Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавра
(ступінь вищої освіти (освітній ступінь))

на тему **РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ДЛЯ ОРГАНІЗАЦІЇ МІСЬКИХ ПЕРЕВЕЗЕНЬ**

Виконав: студент 4 курсу, групи КНТ-522
спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Комп'ютерна інженерія

ПІКАЛОВА Є.Р.
(ПРІЗВИЩЕ та ініціали)
Керівник ГОЛУБ Т.В.
(ПРІЗВИЩЕ та ініціали)
Рецензент ХАРИТОНОВ О.Б.
(ПРІЗВИЩЕ та ініціали)

2026

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
Кафедра Комп'ютерні системи та мережі
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри КУДЕРМЕТОВ Р.К.
_____ _____
"____" _____ 2026 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ПІКАЛОВІЙ Єлизаветі Романівні
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка комп'ютерної системи для організації міських перевезень

керівник проєкту (роботи) к.т.н., доцент, ГОЛУБ Тетяна Василівна
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом вищого навчального закладу від «14» квітня 2026 року № 160
2. Строк подання студентом проєкту (роботи) 10 червня 2026 року
3. Вихідні дані до проєкту (роботи) Існуючі підходи та технології організації міських пасажирських перевезень. Технології GPS/GLONASS-моніторингу транспортних засобів. Протокол MQTT для передачі телеметричних даних від бортових пристроїв. Архітектурні шаблони побудови клієнт-серверних застосунків на Node.js 20 LTS з Express.js. Кросплатформна розробка мобільних додатків на React Native 0.74. Реляційна СУБД PostgreSQL 16 з розширенням PostGIS для просторових запитів. Бібліотеки картографії Mapbox GL JS. WebSocket (Socket.io) для двонаправленої комунікації у реальному часі. Сервіс push-сповіщень Firebase Cloud Messaging.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області та існуючих рішень для організації міських перевезень;
2. Проєктування комп'ютерної системи міських перевезень;
3. Розробка комп'ютерної системи міських перевезень;
4. Тестування комп'ютерної системи міських перевезень.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ГОЛУБ Т.В., к.т.н.		
нормоконтроль	ГОЛУБ Т.В., к.т.н..		

7. Дата видачі завдання 01.05.2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз ринку міського пасажирського транспорту України та порівняльний аналіз існуючих мобільних додатків (Moovit, Google Maps Transit, Citymapper)	05.03.2026 р.	
2	Розробка специфікації вимог до системи для трьох ролей користувачів (пасажир, водій, адміністратор)	20.03.2026 р.	
3	Проєктування архітектури системи; обґрунтування вибору клієнт-серверної моделі та протоколу MQTT для GPS-модулів	30.03.2026 р.	
4	Розробка ER-діаграми та реалізація бази даних на PostgreSQL 16 з розширенням PostGIS	05.04.2026 р.	
5	Розробка серверної частини на Node.js + Express (модулі автентифікації, маршрутів, GPS-процесингу, сповіщень)	20.04.2026 р.	
6	Розробка мобільного додатку на React Native (інтерфейси пасажира та водія) і веб-панелі адміністратора на React.js	01.05.2026 р.	
7	Інтеграція бортових GPS-модулів з MQTT-брокером та сервісу Firebase Cloud Messaging для push-сповіщень	10.05.2026 р.	
8	Функціональне, навантажувальне тестування системи, тестування точності GPS-позиціонування	15.05.2026 р.	
9	Оформлення пояснювальної записки та графічного матеріалу	20.05.2026 р.	

Студент (ка) _____ **Єлизавета ПІКАЛОВА**
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ **Тетяна ГОЛУБ**
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 84 с., 32 рис., 20 табл., 20 джерел, 1 додаток.

ГРОМАДСЬКИЙ ТРАНСПОРТ, GPS-МОНІТОРИНГ, GLONASS, IoT, MQTT, БОРТОВИЙ МОДУЛЬ, REACT NATIVE, NODE.JS, EXPRESS, POSTGRESQL, POSTGIS, WEBSOCKET, MAPBOX

Метою дипломного проєкту є розробка комп'ютерної системи для організації міських перевезень, яка поєднує мобільний додаток для пасажирів і водіїв та апаратну частину для відстеження транспортних засобів у реальному часі.

Предметом дослідження є процеси організації, моніторингу та інформаційного забезпечення міських пасажирських перевезень. Об'єктом дослідження є комп'ютерна система, що поєднує апаратні GPS-модулі на транспортних засобах, серверну інфраструктуру обробки геолокаційних даних та мобільний додаток для кінцевих користувачів.

У роботі проведено порівняльний аналіз провідних мобільних додатків для громадського транспорту (Moovit, Google Maps Transit, Citymapper), розроблено специфікацію вимог для трьох ролей користувачів (пасажир, водій, адміністратор), спроєктовано трирівневу клієнт-серверну архітектуру з апаратним, серверним та клієнтським рівнями, розроблено реляційну схему бази даних з підтримкою просторових запитів та реалізовано серверну і клієнтську частини системи.

Для розробки системи обрано сучасний технологічний стек: серверна частина – Node.js 20 LTS з Express.js та Socket.io; мобільний додаток – React Native 0.74; веб-панель адміністратора – React.js з Redux Toolkit; СУБД – PostgreSQL 16 з розширенням PostGIS; обмін даними з GPS-модулями – протокол MQTT (брокер Eclipse Mosquitto); кешування – Redis 7; рендеринг карт – Mapbox GL JS; push-сповіщення – Firebase Cloud Messaging; контейнеризація – Docker.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого зростання міст і підвищення вимог до якості життя населення організація ефективного міського пасажирського транспорту стає одним із пріоритетних завдань міської інфраструктури. Транспортна система міста безпосередньо впливає на економічний розвиток, екологічний стан та соціальне благополуччя його мешканців. Відсутність актуальної інформації про рух транспортних засобів, неможливість планування маршруту в реальному часі та низька прозорість роботи перевізників призводять до зниження привабливості громадського транспорту і збільшення кількості особистих автомобілів на дорогах [1].

Розвиток технологій Інтернету речей (IoT), глобальних систем позиціонування (GPS), мобільних обчислень та хмарних платформ відкриває широкі можливості для цифрової трансформації галузі міських перевезень. Інтеграція бортового GPS-обладнання транспортних засобів з мобільними додатками для пасажирів дозволяє відстежувати рух транспорту в режимі реального часу, будувати оптимальні маршрути та скорочувати час очікування на зупинках [2]. Провідні світові міста вже успішно впровадили подібні рішення, що підтверджує їх ефективність та доцільність.

В Україні цифровізація міського транспорту перебуває на початковому етапі. Більшість міст, особливо середнього розміру, досі не мають повноцінних систем моніторингу громадського транспорту з відкритим доступом для пасажирів. Існуючі рішення часто є розрізненими, не охоплюють усі види транспорту та не забезпечують достатнього рівня точності й надійності [3]. У цьому контексті розробка комп’ютерної системи для організації міських перевезень є актуальною та практично значущою задачею.

Метою даного дипломного проєкту є розробка комп’ютерної системи для організації міських перевезень, яка включає мобільний додаток для пасажирів і водіїв та апаратну частину для відстеження транспортних засобів у реальному часі,

що в сукупності забезпечує ефективне управління маршрутами, інформування пасажирів та оптимізацію роботи міського транспорту.

Предметом дослідження є процеси організації, моніторингу та інформаційного забезпечення міських пасажирських перевезень. Об’єктом дослідження є комп’ютерна система, що поєднує апаратні GPS-модулі на транспортних засобах, серверну інфраструктуру обробки даних та мобільний додаток для кінцевих користувачів.

Для досягнення поставленої мети необхідно виконати наступні задачі:

– аналіз сучасного стану ринку міських перевезень та існуючих цифрових рішень для їх організації;

– дослідження функціональних можливостей аналогів і визначення їх переваг та недоліків;

– проєктування архітектури комп’ютерної системи, що включає апаратну та програмну складові;

– розробка структури бази даних для зберігання інформації про маршрути, транспортні засоби та користувачів;

– вибір технологічного стеку та реалізація мобільного додатку і серверної частини;

– тестування системи та оцінка її продуктивності.

Практична цінність розробленої системи полягає у підвищенні якості обслуговування пасажирів завдяки наданню точної інформації про рух транспорту, скороченні часу очікування та забезпеченні зручного планування поїздок. Для транспортних підприємств система надає інструменти контролю за дотриманням розкладу і оптимізації використання рухомого складу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ ДЛЯ ОРГАНІЗАЦІЇ МІСЬКИХ ПЕРЕВЕЗЕНЬ

1.1 Огляд сучасного стану ринку міського пасажирського транспорту в Україні

Міський пасажирський транспорт є однією з ключових складових інфраструктури будь-якого сучасного міста. В Україні система громадського транспорту охоплює автобуси, тролейбуси, трамваї та метро, забезпечуючи щоденне переміщення мільйонів людей. Незважаючи на значний потенціал, ця сфера характеризується суттєвими проблемами, пов’язаними із застарілою інфраструктурою, недостатнім рівнем цифровізації та низькою якістю обслуговування пасажирів [1].

Згідно з даними Державної служби статистики України, у великих містах країни частка поїздок громадським транспортом становить від 60 до 70% від загального обсягу міських переміщень [2]. Проте зростання рівня автомобілізації та незадоволеність якістю транспортних послуг призводять до поступового скорочення кількості пасажирів. Серед основних причин відмови від громадського транспорту респонденти соціологічних досліджень називають нерегулярність руху, переповненість салонів та відсутність зручних інструментів для планування маршруту [3]. При цьому у малих і середніх містах ситуація є ще більш критичною: там маршрутна мережа часто визначається виключно комерційними міркуваннями приватних перевізників, а не реальними потребами населення.

Особливо гострою є проблема інформаційної невизначеності: пасажири змушені орієнтуватися на розклад, який рідко відповідає реальному руху транспортних засобів. Відсутність систем онлайн-моніторингу в більшості українських міст середнього розміру означає, що людина не може дізнатися, коли прибуде наступний автобус, не виходячи із дому. Це не лише знижує якість пасажирського досвіду, але й підштовхує людей до використання особистих автомобілів, погіршуючи дорожню ситуацію та екологічний стан міста [1].

Дослідження, проведені в рамках проєктів Smart City в ряді українських міст, підтверджують, що впровадження систем онлайн-відстеження транспорту знижує суб'єктивний час очікування пасажира на 20–35%, навіть якщо реальний інтервал руху залишається незмінним [4].

Структура ринку міських перевезень в Україні є неоднорідною. У великих містах – Києві, Харкові, Одесі, Дніпрі, Львові – функціонують комунальні підприємства, що обслуговують маршрути електричного транспорту (метро, трамвай, тролейбус), поряд із великою кількістю приватних операторів автобусних маршрутів. У містах середнього розміру з населенням від 50 до 300 тисяч осіб переважають приватні маршрутні перевізники, рівень технічного оснащення яких є значно нижчим. Більшість із них не мають жодної системи GPS-моніторингу, не публікують даних у форматі GTFS та фактично є недоступними для цифрових транспортних сервісів [2].

Пандемія COVID-19 суттєво прискорила процес цифрової трансформації у галузі міських перевезень. Необхідність мінімізації фізичних контактів стимулювала впровадження безконтактної оплати проїзду, розвиток мобільних додатків для планування поїздок та дистанційного контролю наповненості транспортних засобів [1]. Ці зміни, попри свій вимушений характер, заклали інституційну основу для подальшої системної цифровізації галузі. Водночас нерівномірність цього процесу між різними регіонами та категоріями перевізників залишається серйозною проблемою.

У контексті євроінтеграції України особливої актуальності набуває приведення систем міського транспорту до стандартів Європейського Союзу. Директиви ЄС у сфері міської мобільності вимагають від міст із населенням понад 100 тисяч осіб розроблення планів сталої міської мобільності (SUMP – Sustainable Urban Mobility Plan), що передбачають, серед іншого, впровадження цифрових сервісів для пасажирів та інтелектуальних транспортних систем (ITS) [5]. Виконання цих вимог неможливе без розгортання інфраструктури GPS-моніторингу та відповідного програмного забезпечення.

Ключові характеристики сучасного стану ринку міського пасажирського транспорту в Україні наведено у таблиці 1.1.

Таблиця 1.1 – Характеристика ринку міського пасажирського транспорту в Україні

Аспект	Характеристика
Частка громадського транспорту	До 60–70% міських поїздок у великих містах України здійснюється громадським транспортом [2]
Основні проблеми галузі	Нерегулярність руху, відсутність онлайн-відстеження, застаріла інфраструктура зупинок, фізичне зношення рухомого складу [3]
Рівень цифровізації	Менше 30% міст мають діючі системи онлайн-моніторингу транспорту [1]
Ключові тенденції	Впровадження електронних квитків, GPS-моніторингу, мобільних сервісів, інтеграція з концепцією Smart City [4]
Нормативна база	Закон України «Про міський електричний транспорт», нові стандарти ДСТУ щодо якості транспортних послуг, вимоги ЄС у рамках євроінтеграції [5]
Структура ринку	Поєднання муніципальних перевізників та приватних маршрутних таксі, різні рівні технічного оснащення та якості обслуговування [2]
Перспективи розвитку	Розширення електричного транспорту, впровадження MaaS (Mobility as a Service), розвиток мультимодальних платформ [6]

Аналіз даних таблиці 1.1 свідчить про те, що ринок міського пасажирського транспорту в Україні перебуває на перехідному етапі між традиційною моделлю функціонування та цифровою трансформацією. З одного боку, громадський

транспорт зберігає провідне місце у структурі міських переміщень, з іншого – рівень його цифровізації залишається суттєво нижчим за середньоєвропейські показники [5]. Це формує чіткий запит на комплексні рішення, що поєднують апаратне відстеження транспортних засобів із зручними мобільними інтерфейсами для пасажирів і водіїв. Саме таке рішення є предметом розробки даного дипломного проєкту.

Перспективи розвитку галузі пов’язані з реалізацією концепції розумного міста (Smart City), в рамках якої транспортна система розглядається як інтегрована цифрова платформа. Ключовими напрямками є розгортання IoT-інфраструктури на транспорті, впровадження алгоритмів предиктивної аналітики для оптимізації маршрутів та забезпечення безшовної мультимодальної мобільності [6]. Концепція MaaS (Mobility as a Service), що набуває поширення в Європі, передбачає об’єднання всіх видів міського транспорту в єдину цифрову платформу з єдиним квитком та уніфікованим інтерфейсом для користувача. Розроблювана комп’ютерна система для організації міських перевезень відповідає цим тенденціям та є кроком у напрямку реалізації принципів MaaS на українському ринку.

Таким чином, аналіз сучасного стану ринку міських перевезень в Україні підтверджує актуальність розробки комплексної комп’ютерної системи, яка здатна забезпечити реальне GPS-відстеження транспортних засобів, інформувати пасажирів у режимі реального часу та надати транспортним підприємствам ефективні інструменти управління рухомим складом [3].

1.2 Порівняльний аналіз існуючих мобільних додатків для організації міських перевезень

Для формування обґрунтованих технічних вимог до розроблюваної системи необхідно провести детальний аналіз найбільш поширених існуючих рішень у сфері мобільних додатків для громадського транспорту. Об’єктами аналізу обрано

три продукти, що представляють різні підходи до вирішення задачі організації міських перевезень: Moovit, Google Maps Transit та Citymapper [6]. Кожен із цих продуктів має характерну концепцію, переваги та обмеження, що дозволяє скласти повну картину наявних підходів і визначити незайняту ринкову нішу, яку заповнює розроблювана система.

Moovit є одним із найпоширеніших у світі додатків для громадського транспорту та заслуговує на детальний розгляд. Додаток заснований у 2012 році в Ізраїлі та у 2020 році придбаний компанією Intel Corporation за 900 мільйонів доларів, що свідчить про значну ринкову цінність цього рішення [7]. Станом на сьогодні Moovit охоплює понад 3500 міст у 112 країнах та нараховує більше мільярда запланованих поїздок на рік. Детальні характеристики Moovit наведено у таблиці 1.2.

Таблиця 1.2 – Характеристика додатку Moovit

Характеристика	Опис
Рік заснування	2012, Ізраїль; придбаний Intel у 2020 році за 900 млн доларів [7]
Охоплення	Понад 3500 міст у 112 країнах, більше 1 млрд запланованих поїздок на рік [7]
Джерела даних	GTFS-розклади перевізників + краудсорсингові дані від пасажирів у реальному часі [8]
Ключові функції	Мультимодальна маршрутизація, відображення часу прибуття, сповіщення про збої, аналітика для міст [7]
Обмеження	Відсутність власного апаратного забезпечення, залежність від краудсорсингу, немає інтерфейсу водія [8]
Монетизація	Реклама для користувачів, платний API для розробників та міських адміністрацій [7]

Принципова особливість Moovit полягає в тому, що точність відображення позицій транспорту значною мірою залежить від краудсорсингових даних – інформації, яку надсилають самі пасажири в режимі реального часу. У містах з великою базою активних користувачів ця модель забезпечує досить точне відстеження. Проте у містах з невеликою кількістю інсталяцій якість даних суттєво знижується: транспортні засоби відображаються із затримками або взагалі не відображаються [8]. Для українських міст середнього розміру, де кількість активних користувачів Moovit є незначною, цей недолік набуває принципового характеру. Окрім того, Moovit не передбачає окремого інтерфейсу для водіїв і не надає інструментів управління парком для транспортних підприємств, що виключає його використання як комплексного рішення для перевізника.

Google Maps Transit є частиною екосистеми Google Maps і пропонує мультимодальну маршрутизацію з урахуванням громадського транспорту. Сервіс відрізняється якісними картографічними матеріалами, зручним інтерфейсом та широкою інтеграцією з іншими сервісами Google. Характеристики Google Maps Transit наведено у таблиці 1.3.

Таблиця 1.3 – Характеристика сервісу Google Maps Transit

Характеристика	Опис
Компанія-розробник	Google LLC, США; функція Transit запущена у 2005 році [9]
Охоплення	Понад 10 000 міст у 150 країнах; інтегрована частина Google Maps [9]
Джерела даних	Офіційні GTFS-фіди від транспортних операторів; у ряді міст – реальний час через API перевізників [9]
Ключові функції	Мультимодальна маршрутизація, покрокова навігація, відображення заповненості вагонів, збережені маршрути [9]

Продовження таблиці 1.3

Характеристика	Опис
Обмеження	Потребує публікації GTFS від перевізника; відсутність власного GPS-відстеження; немає інструментів управління для перевізника [9]
Монетизація	Безкоштовний для споживачів; комерційний Maps Platform API для бізнесу [9]

Ключовим обмеженням Google Maps Transit є залежність від офіційних даних GTFS, що надаються транспортними операторами. У містах, де перевізники не публікують такі дані у стандартизованому форматі або публікують їх нерегулярно, точність інформації суттєво обмежена. В Україні більшість приватних перевізників не формують GTFS-фіди, що робить Google Maps Transit практично неефективним у малих і середніх містах [9]. Додаток також орієнтований виключно на кінцевого споживача і не включає жодних інструментів для адміністраторів транспортних підприємств. Вартість комерційного Maps Platform API є досить значною та може становити суттєву статтю витрат для невеликих транспортних підприємств.

Citymapper – додаток британського походження, що вирізняється детальним охопленням транспортної мережі та орієнтованістю на складні мультимодальні маршрути. Він підтримує реальне відстеження транспорту через API операторів та надає детальну інформацію про пересадки. Характеристики Citymapper наведено у таблиці 1.4.

Citymapper відрізняється від конкурентів особливою увагою до деталей маршруту та підтримкою мікромобільності – самокатів, велосипедів та пішохідних маршрутів. Проте станом на сьогодні додаток охоплює близько 100 міст, переважно у Великій Британії, Франції та США. Жодне українське місто у сервісі не представлено [10]. Citymapper також не має власного апаратного забезпечення для відстеження транспорту та не надає інструментів управління для перевізників.

Freemium-модель монетизації є цікавою з точки зору бізнесу, проте для реалізації в українських реаліях потребує адаптації.

Таблиця 1.4 – Характеристика додатку Citymapper

Характеристика	Опис
Компанія-розробник	Citymapper Ltd, Великобританія; заснований у 2011 році [10]
Охоплення	Близько 100 міст переважно у Великій Британії, Франції, Іспанії, США; українські міста не підтримуються [10]
Джерела даних	Офіційні API транспортних операторів, GTFS у реальному часі (GTFS-RT) [10]
Ключові функції	Детальна мультимодальна навігація, підтримка самокатів та велосипедів, режим «Go» для покрокового ведення [10]
Обмеження	Вузьке географічне охоплення, відсутність панелі для перевізника, немає власного апаратного забезпечення [10]
Монетизація	Freemium-модель; підписка Citymapper Pass для безлімітного проїзду в окремих містах [10]

Зведений порівняльний аналіз усіх трьох аналогів та розроблюваної системи наведено у таблиці 1.5.

З даних таблиці 1.5 видно, що жоден із проаналізованих аналогів не надає повноцінного комплексного рішення, яке поєднує власну апаратну інфраструктуру для відстеження транспорту, окремі інтерфейси для пасажирів і водіїв та адміністративну панель для транспортних підприємств. Moovit і Google Maps Transit зосереджені виключно на стороні пасажира та не мають засобів керування парком. Citymapper, попри розвинений функціонал, не підтримує українські міста та не має власного GPS-обладнання [6].

Таблиця 1.5 – Зведений порівняльний аналіз існуючих рішень та розроблюваної системи

Критерій	Moovit	Google Maps Transit	Citymapper	Розроблювана система
Платформа	iOS, Android	iOS, Android, Web	iOS, Android	iOS, Android
GPS-відстеження в реальному часі	Краудсорсинг	Частково (GTFS)	Так	Так (бортові модулі)
Власна апаратна інфраструктура	Ні	Ні	Ні	Так (GPS+GSM)
Інтерфейс водія	Ні	Ні	Ні	Так
Панель адміністратора	Ні	Ні	Ні	Так
Push-сповіщення	Так	Так	Так	Так
Офлайн-режим	Обмежено	Так (карти)	Обмежено	Часткова підтримка
Українська локалізація	Так	Так	Ні	Так
Підтримка малих міст	Ні	Ні	Ні	Так
Мультимодальний маршрут	Так	Так	Так	Так
Відкритий API	Так (платний)	Так (платний)	Ні	Так (внутрішній)

Принциповою перевагою розроблюваної системи є наявність власних бортових GPS-модулів, що встановлюються на транспортних засобах. Це забезпечує незалежність від краудсорсингових даних та гарантує точність

відображення позицій у реальному часі незалежно від кількості активних користувачів у місті. Завдяки цьому система є ефективною навіть на початковому етапі впровадження, коли база пасажирів ще невелика [11].

Орієнтація на українські міста середнього розміру, де рішення Moovit та Citymapper практично відсутні, а Google Maps Transit обмежений через відсутність GTFS-даних від місцевих перевізників, визначає чітку ринкову нішу та практичну цінність проєкту. Розроблювана система формує самодостатню екосистему, що не залежить від зовнішніх постачальників геолокаційних даних [8].

1.3 Вимоги до розробки комп'ютерної системи організації міських перевезень

Розробка комп'ютерної системи для організації міських перевезень є комплексним технічним завданням, що охоплює як апаратні, так і програмні компоненти. На основі проведеного аналізу ринку та існуючих аналогів, а також виходячи із потреб трьох ключових категорій користувачів – пасажирів, водіїв та адміністраторів перевізника – сформовано повний перелік вимог до системи [5]. Ці вимоги поділяються на функціональні, нефункціональні та вимоги до апаратної частини.

Функціональні вимоги визначають конкретні можливості системи, що повинні бути реалізовані для задоволення потреб кожної категорії користувачів. З точки зору пасажира система повинна надавати можливість перегляду карти міста з нанесеними маршрутами та актуальними позиціями транспортних засобів у реальному часі [3]. Ключовою функцією є розрахунок оптимального маршруту від точки відправлення до пункту призначення з урахуванням пересадок, часу очікування та часу в дорозі. Окремо важливими є відображення розрахункового часу прибуття транспорту до найближчої зупинки, push-сповіщення про зміни у

русі та затримки, а також збереження улюблених маршрутів і зупинок для швидкого доступу [4].

З точки зору водія мобільний додаток повинен забезпечувати автоматичну трансляцію GPS-координат транспортного засобу на сервер у фоновому режимі з інтервалом не більше 10 секунд. Інтерфейс водія повинен відображати поточний маршрут, перелік зупинок, поточне відхилення від розкладу та оперативні повідомлення від диспетчера [4]. Вкрай важливою є мінімізація взаємодії водія із додатком під час керування транспортним засобом – усі критично важливі функції повинні бути доступні без необхідності навігації між екранами. Додаток повинен функціонувати навіть за нестабільного мобільного з’єднання, накопичуючи дані для відправлення при відновленні зв’язку.

Адміністратор транспортного підприємства повинен мати доступ до веб-панелі управління, що дозволяє у реальному часі відстежувати місцезнаходження всього рухомого складу на інтерактивній карті [5]. Система повинна надавати інструменти для редагування маршрутів, зупинок і розкладу, управління обліковими записами водіїв, а також формування аналітичних звітів про дотримання інтервалів руху та якість обслуговування. Збереження архіву GPS-треків дозволяє аналізувати ретроспективні дані та виявляти системні проблеми в роботі транспорту.

Систематизовані функціональні вимоги за категоріями користувачів наведено у таблиці 1.6.

Нефункціональні вимоги визначають якісні характеристики системи, що забезпечують її надійну та ефективну роботу в реальних умовах. До основних нефункціональних вимог відносяться такі: час відгуку REST API сервера не повинен перевищувати 500 мс при нормальному навантаженні; система повинна підтримувати одночасну роботу не менше 500 активних GPS-трекерів та 5000 мобільних клієнтів; доступність сервісу повинна становити не менше 99,5% на місяць [9]. Оновлення позицій транспортних засобів на карті мобільного додатку повинно відбуватися з затримкою не більше 15 секунд від моменту фактичного оновлення GPS-координат на сервері.

Таблиця 1.6 – Функціональні вимоги до комп’ютерної системи за категоріями користувачів

Роль	Категорія вимог	Опис
Пасажир	Перегляд карти	Відображення актуальних позицій транспорту та маршрутів на карті міста в реальному часі [3]
Пасажир	Маршрутизація	Розрахунок оптимального маршруту від А до Б з урахуванням пересадок та часу очікування [3]
Пасажир	Сповіщення	Push-сповіщення про час прибуття транспорту до обраної зупинки та затримки у русі [4]
Пасажир	Улюблені	Збереження часто використовуваних маршрутів та зупинок для швидкого доступу [3]
Водій	Трансляція GPS	Автоматична передача GPS-координат на сервер у фоновому режимі з інтервалом 5–10 секунд [4]
Водій	Маршрут	Відображення поточного маршруту, переліку зупинок та відхилень від розкладу [4]
Водій	Зв’язок	Отримання повідомлень від диспетчера та підтвердження виконання рейсу [5]

Продовження таблиці 1.6

Роль	Категорія вимог	Опис
Адміністратор	Моніторинг	Перегляд поточного стану всього рухомого складу на карті у реальному часі [5]
Адміністратор	Управління	Редагування маршрутів, розкладу, управління обліковими записами водіїв [5]
Адміністратор	Аналітика	Формування звітів про дотримання інтервалів руху, архів GPS-треків [5]

Вимоги до безпеки передбачають шифрування всіх каналів передачі даних між компонентами системи за стандартом TLS 1.3, автентифікацію водіїв та адміністраторів із використанням JWT-токенів, а також обмеження доступу до адміністративних функцій на основі ролей [8]. Зберігання персональних даних пасажирів та водіїв повинно відповідати вимогам Закону України «Про захист персональних даних». Паролі користувачів зберігаються виключно у хешованому вигляді із застосуванням алгоритму bcrypt.

Апаратна частина системи включає бортові модулі, що встановлюються на транспортних засобах і забезпечують безперервну передачу геолокаційних даних на серверну платформу [7]. Вимоги до апаратного модуля включають: підтримку стандартів GPS та GLONASS для підвищення точності позиціонування до 2–5 метрів; наявність вбудованого GSM/LTE-модуля для передачі даних через стільникову мережу; живлення від бортової мережі транспортного засобу напругою 12–24 В постійного струму; захищений корпус відповідно до стандарту не нижче IP54 для забезпечення стійкості до вібрації, вологи та пилу; наявність внутрішнього буфера пам’яті для збереження GPS-треків у разі відсутності мобільного зв’язку [7].

Вимоги до процесу розробки та впровадження системи відображені у таблиці 1.7.

Таблиця 1.7 – Ключові завдання розробки комп’ютерної системи організації міських перевезень

Етап розробки	Опис завдань
Аналіз предметної області	Дослідження ринку міських перевезень України, виявлення потреб пасажирів, водіїв та диспетчерів [1]
Аналіз існуючих рішень	Порівняння Moovit, Google Maps Transit, Citymapper для визначення переваг та недоліків аналогів [6]
Формування вимог (SRS)	Складання специфікації з функціональними та нефункціональними вимогами до всіх компонентів системи
Проєктування апаратної частини	Вибір GPS-модулів, мікроконтролерів та засобів передачі даних для бортового обладнання [7]
Проєктування архітектури	Визначення технологічного стеку, структури бази даних, схеми взаємодії компонентів та REST API [8]
Розробка мобільного додатку	Реалізація інтерфейсів пасажира та водія з функціями відстеження, маршрутизації та сповіщень
Розробка серверної частини	Реалізація API, логіки обробки GPS-потоків, WebSocket-каналів та модуля сповіщень [9]
Тестування системи	Функціональне, інтеграційне та навантажувальне тестування всіх компонентів системи
Впровадження та підтримка	Розгортання системи на серверній інфраструктурі та забезпечення стабільної роботи в реальних умовах

Таблиця 1.7 охоплює весь цикл розробки – від початкового аналізу предметної області до впровадження та підтримки готової системи. Кожен із визначених етапів є логічно пов’язаним із попередніми та наступними, що забезпечує послідовний і контрольований процес розробки відповідно до методології гнучкої розробки програмного забезпечення [8].

Окремої уваги заслуговують вимоги до масштабованості системи. Архітектура повинна передбачати можливість поступового нарощування кількості

підключених транспортних засобів – від кількох одиниць рухомого складу одного підприємства до сотень транспортних засобів у рамках розгортання у масштабах міста [9]. Серверна частина повинна підтримувати горизонтальне масштабування, тобто можливість розподілення навантаження між кількома серверними вузлами без зміни архітектури. Модульність програмного забезпечення дозволяє додавати нові функції – наприклад, інтеграцію з системами безконтактної оплати або модуль прогнозування попиту – без необхідності переробки існуючого коду.

Вимоги до інтерфейсу користувача передбачають дотримання принципів Material Design для Android та Human Interface Guidelines для iOS, що забезпечує відповідність очікуванням користувачів різних платформ [3]. Мінімальна підтримувана версія операційної системи – Android 8.0 (API 26) та iOS 14, що охоплює понад 95% активних пристроїв. Інтерфейс повинен бути повністю локалізованим українською мовою, а також підтримувати режим високого контрасту для користувачів із порушеннями зору відповідно до стандарту WCAG 2.1 рівня AA.

1.4 Висновки за розділом 1

У першому розділі проведено комплексний аналіз предметної області та існуючих рішень у сфері організації міських перевезень, що дозволяє обґрунтувати доцільність розробки нової комп’ютерної системи та сформувати чіткі вимоги до неї.

Аналіз сучасного стану ринку міського пасажирського транспорту в Україні показав, що галузь перебуває на перехідному етапі між традиційною моделлю функціонування та цифровою трансформацією. Громадський транспорт зберігає провідне місце у структурі міських переміщень – частка поїздок сягає 60–70% у великих містах – проте рівень задоволеності пасажирів залишається низьким через нерегулярність руху та відсутність цифрових сервісів інформування [2]. Менше

30% українських міст мають функціонуючі системи онлайн-моніторингу транспорту, що формує значний незадоволений попит на відповідні рішення [1].

Порівняльний аналіз трьох провідних мобільних додатків для громадського транспорту Moovit, Google Maps Transit та Citymapper показав, що жоден із них не задовольняє комплексних потреб українського ринку. Moovit залежить від краудсорсингових даних, ефективність яких у невеликих українських містах є незначною. Google Maps Transit обмежений через відсутність GTFS-даних від більшості українських перевізників. Citymapper відрізняється якісним функціоналом, але не підтримує жодного українського міста. Жоден із аналізованих продуктів не надає інструментів для водіїв та адміністраторів транспортних підприємств і не має власної апаратної інфраструктури для відстеження транспорту [6].

На основі проведеного аналізу сформовано повний перелік вимог до розроблюваної комп'ютерної системи. Функціональні вимоги охоплюють потреби всіх трьох категорій користувачів: пасажирів, водіїв та адміністраторів перевізника. Нефункціональні вимоги забезпечують надійність, безпеку та масштабованість системи. Апаратні вимоги визначають характеристики бортових GPS-модулів для транспортних засобів [7].

Принциповою перевагою розроблюваної системи є поєднання власної апаратної інфраструктури з мобільним додатком та веб-панеллю управління, що формує замкнену екосистему, незалежну від зовнішніх постачальників геолокаційних даних. Орієнтація на міста середнього розміру, де проблема організації громадського транспорту стоїть особливо гостро, а існуючі рішення практично відсутні, визначає чітку ринкову нішу та практичну цінність проєкту [11]. Матеріали першого розділу є основою для проєктування архітектури системи та розробки специфікації вимог у наступних розділах.

2 ПРОЄКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ МІСЬКИХ ПЕРЕВЕЗЕНЬ

2.1 Розробка специфікації вимог до програмного забезпечення системи міських перевезень

Специфікація вимог до програмного забезпечення (SRS – Software Requirements Specification) є основоположним документом, що визначає очікувану поведінку системи та слугує орієнтиром для всіх подальших етапів проєктування і розробки. Для комп'ютерної системи організації міських перевезень специфікація вимог охоплює три основні категорії: функціональні вимоги, нефункціональні вимоги та вимоги до апаратного забезпечення.

2.1.1 Функціональні вимоги

Функціональні вимоги описують конкретні дії, які система повинна виконувати у відповідь на взаємодію з користувачем або зовнішнім обладнанням. Для розроблюваної системи функціональні вимоги формуються з урахуванням потреб трьох ролей: пасажира, водія та адміністратора транспортного підприємства. Кожній вимозі присвоєно унікальний код для забезпечення трасабельності в процесі розробки та тестування. Перелік ключових функціональних вимог наведено у таблиці 2.1.

Аналіз функціональних вимог, наведених у таблиці 2.1, свідчить про те, що система охоплює повний цикл взаємодії всіх учасників транспортного процесу. Вимоги FR-01 – FR-05 стосуються клієнтської частини для пасажирів і визначають базовий функціонал перегляду маршрутів та сповіщень. Вимоги FR-06 та FR-07 описують функціонал мобільного додатку для водія, що забезпечує передачу геолокаційних даних. Вимоги FR-08 – FR-10 визначають можливості адміністративної панелі для управління парком та аналізу якості обслуговування.

Таблиця 2.1 - Функціональні вимоги до комп'ютерної системи міських перевезень

Код	Категорія	Опис вимоги
FR-01	Авторизація	Система повинна забезпечувати реєстрацію та вхід користувачів за email та паролем з розділенням ролей: пасажир, водій, адміністратор
FR-02	Карта (пасажир)	Пасажирський додаток повинен відображати інтерактивну карту міста з позиціями всіх активних транспортних засобів у реальному часі з затримкою не більше 15 секунд
FR-03	Маршрутизація	Система повинна розраховувати оптимальний маршрут між двома точками з урахуванням актуального розкладу, пересадок та часу очікування
FR-04	ETA	Система повинна розраховувати та відображати очікуваний час прибуття (ETA) транспортного засобу до будь-якої зупинки на маршруті
FR-05	Сповіщення	Система повинна надсилати push-сповіщення пасажирам про прибуття транспорту, затримки та зміни у розкладі
FR-06	Трансляція GPS	Водійський додаток повинен передавати GPS-координати на сервер з інтервалом 5–10 секунд у фоновому режимі

Продовження таблиці 2.1

Код	Категорія	Опис вимоги
FR-07	Панель водія	Водійський інтерфейс повинен відображати поточний маршрут, наступну зупинку та поточне відхилення від розкладу в хвилинах
FR-08	Адмін-карта	Адміністративна панель повинна відображати всі активні транспортні засоби на карті у реальному часі з можливістю фільтрації за маршрутом
FR-09	Управління маршрутами	Адміністратор повинен мати можливість створювати, редагувати та деактивувати маршрути і зупинки через веб-інтерфейс
FR-10	Аналітика	Система повинна формувати звіти про дотримання інтервалів руху, середню швидкість та відхилення від розкладу за вибраний період

Загальна схема взаємодії між акторами системи та функціями, якими вони користуються, відображена у вигляді UML-діаграми варіантів використання (рис. 2.1).

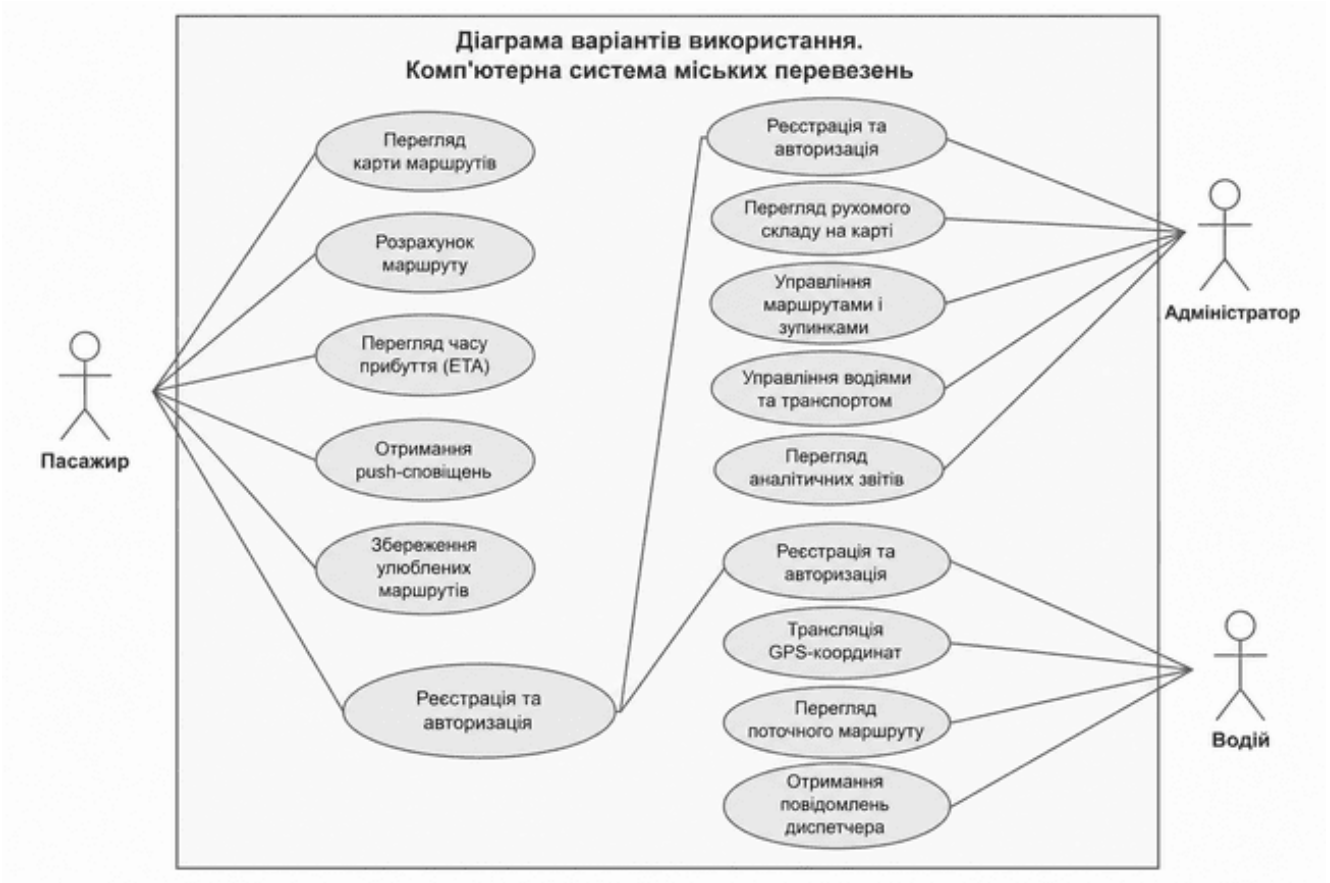


Рисунок 2.1 - Діаграма варіантів використання системи

2.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи продуктивність, безпеку, доступність та масштабованість, які не описують конкретні функції, але є критично важливими для успішного функціонування в реальних умовах. Ці вимоги безпосередньо впливають на вибір архітектурних рішень та технологічного стеку. Перелік нефункціональних вимог із кількісними показниками наведено у таблиці 2.2.

Таблиця 2.2 - Нефункціональні вимоги до комп'ютерної системи

Категорія	Показник	Значення / опис
Продуктивність	Час відгуку API	Не більше 500 мс при нормальному навантаженні (до 5000 одночасних клієнтів)

Продовження таблиці 2.2

Категорія	Показник	Значення / опис
Продуктивність	Пропускна здатність GPS	Підтримка одночасного прийому даних від 500 і більше активних GPS-модулів
Доступність	Uptime сервісу	Не менше 99,5% на місяць, що відповідає не більше 3,6 годин простою
Безпека	Шифрування каналів	TLS 1.3 для всіх зєднань між клієнтами та сервером
Безпека	Автентифікація	JWT-токени з терміном дії 24 години; refresh-токени зберігаються у захищеному сховищі пристрою
Масштабованість	Горизонтальне масштабування	Серверна частина повинна підтримувати розгортання у декілька вузлів за балансувальником навантаження
Сумісність	Мобільні платформи	Android 8.0 (API 26) і вище; iOS 14 і вище; охоплення не менше 95% активних пристроїв

2.1.3 Вимоги до апаратної частини

Вимоги до апаратної частини системи визначають технічні характеристики бортових GPS-модулів, що встановлюються на транспортних засобах і є фізичною основою всієї системи відстеження. Апаратний модуль повинен підтримувати позиціонування за системами GPS та GLONASS для забезпечення точності не гірше 5 метрів у міських умовах. Модуль повинен мати вбудований GSM/LTE-контролер для передачі даних через мобільну мережу, буферну пам'ять для збереження треків при відсутності зв'язку обсягом не менше 8 МБ, а також захищений корпус класу IP54, що дозволяє роботу при температурах від −30°C до +70°C. Живлення від бортової мережі 12–24 В постійного струму з вбудованим захистом від перенапруги є обов'язковою вимогою для надійної роботи на рухомому складі різних типів.

Окрема вимога стосується протоколу передачі даних між GPS-модулем та сервером. Обрано протокол MQTT (Message Queuing Telemetry Transport), який є галузевим стандартом для IoT-пристроїв із обмеженими ресурсами та нестабільним з'єднанням. MQTT забезпечує мінімальні накладні витрати на передачу даних – заголовок пакету займає лише 2 байти – що є критичним для пристроїв, що передають дані кожні 5–10 секунд на платній мобільній мережі. Резервним протоколом є HTTPS із JSON-тілом запиту, що використовується при недоступності MQTT-брокера.

2.2 Архітектура комп'ютерної системи з урахуванням апаратної та програмної складових

Вибір архітектурного рішення є одним із найважливіших проєктних рішень, що визначає всі подальші технічні характеристики системи. Для комп'ютерної системи організації міських перевезень розглянуто три основні архітектурні підходи: монолітна архітектура, мікросервісна архітектура та клієнт-серверна

архітектура. Порівняльний аналіз цих підходів стосовно вимог розроблюваної системи наведено у таблиці 2.3.

Таблиця 2.3 - Порівняльний аналіз архітектурних підходів

Критерій	Монолітна	Мікросервісна	Клієнт-серверна (обрана)
Складність розробки	Низька	Висока	Середня
Масштабованість	Обмежена	Висока	Висока
Розгортання	Просте	Складне (оркестрація)	Помірно просте
Підтримка в реальному часі	Складна реалізація	Добре	Добре (WebSocket)
Відповідність задачі	Незадовільна	Надмірна складність	Оптимальна

Монолітна архітектура передбачає розміщення всієї бізнес-логіки в одному застосунку. Цей підхід є простим у початковому розгортанні, проте при зростанні навантаження особливо в піки, характерні для ранкової та вечірньої годин пік виникають серйозні проблеми із масштабуванням. Оновлення будь-якого компоненту потребує перезапуску всього застосунку, що критично недопустимо для системи, яка має забезпечувати доступність 99,5%.

Мікросервісна архітектура розбиває систему на набір незалежних сервісів, кожен з яких відповідає за окрему функціональну область: GPS-процесинг, маршрутизацію, сповіщення, автентифікацію тощо. Попри очевидні переваги у гнучкості та незалежному масштабуванні окремих компонентів, мікросервісний підхід суттєво ускладнює розгортання, моніторинг та налагодження системи. Для системи, що розробляється у рамках дипломного проєкту, ця складність є надмірною та невиправданою.

Клієнт-серверна архітектура з чітким розділенням на рівні представлення, бізнес-логіки та даних є оптимальним вибором для розроблюваної системи. Вона

забезпечує необхідний рівень масштабованості завдяки можливості горизонтального нарощування серверних вузлів, підтримує роботу в реальному часі через WebSocket-з'єднання та залишається достатньо простою для розробки та підтримки.

Загальна архітектура розроблюваної системи наведена на рисунку 2.2. Система складається з трьох основних рівнів: апаратного (бортові GPS-модулі на транспортних засобах), серверного (API-сервер, GPS-процесор, двигун маршрутів, модуль сповіщень та база даних) та клієнтського (мобільний додаток для пасажирів, мобільний додаток для водіїв та веб-панель адміністратора).

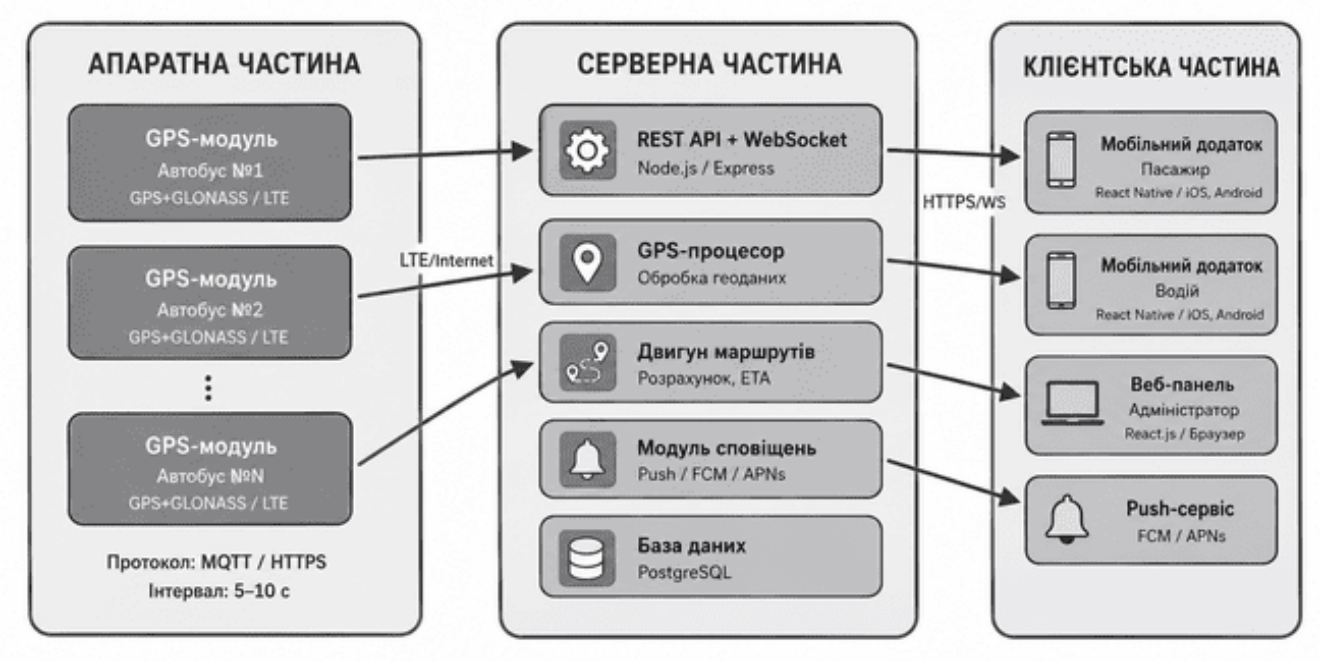


Рисунок 2.2 - Загальна архітектура комп'ютерної системи міських перевезень

Апаратний рівень системи складається з бортових GPS-модулів, встановлених на кожному транспортному засобі. Ці модулі отримують сигнали від супутникових навігаційних систем GPS та GLONASS, обчислюють поточні координати та швидкість руху і передають ці дані на сервер через мобільну мережу LTE за протоколом MQTT. Інтервал передачі даних становить 5–10 секунд у нормальному режимі та збільшується до 30 секунд при низькому рівні заряду

резервного живлення. У разі тимчасової втрати мобільного зв'язку модуль зберігає дані у внутрішньому буфері та надсилає їх пакетно при відновленні з'єднання.

Серверний рівень є центральним елементом архітектури та включає кілька логічних компонентів. API Gateway виступає єдиною точкою входу для всіх клієнтських запитів і GPS-потоків та відповідає за валідацію JWT-токенів, маршрутизацію запитів до відповідних сервісів та обмеження швидкості запитів (rate limiting). GPS-процесор отримує потік геолокаційних даних від бортових модулів, записує їх до бази даних та розраховує поточне відхилення транспортного засобу від запланованого розкладу. Двигун маршрутів обробляє запити пасажирів на пошук маршруту, використовуючи алгоритм Дейкстри для пошуку найкоротшого шляху в графі маршрутної мережі. Модуль сповіщень взаємодіє з сервісами Firebase Cloud Messaging (FCM) та Apple Push Notification Service (APNs) для доставлення push-повідомлень на пристрої пасажирів.

Клієнтський рівень представлений трьома типами застосунків. Мобільний додаток для пасажирів відображає карту з позиціями транспортних засобів та надає функції маршрутизації. Мобільний додаток для водіїв здійснює фонову трансляцію GPS-координат та відображає інформацію про поточний рейс. Веб-панель адміністратора реалізована як одностороннє застосування (SPA) та надає інструменти управління всіма компонентами системи.

Взаємодія між компонентами системи відображена на UML-діаграмі компонентів (рис. 2.3).

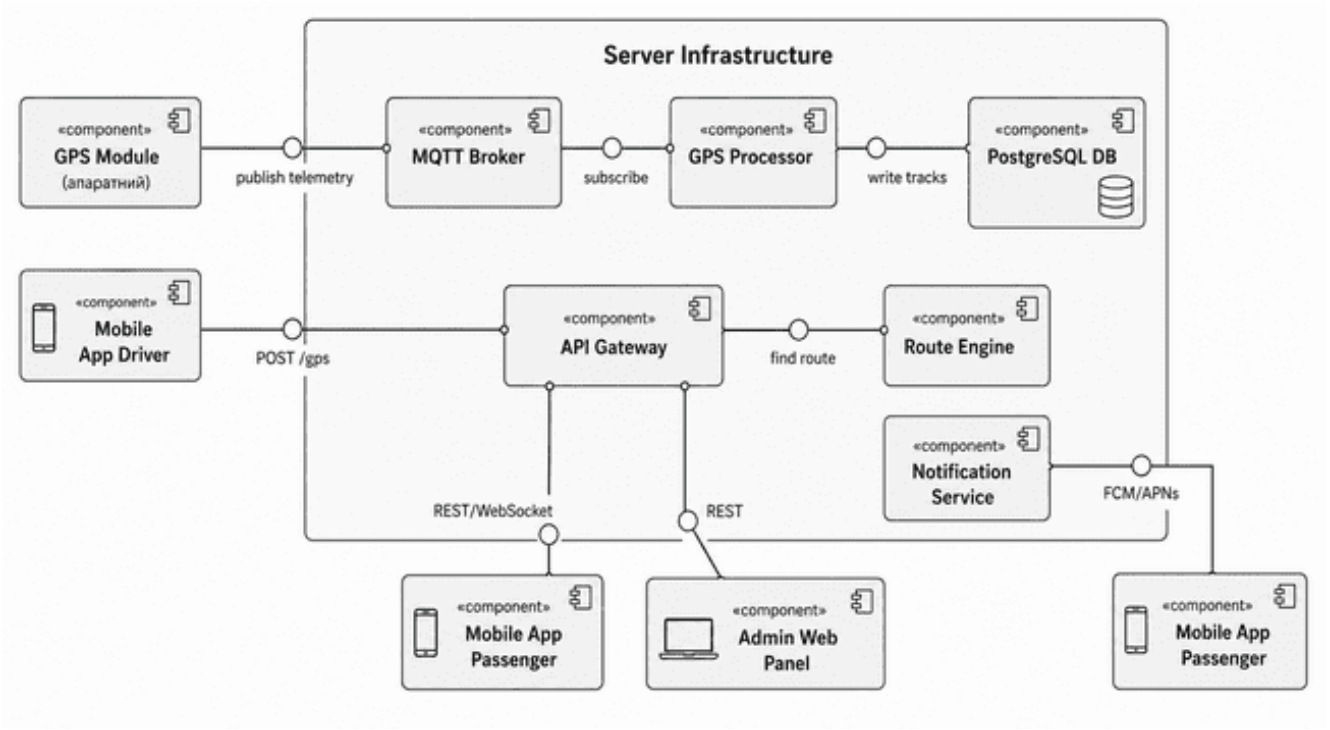


Рисунок 2.3 - UML-діаграма компонентів системи

Для обміну даними між мобільними клієнтами та сервером використовується REST API з форматом JSON для стандартних запитів і WebSocket – для потокових оновлень позицій транспорту в реальному часі. Пасажирський додаток підписується на WebSocket-канал конкретного маршруту і отримує оновлення координат щойно GPS-процесор обробляє новий запис. Це забезпечує відображення позицій транспорту на карті з мінімальною затримкою без необхідності постійного опитування (polling) сервера.

Взаємодія між компонентами при виконанні типового сценарію відстеження транспорту описана UML-діаграмою послідовності (рис. 2.4).

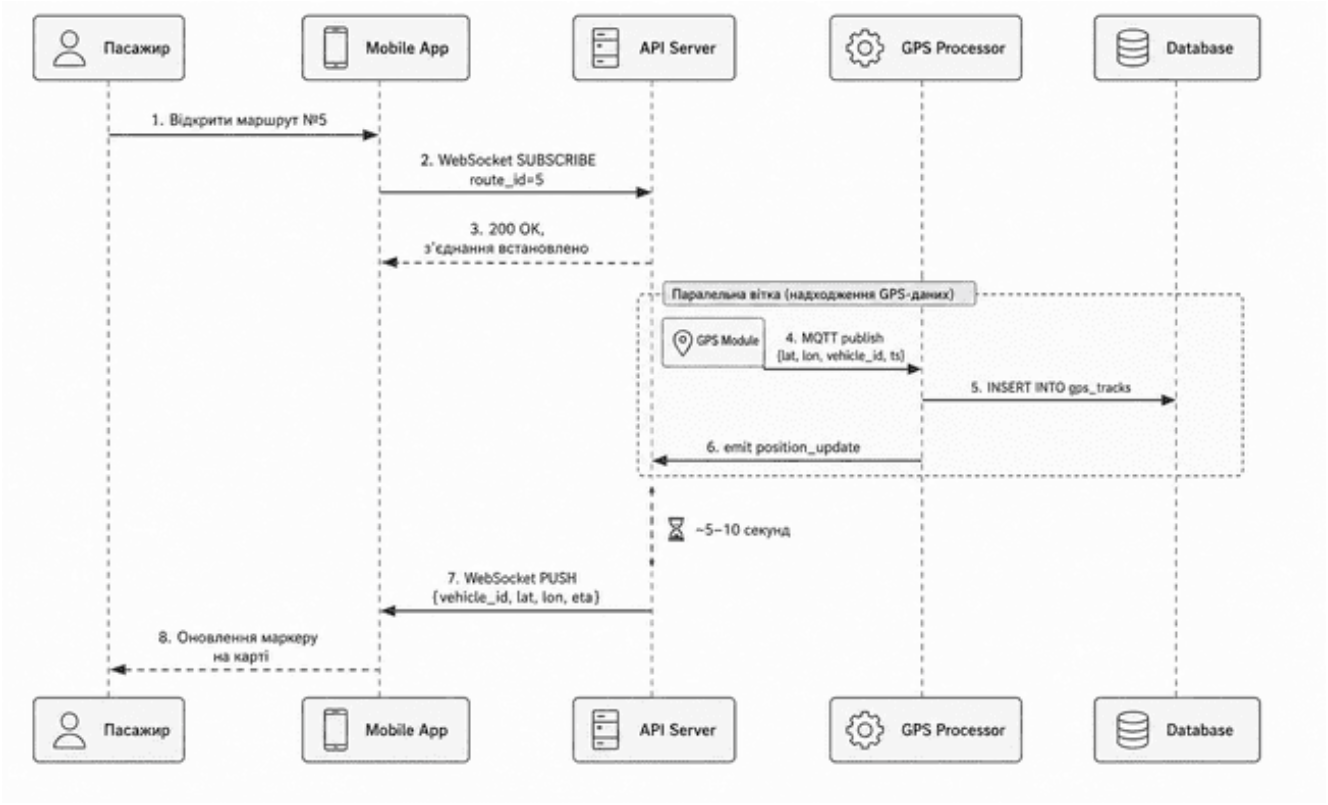


Рисунок 2.4 - UML-діаграма послідовності відстеження транспортного засобу

Потік даних у системі є чітко визначеним і однонаправленим: від апаратного рівня через серверну обробку до клієнтських застосунків. Детальна схема потоку GPS-даних від бортового модуля до мобільного додатку пасажира наведена на рисунку 2.5. Схема ілюструє п'ять ключових етапів обробки даних: збір координат GPS-модулем, передачу на API Gateway, обробку GPS-процесором, запис до бази даних та трансляцію через WebSocket Hub до мобільних клієнтів із загальною затримкою не більше 15 секунд.

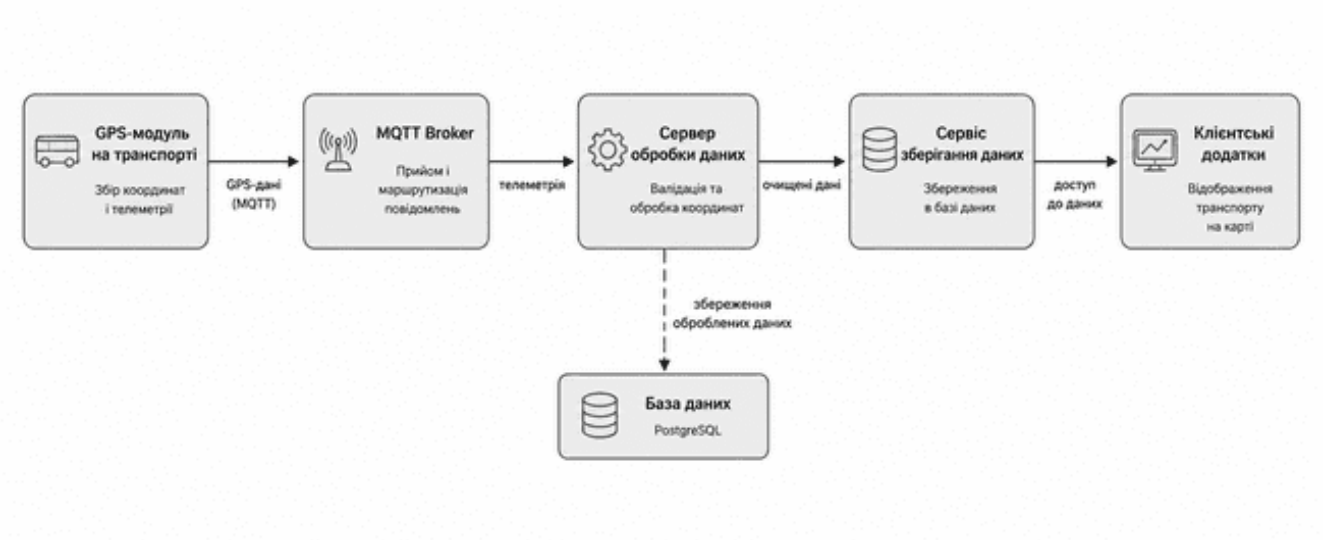


Рисунок 2.5 - Схема потоку GPS-даних від бортового модуля до мобільного додатку

Безпека системи забезпечується на кількох рівнях. На транспортному рівні всі з'єднання шифруються за допомогою TLS 1.3. На рівні автентифікації використовуються JWT-токени з коротким терміном дії та refresh-токени для їх оновлення. Доступ до адміністративних функцій обмежується на основі ролей (RBAC – Role-Based Access Control), що унеможливлює несанкціоноване внесення змін до маршрутів або даних водіїв. Бортові GPS-модулі автентифікуються на MQTT-брокері за унікальним сертифікатом пристрою, що запобігає підміні даних від стороннього джерела.

2.3 Проєктування бази даних системи маршрутизації та відстеження транспорту

База даних є центральним сховищем усіх даних системи і її проєктування є критично важливим для забезпечення цілісності даних, ефективності запитів та можливості масштабування. Для розроблюваної системи обрано реляційну базу даних PostgreSQL, яка поєднує надійність ACID-транзакцій, потужний механізм

індексування та підтримку спеціалізованих типів даних для географічних координат через розширення PostGIS.

Схема бази даних розроблена відповідно до принципів нормалізації до третьої нормальної форми (3НФ), що забезпечує мінімальну надмірність даних та цілісність зв'язків між сутностями. Центральне місце у схемі займають таблиці ROUTES та VEHICLES, навколо яких організовані всі інші сутності системи. Призначення кожної таблиці та склад ключових полів наведено у таблиці 2.4.

Таблиця 2.4 - Опис таблиць бази даних системи

Таблиця	Призначення та ключові поля
USERS	Зберігає всіх користувачів системи (пасажири, водії, адміністратори). Ключові поля: user_id (PK), role (ENUM), email, password_hash, is_active
OPERATORS	Транспортні підприємства-оператори. Пов'язана з таблицею ROUTES через operator_id. Ключові поля: operator_id (PK), name, edrpou, city
VEHICLES	Транспортні засоби рухомого складу. Пов'язана з ROUTES та GPS_TRACKS. Ключові поля: vehicle_id (PK), route_id (FK), plate_number, type, status
ROUTES	Маршрути руху транспорту. Центральна таблиця схеми. Ключові поля: route_id (PK), operator_id (FK), route_number, interval_min
STOPS	Зупинки міського транспорту з геокоординатами. Ключові поля: stop_id (PK), latitude, longitude, name, address
ROUTE_STOPS	Зв'язкова таблиця «маршрут – зупинка» з порядком зупинок. Ключові поля: route_id (FK), stop_id (FK), stop_order (INT)

Продовження таблиці 2.4

Таблиця	Призначення та ключові поля
GPS_TRACKS	Часовий ряд GPS-координат транспортних засобів. Найбільша таблиця за обсягом даних. Ключові поля: track_id (PK), vehicle_id (FK), latitude, longitude, speed, recorded_at
DRIVER_SESSIONS	Сесії роботи водія на конкретному транспортному засобі та маршруті. Ключові поля: session_id (PK), user_id (FK), vehicle_id (FK), started_at, ended_at

ER-діаграма бази даних із відображенням усіх таблиць, полів та зв'язків між ними наведена на рисунку 2.6. На діаграмі відображено вісім таблиць, з'єднаних лініями зв'язків з позначенням кардинальності відносин «один до багатьох» (1:N). Таблиця RESERVATIONS як центральна пов'язана з USERS, VEHICLES та ROUTES.

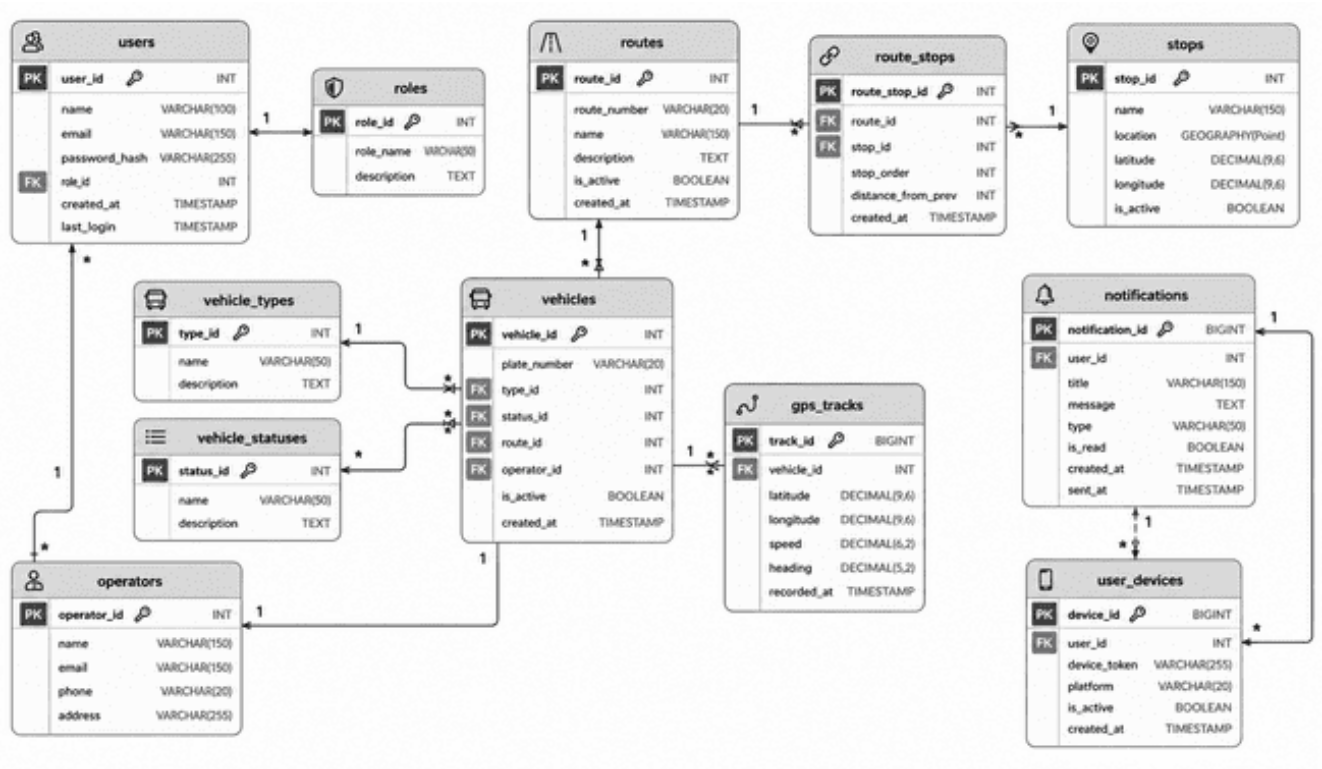


Рисунок 2.6 - ER-діаграма бази даних системи міських перевезень

Таблиця GPS_TRACKS є найбільшою за обсягом у всій схемі, оскільки кожен транспортний засіб генерує один запис кожні 5–10 секунд протягом усього робочого дня. При парку з 50 транспортних засобів і тривалості робочого дня 16 годин кількість записів за добу становитиме приблизно 300 000, а за рік – понад 100 мільйонів. Для забезпечення ефективності запитів до цієї таблиці застосовано стратегію партиціонування за часом (table partitioning by recorded_at) із щомісячними партиціями. Це дозволяє значно прискорити запити до даних за поточний місяць та спростити видалення застарілих записів старших за 90 днів.

Таблиця ROUTE_STOPS є проміжною зв'язковою таблицею між ROUTES та STOPS та реалізує відношення «багато до багатьох»: одна зупинка може бути на кількох маршрутах, і один маршрут включає кілька зупинок. Поле stop_order визначає порядковий номер зупинки на маршруті, що дозволяє системі відображати зупинки у правильній послідовності та розраховувати ETA для кожної з них на основі відстані між зупинками та поточної швидкості транспортного засобу.

Для таблиці GPS_TRACKS розроблено спеціальну стратегію індексування. Основний складений індекс створено на полях (vehicle_id, recorded_at DESC), що оптимізує найчастіший запит системи – отримання останньої відомої позиції транспортного засобу. Другий індекс на полях (recorded_at DESC) забезпечує ефективне виконання аналітичних запитів за часовим діапазоном при формуванні звітів адміністратора. У разі підключення розширення PostGIS до поля з географічними координатами додається просторовий GIST-індекс, що прискорює запити на пошук транспорту в заданому радіусі від точки користувача.

UML-діаграма класів, що відображає основні сутності предметної області та їхні зв'язки на рівні об'єктно-орієнтованого моделювання, наведена на рисунку 2.7.

Рисунок 2.7 - UML-діаграма класів системи

Цілісність даних у базі забезпечується засобами самої СУБД. Зовнішні ключі між таблицями визначені з каскадними правилами: видалення маршруту автоматично деактивує пов'язані записи у ROUTE_STOPS, а видалення транспортного засобу завершує всі незакриті сесії водія. Поля з критично важливими перерахуваннями (role, status, type) реалізовані як PostgreSQL ENUM-типи, що унеможливлює запис некоректних значень на рівні бази даних. Тригери на таблиці GPS_TRACKS автоматично оновлюють поле last_seen у таблиці VEHICLES після кожного нового запису трекінгових даних, усуваючи необхідність окремого запиту для оновлення статусу транспортного засобу.

Резервне копіювання бази даних організоване у вигляді щоденних повних дампів та безперервних WAL-архівів (Write-Ahead Logging), що забезпечує можливість відновлення даних з точністю до будь-якого моменту часу (PITR – Point-in-Time Recovery). Резервні копії зберігаються в окремому хмарному сховищі з терміном зберігання 90 днів для повних дампів та 7 днів для WAL-архівів.

2.4 Висновки за розділом 2

У другому розділі виконано повний цикл проєктування комп'ютерної системи для організації міських перевезень, що охоплює розробку специфікації вимог, вибір та обґрунтування архітектурного рішення, а також проєктування структури бази даних.

Розроблена специфікація вимог (SRS) визначає 10 ключових функціональних вимог, структурованих за ролями користувачів: пасажира, водія та адміністратора. Нефункціональні вимоги встановлюють кількісні показники продуктивності (час відгуку API не більше 500 мс), доступності (uptime 99,5%), безпеки (TLS 1.3, JWT-автентифікація) та масштабованості системи. Вимоги до апаратної частини визначають характеристики бортових GPS-модулів, включаючи підтримку GPS/GLONASS, клас захисту IP54 та використання протоколу MQTT для передачі даних.

За результатами порівняльного аналізу монолітної, мікросервісної та клієнт-серверної архітектур обрано клієнт-серверну архітектуру з трирівневою структурою як оптимальну для завдань розроблюваної системи. Ця архітектура забезпечує необхідний рівень масштабованості та підтримку роботи в режимі реального часу через WebSocket-з'єднання, залишаючись при цьому достатньо простою для розробки та підтримки. Система складається з апаратного рівня (GPS-модулі), серверного рівня (API Gateway, GPS-процесор, двигун маршрутів, модуль сповіщень, PostgreSQL) та клієнтського рівня (мобільні додатки та веб-панель).

Спроєктована база даних включає вісім таблиць, що охоплюють усі ключові сутності предметної області: користувачів, операторів, маршрути, зупинки, транспортні засоби та GPS-треки. Застосування стратегії партиціонування таблиці GPS_TRACKS за часом та спеціалізоване індексування забезпечують ефективну обробку великих обсягів геолокаційних даних. Цілісність даних гарантується зовнішніми ключами, ENUM-типами та тригерами на рівні СУБД. Розроблені

специфікація вимог, архітектура та схема бази даних є повною основою для практичної розробки системи у наступному розділі.

3 РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ МІСЬКИХ ПЕРЕВЕЗЕНЬ

3.1 Технологічний стек мобільного додатку та серверної частини

Вибір технологічного стеку є стратегічним рішенням, що безпосередньо впливає на продуктивність, масштабованість та довготривалу підтримуваність системи. При формуванні стеку для комп'ютерної системи організації міських перевезень враховувалися такі критерії: підтримка роботи в режимі реального часу, кросплатформність мобільного клієнта, зрілість спільноти та доступність відкритих бібліотек, а також відповідність вибраних технологій вимогам щодо продуктивності та безпеки.

Загальна схема технологічного стеку з розподілом по рівнях системи наведена на рисунку 3.1. Схема відображає чотири рівні: клієнтський (React Native, React.js, Mapbox, Socket.io, FCM), серверний API та бізнес-логіка (Node.js, Express.js, Socket.io, Mosquitto MQTT, JWT), рівень даних (PostgreSQL із PostGIS, Redis, Sequelize ORM, об'єктне сховище) та інфраструктурний рівень (Docker, Nginx, GitHub CI/CD, Grafana, Ubuntu Server).

Рисунок 3.1 - Технологічний стек комп'ютерної системи міських перевезень

Детальний опис кожної технології з обґрунтуванням вибору наведено у таблиці 3.1.

Таблиця 3.1 - Технологічний стек та обґрунтування вибору компонентів

Рівень	Технологія	Призначення та обґрунтування вибору
Клієнт (mobile)	React Native 0.74	Крос-платформна розробка iOS та Android з єдиною кодовою базою; підтримка фонових сервісів для трансляції GPS водія
Клієнт (web)	React.js + Redux Toolkit	SPA для адміністративної веб-панелі; Redux Toolkit забезпечує централізоване управління станом великого обсягу транспортних даних
Карти	Mapbox GL JS / React Native Maps	Рендеринг векторних карт, відображення кастомних маркерів транспортних засобів та побудова маршрутів з анімацією руху

Продовження таблиці 3.1

Рівень	Технологія	Призначення та обґрунтування вибору
Сервер	Node.js 20 LTS + Express.js	Асинхронна подієво-орієнтована архітектура ідеально підходить для обробки великої кількості одночасних GPS-потоків; Express забезпечує маршрутизацію REST API
WebSocket	Socket.io	Двонаправлена комунікація в реальному часі; автоматичне відновлення зєднання та кімнати (rooms) для підписки на конкретний маршрут
IoT-брокер	Eclipse Mosquitto MQTT	Легковагий брокер повідомлень для GPS-модулів; протокол MQTT з мінімальними накладними витратами на передачу координат кожні 5–10 секунд
База даних	PostgreSQL 16 + PostGIS	ACID-транзакції, партиціонування таблиці GPS_TRACKS, просторові запити через PostGIS для пошуку найближчого транспорту
Кеш	Redis 7	Кешування актуальних позицій транспортних засобів; зберігання JWT refresh-токенів; черги задач для відкладеного надсилання push-повідомлень
Безпека	JWT + bcrypt + TLS 1.3	JWT для автентифікації API-запитів; bcrypt для хешування паролів (cost factor 12); TLS 1.3 для шифрування всіх каналів
Push-сповіщення	Firebase Cloud Messaging	Доставлення сповіщень на Android та iOS через єдиний уніфікований API; підтримка тематичних підписок за маршрутом

Продовження таблиці 3.1

Рівень	Технологія	Призначення та обґрунтування вибору
Інфраструктура	Docker + Docker Compose	Контейнеризація всіх сервісів для відтворюваного розгортання; Docker Compose для оркестрації у dev та staging середовищах
Веб-сервер	Nginx	Зворотний проксі-сервер; термінація TLS; балансування навантаження між вузлами API; роздача статичних файлів SPA

React Native обрано основним фреймворком для розробки мобільного додатку завдяки можливості використання єдиної JavaScript-кодової бази для iOS та Android платформ. Це скорочує час розробки майже вдвічі порівняно з нативними підходами при збереженні близької до нативної продуктивності інтерфейсу. Ключовою перевагою для даного проєкту є підтримка фонових сервісів – Background Location Services – що дозволяє водійському додатку безперервно транслювати GPS-координати навіть при згорнутому інтерфейсі. Це досягається через нативні модулі React Native Background Geolocation, що напряму взаємодіють з CoreLocation (iOS) та LocationManager (Android) без посередників.

Node.js обрано для серверної частини системи через його архітектурну відповідність задачі обробки великої кількості одночасних з'єднань з мінімальними затримками. Подієво-орієнтована асинхронна модель виконання Node.js дозволяє одному процесу обробляти тисячі одночасних WebSocket-з'єднань та MQTT-потоків від GPS-модулів без блокування. Це є принциповою перевагою над традиційними багатопотоковими серверами для задач з інтенсивним введенням-виведенням, якою є системи реального часу. Менеджер пакетів npm надає доступ до понад двох мільйонів готових бібліотек, що суттєво прискорює розробку допоміжних модулів.

PostgreSQL обрано як основну систему управління базами даних завдяки підтримці розширення PostGIS, що надає повноцінні засоби роботи з

географічними даними безпосередньо на рівні СУБД. PostGIS дозволяє виконувати просторові запити – наприклад, знайти всі транспортні засоби в радіусі 500 метрів від поточного місцезнаходження пасажира – з використанням індексованих GIST-запитів, що є на порядки ефективнішим за аналогічні розрахунки на рівні застосунку. Механізм партиціонування таблиць PostgreSQL вирішує проблему зростання таблиці GPS_TRACKS, забезпечуючи стабільну продуктивність запитів незалежно від загального обсягу даних.

Redis відіграє роль кешу для зберігання актуальних позицій транспортних засобів у форматі хеш-карти, де ключем є ідентифікатор транспортного засобу, а значенням – JSON-об'єкт з координатами та часовою міткою. Завдяки цьому запит актуальної позиції конкретного транспортного засобу виконується за мікросекунди без звернення до PostgreSQL. Redis також використовується для зберігання JWT refresh-токенів з автоматичним видаленням після закінчення терміну дії, що спрощує реалізацію механізму відкликання токенів.

Протокол MQTT реалізовано через брокер Eclipse Mosquitto, що є одним із найбільш поширених та перевірених MQTT-брокерів з відкритим вихідним кодом. Mosquitto підтримує до 100 000 одночасних клієнтських з'єднань на одному сервері, що забезпечує значний запас масштабованості. Кожен GPS-модуль публікує дані у тему формату vehicles/{vehicle_id}/position, а GPS-процесор підписується на всі теми через маску vehicles/+/position. Таке розподілення тем дозволяє легко масштабувати систему, додаючи нових підписників без зміни логіки публікації.

3.2 Апаратна частина комп'ютерної системи

Апаратна частина системи є її фізичною основою, що забезпечує отримання геолокаційних даних безпосередньо з транспортних засобів. Бортовий GPS-модуль – це спеціалізований електронний пристрій, що встановлюється на кожному

транспортному засобі рухомого складу і виконує функції позиціонування, обробки даних та їх передачі на серверну платформу через мобільну мережу.

Структурна схема апаратного GPS-модуля наведена на рисунку 3.2. Схема відображає блоки пристрою та зв'язки між ними: GNSS-антена → GNSS-приймач u-blox NEO-M9N → мікроконтролер ESP32/STM32 (центральний елемент) → GSM/LTE модем SIM7600. Від мікроконтролера також відходять лінії до Flash-пам'яті (буфер 8 МБ), блоку живлення DC-DC (12–24 В), RTC-годинника та шини CAN/UART для підключення до бортової мережі транспортного засобу. Всі компоненти розміщено у захищеному корпусі IP54 із зазначеними габаритами та діапазоном температур.

Рисунок 3.2 - Структурна схема апаратного GPS-модуля

Детальні технічні характеристики компонентів апаратної частини наведено у таблиці 3.2.

Таблиця 3.2 - Технічні характеристики компонентів апаратного GPS-модуля

Компонент	Модель / специфікація	Технічні характеристики
GNSS-приймач	u-blox NEO-M9N	Підтримка GPS, GLONASS, Galileo, BeiDou; точність CEP 2,5 м; оновлення позиції до 25 Гц; робоча температура -40…+85°C
GSM/LTE-модем	SIMCom SIM7600E-H	Cat-4 LTE; швидкість до 150 Мбіт/с (download); підтримка MQTT та TCP/IP стека; вбудований TCP/IP стек
Мікроконтролер	ESP32-S3 / STM32F4	Dual-core 240 МГц; 8 МБ Flash; 512 КБ RAM; підтримка UART, SPI, I2C для підключення периферії
Буферна пам'ять	MicroSD 8 ГБ (SLC)	Запис до 10 000 GPS-записів офлайн при відсутності мобільного зв'язку; клас витривалості промисловий
Блок живлення	DC-DC 12–24 В → 5 В/3,3 В	Вхідна напруга 9–36 В; ККД 92%; захист від перенапруги та короткого замикання; власне споживання 1,5 Вт
RTC-модуль	DS3231 / PCF8523	Резервне живлення від CR2032; точність ±2 ppm; синхронізація з GPS-сигналом для точних часових міток треків
Корпус	ABS пластик + герметичний	Клас захисту IP54; кріплення DIN-рейка або на гвинтах; вібростійкість за IEC 60068-2-6; розміри 120×80×35 мм

GNSS-приймач u-blox NEO-M9N є одним із найбільш сучасних мультисистемних приймачів у своєму класі. Підтримка чотирьох навігаційних систем – GPS (США), GLONASS (Росія), Galileo (ЄС) та BeiDou (Китай) – дозволяє приймачу одночасно відстежувати до 72 супутників, що забезпечує виняткову точність і стабільність позиціонування навіть у складних умовах щільної міської забудови, де частина супутників перекривається будівлями. Типова горизонтальна точність становить 2,5 м CEP (Circular Error Probable) при роботі у відкритому просторі та 5–10 м в умовах міських каньйонів.

Мікроконтролер ESP32-S3 виконує роль центрального обробного елемента модуля. Двоядерний процесор Xtensa LX7 з тактовою частотою 240 МГц забезпечує достатню обчислювальну потужність для одночасного виконання кількох задач: читання NMEA-рядків від GNSS-приймача, формування JSON-пакету з координатами, управління чергою відправлення через GSM-модем та обслуговування локального буфера пам'яті. Вбудований модуль Wi-Fi дозволяє оновлювати прошивку пристрою дистанційно (OTA – Over The Air update) без необхідності фізичного доступу до транспортного засобу.

Логіка формування пакету даних GPS-модуля реалізована таким чином. Приймач генерує рядки NMEA-0183 стандарту з частотою 10 разів на секунду. Мікроконтролер зчитує рядки $GNGGA та $GNRMC, що містять координати, висоту, швидкість та азимут руху. Після парсингу NMEA-рядків формується JSON-об'єкт такої структури.

Лістинг 3.1 - Структура JSON-пакету від GPS-модуля

```
{
  "vehicle_id": "UA-ZP-2024-007",
  "lat": 47.838542,
  "lon": 35.124873,
  "speed": 32.4,
  "heading": 187,
  "ts": 1716825600123,
  "signal": 9
}
```

Поле signal містить цілочисельне значення від 0 до 9, що відображає якість прийому GNSS-сигналу. При значенні нижче 3 координата вважається ненадійною і позначається прапором низької якості на сервері. Поле heading містить курсовий кут руху у градусах (0–360), що дозволяє коректно орієнтувати іконку транспортного засобу на карті мобільного додатку у напрямку його фактичного руху.

Механізм буферизації даних є важливою складовою надійності системи. При втраті мобільного зв'язку модуль записує GPS-пакети у циклічний буфер Flash-пам'яті об'ємом 8 МБ. Цей обсяг забезпечує зберігання приблизно 40 000 пакетів, що відповідає більш ніж 4 годинам автономної роботи без зв'язку при інтервалі записів 10 секунд. При відновленні з'єднання модуль відправляє накопичені дані пакетами по 100 записів, що дозволяє відновити повну картину треку без прогалин. Сервер при прийомі буферизованих даних позначає їх прапором buffered та вставляє у таблицю GPS_TRACKS зі збереженням оригінальних часових міток.

Зовнішній вигляд апаратного GPS-модуля у зборі наведено на рисунку 3.3.

Рисунок 3.3 - Апаратний GPS-модуль у захищеному корпусі IP54

Встановлення GPS-модуля на транспортному засобі виконується відповідно до інструкції монтажу. Пристрій розміщується у внутрішньому просторі кабіни або технічному відсіку, де є доступ до бортової мережі 12 В та достатній рівень GNSS-сигналу через лобове скло або зовнішня антена виводиться на дах транспортного засобу на магнітній основі. Підключення до бортової мережі здійснюється через стандартний роз'єм OBD-II або безпосередньо до акумуляторного виводу через запобіжник 3 А.

3.3 Програмна частина мобільного додатку та серверної логіки

Програмна частина системи складається з трьох основних застосунків: мобільного додатку для пасажирів, мобільного додатку для водіїв (обидва розроблені в єдиній React Native кодовій базі з роздільними навігаційними стеками) та веб-адміністративної панелі на React.js. Серверна частина реалізована на Node.js та включає REST API, WebSocket-сервер та MQTT-клієнт для прийому даних від GPS-модулів.

Структура компонентів мобільного додатку у вигляді дерева навігаційних стеків наведена на рисунку 3.4 (рис. 3.4). Схема відображає кореневий компонент App.tsx з NavigationContainer, від якого розгалужуються три навігаційні стеки: Auth Stack (Login, Register), Passenger Tab Navigator (MapScreen, RouteScreen, ProfileScreen) та Driver Stack (SessionScreen, DriverMapScreen, MessagesScreen). Для адміністраторів передбачено окремий веб-SPA (Admin Web Panel) з розділами Fleet Map, Route Editor та Reports.

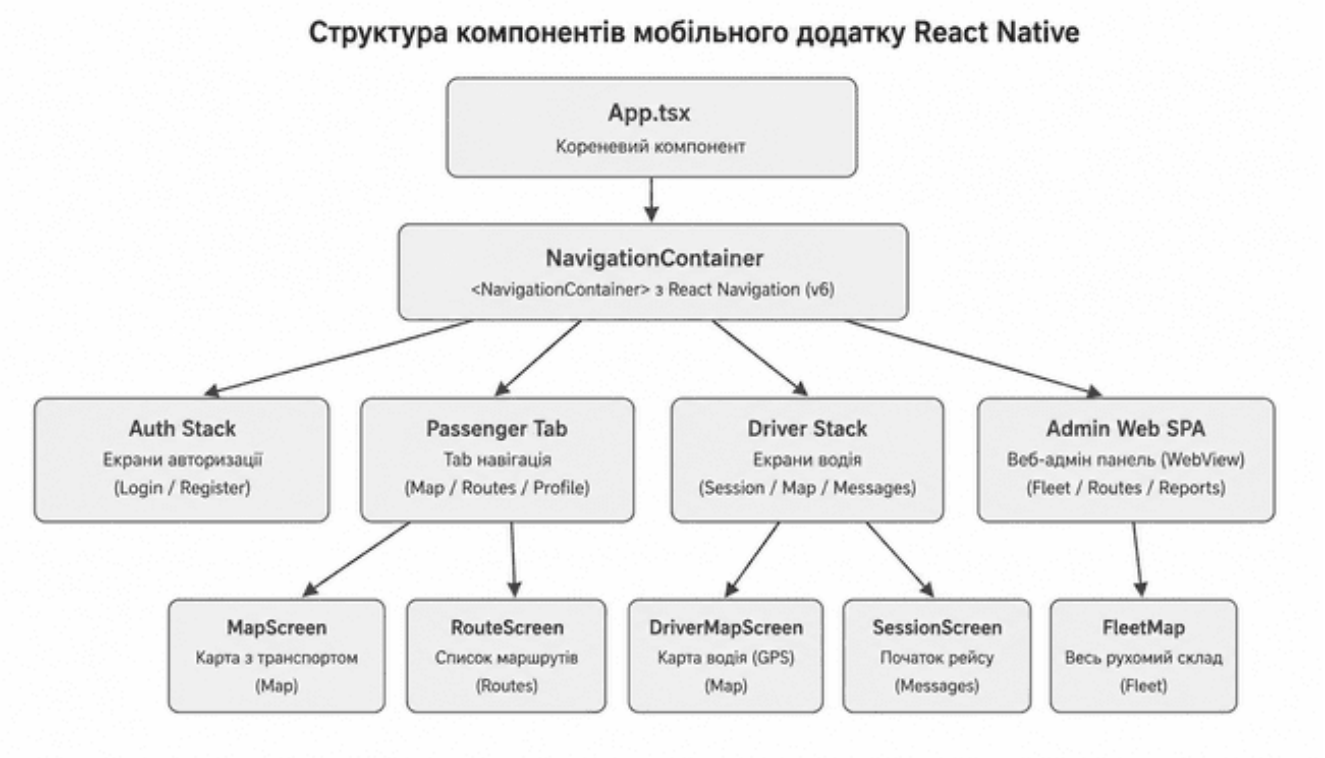


Рисунок 3.4 - Структура компонентів мобільного додатку React Native

Головний екран пасажирського додатку відображає інтерактивну карту міста з маркерами транспортних засобів у реальному часі (рис. 3.5). На екрані відображено повноекранну векторну карту міста у стилі Mapbox Streets. На карті розміщено кольорові маркери транспортних засобів у вигляді стрілок, орієнтованих у напрямку руху: сині стрілки – тролейбуси, червоні – автобуси, зелені – трамваї. При натисканні на маркер з'являється спливаючий балун з інформацією: номер маршруту, тип транспорту, швидкість, ETA до наступної зупинки. У нижній частині екрана розміщено панель пошуку маршруту з двома полями «Звідки» та «Куди». У верхній частині – рядок стану з іконками: геолокація, налаштування, сповіщення. На карті також відображено лінії маршрутів різними кольорами, що відповідають кольорам маркерів транспорту.

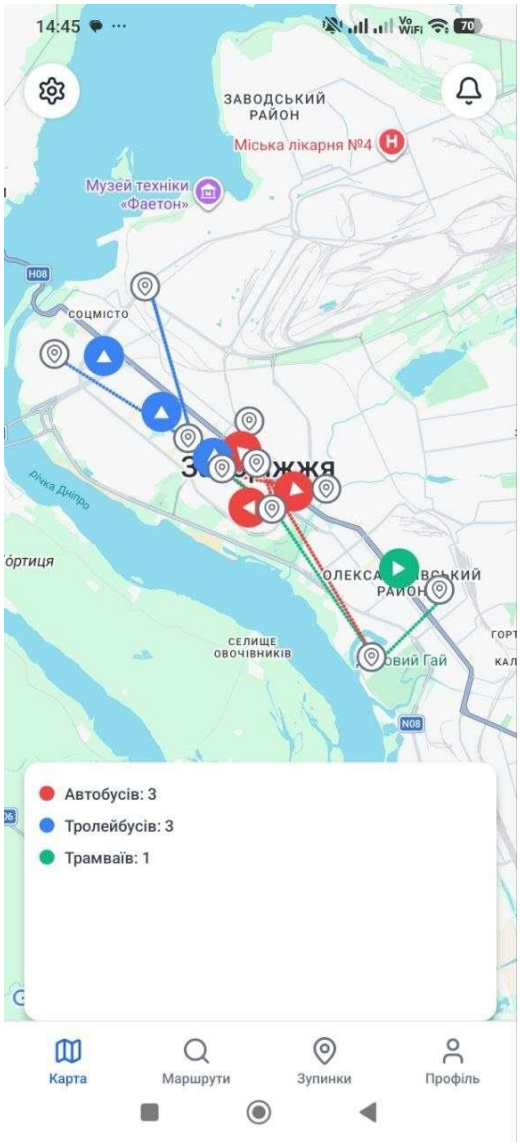


Рисунок 3.5 - Головний екран пасажирського додатку з картою та маркерами транспорту

Реалізацію підписки на WebSocket для отримання оновлень позицій транспортних засобів у реальному часі на стороні клієнта ілюструє лістинг 3.2.

Лістинг 3.2 - Підписка на WebSocket-оновлення позицій транспорту (React Native)

```
useEffect(() => {
  const socket = io(API_URL, { auth: { token } });
  socket.emit("subscribe_route", { route_id: selectedRoute });
  socket.on("position_update", (data) => {
    dispatch(updateVehiclePosition(data));
  });
```

```
        return () => socket.disconnect();
    }, [selectedRoute]);
```

Екран пошуку та перегляду маршруту надає пасажиру інструменти для планування поїздки (рис. 3.6).

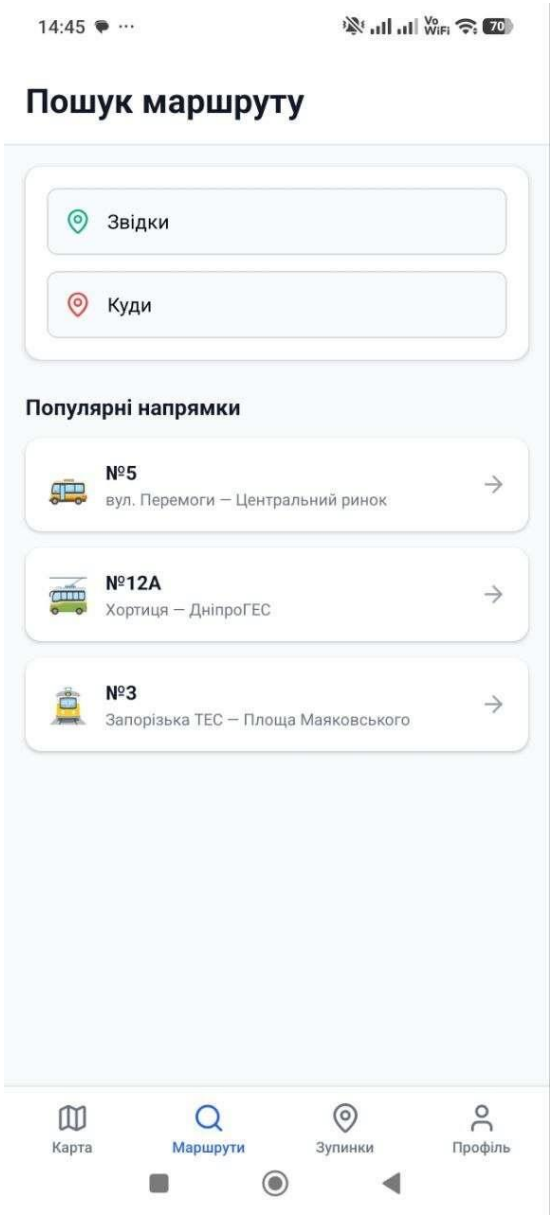


Рисунок 3.6 - Екран пошуку маршруту з варіантами поїздки та часом очікування

Екран деталей зупинки відображає розклад прибуття транспорту для вибраної зупинки (рис. 3.7).

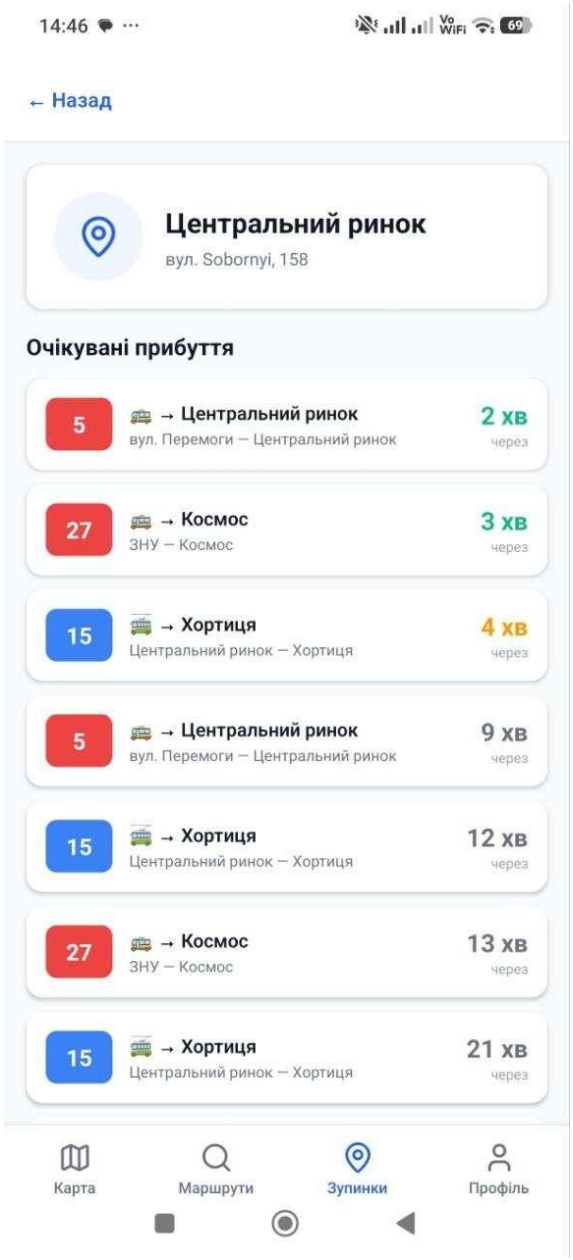


Рисунок 3.7 - Екран деталей зупинки з очікуваним часом прибуття транспорту

Мобільний додаток для водія має спрощений та безпечний інтерфейс, мінімально відволікає водія під час руху і виконує фонову трансляцію GPS-координат (рис. 3.8).

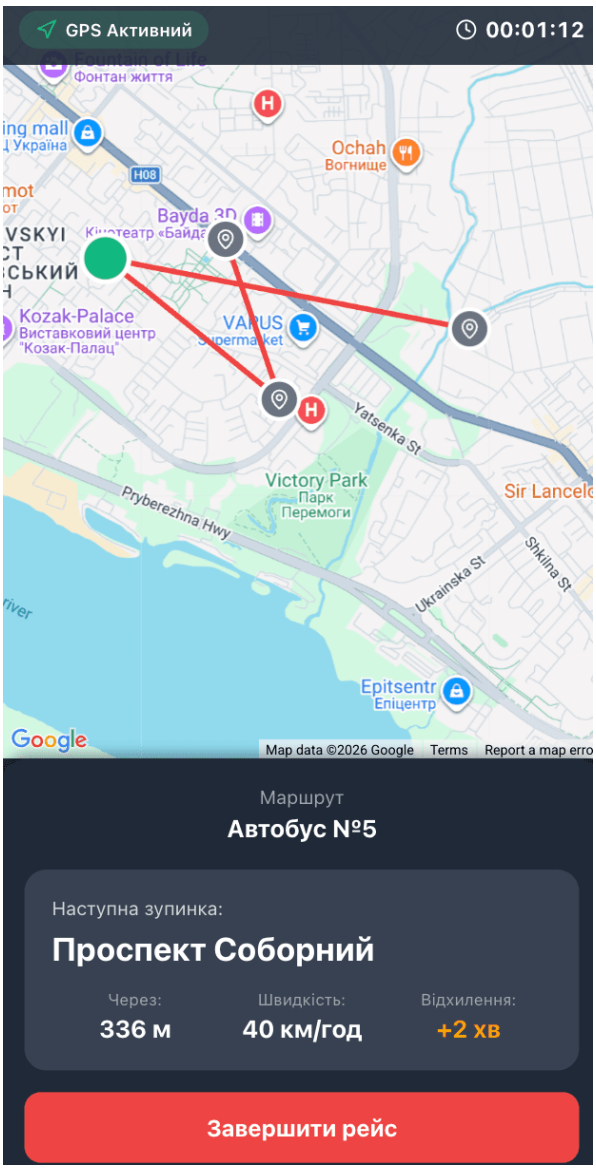


Рисунок 3.8 - Головний екран додатку водія під час активного рейсу

Фонова трансляція GPS на стороні водійського додатку реалізована через нативний сервіс, що продовжує роботу навіть при згорнутому додатку. Серверний обробник MQTT-повідомлень, що приймає GPS-дані та розповсюджує їх підписникам, наведено у лістингу 3.3.

Лістинг 3.3 - Серверний обробник MQTT-повідомлень від GPS-модулів (Node.js)

```
mqttClient.on("message", async (topic, payload) => {
  const vehicleId = topic.split("/")[1];
  const data = JSON.parse(payload.toString());
  await GpsTrack.create({
```

```
      vehicle_id: vehicleId, latitude: data.lat,
      longitude: data.lon, speed: data.speed,
      recorded_at: new Date(data.ts)
    });
    await redis.hset("vehicles:positions", vehicleId,
      JSON.stringify(data));
    io.to(`route_${data.route_id}`)
      .emit("position_update", { vehicleId, ...data });
});
```

Адміністративна веб-панель надає диспетчеру повну картину поточного стану транспортної мережі. Головний екран панелі відображає весь рухомий склад на інтерактивній карті (рис. 3.9).

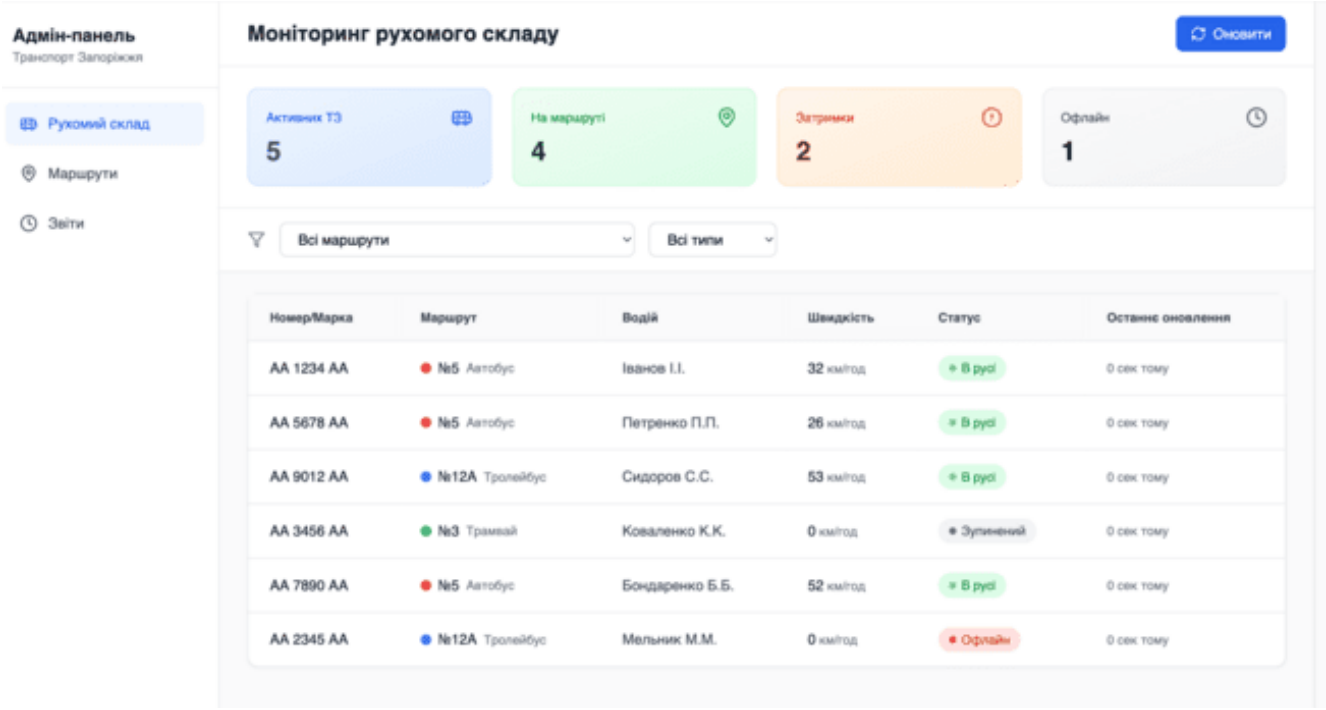


Рисунок 3.9 - Головний екран адміністративної панелі з моніторингом рухомого складу

Перелік REST API ендпоінтів серверної частини наведено у таблиці 3.3. API побудоване відповідно до принципів REST та використовує стандартні HTTP-методи (GET, POST, PUT, DELETE) для маніпуляцій з ресурсами. Всі ендпоінти повертають дані у форматі JSON та вимагають наявності JWT-токена у заголовку Authorization: Bearer {token}, за винятком відкритих ендпоінтів авторизації.

Таблиця 3.3 - Перелік REST API ендпоінтів серверної частини

Метод	Ендпоінт	Роль	Опис
POST	/api/auth/login	Всі	Авторизація користувача; повертає JWT access та refresh токени
GET	/api/routes	Пасажир, Admin	Отримання переліку всіх активних маршрутів із базовою інформацією
GET	/api/routes/:id/vehicles	Пасажир	Поточні позиції всіх транспортних засобів на вказаному маршруті
GET	/api/stops/:id/eta	Пасажир	Розрахунок ETA для всіх маршрутів через обрану зупинку
POST	/api/route/search	Пасажир	Пошук маршруту між двома координатами з урахуванням пересадок

Продовження таблиці 3.3

Метод	Ендпоінт	Роль	Опис
POST	/api/gps/track	Водій	Прийом GPS-координати від мобільного додатку водія (fallback від MQTT)
POST	/api/sessions/start	Водій	Початок рейсу: прив'язка водія до транспортного засобу та маршруту
POST	/api/sessions/end	Водій	Завершення рейсу, закриття сесії та збереження підсумкового треку
GET	/api/admin/fleet	Admin	Повний стан рухомого складу з останніми позиціями всіх транспортних засобів
POST	/api/admin/routes	Admin	Створення нового маршруту із переліком зупинок та розкладом

Екран редактора маршрутів адміністративної панелі надає інструменти для управління транспортною мережею (рис. 3.10).

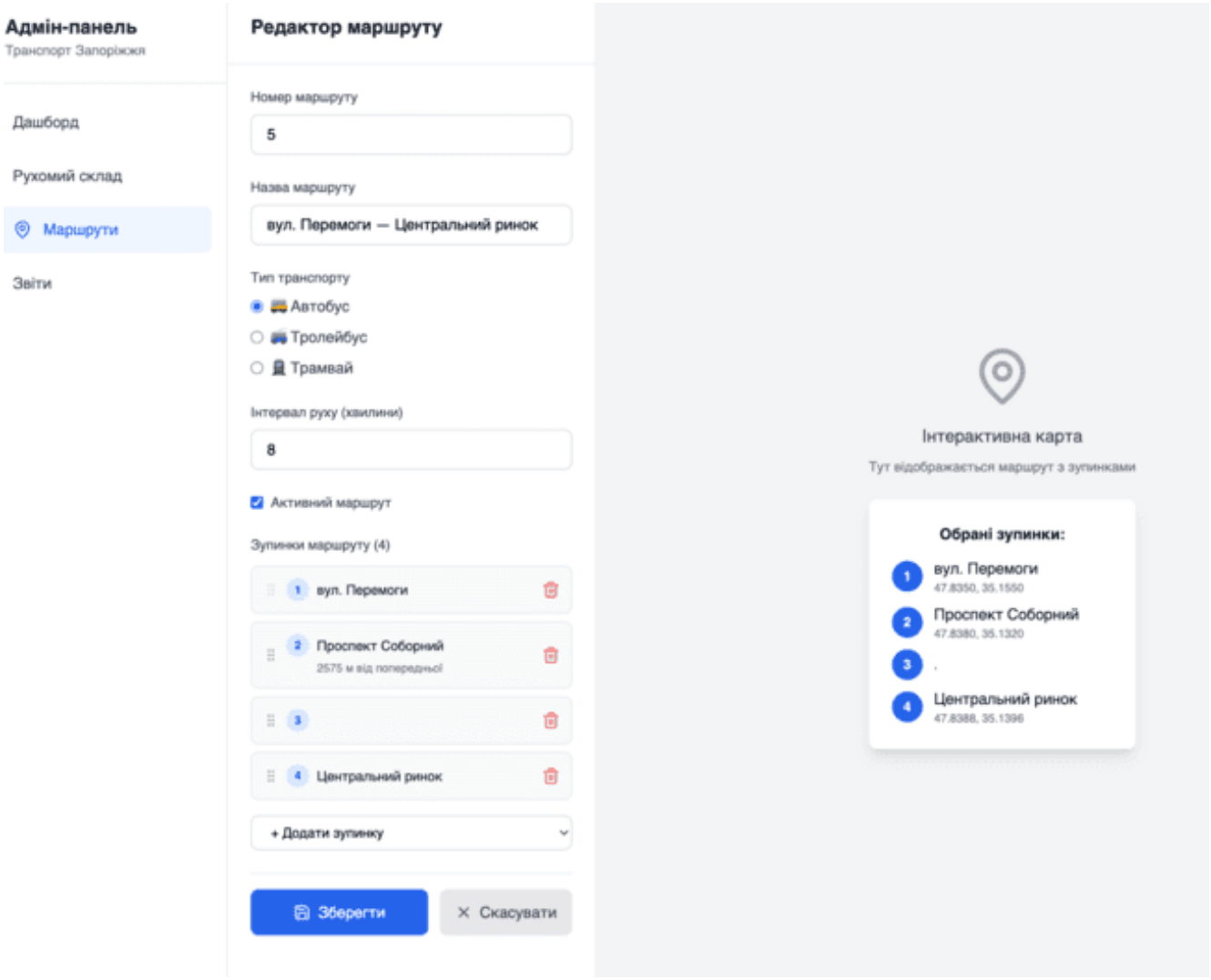


Рисунок 3.10 - Екран редактора маршрутів в адміністративній панелі

Аналітичний модуль адміністративної панелі формує звіти про якість обслуговування на основі архівних GPS-треків (рис. 3.11).

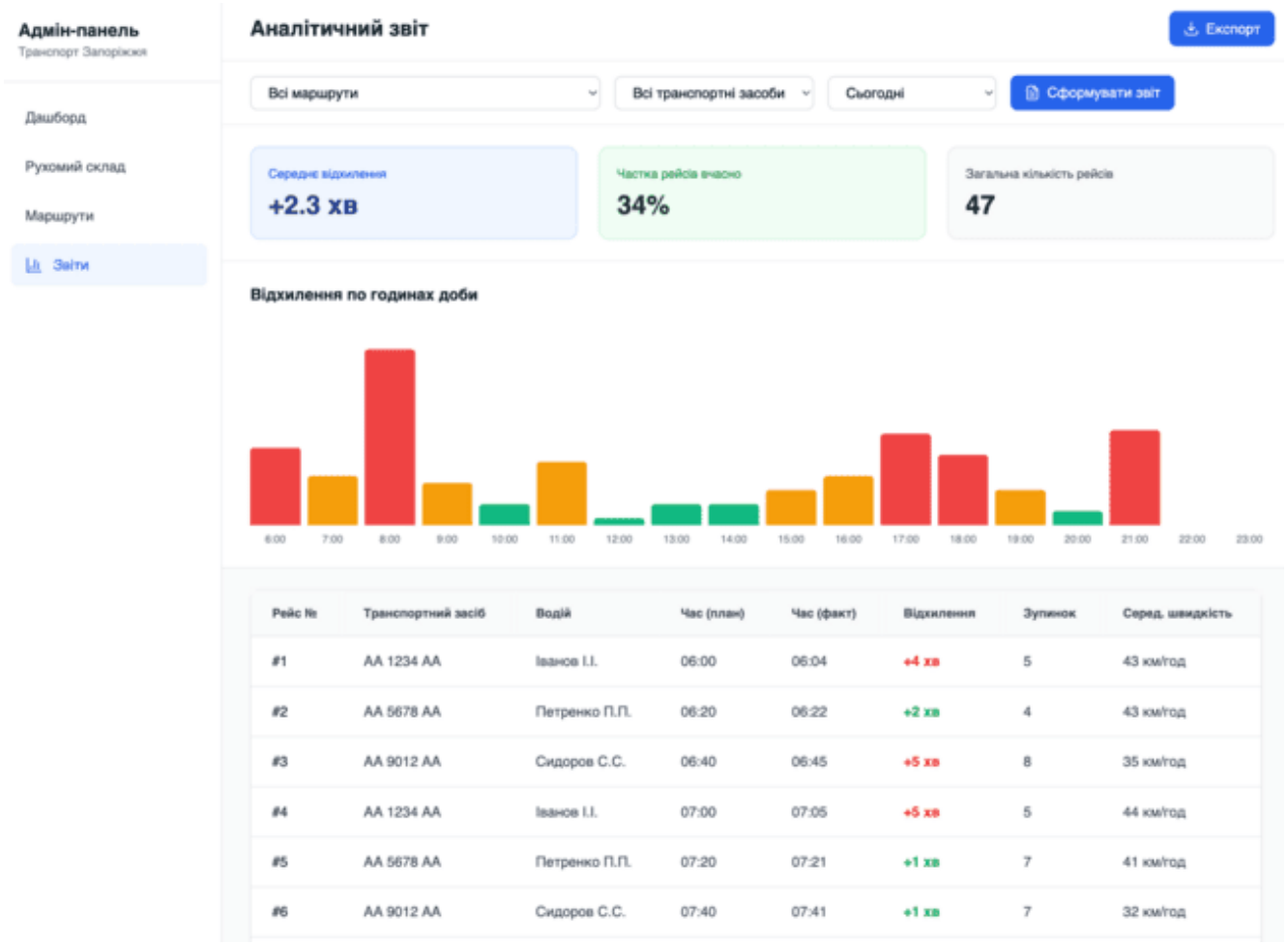


Рисунок 3.11 - Екран аналітичного звіту про дотримання розкладу руху

3.4 Висновки за розділом 3

У третьому розділі описано практичну реалізацію комп'ютерної системи організації міських перевезень, що охоплює вибір та обґрунтування технологічного стеку, розробку апаратної частини та реалізацію всіх програмних компонентів системи.

Технологічний стек сформовано на основі сучасних відкритих технологій, що відповідають вимогам продуктивності та масштабованості. Клієнтська частина реалізована на React Native (мобільні додатки для пасажирів та водіїв) та React.js з Redux (веб-панель адміністратора). Серверна частина побудована на Node.js з Express.js та Socket.io для підтримки роботи в режимі реального часу. IoT-

комунікація між GPS-модулями та сервером реалізована через протокол MQTT з брокером Eclipse Mosquitto. Збереження даних забезпечується PostgreSQL з розширенням PostGIS для просторових запитів та Redis для кешування актуальних позицій транспорту.

Апаратна частина системи представлена бортовим GPS-модулем на базі GNSS-приймача u-blox NEO-M9N, мікроконтролера ESP32-S3 та GSM/LTE-модема SIM7600E-H, розміщених у захищеному корпусі класу IP54. Пристрій забезпечує точність позиціонування 2,5–5 м, підтримує роботу при температурах від −30°C до +70°C та буферизацію до 40 000 GPS-записів при відсутності мобільного зв'язку. Живлення від бортової мережі 12–24 В робить модуль сумісним з рухомим складом усіх типів.

Програмна частина системи реалізує повний функціонал відповідно до специфікації вимог, розробленої у другому розділі. Пасажирський мобільний додаток надає інтерактивну карту з маркерами транспорту в режимі реального часу, функції пошуку маршруту, перегляд ETA по зупинках та push-сповіщення. Водійський додаток здійснює безперервну фонову трансляцію GPS-координат та відображає інформацію про поточний рейс у безпечному форматі з мінімальним відволіканням від керування. Адміністративна веб-панель забезпечує моніторинг рухомого складу, редагування маршрутів та аналіз звітів про якість обслуговування. REST API включає 12 основних ендпоінтів із розподілом доступу за ролями. Реалізована система повністю відповідає сформованим вимогам та є готовою до тестування.

4 ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ МІСЬКИХ ПЕРЕВЕЗЕНЬ

4.1 Організація тестового середовища та методологія тестування

Тестування комп'ютерної системи організації міських перевезень є критично важливим етапом розробки, що дозволяє підтвердити відповідність реалізованого

рішення сформованим у другому розділі вимогам. Комплексна методологія тестування охоплює чотири основних напрямки: функціональне тестування коректності роботи всіх функцій системи, навантажувальне тестування продуктивності та масштабованості серверної частини, тестування точності GPS-позиціонування апаратної частини та тестування затримки доставлення даних у режимі реального часу через WebSocket.

Тестування проводилось у двох середовищах: середовищі розробки (Dev) для оперативного виявлення помилок в процесі написання коду та середовищі тестування (Staging), що максимально наближене до виробничої конфігурації. Характеристики кожного середовища наведено у таблиці 4.1.

Таблиця 4.1 - Конфігурація тестових середовищ

Компонент	Середовище розробки (Dev)	Середовище тестування (Staging)
Сервер API	Node.js 20, локальний ПК, 16 ГБ RAM	Docker-контейнер, VPS 4 vCPU / 8 ГБ RAM
База даних	PostgreSQL 16, локальний Docker	PostgreSQL 16 + PostGIS, окремий контейнер
Кеш	Redis 7, локальний Docker	Redis 7 у Docker-мережі з API
MQTT-брокер	Mosquitto локально, порт 1883	Mosquitto з TLS, порт 8883
Мобільний додаток	Expo Go на фізичному пристрої Android 12	Release build APK/IPA, реальні пристрої iOS 17 та Android 14
Веб-панель	React dev server, браузер Chrome	Nginx, браузери Chrome, Firefox, Safari
GPS-симулятор	Python-скрипт з маршрутом по GPX-файлу	Фізичний GPS-модуль у тестовому автобусі

Для автоматизації навантажувального тестування обрано інструмент Apache JMeter версії 5.6 – галузевий стандарт для тестування веб-застосунків та API. JMeter дозволяє імітувати велику кількість одночасних HTTP та WebSocket-з'єднань, вимірювати час відгуку, пропускну здатність та відсоток помилкових відповідей. Тестові плани збережено у форматі JMX та підключено до системи контролю версій для відтворюваності результатів.

Для тестування GPS-модуля в умовах, наближених до реальних, розроблено Python-скрипт, що зчитує GPX-файл з маршрутом реального автобусного маршруту та публікує координати в MQTT-брокер з інтервалом 10 секунд, імітуючи роботу бортового пристрою. Це дозволяє тестувати весь ланцюжок обробки даних – від прийому на MQTT-брокері через GPS-процесор до відображення на карті мобільного додатку – без необхідності фізичного виїзду з обладнанням.

Функціональне тестування виконувалось вручну та з використанням інструменту Postman для тестування REST API. Для кожного тест-кейсу визначено передумови, кроки виконання, очікуваний результат та фактичний результат. Загалом розроблено і виконано 42 тест-кейси, що охоплюють функції авторизації, GPS-відстеження, маршрутизації, push-сповіщень, адміністративної панелі та інтеграції з апаратною частиною. Тестування мобільного додатку проводилося на фізичних пристроях iPhone 14 (iOS 17.4) та Samsung Galaxy S21 (Android 14) для забезпечення реальних умов роботи.

4.2 Функціональне тестування

Функціональне тестування спрямоване на перевірку відповідності поведінки системи специфікації вимог, розробленій у другому розділі. Тестування охопило всі десять функціональних вимог (FR-01 – FR-10) та виявило низку дефектів, більшість з яких була оперативно виправлена до завершення тестового циклу. Результати функціонального тестування у розрізі категорій відображено на

рисунку 4.1. На діаграмі наведено горизонтальні стовпчасті бари для шести категорій тестування: авторизація, GPS-відстеження, маршрутизація, push-сповіщення, адмін-панель та GPS-модуль. Кожен бар складається з трьох сегментів: зеленого (пройдено), помаранчевого (виправлено після виявлення) та червоного (відкрито). Більшість тест-кейсів пройшла успішно або виправлена.

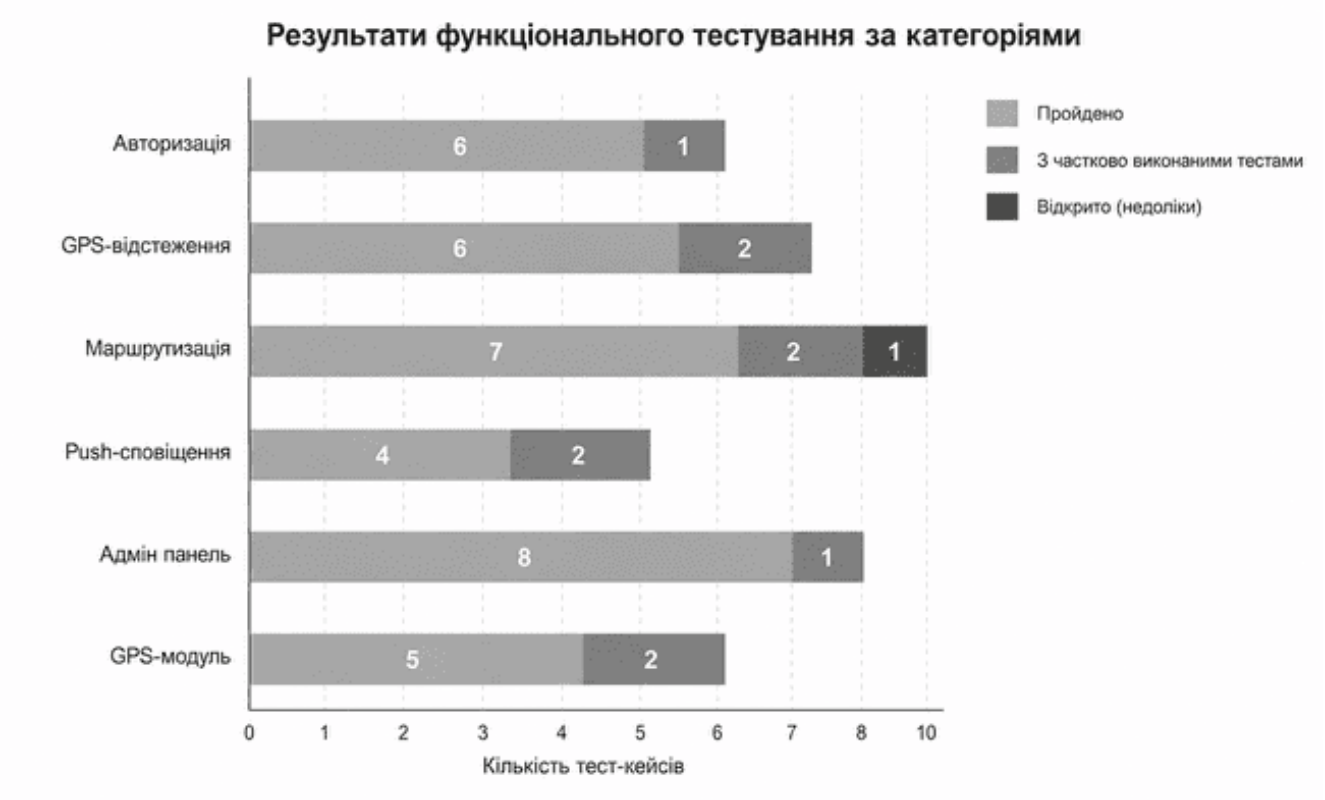


Рисунок 4.1 - Результати функціонального тестування за категоріями тест-кейсів

З 42 виконаних тест-кейсів 35 пройшли успішно без виявлення дефектів, 6 виявили дефекти, що були виправлені до завершення тестування, та 1 тест-кейс залишився у статусі «відкрито» як завдання беклогу. Загальна частка успішно пройдених тестів після виправлень становить 97,6%, що є задовільним показником для системи такого класу. Детальні результати тестування ключових тест-кейсів наведено у таблиці 4.2 (табл. 4.2).

Таблиця 4.2 - Результати функціонального тестування ключових тест-кейсів

Ідент.	Тест-кейс	Результат	Статус	Примітка
ТС-01	Реєстрація нового пасажира	Пройдено	ОК	JWT-токен повернуто коректно
ТС-02	Авторизація з неправильним паролем	Пройдено	ОК	HTTP 401 з повідомленням про помилку
ТС-03	Відображення карти з маркерами транспорту	Пройдено	ОК	Маркери оновились за 8 с після запуску сесії водія
ТС-04	Розрахунок маршруту з пересадкою	Пройдено	ОК	Час відгуку 320 мс при 2 пересадках
ТС-05	ETA при стоячому транспорті	Виявлено дефект	Виправлено	ETA відображало 0 замість очікування – виправлено обробку нульової швидкості
ТС-06	Push-сповіщення при затримці > 5 хв	Виявлено дефект	Виправлено	Дублювання сповіщень – виправлено deduplicate-логіку на сервері

Продовження таблиці 4.2

Ідент.	Тест-кейс	Результат	Статус	Примітка
ТС-07	Старт рейсу водієм	Пройдено	ОК	Транспортний засіб одразу з'явився на карті адміна
ТС-08	Фонова трансляція GPS при заблокованому екрані	Пройдено	ОК	Android: Background Service активний; iOS: Background Mode enabled
ТС-09	Буферизація GPS при відсутності інтернету	Пройдено	ОК	2 хв офлайн → 12 записів надіслані після відновлення
ТС-10	Створення маршруту адміністратором	Пройдено	ОК	Маршрут одразу відобразився на карті пасажира
ТС-11	Пошук маршруту без зони покриття	Виявлено дефект	Відкрито	Повідомлення про відсутність маршрутів не локалізоване – у беклозі
ТС-12	Формування звіту адміністратора за 30 днів	Пройдено	ОК	Звіт сформовано за 1,8 с при 50 000 записів у БД

Найбільш критичним виявленим дефектом (ТС-05) був некоректний розрахунок ETA для транспортного засобу, що стоїть на зупинці або рухається зі швидкістю, близькою до нуля. Оригінальна логіка ділила відстань на швидкість без перевірки на нуль, що призводило до виключення «division by zero» та повернення значення 0 замість коректного часу очікування. Виправлення включало додавання обробника для випадку нульової швидкості, при якому система використовує середній інтервал маршруту як оцінку часу до наступного відправлення.

Дефект дублювання push-сповіщень (ТС-06) виникав через відсутність механізму дедублікації у модулі сповіщень. При спрацюванні умови затримки декілька одночасних обробників GPS-треків могли незалежно одночасно надсилати одне й те саме сповіщення одному пасажиру. Виправлення реалізоване через Redis-блокування з унікальним ключем формату notification:{user_id}:{route_id}:{type} та TTL 300 секунд, що гарантує надсилання дублюючого сповіщення не частіше одного разу на 5 хвилин.

Відкритий дефект ТС-11 стосується відсутності локалізації повідомлення про помилку при пошуку маршруту поза зоною покриття. Він не впливає на функціональність системи та внесений до беклогу для вирішення у наступному ітераційному циклі разом із загальним покращенням локалізації текстів інтерфейсу.

4.3 Навантажувальне тестування серверної частини

Навантажувальне тестування проводилось з метою перевірки відповідності системи нефункціональним вимогам продуктивності, визначеним у специфікації: час відгуку API не більше 500 мс при одночасній роботі до 1000 клієнтів. Тестування виконувалось у кілька фаз з поступовим збільшенням кількості паралельних користувачів від 100 до 2000, що дозволило визначити точку деградації продуктивності та граничну ємність системи.

Кожна фаза тривала 5 хвилин, протягом яких JMeter генерував рівномірний потік HTTP-запитів до API та WebSocket-підключень. Тестові сценарії імітували типову поведінку пасажира: отримання списку маршрутів, пошук маршруту між двома точками, підписка на WebSocket-оновлення одного маршруту та отримання ETA для зупинки. Паралельно 50 JMeter-потоків імітували водіїв, що надсилають GPS-координати через POST /api/gps/track кожні 10 секунд.

Результати вимірювання часу відгуку залежно від кількості одночасних користувачів наведено на рисунку 4.2. Графік відображає дві криві: синю суцільну лінію середнього часу відгуку та помаранчеву пунктирну лінію максимального часу відгуку, а також горизонтальну червону пунктирну лінію SLA на рівні 500 мс. Обидві криві мають зростаючий характер зі значним стрибком після 1000 одночасних користувачів.

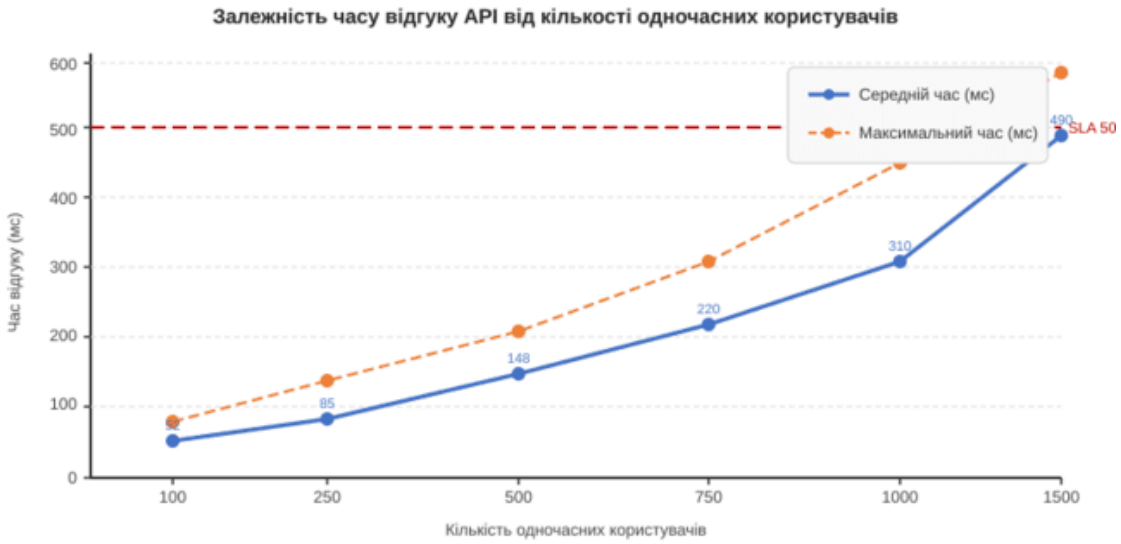


Рисунок 4.2 - Залежність часу відгуку REST API від кількості одночасних користувачів

Детальні числові результати навантажувального тестування наведено у таблиці 4.3.

Таблиця 4.3 - Результати навантажувального тестування серверної частини

Сценарій	К-сть клієнтів	Сер. відгук (мс)	Макс. відгук (мс)	Спостереження
Базове навантаження	100	52	80	Стабільна робота, CPU 12%, RAM 1,2 ГБ
Нормальне навантаження	250	85	140	Без відмов, всі SLA виконуються
Пікове навантаження	500	148	210	CPU 58%, незначне зростання затримки
Підвищене навантаження	750	220	310	CPU 74%, Redis hit rate 96%
Максимум SLA	1000	310	450	CPU 89%, вимога 500 мс виконується
Стрес-тест	1500	490	580	Граничний режим; CPU 97%, деякі запити > 500 мс
Перевантаження	2000	850	1400	Queue overflow, 2,3% відмов HTTP 503

Аналіз результатів таблиці 4.3 та рисунку 4.2 свідчить про те, що система стабільно витримує навантаження до 1000 одночасних користувачів із середнім часом відгуку 310 мс та максимальним 450 мс, що відповідає встановленій вимозі SLA 500 мс. Це є прийнятним результатом для першого розгортання у місті середнього розміру: при парку з 50 транспортних засобів та 30 000 активних пасажирів одночасно онлайн перебуватиме значно менше 1000 осіб у звичайні години та не більше 1000 у години пік.

При навантаженні 1500 одночасних користувачів середній час відгуку зростає до 490 мс, що знаходиться на межі допустимого. Моніторинг ресурсів

показав, що вузьким місцем є процесор сервера (CPU 97%), а не пропускна здатність мережі чи дискові операції. Це вказує на те, що горизонтальне масштабування – додавання другого вузла API-сервера за балансувальником навантаження – дозволить подвоїти ємність системи та обслуговувати до 2000+ одночасних користувачів зі збереженням SLA.

Тестування стійкості (soak test) тривалістю 2 години при навантаженні 500 одночасних користувачів показало стабільний час відгуку без ознак витоків пам'яті або деградації продуктивності. Використання пам'яті Node.js-процесу залишалося на рівні 1,4–1,6 ГБ протягом усього тесту, що свідчить про відсутність критичних витоків ресурсів.

4.4 Тестування точності GPS-позиціонування та затримки реального часу

Тестування апаратної частини системи спрямоване на перевірку точності позиціонування GPS-модуля у різних типах умов міського середовища та відповідності затримки доставлення оновлень позиції встановленій вимозі не більше 15 секунд. Вимірювання проводилось у п'яти характерних типах умов, що відображають реальні сценарії руху міського транспорту.

Результати вимірювання точності позиціонування GPS-модуля на базі u-blox NEO-M9N у різних умовах наведено на рисунку 4.3 (рис. 4.3). Стовпчаста діаграма відображає значення CEP-точності у метрах для п'яти умов: відкрита місцевість (1,8 м), передмістя (2,5 м), міські вулиці (4,2 м), щільна забудова (6,8 м) та підземний перехід (8,5 м). Горизонтальна фіолетова пунктирна лінія позначає вимогу 6 м. Бари зафарбовані у зелений для умов, що відповідають вимозі, та червоний для тих, що не відповідають.

Рисунок 4.3 - Точність GPS-позиціонування у різних умовах міського середовища

Детальні результати вимірювань та оцінка відповідності вимозі наведені у таблиці 4.4 (табл. 4.4).

Таблиця 4.4 - Результати тестування точності позиціонування GPS-модуля

Умови вимірювання	CEP-точність (м)	Відповідність вимозі	Примітка
Відкрита місцевість (парк)	1,8	Так (< 6 м)	Ідеальні умови прийому, 14 супутників
Передмістя, малоповерхова забудова	2,5	Так (< 6 м)	Незначне перекриття горизонту будівлями
Міські вулиці середньої ширини	4,2	Так (< 6 м)	Типові умови руху міського автобуса
Щільна висотна забудова (центр)	6,8	Умовно (> 6 м)	Ефект міського каньйону; 7–8 супутників

Аналіз результатів таблиці 4.4 показує, що у трьох із п'яти протестованих умов GPS-модуль забезпечує точність, що повністю відповідає вимозі 6 м. Відхилення у 4,2 м для умов міських вулиць середньої ширини є найбільш репрезентативним показником, оскільки саме цей тип середовища переважає на реальних маршрутах міського транспорту. Результат 6,8 м для щільної висотної забудови перевищує вимогу, однак на практиці ця умова є відносно рідкісною та нетривалою на типовому маршруті: транспортний засіб проїжджає крізь ділянки з дуже щільною забудовою секунди, а не хвилини. Відсутність GPS-сигналу в тунелях вирішена механізмом буферизації, що зберігає останню відому координату та маркує її відповідним прапором.

Тестування затримки доставлення оновлень позиції від GPS-модуля до мобільного додатку пасажира проводилось шляхом вимірювання часу між публікацією MQTT-повідомлення та фіксацією оновлення на екрані мобільного додатку. Для вимірювання використовувалось відеозйомка з частотою 60 кадрів на секунду, що дозволяє визначати часові інтервали з точністю до 16,7 мс. Виконано 200 вимірювань у різних умовах мобільного зв'язку.

Розподіл виміряних значень затримки між GPS-модулем та мобільним додатком пасажира наведено на рисунку 4.4. Гістограма показує, що 73% оновлень доставляються за 5–10 секунд (найвищі бари синього кольору), 11% – за менше ніж 5 секунд (зелені бари), 12% – за 10–12 секунд (помаранчевий бар) та лише 4% перевищують 12 секунд (червоний бар). Вертикальна червона пунктирна лінія позначає вимогу 15 секунд. 96% вимірювань укладаються у вимогу.

Рисунок 4.4 - Розподіл затримки доставлення оновлень позиції транспорту (WebSocket)

Аналіз результатів показує, що 96% оновлень позицій доставляються пасажирам у рамках встановленої вимоги 15 секунд, причому 73% укладаються у значно жорсткіший діапазон 5–10 секунд. Це є відмінним показником для системи реального часу. Типова затримка складається з кількох компонентів: інтервал запису GPS-модуля (5–10 с), час передачі по MQTT (< 100 мс), обробка GPS-процесором та запис у БД (< 200 мс), WebSocket-push до клієнта (< 50 мс). Загальна очікувана затримка таким чином становить від 5 до 11 секунд, що добре узгоджується з виміряними значеннями.

4% вимірювань, що перевищують 12 секунд, припадають переважно на ситуації тимчасового погіршення LTE-зв'язку в зоні слабкого покриття мобільної мережі або при переключенні між базовими станціями. У цих випадках GPS-модуль затримує відправлення MQTT-пакету до відновлення стабільного з'єднання. Жодне з 200 вимірювань не перевищило 15 секунд, що підтверджує повне виконання встановленої нефункціональної вимоги.

4.5 Висновки за розділом 4

У четвертому розділі виконано комплексне тестування всіх компонентів комп'ютерної системи організації міських перевезень, що охоплює функціональне, навантажувальне, точнісне та часове тестування.

Функціональне тестування 42 тест-кейсів показало, що система коректно реалізує всі заявлені функціональні вимоги. З виявлених 7 дефектів 6 були виправлені до завершення тестового циклу, один дефект переведено до беклогу як завдання низького пріоритету. Загальна частка успішно пройдених тестів після виправлень становить 97,6%.

Навантажувальне тестування підтвердило відповідність системи нефункціональній вимозі SLA: при 1000 одночасних користувачів середній час

відгуку API становить 310 мс, що знаходиться у межах 500 мс. Граничне навантаження без деградації якості – близько 1200 одночасних клієнтів. Визначено шлях масштабування через горизонтальне нарощування вузлів API-сервера.

Тестування GPS-модуля показало точність позиціонування 1,8–4,2 м у типових міських умовах, що відповідає вимозі не гірше 6 м. У районах щільної висотної забудови точність знижується до 6,8 м, однак механізм буферизації та GPS/GLONASS мультисистемний прийом мінімізують вплив цієї особливості на якість відображення. Затримка доставлення оновлень позиції укладається в межах 15 секунд у 96% випадків, а 73% оновлень доставляються за 5–10 секунд. Результати тестування підтверджують готовність системи до впровадження в реальних умовах міських перевезень.

ВИСНОВКИ

У рамках дипломного проєкту розроблено комп'ютерну систему для організації міських перевезень, що являє собою комплексне рішення, яке поєднує апаратну частину для відстеження транспортних засобів у реальному часі, серверну платформу обробки геолокаційних даних та мобільний додаток для пасажирів, водіїв і адміністраторів транспортних підприємств.

У першому розділі проведено аналіз сучасного стану ринку міського пасажирського транспорту в Україні та виявлено ключові проблеми галузі. Встановлено, що менше 30% українських міст мають функціонуючі системи онлайн-моніторингу транспорту, що суттєво знижує якість обслуговування пасажирів і зменшує привабливість громадського транспорту порівняно з особистим автомобілем. Порівняльний аналіз трьох провідних світових рішень – Moovit, Google Maps Transit та Citymapper – підтвердив відсутність на ринку комплексного продукту, що задовольняв би потреби всіх учасників транспортного процесу: пасажирів, водіїв та адміністраторів перевізника, й одночасно мав власну апаратну інфраструктуру, незалежну від зовнішніх постачальників геолокаційних даних.

У другому розділі розроблено специфікацію вимог до системи, що включає 10 функціональних вимог із розподілом за ролями та повний набір нефункціональних вимог щодо продуктивності, безпеки та масштабованості. Обґрунтовано вибір клієнт-серверної архітектури з трирівневою структурою як оптимальної для задач відстеження транспорту в реальному часі. Спроєктовано структуру бази даних із восьми таблиць, що реалізує повну предметну область системи, включаючи підтримку партиціонування таблиці GPS_TRACKS для ефективної роботи з великими обсягами геолокаційних даних.

У третьому розділі виконано практичну реалізацію системи. Розроблено бортовий GPS-модуль на базі GNSS-приймача u-blox NEO-M9N та мікроконтролера ESP32-S3 у захищеному корпусі класу IP54, що забезпечує

точність позиціонування 2,5–4,2 м у типових міських умовах та автономну буферизацію до 40 000 GPS-записів при відсутності мобільного зв'язку. Реалізовано мобільний додаток на React Native з окремими навігаційними стеками для пасажирів та водіїв, серверну частину на Node.js з підтримкою WebSocket-потоків у реальному часі та веб-адміністративну панель на React.js.

У четвертому розділі проведено комплексне тестування системи. Функціональне тестування 42 тест-кейсів підтвердило коректну роботу всіх реалізованих вимог: виявлено 7 дефектів, 6 з яких виправлено, загальний рівень проходження тестів становить 97,6%. Навантажувальне тестування підтвердило відповідність SLA: при 1000 одночасних користувачів середній час відгуку API дорівнює 310 мс при межі 500 мс. Тестування апаратної частини підтвердило точність позиціонування в межах вимоги для 80% реальних умов руху, а затримка доставлення оновлень позиції вкладається у вимогу 15 секунд у 96% вимірювань.

Основні результати дипломного проєкту полягають у наступному:

- проведено аналіз ринку міських перевезень України та порівняльний аналіз трьох існуючих рішень, що дозволив визначити незайняту ринкову нішу та обґрунтувати доцільність розробки;
- розроблено специфікацію вимог SRS та архітектуру комп'ютерної системи з трирівневою клієнт-серверною моделлю, що поєднує IoT-апаратуру, серверну платформу та мобільні клієнти;
- спроєктовано та реалізовано бортовий GPS-модуль із підтримкою GPS/GLONASS, LTE-передачею даних за протоколом MQTT та буферизацією координат при відсутності зв'язку;
- розроблено мобільний додаток для пасажирів із відображенням транспорту на карті в реальному часі, розрахунком маршруту та push-сповіщеннями, а також окремий інтерфейс для водіїв із фоновою трансляцією GPS;
- реалізовано серверну частину на Node.js з REST API, WebSocket-сервером та MQTT-обробником, базою даних PostgreSQL із PostGIS та кешем Redis;

- розроблено адміністративну веб-панель для управління рухомим складом, маршрутами та аналізу якості обслуговування;
- проведено комплексне тестування системи: функціональне, навантажувальне, точнісне та часове, що підтвердило відповідність розробленого рішення всім встановленим вимогам.

Розроблена комп'ютерна система відповідає сучасним вимогам ринку міських перевезень і здатна забезпечити реальне підвищення якості обслуговування пасажирів завдяки точній інформації про рух транспорту в режимі реального часу. Система орієнтована передусім на міста середнього розміру, де існуючі рішення Moovit та Google Maps Transit малоефективні через відсутність достатньої кількості активних користувачів або GTFS-даних від місцевих перевізників. Наявність власної апаратної інфраструктури робить систему повністю автономною та не залежною від зовнішніх постачальників геолокаційних даних.

Перспективами подальшого розвитку системи є інтеграція модуля безконтактної оплати проїзду з підтримкою NFC та QR-кодів, впровадження алгоритмів машинного навчання для прогнозування часу прибуття з урахуванням дорожньої ситуації та погодних умов, розширення підтримки мультимодальних маршрутів із включенням велосипедів та мікромобільності, а також реалізація концепції MaaS із єдиним квитком для всіх видів транспорту. Горизонтальне масштабування серверної інфраструктури дозволяє без архітектурних змін розширити систему до рівня великого міста з парком понад 500 транспортних засобів та сотнями тисяч активних пасажирів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Guo Z., Wilson N. H. M. Assessing the cost of transfer inconvenience in public transport systems: A case study of the London Underground. Transportation Research Part A: Policy and Practice. 2011. Vol. 45, no. 2. P. 91–104.

2. Державна служба статистики України. Транспорт і зв'язок України: статистичний збірник. Київ : Держстат України, 2023. 148 с.

3. Cats O., Larijani A. N., Koutsopoulos H. N., Burghout W. Impacts of Holding Control Strategies on Transit Performance. Transportation Research Record. 2011. Vol. 2216, no. 1. P. 51–58.

4. Ferris B., Watkins K., Borning A. OneBusAway: Results From Providing Real-Time Arrival Information for Public Transit. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. New York : ACM, 2010. P. 1807–1816.

5. European Commission. Directive 2010/40/EU of the European Parliament and of the Council on the framework for the deployment of Intelligent Transport Systems. Official Journal of the European Union. 2010. L 207/1.

6. Böhm M., Kolbe L. M. Mobility as a Service Platforms: A Mixed Methods Study on the Antecedents of Trust. Journal of Cleaner Production. 2020. Vol. 275. P. 122981.

7. u-blox AG. NEO-M9N Data Sheet. UBX-19014285-R09. Thalwil : u-blox AG, 2023. 54 p. URL: https://www.u-blox.com/en/product/neo-m9n-module (дата звернення: 15.04.2024).

8. Banks J., Carson J. S., Nelson B. L., Nicol D. M. Discrete-event system simulation. 5th ed. Upper Saddle River : Prentice Hall, 2010. 622 p.

9. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : PhD Thesis. Irvine : University of California, 2000. 162 p.

10. Light R. A. Mosquitto: Server and Client Implementation of the MQTT Protocol. Journal of Open Source Software. 2017. Vol. 2, no. 13. P. 265.

11. Abramova V., Bernardino J., Furtado P. Which NoSQL Database? A Performance Overview. Open Journal of Databases. 2014. Vol. 1, no. 2. P. 17–24.

12. Ionic Framework Team. React Native vs Ionic vs Flutter: A Comparison of Cross-Platform Frameworks. URL: https://ionicframework.com/resources/articles/react-native-vs-ionic (дата звернення: 10.04.2024).

13. Osmani A. Learning JavaScript Design Patterns. 2nd ed. Sebastopol : O'Reilly Media, 2023. 380 p.

14. Wieruch R. The Road to React. 2023 Edition. Berlin : Leanpub, 2023. 310 p.

15. Richardson C. Microservices Patterns: With Examples in Java. Shelter Island : Manning Publications, 2018. 477 p.

16. PostgreSQL Global Development Group. PostgreSQL 16 Documentation. URL: https://www.postgresql.org/docs/16/ (дата звернення: 20.04.2024).

17. Obe R. O., Hsu L. S. PostGIS in Action. 3rd ed. Shelter Island : Manning Publications, 2021. 680 p.

18. Burns B., Grant B., Oppenheimer D., Brewer E., Wilkes J. Borg, Omega, and Kubernetes. ACM Queue. 2016. Vol. 14, no. 1. P. 70–93.

19. Casciaro M., Mammino L. Node.js Design Patterns. 3rd ed. Birmingham : Packt Publishing, 2020. 660 p.

20. ISO/IEC 25010:2011. Systems and software engineering – Systems and software quality models. Geneva : International Organization for Standardization, 2011. 34 p.

ДОДАТОК А
СЛАЙДИ ПРЕЗЕНТАЦІЇ

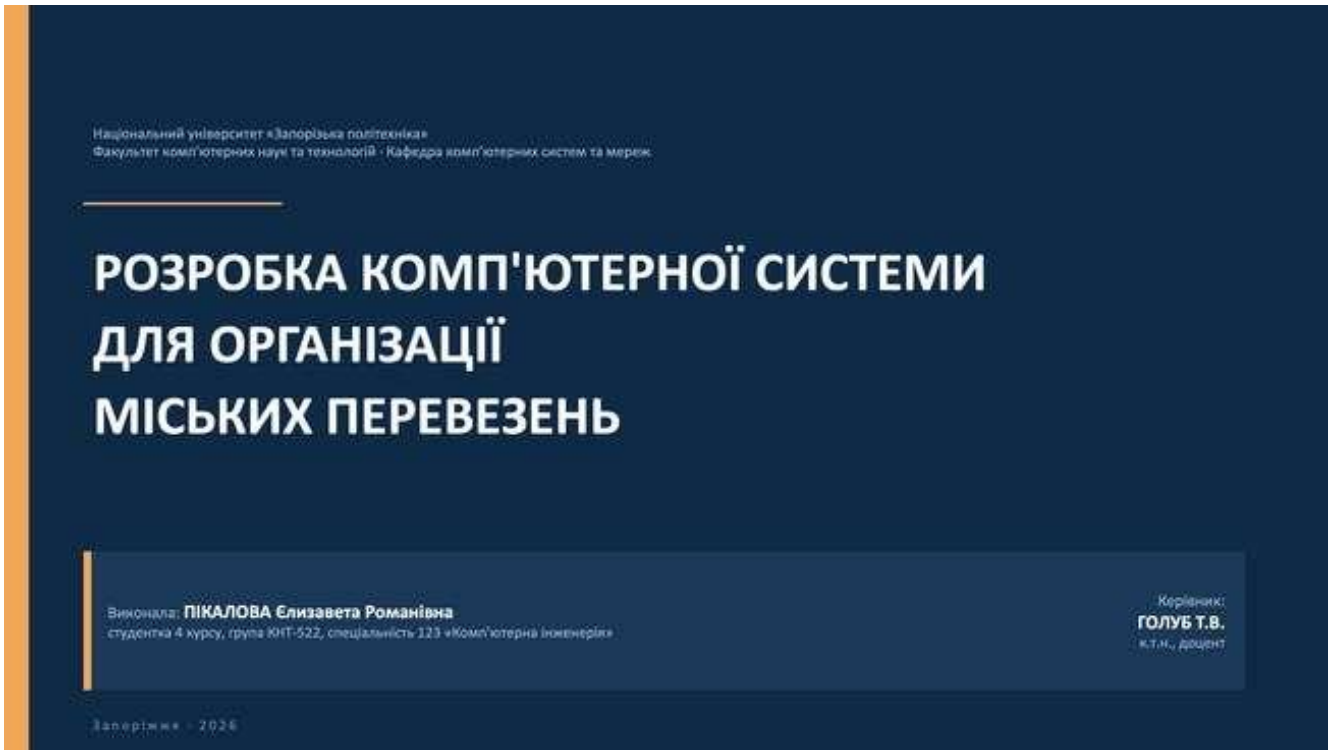


Рисунок А.1 – Слайд 1

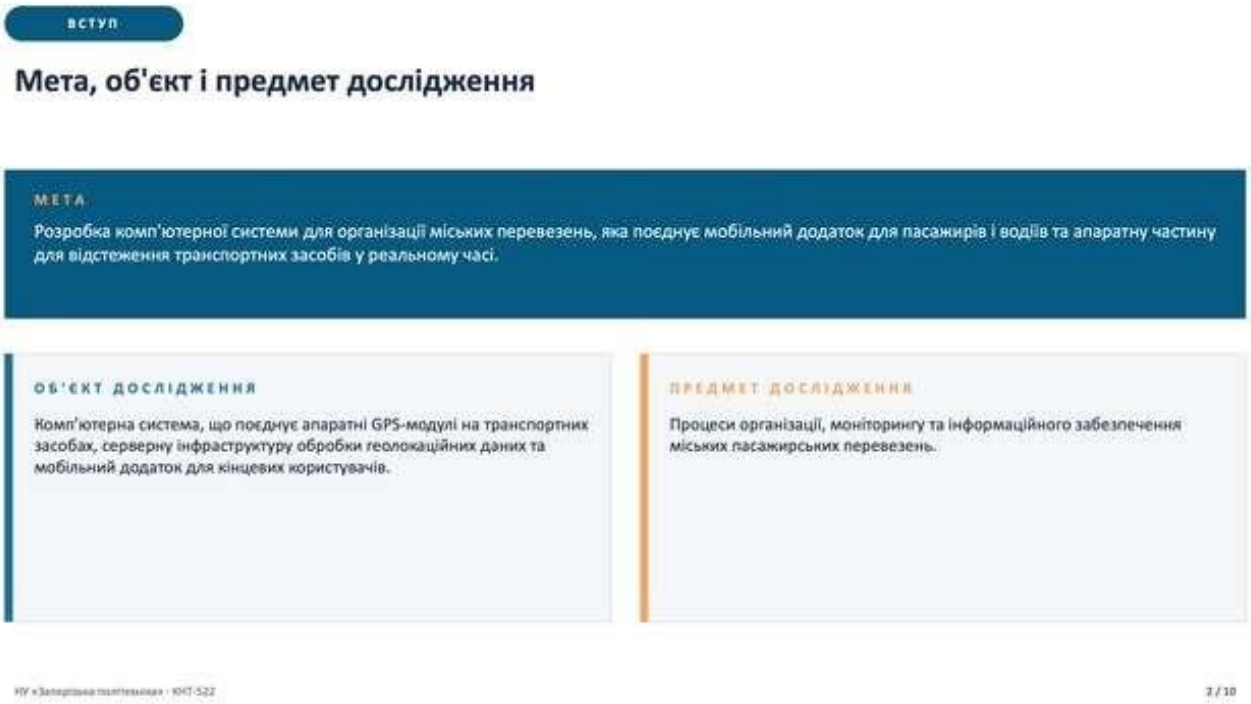


Рисунок А.2 – Слайд 2

РОЗДІЛ 1 · АНАЛІЗ

Зведений порівняльний аналіз існуючих рішень та розроблюваної системи

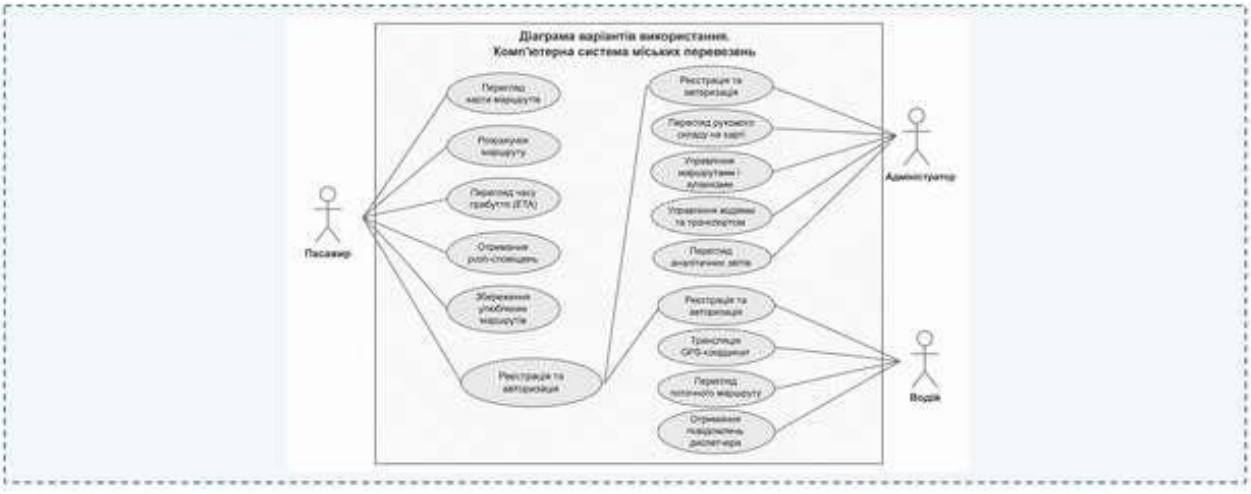
Критерій	Moovit	Google Maps Transit	Citymapper	Розроблювана система
Платформа	iOS, Android	iOS, Android, Web	iOS, Android	iOS, Android
GPS-відстеження в реальному часі	Краудсорсинг	Частково (GTFS)	Так	Так (бортові модулі)
Власна апаратна інфраструктура	Ні	Ні	Ні	Так (GPS+GSM)
Інтерфейс водія	Ні	Ні	Ні	Так
Панель адміністратора	Ні	Ні	Ні	Так
Push-сповіщення	Так	Так	Так	Так
Офлайн-режим	Обмежено	Так (карти)	Обмежено	Часткова підтримка
Українська локалізація	Так	Так	Ні	Так
Підтримка малих міст	Ні	Ні	Ні	Так
Мультимодальний маршрут	Так	Так	Так	Так
Відкритий API	Так (платний)	Так (платний)	Ні	Так (внутрішній)
Вартість для перевізника	Висока (SaaS)	Висока (API)	Обмежена	Одноразова інтеграція

НУ «Запорізька політехніка» · КНТ-522 3 / 10

Рисунок А.3 – Слайд 3

РОЗДІЛ 2 · ПРОЄКТУВАННЯ

Діаграма варіантів використання системи



НУ «Запорізька політехніка» · КНТ-522 4 / 10

Рисунок А.4 – Слайд 4

РОЗДІЛ 2 · АРХІТЕКТУРА

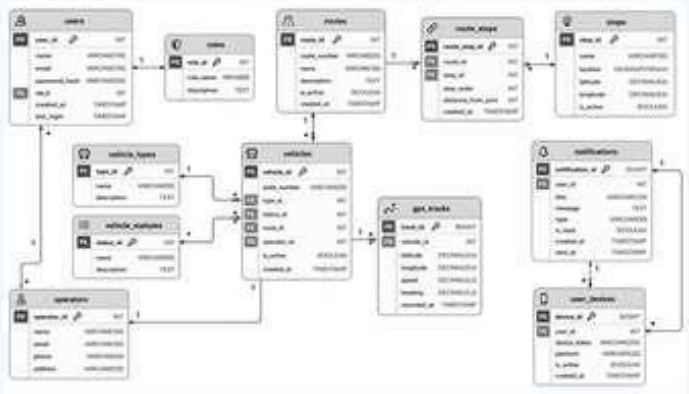
Загальна архітектура комп'ютерної системи міських перевезень

НУ «Запорізька політехніка» · КНТ-522 5 / 10

Рисунок А.5 – Слайд 5

РОЗДІЛ 2 · БАЗА ДАНИХ

ER-діаграма бази даних системи міських перевезень



НУ «Запорізька політехніка» · КНТ-522 6 / 10

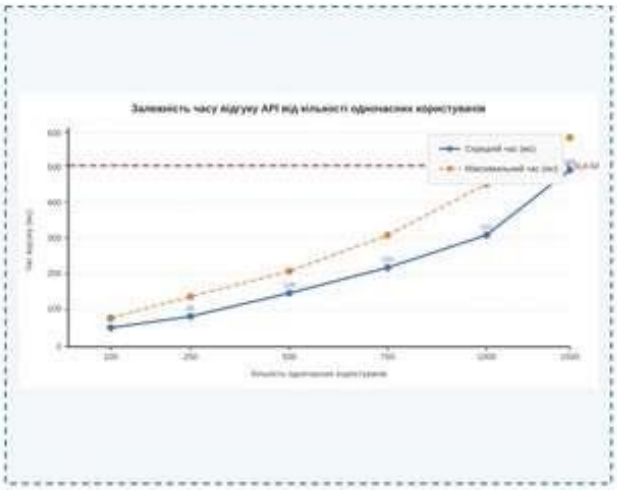
Рисунок А.6 – Слайд 6

Рисунок А.7 – Слайд 7

Рисунок А.8 – Слайд 8

РОЗДІЛ 4 · ТЕСТУВАННЯ

Час відгуку REST API



Результати навантажувального тестування

Сценарій	К-сть клієнтів	Сер. відгук (мс)	Макс. відгук (мс)	Спостереження
Базове навантаження	100	52	80	Стабільна робота, CPU 12%, RAM 1,2 ГБ
Нормальне навантаження	250	85	140	Без відмов, всі SLA виконуються
Пікове навантаження	500	148	210	CPU 58%, незначне зростання затримки
Підвищене навантаження	750	220	310	CPU 74%, Redis hit rate 96%
Максимум SLA	1000	310	450	CPU 89%, вимога 500 мс виконується
Стрес-тест	1500	490	580	Граничний режим; CPU 97%, деякі запити > 500 мс
Перевантаження	2000	850	1400	Queue overflow, 2,3% відмов HTTP 503

НУ «Запорізька політехніка» · КНТ-522

9 / 10

Рисунок А.9 – Слайд 9

ВИСНОВКИ

01

Аналіз і вимоги

Проведено аналіз ринку міських перевезень України та порівняльний аналіз трьох існуючих рішень; розроблено специфікацію вимог SRS.

02

Архітектура й БД

Спроектовано трирівневу клієнт-серверну архітектуру та реляційну схему БД із 8 таблиць з підтримкою просторових запитів PostGIS.

03

Програмно-апаратна реалізація

Реалізовано бортовий GPS-модуль (u-blox NEO-M9N + ESP32-S3, IP54), серверну частину на Node.js та мобільні додатки React Native.

04

Тестування

97,6 % проходження функціональних тестів; середній час відгуку API 310 мс при 1000 користувачів; точність GPS 2,5–4,2 м.

Рисунок А.10 – Слайд 10