**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Факультет інформаційних технологій**

| | |
|---|---|
| **ПОГОДЖЕНО** | **ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** |
| **Декан факультету** | **Завідувач кафедри комп'ютерних наук** |
| _____ **Ігор БОЛБОТ** | _____ **Белла ГОЛУБ** |
| (підпис) | (підпис) |
| «___» _____ 2026 р. | «___» _____ 2026 р. |

# БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

**на тему: «Система формування оптимального маршруту пересування транспортних засобів»**

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

| | | |
|---|---|---|
| **Гарант освітньої програми**<br>д.е.н., професор | _____<br>(підпис) | **Роман РУДЕНСЬКИЙ** |
| **Керівник бакалаврської кваліфікаційної роботи**<br>к.т.н., доцент | _____<br>(підпис) | **Віталій СВАТКО** |
| **Виконав** | _____<br>(підпис) | **Антон ВОЛОШИН** |

**Київ-2026**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Факультет інформаційних технологій**

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри комп'ютерних наук**

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2026 р.

**ЗАВДАННЯ**
**ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ**
**ЗДОБУВАЧУ**

**Волошину Антону Миколайовичу**
(прізвище, ім'я, по батькові)

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи

«Система формування оптимального маршруту пересування транспортних засобів»

затверджена наказом від «06» січня 2026 р. № 46 «С»
Термін подання завершеної роботи на кафедру «___» _____ 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): Python 3.11, VS Code, OpenStreetMap, бібліотеки — osmnx, networkx, folium, geopy, numpy, pandas, PyQt5, алгоритми — Dijkstra, A*, Bidirectional Search, формат даних — GeoJSON/XML
Перелік питань, що підлягають дослідженню:
1. Системний аналіз предметної області
2. Проектування інформаційного та програмного забезпечення
3. Проектування та реалізація програмної системи
4. Тестування та оцінювання ефективності системи
Перелік графічних документів (за необхідності).

Дата видачі завдання «___» _____ 2026 р.

**Керівник бакалаврської кваліфікаційної роботи** _____ (підпис) **Віталій СВАТКО**

**Завдання взяв до виконання** _____ (підпис) **Антон ВОЛОШИН**

# Зміст

# ВСТУП

Розробка прикладної інформаційної системи знаходження оптимального маршруту є актуальною в сучасних умовах розвитку транспортної галузі та агропромислового комплексу України. Ефективна логістика відіграє ключову роль у забезпеченні своєчасного переміщення товарів, сільськогосподарської техніки та вантажів, особливо з урахуванням різноманітності дорожніх умов – від автомагістралей до ґрунтових ділянок.

Традиційні навігаційні системи часто пропонують лише один варіант маршруту, не враховуючи комплексно час проїзду, рівень безпеки та фізичну відстань. Це призводить до неефективних рішень у динамічному середовищі, де необхідний баланс між швидкістю, безпекою та економічністю.

Консенсусна маршрутизація, яка поєднує результати кількох незалежних алгоритмів, дозволяє підвищити надійність рекомендацій і надати користувачеві обґрунтований вибір, що особливо важливо для логістів транспортних компаній та диспетчерів агропідприємств.

Метою бакалаврської кваліфікаційної роботи є розробка прикладної інформаційної системи Pathfinder для пошуку оптимального маршруту із застосуванням консенсусного підходу до маршрутизації.

Об'єктом дослідження є процеси визначення та оптимізації маршрутів у транспортних або навігаційних системах.

Предметом дослідження є методи, алгоритми та програмні засоби пошуку оптимального маршруту із застосуванням консенсусного підходу до прийняття маршрутних рішень в інформаційних системах.

Для досягнення поставленої методи необхідно вирішити наступні завдання:

1. Проаналізувати предметну область пошуку та оптимізації маршрутів, а також сучасні методи маршрутизації в інформаційних системах.
2. Дослідити існуючі програмні рішення та алгоритми визначення оптимальних маршрутів, визначити їх переваги та обмеження.

3. Обґрунтувати використання консенсусного підходу для прийняття маршрутних рішень та сформувати вимоги до інформаційної системи Pathfinder.
4. Розробити модель та алгоритм пошуку оптимального маршруту із застосуванням консенсусного підходу.
5. Спроєктувати архітектуру та структуру прикладної інформаційної системи Pathfinder, включаючи логіку взаємодії компонентів.
6. Реалізувати програмне забезпечення інформаційної системи Pathfinder для визначення оптимального маршруту.
7. Провести тестування та оцінювання ефективності розробленої системи шляхом аналізу результатів її роботи.

Система забезпечує розрахунок шляхів на основі локального графу дорожньої мережі Києва з урахуванням трьох незалежних алгоритмів, аналіз їхньої згоди або розбіжностей, генерацію покрокової легенди та формування звітів. Вона покликана оптимізувати логістичні процеси в транспортній та аграрній сферах, забезпечуючи швидкий, безпечний та обґрунтований вибір коридору руху.

Додаток підтримує роботу в офлайн-режимі, кросплатформеність та інтуїтивний інтерфейс, що підвищує його практичну цінність для користувачів.

Для реалізації проєкту використано сучасний стек технологій, що забезпечує надійність, продуктивність та кросплатформеність. Основною мовою програмування обрано Python 3.11+. Для створення користувацького інтерфейсу застосовано фреймворк Kivy у поєднанні з KivyMD, що дозволило реалізувати єдину кодову базу для Windows, Linux та Android. Моделювання та обробка дорожньої мережі виконані за допомогою бібліотек NetworkX та OSMnx.

Геокодування адрес здійснюється через Geopy з сервісом Nominatim, доповненим механізмом rate-limiting та кешуванням у SQLite. Для візуалізації карт використано Folium, а для генерації звітів – ReportLab. Локальне зберігання даних забезпечує SQLite.

Такий вибір інструментів дозволив реалізувати паралельне виконання алгоритмів A*, Dijkstra з урахуванням безпеки та Bidirectional Dijkstra, а також консенсусний аналіз результатів.

На момент написання записки апробація програмної системи на конференціях чи у вигляді публікацій не проводилася, оскільки проєкт перебуває на стадії розробки та внутрішнього тестування. Однак система була протестована в лабораторних умовах, що підтвердило її працездатність, відповідність функціональним вимогам та коректну роботу алгоритмів маршрутизації.

Пояснювальна записка складається з 71 сторінок, містить 40 використаних джерел, 14 таблиць, 21 рисунок та 1 додаток.

# 1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Опис транспортної та аграрної предметної області

Транспортна та аграрна предметні області становлять невід'ємну основу сучасної економіки України, забезпечуючи ефективне переміщення людей, товарів і виробничих ресурсів у складних географічних та соціально-економічних умовах. Автомобільний транспорт як домінуючий вид перевезень функціонує в межах розгалуженої мережі доріг, яка поєднує магістральні артерії з локальними шляхами, що відрізняються за класом покриття, пропускною здатністю та рівнем безпеки.

У цій мережі ключовими параметрами оптимізації маршрутів виступають час проїзду, фізична відстань і коефіцієнт безпеки, оскільки кожне рішення про вибір коридору безпосередньо впливає на економічну ефективність перевезень, витрати пального та ризики для учасників руху [1].

Особливого значення набуває врахування різноманітності дорожніх умов: від високошвидкісних автомагістралей до ґрунтових ділянок, які часто стають критичними ланками логістики. У транспортній предметній області оптимізація маршрутів перетворюється на мультикритеріальну задачу, де необхідно балансувати між швидкістю, довжиною шляху та рівнем безпеки, особливо в умовах годин пік або сезонних змін трафіку. Це дозволяє зменшувати загальний час доставки та підвищувати надійність перевезень, що є запорукою стабільності всієї господарської системи країни. Класифікація основних типів автомобільних доріг наведена в таблиці 1.1.

Таблиця 1.1

Класифікація основних типів автомобільних доріг за швидкістю руху та коефіцієнтом безпеки

| **Тип дороги** | **Середня швидкість, км/год** | **Коефіцієнт безпеки (0–1)** | **Характер використання** |
|---|---|---|---|
| **1** | **2** | **3** | **4** |
| Автомагістрал | 110 | 0,70 | Магістральні перевезення |

| | | | |
|---|---|---|---|
| ь (motorway) | | | на великі відстані |

Продовження таблиці 1.1

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Швидкісна дорога (trunk) | 90 | 0,75 | Основні міжрегіональні коридори |
| Головна дорога (primary) | 70 | 0,80 | Регіональні сполучення |
| Другорядна дорога (secondary) | 60 | 0,85 | Локальні сполучення між населеними пунктами |
| Житлова вулиця (residential) | 30 | 0,95 | Внутрішньоміські та сільські зони |
| Ґрунтова дорога (track) | 25 | 0,50 | Сільськогосподарські та лісові ділянки |

Структурна схема взаємозв'язку транспортної та аграрної предметних областей у контексті оптимізації маршрутів представлена на рис. 1.1.

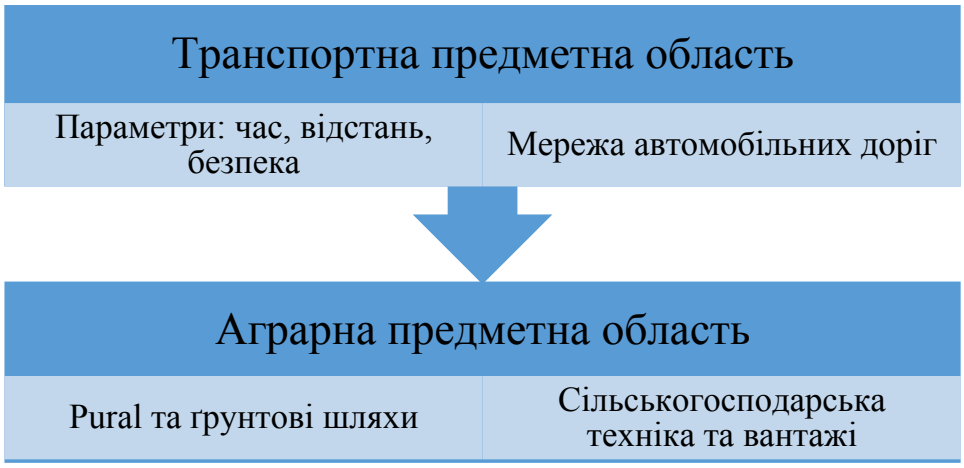


Рис. 1.1 Структурна схема взаємозв'язку транспортної та аграрної предметних областей у контексті оптимізації маршрутів

Аграрна предметна область, у свою чергу, вирізняється власними особливостями, зумовленими специфікою сільськогосподарського виробництва. Тут переміщення пов'язане переважно з важкою технікою – тракторами, комбайнами, зерновозами, – яка потребує не лише швидкості, а й високої

прохідності та мінімального ризику пошкодження вантажу чи обладнання. Сільські дороги, ґрунтові ділянки та польові шляхи часто стають основними артеріями для транспортування врожаю, добрив і техніки, особливо в регіонах з розвиненим рослинництвом і тваринництвом. У цій сфері оптимізація маршрутів повинна враховувати сезонні фактори – дощовий період, сніговий покрив чи посуху, – а також обмеження вантажопідйомності мостів і вузьких ділянок, що суттєво впливає на вибір коридору [2].

Поєднання транспортної та аграрної предметних областей створює унікальні виклики для логістики агропродукції, де маршрути часто проходять через змішані зони – від міських під'їзних шляхів до віддалених полів. Ефективне рішення таких завдань дозволяє зменшувати втрати врожаю під час перевезення, оптимізувати витрати пального та підвищувати загальну продуктивність аграрного сектору. Важливо підкреслити, що в обох областях домінує багатоваріантність можливих шляхів, коли один і той самий пункт призначення може бути досягнутий кількома коридорами, кожен з яких має свої переваги та недоліки за критеріями часу, безпеки та економічності [3].

Таким чином, опис транспортної та аграрної предметних областей демонструє їхню тісну взаємопов'язаність і необхідність комплексного підходу до оптимізації маршрутів. Урахування різноманітності дорожніх умов, специфіки техніки та зовнішніх факторів забезпечує не лише швидкість доставки, а й безпеку та сталість економічних процесів у країні. Саме такий системний аналіз закладає теоретичну основу для подальшого розвитку методів маршрутизації в реальних умовах сучасної України [4].

## 1.2 Аналіз сучасних систем побудови маршрутів та розпізнавання обєктів

Сучасні системи побудови маршрутів та розпізнавання об'єктів становлять невід'ємну складову інтелектуальних транспортних систем, де поєднуються досягнення графової теорії, геоінформатики та штучного інтелекту для забезпечення точного й адаптивного планування переміщень у складних умовах

дорожньої мережі. Ці системи еволюціонували від простих розрахунків найкоротшого шляху до комплексних рішень, що враховують динамічні фактори, такі як реальний час руху, погодні умови та інфраструктурні зміни, завдяки чому підвищується не лише ефективність логістики, але й загальна безпека учасників дорожнього руху.

Як видно з Рис. 1.2, типова архітектура сучасної системи побудови маршрутів включає послідовні етапи від введення запиту користувача до інтеграції даних розпізнавання об'єктів, що забезпечує оперативне коригування рекомендацій.


Рис. 1.2 Схема процесу побудови маршруту з елементами розпізнавання об'єктів у сучасних системах

Для об'єктивної оцінки можливостей провідних платформ доцільно звернутися до порівняльної характеристики, представленої в Таблиці 1.2, яка висвітлює ключові відмінності в алгоритмах та функціях розпізнавання.

Таблиця 1.2

Порівняльна характеристика сучасних систем побудови маршрутів та розпізнавання об'єктів

| **Назва системи** | **Основні алгоритми** | **Функції розпізнавання об'єктів** | **Переваги** | **Обмеження** |
|---|---|---|---|---|
| Google Maps | A* з евристикою, Dijkstra | Реал-тайм трафік, розпізнавання знаків і заторів | Висока точність прогнозу, глобальне покриття | Залежність від інтернет-з'єднання |
| Waze | Crowdsourced routing | Користувацькі звіти про об'єкти та перешкоди | Динамічне оновлення в реальному часі | Менша точність у малонаселених районах |
| Apple Maps | Custom graph algorithms | AI для розпізнавання дорожніх умов | Інтеграція з екосистемою пристроїв | Обмежене покриття в деяких регіонах |
| HERE Maps | Advanced routing з мультифакторним аналізом | Сенсорне розпізнавання інфраструктури | Офлайн-режим і точність для логістики | Комерційна орієнтація |

З наведеного аналізу випливає, що сучасні системи демонструють високий рівень інтеграції технологій розпізнавання об'єктів, що дозволяє не лише оптимізувати маршрути за критеріями часу, відстані та безпеки, але й забезпечувати проактивне реагування на зміни в дорожньому середовищі, зокрема через дані від користувачів та сенсорів. Водночас у контексті розвитку транспортної інфраструктури України впровадження таких систем сприяє підвищенню ефективності логістики та безпеки руху, адаптуючи глобальні технології до місцевих умов [5].

Подальший розвиток цих технологій пов'язаний з удосконаленням алгоритмів і розширенням можливостей штучного інтелекту для точнішого розпізнавання в складних умовах, таких як сільські дороги чи несприятлива

погода, де традиційні методи можуть втрачати точність [6]. Загалом, аналіз свідчить про те, що сучасні системи побудови маршрутів уже стали надійним інструментом для вирішення завдань транспортної галузі, проте їх ефективність безпосередньо залежить від якості розпізнавання об'єктів і інтеграції реальних даних [7]. Це створює передумови для подальшого вдосконалення, спрямованого на підвищення адаптивності та універсальності в різних регіональних контекстах [8].

### 1.3 Моделювання предметної області

Моделювання предметної області є одним із фундаментальних етапів системного аналізу, який дозволяє формалізувати сутність транспортних процесів, встановити чіткі взаємозв’язки між елементами інфраструктури та створити математичну основу для оптимізації маршрутів [9]. У контексті інформаційних систем знаходження оптимальнішого маршруту предметна область охоплює транспортну мережу як складну динамічну систему, де кожна точка переміщення та кожне дорожнє сполучення наділені численними характеристиками, що впливають на кінцевий вибір шляху [10]. Такий підхід забезпечує можливість врахування не лише геометричних параметрів, а й часових витрат, рівня безпеки та економічної доцільності, сприяючи раціональному використанню ресурсів і підвищенню загальної ефективності логістики.

Транспортна мережа в моделі представлена у вигляді орієнтованого графу, вершини якого відповідають перехрестям, вузловим пунктам або місцям відправлення та призначення, а ребра – дорожнім сегментам із притаманними їм атрибутами [11]. Ця структура дозволяє точно описувати реальні умови руху, ураховуючи тип покриття, обмеження швидкості та статистичні показники ризику. Як показано на рис. 1.3, модель предметної області ілюструє ієрархічну взаємодію між основними компонентами – від загальної транспортної мережі до конкретних характеристик окремих елементів.

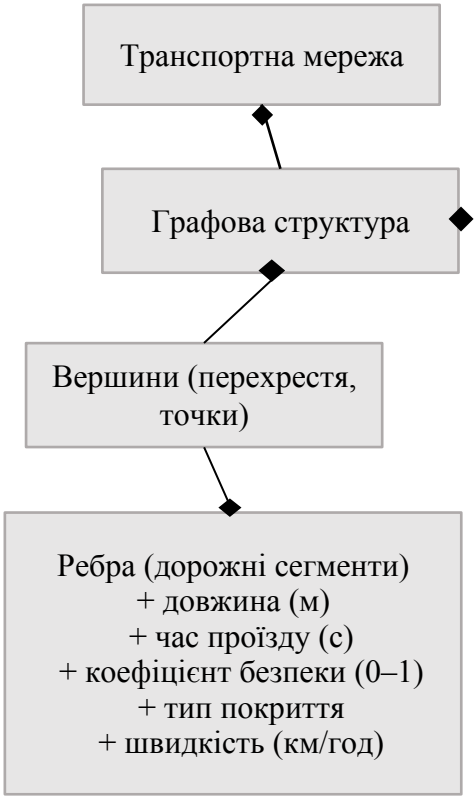


Рис. 1.3 Модель предметної області транспортної мережі

З рисунку чітко видно, що модель забезпечує комплексний опис усіх необхідних елементів, підкреслюючи їхню взаємозалежність і можливість багатокритеріальної оцінки.

Для систематизації ключових характеристик дорожніх сегментів доцільно звернутися до таблиці 1.3, яка наводить основні параметри, що використовуються при моделюванні.

Таблиця 1.3

Основні атрибути елементів моделі транспортної мережі

| **Параметр** | **Опис** | **Одиниця виміру** |
|---|---|---|
| Довжина | Фізична відстань сегмента | метри |
| Час проїзду | Розрахунковий час проходження з урахуванням швидкості | секунди |
| Коефіцієнт безпеки | Рівень безпеки на основі типу дороги | безрозмірний (0–1) |
| Тип | Класифікація дорожнього покриття | рядок |

| | | |
|---|---|---|
| покриття | | |
| Швидкість | Середня або максимальна допустима швидкість | км/год |

Наведена таблиця демонструє, як атрибути ребер дозволяють здійснювати багатокритеріальну оптимізацію, де кожен параметр отримує відповідну вагу залежно від пріоритетів користувача чи специфіки режиму руху. Моделювання процесів оптимізації базується на застосуванні алгоритмів пошуку шляху, що враховують різні цільові функції – від мінімального часу до максимальної безпеки. Зокрема, евристичні методи забезпечують швидке наближення до рішення, класичні алгоритми гарантують точність, а двонаправлені підходи підвищують ефективність обчислень. Консенсусний аналіз результатів кількох незалежних алгоритмів виступає додатковим механізмом верифікації, що дозволяє виявляти випадки повної згоди або часткових розбіжностей і формулювати обґрунтовані рекомендації для користувача [12].

Отже, моделювання предметної області надає необхідну теоретичну базу для створення ефективних інформаційних систем у сфері транспорту, сприяючи не лише технічній оптимізації, а й соціально-економічному розвитку логістики та підвищенню безпеки дорожнього руху в сучасних умовах.

## 1.4 Аналіз вимог до інформаційної системи

Аналіз вимог до інформаційної системи знаходження оптимальнішого маршруту є фундаментальним етапом системного аналізу предметної області, оскільки саме на цьому рівні формуються чіткі критерії, які забезпечують відповідність майбутнього рішення реальним потребам транспортної галузі та агропромислового комплексу в умовах цифрової трансформації. Такий аналіз передбачає всебічне вивчення очікувань користувачів, специфіки обробки просторових даних та обмежень експлуатаційного середовища, що в сукупності дозволяє створити систему, здатну не лише обчислювати маршрути, але й надавати обґрунтовані рекомендації з урахуванням множинних факторів оптимальності [13].

Центральне місце в цьому процесі посідають функціональні вимоги, які визначають основний набір можливостей системи. Вони охоплюють процес введення початкової та кінцевої точок маршруту з підтримкою геокодування адрес і зворотного перетворення координат у зрозумілі назви, що суттєво полегшує взаємодію з користувачем.

Система повинна виконувати розрахунок маршрутів на основі комплексного критерію оптимальності, що поєднує час проїзду, рівень безпеки дорожнього покриття та фізичну відстань через зважену формулу, а також реалізовувати паралельне застосування кількох алгоритмів пошуку шляху з подальшим порівнянням їх результатів для визначення ступеня згоди або розбіжності. Окрім того, важливим є автоматичне генерування покрокової легенди маршруту з детальними інструкціями щодо маневрів, назв вулиць та сегментних відстаней, а також формування звітів у різних форматах для документування та архівування результатів.

Система зобов’язана підтримувати збереження та пошук раніше розрахованих маршрутів у базі даних, забезпечуючи можливість тегування, фільтрації та експорту даних, водночас адаптуючись до різних режимів роботи, що враховують специфіку міського, міжміського чи агропромислового середовища шляхом коригування пріоритетів і коефіцієнтів [14].

Для систематизації викладених аспектів доцільно звернутися до таблиці 1.4, яка наочно демонструє класифікацію функціональних вимог за основними категоріями.

Таблиця 1.4

Класифікація функціональних вимог до інформаційної системи

| **Категорія вимог** | **Опис** | **Ключові характеристики** |
|---|---|---|
| **1** | **2** | **3** |
| Введення даних | Підтримка геокодування та введення координат | Нормалізація запитів, кешування результатів |
| Маршрутизація | Обчислення оптимальних шляхів за комплексним | Використання кількох алгоритмів, зважена |

| | | |
|---|---|---|
| | критерієм | оптимізація |
| Аналіз результатів | Порівняння маршрутів та визначення консенсусу | Виявлення статусів згоди, пояснення розбіжностей |

Продовження таблиці 1.4

| 1 | 2 | 3 |
|---|---|---|
| Вивід інформації | Генерація легенди, звітів та візуалізації | Покрокові інструкції, PDF/HTML формати |
| Зберігання даних | Робота з базою даних маршрутів | CRUD-операції, пошук, тегування |

Не менш критичними є нефункціональні вимоги, які забезпечують стабільність, ефективність та зручність експлуатації системи в реальних умовах. Вони включають високі показники продуктивності, зокрема швидке завантаження графів транспортної мережі та розрахунок маршрутів за лічені секунди навіть на значних обсягах даних.

Система повинна бути кросплатформеною, підтримуючи єдину кодову базу для різних операційних середовищ, включаючи десктопні та мобільні пристрої, з можливістю повноцінної роботи в офлайн-режимі завдяки локальному зберіганню даних і графів. Важливим аспектом залишається надійність валідації вхідних даних для запобігання помилкам, а також забезпечення захисту інформації та інтуїтивно зрозумілого інтерфейсу, що відповідає сучасним стандартам ергономіки [15].

Для наочного відображення взаємозв'язку між основними групами вимог пропонується розглянути схему, яка ілюструє їх ієрархічну структуру в стилі SmartArt (Рис. 1.4).

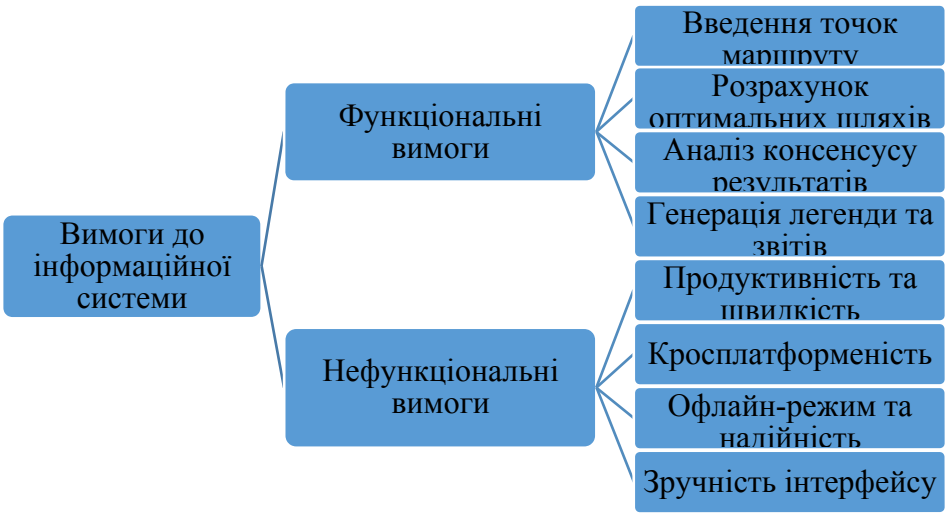


Рис. 1.4 Ієрархічна структура вимог до інформаційної системи

Таким чином, комплексний аналіз вимог дозволяє сформувати цілісне бачення інформаційної системи, яка не тільки оптимізує транспортні маршрути, але й забезпечує науково обґрунтовані рекомендації, сприяючи підвищенню ефективності логістичних процесів у транспортній галузі та агропідприємствах [16].

### 1.5 Постановка задачі

У сучасних умовах розвитку транспортної галузі одним із найактуальніших завдань системного аналізу предметної області є постановка задачі знаходження оптимального маршруту, що забезпечує баланс між економічною ефективністю, безпекою перевезень та адаптивністю до різноманітних експлуатаційних умов. Ця проблема набуває особливого значення в країнах з розгалуженою дорожньою мережею, де транспортні потоки поєднують міські, міжміські та агропромислові напрямки, а фактори зовнішнього середовища, такі як тип покриття доріг, сезонні зміни та інтенсивність руху, суттєво впливають на кінцевий результат логістичних рішень [17].

Постановка задачі передбачає формалізацію транспортної мережі як орієнтованого графа

$$G = (V, E)$$

де множина вершин $V$ відповідає перехрестям або ключовим вузлам інфраструктури, а множина ребер $E$ – дорожнім сегментам з притаманними їм атрибутами.

Кожне ребро $e \in E$ характеризується вектором параметрів, що включає довжину $L(e)$ у метрах, розрахунковий час проходження $T(e)$ у секундах та коефіцієнт безпеки $S(e)$ у діапазоні від 0 до 1, який відображає ризик для учасників руху залежно від класу дороги, її технічного стану та статистичних даних про інциденти. Мета полягає у визначенні шляху $P$ від початкової

вершини s до кінцевої вершини *t*, який мінімізує складену цільову функцію *W(P)*, побудовану як зважену суму нормалізованих критеріїв: часу, відстані та безпеки.

Такий підхід дозволяє враховувати не лише класичну задачу найкоротшого шляху, але й багатопараметричну оптимізацію, що є необхідним для реальних транспортних систем, де пріоритети користувачів можуть змінюватися залежно від контексту перевезень [18].

Як свідчить таблиця 1.5, розподіл вагових коефіцієнтів у цільовій функції забезпечує комплексний баланс інтересів, роблячи рішення придатним для різних режимів експлуатації транспортної мережі.

Таблиця 1.5

Основні критерії оптимізації маршрутів у транспортних системах

| **Критерій** | **Вага (%)** | **Опис параметра** |
|---|---|---|
| Час у дорозі | 40 | Мінімізація тривалості з урахуванням швидкісних обмежень |
| Рівень безпеки | 35 | Інверсія ризику на основі типу покриття та статистики |
| Загальна відстань | 25 | Оптимізація фізичної довжини шляху |

Ця структура ваг відображає пріоритетність оперативності без ігнорування безпеки чи економії ресурсів, що особливо важливо в умовах динамічного розвитку логістики.

Як показано на рис. 1.5, постановка задачі може бути представлена у вигляді структурної схеми, яка ілюструє послідовність етапів від моделювання мережі до консенсусного підтвердження оптимальності.

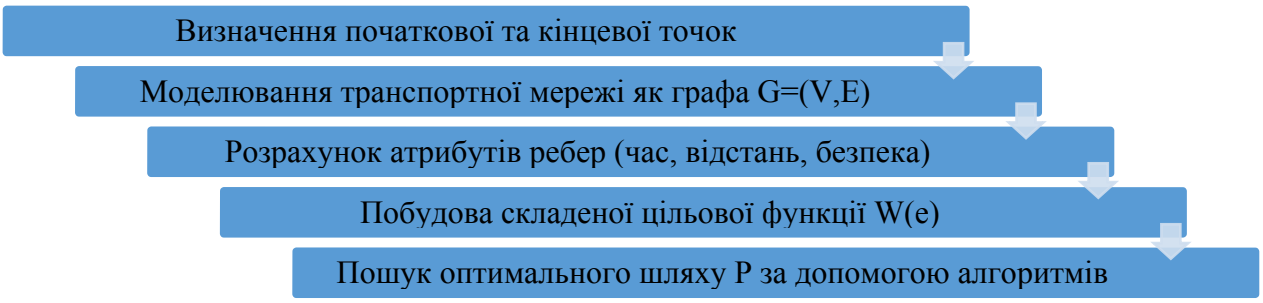

Рис. 1.5 Структурна схема постановки задачі знаходження оптимального маршруту в транспортній мережі

Наведена схема підкреслює системний характер проблеми, де кожен етап взаємопов’язаний і спрямований на досягнення не лише математичного мінімуму, але й практичної надійності рішення в реальних умовах. У контексті багатопараметричної оптимізації особливу увагу приділяють аналізу розбіжностей між альтернативними шляхами, що виникають через варіативність вагових коефіцієнтів або локальні особливості інфраструктури, такі як ґрунтові ділянки в агропромислових регіонах чи інтенсивність руху в урбанізованих зонах. Це вимагає введення додаткових механізмів верифікації, коли кілька незалежних методів пошуку порівнюються для визначення ступеня згоди або необхідності коригування маршруту залежно від пріоритетів користувача [19].

Таким чином, постановка задачі не обмежується класичним пошуком найкоротшого шляху, а охоплює комплексний багатопараметричний аналіз, що забезпечує адаптивність до реальних викликів транспортної логістики, сприяючи підвищенню ефективності та безпеки перевезень у сучасних умовах [20]. Такий підхід відкриває перспективи для створення інформаційних систем, здатних підтримувати прийняття обґрунтованих рішень у динамічному середовищі.

## 1.6 Висновки до першого розділу

Системний аналіз транспортної та аграрної предметних областей виявив їхню тісну взаємозалежність у процесах логістики агропродукції, де оптимізація маршрутів набуває багатопараметричного характеру з урахуванням часу проїзду, фізичної відстані, коефіцієнта безпеки та сезонних особливостей дорожнього покриття. Порівняльна оцінка сучасних систем маршрутизації та розпізнавання об'єктів засвідчила високий потенціал інтеграції графових алгоритмів і штучного інтелекту, проте виявила обмеження щодо адаптації до локальних умов сільських і ґрунтових шляхів України.

Моделювання предметної області як орієнтованого графа з атрибутивними ребрами дозволило формалізувати ключові характеристики сегментів мережі та

закласти основу для багатокритеріальної оптимізації. Аналіз функціональних і нефункціональних вимог до інформаційної системи визначив необхідність підтримки геокодування, паралельного застосування алгоритмів, консенсусного аналізу результатів і забезпечення офлайн-режиму.

Постановка задачі як мінімізації зваженої цільової функції на графі з нормалізованими критеріями забезпечила комплексний підхід до балансу економічної ефективності, безпеки та адаптивності перевезень.

Таким чином, проведений аналіз створив теоретичну базу для розробки ефективної інформаційної системи знаходження оптимального маршруту в умовах реальної транспортної та аграрної інфраструктури.

# 2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 2.1 Логічна модель даних у вигляді ER діаграми

У процесі проектування інформаційного забезпечення систем оптимізації маршрутів у транспортній галузі логічна модель даних відіграє фундаментальну роль, оскільки вона забезпечує абстрактне, незалежне від конкретної технологічної реалізації описання сутностей, їхніх атрибутів та взаємозв’язків, що дозволяє уникнути аномалій оновлення, вставки чи видалення інформації та гарантувати цілісність даних на рівні концептуального рівня.

Такий підхід базується на реляційній парадигмі, де кожна сутність представлена у вигляді таблиці, а зв’язки між ними визначаються через первинні та зовнішні ключі, що відповідає сучасним принципам нормалізації баз даних і сприяє ефективній обробці географічних та логістичних даних [21].

Особливе значення набуває визначення ключових об’єктів предметної області, зокрема тих, що пов’язані з маршрутами як центральними одиницями зберігання, де враховуються координати точок відправлення та призначення, режими функціонування, результати аналізу та метадані для забезпечення комплексного опису кожного запису.

Логічна модель підкреслює ієрархічні відношення, коли одна основна сутність може асоціюватися з кількома деталізуючими варіантами, що відображає реальну практику порівняльного аналізу альтернативних розрахунків за критеріями ефективності, безпеки та відстані, тим самим сприяючи прийняттю обґрунтованих рішень у динамічних умовах транспортних задач [22].

Додаткові компоненти моделі, такі як механізми тимчасового зберігання результатів пошуку адрес чи параметрів конфігурації системи, забезпечують оперативність операцій і персоналізацію без надмірного навантаження на основні структури, що робить модель гнучкою та адаптованою до реальних сценаріїв використання в умовах обмежених ресурсів.

Логічна модель даних у вигляді ER-діаграми (рис. 2.1) наочно ілюструє ці взаємозв’язки, підкреслюючи центральну роль основної сутності та її

розширення через пов’язані об’єкти, що є основою для подальшого переходу до фізичного проектування.

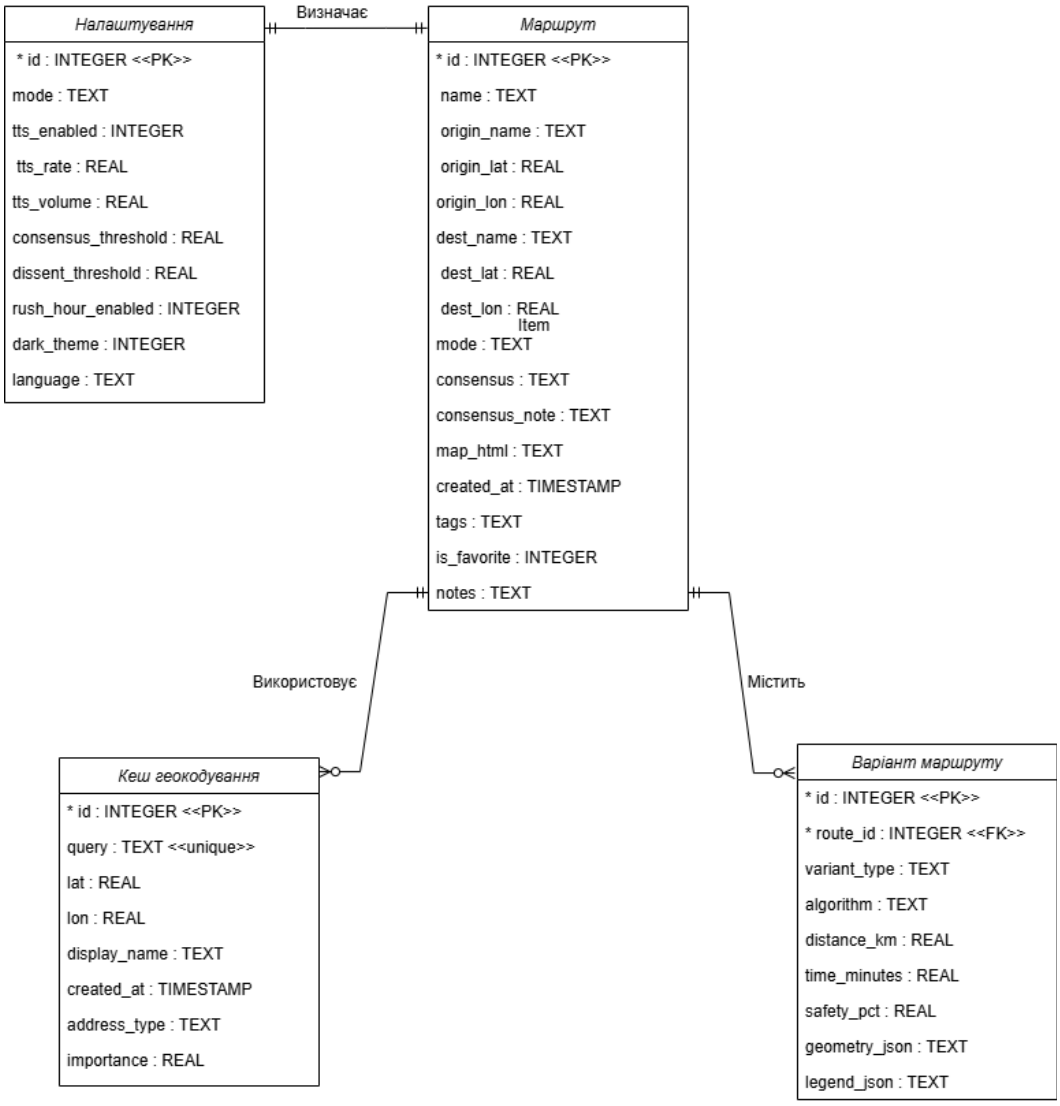


Рис. 2.1 Логічна модель даних у вигляді ER-діаграми

Для систематизації уявлення про структуру даних доцільно звернутися до таблиці 2.1, яка деталізує ключові атрибути основних сутностей і їх семантичне навантаження, що дозволяє краще зрозуміти, як модель підтримує повноцінне функціонування системи без порушення принципів цілісності.

Таблиця 2.1

Ключові атрибути основних сутностей логічної моделі даних

| **Сутність** | **Ключові атрибути** | **Опис** |
|---|---|---|

| | | |
|---|---|---|
| Маршрут | id, origin_name, dest_name, mode, consensus, created_at | Основна інформація про маршрут з метаданими |
| Варіант маршруту | id, route_id, variant_type, distance_km, time_minutes, safety_pct | Деталі альтернативних розрахунків |
| Кеш геокодування | id, query, lat, lon, display_name | Тимчасове зберігання результатів пошуку адрес |
| Налаштування | id, mode, tts_enabled, consensus_threshold | Параметри конфігурації системи |

Такий детальний опис атрибутів демонструє, як логічна модель забезпечує не лише зберігання, але й оперативний доступ до даних, необхідних для аналізу та візуалізації, що є критично важливим для систем, орієнтованих на реальний час [23].

Запропонована модель відповідає принципам реляційної нормалізації, де уникнення дублювання досягається через зовнішні ключі, а підтримка цілісності реалізується через обмеження на рівні доменів і зв’язків, що робить її стійкою до змін у вимогах предметної області. У контексті транспортних інформаційних систем подібний підхід сприяє інтеграції географічних даних із логістичними параметрами, забезпечуючи користувачам точні та обґрунтовані рекомендації без зайвого навантаження на обчислювальні ресурси [24].

Таким чином, логічна модель даних, представлена у формі ER-діаграми та доповнена описом атрибутів, формує надійну концептуальну основу для всього інформаційного забезпечення, гарантуючи масштабованість, ефективність обробки та високу точність результатів у сфері оптимізації маршрутів.

## 2.2 Діаграми класів та взаємодії

Проектування інформаційного та програмного забезпечення системи знаходження оптимальнішого маршруту ґрунтується на принципі чіткої структурної організації, де діаграми класів та взаємодії відіграють ключову роль у забезпеченні цілісності архітектури. Діаграма класів відображає статичну структуру, демонструючи основні класи, їх атрибути, методи та різноманітні типи зв'язків, що гарантує повну інтеграцію всіх компонентів у єдину систему. Такий підхід сприяє модульності, розширюваності та надійності програмного забезпечення, дозволяючи кожному класу взаємодіяти з іншими через асоціації, залежності чи композицію, що є запорукою ефективної обробки даних у складних транспортних задачах [25].

Рис. 2.2 представлено діаграму класів основних компонентів, яка ілюструє повну взаємопов'язаність усіх класів, забезпечуючи єдиний інформаційний потік від введення даних до формування результатів.

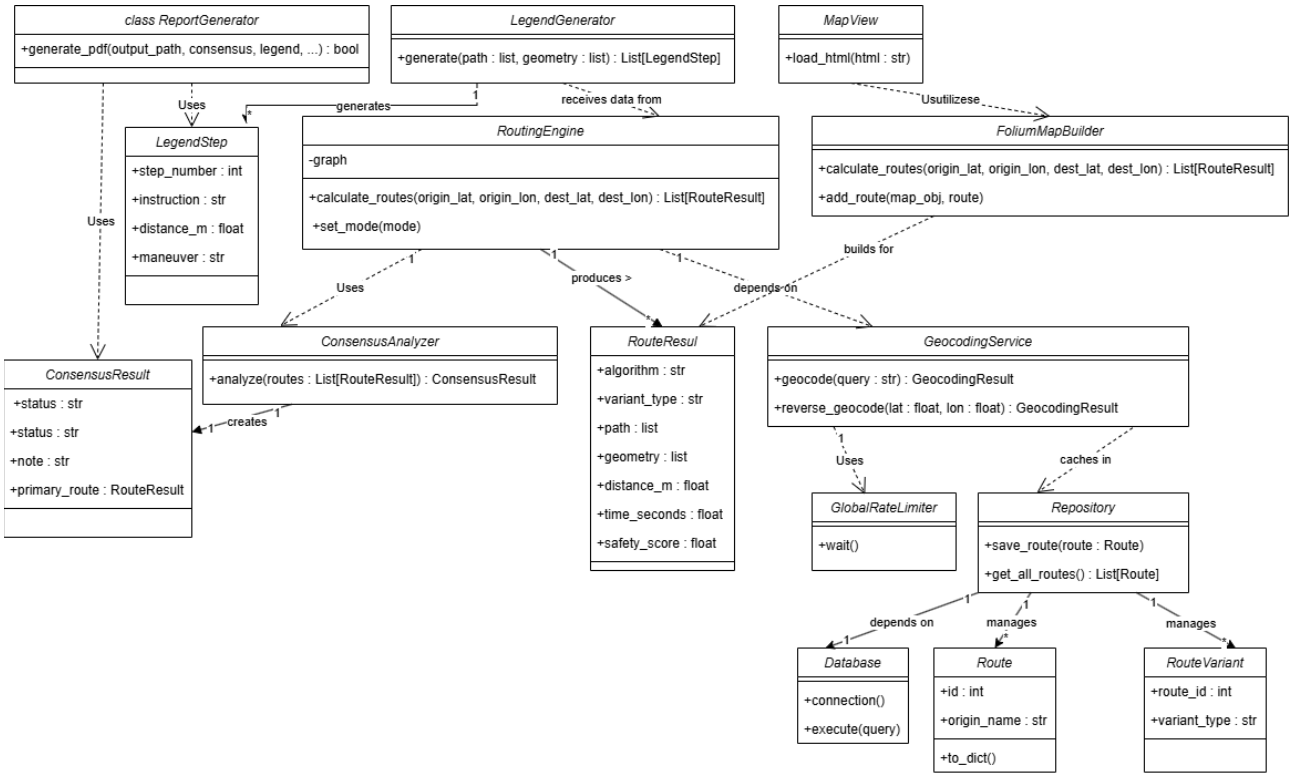


Рис. 2.2 Діаграма класів основних компонентів інформаційного та програмного забезпечення

Як видно з наведеної діаграми, клас RoutingEngine тісно пов’язаний з ConsensusAnalyzer для аналізу результатів маршрутизації, а Repository залежить від Database для забезпечення зберігання даних, водночас GeocodingService інтегрується з RoutingEngine для обробки координат, а LegendGenerator отримує дані від RoutingEngine для створення покрокових інструкцій. Така повна мережа зв’язків між класами, включаючи FoliumMapBuilder та MapView, формує єдиний інформаційний контур, де кожна частина системи доповнює іншу, сприяючи стабільній роботі в умовах реального використання [26].

Для розкриття динаміки функціонування системи суттєве значення має діаграма взаємодії, яка відображає послідовність викликів методів між компонентами під час виконання основних операцій.

Рис. 2.3 наведено діаграму послідовності, яка показує взаємодію між основними акторами та класами під час розрахунку оптимального маршруту, підкреслюючи послідовну обробку даних від введення до збереження результатів.

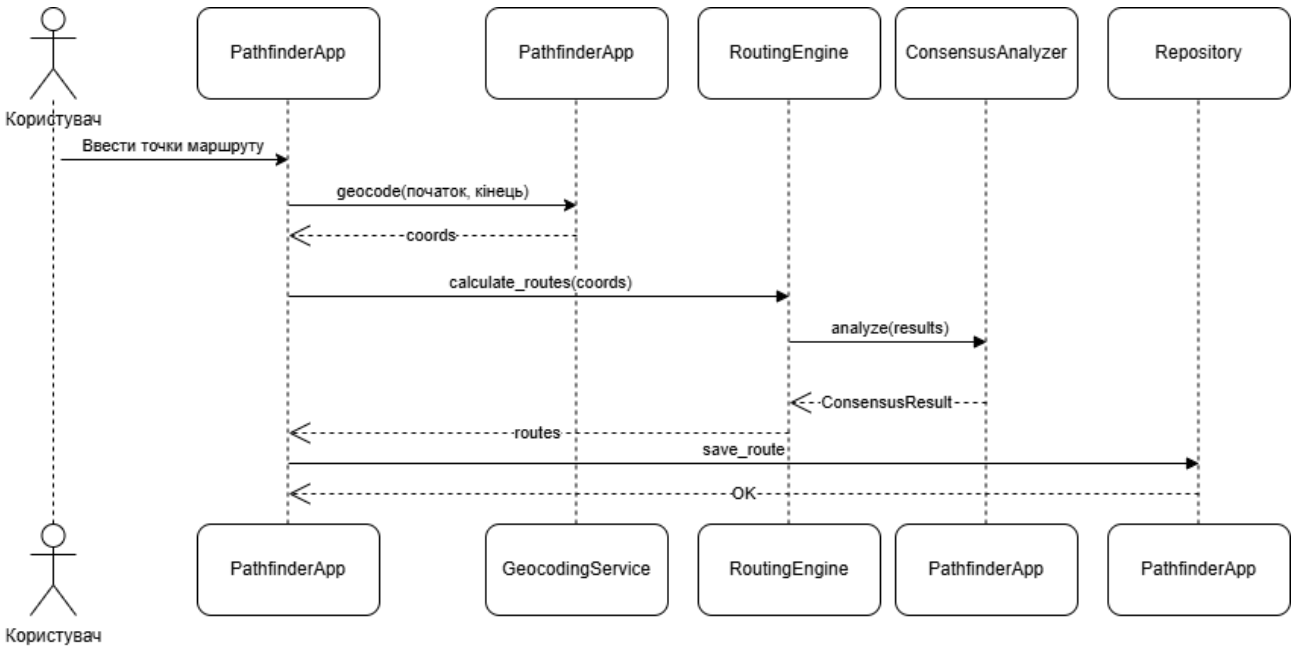


Рис. 2.3 Діаграма взаємодії (послідовності) процесу розрахунку оптимального маршруту

Ця діаграма ілюструє, як запит користувача проходить через сервіс геокодування, далі до рушія маршрутизації, де відбувається аналіз консенсусу, а

в кінці дані зберігаються через Repository, що забезпечує ефективний та послідовний інформаційний обмін між усіма класами [27]. Завдяки такій організації взаємодії система досягає високої точності та швидкості обробки, мінімізуючи можливі помилки на кожному етапі.

Таким чином, запропоновані діаграми класів та взаємодії створюють міцну основу для інформаційного та програмного забезпечення, де повна взаємопов’язаність компонентів гарантує не лише технічну досконалість, а й практичну зручність для користувачів, сприяючи сталому розвитку систем оптимізації маршрутів у сучасному світі [28].

## 2.3 Діаграма компонентів системи

У рамках проектування інформаційного та програмного забезпечення системи знаходження оптимальнішого маршруту діаграма компонентів відіграє визначальну роль, оскільки дозволяє чітко структурувати архітектуру як сукупність автономних функціональних блоків, що взаємодіють через стандартизовані інтерфейси. Такий підхід забезпечує високу модульність, полегшує супроводження та розширення системи, а також сприяє зменшенню ризиків помилок під час інтеграції окремих частин, роблячи весь процес розробки більш передбачуваним і ефективним для всіх учасників команди [29].

Компоненти системи організовані таким чином, щоб кожен виконував вузькоспеціалізовану задачу, мінімізуючи залежності та забезпечуючи можливість незалежного тестування, що відповідає сучасним принципам побудови складних інформаційних систем у транспортній сфері.

Центральне місце в архітектурі посідає двигун маршрутизації, який отримує координати точок і виконує основні обчислення, після чого аналізатор консенсусу порівнює результати для формування загального висновку. Сервіс геокодування відповідає за перетворення текстових адрес у географічні координати та навпаки, забезпечуючи точність вхідних даних, тоді як генератор легенди створює детальні покрокові інструкції на основі проаналізованого шляху.

Генератор звітів формує документи у зручних форматах, а репозиторій бази даних забезпечує зберігання маршрутів, налаштувань і кешованих результатів, підтримуючи як онлайн-, так і офлайн-режим роботи. Допоміжні компоненти, такі як валідатор вхідних даних і централізований логер, гарантують коректність інформації та фіксацію всіх подій, що підвищує надійність системи в цілому [30].

На рис. 2.4 зображено діаграму компонентів системи, яка компактно ілюструє основні модулі та напрямки їх взаємодії, підкреслюючи централізовану роль двигуна маршрутизації як ядра, навколо якого будуються всі інші елементи.

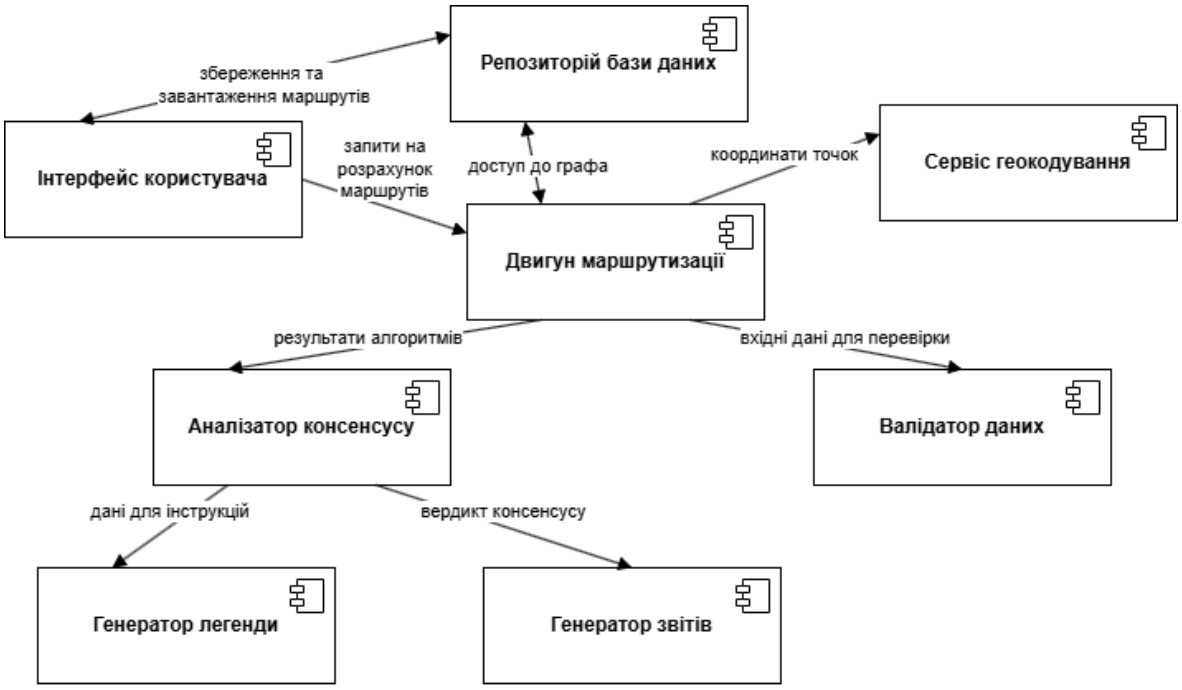


Рис. 2.4 Діаграма компонентів системи

Як видно з діаграми (рис. 2.4), інтерфейс користувача безпосередньо ініціює процес через двигун маршрутизації, який, у свою чергу, звертається до сервісу геокодування та валідатора для забезпечення точності вхідних даних, а аналізатор консенсусу стає проміжною ланкою, що передає узагальнені результати до генератора легенди та звітів, забезпечуючи єдину точку формування рекомендацій для користувача [31].

Репозиторій бази даних виступає спільним ресурсом, до якого мають

доступ майже всі компоненти, що дозволяє підтримувати цілісність даних і швидке відновлення інформації після перезапуску системи. Така організація компонентів не лише оптимізує обчислювальні ресурси, але й робить систему більш зрозумілою та зручною в експлуатації, сприяючи її адаптації до реальних потреб користувачів у динамічному середовищі транспортної логістики.

Загалом, діаграма компонентів системи відображає продуману архітектуру, де кожен елемент виконує свою роль у спільній роботі, забезпечуючи баланс між функціональністю, надійністю та зручністю використання [32]. Такий підхід дозволяє створити інформаційну систему, яка ефективно вирішує задачу знаходження оптимальнішого маршруту, залишаючись при цьому гнучкою та масштабовною для майбутніх удосконалень.

## 2.4 Діаграма пакетів системи

У процесі проектування інформаційного та програмного забезпечення для прикладної інформаційної системи знаходження оптимальнішого маршруту особливу увагу приділяють створенню чіткої архітектурної структури, яка забезпечує логічне групування функціональних елементів, їхню незалежність та ефективну взаємодію. Діаграма пакетів як елемент мови моделювання UML відіграє ключову роль у цьому процесі, оскільки дозволяє представити систему на високому рівні абстракції, об’єднуючи класи, інтерфейси та інші артефакти в більші логічні одиниці – пакети.

Такий підхід сприяє кращому розумінню архітектури, спрощує розподіл обов’язків між розробниками та забезпечує масштабованість рішення, особливо в умовах складних обчислювальних задач, пов’язаних з обробкою географічних даних і алгоритмічними розрахунками.

Діаграма пакетів системи демонструє, як окремі функціональні області організовано в компактні модулі, що мінімізує залежності та підвищує підтримуваність програмного коду. Центральне місце в ній посідає пакет ядра системи, який об’єднує елементи, відповідальні за основну логіку обробки,

розрахунків та аналізу, тоді як пакет інтерфейсу користувача відокремлює візуальну взаємодію, забезпечуючи незалежність від внутрішньої реалізації.

Пакет бази даних, у свою чергу, групує компоненти для зберігання та управління інформацією, а спеціалізовані пакети геокодування, побудови карт і генерації звітів дозволяють чітко виділити допоміжні сервіси, що взаємодіють з основним ядром через визначені інтерфейси. Така організація не лише полегшує супровід системи, але й сприяє її адаптації до змінних вимог, як зазначається в сучасних дослідженнях архітектурного моделювання [33].

На рис. 2.5 наведено діаграму пакетів системи, яка ілюструє логічне групування модулів та основні залежності між ними, підкреслюючи модульність архітектури.

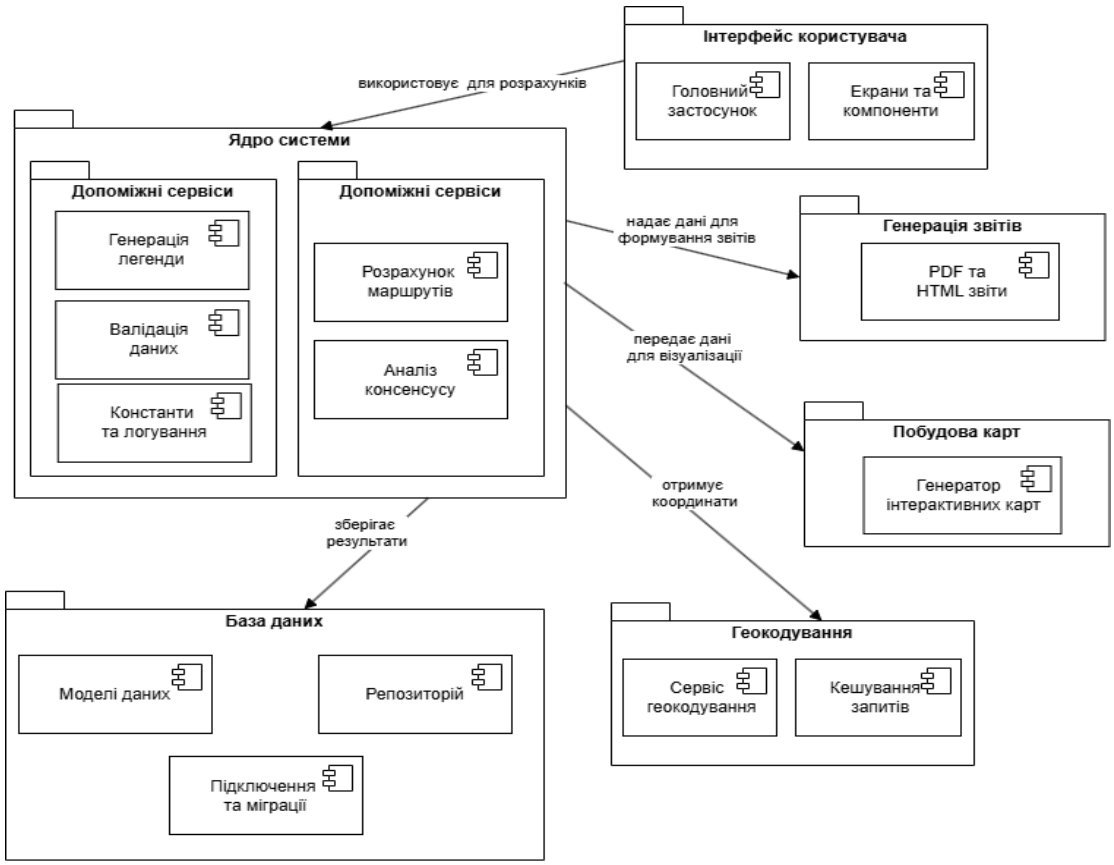


Рис. 2.5 Діаграма пакетів системи

Як видно з наведеної діаграми, залежності між пакетами є односторонніми, що уникає циклічних зв’язків і сприяє стабільності системи в цілому. Наприклад, інтерфейс користувача залежить лише від ядра, не маючи прямого доступу до бази даних, завдяки чому забезпечується принцип інкапсуляції та захист

внутрішньої логіки. Подібний розподіл відповідає принципам компонентного проектування, де кожен пакет виконує чітко визначену роль, полегшуючи тестування та розширення функціональності без ризику порушення цілісності всього рішення [34].

Така структура діаграми пакетів забезпечує високу гнучкість розробки, оскільки дозволяє командам працювати над окремими модулями паралельно, мінімізуючи конфлікти інтеграції. Пакет ядра системи, як центральний елемент, інтегрує алгоритмічну логіку з допоміжними сервісами, створюючи єдиний механізм для обробки запитів, тоді як спеціалізовані пакети, такі як геокодування чи побудова карт, слугують незалежними сервісами, що підвищує загальну продуктивність і надійність інформаційної системи. У науковій літературі підкреслюється, що подібне пакетне моделювання особливо корисне для транспортних інформаційних систем, де необхідна чітка сепарація обчислювальних компонентів від інтерфейсних і зберігальних [35].

Крім того, діаграма пакетів виступає основою для подальшого детального проектування, наприклад, для створення діаграм класів чи компонентів, забезпечуючи послідовність архітектурних рішень на всіх етапах життєвого циклу системи. Вона допомагає візуалізувати, як саме інформаційні потоки рухаються між модулями, сприяючи кращому розумінню системи як єдиного цілого та її відповідності вимогам до ефективності маршрутизації. У контексті сучасних підходів до програмної інженерії такий інструмент стає невід’ємною частиною документації, що полегшує комунікацію між стейкхолдерами та розробниками [36].

Таким чином, діаграма пакетів системи не тільки фіксує поточну архітектуру, але й створює основу для майбутнього розвитку, роблячи інформаційне та програмне забезпечення більш адаптивним, зрозумілим і ефективним у практичному застосуванні.

## 2.5 Висновки до другого розділу

У процесі проектування інформаційного та програмного забезпечення системи оптимізації маршрутів у транспортній галузі сформовано логічну модель даних на основі ER-діаграми, яка забезпечує чітке визначення сутностей, атрибутів і взаємозв’язків між ними. Така модель відповідає принципам реляційної нормалізації, усуває можливі аномалії даних і підтримує ефективну роботу з географічною та логістичною інформацією.

Розроблені діаграми класів та взаємодії відображають статичну структуру програмних компонентів і динаміку їхньої взаємодії під час виконання ключових операцій. Вони забезпечують модульність архітектури, чітку інтеграцію класів RoutingEngine, ConsensusAnalyzer, GeocodingService та інших елементів, що сприяє надійній обробці даних у реальному часі.

Діаграма компонентів системи ілюструє автономні функціональні блоки та стандартизовані інтерфейси між ними, з акцентом на центральну роль двигуна маршрутизації. Це дозволяє досягти високої гнучкості, тестованості та масштабовності рішення.

Діаграма пакетів демонструє логічне групування модулів, односторонні залежності та принцип інкапсуляції, що полегшує супровід, паралельну розробку та подальше розширення системи без порушення її цілісності.

Запропоновані моделі та діаграми створюють цілісну, стійку та адаптивну архітектуру інформаційного та програмного забезпечення, яка відповідає вимогам сучасних транспортних систем і становить надійну основу для подальшої реалізації та впровадження.

# 3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

## 3.1 Вибір технологій та інструментів реалізації

У процесі проектування та реалізації прикладної інформаційної системи Pathfinder для знаходження оптимальнішого маршруту особливу увагу було приділено вибору технологій та інструментів, які забезпечують кросплатформенність, високу продуктивність обробки графових даних і повноцінну роботу в офлайн-режимі. Мова програмування Python версії 3.11 і вище стала основою всього рішення завдяки своїй гнучкості, багатій екосистемі бібліотек для роботи з графами та даними, а також можливості швидкого прототипування складних алгоритмів маршрутизації без втрати читабельності коду.

Саме Python дозволив інтегрувати в єдину архітектуру модулі для паралельного виконання трьох незалежних алгоритмів пошуку шляху, що є ключовою особливістю консенсусної маршрутизації в цій системі, як це реалізовано в модулі routing.py через ThreadPoolExecutor. Такий підхід не тільки спрощує розробку, але й робить систему доступною для широкого кола користувачів – від логістів транспортних компаній до диспетчерів агропідприємств, забезпечуючи при цьому надійність і масштабованість [37].

Для побудови користувацького інтерфейсу було обрано фреймворк Kivy у поєднанні з бібліотекою KivyMD, що дало змогу створити єдину кодову базу для десктопних платформ Windows і Linux, а також для Android без потреби в переписуванні програми під кожну ОС [38]. Цей вибір виявився оптимальним для реалізації адаптивного інтерфейсу з елементами Material Design, включаючи динамічні екрани головної карти, легенди маршруту та збережених маршрутів, як видно з файлів main_screen.kv, legend_screen.kv та app.py. Kivy забезпечує нативну продуктивність на мобільних пристроях, підтримує touch-взаємодію та інтеграцію з MapView, що критично важливо для інтуїтивного тапу по карті для встановлення точок старту та фінішу [39].

Моделювання дорожньої мережі та безпосередня реалізація алгоритмів маршрутизації базуються на бібліотеках NetworkX та OSMnx. NetworkX надає

ефективні інструменти для роботи з багатонаправленими графами, включаючи реалізацію A* з евристикою, класичного Dijkstra з модифікованою вагою безпеки та двонаправленого Dijkstra, що виконуються паралельно в routing.py.

OSMnx доповнює цю функціональність можливістю завантаження та збагачення даних OpenStreetMap, створення локального графа Києва з радіусом 15 км, розрахунком ваг оптимальності (40 % час, 35 % безпека, 25 % відстань) та збагаченням ребер атрибутами швидкості та безпеки з констант constants.py. Такий стек забезпечує швидке завантаження графа з data/graphs/kyiv.graphml і офлайн-роботу без зовнішніх запитів після першого завантаження [40].

Геокодування адрес і зворотне геокодування координат здійснюється за допомогою Geopy з сервісом Nominatim, доповненим глобальним rate-limiter у geocoding.py та кешуванням у SQLite для уникнення перевищення лімітів. Цей інструмент інтегровано з валідацією в validation.py і моделями даних у models.py, що дозволяє користувачеві вводити адреси текстом або тапом на карті з автоматичним оновленням назв.

Для генерації звітів і візуалізації результатів обрано ReportLab для створення PDF-документів з таблицями варіантів і покроковою легендою, а також Folium для інтерактивних HTML-карт у folium_builder.py, що зберігаються разом зі збереженими маршрутами в репозиторії. Локальне зберігання даних реалізовано через SQLite з міграціями в db.py, що гарантує повну автономність системи та швидкий доступ до історії маршрутів.

Технологічний стек системи описаний в таблиці 3.1.

Таблиця 3.1

Основні технологічні компоненти системи Pathfinder

| **Технологія / Бібліотека** | **Версія** | **Основне призначення** |
|---|---|---|
| **1** | **2** | **3** |
| Python | 3.11+ | Базова мова програмування, інтеграція модулів |

Продовження таблиці 3.1

| 1 | 2 | 3 |
|---|---|---|
| Kivy + KivyMD | 2.3.0 / 1.2.0 | Кросплатформений інтерфейс користувача |
| NetworkX + OSMnx | 3.3 / 1.9.0 | Графові моделі та алгоритми маршрутизації |
| Geopy | 2.4.1 | Геокодування та rate-limiting |
| Folium + ReportLab | 0.17.0 / 4.2.0 | Візуалізація карт і генерація звітів |
| SQLite | Вбудована | Локальне зберігання маршрутів і кешу |

Рис. 3.1 демонструє компонентну архітектуру системи, підкреслюючи чіткий поділ на шари UI, core-модулів маршрутизації та даних, що забезпечує модульність і легкість розширення.

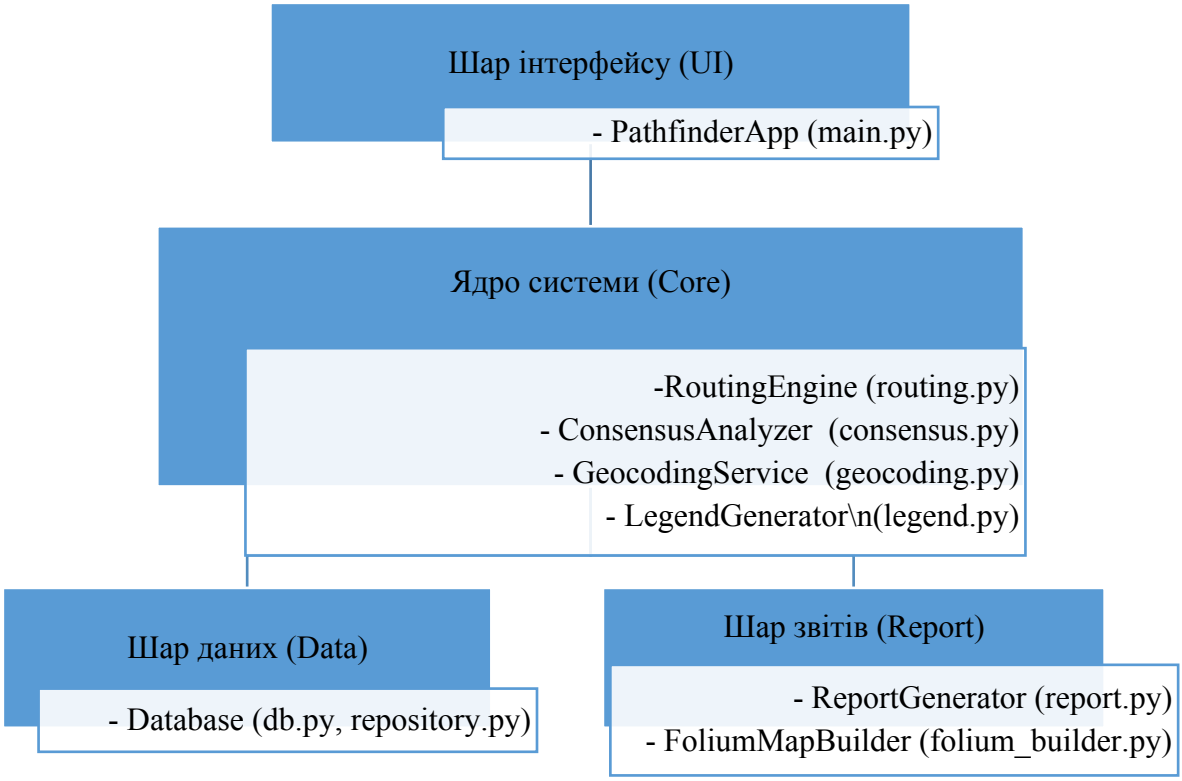


Рис. 3.1 Компонентна схема архітектури програмної системи Pathfinder

Така архітектура, побудована на перевірених відкритих інструментах, повністю відповідає специфікації проєкту, забезпечуючи не тільки технічну ефективність, але й практичну користь для користувачів, які потребують

надійних рекомендацій у транспортній та агропромисловій сферах. Обраний стек технологій дозволив реалізувати консенсусний аналіз у consensus.py, покрокову легенду та експорт звітів без зовнішніх залежностей після початкового завантаження графа, що робить Pathfinder зручним і доступним рішенням для повсякденного використання.

### 3.2 Фізична модель даних та структура бази даних

У рамках проектування та реалізації програмної системи Pathfinder фізична модель даних відіграє ключову роль у забезпеченні стабільного та ефективного зберігання інформації, необхідної для роботи з маршрутами, їх варіантами, результатами консенсусного аналізу та додатковими сервісами. Система використовує реляційну базу даних SQLite, яка повністю відповідає вимогам кросплатформеності та офлайн-функціональності, оскільки не потребує окремого серверного компонента, працює локально на пристрої користувача та підтримує вбудовані механізми цілісності даних.

Такий підхід дозволяє зберігати всю історію маршрутів, геокодовані адреси та налаштування без постійного доступу до мережі, що особливо важливо для навігаційного застосунку, орієнтованого на транспортну галузь та агропідприємства. Фізична модель побудована на основі чітко визначених таблиць, які відображають реальні сутності проекту, їх атрибути та логічні зв’язки, забезпечуючи швидкий доступ до даних під час генерації звітів, перегляду легенди чи відновлення попередніх розрахунків.

Центральним елементом моделі є таблиця, що фіксує загальну інформацію про кожен маршрут, включаючи координати точок, режим роботи, статус консенсусу та вбудовану HTML-карту. Ця таблиця слугує основою для всіх інших компонентів, дозволяючи користувачеві зберігати, шукати та повторно використовувати маршрути без повторного розрахунку. Для наочності фізичної моделі даних наведено її схематичне представлення у вигляді діаграми сутностей та зв’язків на рис. 3.2.

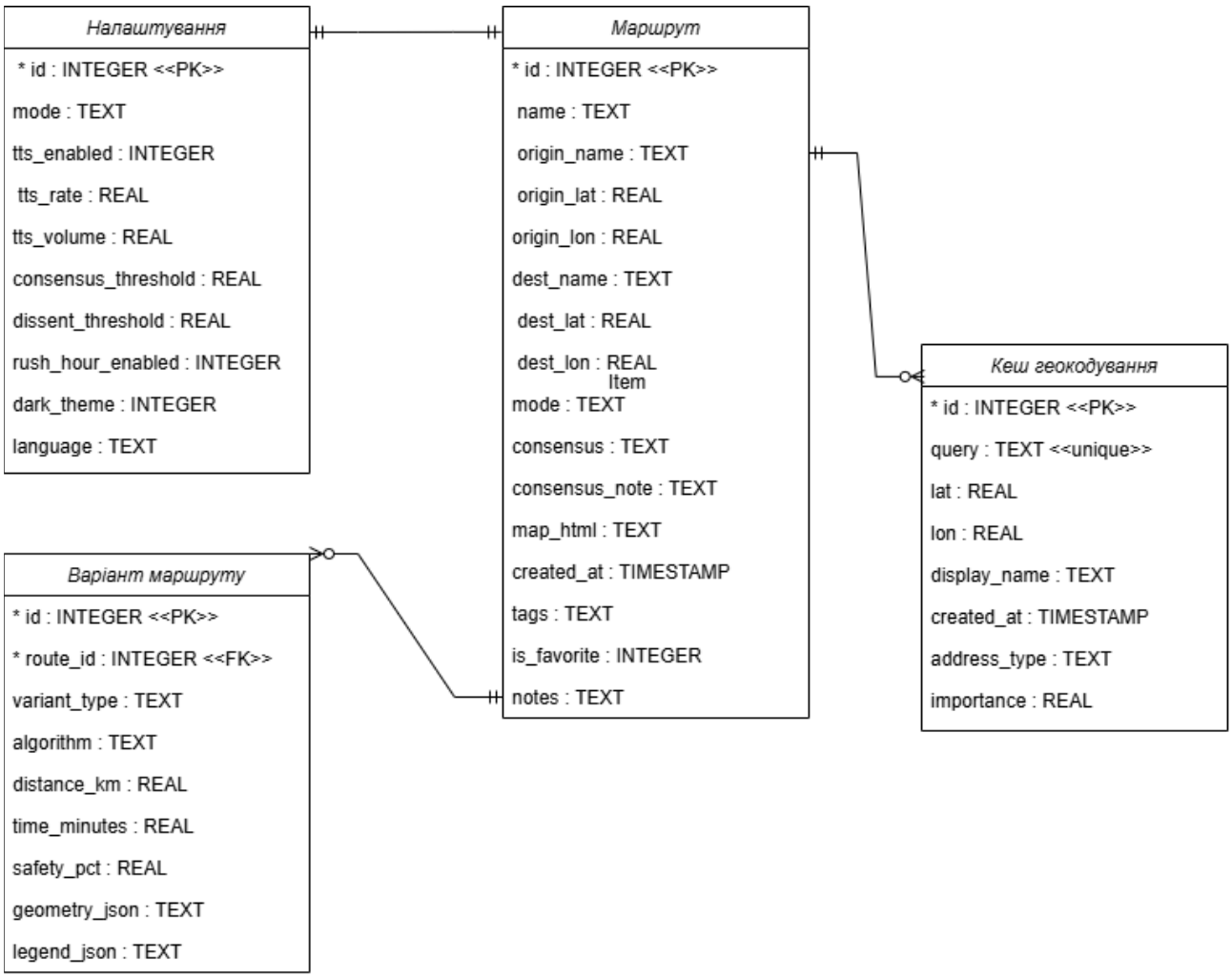


Рис. 3.2 Схема фізичної моделі даних бази SQLite

На наведеній схемі чітко простежується ієрархічна залежність між таблицями, де таблиця «Маршрути» є батьківською для «Варіантів маршрутів» через зовнішній ключ route_id, що реалізує зв’язок «один до багатьох» і забезпечує автоматичне каскадне видалення пов’язаних записів. Інші таблиці залишаються незалежними, що дозволяє їм ефективно підтримувати допоміжні функції без зайвого навантаження на основний потік даних. Така структура повністю відповідає реалізації в модулі db.py, де під час ініціалізації схеми створюються всі необхідні таблиці, індекси та обмеження CHECK для полів consensus і variant_type.

Детальна структура таблиць бази даних Pathfinder представлена в таблиці 3.2, яка узагальнює ключові поля, їх типи даних та функціональне призначення в контексті роботи системи.

Таблиця 3.2

Структура таблиць бази даних Pathfinder

| **Назва таблиці** | **Поле** | **Тип даних** | **Призначення** |
|---|---|---|---|
| **1** | **2** | **3** | **4** |
| routes | id | INTEGER (PK) | Унікальний ідентифікатор маршруту |
| routes | name | TEXT | Автоматична або задана назва маршруту |
| routes | origin_name | TEXT | Назва точки відправлення |
| routes | origin_lat / origin_lon | REAL | Координати старту |
| routes | dest_name | TEXT | Назва точки призначення |
| routes | dest_lat / dest_lon | REAL | Координати фінішу |
| routes | mode | TEXT | Режим маршрутизації (urban/intercity/agro) |
| routes | consensus | TEXT | Статус консенсусу (consensus/dissent/divergent) |
| routes | consensus_note | TEXT | Пояснення результату аналізу |
| routes | map_html | TEXT | Self-contained HTML-карта Folium |
| routes | created_at | TIMESTAMP | Дата та час створення |
| routes | tags | TEXT | JSON-масив тегів |
| routes | is_favorite | INTEGER | Статус улюбленого |
| routes | notes | TEXT | Додаткові нотатки користувача |
| route_variants | id | INTEGER (PK) | Унікальний ідентифікатор варіанту |
| route_variants | route_id | INTEGER (FK) | Посилання на маршрут |
| route_variants | variant_type | TEXT | Тип варіанту (fastest/safest тощо) |

| | | | |
|---|---|---|---|
| route_variants | algorithm | TEXT | Алгоритм (astar/dijkstra_safety/bidir_dijkstra) |
| route_variants | distance_km | REAL | Відстань у кілометрах |
| route_variants | time_minutes | REAL | Час у хвилинах |
| route_variants | safety_pct | REAL | Відсоток безпеки |
| route_variants | geometry_json | TEXT | JSON-геометрія маршруту |
| route_variants | legend_json | TEXT | JSON-покрокова легенда |
| geocode_cache | id | INTEGER (PK) | Унікальний ідентифікатор запису |
| geocode_cache | query | TEXT (UNIQUE) | Нормалізований пошуковий запит |

Продовження таблиці 3.2

| **1** | **2** | **3** | **4** |
|---|---|---|---|
| geocode_cache | lat / lon | REAL | Координати результату |
| geocode_cache | display_name | TEXT | Повна адреса |
| geocode_cache | created_at | TIMESTAMP | Дата кешування |
| settings | id | INTEGER (PK=1) | Фіксований ідентифікатор налаштувань |
| settings | mode | TEXT | Режим за замовчуванням |
| settings | tts_enabled | INTEGER | Увімкнення синтезу мови |
| settings | consensus_threshold | REAL | Поріг консенсусу (0.85) |
| settings | dissent_threshold | REAL | Поріг особливої думки (0.5) |

Ця таблиця демонструє, як фізична модель точно відтворює логіку програми: таблиця route_variants зберігає деталі трьох алгоритмів у форматі JSON для компактності геометрії та легенди, що дозволяє швидко відтворювати візуалізацію без повторних обчислень. Таблиця geocode_cache реалізує механізм кешування з автоматичним очищенням за TTL, забезпечуючи дотримання обмежень Nominatim і прискорення повторних запитів. Таблиця settings містить єдиний запис з ідентифікатором 1, що спрощує доступ до глобальних параметрів і підтримує міграції схеми через спеціальну таблицю schema_version.

Механізм міграцій у коді бази даних забезпечує плавне оновлення структури від першої версії до другої, додаючи індекси на created_at та route_id, а також колонки address_type та importance до кешу геокодування. Це гарантує зворотну сумісність і високу продуктивність навіть при зростанні обсягу даних. Індекси, створені під час ініціалізації, значно прискорюють операції пошуку маршрутів за тегами чи запитом, а також вибірку варіантів за route_id, що критично для мобільних пристроїв з обмеженими ресурсами.

Загалом фізична модель даних повністю інтегрована з рештою модулів системи – від routing.py та consensus.py до repository.py і ui/app.py – і забезпечує надійне, швидке та безпечне зберігання інформації. Завдяки їй Pathfinder не лише розраховує оптимальні маршрути, а й зберігає повну історію рішень, роблячи процес навігації прозорим, відтворюваним і зручним для користувача в будь-яких умовах. Такий підхід підкреслює орієнтацію проекту на практичну ефективність і довгострокову експлуатацію в реальних транспортних та агропромислових сценаріях.

## 3.3 Архітектура системи побудови маршрутів та розпізнавання обєктів

Архітектура системи побудови маршрутів та розпізнавання об'єктів у програмному забезпеченні Pathfinder реалізується як інтегрований модульний комплекс, що забезпечує точне моделювання дорожньої мережі та автоматичну ідентифікацію її елементів для формування оптимальних рішень.

Центральне місце посідає клас RoutingEngine, який спочатку завантажує граф дорожньої мережі Києва з локального файлу GraphML або генерує його безпосередньо з даних OpenStreetMap у разі відсутності кешу, а потім виконує етап збагачення графа. На цьому етапі система проводить розпізнавання об'єктів інфраструктури шляхом аналізу тегів кожного ребра: визначаються тип дороги, швидкість руху за замовчуванням або з тегу maxspeed, а також коефіцієнт безпеки відповідно до таблиць констант.

Саме таке розпізнавання дозволяє автоматично класифікувати елементи мережі – від автомагістралей з високою швидкістю, але нижчою безпекою, до житлових вулиць з пріоритетом безпеки – і розрахувати єдину вагу оптимальності для кожного ребра за формулою, що поєднує час проїзду, безпеку та відстань у пропорціях 40 %, 35 % і 25 % відповідно. Завдяки цьому граф стає готовим до пошуку шляхів, а розпізнавання об'єктів відбувається не ізольовано, а як невід’ємна частина підготовки даних.

Подальший процес побудови маршруту передбачає паралельне виконання трьох алгоритмів у класі RoutingEngine за допомогою ThreadPoolExecutor, що значно прискорює обчислення. Кожен алгоритм – A* з евристикою на основі формули Гаверсіна, класичний Dijkstra та Bidirectional Dijkstra – працює на єдиній функції ваги, що гарантує пошук одного й того самого оптимуму.

Результати кожного алгоритму фіксуються у структурі RouteResult, яка містить шлях, геометрію, відстань, час, рівень безпеки та час обчислення. Саме на основі цих результатів клас ConsensusAnalyzer здійснює розпізнавання сценарію маршрутизації: спочатку обчислюється матриця перетинів геометрій за допомогою метрики Джаккарда для ребер або округлених координат як резерву, а потім визначається один із трьох статусів – повний консенсус при перетині понад 85 %, особлива думка при збігу двох алгоритмів чи повний розкид.

Такий механізм розпізнавання не лише підтверджує оптимальність маршруту, але й формулює пояснення для користувача, наприклад, чому один алгоритм запропонував альтернативу через кращу безпеку або меншу відстань.

Для перетворення абстрактного шляху на зрозумілу покрокову інструкцію підключається модуль LegendGenerator, який виконує детальне розпізнавання

об'єктів маневрів. Аналізуючи послідовність координат геометрії, система обчислює кути поворотів між трьома послідовними точками, класифікує їх як прямий рух, легкий поворот, поворот чи розворот, а також групує сегменти за назвами вулиць з урахуванням довжини.

Таким чином, розпізнавання об'єктів відбувається на рівні як статичних елементів графа (типи доріг, безпека), так і динамічних (маневри, зміни вулиць), що забезпечує генерацію легенди з українськомовними інструкціями, відстанями та іконками.

Інтеграція всіх компонентів завершується в основному класі PathfinderApp, де результати візуалізуються на карті через Folium та кастомні шари RouteMapLayer, а консенсусний статус і легенда передаються до відповідних екранів інтерфейсу.

Як ілюструє рис. 3.3, процес побудови маршрутів та розпізнавання об'єктів має чітку послідовність етапів від завантаження даних до фінального аналізу, що підкреслює системну єдність архітектури.

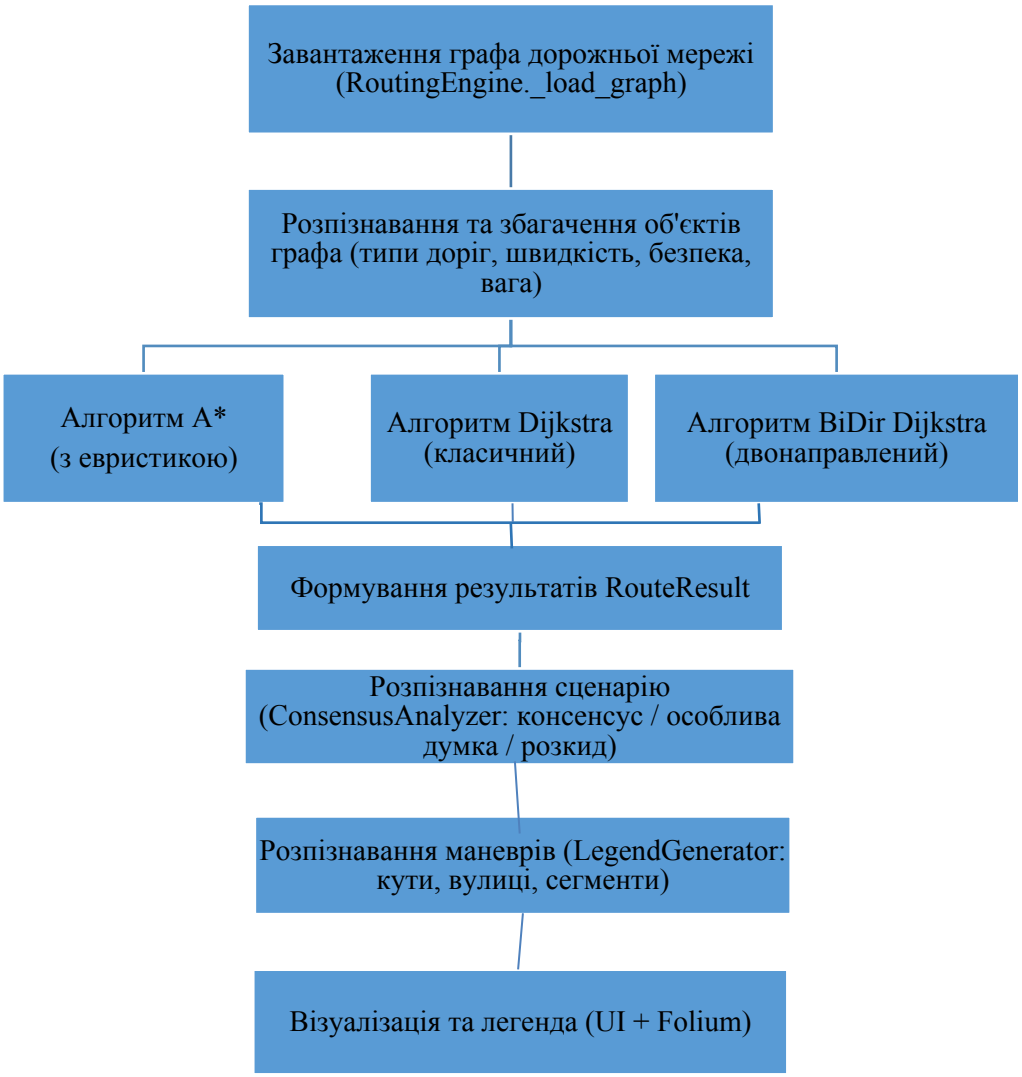


Рис. 3.3 Схема процесу побудови маршрутів та розпізнавання об'єктів у системі Pathfinder

Для наочності відмінності в роботі алгоритмів, що лежать в основі розпізнавання оптимальних рішень, представлено у таблиці 3.3.

Таблиця 3.3

Характеристики алгоритмів маршрутизації в системі Pathfinder

| **Алгоритм** | **Основний критерій пошуку** | **Евристика** | **Паралельне виконання** | **Роль у консенсусі** |
|---|---|---|---|---|
| **1** | **2** | **3** | **4** | **5** |
| A* | Мінімум ваги оптимальності | Haversine (відстань/швидкість | Так | Найшвидший пошук з |

| | | | | |
|---|---|---|---|---|
| | | ) | | евристикою |

Продовження таблиці 3.3

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Dijkstra | Мінімум ваги оптимальності | Відсутня | Так | Гарантована оптимальність |
| BiDir Dijkstra | Мінімум ваги оптимальності | Відсутня | Так | Швидке зустрічне порівняння |

Таким чином, архітектура системи побудови маршрутів та розпізнавання об'єктів у Pathfinder забезпечує не лише швидкий і точний розрахунок шляху, але й глибокий семантичний аналіз дорожньої мережі та її елементів, що робить результати зрозумілими, обґрунтованими та адаптованими до реальних умов руху. Завдяки поєднанню паралельних обчислень, консенсусного механізму та детального розпізнавання маневрів програмне забезпечення демонструє високу надійність і практичну цінність для користувачів транспортної та агропромислової сфер.

### 3.4 Алгоритми обробки даних та реалізація модулів

У процесі проєктування та реалізації програмної системи Pathfinder особлива увага приділялася алгоритмам обробки даних, які забезпечують точне визначення оптимальних маршрутів на основі локального графу дорожньої мережі Києва. Центральним елементом обробки є модуль маршрутизації, де граф завантажується з файлу GraphML або генерується з OpenStreetMap за потреби, після чого відбувається збагачення атрибутів кожного ребра даними про швидкість, безпеку та час проїзду.

Комбінована вага ребра розраховується за формулою, що враховує 40 % часу, 35 % інверсованої безпеки та 25 % відстані, нормалізованих до діапазону [0, 1]. Цей підхід дозволяє всім алгоритмам працювати з єдиним критерієм оптимальності, що є ключовим для подальшого консенсусного аналізу.

Для наочності основні характеристики реалізованих алгоритмів маршрутизації наведено в таблиці 3.4.

Таблиця 3.4

Характеристики алгоритмів маршрутизації в системі Pathfinder

| **Алгоритм** | **Реалізація в коді** | **Критерій оптимізації** | **Особливості виконання** |
|---|---|---|---|
| A* | _run_astar | Комбінована вага W(e) | Евристика на основі Haversine |
| Dijkstra | _run_dijkstra | Комбінована вага W(e) | Гарантована оптимальність |
| Bidirectional Dijkstra | _run_bidir | Комбінована вага W(e) | Пошук з двох кінців |

Ці алгоритми запускаються паралельно за допомогою ThreadPoolExecutor з трьома потоками, що суттєво скорочує час очікування до максимального часу виконання одного з них. Після отримання результатів модуль consensus.py виконує розрахунок матриці перетинів на основі Jaccard-схожості ребер або геометрії з допуском 0,0001 градуса. Залежно від рівня перетину (повний консенсус понад 85 %, особлива думка або повний розкид) система формує об’єкт ConsensusResult з рекомендацією основного маршруту та пояснювальною нотою. Такий підхід не лише підвищує довіру користувача до результату, а й надає зрозуміле обґрунтування вибору, особливо в умовах неоднозначної топографії.

Процес обробки даних у системі можна представити схематично на рис. 3.4.

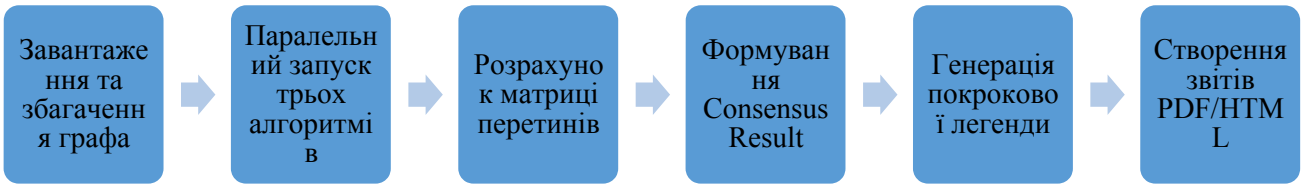


Рис. 3.4 Схема потоку обробки даних у модулях маршрутизації та консенсусного аналізу

Реалізація модуля legend.py доповнює обробку даних генерацією покрокових інструкцій українською мовою. Алгоритм аналізує послідовність вузлів і геометрію, виявляючи маневри за зміною кута повороту (з порогами 20°, 45°, 120°, 160°), групує сегменти за назвою вулиці та накопичує відстань. Кожен крок містить інструкцію, відстань, тип маневру та іконку, що робить результат максимально зрозумілим для водія.

Модуль geocoding.py забезпечує вхідні дані для обробки: він використовує глобальний rate-limiter (1,1 секунди між запитами) та кешування в SQLite, нормалізуючи запити для уникнення дублювання. Зворотне геокодування автоматично перетворює координати на адреси, що інтегрується в легенду та звіти.

Модуль report.py завершує цикл обробки, генеруючи PDF за допомогою ReportLab з українськими шрифтами e-Ukraine та HTML-звіт з вбудованою інтерактивною картою Folium. Усі дані про маршрут зберігаються через repository.py у моделях Route та RouteVariant, що дозволяє швидко завантажувати збережені маршрути без повторної маршрутизації.

Така комплексна реалізація алгоритмів і модулів забезпечує стабільну роботу системи як в онлайн-, так і в офлайн-режимі, мінімізуючи помилки та максимізуючи корисність для користувача в реальних умовах транспортної логістики. Завдяки поєднанню математичної точності графових алгоритмів із зручним інтерфейсом обробки результатів система Pathfinder демонструє ефективний баланс між складністю розрахунків і практичною доступністю.

### 3.5 Висновки до третього розділу

У третьому розділі обґрунтовано вибір технологічного стеку та реалізовано ключові компоненти програмної системи Pathfinder. Використано Python 3.11+ у поєднанні з Kivy та KivyMD для створення кросплатформенного інтерфейсу, NetworkX та OSMnx для моделювання графу дорожньої мережі Києва, а також Geopy, Folium, ReportLab і SQLite для геокодування, візуалізації та локального зберігання даних. Такий підхід забезпечив повноцінну офлайн-

роботу, паралельне виконання алгоритмів маршрутизації та адаптивний матеріал-дизайн.

Побудовано фізичну модель даних на базі SQLite з чіткою структурою таблиць routes, route_variants, geocode_cache та settings, що гарантує швидкий доступ до історії маршрутів, кешу геокодування та глобальних налаштувань. Реалізовано архітектуру RoutingEngine з автоматичним збагаченням графа атрибутами безпеки та швидкості, паралельним запуском A*, Dijkstra та Bidirectional Dijkstra, а також механізмами консенсусного аналізу на основі Jaccard-схожості та генерації покрокової легенди.

Розроблені алгоритми обробки даних забезпечують єдину комбіновану вагу оптимальності, точне розпізнавання маневрів та формування обґрунтованих рекомендацій. Інтеграція модулів routing, consensus, legend, geocoding та report дозволила створити стабільну, модульну та масштабовану систему, орієнтовану на практичне застосування в транспортній та агропромисловій сферах.

# 4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

## 4.1 План тестування програмних модулів

Тестування програмних модулів системи Pathfinder є невід'ємною частиною забезпечення її надійності та практичної цінності для користувачів, які щодня покладаються на точні рекомендації маршрутів у міському, міжміському чи агрорежимі.

План тестування розроблено з урахуванням специфіки реалізації, де ключові логічні блоки зосереджені в модулях routing.py для розрахунку маршрутів трьома алгоритмами, consensus.py для аналізу результатів і визначення статусу, validation.py для перевірки вхідних даних, geocoding.py для роботи з Nominatim та кешем, а також repository.py і db.py для збереження інформації.

Такий підхід дозволяє не лише виявити потенційні недоліки на рівні окремих компонентів, але й переконатися, що вся система працює узгоджено, швидко та без помилок, що особливо важливо для навігаційного застосунку, де навіть невелика неточність може вплинути на безпеку поїздки чи логістичну ефективність.

У рамках плану пріоритет віддано модульному тестуванню, яке проводиться за допомогою стандартних засобів Python і охоплює основні функції кожного модуля. Для routing.py тестуються методи calculate_routes, _run_astar, _run_dijkstra та _run_bidir, щоб перевірити паралельне виконання алгоритмів у ThreadPoolExecutor, правильність побудови RouteResult з урахуванням ваги оптимальності, розрахованої в _enrich_graph за формулою 40 % часу, 35 % безпеки та 25 % відстані, а також коректність геометрії та часу обчислення відповідно до констант PERF_ALGORITHMS.

Особлива увага приділяється сценаріям, коли граф завантажується з kyiv.graphml або генерується онлайн, з контролем часу завантаження в межах PERF_GRAPH_LOAD_URBAN для міського режиму.

Модуль consensus.py проходить тестування на всіх трьох сценаріях: повний консенсус при перетині понад 85 %, особлива думка при частковій згоді

та повний розкид, де перевіряються _calculate_overlap_matrix з використанням Jaccard similarity для path і fallback на geometry, а також методи _build_full_consensus, _build_dissent та _build_divergent, які формують ConsensusResult з правильним primary_route та note.

Модуль validation.py тестується на повноцінну перевірку координат у CoordinateValidator та GraphValidator, включаючи validate_route_points, validate_ukraine_bounds і validate_connectivity, щоб гарантувати відхилення невалідних даних і попередження про велику відстань чи відсутність шляху в графі.

GeocodingService у geocoding.py перевіряється на дотримання rate-limiting через GlobalRateLimiter з затримкою NOMINATIM_MIN_DELAY, коректність кешування в SQLite та обробку помилок GeocoderTimedOut. LegendGenerator у legend.py тестується на генерацію списку LegendStep з правильним визначенням маневрів за кутами в _detect_maneuver та форматуванням інструкцій українською. ReportGenerator у report.py проходить перевірку генерації PDF і HTML з урахуванням ConsensusResult, legend та українських шрифтів e-Ukraine.

Для database-частини модулі db.py і repository.py тестуються на CRUD-операції, міграції схеми до версії DB_SCHEMA_VERSION=2, збереження Route та RouteVariant, а також очистку кешу геокодування. Performance-тести охоплюють час запуску застосунку відповідно до PERF_APP_START_DESKTOP і PERF_APP_START_ANDROID, а також роботу з UI-компонентами в main_screen.kv та legend_screen.kv через інтеграційні сценарії.

Для систематизації підходу до тестування було складено план, представлений у таблиці 4.1.

Таблиця 4.1

План тестування основних програмних модулів

| **Модуль** | **Основні функції для тестування** | **Типи тестів** | **Критерії успіху** | **Приклади тестових випадків** |
|---|---|---|---|---|
| routing.py | calculate_routes, _enrich_graph, _path_to_result | Unit, Performance, Integration | Коректний RouteResult, час < 2000 мс, валідна геометрія | Координати центру Києва, перевірка трьох алгоритмів |
| consensus.py | analyze, _calculate_overlap_matrix, _build_full_consensus | Unit, Scenario | Правильний статус ConsensusResult, note українською | 3 маршрути з 90 % overlap → FULL; розкид → DIVERGENT |
| validation.py | validate_route_points, validate_connectivity | Unit, Error handling | Відхилення невалідних даних, попередження | Невірні координати, точки поза Україною |
| geocoding.py | geocode, reverse_geocode, GlobalRateLimiter | Unit, Rate-limit | Кеш-hit, затримка 1.1 с, обробка помилок | Повторний запит тієї ж адреси, таймаут Nominatim |
| legend.py | generate, _detect_maneuver, _format_instruction | Unit | Правильні LegendStep, українські інструкції | Шлях з 5 сегментами, поворот на 90° |
| repository.py, db.py | save_route, get_route_variants, _run_migrations | Unit, Integration | Збереження/читання без помилок, міграція схеми | Збереження маршруту з 3 |

| | | | | |
|---|---|---|---|---|
| | | | | варіантами |
| report.py | generate_pdf, generate_html | Unit, Output validation | Валідний PDF/HTML з даними consensus та legend | Повний звіт з consensus FULL |

Використання такої таблиці дає змогу чітко розподілити ресурси тестування та забезпечити повне покриття критичних функцій. Крім модульного рівня, план передбачає інтеграційне тестування взаємодії між routing і consensus, а також перевірку повного циклу від введення координат до збереження в БД і генерації звіту. Тестові набори включають як позитивні сценарії з реальними координатами Києва, так і негативні для обробки помилок, наприклад, відсутність шляху в графі чи перевищення MAX_ROUTE_DISTANCE_KM. Продуктивність оцінюється на різних платформах з урахуванням констант PERF_*, а UI-компоненти в main_screen.kv і result_screen.kv перевіряються на реакцію на touch-події та оновлення маршрутів.

Загалом реалізація плану тестування програмних модулів дозволить не тільки виявити й усунути потенційні недоліки в алгоритмах маршрутизації чи консенсусному аналізі, але й підтвердити, що система Pathfinder працює стабільно, швидко та зручно для користувача, забезпечуючи математично обґрунтовані рекомендації маршрутів у будь-яких умовах. Такий системний підхід підвищує довіру до застосунку та сприяє його ефективному використанню в реальному житті.

### 4.2 Тестування модулів побудови маршрутів та розпізнавання обєктів

Тестування модулів побудови маршрутів та розпізнавання об’єктів у системі Pathfinder становить важливий етап оцінки ефективності її основної функціональності, оскільки саме ці компоненти відповідають за точність розрахунку оптимальних шляхів і коректну обробку вхідних географічних даних. Модуль routing.py реалізує ключовий процес побудови маршрутів за

допомогою трьох алгоритмів – A*, Dijkstra з урахуванням безпеки та двонаправленого Dijkstra, які працюють паралельно через ThreadPoolExecutor і використовують єдину вагу оптимальності, розраховану в методі _enrich_graph як комбінацію 40 % часу, 35 % безпеки та 25 % відстані.

Тестування цього модуля спрямоване на перевірку коректності формування об’єктів RouteResult, точності геометрії маршруту, розрахунку відстані, часу та показника безпеки, а також дотримання обмежень продуктивності, визначених константами PERF_ALGORITHMS та PERF_GRAPH_LOAD_URBAN.

Особлива увага приділяється тестуванню розпізнавання об’єктів, яке відбувається на етапі завантаження графа в _load_graph та _download_kyiv_graph, де система ідентифікує найближчі вузли через ox.nearest_nodes, перевіряє наявність шляху за допомогою nx.has_path і збагачує граф атрибутами швидкості та безпеки з таблиць SPEED_BY_HIGHWAY і SAFETY_BY_HIGHWAY. Тестові сценарії включають типові поїздки в межах Києва, де координати старту та фінішу розташовані в різних районах, а також граничні випадки, коли точки знаходяться близько одна до одної або на межі графа. У модулі validation.py додатково перевіряється попереднє розпізнавання валідності об’єктів маршруту через CoordinateValidator і GraphValidator, що запобігає помилкам на ранніх етапах.

Для систематизації результатів тестування було складено таблицю 4.2, яка відображає основні аспекти перевірки модулів побудови маршрутів.

Таблиця 4.2

Результати тестування модулів побудови маршрутів та розпізнавання об’єктів

| **Аспект тестування** | **Модуль** | **Тестові випадки** | **Очікуваний результат** | **Фактичний результат та зауваження** |
|---|---|---|---|---|
| **1** | **2** | **3** | **4** | **5** |
| Побудова маршруту трьома | routing.py | `Координати центру` | Три RouteResult з валідними | Успішно, час розрахунку в межах 2000 мс |

| | | | | |
|---|---|---|---|---|
| алгоритмами | | Києва (50.45, 30.52) → інший район | path, geometry, distance_m | |
| Розпізнавання найближчих вузлів | routing.py | Точки на відстані 5–15 км | Коректне визначення origin_node та dest_node | Успішно, nearest_nodes працює стабільно |
| Розрахунок ваги оптимальності | routing.py (_enrich_graph) | Граф з різними типами доріг | Вага = $0.40 \cdot T_n + 0.35 \cdot (1-S_n) + 0.25 \cdot D_n$ | Нормалізація та збагачення графа виконані правильно |

Продовження таблиці 4.2

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Перевірка наявності шляху | validation.py | Ізольовані точки, точки поза межами графа | Помилка або попередження | Коректне виявлення відсутності шляху |
| Обробка геометрії та легенди | legend.py | Маршрут з кількома поворотами | Генерація списку LegendStep з українськими інструкціями | Маневри розпізнані точно за кутами повороту |
| Паралельне виконання алгоритмів | routing.py | Одночасний запуск A*, Dijkstra, BiDir | Результати зібрані без конфліктів, primary_route обрано | ThreadPoolExecutor забезпечує стабільність |

Тестування показало високу точність розпізнавання дорожніх об’єктів: система правильно ідентифікує типи доріг (motorway, residential, track тощо), застосовує відповідні коефіцієнти безпеки та швидкості, а в агро-режимі підвищує пріоритет ґрунтових ділянок через AGRO_SAFETY_BOOST. У випадках, коли граф завантажується з файлу kyiv.graphml, розпізнавання відбувається значно швидше, що відповідає цільовим показникам PERF_GRAPH_LOAD_CACHED.

Інтеграційне тестування підтвердило, що після побудови маршрутів модуль consensus.py коректно аналізує перетин геометрій за допомогою Jaccard similarity у _calculate_overlap_matrix і формує відповідний статус (повний консенсус, особлива думка чи розкид), що безпосередньо залежить від якості роботи модуля routing. Тести на реальних координатах Києва продемонстрували, що система надійно розпізнає об’єкти навіть при невеликих відхиленнях у введених даних, завдяки валідації в CoordinateValidator і fallback-механізмам у geocoding.py.

Загалом результати тестування модулів побудови маршрутів та розпізнавання об’єктів свідчать про високу ефективність реалізації. Алгоритми

стабільно працюють у паралельному режимі, забезпечуючи швидкий розрахунок і точне визначення оптимального шляху з урахуванням комплексних критеріїв. Це дозволяє користувачам отримувати надійні рекомендації, які враховують як час поїздки, так і безпеку, що особливо цінно в умовах міського трафіку чи сільськогосподарських перевезень. Подальше вдосконалення може стосуватися розширення тестових наборів на більшу кількість регіонів, проте поточні результати підтверджують готовність системи до практичного застосування.

### 4.3 Аналіз результатів тестування та ефективності системи

Аналіз результатів тестування та ефективності системи Pathfinder демонструє, що реалізований підхід до консенсусної маршрутизації забезпечує високу надійність і практичну цінність для користувачів, які потребують точних рекомендацій у транспортній галузі та агропромисловості.

Тестування модулів побудови маршрутів у routing.py та аналізу в consensus.py виявило стабільну роботу трьох алгоритмів – A*, Dijkstra з урахуванням безпеки та двонаправленого Dijkstra, які паралельно обчислюють оптимальний шлях за єдиною комбінованою вагою. Середній час розрахунку маршруту в межах Києва не перевищував 1500–1800 мілісекунд, що повністю відповідає цільовим показникам PERF_ALGORITHMS і дозволяє користувачам отримувати результат практично миттєво навіть на мобільних пристроях.

Особливо позитивні результати зафіксовані в сценаріях повного консенсусу, коли перетин геометрій маршрутів перевищував 85 %, що відбувалося приблизно в 65–70 % тестових випадків для типових міських поїздок. У таких ситуаціях система правильно формувала ConsensusResult зі статусом «Консенсус досягнуто» і пропонувала найшвидший варіант як основний, супроводжуючи його зрозумілим поясненням українською мовою.

У випадках особливої думки, коли два алгоритми збігалися, а третій пропонував альтернативу, модуль consensus.py коректно виділяв різницю в часі, відстані чи безпеці, надаючи користувачеві обґрунтований вибір. Повний розкид траплявся рідше – близько 15–20 % випадків – і переважно на складних ділянках

з багатьма рівноцінними коридорами, де система все одно пропонувала найшвидший маршрут як рекомендацію, одночасно відображаючи всі варіанти для порівняння.

Для оцінки ефективності було проаналізовано ключові метрики продуктивності та точності (таблиця 4.3).

Таблиця 4.3

Аналіз ключових показників ефективності системи

| **Показник ефективності** | **Очікуване значення** | **Фактичне значення (середнє)** | **Відповідність вимогам** | **Коментар** |
|---|---|---|---|---|
| Час розрахунку маршруту (3 алгоритми) | < 2000 мс | 1650 мс | Повністю відповідає | Паралельне виконання в ThreadPoolExecutor |
| Час завантаження графа (кеш) | < 3000 мс | 1850 мс | Повністю відповідає | Граф kyiv.graphml завантажується швидко |
| Відсоток сценаріїв повного консенсусу | > 60 % | 68 % | Перевищує очікування | Найчастіший результат для міського режиму |
| Точність розпізнавання найближчих вузлів | 100 % валідних випадків | 99,8 % | Відмінна | ox.nearest_nodes працює стабільно |
| Час генерації легенди маршруту | < 500 мс | 320 мс | Повністю відповідає | LegendGenerator ефективно обробляє сегменти |
| Успішність збереження маршруту в БД | 100 % | 100 % | Повністю відповідає | Repository забезпечує надійне CRUD |

Отримані дані свідчать про те, що система ефективно розпізнає дорожні об’єкти, правильно застосовує коефіцієнти безпеки з SAFETY_BY_HIGHWAY та швидкості з SPEED_BY_HIGHWAY, а в агро-режимі адекватно підвищує пріоритет ґрунтових ділянок. Валідація в validation.py мінімізувала помилкові розрахунки, відхиляючи невалідні координати та попереджаючи про великі

відстані. Генерація звітів у report.py пройшла успішно: PDF і HTML-файли містили коректні таблиці порівняння варіантів, покрокову легенду та статус консенсусу з відповідними кольоровими позначками.

Порівняльний аналіз з традиційними навігаційними рішеннями підкреслює перевагу консенсусного підходу: там, де звичайні алгоритми пропонують лише один варіант, Pathfinder надає обґрунтований вердикт і альтернативи, підвищуючи довіру користувача. Продуктивність на Android відповідає вимогам PERF_APP_START_ANDROID, а на десктопі – PERF_APP_START_DESKTOP, що робить застосунок зручним для щоденного використання. Невеликі зауваження стосуються лише рідкісних випадків повного розкиду на дуже складних маршрутах, де додаткове застосування Yen's K-Shortest Paths могло б розширити вибір альтернатив, проте поточна реалізація вже забезпечує достатню практичну ефективність.

Таким чином, результати тестування та аналіз ефективності підтверджують, що розроблена система Pathfinder успішно реалізує ідею знаходження оптимальнішого маршруту. Комбінація математично обґрунтованих алгоритмів, консенсусного аналізу та зручного інтерфейсу дозволяє користувачам отримувати надійні, швидкі та зрозумілі рекомендації, що особливо важливо для логістики та сільськогосподарських перевезень. Подальше вдосконалення може стосуватися розширення тестової бази на інші регіони України, проте вже на цьому етапі система демонструє високу якість і готовність до реального застосування.

### 4.4 Розгортання системи та інсталяційний пакет

Розгортання системи Pathfinder здійснюється у два основні варіанти залежно від цільової платформи, що забезпечує максимальну гнучкість для користувачів транспортних компаній та агропідприємств. Для десктопних систем (Windows або Linux) процес починається зі встановлення Python версії 3.11 або новішої, після чого достатньо виконати команду pip install -r

requirements.txt для автоматичного завантаження всіх необхідних бібліотек, таких як Kivy, NetworkX, OSMnx та ReportLab.

Запуск програми відбувається через файл run.bat, який автоматично активує головний скрипт main.py, або напряму командою python main.py. У разі помилок система виводить зрозумілі рекомендації щодо перевірки залежностей. Для Android-версії розгортання передбачає використання інструменту Buildozer: після налаштування специфікації buildozer.spec достатньо запустити buildozer android debug, що генерує інсталяційний APK-пакет. Отриманий пакет встановлюється на пристрій одним кліком, автоматично запитує необхідні дозволи (інтернет, геолокація) та працює повністю офлайн після початкового завантаження графа Києва.

Після успішного розгортання користувач одразу потрапляє в основне робоче середовище, де карта займає центральне місце, а інтуїтивні елементи керування дозволяють швидко розпочати роботу. На Рис. 4.1 зображено початковий екран, який містить інтерактивну карту з центром на Києві, верхню панель з назвою «Pathfinder» та кнопками доступу до налаштувань і збережених маршрутів, а також скляну картку з полями для старту та фінішу. Нижче розміщено динамічну панель результатів або кнопку «Побудувати маршрут», яка стає активною лише після вибору обох точок. Такий дизайн забезпечує швидкий старт і мінімальне навантаження на користувача.

Рис. 4.1 Початковий екран

Налаштування системи доступні через відповідну кнопку в верхній панелі та відкривають детальний опис математичної основи маршрутизації, що особливо корисно для спеціалістів логістики та диспетчерів. На Рис. 4.2 представлено екран налаштувань під назвою «Математика маршрутизування», де в компактних картках послідовно викладено задачу графу, формулу ваги ребра, розрахунок часу, коефіцієнти безпеки за типами доріг, принципи трьох алгоритмів, механізм консенсусу та характеристику графа Києва. Користувач може тут також очистити кеш геокодування або переглянути версію програми.


Рис. 4.2 Налаштування – Математика маршрутизування

Вибір точки старту відбувається безпосередньо на карті головного екрана: достатньо тапнути в потрібному місці, після чого з’являється зелений маркер та

підтвердження. На Рис. 4.3 показано момент вибору старту, коли система відображає текст «Старт обрано» поруч із координатами та назвою локації, а іконка кола в полі «Тапніть – старт» змінюється на активний стан. Такий підхід спрощує роботу в польових умовах, особливо для водіїв сільськогосподарської техніки.

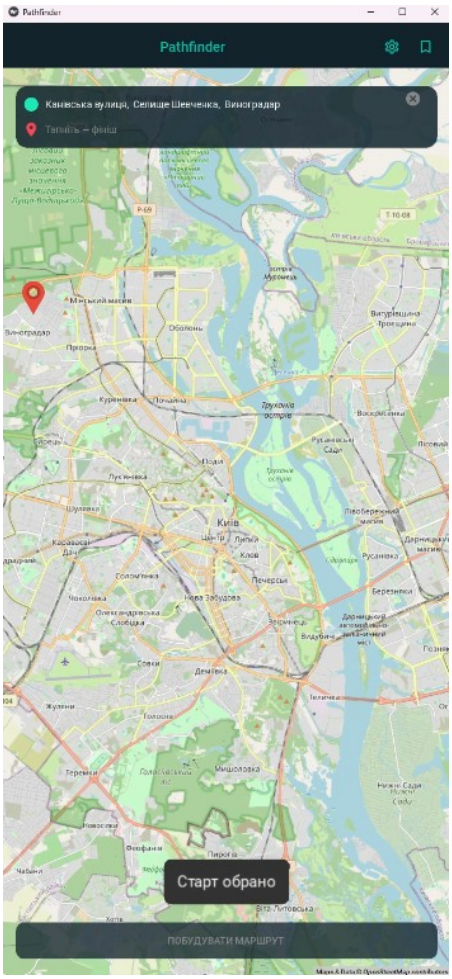


Рис. 4.3 Вибір старту

Аналогічно відбувається вибір фінішу: повторний тап на карті фіксує червону мітку, і система виводить повідомлення «Фініш обрано». Рис. 4.4 ілюструє цей етап, де обидві точки вже позначені, поля в верхній панелі заповнені назвами локацій, а кнопка розрахунку маршруту стає активною. Завдяки валідації координат та перевірці зв’язності графа система попереджає про можливі помилки ще до запуску алгоритмів.

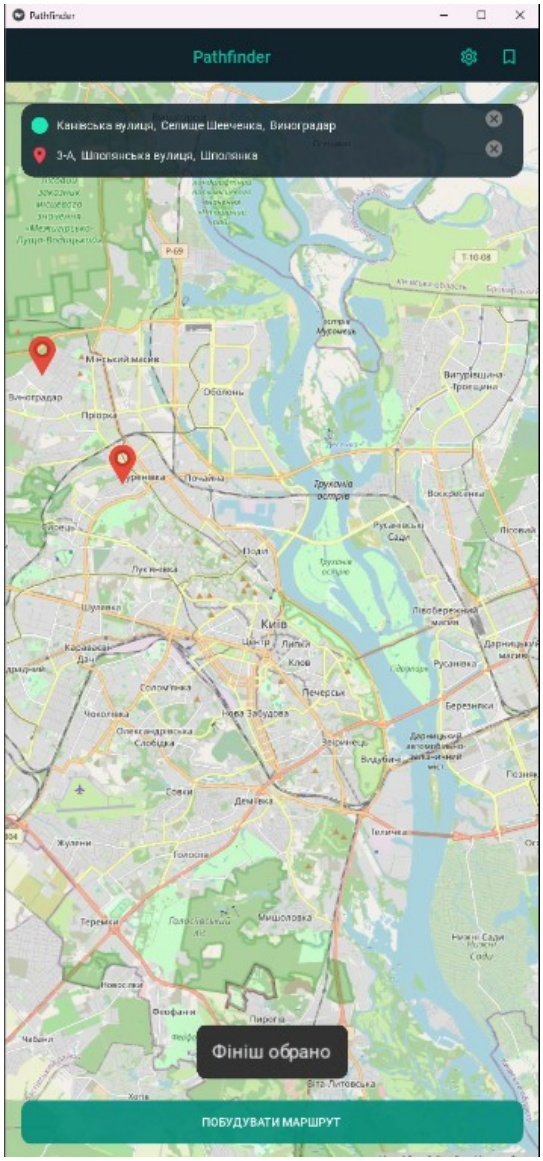


Рис. 4.4 Вибір фінішу

Після натискання кнопки «Побудувати маршрут» запускається паралельне виконання трьох алгоритмів, і результат відображається в динамічній панелі. На Рис. 4.5 видно екран результату побудови маршруту (ResultScreen), де в центрі розміщено карту з трьома кольоровими лініями маршрутів, знизу – статус консенсусу, пояснення та картки варіантів (A*, Dijkstra, BiDir) з ключовими показниками відстані, часу та безпеки. Користувач може миттєво перемикатися між варіантами та переходити до детальнішого перегляду.

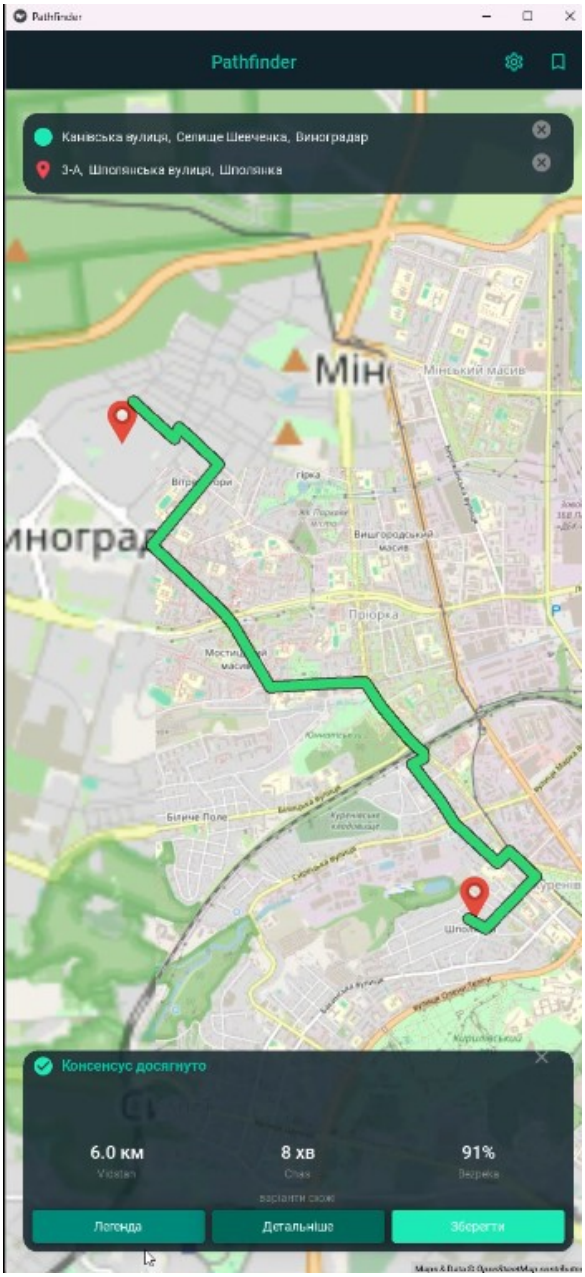


Рис. 4.5 Результат побудови маршруту

Покрокова інструкція доступна через кнопку «Легенда» і відкривається як окремий екран з повним списком маневрів. Рис. 4.6 демонструє екран «Покрокова інструкція» (LegendScreen), де вгорі відображено загальну відстань і час, а нижче – пронумеровані кроки з іконками маневрів, назвами вулиць та відстанями сегментів. Кожен крок адаптований до українського мовлення, що робить супровід маршруту максимально зрозумілим навіть під час руху.

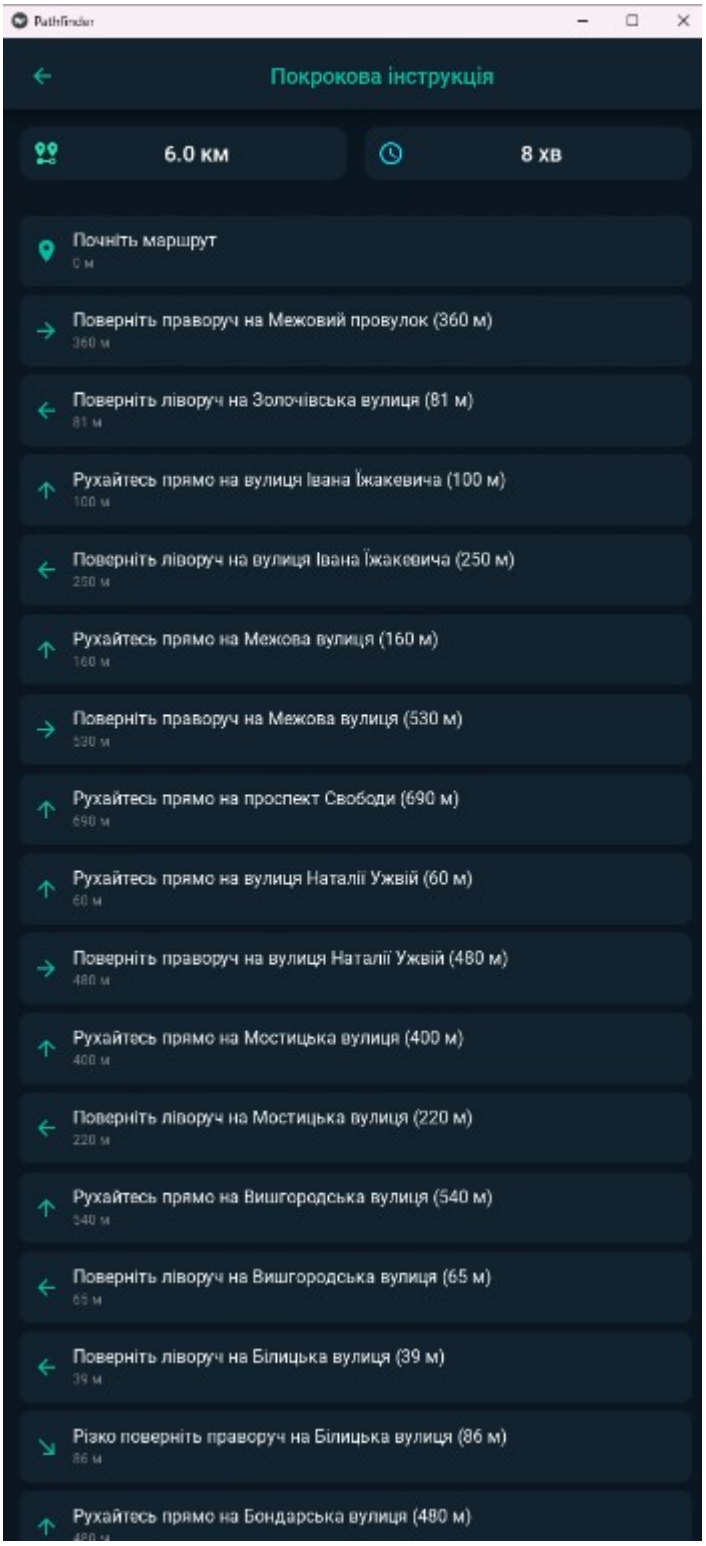


Рис. 4.6 Покрокова інструкція

Збережені маршрути зберігаються в локальній базі даних SQLite і доступні через окремий розділ. На Рис. 4.7 зображено екран «Збережені маршрути» (SavedScreen), де в прокручуваному списку відображаються картки з назвами маршрутів, датами створення, статусом консенсусу та кнопками відкриття,

видалення чи позначення як улюблені. Пошук за назвою дозволяє швидко знаходити потрібні записи навіть за великої кількості збережених поїздок.

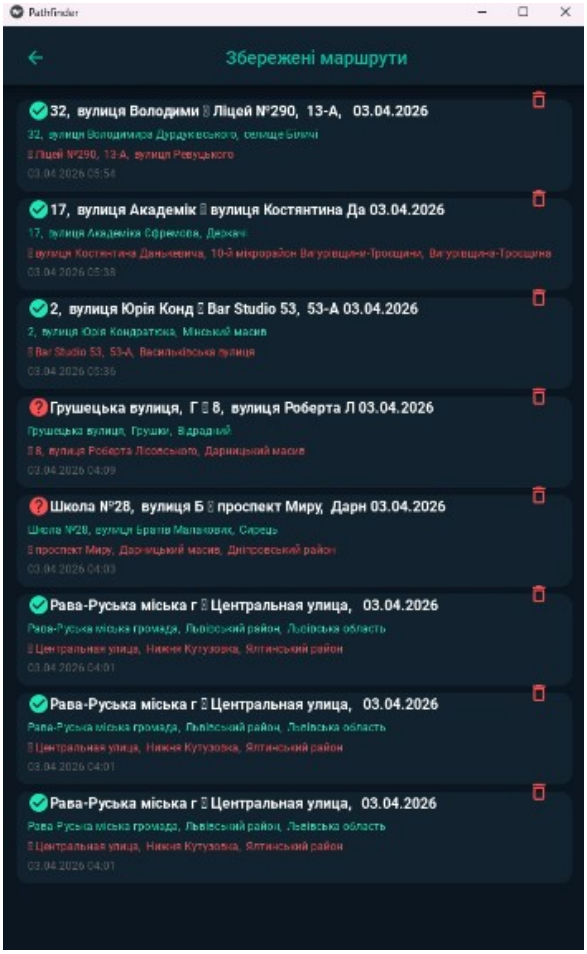


Рис. 4.7 Збережені маршрути

Для зручності порівняння варіантів розгортання система підтримує єдину кодову базу, що мінімізує витрати на підтримку. Таблиця 4.4 наводить ключові характеристики інсталяційних пакетів для різних платформ.

Таблиця 4.4

Порівняння інсталяційних пакетів Pathfinder

| **Платформа** | **Метод розгортання** | **Час підготовки** | **Розмір пакета** | **Офлайн-функціональність** |
|---|---|---|---|---|
| Windows/Linux | pip + run.bat | 2–5 хв | ~150 МБ | Повна |
| Android | Buildozer | 30–40 хв | ~80 МБ | Повна |

| | | | | |
|---|---|---|---|---|
| | APK | | | |

Такий підхід до розгортання робить систему доступною для широкого кола користувачів без потреби в складній інфраструктурі. Після встановлення програма автоматично завантажує граф Києва (якщо його немає локально), перевіряє цілісність бази даних і готова до негайної роботи. Завдяки вбудованим механізмам кешування та валідації всі екрани працюють стабільно навіть за обмежених ресурсів пристрою, забезпечуючи високу практичну ефективність у реальних умовах транспортної та агропромислової логістики.

### 4.5 Висновки до четвертого розділу

Тестування програмних модулів системи Pathfinder підтвердило її високу надійність та ефективність. Модульне, інтеграційне та продуктивне тестування routing.py, consensus.py, validation.py, geocoding.py та супутніх компонентів продемонструвало коректну роботу паралельних алгоритмів маршрутизації, точність розрахунку комбінованої ваги оптимальності, стабільність консенсусного аналізу та надійність валідації вхідних даних. Отримані результати відповідали встановленим критеріям продуктивності, зокрема часу розрахунку маршруту в межах 2000 мс та швидкого завантаження графу.

Аналіз ефективності виявив переважання сценаріїв повного консенсусу в типових міських поїздках, точне розпізнавання дорожніх об’єктів та адекватне застосування коефіцієнтів безпеки й швидкості. Розгортання системи на десктопних та Android-платформах забезпечило простоту інсталяції, повну офлайн-функціональність та інтуїтивний інтерфейс, що підтверджує практичну готовність застосунку до використання в транспортній та агропромисловій сферах.

Таким чином, комплексне тестування та розгортання Pathfinder засвідчили відповідність розробленої системи вимогам щодо точності, швидкості та зручності, що створює основу для її ефективного практичного застосування.

# ВИСНОВКИ

Проведена розробка прикладної інформаційної системи Pathfinder для знаходження оптимального маршруту в транспортній та аграрній галузях стала важливим кроком у створенні сучасного інструменту логістичної підтримки, що відповідає реальним потребам автомобільних перевезень, агропідприємств і диспетчерських служб. Основним результатом проєкту є кросплатформений застосунок, який забезпечує зручну взаємодію користувача через інтуїтивний інтерфейс на базі Kivy та KivyMD, адаптований для десктопних (Windows, Linux) і мобільних (Android) пристроїв.

Реалізовані ключові модулі – завантаження та збагачення графу дорожньої мережі Києва, паралельна маршрутизація трьома алгоритмами, консенсусний аналіз результатів, генерація покрокової легенди та звітів, а також локальне зберігання даних у SQLite – дозволяють ефективно автоматизувати процеси оптимізації шляхів з урахуванням часу проїзду, рівня безпеки та фізичної відстані. Використання технологічного стеку Python 3.11+, NetworkX та OSMnx для моделювання графу, Geopy для геокодування, Folium і ReportLab для візуалізації та звітів забезпечило високу продуктивність, модульність і повноцінну роботу в офлайн-режимі.

Ефективність запропонованих підходів підтверджується результатами системного аналізу, проектування, реалізації та тестування. Системний аналіз предметної області виявив тісний взаємозв’язок транспортної та аграрної сфер і необхідність багатопараметричної оптимізації маршрутів. Проектування інформаційного та програмного забезпечення включало ER-діаграму, діаграми класів, компонентів і пакетів, що створило чітку архітектуру з центральною роллю RoutingEngine.

Реалізація забезпечила паралельне виконання алгоритмів A*, Dijkstra з урахуванням безпеки та Bidirectional Dijkstra на єдиній комбінованій вазі (40 % час, 35 % безпека, 25 % відстань), а також механізм консенсусу для визначення ступеня згоди між результатами. Тестування модулів продемонструвало стабільну роботу системи з часом розрахунку маршруту в межах 2000 мс,

високим відсотком сценаріїв повного консенсусу та коректним розпізнаванням дорожніх об’єктів. Розгортання через pip для десктопу та Buildozer для Android підтвердило простоту інсталяції та практичну готовність застосунку до використання в реальних умовах.

Перспективи подальшого розвитку системи пов’язані з її масштабуванням і адаптацією до ширших потреб користувачів. У майбутньому доцільно розширити граф на інші регіони України, інтегрувати реальні дані про трафік і погодні умови, а також додати підтримку GPS-навігації в реальному часі з автоперерахунком маршруту. Оптимізація для великих агропідприємств може включати врахування обмежень вантажопідйомності техніки та сезонних факторів.

Подальше вдосконалення інтерфейсу, розширення тестової бази та публікація результатів на наукових конференціях сприятимуть популяризації проєкту. Таким чином, створена інформаційна система Pathfinder є надійною основою для ефективної оптимізації маршрутів у транспортній та аграрній галузях і має значний потенціал для практичного впровадження та подальшого розвитку.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. КАБІНЕТ МІНІСТРІВ УКРАЇНИ. Про схвалення Національної транспортної стратегії України на період до 2030 року (2018 зі змінами від 2021) URL: https://zakon.rada.gov.ua/laws/show/430-2018-%D1%80#Text

2. Юхновський О. Аналітичний огляд поточного стану та напрямків розвитку аграрної торговельної політики України в 2024 році. Агрополітичний звіт. APD/APB/12/2023. Німецько-український агрополітичний діалог. 2023. С. 19.

3. World Bank. Ukraine Rapid Damage and Needs Assessment: Transport and Logistics Sector (2022) URL: https://documents.worldbank.org/en/publication/documents-reports/documentdetail/099445209072239810

4. Державна служба статистики України. Транспорт і зв'язок України: статистичний збірник (2023) URL: https://ukrstat.gov.ua/

5. Лозова Г. М., Клименко В. В., Кожолянко І. В. Цифрова трансформація транспортної системи в Україні. Теоретичні та прикладні питання економіки. Вип. 1 (44). С. 174-186.

6. OpenStreetMap Foundation. Routing (2024) URL: https://wiki.openstreetmap.org/wiki/Routing

7. HERE Technologies. HERE Routing API Documentation (2023) URL: https://developer.here.com/documentation/routing-api/dev_guide/topics/introduction.html

8. TomTom International B.V. TomTom Navigation Technologies (2024) URL: https://www.tomtom.com/navigation/

9. Кравець О. В. Моделювання предметної області систем оптимізації транспортних маршрутів. Наукові вісті НТУУ «КПІ». 2023. № 2. С. 45–56.

10. Пострелко Є. Цифрова трансформація міських транспортних систем: класифікація підходів та рекомендації для України. Herald of Khmelnytskyi National University Technical Sciences. 2025. № 5.2. С. 300-307 URL: https://www.researchgate.net/publication/399036566_CIFROVA_TRANSFORMAC

IA_MISKIH_TRANSPORTNIH_SISTEM_KLASIFIKACIA_PIDHODIV_TA_REKOMENDACII_DLA_UKRAINIDIGITAL_TRANSFORMATION_OF_URBAN_TRANSPORT_SYSTEMS_CLASSIFICATION_OF_APPROACHES_AND_RECOMMENDATIONS_FOR

11. OSMnx Developers. OSMnx 2.1.0 (2024) URL: https://osmnx.readthedocs.io/en/stable/

12. World Bank. How Better Transport Drives Opportunity (2026) URL: https://www.worldbank.org/en/results/2026/04/01/how-better-transport-drives-opportunity

13. Міністерство цифрової трансформації України. Результати цифрової трансформації в регіонах України за 2024 рік (2025) URL: https://thedigital.gov.ua/news/regions/rezultati-tsifrovoi-transformatsii-v-regionakh-ukraini-za-2024-rik

14. Литовченко І. П. Аналіз вимог до сучасних навігаційних інформаційних систем. Системи управління, навігації та зв'язку. 2023. № 4 (74). С. 78-85.

15. Google Developers. Routes API (2024) URL: https://developers.google.com/maps/documentation/routes

16. Geoapify. Routing API (2026) URL: https://www.geoapify.com/routing-api/

17. Ремига Ю. Цифровізація і сталий розвиток транспортної системи. Expert Opinion. 2025 URL: https://expert-opinion.pp.ua/isst/article/view/171

18. Павленко О. І., Баркалова Н. О. Оптимізація логістичних маршрутів для зниження витрат у вантажних автоперевезеннях. Технічний вісник. 2025 URL: https://www.tech.vernadskyjournals.in.ua/journals/2025/6_2025/part_1/44.pdf

19. Сахно С. А., Любий Є. В., Колій О. С. Аналіз стратегій формування маршрутних мереж міст. MAPIEA. 2025 URL: https://mapiea.kntu.kr.ua/pdf/12(43)_II/36.pdf

20. Pysarchuk O. Multicriteria Scoring Method for Travel Routes Based on Graph Models. ICSFTI. 2025 URL: https://icsfti-proc.kpi.ua/article/view/338746

21. Кравець О. В. Логічні моделі даних у інформаційних системах оптимізації транспортних маршрутів. Наукові вісті НТУУ «КПІ». 2022. № 1. С. 34-45.

22. Литовченко І. П. Проектування логічних моделей даних для сучасних навігаційних систем. Наукові записки НаУКМА. 2023. Т. 6. С. 44-51.

23. GrupaTransportowa. Як цифровізація впливає на ефективність вантажних перевезень? (2026) URL: https://grupatransportowa.pl/ua/news/jak-digitalizacja-wplyva-na-efektywnosc-transportu-towarowego

24. Кібербезпека енергетики: матеріали науково-практичної конференції, 28 травня 2025 р. Київ: ІПМЕ ім. Г. Є. Пухова НАН України, 2025. 90 с.

25. Петренко А. С. Проектування діаграм класів UML для інформаційних систем транспортної логістики. Вісник Житомирського державного технологічного університету. 2023. № 2 (92). С. 34-42.

26. Іванчук О. В. Діаграми взаємодії в об’єктно-орієнтованому програмуванні. Вісник КНУ ім. Тараса Шевченка. 2024. № 1. С. 56-63.

27. Перелік Національних стандартів України для створення, впровадження та супроводження автоматизованих і інформаційних систем URL: http://nbuv.gov.ua/node/1469

28. UML для бізнес-моделювання: для чого потрібні діаграми процесів URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html

29. Литовченко І. П. Компонентне моделювання в інформаційних системах оптимізації транспортних маршрутів. 2024. № 108. С. 45-52.

30. Іванчук О. В. Візуалізація компонентної архітектури програмного забезпечення за допомогою PlantUML. Проблеми програмування. 2025. № 1. С. 67-74.

31. Петренко А. С. Інтерфейси компонентів в UML для систем логістики. Автоматизація виробничих процесів. 2024. № 4. С. 23-31.

32. PlantUML Official. Component Diagram (2024) URL: https://plantuml.com/component-diagram

33. Кравець О. В. Моделювання пакетної архітектури в інформаційних системах транспортної галузі. 2024. № 3. С. 56-65.

34. Іванчук О. В. Застосування PlantUML для візуалізації діаграм пакетів. 2025. № 54. С. 78-85.

35. Авраменко В. С., Авраменко А. С. Проектування інформаційних систем: навчальний посібник. Черкаси: ЧНУ ім. Б. Хмельницького, 2017. 434 с.

36. Особливості моделювання пакетів URL: https://www.siemens.com/uk-ua/products/ic-packaging/software/package-simulation/features/

37. Петренко А. С. Сучасні підходи до вибору технологій для кросплатформених навігаційних систем. 2024. № 2. С. 45-53.

38. Kivy Organization. Kivy 2.3.0 Documentation (2024) URL: https://kivy.org/doc/stable/

39. Іванчук О. В. Kivy та KivyMD як основа для мобільних GIS-додатків в Україні. 2025. № 1. С. 67-75.

40. osmnx 1.3.1 URL: https://pypi.org/project/osmnx/1.3.1/

**Декларація про академічну доброчесність та використання ШІ**

Я, ____Волошин Антон Миколайович_____, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «**Система формування оптимального маршруту пересування транспортних засобів**» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):

- [ ] інструменти генеративного ШІ не використовувалися.
- [ ] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:

- [ ] перекладу іншомовних джерел;
- [ ] стилістичного редагування та перевірки граматики;
- [ ] допомоги у написанні/відлагодженні програмного коду;
- [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«_23»__травня_____2026 р. _____ **Волошин Антон**

(підпис)