Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Автоматизована клієнт-серверна система транспортних перевезень з функцією інтелектуального моніторингу стану водія»

Виконав: студент 2 курсу, групи 1АКІТР-24м спеціальності 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(шифр і назва спеціальності)

Юрій МОРГУН
(ПІБ студента)

Керівник: к.т.н., доцент кафедри АІІТ
Володимир КОЦЮБИНСЬКИЙ
(науковий ступінь, вчене звання / посада, ПІБ керівника)

« 12 » грудня 2025 р.

Опонент: д.т.н., проф. каф. КСУ
Марія ЮХИМЧУК
(науковий ступінь, вчене звання / посада, ПІБ опонента)

« 12 » грудня 2025 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
(науковий ступінь, вчене звання)

« 12 » грудня 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти II-ий (магістерський)

Галузь знань – 17 – Електроніка, автоматизація та електронні комунікації

Спеціальність – 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка

Освітньо-професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ

Завідувач кафедри АІІТ

д.т.н., проф. Олег БІСІКАЛО

«26» вересня 2025 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Моргуну Юрію Олександровичу

(ПІБ автора повністю)

Тема роботи: Автоматизована клієнт-серверна система транспортних перевезень з функцією інтелектуального моніторингу стану водія.

Керівник роботи: к.т.н., доцент каф. АІІТ Коцюбинський В. Ю.

Затвердженні наказом ВНТУ від «24» вересня 2025 року № 313.

Строк подання роботи студентом: до «12» грудня 2025 року.

Вихідні дані до роботи: Розроблене серверне та клієнтське програмне забезпечення системи моніторингу транспортних перевезень та дій водія. Розмір моделі розпізнавання стану водія – не більше 120 мб; середній час обробки кадру – менше 80 мс; точність виявлення обличчя (IOU) – не менше 0.8.

Зміст текстової частини: Вступ; Аналіз предметної області, Технології розробки системи моніторингу транспортних перевезень; Розробка алгоритмічного та програмного забезпечення; Економічний розділ; Висновки; Список використаних джерел.

Перелік ілюстративного (або графічного) матеріалу: UML Deployment діаграма, UML Data Flow діаграма, UML діаграма послідовності взаємодії пристрою-клієнта та сервера, ER-модель бази даних, UML-діаграма діяльності методу shape_predictor_5, UML-діаграма діяльності методу shape_predictor_68, UML діаграма класів серверної частини.

6. Консультанти розділів роботи

Розділ змістової частини роботи	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Володимир КОЦЮБИНСЬКИЙ к.т.н., доцент каф. АІІТ	25.09.25	20.11.25
4	Володимир КОЗЛОВСЬКИЙ к.е.н., професор каф. ЕПтаВМ	10.11.25	01.12.25

7. Дата видачі завдання: «25» вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	25.09 – 05.10.2025	виконав
2	Підбір та обґрунтування технологій для реалізації	05.10 – 25.10.2025	виконав
3	Розробка архітектури системи та структури бази даних	25.10 – 10.11.2025	виконав
4	Реалізація та тестування програмного забезпечення	05.11 – 20.11.2025	виконав
5	Підготовка економічної частини	до 01.12.2025	виконав
6	Оформлення матеріалів до захисту МКР	20.11 – 02.12.2025	виконав
7	Попередній захист	до 02.12.2025	виконав
8	Захист роботи	до 19.12.2025	виконав

Здобувач _____ Юрій МОРГУН
(підпис) (прізвище та ініціали)

Керівник роботи _____ Володимир КОЦЮБИНСЬКИЙ
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК [004.75+004.932]:656.07

Моргун Ю.О. Автоматизована клієнт-серверна система транспортних перевезень з функцією інтелектуального моніторингу стану водія. Магістерська кваліфікаційна робота зі спеціальності 174 – Автоматизація, комп’ютерно-інтегровані технології та робототехніка. Вінниця: ВНТУ, 2025, 113 с.

Укр. мовою. Бібліогр.: 41 дж.; рис.: 33; табл. 8.

Метою роботи є оптимізація процесів управління логістикою за рахунок розробки автоматизованої системи для моніторингу транспортних перевезень.

У теоретично-методичній частині роботи було детально розглянуто та проаналізовано предметну область та об’єкт автоматизації системи моніторингу транспортних перевезень. Були досліджені особливості існуючих систем-аналогів. Обґрунтовано вибір технологій, що використовувались при розробці системи. У практичній частині розроблено архітектуру системи та проаналізовано шаблони проєктування, що були використані, розроблено алгоритмічне та програмне забезпечення. Результатом роботи є система, що забезпечує моніторинг транспортних перевезень та стану водія, обробку подій, що відбуваються під час перевезення, зручне зберігання та відображення цих даних.

Ключові слова: автоматизація процесів, транспортні перевезення, логістична система, клієнт-сервер, комп'ютерний зір, Qt, подійно-орієнтована архітектура.

ABSTRACT

Morhun Y. O. Automated client-server system for transport operations with intelligent driver status monitoring. Master's thesis in the field of 174 – Automation, Computer-Integrated Technologies, and Robotics. Vinnytsia: VNTU, 2025, 113 p.

In Ukrainian. Bibliography: 41 sources; figs.: 33; tables: 8.

The aim of the work is to optimize logistics management processes by developing an automated system for monitoring transport operations.

The theoretical and methodological part of the work provides a detailed review and analysis of the subject area and the object of automation of the transport monitoring system. The features of existing analog systems were studied. The choice of technologies used in the development of the system was justified. The practical part develops the system architecture, analyzes the design patterns used, and develops the software. The result of the work is a system that monitors transport operations and driver status, processes events that occur during transport, and conveniently stores and displays this data.

Keywords: process automation, transport, logistics system, client-server, computer vision, Qt, event-driven architecture.

ЗМІСТ

ВСТУП

Актуальність. Зростання рівня автоматизації промисловості є характерною рисою розвиненого суспільства. Підвищення рівня життя спонукає людей хотіти більш якісні товари та послуги. Це в свою чергу вимагає розширення існуючих та створенню нових маршрутів перевезень товарів. У такій ситуації застосування інформаційних технологій для моніторингу логістики є особливо актуальною задачею. Відсутність своєчасних даних по стану транспортних перевезень може стани серйозною перешкодою для промисловості, бізнесу та інших сфер життя країни.

Саме тому є необхідність у створенні систем, які будуть не тільки зберігати інформацію про транспортні перевезення, а й здійснювати обробку певних подій що відбуваються під час поїздки для майбутнього покращення процесу перевезень. У найближчі роки системи моніторингу транспортних перевезень будуть активно розвиватись, доповнюючись новим функціоналом. Усі компанії які тим чи іншим чином задіяні у логістиці з кожним роком все більше усвідомлюють перспективи користування цифровими технологіями для оптимізації своїх процесів.

Вирішення задачі автоматизації збирання даних про логістичні перевезення та стан водія дозволяє значно підвищити ефективність та оптимізувати процеси транспортування. Це знижує витрати на паливо та технічне обслуговування завдяки оптимізації маршрутів та своєчасному виявленню технічних проблем. Автоматизовані системи також підвищують безпеку водіїв і вантажів, надаючи можливість контролювати поведінку на дорозі та реагувати на позаштатні ситуації в режимі реального часу. Крім того, такі рішення забезпечують покращення якості обслуговування клієнтів, завдяки точному відстеженню замовлень та своєчасним доставкам. Таким чином

проблема розробки системи моніторингу транспортних перевезень, яка вирішується в даній магістерській кваліфікаційній роботі, є **актуальною**.

Метою даної роботи є покращення процесу управління логістикою та підвищення безпеки за рахунок оптимізації процесу збирання даних під час здійснення перевезень.

Для досягнення поставленої мети необхідно розв’язати наступні задачі:

- Розглянути принципи функціонування систем моніторингу логістики;
- Проаналізувати існуючі системи моніторингу транспортних перевезень;
- Визначити функції, які має реалізувати система моніторингу транспортних перевезень;
- Розробити архітектуру системи;
- Розробити програмне забезпечення серверної частини системи;
- Розробити програмне забезпечення клієнтської частини системи.

Об’єктом дослідження є процес взаємодії водія автомобіля з системою моніторингу транспортних перевезень з використанням мобільних пристроїв.

Предметом дослідження методи та засоби побудови архітектури та технологій реалізації системи моніторингу транспортних перевезень.

Методи дослідження. У процесі дослідження застосовувалися емпіричні методи для збору та оцінки характеристик процесів транспортних перевезень, методи аналізу та узагальнення для формування вимог до системи та структурування отриманих даних, алгоритмічні методи для розробки модулів обробки відеопотоку та серверної взаємодії, а також методи практичного моделювання для побудови та тестування клієнт-серверної архітектури та алгоритмів розпізнавання стану водія.

Науково-практичний результат роботи полягає у оптимізації класичних алгоритмів комп’ютерного зору для розпізнавання ключових точок обличчя у межах мобільного додатку водія, що забезпечує роботу в умовах обмежених обчислювальних ресурсів без використання серверної обробки чи нейронних мереж. Це сприяє зменшенню навантаження на систему з одночасним

збереженням точності виявлення, що дозволяє клієнтській частині працювати на недорогих Android-пристроях без суттєвого зниження продуктивності.

Практична цінність роботи полягає у створенні прототипа, що реалізує базові функції системи моніторингу транспортних перевезень. Таке рішення, в основному, орієнтоване на використання невеликими логістичними компаніями або стартапами, які не можуть собі дозволити більш дороговартісні рішення.

Апробація: Результати виконаня роботи було представлено на конференціях:

- LIII Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024);
- Контроль і управління в складних системах (КУСС-2024);
- LIV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2025).

Робота приймала участь у конкурсах:

- Міжнародний конкурс студентських наукових робіт, м. Кременчук, 2025; [1]
- Конкурс ідей для стартапів серед студентів «IT Idea Battle», м Вінниця, 2025. [2]

Публікації: за результатами роботи було опубліковано тези доповіді [3-5]

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність систем моніторингу транспортних перевезень

На сьогоднішній день велика кількість сфер людського життя тісно пов’язана з логістичними перевезеннями і залежить від них. Часто затримки в доставці бувають руйнівними для бізнесу, що призводить до втрати клієнтів та фінансових збитків. Своєчасна доставка є критично важливою для забезпечення безперебійної роботи ланцюгів постачання, особливо в галузях, де час є вирішальним фактором, таких як охорона здоров'я, харчова промисловість та електронна комерція.

Проблему оптимізації логістичних перевезень допомагають вирішувати системи моніторингу транспортних перевезень (Transportation monitoring system, далі TMS). Термін TMS означає сукупність додатків, які використовують інформаційно-комунікаційні технології в галузі транспорту з метою отримання економічних, соціальних та енергетичних вигод. TMS може бути застосована до багатьох видів транспорту включаючи: наземний транспорт (легкові автомобілі, вантажівки та громадський транспорт), повітряний транспорт (літаки), залізничний транспорт (потяги) та спеціалізований транспорт (будівельна техніка, сільськогосподарські машини тощо). Основною функцією таких систем є підвищення ефективності логістики, а також допомога, з одного боку, в управлінні інфраструктурою за допомогою її експлуатації та систем прийняття рішень, з іншого боку, користувачам, з метою отримання загального задоволення від роботи транспортної системи [6, 7].

Використання автоматизованих логістичних систем пропонує ряд переваг. В першу чергу мова йде про підвищення ефективності та оптимізацію ресурсів. Системи моніторингу дозволяють компаніям оптимізувати маршрути транспортування, що зменшує час у дорозі та витрати на паливо. Зачасту

оптимізація маршрутів включає в себе розумні алгоритми, які фіксують можливі перешкоди на дорозі (аварії, затори та ін.) та завчасно попереджають або навіть пропонують водіям скоригувати свій маршрут, щоб уникнути їх. [8]

Компаніям та невеликим підприємцям подібні системи спрощують процес пошуку водіїв для найму. Основою таких систем обов’язково є деяка система оцінок, що дозволяє переглянути минулі поїздки водія, ознайомитися з відгуками попередніх роботодавців, переконатись в охайності стилю водіння потенційного співробітника.

Ці дані надають роботодавцям можливість приймати більш обґрунтовані рішення при наймі, знижуючи ризики, пов'язані з некваліфікованими або ненадійними працівниками. Крім того, системи оцінок стимулюють водіїв дотримуватися високих стандартів роботи, оскільки їхні рейтинги безпосередньо впливають на їхню репутацію та можливості працевлаштування в майбутньому. Такий підхід сприяє підвищенню якості обслуговування та безпеки на дорогах, а також забезпечує прозорість і довіру між роботодавцями та працівниками.

Такі системи також значно допомагають у підвищенні безпеки на дорогах. Як свідчить статистика дослідження МВС України щодо основних причин ДТП [9] 60 – 70% аварій стаються через помилкові дії людини. Дослідники вважають, що 2/3 усіх пригод виникає з вини людей і лише 1/3 через фактори, які не залежать від їхньої волі та діяльності. Дослідивши 32713 випадків ДТП фахівці помітили що найбільша кількість з них (9777) – це відволікання водія від керування транспортним засобом.

До неуважності за кермом може відноситись відволікання на екран телефону або будь-який інший гаджет, розмови з пасажирами або телефонні розмови. Більшість сучасних систем моніторингу забезпечують функціонал, що в режимі реального часу контролює водія і повідомляє його якщо той відволікається занадто довго.

Більше того, дослідження МВС стверджує, що ще 10 – 15% ДТП стаються через технічні несправності автомобіля. Цю проблему також вирішують системи

моніторингу логістики, так як в транспортних засобах зачасту встановлюється чимала кількість датчиків для контролю стану автомобіля. Постійний моніторинг стану транспортних засобів дозволяє вчасно виявляти проблеми та проводити профілактичне обслуговування, що зменшує ризик дорогих поломок.

Можливість відстежувати транспортні засоби в реальному часі також дозволяє швидко реагувати на позаштатні ситуації, такі як крадіжки або викрадення транспортних засобів.

Системи моніторингу логістики допомагають також зменшити витрати та збільшити екологічність перевезень. Оптимізація маршрутів та зменшення простоїв допомагає знизити викиди парникових газів та зменшити витрати пального. Окрім цього датчики системи моніторингу встановлені на автомобілях допомагають відстежувати екологічні показники транспортних засобів та дотримуватися відповідних стандартів та норм. [10, 11]

Отже, системи моніторингу транспортних перевезень є корисним інструментом оптимізації логістики. Такі системи вже активно використовуються логістиці розвинених країн та грають важливу роль в їх економіці. Вони забезпечують високу ефективність і точність логістичних операцій, що сприяє зниженню витрат і підвищенню конкурентоспроможності бізнесу. Сучасні системи моніторингу транспортних перевезень є незамінним інструментом для розвитку транспортної інфраструктури і підтримки економічного зростання країни.

1.2 Аналіз об’єкта автоматизації

Коли йде мова про автоматизацію логістики – це означає впровадження сучасних технологій комп’ютерних систем для оптимізації процесів даної сфери діяльності.

Нагадаємо, що об'єктом автоматизації у даній роботі є процес взаємодії транспортних засобів з деяким центральним компонентом системи, що здійснює моніторинг над ними, а якщо точніше – це взаємодія пристроїв, що знаходяться всередині транспортних засобів, з серверною частиною системи моніторингу. Важливим у такій ситуації є підтримання з’єднання у реальному часі для миттєвого реагування на події, що стаються під час перевезень.

Таким чином можна описати приблизний алгоритм роботи такої системи:

- Клієнтські пристрої, що встановлені на транспортних засобах, повинні регулярно збирати та передавати дані про місцезнаходження, швидкість, напрямок руху та різноманітні події, які можуть виникнути під час руху. Ці пристрої оснащені GPS-модулями для точного визначення координат, а також можуть мати додаткові датчики для збору інформації про стан транспортного засобу (наприклад, паливний рівень, температура двигуна, стан дверей тощо).
- Серверна частина системи оброблятиме отримані дані, зберігатиме їх у базі даних та надаватиме можливість доступу до цієї інформації через деякий API для подальшого аналізу та візуалізації. Основними задачами серверної частини буде обробка вхідних повідомлень, обробка подій, зберігання даних у базі даних та забезпечення зручного доступу до цієї інформації для кінцевих користувачів (менеджерів транспортного флоту) [12].

Враховуючи все це можна сформувати основні кроки для створення автоматизованої системи моніторингу логістики. Дані кроки зображені на рисунку 1.1.

Рисунок 1.1 – Основні етапи автоматизації моніторингу логістики

1.3 Порівняльна характеристика існуючих систем моніторингу транспортних перевезень

Ключовим етапом у розробці системи моніторингу транспортних перевезень є розгляд конкретних існуючих систем. На сьогоднішній день існує ряд компаній, які надають різноманітні послуги в сфері логістики та займаються розробкою систем для моніторингу транспортних перевезень. Деякі з них: Geotab (система MyGeotab), Samsara (Samsara Fleet Management) та Rosco Vision Systems з їх системою RoscoLive.

1.3.1 Geotab

Geotab – це одна з провідних компаній у сфері управління автопаркам, що надає рішення для моніторингу транспортних засобів та управління логістикою. Заснована у 2000 році, Geotab має штаб-квартиру в Канаді і обслуговує клієнтів більш ніж 50000 клієнтів в 160 країнах по всьому світу. Компанія відома своїми інноваційними рішеннями для відстеження транспортних засобів у реальному часі, збору даних та аналітики [13].

Для своїх клієнтів вони пропонують систему моніторингу яку легко налаштовувати та легко розширювати до необхідної кількості транспортних засобів по мірі того, як росте бізнес клієнта. За необхідності вони забезпечують інтеграцію з уже існуючими системами або рішеннями сторонніх розробників. Geotab пропонує рішення для відстеження місцезнаходження транспортних засобів, моніторингу поведінки водія, аналізу витрати пального та багато іншого. Платформа Geotab допомагає оптимізувати використання автопарку, забезпечувати безпеку, знижувати витрати на технічне обслуговування та покращувати ефективність операцій. Вони також надають потужні інструменти

для аналізу даних, зібраних з транспортних засобів, що допомагає компаніям приймати планувати свої дії та приймати обґрунтовані рішення.

Серед особливостей функціоналу системи Geotab можна виділити:

- Пристрої Geotab GO: Компактні пристрої, які підключаються до порту OBD-II транспортного засобу. Вони збирають дані про місцезнаходження, швидкість, витрату пального, діагностику двигуна та іншу інформацію. Вигляд пристрою Geotab GO зображено на рис. 1.2.

Рисунок 1.2 – Пристрій Geotab Go

- Geotab Drive: Мобільний додаток для водіїв, що забезпечує функції електронного ведення журналу водія, моніторинг відповідності регламентам, перевірки перед виїздом та після повернення на базу, а також реєстрацію робочого часу. Вигляд інтерфейсу мобільного додатку наведено на рис. 1.3.

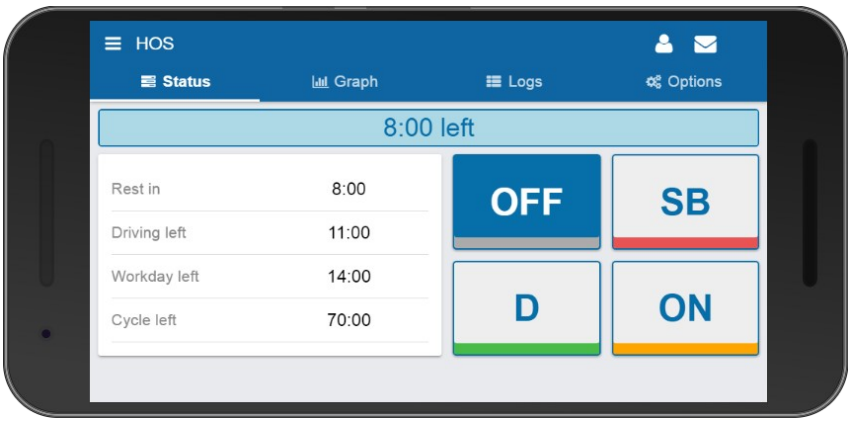

Рисунок 1.3 – Мобільний додаток Geotab Drive

- MyGeotab – це платформа, що дозволяє створювати детальні звіти та аналізувати дані про транспортні засоби та водіїв. Приклад таких данних зображено на рис. 1.4.

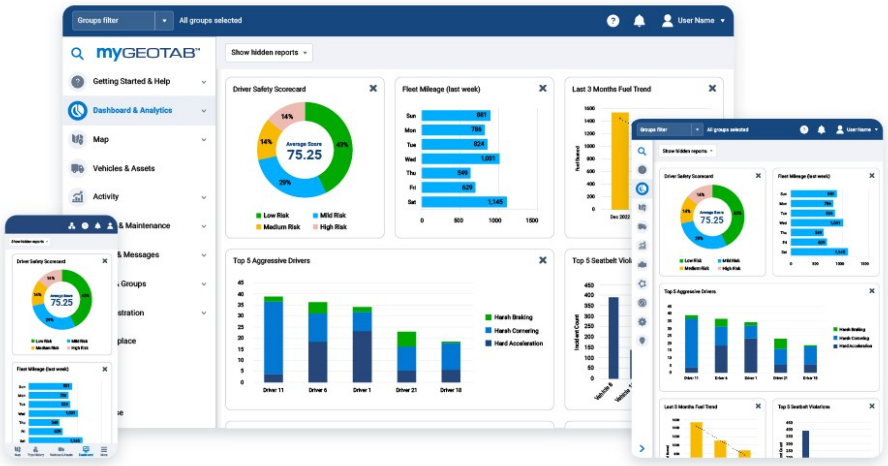

Рисунок 1.4 – Вигляд інтерфейсу користувача платформи MyGeotab з усією необхідною телиметрією

1.3.2 Samsara

Samsara – це ще одна з провідних компаній, яка надає комплексні рішення для управління автопарками, моніторингу та оптимізації бізнес-процесів. Їх відмінною особливістю є активне впровадження Інтернету речей (IoT) у сферу

логістичних перевезень. Samsara також відома своїм інноваційним підходом до збирання і аналізу даних, що дозволяє підвищувати ефективність та безпеку транспортних операцій. Компанія була заснована у 2015 році та має штаб-квартиру у Сан-Франциско, Каліфорнія [14].

Серед особливостей функціоналу, що надає ця компанія свої клієнтам можна виокремити:

- Моніторинг поведінки водіїв, що включає контроль за швидкістю, різким гальмуванням, прискоренням та іншими аспектами поведінки водіїв, що допомагає підвищувати безпеку та знижувати витрати на паливо.
- Електронні журнали водіїв (ELD) які забезпечують відповідність вимогам щодо ведення журналів робочого часу водіїв та дотримання регламентів.
- Відстеження температури та умов зберігання товарів у реальному часі, що особливо важливо для продуктів харчування, ліків та інших чутливих до температури вантажів.
- Відстеження автопарку в реальному часі дозволяє фіксувати місцезнаходження кожного транспортного засобу в режимі реального часу, що забезпечує краще управління маршрутом і своєчасне реагування на будь-які відхилення від плану.
- Контроль за швидкістю, різким гальмуванням, прискоренням та іншими аспектами поведінки водіїв. Це допомагає не лише підвищувати безпеку на дорогах, але й знижувати витрати на паливо, зменшуючи кількість небезпечних маневрів. Система може автоматично генерувати звіти і сповіщення про небезпечну поведінку водія.
- Платформа надає потужні інструменти для аналізу ефективності автопарку. Користувачі можуть генерувати звіти за різними показниками, такими як витрати на паливо, час простою, продуктивність водіїв і багато іншого. Це дозволяє приймати обґрунтовані рішення на основі реальних даних.

Для зручного управління транспортним флотом компанія пропонує веб-застосунок Samsara Fleet Management. Це платформа на якій можна зручно

переглядати весь вищеописаний функціонал, що система Samsara пропонує. Основним вікном такого застосунку є мапа на якій у реальному часі відображається рух маркерів транспортних засобів. Вигляд користувацького інтерфейсу Samsara Fleet Management зображено на рис. 1.5.

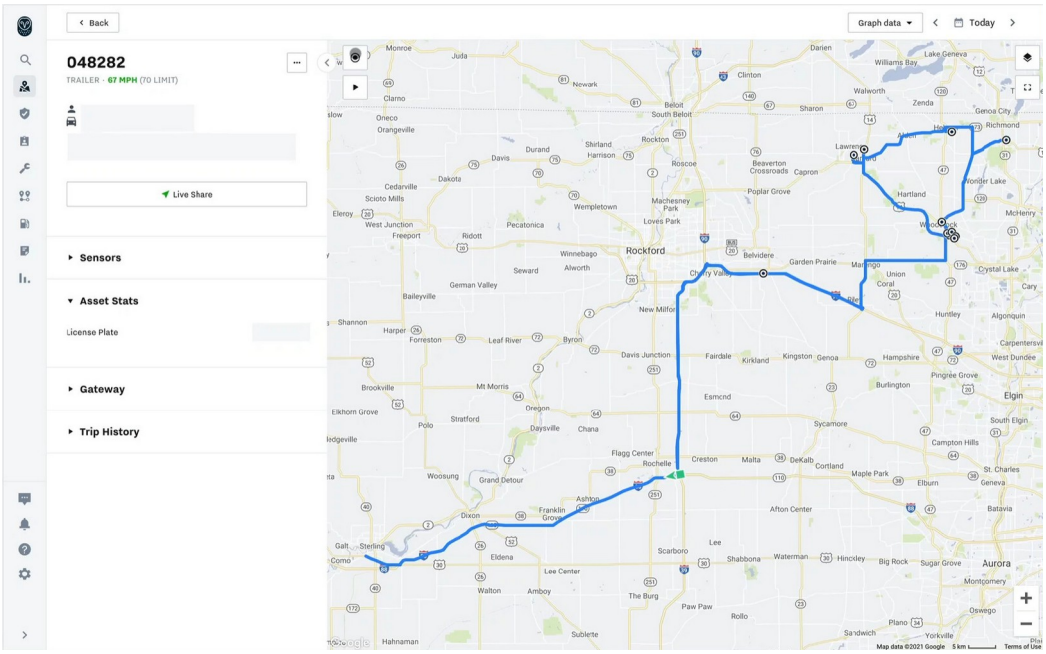

Рисунок 1.5 – Відображення мапи у системі Samsara Fleet Management

1.3.3 Rosco Vision Systems

RoscoLive – це система моніторингу та управління транспортними засобами, що належить компанії Rosco Vision Systems. Вона спеціалізується на наданні рішень для покращення безпеки водіїв та ефективності управління автопарками. Rosco Vision Systems має багаторічний досвід у розробці та виробництві різноманітних систем безпеки для транспортних засобів, включаючи камери заднього виду, системи відеомоніторингу та інші технології [15].

Окрім класичної системи моніторингу транспортних засобів Rosco Vision також пропонує рішення для управління публічним транспортом (наприклад шкільними автобусами), будівельними машинами та багато іншого.

Компанія спеціалізується не тільки на розробці програмного забезпечення а й має широкий асортимент апаратного забезпечення для будь яких потреб: відеорегістратори, сенсори та датчики.

Особливо виділяється серед їх продукції відеореєстратор DV6 HD. Він є передовим інструментом управління автопарком, призначеним для підвищення безпеки, ефективності та моніторингу продуктивності. Цей пристрій на основі штучного інтелекту оснащений двома камерами, які записують відео у форматі Full HD як водія, так і дороги, з можливістю безперервного запису та зйомки окремих кліпів певних подій. Відеореєстратор DV6 HD зображено на рис. 1.6

Рисунок 1.6 – Відеореєстратор DV6 HD

До основних функцій даного пристрою можна віднести:

- Сповіщення та моніторинг у реальному часі: DV6 генерує сповіщення для водіїв у кабіні при виявленні ризикованої поведінки, такої як сонливість, використання телефону або неуважність. Це допомагає підвищити безпеку водія та зменшити ймовірність аварій.

- Пропонуються як локальне, так і хмарне сховище для зберігання даних, що дозволяє гнучко керувати даними. Доступ до записаних відео можна отримати в будь-який час через хмарну платформу RoscoLive.
- DV6 легко інтегрується з іншими системами управління автопарком, такими, як вищеописана Geotab, що дозволяє синхронізувати відеодані з іншими телематичними даними і даними GPS. Ця інтеграція дозволяє менеджерам автопарку переглядати відеокліпи, пов'язані з певними подіями або винятками, покращуючи загальний процес моніторингу.

1.4 Висновки до розділу

У даному розділі було розглянуто сучасний стан систем моніторингу логістичних перевезень та основні існуючі системи моніторингу транспортних перевезень. Ними є системи MyGeotab, Samsara Fleet Management та RoscoLive. Дані системи є лідерами в своїй сфері та можуть запропонувати багатий функціонал для оптимізації процесів перевезень.

Аналіз систем аналогів дозволив прослідкувати схожий базовий функціонал у багатьох розробників. На основі цього можна сформувати вимоги до необхідного функціоналу системи, що розробляється:

- система авторизації в системі та ролі користувачів (адміністратор, менеджер, водій);
- можливість відстеження кожного транспортного засобу в реальному часі під час виконання перевезення;
- можливість отримувати, обробляти та зберігати дані про пересування транспортного засобу та події, що відбуваються під час поїздки;
- система оцінки водія формується з інформації про події, що відбувались під час виконання водієм транспортних перевезень.

2 ТЕХНОЛОГІЇ РОЗРОБКИ СИСТЕМИ МОНІТОРИНГУ ТРАНСПОРТНИХ ПЕРЕВЕЗЕНЬ

2.1 Вибір технологій для реалізації серверної частини системи та додатку пристрою-клієнта

Після проведеного аналізу існуючих систем-аналогів стає зрозуміло, що основою будь-якої системи моніторингу логістики є обмін інформацією між серверною частиною системи та пристроєм, який знаходиться в транспортному засобі, збирає та обробляє перед відправкою необхідну інформацію. Варто зауважити, що логіка серверної частини та додаток пристрою-клієнта будуть виконуватись на різних операційних системах (Linux та Android відповідно), тому при виборі інструментів для розробки кросплатформеність стане ключовим аспектом на який варто звертати увагу. Саме кросплатформеність дозволяє легко підтримувати два різних по своїй суті додатки однією командою розробників.

Як основа для реалізації системи моніторингу транспортних перевезень ідеально підходить мова програмування C++ та фреймворк Qt. Мова C++ добре підходить для написання подібних систем, де продуктивність програми стоїть на першому місці і нею не можна нехтувати, адже кількість транспортних засобів моніторинг яких здійснюється одночасно може сягати десятків або навіть сотень. Фреймворк Qt в свою чергу дозволяє створювати на C++ кросплатформені застосунки, що компілюються як для Linux так і для Android. Функціонал Qt надає ідеальні методи для вирішення задач, які постануть при розробці системи. До таких методів відносяться:

- можливість легко створювати компоненти графічного інтерфейсу для будь-якої операційної системи (QtGui)
- інтуїтивно зрозуміла робота з базами даних мовою структурованих запитів SQL (QtSql)

- набір класів для мережевого програмування (QtNetwork)

2.1.1 Аналіз та обґрунтування використання мови C++

C++ – мова програмування загального призначення з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої, процедурної та ін. Розроблена в 1979 році Б'ярном Страуструпом вона досі залишається однією з найуживаніших мов програмування загального призначення. Мову використовують для системного програмування, розробки прикладного програмного забезпечення, написання потужних серверних та клієнтських програм, драйверів, а також для розробки відеоігор. C++ суттєво вплинула на інші популярні сьогодні мови програмування: C# та Java [16].

При виборі цієї мови для розробки свого програмного забезпечення є фактори які варто враховувати. C++ – це потужна мова програмування, яка може бути складною для початківців. При написанні коду C++ саме на програмісті лежить уся відповідальність за управління пам’яттю. На відміну від більш високорівневих мов, як Python чи C# тут відсутній алгоритм автоматичного керування пам’яттю (так званий Garbage collector). В такій ситуації існує ризик втрати пам’яті (memory leaks), що рано чи пізно призведе зниження продуктивності системи або аварійного завершення програми.

Ці незручності C++ повністю компенсує набагато вищою швидкодією в порівнянні з більш високорівневими мовами програмування. C++ є компільованою мовою і це має свої переваги. Програма, яка скомпільована в машинний код, під час виконання перевершує швидкість інтерпретованої програми в десятки або навіть сотні разів. Компілятор також допомагає виявити велику кількість помилок ще на етапі компіляції в той час як недоліки інтерпретованої програми будуть виявлені тільки після пробного запуску. До того ж ця мова забезпечує багатопотокове програмування, що є однією з

найважливіших технік у C++ розробці. Багатопоточність дозволяє збільшити продуктивність програми та використовувати всі можливості багатоядерних процесорів.

2.1.2 Аналіз та обґрунтування використання фреймворку Qt

Екосистема C++ багата на різноманітні бібліотеки та фреймворки. Вона пропонує рішення для практично будь-якої області програмування, від вбудованих систем до великих програмних проєктів. Бібліотеки, такі як Boost, STL, та Eigen, забезпечують потужні інструменти для розв'язання складних завдань у математиці, обробці даних, графічному програмуванні та багатьох інших сферах. Фреймворки, такі як Qt і wxWidgets, дозволяють створювати кросплатформенні додатки з графічним інтерфейсом, що забезпечує швидку і зручну розробку програми. Така велика різноманітність бібліотек та фреймворків й обумовлює високу популярність мови C++.

Серед цього різноманіття особливо виділяється фреймворк Qt. Цей фреймворк є потужним інструментом для кросплатформеної розробки, що дозволяє створювати програми, які працюють на різних операційних системах, включаючи Windows, macOS, Linux та Android. Qt має широкий набір компонентів та інструментів, які полегшують створення графічного інтерфейсу та взаємодії з користувачем.

Qt використовується в Autodesk Maya, Skype, Медіапрогравач VLC, VirtualBox, Mathematica, на European Space Agency, DreamWorks, Google, HP, Lucasfilm, Panasonic, Philips, Samsung, Siemens, Volvo і Walt Disney Animation Studios та Google Планета Земля [17].

Крім того, на Qt засновано середовище робочого столу KDE, графічний інтерфейс мобільної ОС MeeGo і Qt Creator - середовище розробки на Qt;

Є ряд аргументів на користь вибору даного фреймворку, як основного інструменту розробки системи моніторингу транспортних перевезень. В першу чергу це зручна система спілкування між окремими об’єктами. Коли у 1979 році Б'ярн Страуструп розроблював мову C++ він вже бачив необхідність представлення складних для реалізації речей у формі об’єктів, тобто класів. Не дарма перша версія мови називалась “Сі з класами”.

У класів були поля (атрибути) та методи. Функціонал кожного класу також мав особливу позначку (специфікатор доступу), що визначав його роль як частину класу.

Ці специфікатори могли бути:

- public – загальнодоступним функціоналом для користування як всередині класу так і ззовні
- private – функціонал класу, що був призначаний тільки для внутрішнього користування всередині класу
- protected – що був приватним специфікатором який також дозволяв передавати функціонал при наслідуванні [18]

Усе ж таки не вистачало ще одного специфікатора для функціоналу класу, а саме – можливості спілкування з іншими класами або компонентами програми. Зачасту це призводило до того, що кожен розробник за необхідності для свого застосунку створював свою реалізацію такого функціоналу.

Вирішити цю проблему спробували розробники фреймворку Qt перша версія якого була випущена 20 травня 1995 року. Розробники Qt представили декілька принципів свого фреймворку, які чудово доповнили базовий функціонал мови C++ і які до сьогодні є візитною карткою даного фреймворку.

1. Система сигналів та слотів. При програмуванні складної взаємодії між двома об’єктами може виникнути ситуація коли необхідно, щоб перший об’єкт при зміні свого стану повідомив інший об’єкт про цю подію. Загалом, ми хочемо, щоб об'єкти будь-якого типу могли взаємодіяти один з одним. Наприклад, якщо користувач натискає кнопку «Закрити», ми, ймовірно, хочемо, щоб була

викликана функція close() вікна. Не самим елегантним але швидким та відносні дієвим рішенням в такій ситуації буде використання зворотного виклику.

Зворотний виклик – це вказівник на функцію, тому якщо потрібно, щоб функція обробки сповістила інший компонент програми про якусь подію, необхідно передати вказівник на іншу функцію (зворотний виклик) до функції обробки. Функція обробки викликає функцію зворотного виклику, коли умова виклику стає істинною. Хоча існують успішні фреймворки, що використовують цей метод, зворотні виклики можуть бути інтуїтивно незрозумілими і можуть страждати від проблем із забезпеченням коректності типів аргументів зворотного виклику.

Qt пропонує альтернативу техніці зворотного виклику, а саме – сигнали та слоти. Сигнал випускається, коли відбувається певна подія (наприклад коли користувач натиснув кнопку «Закрити» згенерується сигнал windowClosed()). Слот – це функція, що підключається до конкретного сигналу та викликається у відповідь на нього (наприклад при спрацьовуванні сигналу windowClosed() відпрацює слот close(), що обробить всю необхідну логіку для закриття вікна програми). Принцип роботи механізму сигналів та слотів у Qt зображено на рис. 2.1. [19]

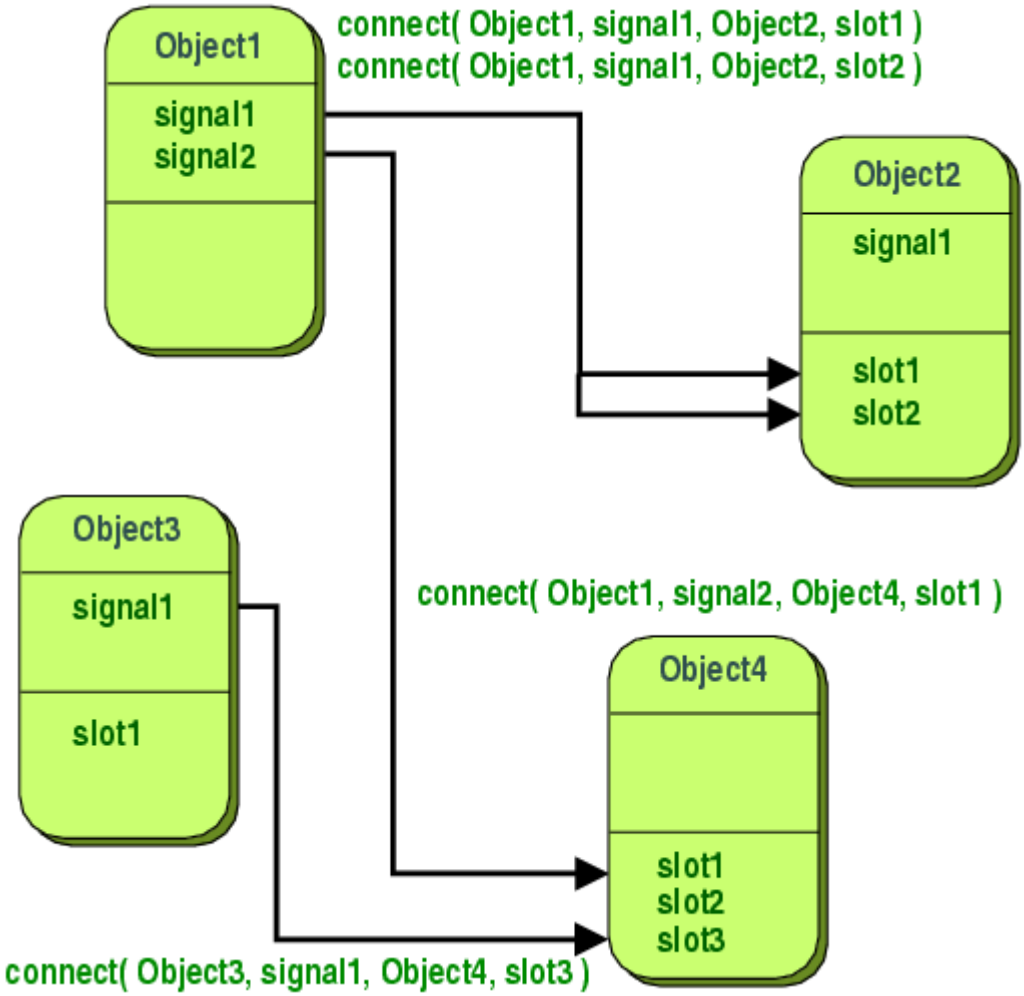

Рисунок 2.1 – Принцип роботи механізму сигналів та слотів у Qt

2. Метаоб’єктний компілятор надає підтримку додаткових можливостей у мові C++, необхідну для додаткової обробки коду при компіляції, надаючи клас QObject з деяким функціоналом, як базовий. Фактично переважна більшість класів у Qt є спадкоємцями цього класу і реалізують цю систему. Саме це дозволяє реалізувати у системі сигналів та слотів взаємодію об’єктів будь-якого типу, адже вказівник на базовий клас завжди буде типу QObject. Усім об’єктам типу QObject, що мають макрос Q_OBJECT, під час компіляції програми генерується додатковий код, що дає низку корисного функціоналу:

- Додається метаоб'єктна інформація. Вона містить інформацію про успадкування класів, що дає змогу визначати, чи є класи безпосередніми спадкоємцями, а також дізнатися ім'я класу. Надається й інформація про

мета-методи класу (сигнали, слоти та інші функції, що викликаються, з макросом Q_INVOKABLE).

- Динамічного приведення типів за допомогою qobject_cast() для класів QObject. Функція qobject_cast () поводиться аналогічно до стандартної C++ dynamic_cast() з тими перевагами, що вона не вимагає підтримки RTTI і працює через межі динамічних бібліотек. Функція намагається привести свій аргумент до типу покажчика, зазначеного в кутових дужках, повертаючи ненульовий покажчик, якщо об'єкт має правильний тип (визначений під час виконання), або nullptr , якщо тип об'єкта несумісний.
- Таймера, і відповідно цикл обробки подій. Додатково є механізм фільтрації, для перехоплення даних подій.
- Можливість інтернаціоналізації додатка за допомогою методів QObject::tr() і QObject::trUtf8().
- Підтримку сигналів і слотів, які використовуються для комунікації між об'єктами.
- Можливість отримання інформації про об'єкти Qt, не з C++ коду, а наприклад з QML або Qt Script [20].

3. Окрім C++ Qt може запропонувати широкий вибір мов програмування. C++ API фреймворку надає можливості для легкої кросплатформної розробки. Є можливість проектувати інтерфейси користувача за допомогою Qt QML – декларативної мови з можливістю описувати бізнес-логіку за допомогою JavaScript. Якщо програміст віддає перевагу Python або іншим більш високорівневим мовам програмування, присутній широкий вибір мовних прив'язок. Фреймворк, окрім C++ доступний, для Python, Java, Rust, Javascript [21].

4. Фреймворк пропонує багату екосистему додаткових інструментів. Важливою частиною цієї екосистеми є власне середовище розробки Qt Creator – інтегроване середовище розробки, призначене для створення кросплатформових застосунків

з використанням бібліотеки Qt. Вигляд середовища розробки Qt Creator зображено на рис. 2.2. [22]

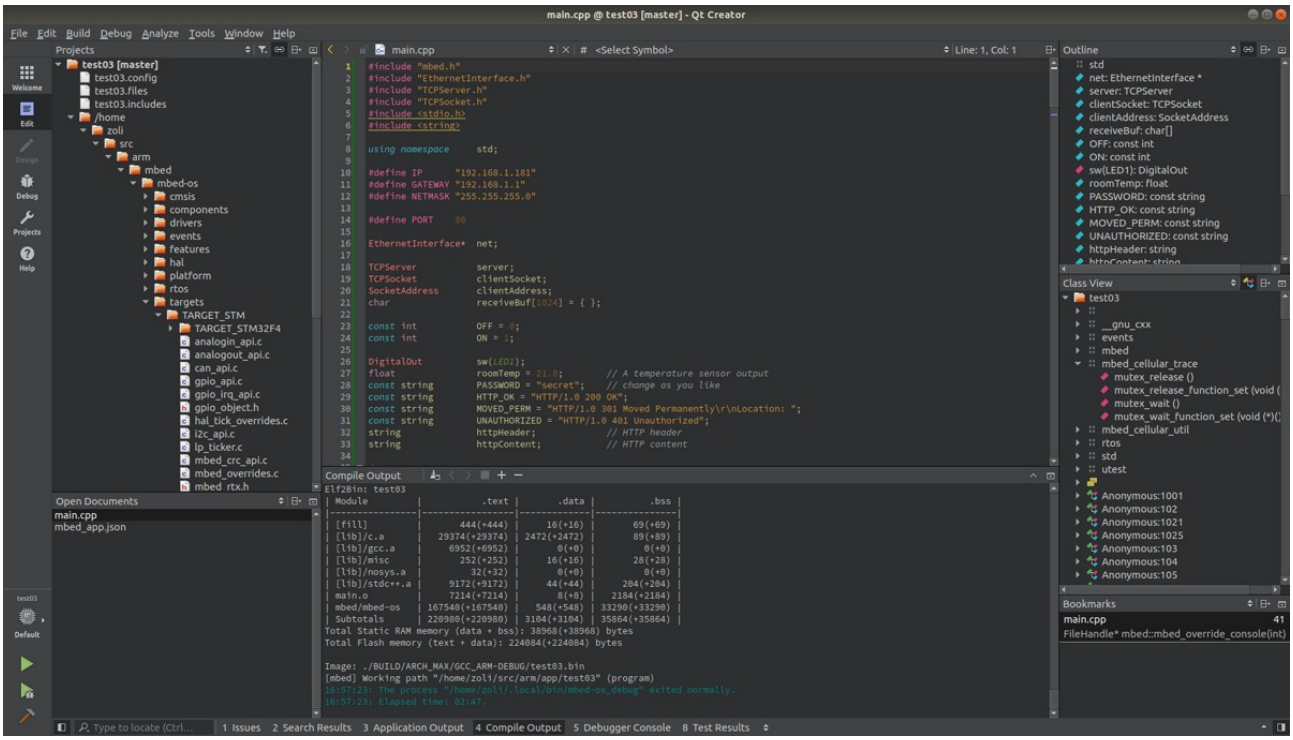

Рисунок 2.2 – Головне вікно середовища Qt Creator

5. Qt також пропонує власну систему збірки програми qmake, яка є зручною і потужною для проектів, розроблених виключно на Qt. Проте, для більш комплексних проектів, що включають різноманітні компоненти та бібліотеки, таких як система моніторингу транспортних перевезень, доцільно використовувати CMake, як основну систему збірки. CMake частіше рекомендується для використання, так як qmake вважається застарілою і використовується в основному для підтримки legacy проєктів, що були створені на більш старих версіях Qt [23].

2.2 Вибір технологій збереження та обробки даних

У комплексних комп’ютерних системах до яких відноситься система моніторингу транспортних перевезень важливою складовою є технологія збереження даних. Вдалий вибір системи керування базами даних особливо важливий для систем, що працюють у реальному часі, де своєчасне отримання та обробка даних може впливати на ефективність та загальну продуктивність усієї системи.

Для системи моніторингу транспортних перевезень база даних повинна зберігати різноманітні дані, необхідні для відстеження поїздок: маршрути (набори точок які відповідають широті та довготі на мапі), події (тип, текстовий опис, файли), що трапляються під час перевезень, інформацію про водіїв (ПІБ, особисті дані, оцінку як співробітника), інформацію про транспортні засоби (марка, модель, номерний знак). Вдалим вибором в такій ситуації буде СКБД PostgreSQL [24].

2.2.1 Обґрунтування вибору СКБД PostgreSQL

PostgreSQL – це об'єктно-реляційна система керування базами даних з відкритим вихідним кодом. Вона відома своєю надійністю, функціональністю та високою продуктивністю. Розробка PostgreSQL почалася в 1986 році в Каліфорнійському університеті в Берклі, і з тих пір вона еволюціонувала в одну з найпотужніших і найбільш використовуваних баз даних у світі. PostgreSQL підтримує стандарт SQL і має численні розширення, які дозволяють вирішувати складні завдання [25].

Серед особливостей PostgreSQL можна виокремити:

- Відкритий вихідний код: PostgreSQL є безкоштовною і відкритою системою, що дозволяє компаніям заощаджувати на ліцензіях і одночасно мати доступ до вихідного коду для модифікацій та покращень.
- Підтримка стандарту SQL: PostgreSQL підтримує більшість стандартних SQL функцій, включаючи транзакції, підзапити, зовнішні ключі, тригери та перегляди.
- Масштабованість і продуктивність: Система PostgreSQL здатна обробляти великі обсяги даних і високі навантаження, що робить її придатною застосувань у подібних системах.
- Розширюваність: PostgreSQL має модульну архітектуру, яка дозволяє додавати нові типи даних, індекси, функції, оператори і процедури, що розширює її функціональні можливості.
- Підтримка геопросторових даних: Завдяки розширенню PostGIS, PostgreSQL підтримує зберігання і обробку геопросторових даних, що є особливо корисним для додатків, які працюють з картами і геолокацією.
- Безпека: PostgreSQL забезпечує високий рівень безпеки даних, включаючи аутентифікацію, авторизацію, шифрування і аудит.
- Широка спільнота та підтримка: Завдяки великій і активній спільноті користувачів та розробників, PostgreSQL постійно вдосконалюється і отримує регулярні оновлення [26].

2.3 Вибір технологій для front-end частини системи

Для розробки клієнтської частини системи моніторингу транспортних перевезень важливим етапом є вибір оптимальних технологій для реалізації інтерфейсу менеджера транспортного флоту. Від правильності цього вибору залежить зручність використання системи, швидкість її роботи, можливість подальшого розширення функціоналу та підтримки. Для створення інтуїтивного,

динамічного та надійного веб-інтерфейсу було обрано низку технологій, що забезпечують ефективну роботу з даними, інтерактивними елементами та інтеграцію з картографічними сервісами. У наступних підпунктах детально розглядаються застосовані інструменти, зокрема JavaScript, бібліотека jQuery та картографічна бібліотека Leaflet.

2.3.1 JavaScript для динамічного відображення контенту

Для зручного та зрозумілого динамічного відображення даних з API серверної частини необхідно створити веб-застосунок. Для розробки такого застосунку є потреба обрати відповідний стек технологій. Основою для динамічного відображення контенту та маніпуляцій з HTML-сторінками було обрано мову JavaScript.

JavaScript – це мова сценаріїв, яка вставляється безпосередньо в HTML сторінки. Це єдина мова програмування такого типу, яка може бути зрозуміла веб-браузерам. Браузери можуть читати Javascript, інтерпретувати його а потім запускати програму, створюючи унікальний інтерактивний клієнтський досвід.

Ця мова досягла такого статусу, тому що є відкритою та стандартизованою. Вона добре підходить для Інтернету завдяки своїй динамічній природі та тісній інтеграції з DOM.

JavaScript також сумісний з іншими мовами. Це дуже важливо, оскільки веб-сервери працюють на різних мовах, будь то PHP, Python, Ruby, Java або .NET. Оскільки JavaScript, що виконується у браузері, на 100% відокремлений від того, як генеруються HTML-сторінки, користувачі завжди матимуть той самий багатий досвід, що й при роботі з JS, незалежно від мови, яка використовується на сервері [27].

2.3.2 Використання бібліотеки jQuery для пришвидшення розробки front-end частини

jQuery – це легка та добре оптимізована JavaScript бібліотека, що діє за принципом «write less, do more». Основна мета jQuery – значно спростити використання JavaScript, пришвидшити написання коду. Функціонал jQuery може запропонувати рішення для багатьох поширених проте складних у вирішенні завдань, для виконання яких потрібно багато рядків коду JavaScript. Ці складні завдання jQuery обгортає у методи, які можна викликати одним рядком коду. Є ряд переваг який jQuery може запропонувати:

- Легка маніпуляція DOM. jQuery може легко маніпулювати об'єктною моделлю документа. DOM - це програмний дизайн для документів HTML та XML. Вона представляє структуру документа і дозволяє програмам маніпулювати його формою, стилем і змістом. jQuery надає простий API для різних маніпуляцій з DOM, таких як зміна діапазону відносного розташування елементів, модифікація атрибутів, зміна CSS тощо. Селектори jQuery - це потужні інструменти, які дозволяють розробникам вибирати та маніпулювати елементами HTML. Вони засновані на тих самих селекторах, що використовуються в CSS, але в jQuery є можливість використовувати ці селектори для виконання різноманітних завдань над вибраними елементами. За допомогою методів маніпулювання DOM jQuery можна динамічно змінювати вміст веб-сторінки.
- Ще однією потужною особливістю jQuery є його проста та ефективна система обробки подій. У JavaScript події можуть бути будь-якими: від натискання користувачем кнопки до того, що веб-сторінка не завантажується, або завершення анімації. jQuery пропонує чудовий спосіб перехоплювати різні події без необхідності перевантажувати ваш HTML-код обробниками подій. Він надає такі методи, як click(), hover(), keypress(),

blur() та інші, які абстрагують процес приєднання слухачів подій до елементів.

- Підтримка AJAX. Асинхронний JavaScript і XML - це методи веб-розробки для створення асинхронних веб-додатків. За допомогою AJAX, наприклад, можна обмінюватися даними з сервером і оновлювати розділи сайту без перезавантаження сайту або всієї сторінки. Кілька методів функціональності AJAX дозволяють завантажувати дані з сервера і безперешкодно розміщувати повернуті дані у вибраних елементах веб-сайту або веб-сторінки. Це робить роботу над веб-сайтами з AJAX набагато менш складною і зводить до мінімуму кількість коду, який потрібно написати. Приклад надсилання асинхронного запиту з бібліотекою jQuery у порівнянні зі звичайним JavaScript зображено на рис. 2.3. [28, 29]

Рисунок 2.3 – Надсилання асинхронного запиту у Jquery та JavaScript

2.3.3 Вибір бібліотеки для відображення мап на HTML-сторінках

Кожна з розглянутих систем аналогів має можливість відображення інформації про транспортний засіб, над яким здійснюється моніторин у веб-браузері. У такій ситуації постає проблема відображення мапи безпосередньо на HTML-сторінці. Екосистема JavaScript може запропонувати ряд готових бібліотек для вирішення цієї проблеми. Одна з таких бібліотек – Leaflet.

Leaflet має невеликий розмір та високу продуктивність, що робить її ідеальною для інтерактивних веб-додатків, де швидкість завантаження і відгуку має велике значення. У випадку з моніторингом транспортних засобів, швидке оновлення даних на карті є критичним, оскільки користувачі системи повинні мати доступ до актуальної інформації в реальному часі.

Бібліотека Leaflet надає різноманітні функції для роботи з картами, включаючи можливості додавання маркерів, ліній, полігонів та інших об'єктів на карту. Це дозволяє не лише відображати поточні місцезнаходження транспортних засобів, але й візуалізувати їхні маршрути, зупинки та інші важливі дані [30]. Приклад відображення мапи Leaflet у браузері зображено на рис. 2.4.

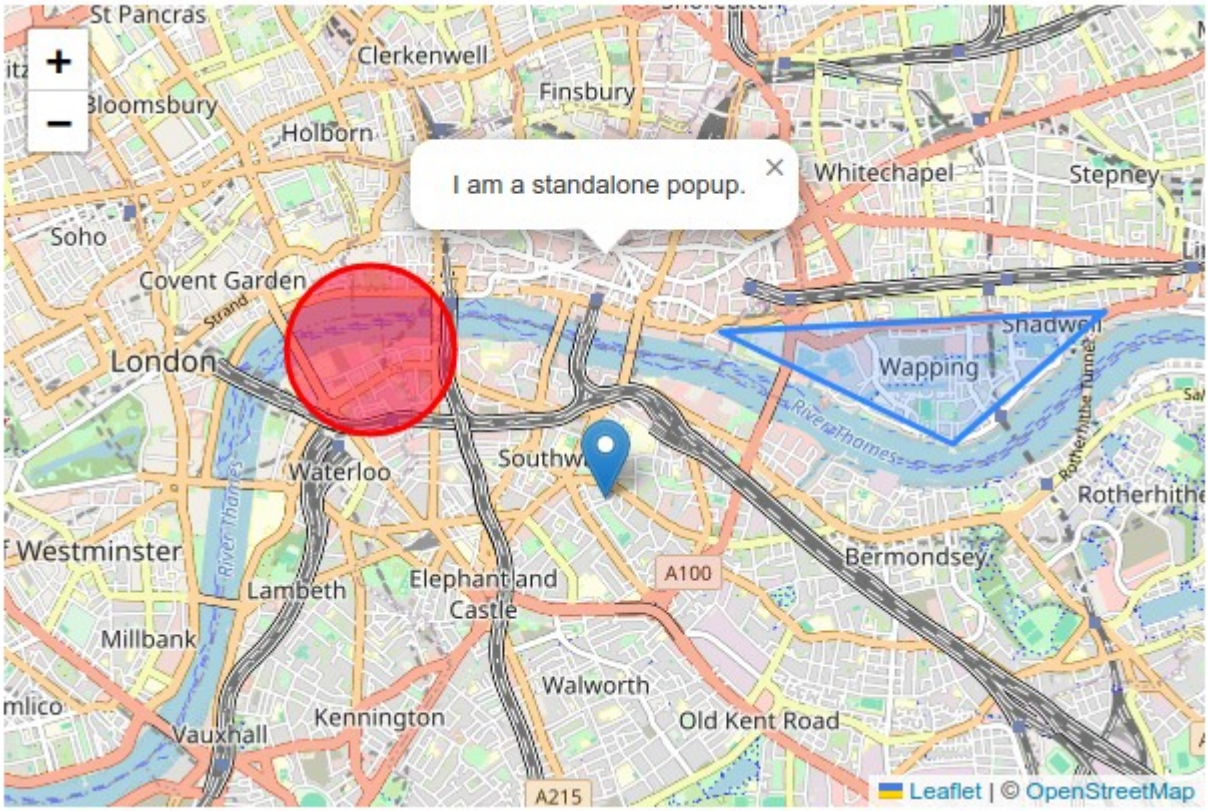

Рисунок 2.4 – Вигляд мапи Leaflet у браузері

2.3.4 Веб-сервер Nginx для хостингу сторінок front-end частини

Сторінки front-end частини системи доцільно розмістити на окремому сервері, що дозволить правильно розподілювати навантаження на систему при великій кількості користувачів. Постає питання вибору вебсерверу для хостингу front-end частини. Вдалим вибором буде використання вебсерверу Nginx який є одним із найпопулярніших веб-серверів у світі, відомим своєю високою продуктивністю, гнучкістю та легкістю налаштування.

Nginx забезпечує ефективну обробку статичних файлів, що дозволяє швидко доставляти веб-сторінки користувачам. Крім того, він підтримує кешування статичних ресурсів, що ще більше знижує час завантаження сторінок та зменшує навантаження на сервер. Це особливо важливо для front-end додатків, які повинні забезпечувати швидкий і зручний доступ до інформації.

Даний веб-сервер також пропонує розширені можливості для забезпечення безпеки веб-додатків. Він підтримує SSL/TLS для шифрування трафіку, захист від атак типу DDoS, налаштування правил доступу та автентифікації користувачів. Використання Nginx для хостингу front-end частини системи допоможе забезпечити захист даних та безпеку користувачів. [31]

2.4 Висновки до розділу

Після детального огляду сучасних технологій для розробки прикладного програмного забезпечення вдалось сформувати стек технологій, компоненти якого гармонійно доповнюють один одного та забезпечують хороший функціонал для розробки системи моніторингу транспортних перевезень.

Мова програмування C++ була обрана завдяки своїм численним перевагам. C++ забезпечує високу продуктивність та ефективність використання ресурсів, що є критичним для систем реального часу, таких як система моніторингу

транспортних перевезень. Мова C++ підтримує об’єктно-орієнтоване програмування, що дозволяє створювати гнучкі та масштабовані архітектури програмного забезпечення. Крім того, C++ має потужну екосистему бібліотек та інструментів, що спрощує розробку складних застосунків.

Фреймворк Qt був обраний для розробки як клієнтської, так і серверної частини системи завдяки своїй гнучкості та можливостям крос-платформної розробки.

Була також обрана система керування базами даних PostgreSQL яка є потужною, надійною та високопродуктивною СКБД з відкритим вихідним кодом, що дозволяє зберігати та обробляти великі обсяги даних, які генеруються системою моніторингу.

Для розробки front-end частини системи було прийнято рішення використовувати мову JavaScript, що дозволяє значно пришвидшити процес розробки завдяки її широкій підтримці та численним бібліотекам. Для спрощення маніпуляцій з елементами DOM та обробки подій було обрано бібліотеку jQuery, яка забезпечує зручний та ефективний спосіб роботи з JavaScript. Для відображення мапи використано бібліотеку Leaflet, яка є потужним інструментом для інтеграції інтерактивних мап у веб-додатки. В якості сервера для обслуговування HTML сторінок було обрано веб-сервер Nginx.

3 РОЗРОБКА АЛГОРИТМІЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Створення архітектури системи

Коли справа доходить до розробки програмного забезпечення, є тисячі речей, які можуть піти правильно і допомогти прискорити розробку. Однак, з іншого боку, є тисячі речей, які можуть піти не так, багато з яких є внутрішніми проблемами процесу розробки, їх можна повністю запобігти за допомогою попереднього планування процесу розробки. Саме тут на допомогу приходить розробка архітектури програмного забезпечення. Нехтування даним кроком при створенні будь якого складного програмного забезпечення може значно затримати розробку, призвести до неочікуваних програмних помилок на стадії тестування та ускладнити подальшу підтримку додатку.

Архітектура програмного забезпечення описує програмне забезпечення, у якому всі компоненти програми добре спроектовані, сплановані та структуровані. Хороший архітектурний дизайн при розробці програмного забезпечення передбачає прийняття правильних рішень щодо організації, компонентів, інтерфейсів і технологій. При розробці архітектури програмного забезпечення важливо обирати інструменти та методи, які допоможуть краще зрозуміти, спроектувати та документувати систему. UML діаграми є одним з таких інструментів та їх використання для розробки архітектури є хорошою практикою, оскільки UML надає широкий набір діаграм для різних аспектів системи [32, 33].

UML (Unified Modeling Language) – уніфікована мова моделювання, що використовується розробниками програмного забезпечення для візуалізації процесів та роботи систем.

Це не мова програмування, скоріше набір правил та стандартів для створення діаграм. Вони дозволяють розробникам програмного забезпечення та інженерам «говорити однією мовою», не заглиблюючись у фактичний код свого

продукту. Складання діаграм за допомогою UML – це чудовий спосіб допомогти собі та іншим розробникам швидко зрозуміти складну ідею чи структуру [34, 35].

Існують різні типи діаграм кожна з яких допомагає краще візуалізувати різні аспекти роботи програмного забезпечення. Класифікація UML діаграм наведена на рис. 3.1.

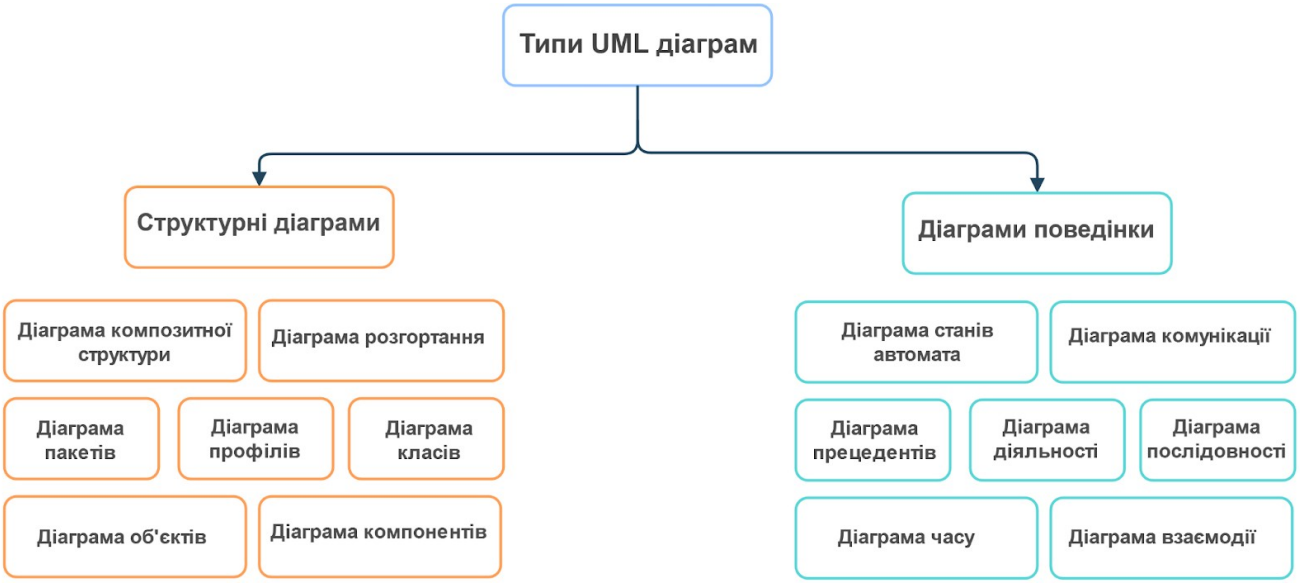

Рисунок 3.1 – Класифікація UML діаграм

Для кращого розуміння загальної архітектури розроблюваної системи варто почати з розгляду структурних діаграм, а саме – UML діаграми розгортання (UML Deployment diagram). Основна мета UML Deployment діаграми полягає у тому, щоб показати, як програмні компоненти системи взаємодіють з фізичними середовищами, в яких вони виконуються. Вона дозволяє моделювати розподіл компонентів, зв'язки між ними та взаємодію з апаратними вузлами. Це важливо для розуміння та оптимізації архітектури системи, особливо у великих розподілених системах, де компоненти системи можуть бути розміщені на різних серверах або пристроях або навіть у різних географічних локаціях.

У системі, що розробляється, серверна частина поділена на Main сервер для безперервного зв’язку серверу з пристроями-клієнтами, API сервер, сервер баз

даних та Nginx вебсервер (Web Server Tier) для хостингу сторінок front-end частини. Усі ці компоненти працюють на сервері з операційною системою Ubuntu 22.04. Окремо знаходиться смартфон на системі Android (Client Tier), що розташований в середині транспортного засобу та відіграє роль пристрою моніторингу, спілкуючись з серверними компонентами по протоколу TCP/IP та з використанням REST API. Ще одним окремим компонентом є веб-браузер працівника, який здійснює моніторинг транспортного флоту (Workstation Tier). Описана архітектура відображена на рис 3.2.

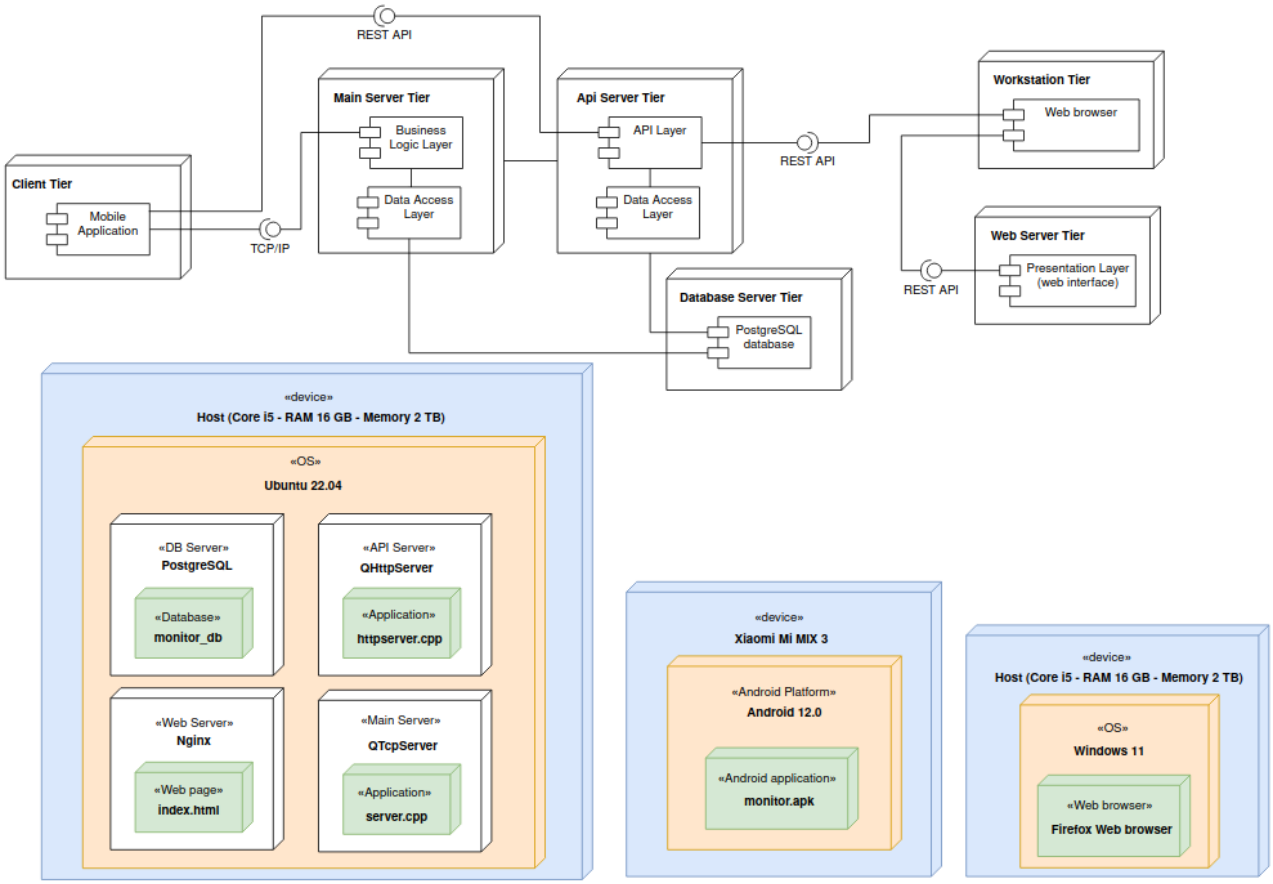

Рисунок 3.2 – Deployment UML діаграма

Ще одним хорошим способом відобразити поведінку системи є Data Flow діаграма. Це тип діаграми, який використовується для опису руху даних всередині системи, зокрема як дані входять, обробляються і виходять із системи. Data Flow діаграма не є частиною стандартного набору UML діаграм. Вона

належить до окремої категорії діаграм, що традиційно використовується в аналізі і проектуванні систем, особливо у системному аналізі та проектуванні програмного забезпечення.

У такому типі діаграм виокремлюють 4 основних компонента:

- Процеси описують обробку даних у системі;
- Потоки даних показують рух даних між процесами, сховищами даних і зовнішніми сутностями;
- Сховища даних, тобто місця, де дані зберігаються;
- Зовнішні сутності, а саме – користувачі, які взаємодіють із системою.

На рис. 3.3 можна спостерігати Data Flow діаграму для системи, що розробляється. Діаграма відображає Android пристрій, що знаходиться у транспортному засобі та додаток для моніторингу, що запущений на ньому (Vehicle device application). По відкритому TCP сокету додаток у форматі JSON відправляє на головний сервер інформацію про зміну місцеположення транспортного засобу та події, що стаються під час здійснення перевезення для подальшого зберігання. Якщо подія передбачає також відправку файлу (наприклад фото) пристрій надсилає його на окремий API сервер. На даній схемі можна також побачити компонент Fleet manager web browser interface, що відображає веб-браузер працівника, який буде здійснювати моніторинг транспортного флоту. Застосунок запущений у веб-браузері отримує HTML сторінки з окремого Nginx вебсерверу та інформацію, яку необхідно відобразити на них з серверу для API.

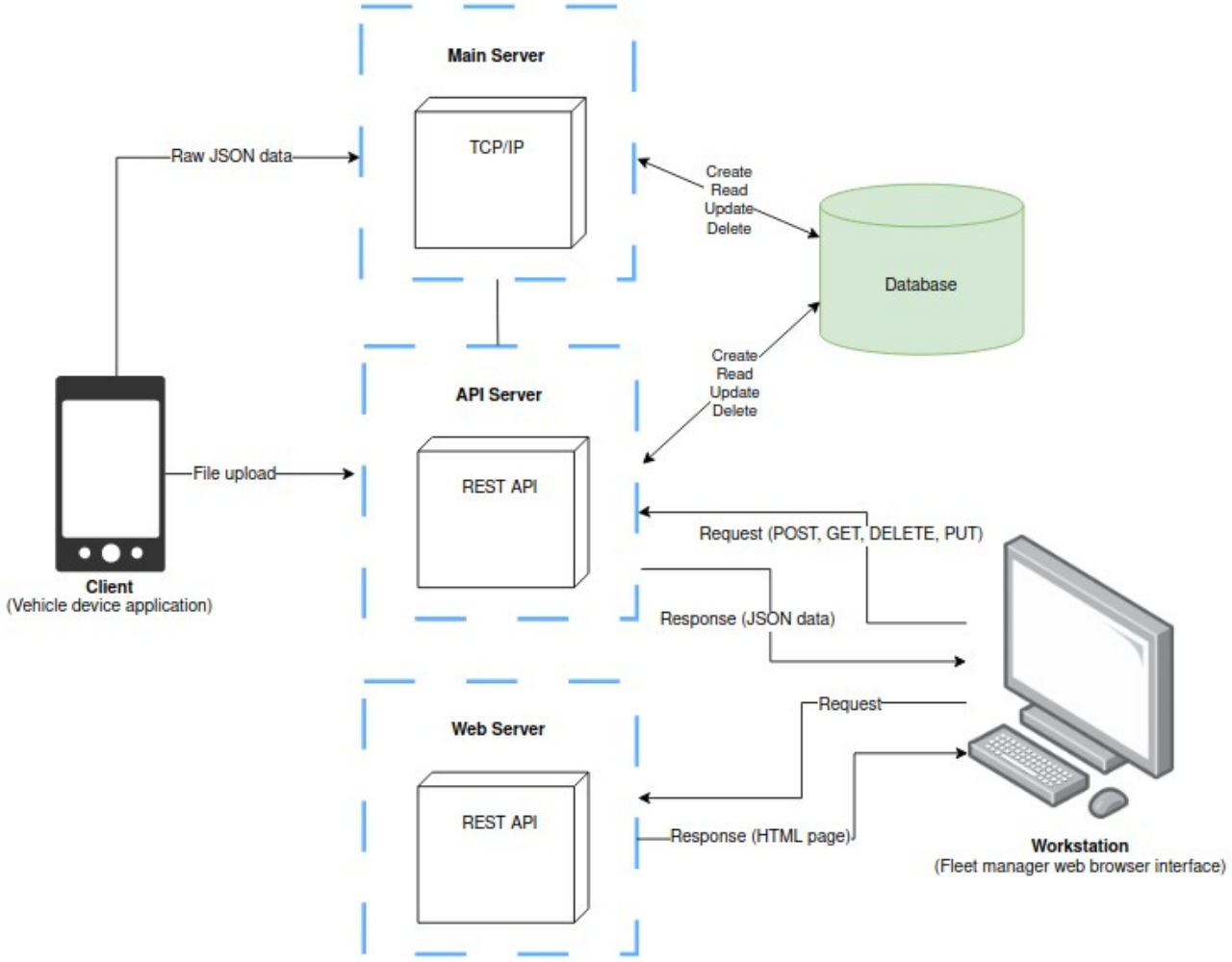

Рисунок 3.3 – Data Flow UML діаграма

Після детального аналізу архітектури системи моніторингу стає зрозуміло, що саме взаємодія між пристроєм-клієнтом та сервером є основною складовою даної системи і потребує більш ретельного дослідження.

Для цього доцільно буде застосувати одну із UML діаграм поведінки, а саме – діаграму послідовності. Діаграма послідовності (Sequence Diagram) – показує часові особливості передачі і прийому повідомлень об’єктами. Впорядкованість за часом слід розуміти як послідовність дій.

Як і усі UML діаграми, діаграма послідовності має чітко виражену структуру та набір компонентів з яких має складатись:

- Лінія життя починається з об’єкта-прямокутника (голова) та зображується вертикальною пунктирною лінією (стеблом). Вона служить для позначення

періоду часу, протягом якого об’єкт існує в системі. Якщо об’єкт існує в системі постійно, то його лінія життя повинна продовжуватися по всій площині діаграми зверху донизу.

- Смуга активації (фокус управління) – тонкий прямокутник на лінії життя, протягом якого елемент виконує операцію. Довжина прямокутника вказує на тривалість перебування об’єктів в активному режимі.

- Повідомлення (виклики) з’являються в послідовному порядку на лінії життя. Повідомлення зображується за допомогою стрілок. Початок стрілки завжди торкається лінії життя відправника та лінії життя об’єкта, що приймає повідомлення [36].

На рис. 3.4 зображено UML діаграму послідовності для опису взаємодії пристрою-клієнта з компонентами серверної частини. Спочатку пристрій надсилає свої облікові дані для того щоб отримати унікальний токен та авторизуватись у системі. Варто зауважити, що кожне повідомлення клієнта надходить у JSON форматі та обробляється на основному сервері. Якщо дані пристрою вдалось знайти у базі це означає, що авторизація успішна і сервер повідомляє про це клієнта та відкриває з ним постійне TCP з’єднання.

Після успішної авторизації у системі пристрій з деякою періодичністю надсилає своє місцезнаходження на сервер. Якщо пристрій фіксує під час поїздки якісь позаштатні ситуації, такі як різке прискорення чи гальмування, він також сформує цю інформацію у JSON та надішле по TCP сокету. Деякі типи подій можуть також вимагати файлу фото або відео для більш чіткого розуміння того що сталось. Файли пристрій надсилає на окремий API-сервер.

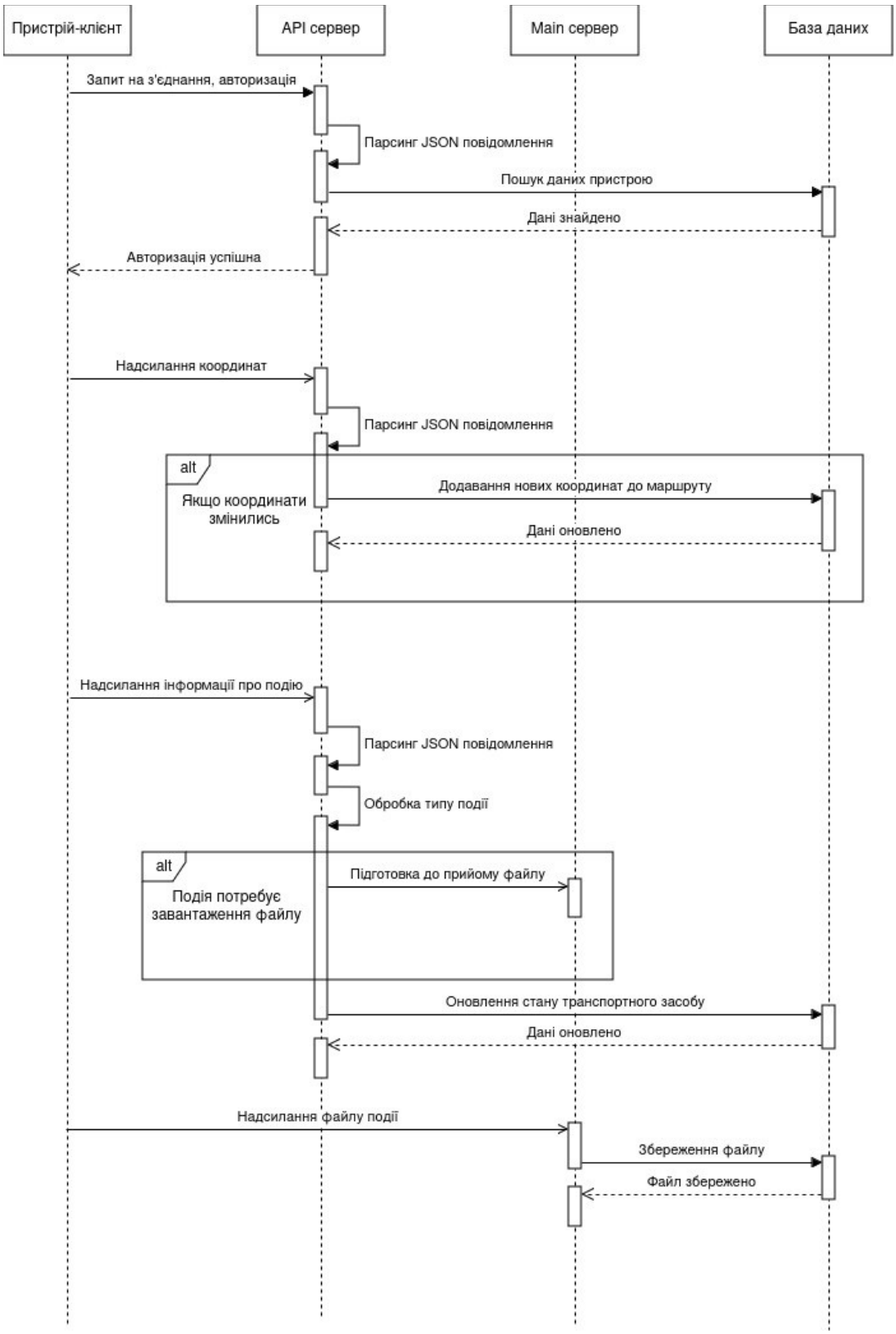

Рисунок 3.4 – UML діаграма послідовності взаємодії пристрою-клієнта та сервера

Доцільно також розглянути, яким паттернам проєктування відповідає розроблена архітектура системи моніторингу транспортних перевезень. Можна проспостерігати, що у даної системи є збіжності з подійно-орієнтованою архітектурою (event-driven architecture або EDA), що є потужним паттерном для систем, де важливою є реакція на асинхронні події [37, 38].

Основними компонентами подійно-орієнтованої архітектури є виробники подій (producers) споживачі подій (consumers) та менеджер подій (або брокер), що знаходиться між ними. Схема подійно-орієнтованої архітектури наведена на рис 3.5.

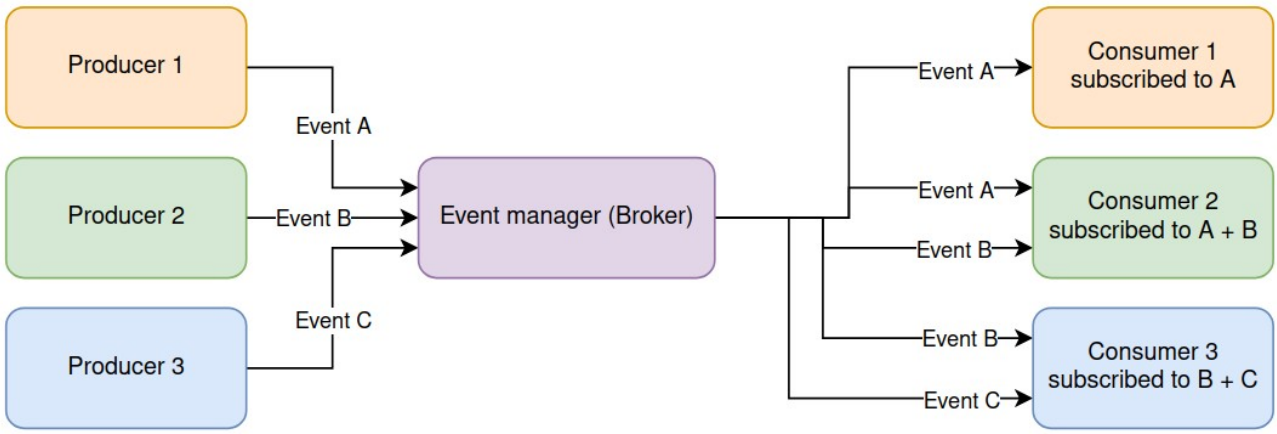

Рисунок 3.5 – Узагальнена схема подійно-орієнтованої архітектури

У системі, що розробляється, цим компонентам відповідають:

- Виробниками подій (Producers) є пристрої-клієнти, які підключаються до основного TCP-сервера, виконують роль виробників подій. Вони постійно надсилають координати транспортних засобів та інформацію про події, такі як відволікання водія, зупинки або технічні проблеми. Якщо подія включає медіафайли, наприклад, фото або відео, ці файли завантажуються на API-сервер, а подія, що супроводжується медіафайлом, включає посилання на цей файл.

- Обробником подій (Broker) є центральний компонент, що зачасту реалізований через систему черг повідомлень або спеціалізовані сервери, які приймають та розподіляють події. У розроблюваній системі TCP-сервер та API-сервер виступають в цій ролі, приймаючи дані від виробників подій та розподіляючи їх до відповідних споживачів. Цей підхід забезпечує надійну та масштабовану обробку подій.
- Consumers (Споживачі подій): База даних та front-end компоненти виступають споживачами подій. База даних зберігає всі події, маршрути та інші дані, що надходять від клієнтських пристроїв. front-end в свою чергу використовує ці дані для відображення актуальної інформації про транспортні засоби та їхні маршрути користувачам у реальному часі.

Варто зауважити, що подійно-орієнтована архітектура буде реалізована не тільки для спілкування між компонентами системи а й в їх середині. Адже для розробки використовується фреймворк Qt для якого події та реакція на них (система сигналів та слотів) є візитною карткою.

Окрім EDA, архітектура системи також відповідає паттерну MVC (Model-View-Controller), який розділяє додаток на три основні компоненти: модель, уявлення та контролер.

Model (Модель) – це компоненти, що відповідають за управління даними, логікою бізнесу та правилами. У розроблюваній системі системі це база даних, де зберігається інформація про користувачів, транспортні засоби, маршрути та події. Модель обробляє дані, отримані від клієнтів, зберігає їх у базі даних та надає необхідну інформацію іншим компонентам системи.

View (Уявлення) – компоненти, що відповідають за відображення даних користувачам. Мова йде про front-end, що використовує JavaScript, jQuery та Leaflet для інтерактивного відображення карт та даних про транспортні засоби. Уявлення взаємодіє з моделлю через контролер для отримання даних та відображення їх у зрозумілій і зручній формі.

Controller (Контролер) – означає компоненти, що відповідають за обробку запитів від користувачів, взаємодію з моделлю та оновлення уявлення. У системі контролери реалізовані як серверні компоненти, що приймають запити від клієнтів, обробляють їх, взаємодіють з моделлю (базою даних) та відправляють необхідні дані до уявлення. Узагальнена схема MVC архітектури зображена на рис. 3.6 [39, 40].

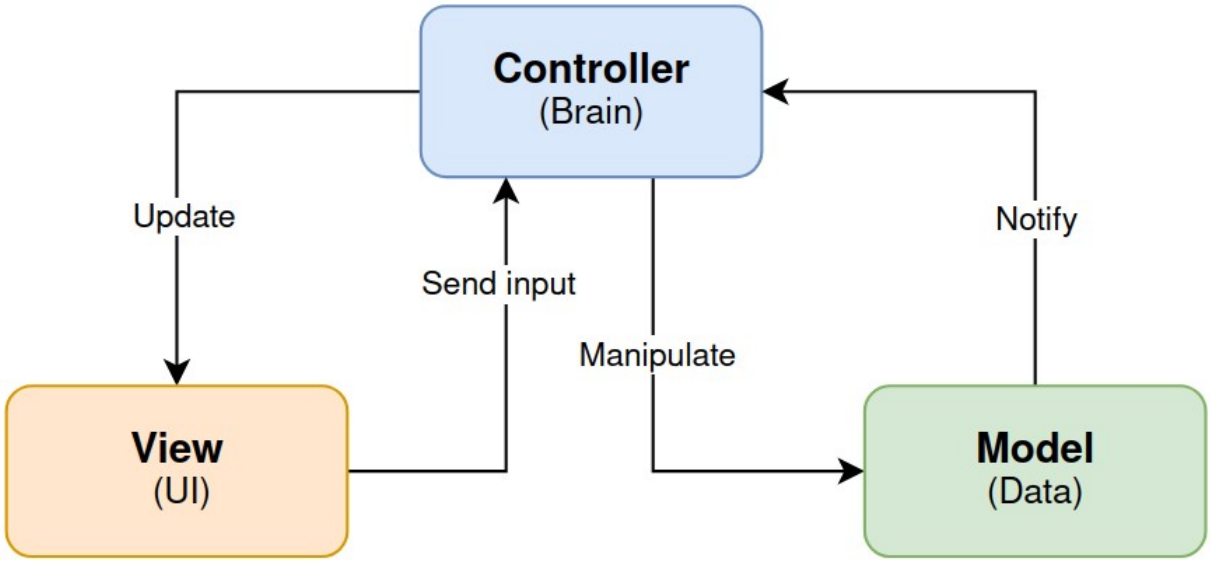

Рисунок 3.6 – Узагальнена схема MVC архітектури

3.2 Розробка структури бази даних

Збереження даних є не менш важливим аспектом розроблюваної системи моніторингу транспортних перевезень, оскільки загальна швидкодія системи залежить від точного та своєчасного зберігання даних про транспортні засоби, маршрути, події та користувачів. Ефективна база даних повинна також забезпечувати можливість швидкого доступу до даних, їх аналізу та використання для прийняття подальших рішень.

Для кращого розуміння структури бази даних, варто відобразити її графічно. Найбільш підходящим способом для цього є використання ER-діаграм (діаграм сутність-зв'язок). ER-діаграми дозволяють візуалізувати основні сутності системи та зв'язки між ними, що допомагає зрозуміти логічну модель бази даних.

ER-діаграми використовують різні нотації для представлення сутностей, атрибутів та зв'язків. Однією з найпоширеніших є нотація Crow’s Foot, яка має наступні основні елементи:

- Сутності (Entities): Представлені прямокутниками, сутності є об'єктами, про які необхідно зберігати інформацію. Наприклад, у розроблюваній системі це можуть бути "Користувачі", "Транспортні засоби", "Маршрути" та "Події".
- Атрибути (Attributes): атрибути описують властивості сутностей. Наприклад, атрибути сутності "Користувачі" можуть включати "Ім'я", "Прізвище", "Роль" тощо.
- Зв'язки (Relationships): Представлені лініями, що з'єднують сутності. Зв'язки показують, як сутності взаємодіють одна з одною. Наприклад, зв'язок між сутностями "Користувачі" та "Маршрути" може вказувати, що користувач може бути водієм певного маршруту.
- Кардинальність (Cardinality): Представлена у вигляді "воронячої лапки" (Crow’s Foot) на кінцях ліній, кардинальність визначає кількість екземплярів однієї сутності, що можуть бути пов'язані з екземплярами іншої сутності. Наприклад, один користувач може мати кілька транспортних засобів, а транспортний засіб може бути закріплений лише за одним користувачем.

  Основні можливі види зв’язків у ER-моделі представлені на рис 3.7.

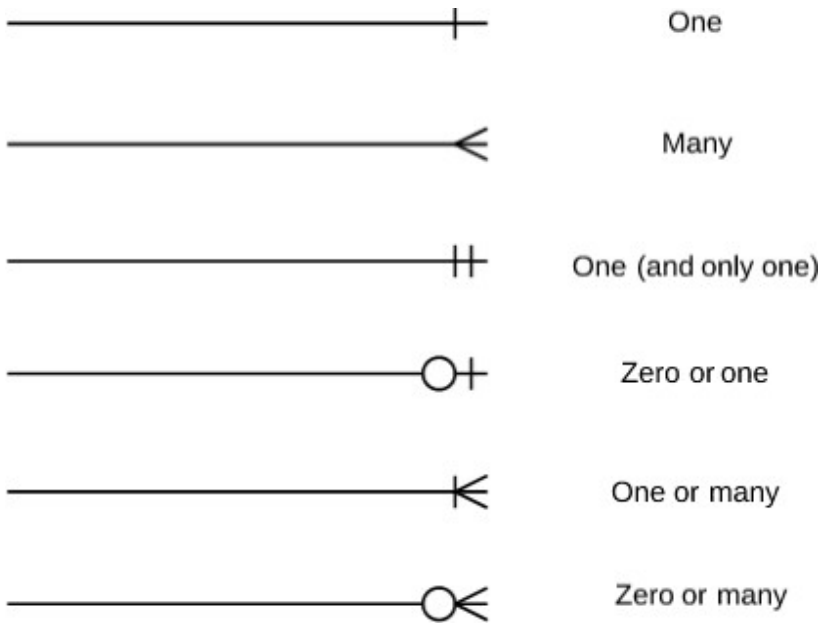

Рисунок 3.7 – Можливі види зв’язків у ER-моделі

У системі, що розробляється центральною сутністю бази даних виступає таблиця Users (Користувачі). Кожен користувач в системі має унікальний id, декілька полів з особистою інформацією та поле role, що визначає його роль в системі (менеджер, водій, адміністратор). Можна спостерігати, що Users має з’єднання “один до одного” з таблицею Drivers (водії). Дана таблиця розширює базову таблицю Users та необхідна для зберігання додаткової інформації про водія (такої як номер водійського посвідчення чи оцінка в системі).

З таблицею Driver поєднана, також з’єнанням “один до одного”, пов’язана таблиця Vehicles (Транспортні засоби). Це означає, що в один момент засу за водієм може бути закріплений тільки один конкретний автомобіль, на якому він здійснює перевезення. Сутність Vehicles зберігає усю базову необхідну інформацію, таку як марка, модель, номерний знак, id водія, що закріплений, про кожен транспортний засіб.

Зображена на схемі також таблиця Routes (Маршрути), що відображає перевезення, здійснені транспортним засобом. Важливим є те, що дана таблиця поєднана з Vehicles з’єднанням “нуль або багато”. Даний зв’язок говорить про те, що історія поїздок конкретного транспортного засобу може складатись з деякої

кількості маршрутів, проте одна сутність Route може належати тільки одному транспортному засобу.

Як вже було з’ясовано при проєктуванні архітектури системи, окрім інформації про своє місцезнаходження транспортний засіб повинен передавати дані про події, якщо такі стаються під час поїздки. Для цього виділена таблиця Events (Події), що тримає в собі всю необхідну інформацію подію. Такою інформацією є індекс поїздки під час якої сталась подія (route_id), час події (time), широта та довгота для зображення точного місця події на мапі (latitude та longtitude відповідно) та тип події (type). З’єднання “нуль або багато” між сутностями Routes та Events говорить про те, що конкретна подія завжди прив’язана до поїздки під час якої сталась. У той же час під час поїздки може статись багато подій, які усі будуть закріплені за даною сутністю маршруту.

При проєктуванні системи було також вирішено, що для більш детального розуміння подій деякі з них будуть вимагати прикріплення до себе одного або декількох файлів. Для файлів виділена окрема таблиця Files, що тримає в собі індекс події та шлях до файлу. Тип з’єднання говорить про те, що до однієї події можна прикріпити декілька файлів.

Описана структура бази даних зображена на рис. 3.8 у вигляді ER-моделі.

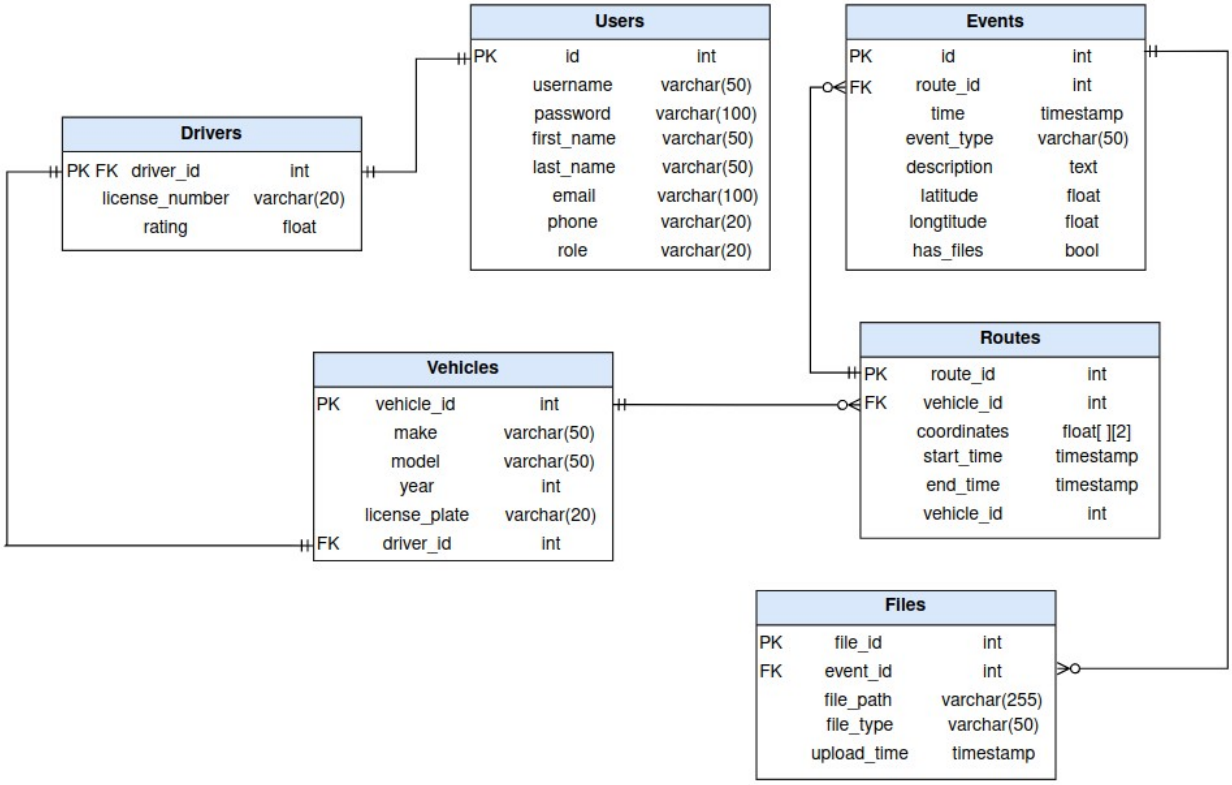

Рисунок 3.8 – ER-модель системи, що розробляється

3.3 Розробка удосконаленого підходу розпізнавання стану водія засобами смартфону

Завдання виявлення аномалій у поведінці водія в реальному часі з використанням мобільних пристроїв супроводжується низкою технічних обмежень: обмежені обчислювальні ресурси, вимоги до енергоефективності та необхідність безперервного відеопотоку. Традиційні підходи, такі як Haar cascade, хоч і є легковаговими, часто демонструють низьку стійкість до поворотів голови, змін освітлення та часткових перекриттів.

У той же час глибокі нейронні мережі, як OpenCV DNN, забезпечують вищу точність, однак потребують суттєвих ресурсів, що не завжди доступно для бюджетних або середніх Android-пристроїв.

З метою пошуку компромісу між точністю розпізнавання та обчислювальними витратами, було використано модель Shape predictor. Вона демонструє значно стабільнішу поведінку при зміні положення голови у порівнянні з Haar cascade. Порівняльне тестування, здійснене засобами смартфону Samsung Galaxy A16, наведено у таблиці 3.1.

Таблиця 3.1 – Порівняння ефективності моделей розпізнавання обличчя

Модель	**Розмір моделі**	**Середній час обробки кадру (мс)**	**Точність виявлення обличчя (IOU)**	**Стійкість до поворотів голови**	**Висновок**
Haar Cascade	1.2 МБ	40	Середня (0.65)	Низька	Висока швидкість, але чутливість до освітлення та положення голови
shape_predictor_5	5.7 МБ	65	Висока (0.79)	Висока	Швидша за 68-точкову модель, але менш точна у складних умовах
shape_predictor_68	95 МБ	140	Дуже висока (0.9)	Висока	Висока точність, але значне навантаження на ресурси пристрою

Тут для оцінювання точності виявлення обличчя використовується метрика IoU (Intersection over Union), яка визначається як відношення площі перетину прямокутників (прогнозованого та еталонного) до площі їх об'єднання:

$$IOU = \frac{A_{predicted} \cap A_{groundtruth}}{A_{predicted} \cup A_{groundtruth}} \quad (3.1)$$

де $A_{predicted}$ – площа прямокутника, передбаченого алгоритмом;

$A_{groundtruth}$ – площа еталонного прямокутника визначеного вручну.

Для оцінки стійкості моделі до поворотів голови було здійснено виявлення обличчя з різними кутами повороту. Кут повороту голови розраховувався за формулою:

$$\theta = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \quad (3.2)$$

де (x1, y1) і (x2, y2) – координати центру лівого та правого ока.

Значення стійкості «Низька» було надано у випадку якщо проблеми з виявленням рис обличчя виникали вже при куті повороту ±20°. Висока стійкість означає що модель стабільно працює при куті до ±40°.

Для наочної оцінки ефективності різних підходів було проведено тестування на трьох моделях смартфонів із різними апаратними характеристиками. Результати часу обробки зображення представлені у вигляді стовпчастої діаграми, що зображена на рис 3.4.

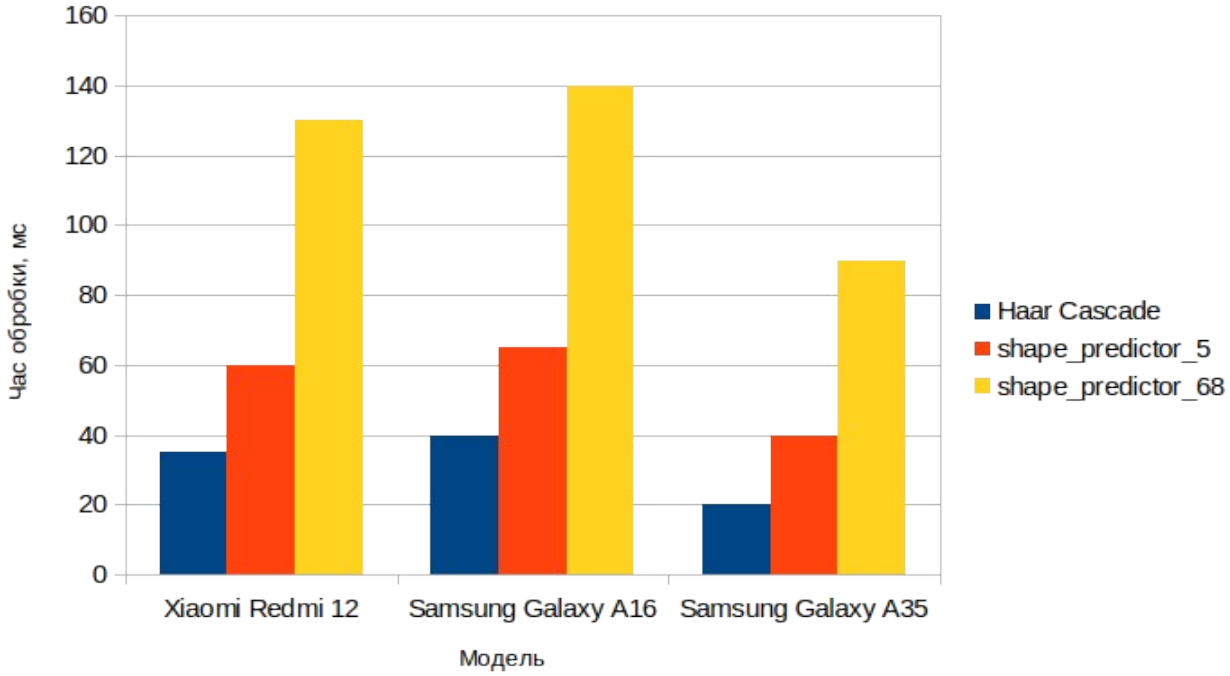

Рисунок 3.4 – Порівняння продуктивності на різних моделях смартфонів

Враховуючи характеристики розглянутих моделей було запропоновано комбінований підхід до розпізнавання обличчя, що поєднує використання моделей shape_predictor_5_face_landmarks та shape_predictor_68_face_landmarks. Такий вибір обумовлено потребою досягнення компромісу між швидкодією та

точністю розпізнавання на мобільних пристроях із обмеженими обчислювальними ресурсами.

Комбінований метод розпізнавання стану водія, реалізований у мобільному додатку, включає в себе наступні кроки:

Крок 1 - Ініціалізація камери та захоплення зображення. Камера смартфону в реальному часі захоплює кадри, які передаються до обробника зображень. Відбувається перевірка доступу до ресурсу та ініціалізація параметрів, включаючи частоту оновлення та роздільну здатність.

Крок 2 - Попереднє оброблення зображення. Перед передачею зображення в алгоритм, воно перетворюється у відтінки сірого, масштабується до уніфікованого розміру та за потреби застосовуються згладжування чи гістограмне вирівнювання.

Крок 3 - Виявлення обличчя з використанням shape_predictor_5. На практиці, модель з 5 ключових точок використовується для оперативного відстеження простих поведінкових ознак водія (наприклад, напрямок погляду, поворот голови). Вона має відносно низьке навантаження на процесор і працює із частотою 20 кадрів на секунду на середньобюджетних смартфонах, що дозволяє застосовувати її у реальному часі без помітної затримки. Детальний опис обробки зображення на даному кроці представлено у вигляді UML-діаграми діяльності на рис 3.5.

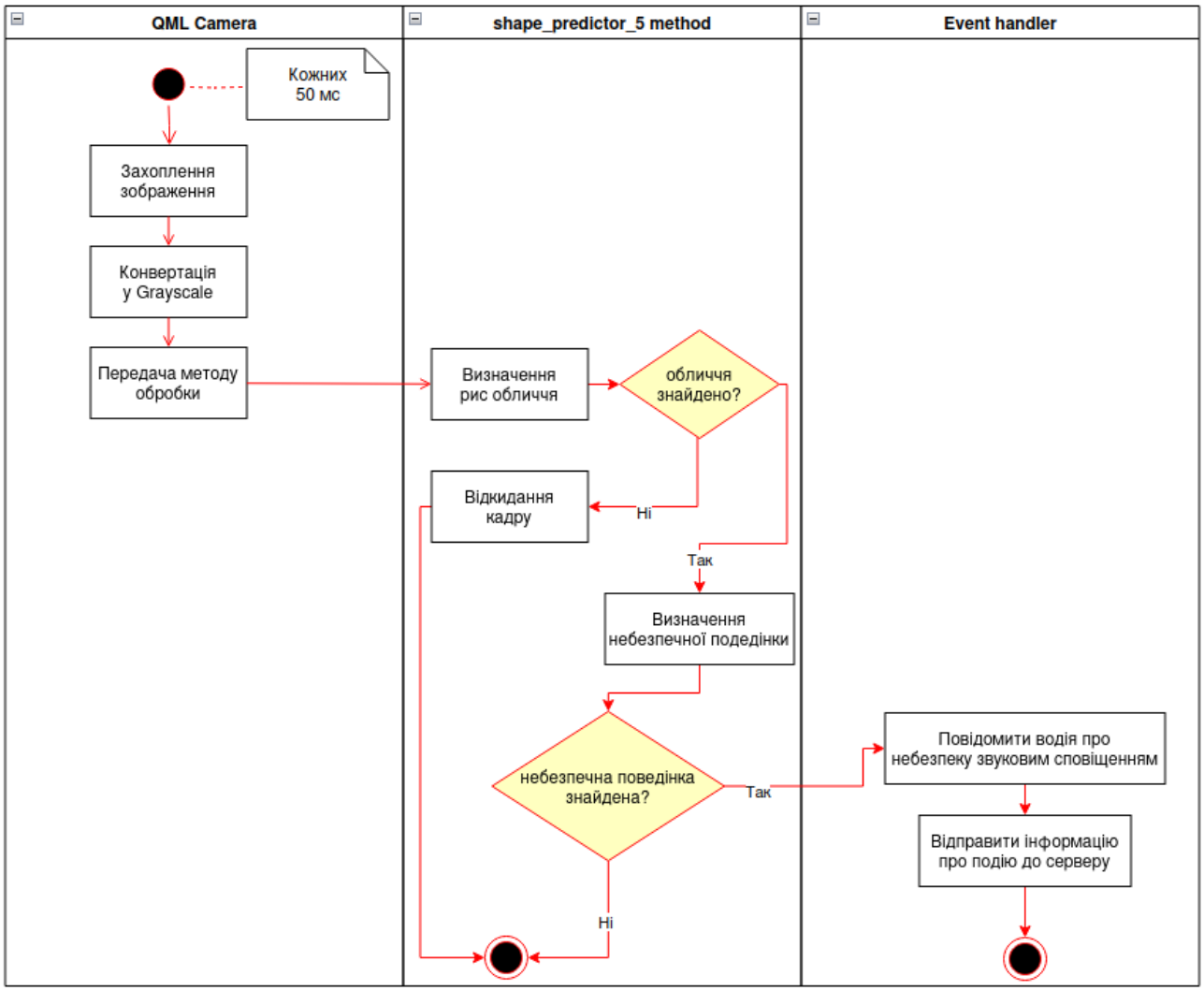

Рисунок 3.5 – UML-діаграма діяльності методу shape_predictor_5

Крок 4 - Виявлення особливо небезпечних поведінкових ознак водія методом shape_predictor_68. У свою чергу, модель з 68 ключових точок активується періодично. Вона призначена для виконання більш складних аналітичних завдань, таких як виявлення ознак сонливості, мікровиразів чи асиметрії у міміці, які можуть сигналізувати про втрату водієм концентрації. Через вищу обчислювальну складність, ця модель використовується з частотою 1–2 кадри на секунду, у фоновому режимі, з обмеженням частоти викликів для уникнення перевантаження системи. Опис обробки зображення на даному кроці представлено у вигляді UML-діаграми діяльності на рис 3.6.

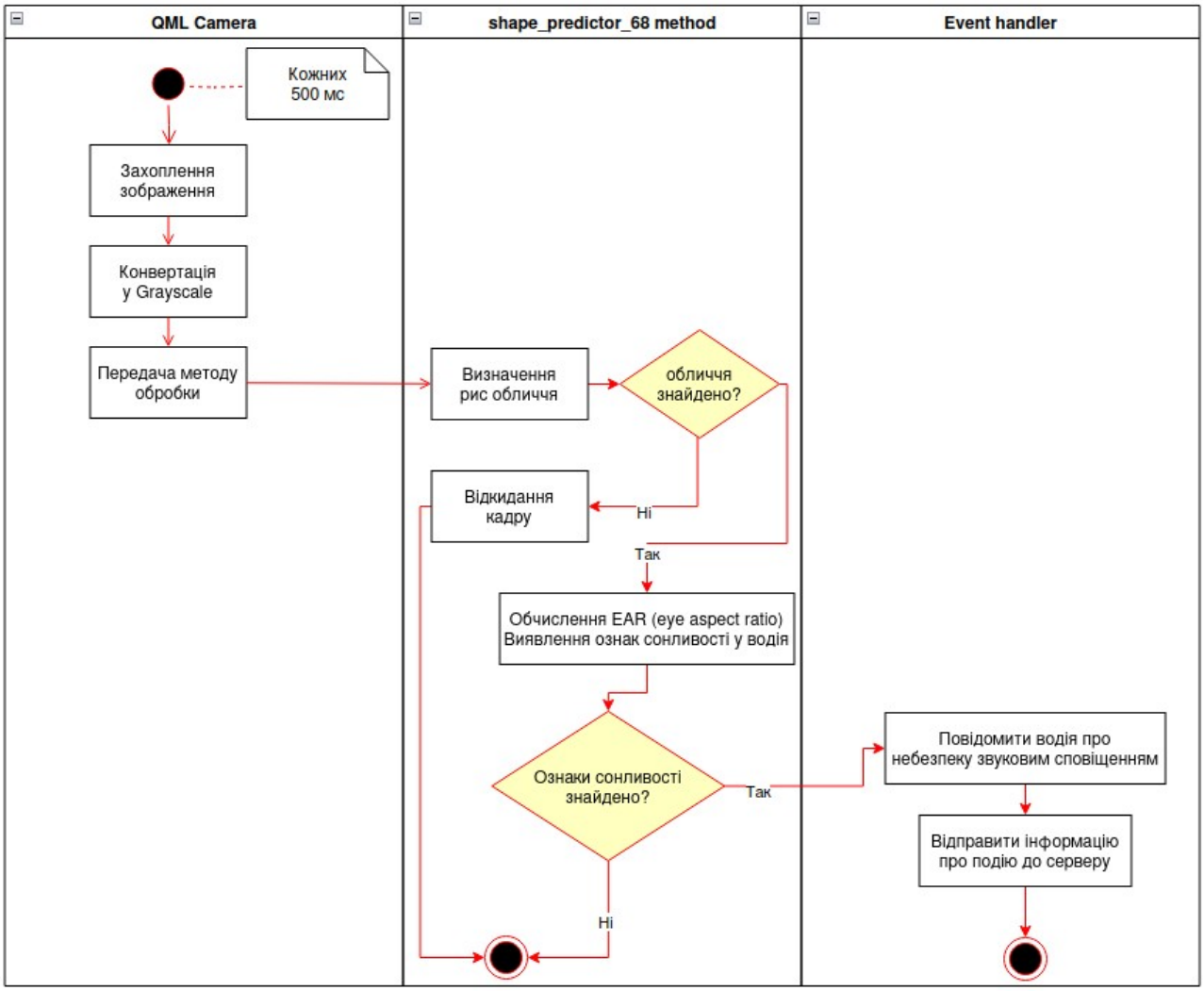

Рисунок 3.6 – UML-діаграма діяльності методу shape_predictor_68

Крок 5 - Передача даних на сервер та візуалізація. Результати роботи методу передаються через REST API до серверної частини, для подальшого зберігання та відображення у вебінтерфейсі.

Як можна спостерігати робота системи розпізнавання націлена на виявлення різноманітних паттернів поведінки водія. Відповідно було запропоновано класифікацію цих поведінкових паттернів відповідно до трьох основних задач системи розпізнавання:

1. Просте відстежування напрямку погляду (базується на 5-точковій моделі). Цей режим активний постійно, оскільки вимагає мінімальних ресурсів. Основними завданнями відстеження погляду водія є:

- Виявлення повторюваних поглядів убік (може вказувати на розмови з пасажирами, відволікання);
- Фіксація довготривалого погляду вниз (ймовірне використання смартфону або ознаки фізичного дискомфорту);
- Реєстрація нестабільного погляду або його відсутності, що свідчить про втрату уваги.

2. Аналіз міміки та виявлення сонливості (на основі 68-точкової моделі). Цей режим активується періодично для зниження навантаження. До особливостей, які виявляє даний режим відносяться:

- Зниження частоти моргання;
- Опущені повіки;
- Асиметрія губ або знижений тонус м’язів обличчя;
- Затримка реакції або втрата фокусу погляду.

Такі ознаки часто пов’язані з втомою та сонливістю і становлять великий ризик під час керування транспортом.

3. Виявлення сторонніх об’єктів у зоні обличчя (на основі 68-точкової моделі). Виявлення часткових перекриттів обличчя може сигналізувати про:

- Паління за кермом (рука з сигаретою біля обличчя);
- Вживання їжі чи напоїв;
- Закриття обличчя сторонніми предметами (телефоном, одягом, тощо).

Таким чином, застосування адаптивного механізму перемикання між моделями залежно від сценарію дозволяє забезпечити високу ефективність та точність розпізнавання без шкоди для продуктивності мобільного пристрою. Це рішення є обґрунтованим у контексті розробки додатку водія для системи моніторингу та становить основу наукової новизни представленої роботи.

3.4 Оцінка ефективності запропонованого підходу

Ефективність алгоритмів виявлення та аналізу обличчя в мобільних системах, особливо в умовах обмежених обчислювальних ресурсів, є ключовим фактором для практичного впровадження. Враховуючи, що запропонований підхід базується на комбінації двох моделей: shape_predictor_5_face_landmarks для швидкого виявлення та shape_predictor_68_face_landmarks для глибшого аналізу стану водія. Постає необхідність обґрунтованої оцінки його ефективності.

Було проведено порівняльне тестування запропонованого комбінованого методу з сучасним підходом на основі глибоких нейронних мереж (DNN), зокрема MobileNetV2, реалізованого засобами OpenCV на базі смартфону Samsung Galaxy A16. Для розрахунку показників точності виявлення обличчя та стійкості до поворотів голови були використані формули 3.1 та 3.2 відповідно. Результати порівняння представлені у таблиці 3.2.

Таблиця 3.2 – Порівняння ефективності запропонованої моделі та DNN

Модель	**Розмір моделі**	**Середній час обробки кадру (мс)**	**Точність виявлення обличчя (IOU)**	**Стійкість до поворотів голови**	**Висновок**
DNN з OpenCV MobileNetV2	14.3 МБ	161	Дуже висока (0.95)	Дуже висока	Висока точність, але значне навантаження
Запропонований метод	100.7 МБ	68	Висока (0.86)	Висока	Відмінний баланс швидкості та точності

Порівняння швидкості обробки зображень двома методами було здійснено на трьох смартфонах із різними апаратними можливостями:

- Xiaomi Redmi 12 – бюджетний рівень
- Samsung Galaxy A16 – середній клас
- Samsung Galaxy A35 – сучасний флагман

Результати тестування представлено у вигляді стовпчастої діаграми на рис. 3.7.

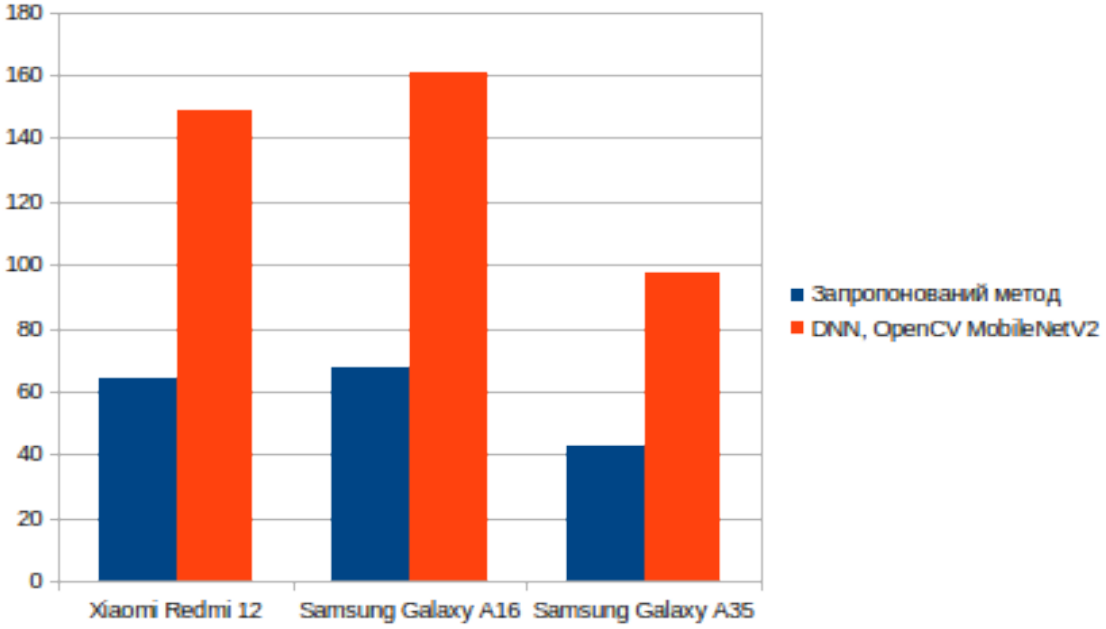

Рисунок 3.7 – Порівняння середнього часу обробки кадру для комбінованого підходу та DNN (MobileNetV2) на різних моделях смартфонів

Проведена оцінка ефективності демонструє, що запропонований комбінований підхід (shape_predictor_5 + shape_predictor_68) є доцільною альтернативою DNN-методам для використання на смартфонах середнього та бюджетного класу. Попри незначну втрату точності у порівнянні з MobileNetV2 (менше 10%), він забезпечує значно вищу швидкодію (у 2–2.5 разів менший час обробки кадру, в залежності від моделі смартфона), що критично важливо для задач у реальному часі, особливо у сфері безпеки водія.

3.5 Розробка програмного забезпечення

Розробка програмного забезпечення є ключовим етапом створення автоматизованої системи моніторингу транспортних перевезень, оскільки саме

на цьому етапі визначені вимоги набувають форми реального функціонального продукту. Процес проєктування та реалізації основних компонентів системи включають розробку серверної частини, клієнтського веб-застостунку для менеджера транспортного флоту та мобільний застосунок водія. Кожен із цих компонентів виконує свою роль у загальній архітектурі рішення, забезпечуючи взаємодію користувачів із системою, збирання та обробку даних.

3.5.1 Розробка серверної частини

Важливим етапом перед написанням основного програмного забезпечення є підготовка класу Logger. Тут буде використана функція qInstallMessageHandler для виводу більш детальної інформації під час відладки, що значно пришвидшить процес подальший процес розробки. Logger забезпечує вивід таких даних, як час виконання коду, функція, файл, та поток, у якому виконується певний фрагмент коду. При цьому даний клас обробляє повідомлення різного типу (інформаційні, попередження, помилки). Це дозволить швидше знаходити та виправляти помилки, а також забезпечить кращий контроль над процесом виконання програмного коду. Приклад виводу повідомлень класом Logger зображено на рис. 3.8.

```
MainThread | 08:39:04.651 | Debug | ../Server/httpserver.cpp | 27 | start | HTTP server started at 6002
MainThread | 08:39:04.652 | Debug | ../Server/server.cpp | 191 | start | Server started at 6001
AssetThread_1 | 08:39:40.701 | Debug | ../Server/asset.cpp | 16 | init | 0 connected
AssetThread_1 | 08:39:41.602 | Debug | ../Server/asset.cpp | 142 | read | 0 "{\"method\": \"positionUpdated\", \"lat\":  37.4221, \"lng\": -122.084}"
AssetThread_1 | 08:39:41.603 | Debug | ../Server/asset.cpp | 58 | isValidJson | json valid
AssetThread_1 | 08:39:41.603 | Debug | ../Server/asset.cpp | 69 | processJson | processJson
AssetThread_1 | 08:39:41.603 | Debug | ../Server/asset.cpp | 93 | positionUpdated | position updated
AssetThread_1 | 08:39:45.207 | Debug | ../Server/asset.cpp | 142 | read | 0 "{\"method\": \"positionUpdated\", \"lat\":  37.2999, \"lng\": -121.801}"
AssetThread_1 | 08:39:45.207 | Debug | ../Server/asset.cpp | 58 | isValidJson | json valid
AssetThread_1 | 08:39:45.207 | Debug | ../Server/asset.cpp | 69 | processJson | processJson
AssetThread_1 | 08:39:45.207 | Debug | ../Server/asset.cpp | 93 | positionUpdated | position updated
```

Рисунок 3.8 – Приклад повідомлень класу Logger

Центральним компонентом у серверній частині буде виступати клас Server. Він наслідується від QTcpServer, що є частиною бібліотеки Qt і забезпечує базовий функціонал для створення TCP-сервера. QTcpServer дозволяє легко налаштовувати сервер для прослуховування вхідних з'єднань та прийому клієнтських запитів.

Основними полями та методами такого класу є:

- Server::start – налаштовує та запускає сервер з використанням методу QtcpServer::listen
- Server::incomingConnection це перевизначений метод QTcpServer, що викликається при кожному новому підключенні клієнта. Він створює новий об'єкт клієнта та ініціалізує обробку повідомлень від нього.
- Поле QHash<uint, QPointer<QObject>> objects; зберігає вказвники на об’єкти клієнтів. Для роботи з конкретним об’єктом клієнта пропонуються методи getHandler, addHandler, removeHandler.

Об’єкти, що зберігаються у objects мають клас Asset, який наслідується від QObject та представляє пристрій клієнта в системі. Цей клас відповідальний за взаємодію з TCP сервером, а також за обробку подій та координат, що надходять від пристрою клієнта.

Серед ключових методів можна виділити:

- read відповідає за читання даних, що надходять від TCP сервера. Він викликається, коли пристрій-клієнт отримує нові дані від сервера.
- processJson відповідає за обробку даних у форматі JSON, що надходять. Даний метод здійснює парсинг JSON-документа щоб з’ясувати тип повідомлення.
- auth відповідає за процес аутентифікації клієнта на сервері. При першому підключенні клієнт надсилає свої облікові дані для перевірки.
- eventOccurred обробляє звичайні події, що відбуваються під час поїздки. Це можуть бути події, пов'язані з транспортним засобом або зовнішніми факторами.

- emergencyEvent обробляє надзвичайні події, які потребують негайної реакції. Ці події можуть включати аварії, поломки та інші критичні ситуації.

Не менш важливим аспектом серверної частини є забезпечення багатопоточності. Кожен клієнт постійно передає координати, події та інші дані, і обробка цих даних в реальному часі вимагає значних ресурсів. Використання багатопоточності дозволяє розподілити ці завдання між кількома потоками, що значно підвищує продуктивність та забезпечує безперебійну роботу системи навіть при великій кількості підключених клієнтів.

Клас WorkerPool відповідає за управління групою потоків (QThread), які використовуються для обробки даних від клієнтів. Цей клас дозволяє створювати, ініціалізувати та керувати кількома потоками, забезпечуючи ефективне розподілення завдань між ними. Даний клас забезпечує наступний функціонал:

- initializeThread ініціалізує новий потік, додаючи його до масиву потоків, якими керує WorkerPool. Цей метод використовується для налаштування потоків перед їх використанням.
- Метод getThread повертає потік, який можна використовувати для виконання певного завдання. Це дозволяє балансувати навантаження між потоками, забезпечуючи рівномірний розподіл завдань.
- next повертає наступний потік у черзі для виконання завдань. Таким чином забезпечується послідовний розподіл завдань між потоками.

Ще одним ключовим класом є HTTPServer, що також наслідується від QTcpServer та працює паралельно основному серверу. Цей клас використовується для реалізації API, формуючи та відправляючи дані у форматі JSON.

- Методи getRawArgs, splitArg та trimArgs використовуються для попередньої обробки параметрів запиту.

- route використовується для визначення обробника запиту на основі шляху URL. Він визначає, який метод або функція повинні обробити конкретний запит.
- processRequest відповідає безпосередньо за обробку HTTP-запиту. Він викликає відповідні методи для отримання та обробки аргументів, визначає маршрут запиту і формує відповідь у форматі JSON.

Нарешті клас DBManager призначений для управління запитами до бази даних PostgreSQL у розроблюваній системі моніторингу транспортних засобів. Він забезпечує інтерфейс для відкриття та закриття підключення до бази даних, вставки даних у таблиці та отримання даних з них. Цей клас полегшує взаємодію з базою даних, інкапсулюючи складність SQL-запитів у прості методи.

Описані класи та зв’язки між ними зображені у вигляді UML діаграми класів на рис. 3.9.

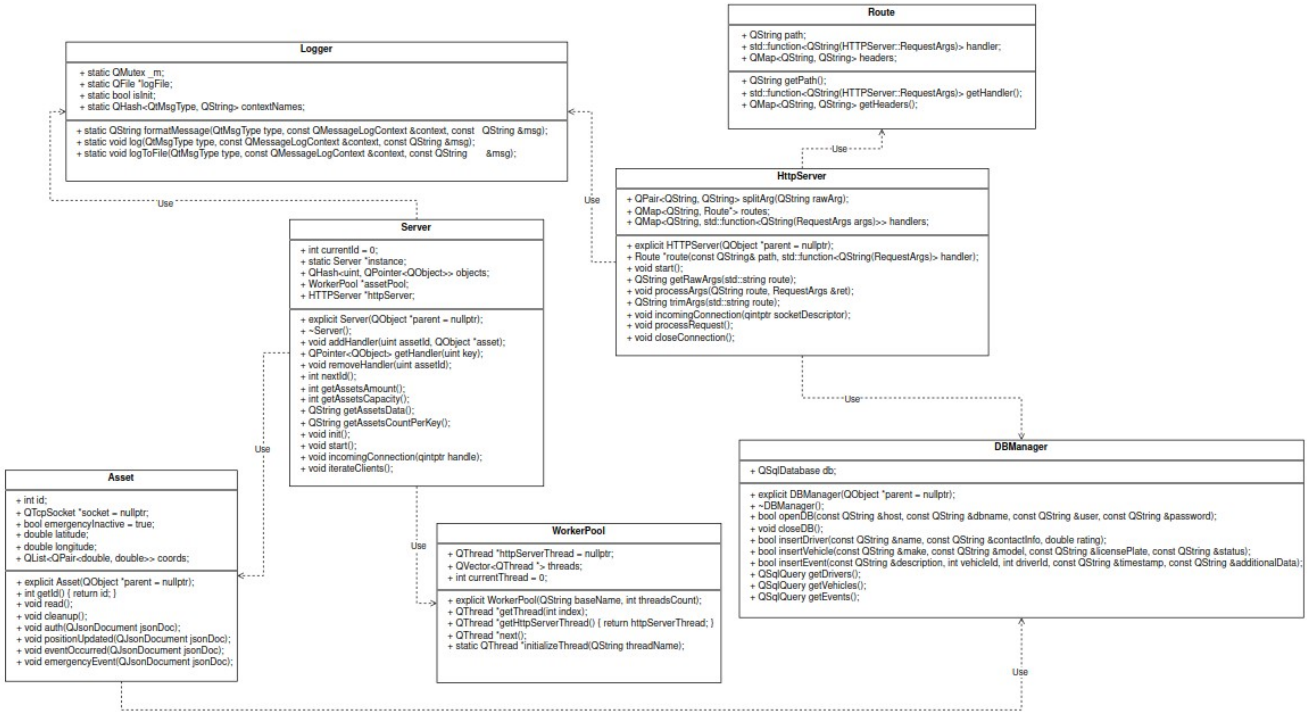

Рисунок 3.9 – UML діаграма класів серверної частини

3.5.2 Створення інтерфейсу менеджера транспортного флоту

Веб-застосунок, що буде виступати в ролі front-end частини системи моніторингу транспортних перевезень повинен мати наступні сторінки: index (основна сторінка з відображенням мапи), drivers (водії), vehicles (транспортні засоби) та events (події). Дані для кожної сторінки будуть отримуватись з розробленого раніше HTTP серверу в форматі JSON та вбудовуватись в сторінку. Допоможуть в цьому методи jQuerry, а саме – $.post (для відправки асинхронних запитів без необхідності перезавантаження сторінки) та селектор елементів $(“”) для зручної вбудови отриманого наповнення в DOM сторінки.

На кожній сторінці веб-застосунку є спільний header, який включає логотип системи та навігаційне меню. Натискання на логотип дозволяє перейти на основну сторінку з картою. Навігаційне меню містить посилання на всі ключові сторінки: "Водії", "Транспортні засоби" та "Події".

Основна сторінка веб-застосунку містить інтерактивну карту, створену за допомогою бібліотеки Leaflet. Карта відображає всі транспортні засоби, що перебувають у мережі, у вигляді маркерів. Крім цього, на карті показуються маршрути, якими рухаються транспортні засоби. Користувач може збільшувати або зменшувати масштаб карти, переміщувати її, а також натискати на маркери для отримання додаткової інформації про транспортні засоби.

З лівого боку сторінки розташований сайдбар, який показує список транспортних засобів, що перебувають у мережі. Кожен запис у списку містить базову інформацію про транспортний засіб та поточний статус. Натискання на транспортний засіб у списку дозволяє швидко знайти його на карті. Вигляд сторінки з мапою зображено на рис. 3.10

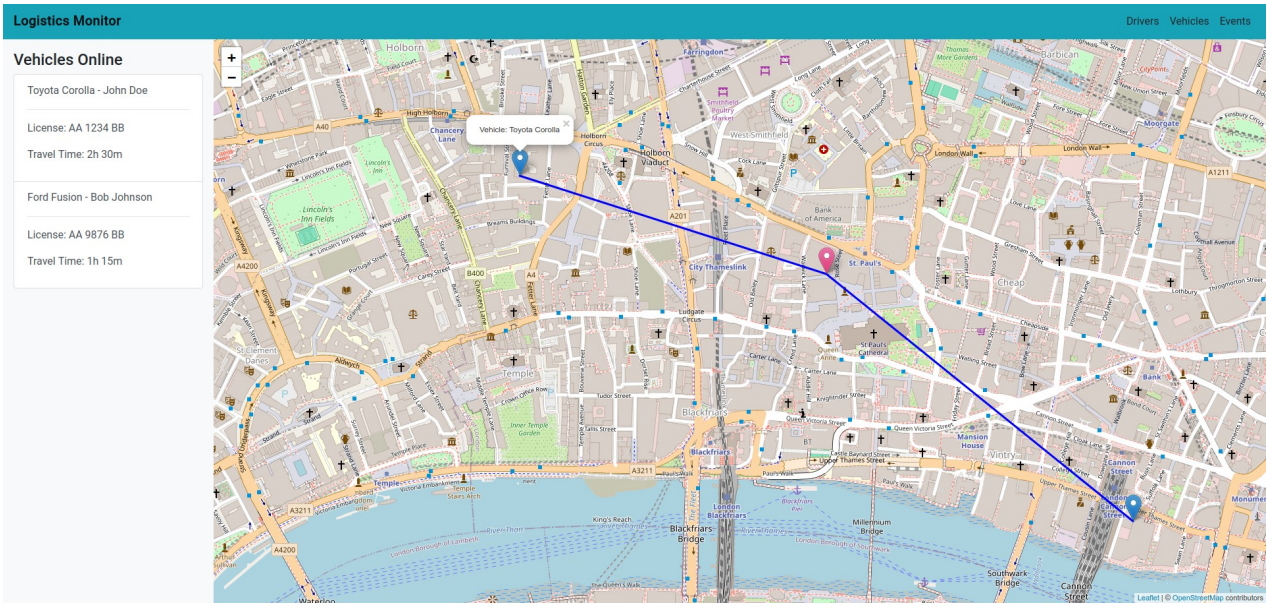

Рисунок 3.10 – Основна сторінка веб-застосунку

Сторінка водіїв призначена для відображення детальної інформації про всіх водіїв, що використовують систему. Вона містить таблицю, яка включає такі поля, як ім'я водія, контактну інформацію, рейтинг та інші релевантні дані. Таблиця забезпечує можливість сортування та фільтрації даних, що дозволяє швидко знаходити потрібну інформацію.

Крім того, є функція пошуку, яка дозволяє користувачам знаходити водіїв за ім'ям або іншими параметрами. Це робить управління великим списком водіїв більш зручним та ефективним. Сторінка “Водії” зображена на рис. 3.11

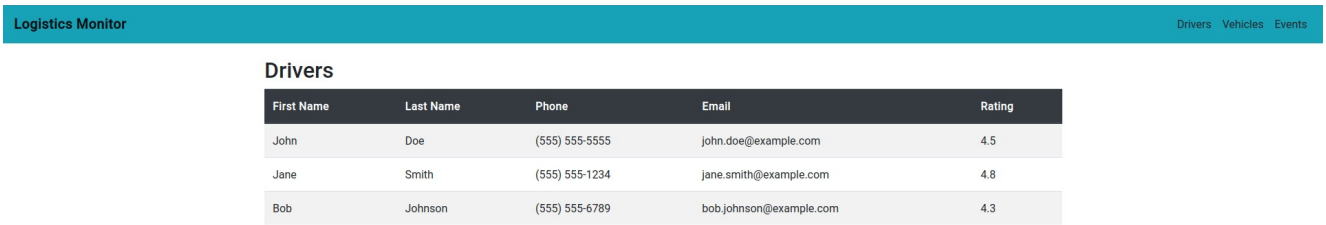

Рисунок 3.11 – Сторінка Drivers

Сторінка транспортних засобів відображає таблицю з інформацією про всі транспортні засоби, зареєстровані в системі. Поля таблиці включають марку та модель транспортного засобу, номерний знак, поточний статус та інші важливі дані. Як і на сторінці водіїв, тут також реалізовано можливість сортування, фільтрації та пошуку.

Така організація даних дозволяє легко управляти парком транспортних засобів, відстежувати їх стан та швидко отримувати необхідну інформацію. Сторінка “Транспортні засоби” зображена на рис. 3.12

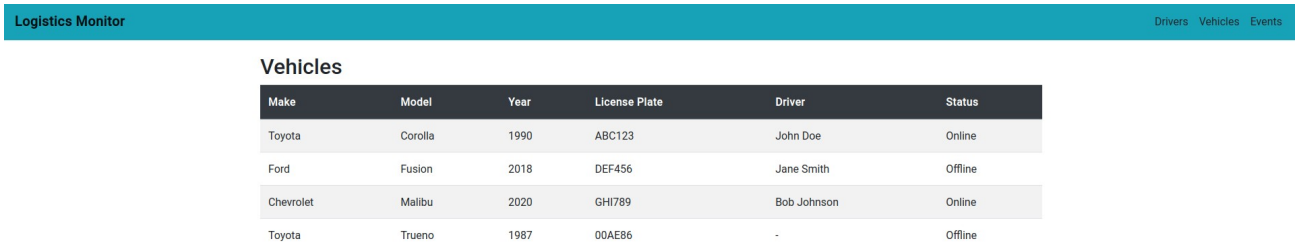

Рисунок 3.12 – Сторінка Vehicles

Сторінка подій відображає таблицю з усіма подіями, що відбулися під час перевезень. Кожен запис у таблиці містить дату та час події, опис події, пов'язаний транспортний засіб та водія, а також будь-які додаткові дані, такі як фотографії або відео, якщо такі є.

Ця сторінка дозволяє детально аналізувати всі інциденти, що сталися під час перевезень, і забезпечує повну прозорість процесу моніторингу транспортних засобів. Сторінка “Події” зображена на рис. 3.13

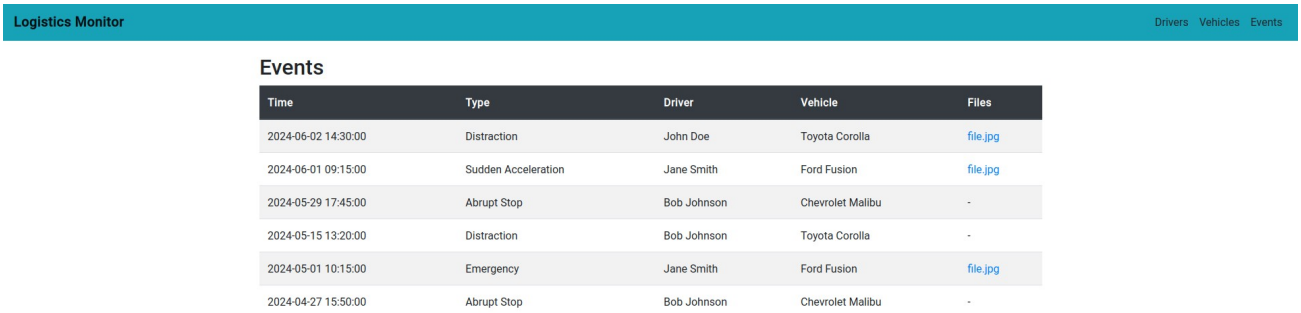

Рисунок 3.13 – Сторінка Events

3.5.3 Розробка мобільного застосунку водія

Для повноцінної системи моніторингу є необхідним також додаток водія. Як і у випадку серверної частини для реалізації буде використано фреймворк Qt. Графічний інтерфейс у свою чергу розроблено на мові QML даного фреймворку.

Перед користуванням додатком водію необхідно авторизуватись в системі ввівши свій логін та пароль. Одразу після авторизації пристрій змінює світ статус в системі на "онлайн" та починає передавати координати свого місцезнаходження. Ці зміни можна буде спостерігати у раніше розробленому веб-застосунку менеджера транспортного флоту. На головній сторінці можна побачити інформацію про водія та поїздку, а також – набір кнопок. Після натискання "Почати поїздку" серверна частина починає зберігати координати транспортного засобу. По ходу руху пари координат формують лінію маршруту, що наглядно відображається у веб-застосунку. Описаний додаток зображено на рис. 3.15.

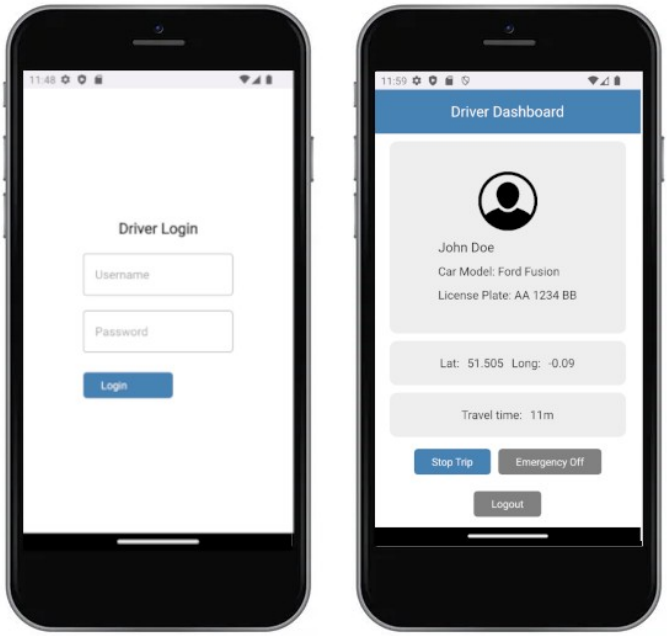

Рисунок 3.15 – Вигляд додатку водія

На макеті користувацького інтерфейсу можна також побачити кнопку "Emergency", що вмикає, так званий, аварійний режим. Така функція у додатку водія дозволяє швидко повідомити про надзвичайну ситуацію. Водій може активувати її одним натисканням у разі:

- Дорожньо-транспортної пригоди (ДТП)
- Нападу або загрози безпеці
- Технічної несправності автомобіля
- Інших екстрених ситуацій, які вимагають швидкого реагування

При ввімкненні аварійного режиму статус транспортного засобу на робочій станції менеджера змінюється.

Під час розробки веб-застосунку логістичної системи була створена спеціальна сторінка "Events" (Події), де відображаються всі важливі інциденти, включаючи активацію аварійного режиму. Якщо водій натискає кнопку "Emergency" у мобільному додатку телефон, закріплений на лобовому склі авто, миттєво робить фото передньою та задньою камерою та надсилає файли на

сервер. Таким чином диспетчер або служба безпеки миттєво отримує інформацію та може вжити заходів. Також у разі ДТП, нападу чи конфліктної ситуації фото можуть стати важливими доказами. Більше того, збережені події дозволяють аналізувати небезпечні зони, поведінку водіїв та потенційні ризики. Лістинг мобільного застосунку водія наведено у Додатку В.

У рамках розробки мобільного застосунку водія було також реалізовано розпізнавання стану та дій водія за допомогою камери смартфону. Цей компонент є ключовим елементом моніторингу безпеки транспортних перевезень, оскільки дозволяє в реальному часі відстежувати поведінку водія та оперативно виявляти потенційно небезпечні ситуації.

Відслідковування стану розпочинається автоматично після натискання кнопки "Почати поїздку". З цього моменту активується камера пристрою та запускаються алгоритми комп’ютерного зору, які аналізують обличчя водія та визначають його поточний стан. У процесі розпізнавання відбувається оцінка таких параметрів, як ступінь відкритості очей, положення голови, напрям погляду та інші характерні ознаки, що дозволяють виявляти ознаки сонливості, відволікання або втрати контролю.

Для забезпечення стабільності роботи та високої точності був використаний раніше описаний метод розпізнавання дій та стану водія, оптимізований для роботи на смартфонах. Застосунок обробляє відеопотік локально, без передачі зображення на сервер. У випадку виявлення небезпечної ситуації застосунок повідомить водія звуковим сигналом. Дані передаються на сервер у вигляді подій, які фіксують зміни стану водія або виявлені порушення. Це дозволяє менеджеру транспортного флоту своєчасно отримувати сповіщення та реагувати на потенційно небезпечні ситуації. Алгоритм розпізнавання було протестовано як на зображеннях з Інтернету так і реальному смартфоні. Приклад тестування методу розпізнавання дій та стану водія на зображеннях з Інтернету зображено на рис. 3.16.

Рисунок 3.16 – Робота методу розпізнавання на тестових зображеннях

3.6 Тестування розробленої системи

Після завершення розробки програмного забезпечення необхідним етапом є проведення тестування, яке дозволяє оцінити відповідність системи вимогам, визначити надійність її роботи та виявити можливі недоліки до впровадження. Представлено результати тестування розробленої системи, які охоплюють як перевірку правильності виконання функціональних можливостей, так і аналіз роботи системи під навантаженням. Проведення комплексного тестування гарантує стабільність, продуктивність та готовність системи до використання в реальних умовах.

3.6.1 Функціональне тестування

Функціональне тестування є критично важливим етапом розробки будь-якого програмного забезпечення, оскільки воно забезпечує перевірку коректності роботи всіх компонентів та взаємодій. У процесі розробки системи моніторингу

транспортних перевезень було здійснено ретельне тестування базових функцій, що включає обмін даними між пристроєм та сервером. Це тестування охоплювало декілька аспектів, таких як авторизація пристрою, передача координат, обробка подій.

Зокрема, було перевірено, як пристрої-клієнти підключаються до TCP сервера, проходять авторизацію та періодично передають координати транспортного засобу. Приклад надходження до сервера даних від пристрою зображено на рис. 3.17

```
MainThread | 20:13:07.758 | Debug | ../Server/httpserver.cpp | 27 | start | HTTP server started at 6002
MainThread | 20:13:07.758 | Debug | ../Server/server.cpp | 191 | start | Server started at 6001
AssetThread_1 | 20:13:09.685 | Debug | ../Server/asset.cpp | 16 | init | 0 connected
AssetThread_1 | 20:13:10.546 | Debug | ../Server/asset.cpp | 142 | read | 0 "{\"method\": \"positionUpdated\", \"lat\":  37.4221, \"lng\": -122.084}"
AssetThread_1 | 20:13:10.547 | Debug | ../Server/asset.cpp | 58 | isValidJson | json valid
AssetThread_1 | 20:13:10.547 | Debug | ../Server/asset.cpp | 69 | processJson | processJson
AssetThread_1 | 20:13:10.547 | Debug | ../Server/asset.cpp | 93 | positionUpdated | position updated
AssetThread_1 | 20:13:14.150 | Debug | ../Server/asset.cpp | 142 | read | 0 "{\"method\": \"positionUpdated\", \"lat\":  37.2999, \"lng\": -121.801}"
AssetThread_1 | 20:13:14.150 | Debug | ../Server/asset.cpp | 58 | isValidJson | json valid
AssetThread_1 | 20:13:14.150 | Debug | ../Server/asset.cpp | 69 | processJson | processJson
AssetThread_1 | 20:13:14.150 | Debug | ../Server/asset.cpp | 93 | positionUpdated | position updated
AssetThread_1 | 20:13:34.015 | Debug | ../Server/asset.cpp | 142 | read | 0 "{\"method\": \"positionUpdated\", \"lat\":  37.2802, \"lng\": -121.808}"
AssetThread_1 | 20:13:34.015 | Debug | ../Server/asset.cpp | 58 | isValidJson | json valid
AssetThread_1 | 20:13:34.015 | Debug | ../Server/asset.cpp | 69 | processJson | processJson
AssetThread_1 | 20:13:34.015 | Debug | ../Server/asset.cpp | 93 | positionUpdated | position updated
AssetThread_1 | 20:13:35.954 | Debug | ../Server/asset.cpp | 142 | read | 0 "{\"method\": \"positionUpdated\", \"lat\":  37.2806, \"lng\": -121.808}"
AssetThread_1 | 20:13:35.955 | Debug | ../Server/asset.cpp | 58 | isValidJson | json valid
AssetThread_1 | 20:13:35.955 | Debug | ../Server/asset.cpp | 69 | processJson | processJson
AssetThread_1 | 20:13:35.955 | Debug | ../Server/asset.cpp | 93 | positionUpdated | position updated
AssetThread_1 | 20:13:38.140 | Debug | ../Server/asset.cpp | 142 | read | 0 "{\"method\": \"positionUpdated\", \"lat\":  37.2808, \"lng\": -121.809}"
AssetThread_1 | 20:13:38.140 | Debug | ../Server/asset.cpp | 58 | isValidJson | json valid
AssetThread_1 | 20:13:38.140 | Debug | ../Server/asset.cpp | 69 | processJson | processJson
AssetThread_1 | 20:13:38.140 | Debug | ../Server/asset.cpp | 93 | positionUpdated | position updated
AssetThread_1 | 20:13:40.031 | Debug | ../Server/asset.cpp | 142 | read | 0 "{\"method\": \"positionUpdated\", \"lat\":  37.2813, \"lng\": -121.809}"
AssetThread_1 | 20:13:40.032 | Debug | ../Server/asset.cpp | 58 | isValidJson | json valid
AssetThread_1 | 20:13:40.032 | Debug | ../Server/asset.cpp | 69 | processJson | processJson
AssetThread_1 | 20:13:40.032 | Debug | ../Server/asset.cpp | 93 | positionUpdated | position updated
AssetThread_1 | 20:13:41.980 | Debug | ../Server/asset.cpp | 142 | read | 0 "{\"method\": \"positionUpdated\", \"lat\":  37.2813, \"lng\": -121.81}"
AssetThread_1 | 20:13:41.980 | Debug | ../Server/asset.cpp | 58 | isValidJson | json valid
AssetThread_1 | 20:13:41.981 | Debug | ../Server/asset.cpp | 69 | processJson | processJson
AssetThread_1 | 20:13:41.981 | Debug | ../Server/asset.cpp | 93 | positionUpdated | position updated
AssetThread_1 | 20:13:43.978 | Debug | ../Server/asset.cpp | 142 | read | 0 "{\"method\": \"positionUpdated\", \"lat\":  37.2815, \"lng\": -121.81}"
AssetThread_1 | 20:13:43.978 | Debug | ../Server/asset.cpp | 58 | isValidJson | json valid
AssetThread_1 | 20:13:43.978 | Debug | ../Server/asset.cpp | 69 | processJson | processJson
AssetThread_1 | 20:13:43.978 | Debug | ../Server/asset.cpp | 93 | positionUpdated | position updated
AssetThread_1 | 20:13:46.087 | Debug | ../Server/asset.cpp | 142 | read | 0 "{\"method\": \"positionUpdated\", \"lat\":  37.2817, \"lng\": -121.811}"
AssetThread_1 | 20:13:46.087 | Debug | ../Server/asset.cpp | 58 | isValidJson | json valid
```

Рисунок 3.17 – Обробка сервером даних від пристрою-клієнта

3.6.2 Тестування навантаження

Окрім функціонального тестування, для системи моніторингу транспортних перевезень дуже важливою є перевірка її здатності витримувати високе навантаження. Тестування навантаження дозволяє визначити, наскільки ефективно система може працювати під час пікових навантажень, коли одночасно

підключено багато пристроїв-клієнтів, і як вона справляється з великим обсягом даних, що передаються.

Тестування навантаження включає моделювання ситуацій, коли система перебуває під інтенсивним використанням. Це можуть бути сценарії, коли десятки або навіть стоні пристроїв одночасно передають дані до сервера. В таких умовах важливо перевірити, чи здатний сервер обробляти всі запити без затримок, чи не виникають помилки або втрати даних. Для цього використовуються спеціальні інструменти, що дозволяють симулювати велику кількість одночасних з'єднань та вимірювати час відповіді сервера, кількість успішно оброблених запитів та інші параметри продуктивності.

Під час тестування навантаження також аналізуються ресурси сервера, такі як використання процесора, оперативної пам'яті та мережевого трафіку. Це дозволяє виявити потенційні вузькі місця у системі та оптимізувати її конфігурацію для забезпечення більшої ефективності. Наприклад, виявлення проблем із перевантаженням процесора може призвести до оптимізації коду або додавання додаткових серверів для балансування навантаження.

Тестування навантаження розробленої системи включало в себе передачу даних до серверу від великої кількості клієнтів та показало, що вона здатна підтримувати кілька десятків підключених пристроїв одночасно без зниження продуктивності. Варто зауважити – клас WorkerPool, що реалізує багатопоточність серверної частини може бути налаштований на більшу кількість потоків за необхідності. Це може допомагати ефективно розвантажувати завдання між потоками, підвищуючи швидкодію системи. Приклад роботи сервера з великою кількістю клієнтів одночасно зображено на рис. 3.18

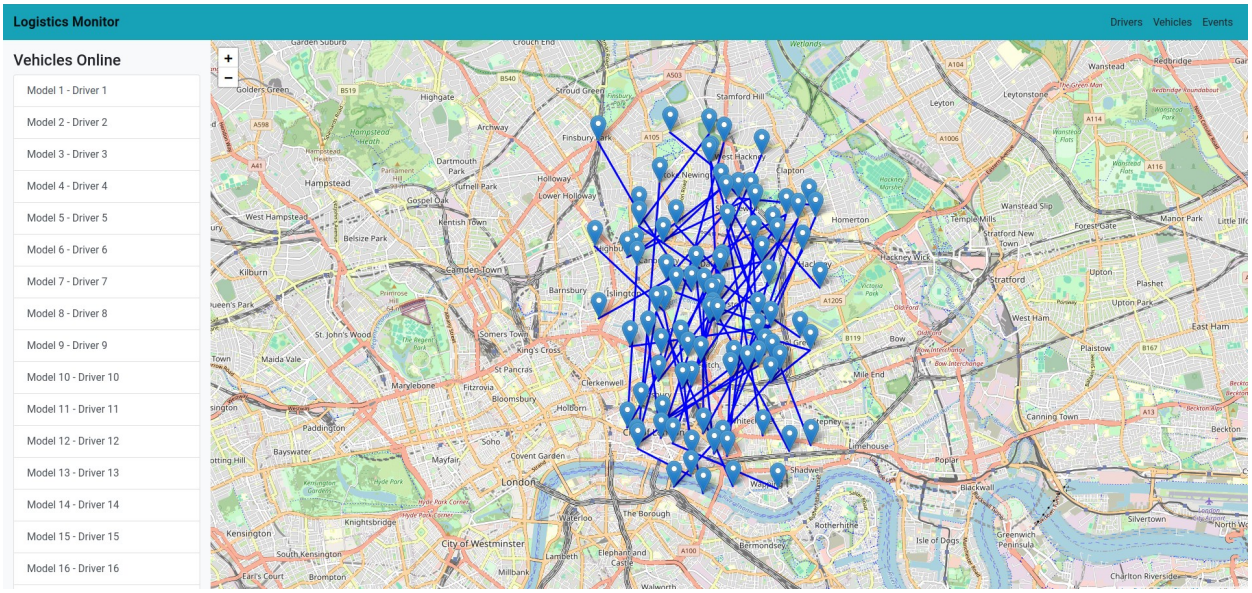

Рисунок 3.18 – Тестування навантаження системи

3.7 Висновки до розділу

У даному розділі було розроблено архітектуру системи, що включає TCP сервер для обміну даними з клієнтами через сокети, API-сервер для приймання файлів та надання API, а також взаємодію з базою даних для збереження всієї інформації. Створено структуру бази даних, що забезпечує ефективне зберігання та доступ до даних про координати транспортних засобів, інформацію про події та файли.

Був також розроблений застосунок для пристрою-клієнта, який підключається до TCP сервера, авторизується в системі та передає координати транспортного засобу та інформацію про події. При необхідності, застосунок також надсилає фото або відео файл, що надає детальну інформацію про подію до якої він прикріплений, до API-сервера. Застосунок забезпечує постійний моніторинг та передачу даних у режимі реального часу. Розроблена front-end частина для візуалізації та роботи з даними, отриманими через API. Вона забезпечує зручний інтерфейс для перегляду та аналізу координат, подій та

файлів, що надходять від пристроїв-клієнтів, а також забезпечує доступність даних для користувачів системи.

Проведено комплексне тестування розробленої системи, що включало в себе тестування продуктивності серверної частини та функціоналу пристрою-клієнта. Тестування продуктивності серверної частини показало, що система здатна обробляти великий обсяг запитів та даних у режимі реального часу без втрат продуктивності. Тестування функціоналу пристрою-клієнта підтвердило коректність авторизації, передачі координат, подій та файлів.

4 ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Технологічний аудит розробленої системи транспортних перевезень з функцією інтелектуального моніторингу стану водія

Як було зазначено раніше, одним із ключових аспектів ефективної організації транспортних перевезень є своєчасне отримання достовірної інформації про місцезнаходження транспортних засобів і стан водіїв. У сучасних умовах, коли логістичні компанії обслуговують значні обсяги перевезень, а вимоги до безпеки дорожнього руху постійно зростають, потреба в оперативному контролі за транспортом стає особливо актуальною. Це пов’язано як із необхідністю оптимізувати маршрути й підвищувати ефективність використання автопарку, так і з потребою мінімізувати ризики, пов’язані з втомою, неуважністю чи іншими небезпечними станами водіїв. Широке впровадження цифрових технологій у сфері логістики та транспортних послуг призвело до того, що більшість компаній орієнтуються на системи, здатні забезпечити комплексний моніторинг транспорту в режимі реального часу.

У зв’язку з цим метою даної роботи стала розробка програмної системи, яка забезпечує безперервний обмін даними між бортовим пристроєм транспортного засобу та серверною частиною, зберігання інформації про координати, події та стан водія, а також можливість приймання додаткових медіафайлів через окремий HTTP-канал.

У процесі розробки було створено серверну частину системи, що працює на Ubuntu Linux та реалізує TCP-взаємодію з клієнтами і приймання файлів через HTTP-інтерфейс, а також клієнтський застосунок для Android пристрою, який виконує авторизацію, періодично передає координати, фіксує події та надсилає відповідні дані до серверу. Розроблена система забезпечує надійний прийом і зберігання інформації в реляційній базі даних PostgreSQL, а також дозволяє легко

масштабувати та розширювати функціональність без значного ускладнення архітектури.

Запропоноване рішення покращує контроль за транспортними перевезеннями та підвищує безпеку як водія, так і вантажу. Система є гнучкою, розширюваною та може бути адаптована для різних сценаріїв використання – від комерційних логістичних компаній до муніципальних транспортних служб, що підтверджує її практичну цінність та доцільність впровадження.

Для встановлення комерційного потенціалу розробленої нами системи транспортних перевезень з функцією інтелектуального моніторингу стану водія було запрошено 3-х відомих експертів – кандидатку технічних наук, доценткиню Барабан М.В., кандидата технічних наук, доцента Кабачія В.В. та кандидатку технічних наук, доценткиню Богач І.В.

Встановлення комерційного потенціалу розробленої нами системи здійснювалося за загально визнаними критеріями, які наведені в таблиці 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу будь-якої розробки і їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у

		комплексі			виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Запрошені експерти оцінили розроблену нами систему таким чином (див. таблицю 4.2):

Таблиця 4.2 – Результати технологічного аудиту розробленої нами системи транспортних перевезень з функцією інтелектуального моніторингу стану водія(за шкалою оцінювання 0-1-2-3-4)

Критерії	Прізвище, ініціали експертів		
	Кабачій В.В.	Барабан М.В.	Богач І.В.
	Бали, що їх виставили експерти:		
1	3	3	3
2	2	2	2
3	4	4	4
4	3	2	3
5	4	3	3

6	2	3	2
7	2	3	2
8	4	3	3
9	3	2	3
10	3	4	3
11	3	2	2
12	3	3	4
Сума балів	$СБ_1 = 36$	$СБ_2 = 34$	$СБ_3 = 34$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{36+34+34}{3} = 34.6$		

Встановлення комерційного потенціалу розробленої нами системи будемо здійснювати на основі рекомендацій, наведених в таблиці 4.3 [41].

Таблиця 4.3 – Рівні комерційного потенціалу будь-якої наукової розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Оскільки середньоарифметична сума балів, що їх виставили експерти, складає 34 бали, то це свідчить про те, що нами система транспортних перевезень з функцією інтелектуального моніторингу стану водія має рівень комерційного потенціалу, який вважається «вище середнього».

Це пояснюється тим, що розроблена нами система має значно кращі експлуатаційні показники і може бути використана для точного моніторингу стану та дій водія під час здійснення перевезення.

4.2 Розрахунок витрат на розроблення системи транспортних перевезень з функцією інтелектуального моніторингу стану водія

При розробці системи транспортних перевезень з функцією інтелектуального моніторингу стану водія були зроблені певні витрати. Зокрема:

А). Основна заробітна плата $З_о$ розробників, яка визначається за формулою:

$$З_о = \frac{М}{Т_р} \cdot t \text{ грн}, \quad (4.1)$$

де М – місячний посадовий оклад розробника, грн; приймемо, що

М = (8000…35000) грн/місяць;

$Т_р$ – число робочих днів в місяці; приймемо $Т_р$ = 20 днів;

t – число днів роботи розробників.

Зроблені розрахунки зведемо до таблиці 4.4:

Таблиця 4.4 – Основна заробітна плата розробників

Найменування посади виконавця	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на оплату праці, грн
1. Науковий керівник магістерської роботи	25000	1250	20 годин	≈ 4167
2. Магістрант-студент-виконавець	Беремо мінімальну оплату 8000 грн	400	87	≈ 34800
3. Консультант з економічної частини	20000	1000	1,5 години	≈ 250 (при 6-годинному робочому дні)
Загалом				$З_о$ = 39217 грн

Б). Додаткова заробітна плата $З_д$ розробників розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = \alpha \cdot З_о = (0,1…0,12) \cdot З_о. \quad (4.2)$$

Приймемо, що α = 0,1. Тоді для нашого випадку отримаємо:

$З_д$ = 0,1 × 39217 = 3921,7 ≈ 3922 грн.

В). Нарахування на заробітну плату $НЗП_{зп}$ розробників (дослідників) розраховуються за формулою:

$$НЗП_{зп}=(З_о+З_д)\cdot\frac{\beta}{100}, \quad (4.3)$$

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування, %. β = 22%. Тоді:

$$НЗН_{зп} = (39217 + 3922) \times 0,22 = 9490,58 \approx 9491 \text{ грн.}$$

Г). Амортизація основних засобів А, які використовувались під час виконання цієї роботи:

$$A=\frac{Ц\cdot Н_а}{100}\cdot\frac{Т}{12} \text{ грн}, \quad (4.4)$$

де Ц – загальна балансова вартість основних засобів, грн;

$Н_а$ – річна норма амортизаційних відрахувань. Для нашого випадку можна прийняти, що $Н_а$ = (2,5...25)%;

Т – термін використання основних засобів, місяці.

Зроблені розрахунки зведено в таблицю 4.5.

Таблиця 4.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, місяці	Величина амортизаційних відрахувань, грн
1. Комп'ютерна техніка, обладнання, принтери тощо	64000	25	3,5 (при 80% використанні	3733,33 ≈ 3734
2. Приміщення університету, кафедри	26000	3	3,5 при 40% використанні	91
Всього				А = 3825 грн

Д). Витрати на матеріали М розраховуються за формулою:

$$M=\sum_1^n Н_i\cdot Ц_i\cdot К_i-\sum_1^n В_i\cdot Ц_в \text{ грн.}, \quad (4.5)$$

де $Н_і$ – витрати матеріалу *і*-го найменування, кг; $Ц_і$ – вартість матеріалу *і*-го най-менування; $К_і$ – коефіцієнт транспортних витрат, $К_і$ = (1,1…1,15); $В_і$ – маса відходів матеріалу *і*-го найменування; $Ц_в$ – ціна відходів матеріалу *і*-го найменування; n – кількість видів матеріалів.

Е). Витрати на комплектуючі К розраховуються за формулою:

$$K = \sum_{1}^{n} H_i \cdot Ц_i \cdot K_i \text{ грн}, \quad (4.6)$$

де $Н_і$ – кількість комплектуючих *і*-го виду, шт.; $Ц_і$ – ціна комплектуючих *і*-го виду; $К_і$ – коефіцієнт транспортних витрат, $К_і$ = (1,1…1,15); n – кількість видів комплектуючих.

Під час виконання магістерської кваліфікаційної роботи загальні витрати на матеріали та комплектуючі склали приблизно 1450 грн.

Ж). Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$B_e = \frac{B \cdot П \cdot Ф \cdot K_n}{K_д}, \quad (4.7)$$

де В – вартість 1 кВт-год. електроенергії, в 2025 р. В ≈ 4,5 грн/кВт;

П – установлена потужність обладнання, кВт; П = 1,2 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

Приймемо, що Ф = 280 годин;

$К_п$ – коефіцієнт використання потужності; $К_п$ < 1 = 0,85.

$К_д$ – коефіцієнт корисної дії, $К_д$ = 0,8.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$B_c = \frac{B \cdot П \cdot Ф \cdot K_n}{K_д} = \frac{4,5 \cdot 1,2 \cdot 280 \cdot 0,85}{0,8} = 1606,5 \approx 1607 \text{ грн.}$$

И). Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати розробників, тобто:

$$В_{інш} = (0,5....3) \times З_о. \quad (4.8)$$

Для нашого випадку отримаємо:

$$В_{інш} = 1,2 \times 39217 = 47060,4 \approx 47061 \text{ грн.}$$

К). Сума всіх попередніх статей витрат складає витрати на виконання роботи безпосередньо розробником-магістрантом – В.

В = 39217+ 3922 + 9491+ 3825 + 1450 +1607 + 47061 = 106573 грн.

Л). Загальні витрати на розробку системи $В_{заг}$ розраховуються за формулою:

$$B_{заг} = \frac{B}{\beta}, \quad (4.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи. Можна прийняти, що, $\beta \approx 0{,}9$ [41].

Тоді: $B_{заг} = \frac{106573}{0{,}9} = 118414{,}44$ грн або приблизно 119 тисяч грн.

Тобто прогнозовані загальні витрати на розробку системи транспортних перевезень з функцією інтелектуального моніторингу стану водія становлять приблизно 119 тисяч грн.

4.3 Розрахунок економічного ефекту від можливої комерціалізації нашої розробки

Економічний ефект від впровадження та можливої комерціалізації розробленої нами системи транспортних перевезень з функцією інтелектуального моніторингу стану водія пояснюється її значно кращими функціональними можливостями. Тому нашу розробку можна реалізовувати на ринку дещо дорожче, ніж аналогічні за функціями розробки. Так, якщо подібна за функціями розробка коштує на ринку приблизно 110 тисяч грн, то нашу розробку можна буде реалізовувати на ринку приблизно за 140 тисяч грн або на 30 тисяч грн дорожче.

Аналіз місткості ринку показує, що на сьогодні в Україні кількість потенційних споживачів розробленої нами системи може становити приблизно 90 фізичниї та юридичрих осіб. Це, насамперед, компанії-перевізники, служби логістики та доставки, таксі-парки, автотранспортні підприємства, муніципальні

служби громадського транспорту, служби безпеки та страхові компанії, що потребують контролю місцезнаходження, стану та поведінки водіїв під час руху. Тому можна очікувати зростання попиту на нашу розробку принаймні протягом 3-х років після її впровадження.

Тобто, якщо наша розробка буде впроваджена з 1 січня 2026 року, то її результати будуть виявлятися протягом 2026-го, 2027-го та 2028-го років.

Прогноз зростання попиту на нашу розробку складає по роках:

а) 2026 р. – приблизно +15 шт. до базового року;

б) 2027 р. – +20 шт. до базового року;

в) 2028 р. – +25 шт. до базового року.

Можливе збільшення чистого прибутку $\Delta П_i$, що його може отримати потенційний інвестор від комерціалізації нашої розробки, тобто виведення нашої розробки на ринок, становитиме:

$$\Delta П_i = ¿ \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (4.10)$$

де $\Delta Ц_o$ – покращення основного якісного показника від впровадження результатів нашої розробки у цьому році. Для нашого випадку це є збільшення ціни реалізації нашої розробки $\Delta Ц_o$ = (140 – 110) = + 30 тисяч грн;

N – основний кількісний показник, який визначає обсяг діяльності у році до впровадження результатів розробки; N = 90 шт.;

ΔN – покращення основного кількісного показника від впровадження результатів розробки. Таке покращення становитиме по роках, відповідно: у 2026 році – + 15 шт., у 2027 році +20 шт, та у 2028 році + 25 шт. (відносно базового 2025 року);

$Ц_o$ – основний якісний показник (тобто ціна), який визначає обсяг діяльності у році після впровадження результатів розробки, грн; $Ц_o$= 140 тисяч грн;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки; для нашого випадку n = 3;

λ – коефіцієнт, який враховує сплату податку на додану вартість; $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати ρ= (0,2...0,5); візьмемо ρ= 0,4;

υ – ставка податку на прибуток. У 2025 υ = 18%. У 2026 році і в наступних роках також очікуємо ставку υ = 18%.

Тоді можливе зростання чистого прибутку ΔП1 для потенційного інвестора протягом першого року від можливого впровадження (комерціалізації) нашої розробки (2026 р.) складе:

$$\Delta П_1 = [30 \cdot 90 + 140 \cdot 15] \cdot 0{,}8333 \cdot 0{,}4 \cdot \left(1 - \frac{18}{100}\right) = 1312 \text{ тис. грн.}$$

Можливе зростання чистого прибутку ΔП2 для потенційного інвестора від можливого впровадження (комерціалізації) нашої розробки протягом другого (2027 р.) року складе:

$$\Delta П_2 = [30 \cdot 90 + 140 \cdot 20] \cdot 0{,}8333 \cdot 0{,}4 \cdot \left(1 - \frac{18}{100}\right) = 1503 \text{ тис. грн.}$$

Можливе зростання чистого прибутку ΔП3 для потенційного інвестора від можливого впровадження (комерціалізації) нашої розробки протягом третього (2028 р.) року становитиме:

$$\Delta П_2 = [30 \cdot 90 + 140 \cdot 25] \cdot 0{,}8333 \cdot 0{,}4 \cdot \left(1 - \frac{18}{100}\right) = 1695 \text{ тис. грн.}$$

Приведена вартість зростання всіх чистих прибутків потенційного інвестора від можливого впровадження нашої розробки становитиме:

$$ПП = \sum_{1}^{m} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої роботи, грн;

т – період часу, протягом якого виявляються результати впровадженої роботи, роки. Для нашого випадку т = 3 роки;

τ – ставка дисконтування. Приймемо τ = 0,10 (10%);

t – період часу від моменту початку розроблення системи моніторингу до моменту отримання можливих чистих прибутків потенційним інвестором.

Тоді приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від комерціалізації нашої розробки, складе:

$$ПП^{10\%} = \frac{1312}{(1+0,1)^2} + \frac{1503}{(1+0,1)^3} + \frac{1695}{(1+0,1)^4} \approx 1084 + 1129 + 1158 = 3371 \text{ тисяч грн.}$$

Теперішня вартість інвестицій PV, що можуть бути вкладені для реалізації нашої розробки: PV = (1,0…5) × Взаг.

Для нашого випадку приймемо:

PV = (1,0…5) × 90 = 5 × 90 = 450 тисяч грн.

Розраховуємо абсолютний ефект від можливих вкладених інвестицій Еабс.

$$Е_{абс} = ПП – PV, \quad (4.12)$$

де ПП – приведена вартість збільшення всіх чистих прибутків, що їх може отримати потенційний інвестор від можливого впровадження нашої розробки, грн;

PV – теперішня вартість інвестицій PV = 450 тисяч грн.

Абсолютний ефект від можливого впровадження нашої розробки (при прогнозованому ринку збуту) за три роки складе:

$$Е_{абс} = 3371 – 450 = 2921 \text{ тисяч грн.}$$

Оскільки $Е_{абс} > 0$, то комерціалізація нашої розробки може бути доцільною.

Далі розрахуємо внутрішню дохідність $Е_в$ вкладених інвестицій:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.13)$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс}$ = 2921 тис. грн;

PV –теперішня вартість початкових інвестицій PV = 450 тис. грн;

$Т_ж$ – життєвий цикл розробки, роки.

$Т_ж$ = 4 років (2025-й, 2026-й, 2027-й, 2028-й роки)

Для нашого випадку отримаємо:

$$E_B = \sqrt[4]{\frac{2921}{450}} - 1 = 1,596 - 1 = 0,596 = 59,6\%$$

Далі визначимо ту мінімальну дохідність, нижче за яку потенційному інвестору не вигідно буде займатися комерціалізацією нашої розробки.

Мінімальна дохідність або мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = ¿ d + f, \quad (4.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d = (0,15...0,19);

f – показник, що характеризує ризикованість вкладень; f = (0,05...0,40).

Для нашого випадку отримаємо:

$$\tau_{мін} = 0,18 + 0,37 = 0,55 \text{ або } \tau_{мін} = 55\%.$$

Оскільки величина $E_в = 59,6\% > \tau_{мін} = 55\%$, то потенційний інвестор у принципі може бути зацікавлений у фінансуванні та комерціалізації розробленої нами системи.

Далі розраховуємо термін окупності коштів, вкладених у можливу комерціалізацію розробленої нами системи моніторингу.

Термін окупності $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}. \quad (4.15)$$

Для нашого випадку термін окупності $T_{ок}$ коштів (інвестицій) становитиме:

$$T_{ок} = \frac{1}{0,596} = 1,7 \text{ років} < 3 \text{ років},$$

що свідчить про потенційну доцільність комерціалізації розробленої нами системи моніторингу.

Далі проведено моделювання залежності величини внутрішньої дохідності вкладених потенційних інвестицій від рівня інфляції в країні (який в умовах війни, на жаль, може зростати).

Прийнявши рівень інфляції у 20%, отримаємо:

$$ПП^{10\%} = \frac{1312}{(1+0,2)^2} + \frac{1503}{(1+0,2)^3} + \frac{1695}{(1+0,2)^4} = 2598 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження нашої розробки за три роки складе:

$$Е_{абс} = 2598 – 450 = 2148 \text{ тисяч грн.}$$

Внутрішня дохідність $Е_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[T_ж]{1+\frac{E_{абс}}{PV}} - 1,$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс}$ = 2148 тисяч грн;

PV –теперішня вартість початкових інвестицій PV = 450 тисяч грн.

Для нашого випадку отримаємо:

$$E_B = \sqrt[4]{\frac{2148}{450}} - 1 = 1{,}478 - 1 = 0{,}478 = 47{,}8\%$$

Оскільки величина $Е_в = 47{,}8\% < \tau_{мін} = 55\%$, то потенційний інвестор у принципі може бути не зацікавлений у фінансуванні та комерціалізації розробленої нами системи.

Прийнявши рівень інфляції у 30%, отримаємо:

$$ПП^{10\%} = \frac{1312}{(1+0{,}3)^2} + \frac{1503}{(1+0{,}3)^3} + \frac{1695}{(1+0{,}3)^4} = 2054 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження нашої розробки за три роки складе:

$$Е_{абс} = 2054 – 450 = 1604 \text{ тисячі грн.}$$

Внутрішня дохідність $Е_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[T_ж]{1+\frac{E_{абс}}{PV}} - 1,$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс}$ = 1604 тисяч грн;

PV – теперішня вартість початкових інвестицій PV = 450 тисяч грн.

Для нашого випадку отримаємо:

$$E_B = \sqrt[4]{\frac{1604}{450}} - 1 = 1{,}374 - 1 = 0{,}374 = 37{,}4\%$$

Оскільки величина $Е_в = 37{,}4\% < \tau_{мін} = 55\%$, то потенційний інвестор ще більшою мірою може бути не зацікавлений у фінансуванні та комерціалізації розробленої нами системи.

Зроблені розрахунки у вигляді графіків наведено на рис. 4.1.

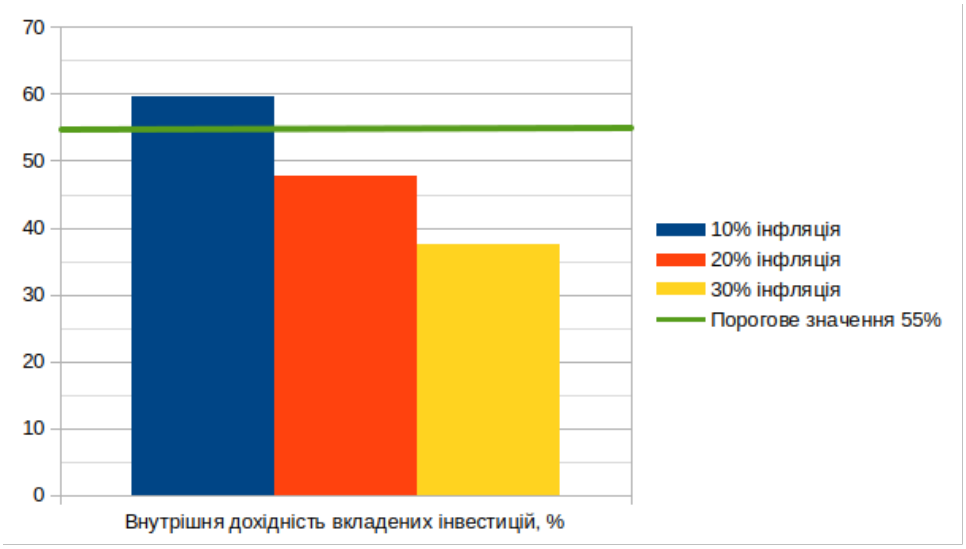

Рисунок 4.1 – Моделювання залежності величини внутрішньої дохідності потенційних інвестицій, вкладених у комерціалізацію розробленої нами системи моніторингу, від рівня інфляції в країні

Аналіз графіка на рис 4.1 показує, що при рівні інфляції в 10% величина внутрішньої дохідності потенційних інвестицій, вкладених у комерціалізацію розробленого нами системи моніторингу водія, може становити $E_в$ = 59,6%, що більше порогового значення $\tau_{мін}$ = 55%, і тому комерціалізація нашої розробки може бути доцільною. При рівні інфляції в 20%, 30% і вище величина внутрішньої дохідності інвестицій $E_в$, вкладених в комерціалізацію нашої розробки, буде меншою за $\tau_{мін}$ = 55%, і тому комерціалізація нашої розробки (при нинішніх зовнішніх умовах) може бути для потенційного інвестора не доцільною. Для прийняття остаточного рішення потрібно провести додаткові обгрунтування і розрахунки (наприклад, знизити рівень прийнятого ризику, підняти ціну реалізації нашої розробки тощо).

Результати виконаної економічної частини магістерської кваліфікаційної роботи зведено у таблицю 4.6:

Таблиця 4.6 – Результати виконаної економічної частини

Показники	Задані у ТЗ	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку системи	Не більше 125 тис. грн	119 тис. грн.	Практично досягнуто
2. Абсолютний ефект від впровадження розробки, тисяч грн	В межах 2800 тисяч грн (за три роки)	2921 тисяч грн	Практично виконано
3. Внутрішня дохідність потенційних інвестицій, %	не менше 55% (при 10% інфляції)	59,6%	Досягнуто
4. Термін окупності інвестицій, роки	до 3-ти років	1,7 років	Виконано

Таким чином, основні техніко-економічні показники розробленої нами системи транспортних перевезень з функцією інтелектуального моніторингу стану водія, визначені у технічному завданні, виконані.

ВИСНОВКИ

У ході виконання даної магістерської кваліфікаційної роботи було розроблено повноцінну автоматизовану системи моніторингу транспортних перевезень та стану водія. Розглянуто основні етапи розробки такої системи моніторингу.

Перший етап роботи включав в себе детальний аналіз предметної області логістичних перевезень. Були вивчені вже існуючі на ринку рішення та особливості їх реалізації.

Не менш важливим кроком був вибір відповідних технологій для розробки системи. Після ретельного аналізу можливих варіантів було вирішено використати мову програмування C++ з фреймворком Qt для серверної частини, що забезпечує високу продуктивність та багатопоточність. Для бази даних було обрано PostgreSQL завдяки її надійності та масштабованості. Для фронт-енд частини використовувались JavaScript, бібліотека jQuery та Leaflet для інтерактивного відображення мапи та даних про транспортні засоби.

Нарешті розробка алгоритмічного та програмного забезпечення включала кілька етапів, такі як: створення архітектури системи та структури бази даних, розробка серверної та клієнтської частин, тестування розробленої системи. Особлива увага приділялась етапу проєктування алгоритмів розпізнавання рис обличчя водія. Оптимізований метод дозволяє не лише покращити якість аналізу стану водія, але й забезпечити стабільну роботу додатку в умовах обмеженої обчислювальної потужності смартфонів. Здійснений порівняльний аналіз показав, що попри незначну втрату точності у порівнянні з MobileNetV2 (менше 10%), запропонований метод забезпечує значно вищу швидкодію (у 2–2.5 разів менший час обробки кадру, в залежності від моделі смартфона), що критично важливо для задач у реальному часі, особливо у сфері безпеки водія.

У межах економічного розділу було проведено аналіз доцільності розробки та впровадження розробленої автоматизованої системи моніторингу транспортних перевезень з функцією інтелектуального моніторингу стану водія. У результаті розрахунків встановлено, що витрати на розробку системи становили 119 тис. грн, що відповідає запланованому бюджету. Абсолютний економічний ефект від впровадження системи протягом трирічного періоду очікується на рівні 2921 тис. грн. Внутрішня дохідність інвестицій склала 59,6%, що перевищує необхідний мінімальний рівень, а термін окупності інвестицій – 1,7 року, що є значно нижчим за допустимий поріг. Таким чином, усі визначені техніко-економічні показники для розробленої системи були виконані, що підтверджує економічну доцільність її створення та подальшого впровадження у практичну діяльність підприємств транспортної галузі.

Результатом роботи стала комплексна система транспортних перевезень, що реалізує весь необхідний функціонал для відстеження стану та дій водія під час здійснення перевезення. Система довела свою надійність та стійкість до навантажень під час тестування.

СПИСОК ЛІТЕРАТУРИ

1. Моргун Ю.О., Коцюбинський В.Ю. Реалізація системи транспортних перевезень з функцією інтелектуального моніторингу стану водія / Міжнародний конкурс студентських наукових робіт. Напрям «Моделювання та оптимізація процесів керування». Кременчук, КрНУ імені Михайла Остроградського, (2025): Веб-ресурс. https://krnukonkurs.kdu.edu.ua/statti/win_174_2025.pdf (дата звернення 02.10.2025)
2. Конкурс ідей для стартапів серед студентів “IT Idea Battle: Твоя ідея – майбутній стартап” URL: https://blog.sikorskychallenge.com/2025/04/IT-Idea-Battle.html (дата звернення 02.10.2025)
3. Моргун Ю.О., Коцюбинський В.Ю., Юхимчук М.С. Розробка клієнт-серверної системи моніторингу транспортних перевезень / Матеріали LIII НКТП ВНТУ. Секція комп’ютерних систем управління. Вінниця, ВНТУ, (2024): Веб-ресурс. https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/41738/20482.pdf
4. Моргун Ю.О., Коцюбинський В.Ю. Реалізація системи моніторингу транспортних перевезень на основі клієнт-серверної архітектури / Матеріали конференції Контроль і управління в складних системах. Секція “Перспективні методи, програмні і технічні засоби систем контролю і управління”. Вінниця, ВНТУ, (2024): Веб-ресурс. https://doi.org/10.31649/mccs2024.2-06
5. Моргун Ю.О., Коцюбинський В.Ю. Розробка мобільного додатку водія для клієнт-серверної системи моніторингу транспортних перевезень / Матеріали LIV НКТП ВНТУ. Секція автоматизації та інтелектуальних інформаційних технологій. Вінниця, ВНТУ, (2025): Веб-ресурс.

https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/23903/19753

6. What Is an In-Vehicle Monitoring System? How Does It Work? URL: https://sensordynamics.com.au/what-is-an-in-vehicle-monitoring-system/ (дата звернення 05.10.2025)
7. Enrique Onieva, Ignacio Julio García Zuazola, Asier Perallos, Unai Hernandez-Jayo Intelligent Transport Systems: Technologies and Applications, 2015. P. 3 - 17.
8. Transportation Management System: Meaning, Importance, and Benefits URL: https://www.inboundlogistics.com/articles/transportation-management-system/
9. Дослідження МВС щодо основних причин ДТП URL: https://forinsurer.com/public/06/03/02/2196 (дата звернення 26.09.2025)
10. Філ Саєґ, Філ Чарльз Інтелектуальні транспортні системи (відредагована версія), 2009. С. 2 - 18.
11. Автоматизація в логістиці – це ефективність і менші витрати URL: https://www.trans.eu/ua/blog/lohistyka-4-0/koly-systema-praciuie-za-vas/ (дата звернення 07.10.2025)
12. Azim Eskandarian, Handbook of Intelligent Vehicles Volume 2, 2012. P. 2 - 10.
13. Geotab: One Platform - Total Fleet Management URL: https://www.geotab.com/ (дата звернення 08.10.2025)
14. Samsara: The leading fleet management and safety platform URL: https://www.samsara.com/ (дата звернення 08.10.2025)
15. RoscoLive - Real-time fleet management at your fingertips URL: https://roscolive.com/ (дата звернення 08.10.2025)
16. What is C++ & How It Compares to Other C Programming Languages URL: https://www.simplilearn.com/what-is-cpp-programming-article (дата звернення 11.10.2025)
17. What Is Qt framework, Why to Use It, and How? URL: https://lembergsolutions.com/blog/why-use-qt-framework

18. C++ Public, Protected and Private InheritanceURL: https://www.programiz.com/cpp-programming/public-protected-private-inheritance (дата звернення 13.10.2025)
19. Signals & Slots URL: https://doc.qt.io/qt-6/signalsandslots.html (дата звернення 14.10.2025)
20. Using the Meta-Object Compiler (moc) URL: https://doc.qt.io/qt-6/moc.html (дата звернення 14.10.2025)
21. Supported Platforms & Development Languages URL: https://www.qt.io/product/supported-platforms-languages (дата звернення 14.10.2025)
22. Qt Creator Manual URL: https://thinkinginqt.com/doc/qtcreator/index.html (дата звернення 14.10.2025)
23. CMake Documentation and Community URL: https://cmake.org/documentation/ (дата звернення 15.10.2025)
24. Why Use PostgreSQL For Your Next Project? URL: https://cleancommit.io/blog/why-use-postgresql-for-your-next-project/ (дата звернення 08.17.2025)
25. What is PostgreSQL? Key Features, Benefits, and Real-World Uses URL: https://www.percona.com/blog/what-is-postgresql-used-for/ (дата звернення 17.10.2025)
26. Documentation PostgreSQL Chapter 8. Data Types URL: https://www.postgresql.org/docs/current/datatype.html (дата звернення 17.10.2025)
27. Reasons Why JavaScript is Omnipresent in Modern Development URL: https://snipcart.com/blog/why-javascript-benefits (дата звернення 22.10.2025)
28. What Is JQuery & Why’s It The Top JS Library In 2024 URL: https://www.learnenough.com/blog/what-is-jquery (дата звернення 22.10.2025)

29. jQuery Introduction URL: https://www.w3schools.com/jquery/jquery_intro.asp (дата звернення 22.10.2025)
30. Leaflet - a JavaScript library for interactive maps URL: https://leafletjs.com/ (дата звернення 22.10.2025)
31. Why you should use NGINX to serve your web application URL: https://www.slingacademy.com/article/why-use-nginx-web-application-serving/ (дата звернення 23.10.2025)
32. How important is architecture in software development? URL: https://www.ciklum.com/resources/blog/expert-advice-how-important-is-architecture-in-software-development (дата звернення 26.10.2025)
33. Principles of Network Applications URL: https://www.geeksforgeeks.org/principles-of-network-applications/ (дата звернення 26.10.2025)
34. What are UML diagrams, and how can you use them? URL: https://miro.com/blog/uml-diagram/ (дата звернення 26.10.2025)
35. UML для бізнес-моделювання URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення 27.10.2025)
36. Як будувати UML-діаграми URL: https://dou.ua/forums/topic/40575/ (дата звернення 27.10.2025)
37. What is an Event-Driven Architecture? URL: https://aws.amazon.com/event-driven-architecture/ (дата звернення 28.10.2025)
38. Monitoring of Event-Driven System Architecture URL: https://coralogix.com/blog/monitor-event-driven-system-architecture/ (дата звернення 28.10.2025)
39. MVC Architecture Explained: Model, View, Controller URL: https://www.codecademy.com/article/mvc-architecture-model-view-controller (дата звернення 29.10.2025)

40. MVC - MDN Web Docs Glossary: Definitions of Web-related terms URL: https://developer.mozilla.org/en-US/docs/Glossary/MVC (дата звернення 29.10.2025)
41. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. / Укладачі В.О. Козловський, О.Й. Лесько, В.В.Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

Додаток А (обов'язковий)

Технічне завдання

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«17» жовтня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
«АВТОМАТИЗОВАНА КЛІЄНТ-СЕРВЕРНА СИСТЕМА ТРАНСПОРТНИХ ПЕРЕВЕЗЕНЬ З ФУНКЦІЄЮ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ СТАНУ ВОДІЯ»
08-31.МКР.007.02.000 ТЗ

Керівник роботи:
к.т.н., доц. каф. АІІТ
Володимир КОЦЮБИНСЬКИЙ
«16» жовтня 2025 р.

Виконавець:
ст. гр. 1АКІТР-24м
Юрій МОРГУН
«16» жовтня 2025 р.

Вінниця ВНТУ – 2025

1. Назва та галузь застосування

Автоматизована клієнт-серверна система транспортних перевезень з функцією інтелектуального моніторингу стану водія.

Інформаційні технології у сфері збору даних. Логістичні системи. Автоматизація контролю, аналізу стану водія та обліку поїздок із використанням мобільних застосунків та серверних технологій.

2. Підстава для розробки

Розробку системи здійснювати на підставі наказу по університету № 313 від 24 вересня 2025 року та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій»

3. Мета та призначення розробки

Метою роботи є розробка автоматизованої системи для моніторингу транспортних перевезень та водія, яка забезпечить:

- Автоматичний збір та обробку телеметричних даних про поїздку (час старту/фінішу, маршрут, геолокацію, тривалість) із мобільного застосунку водія;
- Розпізнавання стану та дій водія за допомогою камери смартфона, включаючи контроль уваги, наявність відволікань та потенційних небезпечних поведінкових патернів під час поїздки;
- Моніторинг маршруту та статусів водія на сервері з можливістю перегляду активних поїздок, журналу дій, історії переміщень та інцидентів;
- Синхронізацію даних між мобільним застосунком та сервером у режимі реального часу через REST API для забезпечення актуальності та цілісності інформації;
- Адміністрування та управління транспортною діяльністю, включаючи перегляд інформації про водіїв, транспортні засоби, списки поїздок, статуси та логи подій;
- Інтеграцію з базою даних PostgreSQL для зберігання телеметрії, профілів водіїв, історії поїздок та результатів розпізнавання стану.

Призначення системи: забезпечення комплексного контролю та моніторингу транспортних перевезень, підвищення безпеки руху завдяки

автоматичному аналізу стану водія, збирання та зберігання інформації про поїздки в реальному часі.

4. Джерела розробки

- Філ Саєг, Філ Чарльз Інтелектуальні транспортні системи (відредагована версія), 2009. С. 2 - 18.
- Enrique Onieva, Ignacio Julio García Zuazola, Asier Perallos, Unai Hernandez-Jayo Intelligent Transport Systems: Technologies and Applications, 2015. P. 3 - 17.
- Azim Eskandarian, Handbook of Intelligent Vehicles Volume 2, 2012. С. 2 - 10.
- Дослідження МВС щодо основних причин ДТП URL: https://forinsurer.com/public/06/03/02/2196 (дата звернення 26.09.2025)

5. Показники призначення

5.1. Основні технічні характеристики системи

Функціональні можливості:

5.1.1 Серверний модуль обробки та збереження даних:

Прийом даних від мобільного застосунку водія через REST API.

- Збереження телеметрії руху: координати (lat, lon), швидкість, час, статус поїздки;
- Обробка інформації про стан водія (розпізнані події, тип дії);
- Реалізація механізмів автентифікації та обмеження доступу;
- Генерація службових логів та журналів подій для подальшого аудиту.

5.1.2. Web-інтерфейс менеджера транспортного флоту

- Перегляд карти руху транспортних засобів у реальному часі;
- Візуалізація ключових параметрів: поточна локація, статус поїздки, події;
- Перегляд списку водіїв, історії поїздок та їх тривалості;
- Перегляд станів водія, розпізнаних системою (відволікання, сонливість тощо);
- Робота з фільтрами та пошуком по історичних даних.

5.1.3. Мобільний застосунок водія

- Авторизація користувача та отримання службової інформації про поїздку;
- Відображення власних координат, статусу поїздки, кнопок управління (почати/закінчити поїздку);
- Запис телеметрії руху та її передача на сервер;
- Робота при нестабільному інтернет-з’єднанні (локальна буферизація даних).

5.2. Мінімальні системні вимоги

5.2.1 Апаратне забезпечення:

Серверна частина:

- Процесор: 2.5 GHz+, рекомендовано 4 ядра;
- Оперативна пам'ять: від 8 GB;
- Місце на диску: 1 GB - програмне забезпечення; 5–20 GB - історичні дані поїздок;
- Мережа: стабільний доступ до Інтернет (від 50 Mbps).

Мобільний застосунок водія (Android):

- Android 15;
- 2 GB оперативної пам’яті;
- Наявність фронтальної камери;
- Датчики GPS;
- Можливість передачі мобільних даних (4G/LTE).

5.2.2 Програмне забезпечення:

- Серверна ОС: Windows / Linux (Ubuntu 20.04+) / macOS;
- Qt 6.x + C++17 для розробки клієнтської та серверної частин;
- PostgreSQL 14+;
- Android SDK – для мобільного застосунку.

5.3. Вхідні дані

- Дані GPS-позиціонування (широта, довгота, час);
- Дані камерного модуля (потік кадрів з фронтальної камери смартфону);
- Дані водія: ID, статус, інформація про транспортний засіб;
- Службові події мобільного застосунку: «Початок поїздки», «Завершення поїздки», «Втрачено з’єднання»;
- Небезпечні стани та дії, розпізнані алгоритмами комп’ютерного зору.

5.4 Результати роботи системи

- Повний маршрут руху за кожну поїздку;

- Історія станів та подій поведінки водія;
- Звіти про поїздки та супровідні журнали;
- Профілі водіїв, транспортних засобів та облікові записи.

6. Економічні показники

До економічних показників входять:

- витрати на розробку – до 125 тис. грн.
- абсолютний ефект від впровадження розробки – в межах 2800 тис. грн.
- внутрішня дохідність потенційних інвестицій – не менше 55%
- термін окупності – до 3х років

7. Стадії розробки:

1. Розділ 1 «Аналіз предметної області» має бути виконаний до 05.10.2025 р.

1. Розділ 2 «Технології розробки системи моніторингу транспортних перевезень» має бути виконаний до 25.10.2025 р.

1. Розділ 3 «Розробка алгоритмічного та програмного забезпечення» має бути виконаний до 20.11.2025 р.

1. Розділ 4 «Економічний розділ» має бути виконаний до 01.12.2025 р.

8. Порядок контролю та приймання

- Рубіжний контроль провести до 14.11.2025.
- Попередній захист магістерської кваліфікаційної роботи провести до 02.12.2025.
- Захист магістерської кваліфікаційної роботи провести в період з 15.12.2025 р. до 19.12.2025 р.

Додаток Б

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«АВТОМАТИЗОВАНА КЛІЄНТ-СЕРВЕРНА СИСТЕМА ТРАНСПОРТНИХ ПЕРЕВЕЗЕНЬ З ФУНКЦІЄЮ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ СТАНУ ВОДІЯ»

Студент групи 1АКІТР-24м

Юрій МОРГУН

Керівник к.т.н., доц. каф. АІІТ

Володимир КОЦЮБИНСЬКИЙ

Вінниця ВНТУ – 2025

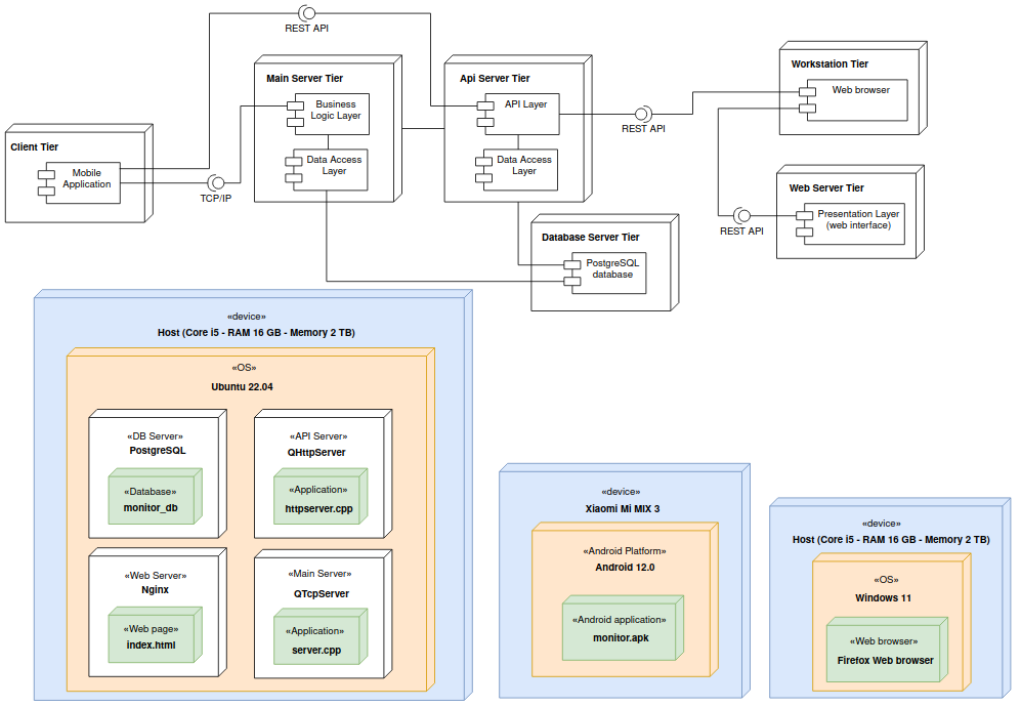

Рисунок Б.1 – Deployment UML діаграма

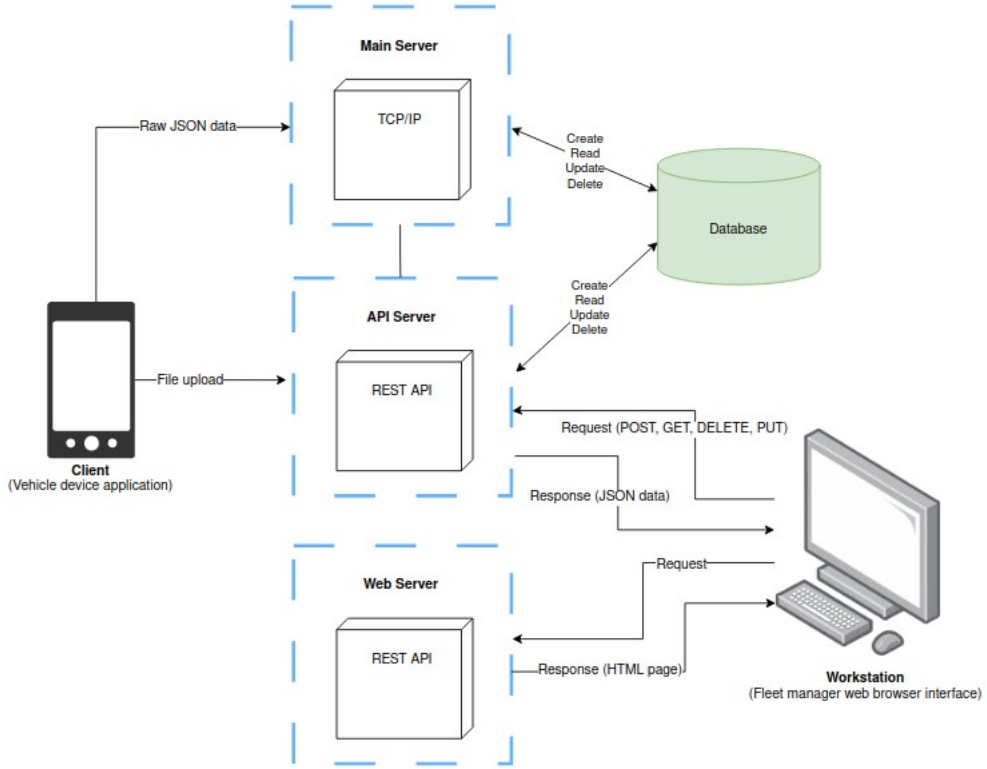

Рисунок Б.2 – Data Flow UML діаграма

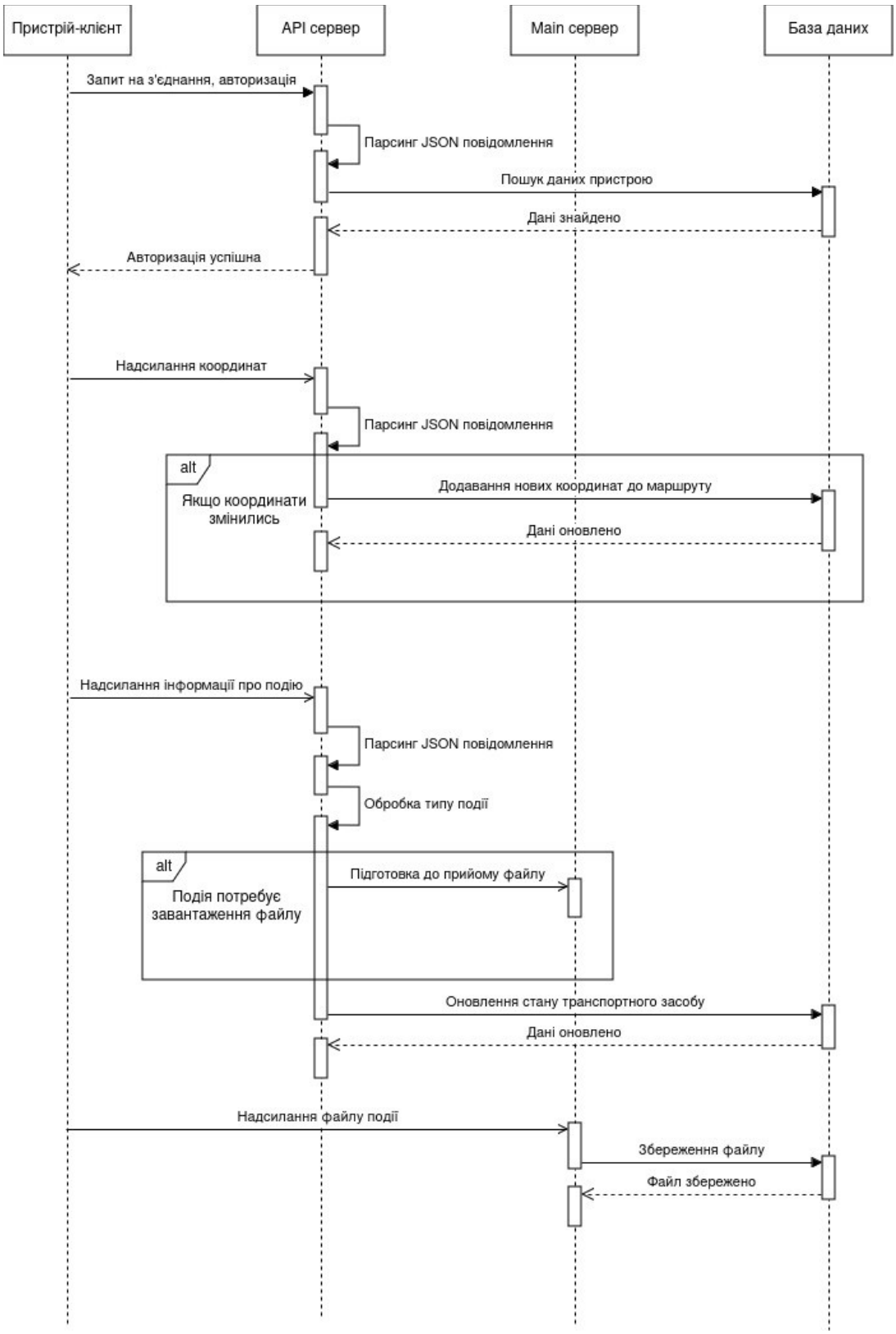

Рисунок Б.3 – UML діаграма послідовності взаємодії пристрою-клієнта та сервера

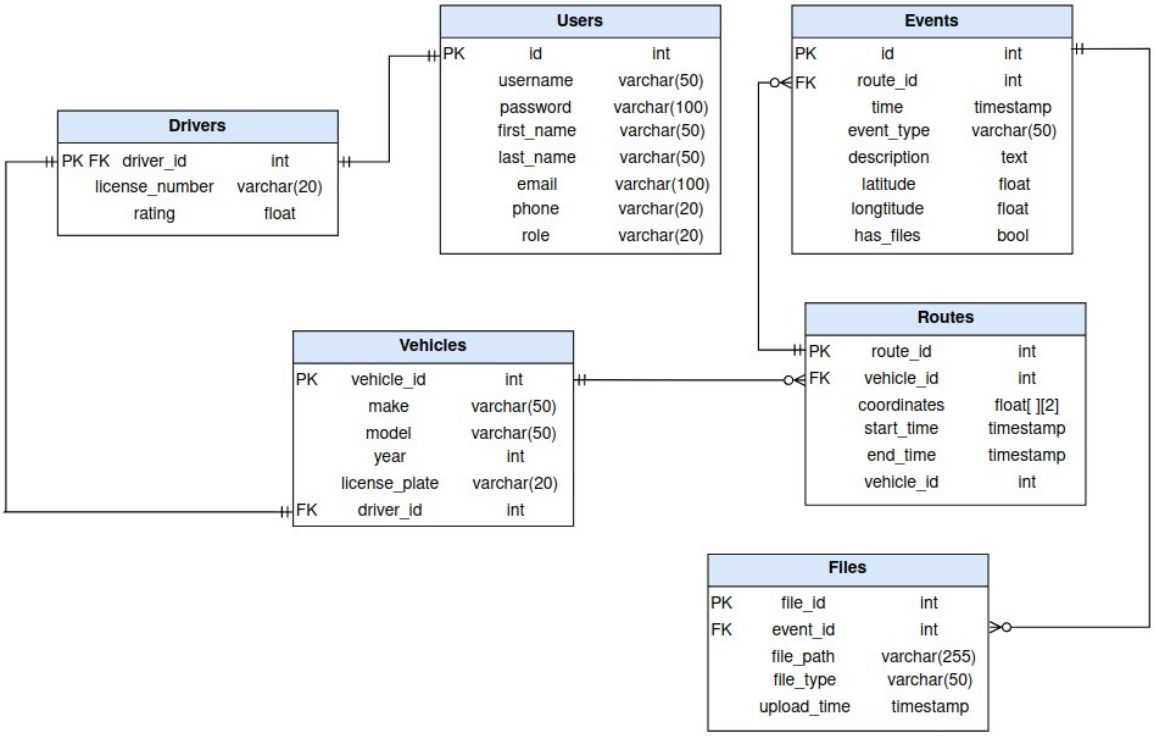

Рисунок Б.4 – ER-модель системи, що розробляється

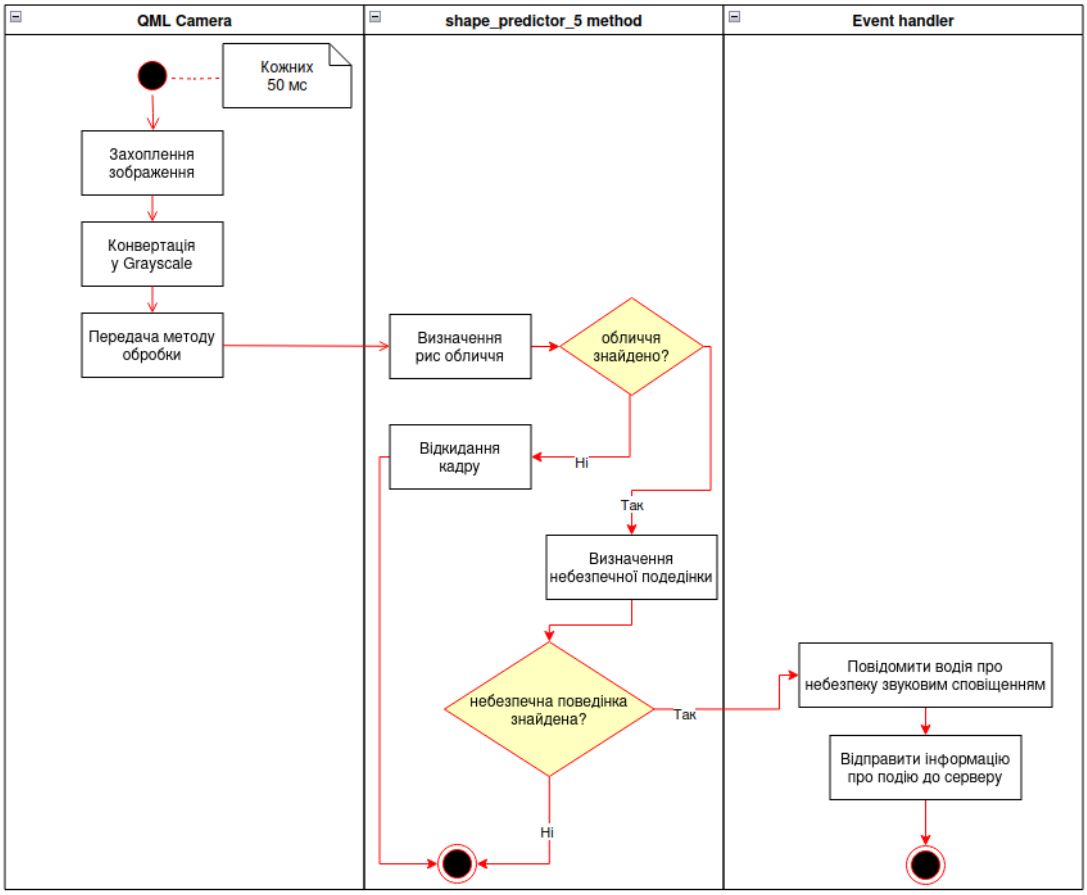

Рисунок Б.5 – UML-діаграма діяльності методу shape_predictor_5

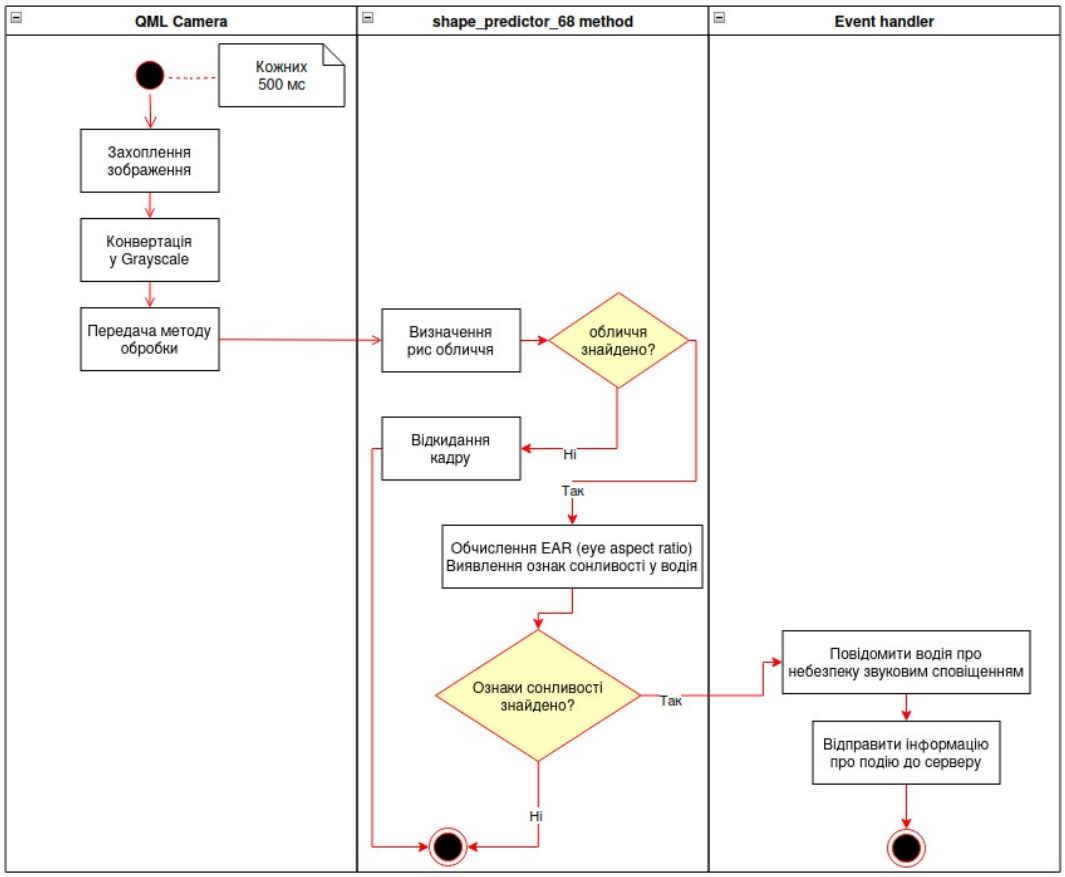

Рисунок Б.6 – UML-діаграма діяльності методу shape_predictor_68

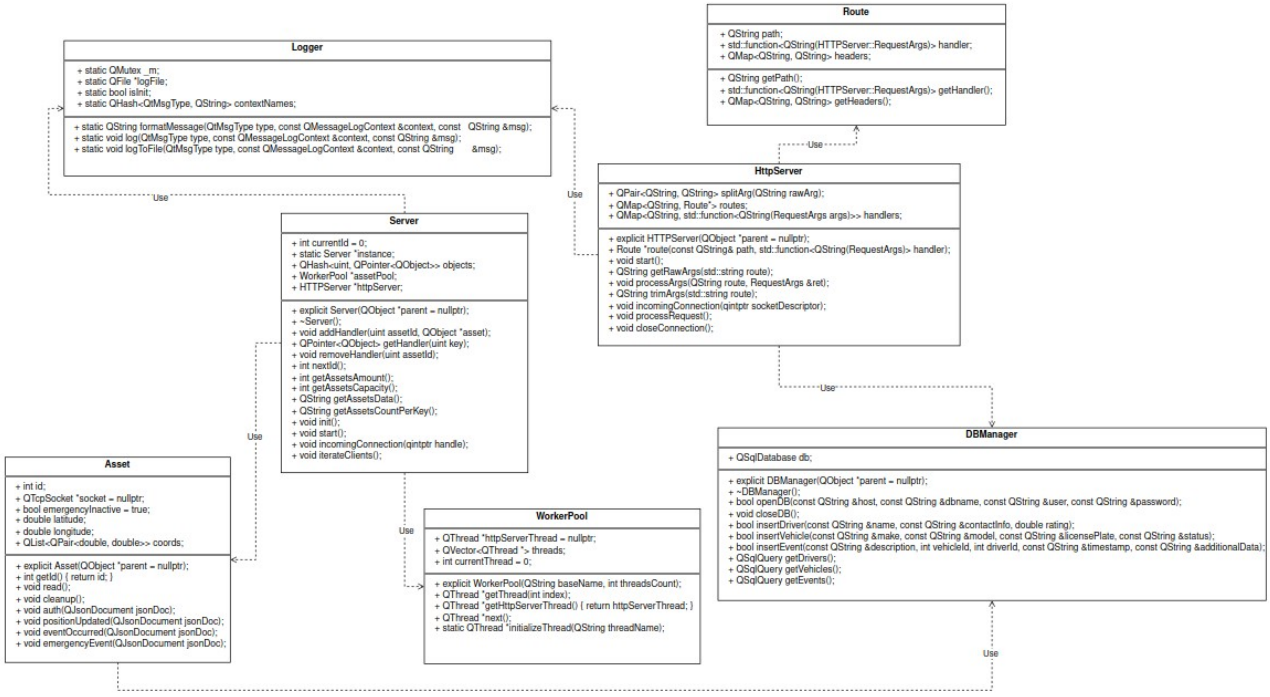

Рисунок Б.7 – UML діаграма класів серверної частини

Додаток В (обов’язковий) Лістинг програмного забезпечення

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QDebug>
#include <QMainWindow>
#include <QGeoPositionInfoSource>
#include <QTcpSocket>
#include <fileuploader.h>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow {
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void connectToServer();
    void sendToServer(QString message = QString());
    void receiveFromServer();
    void onPositionUpdated(const QGeoPositionInfo &info);

private:
    FileUploader uploader;

    Ui::MainWindow *ui;
    QGeoPositionInfoSource *positionSource;

    bool waitingForAck = false;
    QList<QString> queuedMessages;
    QTcpSocket *socket;

    double oldLatitude = 0;
    double oldLongitude = 0;
```

```
};

#endif // MAINWINDOW_H

#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent), ui(new
Ui::MainWindow) {
   ui->setupUi(this);
   socket = new QTcpSocket(this);
   connect(socket, &QTcpSocket::readyRead, this, &MainWindow::receiveFromServer);

   connect(socket, &QTcpSocket::connected, this, []() {
      qDebug() << "Connected successfully";
   });

   connect(socket, &QTcpSocket::errorOccurred, this, [this]() {
      qDebug() << "Failed to connect to server. Error: " << socket->errorString();
      socket->close();
   });

   connect(socket, &QTcpSocket::disconnected, this, []() {
      qDebug() << "disconnectFromServer";
   });

                        connect(ui->connectButton,      &QPushButton::released,      this,
&MainWindow::connectToServer);
   connect(ui->uploadButton, &QPushButton::released, this, [this](){
      uploader.captureAndUpload();
   });

   positionSource = QGeoPositionInfoSource::createDefaultSource(this);
   if (positionSource) {
                                                                          positionSource-
>setPreferredPositioningMethods(QGeoPositionInfoSource::AllPositioningMethods);
      positionSource->setUpdateInterval(2000);

            connect(positionSource,  &QGeoPositionInfoSource::positionUpdated,  this,
&MainWindow::onPositionUpdated);
```

```
    }
}

void MainWindow::connectToServer() {
    if(socket->state() != QAbstractSocket::UnconnectedState) {
        return;
    }

    qDebug() << "connectToServer";
    QStringList address = {"10.0.2.2", "6001"};
    if(address.size() != 2) {
        qWarning() << "Failed to parse address";
        return;
    }
    socket->connectToHost(address[0], address[1].toUInt());

    positionSource->startUpdates();
}

void MainWindow::sendToServer(QString message) {
    qDebug() << "sendToServer";

    if(message == "" || !socket->isValid()) {
        return;
    }

    if (waitingForAck) {
        qDebug() << "Socket is busy. Message is queued.";
        queuedMessages.append(message);
        return;
    }

    QByteArray data;
    QDataStream out(&data, QIODevice::WriteOnly);
    out.setVersion(QDataStream::Qt_DefaultCompiledVersion);
    out << quint16(0) << message;

    out.device()->seek(0);
    out << quint16(data.size() - sizeof(quint16));
```

```
    waitingForAck = true;
    socket->write(data);
}

void MainWindow::receiveFromServer() {
    qDebug() << "receiveFromServer";

    if(!socket) {
        qDebug() << "Socket is NULL";
        return;
    }

    QString message;
    QByteArray data;
    while (socket->bytesAvailable() > 0) {
        QByteArray receivedData = socket->readAll();
        data += receivedData;

        while (static_cast<quint16>(data.size()) >= sizeof(quint16)) {
            QDataStream in(&data, QIODevice::ReadOnly);
            in.setVersion(QDataStream::Qt_DefaultCompiledVersion);

            quint16 messageSize;
            in >> messageSize;

            if (data.size() < messageSize)
                break;

            in >> message;

            data = data.mid(messageSize);
        }
    }

    if(message == "-1") {
        qDebug() << "Acknowledgment received";
        waitingForAck = false;
        if(!queuedMessages.isEmpty()) {
            sendToServer(queuedMessages.takeFirst());
        }
```

```
        return;
    }
}

void MainWindow::onPositionUpdated(const QGeoPositionInfo &info) {
    if (!info.isValid()) {
        qWarning() << "QGeoPositionInfo not valid";
        return;
    }

    double latitude = info.coordinate().latitude();
    double longitude = info.coordinate().longitude();
    if (latitude == oldLatitude && longitude == oldLongitude)
        return;

    oldLatitude = latitude;
    oldLongitude = longitude;

    qDebug() << "[" << latitude << "," << longitude << "],";

        sendToServer(QString("{\"method\": \"positionUpdated\", \"lat\": %1, \"lng\":
%2}").arg(latitude).arg(longitude));
}

MainWindow::~MainWindow() {
    qDebug() << "~MainWindow";
    socket->deleteLater();
    delete ui;
}
```

Додаток Г (обов'язковий) Акт впровадження

Науково-виробниче підприємство
„СПІЛЬНА СПРАВА“ Товариство з обмеженою відповідальністю
Україна, 21000, м. Вінниця, вул. Магістратська, 36/3, тел. +38(096) 558 24 64
код ЄДРПОУ 31041670, код платників податків 310416702282, свідоцтво № 0183329
IBAN : UA 41 322313 0000026004000003226 в філії АТ «Укрексімбанк» в м. Вінниці

Затверджую
Директор
НВП «СПІЛЬНА СПРАВА», ТОВ

Малачковська Роксолана Ігорівна

« 01 » грудня 2025 р.

АКТ

впровадження результатів магістерської роботи

Моргуна Юрія Олександровича «Автоматизована клієнт-серверна система транспортних перевезень з функцією інтелектуального моніторингу стану водія»

Комісія у складі головного спеціаліста, керівника відділу обробки даних С. Житанського та керівника відділу розробки інформаційних систем О. Кириленка склали цей акт про те, що у Науково-виробничому підприємстві „Спільна Справа“, ТОВ впроваджуються результати, які отримані магістром Моргуном Ю. О. при виконанні магістерської кваліфікаційної роботи. Розроблений метод розпізнавання дій та стану водія забезпечує роботу в умовах обмежених обчислювальних ресурсів без використання серверної обробки чи нейронних мереж.

Таким чином, актуальність роботи Ю. О. Моргуна не викликає сумнівів, а низка методик та підходів та результати їх застосування можуть бути застосованими у практичній діяльності.

Науково-виробничому підприємству „Спільна Справа“, ТОВ магістром Моргуном Ю. О. передано програмне забезпечення, яке дозволяє підвищити ефективність процесу моніторингу стану водія.

Члени комісії:

Головний спеціаліст Сергій ЖИТАНСЬКИЙ

Керівник відділу Олександр КИРИЛЕНКО

113

Додаток Д (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Автоматизована клієнт-серверна система транспортних перевезень з функцією інтелектуального моніторингу стану водія»

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

Підрозділ кафедра АІІТ
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 6,75 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☑ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Бісікало О.В., зав. каф. АІІТ (підпис)
(прізвище, ініціали, посада)

Овчинников К.В., доц. каф. АІІТ (підпис)
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку (підпис) Маслій Р.В.
(прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник (підпис) Коцюбинський В.Ю., доц. каф. АІІТ
(прізвище, ініціали, посада)

Здобувач (підпис) Моргун Ю.О.
(прізвище, ініціали)