НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Програмне забезпечення для збору та інтелектуального аналізу даних дорожнього руху з використанням камер відеоспостереження»

Виконав:
студент IV курсу, групи КП-02
Простаков Олег Дмитрович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Олещенко Любов Михайлівна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри ІСТ, к.т.н., доцент,
Полторак Вадим Петрович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____ 2023 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Простакову Олегу Дмитровичу

1. Тема проєкту «Програмне забезпечення для інтелектуального аналізу даних дорожнього руху з використанням камер відеоспостереження», керівник проекту Олещенко Любов Михайлівна, к.т.н., доцент, затверджені наказом по університету від «30» травня 2024 р. №2205-с

2. Термін подання студентом проєкту «14» червня 2024 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз предметної області та існуючих рішень;
   - обґрунтування вибору засобів реалізації;
   - розроблення архітектури та програмних модулів;
   - аналіз розробленої програмної системи.

5. Перелік обов'язкового графічного матеріалу:
   - структура бази даних (креслення);
   - варіанти використання. UML-діаграма (креслення);
   - архітектура програмної системи (плакат);
   - дерево проблем (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	13.11.2023	
2.	Розроблення та узгодження технічного завдання	22.11.2023	
3.	Розроблення структури програмного забезпечення	15.12.2023	
4.	Підготовка матеріалів першого розділу дипломного проєкту	28.12.2023	
5.	Розроблення дизайну сторінок та графічних елементів	01.02.2024	
6.	Підготовка матеріалів другого розділу дипломного проєкту	19.02.2024	
7.	Реалізація програмного забезпечення	11.03.2024	
8.	Тестування програмного забезпечення	18.03.2024	
9.	Підготовка матеріалів третього розділу дипломного проєкту	27.03.2024	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	15.04.2024	
11.	Підготовка графічної частини дипломного проєкту	22.04.2024	
12.	Оформлення документації дипломного проєкту	27.05.2024	

Студент Олег ПРОСТАКОВ

Керівник проєкту Любов ОЛЕЩЕНКО

АНОТАЦІЯ

Даний дипломний проєкт присвячений створенню програмного забезпечення для інтелектуального аналізу даних дорожнього руху з використанням камер відеоспостереження.

Програмне забезпечення являє собою набір сервісів, призначених для отримання, структуризації та зберігання даних з відео потоків камер відеоспостереження; аналізу цих даних в режимах реального та близького до реального часів; виведення результатів аналізу на web-ресурсі. Будучи зареєстрованим на даному ресурсі, за натисканням на іконку камери відеоспостереження, що розміщено на фоні мапи міста, користувач може дізнатись кількість унікальних транспортних засобів та їхню середню швидкість, що були зафіксовані за короткий проміжок часу. Така інформація оновлюється в режимі реального часу. Окрім цього, користувач має можливість переглянути історичну інформацію з даної камери, що подається у вигляді графіку. Також, користувачі з правами адміністратора можуть додавати, редагувати та видаляти реєстраційні дані камер відеоспостереження з web-ресурсу.

Стосовно даних, що отримуються з відео потоків, на кожному кадрі і для кожного транспортного засобу розроблене програмне забезпечення знаходить координати прямокутної області, що охоплює засіб, та додає цю область до ідентифікатору транспортного засобу в режимі реального часу. Ідентифікатори є унікальними в межах кожної камери та закріплюються за транспортними засобами між різними кадрами. Також, програмне забезпечення вилучає значення кольору, моделі та виробнику кожного поміченого транспортного засобу в режимі близького до реального часу.

У даному дипломному проєкті розроблено: архітектуру серверної та клієнтської частини програмного забезпечення, схему бази даних, графічні елементи та дизайн web-сторінок, моделі машинного навчання для

отримання даних з кадрів відео потоку, алгоритм обчислення швидкості транспортного засобу на основі спостережень з декількох кадрів.

ABSTRACT

This diploma project is dedicated to the development of software aimed at the collection and intelligent processing of vehicle traffic data using CCTV cameras.

The software is a set of services developed to extract, structure, and store data from the CCTV cameras' video streams; analyze this data in real- and close-to-real times; and display the analysis results on a website. When registered on this website, by clicking on a CCTV camera icon, displayed on an overlay of a city, a user can discover the number of unique vehicles along with their average velocity tracked over a short period of time. Such information is updated in real-time. In addition, the user can also inspect the historical information of the camera, plotted over time. Furthermore, users with administrative permissions are able to add, modify, and remove the registration data of CCTV cameras from the website.

Regarding the information obtained from the video streams, on every frame and for each vehicle the software detects a rectangular area, which surrounds the vehicle, and appends this area to the vehicle's identifier in real time. The identifiers are unique in the scope of each camera and persist for vehicles between different frames. Furthermore, the software extracts the values of color, make, and model of every detected vehicle in the close-to-real-time mode.

The following structures and algorithms have been developed for this project: an architecture for the client and server sides of the software, a database schema, graphical elements and design of the web pages, machine learning models for data extraction from frames of a video stream, and an algorithm for calculation of a vehicle's velocity based on several observations of the vehicle over different frames.

ДП.045440-01-90 Програмне забезпечення для інтелектуального аналізу даних дорожнього руху з використанням камер відеоспостереження. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Програмне забезпечення	5	
	для збору та		
	інтелектуального аналізу		
	даних дорожнього руху		
	з використанням камер		
	відеоспостереження.		
	Технічне завдання		
ДП.045440-03-81	Програмне забезпечення	76	
	для збору та		
	інтелектуального аналізу		
	даних дорожнього руху		
	з використанням камер		
	відеоспостереження.		
	Пояснювальна записка		
ДП.045440-04-51	Програмне забезпечення	4	
	для збору та		
	інтелектуального аналізу		
	даних дорожнього руху		
	з використанням камер		
	відеоспостереження.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Програмне забезпечення	12	
	для збору та		
	інтелектуального аналізу		
	даних дорожнього руху		
	з використанням камер		
	відеоспостереження.		
	Керівництво користувача		

ДП.045440-06-99	Програмне забезпечення	1	
	для збору та		
	інтелектуального аналізу		
	даних дорожнього руху		
	з використанням камер		
	відеоспостереження.		
	Структура бази даних.		
	ERD-діаграма.		
ДП.045440-07-99	Програмне забезпечення	1	
	для збору та		
	інтелектуального аналізу		
	даних дорожнього руху		
	з використанням камер		
	відеоспостереження.		
	Варіанти використання.		
	UML-діаграма		
ДП.045440-08-98	Програмне забезпечення	1	
	для збору та		
	інтелектуального аналізу		
	даних дорожнього руху		
	з використанням камер		
	відеоспостереження.		
	Компакт-диск		

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Євгенія СУЛЕМА
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ ДОРОЖНЬОГО РУХУ З ВИКОРИСТАННЯМ КАМЕР ВІДЕОСПОСТЕРЕЖЕННЯ

Технічне завдання

ДП.045440-02-91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Любов ОЛЕЩЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олег ПРОСТАКОВ

2023

ЗМІСТ

1. Найменування та галузь застосування 3
2. Підстава для розроблення 3
3. Призначення розробки 3
4. Вимоги до програмного продукту 3
5. Вимоги до проєктної документації 4
6. Етапи проєктування 5
7. Порядок тестування розробки 5

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для інтелектуального аналізу даних дорожнього руху з використанням камер відеоспостереження

Галузь застосування: інформаційні технології

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для полегшення та оптимізації процесів пошуку інформації та отримання необроблених структурованих даних дорожнього руху в різних частинах міста.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмне забезпечення повинне забезпечувати такі основні функції:

1) можливість збору даних з відео потоків камер відеоспостереження;
2) можливість виведення даних, що дозволяють обчислювати кількість та швидкість транспортних засобів в режимі реального часу;

3) можливість збільшення кількості властивостей, що збираються з відео потоків.
4) надання інструментарію для отримання інформації з зібраних даних.
5) візуалізація результатів оброблення даних на web-ресурсі.
6) можливість додавання, редагування та видалення джерел відео потоків до системи.

Додаткові вимоги:

1) зручність та зрозумілість користувацього інтерфейсу web-ресурсу;
2) ефективність використання ресурсів систем обробки, транспортування та зберігання даних;
3) можливість масштабування системи під різну кількість джерел відео потоків без втрати швидкодії;
4) безпека зібраних даних.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   - «Структура бази даних. ERD-діаграма»;
   - «Алгоритм обчислення швидкості транспортного засобу на основі декількох спостережень зі статичної камери відеоспостереження. Блок-схема».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи……..16.11.2023

Розроблення та узгодження технічного завдання……....27.11.2023

Розроблення структури програмного забезпечення…….....15.12.2023

Розроблення дизайну сторінок та графічних елементів……..05.02.2024

Програмна реалізація програмного забезпечення……....15.03.2024

Тестування програмного забезпечення…….....08.04.2024

Підготовка матеріалів текстової частини проєкту……..….29.04.2024

Підготовка матеріалів графічної частини проєкту……...…15.05.2024

Оформлення технічної документації проєкту……..…28.05.2024

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2024 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ ДОРОЖНЬОГО РУХУ З ВИКОРИСТАННЯМ КАМЕР ВІДЕОСПОСТЕРЕЖЕННЯ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Любов ОЛЕЩЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олег ПРОСТАКОВ

2024

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ 3
ВСТУП 8
1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ 10
1.1. Огляд проблеми, яка вирішується ПЗ 10
1.2. Аналіз існуючих програмних рішень 11
1.3. Результати проведеного аналізу 17
1.4. Перелік вимог до програмного забезпечення 18
1.5. Висновки до розділу 1 19
2. ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ 20
2.1. Вибір мов програмування 20
2.2. Вибір СУБД 26
2.3. Вибір брокеру повідомлень 34
2.4. Додаткові технології та бібліотеки 38
2.5. Висновки до розділу 2 40
3. АРХІТЕКТУРА ПРОГРАМНИХ МОДУЛІВ 42
3.1. Загальний опис архітектури 42
3.2. Огляд модулів програмного забезпечення 43
3.3. Висновки до розділу 3 55
4. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ . 57
4.1. Тестування програмного забезпечення 57
4.2. Порівняння з існуючими рішеннями 65
4.3. Рекомендації щодо подальшого вдосконалення 65
4.4. Висновки до розділу 4 66
ВИСНОВКИ 67
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ 69
ДОДАТКИ 76

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення.

СУБД – система управління базами даних.

ТЗ – транспортний засіб.

Атрибут даних – властивість, що характеризує об’єкт. Наприклад: колір автомобіля, його виробник або модель.

Автентифікація – процес встановлення особливості користувача програмної системи.

Агрегація даних – процес збору даних з різних джерел та перетворення їх у одне узагальнене представлення.

Архітектура нейронної мережі – структура нейронної мережі та типи штучних нейронів.

Асинхронна комунікація – тип комунікації з відкладеної відповіддю на повідомлення. Наприклад, асинхронна комунікація присутня в секціях коментарів в соціальних мережах.

Брокер повідомлень – програма посередник, що забезпечує асинхронну комунікація між декількома компонентами системи. Повідомленням вважається будь-який набір даних, що передається до або від брокера.

Відстеження (у комп'ютерному зорі) – визначення розташування рухомого об’єкта з часом.

Динамічна типізація – система типізації, де тип змінної з’ясовується під час виконання програмного коду, а не на стадії компіляції.

Згорткова нейронна мережа – клас нейронних мереж, що часто застосовується для аналізу візуальних зображень.

Зовнішня реклама – реклама, розміщена на вулицях (рекламні щити, перетяжки тощо).

Інтерпретовані мови програмування – мови програмування, де вихідний код одразу виконується інтерпретатором мови, без попереднього його перетворення на машинний код.

Кешування – тимчасове зберігання даних задля зменшення затримок під час отримання даних зі стійких джерел.

Класифікація (у комп’ютерному зорі) – віднесення вхідного зображення до певного, наперед-визначеного класу.

Компільовані мови програмування – мови програмування, де вихідний код має перетворитись на машинний код перед виконанням інтерпретатором мови.

Комп’ютерний зір - теорія та технологія створення машин, які можуть проводити виявляння, відстежування та визначення об'єктів.

Метрики успішності рекламних кампаній – числові показники, які вимірюють успішність рекламної кампанії. Наприклад: кількість осіб, що контактували з рекламним повідомленням або кількість осіб, що придбали рекламований товар.

Нейронна мережа (штучна) – математична програмна модель, що ґрунтується на сукупності великої кількості з’єднаних вузлів.

Охоплення рекламної кампанії – відсоток показів оголошень із рекламної кампанії користувачам, які є її цільовою аудиторією.

Проєкція даних – перетворення даних з одного виду на інший.

ПЗ з відкритим кодом – ПЗ, вихідний код якого доступний для перегляду, вивчення, зміни, використання для створення нових програм та їх виправлення.

Розподілена система – набір незалежних комп'ютерів, представлений користувачеві єдиною об'єднаною системою.

Розпізнавання (у комп’ютерному зорі) – визначення того, чи містять одне або послідовність зображень деякий характерний об'єкт, особливість чи активність.

Снепшот – статичний зріз стану деякої програмної системи, наприклад, бази даних в певний момент часу.

Транскодування – конвертація потоку з одного формату на інший.

Тренування (моделі машинного навчання) – процес навчання моделі робити передбачення базуючись на вхідних даних.

Фреймворк – інфраструктура програмних рішень, що полегшує розробку складних систем. Фреймворк можна вважати своєрідною комплексною бібліотекою, що містить ряд обмежень, задаючих правила створення проєкту та написання коду.

Хешування – перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини.

Часова мітка – послідовність символів або закодованої інформації, що показує, коли відбулася певна подія.

ACID (Atomicity Consistency Isolation Durability) – набір принципів, що допомагають підтримувати цілісність даних та забезпечувати надійність транзакцій в базі даних.

AdTech – термін, що поєднує всі види інструментів та програмних платформ для налаштування та показу реклами потенційним клієнтам.

AMQP (Advanced Message Queuing Protocol) – відкритий мережевий протокол на рівні застосунку, що використовується у програмах-посередниках, орієнтованих на передачу повідомлень.

ANSI SQL – стандарт мови SQL, метою якого є забезпечення сумісності SQL коду між різними СУБД.

API (Application Programming Interface) – це набір правил та протоколів, які визначають спосіб взаємодії між різними програмними компонентами чи сервісами через мережу Інтернет, дозволяючи їм обмінюватися даними та функціональністю.

B2B (Business to Business) – термін, що означає обмін послугами між компаніями, не включаючи в цей процес кінцевого фізичного споживача товару чи послуги.

DOM (Document Object Model) – інтерфейс, що дозволяє програмам та скриптам динамічно доступатись та оновлювати зміст, структуру та стиль HTML документів.

ERD (Entity-relationship diagram) – діаграма, що візуалізає зв’язки між сутностями в базі даних.

FIFO (First In, First Out) – у веброзробці цей принцип відноситься до алгоритму кешування, де найстаріші запити обробляються першими, що дозволяє ефективно управляти ресурсами сервера та зберігати стабільну швидкодію вебсайту.

IDE (Integrated Development Environment) – комплексне середовище розробки, яке надає інтегровані інструменти та можливості для створення програмного забезпечення. У веброзробці IDE включає в себе різноманітні інструменти, такі як текстовий редактор, компілятор, дебагер, підтримку версій, автоматизоване виведення коду та інші, які полегшують та прискорюють процес розробки вебдодатків.

IoT (Internet of Things) – концепція мережі, що складається із взаємозв'язаних фізичних пристроїв, а також програмного забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом і комп'ютерними системами в автоматичному режимі, за допомогою використання стандартних протоколів зв'язку.

MVCC (Multi-Version Concurrency Control) – архітектурне рішення в базах даних для управління транзакціями, що виконуються паралельно.

P2P (peer-to-peer) – варіант архітектури системи, в основі якої стоїть мережа рівноправних вузлів.

SPA (Single Page Application) – це тип вебдодатка, який працює в одній HTML-сторінці, завантажуючи динамічний контент за допомогою AJAX-запитів.

SSR (Server-Side Rendering) – механізм генерації вмісту сторінок HTML на стороні сервера, що допомагає підвищити швидкодію та швидкість завантаження сторінок вебзастосунків.

TCP (Transmission Control Protocol) – один з головних мережевих протоколів транспортного рівня, що використовуються в комп’ютерних системах.

ВСТУП

Кількість компаній, що використовують великі дані для оптимізації своїх бізнес-моделей, зростає з кожним днем. Дослідження ілюструють цей зростаючий інтерес тим, що капіталізація ринку великих даних збільшилась вдвічі з 2017 по 2023 роки [1]. Даний тренд не обійшов стороною маркетингові компанії.

Зростання популярності інтернет-маркетингу [2] демонструє високу ефективність реклами, що персоналізується під різні сегменти цільової аудиторії та адаптується в залежності від отриманих метрик успішності. Проте така персоналізація стала можливою лише завдяки тому, що платформи для впровадження цифрової реклами, такі як Google Ads та Facebook Ads зберігають та аналізують величезні обсяги даних про своїх користувачів [3]. Незважаючи на це, компанії, що задіяні в інших сферах маркетингу, таких як зовнішня реклама, також хотіли б мати доступ до даних про свою цільову аудиторію, що забезпечують успіх інтернет-маркетингу.

Нещодавнє зниження цін на сенсорні пристрої [4] разом зі стрімким зростанням технологій впровадження мереж IoT зробили можливим постійний та автоматизований моніторинг локацій проведення рекламних кампаній. Такий моніторинг дозволяє отримувати та аналізувати великі обсяги даних про свою цільову аудиторію компаніям, що задіяні в індустрії зовнішньої реклами. Очевидно, що обсяги зібраних таким чином даних є суттєво меншими за обсяги даних, якими оперують глобальні платформи інтернет-маркетингу. Проте навіть вони дозволяють оптимізувати портрет цільової аудиторії та покращити процеси збору метрик успішності кампаній.

Одним з найпопулярніших каналів пасивного збору даних для рекламних кампаній є системи моніторингу транспортних потоків,

розташовані в місцях проведення рекламних кампаній. Дані, отримані з камер відеоспостереження, дозволяють складати оцінку популярності локацій в залежності від зовнішніх факторів, збагачувати портрет аудиторії такими деталями як виробник та модель автомобіля, адаптувати рекламний контент в залежності від швидкості дорожнього потоку, тощо.

Не дивлячись на вище розглянуті переваги, ринок таких систем не є занадто активним, і компаніям, що хочуть їх використовувати, зазвичай доводиться розроблювати власне рішення. Отже, розроблення програмного забезпечення для моніторингу транспортних потоків є актуальною задачею.

У даному дипломному проєкті було проаналізовано існуючі програмні системи моніторингу транспортних потоків з метою отримання даних для оптимізації рекламних кампаній, сформовано вимоги до розроблюваного програмного забезпечення з урахуванням їхніх переваг та недоліків. Після цього було проаналізовано та відібрано технології програмування для розробки програмного забезпечення. Далі, програмне забезпечення було розроблено з урахуванням функціональних і нефункціональних вимог та протестовано, виділено його основні переваги та напрямки подальшого розвитку.

1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Огляд проблеми, яка вирішується ПЗ

Маркетингові компанії, що працюють в індустрії зовнішньої реклами, стикаються з декількома спільними проблемами – пошуком оптимальної локації для розміщення такої реклами, створенням послідовності показу рекламного контенту, а також збором метрик успішності кампаній. Розглянемо ці проблеми детальніше.

- Пошук оптимальної локації: навіть невелике місто може налічувати сотні або тисячі локацій, придатних для розміщення реклами. Проте, окрема кампанія буде використовувати лише декілька з них [5]. Відповідно, виникає необхідність пошуку декількох найкращих місць з урахуванням ціни за оренду локації, очікуваного охоплення, та частки цільової аудиторії серед цього охоплення. Хоча ціна оцінюється достатньо просто, оцінку охоплення та частки цільової аудиторії з високою мірою точності вручну провести майже неможливо.
- Динамічне оновлення рекламного контенту: маркетингові компанії нерідко використовують одну й ту саму локацію для декількох рекламних кампаній одночасно. Прикладом такої локації може бути цифровий щит (білборд). Зазвичай контент в таких локаціях оновлюється за певним таймером, незважаючи на чинники оточуючого середовища, такі як швидкість дорожнього руху. Це є проблемою, адже, ефективність більшості рекламних відеороликів залежить від умов навколишнього середовища [6]. Наприклад, довші ролики більш ефективні за умов повільного трафіку. В той же час, коротші, більш захоплюючі відео, досягають максимальної ефективності за умов швидкого трафіку.

- Збір метрик успішності: на відміну від онлайн-маркетингу, де самі рекламні платформи надають рекламодавцю різноманітні метрики успішності кампаній, такі як охоплення або кількість переходів до веб-банеру, в зовнішній рекламі метрики успішності кампаній збираються лише за дотичними показниками та з достатньо низькою точністю.

Логічним вирішенням даних проблем, за умов того, що цільовою аудиторією є учасники дорожнього руху, є використання системи моніторингу дорожнього руху. Огляду та аналізу існуючих програмних рішень присвячено наступний підрозділ.

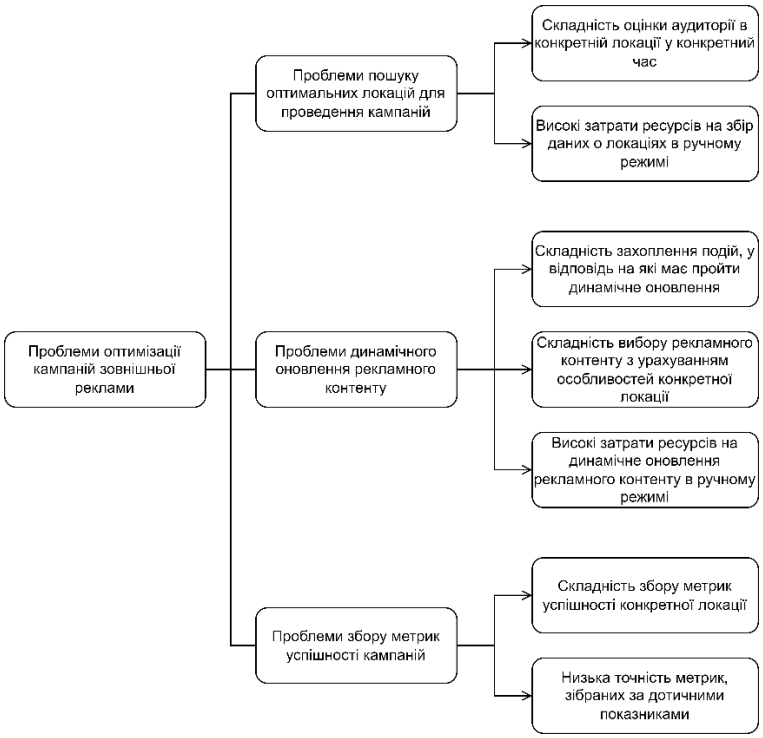


Рис. 1.1. Дерево проблем проєкту

1.2. Аналіз існуючих програмних рішень

1.2.1. *Outfront AdServer та smartSCOUT*

Outfront AdServer та Outfront smartSCOUT – платформи, розроблені компанією Outfront Media для використання у власних рекламних кампаніях.

Платформа Outfront AdServer призначена для оптимізації виведення рекламного контенту в залежності від зовнішніх чинників. Дана платформа збирає дані про зовнішнє середовище, аналізує їх в режимі реального часу, визначає рекламний контент, що підходить аудиторії найбільше за поточних умов, та виводить його на екрани [7]. Дані, що збирає AdServer, структуризуються за багатьма атрибутами – від кількості пішоходів або автомобилів навколо до поточних температури, вологості та швидкості вітру. Після збору дані залишаються у внутрішньому сховищі AdServer для подальшого історичного аналізу.

Платформа Outfront smartSCOUT призначена для пошуку та оцінки цільових аудиторій, використовуючи агреговані та анонімізовані дані, одна частина яких є придбаною у сторонніх провайдерів даних, а інша – зібрана за допомогою AdServer [8]. Результати роботи smartSCOUT допомагають оптимізовувати рекламні кампанії шляхом знаходження найкращих локацій для їх проведення. Варто зазначити, що як AdServer так і smartSCOUT є інтелектуальною власністю Outfront Media, яку дана компанія використовує виключно для проведення власних рекламних кампаній. Окрім цього, сама Outfront Media працює виключно на територіях країн Північної Америки.

Отже, підсумуємо переваги та недоліки платформ від Outfront Media.

Переваги:

- Можливість самостійного збору даних.
- Можливість збору та аналізу даних як транспортних, так і пішохідних потоків.
- Наявність інструментів для проведення ретельного аналізу зібраних даних, в тому числі, для пошуку оптимальних локацій для проведення кампаній.
- Можливість збору та аналізу метрик рекламних кампаній з багатьох джерел.

Недоліки:

- Відсутність доступу до платформ за межі компанії Outfront Media.
- Обмеженість роботи компанії до ринків країн Північної Америки.

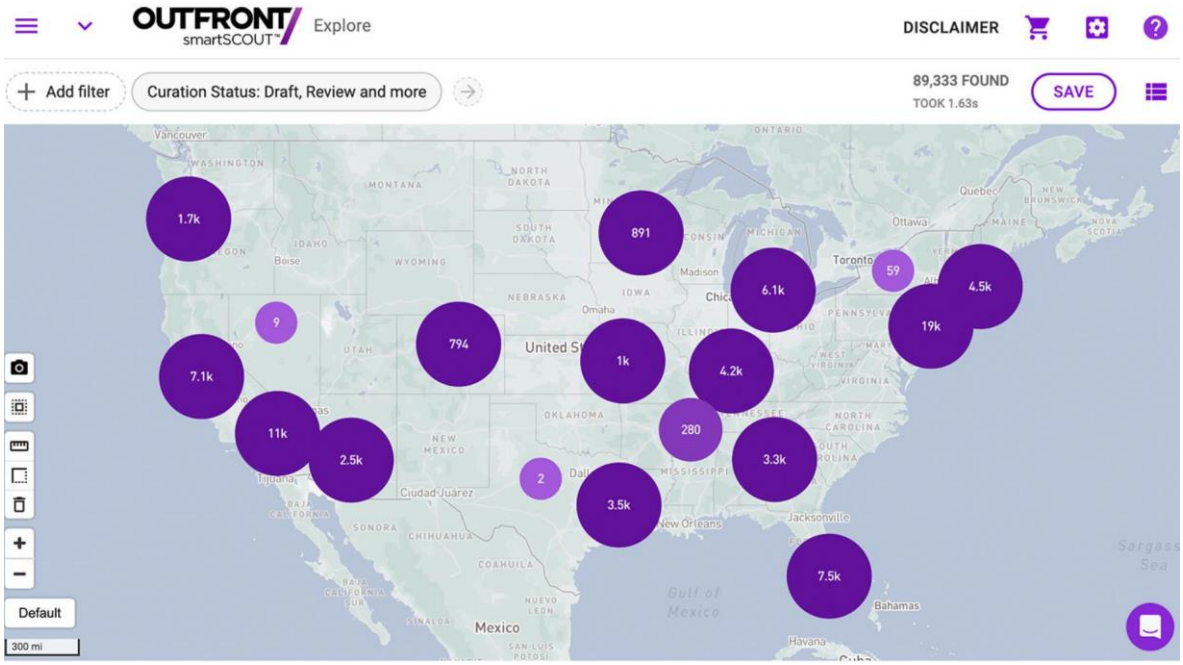


Рис. 1.2. Мапа розподілу цільових аудиторій містами США на платформі OUTFRONT smartSCOUT [8]

1.2.2. *JCDecaux Data Solutions*

JCDecaux Data Solutions – це комплекс програмних рішень від компанії JCDecaux Group, спрямований на збір та аналіз даних для покращення рекламних кампаній [9]. Даний комплекс допомагає клієнтам JCDecaux оптимізовувати контент рекламних компаній з урахуванням найдрібніших особливостей цільових аудиторій, доводити цей контент до аудиторії через різноманіття цифрових та фізичних каналів та проводити оцінку успішності рекламних кампаній на кожному з цих каналів. Також комплекс рішень від JCDecaux підтримує динамічну зміну рекламного контенту в залежності від зовнішніх чинників. Приклади такого контенту наведено на рисунках 1.2. та 1.3.

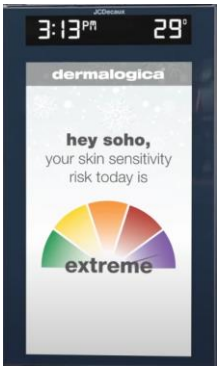


Рис. 1.3. Цифровий щит JCDecaux в районі SoHo м. Нью-Йорк за зовнішньої температури 29° за Фаренгейтом [10]

Рис. 1.4. Цифровий щит JCDecaux в районі SoHo м. Нью Йорк за зовнішньої температури 32° за Фаренгейтом [10]

Розглянемо детальніше переваги та недоліки даного програмного рішення.

Переваги:

- Здатність до оптимізації рекламних кампаній, в тому числі, шляхом пошуку оптимальних локацій для їхнього проведення.
- Можливість самостійного збору даних.
- Здатність до збору різноманітних метрик успішності рекламних кампаній з великої кількості цифрових та фізичних джерел.
- Наявність інструментів для отримання найдрібніших особливостей цільових аудиторій.

Недоліки:

- Фокус на рекламу, та, відповідно, збір даних з пішохідних потоків.
- Нерозповсюдженість доступу до програмного рішення за межі компанії JCDecaux.

1.2.3. *RADAR Data Solutions*

RADAR Data Solutions – комплекс рішень, розроблений маркетинговою компанією Clear Channel Outdoor (CCO) для планування, посилення та оцінювання кампаній зовнішньої реклами за допомогою даних, зібраних з декількох різних джерел [11].

По-перше, для планування рекламних кампаній RADAR використовує агреговані та/або анонімізовані дані про локації мобільних пристроїв. Такі дані допомагають компанії CCO розуміти популярні маршрути та види занять своєї аудиторії. Аналіз цих даних надає компанії уявлення про те, представники якої аудиторії будуть переважно охоплюватись кампанією, що проводиться в конкретному місці та конкретний час (рис. 1.4).

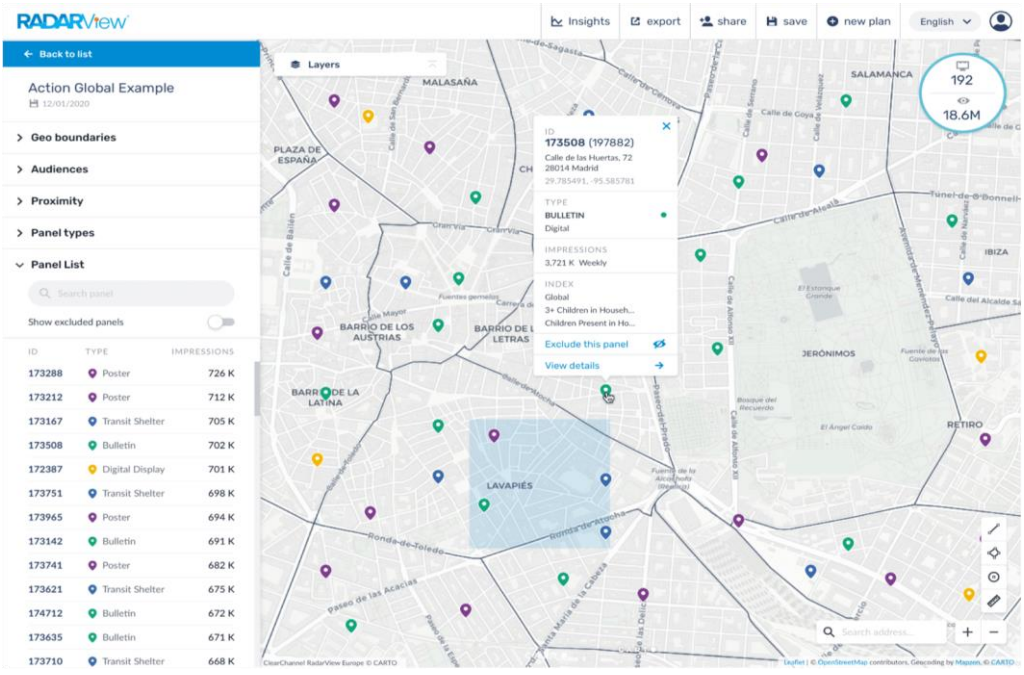


Рис. 1.5. Мапа охоплень цифрових щитів в різних точках міста в застосунку RADAR [12]

По-друге, для оцінювання успішності кампаній, RADAR використовує візуальні дані, що збираються камерами відеоспостереження, розташованими поблизу рекламних цифрових, щитів разом з інформацією про активність мобільних пристроїв поблизу цих щитів. Аналіз цих даних дозволяє знаходити проміжок часу, протягом якого пішохід або автомобіль знаходяться в оптимальній позиції відносно цифрового щита [13] та проводити оцінку зміни активності мобільних пристроїв поблизу тієї позиції протягом цього проміжку часу, отримуючи ймовірність взаємодії потенційного клієнта з рекламою.

Розглянемо більш детально переваги та недоліки RADAR Data Solutions.

Переваги:

- Використання даних, зібраних з багатьох каналів.
- Можливість збору даних як з транспортних, так і цифрових потоків.
- Наявність інструментів для знаходження популярних маршрутів цільових аудиторій, що дає більше можливостей для оптимізації кампаній, ніж знаходження популярних локацій.

Недоліки:

- Неможливість автоматичної зміни рекламного контенту в залежності від зовнішніх чинників.
- Залежність від даних про локації та активностях мобільних пристроїв, які компанія ССО не може збирати самостійно.
- Нерозповсюдження доступу до рішення RADAR Data Solutions за межі компанії ССО.
- Обмеженість роботи компанії до ринків країн Північної Америки.

1.3. Результати проведеного аналізу

Як результат проведення аналізу існуючих рішень, що надають маркетинговим компаніям можливість моніторингу дорожнього руху в місцях інтересу, було сформовано ряд висновків.

По-перше, варто зазначити, що кожне з рішень було розроблено під потреби конкретних компаній, та використовується виключно компаніями-розробниками. На відміну від ринку цифрового маркетингу, де існує ряд B2B-платформ для оптимізації та оцінки рекламних кампаній, на ринку зовнішнього маркетингу подібні платформи розробляються та використовуються лише найбільшими компаніями.

По-друге, кожне з проаналізованих рішень фокусується на одному, хоч і значному за розміром, сегменті ринку. Рішення від Outfront та Clear Channel Outdoor були розроблені під роботу на ринках країн Північної Америки. Відповідно, навіть якщо ad-tech компанія, що веде бізнес в країнах Європи або Південно-Східної Азії, отримає до них доступ, значна кількість даних, що використовуються цими платформами, буде для такої компанії нерелевантною. В свою чергу, набір рішень JCDecaux Data Solutions розрахований на роботу на ринках країн в декількох географічних регіонах. Проте, сама компанія фокусується на маркетингу аудиторіям, що не використовують транспортні засоби. Відповідно, даних про транспортні потоки, що збирає дане рішення, буде недостатньо для проведення оптимізації або оцінки рекламних кампаній.

Таблиця 1.1

Порівняння існуючих програмних рішень

Критерій	Outfront AdServer та smartSCOUT	JCDecaux Data Solutions	RADAR Data Solutions
Доступність ПЗ на ринку	Ні	Ні	Ні

Продовження табл. 1.1

Можливість збору даних про аудиторії в конкретній локації	Так	Так	Так
Наявність інструменту для візуалізації даних	Так	Ні	Так
Можливість оцінки охоплення в конкретній локації	Так	Так	Так
Сумісність з відео потоками з різними параметрами	Ні	Так	Ні
Можливість аналізу даних про зовнішнє середовище в режимі реального часу	Ні	Так	Ні

1.4. Перелік вимог до програмного забезпечення

На основі проведених огляду проблеми використання даних про дорожній рух при оптимізації рекламних кампаній та аналізу існуючих програмних рішень було сформовано вимоги до програмного забезпечення у даному дипломному проєкті.

Функціональні вимоги:

- Збір інформації про кількість унікальних ТЗ, помічених камерою відеоспостереження за певний проміжок часу.
- Збір інформації про середню швидкість ТЗ, помічених камерою відеоспостереження за певний проміжок часу.

- Збір даних про колір, моделі та виробника ТЗ, помічених камерою відеоспостереження.
- Візуалізація зібраної інформації на вебресурсі.
- Можливість реєстрації та авторизації користувача вебресурсу.

Нефункціональні вимоги:

- Ефективність використання обчислювальних ресурсів під час обробки відео потоків.
- Ефективність використання обчислювальних ресурсів під час візуалізації інформації.
- Простота користувацького інтерфейсу вебресурсу.
- Висока точність зібраних даних.

1.5. Висновки до розділу 1

Під час підготовки першого розділу дипломного проєкту було виконано наступні задачі:

- проаналізовано ринок ПЗ для моніторингу транспортних потоків з метою оптимізації рекламних кампаній;
- встановлено перелік проблем, з якими стикаються маркетингові компанії, що не використовують подібних систем;
- проаналізовано декілька найпопулярніших програмних систем моніторингу транспортних потоків, що використовуються маркетинговими компаніями;
- виконано порівняльний аналіз існуючих програмних рішень, зазначено переваги та недоліки кожного з них;
- побудовано таблицю для порівняння результатів аналізу;
- сформовано перелік функціональних та нефункціональних вимог, яким має слідувати ПЗ, створюване у рамках даного дипломного проєкту.

2. ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Вибір мов програмування

У цьому підрозділі розглянуто популярні мови програмування, які використовуються для розробки клієнтських та серверних компонентів програмних систем, проведено аналіз їх переваг та недоліків.

2.1.1. *Огляд мови програмування Python*

Мова Python була створена Гвідо ван Россумом і вперше випущена у 1991 році як спадкоємиця мови ABC з метою дозволити програмістам писати зрозумілий, логічний код для малих і великих проєктів [14]. Остання версія даної мови, Python 3.0, була випущена у 2008 році, і станом на сьогодні мала 12 значних оновлень [15]. Наразі Python є третьою найпопулярнішою мовою програмування у світі. За результатами досліджень, у 2024 році з нею часто взаємодіють не менш ніж 49.27% розробників програмного забезпечення [16].

Найчастіше, мова Python використовується в галузях аналізу даних та машинного навчання [17], що пояснюється наявністю в ній цілої екосистеми бібліотек та фреймворків для роботи в цих галузях. Наприклад, такі бібліотеки, як NumPy та pandas, надають всі необхідні інструменти для ретельного аналізу даних [18]. Matplotlib та Seaborn є стандартним вибором для візуалізації даних, а SciPy – для наукових обчислень. Популярності ж Python в галузі машинного навчання сприяють бібліотеки та фреймворки для глибокого навчання, такі як TensorFlow та PyTorch. На останок зазначимо, що з ряду причин Python вважається достатньо повільною мовою [19]. Проте легкість інтеграції до мови модулів, написаних мовами C та C++ [20], дозволяє нівелювати цей недолік при роботі над фрагментами коду, де швидкість виконання є критично-важливим параметром.

Другою за популярністю галуззю, де використовується Python, є веброзробка. Цьому сприяє наявність таких фреймворків як Django та FastAPI, що допомагають спеціалістам у ефективній розробці безпечних і зручних для підтримки вебсайтів. Зокрема, фреймворк Django відомий своєю філософією оптимізації частих завдань у веброзробці, пропонує цілий набір готових до використання компонентів, таких як панель адміністрації вебсайту, модуль автентифікації, тощо [21]. В той же час, FastAPI є модульним фреймворком [22], надаючи розробникам більше контролю над компонентами, що вони використовують, та, відповідно, ресурсами, що споживатиме готовий вебзастосунок.

Варто зазначити, що у веброзробці мова Python використовується для написання серверної частини вебзастосунків: валідації вхідних даних, взаємодії з базами даних, маршрутизації URL-адрес, управління сесіями, тощо [23]. Написання ж клієнтських частин потребує використання інших інструментів.

Розглянемо більш детально переваги та недоліки мови Python.

Переваги:

- Вивчення та використання: мова Python має дуже простий синтаксис, що за своєю структурою схожий на натуральну англійську мову [24]. Така особливість суттєво спрощує поріг входу нових користувачів та підвищує ефективність роботи вже досвідчених розробників.
- Універсальність: Python підтримується багатьма системами і платформами, від популярних операційних систем до IoT- сенсорів, і може використовуватися в різних сферах, від веброзробки до машинного навчання.
- Розширюваність: такі системи управління пакетами, як pip і conda, налічують сотні тисяч модулів і бібліотек [25], що дозволяє

програмістам не продумувати власну алгоритмічну реалізацію для багатьох сценаріїв.

- Спільнота: популярність мови Python забезпечує їй наявність великої онлайн-спільноти. Завдяки цьому розробники можуть знайти відповіді на майже всі запитання та приклади розв’язання більшості проблем, що можуть виникати під час написання коду даною мовою.
- Сумісність з іншими мовами: Python підтримує інтеграцію зі сторонніми мовами програмування, такими як C та C++. Це дозволяє розробникам використовувати звичну для них мову Python навіть в тих проєктах, де для деяких сценаріїв можливостей даної мови може бути недостатньо.

Недоліки:

- Швидкодія: інтерпретованість мови Python робить її повільнішою за компільовані мови, такі як C, Java або GoLang.
- Однопотоковість: для роботи Python використовує технологію під назвою “Глобальне Блокування Інтерпретатору” (GIL) [26], що дозволяє лише одному потоку мати доступ до інтерпретатору мови у конкретний момент часу. Така особливість суттєво знижує ефективність коду, що виконується на ядрах центрального процесору.
- Споживання пам'яті: гнучкість і простота використання Python досягається за рахунок більшого споживання пам'яті порівняно з іншими мовами.
- Помилки під час виконання: інтерпретованість, та, відповідно, динамічна типізація мови, призводять до того, що значна частина помилок помічається лише після початку виконання коду.

2.1.2. *Огляд мови програмування JavaScript і її розширення TypeScript*

JavaScript – високорівнева динамічна мова, що була створена у 1995 році компанією Netscape, а останній стандарт цієї мови, відомий як ECMAScript 6, було випущено у 2015 році. JavaScript нерозривно пов'язана з веброзробкою, адже, за інформацією з відкритих джерел, цю мову використовують не менш ніж 99% вебсайтів [27]. Окрім цього, JavaScript вважається найпопулярнішою мовою програмування в світі. На момент проведення опитування, її активно використовували не менш ніж 63% опитаних розробників.

JavaScript лежить в основі сучасної фронтенд розробки. Завдяки можливостям взаємодії даної мови з HTML та CSS кодом вебсторінок, що підтримуються всіма сучасними браузерами [28], дана мова дозволяє спеціалістам створювати інтерактивні та динамічні користувацькі інтерфейси. Саме завдяки JavaScript програмісти реалізують на вебсторінках функціональність валідації форм, обробки подій, таких як кліки миші або натискання клавіш на клавіатурі, створення анімацій, тощо. Фреймворки та бібліотеки, такі як React, Angular та Vue.js, ще більше розширили можливості JavaScript, дозволивши створювати складні односторінкові додатки, які пропонують користувацький досвід, подібний до десктопних додатків. Такі фреймворки допомагають структурувати і оптимізувати код вебдодатків, позбувшись при цьому суттєвої кількості сегментів дубльованого коду, що неодмінно виникають при роботі над великими проєктами. Загалом, ці інструменти забезпечують надійну основу для створення масштабованих і підтримуваних клієнтських додатків.

Для багатьох спеціалістів суттєвим недоліком мови JavaScript є динамічна типізація. Така особливість мови може бути корисною при розробці невеликих додатків, проте, при роботі над проєктами, що містять мільйони рядків коду, записаного в тисячі файлів, розуміння того, яка

змінна в коді має який тип – може бути складним як для розробників, так а для алгоритмів IDE. Для вирішення даною проблемою, компанія Microsoft у 2014 році випустила розширення для мови JavaScript під назвою TypeScript, що впроваджує можливість статичної типізації і компіляції коду, суттєво спрощуючи процес роботи над великими проєктами. На сьогоднішній день, TypeScript активно підтримується, розповсюджується як ПЗ з відкритим кодом та вважається стандартом в індустрії.

Розглянемо більш детально переваги та недоліки мови JavaScript/TypeScript.

Переваги:

- Інтерактивність: вбудована в усі сучасні браузери можливість взаємодії JavaScript з DOM надає програмістам можливість робити вебсторінки динамічними та інтерактивними.
- Універсальність: JavaScript може використовуватись як для клієнтської, так і для серверної розробки. Така властивість є особливо корисною для невеликих проєктів, дозволяючи одному розробнику працювати як над клієнтською, так і над серверною частинами застосунку.
- Розширюваність: npm, найбільший менеджер пакетів для JavaScript, налічував більш ніж 2.1млн різноманітних модулів та бібліотек станом на 2022 рік [29].
- Спільнота: JavaScript є найпопулярнішою мовою програмування у світі, що надає даній мові величезну онлайн спільноту.
- Асинхронність: можливість виконання асинхронного коду є вбудованою в мову JavaScript і не потребує встановлення додаткових пакетів та бібліотек. Це дозволяє розробникам ефективно працювати зі сценаріями, в яких потрібно певний час очікувати на отримання даних зі сторонніх джерел, наприклад, з серверної частини вебзастосунку.

Недоліки:

- Швидкодія: JavaScript є інтерпретованою мовою. Відповідно код, написаний з використанням JavaScript, виконується повільніше ніж код, написаний з використанням компільованих мов.
- Однопотоковість: у своїй основі, JavaScript є однопоточною мовою. В якійсь мірі, це спрощує процес розробки коду даною мовою, проте така особливість також не дозволяє мові використовувати декілька ядер центральних процесорів, що суттєво впливає на швидкодію.
- Розміри готових продуктів: інтерпретованість мови та залежність від бібліотек нерідко змушують готові вебсайти використовувати декілька мегабайтів JavaScript коду. Це є проблемою, адже увесь код має пересилатись мережею Інтернет від сервера, де розгорнуто вебсайт, до клієнтів вебзастосунку. Збільшення розмірів цього коду призводить до сповільнення завантаження вебсайту та посилення навантаження на мережу Інтернет.
- Залежність від TypeScript: майже всі сучасні клієнтськи додатки розроблюються з використанням TypeScript. Той факт, що TypeScript не є частиною офіційного JavaScript, змушує розробників витрачати час на встановлення та налаштування його на своїх проєктах.

Python і JavaScript/TypeScript – це дві популярні мови програмування зі своїми унікальними особливостями. Мова Python використовується для широкого кола задач, включаючи веброзробку, аналіз даних та штучний інтелект. JavaScript і TypeScript, з іншого боку, широко використовуються у веброзробці, особливо для динамічного створення вебсайтів. TypeScript надає більшу надійність завдяки статичній типізації, тоді як JavaScript використовується для швидкого прототипування з розширеними функціональними можливостями завдяки широкому спектру бібліотек і фреймворків (Табл. 2.1).

Таблиця 2.1

Порівняння мов програмування Python, JavaScript/TypeScript

Критерій	Python	JavaScript
Опис	Динамічна інтерпретована високорівнева мова, що переважно використовується для розробки серверних додатків	Динамічна інтерпретована високорівнева мова, що переважно використовується для розробки клієнтських додатків
Простота вивчення	Так	Ні
Багатопоточність	Ні	Ні
Швидкодія	Частково	Ні
Сумісність з іншими мовами програмування	Так	Ні
Кросплатформеність	Так	Так
Статична типізація	Ні	Частково
Можливість розробки клієнтських додатків	Ні	Так
Можливість розробки серверних додатків	Так	Так
Використання в науці про дані	Так	Ні

2.2. Вибір СУБД

Даний підрозділ присвячено СУБД, що часто використовуються при побудові клієнт-серверних програмних систем.

2.2.1. *Огляд СУБД PostgreSQL*

PostgreSQL – потужна об’єктно-реляційна СУБД з відкритим кодом, що була випущена командою розробників з Каліфорнійського Університету Берклі в 1996 році. Дана СУБД створювалась для розширення можливостей свого попередника, Ingres за межі, що встановлюються стандартними реляційними моделями [30]. На сьогоднішній день, PostgreSQL є однією з найпопулярніших СУБД у світі і використовуються в різноманітних галузях – від веб та десктопних застосунків до систем обробки геопросторових даних та управління IoT мережами.

Однією з головних особливостей PostgreSQL є підтримка двигуном бази даних операцій над неструктурованими та напівструктурованими даними. Наприклад, дана СУБД відома своїми можливостями роботи з JSON, XML та геопросторовими даними [31], для яких, за інших умов, доводиться використовувати окрему NoSQL СУБД.

PostgreSQL є повністю відповідною до стандартів ANSI SQL, що включає в себе підтримку складних SQL запитів та підзапитів, транзакційність, тощо. Окрім цього, PostgreSQL надає розробникам широкі можливості індексації даних, що включають до себе контроль за структурою індексів, можливість повнотекстової індексації, а також підтримку фільтрованих та часткових індексів. Всі ці можливості роблять PostgreSQL одним з найкращих варіантів при роботі з великими об’ємами даних.

Іншою перевагою PostgreSQL є підтримка паралельного виконання декількох транзакцій завдяки механізму мультиверсійного контролю паралельних виконань (MVCC). На відміну від інших баз даних, де обробка декількох паралельних транзакцій в одній таблиці призводить до зачинення рядків даних, змушуючи інші транзакції, що потребують доступу до цих рядків, чекати на завершення попередніх, транзакції PostgreSQL використовують снепшоти рядків (знімки або версії текстових рядків у репозиторії контролю версій, які використовуються для відстеження змін

для збереження стану файлів на певний момент часу та для порівняння з попередніми версіями), що суттєво зменшує використання замків та, відповідно, частоту конфліктів між декількома транзакціями [32].

На останок, варто зазначити можливості масштабованості PostgreSQL. Дана СУБД підтримує як стандартне для SQL баз даних вертикальне масштабування, так і більш складне горизонтальне. Без потреби у встановленні додаткових модулів, PostgreSQL дозволяє проводити реплікацію баз даних у різних форматах – multi-master, master-slave, тощо. Окрім цього, PostgreSQL фрагментувати таблиці, по-суті розділяючи великі таблиці на набір малих, що сприймаються двигуном як одне ціле. Сторонні модулі PostgreSQL дозволяють забезпечити балансування навантаження на бази даних. Утиліти по типу PgBouncer та PgPool були створені для перенаправлення потоків запитів на з'єднання з БД до різних клієнтів бази даних, або навіть до різних кластерів.

Розглянемо більш детально переваги та недоліки СУБД PostgreSQL.

Переваги:

- Цілісність даних: завдяки дотриманню принципів ACID та використанню MVCC, дана СУБД забезпечує користувачів цілісними даними в будь-який момент часу.
- Розширюваність: незважаючи на те, що навіть стандартний варіант PostgreSQL підтримує виконання великої кількості різноманітних функцій, модель великого коду забезпечує дану СУБД сторонніми модулюми, що покривають функціональність, яка є відсутньою в стандартному варіанті.
- Різноманітність даних: PostgreSQL оптимізована для роботи зі значною кількістю як примітивних, так і складних типів даних, від числових значень та рядків тексту в різних кодуваннях до масивів, геопросторових та документних моделей.

- Масштабованість: розробники можуть легко масштабувати PostgreSQL відповідно до потреб програмних систем як вертикально, додаючи більше ресурсів до серверів, так і горизонтально – завдяки реплікації, фрагментації та балансувальникам навантаження.

Недоліки:

- Споживання пам’яті: снепшоти даних, що створюються механізмом MVCC під час роботи, зберігаються в пам’яті сервера, збільшуючи її споживання під час роботи СУБД.
- Складність: через велику кількість можливостей СУБД, деякі програмісти можуть губитися при роботі з нею. Особливо це стосується розробників-початківців.
- Реплікація: незважаючи на їхню наявність в стандартному варіанті СУБД, механізми реплікації в PostgreSQL налаштовуються складніше ніж в деяких інших СУБД, таких як MySQL.

2.2.2. *Огляд СУБД SQL Server*

SQL Server або Microsoft SQL Server – пропрієтарна реляційна СУБД, що була розроблена та підтримується компанією Microsoft. SQL Server була випущена у 1989 році і на даний момент є найпопулярнішею СУБД для великих комерційних застосунків.

Головною перевагою SQL Server є оптимізація двигуна для роботи з запитами різної складності над таблицями великого розміру. Це досягається завдяки можливостям двигуна оптимізовувати запити для досягнення ними максимальної швидкодії, будувати плани виконання запитів, що кешуються та повторно використовуються для наступних запитів, а також автоматичній фрагментації таблиць, що дозволяє зменшити кількість нерелевантних рядків, які має прочитати двигун перед виконанням запиту.

На відміну від PostgreSQL, СУБД SQL Server розповсюджується в декількох виданнях, оптимізованих під різні бізнес-потреби. Серед звичайних форматів варто зазначити Express та Standard, що оптимально підходять для проєктів невеликих та середніх розмірів, а також Enterprise, оптимізований під потреби великих компаній, що допомагає спеціалістам управляти базами даних розміром до 524ПБ [33]. Серед спеціалізованих видань можна визначити SQL Server Azure, що було розроблено спеціально для використання в хмарі Microsoft Azure, SQL Server APS, оптимізований для аналітики сотень терабайт даних, а також SQL Server Developer, ліцензійна угода дозволяє використання СУБД лише для локальної розробки або тестування.

Як і PostgreSQL, СУБД SQL Server підтримує одночасне виконання декількох транзакцій на одних й тих самих рядках таблиці. Досягається така функціональність завдяки підходу під назвою «оптимістичне зачинення рядків» [34]. Відповідно до такого підходу, рядок таблиці може зчитуватись декількома транзакціями одночасно, проте, лише одна з читаючих транзакцій зможе його модифікувати. Окрім цього, в SQL Server використовується спеціальний менеджер зачинень рядків, що зберігає в пам’яті таблицю всіх об’єктів та замків в базі даних і допомагає розв’язувати конфліктні ситуації між декількома потоками, зазвичай, знищуючи при цьому один з потоків та відкатуючи транзакцію.

Для масштабування, SQL Server використовує реплікацію. На день написання даної роботи, ця СУБД підтримує 3 різних типи реплікацій – реплікацію транзакцій, реплікацію злиттям та реплікацію снепшотами. Реплікація транзакцій працює за схемою master-slave. В ній кожна транзакція зроблена в master БД оновляє slave бази даних. Реплікація злиттям працює за схемою multimaster, де зміни ззовні вносяться до декількох master баз даних і періодично синхронізуються між всіма БД. На

останок, реплікація снепшотами передбачає публікацію копії основної БД без відстеження подальших змін в ній.

Розглянемо більш детально переваги та недоліки СУБД SQL Server.

Переваги:

- Швидкодія: експертиза розробників БД та фокус на роботу з великими масивами даних роблять SQL Server однією з найшвидших SQL баз даних.
- Оптимізація під різні сценарії: SQL Server розповсюджується у декількох форматах, оптимізованих під різні потреби – від аналітики до машинного навчання.
- Інтеграція з іншими продуктами Microsoft: багато інших відомих продуктів від Microsoft, таких як .NET Core, Excel та PowerBI одразу підтримують інтеграцію SQL Server як провайдера даних.
- Профілювання: SQL Server має одну з найякісніших систем профілювання серед усіх SQL баз даних. Такий результат досягається завдяки вбудованому профілюванню подій, наявності інструментів для перегляду планів виконання, збору статистики з таблиць та індексів, тощо.

Недоліки:

- Ціна: SQL Server не є продуктом з відкритим кодом. Відповідно, майже усі формати, в яких розповсюджується SQL Server, є платними.
- Складність: велика кількість різноманітних функцій та інструментів, що надає SQL Server, ускладнює роботу з нею спеціалістів. Особливо актуальним це є для розробників-початківців.
- Споживання ресурсів: не в останню чергу, швидкодія SQL Server досягається шляхом споживання значної кількості ресурсів сервера. Варто зазначити, що ця проблема може частково нівелюватись шляхом розгортання БД в хмарі.

2.2.3. *Огляд СУБД Redis*

Redis – це NoSQL СУБД, що зберігає дані відповідно до моделі ключ-значення. Дана СУБД розповсюджується відповідно до моделі доступного коду і переважно розробляється компанією Redis Labs.

На сьогодні Redis вважається однією з найшвидших СУБД у світі [35]. Швидкодія цією СУБД досягається завдяки постійному зберіганню всіх даних у пам’яті, а не на диску. Це робить Redis популярним вибором при роботі над сценаріями, в яких потрібно зберігати дані, що швидко втрачають свою релевантність, доступ до яких має бути якнайвшидшим, а втрата яких не є критичною. Варто зазначити, що зберігаючи дані у форматі ключ-значення, Redis підтримує лише дуже примітивні пошукові запити.

Будучи NoSQL базою даних, Redis підтримує декілька форматів зберігання даних, оптимізовані під різні потреби. Одразу варто зазначити, що в якості ключів Redis підтримує виключно рядки. Проте значення можуть бути числами з плаваючою крапкою, рядкими, масивами та наборами, хеш-мапами, бінарними послідовностями, тощо. Також, за допомогою сторонніх модулів, Redis може зберігати числові послідовності, де значеннями є виключно числа з плаваючою крапкою, вектори повнотекстового пошуку, JSON документи та інші формати даних.

Для підвищення ступеня цілісності даних Redis записує дані з пам’яті на диск [36]. В даній СУБД це може відбуватись в одним з двох методів, відповідно до налаштувань самої СУБД. Перший метод передбачає періодичне створення снепшотів даних з пам’яті та запису їх на диск, а другий – виконання запису на диск у неосновному потоці після кожної модифікації даних в пам’яті.

Масштабування в СУБД Redis забезпечується завдяки фрагментації та реплікації. Фрагментація дозволяє розділити дані, що зберігаються в Redis, на декілька різних вузлів, що зазвичай представляють з себе різні сервери. Такі вузли переважно розташовані неподалік один від одного – в одній

локальній мережі або в межах одного датацентру для забезпечення максимальної швидкодії. Реплікація в Redis переважно використовується для прискорення операцій читання шляхом копіювання даних з головного вузла до вузлів реплік. Зазначимо, що Redis також підтримує реплікацію multi-master, хоча така можливість доступна лише в комерційному виданні Redis під назвою Redis Enterprise.

Таблиця 2.2

Порівняння СУБД PostgreSQL, SQL Server, Redis

Критерій	PostgreSQL	SQL Server	Redis
Тип	Об’єктно-реляційна	Реляційна	Нереляційна, модель ключ-значення
Швидкодія	Висока	Висока	Надвисока
Споживання пам’яті	Низьке	Помірне	Високе
Споживання ресурсів процесору	Помірне	Високе	Низьке
Відповідність ACID	Відповідна	Відповідна	Невідповідна
Підтримка складних запитів	Наявна	Наявна	Відсутня
Горизонтальне масштабування	Обмежене	Наявне	Наявне
Обмеження безкоштовної ліцензії	Відсутні	Можливість виключно локальної розробки	Відсутність доступу до multi-master реплікації

Продовження табл. 2.2

Можливість профілювання запитів	Наявна частково	Наявна	Відсутня

2.3. Вибір брокеру повідомлень

В сучасній інженерії програмного забезпечення брокери повідомлень є стандартним методом забезпечення асинхронної комунікації між компонентами програмної системи [37]. Даний підрозділ присвячено розгляду таких брокерів.

2.3.1. *Брокер повідомлень RabbitMQ*

RabbitMQ – це брокер повідомлень з відкритим кодом, що використовує протокол AMQP для передачі даних. Першу версію даного брокера було випущено у 2007 році, а на сьогоднішній день більшість сучасних мов програмування мають бібліотеки для роботи з RabbitMQ.

Головною перевагою RabbitMQ є розвинена система маршрутизації повідомлень. Відповідно до своєї архітектури, RabbitMQ працює з обмінниками та чергами, підключеними до них. Коли відправник надсилає повідомлення в RabbitMQ, він має вказати ключ маршрутизації та назву обмінника, який має обробити повідомлення. Коли повідомлення доходить до брокера, RabbitMQ знаходить релевантні для повідомлення черги, відповідно до налаштувань обмінника. Це можуть бути всі черги, підключені до обмінника, що є корисним коли повідомлення описує подію, що сталась в системі. Іншим варіантом є одна черга, назва якої чітко співпадає з ключем маршрутизації, що є характерним коли повідомлення описує певну команду. Також, обмінник може знайти декілька черг, що відповідають шаблону, вказаному в ключі маршрутизації. Після знаходження всіх потрібних черг, обмінник передає до них отримане повідомлення для його подальшого прийому споживачами.

Черги в RabbitMQ працюють за принципом FIFO [38]. Це допомагає підтримувати цілісність даних при асинхронній комунікації, забезпечуючи те, що повідомлення будуть оброблені в тому порядку, в якому вони були надіслані. Також, стандартна версія RabbitMQ підтримує чергу «мертвих» повідомлень. Розробник системи може вказати умови, за виконання яких повідомлення вважатиметься таким, що не може бути опрацьоване, та буде відправлене до цієї черги для подальшого аналізу.

RabbitMQ підтримує горизонтальне масштабування. Обмінники та черги в RabbitMQ можуть бути розміщені на декількох вузлах системи, що разом утворюють кластер. Самі черги також підлягають фрагментації, що дозволяє знизити навантаження на вузли, що містять черги з високою пропускною здатністю. Суттєвим недоліком горизонтального масштабування в RabbitMQ є відсутність вбудованого балансувальника мережевого навантаження на вузли кластера, що вимагає від програміста використання стороннього балансувальника, такого як NGINX.

Розглянемо детальніше переваги та недоліки брокера RabbitMQ.

Переваги:

- Розвинена система маршрутизації: набір налаштувань та фільтрів дозволяє розробникам надсилати повідомлення в самі ті черги, які на ці повідомлення очікують.
- FIFO формат черг: повідомлення, що були надіслані брокеру раніше за інші, завжди потраплять до обробки також раніше за інші повідомлення.
- Підтримка черги мертвих повідомлень: наявність такої черги дозволяє програмістам проводити більш ефективний аналіз помилок системи.
- Необмежена кількість споживачів: на відміну від багатьох інших брокерів, повідомлення з черг RabbitMQ можуть отримуватись багатьма споживачами.

Недоліки:

- Швидкодія: складна система маршрутизації сповільнює швидкодію системи. Варто зазначити, що цей недолік може частково нівелюватись можливістю декількох споживачів асинхронно читати повідомлення.
- Відсутність балансувальника мережевого навантаження: кластер RabbitMQ потребує використання стороннього балансувальника.
- Складність: через свою систему маршрутизації, RabbitMQ є складнішим за більшість інших брокерів повідомлень.

2.3.2. *Брокер повідомлень Apache Kafka*

Apache Kafka або Kafka – брокер повідомлень та платформа передачі потоків, оптимізована для забезпечення найменшої затримки та найвищої пропускної здатності для обробки повідомлень [39]. Для своєї роботи Kafka використовує власний протокол, що було створено на основі TCP.

У своїй архітектурі Kafka використовує теми, які розбиті на фрагменти. Коли відправник надсилає повідомлення, він вказує тему, до якої повідомлення буде передано, та опціонально – ключ фрагмента. За умови наявності ключа, Kafka передає повідомлення до фрагмента, назва якого співпадає з ключем. За відсутності ключа, повідомлення потрапляє до фрагмента з найменшим навантаженням. Споживачі в брокері Kafka є об’єднаними в так звані групи споживачів. Для забезпечення максимальної швидкодії повідомлення з кожного фрагменту в Kafka можуть споживатись не більш ніж одним споживачем з кожної групи. Також, на відміну від RabbitMQ, після споживання повідомлення не видаляються з брокера. Пізніше, їх може прочитати споживач з іншої групи.

Фрагменти в Kafka забезпечують порядок FIFO для запису та споживання повідомлень. Варто зазначити, що через зберігання

повідомлень навіть після його споживання Kafka не підтримує чергу «мертвих» повідомлень.

Kafka підтримує і заохочує горизонтальне масштабування [40]. Даний брокер автоматично підтримує повторне балансування повідомлень між фрагментами, у відповідь на збільшення чи зменшення кількості вузлів, а також підтримує реплікацію фрагментів для підвищення пропускної здатності при обробці запитів на споживання повідомлень.

Розглянемо детальніше переваги та недоліки брокера Kafka.

Переваги:

- Пропускна здатність: брокер Kafka оптимізовано для забезпечення максимальної пропускної здатності. Це робить його оптимальним варіантом для роботи в сценаріях обробки повідомлень в режимі реального часу.
- FIFO формат фрагментів: повідомлення, що були надіслані до конкретного фрагменту раніше за інші, завжди потраплять до обробки також раніше за інші повідомлення.
- Зберігання повідомлень після споживання: така особливість даного брокера дозволяє декільком групам споживачів прочитати одне й те саме повідомлення. Це може бути корисним, наприклад, у сценарії наявності одної групи споживачів, що оброблюють повідомлення, та наявності іншої групи, що проводять аналітику вхідних даних.
- Масштабованість: Kafka було створено для роботи в великих кластерах з тисячами вузлів. Стандартна версія даного брокера містить всі необхідні інструменти для такого масштабування – від балансувальника мережевого навантаження до балансувальника фрагментів.

Недоліки:

- Ситуативність: брокер Kafka було оптимізовано для сценаріїв, що вимагають максимально високої пропускної здатності. Через це він

є менш гнучким і може бути менш зручним ніж інші брокери, такі як RabbitMQ, за звичайних умов.

- Маршрутизація: кожне повідомлення в Kafka потрапляє виключно до одного фрагменту. Це робить даний брокер несумісним з такими сценаріями як, наприклад, передача подій декільком підписникам.

Таблиця 2.3

Порівняння брокерів повідомлень RabbitMQ та Kafka

Критерій	RabbitMQ	Kafka
Налаштування маршрутизації	Так	Частково
Пропускна здатність	Тисячі повідомлень на секунду	Мільйони повідомлень на секунду
Горизонтальне масштабування	Частково	Так
Видалення повідомлень після споживання	Так	Ні
FIFO в чергах	Так	Так

2.4. Додаткові технології та бібліотеки

2.4.1. *Огляд бібліотеки VueJS*

VueJS – це широко використовуваний фреймворк JavaScript, що призначений для полегшення процесу розробки інтерфейсів користувача. Даний фреймворк був створений незалежними розробниками та розповсюджується за моделлю з відкритим кодом.

Схожим чином на інші SPA-фреймворки, VueJS забезпечує декларативний підхід до створення інтерфейсу користувача, що можуть бути повторно використані на різних псевдосторінках застосунку. При порівнянні з іншими популярними фреймворками, такими як React та

Angular, VueJS є суттєво простішим для вивчення. Також VueJS не потребує додаткового шару абстракції для компіляції проміжної мови в JavaScript чи TypeScript як React з JSX. Окрім цього, VueJS має найвищу швидкодію серед усіх трьох перелічених фреймворків [41] завдяки системі відслідковування залежностей, що працює схожим чином, як було реалізовано в Angular, мінімізуючи кількість оновлень елементів в DOM при перезавантаженні інтерфейсу. Ці особливості роблять VueJS оптимальним варіантом для розроблення інтерфейсів малих та середніх за розміром програмних систем.

Протягом декількох років існування фреймворку, навколо VueJS сформувалась ціла екосистема інструментів та бібліотек для розширення його базових можливостей. Офіційна бібліотека Vue Router забезпечує маршрутизацію для SPA-застосунків, написаних на VueJS, дозволяючи користувачам переходити між різними псевдосторінками застосунку так, наче вони були б окремими HTML-сторінками. Vite є офіційним збірником для VueJS застосунків, що розширює та оптимізує процеси збірки такими інструментами як Webpack конкретно під потреби фреймворку. Рекомендованим менеджером станів для Vue застосунків вважається Pinia, що відносно нещодавно замінила Vuex завдяки спрощенню процесів менеджменту станів, підтримки розширення TypeScript, наявності системи підключення плагінів та підтримки технології SSR.

2.4.2. *Огляд фреймворку FastAPI*

FastAPI – сучасний вебфреймворк для побудови API з використанням мови Python, що було вперше випущено у 2018 році. На сьогоднішній день, він вважається одним з найшвидших серверних фреймворків, адже API, написані за допомогою FastAPI, є лише трохи повільнішими за ті, що написані за використання компільованої мови Go [42].

FastAPI надає розробникам всі інструменти, необхідні для створення веб-API. По-перше, даний фреймворк включає в себе інструменти для

валідації та серіалізації вхідних і вихідних даних, на які зазвичай припадає найбільша кількість коду в застосунку при створенні таких інструментів вручну. Також FastAPI забезпечує розробників зручною системою маршрутизації, що здатна перенаправляти запити на різні кінцеві точки застосунку в залежності від конфігурації, наданої програмістам. На останок, варто зазначити сумісність даного фреймворку зі стандартом OpenAPI та підтримку SwaggerUI його стандартною версії. Така особливість допомагає розробникам візуалізувати кінцеві точки розроблюваного застосунку та відправляти запити до них просто зі сторінки веббраузеру.

Унікальною перевагою FastAPI є вбудована система ін'єкції залежностей. Така особливість надає програмістам можливість створювати одну вхідну точку для застосунку, що містить усю необхідну конфігурацію. Окрім цього, в архітектуру FastAPI закладено підтримку асинхронного програмування, що забезпечує більш ефективну обробку довгих операцій I/O [43]. Це є суттєвою відмінністю FastAPI від таких фреймворків як Flask та Django, що переважно, оброблюють усі запити синхронно.

2.5. Висновки до розділу 2

У даному розділі було оглянуто інструменти та технології, що використовуються у розробці сучасних серверних та клієнтських частин програмних систем. Огляд включав порівняння даних інструментів між собою та аналіз їхніх сильних та слабких систем. На основі результатів аналізу було вибрано засобу реалізації програмної системи для інтелектуального аналізу даних дорожнього руху.

При розробці серверної частини використовуватиметься мова Python. Такий вибір було зроблено з урахуванням того, що серверні компоненти системи повинні мати змогу проводити обробку даних з використанням методів машинного навчання, зберігати ці дані до стійких або нестійких сховищ та передавати дані зовнішнім компонентам, використовуючи

синхронні та асинхронні методи комунікації. Забезпечення усіх трьох наведених вимог було б проблематичним за використанням мов програмування, окрім Python.

За фреймворк для розробки API було обрано FastAPI. Він є сучасним, швидким, забезпечує розробників комплектом інструментів для побудови якісних API, а також містить мінімум функціональності, що не використовуватиметься на даному проєкті, наприклад, MVC компонентів Django.

Клієнтська частина розроблюватиметься мовою JavaScript з використанням розширення TypeScript та за допомогою фреймворку VueJS. Така комбінація технологій допомагає в написанні структурованого коду з передбачуваною поведінкою та мінімум шаблонних компонентів, що не впливають на бізнес-логіку застосунку.

Для зберігання даних було обрано дві різні СУБД. В даному проєкті, стійке сховище даних забезпечуватиметься PostgreSQL. Дана база даних є повністю відповідною стандартам ACID та ANSI SQL, має бібліотеки для інтеграції з Python, є простішою для користування та споживає менше ресурсів, ніж SQL Server, а також має підтримку паралельного виконання транзакцій. Це робить її оптимальним варіантом для даного проєкту. Окрім PostgreSQL, в якості сховища для короткочасного зберігання даних було обрано Redis. Такий вибір було зроблено з урахуванням надвисокої швидкодії, простим API а також можливістю кластеризації даної СУБД.

Програмний продукт, розроблений як результат завершення даного проєкту, використовуватиме два брокери повідомлень – Kafka і RabbitMQ. Через перший брокер Kafka камери відеоспостереження передаватимуть потоки відео системі для обробки в режимі реального часу. Другий брокер використовуватиметься для передачі фрагментів оброблених потоків іншим компонентам системи для обробки поза режимом реального часу.

3. АРХІТЕКТУРА ПРОГРАМНИХ МОДУЛІВ

3.1. Загальний опис архітектури

Для реалізації програмного забезпечення для інтелектуального аналізу даних дорожнього руху з використанням камер відеоспостереження було обрано клієнт-серверну та P2P архітектури.

Клієнт-сервера архітектура – модель організації комп'ютерних мереж, що розділяє вузли мережі на 2 категорії – клієнти та сервери. В даній моделі, вузли-клієнти надсилають запити на отримання або оброблення інформації до вузлів-серверів, а вузли-сервери їх опрацьовують. Опрацювання запитів в більшості випадків відбувається синхронно, адже працездатність вузлів-клієнтів залежить від даних, що отримуються від вузлів-серверів. При цьому, самі вузли-сервери жодним чином не комунікують між собою. Така модель є стандартною при розробці вебзастосунків [44], забезпечуючи розподіл обов'язків між клієнтами та серверами таким чином, що клієнти фокусуються на управлінні довгими користувацькими сесіями, а сервери – на управлінні даними та інфраструктурними ресурсами системи.

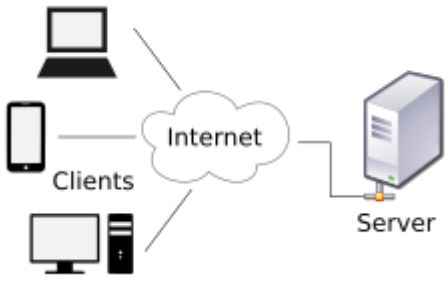


Рис. 3.1. Клієнт-серверна модель [45]

P2P архітектура – модель організації мереж, за яким всі вузли мережі є рівноправними та самодостатніми. За даної організації, кожен вузол фокусується на виконанні конкретних завдань та може працювати за відсутності або недоступності інших вузлів. Самі ж вузли можуть

комунікувати між собою, проте, на відміну від вузлів клієнт-серверних мереж, де комунікація відбувається синхронно, вузли P2P мереж зазвичай комунікують між собою асинхронно, використовуючи певну програму-посередник для передачі даних. Такий підхід підвищує рівень незалежності вузлів P2P мереж, адже їм ніколи не доводиться чекати зовнішньої реакції на дані, що надсилаються з цих вузлів до загальної мережі. Загалом, P2P модель є стандартом при побудові розподілених програмних систем [46].

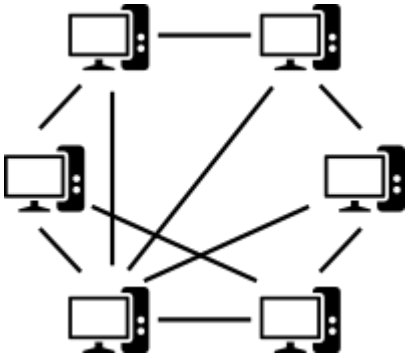


Рис. 3.2. P2P модель [47]

У розробленій програмній системі, користувач підключатиметься до клієнтської частини, SPA-застосунку, що для отримання даних синхронно відправлятиме запити до одного з кластерів серверних модулів. При цьому, серверні модулі, що належать до різних кластерів та, відповідно, незалежно один-від-одного виконують свої обов’язки, комунікуватимуть між собою асинхронно, за допомогою повідомлень.

3.2. Огляд модулів програмного забезпечення

У структурі побудованого програмного забезпечення можна виділити наступні модулі:

- модуль оброблення відео потоків;
- модуль відбору інформації;
- модуль візуалізації даних.

Модуль оброблення відео потоків і модуль відбору інформації вважаються автономними вузлами мережі побудованого програмного забезпечення та використовують брокер повідомлень для комунікації між собою. Модуль візуалізації даних вважається вузлом-клієнтом в розрізі клієнт-серверної моделі. Його вузлом сервером є модуль відбору інформації. Комунікація між цими двома модулюми відбувається без використання програми-посередника, за допомогою протоколів HTTP або HTTPS.

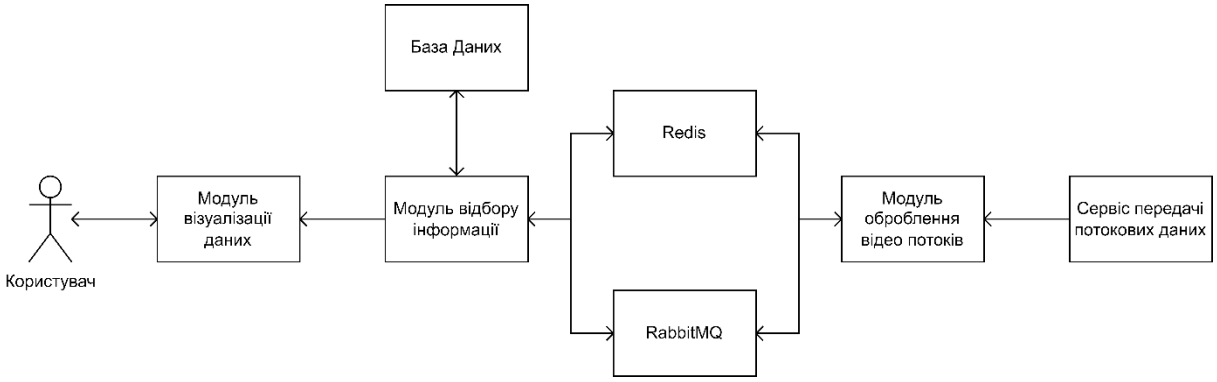


Рис. 3.1. Загальна архітектура програмного забезпечення

3.2.1. *Модуль оброблення відео потоків*

Модуль оброблення відео потоків напряму працює з потоками з камер відеоспостереження. Даний модуль виконує транскодування вхідного потоку, витягує дані з відео кадрів, розділяє їх на ті, що мають оброблюватись в режимі реального часу та ті, що мають оброблюватись в режимі близького до реального часу, і відправляє їх в мережу програмного забезпечення для подальшої обробки. Робота даного модулю виконується в режимі реального часу відносно до транскодованого потоку.

Транскодування

Відео потоки з різних камер відео спостереження можуть мати різну часту та формат кадрів. Для забезпечення стабільного та передбачуваного

навантаження на систему, дані параметри приводяться до наперед визначених стандартів.

Частота кадрів обмежується до 20 кадрів на секунду. Експериментальним шляхом було встановлено, що такий показник є достатньо високим для отримання достатньої кількості знімків кожного автомобіля-учасника дорожнього руху навіть за умов високої швидкості або великої кількості автомобілів на дорозі.

При форматуванні кадрів, модуль оброблення відео потоків зменшує їх таким чином, щоб розмір найбільшої сторони кадру не перевищував 640 пікселів, зберігаючи при цьому пропорції. Значення 640 вважається стандартним розміром [48] найбільшої сторони вхідних зображень для моделей комп'ютерного зору, що побудовані за архітектурою YOLOv8 [49]. Відповідно, форматування кадрів до такого формату дозволяє зменшити кількість пам'яті, що споживалась би програмним забезпеченням для зберігання зображень більшого розміру, не жертвуючи при цьому точністю моделей машинного навчання, що використовує програмне забезпечення.

Витягування даних з кадрів

На кожному кадрі відео потоку, модуль оброблення розпізнає транспортні засоби різних типів – легкові та вантажні автомобілі, мотоцикли, автобуси, тощо. Для цього використовується згорткова нейронна мережа побудована за архітектурою YOLOv8s та натренована на наборі даних COCO [50].

Після розпізнавання транспортних засобів, модуль відстежує їхнє переміщення відносно до попередніх кадрів, співвідносячи кожен розпізнаний ТЗ з унікальним в межах відео потоку ідентифікатором. Ці ідентифікатори зберігаються за транспортними засобами, між різними кадрами, навіть за умов втрат конкретних ТЗ на деяких кадрах.

Наявність ідентифікаторів дозволяє програмному забезпеченню співвідносити атрибути даних з конкретними транспортними засобами та проводити більш якісний аналіз даних. Головним же недоліком такого підходу є необхідність обробки всього потоку одним і лише одним пристроєм, адже обробка кожного поточного кадру потребує інформації з попередніх кадрів. За метод співвідношення транспортних засобів з ідентифікаторами було обрано алгоритм множинного відстеження ByteTrack [51].

Розділення і відправлення даних

Коли дані були витягнуті з кадру, модуль оброблення відео потоків публікує їх для аналізу іншими частинами програмного забезпечення. Опубліковані дані включають в себе часову мітку моменту закінчення витягування даних, порядковий номер кадру відео потоку, а також масив координат прямокутних областей зі співставленими до них ідентифікаторами, що були отримані з кадру.

Окрім публікації, дані з кадру відео потоку кешуються модулем оброблення потоків, разом з самим кадром. Як тільки кеш починає містити дані з більш ніж 20 кадрів, модуль стискає послідовність кадрів в кеші до формату mp4, формує повідомлення з них та закешованого масиву даних з кадрів, відправляє повідомлення до брокера повідомлень для подальшого оброблення поза режимом реального часу та очищує кеш.

3.2.2. *Модуль відбору інформації*

У розробленому програмному забезпеченні, модуль відбору інформації відповідає за отримання з кадрів даних, що не аналізуються в режимі реального часі, зберігання всіх отриманих даних з кадру до стійкого сховища, проведення аналізу історичних даних і даних, що отримуються в

режимі реального часу, а також оброблення запитів зовнішніх систем. Відповідно, даний модуль можна розділити на 5 логічних компонентів:

- компонент отримання даних;
- компонент зберігання даних;
- компонент оброблення запитів зовнішніх систем;
- компонент аналізу історичних даних;
- компонент аналізу даних в режимі реального часу.

На рисунку 3.3 наведено ERD-діаграму БД, що використовується модулем.

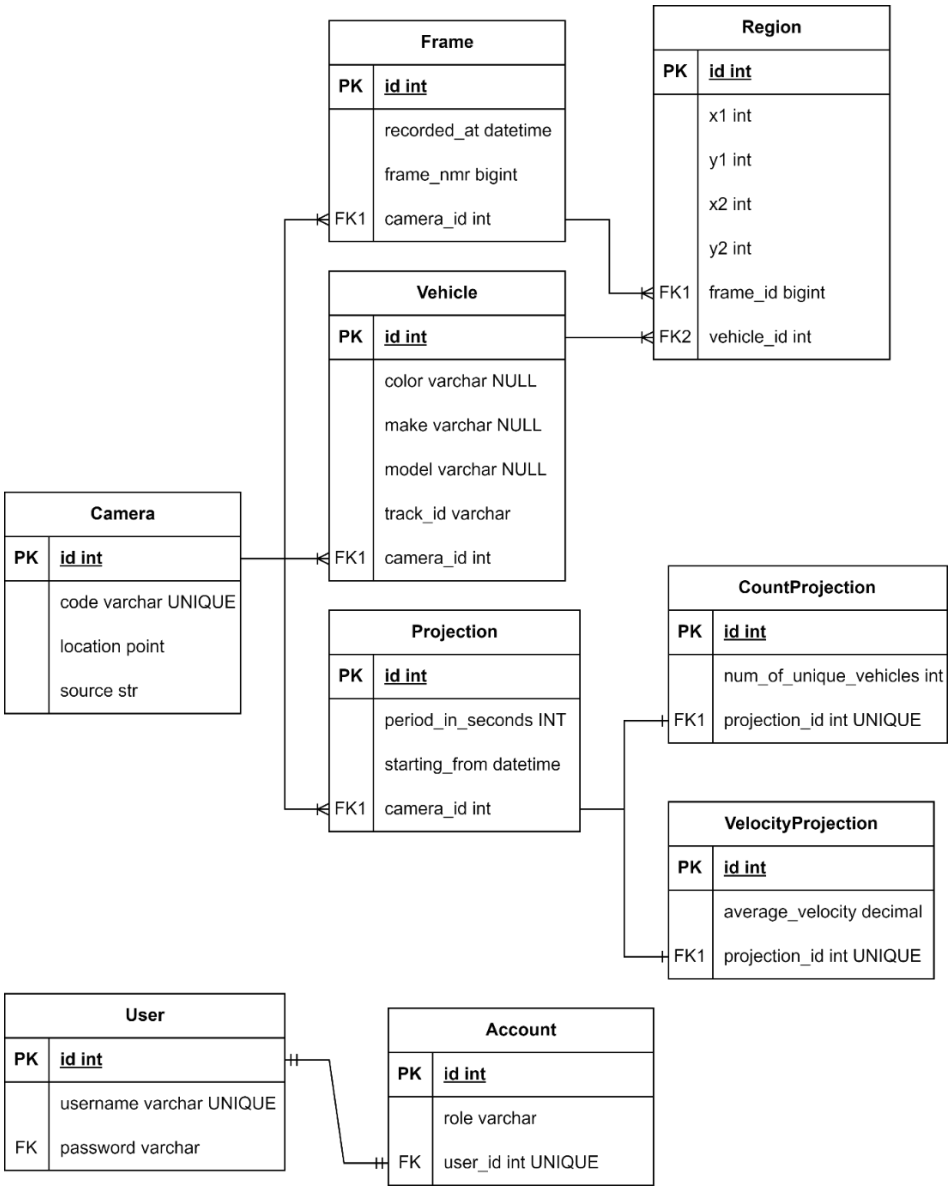


Рис. 3.3. ERD-діаграма БД

Компонент отримання даних

Деякі атрибути даних, що збирає система, використовуються виключно для збагачення портрету аудиторії в конкретних локацій. В розробленому ПЗ такими даними є значення кольору, марки та моделі ТЗ. Оскільки процес збагачення портрету зазвичай проводиться під час історичного аналізу даних, збирання зазначених атрибутів даних в режимі реального часу лише додаватиме навантаження на модуль оброблення відео потоків, не несучи при цьому жодних переваг. Зважаючи на це, в розробленому ПЗ обов'язок збирання зазначених даних о кольорі, марок та моделей ТЗ було призначено модулю відбору інформації, а конкретніше, компоненту отримання даних.

Отримання натурального кольору автомобіля за його зображенням в кадрі не є тривіальною задачею. Зовнішні чинники, такі як погодні умови, час доби та світлове забруднення міста суттєво впливають на сприйняття кольору камерою відео спостереження. На рисунку 3.4 наведено приклад зображення в двох варіантах – до (варіант a) та після (варіант b) нейтралізації туману алгоритмічними методами.

Для підвищення точності сприйняття кольорів на кадрах відео потоку, компонент отримання даних проводить обробку кадру для нейтралізації деяких зовнішніх чинників з особливим фокусом на світлове забруднення, адже в потоці дорожнього руху кожен транспортний засіб може мати своє джерело світла, спотворюючи сприйняття кольору транспортних засобів навколо. Обробка виконується алгоритмічно, відповідно до методу описаному в статті “Single image haze removal using dark channel prior” [52]. Після обробки, сам колір зчитується з фрагменту кадру, що містить транспортний засіб, за допомогою згорткової нейронної мережі відповідно до методу, описаному в статті “Vehicle Color Recognition using Convolutional Neural Network” [53].

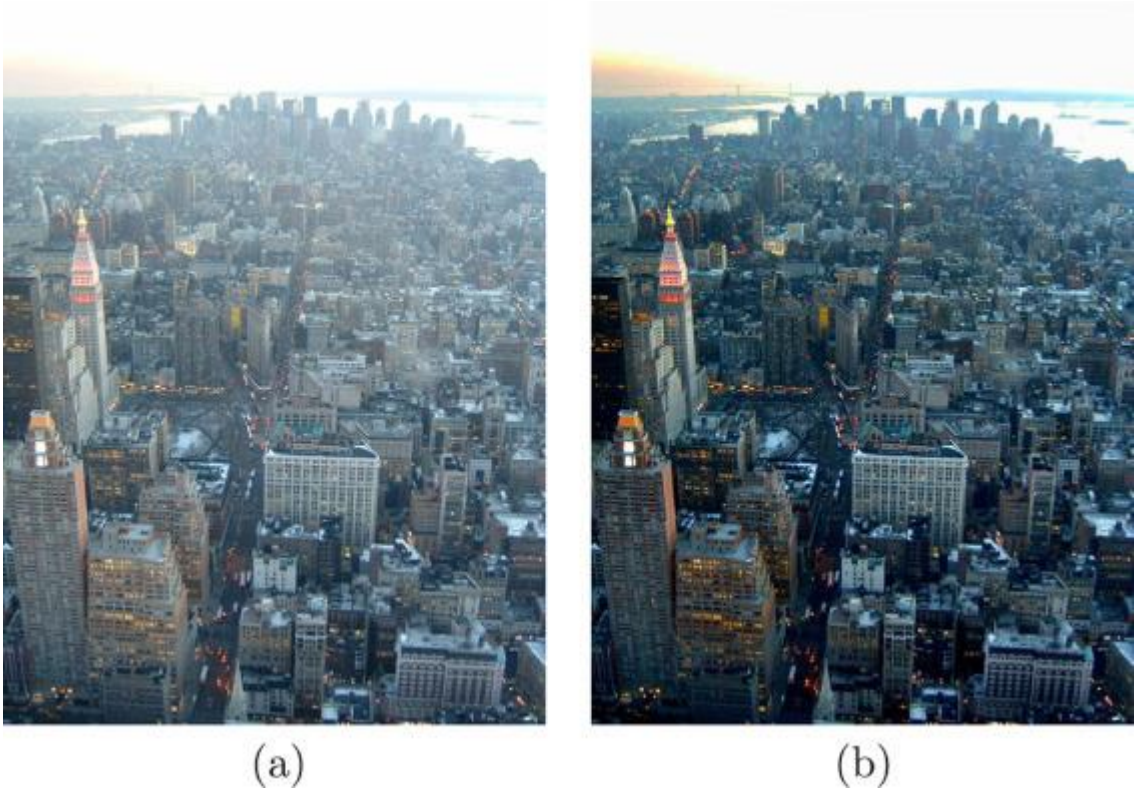


Рис. 3.4. Приклад зображення до і після нейтралізації туману [54]

Отримання моделі та марки за зображенням транспортного засобу є не менш складною задачею, ніж отримання кольору. Станом на 2024 рік, у світі існують тисячі комбінацій марок та моделей. Окрім цього, транспортні засоби однією марки але різних моделей не рідко є схожими між собою, що так само ускладнює задачу класифікації. В даній дипломній роботі, для класифікації моделей та марок транспортних засобів було використано глибоку нейронну мережу з архітектурою ResNet-50 [55], що було натреновано на наборі даних ImageNet [56] та дотреновано на наборі даних “A Large and Diverse Dataset for Improved Vehicle Make and Model Recognition” [57].

Компонент зберігання даних

Модуль відбору інформації є відповідальним за збергіання всіх даних, що оброблює система, включаючи ті дані, що отримуються модулем оброблення потоків. Таке рішення було прийнято з урахуванням того, що

модуль оброблення потоків не виконує жодного аналізу даних. Відповідно, зберігання даних цим модулем призвело б лише до непотрібних затримок під час оброблення кадрів.

Компонент зберігання даних працює з даними чотирьох сутностей – камери (Camera), ТЗ (Vehicle), кадру (Frame) та регіону кадру (Region). Схему атрибутів та зв'язків цих сутностей наведено на рисунку 3.5.

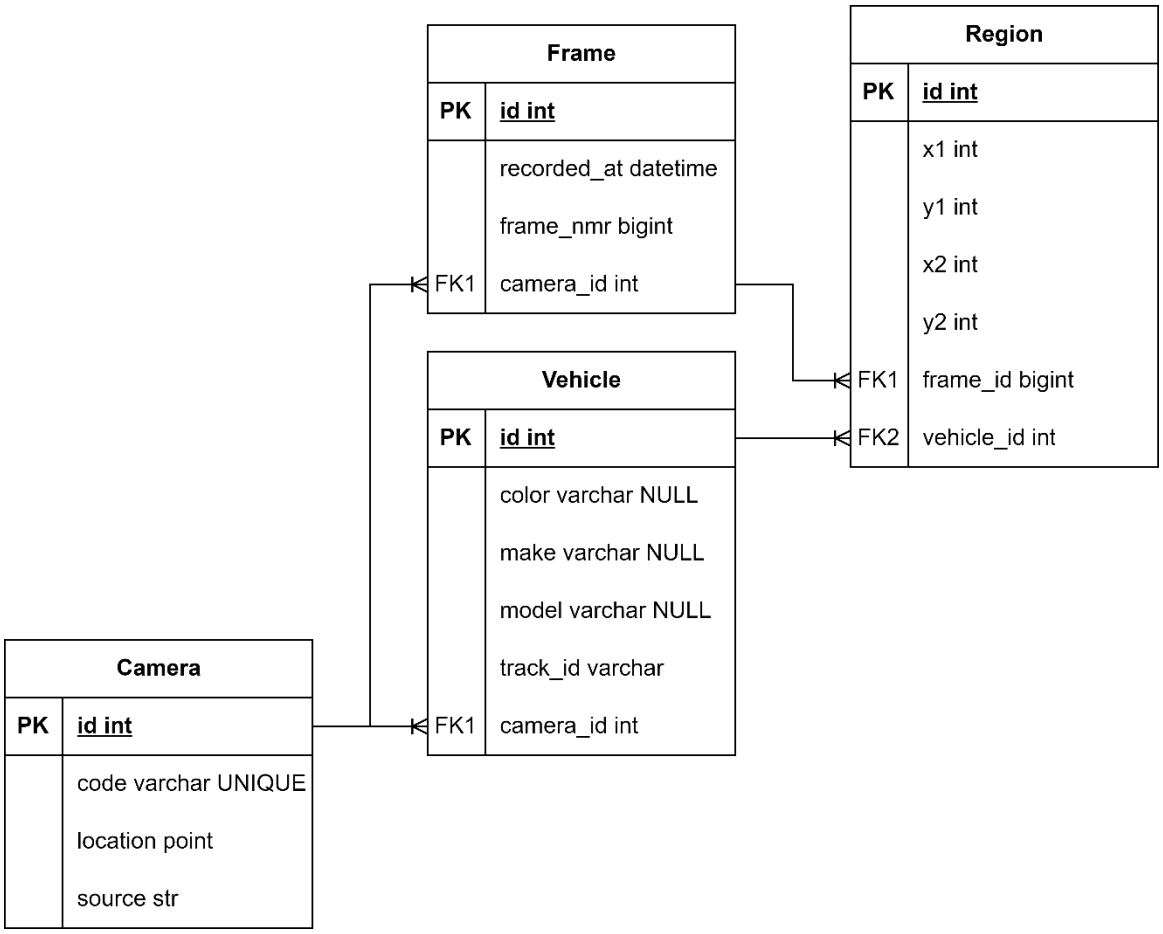


Рис. 3.5. Схема сутностей камери, транспортного засобу, кадру та регіону кадру

Компонент аналізу історичних даних

Компонент аналізу історичних даних збирає інформацію о кількості унікальних ТЗ, а також їхньої середньої швидкості за певні проміжки часу – від однієї години до 30 днів, базуючись на даних, що додаються до системи компонентом зберігання даних. Для цього, компонент аналізу періодично

дістає та агрегує дані з БД, будуючи проєкції даних, які потім так само зберігаються в БД.

Компонент зберігання даних працює з даними трьох сутностей – проєкція (Projection), що містить дані спільні для проєкцій всіх типів, проєкція кількості ТЗ (CountProjection). та проєкція середньої швидкості ТЗ (VelocityProjection). Схему атрибутів та зв'язків цих сутностей наведено на рисунку 3.6.

Компонент аналізу даних в режимі реального часу

Компонент аналізу даних в режимі реального часу працює схожим чином до компоненту аналізу історичних даних. Проте, на відміну від попереднього, даний компонент отримує вхідні дані з розподіленого кешу, куди їх публікує модуль обробки відео потоків. Окрім цього, створені даним компонентом проєкції зберігаються в кеші, а також мають дуже коротке часове вікно – від 1 до 10 секунд.

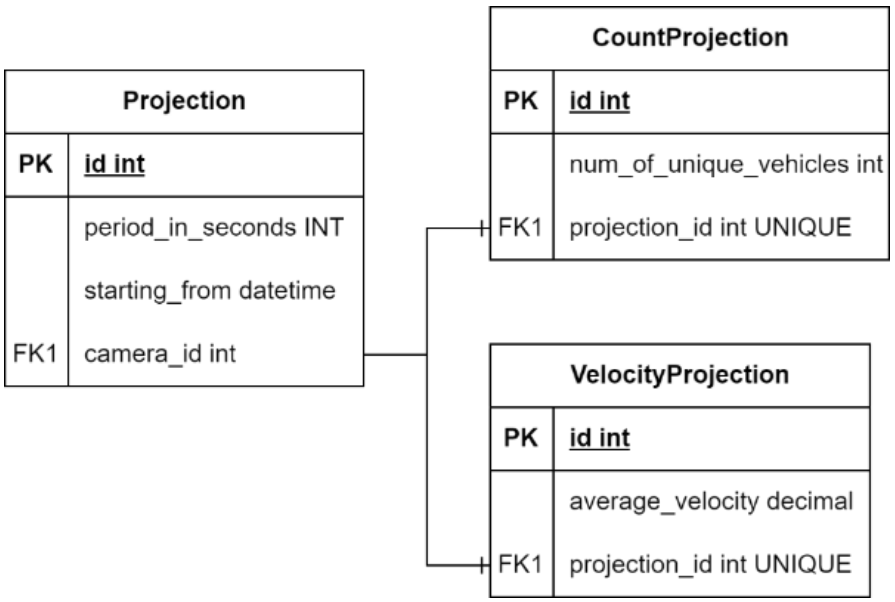


Рис. 3.6. Схема сутностей проєкції, проєкції швидкості та проєкції кількості ТЗ

Компонент оброблення запитів зовнішніх систем

Компонент оброблення запитів є інтерфейсом розробленого програмного забезпечення для комунікації з зовнішнім світом. Завдяки цьому інтерфейсу, користувачі системи можуть отримувати зібрану системою інформацію, реєструвати нові джерела даних (камери відео спостереження) до системи, в майбутньому, підписуватись на події, що помічаються системою. Прикладом такої події є різке зниження середньої швидкості транспортних засобів, помічених певною камерою, що може означати загорання червоного сигналу світлофору в локації поблизу камери.

Для того, щоб система оброблювала запити користувача, користувач має бути зареєстрованим в системі, автентифікованим та авторизованим системою.

Реєстрація користувачів виконується користувачами з роллю адміністратора. При цьому, найперший адміністратор системи автоматично створюється при першому запуску модулю відбору інформації. При реєстрації, адміністратор має вказати електронну пошту та пароль користувача. Варто зазначити, що електронна пошта має бути унікальною, а пароль – підходити під вимоги безпеки системи: його довжина має потрапляти в певний діапазон, а складові символи мають містити хоча б один спеціальний символ. Якщо електронна пошта та пароль задовольняють вимогам, вони зберігаються в системі. Зазначимо, що пароль зберігаються в хешованому вигляді.

Для автентифікації, користувач має ввести свої електронну пошту та пароль на сторінці автентифікації системи. Після потрапляння на сервер, значення пароля хешується таким же чином як при реєстрації, після чого система шукає користувача з переданими значеннями електронної пошти та пароля в базі даних. Якщо пошук є успішним, система генерує токен автентифікації, повертає його користувачу та зберігає в системі для подальшої ідентифікації користувача. Якщо ні – токен не генерується, а

користувачу повертається повідомлення о помилці. Зазначимо, що не автентифікованим користувачам система не дає виконувати будь-які дії окрім автентифікації.

Деякі дії в системі, такі як реєстрація нової камери відео спостереження, додавання нового користувача або зміна повноважень існуючого користувача вимагають від користувачів мати роль адміністратора системи. За відсутності такої ролі у користувача, система не дозволить йому виконувати наведені вище дії.

Автентифіковані користувачі можуть відправляти системі HTTP запити на отримання історичних даних. Дані для відповіді на ці запити беруться з проєкцій, що генеруються компонентом аналізу історичних даних. Також, користувачі можуть підписуватись на отримання даних з цікавлячих їх камер відео спостереження в режимі реального часу. Системою такі дані беруться з кешу, куди їх зберігає компонент аналізу даних в режимі реального часу, та передаються користувачеві за допомогою технології WebSockets.

Варіанти використання ПЗ користувачами з різними ролями наведено на рисунку 3.7.

3.2.3. *Модуль візуалізації даних*

Модуль візуалізації даних представляє собою клієнтську частину розробленого програмного забезпечення, метою якої є спрощення роботи користувачів ПЗ з API системи. Даний модуль представляє собою SPA застосунок, написаний за допомогою фреймворку VueJS.

Деякі дані, з якими взаємодіє система, є складними для сприйняття людьми. Особливо це стосується значень локацій камер відео спостереження, що зберігаються в системі як значення координат в координатній системі Землі [58]. Маючи лише значення широти і довготи конкретної точки, більшість фахівців не зможуть вказати навіть на країну, в

якій знаходитиметься така точка, не кажучи вже про локації меншого розміру, такі як місто чи перехрестя вулиць. Через це, в розробленому продукті, камери в певній місцевості відображаються як крапки на інтерактивній мапі цієї місцевості. Саму інтерактивну мапу було розроблено за використання бібліотеки Mapbox GL JS [59].

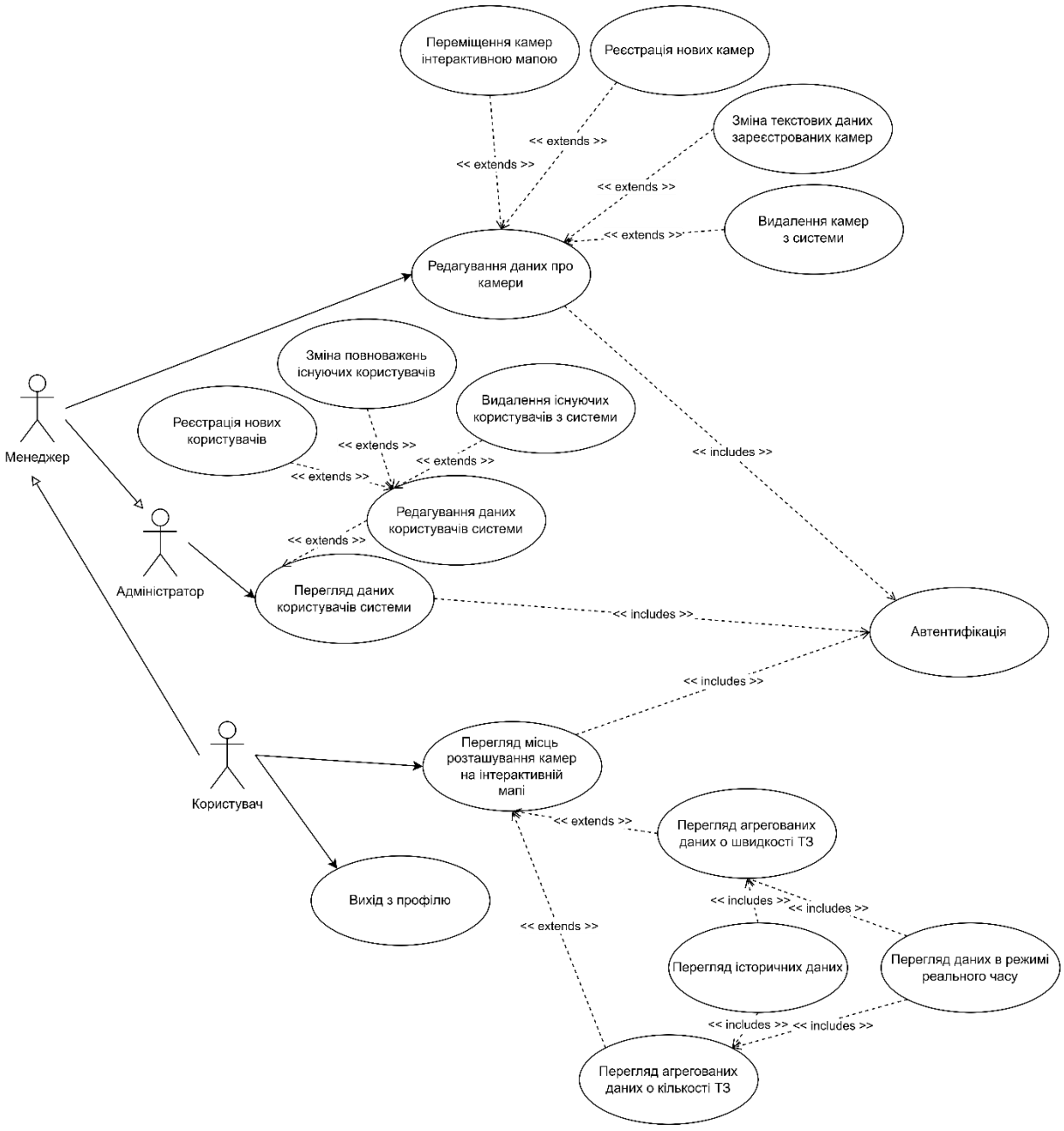


Рис. 3.7. Варіанти використання ПЗ

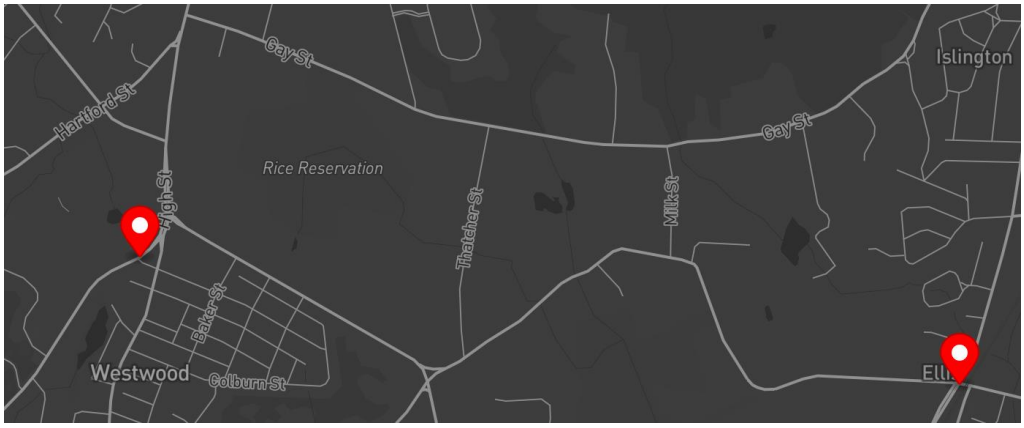


Рис. 3.8. Приклад відображення камер на мапі

Дані, отримані в результаті аналізу потоків з камер відео спостереження подаються модулем візуалізації у вигляді інтерактивних графіків (рис. 3.8), що було реалізовано за допомогою бібліотеки ApexCharts [60].

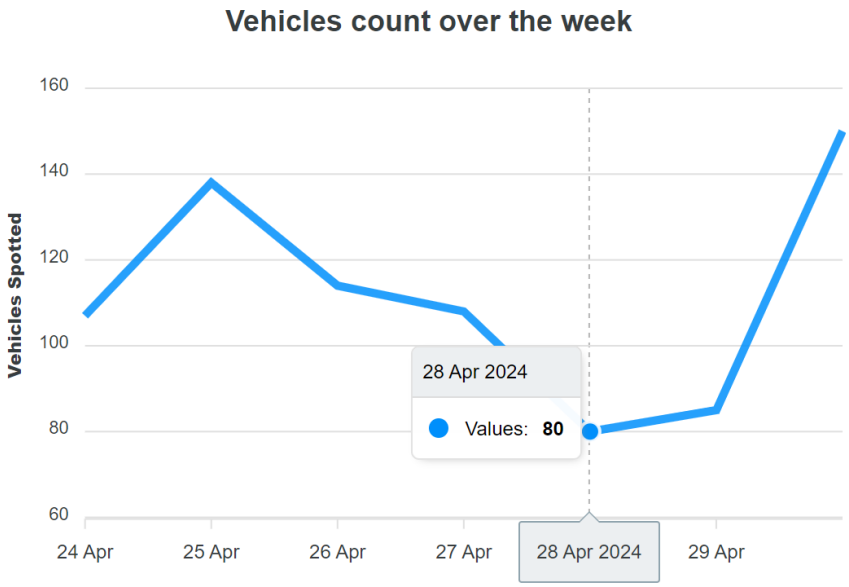


Рис. 3.9. Приклад візуалізації історичних даних о кількості ТЗ

3.3. Висновки до розділу 3

У розділі 3 було описано архітектуру, основні модулі системи, та їхні компоненти. Було згадано технології та особливості реалізації певних

алгоритмів системи. Також, було наведено ERD-діаграму БД одного з модулів застосунку, описано сутності та зв’язки між ними.

4. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Тестування програмного забезпечення

Тестування є неодмінним етапом життєвого циклу розробки програмного продукту [61]. Саме на даному етапі перевіряється коректність роботи, а також відповідність програмного продукту зібраним для нього функціональним та нефункціональним вимогам.

Існує багато типів тестування ПЗ [62]. В залежності від свого призначення, кожен з них проводиться на різних стадіях розробки програмного продукту. Наприклад, однією з найкращих практик при розробці ПЗ вважається написання юніт- та інтеграційних тестів паралельно з написанням коду програмного продукту. В той же час, проведення димового тестування вимагає наявності готового і розгорнутого на пристрої-носії програмного забезпечення. Також, тестування ПЗ нерідко класифікується за типом його проведення – на ручне та автоматизоване. При ручному тестуванні, тестові сценарії виконуються безпосередньо розробниками або тестувальниками ПЗ. При автоматизованому, набори тестових сценаріїв запускаються, а результати виконання сценаріїв перевіряються за допомогою спеціалізованих інструментів [63].

Тестування програмного забезпечення, розробленого в рамках даного дипломного проєкту, було проведено наступними методами.

4.1.1. *Димове тестування*

Димове тестування або тестування впевненості є типом тестування ПЗ, що перевіряє готовність нової збірки ПЗ для подальшого тестування або навіть розгортання. Під час виконання сценаріїв даного тестування перевіряється коректність роботи найважливіших елементів функціональності програмного забезпечення. Не проходження ПЗ даного типу тестування зазвичай говорить о наявності критичних помилок в коді.

Подальше тестування або розгортання такого ПЗ має мало сенсу і, як правило, воно повертається команді розробників на доопрацювання.

Димове тестування розробленого ПЗ було виконано вручну. Для вебзастосунку, що було створено як частину ПЗ, було перевірено наступну функціональність:

- авторизація у системі (рис. 4.1);
- виведення маркерів камер відеоспостереження на фоні мапі міста (рис. 4.2);
- можливість зміни локацій маркерів (рис. 4.3);
- можливість зміни деталей маркерів (рис. 4.4);
- виведення даних о кількості ТЗ в режимі реального часу на графіку (рис. 4.5);
- виведення історичних даних о кількості ТЗ на графіку (рис. 4.6, 4.7);
- реєстрація нових камер до системи (рис. 4.8);
- реєстрація нових користувачів у системі (рис. 4.9);
- редагування даних існуючих користувачів (рис. 4.10).

Також, для модуля обробки відео потоків, що не має клієнтської частини, було перевірено наступну функціональність:

- публікація повідомлень до брокера повідомлень (рис. 4.11);
- публікація даних до розподіленої системи кешування (рис. 4.12).

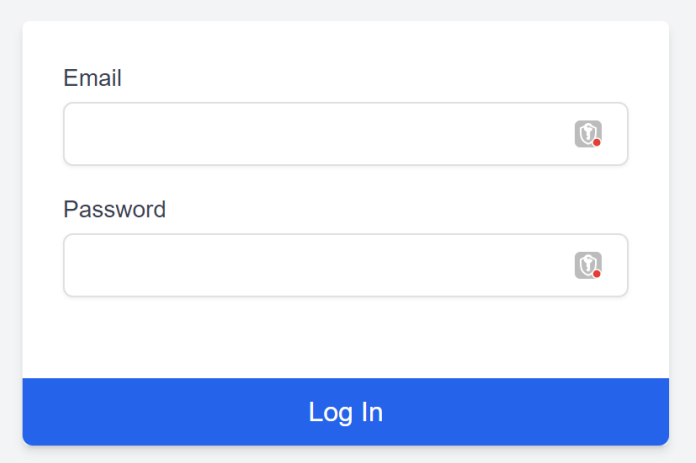


Рис. 4.1. Сторінка авторизації

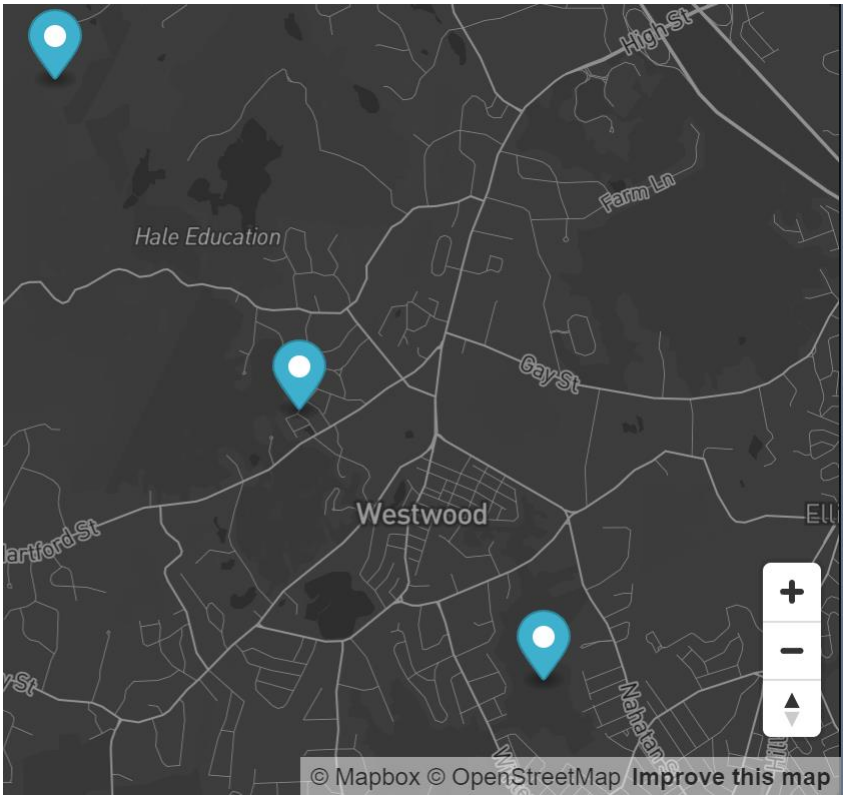


Рис. 4.2. Головна сторінка, фрагмент мапи з розташованими на ній маркерами

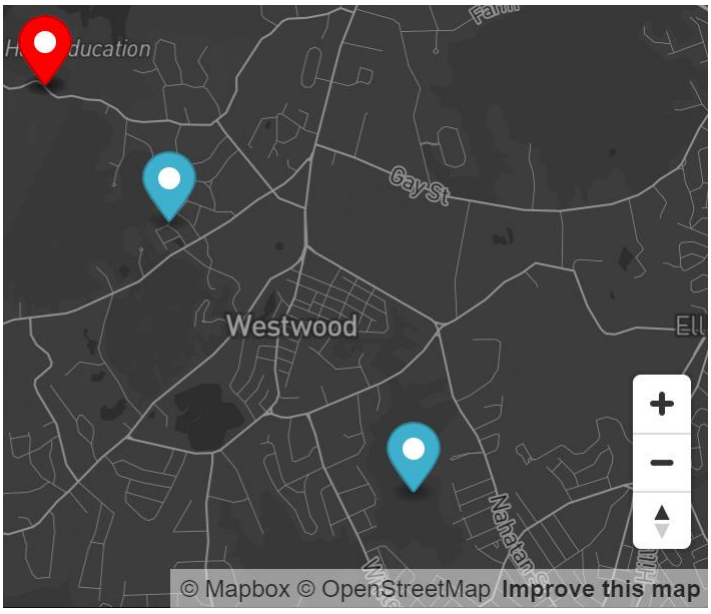


Рис. 4.3. Головна сторінка, фрагмент мапи з розташованими на ній маркерами. Червоним кольором відмічено пересунутий маркер, що досі не було збережено

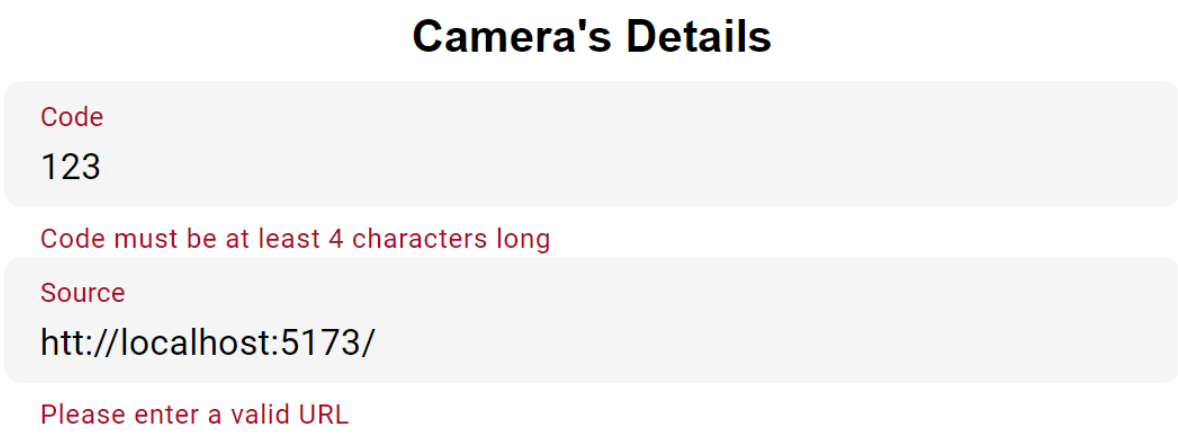


Рис. 4.4. Головна сторінка, форма редагування даних камери відео спостереження

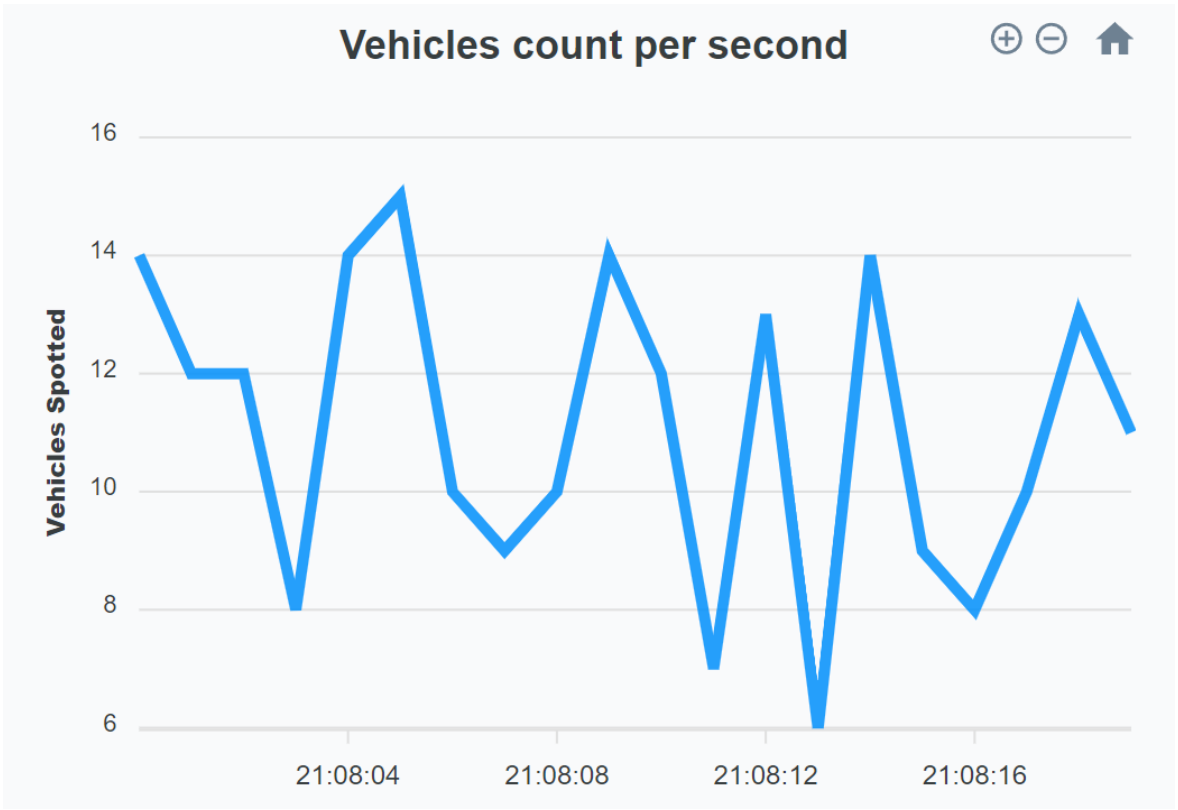


Рис. 4.5. Головна сторінка, фрагмент графіку даних кількості ТЗ, що оновлюється в режимі реального часу

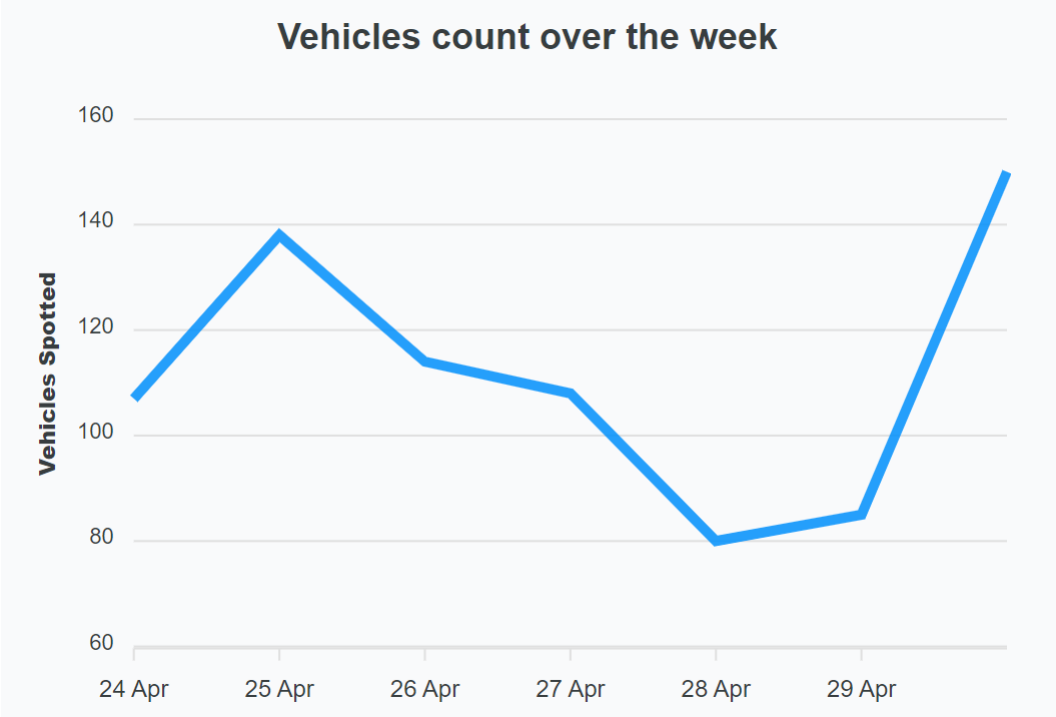


Рис. 4.6. Головна сторінка, фрагмент графіку кількості ТЗ, помічених кожен день протягом 1 тижня

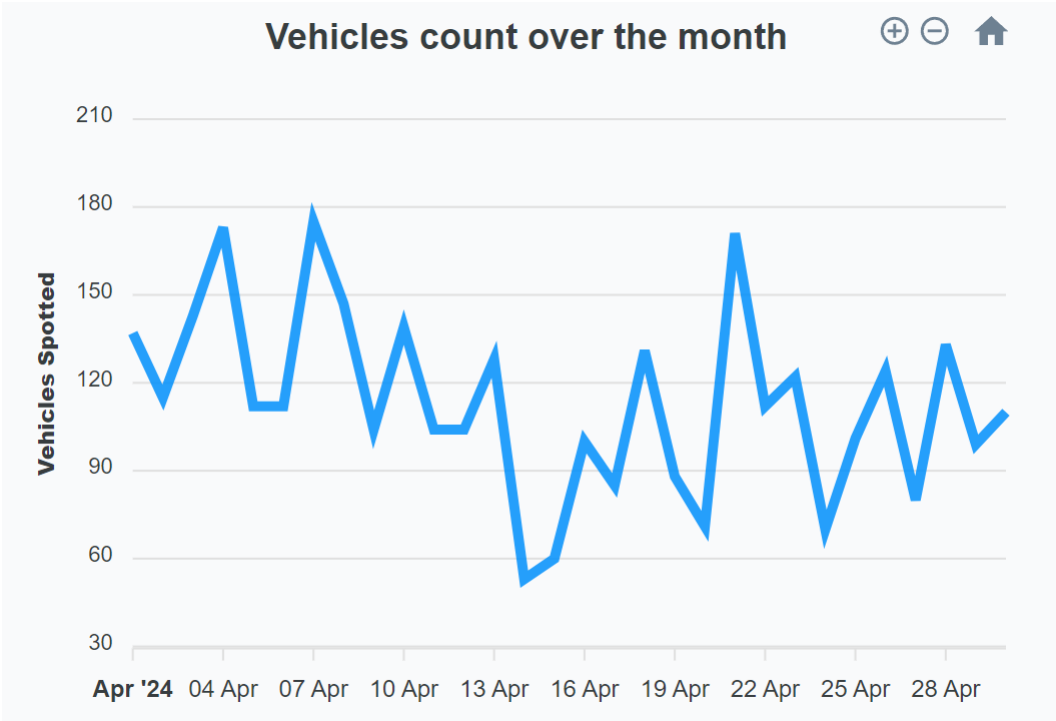


Рис. 4.7. Головна сторінка, фрагмент графіку кількості ТЗ, помічених кожен день протягом 1 місяця

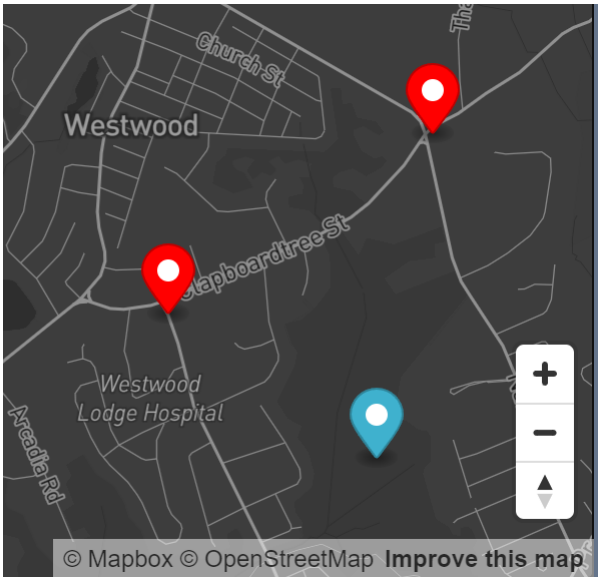


Рис. 4.8. Головна сторінка, фрагмент мапи з розташованими на ній маркерами. Червоним кольором відмічено додані маркери, що досі не було збережено

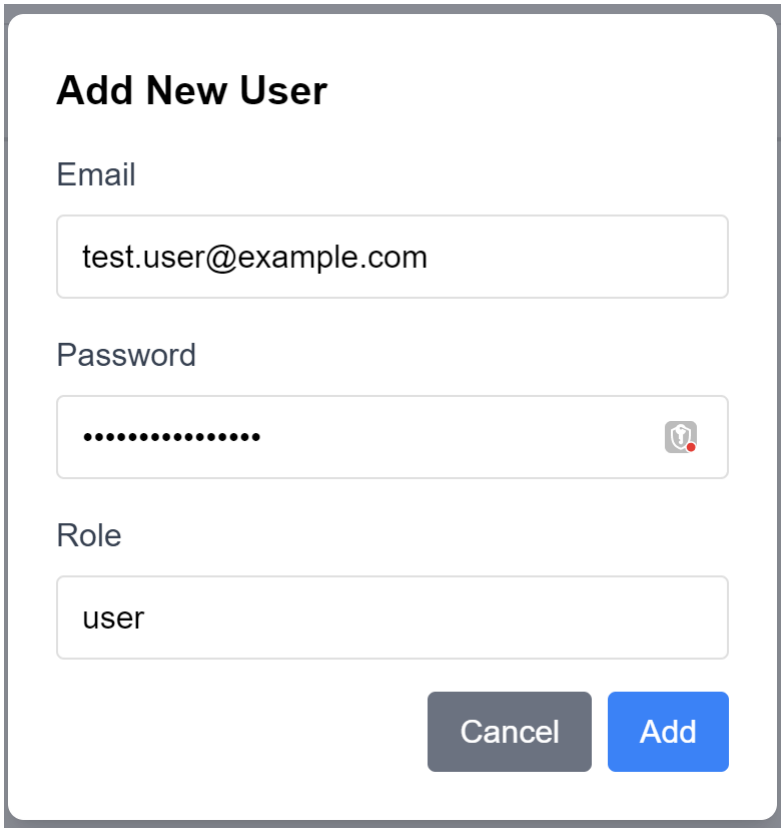


Рис. 4.9. Сторінка користувачів, форма реєстрації нового користувача

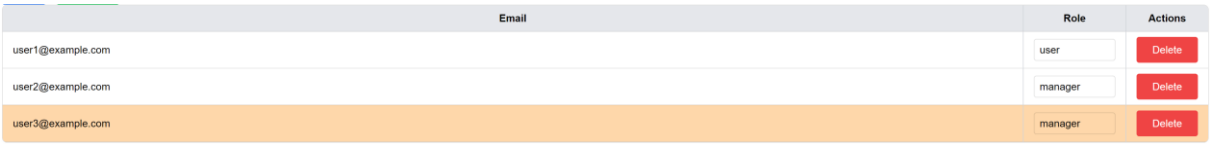


Рис. 4.10. Сторінка користувачів, редагування даних користувача (оранжевим підсвічено запис користувача, дані якого було змінено)

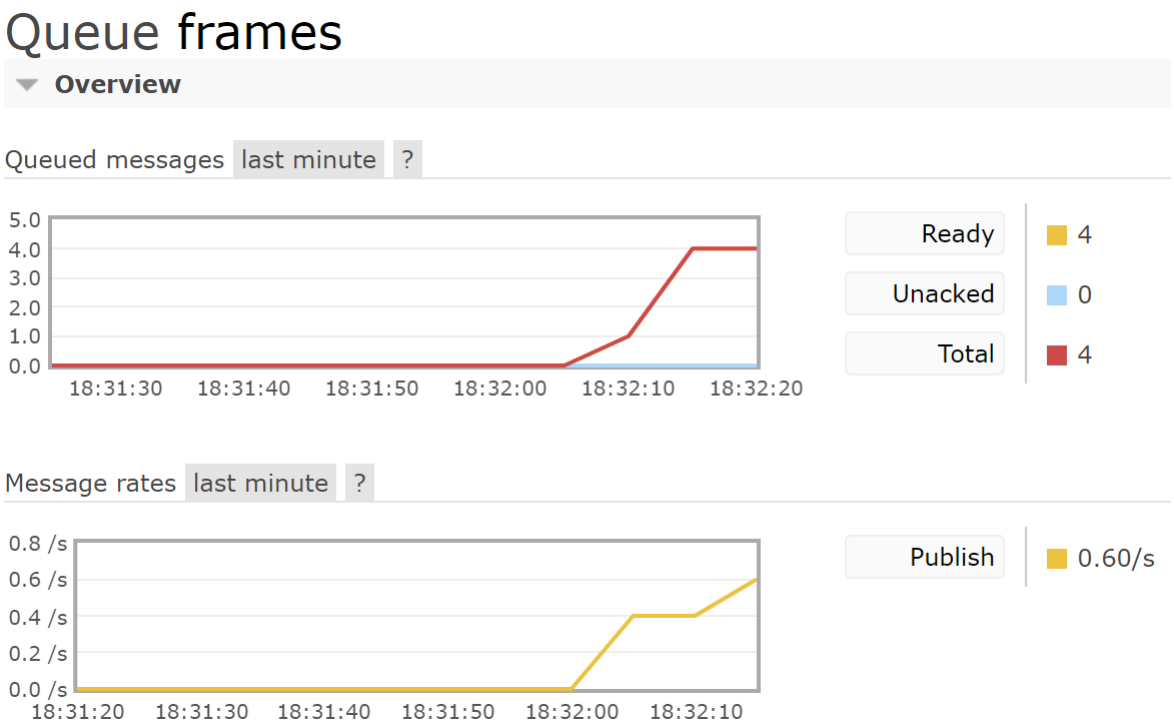


Рис. 4.11. Сторінка консолі адміністратора RabbitMQ, метрики черги повідомлень

```
redis-cli
127.0.0.1:6379> INFO keyspace
# Keyspace
db0:keys=1,expires=0,avg_ttl=0
db1:keys=306,expires=0,avg_ttl=0
```

Рис. 4.12. Консольний інтерфейс Redis, кількість записів у базах даних, що використовуються ПЗ

4.1.2. *Тестування швидкодії*

Модуль оброблення відео потоків призначено для оброблення потоків у режимі реального часу. Відповідно, кожен кадр потоку має бути оброблено до отримання наступного кадру.

Для отримання інформації о швидкодії модуля оброблення потоків, було зібрано значення часів оброблення 1200 різних кадрів, знайдено мінімальний, максимальний, та середній часи. Дані для аналізу було зібрано як за умови виконання програмних алгоритмів на ядрах центрального процесору Intel Core I5-12450H, так і на CUDA ядрах графічного процесору RTX 3050 6GB. Результати аналізу наведено в таблиці 4.1.

Таблиця 4.1

Результати аналізу швидкодії

Пристрій	Мінімальний час	Середній час	Максимальний час
CUDA	22.5мс	27.8мс	39.1мс
CPU	201.3мс	225.6мс	301.3мс

При виконанні алгоритмів на CUDA ядрах графічного процесору, середній час обробки становить 27.8мс. Це значення є суттєво меншим за поріг у 50мс, що має досягатись для оброблення відео потоків з частотою 20 кадрів на секунду в режимі реального часу. Більше того, за умови покращення швидкодії модуля оброблення потоків та досягнення зменшення середнього часу обробки кадрів графічним процесором RTX 3050 до 20-25мс, даний модуль зможе працювати на пристроях типу NVIDIA Jetson Orin Nano [64], що, відповідно до принципів архітектури туманних обчислень [65], суттєво зменшить навантаження на центральні сервери системи.

Очікувано, інтелектуальне оброблення кадрів відео потоків ядрами центрального процесору займає значно більше часу за оброблення ядрами

графічного процесору. Оптимізація алгоритмів розробленого ПЗ для зменшення часу обробки ядрами центрального процесору не має практичного сенсу.

4.2. Порівняння з існуючими рішеннями

На початку даної пояснювальної записки було проведено аналіз існуючих програмних рішень для моніторингу дорожнього руху з метою отримання корисної інформації для проведення рекламних кампаній. На основі цього аналізу було сформовано вимоги до розроблюваного програмного забезпечення.

Маючи отримані в першому розділі результати аналізу, а також готове програмне забезпечення, можна навести його переваги відносно до існуючих рішень. Такими перевагами є:

- сумісність з відео потоками різних форматів;
- наявність платформи для візуалізації даних;
- продуктивність;
- можливість горизонтального масштабування;
- підтримка моделі туманних обчислень;
- можливість оброблення даних в режимі реального часу.

4.3. Рекомендації щодо подальшого вдосконалення

Наведемо напрямки можливого розвитку розробленого ПЗ:

- оптимізація використаних моделей машинного навчання;
- збільшення кількості атрибутів даних, що збирає система;
- інтеграція можливості аналізу пішохідних потоків;
- адаптація вебінтерфейсу під мобільні пристрої;
- інтеграція сторонніх джерел даних, таких як оператори мобільного зв’язку, до системи;
- використання хмарних технологій для розгортання системи.

4.4. Висновки до розділу 4

Четвертий розділ пояснювальної записки було присвячено аналізу розробленого програмного забезпечення.

В даному розділі було розглянуто різні методики тестування ПЗ. Також, при написанні розділу було проведено димове тестування та тестування швидкодії розробленого ПЗ, наведено аналіз результатів тестування. Також, було проведено порівняння розробленого ПЗ з існуючими програмними рішеннями. Було наведено переваги розробленого програмного продукту, а також рекомендації щодо його подальшого вдосконалення.

ВИСНОВКИ

Метою даного дипломного проєкту було створення програмного забезпечення для інтелектуального аналізу даних дорожнього руху з використанням камер відео спостереження, що збирає та аналізує дані для оптимізації кампаній зовнішньої реклами.

Перший розділ було присвячено огляду та аналізу існуючих програмних рішень проблеми збору та аналізу даних для кампаній зовнішньої реклами, виявлено їхні сильні та слабкі сторони. Базуючись на проведеному аналізі та маючи за мету усунення слабких та вдосконалення сильних сторін оглянутих рішень, було сформовано функціональні та нефункціональні вимоги для розроблюваного програмного продукту.

У другому розділі фокус було поставлено на огляд інструментів розроблення систем для інтелектуального аналізу відео потоків та вибір оптимальних технологій для розроблюваного програмного продукту. Як результат, для розроблення серверної частини було обрано мову Python, фреймворк для веброзробки FastAPI, та даних PostgreSQL; для розроблення клієнтської частини – фреймворк VueJS та мову JavaScript з її розширенням TypeScript; як посередників для асинхронної комунікації між серверними компонентами – брокера повідомлень RabbitMQ та систему розподіленого кешування Redis.

Третій розділ пояснювальної записки було присвячено опису архітектури розробленого програмного забезпечення. В цьому розділі було розглянуто основні модулі ПЗ та їхні важливі компоненти, структуру та взаємозв’язок сутностей бази даних, описано особливості реалізації певних алгоритмів та технології, завдяки яким ці алгоритми було створено.

У четвертому розділі було проведено аналіз розробленого програмного забезпечення. На початку розділу, було наведено та інтерпретовано результати тестування програмної системи. Також, новий

програмний продукт було порівняно з існуючими програмними рішеннями, наведено рекомендації для його вдосконалення.

Як результат роботи над дипломним проєктом, було розроблено багатомодульну програмну систему для збору структурованих даних з відео потоків, їхнього аналізу, та візуалізації отриманої інформації. Для збору структурованих даних використовуються найсучасніші моделі комп'ютерного зору, а самі дані проходять попередню обробку за допомогою різноманітних алгоритмів для поліпшення їхнього сприйняття моделями. Також, зважаючи на високе навантаження на обчислювальні системи, що неодмінно виникає як при обробці потокових даних, так і при використання методів машинного навчання, програмну систему було побудовано таким чином, що кількість її модулів можна було горизонтально масштабувати як за допомогою хмарних технологій, так і за використання набору пристроїв для хмарних обчислень.

Зважаючи на актуальність проблеми, яку розроблене програмне забезпечення вирішує, а також наявність опису його різних аспектів, подальші підтримка та вдосконалення ПЗ є цілком здійсненими та рекомендованими.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Big data market size revenue forecast worldwide from 2011 to 2027 [Електронний ресурс]. – Режим доступу: https://www.statista.com/statistics/254266/global-big-data-market-forecast/ – (10.05.2023).
2. Internet advertising spending worldwide from 2007 to 2024, by format [Електронний ресурс]. – Режим доступу: https://www.statista.com/statistics/276671/global-internet-advertising-expenditure-by-type/ – (10.05.2023).
3. Will Google Ever Run Out of Storage Space? [Електронний ресурс]. – Режим доступу: https://www.makeuseof.com/will-google-run-out-storage-space/ – (10.05.2023).
4. Evolution of IoT Technology Pricing [Електронний ресурс]. – Режим доступу: https://www.link-labs.com/blog/evolution-of-iot-technology-pricing/ – (10.05.2023).
5. The Definitive Guide to Location Intelligence for Out-of-Home Advertising [Електронний ресурс]. – Режим доступу: https://gravyanalytics.com/definitive-guide-location-intelligence-out-of-home-advertising/ – (10.05.2023).
6. What is Dynamic Digital OOH and Why You Should Use It? [Електронний ресурс]. – Режим доступу: https://theneuron.com/what-is-dynamic-digital-ooh/ – (10.05.2023).
7. DIGITAL DIRECT ADVERTISING SERVER [Електронний ресурс]. – Режим доступу: https://www.outfront.com/ad-tech/digital-direct-ad-server/ – (10.05.2023).
8. SMARTSCOUT™ [Електронний ресурс]. – Режим доступу: https://www.outfront.com/ad-tech/smartscout/ – (10.05.2023).

9. Data-driven OOH [Електронний ресурс]. – Режим доступу: https://www.jcdecaux.com/brands/data-driven-ooh/ – (10.05.2023).
10. Dynamic content [Електронний ресурс]. – Режим доступу: https://www.jcdecaux.com/brands/dynamic-content/ – (11.05.2023).
11. Get more results with Clear Channel Outdoor RADAR [Електронний ресурс]. – Режим доступу: https://clearchanneloutdoor.com/radar-data-solutions/ – (11.05.2023).
12. 5 Reasons Why OOH Professionals Are Turning to Location Intelligence [Електронний ресурс]. – Режим доступу: https://carto.com/blog/why-ooh-advertisers-use-location-intelligence/ – (11.05.2023).
13. Constructing An Out-Of-Home Rating [Електронний ресурс]. – Режим доступу: https://support.geopath.io/hc/en-us/articles/360006951491-Constructing-An-Out-Of-Home-Rating/ – (11.05.2023).
14. Python (programming language) [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Python_(programming_language) – (13.05.2023).
15. Python Documentation by Version [Електронний ресурс]. – Режим доступу: https://www.python.org/doc/versions/ – (13.05.2023).
16. Most used programming languages among developers worldwide as of 2023 [Електронний ресурс]. – Режим доступу: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/ – (13.05.2023).
17. Purposes for Using Python [Електронний ресурс]. – Режим доступу: https://lp.jetbrains.com/python-developers-survey-2022/#PurposesUsingPython/ – (13.05.2023).
18. EDA | Exploratory Data Analysis in Python [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/exploratory-data-analysis-in-python/ – (13.05.2023).

19. How Fast Does Python Execute Code? [Електронний ресурс]. – Режим доступу: https://www.rebellionresearch.com/how-fast-does-python-execute-code/ – (13.05.2023).
20. Extending Python with C or C++ [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/extending/extending.html/ – (13.05.2023).
21. A dynamic admin interface: it’s not just scaffolding – it’s the whole house [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/en/5.0/intro/overview/#a-dynamic-admin-interface-it-s-not-just-scaffolding-it-s-the-whole-house/ – (13.05.2023).
22. FastAPI [Електронний ресурс]. – Режим доступу: https://blog.jetbrains.com/pycharm/2023/12/django-vs-fastapi-which-is-the-best-python-web-framework/ – (13.05.2023).
23. Can I Use Python for Front-end Development? [Електронний ресурс]. – Режим доступу: https://medium.com/@etherservices.vimalraj/can-i-use-python-for-front-end-development-e80033f9c175/ – (13.05.2023).
24. Python: Unleashing the Power of Simplicity and Versatility [Електронний ресурс]. – Режим доступу: https://medium.com/@shaikhhashim070/python-unleashing-the-power-of-simplicity-and-versatility-243f29b6914/ – (13.05.2023).
25. all-packages 1.0.1 [Електронний ресурс]. – Режим доступу: https://pypi.org/project/all-packages/ – (13.05.2023).
26. What Is the Python Global Interpreter Lock (GIL)? [Електронний ресурс]. – Режим доступу: https://realpython.com/python-gil/ – (13.05.2023).
27. JavaScript [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/JavaScript/ – (13.05.2023).

28. MOST POPULAR WEB BROWSERS IN 2024 [Електронний ресурс]. – Режим доступу: https://www.oberlo.com/statistics/browser-market-share/ – (13.05.2023).
29. An introduction to the npm package manager [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/learn/getting-started/an-introduction-to-the-npm-package-manager/ – (13.05.2023).
30. PostgreSQL [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/PostgreSQL/ – (13.05.2023).
31. Chapter 8. Data Types [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/current/datatype.html/ – (13.05.2023).
32. Chapter 9. Multi-Version Concurrency Control [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/7.1/mvcc.html#MVCC-INTRO/ – (13.05.2023).
33. Microsoft SQL Server [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Microsoft_SQL_Server/ – (13.05.2023).
34. Optimistic locking [Електронний ресурс]. – Режим доступу: https://www.ibm.com/docs/en/db2/11.5?topic=overview-optimistic-locking/ – (13.05.2023).
35. Why is Redis Incredibly Fast? Unpacking the Secrets of its Speed [Електронний ресурс]. – Режим доступу: https://medium.com/hprog99/why-is-redis-incredibly-fast-unpacking-the-secrets-of-its-speed-f10f051b3f23/ – (13.05.2023).
36. Redis persistence [Електронний ресурс]. – Режим доступу: https://redis.io/docs/latest/operate/oss_and_stack/management/persistence/ – (13.05.2023).
37. Asynchronous communication of microservices [Електронний ресурс]. – Режим доступу:

https://journals.nmetau.edu.ua/index.php/st/article/view/1227/ – (14.05.2023).

38. Message Ordering in RabbitMQ [Електронний ресурс]. – Режим доступу: https://www.rabbitmq.com/docs/queues#message-ordering/ – (14.05.2023).
39. Benchmarking Apache Kafka: 2 Million Writes Per Second (On Three Cheap Machines) [Електронний ресурс]. – Режим доступу: https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines/ – (14.05.2023).
40. Different types of scaling in Apache Kafka [Електронний ресурс]. – Режим доступу: https://www.cloudkarafka.com/blog/scaling-in-apache-kafka.html/ – (14.05.2023).
41. Comparative Web Performance evaluation of Vue & React using JSWFB [Електронний ресурс]. – Режим доступу: https://medium.com/@danialeshete/comparative-web-performance-evaluation-benchmarking-of-vue-react-using-jswfb-a76982097225/ – (14.05.2023).
42. fastapi [Електронний ресурс]. – Режим доступу: https://github.com/tiangolo/fastapi/ – (14.05.2023).
43. Benefits of FastAPI [Електронний ресурс]. – Режим доступу: https://blog.jetbrains.com/pycharm/2023/12/django-vs-fastapi-which-is-the-best-python-web-framework/#benefits-of-fastapi/ – (14.05.2023).
44. Client-server model [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Client%E2%80%93server_model#Examples – (14.05.2023).
45. A computer network diagram of clients communicating with a server via the Internet [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Client%E2%80%93server_model#/media/File:Client-server-model.svg – (14.05.2023).

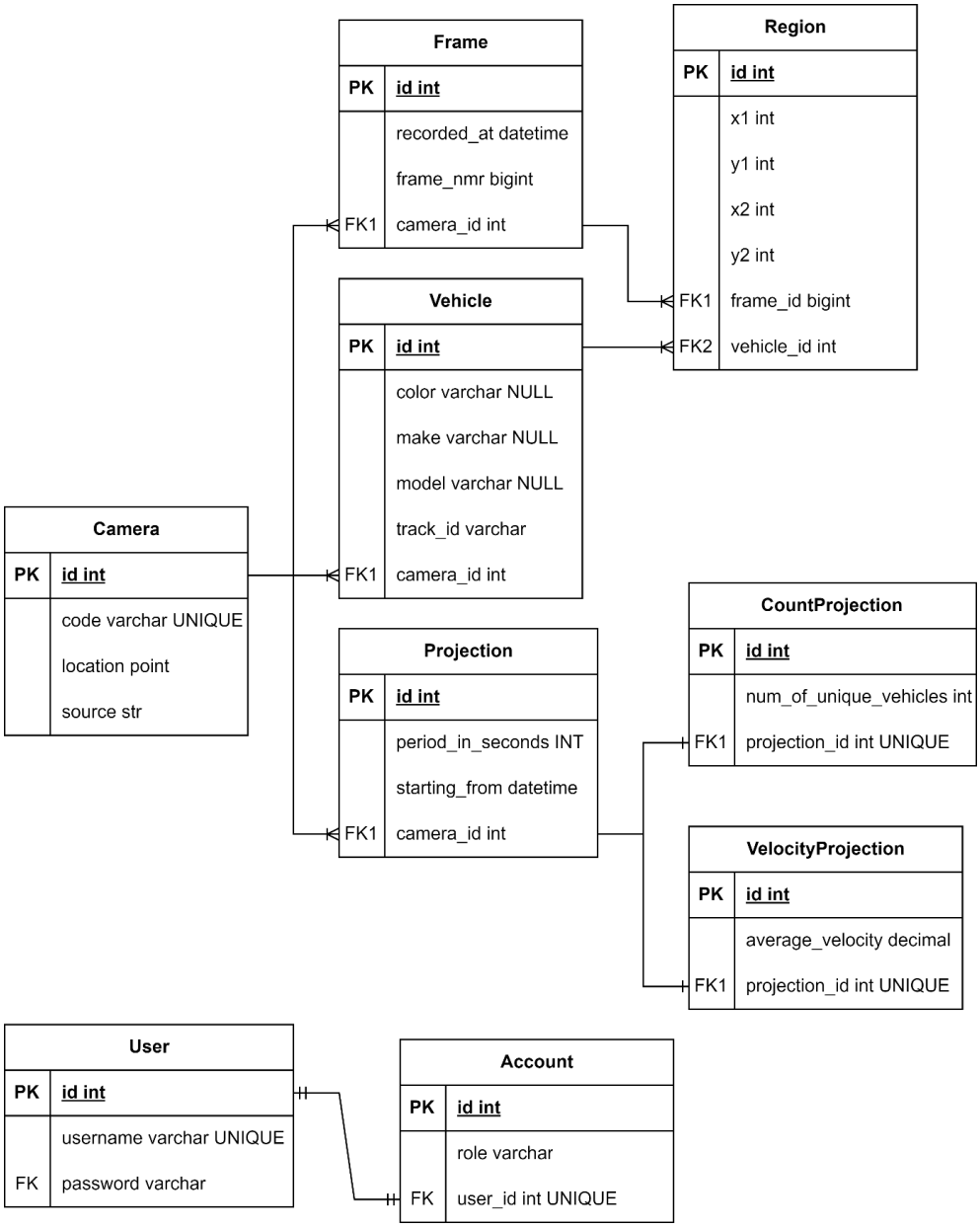
46. Distributed Computing [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Distributed_computing – (14.05.2023).
47. Peer-to-peer [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Peer-to-peer#/media/File:P2P_network.svg – (14.05.2023).
48. Reference for ultralytics/data/base.py [Електронний ресурс]. – Режим доступу: https://docs.ultralytics.com/reference/data/base – (16.05.2023).
49. Models [Електронний ресурс]. – Режим доступу: https://github.com/ultralytics/ultralytics?tab=readme-ov-file#models – (16.05.2023).
50. COCO Common Objects in Context [Електронний ресурс]. – Режим доступу: https://cocodataset.org/#home – (16.05.2023).
51. ByteTrack: Multi-Object Tracking by Associating Every Detection Box [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/2110.06864 – (16.05.2023).
52. Single image haze removal using dark channel prior [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/5206515
53. Vehicle Color Recognition using Convolutional Neural Network [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1510.07391 – (16.05.2023).
54. Single image dehazing by approximating and eliminating the additional airlight component [Електронний ресурс]. – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0925231220303726 – (16.05.2023).
55. Deep Residual Learning for Image Recognition [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1512.03385 – (16.05.2023).
56. ImageNet [Електронний ресурс]. – Режим доступу: https://www.image-net.org – (16.05.2023).

57. A Large and Diverse Dataset for Improved Vehicle Make and Model Recognition [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/8014855 – (16.05.2023).
58. The Earth’s Coordinate System (Latitude and Longitude) [Електронний ресурс]. – Режим доступу: https://www.icsm.gov.au/education/fundamentals-mapping/earths-coordinate-system – (16.05.2023).
59. API Reference [Електронний ресурс]. – Режим доступу: https://docs.mapbox.com/mapbox-gl-js/api – (20.05.2023).
60. Using ApexCharts to create charts in Vue.js [Електронний ресурс]. – Режим доступу: https://apexcharts.com/docs/vue-charts – (20.05.2023).
61. What is SDLC (Software Development Lifecycle)? [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/what-is/sdlc/?nc1=h_ls – (20.05.2023).
62. Software quality control [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Software_quality_control – (20.05.2023).
63. Top 15 Automation Testing Tools [Електронний ресурс]. – Режим доступу: https://katalon.com/resources-center/blog/automation-testing-tools – (20.05.2023).
64. View Jetson Orin Technical Specifications [Електронний ресурс]. – Режим доступу: https://www.nvidia.com/en-sg/autonomous-machines/embedded-systems/jetson-orin – (20.05.2023).
65. Fog computing [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Fog_computing – (20.05.2023).

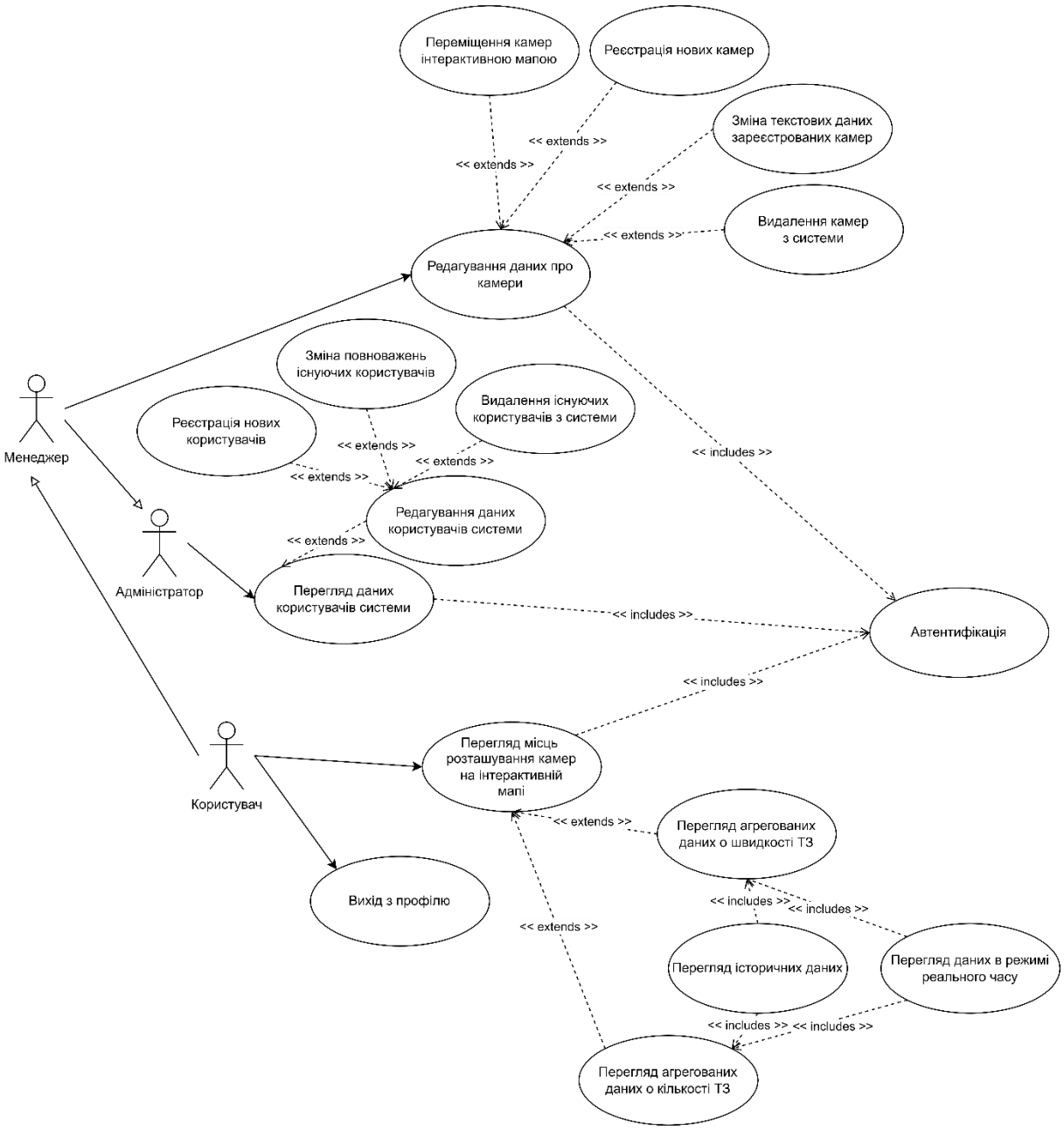
ДОДАТКИ

Додаток 1

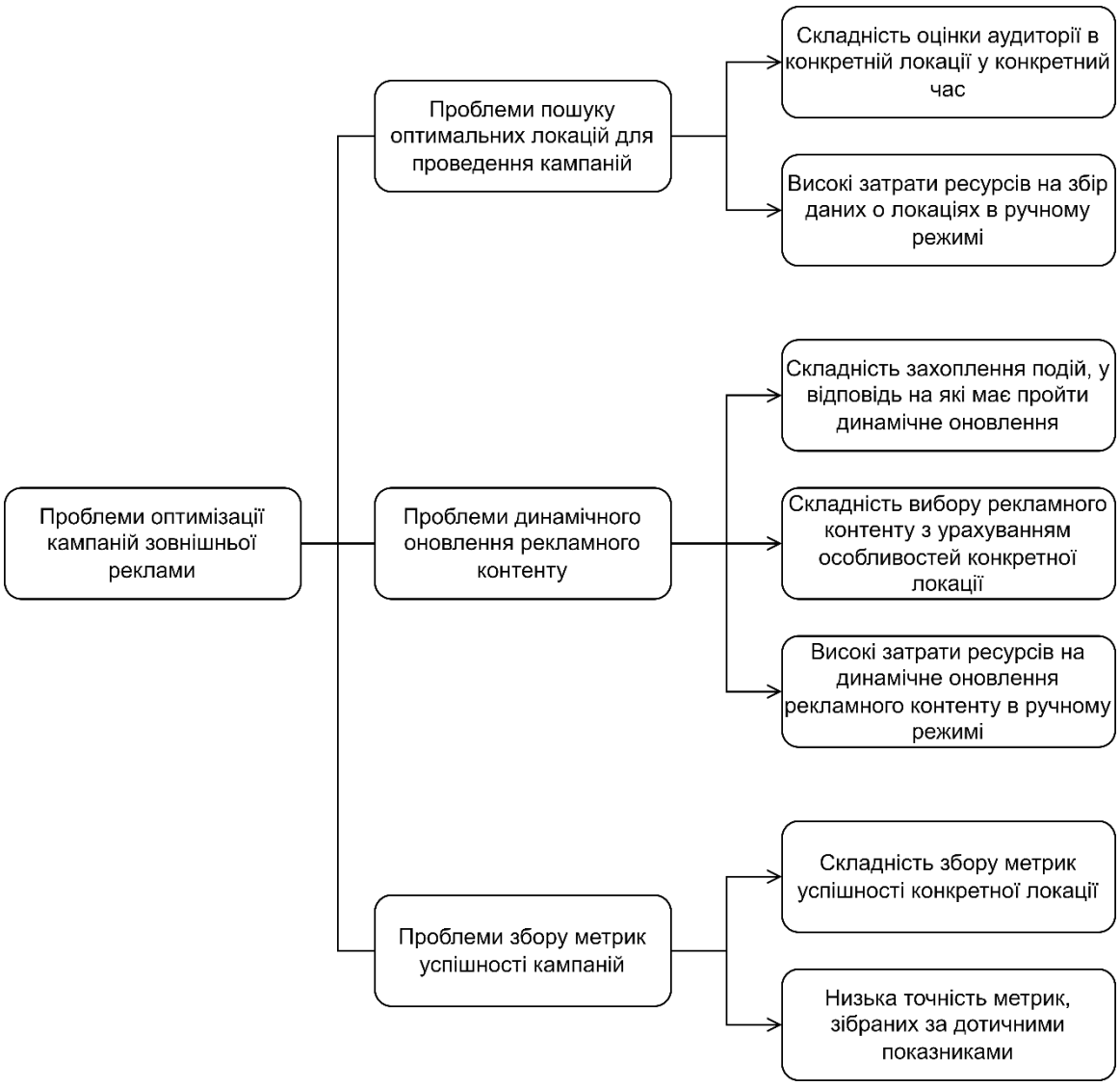
Копії графічних матеріалів



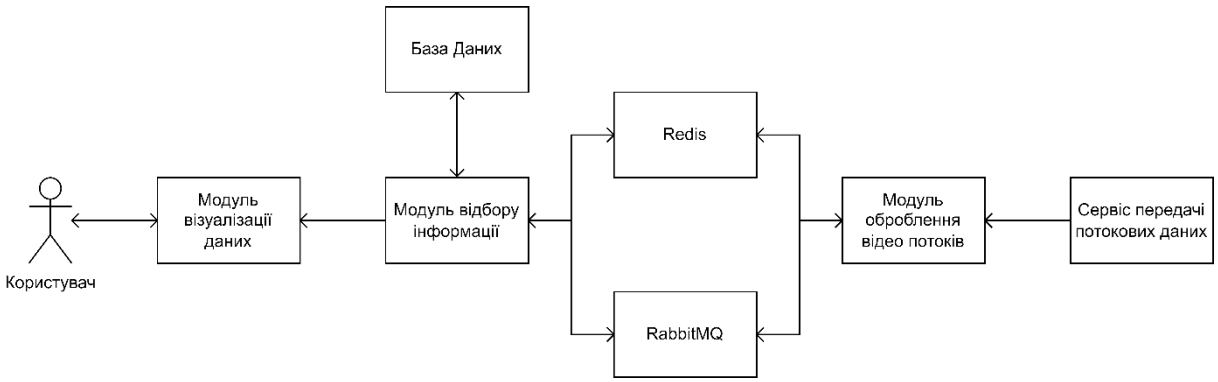
ДП.045440-06-99
Програмне забезпечення для інтелектуального аналізу даних дорожнього руху з використанням камер відеоспостереження. Структура бази даних. ERD-діаграма



ДП.045440-06-99
Програмне забезпечення для інтелектуального аналізу даних дорожнього руху з використанням камер відеоспостереження. Варіанти використання.
UML-діаграма



Дерево проблем проєкту
Простаков Олег, група КП-02



Узагальнена архітектура ПЗ
Простаков Олег, група КП-02

Додаток 2

Лістинг коду деяких класів-сервісів для відбору даних

```
from abc import ABC, abstractmethod
from typing import List, Dict, TypeVar, Generic
import numpy as np
from entities import Vehicle, Region

Frame = np.ndarray
TFeature = TypeVar('TFeature')


class BaseService(ABC, Generic[TFeature]):
    @abstractmethod
    def _should_process_vehicle(self, vehicle: Vehicle) -> bool:
        pass

    @abstractmethod
    def _preprocess_frame(self, frame: Frame) -> np.ndarray:
        pass

    @abstractmethod
    def _extract_feature(self, region: Region, preprocessed_frame: Frame) ->
TFeature | None:
        pass

    @abstractmethod
    def _update_vehicle(self, vehicle: Vehicle, feature: TFeature):
        pass

    def process(self, vehicles: List[Vehicle], frames: Dict[int, Frame]) ->
List[Vehicle]:
        for vehicle in vehicles:
            if not self._should_process_vehicle(vehicle):
                continue
            for region in vehicle.regions:
                sliced_frame = frames[region.frame_id][region.y1:region.y2,
region.x1:region.x2]
                preprocessed_frame = self._preprocess_frame(sliced_frame)
                feature = self._extract_feature(region, preprocessed_frame)
                if feature is not None:
                    self._update_vehicle(vehicle, feature)
                    break

        return vehicles



import cv2
import keras
import numpy as np
import tensorflow as tf
from numpy.compat import long

from entities import Vehicle, Region
from services.BaseService import BaseService, Frame


class _ChannelValue:
    val = -1.0
    intensity = -1.0
```

```
class ColorInferenceService(BaseService[str]):
    color_codes = {0: 'black', 1: 'blue', 2: 'cyan', 3: 'gray', 4: 'green', 5:
'red', 6: 'white', 7: 'yellow'}

    def __init__(self, color_picker_model: keras.Sequential):
        self._model = color_picker_model

    def _update_vehicle(self, vehicle: Vehicle, feature: str):
        vehicle.color = feature

    def _extract_feature(self, region: Region, preprocessed_frame: Frame) -> str |
None:
        tensor = preprocessed_frame.reshape((1, 100, 100, 3))
        prediction_tensor = self._model(tensor)
        color_code: int = int(tf.argmax(prediction_tensor, axis=1).numpy()[0])
        color_conf: float = prediction_tensor.numpy()[0][color_code]
        return self.color_codes[color_code] if color_conf > 0.8 else None

    def _preprocess_frame(self, frame: Frame) -> np.ndarray:
        rescaled_frame = cv2.resize(frame, (100, 100),
interpolation=cv2.INTER_AREA)
        light_intensity = self.__get_atmospheric_light_intensity(rescaled_frame)
        return self.__dehaze(rescaled_frame, light_intensity)

    def _should_process_vehicle(self, vehicle: Vehicle) -> bool:
        return vehicle.color is None

    def __get_atmospheric_light_intensity(self, img: np.ndarray) -> float:
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        top_num = int(img.shape[0] * img.shape[1] * 0.001)
        top_list = [_ChannelValue()] * top_num
        dark_channel = self.__find_dark_channel(img)

        for y in range(img.shape[0]):
            for x in range(img.shape[1]):
                val = img.item(y, x, dark_channel)
                intensity = gray.item(y, x)
                for t in top_list:
                    if t.val < val or (t.val == val and t.intensity < intensity):
                        t.val = val
                        t.intensity = intensity
                        break
        max_channel = _ChannelValue()
        for t in top_list:
            if t.intensity > max_channel.intensity:
                max_channel = t
        return max_channel.intensity

    # Finding the dark channel i.e. the pixel with the lowest R/G/B value
    @staticmethod
    def __find_dark_channel(img: np.ndarray) -> long:
        return np.unravel_index(np.argmin(img), img.shape)[2]

    # Finding a coarse image which gives us a transmission map
    @staticmethod
    def __coarse(minimum, x, maximum):
        return max(minimum, min(x, maximum))

    # Uses values from other functions to aggregate and give us a clear image
```

```
    def __dehaze(self, img: np.ndarray, light_intensity: float, window_size=20,
t0=0.55, w=0.95) -> np.ndarray:
        size = (img.shape[0], img.shape[1])

        out_img = np.zeros(img.shape, img.dtype)

        for y in range(size[0]):
            for x in range(size[1]):
                x_low = max(x - (window_size // 2), 0)
                y_low = max(y - (window_size // 2), 0)
                x_high = min(x + (window_size // 2), size[1])
                y_high = min(y + (window_size // 2), size[0])

                slice_img = img[y_low:y_high, x_low:x_high]

                dark_channel = self.__find_dark_channel(slice_img)
                t = 1.0 - (w * img.item(y, x, dark_channel) / light_intensity)

                out_img.itemset((y, x, 0), self.__coarse(0, ((img.item(y, x, 0) -
light_intensity) /
                                                             max(t, t0) +
light_intensity), 255))
                out_img.itemset((y, x, 1), self.__coarse(0, ((img.item(y, x, 1) -
light_intensity) /
                                                             max(t, t0) +
light_intensity), 255))
                out_img.itemset((y, x, 2), self.__coarse(0, ((img.item(y, x, 2) -
light_intensity) /
                                                             max(t, t0) +
light_intensity), 255))
        return out_img



import torch
import numpy as np
import pandas as pd
from PIL import Image
from torch import nn
from torchvision import transforms

from entities import Vehicle, Region
from services.BaseService import BaseService, Frame


class _ModelMake:
    def __init__(self, model: str, make: str):
        self.model = model
        self.make = make


class VmmrPredictorService(BaseService[_ModelMake]):
    def __init__(self, predictor_model: nn.Module, vmmr_classes: pd.DataFrame):
        self._model = predictor_model
        self._vmmr_classes = vmmr_classes
        self._device = torch.device('cuda')

    def _update_vehicle(self, vehicle: Vehicle, feature: _ModelMake):
        vehicle.model = feature.model
        vehicle.make = feature.make
```

```
    def _extract_feature(self, region: Region, preprocessed_frame: Frame) ->
_ModelMake | None:
        with torch.no_grad():
            output = self._model(preprocessed_frame)
        _, preds = torch.topk(output, 3)

        preds = torch.transpose(preds, 0, 1)
        preds = preds.cpu()  # Send tensor to cpu
        preds = pd.DataFrame(preds.numpy(), columns=["Classencoded"])  # Convert
to dataframe

        class_encoded_matches = pd.merge(self._vmmr_classes, preds, how="inner")
        class_encoded_matches = pd.merge(preds, class_encoded_matches, how="left",
on="Classencoded", sort=False)
        classname_matches = class_encoded_matches["Classname"].unique()
        class_parts = str(classname_matches[0]).split(" ")
        make = class_parts[0]
        model = " ".join(class_parts[1:-1])
        return _ModelMake(model, make)

    def _should_process_vehicle(self, vehicle: Vehicle) -> bool:
        return vehicle.model is None and vehicle.model is None

    def _preprocess_frame(self, frame: Frame) -> np.ndarray:
        img_transforms = transforms.Compose([transforms.Resize((256, 256)),
                                             transforms.ToTensor(),
                                             transforms.Normalize(mean=[0.485,
0.456, 0.406], std=[0.229, 0.224, 0.225])])

        img = Image.fromarray(frame.astype('uint8'), 'RGB')
        img = img_transforms(img)
        img = img.to(self._device)
        img = img.unsqueeze(0)
        return img
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ ДОРОЖНЬОГО РУХУ З ВИКОРИСТАННЯМ КАМЕР ВІДЕОСПОСТЕРЕЖЕННЯ

Виконав: Простаков Олег Дмитрович

Керівник: доц. кафедри ПЗКС, к.т.н., доц. Олещенко Любов Михайлівна

Київ – 2024

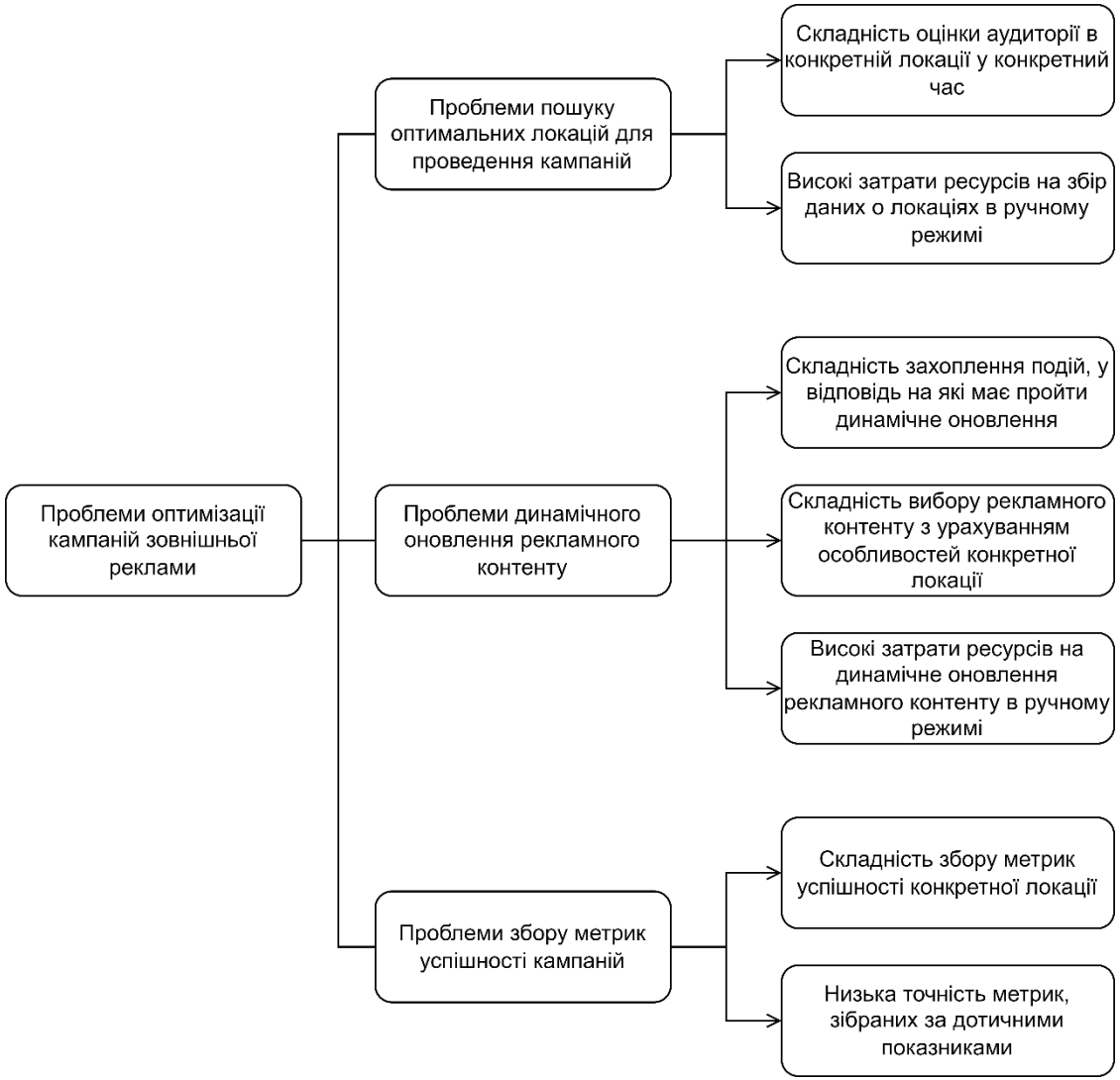
ПОСТАНОВКА ЗАДАЧІ

Мета проєкту: розробити програмне забезпечення для збору та аналізу даних дорожнього руху з місць проведення кампаній зовнішньої реклами.

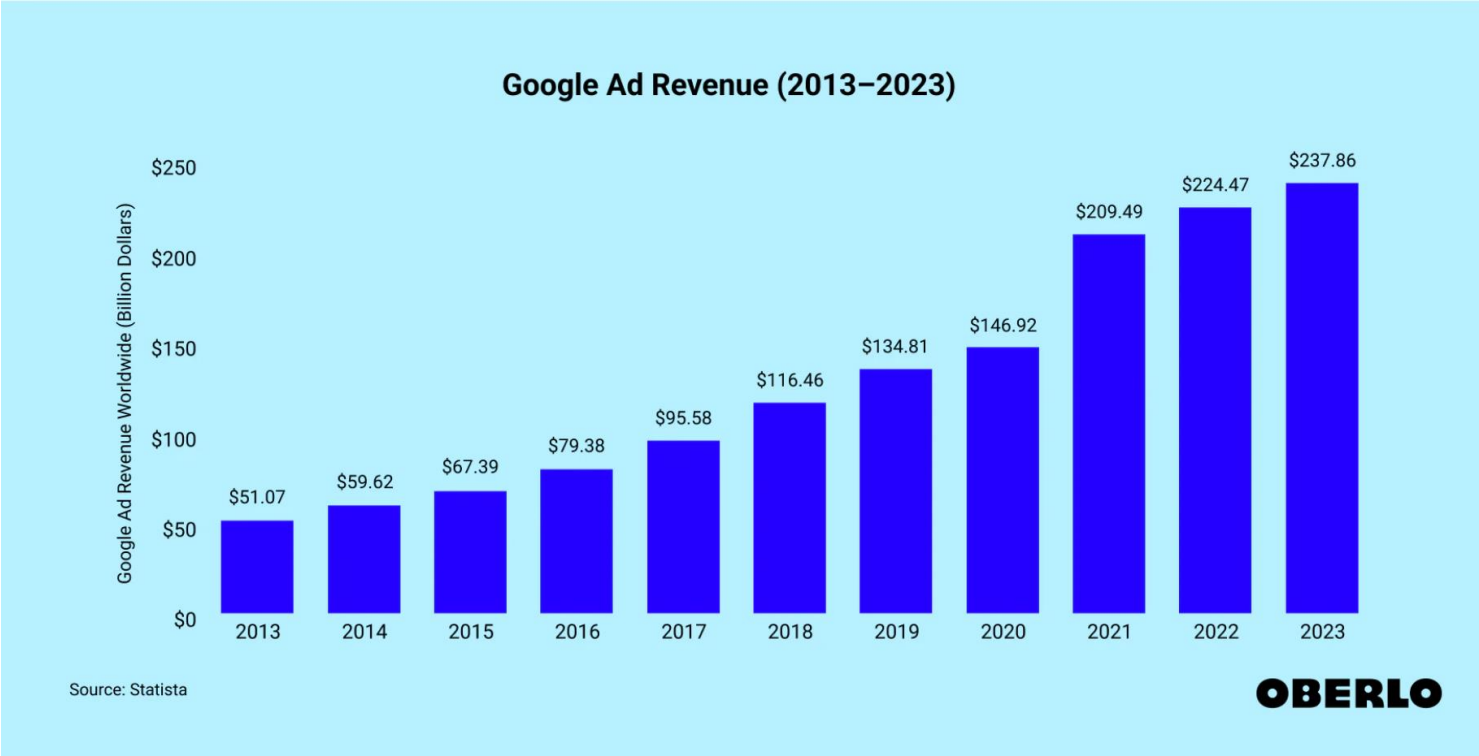
Завдання:

1. Дослідити проблеми, що виникають при проведенні кампаній зовнішньої реклами.
2. Проаналізувати існуючі програмні рішення.
3. Визначити вимоги до розроблюваного ПЗ.
4. Обрати засоби реалізації ПЗ.
5. Розробити та протестувати ПЗ відповідно до вимог.
6. Сформулювати шляхи подальшого розвитку проєкту.

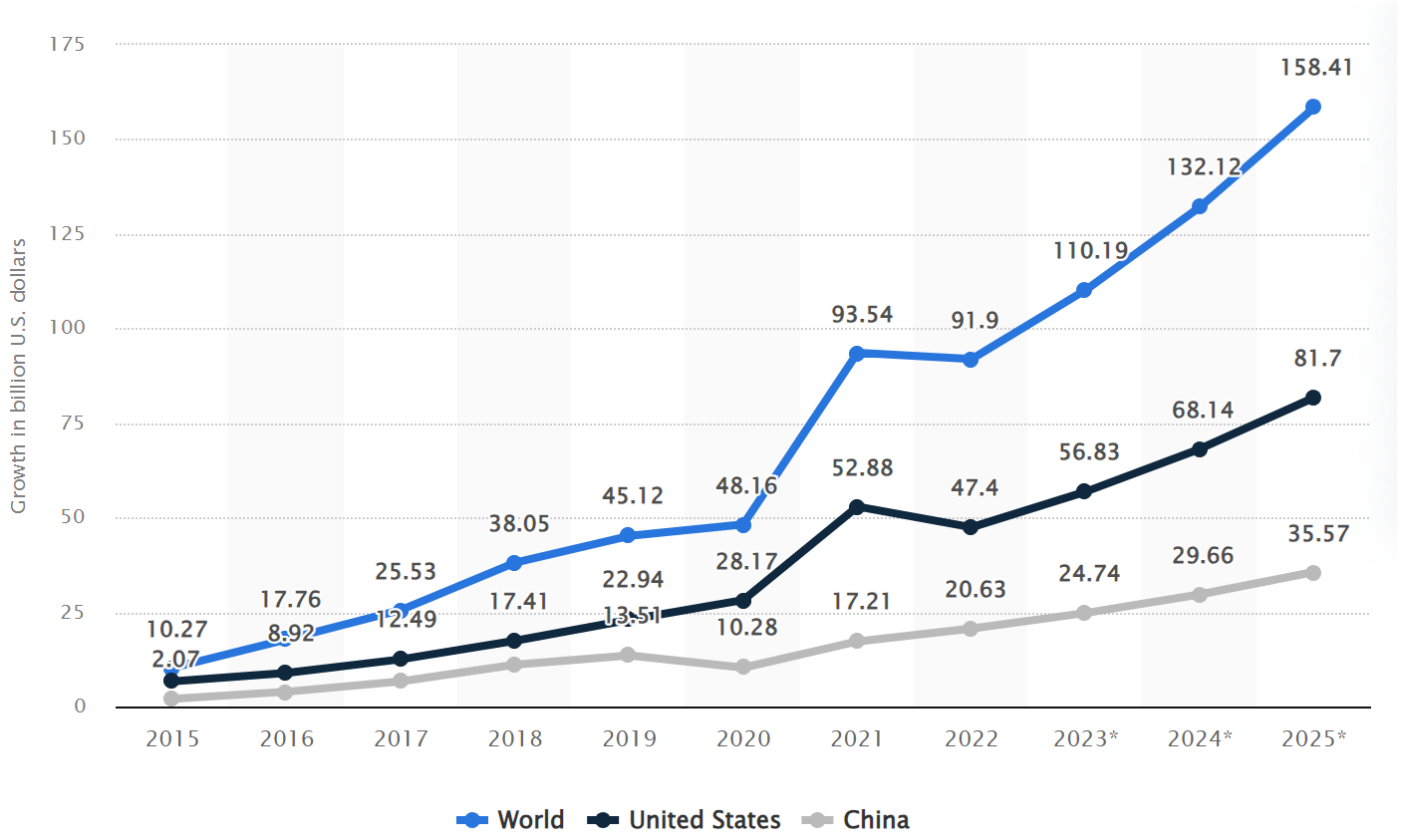
ДЕРЕВО ПРОБЛЕМ



АКТУАЛЬНІСТЬ

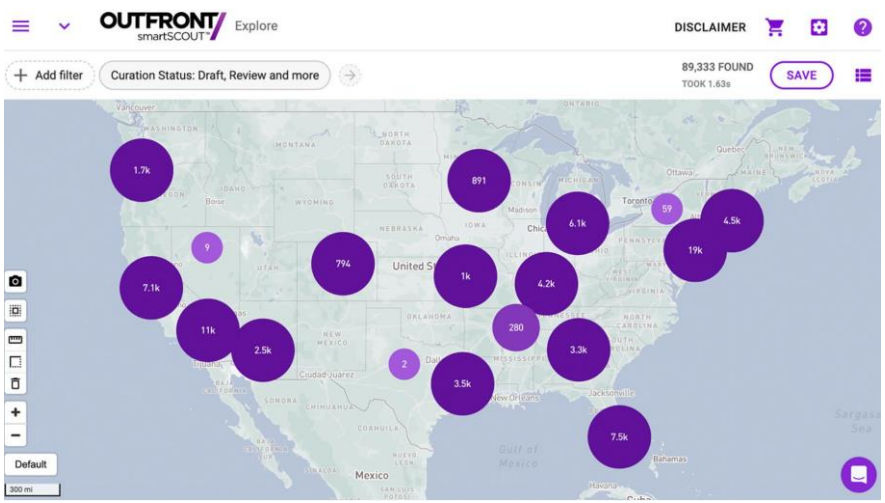


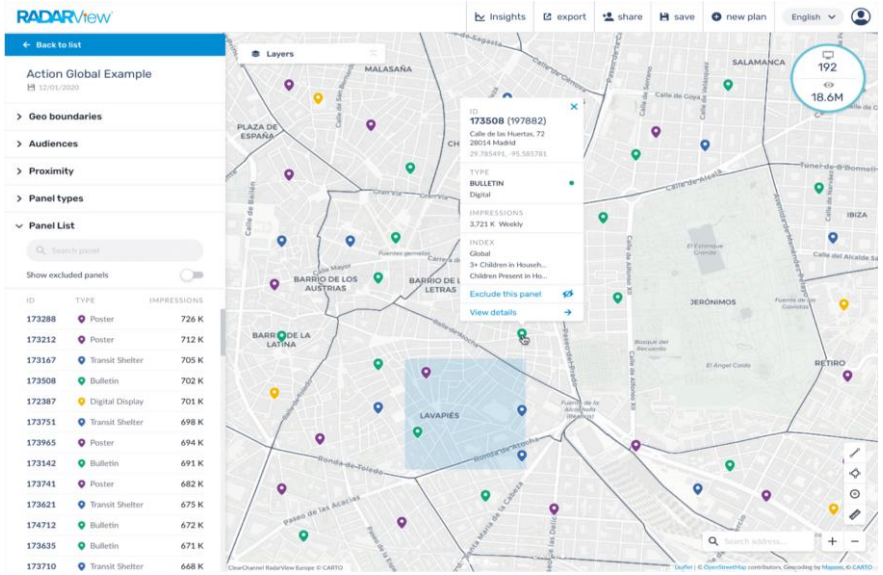
АКТУАЛЬНІСТЬ



Щорічний розмір інвестицій в технології ШІ

ІСНУЮЧІ ПРОГРАМНІ РІШЕННЯ





JCDecaux

ВИМОГИ ДО ПЗ

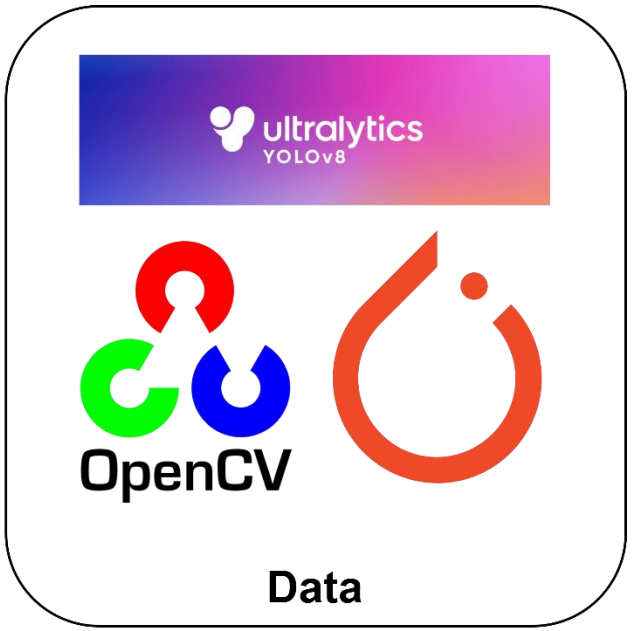
Функціональні:

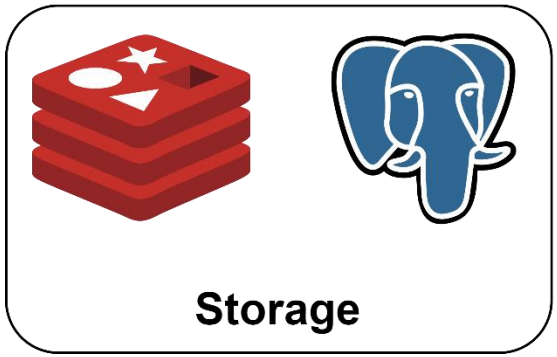
- Збір інформації про кількість унікальних ТЗ проміжок часу.
- Збір інформації про середню швидкість ТЗ.
- Збір даних про колір, моделі та виробників ТЗ, помічених камерою відеоспостереження.
- Візуалізація зібраної інформації на вебресурсі.
- Можливість реєстрації та авторизації користувача вебресурсу.

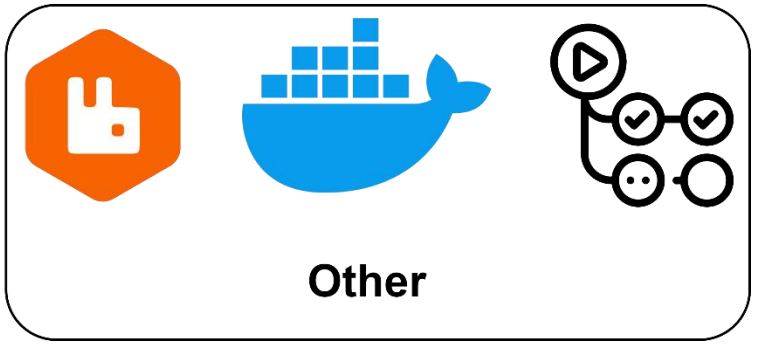
Нефункціональні:

- Ефективність використання обчислювальних ресурсів під час оброблення відео потоків.
- Ефективність використання обчислювальних ресурсів під час візуалізації інформації.
- Простота користувацького інтерфейсу вебресурсу.
- Висока точність зібраних даних.

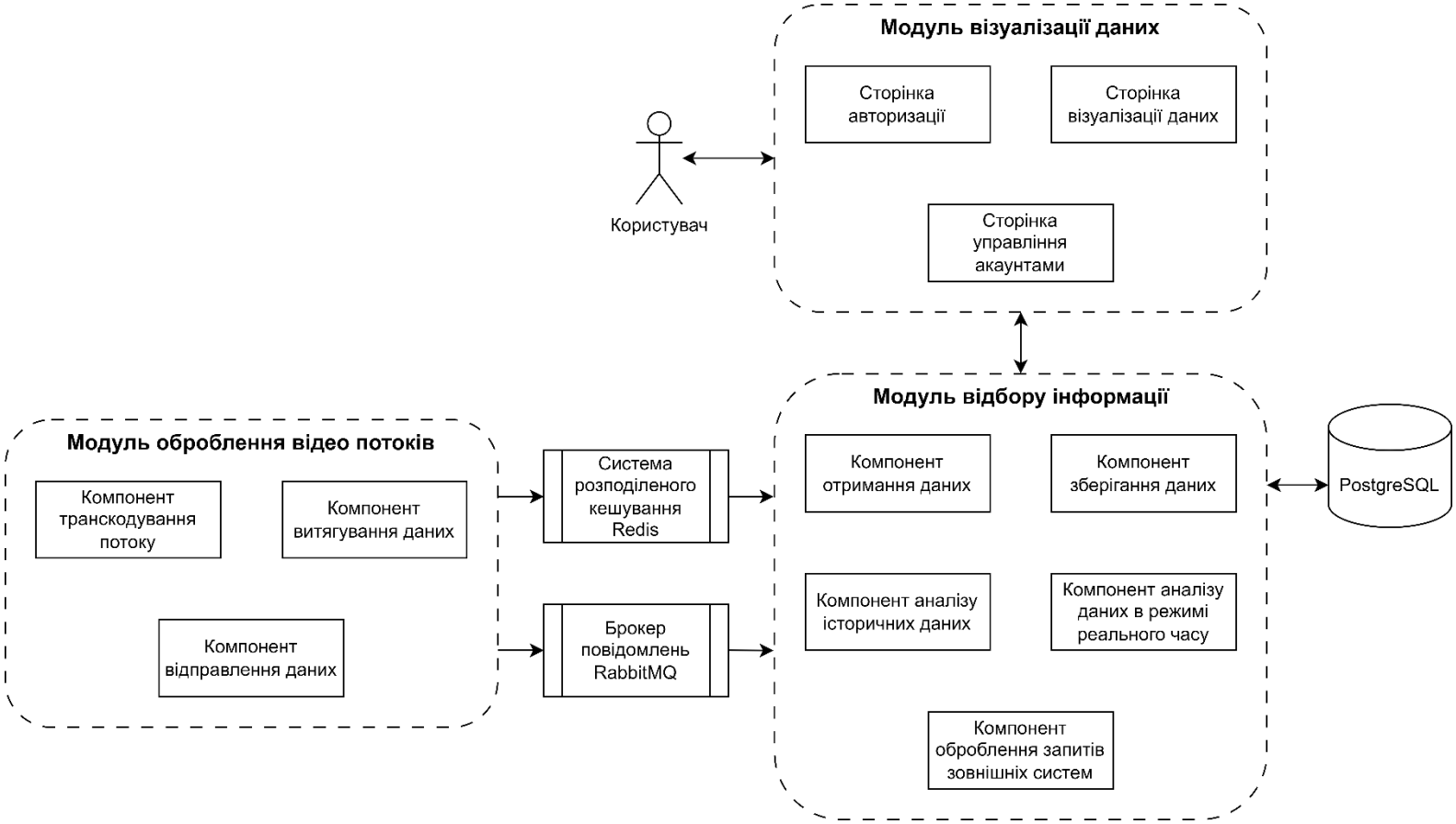
ТЕХНОЛОГІЇ, ЩО ВИКОРИСТОВУЄ ПРОДУКТ



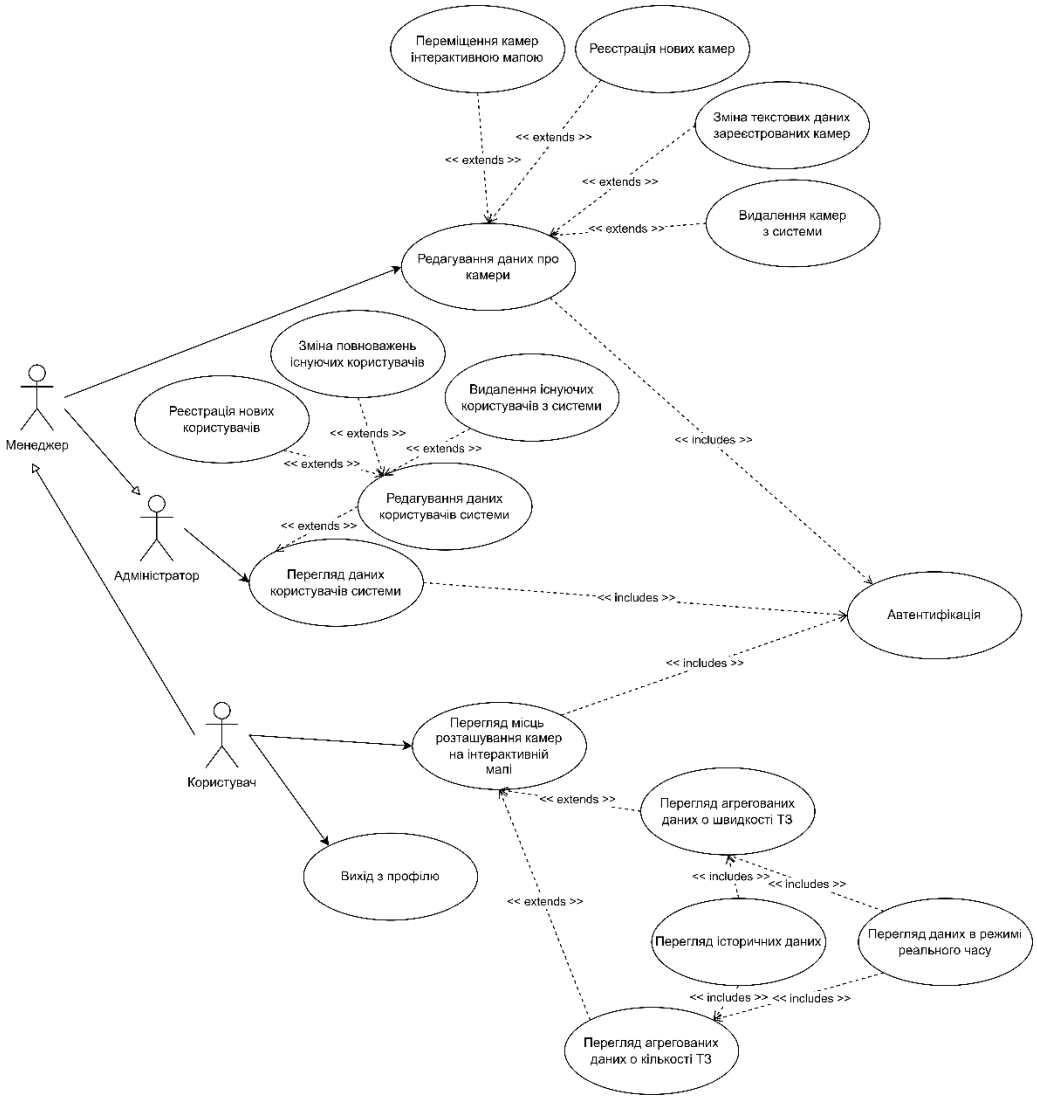




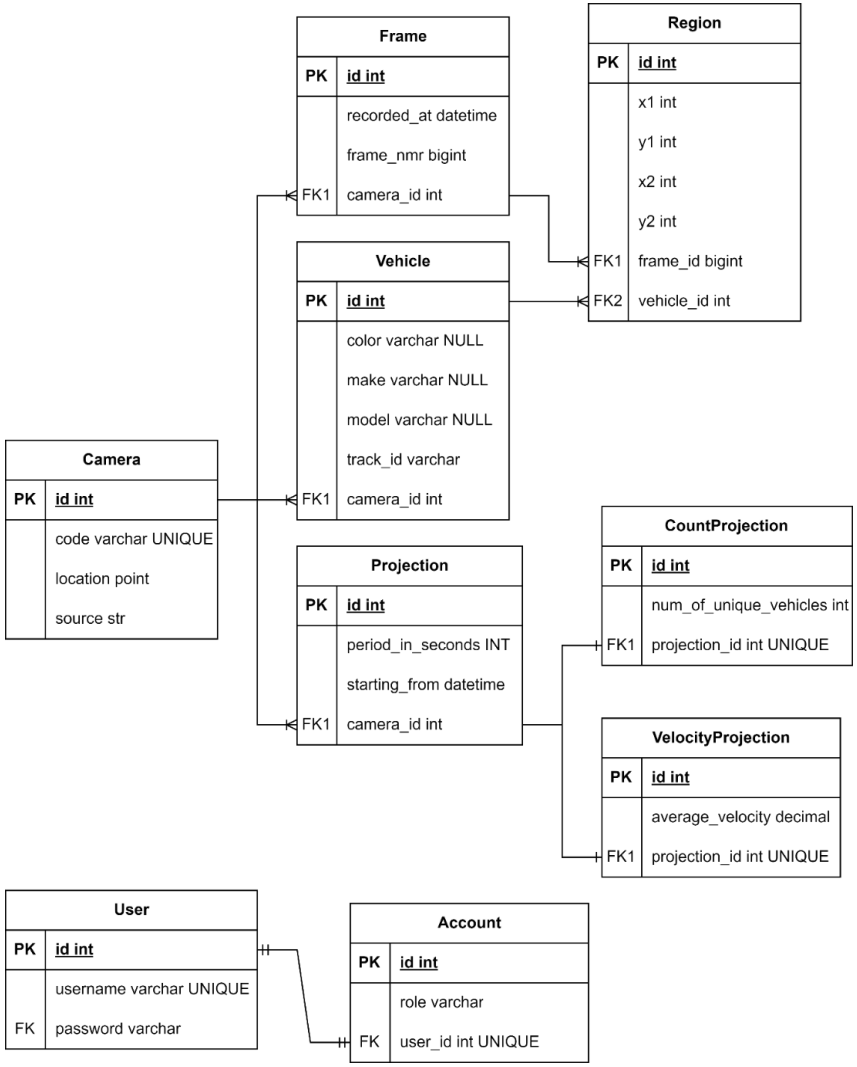
УЗАГАЛЬНЕНА АРХІТЕКТУРА



ВАРІАНТИ ВИКОРИСТАННЯ.
UML-діаграма



СТРУКТУРА БД. ERD-діаграма



ПРИКЛАД ВІЗУАЛІЗАЦІЇ ДАНИХ, ЩО ЗБИРАЮТЬСЯ

Приклад візуалізації даних на фоні відео потоку

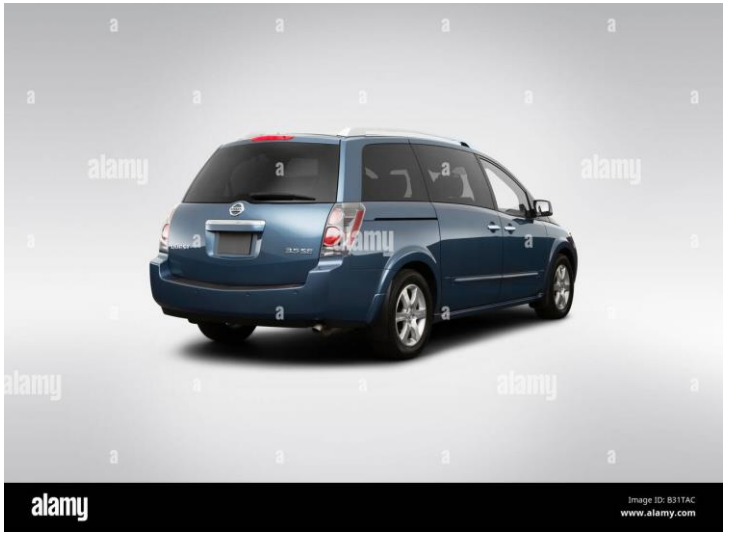


Фото автомобіля моделі Nissan Quest

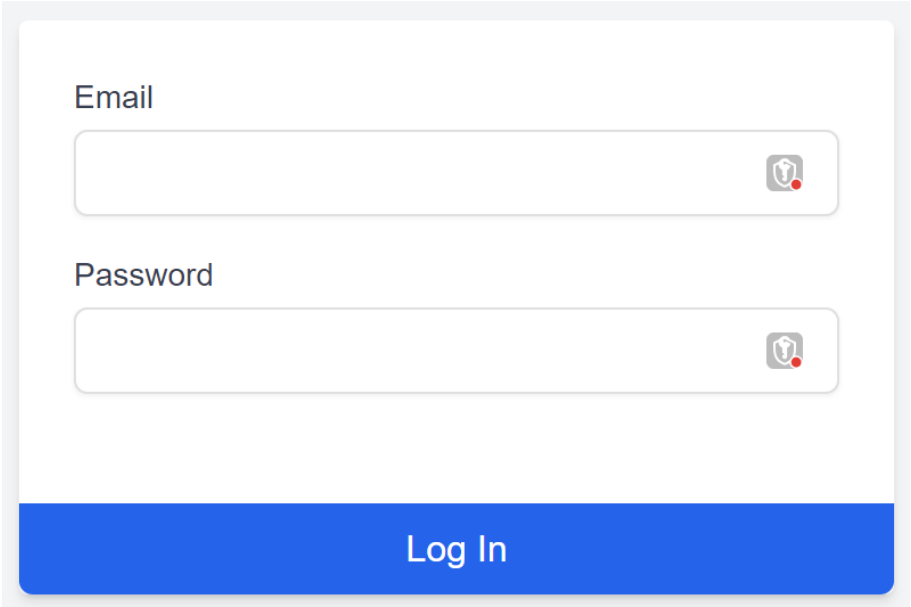
ПРИКЛАД НЕЙТРАЛІЗАЦІЇ ТУМАННОСТІ ЗОБРАЖЕННЯ

Оригінальні зображення

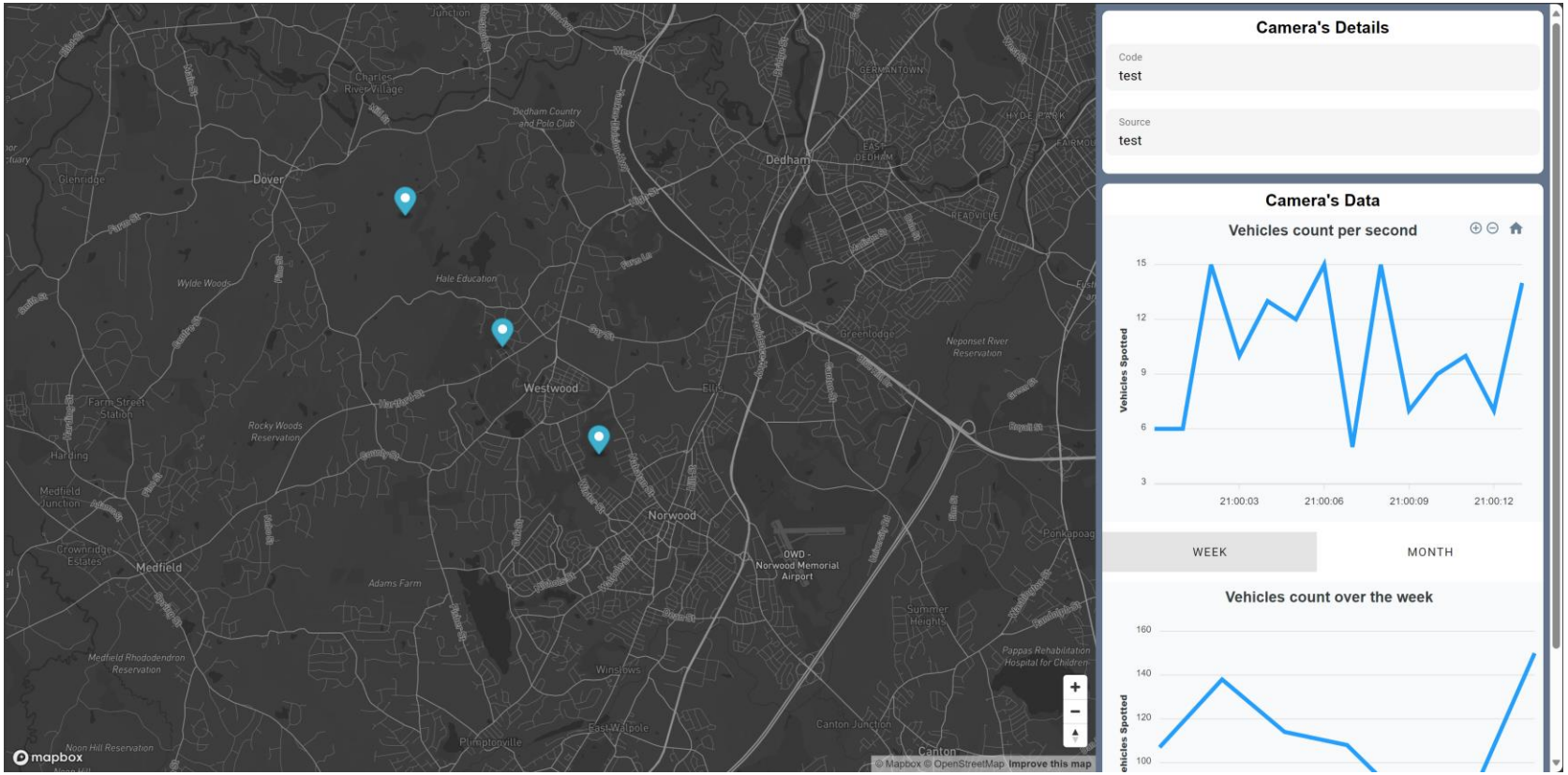
Зображення після зміни масштабу

Зображення після нейтралізації забруднення

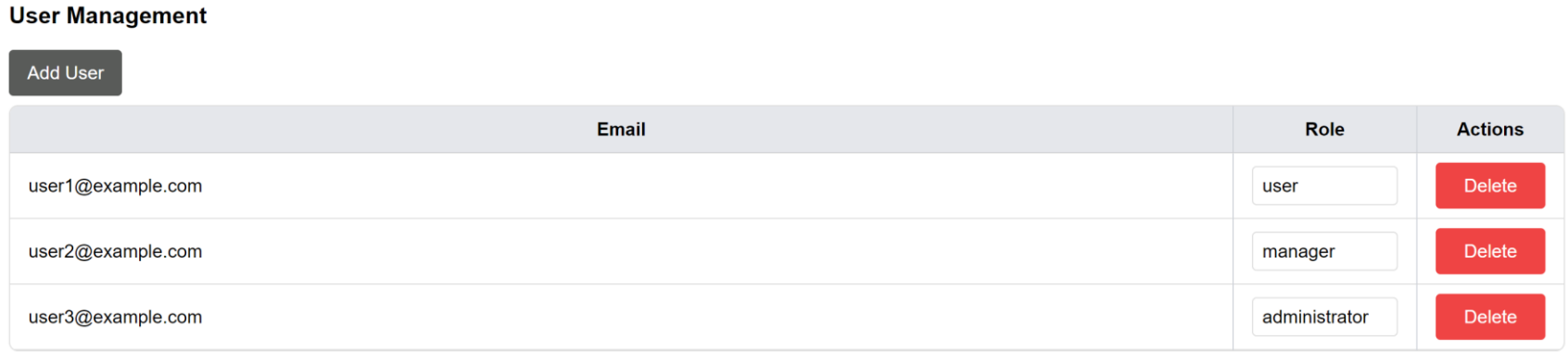
СТОРІНКА АВТОРИЗАЦІЇ



СТОРІНКА ВІЗУАЛІЗАЦІЇ ДАНИХ



СТОРІНКА УПРАВЛІННЯ АКАУНТАМИ КОРИСТУВАЧІВ



РЕЗУЛЬТАТИ ТЕСТУВАННЯ ПРОДУКТИВНОСТІ

Пристрій	Мінімальний час	Середній час	Максимальний час
CUDA	22.5мс	27.8мс	39.1мс
CPU	201.3мс	225.6мс	301.3мс

Час оброблення кадру на різних пристроях

ПЕРЕВАГИ РОЗРОБЛЕНОГО ПЗ

Порівнюючи з існуючими програмними рішеннями, можна виділити наступні переваги розробленого продукту:

- сумісність з відео потоками різних форматів;
- наявність платформи для візуалізації даних;
- продуктивність;
- можливість горизонтального масштабування;
- підтримка моделі туманних обчислень;
- можливість оброблення даних в режимі реального часу.

ШЛЯХИ ПОДАЛЬШОГО РОЗВИТКУ ПРОЄКТУ

Розроблене ПЗ може бути вдосконаленим шляхами:

- оптимізації використаних моделей машинного навчання;
- збільшенню кількості атрибутів даних, що збирає система;
- інтеграції можливості аналізу пішохідних потоків;
- адаптації вебінтерфейсу під мобільні пристрої;
- інтеграції сторонніх джерел даних, таких як оператори мобільного зв’язку, до системи;
- використання хмарних технологій для розгортання системи.

ВИСНОВКИ

В процесі роботи над даним проєктом було:

1. Досліджено проблеми, що виникають при проведенні кампаній зовнішньої реклами.
2. Проаналізовано існуючі програмні рішення.
3. Визначено вимоги до розроблюваного ПЗ.
4. Проаналізовано та обґрунтовано вибір засоби реалізації ПЗ.
5. Розроблено ПЗ відповідно до поставлених вимог.
6. Проведено тестування отриманого ПЗ та отримано результати, згідно з якими воно відповідає вимогам.
7. Сформульовано шляхи подальшого розвитку проєкту.

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

WEB-РЕСУРС ЦЕНТРУ ЕЛЕКТРОННОЇ ОСВІТИ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Любов ОЛЕЩЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олег ПРОСТАКОВ

2023

ЗМІСТ

1. Об’єкт випробувань……...3
2. Мета тестування…….3
3. Методи тестування……3
4. Засоби та порядок тестування……..4

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмна система для інтелектуального аналізу відео потоків, серверна частина якої розроблена мовою програмування Python з використанням моделей машинного навчання YOLOv8, ByteTrack, ResNet-50, тощо для аналізу даних, а також фреймворку FastAPI для вебкомпоненту; клієнтська – мовою програмування JavaScript з розширенням TypeScript та фреймворком VueJS; а в якості СУБД якої було обрано PostgreSQL.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) функціональна працездатність модулів програмної системи;
2) коректність взаємодії різних модулів програмної системи;
3) належна швидкість обробки кадрів модулем обробки відео потоків;
4) забезпечення належного рівня безпеки даних;
5) зручність роботи з вебзастосунком програмної системи;
6) відповідність функціональності вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівнях «системного» та «інтеграційного» тестувань.

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Critical path test (базове тестування);

2) ручне тестування;

3) тестування продуктивності різних компонентів системи;

4) димове тестування

5) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується вручну, засобами мов Python та JavaScript, а також платформи Google Colab для ізольованого тестування різних моделей машинного навчання, що використовує програмна система.

Працездатність перевіряється шляхом:

1) динамічного ручного тестування вебзастосунку – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування вебзастосунку в різних браузерах;
5) тестування при максимальному навантаженні;
6) тестування стабільності роботи при різних умовах;
7) тестування зручності використання;
8) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2024 р.

WEB-РЕСУРС ЦЕНТРУ ЕЛЕКТРОННОЇ ОСВІТИ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Любов ОЛЕЩЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олег ПРОСТАКОВ

2024

ЗМІСТ

1. Опис структури вебзастосунку ……..……3
2. Процедура авторизації користувача …….…….…….……..3
3. Візуалізація даних на інтерактивній мапі …….……...……4
4. Візуалізація даних на графіках ;...……...……..…….……..7
5. Редагування даних в системі…….……..……9
6. Управління акаунтами користувачів …….……..……..…...11

1. Опис структури вебзастосунку

У розробленому програмному забезпеченні вебзастосунок виконує роль інтерфейсу, завдяки якому користувачі взаємодіють з усією програмною системою. Складовими вебзастосунку є серверна та клієнтська частини. Напряму користувачі можуть взаємодіяти з клієнтською частиною, а саме, вебсторінками, через браузер.

Неавторизовані користувачі мають доступ виключно до сторінки авторизації. Авторизованим користувачам доступна більша кількість сторінок та дій на цих сторінках, в залежності від рівня повноважень акаунта користувача. Розроблене програмне забезпечення розрізняє три рівня прав доступу акаунтів: “Користувач”, “Менеджер” та “Адміністратор”, де кожен наступний рівень розширює можливості попереднього.

2. Процедура авторизації користувача

При підключенні до вебзастосунку, користувача автоматично перенаправляє на сторінку авторизації за відносною адресою /login (рис. 1).

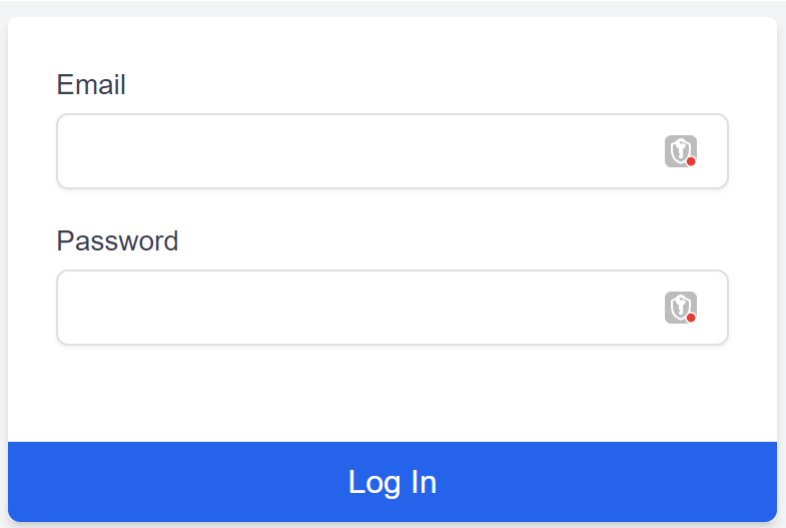


Рис. 1. Сторінка авторизації користувача

Для авторизації, користувач має ввести електронну пошту та пароль, за якими адміністратор системи раніше зареєстрував користувача. За умови неможливості знаходження зареєстрованого користувача за введеними даними, система виведе користувачу повідомлення про помилку (рис. 2).

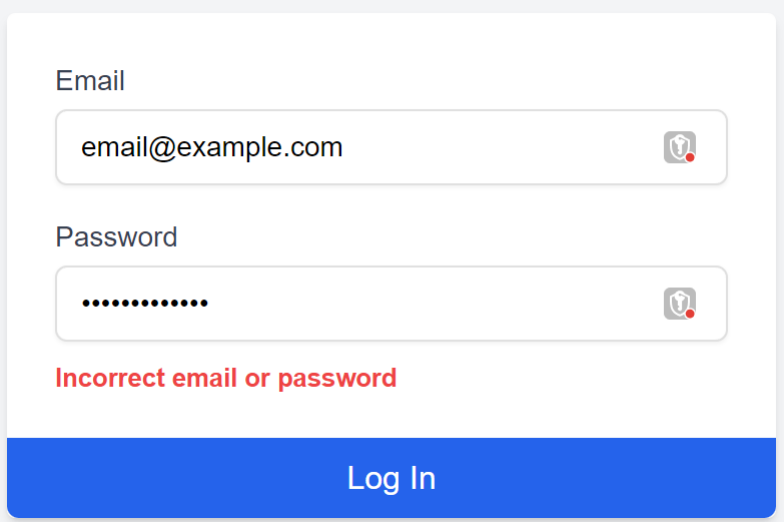


Рис. 2. Спроба авторизації з невірними даними

3. Візуалізація даних на інтерактивній мапі

Перегляд результатів візуалізації даних на інтерактивній мапі доступний користувачам з рівнем прав доступу “Користувач”. Сама інтерактивна мапа знаходиться в лівій частині головної сторінки вебзастосунку за відносною адресою /data. На мапі блакитними маркерами відмічені камери відеоспостереження, що розташовані в різних місцях проведення кампаній зовнішньої реклами (рис. 3).

Рис. 3. Інтерактивна мапа

Як було згадано раніше, мапа є інтерактивною. Затиснувши ліву кнопку миші, користувачі можуть змінювати місцевість, що відображається, пересуваючи мишу. Прогортаючи колесо миші, користувачі можуть змінювати масштаб мапи (рис. 4). Затиснувши праву кнопку миші, користувачі можуть обертати мапу, рухаючи мишу вліво та вправо (рис. 5), або змінювати кут нахилу мапи, рухаючи мишу вгору або вниз (рис. 6)

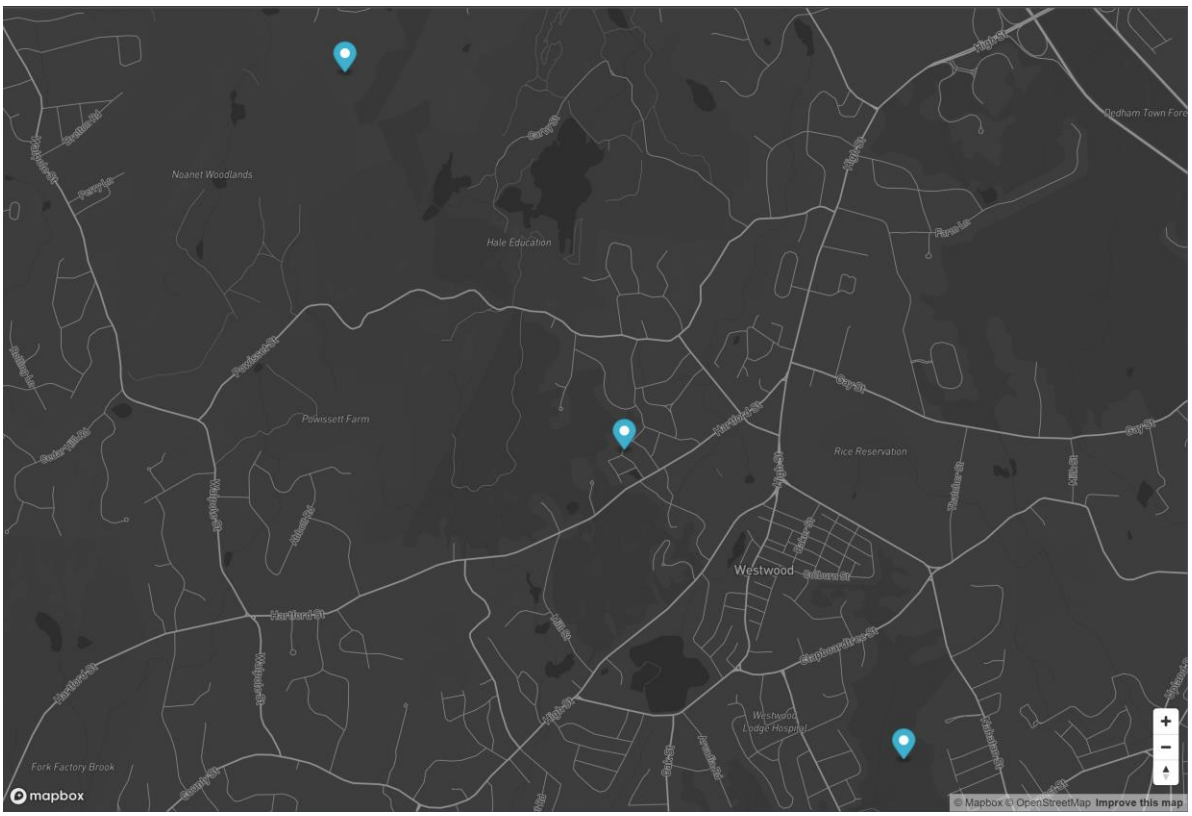


Рис. 4. Інтерактивна мапа, збільшений масштаб

Рис. 5. Інтерактивна мапа обернута на -90°

Рис. 6. Інтерактивна мапа зі зміненим кутом нахилу

4. Візуалізація даних на графіках

Перегляд результатів візуалізації даних на графіках так само доступний користувачам з рівнем прав доступу “Користувач”. Компонент графіків, побудованих на основі даних зібраних з конкретної камери, знаходиться в правій частині головної сторінки вебзастосунку за відносною адресою /data (рис. 7).

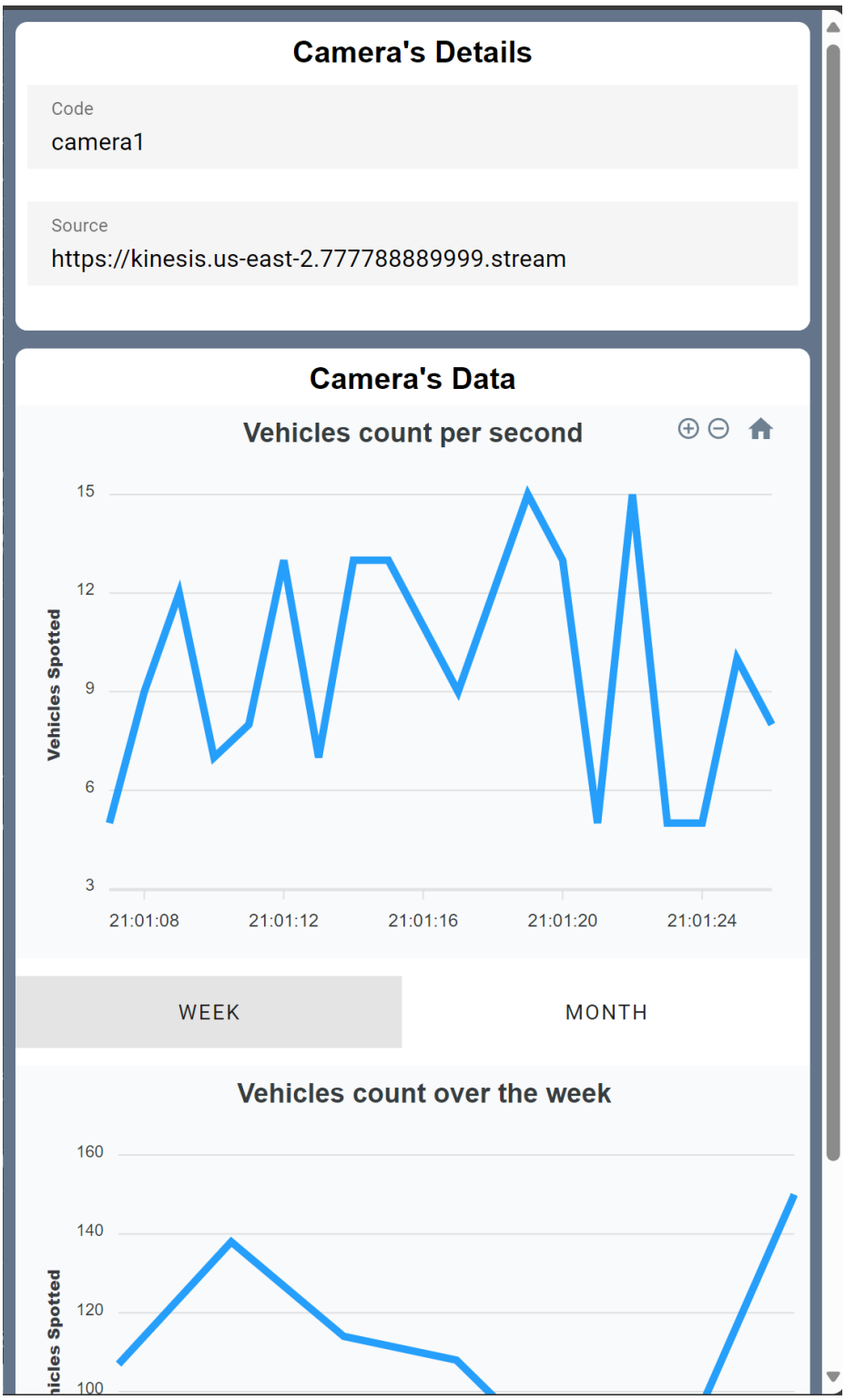


Рис. 7. Візуалізація за допомогою графіків

На графіку історичних даних, користувач може обрати період, за який візуалізуються дані – тиждень або місяць (рис. 8).

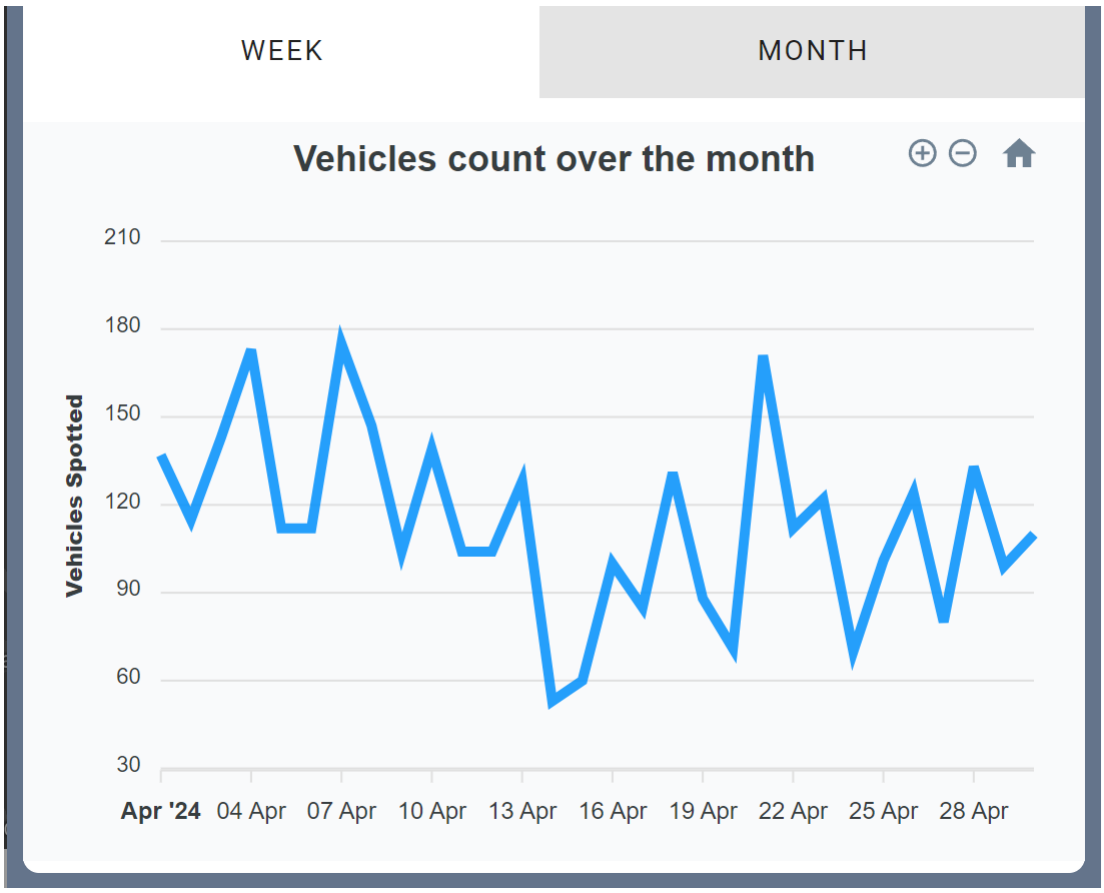


Рис. 8. Візуалізація історичних даних за місяць

5. Редагування даних в системі

Редагування даних доступно користувачам з рівнем прав доступу “Менеджер”. Такі користувачі можуть додавати нові камери до системи, змінювати та видаляти існуючі камери з системи.

Додавання камер виконується за допомогою інтерактивної мапи. Для додавання нової камери, користувачу потрібно зробити подвійне натискання в точку мапи, що відповідає локації нової камери. Маркер новоствореної камери матиме червоний колір (рис. 9), повідомляючи користувача про те, що додані дані ще не було збережено.

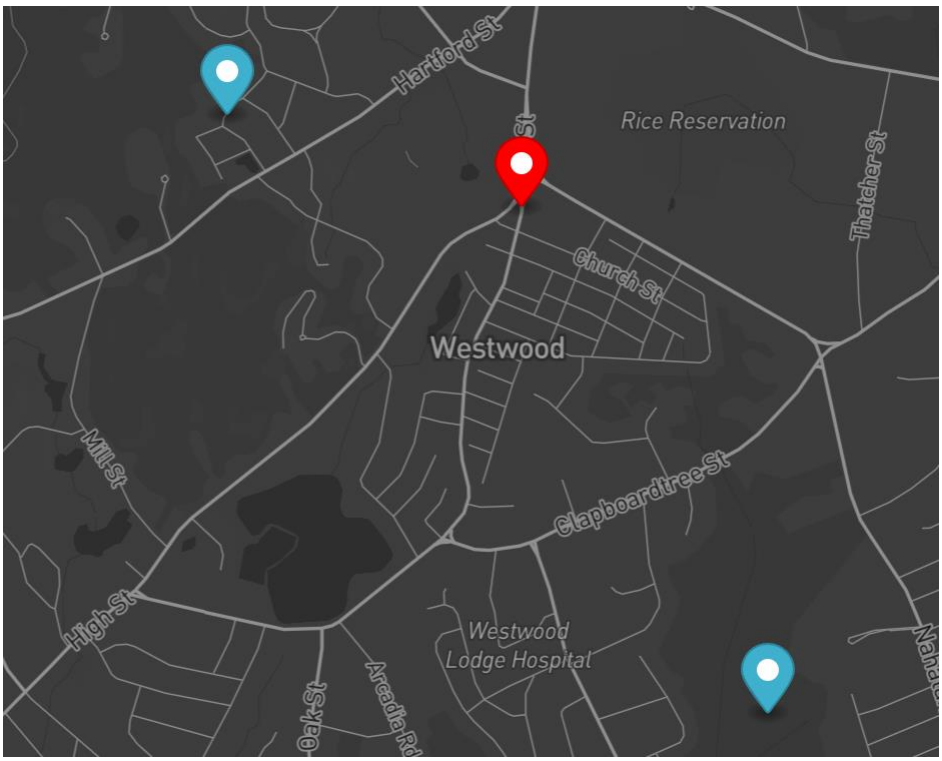


Рис. 9. Фрагмент мапи з новоствореним маркером

Також, користувачі можуть змінювати локації існуючих камер. Для цього потрібно клацнути мишею по існуючому маркеру та, не відпускаючи ліву кнопку, перетягнути його в потрібну локацію.

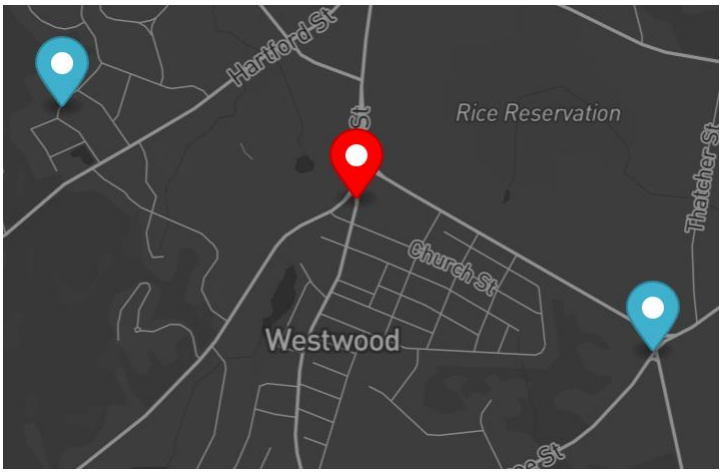
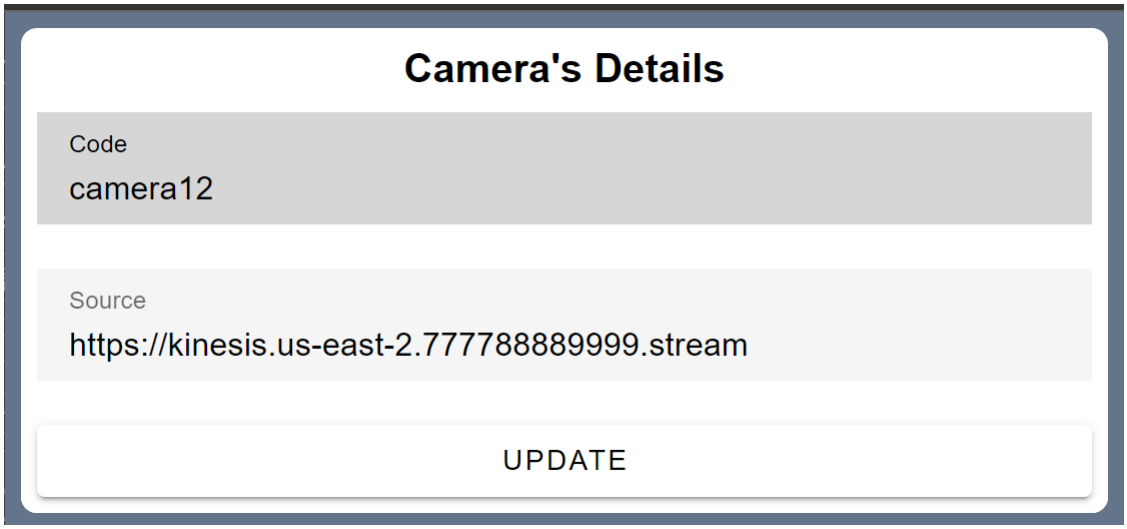


Рис. 10. Фрагмент мапи з пересунутим маркером

Для зміни інших даних камери, користувачам необхідно ввести потрібні значення в поля форми “Camera’s Details”, що розташована вгорі компоненту з графіками (рис. 11), та натиснути кнопку “Update”.



6. Управління акаунтами користувачів

Управління акаунтами користувачів доступно виключно користувачам з рівнем прав доступу “Адміністратор” і здійснюється завдяки сторінці управління акаунтами, що знаходиться за відносною адресою /users. На даній сторінці (рис. 12), користувачі можуть переглядати дані зареєстрованих акаунтів, змінювати рівні прав доступу акаунтів, видаляти або додавати (рис. 13) нові акаунти, натиснувши на кнопку “Add User” та заповнивши необхідні поля.

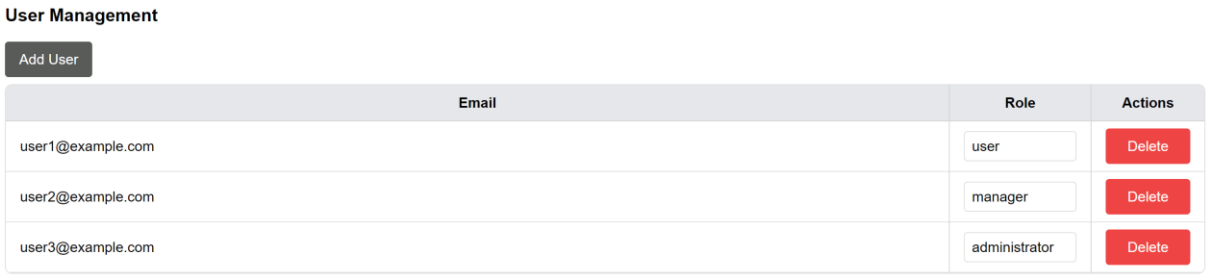


Рис. 12. Сторінка управління акаунтами

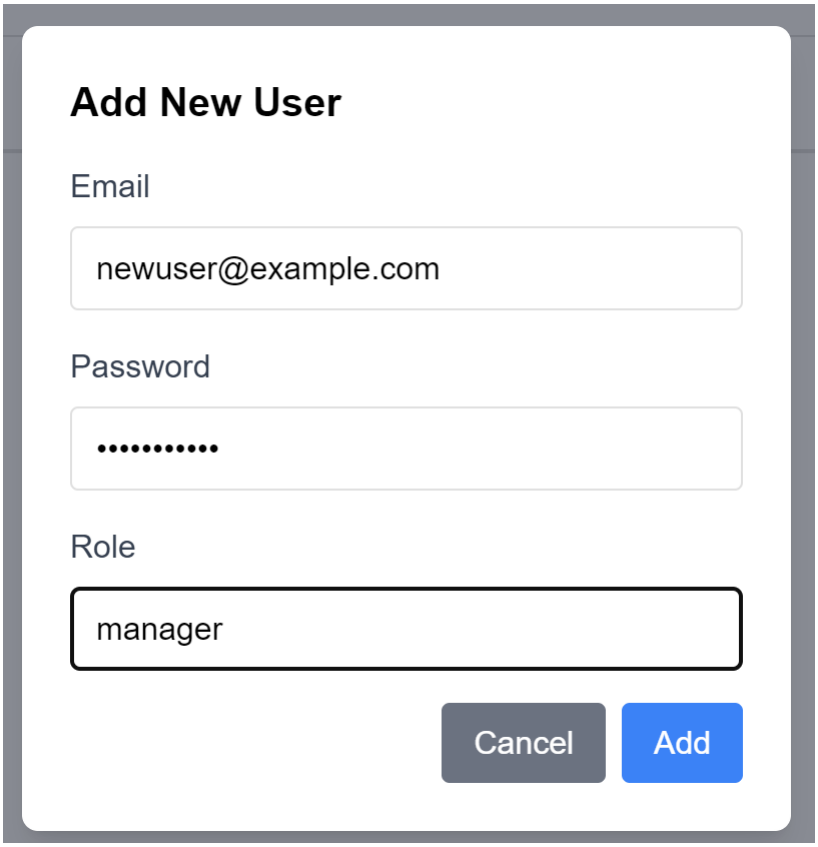


Рис. 13. Форма реєстрації нового акаунта