МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту

"_____" _____ 2025 р.

Зав. кафедри, к.т.н., доцент

_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Програмне забезпечення для інтелектуального керування рухом транспорту

Виконав – студент 4 курсу, групи 41-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки
Бойченко Максим Олегович
Науковий керівник – д.т.н., професор
Кучерук Володимир Юрійович

Умань – 2025

Реферат

на кваліфікаційну роботу «Програмне забезпечення для інтелектуального керування рухом транспорту»

студента факультету економіки та підприємництва

Уманського національного університету

Бойченка Максима Олеговича

Структура та обсяг роботи - пояснювальна записка до кваліфікаційної роботи бакалавра складається з 3 розділів, містить 72 сторінки, 35 рисунків, 1 додаток, 20 літературних джерел.

У рамках кваліфікаційної роботи бакалавра було створено програмні модулі для інтелектуального керування транспортним рухом. Основні виконані завдання включали: загальна схема роботи системи; алгоритм моделювання транспортного потоку; адаптивний графічний інтерфейс користувача; модуль збереження даних; функціонал для виявлення порушників швидкісного режиму; тестування системи відповідно до поставлених вимог.

Перший розділ присвячено аналізу сучасного стану систем керування транспортним рухом. Здійснено порівняння наявних рішень із запропонованим модулем, що дозволило обґрунтувати доцільність розробки власного застосунку.

У другому розділі розглянуто структуру вхідних даних, а також описано етапи розробки програмних компонентів. Запропоновано модель системи, яку проілюстровано за допомогою UML-діаграми активності. Розроблено алгоритм автоматичного переміщення транспортного засобу та представлено блок-схеми алгоритмів.

Третій розділ містить огляд мов програмування та інструментів, що розглядалися для реалізації проекту. Аргументовано вибір мови Java для розробки системи. Створено інтерфейс користувача, реалізовано інтеграцію з базою даних. Також створено модуль адаптивного керування світлофором і підсистему моніторингу порушників правил дорожнього руху. Проведено повне тестування системи. На завершення розроблено інструкцію для користувачів програмного продукту.

Анотація

У кваліфікаційній роботі бакалавра розроблено програмні засоби для системи інтелектуального керування рухом транспорту, яка призначена для десктопних платформ. Розроблювані програмні модулі можуть бути спрямовані на створення ефективної системи керування рухом транспорту з використанням інтелектуальних алгоритмів та аналізу даних.

Застосунок розроблено з використанням мови програмування Java та середовища розробки IntelliJ IDEA. Для графічного відображення вікон використано JavaFX.

Отримані результати можна використати для розробки ефективних систем моніторингу та управління транспортними потоками для зменшення транспортних заторів, підвищення безпеки на дорогах та оптимізації використання інфраструктури міст і доріг.

Ключові слова: десктопний застосунок, транспорт, програмні засоби, оптимізація транспортних потоків.

Abstract

In this bachelor's thesis, software tools for an intelligent traffic management system designed for desktop platforms were developed. The developed software modules can be used to create an efficient traffic management system using intelligent algorithms and data analysis.

The application was developed using the Java programming language and the IntelliJ IDEA development environment. JavaFX is used for graphical display of windows.

The results obtained in the bachelor's thesis can be used to develop effective traffic monitoring and management systems to reduce traffic congestion, improve road safety, and optimize the use of city and road infrastructure.

Keywords: desktop application, transport, software, traffic flow optimization.

ЗМІСТ

Перелік умовних позначень, скорочень і термінів ……..7

ВСТУП 8

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ У СФЕРІ СИСТЕМ ІНТЕЛЕКТУАЛЬНОГО КЕРУВАННЯ РУХОМ ТРАНСПОРТУ ТА ФОРМУЛЮВАННЯ ОСНОВНИХ ЗАВДАНЬ ДОСЛІДЖЕННЯ 10

1.1 Аналіз стану питання керування рухом транспорту 10

1.2 Порівняльний аналіз аналогів 11

1.3 Аналіз методів розв’язання поставленої задачі 15

РОЗДІЛ 2. РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ЗАСТОСУНКУ 19

2.1 Аналіз вхідних даних системи 19

2.2 Розробка моделі системи 20

2.3 Розробка алгоритмів роботи системи..... 22

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ СИСТЕМИ ДЛЯ КЕРУВАННЯ РУХОМ ТРАНСПОРТУ 25

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного застосунку та розробка його інтерфейсу 25

3.2 Розробка бази даних ……. 32

3.3 Тестування системи ……..... 41

ВИСНОВКИ 49

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 50

ДОДАТОК А. Лістинг програми 52

Перелік умовних позначень, скорочень і термінів

SUMO	Simulation of Urban MObility
LINQ	Language Integrated Query
XML	EXtensible Markup Language
JVM	Java Virtual Machine
MVC	Model-View-Controller
SQL	Structured Query Language
JDK	Java Development Kit

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному світі транспорт є невід'ємною складовою повсякденного життя людей та функціонування економіки. З огляду на це, питання безпеки та ефективної організації дорожнього руху набувають особливої актуальності. Одним із ключових викликів є необхідність раціонального управління транспортними потоками з урахуванням багатьох чинників - інтенсивності руху, безпеки учасників дорожнього руху, дотримання правил тощо.

У цьому контексті виникає потреба у впровадженні систем інтелектуального керування транспортом, здатних забезпечити безпечне, зручне та ефективне пересування транспортних засобів. Такі системи ґрунтуються на використанні сучасних інформаційно-комунікаційних технологій, аналізу даних та алгоритмів штучного інтелекту для прийняття оперативних і оптимальних рішень у реальному часі [1].

Незважаючи на наявність низки програмних рішень для реалізації інтелектуального управління рухом транспорту [2], багато з них мають обмежену функціональність або недостатню адаптивність до умов реального міського середовища. Це створює потребу в розробці власної, гнучкої та ефективної системи, орієнтованої на вирішення актуальних проблем транспортного управління.

Мета та завдання дослідження. Метою дослідження є підвищення безпеки дорожнього руху та покращення якості управління транспортними потоками шляхом розробки програмного забезпечення з використанням сучасних інформаційних технологій. Очікується, що впровадження такої системи дозволить зменшити кількість ДТП, підвищити пропускну здатність доріг і скоротити час затримок у русі.

Основні завдання дослідження:

- сформулювати загальний алгоритм функціонування системи;
- розробити алгоритм симуляції транспортного руху;

- створити адаптивний та інтуїтивно зрозумілий графічний інтерфейс користувача;
- реалізувати модуль збереження та обробки даних;
- створити підсистему моніторингу правопорушень (наприклад, перевищення швидкості);
- провести тестування програмного продукту відповідно до поставлених задач.

Об’єкт і предмет дослідження. Об’єкт дослідження - процес розробки програмних компонентів для системи інтелектуального керування рухом транспорту. Предмет дослідження - алгоритми та засоби реалізації програмного застосунку для управління транспортними потоками.

Методи дослідження. У процесі розробки було використано такі методи: побудова графічного інтерфейсу за допомогою JavaFX; розробка блок-схем алгоритмів і структурних діаграм; модульна розробка програмного забезпечення з використанням принципів об’єктно-орієнтованого програмування.

Практична цінність дослідження. Розроблене програмне забезпечення має практичне значення і може бути застосоване як частина комплексної системи керування дорожнім рухом. Його використання сприятиме оптимізації транспортних потоків, зниженню рівня аварійності, покращенню якості транспортного обслуговування та підвищенню безпеки учасників дорожнього руху.

Отримані результати можуть бути адаптовані до потреб сучасних міст і регіонів, сприяючи розвитку сталої транспортної інфраструктури та підвищенню якості життя громадян.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ У СФЕРІ СИСТЕМ ІНТЕЛЕКТУАЛЬНОГО КЕРУВАННЯ РУХОМ ТРАНСПОРТУ ТА ФОРМУЛЮВАННЯ ОСНОВНИХ ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз стану питання керування рухом транспорту

Аналіз сучасного стану управління транспортними потоками свідчить про складність і багатогранність викликів, із якими стикаються існуючі транспортні системи [4]. Серед основних проблем - порушення правил дорожнього руху, недостатня концентрація уваги водіїв, технічні несправності транспортних засобів, а також перевантаженість дорожньої інфраструктури.

1. Безпека дорожнього руху. Зростання кількості дорожньо-транспортних пригод стало однією з найгостріших проблем сучасного міського середовища. Основними причинами аварій залишаються: недотримання правил дорожнього руху, перевищення швидкості, керування у нетверезому стані та технічна несправність транспортних засобів. Наслідки таких подій - значні матеріальні збитки, травмування та загибель людей.

2. Затори та перевантаження інфраструктури. Інтенсивне зростання кількості автомобілів призводить до перенавантаження міських доріг. Обмежена пропускна здатність та недостатній розвиток інфраструктури у поєднанні з відсутністю дієвих механізмів регулювання потоків спричиняють затори, підвищення рівня викидів і втрати часу для учасників руху.

3. Недостатній рівень технологічної оснащеності. Нестача сучасних інформаційно-комунікаційних технологій та обмежене впровадження автоматизованих систем управління транспортом знижують здатність реагувати на динамічні зміни у дорожній ситуації. Це унеможливлює оперативне прийняття рішень, що критично важливо для ефективного керування рухом у реальному часі.

4. Порушення швидкісного режиму. Недотримання встановлених обмежень швидкості є системною проблемою, що суттєво підвищує ризик виникнення аварій. Ускладнює ситуацію й обмежена здатність контролювати

швидкість транспортних засобів через відсутність або недостатню точність наявних систем моніторингу.

Необхідність пошуку інноваційних рішень. Зазначені проблеми потребують впровадження новітніх підходів до управління дорожнім рухом, зокрема на основі інтелектуальних систем. Вирішення цих завдань можливе шляхом розробки програмних засобів, які поєднують аналіз великих обсягів даних, алгоритми машинного навчання та технології моніторингу в реальному часі.

Значення програмних рішень для транспортного управління. Розробка та впровадження програмних модулів інтелектуального управління рухом транспорту є важливим кроком у напрямку вдосконалення транспортних систем. Такі системи здатні: підвищити безпеку на дорогах; зменшити кількість заторів; оптимізувати потоки транспортних засобів; забезпечити стабільну роботу інфраструктури; покращити загальний комфорт для учасників дорожнього руху.

Інтеграція інтелектуальних рішень у транспортну сферу відкриває шлях до створення стійких, адаптивних і ефективних міських транспортних систем, що відповідатимуть вимогам сучасності та сприятимуть сталому розвитку міського середовища.

1.2 Порівняльний аналіз аналогів

Проведення порівняльного аналізу аналогів є ключовим етапом у розробці програмного забезпечення для системи інтелектуального керування рухом транспорту [5]. Такий аналіз дозволяє здійснити обґрунтований вибір технологічних рішень та методів, які найбільше відповідають вимогам проєкту. Вивчення існуючих систем дає змогу виявити їхні переваги, недоліки та функціональні можливості, що в подальшому допоможе уникнути типових помилок і застосувати найефективніші підходи.

Аналіз аналогічних розробок також дозволяє отримати цілісне уявлення про рівень розвитку інтелектуальних транспортних систем у світі, а також

оцінити функціональність, продуктивність, масштабованість, зручність у використанні та технічну реалізацію наявних рішень.

До найвідоміших програмних продуктів, що можуть вважатися аналогами, належать: SUMO (*Simulation of Urban MObility*); MATSim (*Multi-Agent Transport Simulation*); Aimsun; VISSIM.

SUMO (Simulation of Urban MObility) - це відкрите програмне забезпечення для моделювання дорожнього руху в міському середовищі, розроблене Німецьким аерокосмічним центром (German Aerospace Center) (рис. 1.1) [6]. Система орієнтована на дослідження, аналіз та оптимізацію транспортної інфраструктури з урахуванням складної взаємодії між різними видами транспорту - автомобілями, автобусами, трамваями, велосипедистами та пішоходами [7].

SUMO підтримує гнучке моделювання транспортних мереж, управління сигналами світлофорів, а також аналіз трафіку в реальному часі або у симульованих сценаріях. Її перевагою є відкритий код, що дозволяє адаптувати систему до конкретних потреб дослідника або розробника. Водночас, система має обмеження у графічному інтерфейсі та потребує додаткових знань у сфері роботи з XML-файлами та скриптовими мовами для повноцінного використання.

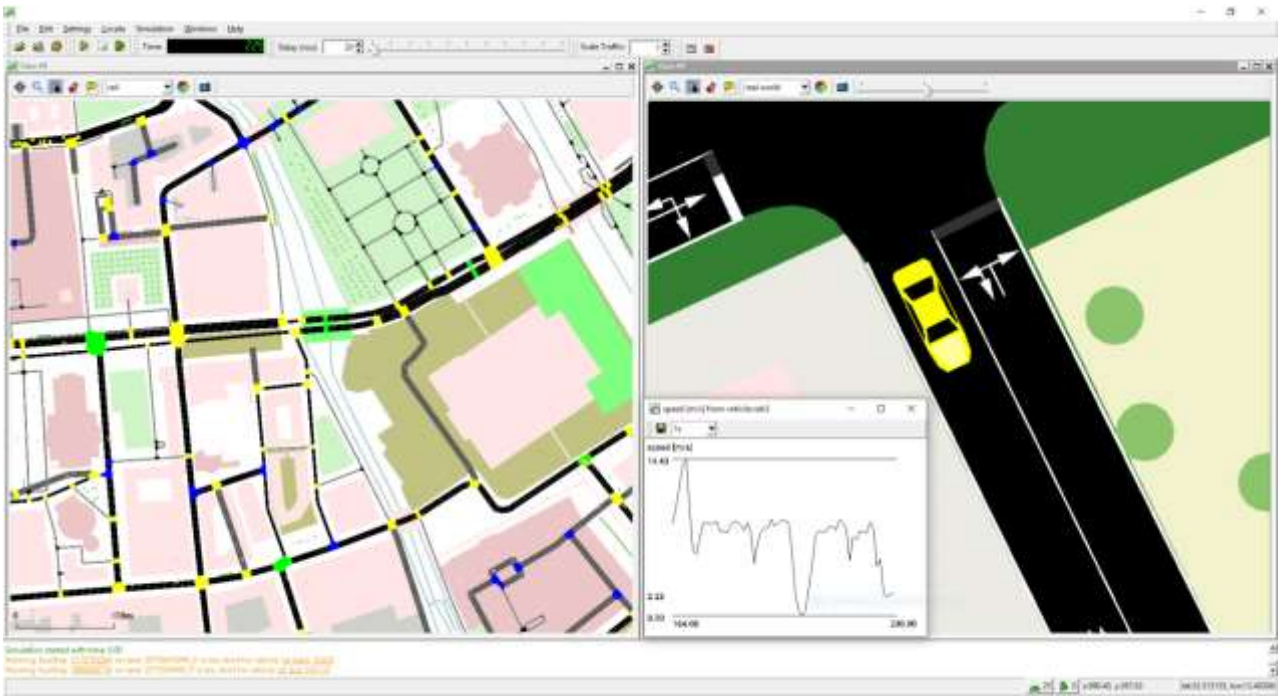


Рисунок 1.1. SUMO

Система SUMO дає змогу користувачам моделювати складні транспортні мережі, проводити аналіз їхньої ефективності та здійснювати оптимізацію різних параметрів, таких як трафік, черги на перехрестях, швидкісні режими тощо. Завдяки цим можливостям, SUMO є потужним інструментом для дослідження та вдосконалення міської транспортної інфраструктури, а також для підтримки прийняття рішень у сфері планування мобільності.

MATSim (*Multi-Agent Transport Simulation*) - це ще один приклад відкритого програмного забезпечення, призначеного для моделювання транспортних систем на основі агентного підходу (рис. 1.2) [8]. Ключова концепція MATSim полягає в тому, що кожен учасник дорожнього руху (автомобіль, автобус, велосипедист, пішохід тощо) моделюється як окремий агент, який самостійно приймає рішення щодо свого маршруту та поведінки у кожен момент часу.

Ці агенти враховують індивідуальні переваги, обмеження та динамічні фактори довкілля, зокрема поточний стан трафіку, доступність маршрутів і час у дорозі [9]. Завдяки цьому MATSim дозволяє отримати детальну картину транспортної поведінки на мікро- та макрорівні, що робить його ефективним засобом для аналізу транспортної політики, оптимізації маршрутів і планування сталого розвитку міського середовища.

Aimsun - це комерційне програмне забезпечення для моделювання та симуляції транспортних систем, яке дає змогу аналізувати рух транспортних засобів у різних середовищах - від міських вулиць до автомагістралей (рис. 1.3) [10]. Основне призначення Aimsun полягає в наданні ефективних інструментів для планування, проєктування та оптимізації транспортної інфраструктури з урахуванням таких факторів, як щільність трафіку, структура дорожньої мережі, розташування перехресть, світлофорне регулювання та інші динамічні параметри [11].

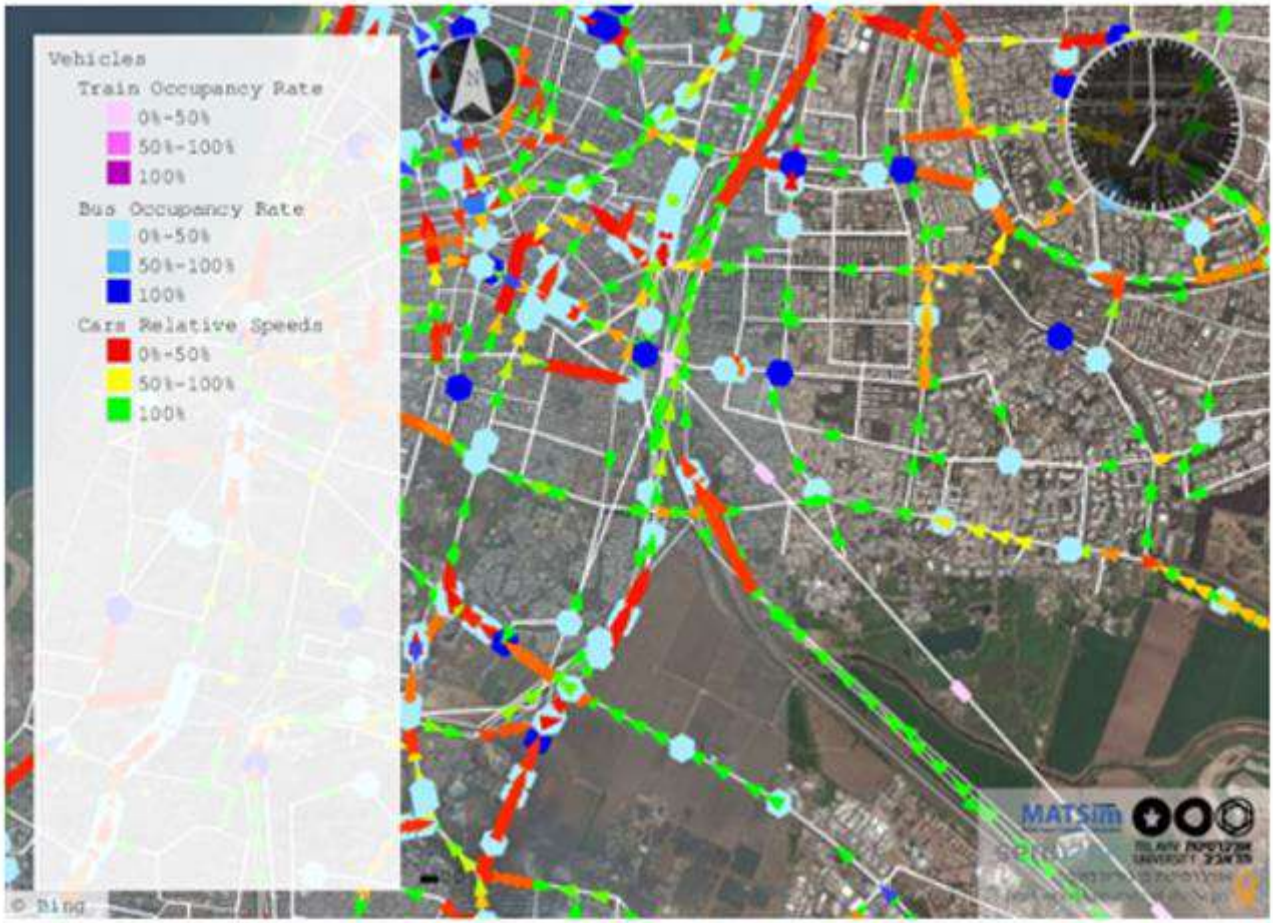


Рисунок 1.2. MATSim

Програмне забезпечення підтримує як макро-, так і мікроскопічне моделювання, що дозволяє проводити комплексний аналіз транспортних потоків у режимі реального часу або за допомогою сценаріїв прогнозування.

VISSIM - це ще одна потужна комерційна платформа для симуляції дорожнього руху та транспортних потоків, розроблена компанією PTV Group (рис. 1.4) [12]. VISSIM широко використовується інженерами, дослідниками та міськими планувальниками для аналізу, планування та вдосконалення транспортних систем у різноманітних середовищах - від урбанізованих районів до міжміських магістралей [13].

Ця система вирізняється високою точністю моделювання поведінки водіїв, пішоходів, взаємодії з елементами інфраструктури, а також можливістю інтеграції з іншими транспортними моделями та зовнішніми даними. Завдяки гнучкості налаштувань та багатофункціональному інтерфейсу, VISSIM

забезпечує високу точність при прогнозуванні наслідків різних управлінських рішень, таких як зміна організації руху, оптимізація світлофорів чи впровадження нових маршрутів.



Рисунок 1.3. Aimsun

Результати порівняння аналогів зведено в табл. 1.1.

Отже, з урахуванням виявлених обмежень існуючих рішень у сфері управління транспортними системами, розробка власного програмного продукту є обґрунтованим і актуальним кроком. Створення нового програмного забезпечення дозволить усунути наявні недоліки, розширити функціональні можливості та запропонувати користувачам більш зручний, адаптивний та ефективний інструмент. Усвідомлення переваг, які може надати така система, сприятиме впровадженню інновацій у транспортну галузь, підвищенню безпеки

дорожнього руху та загальній ефективності транспортної інфраструктури. Це, у свою чергу, забезпечить конкурентоспроможність розробки на ринку сучасних інтелектуальних транспортних рішень.



Рисунок 1.4. VISSIM

Таблиця 1.1 – Порівняльні характеристики аналогів

Критерії	Модулі				
	SUMO	MATSim	Aimsun	VISSIM	TrafficT
Моделювання транспортного потоку	+	+	+	+	+
Оптимізація транспортного потоку	+	+	+	+	+
Сценарії та аналіз	+	+	+	+	+
База даних	–	–	–	–	+
Моніторинг правопорушників	–	–	–	–	+
Загальна оцінка	60%	60%	60%	60%	100%

1.3 Аналіз методів розв’язання поставленої задачі

Завдання розробки полягає у створенні програмного продукту для інтелектуального управління транспортними потоками, а також у дослідженні підходів і методів, що застосовуються для вирішення проблем у цій сфері. Від традиційних математичних моделей до сучасних технологій на основі штучного

інтелекту - розглядаються методології, які спрямовані на оптимізацію ефективності транспортних систем і підвищення якості управління дорожнім рухом. У процесі аналізу враховуються такі чинники, як точність прогнозування трафіку, швидкість реагування на зміни дорожньої ситуації, зниження заторів і покращення загального рівня безпеки.

Першим підходом є використання класичних математичних моделей [14], зокрема методів теорії черг та оптимізації. Ці підходи базуються на обробці статистичних даних та математичному моделюванні для розрахунку оптимальних маршрутів, часу пересування і розподілу транспортних потоків.

Другий підхід - агентне моделювання [15], при якому кожен транспортний засіб розглядається як окремий агент, що приймає рішення на основі власного стану та впливу навколишнього середовища. Цей метод забезпечує більшу гнучкість і адаптивність системи, проте потребує значних обчислювальних ресурсів для моделювання великої кількості агентів.

Третій підхід - використання сучасних інформаційних технологій, зокрема методів штучного інтелекту та машинного навчання. Ці технології дають змогу системам навчатися на основі даних і адаптуватися до змін у режимі реального часу, що значно підвищує точність і ефективність управління транспортними потоками.

Порівняльний аналіз цих підходів показує, що кожен із них має свої переваги та обмеження, які слід враховувати залежно від особливостей конкретної транспортної системи. З огляду на потребу в динамічності та адаптивності, було обрано саме агентний підхід як найбільш придатний для розробки сучасної системи інтелектуального керування рухом транспорту.

У результаті проведеного аналізу було визначено основні завдання, необхідні для реалізації проєкту: розробити концептуальну модель системи; сформувати загальний алгоритм її функціонування; створити алгоритм симуляції руху транспорту на дорожній мережі; розробити зручний та адаптивний графічний інтерфейс користувача; спроєктувати та реалізувати базу даних; впровадити модуль моніторингу порушень швидкісного режиму; провести

повноцінне тестування програмного забезпечення відповідно до поставлених завдань.

Таким чином, ефективна реалізація системи інтелектуального керування транспортом передбачає послідовне виконання зазначених ключових завдань, без яких створення повноцінного програмного продукту неможливе.

РОЗДІЛ 2. РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ЗАСТОСУНКУ

2.1 Аналіз вхідних даних системи

Реалізація системи інтелектуального керування рухом транспорту ґрунтується на використанні мови програмування Java та фреймворку JavaFX для створення графічного інтерфейсу користувача.

Java є однією з найпоширеніших мов програмування, що вирізняється платформонезалежністю, високою продуктивністю та широким спектром застосування. Завдяки великій спільноті розробників і наявності численних бібліотек та інструментів, Java є надійною основою для створення складних програмних рішень, зокрема систем управління транспортними потоками.

Для реалізації графічного інтерфейсу в системі використовується JavaFX - сучасна технологія, що забезпечує розробку інтерактивних, динамічних та візуально привабливих інтерфейсів. JavaFX підтримує анімацію, роботу з мультимедіа та векторною графікою, що дозволяє створювати інтуїтивно зрозумілі й функціональні елементи керування для користувачів.

Комбінація Java та JavaFX гарантує високу якість розробки, зручність супроводу системи та ефективність її використання.

Основні етапи реалізації модулів системи. Розробка програмної системи передбачає виконання кількох ключових етапів:

1. Симуляція руху транспорту. Розробляються алгоритми моделювання транспортних потоків з урахуванням умов дорожнього руху: функціонування перехресть, розгалуження доріг, різні швидкісні режими та ймовірність утворення заторів. Це дає змогу аналізувати типові сценарії й оптимізувати поведінку учасників руху.

2. Адаптивне управління світлофорами. Створюються алгоритми, які реагують на зміну інтенсивності руху в режимі реального часу, автоматично регулюючи фази світлофорів для покращення пропускної здатності та зменшення заторів.

3. Моніторинг порушників. Реалізується система виявлення правопорушників, зокрема водіїв, які перевищують дозволену швидкість. Алгоритми фіксують порушення та передають дані для подальшої обробки.

4. Інтеграція бази даних. Створюється база даних для авторизації та реєстрації користувачів. У ній зберігається інформація про облікові записи, рівні доступу та інші необхідні атрибути користувача.

5. Система логування правопорушень. Розробляється механізм зберігання інформації про виявлені порушення, що дозволяє проводити подальший аналіз, формувати звітність та приймати відповідні заходи реагування.

У підсумку, реалізація кожного з зазначених етапів є невід'ємною складовою створення комплексної, ефективної та адаптивної системи інтелектуального керування рухом транспорту. Така система сприятиме покращенню управління дорожнім рухом, зменшенню заторів та підвищенню загального рівня безпеки на дорогах.

2.2 Розробка моделі системи

Для кращого розуміння функціонування програмного застосунку було прийнято рішення побудувати UML-діаграму діяльності [16], яка є важливим елементом процесу аналізу та проєктування системи інтелектуального керування рухом транспорту (рис. 2.1).

Діаграма діяльності дозволяє наочно відобразити послідовність дій та процесів, що відбуваються всередині системи, а також проаналізувати логіку їх виконання та взаємозв’язки між етапами. Її основне призначення - продемонструвати покрокове виконання конкретної функціональності або бізнес-процесу в рамках системи, що значно полегшує як розробку, так і подальше вдосконалення програмного продукту.

Ця діаграма діяльності демонструє роботу моделі системи.

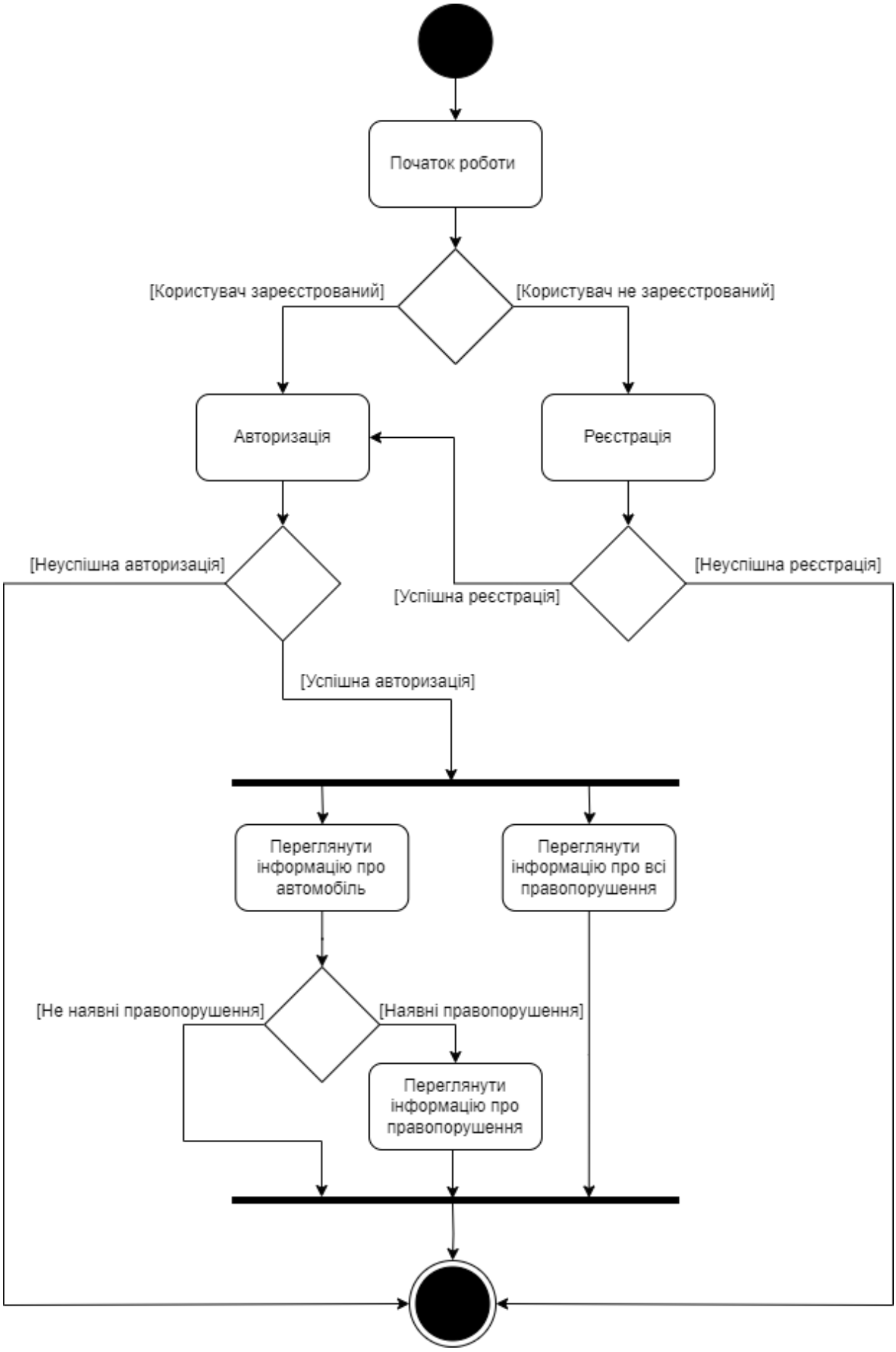


Рисунок 2.1. Діаграма діяльності

2.3 Розробка алгоритмів роботи системи

Одним із ключових етапів у процесі створення програмного застосунку для інтелектуального керування рухом транспорту є формування загального алгоритму функціонування системи.

На рис. 2.2 представлено блок-схему, яка відображає загальний алгоритм роботи системи, демонструючи основні етапи її функціонування та взаємозв’язок між ними.

На початковому етапі користувач проходить авторизацію. Якщо обліковий запис відсутній, система пропонує пройти процедуру реєстрації, після чого користувач здійснює вхід до системи.

Після успішної авторизації у головному вікні застосунку користувач спостерігає за появою транспортних засобів у межах видимої області панелі «Дорога». Автомобілі рухаються по дорозі, можуть зупинятися на перехрестях і зникати з поля зору при виїзді за межі панелі.

Користувач має змогу:

- переглянути детальну інформацію про транспортний засіб, натиснувши на нього;
- ознайомитися зі списком виявлених порушень, якщо такі є;
- переглянути повний реєстр усіх зафіксованих правопорушень.

Важливим компонентом функціонування системи є алгоритм автоматичного руху автомобіля. На рис. 2.3 представлено блок-схему, яка ілюструє логіку цього алгоритму.

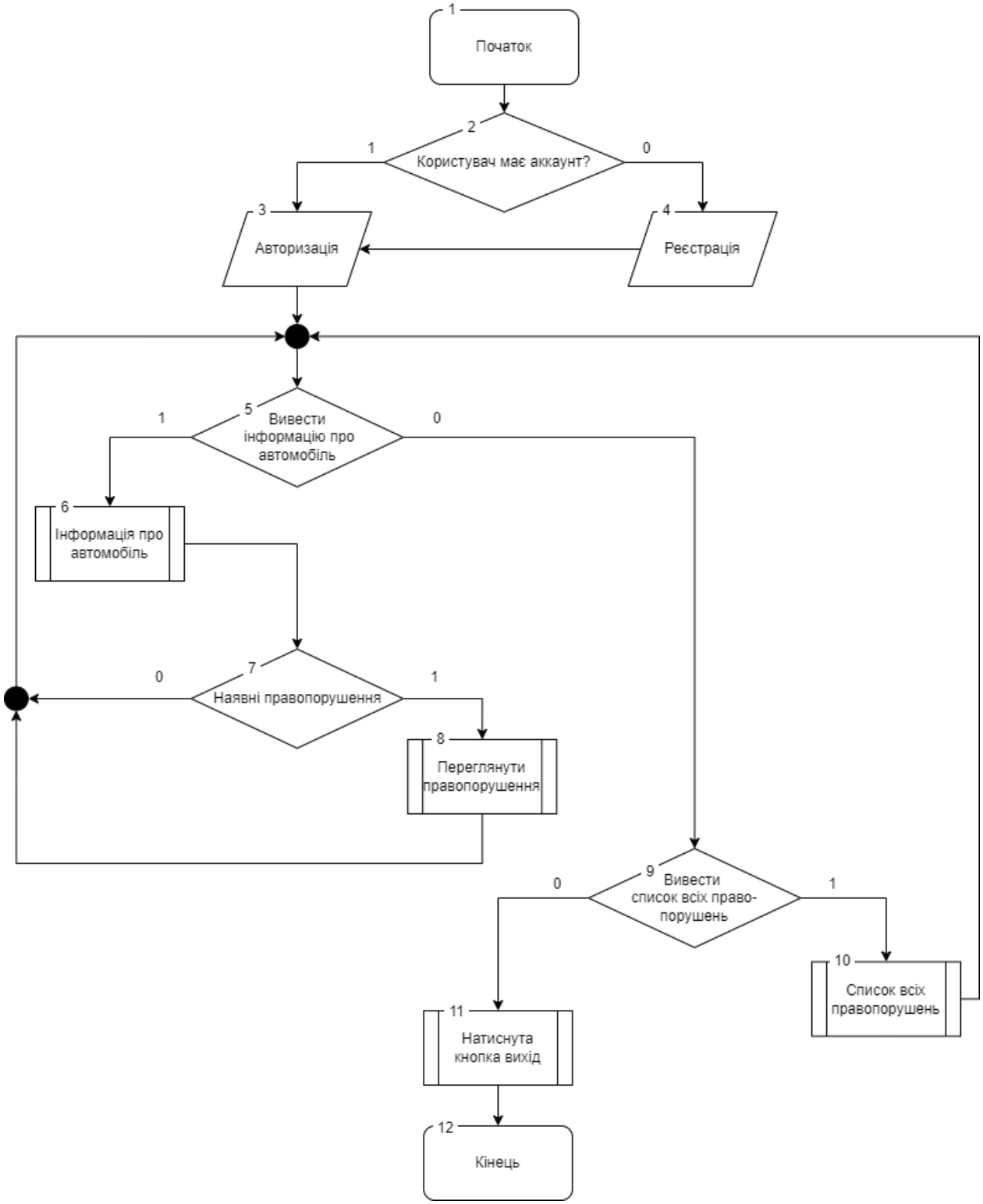


Рисунок 2.2. Блок-схема загального алгоритму роботи системи

Коли автомобіль з’являється у межах видимої області панелі, він розпочинає рух відповідно до заданого напрямку.

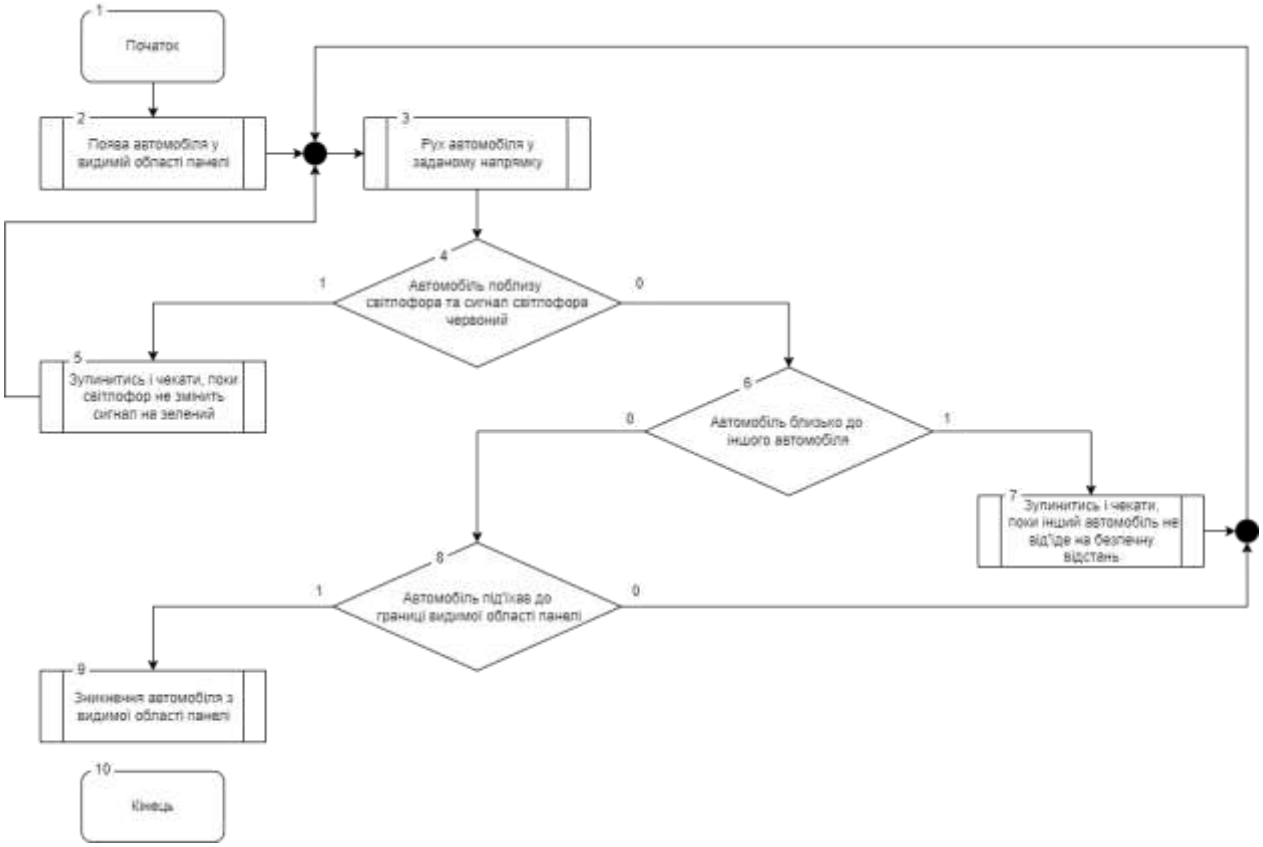


Рисунок 2.3. Блок-схема алгоритму автоматичного руху автомобіля

У разі наближення до світлофора з червоним сигналом, транспортний засіб зупиняється та очікує на зміну сигналу на зелений, після чого продовжує рух.

Якщо автомобіль під'їжджає до іншого транспортного засобу, що зупинився перед світлофором, він також зупиняється та очікує, поки попередній автомобіль не почне рух.

Коли автомобіль досягає межі видимої області, він зникає з поля зору користувача й завершує свій життєвий цикл у межах симуляції.

Таким чином, було реалізовано логіку загального алгоритму керування транспортним потоком, розроблено алгоритм автоматичного руху транспортних засобів, а також представлено відповідні блок-схеми для наочного відображення їхньої роботи.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ СИСТЕМИ ДЛЯ КЕРУВАННЯ РУХОМ ТРАНСПОРТУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного застосунку та розробка його інтерфейсу

Для реалізації програмного застосунку для інтелектуального керування рухом транспорту необхідно уважно обрати мову програмування та відповідні інструменти розробки.

C# - об'єктно-орієнтована мова програмування, створена компанією Microsoft для платформи .NET Framework. Вона має C-подібний синтаксис, сувору статичну типізацію та підтримує сучасні можливості, такі як перевантаження операторів, анонімні функції, узагальнені типи та делегати. Хоча C# не підтримує множинне спадкування класів (на відміну від C++), вона дозволяє реалізовувати множинне спадкування інтерфейсів.

Основні переваги C#: Автоматичне управління пам'яттю завдяки вбудованому збирачу сміття; LINQ - інтеграція запитів безпосередньо в код; Асинхронне програмування через ключові слова async та await; Подієво-орієнтована модель за допомогою подій та делегатів.

Python - високорівнева мова, що використовується в різноманітних галузях, включаючи веб-розробку, аналітику, машинне навчання тощо. Код у Python інтерпретується у байт-код, який виконується віртуальною машиною.

Ключові особливості Python: Простий синтаксис, зручний для початківців; Динамічна типізація, що прискорює написання коду; Автоматичне управління пам'яттю; Велика стандартна бібліотека; Підтримка кількох парадигм програмування.

Java - платформонезалежна мова, що дозволяє створювати кросплатформені застосунки завдяки використанню віртуальної машини Java (JVM). Java має широку екосистему бібліотек і фреймворків, що спрощують розробку.

Переваги Java: Статична типізація та виявлення помилок на етапі компіляції; Повна підтримка ООП; Мультипотоковість; Платформонезалежність завдяки байт-коду; Динамічне завантаження класів; Збирання сміття; Масштабованість для великих систем.

Java також відзначається надійністю та безпекою, має активну спільноту, що сприяє навчанню та вирішенню технічних питань.

З огляду на ці фактори, вибір Java як основної мови розробки програмного застосунку є доцільним та забезпечує надійність, масштабованість і безпеку.

Для створення графічного інтерфейсу користувача у середовищі Java можливе використання двох основних технологій: Swing і JavaFX.

Swing - це набір бібліотек Java для створення десктопних графічних інтерфейсів. Swing надає багатий набір компонентів (кнопки, текстові поля, таблиці тощо) та можливості для кастомізації зовнішнього вигляду. Він реалізує подієво-орієнтовану модель та має гнучку систему розташування елементів на формі.

JavaFX - сучасна платформа для побудови багатофункціональних інтерфейсів. Вона підтримує архітектуру MVC, має потужні графічні можливості та підтримує CSS-стилізацію.

Переваги JavaFX над Swing: Сучасний дизайн; CSS для стилізації компонентів; Модульна архітектура; Підтримка 3D-графіки; Мультимедіа та FXML для опису UI.

У результаті вибір JavaFX для реалізації інтерфейсу програмного застосунку виявляється найкращим варіантом.

Інтерфейс є критичним елементом взаємодії користувача із системою. Тому в межах цього проєкту було прийнято рішення створити зручний, адаптивний та візуально привабливий графічний інтерфейс.

Основним кольором обрано сірий, який забезпечує нейтральний, професійний вигляд інтерфейсу. Він добре підходить для фону, кнопок і тексту, сприяючи контрастності з яскравими елементами та поліпшуючи сприйняття інформації.

При запуску застосунку користувач потрапляє у вікно авторизації, де вводить свої облікові дані для доступу до функціоналу системи (рис. 3.1).

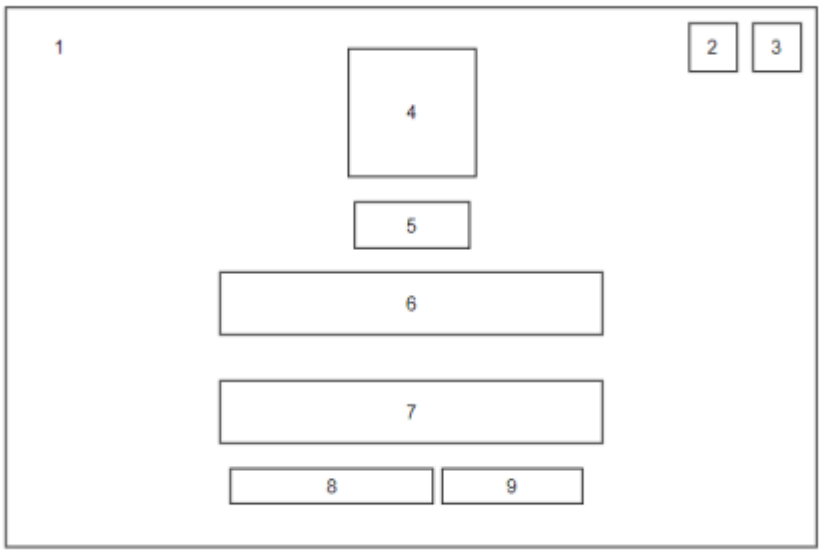


Рисунок 3.1. Схема вікна авторизації

На рис. 3.2 зображено вікно авторизації. Воно містить такі елементи: 1. Вікно; 2. Кнопка «Згорнути»; 3. Кнопка «Закрити»; 4. Логотип застосунку; 5. Заголовок «Sign in»; 6. Поле для введення логіну; 7. Поле для введення паролю; 8. Напис «Don’t have an account?»; 9. Посилання для переходу до реєстрації.

Якщо користувач не має облікового запису, він може створити його, натиснувши на посилання «Create», після чого відкривається вікно реєстрації.

На рис. 3.3 подано схему вікна реєстрації, яке включає такі елементи: 1. Вікно; 2. Кнопка «Згорнути»; 3. Кнопка «Закрити»; 4. Логотип застосунку; 5. Заголовок «Sign up»; 6. Поле для введення логіну; 7. Поле для введення email; 8. Поле для введення паролю; 9. Поле для підтвердження паролю; 10. Напис «Already have an account?»; 11. Посилання для повернення до авторизації.

На рис. 3.4 представлено вікно реєстрації. За потреби користувач може повернутися до вікна авторизації, натиснувши на посилання «Login». Після успішного завершення процесу реєстрації система автоматично перенаправляє користувача до вікна авторизації, де він може увійти до системи, використовуючи створені облікові дані.

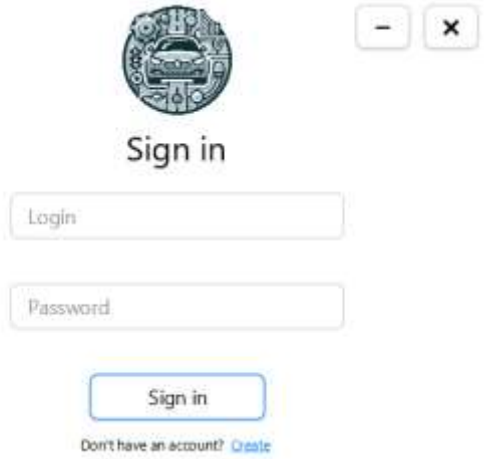


Рисунок 3.2 - Вікно авторизації

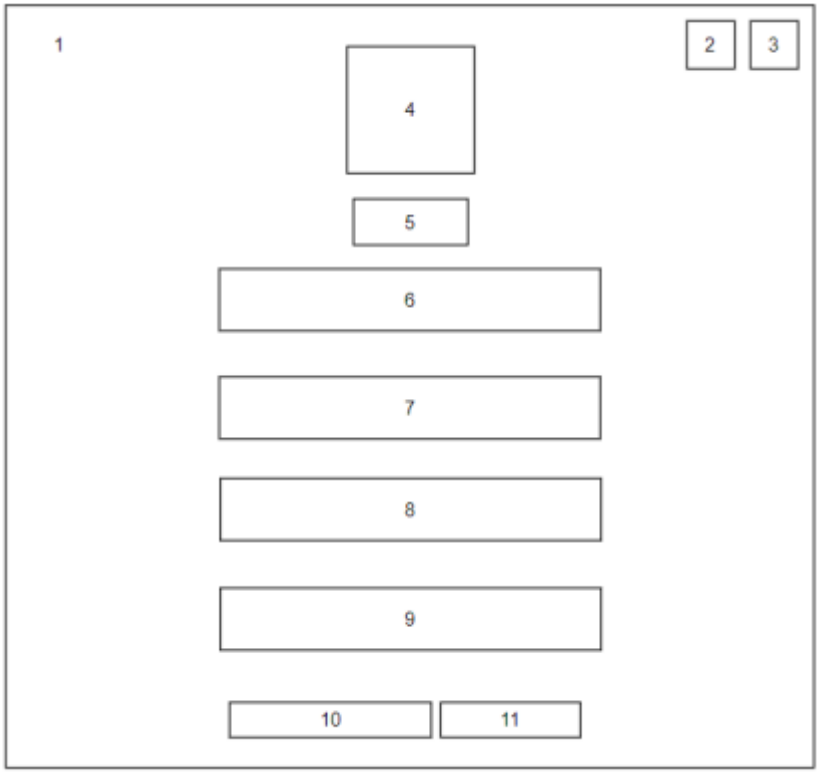


Рисунок 3.3. Схема вікна реєстрації

Після проходження авторизації відкривається головне вікно програми, яке є основним середовищем взаємодії користувача з функціоналом системи. Схематичне зображення головного вікна наведено на рис. 3.5.



Рисунок 3.4. Вікно реєстрації

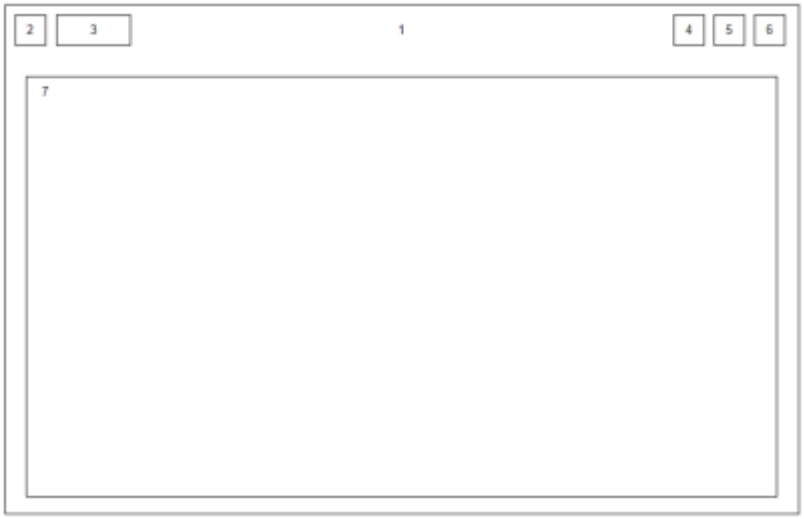


Рисунок 3.5. Схема головного вікна

На рис. 3.6 представлено головне вікно програми. Його структура включає такі елементи: 1 – Вікно, 2 – Кнопка «Меню», 3 – Нікнейм користувача, 4 – Кнопка «Переглянути список правопорушень», 5 – Кнопка «Згорнути», 6 – Кнопка «Закрити», 7 – Панель «Дорога».

Панель «Дорога» є центральним елементом інтерфейсу, що візуалізує рух транспортних засобів на дорожньому полотні та слугує основною зоною для взаємодії користувача з динамічними даними системи.



Рисунок 3.6. Головне вікно

При натисканні мишею по іконці автомобіля з’являється вікно, що дає додаткову інформацію про даний автомобіль.

На рис. 3.7 зображено схему інформаційного вікна.

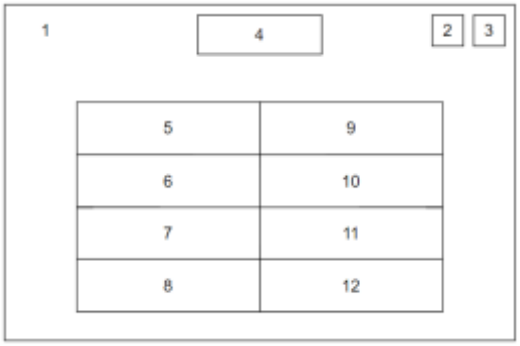


Рисунок 3.7. Схема інформаційного вікна про автомобіль

На рис. 3.8 представлено інформаційне вікно, яке відображає детальні відомості про обраний транспортний засіб. Структура вікна містить такі елементи: 1 – Вікно, 2 – Кнопка «Згорнути», 3 – Кнопка «Закрити», 4 – Заголовок «Car Info», 5 – Напис «Number», 6 – Напис «Model», 7 – Напис «Speed coefficient», 8 – Напис «Offences», 9 – Значення номера автомобіля, 10 – Значення моделі автомобіля, 11 – Значення коефіцієнта швидкості, 12 – Значення кількості зафіксованих правопорушень.

Це вікно забезпечує швидкий доступ до основних характеристик транспортного засобу та його історії порушень.



Рисунок 3.8. Інформаційне вікно про автомобіль

Якщо кількість правопорушень не дорівнює нулю, то при натисканні на відповідне поле відкривається вікно зі списком порушень, які були зареєстровані в базі даних для цього автомобіля. На рис. 3.9 наведено вікно перегляду правопорушень, пов’язаних із обраним транспортним засобом.



Рисунок 3.9. Схема вікна зі списком правопорушень

На схемі показано: 1 – Вікно, 2 – Кнопка «Згорнути», 3 – Кнопка «Закрити», 4 – Напис «Offences», 5 – Таблиця.

Таблиця містить такі заголовки: Type - тип правопорушення, Car Number - номер автомобіля, Car Model - марка автомобіля, Time - дата та час вчинення правопорушення.

На рис. 3.10 зображено це вікно.



Рисунок 3.10. Вікно зі списком правопорушень

Якщо користувач натисне кнопку «Переглянути список правопорушень» у головному вікні, відкриється аналогічне вікно зі списком, як і раніше, проте воно міститиме всі зареєстровані правопорушення, наявні в базі даних (рис. 3.11).



Рисунок 3.11. Вікно зі списком всіх правопорушень

Отже, було розроблено інтерфейс програмного застосунку для інтелектуального керування рухом транспорту.

3.2 Розробка бази даних

Для забезпечення можливості реєстрації та авторизації користувачів у програмному застосунку необхідно організувати збереження відповідних даних. З цією метою було прийнято рішення використовувати базу даних.

Розробка бази даних передбачає низку важливих етапів, зокрема: вибір системи управління базами даних (СУБД), проєктування та створення таблиць, підключення бази даних до застосунку.

Вибір СУБД. Для реалізації бази даних було обрано MySQL Workbench - інтегроване середовище для проєктування, моделювання та адміністрування баз даних MySQL. Воно забезпечує ефективну розробку завдяки таким ключовим перевагам:

1. Зручний графічний інтерфейс - інтуїтивно зрозумілий GUI дозволяє швидко створювати, редагувати та керувати об’єктами БД.

2. Моделювання структури даних - графічний редактор моделей дає змогу визначити структуру таблиць і зв’язки між ними.

3. Синхронізація з БД - полегшує оновлення схеми БД відповідно до моделі та навпаки.

4. Візуалізація SQL-запитів - SQL-редактор з можливістю миттєвого перегляду результатів.

5. Автоматизація задач - підтримка сценаріїв для резервного копіювання, планування завдань тощо.

6. Підтримка різних версій MySQL та MariaDB - забезпечує сумісність з різними СУБД.

З огляду на ці переваги, MySQL Workbench було обрано як основний інструмент розробки бази даних.

Структура бази даних. База даних використовується для зберігання: даних користувачів (авторизація та реєстрація), інформації про зафіксовані правопорушення.

Було створено дві сутності:

- users (user_id, username, email, password, status). Таблиця зберігає унікальний ідентифікатор користувача (user_id), облікові дані та поле status, що визначає права доступу до системи.
- offences (offence_id, type, car_number, car_model, time). Таблиця зберігає відомості про зафіксовані правопорушення, включаючи тип порушення, дані автомобіля та час події.

Підключення бази даних до застосунку. Для підключення MySQL до Java-застосунку необхідно:

1. Завантажити бібліотеку mysql-connector-j.jar.
2. Додати її до структури проєкту.
3. Реалізувати логіку підключення до бази даних у коді застосунку.

На рис. 3.12 представлено діаграму послідовностей, яка ілюструє процес підключення до бази даних у контексті роботи програмного застосунку.

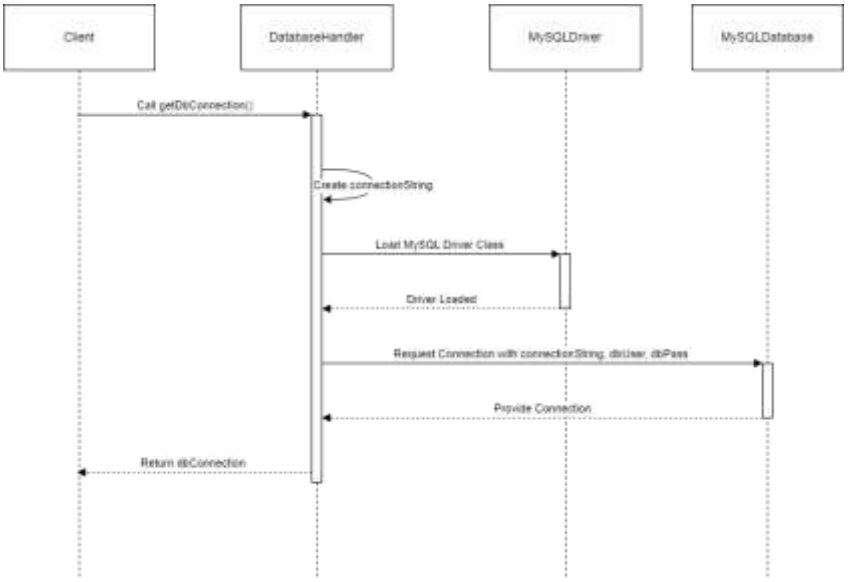


Рисунок 3.12. Підключення до бази даних

Після успішного підключення до бази даних програмний застосунок отримує можливість виконувати з нею різноманітні операції. Однією з ключових функцій є реєстрація нового користувача, яка передбачає виконання відповідного SQL-запиту. На рис. 3.13 представлено діаграму послідовності процесу реєстрації користувача.

Перед створенням нового облікового запису система виконує запит до бази даних з метою перевірки, чи існує вже введене ім’я користувача (username). Така перевірка необхідна для забезпечення унікальності ідентифікаторів користувачів у системі. Цей механізм: запобігає дублюванню облікових записів; мінімізує конфлікти при ідентифікації; підвищує надійність і безпеку процесу автентифікації; гарантує коректну роботу механізмів авторизації.

Таким чином, перевірка унікальності імені користувача є важливою частиною логіки реєстрації, що дозволяє забезпечити стабільність функціонування системи та комфортну взаємодію користувачів із застосунком.

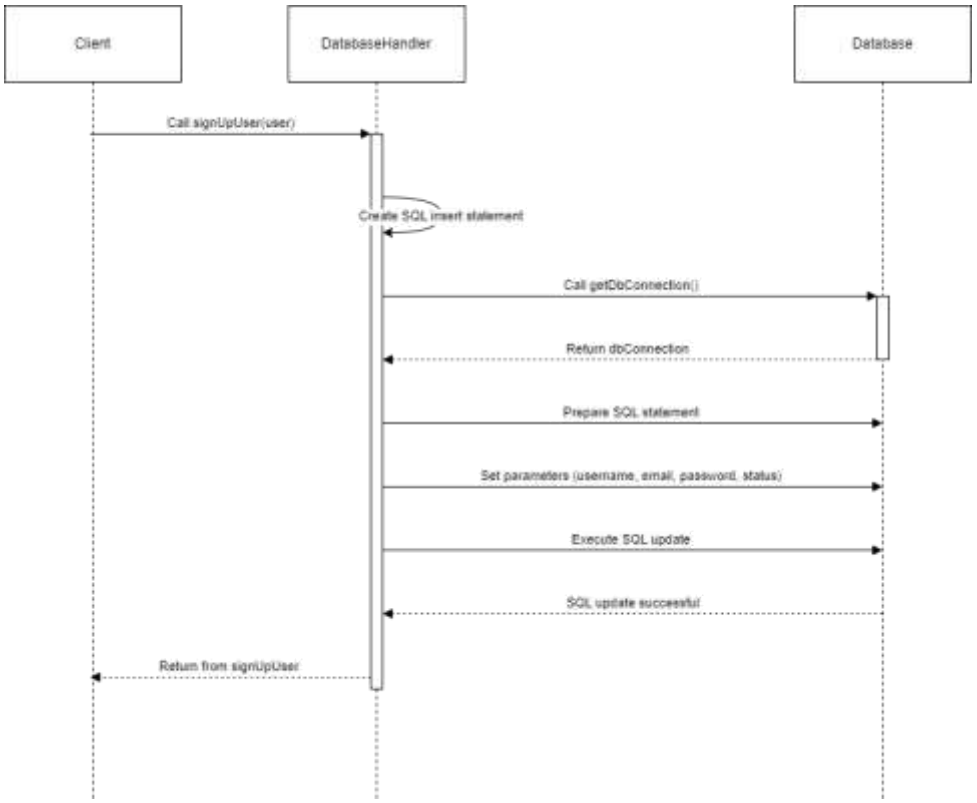


Рисунок 3.13. Реєстрація користувача

На рис. 3.14 представлено діаграму послідовності перевірки наявності імені користувача (username) в базі даних.

Одним з ключових етапів під час реєстрації нового користувача або оновлення його профілю є також перевірка унікальності електронної пошти (email). Відповідний запит до бази даних дозволяє переконатися, що введена електронна адреса ще не використовується в існуючих облікових записах. Така перевірка є важливою для:

- запобігання дублюванню облікових записів**;**
- забезпечення унікальності користувачів у системі**;**
- підвищення точності та достовірності даних, що зберігаються у базі;
- гарантування безперебійної роботи функцій відновлення паролю та сповіщень.

На рис. 3.15 зображено діаграму послідовності перевірки наявності електронної пошти в базі даних.

Під час авторизації користувача система виконує перевірку коректності введеного паролю шляхом відповідного запиту до бази даних. Це є вирішальним кроком для підтвердження особи користувача та забезпечення доступу до системи лише авторизованим особам.

На рис. 3.16 представлено діаграму послідовності перевірки паролю при авторизації.

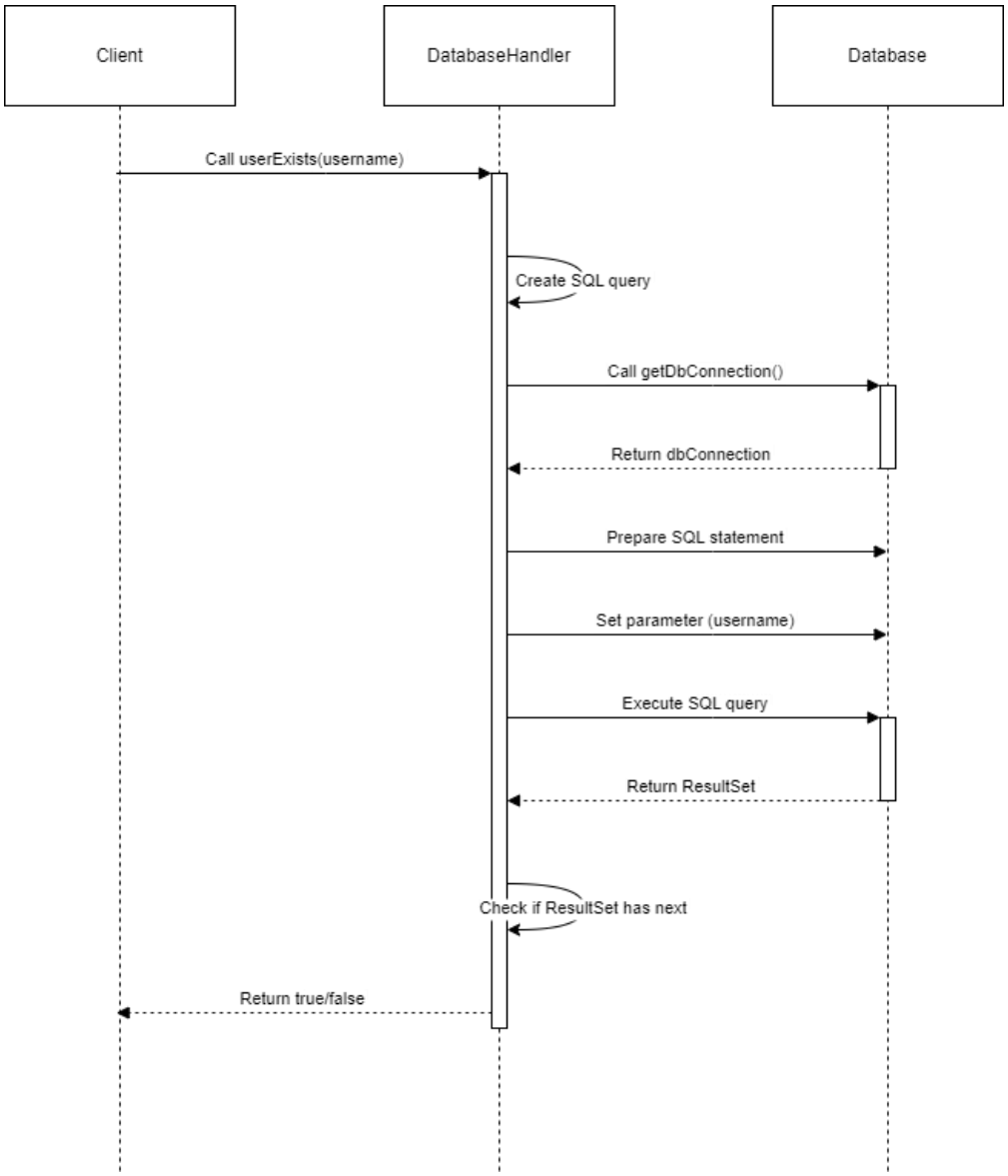


Рисунок 3.14. Перевірка існування username

Таким чином, було обґрунтовано вибір системи управління базами даних (СУБД), створено необхідні таблиці, здійснено підключення бази даних до програмного застосунку, а також описано ключові запити для взаємодії з базою даних.

Окрему увагу в системі приділено модулю адаптивного керування світлофорами, що є важливою складовою інтелектуального управління дорожнім рухом.

У стандартному режимі роботи світлофор змінює сигнал на протилежний кожні 15 секунд, при цьому таймер оновлюється щосекунди.

На рис. 3.17 наведено діаграму послідовності, яка ілюструє логіку зміни сигналу світлофора та періодичне оновлення таймера.

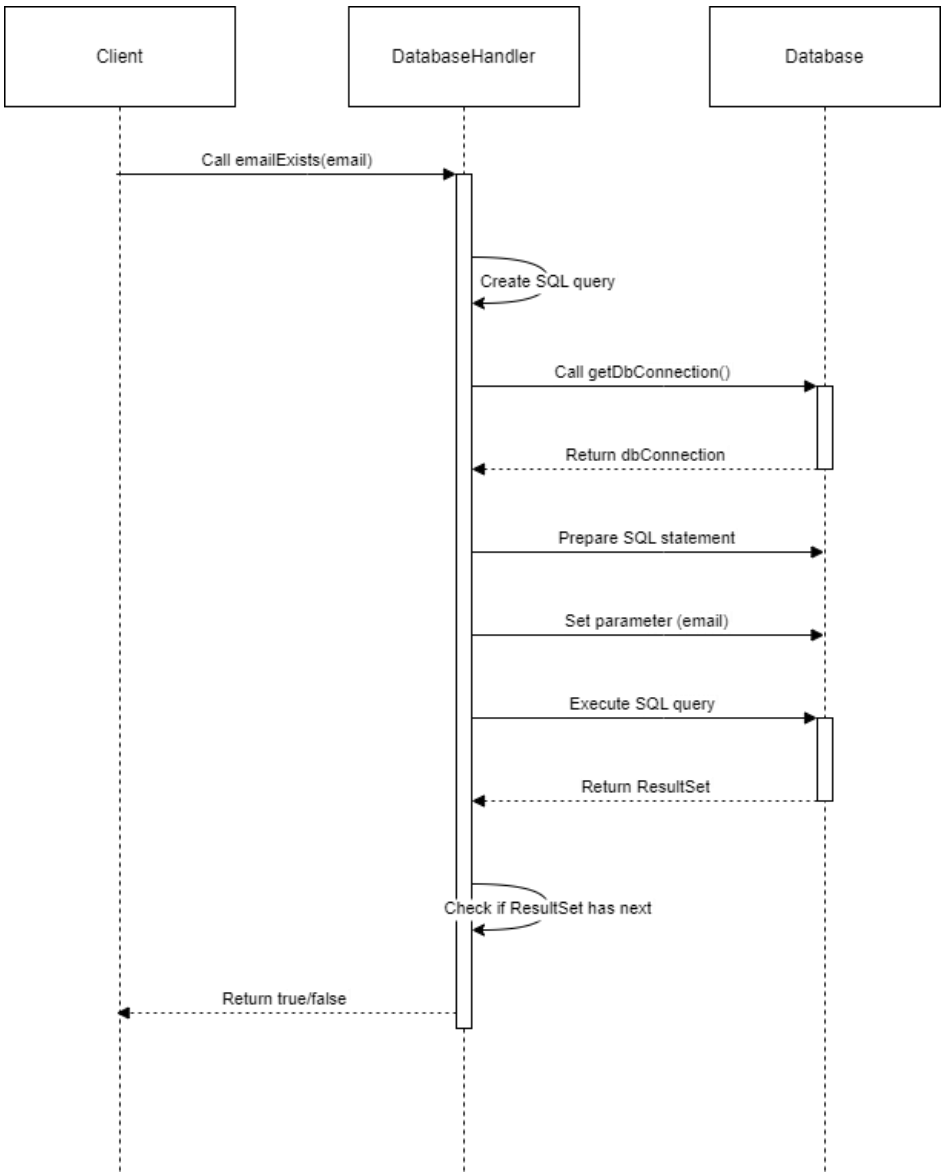


Рисунок 3.15. Перевірка існування email

Крім стандартного режиму роботи, світлофор обладнаний датчиком, який підраховує кількість автомобілів, що зупинилися на червоний сигнал. Якщо

кількість транспортних засобів перевищує чотири, система автоматично змінює сигнали світлофорів на перехресті з метою розвантаження дороги та запобігання заторам.

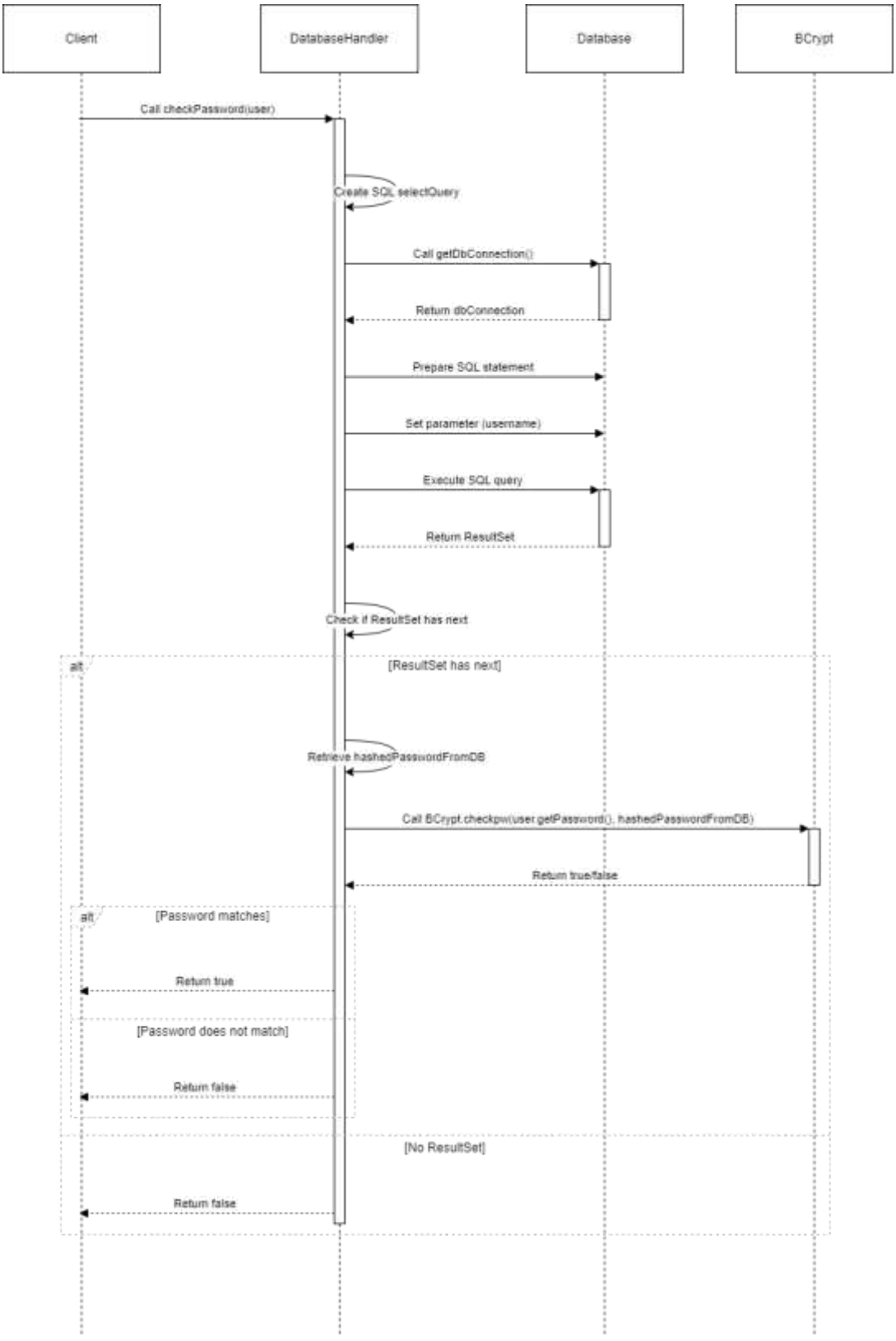


Рисунок 3.16. Перевірка паролю

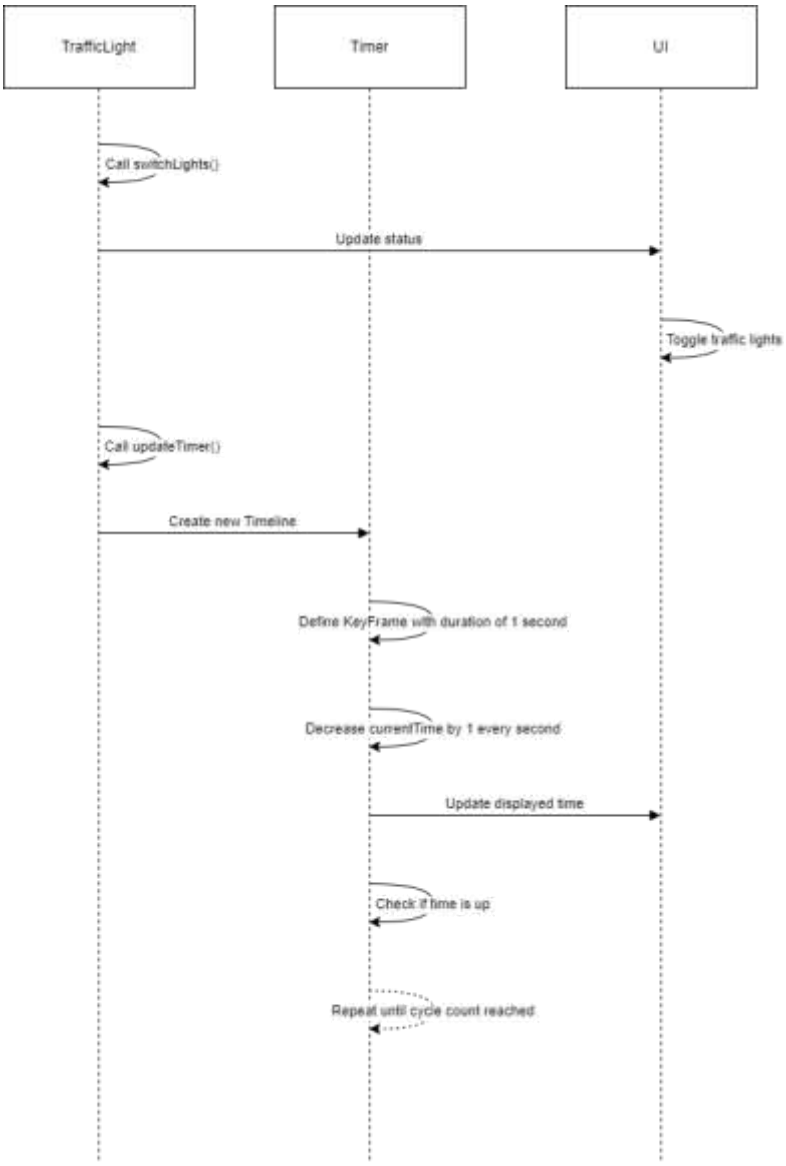


Рисунок 3.17. Зміна сигналу світлофора

На рис. 3.18 зображено діаграму послідовності, яка демонструє цю логіку адаптивного реагування.

Ще одним важливим модулем системи є моніторинг порушень правил дорожнього руху. Під час руху транспортного засобу існує ймовірність перевищення допустимої швидкості.

Для контролю цього параметра на дорозі встановлені спеціальні датчики, які фіксують факт перевищення швидкості. У випадку виявлення такого порушення система автоматично реєструє подію в базі даних, забезпечуючи збереження інформації, яка може бути використана для подальшого накладення штрафних санкцій.

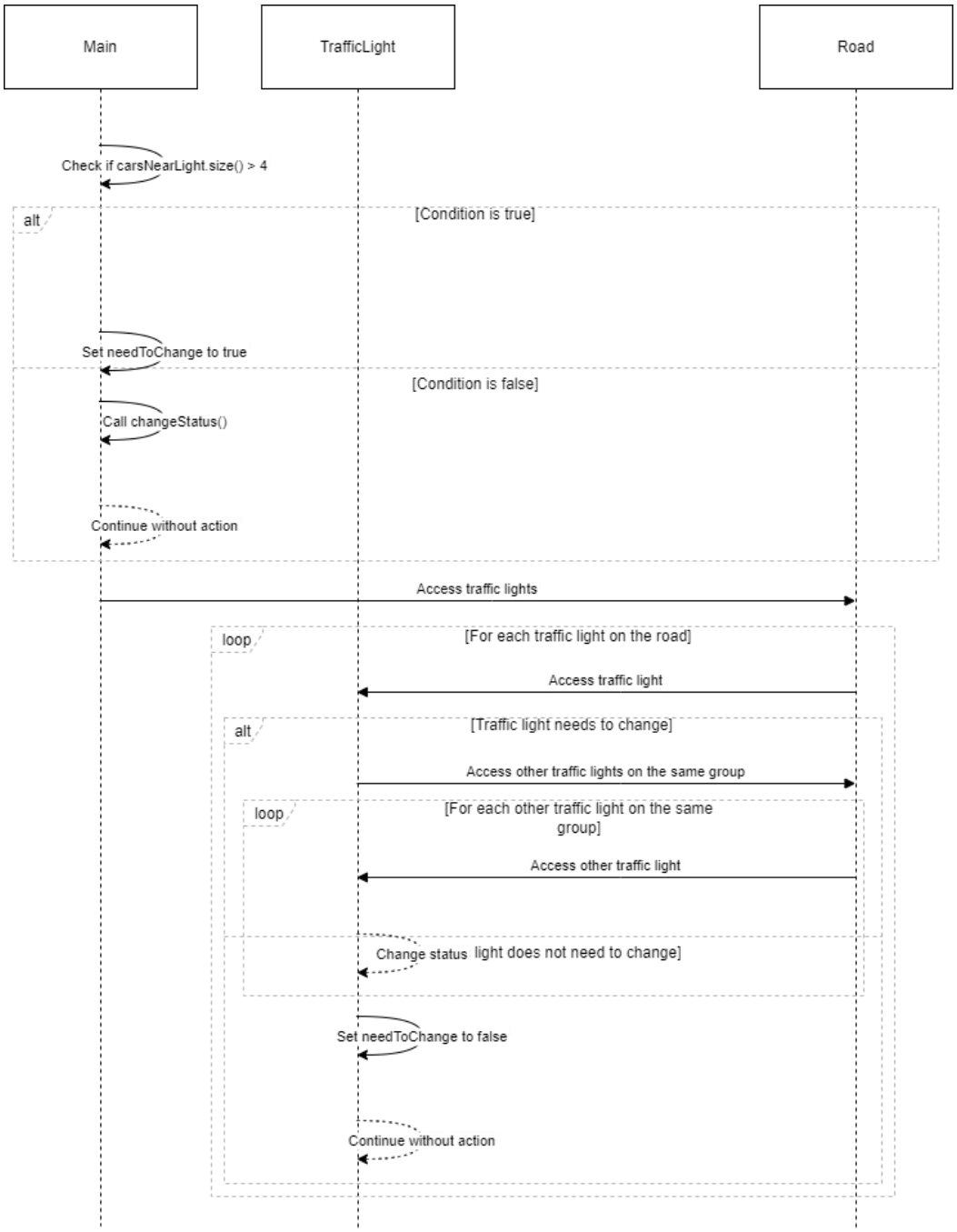


Рисунок 3.18. Адаптивне регулювання світлофора

На рис. 3.19 представлено діаграму послідовності фіксації правопорушення у базі даних.

Запис порушення до бази даних здійснюється у такому форматі: тип правопорушення (у цьому випадку – перевищення швидкості), номер автомобіля**,** марка автомобіля**,** дата та час фіксації події.

Таким чином, у рамках проєкту було реалізовано два ключові функціональні модулі:

– адаптивне керування світлофором, що дозволяє регулювати трафік у реальному часі;

– система моніторингу правопорушень, яка забезпечує автоматичне виявлення і реєстрацію фактів перевищення швидкості.

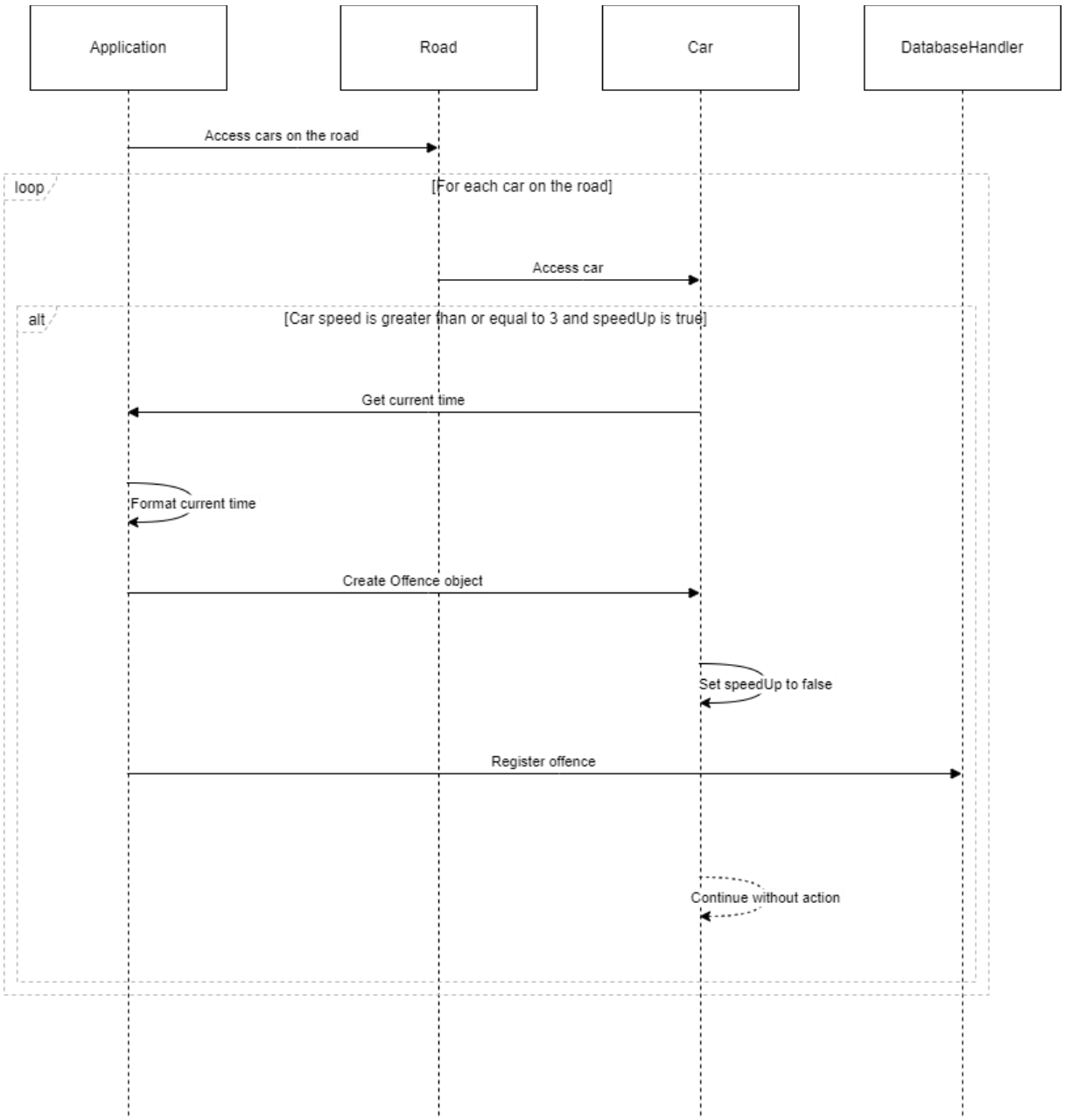


Рисунок 3.19. Фіксація правопорушення

3.3 Тестування системи

Тестування програмного забезпечення є невід’ємною частиною процесу розробки, адже саме на цьому етапі перевіряється якість продукту перед його

впровадженням або випуском на ринок. Основна мета тестування - виявлення і усунення помилок у програмному коді для забезпечення надійності, стабільності, ефективності та безпеки роботи системи.

Крім технічного аспекту, тестування гарантує, що програмний продукт відповідає функціональним вимогам, очікуванням користувачів і бізнес-цілям, що безпосередньо впливає на рівень задоволеності клієнтів і конкурентоспроможність на ринку. Також своєчасне тестування дозволяє ідентифікувати проблеми на ранніх етапах, знижуючи витрати на їх виправлення в майбутньому.

Основні методи тестування. Існує багато підходів до тестування ПЗ, кожен з яких має свої переваги і сфери застосування. Серед них:

1. Тестування «чорної скриньки» – орієнтоване на перевірку функціональності програми без знання внутрішньої структури. Основна увага приділяється відповідності очікуваній поведінці згідно з вимогами користувача.

2. Тестування «білої скриньки» – зосереджене на внутрішній логіці програми. Дозволяє детально перевірити алгоритми, цикли та гілки виконання для виявлення помилок, які не можна помітити ззовні.

3. Модульне тестування – передбачає окрему перевірку кожного модуля (функції, класу) на коректність роботи. Такий підхід дає змогу виявити помилки на ранньому етапі й забезпечити якісний фундамент для всієї системи.

4. Функціональне тестування – перевіряє, чи відповідає функціональність програмного забезпечення технічному завданню та потребам користувача. Тести виконуються для кожної функції системи.

5. Інтеграційне тестування – спрямоване на перевірку взаємодії між окремими модулями. Його мета - виявити помилки, що виникають при об’єднанні частин програми в єдину систему.

6. Системне тестування – охоплює перевірку всієї програми як цілісного продукту. Тестується вся функціональність у відповідності до вимог технічної документації.

7. Автоматизоване тестування – застосовує спеціалізовані інструменти для автоматизації процесу тестування. Це дозволяє ефективно проводити регресійне, повторюване чи навантажувальне тестування, значно економлячи ресурси.

Переваги методу «чорної скриньки». Обраним для даної системи методом є тестування «чорної скриньки», яке має низку вагомих переваг:

1. Незалежність від коду - тестувальник не потребує доступу до вихідного коду, що дозволяє перевірку навіть у випадках закритих систем.

2. Фокус на користувацькому досвіді - метод дозволяє оцінити інтерфейс та взаємодію користувача з програмою, що є ключовим чинником у визначенні її якості.

3. Ширше охоплення сценаріїв - тестувальник змушений мислити з позиції користувача, перевіряючи різні варіанти введення та поведінки.

4. Виявлення логічних помилок - дозволяє ідентифікувати несподівані помилки або аномалії, що не залежать безпосередньо від коду.

5. Реалістичність тестування - симулює реальні умови використання, забезпечуючи практичну перевірку роботи системи.

Зважаючи на зазначені переваги, для перевірки працездатності системи було обрано саме цей підхід.

Першим кроком у тестуванні стала перевірка авторизації користувача. Наприклад, якщо користувач намагається виконати вхід із порожніми полями, система відображає повідомлення: «Empty fields!», як показано на рис. 3.20.

При введенні неіснуючого Login буде виведене повідомлення «Login is wrong!» (рис. 3.21).

Якщо був введений неправильний пароль, буде виведене повідомлення «Password is wrong!» (рис. 3.22).

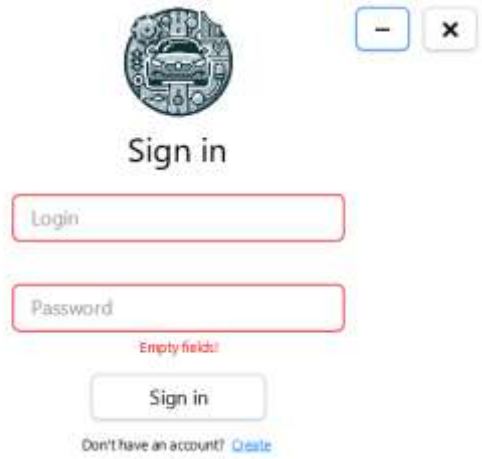


Рисунок 3.20. Порожні поля



Рисунок 3.21. Введення неіснуючого логіну

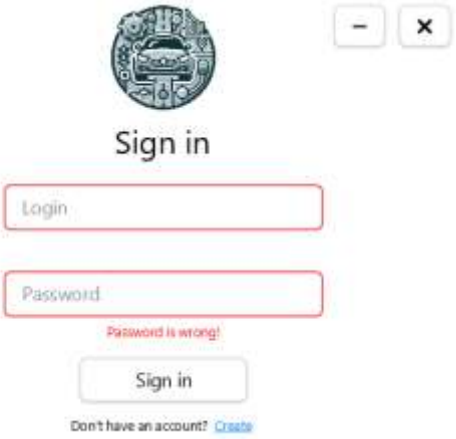


Рисунок 3.22. Введення неправильного паролю

У разі введення коректних даних для авторизації користувач успішно переходить до головного вікна застосунку.

Наступним етапом тестування є перевірка функціональності реєстрації. Якщо під час спроби створення облікового запису деякі поля залишено порожніми, система виведе повідомлення: «Empty fields!» (рис. 3.23).



Рисунок 3.23. Порожні поля

При спробі виконати реєстрацію з існуючим нікнеймом, буде виведене повідомлення «This username already in use!» (рис. 3.24).



Рисунок 3.24. Існуючий нікнейм

При спробі виконати реєстрацію з існуючим email, буде виведене

повідомлення «This email already in use!» (рис. 3.25).

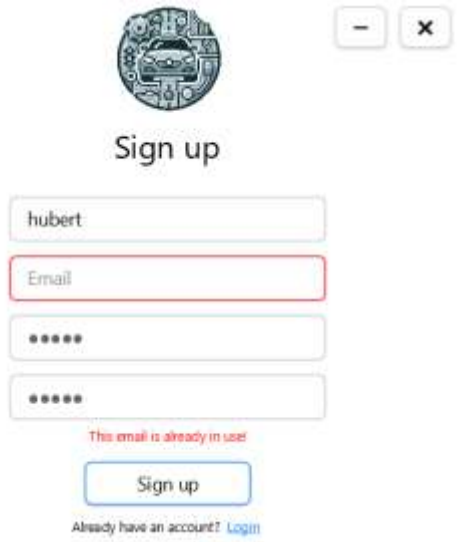


Рисунок 3.25. Існуючий email

Якщо при спробі реєстрації дані в полях «Пароль» та «Підтвердити пароль» не збігаються, буде виведене повідомлення «The passwords don’t match!» (рис. 3.26).

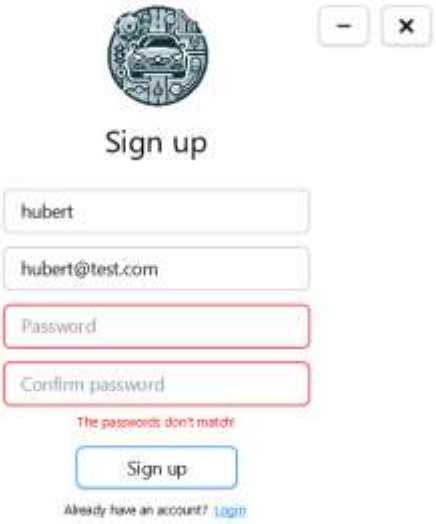


Рисунок 3.26. Паролі не збігаються

Якщо користувач успішно зареєструвався, буде виведене повідомлення «Success!» і через декілька секунд його перекине на вікно авторизації (рис. 3.27).

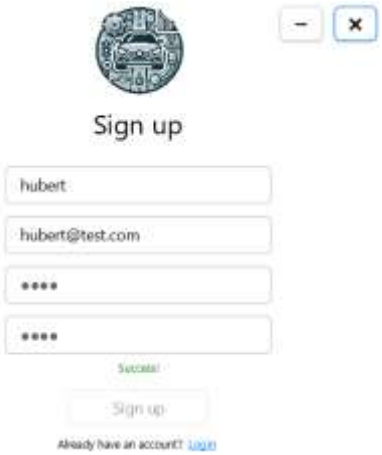


Рисунок 3.27. Успішна реєстрація

Після натискання кнопки «Меню» відкривається контекстне меню, яке дає змогу виконати вихід із системи (рис. 3.28).

За підсумками тестування було підтверджено, що система функціонує відповідно до вимог технічного завдання.

Щоб розпочати роботу з програмою, спочатку необхідно встановити Java Development Kit (JDK). Його можна завантажити з офіційного сайту Oracle або з іншого надійного джерела. Після завантаження потрібно запустити інсталяційний файл і виконати кроки, вказані в інсталяторі. Під час установки, як правило, буде запропоновано вказати шлях для інсталяції JDK (за замовчуванням), а також можуть бути доступні додаткові параметри для налаштування середовища.



Рисунок 3.28. Вихід з системи

Наступним етапом є встановлення виконуваного файлу застосунку з розширенням .exe. Файли такого типу призначені для запуску програм в операційній системі Windows і містять виконуваний код, що дозволяє ініціювати роботу програмного забезпечення на комп’ютері. Після завершення інсталяції користувач може запустити програму.

У табл. 3.1 наведено мінімальні системні вимоги, необхідні для коректної роботи всіх модулів застосунку.

Таблиця 3.1. Мінімальна конфігурація

Процесор	x64 1 ГГц
Обсяг оперативної пам’яті	2 ГБ для 64-разрядної системи
Вільний простір диску	100 MB
Операційна система	Windows 10

Таблиця 3.2. Рекомендована конфігурація

Процесор	x64 2 ГГц
Обсяг оперативної пам’яті	2 ГБ для 64-разрядної системи
Вільний простір диску	100 MB
Операційна система	Windows 11

ВИСНОВКИ

У кваліфікаційній роботі бакалавра було розроблено програмні модулі для інтелектуального керування дорожнім рухом. Основні завдання, виконані в межах проєкту: визначено загальну архітектуру та алгоритм функціонування системи; розроблено алгоритм симуляції руху транспорту; реалізовано адаптивний і зручний графічний інтерфейс; створено модуль для збереження та обробки даних; впроваджено систему моніторингу порушень швидкісного режиму; виконано тестування програмного забезпечення відповідно до поставлених завдань.

У першому розділі досліджено сучасний стан проблеми керування транспортними потоками, проведено аналіз існуючих рішень та обґрунтовано доцільність розробки власного застосунку. У ході аналізу методів вирішення задачі було обрано підхід із використанням агентних моделей, а також визначено основні функціональні компоненти системи.

У другому розділі здійснено аналіз вхідних даних і визначено етапи розробки програмних модулів. Створено модель системи на основі UML-діаграм діяльності, сформовано логіку загального алгоритму, розроблено алгоритм автоматичного руху транспортних засобів та представлено блок-схеми ключових процесів.

У третьому розділі проаналізовано інструменти та мови програмування, обґрунтовано вибір мови Java та графічної бібліотеки JavaFX для реалізації інтерфейсу. Розроблено повнофункціональний користувацький інтерфейс. Також обґрунтовано вибір системи управління базами даних, створено таблиці та реалізовано підключення до СУБД. Окрему увагу приділено розробці модуля адаптивного керування світлофорами та системи моніторингу порушень. Проведено аналіз методів тестування ПЗ, зокрема обрано метод «чорної скриньки», за допомогою якого було перевірено роботу системи. На завершення підготовлено інструкцію користувача для ефективного використання програмного застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Zear A., Kumar Singh P., Singh Y. Intelligent Transport System: A Progressive Review. Indian Journal of Science and Technology. 2016. Vol. 9, no. 32. [Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.17485/ijst/2016/v9i32/100713

2. A Reconfigurable Method for Intelligent Manufacturing Based on Industrial Cloud and Edge Intelligence / H. Tang et al. *IEEE Internet of Things Journal*. 2020. Vol. 7, no. 5. P. 4248–4259. [Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.1109/jiot.2019.2950048

3. Пліхта О.О. Переваги JavaFX над Swing у розробці інтерфейсу / О. О. Пліхта, Г.О. Черноволик // Матеріали LIII Науково-технічної конференції підрозділів Вінницького національного технічного університету, Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20695/17204

4. Analysis of Existing Problems and Solutions in Traffic Engineering Management. Foreign Language Science and Technology Journal Database Engineering Technology. 2021. [Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.47939/et.v2i9.252

5. Порівняльний аналіз аналогів [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Порівняльний_аналіз

6. SUMO [Електронний ресурс] – Режим доступу до ресурсу: https://eclipse.dev/sumo/

7. Simulation of Urban Mobility [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Simulation_of_Urban_MObility

8. MATSim [Електронний ресурс] – Режим доступу до ресурсу: https://www.matsim.org/

9. Multi-Agent Transport Simulation [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/MATSim

10. Aimsun [Електронний ресурс] – Режим доступу до ресурсу: https://aimsun.com/

11. Aimsun [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Aimsun

12. VISSIM [Електронний ресурс] – Режим доступу до ресурсу: https://www.ptvgroup.com/en/products/ptv-vissim

13. VISSIM [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/PTV_Vissim

14. Математична модель [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Математична_модель

15. Агентна модель [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Агентне_моделювання

16. Діаграма діяльності [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/mw7Jdke4

17. JVM [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Java_virtual_machine

18. Java [Електронний ресурс] – Режим доступу до ресурсу: https://www.oracle.com/java/

19. MySQL Workbench [Електронний ресурс] – Режим доступу до ресурсу: https://www.mysql.com/products/workbench/

20. Тестування методом чорної скриньки [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/Uw7Jf0mA