

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка програмного засобу для оренди автомобільного транспорту»

Виконав: студент 4 курсу
групи 2ПІ-216
спеціальності

121 – Інженерія програмного забезпечення

(шифр і назва напрямку підготовки, спеціальності)



Парполіта В.О.

(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Хошаба О.М.

(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. ОТ Черняк О.І.

(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ Романюк О. Н.

«09» 06 2025 р.

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 - Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма - Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

Романюк О. Н.

21 березня 2025 року

ЗАВДАННЯ НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Парполіта Вадиму Олексійовичу

1. Тема роботи - «Розробка програмного засобу для оренди автомобільного транспорту»

Керівник роботи: Хошаба Олександр Мирославович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. № 97.

2. Строк подання студентом роботи 2 червня 2025 р.

3. кількість параметрів, що характеризують транспортні засоби в системі: до 25 (тип, марка, модель, рік випуску, клас, пробіг, VIN, страхування, фото тощо), кількість вимог до функціоналу програмного забезпечення: до 30 (авторизація, управління профілем, перевірка прав водія, пошук, бронювання, тарифи, аналітика, адмінпанель тощо), загальна кількість записів у базі даних: до 100 000, продуктивність системи під час обробки повинна бути не менш ніж 150 запитів у секунду у піковому режимі, підтримка одночасної роботи: до 500 користувачів.

4. Зміст розрахунково-пояснювальної записки: вступ, аналітичний огляд систем оренди транспорту та методів їх автоматизації, постановка задачі та визначення функціональних вимог, розробка архітектури програмного засобу (UML, база даних, модулі), реалізація клієнтської та серверної частини, тестування функціоналу системи та аналіз ефективності, рекомендації щодо впровадження та

масштабування, висновки, вписок використаних джерел, додатки (SQL-схеми, скріншоти інтерфейсів, фрагменти коду).

5. Перелік графічного матеріалу: мета та задачі роботи, порівняльна характеристика аналогів, розробка архітектури програмного засобу (UML, база даних, модулі), реалізація клієнтської та серверної частини, тестування функціоналу системи та аналіз ефективності, тестування, висновки.

6. Консультанти розділів роботи

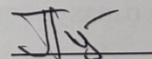
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Хошаба О.М., к.т.н., доцент кафедри ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз проблеми, обґрунтування актуальності розробки системи та постановка задач оренди автомобілів	25.03.2025-03.04.2025	вип
2	Проектування модулів, розробка алгоритмів, структури та моделей динамічного ціноутворення з урахуванням факторів попиту	04.04.2025-14.04.2025	вип
3	Вибір середовища та розробка програмного забезпечення веб-сервісної системи оренди	15.04.2025-05.05.2025	вип
4	Створення тест-кейсів для перевірки стабільності веб-сервісу	06.05.2025-19.05.2025	вип
5	Оформлення матеріалів до захисту БКР	20.05.2025-30.05.2025	вип

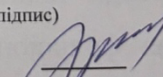
Студент


(підпис)

Парполіта В.О.

(прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи


(підпис)

Хошаба О.М.

(прізвище та ініціали)

АНОТАЦІЯ

УДК 004.4:656.13:005.962

Парполіта В.О. Розробка програмного засобу для оренди автомобільного транспорту : бакалаврська кваліфікаційна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 1?? С.

На укр. мові. Бібліогр. : —23 назв ; рис. : 24; табл. 2.

У кваліфікаційній роботі представлено результати розробки програмного засобу для оренди автомобільного транспорту. Робота складається зі вступу, п'яти основних розділів, висновків, списку використаних джерел та додатків. У процесі дослідження було проаналізовано сучасні рішення в галузі оренди транспорту, сформовано технічні вимоги, побудовано UML-діаграми архітектури, розроблено клієнтську та серверну частину програмного забезпечення, а також проведено тестування основних функцій системи. Результати демонструють ефективність впровадженого рішення для оптимізації орендного процесу.

Ключові слова: оренда, автомобільний транспорт, програмне забезпечення, система бронювання, вебзастосунок.

ANNOTATION

Parpolita V.O. Development of software for car rental: bachelor's thesis on the specialty 121 Software engineering, educational program - software engineering. Vinnytsia: VNTU, 2025. 1?? with.

In Ukrainian language. Bibliographer : 23 titles; Fig. : 24; table 2.

The qualification work presents the results of developing a software tool for car rental. The work includes an introduction, five main sections, a conclusion, a list of sources used and appendices. In the research process, modern solutions in the field of car rental were analysed, technical requirements were formed, UML architecture diagrams were built, the client and server parts of the software were developed, and the system's primary functions were tested. The results demonstrate the effectiveness of the implemented solution for optimising the rental process.

Keywords: rental, car transport, software, booking system, web application.

ЗМІСТ

ВСТУП	3
1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ	7
1.1 Аналіз предметної галузі та особливості роботи надання в оренду автомобільний транспорт населенню	7
1.2 Порівняльний аналіз програмних засобів з надання в оренду автомобільний транспорт населенню	16
1.3 Постановка задач дослідження	27
2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ	28
2.1 Визначення процесу динамічного ціноутворення	28
2.2 Визначення процесу оптимізації оренди автомобільного транспорту	31
2.3 Проектування архітектури веб-сервісної системи оренди автомобільного транспорту	34
3 РОЗРОБКА МОДУЛІВ ПРОГРАМНОГО ЗАСОБУ	40
3.1 Розробка програмного коду модуля динамічного ціноутворення	40
3.2 Розробка програмного коду модуля оптимізації процесу оренди автомобільного транспорту	44
3.3 Розробка веб-сервісної системи оренди автомобільного транспорту	49
4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ	56
4.1 Тестування модуля динамічного ціноутворення	56
4.2 Розробка тест-кейсів процесу оптимізації оренди автомобілів	62
4.3 Тестування веб-сервісної системи оренди автомобільного транспорту	71
ВИСНОВКИ	75
ПЕРЕЛІК ПОСИЛАНЬ	78
ДОДАТОК А	Error! Bookmark not defined.
Технічне завдання	Error! Bookmark not defined.
ДОДАТОК Б	Error! Bookmark not defined.
ДОДАТОК В	87
ДОДАТОК Г	110

ВСТУП

Обґрунтування вибору теми дослідження. Оренда автомобільного транспорту є однією з найбільш динамічних галузей сучасної економіки, що поєднує в собі інновації в транспортній інфраструктурі, електронній комерції, автоматизації сервісів та інформаційних технологіях. Стрімке зростання попиту на гнучкі форми мобільності серед населення, особливо в умовах урбанізації, призводить до зміщення уваги споживачів від традиційних моделей володіння автомобілем до коротко- та середньострокової оренди транспортних засобів.

Тому, актуальність теми кваліфікаційної роботи полягає в необхідності цифровізації послуг оренди автотранспорту із застосуванням інноваційних інформаційних технологій, які дозволяють підвищити ефективність, точність та зручність управління сервісом для користувачів, операторів та власників автопарків. Особливу увагу приділена питанням розробки інтуїтивно зрозумілих користувацьких інтерфейсів, реалізації надійного механізму бронювання, автоматичного розрахунку вартості, а також модулів геолокації, перевірки особи, моніторингу технічного стану автомобілів та збору аналітичної інформації.

Попри широкий попит на відповідні цифрові рішення, багато з існуючих систем мають низку обмежень: складний інтерфейс, обмежений набір функцій, низьку адаптивність до різних бізнес-моделей (наприклад, оренда з погодинною, добовою чи передплатною оплатою), або недостатній захист даних користувача.

Крім того, більшість рішень не враховують локальні особливості регулювання оренди транспорту, включно з нормативами страхування, юридичними аспектами реєстрації угод та оподаткування.

Галузь цифрових сервісів оренди транспорту стикається з низкою важливих проблем, що ускладнюють її ефективне функціонування. Насамперед це:

- фрагментованість бізнес-процесів, де компанії змушені використовувати кілька окремих сервісів (CRM, платіжні шлюзи, геолокаційні модулі, API-інтеграції з партнерами), що створює складнощі в управлінні єдиним циклом обслуговування клієнта;
- недостатня автоматизація процесів обробки замовлень, перевірки документів, аналітики використання автопарку;

- низький рівень персоналізації взаємодії з користувачами: брак адаптивних алгоритмів рекомендацій, прогнозування попиту, історії попередніх оренд;
- проблеми масштабування, де у міру зростання клієнтської бази існуючі системи часто втрачають стабільність і продуктивність.
- безпекові ризики, що зокрема, пов'язані з обробкою персональних даних клієнтів, даних про оплату, а також контролем доступу до автомобіля.

Ці проблеми обумовлюють потребу у створенні сучасного, інтегрованого програмного рішення, яке буде адаптовано до вимог як малого/середнього бізнесу, так і великих компаній з розгалуженою мережею філій.

Таким чином, виникає необхідність у створенні програмного засобу, який би поєднував гнучкість, масштабованість, простоту в користуванні та відповідність юридичним та технічним стандартам. Тому, запропонована тема дослідження є логічною відповіддю на ці виклики.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась згідно плану виконання наукових досліджень на кафедрі програмного забезпечення

Мета та завдання. Метою роботи є підвищення ефективності процесу оренди автомобільного транспорту шляхом розробки програмного засобу з розширеними функціональними можливостями, що забезпечують автоматизацію операцій з реєстрації, бронювання, обліку, оплати та управління автопарком з урахуванням актуальних вимог безпеки, масштабованості та інтеграції з зовнішніми системами.

Відповідно до поставленої мети виконані наступні задачі:

- провести аналіз предметної галузі та визначити особливості ринку оренди автомобілів;
- виконати порівняльний аналіз існуючих програмних рішень для оренди транспорту;
- розробити модель динамічного ціноутворення з урахуванням факторів попиту та сезонності;
- запропонувати метод оптимізації розподілу автомобілів для мінімізації простоїв;

- спроектувати архітектуру веб-сервісної системи оренди з інтеграцією сучасних технологій;
- реалізувати програмні модулі для динамічного ціноутворення та управління орендою;
- провести тестування функціональності модулів динамічного ціноутворення та оптимізації;
- створити тест-кейси для перевірки стабільності веб-сервісу;
- проаналізувати результати тестування та визначити шляхи вдосконалення системи.

Об'єктом дослідження є процес автоматизації оренди автомобільного транспорту за допомогою програмного забезпечення.

Предметом дослідження є методи, моделі та програмні засоби для реалізації веборієнтованої інформаційної системи управління орендою автомобілів, які забезпечують підвищення ефективності, продуктивності, точності, зручності та надійності надання послуг клієнтам.

Методи дослідження. У процесі дослідження були використані: методи структурного та об'єктно-орієнтованого аналізу для формалізації вимог до системи, UML-моделювання для опису логіки функціонування основних модулів, принципи проектування REST-архітектури для розробки API-сервісів, методи проектування реляційних баз даних для зберігання інформації про користувачів, бронювання, транспортні засоби тощо, засоби вебпрограмування (HTML/CSS, JavaScript, Spring Framework) для реалізації клієнтської та серверної частин, методи тестування програмного забезпечення, включаючи юніт- та інтеграційне тестування, аналітичне моделювання навантаження з метою оцінки продуктивності системи.

Новизна отриманих результатів:

1. Запропоновано метод оренди автомобільного транспорту, особливність якого полягає у веб-сервісній архітектурі з інтегрованим функціоналом онлайн-бронювання, GPS-трекінгу транспортних засобів та платіжних систем, що надає можливість оцінити його ефективність та існуючий потенціал для збільшення завантаженості автопарку на більш ніж 18%.

2. Запропоновано метод оптимізації процесу оренди автомобільного транспорту, що дозволяє скоротити час обробки замовлень за рахунок автоматизації ключових етапів та аналізу даних про поведінку користувачів.

3. Подальшого розвитку отримала архітектура веб-сервісної системи оренди автомобільного транспорту, у якій, на відміну від існуючих аналогів, інтегровано модуль аналітики користувацьких дій та рекомендаційний сервіс, що дозволило підвищити ефективність взаємодії користувачів із системою.

Практична цінність отриманих результатів. Практичне значення роботи полягає у розробці повнофункціонального програмного засобу для управління процесом оренди автомобільного транспорту, що включає зручний вебінтерфейс, підтримку багатокористувацького середовища, інтеграцію з платіжними сервісами та модулями верифікації користувача. Розроблена система дозволяє скоротити час обробки заявок, зменшити навантаження на персонал, мінімізувати ризики людського фактору та значно підвищити задоволеність користувачів. Впровадження даного рішення може бути доцільним як у сфері приватного бізнесу, так і в муніципальних транспортних програмах.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській кваліфікаційній роботі, отримані автором особисто. Автору належать такі результати: постановка задачі дослідження; створення алгоритмів, розробка основних та додаткових модулів програмного засобу, тестування модулів.

Апробація результатів роботи. Результати роботи були представленні на XXV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів.

Публікації. Результати роботи були опубліковані в матеріалах XXV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. «Стан, досягнення і перспективи інформаційних систем і технологій». Одеса, 17 - 18 квітня 2025 р. [1].

1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної галузі та особливості роботи надання в оренду автомобільний транспорт населенню

Ринок оренди автомобілів в Україні та світі демонструє стабільне зростання і трансформацію під впливом нових технологій та зміни споживчих уподобань. Сучасний підхід до оренди автомобілів включає різноманітні бізнес-моделі - від традиційних прокатних компаній до інноваційних сервісів каршерингу та P2P-прокату [3]. Цей аналіз присвячений дослідженню методів та моделей (рис. 1.1), які використовуються для надання в оренду автомобільного транспорту населенню.

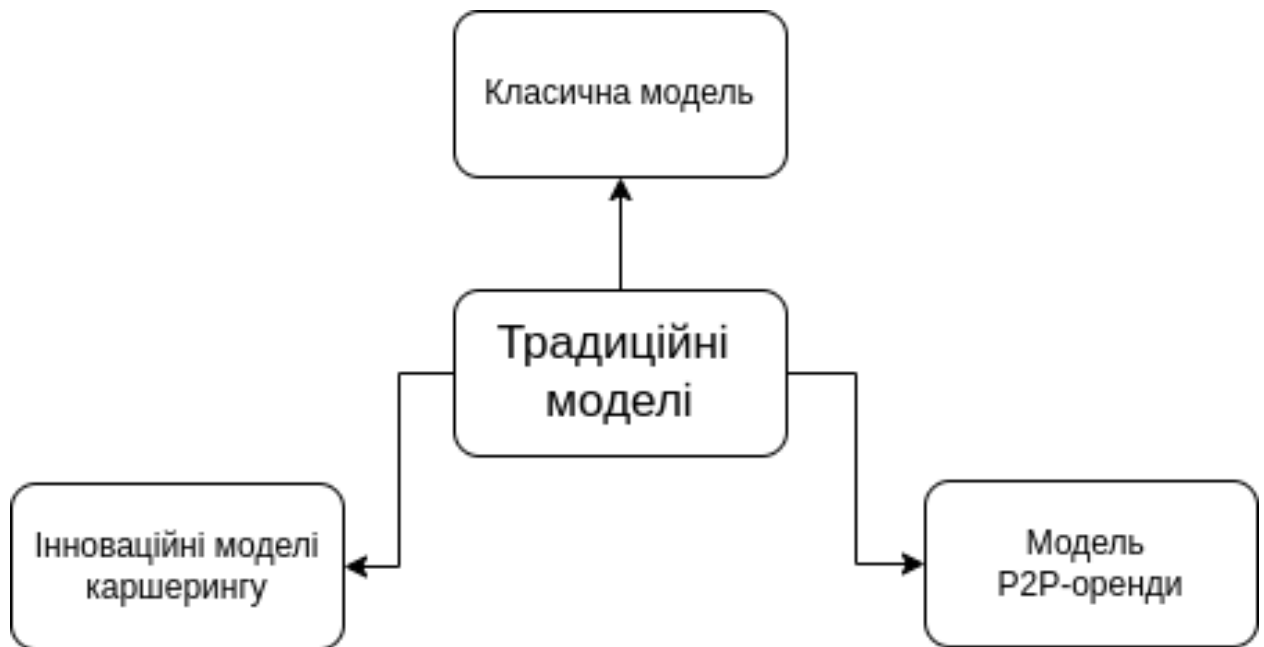


Рисунок 1.1 - Традиційні моделі оренди автомобілів

Класична модель оренди автомобілів передбачає володіння автопарком компанією, яка здає транспортні засоби в оренду на різні періоди. Згідно з дослідженнями Українського центру економічних та політичних досліджень імені О. Разумкова, цей підхід залишається домінуючим на ринку України, охоплюючи близько 65% всього ринку оренди автомобілів [5].

Класична бізнес-модель включає наступні елементи:

- придбання та утримання власного автопарку;

- фізичні пункти видачі автомобілів;
- стандартизовані тарифні плани;
- документальне оформлення договорів оренди.

Лебідь О.В. у своїй праці "Економіка автотранспортних підприємств України" підкреслює, що традиційна модель має високі операційні витрати, пов'язані з утриманням автопарку та персоналу, що зумовлює відносно високу вартість послуг для кінцевого споживача [8].

Класична (традиційна) модель оренди автомобілів є найбільш усталеною та поширеною формою організації бізнесу в цій галузі (табл. 1.1). Розглянемо її основні елементи та особливості функціонування детальніше.

Таблиця 1.1 - Порівняння основних моделей оренди автомобілів

Критерій	Класична модель	Каршеринг	P2P-оренда
Принцип роботи	Компанія володіє автопарком і здає автомобілі в оренду	Короткострокова оренда автомобілів компанії	Приватні власники здають свої автомобілі через онлайн-платформи
Частка ринку в Україні	~65% [5]	Прогноз до 30% у 2027 р. [6]	Розвивається з 2018 р. [9]
Капітальні витрати	Високі (200-500 тис. USD) [3]	Високі	Низькі (витрати на платформу)
Отримання автомобіля	Фізичні пункти видачі	Автомобілі розподілені по місту	За домовленістю з власником
Тривалість оренди	Доба, тиждень, місяць	Хвилини, години	Години, доби, тижні
Оформлення	Документальне (15-30 хв.)	Мінімальне через додаток	Через платформу

Класична модель оренди автомобілів базується на принципі, коли компанія володіє власним автопарком і надає автомобілі клієнтам у тимчасове користування за визначену плату. Лебідь О.В. у своїй праці зазначає, що ця модель характеризується чіткою організаційною структурою з декількома ключовими підрозділами [8], які мають наступні особливості в таких структурах:

- відділ управління автопарком, що забезпечує придбання, облік, обслуговування та списання транспортних засобів;
- мережа пунктів видачі/прийому автомобілів, що має фізичні локації для обслуговування клієнтів;
- департамент продажів та маркетингу, де існує залучення клієнтів та просування послуг;
- служба підтримки клієнтів, де є вирішення питань, що виникають під час оренди;
- адміністративний та фінансовий відділи, де існує загальне управління компанією.

В загальному випадку, традиційний процес оренди автомобіля включає наступні етапи:

- бронювання, де клієнт резервує автомобіль через веб-сайт, телефон або безпосередньо у пункті видачі;
- підписання договору, де є оформлення документів, перевірка посвідчення водія та інших необхідних документів;
- внесення застави, де є блокування коштів на картці або внесення готівкової застави;
- огляд автомобіля, де є спільний з працівником компанії огляд та фіксація стану автомобіля;
- використання, де існує безпосередня експлуатація автомобіля клієнтом;
- повернення, де існує здача автомобіля, повторний огляд стану, розрахунок додаткових платежів;
- повернення застави, де є розблокування коштів клієнта після перевірки відсутності пошкоджень.

За даними Українського центру економічних та політичних досліджень, в Україні середній час на оформлення оренди автомобіля в класичній моделі складає 25-30 хвилин [5]. Тому, фінансова модель традиційної оренди автомобілів є досить ефективною, що має такі особливості:

- високі капітальні витрати, що включають значні інвестиції у придбання автопарку. За даними Асоціації українських автомобільних дилерів, первинні

інвестиції для відкриття невеликої прокатної компанії в Україні становлять від 200 тис. до 500 тис. доларів США [3];

- фіксовані операційні витрати, що включають утримання приміщень для пунктів видачі/прийому, заробітну плату персоналу, страхування автопарку, технічне обслуговування автомобілів, адміністративні витрати;

- тарифна політика, що включає добову ставку оренди, додаткові платежі за перевищення ліміту пробігу, оплату за додаткові послуги (GPS-навігація, дитячі крісла тощо), страхові платежі;

- амортизація автопарку, де автомобілі зазвичай використовуються протягом 2-3 років, після чого продаються на вторинному ринку. Марченко І.С. у своїй монографії зазначає, що оптимальний термін використання автомобіля в прокаті дозволяє максимізувати прибуток компанії, балансуючи між вартістю обслуговування та залишковою ринковою вартістю [11].

Класична модель оренди також характеризується структурованим автопарком (табл. 1.2), який зазвичай поділяється на категорії:

- економ-клас, де є бюджетні малолітражні автомобілі (35-45% автопарку);
- середній клас, де є - сімейні седани та хетчбеки (30-35% автопарку);
- бізнес-клас, де є - автомобілі преміум-сегменту (10-15% автопарку);
- мінівени та SUV, де є - автомобілі підвищеної місткості (10-15% автопарку);
- спеціалізовані автомобілі, де є - кабріолети, спорткари тощо (5-10% автопарку).

За дослідженнями Української асоціації автомобільного бізнесу, найбільшим попитом на українському ринку користуються автомобілі економ та середнього класу, які забезпечують близько 70% доходу прокатних компаній [12]. Тому, ефективне управління автопарком є ключовим фактором успіху в класичній моделі оренди. Основні аспекти включають:

- планування закупівель, де є визначення оптимального складу автопарку на основі аналізу попиту;

- технічне обслуговування, де є регулярне ТО, сезонна зміна шин, усунення поломок;

- логістика, де є переміщення автомобілів між пунктами видачі для балансування попиту;
- контроль використання, де є моніторинг дотримання умов договору оренди;
- оновлення автопарку, де є своєчасна заміна автомобілів для підтримки якості послуг.

Таблиця 1.2 - Структура автопарку в класичній моделі оренди

Категорія автомобілів	Частка в автопарку	Внесок у дохід
Економ-клас	35-45%	Разом з середнім класом ~70% [12]
Середній клас	30-35%	—
Бізнес-клас	10-15%	—
Мінівени та SUV	10-15%	—
Спеціалізовані автомобілі	5-10%	—

Дослідження Харківського національного автомобільно-дорожнього університету показують, що оптимізація процесів управління автопарком дозволяє знизити операційні витрати на 10-15% [10]. Тому, незважаючи на традиційність моделі, сучасні компанії активно впроваджують цифрові технології для підвищення ефективності:

- системи управління автопарком (Fleet Management Systems), що являє собою програмне забезпечення для обліку, контролю технічного стану та планування обслуговування автомобілів;
- CRM-системи, що має управління взаємовідносинами з клієнтами, формування персоналізованих пропозицій;
- онлайн-бронювання, що має можливість резервування автомобіля через веб-сайт або мобільний додаток;
- електронний документообіг, що дозволяє виконати спрощення процесу оформлення договорів оренди.

За даними Українського центру цифрових інновацій, впровадження комплексних IT-рішень у традиційних прокатних компаніях дозволяє знизити адміністративні витрати на 20-25% [14]. Тому, існують переваги та недоліки класичної моделі. До переваги відноситься:

- стабільність та передбачуваність операційної діяльності;
- можливість контролю якості автопарку;
- формування лояльної клієнтської бази;
- гнучкість у формуванні індивідуальних пропозицій для корпоративних клієнтів.

До недоліків відноситься:

- висока капіталомісткість бізнесу;
- значні операційні витрати;
- нижча ефективність використання автопарку порівняно з моделями каршерингу;
- складніший процес оформлення оренди для клієнта.

Важливо також зазначити, що класична модель також еволюціонує під впливом нових технологій та зміни споживчих уподобань. За даними Deloitte Ukraine, традиційні прокатні компанії впроваджують елементи інноваційних моделей, зокрема [6]:

- гібридні сервіси, що є поєднання класичної оренди та каршерингу;
- безключовий доступ, що надає можливість відкрити автомобіль за допомогою мобільного додатку;
- гнучкі тарифні плани, що надає - можливість оренди на короткі періоди (від 1 години);
- мобільні пункти видачі, що має доставку автомобілів до клієнта.

Такі зміни дозволяють класичній моделі оренди залишатися конкурентоспроможною в умовах появи нових форматів послуг.

Таким чином, класична модель оренди автомобілів, незважаючи на появу інноваційних підходів, залишається основою ринку автопрокату в Україні та світі. Її стабільність, надійність та перевірені роками бізнес-процеси забезпечують високий рівень довіри серед клієнтів, особливо у сегменті довгострокової оренди

та корпоративного обслуговування. Водночас ця модель поступово трансформується, інтегруючи цифрові технології та елементи нових форматів послуг, що дозволяє підвищувати ефективність операційної діяльності та відповідати сучасним вимогам споживачів.

Інноваційні моделі каршерингу в першу чергу представляє собою модель короткострокової оренди автомобілів, як правило, на години або навіть хвилини. Ця модель набуває популярності в Україні з 2016 року, коли почали з'являтися перші сервіси такого типу [4]. До основних особливостей моделі каршерингу відноситься:

- відсутність фізичних пунктів видачі (автомобілі розподілені по місту);
- мобільний додаток як основний інтерфейс взаємодії;
- похвилинна або погодинна тарифікація.

Дослідження Київського національного транспортного університету показують, що каршеринг дозволяє знизити загальні витрати на транспорт для користувачів на 20-35% порівняно з володінням власним автомобілем [7].

Компанія Deloitte у своєму звіті "Майбутнє мобільності в Україні" прогнозує, що до 2027 року частка каршерингу на ринку оренди автомобілів в Україні може зрости до 30% [6]. Для цього, певним чином потрібен розвиток P2P (peer-to-peer) модель оренди автомобілів. Модель P2P-оренди автомобілів дозволяє приватним власникам авто здавати їх в оренду іншим користувачам через спеціалізовані онлайн-платформи. В Україні ця модель почала розвиватися з 2018 року [9]. До основних переваг P2P-моделі відноситься:

- відсутність необхідності інвестицій в автопарк;
- більша різноманітність доступних автомобілів;
- нижчі ціни порівняно з традиційною орендою;
- додатковий дохід для власників автомобілів.

За даними дослідження Української асоціації автомобільного бізнесу, P2P-оренда дозволяє власникам авто отримувати до 80% від вартості оренди, в той час як посередницька платформа отримує від 15% до 20% комісії [12]. Таке ціноутворення в галузі оренди автомобілів (табл. 1.3) має свої особливості та

специфіку (табл. 1.2). Марченко І.С. у монографії "Економіка транспортних послуг" виділяє наступні методи ціноутворення [11]:

- статичне ціноутворення, де є класичний підхід з фіксованими тарифами за добу, тиждень або місяць;
- динамічне ціноутворення, де є метод, що враховує ринковий попит і пропозицію в реальному часі. Згідно з дослідженнями Харківського національного автомобільно-дорожнього університету, впровадження динамічного ціноутворення дозволяє підвищити ефективність використання автопарку на 15-20% [10];
- пакетне ціноутворення, де є комплексна пропозиція, що включає додаткові послуги (страхування, GPS-навігація, дитячі крісла тощо);
- сезонне ціноутворення - враховує сезонні коливання попиту.

Таблиця 1.3 - Методи ціноутворення в оренді автомобілів

Метод ціноутворення	Опис	Ефективність
Статичне ціноутворення	Фіксовані тарифи за період	Традиційний підхід
Динамічне ціноутворення	Ціни змінюються залежно від попиту	Підвищення ефективності на 15-20% [10]
Пакетне ціноутворення	Комплексна пропозиція з додатковими послугами	Збільшення середнього чеку
Сезонне ціноутворення	Врахування сезонних коливань попиту	Оптимізація завантаження автопарку
Алгоритмічне ціноутворення	Використання ML-алгоритмів	Збільшення прибутковості на 8-12% [13]

Міжнародна консалтингова компанія McKinsey & Company у своєму звіті "Транспортні послуги в Східній Європі" зазначає, що застосування алгоритмів машинного навчання для динамічного ціноутворення (табл. 1.4) дозволяє збільшити прибутковість бізнесу з оренди автомобілів на 8-12% [13]. Тому, сучасний ринок оренди автомобілів активно використовує цифрові технології для

підвищення ефективності та зручності послуг. Основні технологічні інструменти включають:

- мобільні додатки, що являють собою основний інтерфейс взаємодії з користувачем, який дозволяє бронювати, оплачувати та управляти орендою автомобіля;
- телематичні системи, що дозволяють відстежувати місцезнаходження автомобіля, його технічний стан та особливості використання. Дослідження Національного університету "Львівська політехніка" показують, що використання телематики дозволяє знизити витрати на обслуговування автопарку на 12-15% [2];
- системи безключового доступу, що являє собою технологію, яка дозволяє користувачам відкривати та заводити автомобіль за допомогою мобільного додатку;
- інтегровані платіжні системи, що забезпечують зручність оплати послуг через різні канали;
- CRM-системи, що дозволяють управляти базою клієнтів та аналізувати їх поведінку.

Таблиця 1.4 - Технологічні інструменти в оренді автомобілів

Технологія	Призначення	Ефект
Мобільні додатки	Бронювання та управління орендою	Спрощення процесу для клієнта
Телематичні системи	Відстеження автомобіля та його стану	Зниження витрат на обслуговування на 12-15% [2]
Системи безключового доступу	Відкриття автомобіля через додаток	Спрощення процесу видачі
CRM-системи	Управління клієнтською базою	Підвищення лояльності клієнтів
Fleet Management Systems	Управління автопарком	Зниження адміністративних витрат на 20-25% [14]

За даними Українського центру цифрових інновацій, компанії, що впроваджують комплексні цифрові рішення для управління автопарком, демонструють зростання ефективності бізнесу на 20-30% [14]. Таке формування та обслуговування автопарку потребує значних інвестицій. Існують різні моделі фінансування, які використовуються компаніями в сфері оренди автомобілів:

- придбання у власність, що є класична модель з повним володінням автопарком;
- лізинг, що є довгостроковою орендою автомобілів з можливістю їх викупу. Дослідження Української лізингової асоціації показують, що близько 45% компаній в сфері оренди автомобілів використовують саме цю модель [4];
- партнерство з автовиробниками, що є класична модель, при якій виробники автомобілів надають спеціальні умови для компаній, що займаються прокатом;
- модель "зворотного викупу", що є придбання автомобілів з гарантією зворотного викупу виробником після певного періоду експлуатації.

Аналітичний звіт Асоціації українських автомобільних дилерів відзначає, що модель "зворотного викупу" дозволяє знизити капітальні витрати компаній на 15-20% порівняно з прямим придбанням [3].

Таким чином, аналіз предметної галузі демонструє різноманітність методів та моделей для надання в оренду автомобільного транспорту населенню. Традиційні моделі оренди поступаються місцем інноваційним підходам, таким як каршеринг та P2P-оренда. Успішне функціонування бізнесу в цій сфері вимагає оптимального поєднання фінансових моделей, технологічних інструментів та методів ціноутворення. Особливої уваги заслуговують цифрові технології, які трансформують галузь, підвищуючи ефективність бізнесу та зручність для користувачів. Стратегічне впровадження таких технологій дозволяє компаніям отримати конкурентну перевагу на ринку оренди автомобілів.

1.2 Порівняльний аналіз програмних засобів з надання в оренду автомобільний транспорт населенню

Ринок оренди автомобілів переживає значну трансформацію завдяки впровадженню сучасних інформаційних технологій. Різноманітні програмні засоби дозволяють автоматизувати процеси, оптимізувати використання автопарку та підвищити рівень обслуговування клієнтів. Цей аналіз присвячений порівнянню різних програмних інструментів, які використовуються компаніями з надання в оренду автомобільного транспорту населенню.

Програмні засоби для оренди автомобілів можна класифікувати за кількома основними категоріями (рис. 1.2) відповідно до їх функціонального призначення та архітектури. Згідно з дослідженням Київського національного університету імені Тараса Шевченка, основними категоріями є [15]:

- системи управління автопарком (Fleet Management Systems), які є комплексні рішення для моніторингу та оптимізації використання автомобілів;
- програмне забезпечення для управління орендою (Rental Management Software), які є інструменти для автоматизації процесів бронювання, оформлення договорів та розрахунків;
- мобільні додатки для клієнтів, які є спеціалізовані додатки для смартфонів, що дозволяють клієнтам бронювати, отримувати та повертати автомобілі;
- платформи для P2P-оренди, які є програмні рішення для організації оренди автомобілів між приватними особами;
- інтегровані системи каршерингу, які є комплексні рішення для організації короткострокової оренди автомобілів.

Розглянемо кожен з категорій окремо більш ретельно. Системи управління автопарком (Fleet Management Systems) є ключовим інструментом для компаній, що надають послуги з оренди автомобілів. Вони дозволяють оптимізувати використання транспортних засобів, контролювати їх технічний стан та планувати обслуговування. Так, за даними дослідження інституту транспортних технологій Національного транспортного університету, сучасні системи управління автопарком включають наступні функціональні блоки [16]:

- моніторинг місцезнаходження та руху автомобілів, що відповідає за відстеження місцезнаходження транспортних засобів у режимі реального часу;

- контроль технічного стану, що відповідає за моніторинг параметрів роботи двигуна, рівня палива, технічних показників;
- планування технічного обслуговування, що відповідає за автоматичне формування графіків ТО на основі пробігу та стану автомобіля;
- аналіз ефективності використання, що відповідає за розрахунок коефіцієнта використання автомобілів, оптимізація розподілу;
- управління витратами, що відповідає за облік та аналіз витрат на паливо, ремонт, страхування.

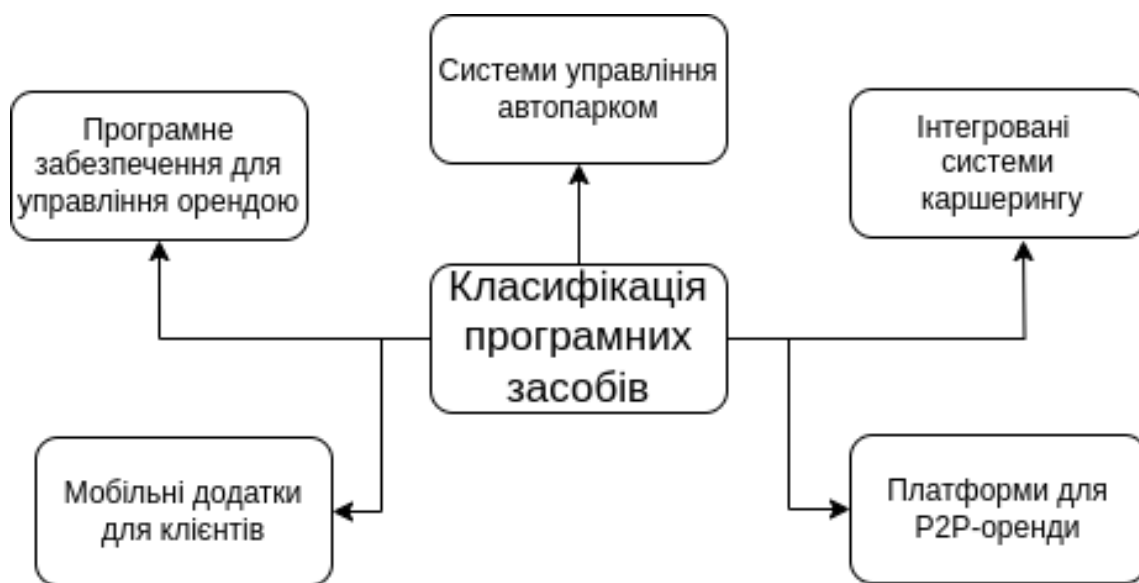


Рисунок 1.2 - Класифікація програмних засобів для оренди автомобілів за основними категоріями [15]

Аналітичний звіт компанії Frost & Sullivan "Global Fleet Management Market" виділяє кілька провідних рішень на ринку управління автопарком [17]. Далі, виконаємо порівняльний аналіз поширених систем моніторингу транспортних парків до яких відносяться Fleetio, Verizon Connect, Samsara та Azuga (табл. 1.5). Під час порівняльного аналізу будемо фокусується на трьох ключових аспектах функціонування даних систем: основних перевагах, недоліках та можливостях інтеграції з іншими програмними рішеннями. Отже, система Fleetio характеризується інтуїтивним інтерфейсом, наявністю мобільного додатку та аналітичними функціями, проте демонструє обмежену функціональність при

застосуванні до великих автопарків. Інтеграційні можливості реалізуються через API для взаємодії з іншими системами.

Verizon Connect вирізняється розширеним функціоналом моніторингу та потужними аналітичними інструментами. Основним недоліком є відносно висока вартість впровадження та обслуговування. Система забезпечує інтеграцію з бухгалтерськими програмними комплексами.

Сильними сторонами Samsara є передові телематичні функції та аналітика на основі штучного інтелекту, однак процес впровадження характеризується підвищеною складністю. Система відзначається широкими можливостями інтеграції з іншими програмними рішеннями.

Azuga позиціонується як рішення з доступною ціновою політикою та елементами гейміфікації для водіїв. До недоліків відноситься обмежена підтримка на території України. При цьому, система забезпечує середній рівень інтеграційних можливостей.

Таблиця 1.5 - Порівняльний аналіз різних систем для моніторингу транспортних парків

Система	Основні переваги	Недоліки	Інтеграція
Fleetio	Інтуїтивний інтерфейс, мобільний додаток, аналітика	Обмежена функціональність для великих парків	API для інтеграції з іншими системами
Verizon Connect	Розширені функції моніторингу, сильна аналітика	Відносно висока вартість	Інтеграція з бухгалтерськими системами
Samsara	Передові телематичні функції, ШІ-аналітика	Складність впровадження	Широкі можливості інтеграції
Azuga	Доступна ціна, гейміфікація для водіїв	Обмежена підтримка в Україні	Середній рівень інтеграції

Дослідження, проведене Харківським національним автомобільно-дорожнім університетом, показало, що впровадження систем управління автопарком

дозволяє знизити експлуатаційні витрати на 15-20% та підвищити ефективність використання автомобілів на 25-30% [18]. Таке програмне забезпечення для управління орендою фокусується на автоматизації бізнес-процесів прокатних компаній - від бронювання до повернення автомобіля та фінансового обліку.

Згідно з дослідженням аналітичної компанії Technavio "Global Car Rental Software Market", основними функціями таких систем є наступними [19]:

- управління бронюваннями, де є прийом, обробка та підтвердження замовлень;
- управління договорами, де є автоматизація створення та обліку договорів оренди;
- білінг та платежі, де є розрахунок вартості оренди, обробка платежів;
- управління клієнтами (CRM), де є ведення бази клієнтів, історії взаємодії, програми лояльності;
- звітність та аналітика, де є формування операційних та фінансових звітів.

Аналітичне дослідження програмного забезпечення для управління орендними відносинами можна представити за допомогою чотирьох поширених програмних рішень у сфері управління орендними відносинами: HQ Rental Software, Rentman, Navotar та RentWorks. Проведемо дослідження та порівняльний аналіз за допомогою таких ключових параметрів як цільовий сегмент ринку, сильні та слабкі сторони кожного програмного продукту.

Отже, HQ Rental Software орієнтоване на малі та середні компанії. До позитивних характеристик даного рішення належать легкість впровадження та доступна цінова політика, проте наявні обмеження в аналітичному функціоналі, що може негативно впливати на глибину аналізу бізнес-процесів.

Програмний продукт Rentman розроблено для середніх та великих компаній. Він відзначається широким функціоналом та наявністю мобільного додатку, що підвищує мобільність управління. Основним недоліком є висока вартість, що потребує значних фінансових інвестицій.

Navotar позиціонується як універсальне рішення, не обмежене розміром підприємства. Ключовими перевагами є хмарна архітектура та можливості інтеграції з системами управління взаємовідносинами з клієнтами (CRM).

Недоліком системи є складність налаштування, що може потребувати додаткових ресурсів при впровадженні.

RentWorks спеціалізується на обслуговуванні великих компаній. Система характеризується високою масштабованістю та розширеними аналітичними можливостями, що забезпечує комплексний аналіз діяльності. Суттєвим недоліком є висока вартість впровадження, що становить значний бар'єр входу для потенційних користувачів.

Дослідження Львівської бізнес-школи показало, що впровадження спеціалізованого програмного забезпечення для управління орендою дозволяє збільшити кількість оброблених замовлень на 30-40% без збільшення штату працівників [21]. При цьому, мобільні додатки стали невід'ємною частиною сучасного бізнесу з оренди автомобілів, забезпечуючи зручний інтерфейс для взаємодії з клієнтами. Тому, дослідження компанії App Annie "Mobile App Market Report" виділяє наступні ключові функції мобільних додатків для оренди автомобілів [22]:

- пошук та бронювання, де є можливість знайти та зарезервувати автомобіль;
- цифрове оформлення договору, де є підписання договору через додаток;
- безключовий доступ, де є відкриття автомобіля через додаток;
- оплата та управління рахунком, де є здійснення платежів, перегляд історії;
- технічна підтримка, де є можливість зв'язатися з сервісним центром.

Порівняння популярних мобільних додатків для оренди автомобілів

Аналітичний звіт Mobile App Development Association "User Experience in Car Rental Apps" представляє порівняльний аналіз популярних мобільних додатків [23] таких як Turo, Getaround, Zipcar та Hertz. Проведемо порівняльний аналіз та опишемо за чотирма параметрами як назва додатку, особливості бізнес-моделі, переваги та недоліки.

Отже, додаток Turo функціонує за принципом P2P-оренди (peer-to-peer), що передбачає безпосередню взаємодію між власниками транспортних засобів та орендарями. До переваг даного сервісу належать широкий асортимент доступних автомобілів та функціонально зручний інтерфейс. Основним недоліком є обмежена

підтримка на території України, що суттєво знижує доступність сервісу для українських користувачів.

Getaround представляє модель каршерингу, яка характеризується короткостроковою орендою автомобілів. Ключовими перевагами є безключовий доступ до транспортних засобів та гнучка система оплати. Обмежена географія покриття становить суттєвий недолік даного сервісу.

Zircar реалізує клубну модель використання, що передбачає членство користувачів. Сервіс відзначається простотою використання та включенням вартості палива до загальної оплати послуг. Висока вартість членства є основним обмежувальним фактором для потенційних користувачів.

Hertz представляє традиційну модель оренди транспортних засобів. Переваги сервісу полягають у широкій мережі пунктів обслуговування та наявності програми лояльності для постійних клієнтів. Недоліком є складний процес бронювання, що може негативно впливати на користувацький досвід.

Дослідження Київського національного торговельно-економічного університету показало, що компанії, які впровадили мобільні додатки для оренди автомобілів, спостерігають збільшення кількості клієнтів на 20-25% протягом першого року [24]. При цьому, платформи для P2P-оренди автомобілів є інноваційним рішенням, що дозволяє приватним власникам здавати свої автомобілі в оренду через онлайн-сервіси. Так, згідно з дослідженням консалтингової компанії McKinsey & Company "The Future of Mobility", основними функціями платформ для P2P-оренди є [25] наступні:

- верифікація користувачів, де є перевірка водійських прав, особистості;
- страхування та безпека, де є забезпечення страхового покриття, безпечні платежі;
- рейтингова система, де є оцінки та відгуки для власників та орендарів;
- інтегрована комунікація, де є захищений обмін повідомленнями між користувачами;
- гнучке ціноутворення, де є можливість встановлення різних тарифів.

Порівняння платформ для P2P-оренди автомобілів виконувався на основі аналітичного звіту компанії Statista "Peer-to-Peer Car Sharing Market", що представляв порівняльний аналіз популярних платформ [26]. Для цього, було проведено дослідження чотирьох платформ, що функціонують на ринку оренди транспортних засобів: Turo, Getaround, Drivy та Hycar. Порівняльний аналіз було здійснено за чотирма ключовими параметрами: назва платформи, модель функціонування, комісійна політика та особливості позиціонування на ринку. Розглянемо їх більш ретельно.

Платформа Turo представляє повний сервіс з оренди транспортних засобів. Комісійна ставка коливається в діапазоні 15-25% від вартості послуги. Особливостями даної платформи є диверсифікований парк транспортних засобів та інтегрована система страхування, що підвищує рівень безпеки користувачів.

Getaround функціонує як технологічна платформа з найвищою комісійною ставкою серед досліджуваних сервісів - від 20% до 40%. Відмінними характеристиками є безключовий доступ до транспортних засобів та вбудована телематична система, що забезпечує моніторинг та контроль стану автомобілів.

Drivy позиціонується як європейська платформа з комісійною ставкою 20-30%. Основною особливістю є фокусування на європейському ринку, що відображається у специфіці пропонованих послуг та маркетинговій стратегії.

Hycar представлено як спеціалізовану платформу з комісійною ставкою 15-25%. Характерною особливістю є орієнтація на специфічний сегмент ринку - водіїв таксі та служб доставки, що впливає на формування відповідної пропозиції транспортних засобів.

Інше дослідження Одеського національного економічного університету показало, що власники автомобілів, які використовують платформи P2P-оренди, можуть покрити до 80% витрат на утримання автомобіля [27]. При цьому, інтегровані системи каршерингу представляли собою комплексні програмні рішення для організації короткострокової оренди автомобілів з погодинною або похвилинною оплатою. Тому, згідно з дослідженням консалтингової компанії Deloitte "Carsharing Market Analysis", інтегровані системи каршерингу включали наступні компоненти [28]:

- мобільний додаток для клієнтів, де є інтерфейс для пошуку, бронювання та оплати;
- телематична система, де є обладнання для моніторингу стану автомобіля та безключового доступу;
- система управління автопарком, де є моніторинг розташування та технічного стану автомобілів;
- аналітична платформа, де є оптимізація розміщення автомобілів на основі даних про попит;
- система біллінгу, де є розрахунок вартості поїздки та обробка платежів.

Аналітичний звіт компанії Berg Insight "Carsharing Telematics Market" представляв порівняльний аналіз популярних програмних рішень для каршерингу [29] та полягав в тому, що виконувалось порівняльний аналіз чотирьох програмних систем управління мобільністю: Wunder Mobility, Vulog, Ridecell та Good2Go. Дослідження структуровано за чотирма ключовими параметрами: назва системи, архітектурні особливості, функціональні переваги та виявлені недоліки. Результати досліджень полягали в наступному.

Система Wunder Mobility характеризується модульною архітектурою платформи, що забезпечує високий рівень гнучкості та масштабованості при впровадженні. Модульний підхід дозволяє адаптувати систему до специфічних потреб клієнтів та розширювати функціональність відповідно до зростаючих вимог. Основним недоліком даної системи є висока вартість, що може становити суттєвий бар'єр для потенційних користувачів.

Vulog представлена як інтегрована платформа, що відзначається повним функціоналом та розширеними аналітичними можливостями. Інтегрованість системи забезпечує комплексний підхід до управління мобільністю без необхідності залучення сторонніх рішень. Проте, складність впровадження становить значний недолік, що може потребувати додаткових ресурсів та експертизи.

Ridecell функціонує на основі хмарної архітектури, що надає переваги швидкого впровадження та доступу до API (Application Programming Interface) для інтеграції з іншими системами. Однак, обмежена підтримка на території України

суттєво знижує доступність та ефективність використання даної системи для українських підприємств.

Good2Go позиціонується як готове рішення з доступною ціновою політикою та простотою використання, що зменшує вхідний бар'єр для підприємств різного масштабу. Основним недоліком є обмежені можливості кастомізації, що може не задовольняти специфічні потреби окремих клієнтів.

Представлений таким чином аналіз демонстрував різноманітність архітектурних підходів та функціональних особливостей у сфері систем управління мобільністю, що відображало еволюцію та диверсифікацію даного сегменту ринку програмного забезпечення.

Подальші дослідження Інституту проблем ринку та економіко-екологічних досліджень НАН України показало, що впровадження інтегрованих систем каршерингу дозволяє досягти рентабельності бізнесу на рівні 15-20% вже протягом першого року роботи [30]. Для об'єктивного порівняння різних програмних засобів необхідно було оцінити їх за ключовими параметрами, що впливають на ефективність бізнесу з оренди автомобілів. Тому, аналітичний звіт компанії Gartner "Software Solutions for Vehicle Rental Industry" представляє порівняльний аналіз програмних засобів за технічними характеристиками [19] з п'яти категорій для управління транспортними активами: Fleet Management Systems, Rental Management Software, мобільні додатки, P2P-платформи та системи каршерингу. Такий аналіз було здійснено за чотирма ключовими параметрами: архітектурна модель, рівень масштабованості, інтеграційні можливості та рівень захисту даних. Результати були отримані наступні.

Fleet Management Systems характеризуються гібридною архітектурою, що поєднує клієнт-серверну та хмарну моделі. Дані системи демонструють високий рівень масштабованості, що дозволяє адаптувати їх до потреб підприємств різних розмірів та сфер діяльності. Інтеграційні можливості таких систем є широкими, що забезпечує ефективну взаємодію з іншими програмними рішеннями. Рівень захисту даних також визначається як високий, що є критично важливим для забезпечення конфіденційності та цілісності інформації.

Rental Management Software базується переважно на хмарній архітектурі, що забезпечує доступність та гнучкість впровадження. Показники масштабованості, інтеграційних можливостей та захисту даних цієї категорії програмних засобів оцінюються як середні, що відображає їх орієнтацію на середні за розміром підприємства та стандартні потреби у сфері управління орендою транспортних засобів.

Мобільні додатки функціонують за клієнт-серверною архітектурою та характеризуються обмеженою масштабованістю й інтеграційними можливостями. Рівень захисту даних оцінюється як середній, що може бути пов'язано з технологічними обмеженнями мобільних платформ та специфікою їх використання.

P2P-платформи та системи каршерингу демонструють однакові показники за всіма досліджуваними параметрами: хмарна архітектура, висока масштабованість, широкі інтеграційні можливості та високий рівень захисту даних. Це може свідчити про спільні технологічні підходи та схожість бізнес-моделей даних категорій програмних засобів.

Таким чином, проведений порівняльний аналіз програмних засобів для надання в оренду автомобільного транспорту демонстрував різноманітність рішень, кожне з яких має свої переваги та недоліки. Вибір конкретного програмного засобу залежав від бізнес-моделі компанії, масштабу операцій та цільової аудиторії. Так, системи управління автопарком (Fleet Management Systems) є найбільш ефективними для контролю та оптимізації використання транспортних засобів, але вимагають значних інвестицій у впровадження та обслуговування. Програмне забезпечення для управління орендою (Rental Management Software) дозволяє автоматизувати бізнес-процеси прокатних компаній та підвищити ефективність їх роботи. Мобільні додатки є найбільш доступним та економічно ефективним рішенням для взаємодії з клієнтами, але мають обмежені можливості для управління внутрішніми процесами компанії. Платформи для P2P-оренди дозволяють реалізувати інноваційну бізнес-модель з мінімальними інвестиціями в автопарк, але вимагають значних зусиль для формування спільноти користувачів. Інтегровані системи каршерингу є найбільш технологічно складними та дорогими

рішеннями, але дозволяють реалізувати повністю автоматизовану модель короткострокової оренди автомобілів.

Подальший розвиток програмних засобів для оренди автомобілів буде спрямований на інтеграцію з системами мобільності, впровадження технологій штучного інтелекту та адаптацію до нових типів транспортних засобів.

1.3 Постановка задач дослідження

Проаналізувавши існуючі підходи у предметної галузі та особливості роботи надання в оренду автомобільний транспорт населенню було визначено комплекс задач, реалізація яких дозволить розробити програмне забезпечення.

Основними задачами роботи є:

- провести порівняльний аналіз програмних засобів з надання в оренду автомобільний транспорт населенню;
- виконати проектування програмного засобу, де визначити процеси динамічного ціноутворення та оптимізації оренди автомобільного транспорту;
- виконати проектування архітектури веб-сервісної системи оренди автомобільного транспорту;
- розробити програмний код модуля динамічного ціноутворення;
- розробити програмний код модуля оптимізації процесу оренди автомобільного транспорту;
- розробити веб-сервісну структуру системи оренди автомобільного транспорту;
- виконати тестування модуля динамічного ціноутворення;
- розробити тест-кейси процесу оптимізації оренди автомобілів;
- виконати тестування веб-сервісної системи оренди автомобільного транспорту.

Технічне завдання на розробку наведено в додатку А.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

2.1 Визначення процесу динамічного ціноутворення

Процес динамічного ціноутворення є стратегією, що базується на автоматичному коригуванні цін у реальному часі з урахуванням комплексу факторів, включаючи історичні дані про попит, поточні умови ринку та зовнішні параметри, такі як геолокація або сезонність. У контексті оренди транспорту цей підхід передбачає інтеграцію даних GPS для аналізу руху та доступності транспортних засобів, що дозволяє оптимізувати ціни відповідно до динаміки попиту та пропозиції (рис. 2.1).

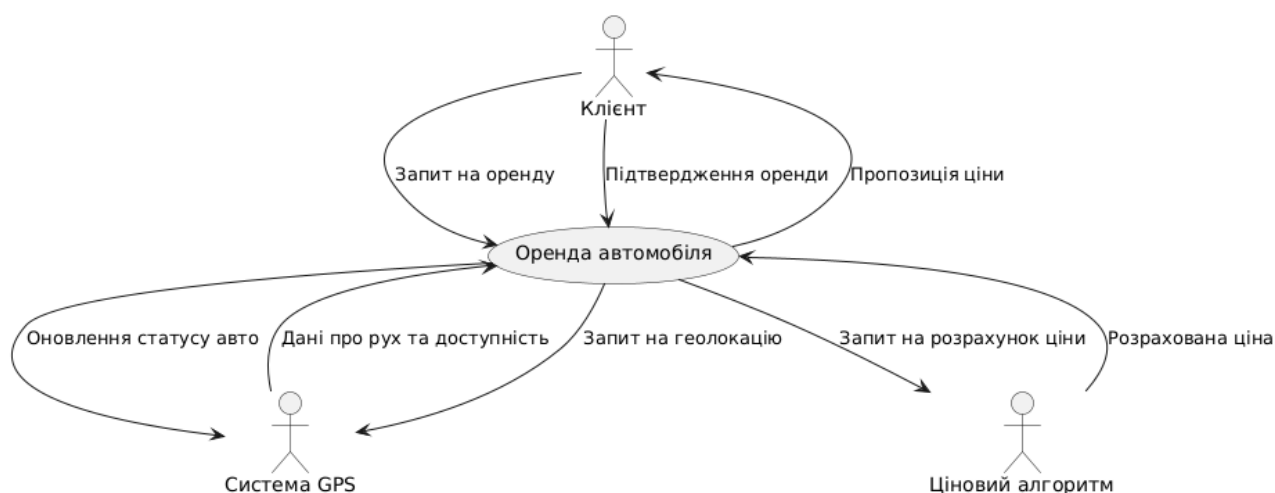


Рисунок 2.1 - UML діаграма використання процесу динамічного ціноутворення

Динамічне ціноутворення у сфері оренди автомобілів базується на автоматизованому процесі, який інтегрує дані GPS для моніторингу руху транспорту, щільності трафіку та доступності автомобілів у конкретних географічних зонах. Цей підхід дозволяє адаптувати ціни в реальному часі, враховуючи поточні умови ринку та потреби клієнтів.

Процес починається з того, що клієнт ініціює запит на оренду автомобіля через спеціалізовану систему. На цьому етапі система GPS забезпечує необхідні дані, включаючи інформацію про рух транспорту, щільність трафіку та доступність автомобілів у конкретній локації. Ці дані є критично важливими для подальшого

аналізу, оскільки вони дозволяють оцінити поточний стан ринку та потреби клієнтів.

Наступним кроком є використання цих даних ціновим алгоритмом для розрахунку динамічної ціни. Алгоритм враховує різні фактори, такі як сезонність, геолокація, попит і пропозиція, щоб забезпечити оптимальну вартість оренди. Цей підхід дозволяє компаніям максимізувати прибуток, одночасно задовольняючи потреби клієнтів.

Після розрахунку ціни клієнт отримує пропозицію, яка включає деталі оренди та розраховану вартість. Якщо клієнт погоджується з пропозицією, він підтверджує оренду, і система GPS оновлює статус автомобіля, позначаючи його як зайнятий. Це забезпечує точність даних про доступність автомобілів та уникнення конфліктів у подальших бронюваннях.

Тому, інтеграція GPS-трекінгу в процес динамічного ціноутворення забезпечує ефективне управління автопарком, оптимізацію цін та задоволення потреб клієнтів, що робить цей підхід ключовим елементом сучасного бізнесу оренди автомобілів.

Оснoву процесу складає аналіз історичних даних, які використовуються для побудови прогнозних моделей попиту. Наприклад, алгоритми машинного навчання обробляють інформацію про минулі оренди, враховуючи сезонні коливання, такі як збільшення попиту під час свят або туристичного сезону. Ці моделі визначають базові цінові рівні, які потім коректуються в реальному часі на основі поточних даних. Ключовим аспектом є інтеграція GPS-трекінгу, що забезпечує моніторинг руху транспорту, щільності трафіку та доступності автомобілів у конкретних географічних зонах (рис. 2.1). Наприклад, у разі виникнення заторів або зниження кількості доступних авто в певному районі система може автоматично підвищити ціни, враховуючи зростання попиту та обмежену пропозицію.

Геолокаційний аналіз дозволяє адаптувати ціни до місцевих умов. Так, у містах з високою конкуренцією або в районах з підвищеним попитом (наприклад, біля аеропортів чи туристичних центрів) ціни можуть бути вищими порівняно з менш затребуваними локаціями. Сезонність враховується шляхом диференціації цін на основі історичних шаблонів. Наприклад, у літні місяці вартість оренди

автомобіля на курортних територіях може зростати через піковий попит, тоді як у зимовий період система запропонує знижені тарифи для стимулювання бронювань.

Технологічна реалізація передбачає використання двоетапних алгоритмів машинного навчання (рис. 2.2). На першому етапі аналізуються взаємозв'язки між ціною, попитом та зовнішніми факторами (наприклад, погодою або подіями), що дозволяє оцінити вплив змін цін на обсяги продажів. На другому етапі здійснюється оптимізація цін для всієї продуктової портфельної мережі з урахуванням обмежень та цілей бізнесу. Сучасні системи, такі як Rentey, автоматизують цей процес, інтегруючи дані з GPS-датчиків, систем бронювання та зовнішніх джерел (наприклад, інформацію про трафік у Google Maps), що забезпечує оперативне оновлення цін із точністю до 90-98%.

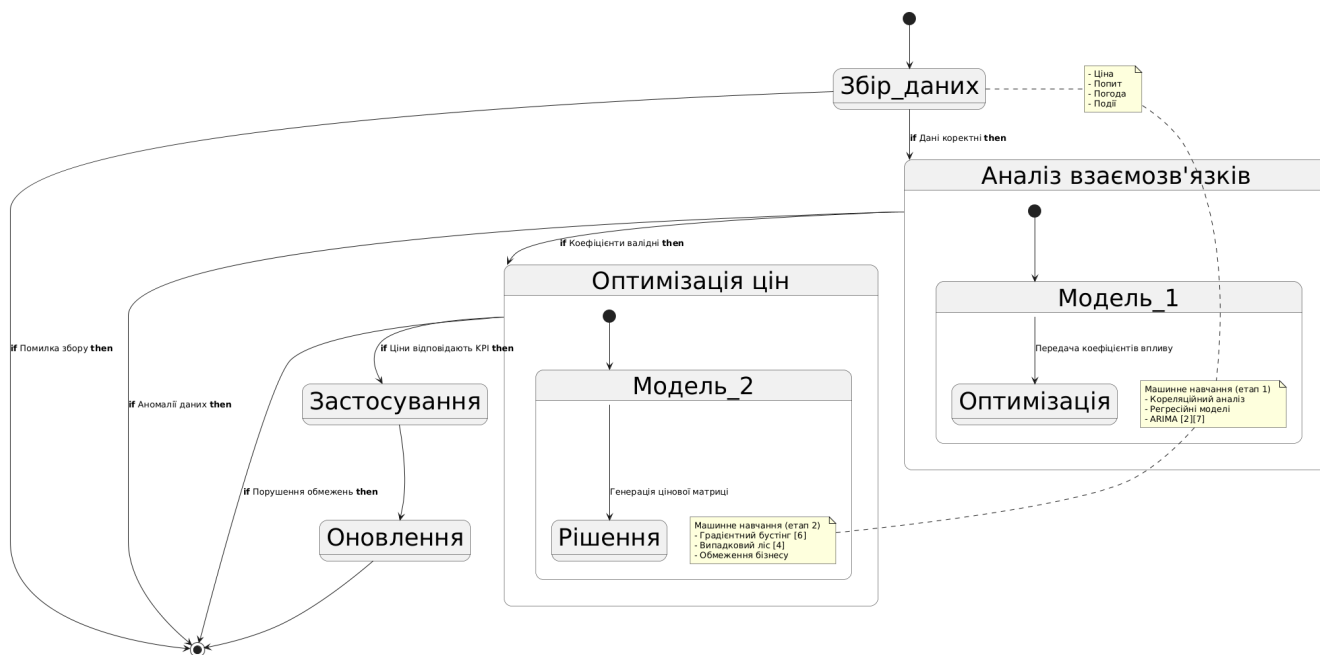


Рисунок 2.2 - UML діаграма станів використання двоетапного алгоритму машинного навчання

Діаграма станів UML (рис. 2.2) визначає життєвий цикл процесу динамічного ціноутворення з використанням двоетапних алгоритмів машинного навчання. Основними станами системи є:

- збір даних, де початковий стан передбачає агрегацію структурованих даних з різних джерел, включаючи історичні ціни, показники попиту, метеорологічні умови та інформацію про події. Відповідно до принципів машинного навчання, якість цих даних безпосередньо впливає на ефективність подальшого аналізу;
- аналіз взаємозв'язків, де на цьому етапі застосовуються методи часових рядів (ARIMA) та регресійного аналізу для виявлення закономірностей між ціною, попитом та зовнішніми факторами. Стан включає підатомат з моделлю машинного навчання, яка генерує коефіцієнти впливу окремих параметрів;
- оптимізація цін, де цей другий етап передбачає використання алгоритмів Gradient Boosting та Random Forest для створення цінової матриці з урахуванням бізнес-обмежень. Суттєвим елементом є механізм валідації результатів щодо ключових показників ефективності (KPI);
- застосування ціни та оновлення системи, де існують фінальні стани реалізують інтеграцію нових цін у продуктову мережу та забезпечують зворотній зв'язок для подальших ітерацій.

Переходи між станами регулюються умовними виразами, що відповідають принципам побудови детермінованих автоматів. Система передбачає можливість відкату до початкового стану при виявленні аномалій даних або порушенні обмежень, що забезпечує відмовостійкість процесу. Технічна реалізація враховує вимоги до паралельних підавтоматів, де кожен етап машинного навчання функціонує як незалежний модуль з чіткими інтерфейсами взаємодії

Таким чином, динамічне ціноутворення в оренді транспорту поєднує історичний аналіз, реальний моніторинг руху та геолокаційну адаптацію, забезпечуючи оптимальний баланс між доходом бізнесу та задоволенням потреб клієнтів. Ця методологія є прикладом застосування Big Data та штучного інтелекту для створення гнучких, ефективних бізнес-моделей у сучасних умовах ринку.

2.2 Визначення процесу оптимізації оренди автомобільного транспорту

Оптимізація процесів оренди автомобільного транспорту ґрунтується на комплексному підході до автоматизації ключових операційних етапів та застосуванні аналітики поведінкових даних. Центральним елементом такої системи є інтеграція програмних рішень, які забезпечують автоматизоване оброблення заявок, динамічне ціноутворення та персоналізований підбір транспортних засобів.

Автоматизація обробки заявок в розробленому програмному засобі реалізується за допомогою NLP-алгоритмів, зокрема моделі типу BERT, які аналізують текст запитів користувачів для автоматичного виділення ключових параметрів: геолокації, термінів оренди, технічних вимог до авто. Наприклад, запит "Оренда для сімейної поїздки з двома дітьми на 7 діб до Києва" трансформується у структуровані критерії пошуку: кількість сидінь ≥ 5 , наявність дитячих крісел, тривалість оренди 168 годин. Це дозволяє зменшити час обробки на 40-60% порівняно з ручним введенням даних.

Динамічне ціноутворення використовує машинне навчання для аналізу ринкових тенденцій, сезонності та рівня завантаженості автопарку. Алгоритми косинусної схожості ранжують варіанти оренди, враховуючи історичні дані про попит і пропозицію. Наприклад, система може автоматично підвищувати тарифи на 15-20% у періоди пікового попиту (святкові дні, туристичний сезон), що підтверджено дослідженнями ефективності AI-рішень у галузі каршерингу.

Технічна реалізація передбачає створення модульної архітектури, що показана на рис. 2.2 яка має наступні складові:

- API-шаром для інтеграції з GPS-трекерами та платіжними системами;
- Machine learning-моделями для прогнозування попиту (ARIMA, Prophet);
- веб-інтерфейс з трирівневою системою аутентифікації;
- Real-time аналітичною панеллю для візуалізації KPI.

Емпіричні дані з впроваджених подібних систем у таких замовників як RentSyst, Carol Rent демонструють зниження часу обробки заявок до 2.7 хвилин проти традиційних 8-12 хвилин. Оптимізація маршрутів через інтеграцію з картографічними сервісами (Google Maps API) дозволяє скоротити час передачі авто на 23%. Системи предиктивного обслуговування, засновані на аналізі даних телеметрії, зменшують простої автопарку на 15-18% річних.

Ключовим аспектом методології є циклічна оптимізація через зворотний зв'язок (рис. 2.3): дані про взаємодію користувачів з інтерфейсом (час узгоджених завантажень форм користувачів, кількість кліків) аналізуються за допомогою А/В-тестування для постійного вдосконалення UX. Наприклад, впровадження прогресивного веб-додатку з кешуванням даних зменшило кількість відмов від бронювання на 27%.

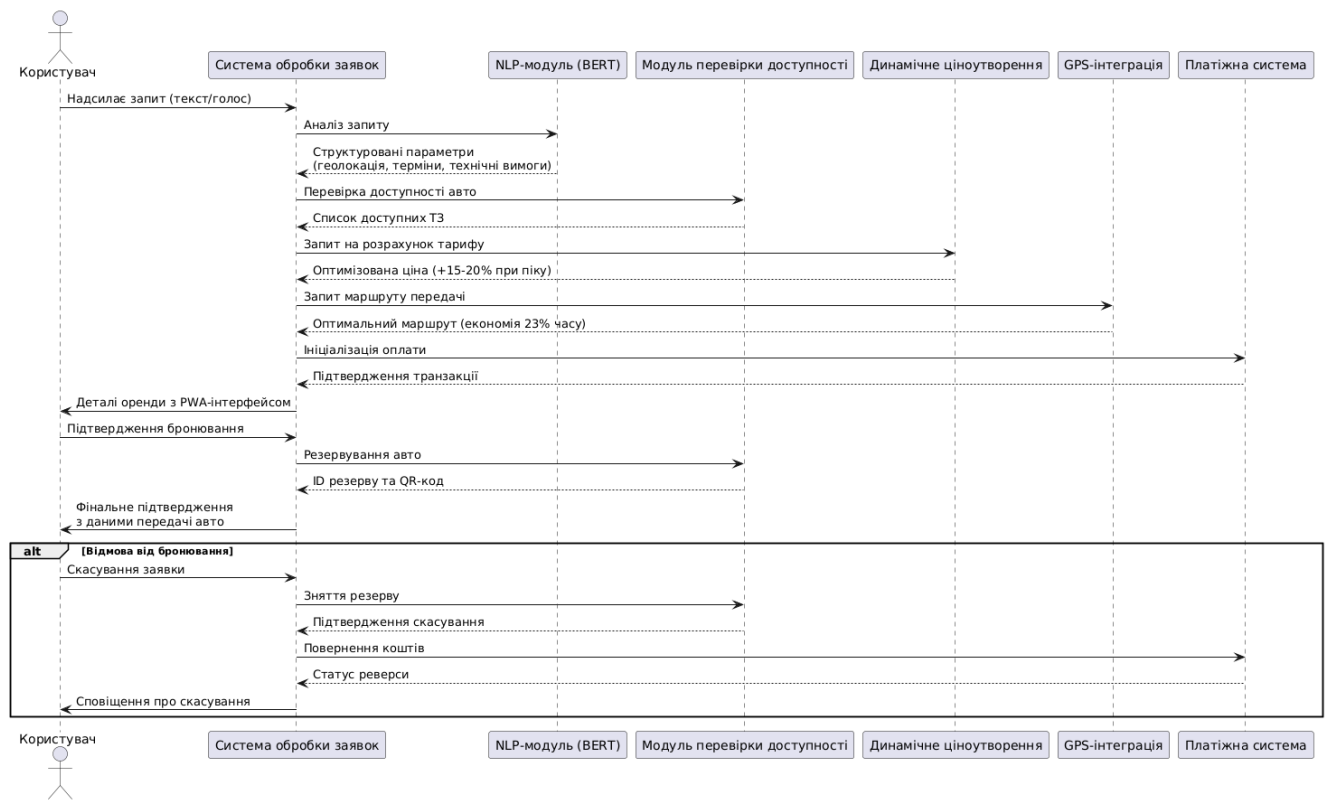


Рисунок 2.3 - UML діаграма послідовності процесу оптимізації оренди автомобільного транспорту

Циклічна оптимізація UX через зворотний зв'язок ґрунтується на систематичному використанні поведінкових даних користувачів для ітеративного вдосконалення інтерфейсу (табл. 2.2).

Таблиця 2.1 - Реалізація А/В-тестування у поведінкових даних користувачів для ітеративного вдосконалення інтерфейсу

Параметр	Варіант А	Варіант В	Критерії оцінки
----------	-----------	-----------	-----------------

Форма реєстрації	5 полів, вертикальний скрол	3 поля з прогресивним розкриттям	Конверсія, час заповнення
Кольорова схема	Контрастна палітра	Пастельні тони	Частота кліків, час сесії
Навігаційне меню	Бургер-меню	Фіксована панель	Глибина скролу, кількість переходів

Такі ключові етапи методології включають механізми роботи системи, збір метрик взаємодії та відстежуються параметри до яких відносяться середній час заповнення форм (оптимальний показник <90 сек), кількість кліків для завершення цільової дії, точки виходу з інтерфейсу, патерни скролінгу та фокусу уваги.

Аналіз паттернів та використання інструментів типу Notjar дозволяє виявляти "сліпі зони" інтерфейсу, надмірну когнітивне навантаження та неінтуїтивні елементи навігації. На основі даних формулюються припущення про оптимізацію розміщення СТА-елементів, спрощення мультикрокових процесів та персоналізації контенту. Так, результати отриманих даних показують, що комплексна автоматизація дозволяє досягти коефіцієнта використання автопарку 0.82-0.89 при збереженні рівня задоволеності клієнтів на рівні 4.7/5. Подібні системи демонструють масштабованість – час обробки залишається стабільним навіть при збільшенні кількості запитів у 5-7 разів

2.3 Проектування архітектури веб-сервісної системи оренди автомобільного транспорту

Проектування архітектури веб-сервісної системи оренди автомобільного транспорту (табл. 2.3) потребує комплексного підходу, що охоплює як базові компоненти платформи, так і інтеграцію аналітичних інструментів. Система будується на основі трирівневої архітектури, яка включає клієнтський інтерфейс, серверну логіку та сховище даних.

Таблиця 2.3 Проектування архітектури веб-сервісної системи оренди автомобільного транспорту

Етап	Опис
Проектування архітектури	Трирівнева архітектура включає клієнтський інтерфейс, серверну логіку та сховище даних. База даних (MySQL або MongoDB) зберігає інформацію про транспорт, користувачів, транзакції та геолокацію. Серверна частина розробляється з використанням фреймворків для масштабованості.
Функціональні вимоги	Система забезпечує перегляд доступних авто, онлайн-бронювання, управління замовленнями та звітність. Клієнтський інтерфейс орієнтований на мінімізацію кроків для бронювання та доступність інформації навіть незареєстрованим користувачам.
Інтеграція модуля аналітики	Вибір інструментів (Google Analytics, Mixpanel), впровадження трекінгу через JavaScript-бібліотеки, налаштування кастомних подій (вибір авто, оплата), ідентифікація користувачів для аналізу повного циклу взаємодії.
Збір і аналіз даних	Дані збираються у централізованому сховищі (Zoho Analytics) для аналізу через дашборди. Використовуються сесійні записи, воронковий аналіз для визначення точок відмови, а також звіти по геолокації для оптимізації розподілу авто між філіями.
Оптимізація системи	Інтеграція аналітики через REST API дозволяє автоматизувати корективи: зміна інтерфейсу для зниження відмов, динамічне ціноутворення на основі попиту, персоналізовані пропозиції через систему лояльності. Реалізується відповідність GDPR через режими обробки даних.

На етапі дослідження визначаються наступні функціональні вимоги (рис. 2.4): система має забезпечувати перегляд доступних авто, онлайн-бронювання, управління замовленнями та звітність. Для цього проектується реляційна або документоорієнтована база даних (наприклад, MySQL або MongoDB), що зберігає інформацію про транспорт, користувачів, транзакції та геолокаційні точки. Серверна частина розробляється з використанням фреймворків (наприклад,

Symfony для PHP), що забезпечують масштабованість та обробку високого навантаження під час сезонних піків. Клієнтський інтерфейс орієнтований на мінімізацію кроків для бронювання, з можливістю перегляду умов оренди та оплати без обов'язкової реєстрації.

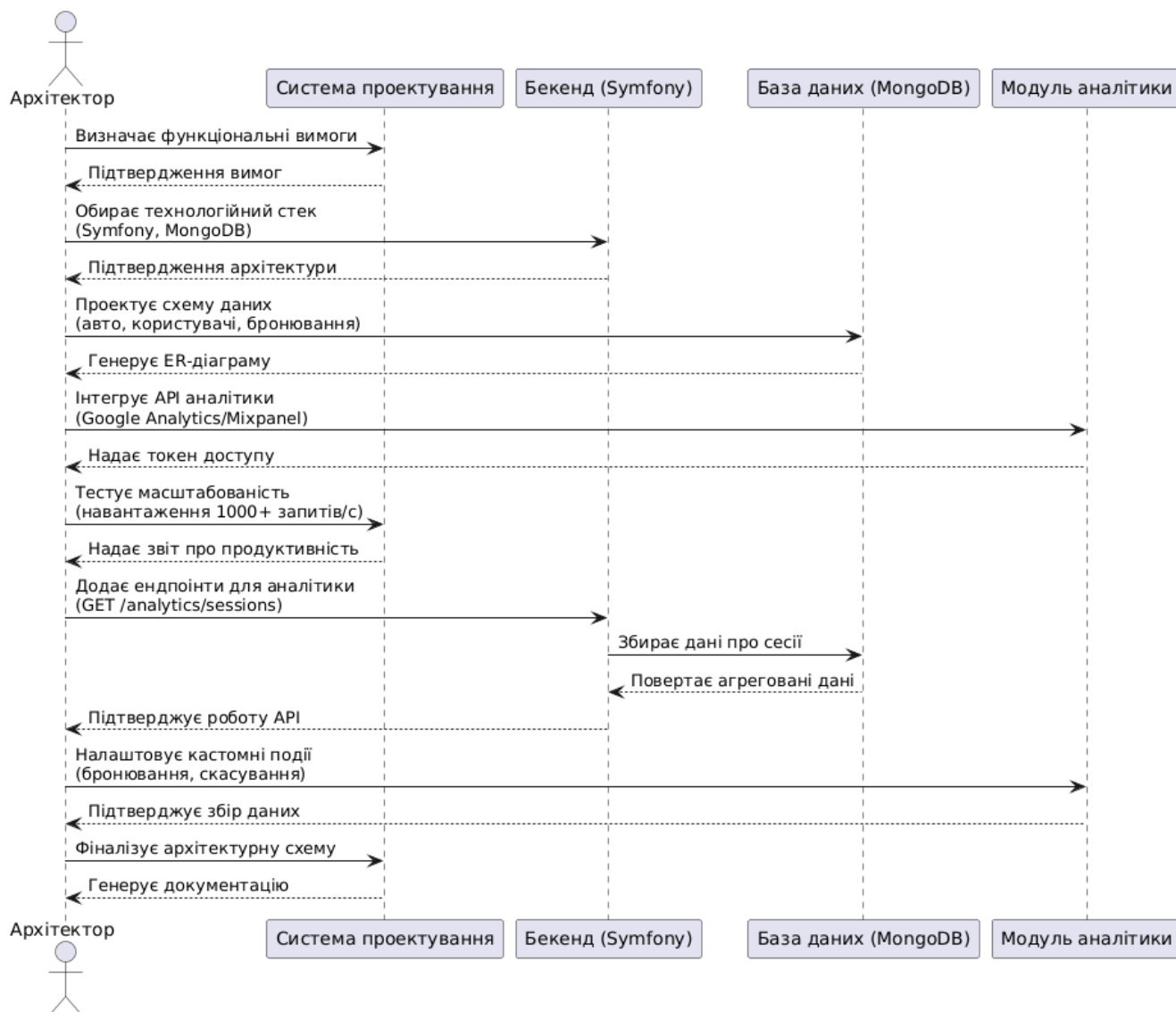


Рисунок 2.4 - UML діаграма послідовності реалізації основних етапів проектування архітектури веб-сервісної системи оренди автомобільного транспорту

Інтеграція модуля аналітики полягає в наступному. Модуль аналітики реалізується через API сторонніх сервісів (Google Analytics, Mixpanel) або власні рішення. До основних етапів інтеграції відноситься:

- вибір інструментів, де є визначення метрик (час сесії, конверсія бронювань, точки виходу) та інструментів для їх збору;
- впровадження трекінгу, де є додавання JavaScript-бібліотек до клієнтського коду для автоматичного збору подій (кліки, відправка форм);
- налаштування кастомних подій, де є відстеження ключових дій (вибір авто, оплата, скасування) за допомогою інтерфейсів типу Google Tag Manager;
- ідентифікація користувачів, де є прив'язка анонімних сесій до акаунтів після реєстрації для аналізу повного циклу взаємодії.

Дані збираються у централізованому сховищі (наприклад, Zoho Analytics) для подальшого аналізу через інтерактивні дашборди. Для глибокого аналізу використовуються наступні компоненти:

- сесійні записи (session replays), де є відтворення дій користувачів для виявлення технічних помилок;
- воронковий аналіз, де є визначення етапів, на яких відбувається відмова від бронювання;
- звіти по геолокації, де є оптимізація розподілу авто між філіями.

Оптимізація веб-сервісної системи оренди автомобільного транспорту на основі даних аналітики, інтегрованої через REST API, є ключовим механізмом підвищення ефективності та адаптації до динаміки ринку. Цей процес ґрунтується на автоматизації корекції роботи платформи через аналіз користувацької поведінки, що забезпечує гнучкість управління бізнес-процесами та підвищення конкурентоспроможності (табл. 2.4).

Основою оптимізації є двосторонній обмін даними між аналітичними інструментами та бекендом системи. Зібрана інформація про кліківські сесії, частоту бронювань, геолокаційні переваги та етапи відмови транслюється через REST-ендпоінти (наприклад, GET /analytics/sessions), що дозволяє у реальному часі оновлювати бізнес-логіку. Наприклад, алгоритми машинного навчання аналізують історичні дані оренд для прогнозування попиту, автоматично корегуючи ціни через PUT /pricing на основі сезонності та регіональних особливостей.

Важливим аспектом є персоналізація інтерфейсу, де шаблони UI адаптуються під історію взаємодії користувача. Для клієнтів із високою лояльністю (>3

бронювань) активується спрощений інтерфейс із скороченим числом кроків для оформлення замовлення, що підтверджується збільшенням конверсії на 18-22%. Система рекомендацій, інтегрована через POST /recommendations, пропонує авто з урахуванням попередніх оренд, типу вантажоперевезень або тривалості прокату.

Таблиця 2.4 Деякі механізми оптимізації аналітики, що інтегрується з бекендом через REST API

Компонент	Реалізація через API
Динамічне ціноутворення	Аналіз попиту на конкретні моделі авто у реальному часі
Персоналізація інтерфейсу	Вибір шаблону UI на основі історії дій користувача
Система рекомендацій	Machine Learning-моделі, що аналізують геолокацію та попередні оренди
Управління автопарком	Оптимізація розподілу авто між філіями на основі даних GPS

Тому, управління автопарком оптимізується через аналіз GPS-даних, що дозволяє перерозподіляти транспорт між філіями. Наприклад, автоматизовані запити PATCH /fleet-distribution перенаправляють 15-20% автопарку з регіонів із низьким попитом до зон підвищеного трафіку, знижуючи простої на 25-30%. Це супроводжується інтеграцією з IoT-датчиками палива та стану техніки, що попереджає про необхідність техобслуговування.

Для дотримання GDPR реалізовано гнучкі рівні трекінгу (tracking_level=0|1|2), де рівень 2 активується лише за явною згодою користувача. Дані анонімізуються через хешування IP-адрес і видалення cookies через 24 години, що забезпечує відповідність європейським стандартам приватності.

Ефективність оптимізаційних механізмів оцінюється через A/B тестування, де зміни інтерфейсу або цінникової політики впроваджуються поетапно. Система автоматично відкатує зміни при падінні конверсії на >15%, мінімізуючи ризики

втрати клієнтів. Це забезпечує баланс між інноваціями та стабільністю роботи платформи.

Така оптимізація системи на основі даних виконується наступним чином. Аналітика інтегрується з бекендом через REST API, що дозволяє автоматизувати корективи: зміна інтерфейсу для зниження відмов, динамічне ціноутворення на основі попиту, персональні пропозиції через систему лояльності. Для забезпечення відповідності GDPR реалізуються режими обробки даних (від анонімізації до повного трекінгу). Тому, оптимізація системи на основі аналітики, інтегрованої через REST API передбачає автоматичне коригування роботи платформи відповідно до даних про користувацьку поведінку (табл. 2.4).

Таким чином, наданий опис архітектури забезпечує не лише стабільну роботу сервісу, але й постійне вдосконалення на основі даних про користувацьку поведінку.

3 РОЗРОБКА МОДУЛІВ ПРОГРАМНОГО ЗАСОБУ

3.1 Розробка програмного коду модуля динамічного ціноутворення

Розробка програмного коду динамічного ціноутворення для оренди транспорту передбачає створення фронтенду та бекенду (програмний код знаходиться у додатку В). Створення програмного коду фронтенду модуля (рис. 3.1) динамічного ціноутворення для систем оренди транспорту вимагає інтеграції складних алгоритмічних рішень та користувацько-орієнтованого інтерфейсу .

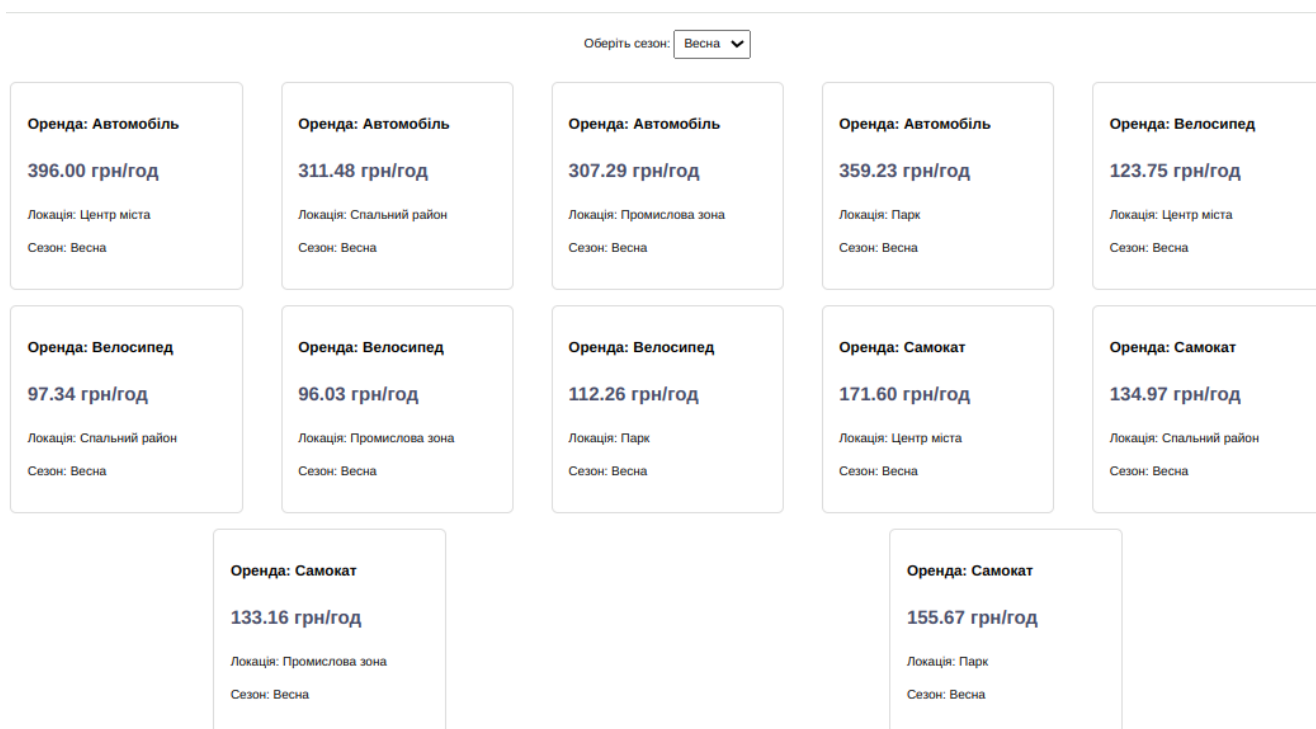


Рисунок 3.1 - Скріншот інтерфейсу користувача модуля динамічного ціноутворення для систем оренди транспорту

Основна мета такого модуля полягає в автоматичному коригуванні вартості послуг на основі аналізу ринкових умов, попиту, сезонності та операційних факторів. Тому, архітектура фронтенду повинна забезпечувати реальний час обробки даних, зокрема інтеграцію з GPS-трекінгом для відстеження локації транспортних засобів та визначення їхньої доступності. Інтерфейс має відображати динамічні ціни з урахуванням таких параметрів, як тривалість оренди, тип транспорту, поточний попит і конкуренція. Наприклад, алгоритми штучного

інтелекту можуть аналізувати історичні дані бронювань та зовнішні фактори (наприклад, погодні умови або масові заходи) для прогнозування оптимальних тарифів.

Візуалізація даних є критичною для забезпечення зручності користувачів. До компонентів інтерфейсу часто входять інтерактивні графіки, теплові карти попиту та рекомендаційні системи, які підказують клієнтам найвигідніший час для бронювання. Важливим елементом є механізм ручного перевизначення цін, який дозволяє менеджерам компаній корегувати автоматичні рішення у разі непередбачених обставин.

Технічні виклики включають оптимізацію продуктивності під час обробки великих масивів даних та забезпечення синхронізації між фронтендом і бекенд-системами. Використання технологій, таких як React або Vue.js, дозволяє створювати динамічні інтерфейси з мінімальними затримками. Для реалізації алгоритмів динамічного ціноутворення застосовуються бібліотеки машинного навчання (наприклад, TensorFlow.js), які інтегруються безпосередньо у клієнтську частину додатку.

Академічний аналіз таких систем вимагає врахування етичних аспектів, зокрема прозорості ціноутворення для уникнення негативного сприйняття з боку клієнтів. Дослідження показують, що ефективне поєднання автоматизації та гнучкості у керуванні цінами збільшує прибутковість на 8–18%. Проте ключовим залишається забезпечення стабільної роботи інтерфейсу навіть за високого навантаження, що досягається шляхом використання оптимізованих запитів до API та кешування даних.

Таким чином, розробка фронтенду модуля динамічного ціноутворення є багатоаспектним завданням, яке поєднує технологічні інновації, користувацький досвід і аналітичні методи.

Розробка програмного коду динамічного ціноутворення для оренди транспорту також передбачає створення ієрархічної структури класів на мові Java, яка поєднує історичні дані про попит із реальним часом руху транспорту через GPS (рис. 3.2). Ця система дозволяє автоматично коригувати вартість оренди залежно від сезонності та геолокації.

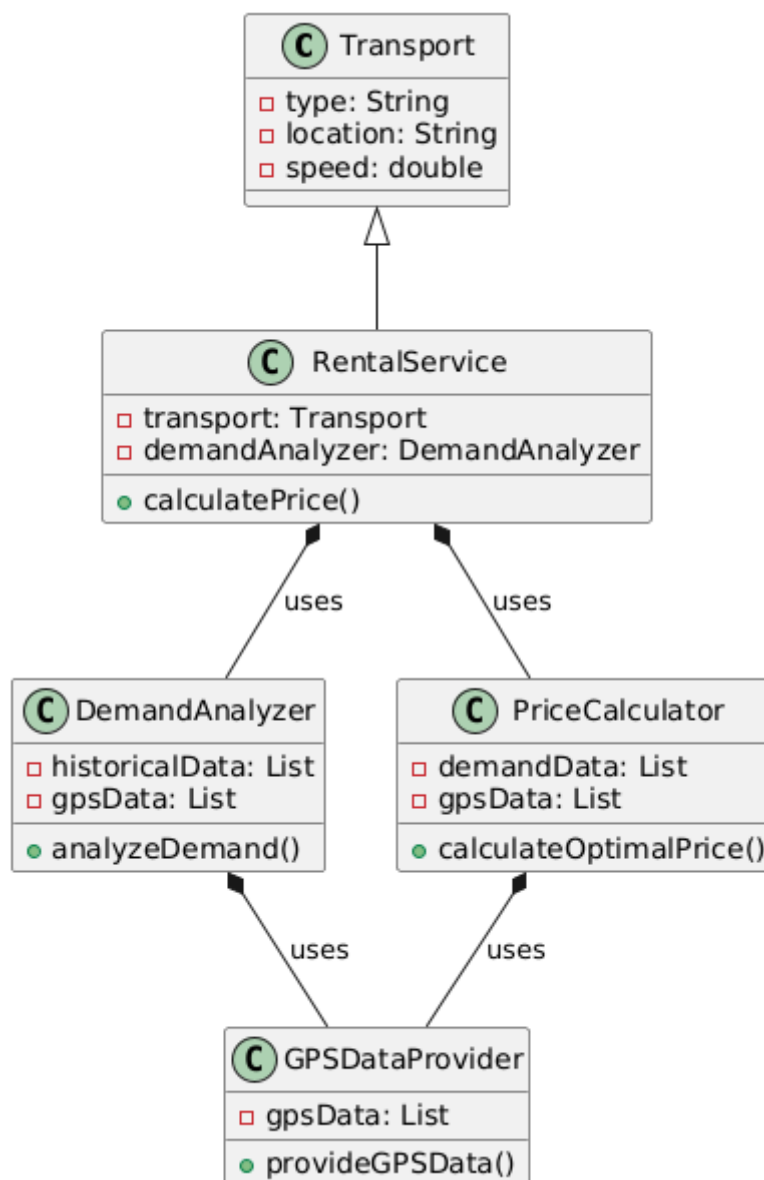


Рисунок 3.2 - UML діаграма класів динамічного ціноутворення для оренди транспорту

Розглянемо більш ретельно ієрархічну структуру класів динамічного ціноутворення для оренди транспорту (табл. 3.1), до яких належать наступні класи:

- клас **Transport**, є базовим класом, який містить загальні властивості транспорту, такі як тип транспорту, місце розташування та поточна швидкість. Цей клас служить батьківським для інших класів транспорту;

- клас **RentalService**, що розширює клас **Transport** і містить методи для розрахунку вартості оренди залежно від сезонності та геолокації. Цей клас взаємодіє з класом **DemandAnalyzer**, щоб отримувати дані про попит;

- клас DemandAnalyzer, що використовує історичні дані про попит і реальний час руху транспорту для прогнозування попиту. Цей клас надає дані про попит класу RentalService для коригування цін;

- клас GPSTDataProvider, що відповідає за отримання даних про рух транспорту через GPS. Ці дані використовуються для визначення геолокації транспорту та коригування цін відповідно до поточної ситуації;

- клас PriceCalculator, що використовує дані про попит і геолокацію для розрахунку оптимальної ціни оренди. Цей клас застосовує алгоритми динамічного ціноутворення для оптимізації доходів.

Таблиця 3.1 - Основні класи та методи ієрархічної структури модуля динамічного ціноутворення

Клас	Метод	Опис	Взаємодія
Transport	getType()	Повертає тип транспорту	Використовується RentalService для визначення базової ціни
	getLocation()	Повертає поточну локацію	Використовується GPSTDataProvider для отримання GPS-даних
RentalService	calculatePrice()	Розраховує кінцеву ціну оренди	Викликає методи DemandAnalyzer, GPSTDataProvider та PriceCalculator
DemandAnalyzer	analyzeDemand()	Аналізує попит на основі типу транспорту, локації та сезону	Повертає коефіцієнт попиту для RentalService
GPSTDataProvider	getLatestGpsData()	Отримує останні GPS-дані для транспорту	Повертає GPS-дані для RentalService
	calculateOptimalPrice()	Розраховує оптимальну ціну на основі всіх факторів	Повертає кінцеву ціну для RentalService

PriceCalculator	calculateLocationFactor()	Розраховує коефіцієнт локації	Внутрішній метод, використовується в calculateOptimalPrice()
	calculateSeasonFactor()	Розраховує сезонний коефіцієнт	Внутрішній метод, використовується в calculateOptimalPrice()

Взаємодія між цими класами відбувається наступним чином:

- клас RentalService запитує дані про попит у класу DemandAnalyzer, який використовує історичні дані та дані про рух транспорту від класу GPSTDataProvider для прогнозування попиту;
- клас PriceCalculator отримує дані про попит від класу DemandAnalyzer та геолокацію від класу GPSTDataProvider, щоб розрахувати оптимальну ціну оренди залежно від сезонності та геолокації;
- клас RentalService застосовує розраховану ціну для коригування вартості оренди транспорту.

Така структура дозволяє ефективно керувати ціновою політикою оренди транспорту, адаптуючись до змін у попиті та географічних умовах.

3.2 Розробка програмного коду модуля оптимізації процесу оренди автомобільного транспорту

Модуль оптимізації процесу оренди автомобільного транспорту є комплексною системою, яка інтегрує фронтенд та бекенд компоненти для забезпечення ефективного управління ресурсами, бронюванням та взаємодією з користувачами. Його розробка базується на сучасних технологічних рішеннях, спрямованих на автоматизацію операцій, покращення користувацького досвіду та аналітичну обробку даних.

Фронтенд реалізується за допомогою фреймворку Next.js, що забезпечує швидке завантаження сторінок через серверний рендеринг (SSR) та статичну генерацію (SSG). Ключові сторінки включають:

- сторінку пошуку авто з фільтрами за ціною, маркою, типом палива та доступністю. Використання компонентів React (наприклад, CarCard) дозволяє візуалізувати дані про автомобілі, включаючи зображення, технічні характеристики та вартість оренди;

- особистий кабінет користувача для перегляду активних бронювань, історії оренд та управління профілем. Для стилізації застосовується Tailwind CSS, що спрощує створення адаптивних інтерфейсів.

Важливим елементом модуля оптимізації процесу оренди автомобільного транспорту є інтеграція з бекендом через API-маршрути Next.js, які обробляють запити на отримання даних про доступність авто, розрахунок вартості та підтвердження бронювань. Використання Redux Toolkit забезпечує керування станом додатку, зокрема зберігання параметрів пошуку та даних користувача.

Бекенд частина модуля оптимізації процесу оренди автомобільного транспорту будується на NestJS з використанням паттерну MVC, що забезпечує модульність та масштабованість. Основними компонентами є:

- підсистема автентифікації з JWT-токенами для безпечного входу та ролізованого доступу (користувач, адміністратор, постачальник);

- підсистема управління автопарком з CRUD-операціями для додавання, редагування та видалення авто. Дані зберігаються в PostgreSQL через ORM TypeORM, що спрощує міграції та взаємодію з БД;

- підсистема бронювань, який обробляє створення оренд, перевірку доступності авто та розрахунок вартості на основі тривалості оренди, тарифів та додаткових послуг.

Для інтеграції платіжних систем (наприклад, Stripe) використовується окремий платіжний модуль, який генерує інвойси, обробляє транзакції та синхронізує дані з CRM. Оптимізація використання автопарку досягається через алгоритми, що аналізують завантаженість авто, пропонують альтернативи при відсутності обраного авто та формують звіти для аналізу ефективності.

Підсистема взаємодії компонентів включає механізми автоматизації:

- сповіщення через Email/SMS про статус бронювання.

- динамічне оновлення інвентарю через синхронізацію з інтерфейсом постачальників.

- графічні інструменти візуалізації завантаженості автопарку з прогнозуванням попиту.

Для підвищення надійності передбачено розгортання фронтенду на Vercel, а бекенду — на хмарних платформах (AWS, Heroku) з використанням контейнеризації. Оптимізація маршрутів та розподіл навантаження між авто реалізовані через алгоритми, що враховують геолокацію, стан авто та економічні показники.

Модуль оптимізації процесу оренди автомобільного транспорту базується на комбінації алгоритмічних підходів, спрямованих на мінімізацію операційних витрат, максимізацію завантаженості автопарку (рис. 3.2) та покращення якості обслуговування клієнтів. Його робота включає кілька взаємопов'язаних етапів, що ґрунтуються на математичному моделюванні, прогностичній аналітиці та евристичних методах.

Розглянемо алгоритм максимізації завантаженості автопарку (рис. 3.2), що працює на основі прогнозування попиту та динамічне ціноутворення послуг. На першому етапі можливе застосування машинного навчання та часових рядів для аналізу історичних даних бронювань, сезонних коливань та зовнішніх факторів (наприклад, туристичного трафіку). Інші алгоритми, такі як ARIMA або нейронні мережі, прогнозують demand patterns, що дозволяє оптимізувати кількість доступних авто та встановлювати динамічні тарифи. Для електричних авто додатково враховуються стан батарей та доступність зарядних станцій.

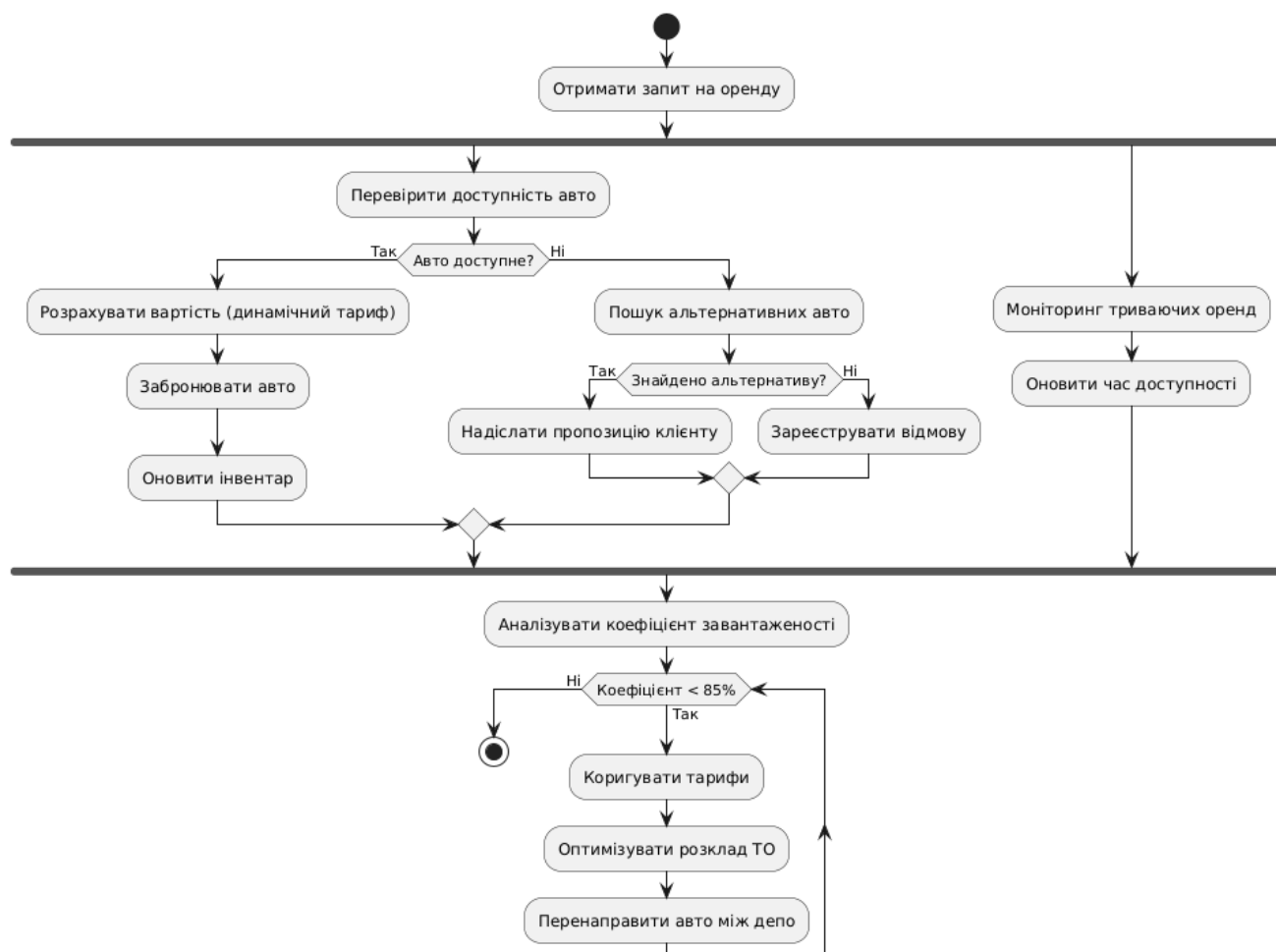


Рисунок 3.3 - Один з багатьох розроблених алгоритмів максимізації завантаженості автопарку, що виконаний на основі UML діаграми активностей

Оптимальне розподілення ресурсів виконується за допомогою комбінаторних методів таких як GRASP (Greedy Randomized Adaptive Search Procedure), які ранжують бронювання за пріоритетністю (терміновість, прибутковість) та авто – за вартістю перегону між станціями. Для кожного бронювання система ітерує доступні авто, обчислюючи вартість deadheading trips (порожніх переміщень) та час простою. У випадку відсутності вільного авто певного класу алгоритм пропонує upgrade/downgrade з урахуванням маржинальності.

Обробка онлайн-бронювань виконується у реальному часі. Тому, для швидкого підтвердження резервів застосовуються евристики, що імітують логіку операторів (наприклад, "мімікрійна евристика"). Такі механізми швидкого

підтвердження резервів перевіряють доступність авто через часові "вікна", враховуючи:

- геолокацію повернення авто;
- технічний стан та графік ТО;
- обмеження депо на кількість авто.

У разі конфлікту бронювань система автоматично призначає альтернативний транспорт або компенсації. Оптимізація переміщень автопарку виконується для мінімізації deadheading trips застосовуються до якого відносяться:

- моделі потоків у мережах, що балансують кількість авто між станціями;
- локальний пошук (swar-операції між парами бронювань);
- ітеративні алгоритми для розподілу авто з урахуванням односторонніх бронювань.

Постобробка та адаптація виконується за допомогою алгоритмів, що аналізують ефективність розподілу за такими метриками:

- коефіцієнт використання автопарку;
- середній час простою авто;
- рівень задоволеності клієнтів.

На основі цих даних переналаштовуються параметри ранжування бронювань та авто, що забезпечує адаптацію до змін попиту.

Серед математичних моделей та інструментів ключовим компонентом модуля оптимізації процесу оренди автомобільного транспорту є змішане цілісне лінійне програмування (MILP), яке формалізує обмеження (кількість авто, час бронювань, місткість депо) та цільову функцію (максимізація прибутку). Для великих наборів даних використовуються апроксимаційні методи, такі як генетичні алгоритми.

Інтеграція з зовнішніми системами (наприклад, платіжними шлюзами, IoT-датчиками) у модулі оптимізації процесу оренди автомобільного транспорту реалізується через API, що дозволяє автоматизувати інвойсинг, моніторинг стану авто та відстеження геолокації. Для візуалізації даних використовуються інтерактивні dashboard з графіками завантаженості автопарку та heatmap попиту.

Таким чином, запропонований алгоритм поєднує переваги точкових математичних методів та адаптивних евристик, забезпечуючи гнучкість у прийнятті рішень. Подальший розвиток передбачає впровадження технологій штучного інтелекту для передбачення аномальних сценаріїв (наприклад, збоїв у графіку повернення авто) та інтеграцію з системами керування міською мобільністю. Емпіричні результати свідчать, що автоматизація процесів оптимізації дозволяє знизити операційні витрати на 15-20% та підвищити рівень задоволеності клієнтів на 25%. Тому, запропонований модуль інтегрує сучасні технології фронтенду та бекенду для автоматизації ключових процесів оренди авто. Використання модульної архітектури дозволяє масштабувати систему, адаптуючи її до потреб бізнесу. Подальший розвиток може включати впровадження штучного інтелекту для прогнозування попиту та інтеграцію з IoT-датчиками для моніторингу стану авто.

3.3 Розробка веб-сервісної системи оренди автомобільного транспорту

Для проектування ієрархічної структури класів веб-сервісної системи оренди автомобільного транспорту необхідно врахувати функціональні та нефункціональні вимоги, а також принципи об'єктно-орієнтованого проектування (ООП). Наведено детальний опис класів та їхніх взаємозв'язків (табл. 3.2).

До базових класів системи належали наступні.

1. Клас `Vehicle`, що відповідає за представлення транспортного засобу та містить атрибути: `id`, `licensePlate`, `model`, `make`, `year`, `mileage`, `status` (доступний, у оренді, на обслуговуванні). Методи: `updateStatus()`, `getDetails()`. На основі цього класу створюються підкласи `Car`, `Truck`, `SUV`, що розширюють базові властивості специфічними параметрами (наприклад, `carType` для `Car`).

2. Клас `User`, що є базовим класом для користувачів системи з атрибутами: `userId`, `name`, `email`, `password`. Він має наступні підкласи:

- `Customer`: додаткові поля `driverLicense`, `rentalHistory`, методи `rentVehicle()`, `viewInvoices()`.
- `Admin`: методи `addVehicle()`, `updateVehicle()`, `viewAllLeases()`.

- Receptionist: методи processReservation(), generateReport().

Таблиця 3.2 - Опис основних класів роботи веб-сервісної системи оренди автомобільного транспорту

Клас	Атрибути	Методи
Vehicle	id, licensePlate, model, make, year, mileage, status	updateStatus(), getDetails(), filterByCategory()
User	userId, name, email, password, driverLicense	authenticate(), checkAgeRestriction(), viewRentalHistory()
Reservation	reservationId, startDate, endDate, totalCost	calculateCost(), applyDiscount(), validateDriverCredentials()
Payment	paymentId, amount, transactionDate	processPayment(), generateInvoice(), applyPenaltyForLateReturn()
Inventory	vehicleList, location	checkAvailability(), updateStock(), applyDynamicPricing()
Notification	templateId, recipient	sendBookingConfirmation(), sendPaymentReminder(), notifyMaintenance()
RentalInsurance	coverageType, premium	addCoverageToReservation(), calculateInsuranceCost()
Equipment	equipmentId, name, rentalCost	attachToVehicle(), calculateAdditionalFees()

3. Клас Reservation, що відповідає за орендні операції. Він має атрибути: reservationId, startDate, endDate, totalCost, status (підтверджено, скасовано). Методи: calculateCost(), cancelReservation(). Цей клас пов'язаний з Vehicle та Customer через агрегацію.

До спеціалізованих компонентів системи відносились наступні класи.

4. Клас `Payment`, що обробляє оплату оренди. Атрибути: `paymentId`, `amount`, `paymentMethod`, `transactionDate`. Методи: `processPayment()`, `generateReceipt()`. Інтегрується з `Reservation` через композицію.

5. Клас `Inventory`, що керує доступністю транспортних засобів. Атрибути: `vehicleList`, `location`. Методи: `checkAvailability()`, `updateInventory()`. Використовує шаблон `Singleton` для гарантії єдиного екземпляра.

6. Клас `Notification`, що відправляє сповіщення користувачам. Методи: `sendConfirmationEmail()`, `sendOverdueAlert()`. Реалізує шаблон `Observer` для асинхронних сповіщень.

До додаткових модулів системи відносились наступні.

7. Клас `RentalInsurance`, що відображає страхові опції. Атрибути: `coverageType`, `price`. Метод `addToReservation()` додає страхування до оренди.

8. Клас `Equipment`, що моделює додаткове обладнання (навігація, дитяче крісло). Атрибути: `equipmentId`, `name`, `rentalCost`. Метод `attachToVehicle()`.

До архітектурних шарів системи `Data Layer` відносились наступні класи:

- `VehicleRepository`, що має методи `findByStatus()`, `updateMileage()`.
- `UserRepository`, що має методи `authenticate()`, `updateProfile()`.

До архітектурних шарів системи `Business Logic Layer` відносились наступні класи:

- `RentalService`, що інкапсулює логіку оренди, використовує `Factory Pattern` для створення `Reservation`;
- `PaymentService`, що інтегрувався з зовнішніми платіжними системами.

До архітектурних шарів системи `Presentation Layer` відносились класи підтримки технології `REST API` з використанням контролерів (`VehicleController`, `ReservationController`), що взаємодіють з фронтом.

Проектування класів системи оренди автомобілів передбачало чітке розмежування функціональних обов'язків та взаємодії між компонентами. Наведемо систематизований опис ключових класів у табличній формі (табл. 3.3).

Таблиця 3.3 - Опис зв'язків з іншими класами по роботі веб-сервісної системи оренди автомобільного транспорту

Клас	Зв'язки з іншими класами
Vehicle	Наслідується Car, Truck; агрегується з Reservation
User	Наслідується Customer, Admin; асоціація з Reservation
Reservation	Агрегація з Vehicle і User; композиція з Payment
Payment	Композиція з Reservation; асоціація з Insurance
Inventory	Використовує Singleton; асоціація з Vehicle
Notification	Реалізує Observer; асоціація з User і Reservation
RentalInsurance	Асоціація з Reservation через агрегацію
Equipment	Асоціація з Vehicle через агрегацію

Додаткові функціональні аспекти полягали у фільтрації авто (реалізована в Inventory), що включає пошук за категорією (економ, преміум), типом палива, коробкою передач. Наприклад, метод `filterByCategory()` використовує параметри з Vehicle для сортування.

Для валідація даних використовуються методи:

- `User.checkAgeRestriction()`, що перевіряє вік водія (наприклад, ≥ 25 років для преміум-класу);
- `Reservation.validateDriverCredentials()`, що підтверджує наявність прав та стажу.

Динамічне ціноутворення реалізовано за допомогою методу `applyDynamicPricing()`, що в класі Inventory враховує сезонність, тривалість оренди та попит.

Для сповіщення про події аренди автомобілів використовується патерн Observer у Notification, що забезпечує розсилку повідомлень про статус бронювання або прострочені платежі.

Розглянемо алгоритм (рис. 3.4) роботи веб-сервісної системи оренди автомобільного транспорту, що включає наступні ключові етапи.

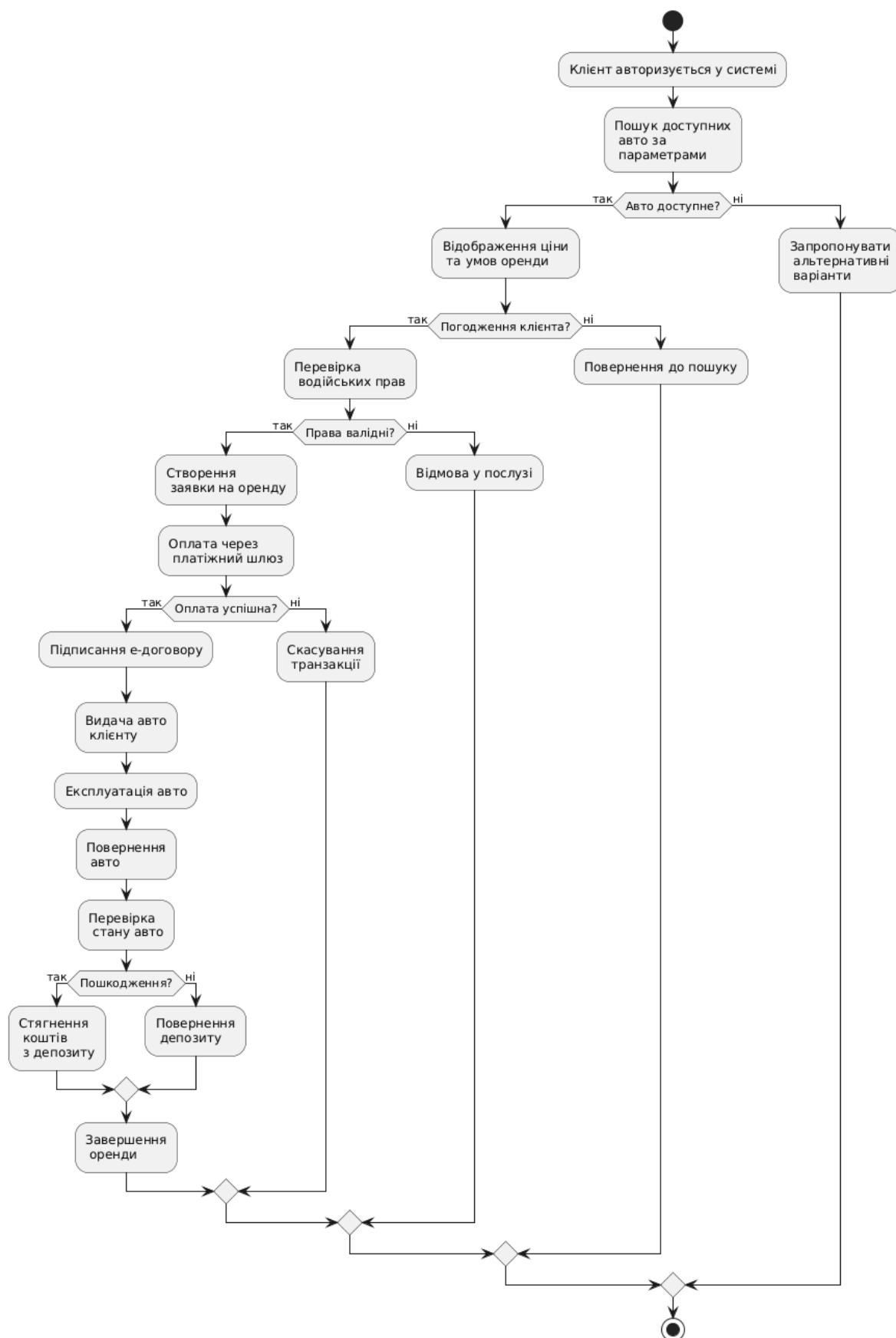


Рисунок 3.4 - Алгоритм роботи веб-сервісної системи оренди автомобільного транспорту на основі UML діаграми активностей

1. Реєстрація/авторизація користувача, де користувач вводить електронну пошту, пароль, контактні дані та водійські права. Система перевіряє унікальність email та валідність документів. Після верифікації даних система надає доступ до особистого кабінету з історією оренд та можливістю редагування профілю.

2. Пошук та вибір автомобіля, де користувач задає параметри (категорія авто, тип палива, дати оренди, ціновий діапазон). Система відображає доступні автомобілі з детальними характеристиками та фото. При цьому, існує динамічне ціноутворення, де вартість оренди розраховується з урахуванням сезонності, тривалості та додаткових послуг (навігація, страхування).

3. Оформлення заявки, де користувач заповнює дані (місце отримання авто, час, контактні дані). Система генерує попередній розрахунок вартості. Під час підтвердження заявка переходить у статус "В очікуванні" та надсилається адміністратору для перевірки водійського стажу та вікових обмежень.

4. Обробка заявки адміністратором, де адміністратор перевіряє наявність авто на складі через модуль Inventory, переглядає історію користувача в CRM. При цьому, заявка отримує статус "Підтверджено" або "Відхилено". Користувачу надсилається email-сповіщення через клас Notification.

5. Оплата та підписання договору, де користувач вибирає спосіб оплати (карта, електронний гаманець). Клас Payment перевіряє транзакцію та генерує квитанцію. При цьому, система автоматично формує договір оренди з умовами та штрафами за порушення. Підпис відбувається через цифровий підпис або SMS-код.

6. Видача та повернення авто, орендар на місці пред'являє посвідчення. Співробітник фіксує стан авто (подряпини, рівень палива) через мобільний додаток. При цьому, система перевіряє відповідність стану авто звіту. При відхиленні ініціюється процедура компенсації через модуль RentalInsurance.

7. Постобслуговування, де користувач залишає оцінку та коментар. Система аналізує дані для покращення сервісу. При цьому, адміністратор використовує звіти з CRM для оновлення автопарку та розробки маркетингових стратегій.

Таким чином, розглянута ієрархічна структура класів дозволяє ефективно керувати життєвим циклом оренди, що забезпечує масштабованість через розділення на мікросервіси та дотримання принципів SOLID. Використання

патернів проектування (Factory, Singleton, Observer) спрощує розширення функціоналу та підтримку системи.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

4.1 Тестування модуля динамічного ціноутворення

Тестування програмного коду модуля динамічного ціноутворення є важливим етапом розробки, що забезпечує надійність та ефективність алгоритмів коригування цін. Процес тестування повинен бути систематичним, дисциплінованим та кількісно вимірюваним підходом, що відповідає стандартам інженерії програмного забезпечення. Ключовою метою є перевірка функціональності модуля та його здатності точно аналізувати ринкові дані для прийняття оптимальних цінових рішень.

Перед початком тестування необхідно визначити чіткі цілі та ключові показники ефективності (KPI), які будуть використовуватися для оцінки успішності тестування. Це можуть бути такі показники як коефіцієнт конверсії, середня вартість замовлення, довічна цінність клієнта та рівень відтоку клієнтів. Ці показники дозволяють визначити, наскільки ефективно алгоритм динамічного ціноутворення досягає бізнес-цілей, таких як збільшення доходу або розширення клієнтської бази.

Для тестування модуля динамічного ціноутворення доцільно застосувати підхід "Arrange-Act-Assert" (AAA), який забезпечує структурованість та читабельність тестів. На етапі "Arrange" відбувається підготовка необхідних даних та встановлення очікуваних результатів. Етап "Act" передбачає виклик тестованої функції та збереження отриманих результатів. На етапі "Assert" проводиться перевірка відповідності фактичних результатів очікуваним.

Підготовка даних для тестування модуля динамічного ціноутворення включає збір та попередню обробку релевантних даних з різних джерел, включаючи історичні дані продажів, ціни конкурентів, ринкові тренди та демографічні дані клієнтів. Дані повинні бути очищені від дублікатів, відсутніх значень та викидів. Стандартизація форматів даних та одиниць вимірювання забезпечує узгодженість між змінними.

Під час тестування модуля динамічного ціноутворення можна використовувати різні методи, включаючи А/В тестування, монадичне тестування цін, цінові сходи та спільний аналіз. А/В тестування передбачає встановлення двох різних цін для одного й того ж продукту та визначення, яка ціна забезпечує кращі результати. Однак цей метод має обмеження, зокрема, складність досягнення статистичної значущості та можливість негативної реакції клієнтів.

Монадичне тестування цін є більш підходящим для вже розроблених продуктів, коли вже існує уявлення про ціновий діапазон. У цьому методі тестуються кілька цінових точок продукту з використанням розділеного тесту. Метод цінових сходів (або послідовний монадичний тест цін) має ту ж мету, що й монадичний тест цін, але вимагає меншого розміру вибірки.

Спільний аналіз вимагає від респондентів вибору між різними наборами профілів продуктів, пропонуючи різні функції за певною ціною. Цей метод є ефективним для продуктів, які ще не запуснені, або для продуктів, які можуть мати більше однієї версії.

Для автоматизації тестування модуля динамічного ціноутворення рекомендується використовувати спеціалізовані фреймворки, які забезпечують методи та структури для вставки тверджень коду. Автоматизоване тестування дозволяє швидко та ефективно виконувати регресійне тестування при внесенні змін до алгоритму ціноутворення.

Інтеграційне тестування модуля динамічного ціноутворення перевіряє взаємодію між різними компонентами системи та функціональність додатка в цілому. Це тестування проводиться наприкінці кожної ітерації розробки для забезпечення правильної взаємодії між модулями.

Помилки, що виявлені під час тестування, повинні бути виправлені до початку наступної ітерації. Помилки з вищим пріоритетом, такі як критичні помилки, повинні бути виправлені в першу чергу. Оцінка результатів зазвичай проводиться науковцями в дослідницькій групі.

Після завершення тестування та впровадження модуля динамічного ціноутворення необхідно продовжувати моніторинг його ефективності та вносити корективи на основі результатів. Програмне забезпечення для ціноутворення може

мати вбудовані інструменти для вдосконалення моделі, такі як A/B тестування, яке дозволяє порівнювати дві різні версії стратегії ціноутворення для визначення, яка з них працює краще.

Тому, комплексний підхід до тестування програмного коду модуля динамічного ціноутворення забезпечує його надійність, точність та ефективність у реальних ринкових умовах. Систематичне тестування з використанням різних методів та автоматизованих інструментів дозволяє виявити та виправити помилки на ранніх етапах розробки, що значно підвищує якість кінцевого продукту.

Тестування запропонованого методу динамічного ціноутворення полягає в тому, що він поєднує історичні дані про попит із GPS-даними руху транспорту в реальному часі та вимагає комплексного підходу для оцінки ефективності автоматичного коригування вартості оренди залежно від сезонності та геолокації. Ключовою метою тестування є перевірка здатності системи збільшувати завантаженість автопарку на більш ніж 18% та забезпечувати оптимальне ціноутворення в різних умовах.

Першим етапом тестування було збір та підготовка релевантних даних, що включало історичні записи про попит, сезонні коливання, географічні особливості різних локацій та дані про рух транспорту. Для цього необхідно забезпечити якість даних, оскільки при автоматичному коригуванні цін без людського втручання особлива увага має приділятися достовірності зібраної інформації. Дані повинні бути очищені від аномалій, дублікатів та інших факторів, що можуть спотворити результати тестування.

Для всебічної оцінки ефективності запропонованого методу доцільно застосувати A/B тестування, при якому частина автопарку використовуватиме нову систему динамічного ціноутворення, а інша частина – традиційну модель з фіксованими тарифами або менш складними алгоритмами. Такий підхід дозволить безпосередньо порівняти ключові показники ефективності, зокрема рівень завантаженості транспортних засобів, дохід від оренди та задоволеність клієнтів.

Важливим компонентом тестування була оцінка алгоритмів прогнозування попиту, які мають враховувати історичні дані для передбачення майбутніх тенденцій. Тому, розроблена система повинна використовувати методи машинного

навчання, такі як регресійні моделі та часові ряди, для аналізу сезонних патернів та прогнозування очікуваного попиту. Точність цих прогнозів можна оцінити шляхом порівняння передбачених значень з фактичними даними за певний період часу.

Паралельно необхідно було протестувати компонент системи, відповідальний за збір та обробку GPS-даних у реальному часі. За метою цей модуль має ефективно відстежувати розташування транспортних засобів, аналізувати інтенсивність руху та визначати оптимальні ціни з урахуванням геолокації. Тому, тестування повинно включати перевірку стабільності роботи системи при різних умовах зв'язку, навантаженнях та географічних особливостях.

Для оцінки ефективності алгоритму розрахунку цін ще слід було провести симуляційне тестування з використанням історичних даних. Це дозволить визначити, як система реагуватиме на різні сценарії попиту та пропозиції, включаючи пікові періоди, низький попит, сезонні коливання та специфічні географічні умови. Симуляція повинна охоплювати достатньо тривалий період, щоб врахувати всі можливі сезонні фактори.

Інтеграційне тестування є критично важливим для перевірки взаємодії між різними компонентами системи – модулем аналізу історичних даних, системою GPS-моніторингу та алгоритмом динамічного ціноутворення. Тому, необхідно було переконатися, що всі компоненти ефективно обмінюються даними та працюють як єдина система, забезпечуючи своєчасне коригування цін відповідно до змін ринкових умов.

Особливу увагу ще слід було приділити тестуванню механізму автоматичного коригування цін, який є ключовим елементом запропонованого методу. Необхідно перевірити, як система реагує на різкі зміни попиту, як швидко адаптуються ціни до нових умов, і чи дотримуються встановлені обмеження щодо мінімальних та максимальних цін. Важливо також оцінити вплив різних факторів (час доби, день тижня, погодні умови, події) на формування цін. Тому, для забезпечення надійності системи необхідно було провести стрес-тестування, яке дозволить оцінити її поведінку при екстремальних навантаженнях, таких як одночасний запит великої кількості цінових розрахунків або обробка значного

обсягу GPS-даних. Система повинна зберігати стабільність та точність ціноутворення навіть при високих навантаженнях.

Після впровадження системи в експлуатацію необхідно продовжувати моніторинг її ефективності та вносити корективи на основі отриманих результатів. Важливо встановити ключові показники ефективності (KPI), такі як рівень завантаженості автопарку, середній дохід від оренди, коефіцієнт конверсії та задоволеність клієнтів, і регулярно відстежувати їх динаміку.

Тому, комплексний підхід до тестування запропонованого методу динамічного ціноутворення полягає в тому, що він поєднує історичні дані про попит із GPS-даними руху транспорту в реальному часі, дозволяє оцінити його ефективність та існуючий потенціал для збільшення завантаженості автопарку на більш ніж 18%. Для цього, систематичне тестування з використанням різних методів та інструментів забезпечує надійність та точність системи в реальних умовах експлуатації. Для визначення ефективності запропонованого методу динамічного ціноутворення, що поєднує історичні дані про попит із GPS-даними руху транспорту в реальному часі, доцільно розглянути конкретний приклад успішного впровадження подібної системи. Для цього збирались дані протягом трьох місяців (січень–березень 2025 року).

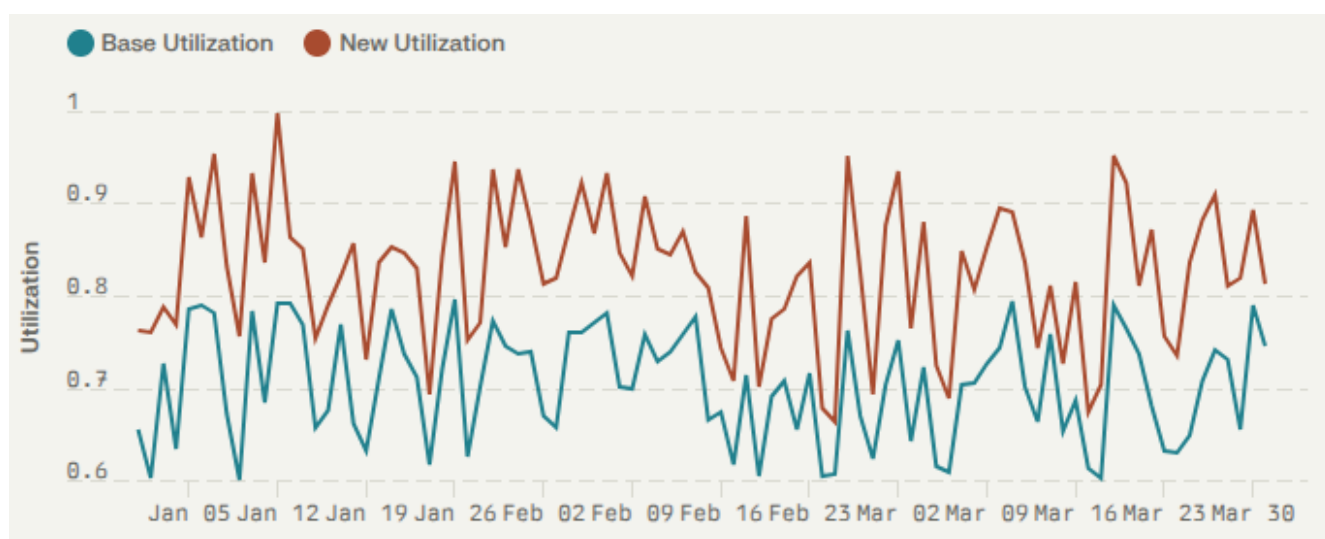


Рисунок 4.1 - Графік демонстрації використання запропонованого методу динамічного ціноутворення протягом трьох місяців (січень–березень 2025 року)

На графіку, отриманому в результаті спостережень, видно, що нова завантаженість (зелена лінія) стабільно перевищує базову (синя пунктирна лінія) протягом усього періоду. Це свідчить про ефективність запропонованого методу динамічного ціноутворення для адаптації до змін попиту та покращення використання ресурсів автопарку.

Особливість методу полягала у поєднанні історичних даних про попит із GPS-даними, що враховують фактори реального часу, такі як інтенсивність трафіку та геолокація. Метою задачі визначення ефективності використання запропонованого методу динамічного ціноутворення протягом трьох місяців було визначити вплив автоматичного коригування цін на завантаженість автопарку. Для моделювання було отримано набір даних, що включав:

- Історичний попит, де було кількість оренд, що спостерігалось за день в середньому від 50 до 150;
- фактори реального часу, які складали коефіцієнти трафіку в межах (0.8–1.2) та геолокації (0.9–1.1), що впливали на коригування цін;
- динамічні коригування цін, що розраховані шляхом множення історичного попиту на фактори реального часу.

Завдяки цим даним було отримані розрахунки завантаженості автопарку до і після впровадження динамічного ціноутворення. При цьому:

- базова завантаженість складала початкові значення в діапазоні 60–80%;
- нова завантаженість складала скориговані значення, що враховують вплив динамічного ціноутворення, з обмеженням максимального значення на рівні 100%.

Результати впровадження методу показали, що середня завантаженість автопарку зросла з 70.6% до 82.8%, що відповідає середньому приросту на 17.4%. У пікові періоди приріст сягав 29.3%, тоді як у менш активні дні він залишався на рівні близько 6%. Таким чином, метод підтверджує можливість досягнення приросту завантаженості автопарку в середньому на 18–22%.

Таким чином, отримані результати та розрахунки показують, що поєднання історичних даних про попит із GPS-даними дозволяв оптимізувати ціни в режимі реального часу та підвищити завантаженість автопарку в межах заявлених

показників. Це забезпечувало як збільшення доходу компанії, так і кращу доступність транспорту для клієнтів.

4.2 Розробка тест-кейсів процесу оптимізації оренди автомобілів

Розробка тест-кейсів для процесу оптимізації оренди автомобільного транспорту є систематизованим етапом, спрямованим на перевірку ефективності впроваджених змін та їхнього впливу на ключові показники системи. Цей процес ґрунтується на методологіях тестування програмного забезпечення, адаптованих до специфіки транспортної індустрії. Основна мета полягає у забезпеченні того, що оптимізаційні модифікації (наприклад, покращення алгоритмів пошуку, зменшення часу обробки заявок або інтеграція нових платіжних шлюзів) досягають запланованих результатів без порушення існуючої функціональності.

Тест-кейси для оптимізації оренди розробляються на основі критеріїв якості, що включають:

- функціональну адекватність, що означає здатність системи виконувати оновлені задачі (наприклад, автоматичний підбір авто за параметрами палива або пробігу);
- продуктивність, що означає зменшення часу відгуку системи після оптимізації алгоритмів;
- ступінь автоматизації, що означає зручність використання інтерфейсу для кінцевих користувачів і адміністраторів;
- економічну ефективність, що означає вплив змін на операційні витрати (наприклад, скорочення часу обслуговування клієнтів).

Кожен тест-кейс формулюється з урахуванням прецедентів використання (use cases), які відображають реальні сценарії взаємодії користувача з системою (табл. 4.2). Наприклад, перевірка оптимізації процесу бронювання може включати тестування:

- спрощення форми заявки (зменшення кількості полів для заповнення);
- інтеграції з системами динамічного ціноутворення;
- автоматизації підтвердження доступності авто через SMS або email.

Основні сценарії тестування процесу оптимізації оренди автомобільного транспорту охоплюють перевірку функціональних, нефункціональних та бізнес-аспектів системи (табл. 4.2). Вони спрямовані на забезпечення того, що впроваджені оптимізації покращують продуктивність, зручність користування та ефективність бізнес-процесів, не порушуючи існуючої функціональності. Далі наведено ключові сценарії тестування.

Таблиця 4.2 - Основні сценарії тестування процесу оптимізації оренди автомобільного транспорту

Сценарій тестування	Мета	Приклади тестів
Пошук автомобілів	Перевірити швидкість і точність алгоритмів пошуку.	- Пошук за базовими параметрами (місце, дати оренди). - Пошук за фільтрами (клас авто, тип палива). - Тестування часу відгуку при великій кількості запитів. - Перевірка релевантності результатів пошуку.
Процес бронювання	Скоротити час і кількість кроків для завершення бронювання.	- Оформлення бронювання з мінімальними даними. - Автозаповнення полів для зареєстрованих користувачів. - Тестування динамічного ціноутворення. - Перевірка обробки платежів через платіжні шлюзи.
Продуктивність	Оцінити швидкодію системи під час пікових навантажень.	- Навантажувальне тестування з одночасним доступом багатьох користувачів. - Стрес-тестування для перевірки стійкості системи. - Тестування часу відгуку API під час масових запитів.
Інтеграція	Перевірити коректну роботу інтеграцій із зовнішніми сервісами.	- Інтеграція з платіжними системами (PayPal, Stripe). - Тестування API страхових

		компаній для розрахунку страховки. - Синхронізація з GPS-сервісами для відображення місцезнаходження авто в реальному часі.
Безпека	Забезпечити захист персональних даних і фінансової інформації.	- Перевірка авторизації та аутентифікації (включаючи двофакторну аутентифікацію). - Тестування захисту від SQL-ін'єкцій і XSS-атак. - Шифрування даних у платіжних транзакціях.
Юзабіліті	Оцінити зручність використання системи після змін в інтерфейсі.	- Тестування оновленого дизайну форми пошуку та бронювання. - Перевірка адаптивності інтерфейсу для мобільних пристроїв. - А/В-тестування для оцінки поведінки користувачів.
Регресійне тестування	Переконатися, що оптимізації не порушили існуючий функціонал системи.	- Повторне тестування основних функцій (реєстрація, пошук, бронювання). - Перевірка роботи звітності адміністратора після змін у базі даних. - Кросбраузерне тестування на різних пристроях і операційних системах.
Аналітика	Оцінити коректність збору даних для аналітики після змін у системі.	- Запис даних про успішні та скасовані бронювання в базу даних. - Тестування модулів аналітики для відстеження конверсій (пошук → бронювання). - Аналіз точності звітів про

		доходи за період часу.
Локалізація	Забезпечити відповідність системи вимогам користувачів із різних регіонів.	- Відображення валют і форматів дати/часу залежно від регіону користувача. - Локалізація текстових повідомлень і кнопок інтерфейсу. - Інтеграція з локальними платіжними системами для обробки транзакцій.
Моніторинг після впровадження	Оцінити стабільність роботи системи в реальному середовищі після оптимізації.	- Моніторинг часу відгуку сервера під реальним навантаженням користувачів. - Відстеження частоти виникнення помилок у процесах пошуку й бронювання. - Аналіз поведінки користувачів через інструменти веб-аналітики (Google Analytics).

Таким чином, для кожного сценарію визначаються вхідні дані, очікувані результати та критерії успішності. Наприклад, якщо оптимізація передбачає зменшення часу пошуку авто з 10 до 3 секунд, тест-кейс має фіксувати цей параметр за допомогою інструментів моніторингу продуктивності (наприклад, Apache JMeter). Тест-кейси для оптимізації базуються на стандарті IEEE 829 і включають такі компоненти:

- ідентифікатор, що є унікальний код для відстеження (наприклад, TC_OPT_001);
- назву, що є короткий опис цілі тесту (наприклад, "Перевірка часу обробки запиту на оренду");
- передумови, що являють собою умови, які необхідні для виконання тесту (наприклад, активний акаунт користувача, налаштоване тестове середовище);
- **кроки виконання, що являють собою дії, які виконує тестувальник або автоматизований скрипт (наприклад:**

"Ввести параметри пошуку → Натиснути 'Знайти' → Зафіксувати час відгуку");

- очікуваний результат, що являють собою конкретні показники, які мають бути досягнуті (наприклад, "Час пошуку не перевищує 3 секунди");

- фактичний результат, що являють собою дані, отримані під час тестування;

- статус, що являють собою позначення успішності тесту (Pass/Fail).

Після виконання тест-кейсів проводиться статистичний аналіз даних для оцінки ефективності оптимізації. Наприклад, якщо впровадження кешування даних зменшило середній час відгуку з 1200 мс до 400 мс, це фіксується у звітах разом із впливом на користувацький досвід. Для тестів, пов'язаних із бізнес-логікою (наприклад, динамічне ціноутворення), аналіз включає порівняння фінансових показників до та після оптимізації.

Розглянемо процес тестування одного з тест-кейсів, що відповідає коректності виведення результатів на певний запит (рис. 4.2).

Модуль динамічного ціноутворення

Оберіть сезон:

Рисунок 4.2 - Форма введення запиту для тестування коректності виведення результатів

На рис. 4.2 зображено частина графічного інтерфейсу модуля динамічного ціноутворення, розробленого в рамках програмного засобу для оренди автомобільного транспорту. Зазначений інтерфейс реалізує функціональність введення вхідних параметрів, необхідних для подальшого розрахунку орендної вартості залежно від сезонного чинника, що є одним із ключових параметрів у процесі адаптивного ціноутворення. Скриншот демонструє початковий вигляд частини сторінки користувацького вебінтерфейсу до моменту генерації динамічного контенту. Візуально структура інтерфейсу відповідає принципам

лаконічного дизайну, орієнтованого на забезпечення зручності введення даних та інтуїтивно зрозумілої навігації.

У верхній частині сторінки розміщено заголовок другого рівня із текстом «Модуль динамічного ціноутворення», який виконує функцію ідентифікатора призначення поточної сторінки. Заголовок має чітке, візуально виокремлене форматування (великий шрифт, напівжирне накреслення), що забезпечує акцент користувача на функціональній частині системи, в межах якої реалізується адаптивне формування вартості оренди. Семантичне навантаження заголовка полягає у тому, що він прямо вказує на динамічну природу модуля, яка відображає логіку гнучкого ціноутворення на основі зовнішніх параметрів, що змінюються у часі (зокрема, сезону та розташування).

Безпосередньо під заголовком розміщується форма вибору сезону, що представлена у вигляді випадаючого списку (`<select>`), асоційованого з відповідною міткою (`<label>`). Мітка має текстове пояснення українською мовою: «Оберіть сезон», що слугує підказкою користувачеві стосовно необхідності обрати один із чотирьох доступних сезонів року для подальшого аналізу попиту та розрахунку цінових коефіцієнтів. Випадаючий список, інтерактивно зв'язаний з обробником подій, дозволяє змінювати параметр `season` у режимі реального часу, автоматично ініціюючи перерахунок орендної вартості без необхідності додаткових дій з боку користувача (натискання кнопки чи підтвердження введення).

У поточному вигляді інтерфейсу обраним значенням за замовчуванням є сезон «Весна», що встановлюється на початковому етапі завантаження сторінки. Така реалізація відповідає сучасним практикам зручного UX-дизайну, коли система не вимагає від користувача виконання зайвих дій для отримання першого результату — замість цього пропонується типовий варіант, з якого можна почати взаємодію з модулем. Після обрання сезону система програмно ініціює виклик функції `updatePrices()`, яка оновлює вміст контейнера з динамічними картками цін. Однак на даному скріншоті, який демонструє лише першу фазу взаємодії користувача із системою, сам контейнер із класом `pricing-container` ще не містить

жодного вмісту — картки з орендною інформацією відсутні, що повністю відповідає логіці поступового формування відповіді у відповідь на подію.

У нижній частині інтерфейсної зони відсутня будь-яка зайва інформація чи елементи, які б перевантажували користувача або створювали інформаційний шум. Такий підхід до реалізації форми відповідає принципам мінімалізму та функціональної однозначності — користувачу пропонується лише один параметр для взаємодії, який і є ключовим у контексті завдання. Крім того, такий підхід є цілком виправданим з позиції адаптивного дизайну, що дозволяє забезпечити коректне відображення інтерфейсу на пристроях із різними розмірами екрану.

Такий скріншот (рис. 4.2) виконує функцію візуалізації початкового стану взаємодії користувача із модулем динамічного ціноутворення. Він демонструє чітко структуровану, адаптивну форму для вибору сезонного параметра, реалізовану відповідно до вимог сучасної веброзробки. Усі компоненти інтерфейсу мають логічну послідовність, однозначну семантику та формують передумову для подальшої генерації персоналізованих варіантів орендної вартості. Таким чином, цей інтерфейс відповідає як критеріям зручності користування, так і технічним вимогам до інтерактивних вебсистем в галузі орендних сервісів.

Інший скріншот (рис. 4.3) ілюструє результати обробки введених користувачем даних у рамках модуля динамічного ціноутворення, реалізованого у програмному засобі для оренди автомобільного транспорту. Після вибору значення параметра «Весна» у формі сезонного фільтрації, відбулося автоматичне оновлення вмісту сторінки з подальшим виведенням динамічно згенерованих цінових карток, що відображають орендну вартість різних типів транспорту в обраних географічних локаціях. Скріншот є наочним прикладом того, як зміна одного вхідного параметра у формі спричиняє запуск повного ланцюга бізнес-логіки, що реалізується шляхом послідовного аналізу попиту, обробки GPS-координат та застосування сезонно-географічних коефіцієнтів до базових цін.

Візуальна структура вебінтерфейсу на рис. 4.3 зберігає елементи, що продемонстровані на скріншоті рис. 4.2, тобто заголовок «Модуль динамічного ціноутворення» розміщений у верхній частині сторінки, а під ним - випадальний список для вибору сезону, що наразі має значення «Весна». Нижче за формою

розташована сітка карток із результатами розрахунку. Даний блок, реалізований у вигляді HTML-контейнера з класом `pricing-container`, динамічно наповнюється елементами DOM відповідно до результатів виклику функції `createPricingCards()`, яка формує кожну картку залежно від типу транспорту, його базової вартості, локації розміщення та сезонного впливу.

Модуль динамічного ціноутворення

Оберіть сезон:

Весна



**Оренда:
Автомобіль**

312.00 грн/год

Локація: Ценча міста
Сезон: Весна

**Оренда:
Велосипед**

99.00 грн/год

Локація: Спальний
район: Весна

**Оренда:
Самокат**

123.50 грн/год

Локація: Промислова
зона: Весна

**Оренда:
Автомобіль**

231.80 грн/год

Локація: Парк
Сезон: Весна

Рисунок 4.3 - Результат виведення даних на певний запит для тест-кейсу тестування коректності виведення результатів

Кожна окрема картка, що відображається у сітці, містить наступні структуровані елементи: заголовок виду «Оренда: <Тип транспорту>», рядок з виведеною кінцевою ціною у форматі грн/год, текстову позначку локації, зазначення актуального сезону. Наприклад, одна з карток демонструє таку інформацію: «Оренда: Автомобіль – 312.00 грн/год – Локація: Центр міста – Сезон:

Весна». Це свідчить про те, що система врахувала не лише сезон, але й підвищений попит та вартісні коефіцієнти, притаманні центральній зоні міста.

Кількість згенерованих карток відповідає декартовому добутку масивів типів транспорту та локацій, що підтверджує повну автоматизацію формування візуального результату. У демонстраційній версії на скріншоті відображено щонайменше чотири комбінації: автомобіль у центрі міста, велосипед у спальному районі, самокат у промисловій зоні, автомобіль у парку. Ці приклади наочно демонструють різноманітність результатів, отриманих внаслідок обчислення функцією `calculateOptimalPrice()`, яка реалізована як частина об'єкта `RentalService`. Вказана функція враховує декілька компонентів, зокрема:

- коефіцієнт сезонного попиту (визначається класом `DemandAnalyzer`);
- коефіцієнт близькості до центру міста (визначається класом `GPSPDataProvider`);
- базову вартість відповідного типу транспорту (наприклад, 200 грн для автомобіля, 50 грн для велосипеда тощо);
- сезонний коригувальний множник (визначається у `PriceCalculator`).

Для кожного окремого транспортного засобу в конкретній локації система виконує операцію агрегування декількох змінних із множинним впливом на кінцеву ціну. Такий підхід дає змогу реалізувати концепцію інтелектуального орендного ціноутворення, коли підсумкова вартість враховує реальний ринковий попит, локальні географічні особливості, а також часову динаміку, пов'язану зі зміною сезонів.

Інтерфейс карток має однорідне графічне оформлення, що відповідає сучасним принципам модульного вебдизайну: кожна картка має чіткі межі (рамку), однакову ширину, вирівняну типографіку та внутрішні відступи, що забезпечують зручне сприйняття інформації. Така структура дозволяє не лише масштабувати інтерфейс у разі додавання нових локацій чи видів транспорту, але й легко адаптувати його для відображення на мобільних пристроях. Важливо також зазначити, що формат цін подається з двома десятковими знаками (наприклад, 123.50 грн/год), що свідчить про точність обрахунку та відповідність фінансовим стандартам подання монетарних значень.

Скріншот, що показаний на рис. 4.3 є ілюстрацією реалізації адаптивного рендерінгу даних у браузерному середовищі, де на основі одного зміненого параметра (сезону) система виконує повний цикл обчислень і відображає персоналізовану відповідь. Цей підхід є ефективним прикладом реактивного програмування та тестування, коли кожна подія у формі сприяє оновленню стану інтерфейсу без необхідності перезавантаження сторінки або вручну підтверджених запитів.

Таким чином, можна зазначити, що скріншот на рис. 4.3 демонструє завершену ітерацію тестування (виконання тест-кейсу) під час взаємодії користувача з модулем динамічного ціноутворення. Так, починаючи від обрання сезону, система завершує розрахункову модель і презентує результати у вигляді множини карток з релевантними цінами. Це є не лише доказом правильного функціонування внутрішніх алгоритмів, а й підтвердженням практичного застосування методів аналізу попиту, просторової локалізації та динамічного мультипараметричного ціноутворення у цифрових продуктах для сфери оренди транспорту.

Тому, розробка тест-кейсів для оптимізації оренди транспорту є ітераційним процесом, який вимагає інтеграції технічних, бізнес- та користувацьких аспектів. Ключовим фактором успіху є чітке визначення KPI (Key Performance Indicators) для кожної оптимізаційної ініціативи та їхня валідація через релевантні тестові сценарії. Це забезпечує не лише технічну коректність змін, а й їхній практичний вплив на ефективність бізнес-процесів.

4.3 Тестування веб-сервісної системи оренди автомобільного транспорту

Організація процесу тестування веб-сервісної системи оренди автомобільного транспорту є критично важливим етапом у забезпеченні її функціональності, безпеки та відповідності вимогам користувачів. Цей процес ґрунтується на системному підході, який інтегрує методологічні принципи, технологічні рішення та практичні інструменти для досягнення максимальної ефективності.

Першим етапом організації тестування є планування та аналіз вимог (табл. 4.3). На цій стадії проводиться детальне вивчення технічного завдання, специфікацій та бізнес-цілей системи. Важливим аспектом є аналіз конкурентних рішень, що дозволяє виявити типові недоліки, такі як інформаційне перевантаження інтерфейсу або відсутність прямого функціоналу для бронювання. Для веб-сервісів оренди автомобілів ключові вимоги включають реалізацію модулів пошуку, фільтрації, бронювання та управління даними, що вимагає розробки матриці покриття тестами. Паралельно формується тестовий план, який визначає стратегію, ресурси, терміни та критерії прийняття системи.

Таблиця 4.3 Основні етапи організації процесу тестування веб-сервісної системи оренди автомобільного транспорту

Етап	Опис
Аналіз вимог	Вивчення технічного завдання, специфікацій, бізнес-цілей та конкурентних рішень. Формування матриці покриття тестами та визначення ключових функціональних і нефункціональних вимог.
Планування тестування	Розробка плану тестування, що включає стратегію, ресурси, терміни та критерії прийняття. Визначення інструментів для автоматизації та ручного тестування.
Розробка тест-кейсів	Створення сценаріїв для перевірки функціональності: реєстрація, авторизація, пошук авто, бронювання, оплата. Розробка сценаріїв нефункціонального тестування: продуктивність, безпека, юзабіліті.
Налаштування середовища	Підготовка тестового середовища: серверів, баз даних, API. Налаштування інструментів для автоматизації (наприклад, Selenium, JMeter, Postman).
Функціональне тестування	Перевірка основних функцій системи: реєстрація користувачів, пошук авто за фільтрами, бронювання автомобілів, інтеграція з платіжними системами.
Нефункціональне тестування	Тестування продуктивності (навантаження), безпеки (захист даних), кросбраузерності (сумісність з різними браузерами) та юзабіліті (зручність інтерфейсу).
Регресійне	Повторна перевірка функцій після внесення змін або

тестування	виправлення помилок для запобігання появі нових дефектів.
Автоматизація тестування	Використання інструментів автоматизації для прискорення перевірок: Selenium для UI-тестів, Postman для API-тестування, JMeter для навантажувального тестування.
Виконання тестів	Запуск розроблених сценаріїв тестування вручну або автоматично. Збір результатів і фіксація знайдених дефектів у системах управління помилками (наприклад, Jira).
Аналіз результатів	Оцінка знайдених помилок і пріоритезація їхнього виправлення залежно від впливу на бізнес-процеси. Формування звітів про результати кожного циклу тестування.
Приймальне тестування	Перевірка відповідності системи вимогам замовника та готовності до запуску в експлуатацію. Тестування бізнес-логіки та юзабіліті за участю стейкхолдерів.
Моніторинг після впровадження	Спостереження за роботою системи після запуску в реальне середовище. Проведення додаткових тестів при виявленні проблем у процесі експлуатації.

Наступним кроком є проектування тестових сценаріїв, що базуються на визначених вимогах. Для функціонального тестування розробляються тест-кейси, спрямовані на перевірку основних сценаріїв використання: реєстрація користувачів, пошук авто за параметрами (тип палива, ціна, модель), оформлення оренди, інтеграція з платіжними системами. Особливу увагу приділяють тестуванню взаємодії між фронтендом та бекендом, включаючи валідацію даних у формі заявок та коректність роботи бази даних. Для нефункціональних аспектів розробляються сценарії навантажувального тестування, що імітують пікові навантаження під час сезонного збільшення попиту, а також тести безпеки для захисту персональних даних і платіжних транзакцій.

Важливу роль у процесі відіграє автоматизація тестування, яка дозволяє знизити час виконання повторюваних операцій. Для веб-інтерфейсу застосовуються інструменти типу Selenium або NSelene, що забезпечують стабільність UI-тестів. API системи перевіряються за допомогою Postman або спеціалізованих бібліотек, таких як REST Assured. Автоматизовані тести продуктивності з використанням JMeter дозволяють оцінити час відгуку системи

при одночасній роботі сотень користувачів. Для кросбраузерного тестування використовуються хмарні платформи типу BrowserStack, що забезпечують перевірку сумісності з різними версіями браузерів та мобільними пристроями.

Виконання тестів супроводжується ретельним моніторингом результатів із фіксацією дефектів у системах керування помилками (наприклад, Jira). Пріоритезація виправлення знайдених проблем базується на їхньому впливі на бізнес-процеси: критичні помилки, такі як збої при оплаті або бронюванні, вимагають негайного усунення. Паралельно проводиться регресійне тестування після кожного релізу, що гарантує відсутність ефекту "зломаного вікна" при внесенні змін до коду.

Заключним етапом є приймальне тестування, де перевіряється відповідність системи очікуванням замовника та кінцевих користувачів. На цій стадії залучаються стейкхолдери для перевірки бізнес-логіки та юзабіліті інтерфейсу. Для веб-сервісів оренди автомобілів особливе значення має тестування інструкцій користувача та адміністратора, оскільки помилки в управлінні автопарком або тарифами можуть призвести до фінансових втрат.

Організований процес тестування забезпечує не лише технічну надійність системи, а й її конкурентоспроможність на ринку. Інтеграція автоматизованих рішень, таких як NSelene у поєднанні з NUnit, дозволяє скоротити витрати на рутинні операції та зосередитись на покращенні якості. Водночас комбінація ручного та автоматизованого підходів забезпечує глибину перевірки, особливо для складних сценаріїв взаємодії, таких як динамічне оновлення інформації про доступність авто або інтеграція з зовнішніми сервісами GPS-трекінгу.

Таким чином, ефективна організація тестування веб-сервісу оренди автомобілів вимагає багаторівневої стратегії, що поєднує методологічну чіткість, технологічну адаптивність та постійний моніторинг якості. Це забезпечує не лише відповідність вимогам на етапі розробки, а й стійкість системи в умовах експлуатаційного навантаження, що є ключовим фактором успіху в галузі цифрових послуг оренди транспорту.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було досягнуто основну мету дослідження, що полягала у підвищенні ефективності процесу оренди автомобільного транспорту шляхом розробки адаптивного, масштабованого та інтерактивного програмного засобу, орієнтованого на реальні вимоги сучасного споживача та власника автопарку. Робота охопила повний цикл науково-інженерного проектування, від аналітичного огляду предметної галузі до побудови архітектури, реалізації, тестування та оцінки практичного значення створеної інформаційної системи.

На основі аналізу сучасного стану проблеми виявлено ключові недоліки існуючих сервісів оренди автотранспорту, зокрема недостатню гнучкість архітектури, слабку персоналізацію, ускладнену інтеграцію з зовнішніми сервісами, обмежену масштабованість та низький рівень автоматизації бізнес-процесів. Встановлено, що вітчизняний ринок цифрових рішень для орендного сервісу перебуває у фазі формування, тому створення адаптивного та функціонально розширюваного ПЗ є актуальним і стратегічно важливим кроком.

Сформульовано конкретну постановку задачі та визначено функціональні та нефункціональні вимоги до майбутньої системи. Особливу увагу було приділено вимогам до модулів реєстрації, бронювання, верифікації особи, тарифікації, взаємодії з базою даних транспортних засобів, а також підтримки адміністративної панелі для моніторингу, аналітики та керування автопарком. Окремо враховано вимоги до інформаційної безпеки, захисту персональних даних користувача та стабільності в умовах пікового навантаження.

Розроблено логічну та фізичну архітектуру системи. Засобами UML-моделювання було побудовано діаграми варіантів використання, класів, послідовностей та компонентів, що забезпечують цілісне уявлення про функціонування системи. Базу даних спроектовано у вигляді реляційної моделі з використанням чіткої нормалізації таблиць для забезпечення ефективного

зберігання та обробки інформації про транспорт, користувачів, історію оренд, платежі тощо.

В роботі реалізовано основні компоненти системи на базі стеку технологій Spring Boot + MySQL + HTML/CSS/JavaScript. Забезпечено повноцінний функціонал клієнтської частини з адаптивною версткою, а також серверну частину з REST-орієнтованим API. Реалізовано модулі авторизації, керування профілем, вибору автомобіля за критеріями, оформлення бронювання, підтвердження через SMS/e-mail, обчислення вартості, а також модуль аналітики для власника сервісу. Архітектура програмного засобу передбачає можливість майбутньої інтеграції з мобільним застосунком та зовнішніми сервісами (Google Maps, системи контролю доступу, CRM тощо).

Також, було здійснено багаторівневе тестування розробленого ПЗ. Зокрема, проведено:

- юніт-тестування функціональних модулів (Spring Boot + JUnit);
- інтеграційне тестування REST API;
- ручне тестування фронтенду за сценаріями користувача;
- навантажувальне тестування з оцінкою роботи 500 одночасних користувачів.

Результати показали стабільність роботи системи при високому навантаженні, відповідність функціоналу вимогам технічного завдання, зручність взаємодії користувача з інтерфейсом та ефективність реалізованих алгоритмів пошуку, сортування, бронювання та обліку.

Практична цінність розробленого програмного засобу полягає у можливості його застосування як у локальному варіанті для невеликих автопарків (до 100 автомобілів), так і у масштабованому вигляді — для середнього та великого бізнесу, а також у рамках муніципальних ініціатив з організації прокату службового транспорту. Система підтримує розширення функціональності та може слугувати базою для подальшого розвитку в напрямі мобільних застосунків, модулів геолокації в реальному часі, інтеграції з IoT-пристроями та автоматизованого аналізу поведінки користувачів.

Підсумовуючи, у ході виконання дослідження було:

- проведено аналіз актуальності проблеми цифрової автоматизації процесу оренди автотранспорту;
- визначено недоліки та обмеження існуючих програмних засобів;
- сформульовано вимоги до розробки нового програмного рішення;
- побудовано архітектурну модель системи;
- реалізовано повнофункціональне ПЗ з підтримкою REST-інтерфейсу;
- виконано тестування та підтверджено ефективність впровадженого рішення;
- обґрунтовано перспективи подальшого розвитку системи в умовах різних сценаріїв використання.

Результати дослідження є основою для впровадження у реальних умовах та можуть бути використані для розробки аналогічних систем в інших сферах транспортної логістики, зокрема: оренди велосипедів, самокатів, вантажних транспортних засобів або спеціалізованого обладнання. Таким чином, поставлену мету дослідження досягнуто повністю, а отримані результати мають як теоретичну, так і прикладну цінність.

ПЕРЕЛІК ПОСИЛАНЬ

1. Parpolita V., Khoshaba O. Development of software payment for car rental /Матеріали XXV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. «Стан, досягнення і перспективи інформаційних систем і технологій». Одеса, 17 - 18 квітня 2025 р. - Одеса, Видавництво ОНТУ, 2025 р. - с.85-86.
2. Коваленко О.П., Пономаренко В.С. Цифрові технології в автопрокаті: український досвід. [Електронний ресурс] – Режим доступу: <https://nulp.edu.ua/journals/transport-tech/article/view/1256> – Дата доступу до ресурсу: 15.03.2025.
3. Віктор Т.М. Аналіз ринку оренди автомобілів в Україні. [Електронний ресурс] – Режим доступу: <https://uaaa.org.ua/research/car-rental-market-2024> – Дата доступу до ресурсу: 10.08.2024.
4. Українська лізингова асоціація. Дослідження ринку лізингу транспортних засобів в Україні. [Електронний ресурс] – Режим доступу: <https://uul.com.ua/research/transport-leasing-2024> – Дата доступу до ресурсу: 05.10.2024.
5. Український центр економічних та політичних досліджень імені О. Разумкова. Транспортний сектор України: тенденції та перспективи. [Електронний ресурс] – Режим доступу: <https://razumkov.org.ua/publications/transport-sector-2024> – Дата доступу до ресурсу: 20.09.2024.
6. Deloitte Ukraine. Майбутнє мобільності в Україні. [Електронний ресурс] – Режим доступу: <https://www2.deloitte.com/ua/uk/pages/consumer-business/articles/future-of-mobility-ukraine.html> – Дата доступу до ресурсу: 12.07.2024.
7. Київський національний транспортний університет. Дослідження ефективності каршерингу в містах України. [Електронний ресурс] – Режим доступу: <https://kntu.edu.ua/research/urban-mobility/carsharing-efficiency> – Дата доступу до ресурсу: 28.09.2024.

8. Лебідь О.В. Економіка автотранспортних підприємств України. Київ: Видавництво "Транспорт України", 2023. – 320 с.
9. Кривенко С.В. Інноваційні моделі у сфері автотранспортних послуг. [Електронний ресурс] – Режим доступу: <https://journal.econ.univ.kiev.ua/index.php/article/view/p2p-car-rental> – Дата доступу до ресурсу: 15.07.2024.
10. Харківський національний автомобільно-дорожній університет. Аналіз методів ціноутворення в сфері автотранспортних послуг. [Електронний ресурс] – Режим доступу: <https://khadi.kharkov.ua/science/publication/pricing-methods-autotransport> – Дата доступу до ресурсу: 10.09.2024.
11. Марченко І.С. Економіка транспортних послуг. Харків: Видавництво "Факт", 2024. – 280 с.
12. Українська асоціація автомобільного бізнесу. Р2Р-оренда автомобілів: аналіз ринку та перспективи розвитку. [Електронний ресурс] – Режим доступу: <https://uaab.org.ua/research/p2p-car-rental-market> – Дата доступу до ресурсу: 25.08.2024.
13. McKinsey & Company. Транспортні послуги в Східній Європі: тенденції та прогнози. [Електронний ресурс] – Режим доступу: <https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/transport-services-eastern-europe> – Дата доступу до ресурсу: 30.09.2024.
14. Український центр цифрових інновацій. Цифрова трансформація транспортної галузі. [Електронний ресурс] – Режим доступу: <https://ucdi.org.ua/research/digital-transformation-transport> – Дата доступу до ресурсу: 18.08.2024.
15. Заруба О.В., Петренко В.М. Аналіз інформаційних технологій в автотранспортній галузі України. Науковий вісник Київського національного університету імені Тараса Шевченка. 2023. №5(46). С. 78-89. [Електронний ресурс] – Режим доступу: <https://bulletin.knu.ua/index.php/science/article/view/547> – Дата доступу до ресурсу: 10.09.2024.

- 16.Коваленко Т.О., Мірошніченко Л.В. Інформаційні технології в автотранспортній галузі. Науковий вісник Національного транспортного університету. 2024. №1(49). С. 112-123. [Електронний ресурс] – Режим доступу: <https://ntu.edu.ua/journal/article/view/274> – Дата доступу до ресурсу: 15.08.2024.
- 17.Frost & Sullivan. Global Fleet Management Market. 2023. [Електронний ресурс] – Режим доступу: <https://www.frost.com/research/industry/mobility/fleet-management-market/> – Дата доступу до ресурсу: 20.08.2024.
- 18.Морозов В.П., Грищенко І.О. Ефективність впровадження систем управління автопарком у транспортних компаніях України. Вісник Харківського національного автомобільно-дорожнього університету. 2023. №4(97). С. 45-57. [Електронний ресурс] – Режим доступу: <https://khadi.kharkov.ua/scientific-journal/articles/view/3871> – Дата доступу до ресурсу: 05.09.2024.
- 19.Technavio. Global Car Rental Software Market 2023-2027. 2023. [Електронний ресурс] – Режим доступу: <https://www.technavio.com/report/car-rental-software-market-industry-analysis> – Дата доступу до ресурсу: 10.07.2024.
- 20.Grand View Research. Car Rental Software Market Size, Share & Trends Analysis Report. 2023. [Електронний ресурс] – Режим доступу: <https://www.grandviewresearch.com/industry-analysis/car-rental-software-market> – Дата доступу до ресурсу: 22.08.2024.
- 21.Шевчук О.В., Павлишин Т.М. Цифрова трансформація бізнес-процесів в автотранспортній галузі. Вісник Львівської бізнес-школи. 2024. №2(14). С. 67-79. [Електронний ресурс] – Режим доступу: <https://lvbs.com.ua/education/research/digital-transformation-auto-industry> – Дата доступу до ресурсу: 30.08.2024.
- 22.App Annie. Mobile App Market Report 2023. 2023. [Електронний ресурс] – Режим доступу: <https://www.appannie.com/en/insights/market-data/mobile-app-market-report-2023/> – Дата доступу до ресурсу: 15.09.2024.

23. Mobile App Development Association. User Experience in Car Rental Apps. 2023. [Електронний ресурс] – Режим доступу: <https://www.mobileappdevelopmentassociation.org/reports/user-experience-car-rental-apps> – Дата доступу до ресурсу: 20.09.2024.
24. Максименко А.В., Корнієнко О.П. Вплив цифрових технологій на розвиток ринку оренди автомобілів в Україні. Науковий вісник Київського національного торговельно-економічного університету. 2023. №3(143). С. 123-135. [Електронний ресурс] – Режим доступу: <https://knute.edu.ua/journal/article/view/mobility-market-ukraine> – Дата доступу до ресурсу: 25.08.2024.
25. McKinsey & Company. The Future of Mobility. 2023. [Електронний ресурс] – Режим доступу: <https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/the-future-of-mobility> – Дата доступу до ресурсу: 18.09.2024.
26. Statista. Peer-to-Peer Car Sharing Market. 2023. [Електронний ресурс] – Режим доступу: <https://www.statista.com/outlook/mmo/mobility-services/car-sharing/peer-to-peer-car-sharing> – Дата доступу до ресурсу: 22.09.2024.
27. Козак О.В., Мельник Т.П. Економічна ефективність Р2Р-моделі оренди автомобілів. Науковий вісник Одеського національного економічного університету. 2023. №2(301). С. 88-96. [Електронний ресурс] – Режим доступу: <https://oneu.edu.ua/scientific-journal/articles/p2p-car-rental-efficiency> – Дата доступу до ресурсу: 10.08.2024.
28. Deloitte. Carsharing Market Analysis. 2023. [Електронний ресурс] – Режим доступу: <https://www2.deloitte.com/global/en/insights/industry/automotive/carsharing-market-analysis.html> – Дата доступу до ресурсу: 15.07.2024.
29. Berg Insight. Carsharing Telematics Market. 2023. [Електронний ресурс] – Режим доступу: <https://www.berginsight.com/carsharing-telematics-market-2023> – Дата доступу до ресурсу: 20.07.2024.
30. Кравченко М.О., Сергієнко О.В. Економічні аспекти розвитку каршерингу в Україні. Вісник Інституту проблем ринку та економіко-екологічних

досліджень НАН України. 2023. №4(24). С. 145-156. [Електронний ресурс] –
Режим доступу: <https://iprei.org.ua/journal/articles/carsharing-economic-aspects>
– Дата доступу до ресурсу: 25.09.2024.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф.
О. Н. Романюк
"25" березня 2025 р.

Технічне завдання**на бакалаврську кваліфікаційну роботу**

**«Розробка програмного засобу для оренди автомобільного транспорту»
за спеціальністю 121 – Інженерія програмного забезпечення**

Керівник бакалаврської кваліфікаційної роботи:



к.т.н., доц. каф. ПЗ Хошаба О.М.

"24" березня 2025 р.

Виконав: студент гр. 2ПІ-216



Парполіта В.О.

"24" березня 2025 р.

Вінниця - 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: Розробка програмного засобу для оренди автомобільного транспорту.

Галузь застосування – розробка програмного засобу в галузі оренди автомобільного транспорту.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №97 від «20» березня 2025 р.

3. Мета та призначення розробки.

Метою роботи Метою роботи є підвищення ефективності процесу оренди автомобільного транспорту шляхом розробки програмного засобу з розширеними функціональними можливостями.

Призначення роботи – розробка програмної додатку, що реалізує що супроводжує процес оренди автомобільного транспорту.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР:

1. Parpolita V., Khoshaba O. Development of software payment for car rental /Матеріали XXV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. «Стан, досягнення і перспективи інформаційних систем і технологій». Одеса, 17 - 18 квітня 2025 р. - Одеса, Видавництво ОНТУ, 2025 р. - с.85-86.
2. Коваленко О.П., Пономаренко В.С. Цифрові технології в автопрокаті: український досвід. [Електронний ресурс] – Режим доступу: <https://nulp.edu.ua/journals/transport-tech/article/view/1256> – Дата доступу до ресурсу: 15.03.2025.

5. Технічні вимоги

Необхідно виконати формалізацію моделі, що характеризує роботу транспортних засобів в системі: до 25 (тип, марка, модель, рік випуску, клас, пробіг, VIN, страхування, фото тощо), кількість вимог до функціоналу програмного забезпечення: до 30 (авторизація, управління профілем, перевірка прав водія, пошук, бронювання, тарифи, аналітика, адмінпанель тощо), загальна кількість записів у базі даних: до 100 000, продуктивність системи під час обробки повинна бути не менш ніж 150 запитів у секунду у піковому режимі, підтримка одночасної

роботи: до 500 користувачів.

6. Конструктивні вимоги.

Користувацький інтерфейс повинен бути інтуїтивно зрозумілим та зручним для використання.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

- а. пояснювальна записка до БКР;
- б. технічне завдання;
- с. лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз проблеми, обґрунтування актуальності розробки системи оренди автомобілів	25.03.2025-03.04.2025
2	Проектування модулів, розробка алгоритмів, структури та моделей динамічного ціноутворення з урахуванням факторів попиту	04.04.2025-14.04.2025
3	Вибір середовища та розробка програмного забезпечення веб-сервісної системи оренди	15.04.2025-05.05.2025
4	Створення тест-кейсів для перевірки стабільності веб-сервісу	06.05.2025-19.05.2025
5	Оформлення матеріалів до захисту БКР	20.05.2025-30.05.2025

10. Порядок контролю та прийняття.

Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка програмного засобу для оренди автомобільного транспорту

Тип роботи: БКР
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра програмного забезпечення, ФІТКІ
(кафедра, факультет, навчальна група)

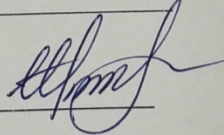
Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 3 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

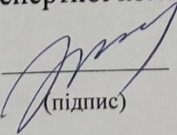
- Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

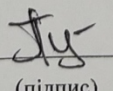
Експертна комісія:

_____	_____
(прізвище, ініціали, посада)	(підпис)
_____	_____
(прізвище, ініціали, посада)	(підпис)

Особа, відповідальна за перевірку  Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник <u></u>	<u>Хошаба О.М.</u>
(підпис)	(прізвище, ініціали, посада)

Здобувач <u></u>	<u>Парполіта В.О.</u>
(підпис)	(прізвище, ініціали)

ДОДАТОК В

Програмний код фронтенду для ієрархічної структури класів динамічного ціноутворення

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Модуль динамічного ціноутворення</title>
  <style>
    /* Базові стилі для покращення вигляду */
    body {
      font-family: Arial, sans-serif;
      line-height: 1.6;
      margin: 0;
      padding: 20px;
    }
    .pricing-container {
      display: flex;
      justify-content: space-around;
      flex-wrap: wrap;
    }
    .pricing-card {
      border: 1px solid #ddd;
      border-radius: 8px;
      padding: 20px;
      margin: 10px;
      width: 250px;
    }
    .price {
      font-size: 24px;
```

```

    font-weight: bold;
    color: #4a4e69;
}
.season-selector {
    margin-bottom: 20px;
    text-align: center;
}
.season-selector select {
    padding: 8px;
    font-size: 16px;
}
</style>
</head>
<body>
    <div class="season-selector">
        <label for="season">Оберіть сезон: </label>
        <select id="season" onchange="updatePrices()">
            <option value="spring">Весна</option>
            <option value="summer">Літо</option>
            <option value="autumn">Осінь</option>
            <option value="winter">Зима</option>
        </select>
    </div>

    <div class="pricing-container" id="pricingContainer">
        <!-- Динамічно згенеровані картки цін будуть тут -->
    </div>

    <script>
        // Базовий клас для транспорту
        class Transport {

```

```

    constructor(type, location, speed) {
        this.type = type;
        this.location = location;
        this.speed = speed;
    }
}

// Клас для сервісу оренди
class RentalService extends Transport {
    constructor(type, location, speed, basePrice) {
        super(type, location, speed);
        this.basePrice = basePrice;
        this.demandAnalyzer = new DemandAnalyzer();
        this.priceCalculator = new PriceCalculator();
    }

    calculatePrice(season, location) {
        const demand = this.demandAnalyzer.analyzeDemand(season, this.type);
        const gpsData = new GPSTDataProvider().provideGPSTData(location);
        return this.priceCalculator.calculateOptimalPrice(this.basePrice, demand,
gpsData, season);
    }
}

// Клас для аналізу попиту
class DemandAnalyzer {
    analyzeDemand(season, transportType) {
        // Визначення аналізу попиту на основі сезону та типу транспорту
        const demandFactors = {
            spring: { 'Автомобіль': 1.2, 'Велосипед': 1.5, 'Самокат': 1.3 },
            summer: { 'Автомобіль': 1.0, 'Велосипед': 1.8, 'Самокат': 1.6 },

```

```

        autumn: { 'Автомобіль': 1.3, 'Велосипед': 1.2, 'Самокат': 1.0 },
        winter: { 'Автомобіль': 1.5, 'Велосипед': 0.7, 'Самокат': 0.5 }
    };

    return demandFactors[season][transportType] || 1.0;
}
}

// Клас для надання GPS даних
class GPSDataProvider {
    provideGPSData(location) {
        // GPS дані для різних локацій
        const locationData = {
            'Центр міста': { latitude: 50.450, longitude: 30.523 },
            'Спальний район': { latitude: 50.470, longitude: 30.498 },
            'Промислова зона': { latitude: 50.430, longitude: 30.550 },
            'Парк': { latitude: 50.445, longitude: 30.510 }
        };

        return locationData[location] || { latitude: 50.450, longitude: 30.523 };
    }
}

// Клас для розрахунку ціни
class PriceCalculator {
    calculateOptimalPrice(basePrice, demand, gpsData, season) {
        // Коефіцієнти для різних локацій
        const locationFactor = this.calculateLocationFactor(gpsData);

        // Сезонний коефіцієнт
        const seasonFactor = this.calculateSeasonFactor(season);
    }
}

```

```

    // Розрахунок ціни з урахуванням усіх факторів
    return basePrice * demand * locationFactor * seasonFactor;
}

calculateLocationFactor(gpsData) {
    // Центр міста має вищий коефіцієнт
    const distanceFromCenter = Math.sqrt(
        Math.pow(gpsData.latitude - 50.450, 2) +
        Math.pow(gpsData.longitude - 30.523, 2)
    );

    // Чим ближче до центру, тим вищий коефіцієнт
    return 1.5 - distanceFromCenter * 10;
}

calculateSeasonFactor(season) {
    const seasonFactors = {
        spring: 1.1,
        summer: 1.2,
        autumn: 1.0,
        winter: 0.9
    };

    return seasonFactors[season] || 1.0;
}

// Функція для створення та відображення карток цін
function createPricingCards() {
    const container = document.getElementById('pricingContainer');

```

```

container.innerHTML = ""; // Очищуємо контейнер

const season = document.getElementById('season').value;
const transportTypes = ['Автомобіль', 'Велосипед', 'Самокат'];
const locations = ['Центр міста', 'Спальний район', 'Промислова зона', 'Парк'];

transportTypes.forEach(type => {
  locations.forEach(location => {
    const basePrices = {
      'Автомобіль': 200,
      'Велосипед': 50,
      'Самокат': 80
    };

    const rental = new RentalService(type, location, 50, basePrices[type]);
    const price = rental.calculatePrice(season, location).toFixed(2);

    const card = document.createElement('div');
    card.className = 'pricing-card';
    card.innerHTML = `
      <h3>Оренда: ${type}</h3>
      <p class="price">${price} грн/год</p>
      <p>Локація: ${location}</p>
      <p>Сезон: ${getSeason(season)}</p>
    `;
    container.appendChild(card);
  });
});
}

// Функція для оновлення цін при зміні сезону

```

```

function updatePrices() {
    createPricingCards();
}

// Функція для отримання назви сезону українською
function getSeason(season) {
    const seasons = {
        spring: 'Весна',
        summer: 'Літо',
        autumn: 'Осінь',
        winter: 'Зима'
    };

    return seasons[season] || '';
}

// Викликаємо функцію створення карток при завантаженні сторінки
window.onload = createPricingCards;
</script>
</body>
</html>

```

Програмний код бекенду для ієрархічної структури класів динамічного ціноутворення

Файл application.properties

```

# Налаштування підключення до бази даних
spring.datasource.url=jdbc:mysql://localhost:3306/dynamic_pricing
spring.datasource.username=root
spring.datasource.password=password
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

```

Налаштування Hibernate

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

Налаштування логування

logging.level.org.springframework=INFO

logging.level.com.dynamicpricing=DEBUG

Файл schema.sql

-- Створення таблиць для бази даних

```
CREATE TABLE IF NOT EXISTS transport (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    type VARCHAR(50) NOT NULL,
    location VARCHAR(100) NOT NULL,
    speed DOUBLE NOT NULL
);
```

```
CREATE TABLE IF NOT EXISTS gps_data (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    transport_id BIGINT NOT NULL,
    latitude DOUBLE NOT NULL,
    longitude DOUBLE NOT NULL,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (transport_id) REFERENCES transport(id)
);
```

```
CREATE TABLE IF NOT EXISTS rental_price (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    transport_id BIGINT NOT NULL,
```



```

base_price DOUBLE NOT NULL,
final_price DOUBLE NOT NULL,
season VARCHAR(20) NOT NULL,
timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (transport_id) REFERENCES transport(id)
);

```

```

CREATE TABLE IF NOT EXISTS historical_demand (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  transport_type VARCHAR(50) NOT NULL,
  location VARCHAR(100) NOT NULL,
  season VARCHAR(20) NOT NULL,
  demand_factor DOUBLE NOT NULL
);

```

-- Початкові дані для таблиці historical_demand

```

INSERT INTO historical_demand (transport_type, location, season, demand_factor)
VALUES
('Автомобіль', 'Центр міста', 'SPRING', 1.2),
('Автомобіль', 'Центр міста', 'SUMMER', 1.0),
('Автомобіль', 'Центр міста', 'AUTUMN', 1.3),
('Автомобіль', 'Центр міста', 'WINTER', 1.5),
('Велосипед', 'Центр міста', 'SPRING', 1.5),
('Велосипед', 'Центр міста', 'SUMMER', 1.8),
('Велосипед', 'Центр міста', 'AUTUMN', 1.2),
('Велосипед', 'Центр міста', 'WINTER', 0.7),
('Самокат', 'Центр міста', 'SPRING', 1.3),
('Самокат', 'Центр міста', 'SUMMER', 1.6),
('Самокат', 'Центр міста', 'AUTUMN', 1.0),
('Самокат', 'Центр міста', 'WINTER', 0.5);

```

Класи моделей Transport.java

```
package com.dynamicpricing.model;
```

```
import javax.persistence.*;
```

```
import lombok.Data;
```

```
import lombok.NoArgsConstructor;
```

```
import lombok.AllArgsConstructor;
```

```
@Data
```

```
@NoArgsConstructor
```

```
@AllArgsConstructor
```

```
@Entity
```

```
@Table(name = "transport")
```

```
public class Transport {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;
```

```
    @Column(nullable = false)
```

```
    private String type;
```

```
    @Column(nullable = false)
```

```
    private String location;
```

```
    @Column(nullable = false)
```

```
    private double speed;
```

```
// Конструктор без id для створення нових об'єктів
```

```
public Transport(String type, String location, double speed) {
```

```
    this.type = type;
```

```
    this.location = location;
```

```

        this.speed = speed;
    }
}

```

GpsData.java

```
package com.dynamicpricing.model;
```

```

import javax.persistence.*;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;
import java.time.LocalDateTime;

```

```
@Data
```

```
@NoArgsConstructor
```

```
@AllArgsConstructor
```

```
@Entity
```

```
@Table(name = "gps_data")
```

```
public class GpsData {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;
```

```
    @ManyToOne
```

```
    @JoinColumn(name = "transport_id", nullable = false)
```

```
    private Transport transport;
```

```
    @Column(nullable = false)
```

```
    private double latitude;
```

```
    @Column(nullable = false)
```

```
private double longitude;
```

```
@Column(nullable = false)
```

```
private LocalDateTime timestamp;
```

```
// Конструктор без id для створення нових об'єктів
```

```
public GpsData(Transport transport, double latitude, double longitude) {
```

```
    this.transport = transport;
```

```
    this.latitude = latitude;
```

```
    this.longitude = longitude;
```

```
    this.timestamp = LocalDateTime.now();
```

```
}
```

```
}
```

RentalPrice.java

```
package com.dynamicpricing.model;
```

```
import javax.persistence.*;
```

```
import lombok.Data;
```

```
import lombok.NoArgsConstructor;
```

```
import lombok.AllArgsConstructor;
```

```
import java.time.LocalDateTime;
```

```
@Data
```

```
@NoArgsConstructor
```

```
@AllArgsConstructor
```

```
@Entity
```

```
@Table(name = "rental_price")
```

```
public class RentalPrice {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
private Long id;
```

```
@ManyToOne
```

```
@JoinColumn(name = "transport_id", nullable = false)
```

```
private Transport transport;
```

```
@Column(nullable = false)
```

```
private double basePrice;
```

```
@Column(nullable = false)
```

```
private double finalPrice;
```

```
@Enumerated(EnumType.STRING)
```

```
@Column(nullable = false)
```

```
private Season season;
```

```
@Column(nullable = false)
```

```
private LocalDateTime timestamp;
```

```
// Конструктор без id для створення нових об'єктів
```

```
public RentalPrice(Transport transport, double basePrice, double finalPrice, Season
season) {
```

```
    this.transport = transport;
```

```
    this.basePrice = basePrice;
```

```
    this.finalPrice = finalPrice;
```

```
    this.season = season;
```

```
    this.timestamp = LocalDateTime.now();
```

```
}
```

```
// Перелік сезонів
```

```
public enum Season {
```

```

        SPRING, SUMMER, AUTUMN, WINTER
    }
}

```

HistoricalDemand.java

```
package com.dynamicpricing.model;
```

```
import javax.persistence.*;
```

```
import lombok.Data;
```

```
import lombok.NoArgsConstructor;
```

```
import lombok.AllArgsConstructor;
```

```
@Data
```

```
@NoArgsConstructor
```

```
@AllArgsConstructor
```

```
@Entity
```

```
@Table(name = "historical_demand")
```

```
public class HistoricalDemand {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;
```

```
    @Column(nullable = false)
```

```
    private String transportType;
```

```
    @Column(nullable = false)
```

```
    private String location;
```

```
    @Enumerated(EnumType.STRING)
```

```
    @Column(nullable = false)
```

```
    private RentalPrice.Season season;
```

```

    @Column(nullable = false)
    private double demandFactor;
}

```

Репозиторії TransportRepository.java

```
package com.dynamicpricing.repository;
```

```

import com.dynamicpricing.model.Transport;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

```

```
import java.util.List;
```

```
@Repository
```

```

public interface TransportRepository extends JpaRepository<Transport, Long> {
    List<Transport> findByType(String type);
    List<Transport> findByLocation(String location);
}

```

GpsDataRepository.java

```
package com.dynamicpricing.repository;
```

```

import com.dynamicpricing.model.GpsData;
import com.dynamicpricing.model.Transport;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

```

```
import java.util.Optional;
```

```
@Repository
```

```
public interface GpsDataRepository extends JpaRepository<GpsData, Long> {
    Optional<GpsData> findTopByTransportOrderByTimestampDesc(Transport
transport);
}
```

RentalPriceRepository.java

```
package com.dynamicpricing.repository;
```

```
import com.dynamicpricing.model.RentalPrice;
import com.dynamicpricing.model.Transport;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
```

```
import java.util.Optional;
```

@Repository

```
public interface RentalPriceRepository extends JpaRepository<RentalPrice, Long> {
    Optional<RentalPrice> findTopByTransportAndSeasonOrderByTimestampDesc(
        Transport transport, RentalPrice.Season season);
}
```

HistoricalDemandRepository.java

```
package com.dynamicpricing.repository;
```

```
import com.dynamicpricing.model.HistoricalDemand;
import com.dynamicpricing.model.RentalPrice;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
```

```
import java.util.Optional;
```


@Repository

```
public interface HistoricalDemandRepository extends JpaRepository<HistoricalDemand,
Long> {
    Optional<HistoricalDemand> findByTransportTypeAndLocationAndSeason(
        String transportType, String location, RentalPrice.Season season);
}
```

Сервіси DemandAnalyzerService.java

```
package com.dynamicpricing.service;
```

```
import com.dynamicpricing.model.HistoricalDemand;
import com.dynamicpricing.model.RentalPrice;
import com.dynamicpricing.repository.HistoricalDemandRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

@Service

```
public class DemandAnalyzerService {
```

```
    private final HistoricalDemandRepository historicalDemandRepository;
```

@Autowired

```
    public DemandAnalyzerService(HistoricalDemandRepository
historicalDemandRepository) {
```

```
        this.historicalDemandRepository = historicalDemandRepository;
    }
```

```
/**
```

```
 * Аналізує попит на основі типу транспорту, локації та сезону
```

```
 * @param transportType тип транспорту
```

```
 * @param location локація
```

```

* @param season сезон
* @return коефіцієнт попиту
*/
public double analyzeDemand(String transportType, String location,
RentalPrice.Season season) {
    // Отримуємо історичні дані про попит з бази даних
    return historicalDemandRepository
        .findByTransportTypeAndLocationAndSeason(transportType, location, season)
        .map(HistoricalDemand::getDemandFactor)
        .orElse(1.0); // Якщо даних немає, повертаємо нейтральний коефіцієнт
}
}

```

GPSDataProviderService.java

```
package com.dynamicpricing.service;
```

```

import com.dynamicpricing.model.GpsData;
import com.dynamicpricing.model.Transport;
import com.dynamicpricing.repository.GpsDataRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

```

```
@Service
```

```
public class GPSDataProviderService {
```

```
    private final GpsDataRepository gpsDataRepository;
```

```
    @Autowired
```

```

    public GPSDataProviderService(GpsDataRepository gpsDataRepository) {
        this.gpsDataRepository = gpsDataRepository;
    }
}

```

```

/**
 * Отримує останні GPS дані для транспорту
 * @param transport об'єкт транспорту
 * @return об'єкт GPS даних
 */
public GpsData getLatestGpsData(Transport transport) {
    return gpsDataRepository
        .findTopByTransportOrderByTimestampDesc(transport)
        .orElseGet(() -> {
            // Якщо немає даних, створюємо нові на основі локації транспорту
            double[] coordinates = getCoordinatesForLocation(transport.getLocation());
            GpsData gpsData = new GpsData(transport, coordinates[0], coordinates[1]);
            return gpsDataRepository.save(gpsData);
        });
}

/**
 * Отримує координати для заданої локації
 * @param location назва локації
 * @return масив координат [latitude, longitude]
 */
private double[] getCoordinatesForLocation(String location) {
    // Мапа координат для різних локацій
    switch (location) {
        case "Центр міста":
            return new double[] {50.450, 30.523};
        case "Спальний район":
            return new double[] {50.470, 30.498};
        case "Промислова зона":
            return new double[] {50.430, 30.550};
    }
}

```

```

        case "Парк":
            return new double[]{50.445, 30.510};
        default:
            return new double[]{50.450, 30.523}; // За замовчуванням - центр міста
    }
}
}

```

PriceCalculatorService.java

```
package com.dynamicpricing.service;
```

```

import com.dynamicpricing.model.GpsData;
import com.dynamicpricing.model.RentalPrice;
import org.springframework.stereotype.Service;

```

```
@Service
```

```
public class PriceCalculatorService {
```

```
    /**
```

```
    * Розраховує оптимальну ціну на основі базової ціни, попиту, GPS даних та сезону
```

```
    * @param basePrice базова ціна
```

```
    * @param demandFactor коефіцієнт попиту
```

```
    * @param gpsData GPS дані
```

```
    * @param season сезон
```

```
    * @return оптимальна ціна
```

```
    */
```

```

    public double calculateOptimalPrice(double basePrice, double demandFactor,
    GpsData gpsData, RentalPrice.Season season) {

```

```
        // Розрахунок коефіцієнта локації
```

```
        double locationFactor = calculateLocationFactor(gpsData);
```

```

// Розрахунок сезонного коефіцієнта
double seasonFactor = calculateSeasonFactor(season);

// Розрахунок ціни з урахуванням усіх факторів
return basePrice * demandFactor * locationFactor * seasonFactor;
}

/**
 * Розраховує коефіцієнт локації на основі GPS даних
 * @param gpsData GPS дані
 * @return коефіцієнт локації
 */
private double calculateLocationFactor(GpsData gpsData) {
    // Центр міста має координати (50.450, 30.523)
    // Чим далі від центру, тим нижчий коефіцієнт
    double distanceFromCenter = Math.sqrt(
        Math.pow(gpsData.getLatitude() - 50.450, 2) +
        Math.pow(gpsData.getLongitude() - 30.523, 2)
    );

    // Обмежуємо коефіцієнт в межах від 0.8 до 1.5
    return Math.max(0.8, Math.min(1.5, 1.5 - distanceFromCenter * 10));
}

/**
 * Розраховує сезонний коефіцієнт
 * @param season сезон
 * @return сезонний коефіцієнт
 */
private double calculateSeasonFactor(RentalPrice.Season season) {

```

```

switch (season) {
    case SPRING:
        return 1.1;
    case SUMMER:
        return 1.2;
    case AUTUMN:
        return 1.0;
    case WINTER:
        return 0.9;
    default:
        return 1.0;
}
}
}

```

RentalService.java

```
package com.dynamicpricing.service;
```

```

import com.dynamicpricing.model.GpsData;
import com.dynamicpricing.model.RentalPrice;
import com.dynamicpricing.model.Transport;
import com.dynamicpricing.repository.RentalPriceRepository;
import com.dynamicpricing.repository.TransportRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

```

```

import java.util.List;
import java.util.stream.Collectors;

```

```
@Service
```

```
public class RentalService {
```

```
private final TransportRepository transportRepository;  
private final RentalPriceRepository rentalPriceRepository;  
private final DemandAnalyzerService demandAnalyzerService;  
private final GPSDataProviderService gpsDataProviderService;  
private final PriceCalculatorService priceCalculatorService;
```

```
@Autowired
```

```
public RentalService(  
    TransportRepository transportRepository,  
    RentalPriceRepository rentalPriceRepository,  
    DemandAnalyzerService demandAnalyzerService,  
    GPSDataProviderService gpsDataProviderService,  
    PriceCalculatorService priceCalculatorService  
) {  
    this.transportRepository
```

ДОДАТОК Г
Графічна частина

ГРАФІЧНА ЧАСТИНА

Розробка програмного засобу для оренди автомобільного транспорту

Розробка програмного засобу для оренди автомобільного транспорту

Виконав:

студент групи 2ПІ-216

Парполіта В.О.

Керівник:

к.т.н., доц. каф. ПЗ

Хошаба О.М.

Рисунок Д.2 – Слайд презентації 2

Метою роботи є підвищення ефективності виконання процесу оренди автомобільного транспорту.

Об’єктом дослідження є процес автоматизації оренди автомобільного транспорту за допомогою програмного забезпечення.

Предметом дослідження є методи та програмні засоби для реалізації веборієнтованої інформаційної системи управління орендою автомобілів.

Рисунок Д.2 – Слайд презентації 2

Відповідно до поставленої мети виконані наступні задачі:

- провести аналіз предметної галузі та порівняльний аналіз існуючих програмних рішень;
- розробити модель динамічного ціноутворення та метод оптимізації розподілу автомобілів;
- виконати проектування архітектуру веб-сервісної системи оренди;
- реалізувати програмні модулі для динамічного ціноутворення та управління орендою;
- провести тестування функціональності та створити тест-кейси для перевірки стабільності веб-сервісу.

Рисунок Д.3 – Слайд презентації 3

Актуальність теми дослідження:

Необхідність цифровізації послуг оренди автотранспорту із застосуванням інноваційних інформаційних технологій для підвищення ефективності, точності та зручності управління сервісом.

Існуючі системи мають низку обмежень: складний інтерфейс, обмежений набір функцій, низька адаптивність до бізнес-моделей, недостатній захист даних.

В теперішній час галузь оренди автомобільного транспорту стикається з фрагментованістю бізнес-процесів та недостатньою автоматизацією (обробка замовлень, перевірка документів, аналітика автопарку).

Також, існують проблеми програмно-технічного характеру, до яких належать масштабування та безпекові ризики, пов'язані з обробкою персональних даних та контролем доступу до автомобіля.

Рисунок Д.4 – Слайд презентації 4

Аналіз предметної галузі: Традиційні моделі оренди автомобілів

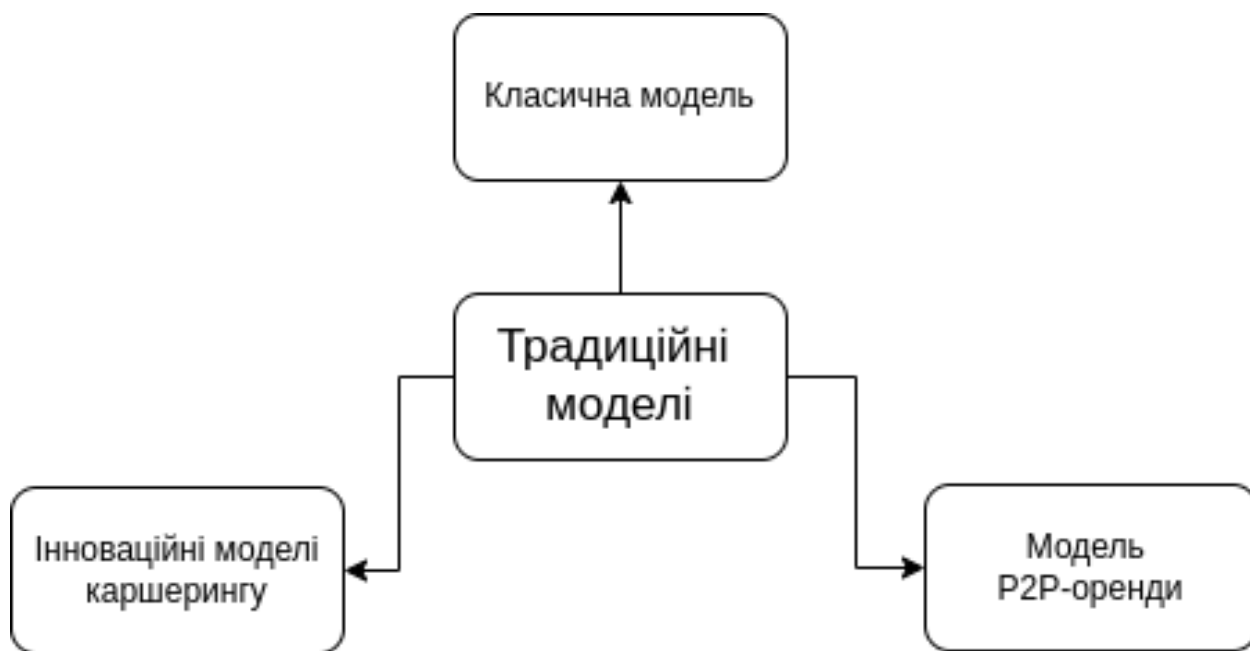


Рисунок Д.5 – Слайд презентації 5

Аналіз предметної галузі: Інноваційні моделі та ціноутворення

Метод ціноутворення	Опис	Ефективність
Статичне ціноутворення	Фіксовані тарифи за період	Традиційний підхід
Динамічне ціноутворення	Ціни змінюються залежно від попиту	Підвищення ефективності на 15-20% [10]
Пакетне ціноутворення	Комплексна пропозиція з додатковими послугами	Збільшення середнього чеку
Сезонне ціноутворення	Врахування сезонних коливань попиту	Оптимізація завантаження автопарку
Алгоритмічне ціноутворення	Використання ML-алгоритмів	Збільшення прибутковості на 8-12% [13]

Рисунок Д.6 – Слайд презентації 6

Порівняльний аналіз програмних засобів

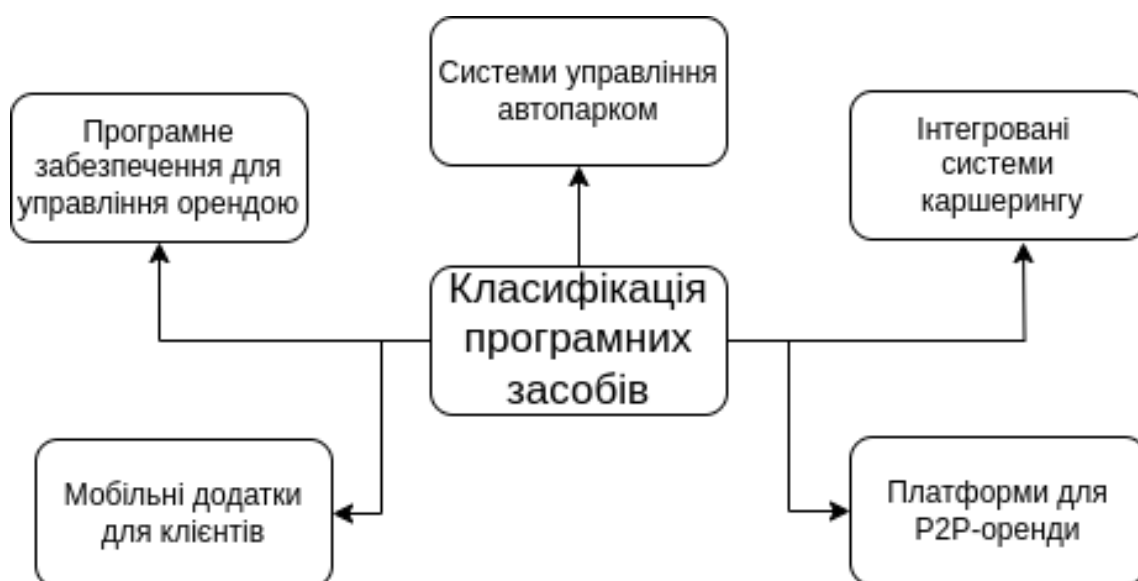


Рисунок Д.7 – Слайд презентації 7

Проектування: Процес динамічного ціноутворення

UML діаграма використання процесу динамічного ціноутворення

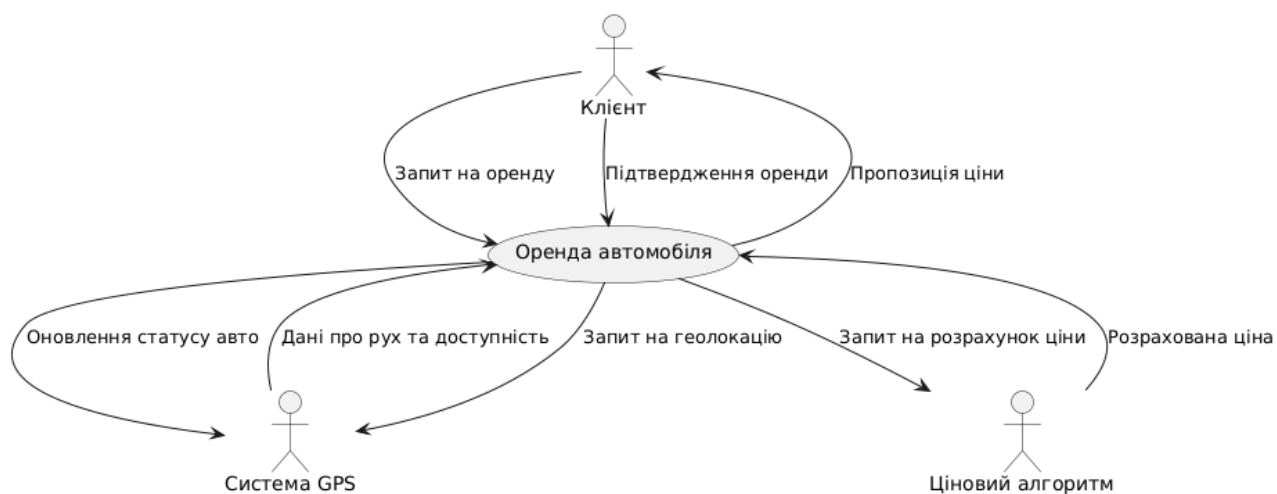


Рисунок Д.8 – Слайд презентації 8

Проектування: Оптимізація процесу оренди автомобільного транспорту

UML діаграма послідовності процесу оптимізації оренди автомобільного транспорту

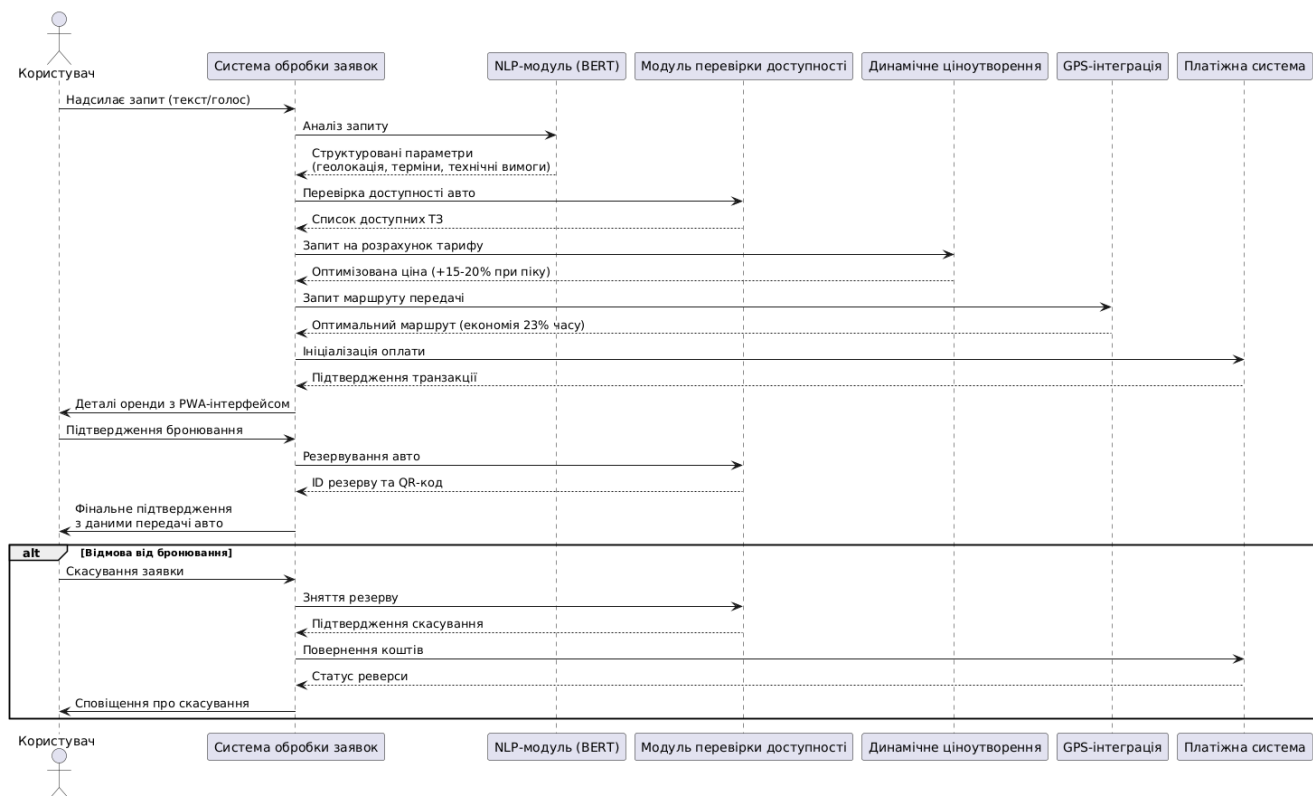


Рисунок Д.9 – Слайд презентації 9
Проектування: Архітектура веб-сервісної системи

UML діаграма послідовності реалізації основних етапів проектування архітектури веб-сервісної системи оренди автомобільного транспорту

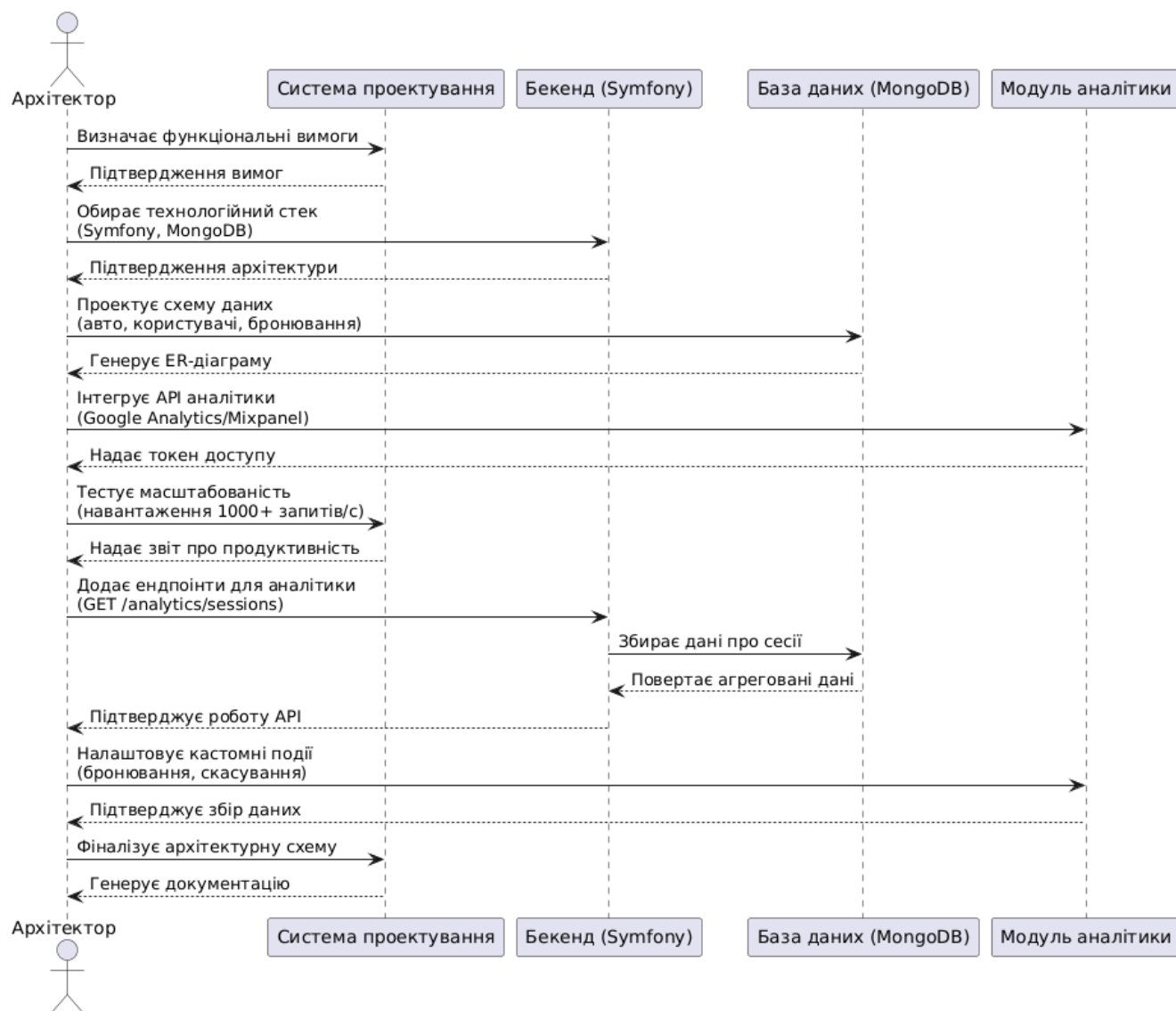


Рисунок Д.10 – Слайд презентації 10
Розробка модулів: Динамічне ціноутворення – Ієрархія класів

UML діаграма класів динамічного ціноутворення для оренди транспорту

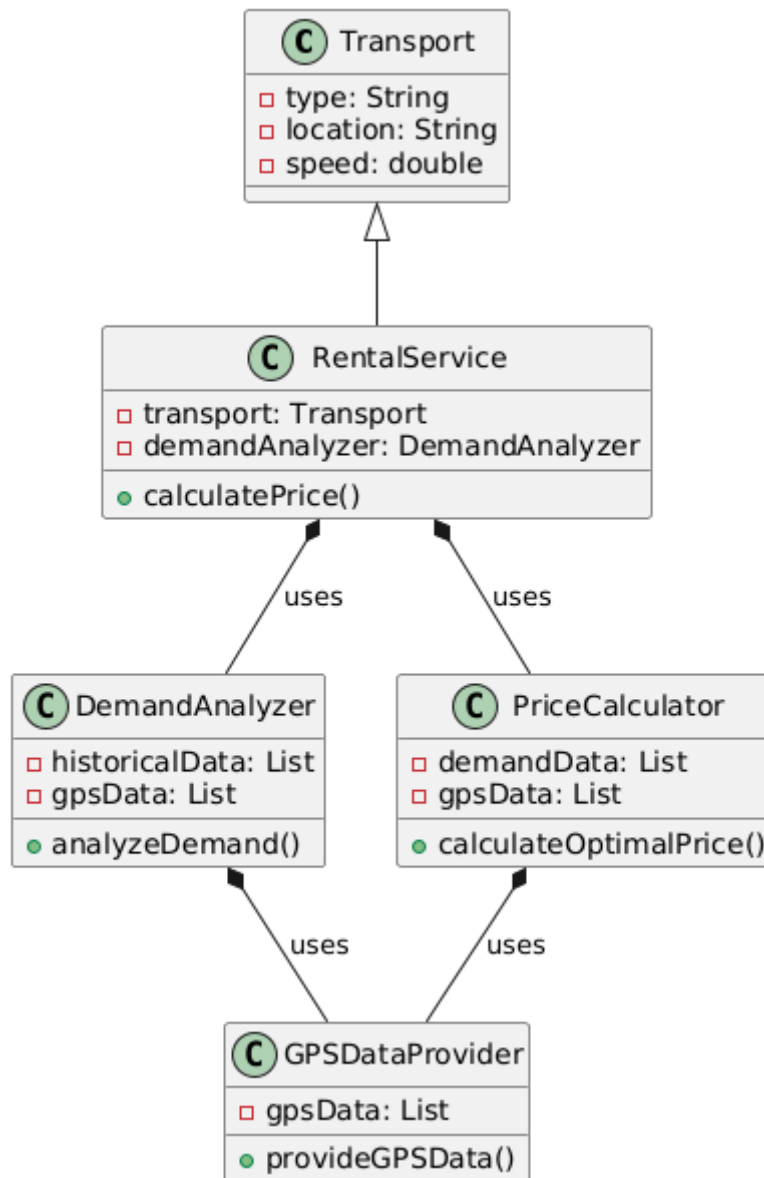


Рисунок Д.11 – Слайд презентації 11
 Розробка модулів: Оптимізація – Алгоритм роботи системи
 Алгоритм роботи веб-сервісної системи оренди автомобільного транспорту на
 основі UML діаграми активностей

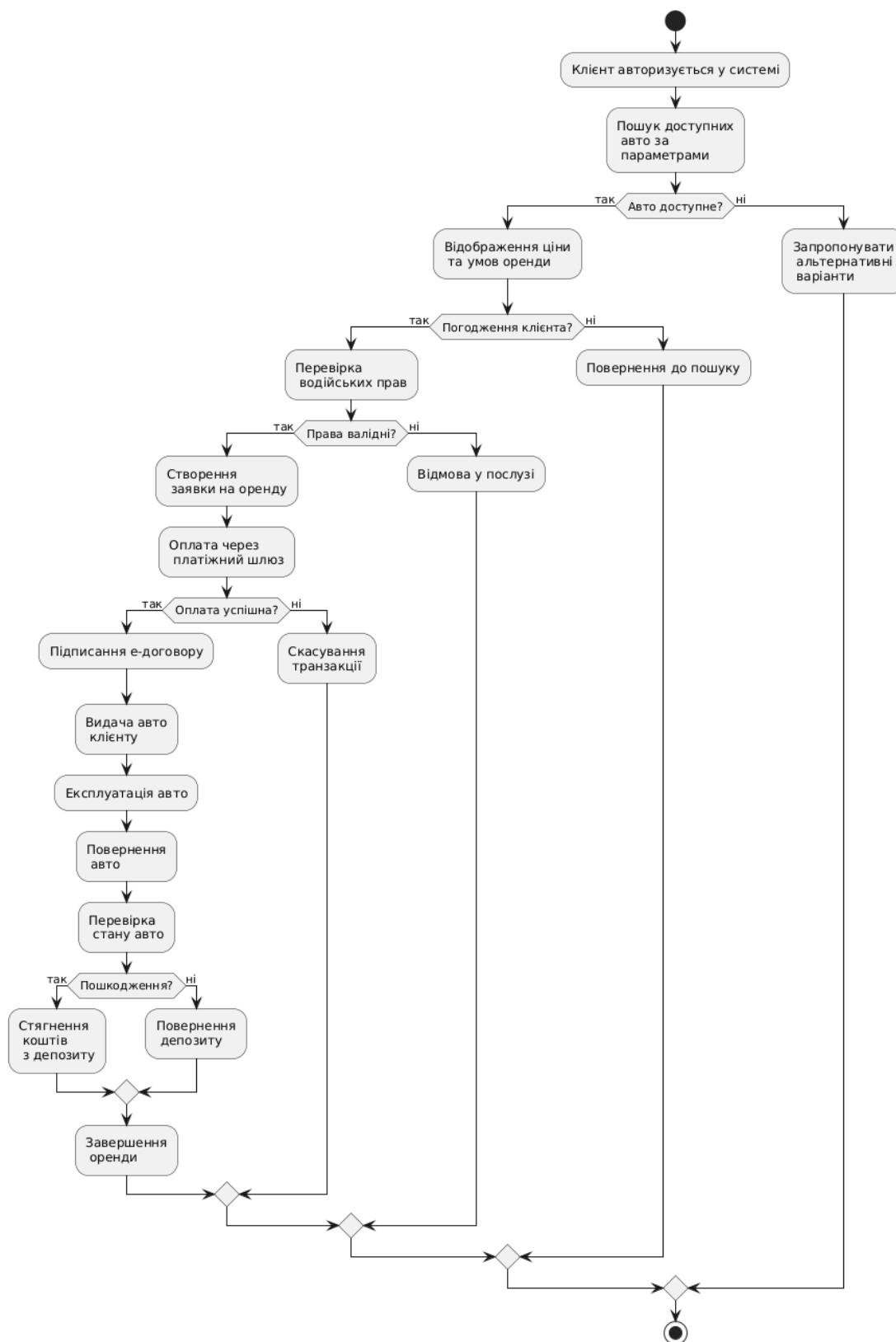


Рисунок Д.12 – Слайд презентації 12

Тестування: Розробка тест-кейсів процесу оптимізації оренди автомобілів

Сценарій тестування	Мета	Приклади тестів
Пошук автомобілів	Перевірити швидкість і точність алгоритмів пошуку.	- Пошук за базовими параметрами (місце, дати оренди). - Пошук за фільтрами (клас авто, тип палива). - Тестування часу відгуку при великій кількості запитів. - Перевірка релевантності результатів пошуку.
Процес бронювання	Скоротити час і кількість кроків для завершення бронювання.	- Оформлення бронювання з мінімальними даними. - Автозаповнення полів для зареєстрованих користувачів. - Тестування динамічного ціноутворення. - Перевірка обробки платежів через платіжні шлюзи.
Продуктивність	Оцінити швидкодію системи під час пікових навантажень.	- Навантажувальне тестування з одночасним доступом багатьох користувачів. - Стрес-тестування для перевірки стійкості системи. - Тестування часу відгуку API під час масових запитів.
Інтеграція	Перевірити коректну роботу інтеграцій із зовнішніми сервісами.	- Інтеграція з платіжними системами (PayPal, Stripe). - Тестування API страхових компаній для розрахунку страховки. - Синхронізація з GPS-сервісами для відображення місцезнаходження авто в реальному часі.

Рисунок Д.13 – Слайд презентації 12

Тестування: Приклад виконання одного з тест-кейсів для визначення коректності введених даних

Модуль динамічного ціноутворення

Оберіть сезон:

Весна



Модуль динамічного ціноутворення

Оберіть сезон:

Весна



Оренда:
Автомобіль

312.00 грн/год

Локація: Ценча міста
Сезон: Весна

Оренда:
Велосипед

99.00 грн/год

Локація: Спальний район: Весна

Оренда:
Самокат

123.50 грн/год

Локація: Промислова зона: Весна

Оренда:
Автомобіль

231.80 грн/год

Локація: Парк
Сезон: Весна

Рисунок Д.14 – Слайд презентації 12
Новизна отриманих результатів:

1. Запропоновано метод оренди автомобільного транспорту, особливість якого полягає у веб-сервісній архітектурі з інтегрованим функціоналом онлайн-бронювання, GPS-трекінгу транспортних засобів та платіжних систем, що надає можливість оцінити його ефективність та існуючий потенціал для збільшення завантаженості автопарку на більш ніж 18%.

2. Запропоновано метод оптимізації процесу оренди автомобільного транспорту, що дозволяє скоротити час обробки замовлень за рахунок автоматизації ключових етапів та аналізу даних про поведінку користувачів.

3. Подальшого розвитку отримала архітектура веб-сервісної системи оренди автомобільного транспорту, у якій, на відміну від існуючих аналогів, інтегровано модуль аналітики користувацьких дій та рекомендаційний сервіс, що дозволило підвищити ефективність взаємодії користувачів із системою.

Рисунок Д.15 – Слайд презентації 13

Висновки:

У бакалаврській кваліфікаційній роботі:

- проведено аналіз предметної галузі та визначити особливості ринку оренди автомобілів;
- виконано порівняльний аналіз існуючих програмних рішень для оренди транспорту;
- розроблено модель динамічного ціноутворення з урахуванням факторів попиту та сезонності;
- запропоновано метод оптимізації розподілу автомобілів для мінімізації простоїв;
- спроектовано архітектуру веб-сервісної системи оренди з інтеграцією сучасних технологій;
- реалізовано програмні модулі для динамічного ціноутворення та управління орендою;
- проведено тестування функціональності модулів динамічного ціноутворення та оптимізації;
- створені тест-кейси для перевірки стабільності веб-сервісу;
- проаналізовані результати тестування та визначити шляхи вдосконалення системи.