Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: Розробка методів і програмних засобів для інтелектуального керування рухом транспорту

Виконав: студент 4 курсу
групи 4ПІ-20Б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Пліхта О.О.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Черноволик Г.О.
(прізвище та ініціали)
Рецензент:. к.т.н., доц. каф ОТ Томчук М.А.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ _____
« 10 » червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 12 » березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Пліхті Олександру Олександровичу

1. Тема роботи – розробка методів і програмних засобів для інтелектуального керування рухом транспорту.

Керівник роботи: Черноволик Галина Олександрівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. №80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: операційна система — Windows 10, середовище розробки — IntelliJ IDEA, мова програмування — Java.

4. Зміст текстової частини: вступ, аналіз стану питання та постановка задач дослідження; розробка структури та алгоритмів програмного застосунку; розробка програмних модулів; тестування програмного застосунку; висновки; список використаних джерел.

5. Перелік ілюстративного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження; наукова новизна; порівняльний аналіз аналогів; розробка структури інтерфейсу програмного застосунку; діаграма діяльності моделі системи; блок-схема загального алгоритму роботи програмного застосунку; тестування модулів системи; апробація та публікація результатів роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Черноволик Г.О., к.т.н., доцент кафедри ПЗ	13.03.2024	03.06.2024

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

Ч.ч.	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання та постановка задач	14.03.24 – 05.04.24	вик.
2	Розробка структури та моделі системи	06.04.24 – 15.04.24	вик.
3	Розробка алгоритмів роботи програмного застосунку	16.04.24 – 28.04.24	вик.
4	Аналіз та вибір засобів для реалізації програми	29.04.24 – 05.05.24	вик.
5	Розробка програмних модулів	06.05.24 – 20.05.24	вик.
6	Тестування роботи розробленого програмного застосунку	21.05.24 – 26.05.24	вик.
7	Оформлення матеріалів до захисту БДР	27.05.24 – 30.05.24	вик.

Студент _____ **О. О. Пліхта**
(підпис) (ініціали та прізвище)

Керівник бакалаврської дипломної роботи _____ **Г. О. Черноволик**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 56 сторінок формату А4, на яких є 35 рисунків, 3 таблиць, список використаних джерел містить 20 найменувань.

У бакалаврській дипломній роботі 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 в бакалаврській дипломній роботі 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

The bachelor's thesis consists of 56 pages of A4 format, with 35 figures, 3 tables, and a list of references containing 20 titles.

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

ВСТУП

Обґрунтування вибору теми дослідження. У світі, де транспорт відіграє ключову роль у повсякденному житті людей і економіці, питання безпеки та ефективності руху транспорту стають все більш актуальними. Однією з основних складнощів, що виникають у цьому контексті, є необхідність управління рухом транспорту на дорогах з урахуванням різноманітних факторів, таких як об’єм транспортного потоку, безпека учасників руху та регулювання правил дорожнього руху.

У зв’язку з цим виникає потреба в розробці систем інтелектуального керування рухом транспорту, які б забезпечували ефективне та безпечне переміщення транспортних засобів [1]. Ці системи базуються на використанні сучасних технологій інформаційно-комунікаційних систем, аналізу даних та інтелектуальних алгоритмів для прийняття оптимальних рішень у реальному часі.

Існують різноманітні програмні застосунки, спрямовані на системи інтелектуального керування рухом транспорту [2], проте деякі з них можуть бути обмежені у функціональності. Тому є актуальним розробка власної системи інтелектуального керування рухом транспорту.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення безпеки на дорогах та якості керування рухом транспорту за рахунок використання методів і програмних засобів, що дозволить зменшити кількість аварій та інцидентів, покращити пропускну здатність доріг та скоротити час очікування для учасників дорожнього руху. Впровадження такої системи дозволить зменшити кількість аварій та інцидентів на дорогах, покращити пропускну здатність та зменшити час очікування для учасників дорожнього руху. Крім того, система буде сприяти

підвищенню рівня комфорту та безпеки на дорогах, що є важливим аспектом для сталого розвитку сучасних міст і регіонів.

Основними задачами, які необхідно виконати, є:

– визначити загальний алгоритм системи;

– розробити алгоритм симуляції руху транспорту на дорозі;

– розробити зручний та адаптивний графічний інтерфейс системи;

– розробити модуль для збереження даних;

– реалізувати систему моніторингу правопорушників, що перевищують швидкість;

– здійснити тестування системи згідно поставлених задач.

Об'єктом дослідження є процес розробки програмних модулів для інтелектуального керування рухом транспорту.

Предметом дослідження є алгоритми і засоби реалізації застосунку для управління рухом транспорту.

Методи дослідження. В роботі було використано наступні методи:

– метод побудови вікон за допомогою JavaFX для реалізації графічного інтерфейсу програмного застосунку;

– методи побудови блок-схем алгоритмів та діаграм;

– методи розробки програмних модулів;

Новизна отриманих результатів.

Подальшого розвитку отримала модель системи для інтерактивного керування рухом транспорту, яка, на відміну від існуючих, дозволяє більш ефективно оптимізувати транспортні потоки в реальному часі, що дає можливість зменшити затори та покращити ефективність використання транспортної інфраструктури.

Практична цінність отриманих результатів. Практична цінність полягає в можливості використання розробленої програми для системи інтелектуального керування рухом транспорту. Цей програмний продукт спрямований на оптимізацію управління транспортними потоками на дорогах, що дозволить ефективно використовувати дорожню інфраструктуру та зменшити ймовірність

аварій. Результати дослідження можуть бути застосовані для подальшого впровадження в сучасні міста та регіони з метою підвищення безпеки дорожнього руху та покращення транспортних послуг для населення.

Особистий внесок. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримано здобувачем самостійно.

Апробація матеріалів бакалаврської дипломної роботи. Матеріали бакалаврської кваліфікаційної роботи доповідались на науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії «LIII Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2024)».

Публікації. Результати роботи були опубліковані в матеріалах конференції «LIII Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2024)» [3].

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 стану питання керування рухом транспорту відображає складність та різноманітність викликів, що стикаються сучасні системи транспортного управління [4]. Порушення правил дорожнього руху, недостатня увага водіїв та технічні несправності автомобілів є лише деякими з факторів, які призводять до аварій.

1. Безпека дорожнього руху.

Проблема безпеки на дорогах стала актуальною через збільшення кількості аварій та інцидентів, які спричиняють травми та загибель учасників руху. Порушення правил дорожнього руху, водії, які не дотримуються швидкісного режиму, а також несправність транспортних засобів є основними причинами таких подій.

2. Затори та перенавантаження.

Збільшення кількості транспортних засобів на дорогах призводить до заторів та перенавантаження дорожньої інфраструктури. Це відбувається через обмежену пропускну здатність доріг та відсутність ефективних систем управління транспортними потоками.

3. Недооснащеність інфраструктури.

Брак інформаційно-комунікаційних технологій та неефективне використання інфраструктурних ресурсів призводить до обмежень у здатності системи керування рухом транспорту ефективно реагувати на зміни у транспортних потоках та управляти ними.

4. Невідповідність обмежень швидкості.

Порушення встановлених правил швидкісного режиму є поширеною проблемою, що призводить до збільшення кількості аварій та інцидентів на

дорогах. Недостатня контроль та недостовірність інформації про швидкість руху транспортних засобів також сприяють цьому.

Ці питання вимагають уваги та пошуку інноваційних рішень для поліпшення керування рухом транспорту, з метою забезпечення безпеки, ефективності та сталості транспортних систем.

Отже, розробка програмних модулів для розв'язання цих питань є важливим кроком у напрямку покращення систем керування рухом транспорту. Інтеграція інтелектуальних алгоритмів, аналізу даних та передових технологій дозволить створити систему, яка забезпечить не лише ефективне управління рухом транспорту, але й зменшить кількість аварій, покращить пропускну здатність доріг та сприятиме загальному покращенню безпеки та комфорту для учасників дорожнього руху. Такі програмні засоби мають потенціал стати основою для майбутніх інноваційних транспортних систем, що пристосовані до вимог сучасного міського життя та забезпечать стабільний та безпечний транспортний рух.

1.2 Порівняльний аналіз аналогів

Порівняльний аналіз аналогів є важливим етапом у дослідженні, спрямованому на розробку програмних засобів системи інтелектуального керування рухом транспорту [5]. Обґрунтування вибору конкретних рішень, визначення їх переваг і недоліків порівняно з аналогами, що вже існують на ринку — це важлива частина процесу розробки, оскільки дозволяє здійснити інформований вибір технологій та підходів, які найбільш відповідають потребам і вимогам проекту. Аналіз аналогів дозволяє отримати повніше уявлення про можливості іншими розробниками, зокрема щодо функціональності, ефективності, зручності використання та інших аспектів, які можуть бути враховані під час розробки власної системи.

Рішення, що можна назвати аналогами:

– SUMO (Simulation of Urban MObility).
– MATSim (Multi-Agent Transport Simulation).
– Aimsun.
– VISSIM.

SUMO (Simulation of Urban MObility) — це відкрите програмне забезпечення для симуляції дорожнього руху в міських середовищах від German Aerospace Center (рис. 1.1) [6]. Основна мета SUMO — дослідження та оптимізація транспортних систем, включаючи дороги, вулиці, перехрестя та різні види транспорту, такі як автомобілі, автобуси, трамваї та пішоходи [7].

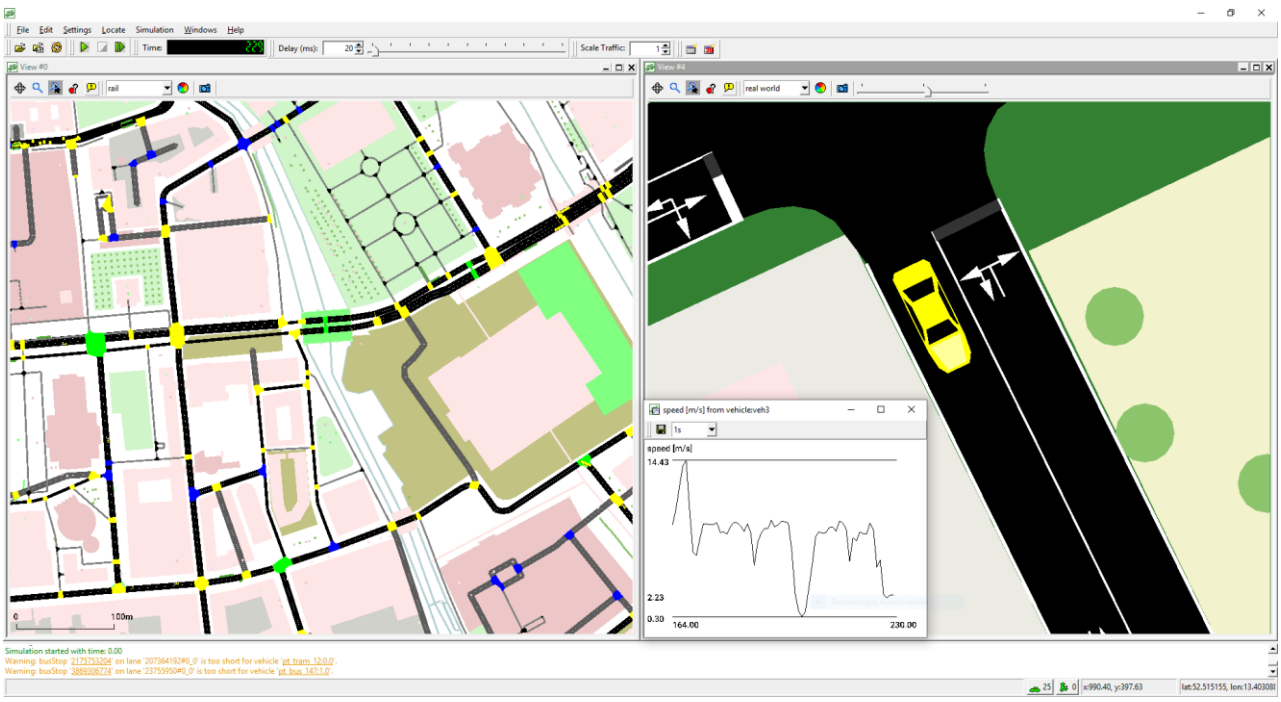


Рисунок 1.1 — SUMO

SUMO дозволяє користувачам створювати складні моделі транспортних систем, аналізувати їх ефективність та оптимізувати різні параметри, що робить його хорошим інструментом для дослідження та розвитку міської інфраструктури та транспортних систем.

MATSim (Multi-Agent Transport Simulation) — це відкрите програмне забезпечення для моделювання транспортних систем на основі агентів (рис. 1.2) [8]. Основна ідея MATSim полягає в тому, щоб кожен транспортний засіб (автомобіль, автобус, велосипед, пішохід тощо) був представлений окремим агентом, який рішає, куди рухатися в кожний момент часу, з урахуванням його власних переваг та обмежень, а також інших факторів, таких як трафік та доступність транспортних маршрутів [9].

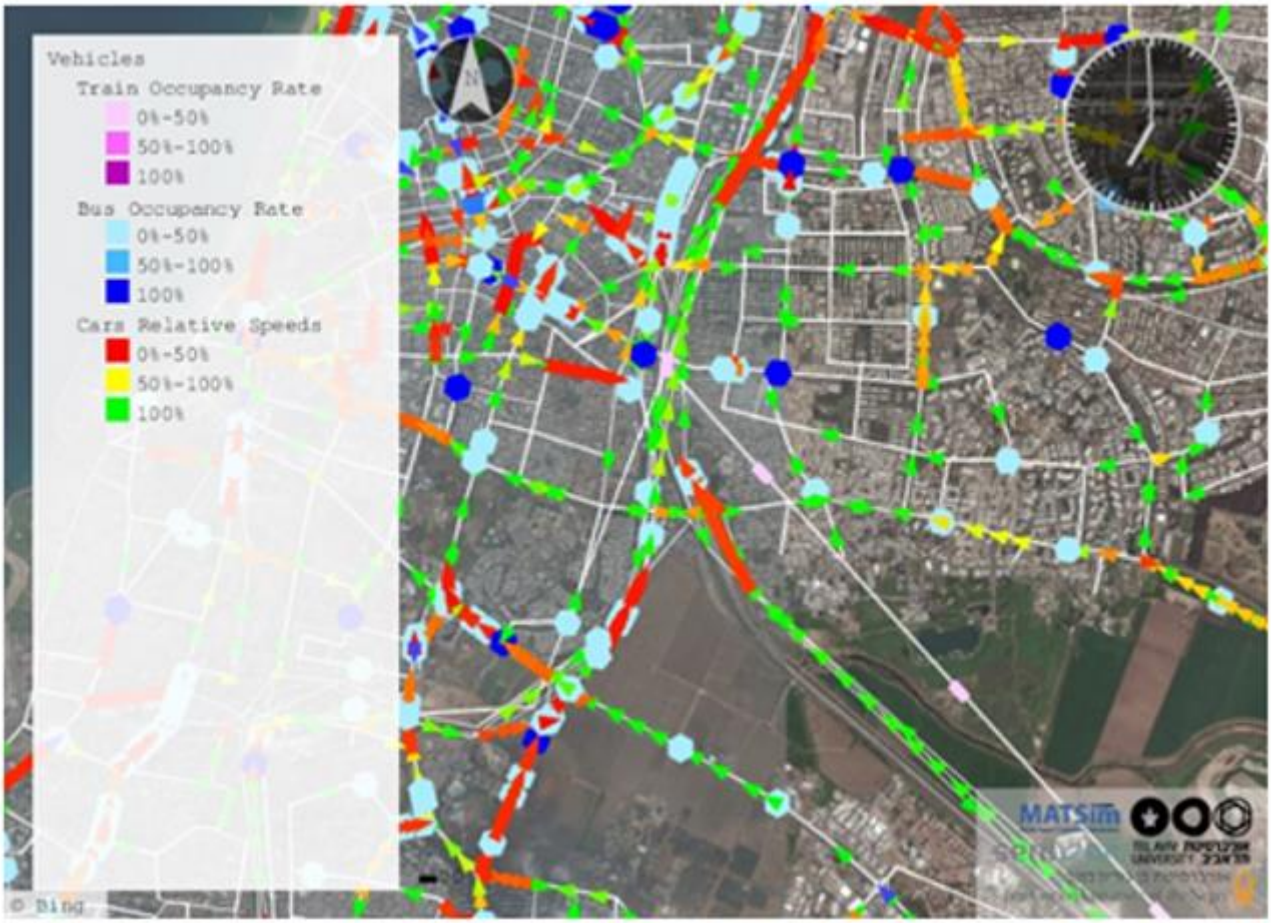


Рисунок 1.2 — MATSim

Aimsun — це комерційне програмне забезпечення для симуляції транспортних систем, що дозволяє моделювати та аналізувати рух транспортних засобів у міських, автодорожніх та інших середовищах (рис. 1.3) [10]. Основна мета Aimsun — надання інструментів для планування, проектування та

оптимізації транспортних систем з урахуванням різноманітних факторів, таких як трафік, інфраструктура доріг, розташування перехрестя та інше [11].

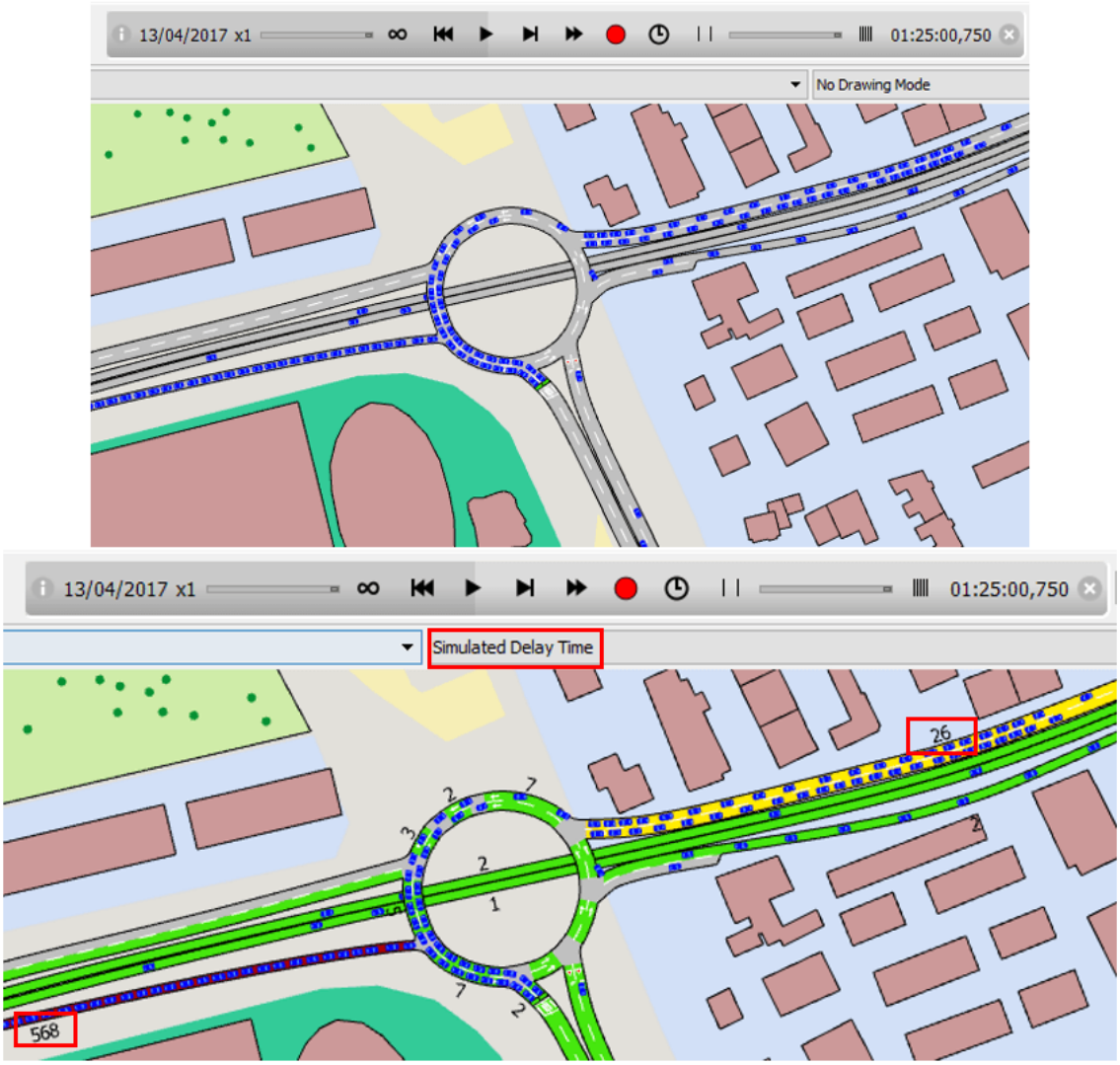


Рисунок 1.3 — Aimsun

VISSIM — це комерційне програмне забезпечення для симуляції та моделювання дорожнього руху та транспортних потоків (рис. 1.4) [12]. Воно використовується для аналізу, планування та оптимізації транспортних систем у міських, автодорожніх та інших середовищах. VISSIM дозволяє інженерам, планувальникам та дослідникам створювати реалістичні моделі транспортних систем та аналізувати їх ефективність, що робить його потужним інструментом для планування та управління дорожнім рухом [13].

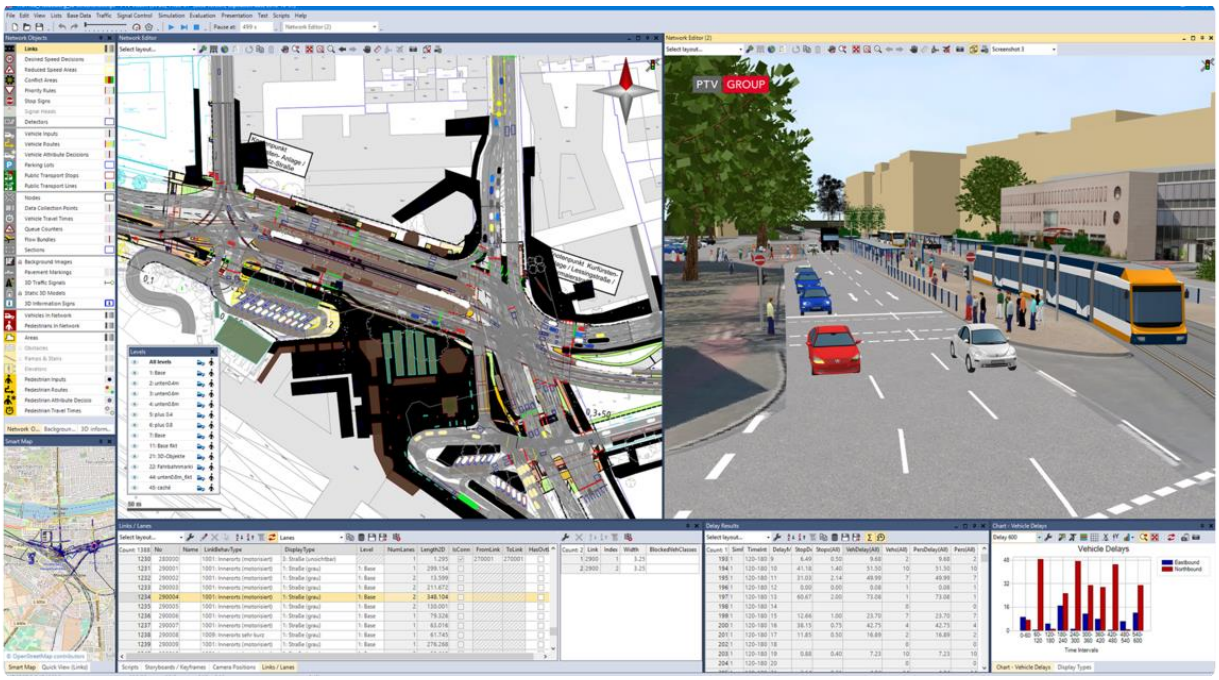


Рисунок 1.4 — VISSIM

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристика аналогів

Модулі / Критерії	SUMO	MATSim	Aimsun	VISSIM	TrafficT
Моделювання транспортного потоку	+	+	+	+	+
Оптимізація транспортного потоку	+	+	+	+	+
Сценарії та аналіз	+	+	+	+	+
База даних	–	–	–	–	+

Продовження таблиці 1.1

Модулі / Критерії	SUMO	MATSim	Aimsun	VISSIM	TrafficT
Моніторинг правопорушників	–	–	–	–	+
Загальна оцінка	60%	60%	60%	60%	100%

Отже, враховуючи недоліки існуючих рішень у сфері транспортної системи, розробка власного програмного продукту є доцільною. Новий продукт матиме потенціал покрити прогалини, що залишилися у вже існуючих рішеннях, і відкрити нові можливості для покращення користувацького досвіду. Розуміння потенційних переваг використання цього програмного забезпечення дозволить створювати інноваційні та ефективні системи для управління транспортом, що сприятиме їхньому успіху на ринку.

1.3 Аналіз методів розв’язання поставленої задачі

Задача розробки полягає в створенні програмного продукту для управління транспортними потоками, дослідження різноманітності підходів та методів, що застосовуються для вирішення проблеми управління рухом транспорту. Починаючи від традиційних алгоритмів і до сучасних інноваційних рішень, проводиться огляд методологій, які дозволяють оптимізувати ефективність транспортних систем та забезпечити покращення управління рухом транспорту. В аналізі враховуються такі аспекти, як точність передбачення трафіку, швидкість реакції на зміни в умовах руху, вплив на зменшення заторів та підвищення загальної безпеки дорожнього руху.

Перший підхід, який варто розглянути, це використання традиційних математичних моделей [14], що базуються на теорії чергових систем та оптимізації, для прогнозування та управління рухом транспорту. Ці методи зазвичай використовують статистичні дані та математичні моделі для розрахунку оптимальних маршрутів, часів подорожей та розподілу трафіку.

Другий підхід передбачає використання агентних моделей [15], де кожен транспортний засіб виступає як окремий агент, що приймає рішення на основі свого поточного стану та впливу навколишнього середовища. Ці методи забезпечують більш гнучкий та адаптивний підхід до управління рухом транспорту, але вимагають складних обчислень та моделювання великої кількості агентів.

Третій підхід — це використання сучасних технологій, таких як штучний інтелект та машинне навчання, для автоматизації процесу управління рухом транспорту. Ці методи дозволяють системі навчатися на основі даних та адаптуватися до змін в реальному часі, що сприяє покращенню ефективності та точності управління.

Аналізуючи різні методи розв'язання поставленої задачі, необхідно враховувати їхню ефективність, швидкодію та складність впровадження, зокрема в контексті потреб та обмежень конкретної транспортної системи.

Розглянувши різноманітні підходи до моделювання транспортної системи, їх переваги та недоліки, було вирішено використовувати агентну модель. Такий підхід дозволяє створити більш динамічну та адаптивну систему управління рухом транспорту, що відповідає сучасним вимогам ефективності та точності в управлінні транспортними потоками.

1.4 Постановка задач розробки програмного застосунку

Після проведення аналізу особливостей розробки програмного застосунку для інтелектуального керування рухом транспорту, було визначено наступні завдання, які необхідно виконати для розробки системи:

- розробити модель системи;
- визначити загальний алгоритм системи;
- розробити алгоритм симуляції руху транспорту на дорозі;
- розробити зручний та адаптивний графічний інтерфейс системи;
- розробити базу даних;

– реалізувати систему моніторингу правопорушників, що перевищують швидкість;

– здійснити тестування системи згідно поставлених задач.

Отже, було визначено, що для успішної розробки програмного застосунку для інтелектуального керування рухом транспорту необхідно виконати певний ряд ключових завдань, без яких система не може існувати.

1.5 Висновки

У першому розділі було розглянуто стан питання керування рухом транспорту. Також було проведено порівняльний аналіз аналогів, таких як SUMO (Simulation of Urban MObility), MATSim (Multi-Agent Transport Simulation), Aimsun, VISSIM з власним модулем, було доведено доцільність розробки власного програмного застосунку. Під час аналізу методів розв’язання поставленої задачі було вирішено використовувати агентні моделі. Було встановлено основні завдання, які необхідно виконати для розробки програмного застосунку.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ЗАСТОСУНКУ

2.1 Аналіз вхідних даних системи

Реалізація системи для інтелектуального керування рухом транспорту базується на мові програмування Java та використовує JavaFX для реалізації графічного інтерфейсу.

Мова програмування Java широко розповсюджена, вона має платформонезалежність та високу ефективність. Java є потужним інструментом для розробки різноманітних програмних продуктів, включаючи системи управління транспортними потоками. Вона має велику спільноту розробників та багато готових бібліотек та інструментів, що полегшують процес розробки.

JavaFX використовується для створення графічного інтерфейсу користувача (GUI). Ця технологія надає потужні засоби для реалізації інтерактивних та візуально привабливих інтерфейсів. JavaFX має вбудовану підтримку анімації, мультимедіа та векторної графіки, що дозволяє розробляти динамічні та ефективні користувацькі інтерфейси.

Мова програмування Java та використання JavaFX у системі для інтелектуального керування рухом транспорту забезпечує високу якість та ефективність розробленого програмного продукту, а також сприяє зручності у використанні та підтримці системи.

При розробці модулів для системи інтелектуального керування рухом транспорту необхідно виконати ряд етапів.

Першим етапом є симуляція руху транспорту на дорозі. Для цього необхідно розробити алгоритми, які враховують різноманітні умови дорожнього руху, такі як рух на перехрестях, розгалуження доріг, різні швидкості руху транспортних засобів та можливість утворення заторів. Симуляція дозволить ефективно моделювати та аналізувати різні сценарії руху транспорту.

Другий етап полягає в адаптивному керуванні світлофорами на дорозі. Для цього розробляються алгоритми, що враховують поточний стан руху на дорозі

та роблять відповідні корекції у роботі світлофорів для оптимізації транспортного потоку та зменшення заторів.

Третій етап передбачає реалізацію системи моніторингу правопорушників. Для цього потрібно розробити алгоритм, який виявляє порушників, зокрема тих, хто перевищує швидкість.

Четвертий етап включає підключення бази даних для авторизації та реєстрації користувачів. Це передбачає створення бази даних, яка зберігатиме інформацію про користувачів, їх облікові записи та права доступу до системи.

На п'ятому етапі необхідно реалізувати систему логування правопорушників. Це означає розробку механізму, який буде реєструвати інформацію про порушення правил дорожнього руху, зокрема перевищення швидкості, та зберігати цю інформацію для подальшого аналізу та вжиття необхідних заходів.

Отже, розробка програмних модулів — це завдання з декількох етапів, виконання яких дозволить створити комплексну та ефективну систему для інтелектуального керування рухом транспорту, яка забезпечить покращення управління дорожнім рухом та забезпечить більш безпечне та ефективне переміщення транспортних засобів.

2.2 Розробка моделі системи

Для кращого розуміння роботи програмного застосунку було вирішено створити UML діаграму діяльності [16], яка є важливою складовою частиною аналізу та проектування програмної системи для інтелектуального керування рухом транспорту (рис. 2.1).

Діаграма діяльності дозволяє візуалізувати послідовність дій та процесів, які відбуваються в системі, та допомагає зрозуміти їхню логіку та взаємозв’язки. Основна мета діаграми діяльності — показати послідовність кроків і дій, які відбуваються в рамках конкретної функціональності або процесу.

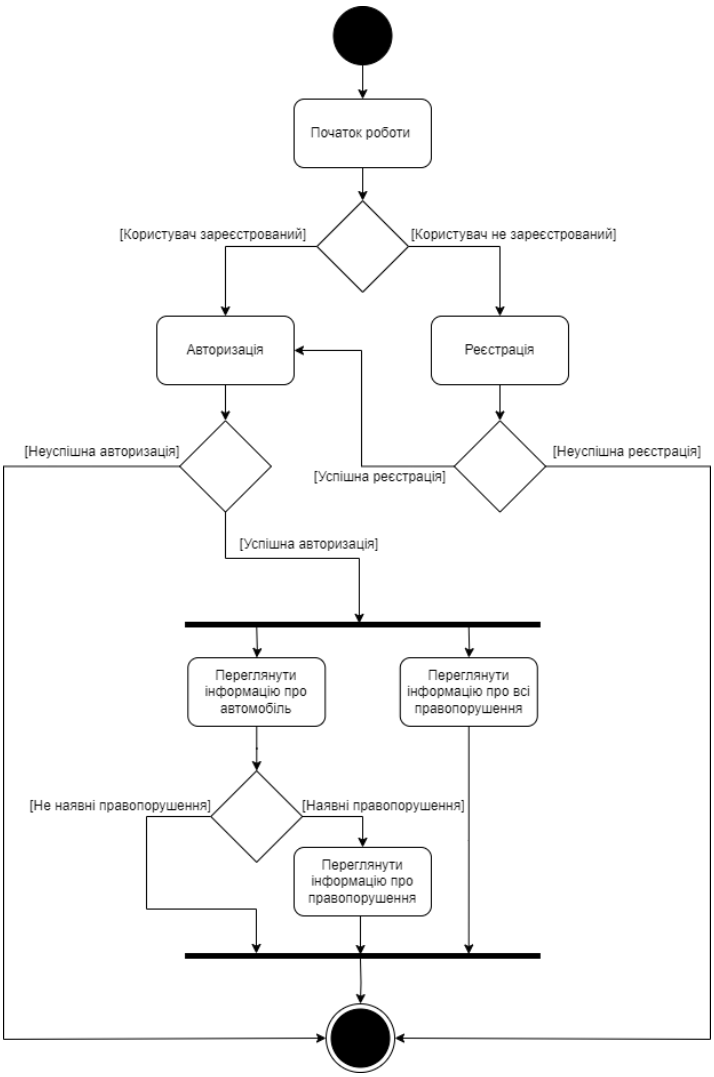


Рисунок 2.1 — Діаграма діяльності

Ця діаграма діяльності демонструє роботу моделі системи.

2.3 Розробка алгоритмів роботи системи

Одним із основних завдань при розробці програмного застосунку для інтелектуального керування рухом транспорту є розробка загального алгоритму роботи системи.

На рисунку 2.2 зображено блок-схему загального алгоритму роботи системи.

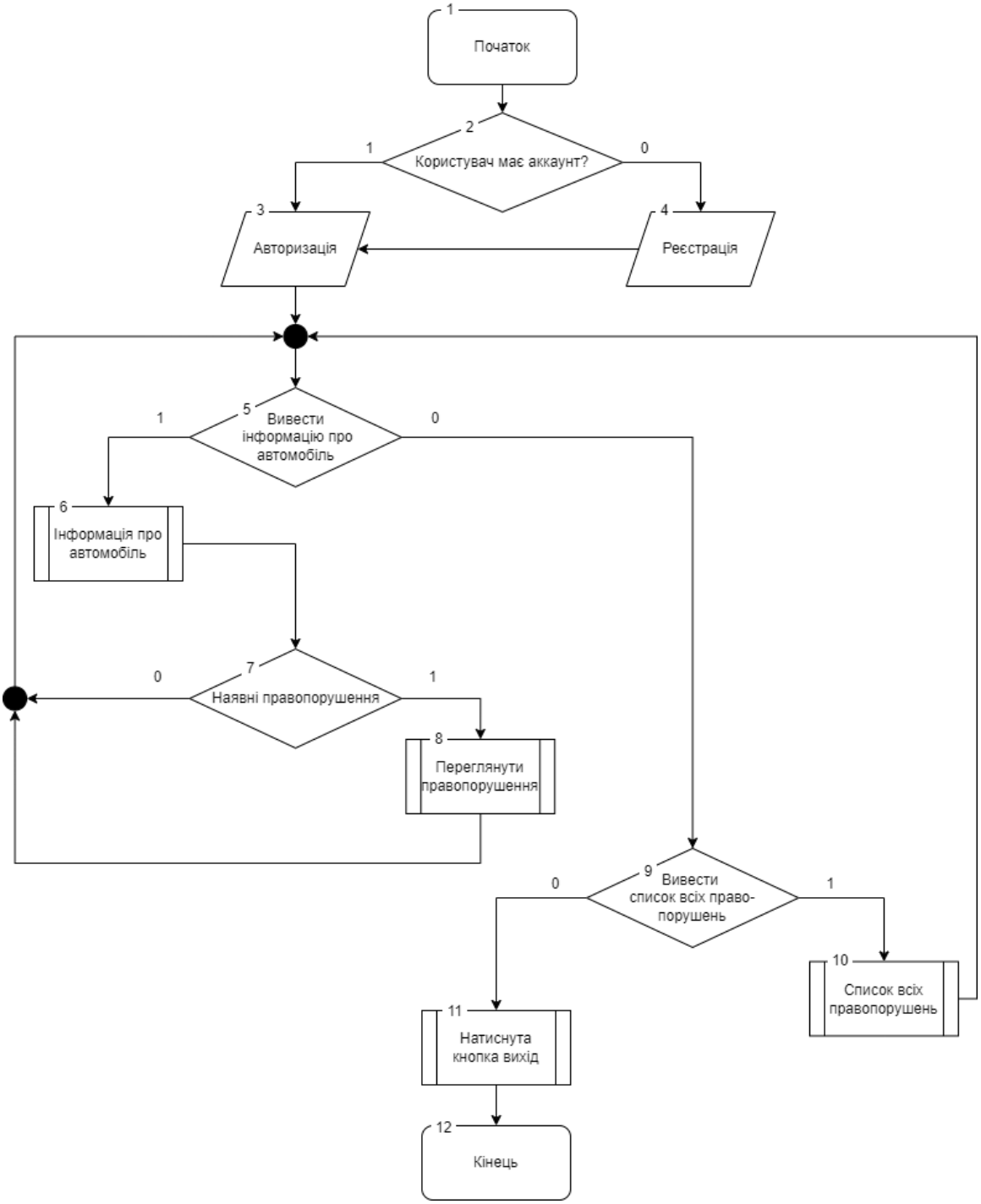


Рисунок 2.2 — Блок-схема загального алгоритму роботи системи

Спочатку користувач проходить процес авторизації. Якщо користувач не має особистого аккаунту, то спочатку він проходить процес реєстрації, а вже потім процес авторизації.

Після цього користувач може спостерігати на головному вікні появу автомобілів у видимій області панелі «Дорога», їх рух по дорозі, можливі зупинки на перехрестях та зникнення з видимої області панелі.

Також користувач може переглянути інформацію про автомобіль при натисненні на нього, переглянути список правопорушень, якщо такі є, та переглянути повний список всіх правопорушень.

Наступним важливим алгоритмом є алгоритм автоматичного руху автомобіля.

На рисунку 2.3 зображено блок-схему алгоритму автоматичного руху автомобіля.

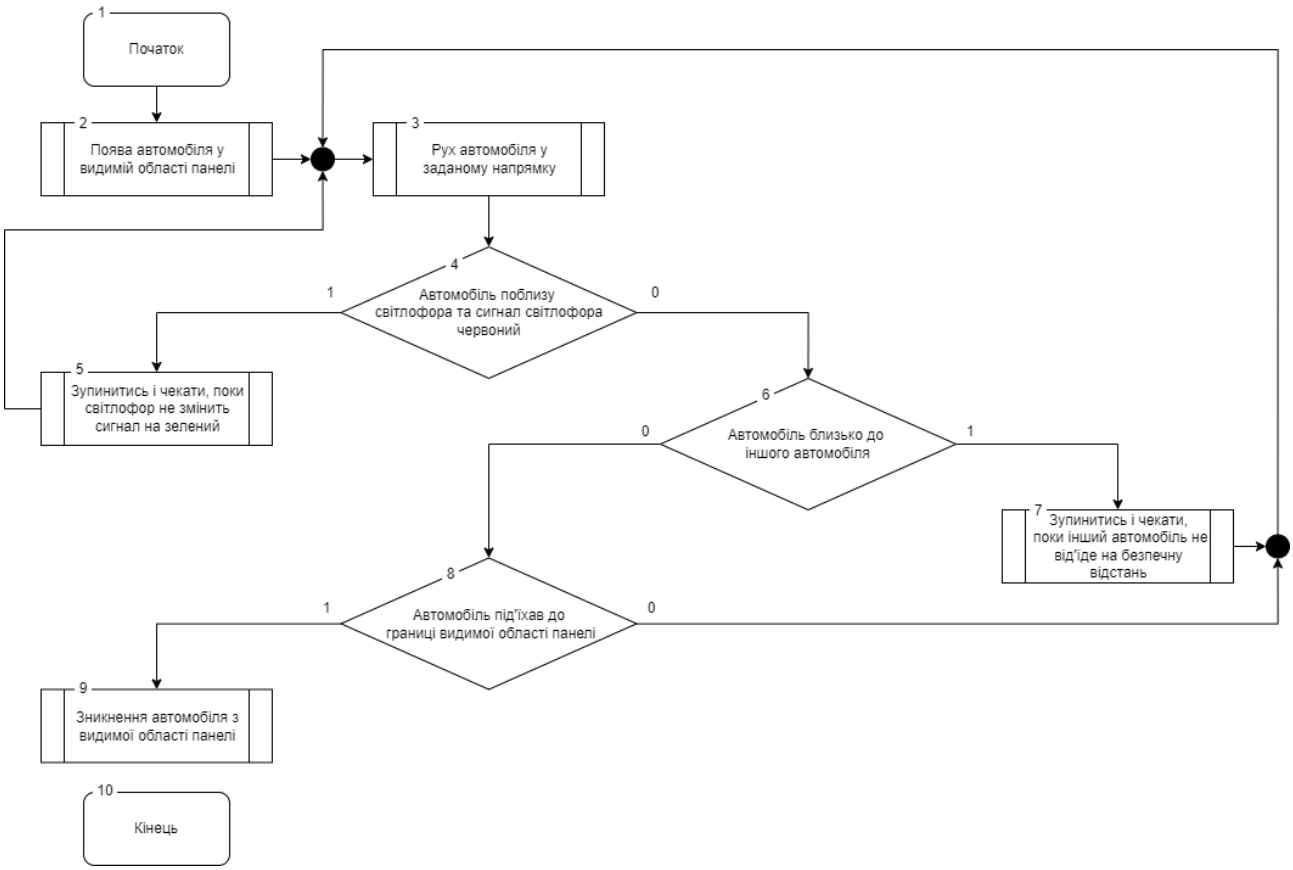


Рисунок 2.3 — Блок-схема алгоритму автоматичного руху автомобіля

Коли автомобіль з’являється в у видимій області панелі, він починає виконувати рух по своєму заданому напрямку.

Якщо автомобіль знаходиться біля світлофора і сигнал цього світлофора червоний, він зупиниться і буде чекати, поки сигнал світлофора не стане зеленим, після чого продовжить свій рух.

Якщо автомобіль під’їхав до іншого автомобіля, який стоїть на світлофорі, він зупиняється і чекає, поки інший автомобіль не почне свій рух.

Якщо автомобіль доїхав до границі видимої області панелі, то він зникає з поля зору наглядача та завершує свій життєвий цикл.

Отже, було розроблено логіку загального алгоритму системи для керування рухом транспорту, розроблено алгоритм автоматичного руху автомобіля, представлено блок-схеми алгоритмів.

2.4 Висновки

У другому розділі було проведено аналіз вхідних даних системи, визначено етапи розробки програмних модулів. Було розроблено модель системи, що відображена за допомогою UML діаграми діяльності. Було розроблено логіку загального алгоритму системи для керування рухом транспорту, розроблено алгоритм автоматичного руху автомобіля, представлено блок-схеми алгоритмів.

3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ СИСТЕМИ ДЛЯ КЕРУВАННЯ РУХОМ ТРАНСПОРТУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного застосунку

Для реалізації програмного застосунку для інтелектуального керування рухом транспорту слід ретельно розглянути можливі варіанти мов програмування та інструментів.

C# – об’єктно орієнтована мова, розроблена компанією Microsoft спеціально для використання можливостей .NET Framework. C# відноситься до мов програмування з C-подібним синтаксисом. Мова має строгу статичну типізацію, підтримує перевантаження операторів, анонімні функції, узагальнені типи і методи, вказівники на члени функції класів. C# не підтримує множинне спадкування класів, на відміну від C++, але підтримує множинне спадкування інтерфейсів.

Основні особливості мови програмування C#:

1. Автоматичне управління пам'яттю: C# використовує збирач сміття (Garbage Collector), який автоматично звільняє пам'ять, що більше не використовується, що зменшує ймовірність витоків пам'яті.
2. LINQ (Language Integrated Query): інтеграція мови запитів, що дозволяє писати запити до колекцій, баз даних, XML та інших джерел даних безпосередньо в коді C#.
3. Асинхронне програмування: підтримка асинхронних методів за допомогою ключових слів async і await, що спрощує написання асинхронного коду.
4. Події та делегати: механізми для обробки подій і делегатів, що дозволяють визначати і викликати методи через посилання на них, що корисно для створення подієво-орієнтованих програм.

Python – високорівнева мова програмування, як призначена для створення застосунків різного типу: веб-застосунки, відеоігри, настільні програми, утиліти

для роботи з базами даних тощо. Велике розповсюдження Python отримав в області машинного навчання та штучного інтелекту. Код написаний на Python інтерпретується байт-код який переводиться віртуальною машиною в набір команд, які виконуються операційною системою.

Основні особливості мови програмування Python:

1. Простий та зрозумілий синтаксис: Python має чіткий і читабельний синтаксис, що робить код зрозумілим навіть для новачків. Це спрощує процес навчання і дозволяє швидко писати чистий код.

2. Динамічна типізація: у Python типи змінних визначаються автоматично під час виконання програми, що зменшує кількість коду, необхідного для оголошення змінних.

3. Автоматичне управління пам'яттю: Python має вбудований механізм збирання сміття, який автоматично керує виділенням і звільненням пам'яті.

4. Велика стандартна бібліотека: Python має велику кількість вбудованих модулів і пакетів, які забезпечують широкий спектр функціональності, від роботи з файловою системою до мережевих протоколів і веб-сервісів.

5. Підтримка різних парадигм програмування: Python підтримує об'єктно-орієнтоване, процедурне і функціональне програмування, що робить її гнучкою для різних типів завдань.

Java є платформонезалежною мовою, що дозволяє розробляти програми, які можна запускати на будь-якому пристрої, що підтримує віртуальну машину Java (JVM) [17]. Це забезпечує широку сумісність та мобільність програмного продукту. Крім того, Java має велику кількість бібліотек та фреймворків, що значно спрощує розробку та розширення функціоналу застосунку.

Основні особливості мови програмування Java:

1. Сувора типізація та перевірка типів: Java забезпечує статичну типізацію, що дозволяє виявляти багато помилок на етапі компіляції, що робить код більш безпечним і надійним.

2. Об'єктно-орієнтоване програмування: Java повністю підтримує об'єктно-орієнтоване програмування, що сприяє модульності, повторному використанню коду та легкому обслуговуванню великих проектів.

3. Мультипотоковість: вбудована підтримка багатонитковості дозволяє ефективно використовувати багатоядерні процесори для створення високопродуктивних багатозадачних додатків.

4. Платформонезалежність на рівні байт-коду: Java програми компілюються в байт-код, який виконується на будь-якій JVM, що робить можливим запуск Java програм на будь-якій платформі, де доступна JVM.

5. Динамічне завантаження класів: Java дозволяє динамічно завантажувати нові класи і методи під час виконання програми, що забезпечує гнучкість і можливість розширення функціоналу без перезапуску додатка.

6. Автоматичне управління пам'яттю: Java використовує збирач сміття для автоматичного управління пам'яттю, що звільняє розробників від необхідності вручну керувати виділенням і звільненням пам'яті.

7. Масштабованість: Java добре підходить для створення великих, масштабованих систем, таких як корпоративні додатки та веб-служби.

Важливою перевагою Java є її висока надійність та безпека [18]. Мова включає в себе вбудовані механізми безпеки, що дозволяють попереджати багато типових помилок програмістів, таких як витіки пам'яті або виклики невизначеного стану. Крім того, Java має вбудовану систему керування пам'яттю, яка автоматично видаляє непотрібні об'єкти, що робить програми на Java більш стабільними та ефективними у використанні ресурсів.

Аргумент на користь Java — це велика спільнота розробників та багато ресурсів для самоосвіти. Java є однією з найпопулярніших мов програмування у світі, що означає, що знайдення допомоги або рішення проблем не становить проблеми. Велика кількість книг, курсів, форумів та інших джерел інформації робить вивчення та розробку програм на Java досить доступними для будь-якого розробника.

Ці переваги є ключовими для розробки програмних модулів, тому обрання мови програмування Java є обґрунтованим кроком, що забезпечує надійність, безпеку та ефективність розробки, а також широкі можливості для подальшого розвитку та супроводу програмного забезпечення.

Розглянемо можливості використання платформ для розробки інтерфейсу в середовищі Java.

Swing — це набір бібліотек Java, який дозволяє розробникам створювати різноманітні користувацькі інтерфейси для десктопних застосунків. Він є частиною Java Foundation Classes (JFC) і надає засоби для створення графічних інтерфейсів з використанням різноманітних компонентів, таких як кнопки, поля для введення, списки, таблиці, вкладки і багато інших. Однією з ключових особливостей Swing є його можливість кастомізації. Розробники можуть легко налаштовувати вигляд і поведінку компонентів, використовуючи механізми рендерингу і моделювання подій. Це дозволяє створювати інтерфейси, які відповідають конкретним дизайнерським концепціям або вимогам брендингу. Крім того, Swing підтримує подійно-орієнтовану модель програмування, що робить його хорошим інструментом для обробки користувацьких взаємодій. Розробники можуть відслідковувати події, такі як натискання кнопок, введення тексту або переміщення миші, і реагувати на них відповідним чином. Також, Swing надає різноманітні менеджери розташування, які допомагають автоматизувати розміщення компонентів на вікні, що робить процес розробки інтерфейсу більш простим і ефективним.

JavaFX — це платформа для розробки користувацьких інтерфейсів в середовищі Java. Вона надає широкий набір інструментів і можливостей для створення багатофункціональних та стильних графічних інтерфейсів. Однією з ключових особливостей JavaFX є його модульність та вбудована підтримка структури MVC (Model-View-Controller), що спрощує організацію коду та розділення логіки застосунку від його візуального представлення. JavaFX має різноманітні вбудовані елементи управління (controls), такі як кнопки, тексти, таблиці, списки, діаграми, відеоплеєри та інші, що дозволяють швидко

побудувати потрібний інтерфейс. Ще однією важливою особливістю JavaFX є CSS-подібна система стилів, яка дозволяє розробникам легко змінювати вигляд інтерфейсу, використовуючи декларативний підхід до оформлення. Також варто відзначити високу продуктивність JavaFX та його можливості інтеграції з іншими технологіями Java, що робить його чудовим вибором для розробки як малих, так і великих застосунків з графічним інтерфейсом.

Можна виділити наступні переваги JavaFX над Swing:

1. Модернізований вигляд: JavaFX надає більш сучасний та естетичний вигляд для створення інтерфейсу користувача порівняно з більш застарілим виглядом Swing.

2. CSS стилізація: JavaFX дозволяє використовувати CSS для стилізації віджетів, що робить процес розробки інтерфейсу більш гнучким та простим у використанні.

3. Модульність: JavaFX підтримує модульну архітектуру, що полегшує організацію та управління компонентами інтерфейсу.

4. Графічні можливості: JavaFX має більш потужні графічні можливості, такі як вбудована підтримка тривимірної графіки, що дозволяє створювати більш складні та привабливі інтерфейси.

5. Підтримка мультимедія: JavaFX має вбудовану підтримку мультимедіа, що робить його ідеальним вибором для розробки застосунків, які вимагають обробки відео, аудіо тощо.

6. Підтримка FXML: JavaFX надає можливість використовувати FXML для опису інтерфейсу користувача, що дозволяє розділити логіку застосунку від його візуальної частини і спрощує розробку та обслуговування.

Отже, вибір JavaFX для реалізації програмного застосунку для інтелектуального керування рухом транспорту є обґрунтованим рішенням.

3.2 Розробка інтерфейсу програмного застосунку

Успішність будь-якого програмного продукту часто залежить від його інтерфейсу — того, як користувач взаємодіє з програмою. Тут розглянуто процес

розробки інтерфейсу програмного застосунку. Було вирішено розробити зручний та адаптивний графічний інтерфейс програмного застосунку.

Кольорова гама важливий аспект розробки інтерфейсу програмного застосунку, а основний колір — сірий — може відображати його стиль та атмосферу. Використання сірого кольору надає інтерфейсу відмінний вигляд, забезпечуючи при цьому відчуття солідності та професіоналізму. Сірий колір часто використовується для фонів, кнопок, рамок та текстових елементів, що додає зручності та легкості в сприйнятті інтерфейсу користувачем. Крім того, сірий колір може створювати контраст з яскравішими кольорами, що допомагає підкреслити важливі елементи інтерфейсу та поліпшує його зовнішній вигляд.

При відкритті застосунку користувач бачить на екрані вікно авторизації, де може ввести свої облікові дані для доступу до функціоналу програми (рис. 3.1).

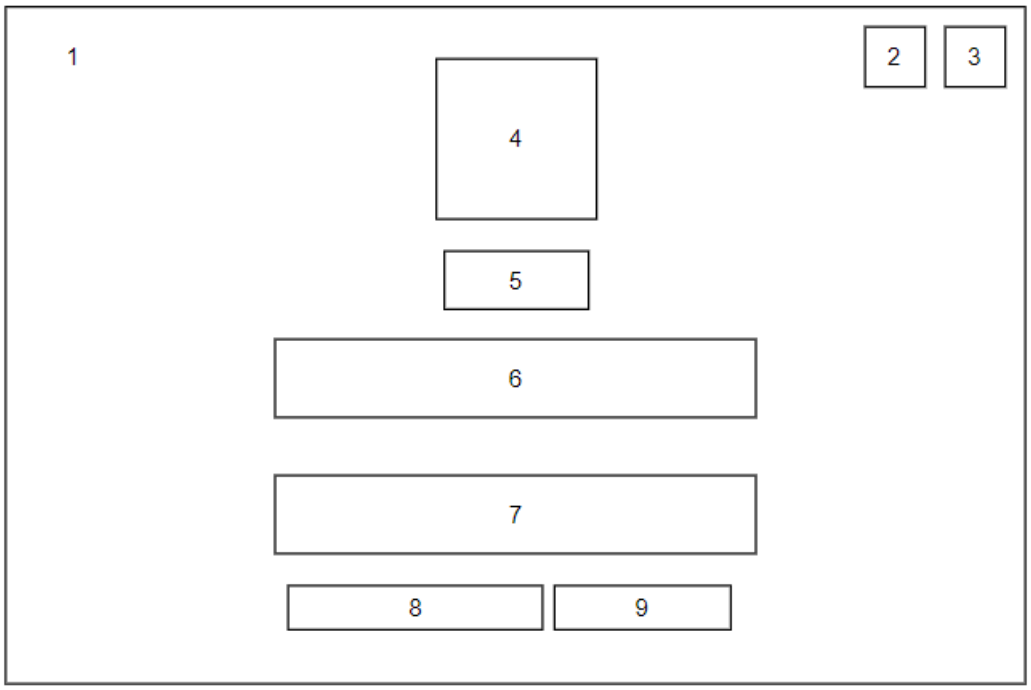


Рисунок 3.1 — Схема вікна авторизації

На схемі зображено:

1. Вікно.
2. Кнопка «Згорнути»

3. Кнопка «Закрити».
4. Логотип.
5. Напис «Sign in»
6. Поле для вводу логіну.
7. Поле для вводу паролю.
8. Напис «Don’t have an account?».
9. Перейти до реєстрації.

На рисунку 3.2 зображено вікно авторизації.

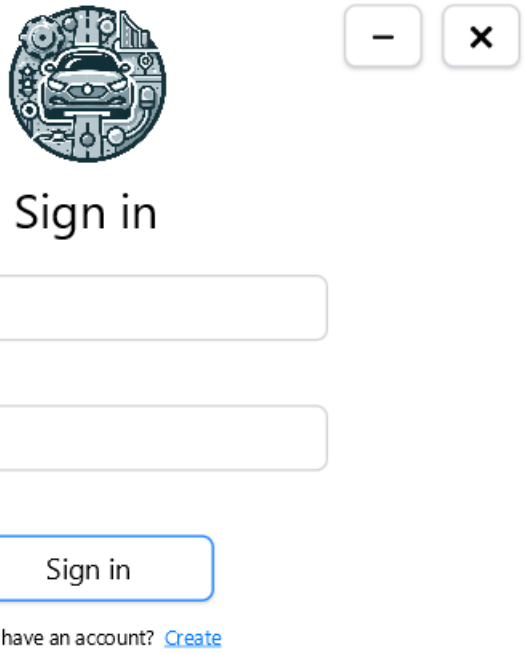


Рисунок 3.2 — Вікно авторизації

Якщо користувач не має аккаунту, він може його зареєструвати, перейшовши за посиланням «Create», після чого він побачить вікно реєстрації.

На рисунку 3.3 зображено схему вікна реєстрації.

На схемі зображено:

1. Вікно.
2. Кнопка «Згорнути»
3. Кнопка «Закрити».

4. Логотип.
5. Напис «Sign up»
6. Поле для вводу логіну.
7. Поле для вводу email.
8. Поле для вводу паролю.
9. Поле для підтвердження паролю
10. Напис «Already have an account?».
11. Перейти до авторизації.

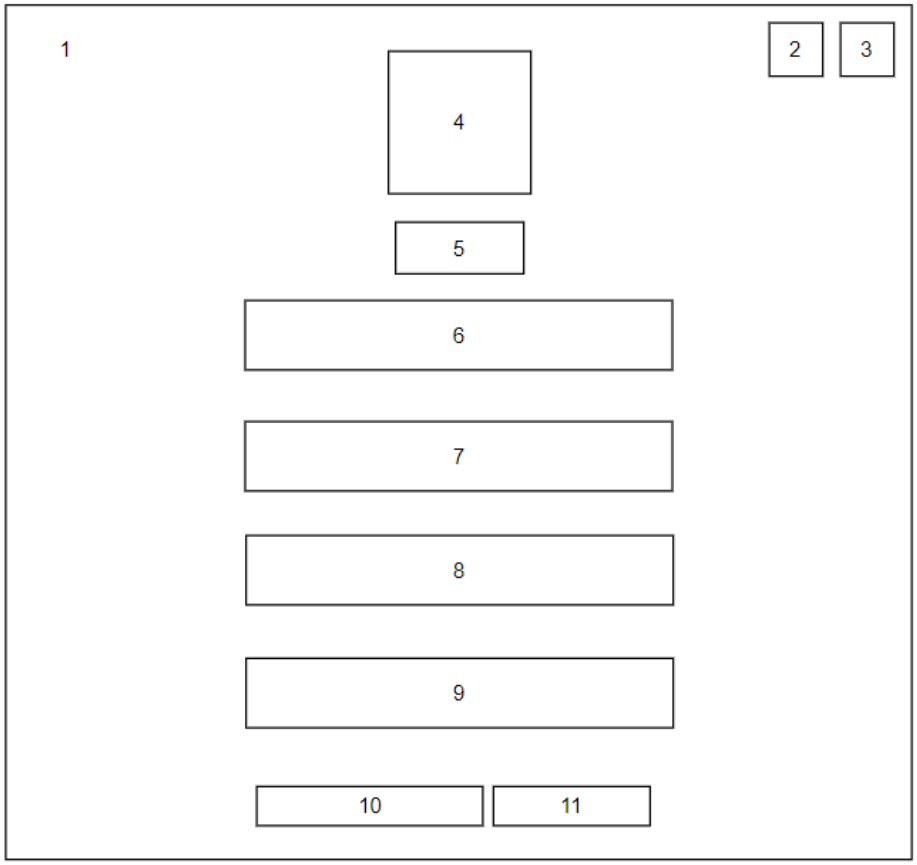


Рисунок 3.3 — Схема вікна реєстрації

На рисунку 3.4 зображено вікно реєстрації.

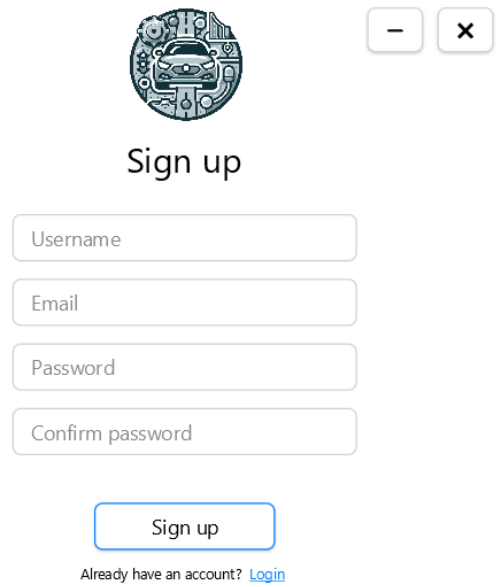


Рисунок 3.4 — Вікно реєстрації

Користувач при бажанні може повернутись назад до вікна авторизації, натиснувши на посилання «Login». Після реєстрації аккаунту користувача автоматично перекине на вікно авторизації, де він зможе успішно авторизуватись.

Після успішної авторизації користувач бачить головне вікно, з яким він і буде в основному взаємодіяти.

На рисунку 3.5 зображено схему головного вікна.

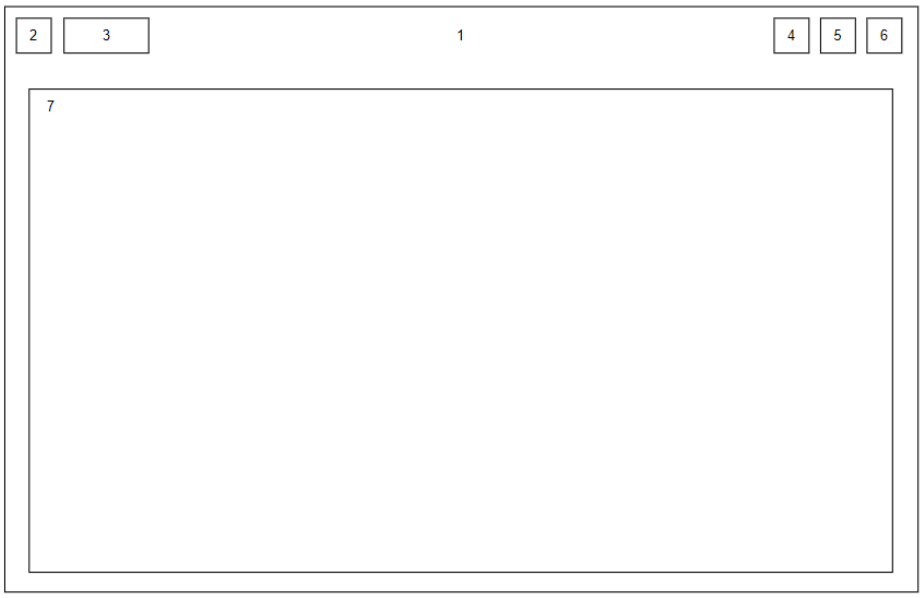


Рисунок 3.5 — Схема головного вікна

На схемі зображено:

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 основною частиною, що відображає рух транспорту на дорозі.

На рисунку 3.6 зображено головне вікно.

Рисунок 3.6 — Головне вікно

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

На рисунку 3.7 зображено схему інформаційного вікна.

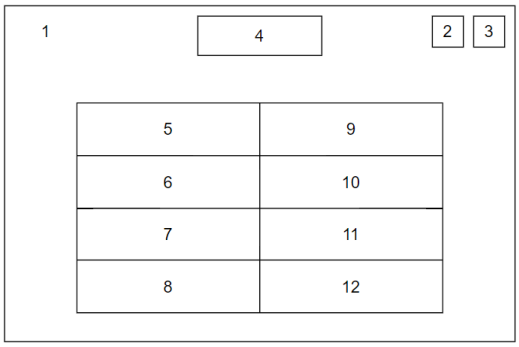


Рисунок 3.7 — Схема інформаційного вікна про автомобіль

На схемі зображено:

1. Вікно.
2. Кнопка «Згорнути»
3. Кнопка «Закрити».
4. Напис «Car Info».
5. Напис «Number».
6. Напис «Model».
7. Напис «Speed coefficient».
8. Напис «Offences».
9. Номер автомобіля.
10. Марка автомобіля.
11. Коефіцієнт швидкості.
12. Кількість порушень.

На рисунку 3.8 зображено інформаційне вікно.

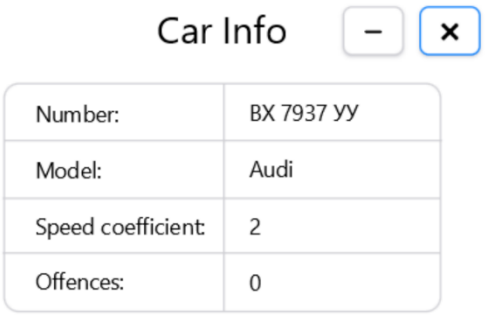


Рисунок 3.8 — Інформаційне вікно про автомобіль

Якщо кількість порушень не 0, то при натисненні на кількість порушень з’явиться вікно зі списком цих правопорушень, що були зареєстровані в базі даних.

На рисунку 3.9 зображено вікно зі списком правопорушень, що знайдені за цим автомобілем.

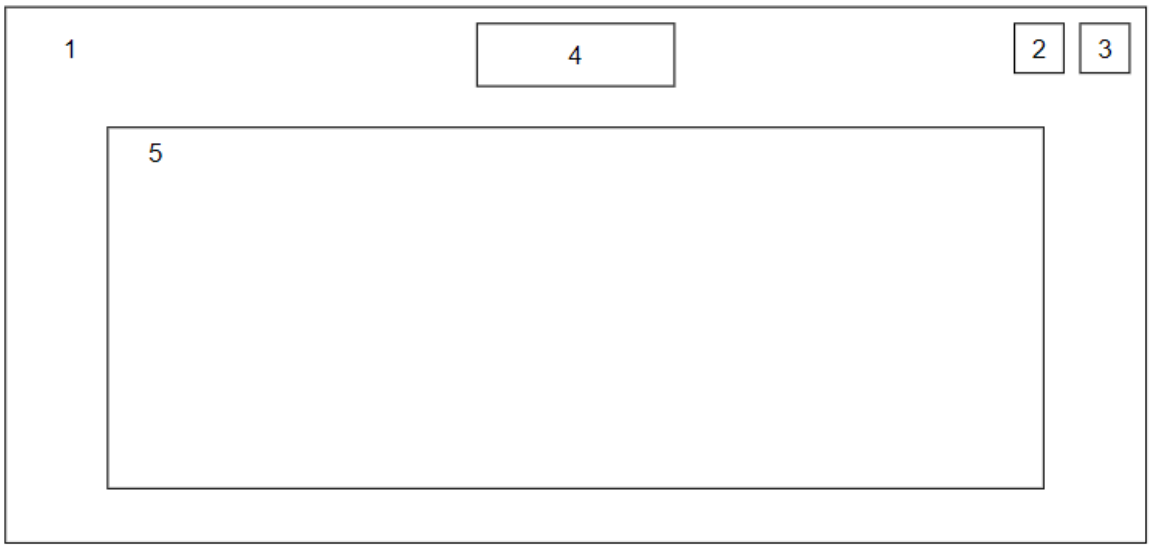


Рисунок 3.9 — Схема вікна зі списком правопорушень

На схемі зображено:

1. Вікно.
2. Кнопка «Згорнути»
3. Кнопка «Закрити».
4. Напис «Offences».
5. Таблиця.

Таблиця містить наступні заголовки:

1. Type — тип правопорушення.
2. Car Number — номер автомобіля.
3. Car Model — марка автомобіля.
4. Time — дата та час здійснення правопорушення.

На рисунку 3.10 зображено це вікно.

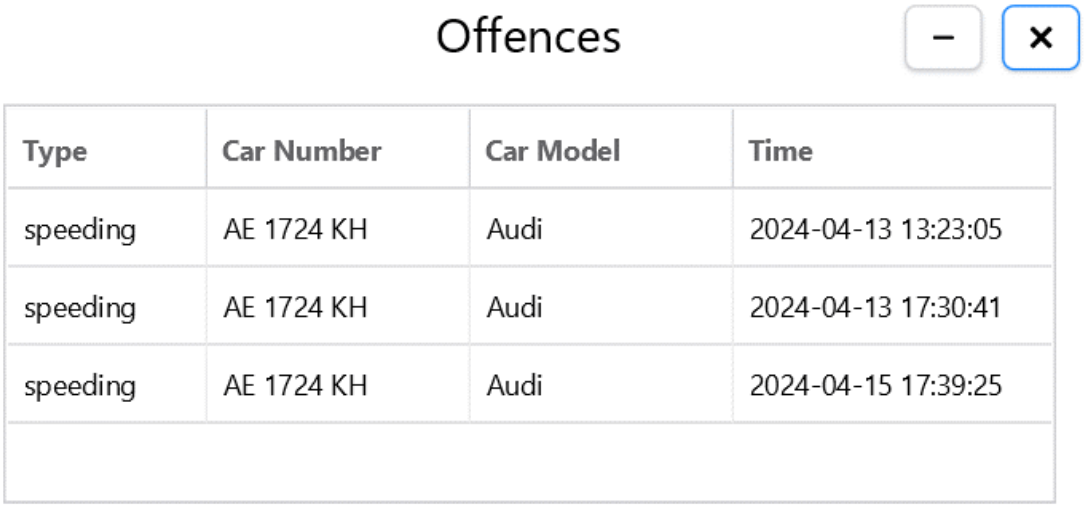


Рисунок 3.10 — Вікно зі списком правопорушень

Якщо натиснути кнопку «Переглянути список правопорушень» на головному вікні, з’явиться подібне вікно зі списком правопорушень, як попереднє, але воно буде містити всі наявні зареєстровані правопорушення в базі даних (рис. 3.11).

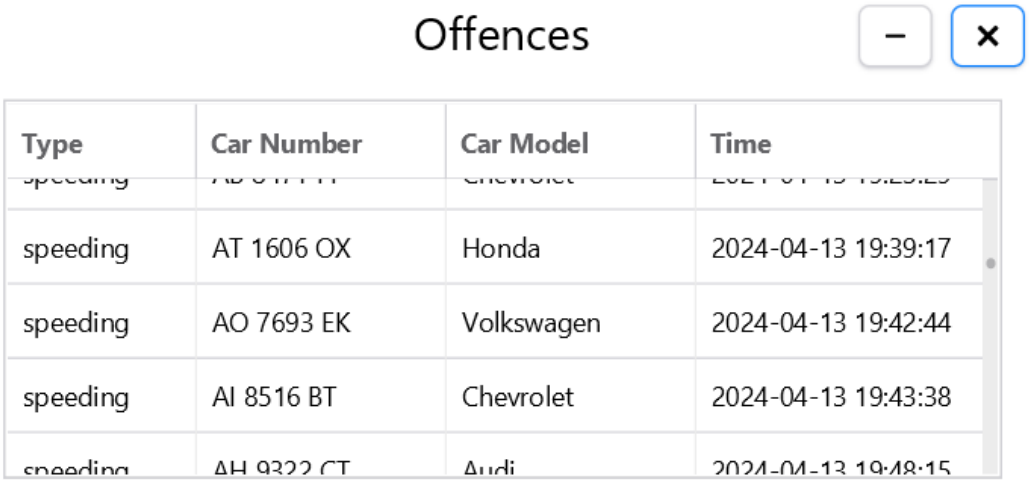


Рисунок 3.11 — Вікно зі списком всіх правопорушень

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

3.3 Розробка та підключення бази даних

Для того, щоб користувачі могли виконувати авторизацію та реєстрацію, потрібно мати місце де зберігати ці дані. Тому було вирішено використовувати базу даних.

Для її розробки потрібно охоплити всі аспекти розробки бази даних, включаючи вибір системи управління базами даних (СУБД), створення таблиць, а також підключення бази даних до програмного застосунку.

MySQL Workbench — це інтегроване середовище розробки баз даних, що надає зручні інструменти для проектування, моделювання, роботи з базами даних MySQL та керування ними [19]. Основні переваги використання MySQL Workbench включають:

1. Графічний інтерфейс користувача: MySQL Workbench має інтуїтивно зрозумілий і легкий у використанні графічний інтерфейс, який дозволяє розробникам швидко та зручно виконувати різні завдання, такі як створення, редагування та видалення об'єктів баз даних.

2. Моделювання баз даних: MySQL Workbench дозволяє розробникам створювати моделі баз даних з використанням графічного інтерфейсу, що дозволяє визначати структуру даних, відносини між таблицями та інші аспекти дизайну бази даних.

3. Синхронізація з базою даних: Завдяки функціоналу синхронізації, MySQL Workbench дозволяє легко і швидко синхронізувати структуру бази даних та її дані між різними середовищами розробки.

4. Візуальне відображення запитів: Інтегрований SQL редактор надає можливість створювати SQL-запити та візуалізувати їх результати безпосередньо у редакторі.

5. Автоматизація задач: MySQL Workbench підтримує автоматизацію рутинних задач, таких як резервне копіювання баз даних, виконання пакетних завдань та інші.

6. Підтримка версій баз даних: MySQL Workbench підтримує роботу з різними версіями сервера MySQL, а також з іншими системами управління базами даних, такими як MariaDB.

На основі цих переваг було вирішено використовувати MySQL Workbench.

База даних використовуватиметься для:

- Авторизації користувача.
- Реєстрації користувача.
- Логування правопорушень на дорозі.

Для збереження описаних даних було створено дві сутності, а саме:

- users (<user_id>, username, email, password, status);
- offences (<offence_id>, type, car_number, car_model, time).

Таблиця users містить ключ <user_id> та інформацію про користувача. Поле status визначає права доступу до системи.

Таблиця offences містить ключ <offence_id> та інформацію про залоговані правопорушення на дорозі.

Для підключення бази даних до програмного застосунку спершу потрібно завантажити бібліотеку mysql-connector-j.jar та підключити її до проекту.

Далі потрібно виконати підключення до бази даних

На рисунку 3.12 зображено діаграму послідовностей підключення до бази даних.

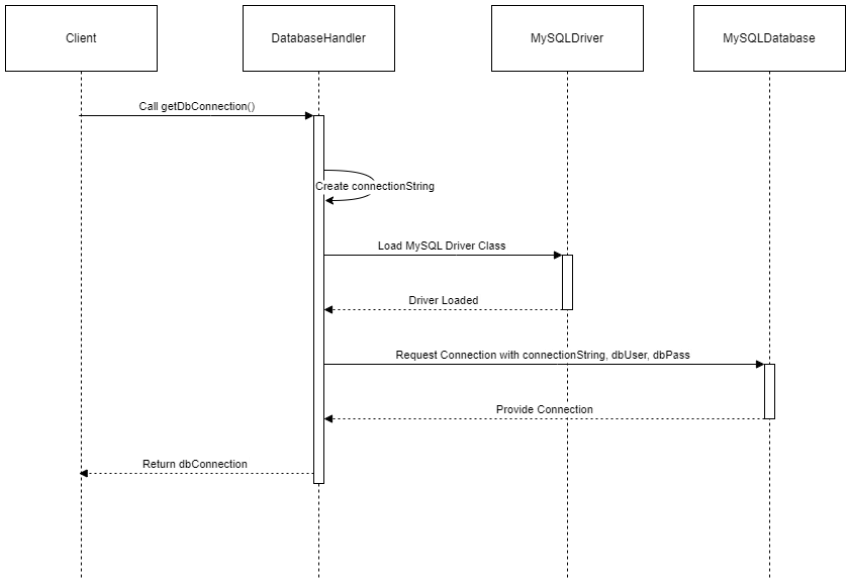


Рисунок 3.12 — Підключення до бази даних

Після цього можна взаємодіяти з базою даних. Наприклад запит до бази даних для реєстрації користувача

На рисунку 3.13 зображено діаграму послідовності реєстрації користувача.

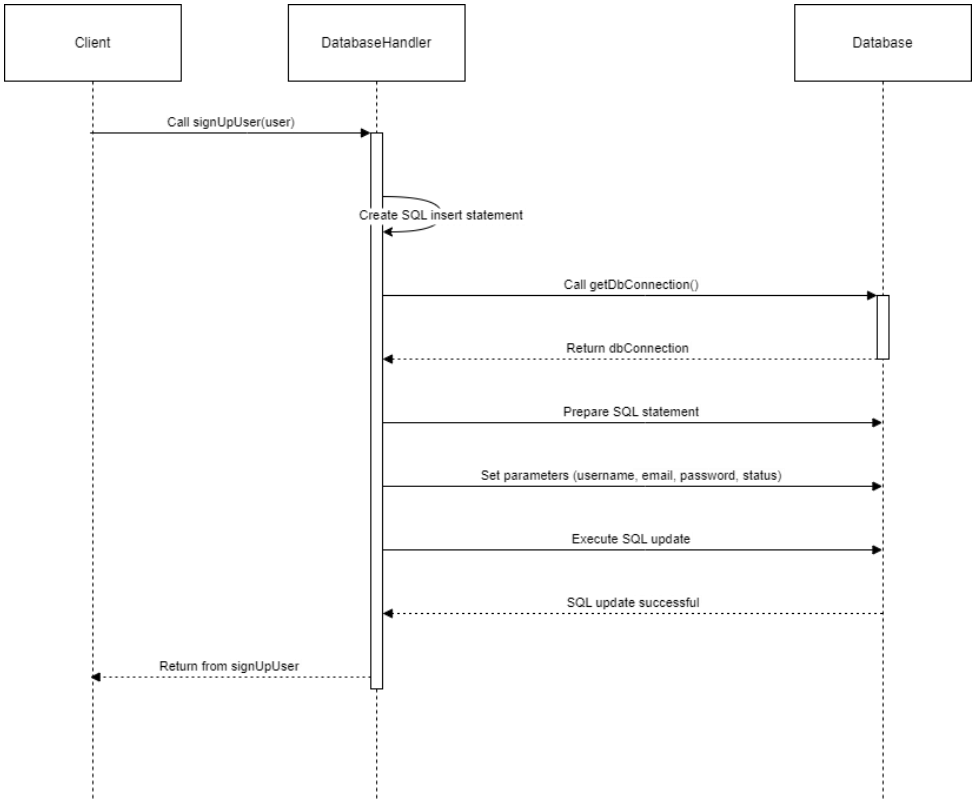


Рисунок 3.13 — Реєстрація користувача

Запит до бази даних на перевірку існування username дозволить забезпечити унікальність ідентифікаторів користувачів у системі. Це важливо для запобігання конфліктів та недопущення створення дублікатів облікових записів. Шляхом перевірки існування username в базі даних перед створенням нового облікового запису можна уникнути ситуацій, коли користувачі обирають однакові імена. Такий підхід сприяє покращенню безпеки системи та забезпеченню коректної роботи функцій автентифікації та ідентифікації користувачів.

На рисунку 3.14 зображено діаграму послідовності перевірки існування username.

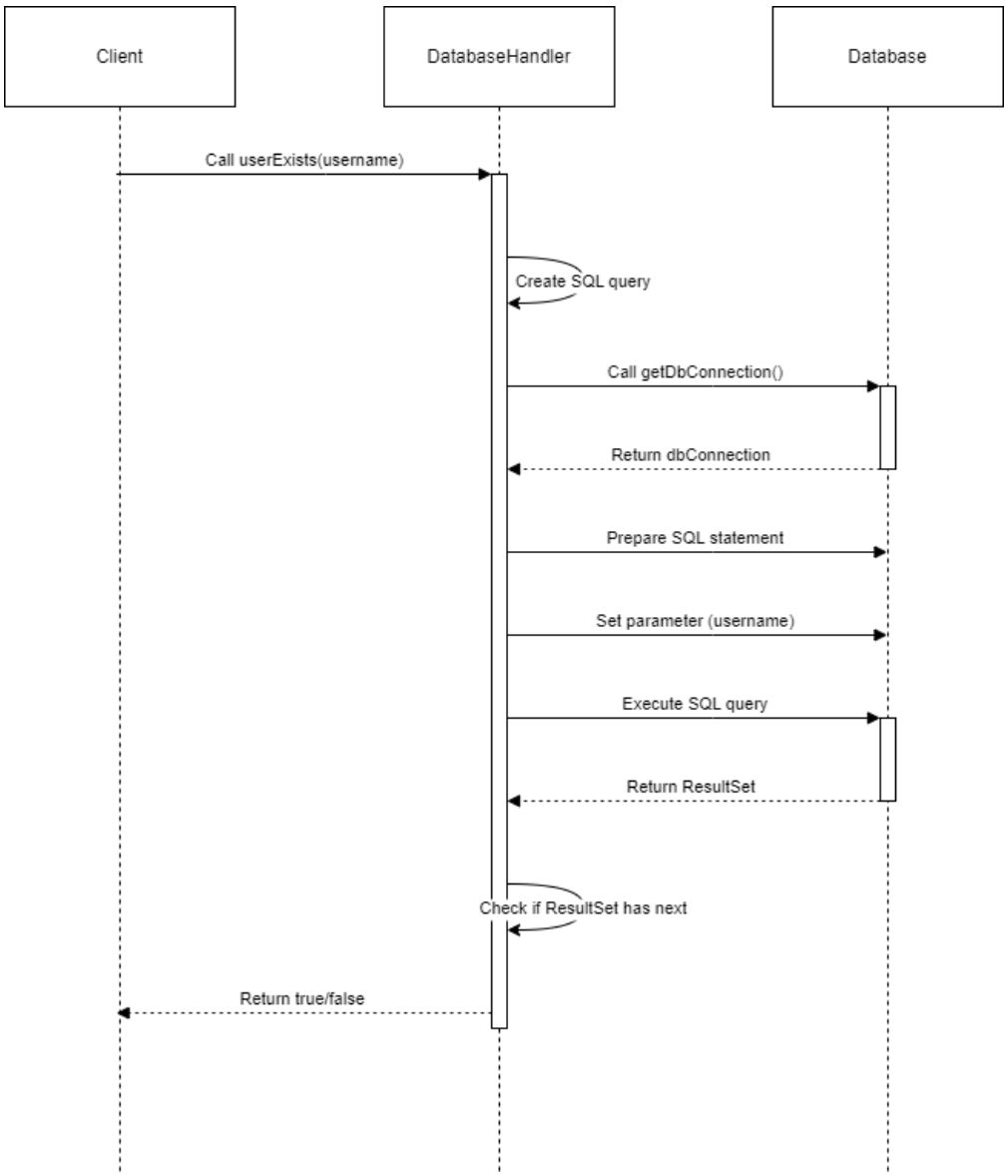


Рисунок 3.14 — Перевірка існування username

Запит до бази даних на перевірку існування email є важливим етапом в процесі реєстрації користувача або оновлення його даних. Це дозволяє перевірити, чи існує вже обліковий запис з вказаною електронною адресою в системі. Перевірка існування email перед реєстрацією допомагає уникнути створення дублікатів облікових записів та забезпечує унікальність ідентифікаторів користувачів. Цей запит також може використовуватися для перевірки актуальності та коректності введених даних, забезпечуючи високий рівень якості та надійності інформації у базі даних.

На рисунку 3.15 зображено діаграму послідовності перевірки існування email.

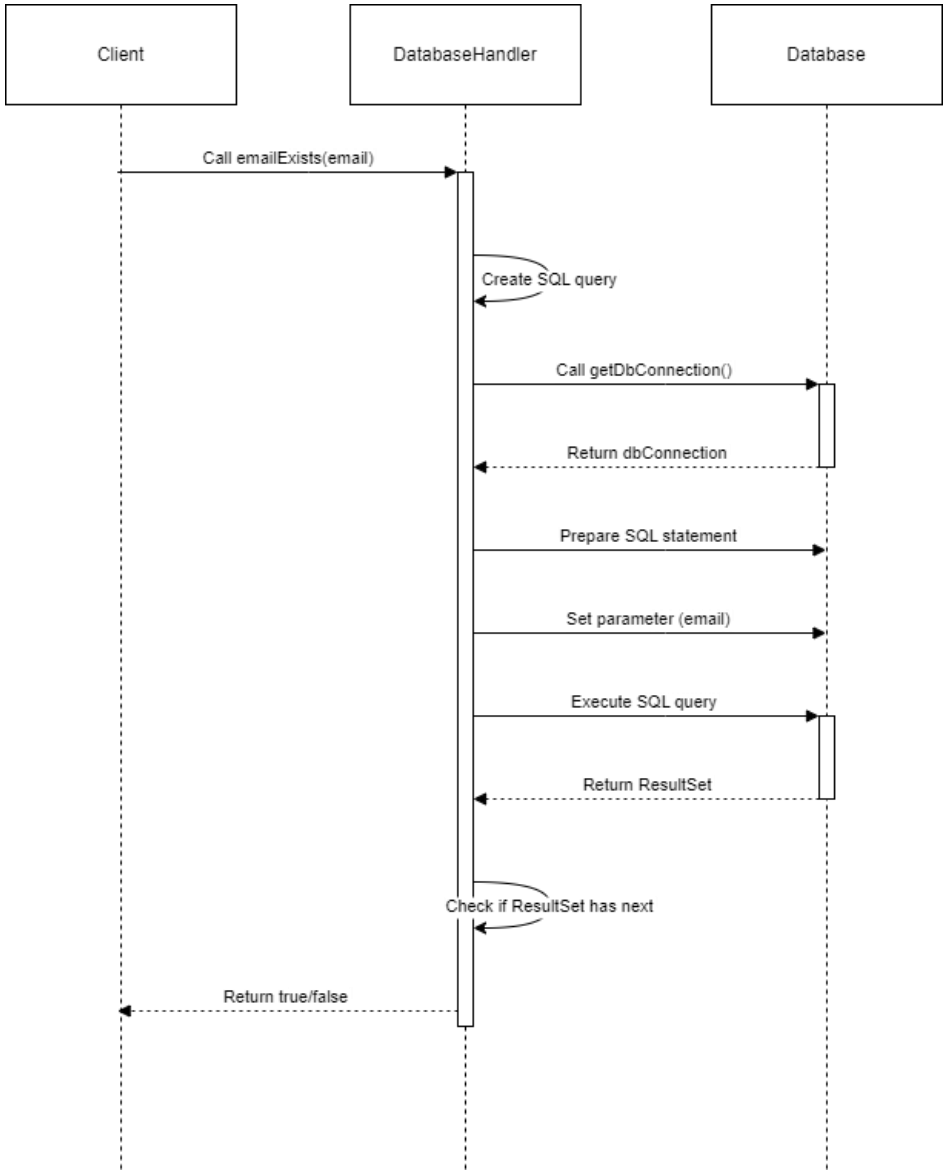


Рисунок 3.15 — Перевірка існування email

При спробі авторизації виконується запит до бази даних на перевірку паролю.

На рисунку 3.16 зображено діаграму послідовності перевірки паролю.

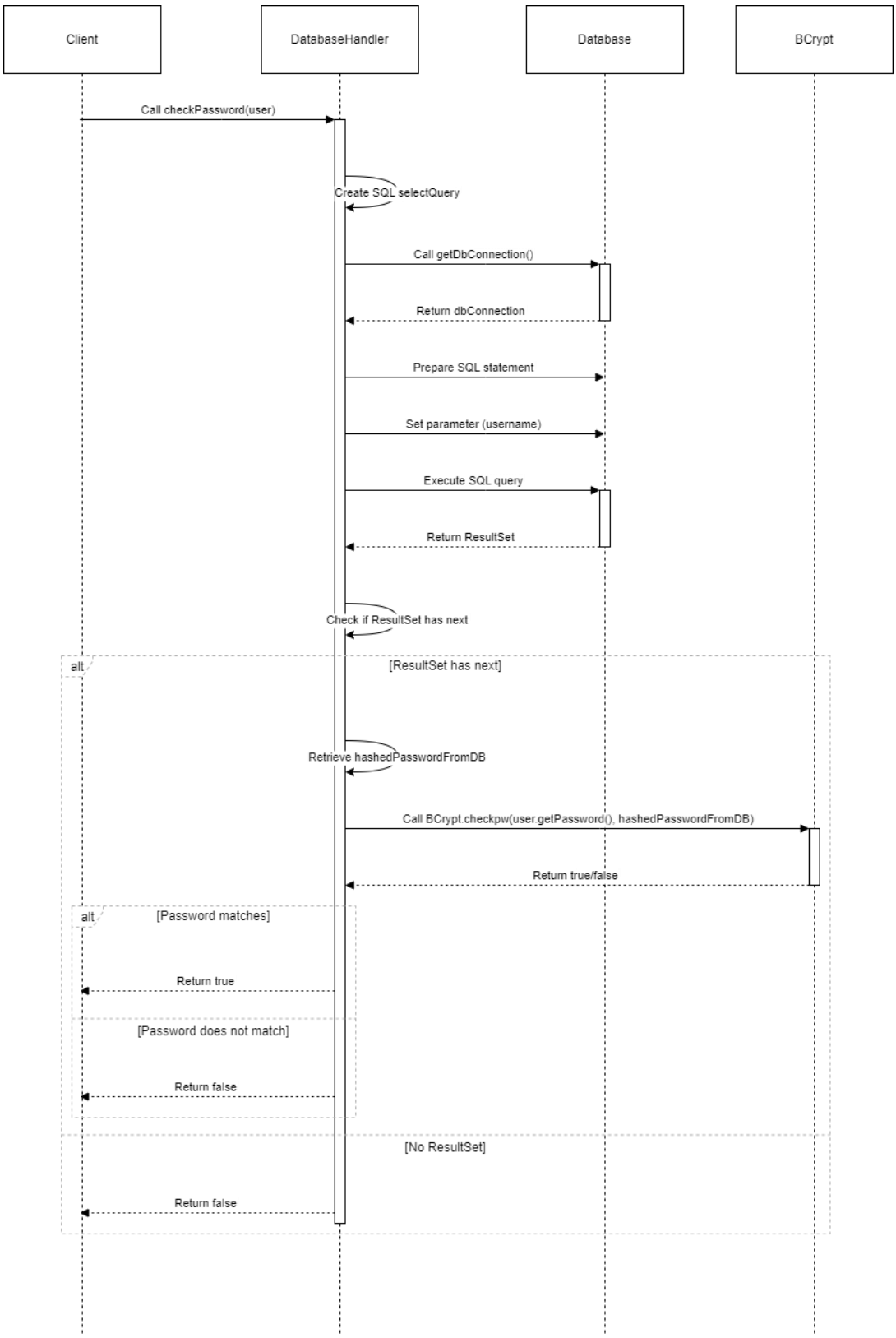


Рисунок 3.16 — Перевірка паролю

Отже, було розглянуто аргументи щодо вибору системи управління базами даних (СУБД), створено таблиці, виконано підключення бази даних до програмного застосунку, описано основні запити до бази даних.

3.4 Розробка модулів системи для адаптивного керування рухом транспорту

Одним із важливих модулів системи є адаптивне керування світлофорами на дорозі. Звичайний стан світлофору: змінювати сигнал на протилежний кожні 15 секунд. На рисунку 3.17 зображено діаграму послідовності, що демонструє зміну сигналу світлофора та оновлення таймеру кожної секунди.

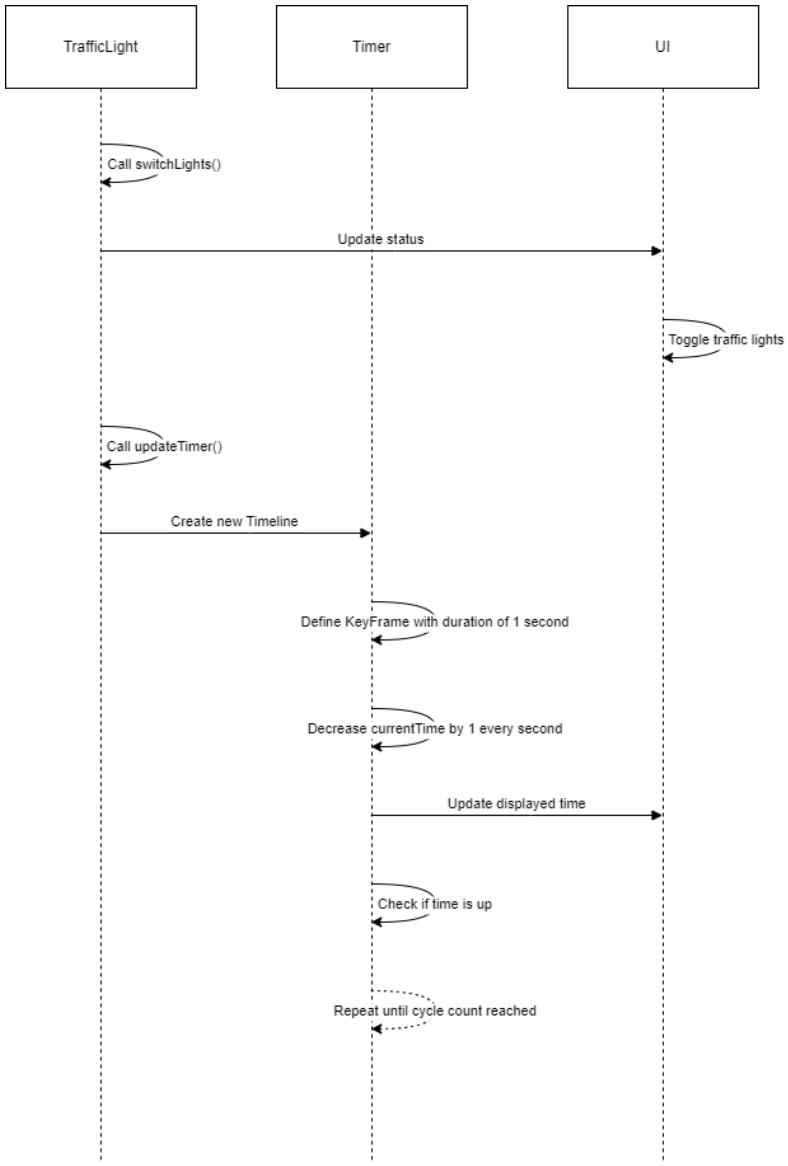


Рисунок 3.17 — Зміна сигналу світлофора

Крім цього, датчик на світлофорі підраховує, скільки автомобілів під’їхало та стоїть на червоному сигналі. Якщо автомобілів більше ніж 4, сигнал всіх

світлофорів на перехресті буде змінено для зменшення кількості автомобілів, що стоять, та запобіганню затору (рис 3.18).

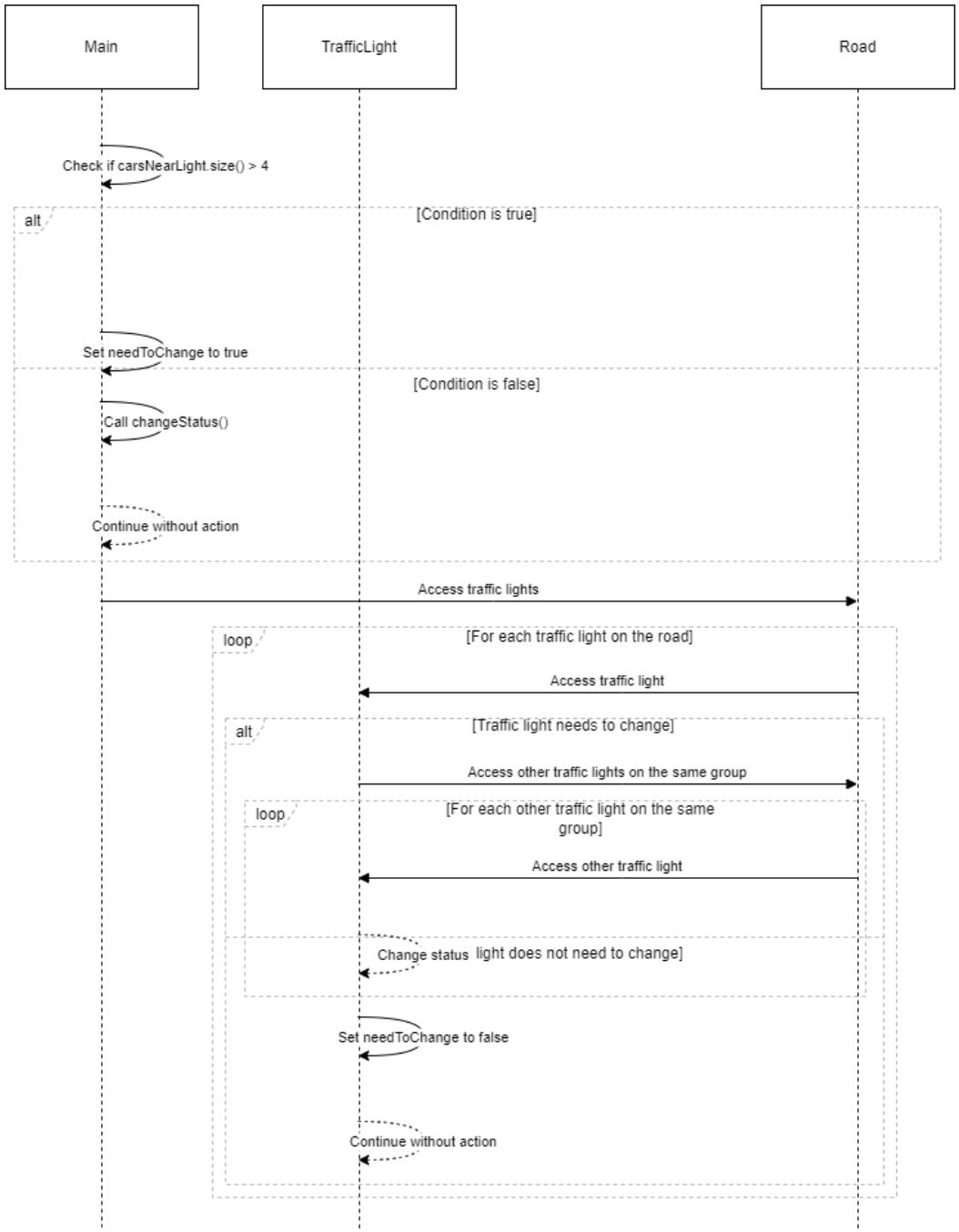


Рисунок 3.18 — Адаптивне регулювання світлофора

Наступним важливим модулем є моніторинг правопорушників. При русі автомобіля є невелика ймовірність, що швидкість буде перевищена.

Для відслідковування цього на дорозі присутні спеціальні датчики, які фіксують перевищення швидкості. Якщо датчик зафіксував перевищення

швидкості, система виконує збереження правопорушення в базу даних, таким чином фіксуючи правопорушення, що в подальшому може бути використане для виписування штрафу (рис. 3.19).

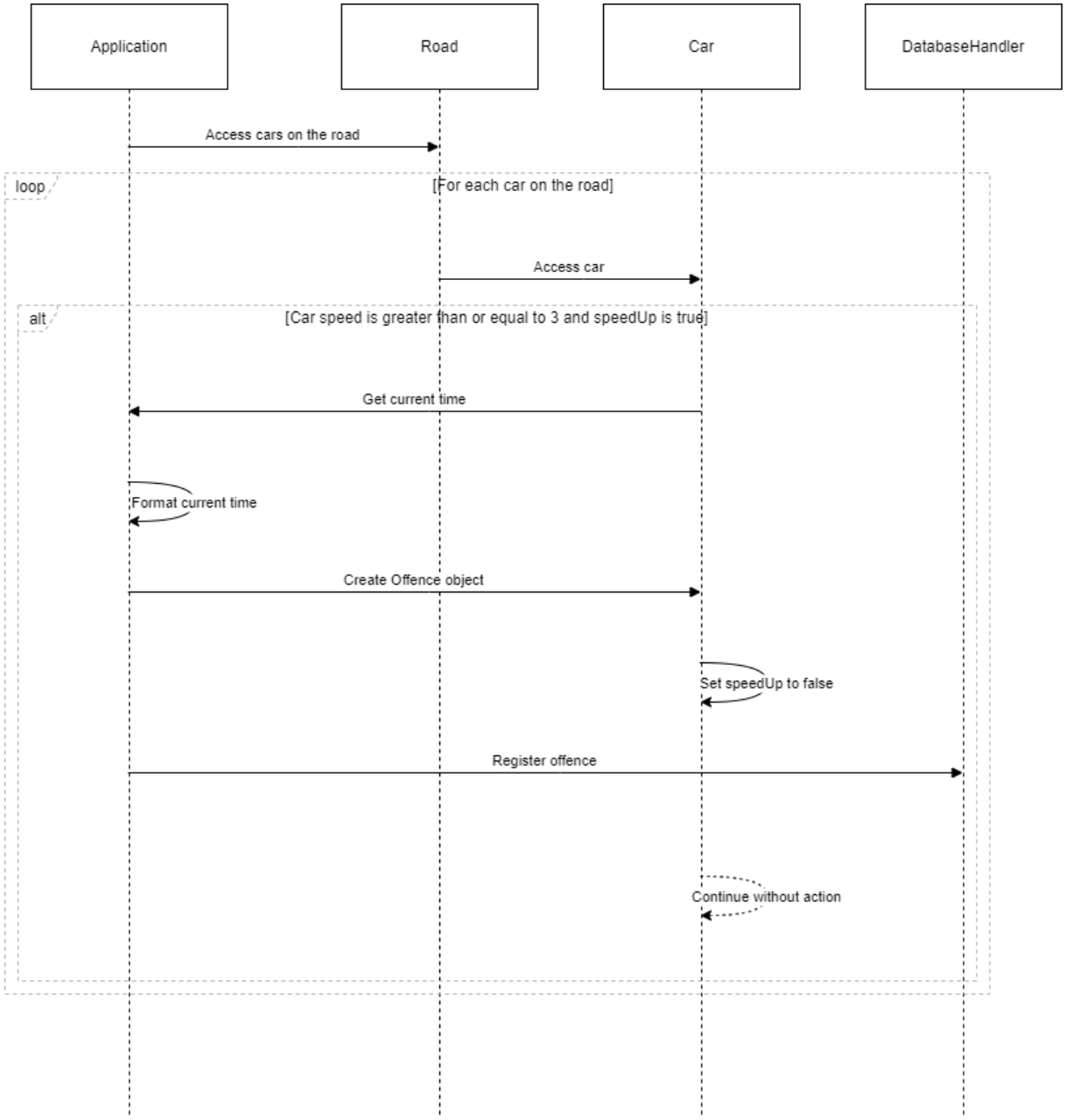


Рисунок 3.19 — Фіксація правопорушення

Запис в базу даних відправляється в наступному форматі:

- Тип правопорушення (в даному випадку перевищення швидкості).
- Номер автомобіля.
- Марка автомобіля.

- Дата та час скоєння правопорушення.

Отже, було розроблено наступні важливі модулі: адаптивне керування світлофора, моніторинг правопорушників.

3.5 Висновки

У третьому розділі було проаналізовано мови програмування та інструменти для розробки. Було обґрунтовано вибір мови програмування Java для розробки програмного застосунку. Було обрано JavaFX для реалізації інтерфейсу програмного застосунку. Було розроблено інтерфейс програмного застосунку для інтелектуального керування рухом транспорту. Було розглянуто аргументи щодо вибору системи управління базами даних (СУБД), створено таблиці, виконано підключення бази даних до програмного застосунку. Було розроблено модуль адаптивного керування світлофора та модуль моніторингу правопорушників.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення є важливим етапом в розробці будь-якого програмного продукту. Воно проводиться з метою забезпечення високої якості продукту перед його випуском на ринок або в експлуатацію. Це допомагає виявити і виправити помилки та недоліки в програмному коді, що сприяє підвищенню надійності, ефективності та безпеки програми. Крім того, тестування допомагає забезпечити відповідність програмного продукту вимогам користувачів та бізнес-потребам, що є ключовим аспектом у забезпеченні задоволення клієнтів і конкурентоспроможності на ринку. Також тестування є важливою складовою процесу розробки програмного забезпечення, що допомагає виявити проблеми на ранніх етапах і знизити витрати на їх виправлення в подальшому.

Існує безліч методів тестування програмного забезпечення, кожен з яких має свої переваги та застосування, залежно від вимог проекту, типу програми і доступних ресурсів. До них можна віднести:

1. Тестування «чорної скриньки» — це оцінка функціональності програми без знання внутрішньої реалізації [20]. Дозволяє перевірити програму на відповідність вимогам і очікуванням користувача.

2. Тестування «білої скриньки» — це перевірка внутрішньої структури та логіки програми. Таке тестування може виявити помилки, які не видно при зовнішньому тестуванні.

3. Модульне тестування полягає у перевірці окремих модулів програми на коректність їх роботи. Кожен модуль, який складається з функцій або класів, тестується окремо. Модульне тестування дозволяє виявити помилки на ранніх етапах розробки та покращити якість коду.

4. Функціональне тестування перевіряє відповідність функціональності програмного продукту вимогам та очікуванням користувача.

Тести виконуються для кожної функції або операції програми, щоб переконатися, що вони працюють правильно.

5. Інтеграційне тестування перевіряє взаємодію між різними модулями або компонентами програми. Його мета - виявити помилки, які можуть виникнути при інтеграції окремих частин програми в єдину систему.

6. Системне тестування полягає в перевірці програми як цілісної системи з урахуванням всіх її функцій і можливостей. Під час системного тестування перевіряється відповідність програми вимогам, що визначаються в специфікації системи.

7. Автоматизоване тестування використовує спеціальні програмні засоби для автоматизації процесу тестування. Він дозволяє швидко та ефективно виконувати тести, знижуючи час і зусилля, необхідні для їх проведення вручну. Автоматизоване тестування особливо корисне для повторюваних тестів, регресійного тестування та навантажувального тестування.

Метод «чорної скриньки» має ряд переваг:

1. Незалежність від внутрішньої реалізації. Тестувальник не повинен знати деталей реалізації програми. Це означає, що тести можуть бути проведені навіть без доступу до вихідних кодів, що є важливим у випадках, коли код знаходиться під захистом або коли розробник не розголошує вихідні файли.

2. Сконцентрованість на взаємодії з користувачем. Метод «чорної скриньки» дозволяє зосередитися на взаємодії програми з користувачем, його інтерфейсом та поведінкою, що є критично важливим для визначення якості програми з точки зору кінцевого користувача.

3. Стимулює тестувальника мислити ширше. Тестування «чорної скриньки» стимулює тестувальника думати ширше і спробувати всі можливі варіанти введення та сценарії взаємодії з програмою, що може виявити більше помилок та аномалій.

4. Виявлення помилок, які не пов'язані з реалізацією. Тестування «чорної скриньки» дозволяє виявляти помилки та недоліки, які пов'язані з

неочікуваними взаємодіями, алгоритмічними помилками або неправильною роботою програми, які можуть бути пропущені при інших методах тестування.

5. Більш реалістичне тестування. Оскільки тестування виконується на рівні користувача, цей метод дозволяє більш реалістично оцінити якість програми та її придатність для використання в реальних умовах.

Отже, зважаючи на ці переваги, було обрано метод тестування «чорної скриньки».

4.2 Тестування системи

Першим кроком слід перевірити авторизацію. При спробі авторизації з порожніми полями буде виведене повідомлення «Empty fields!» (рис. 4.1).

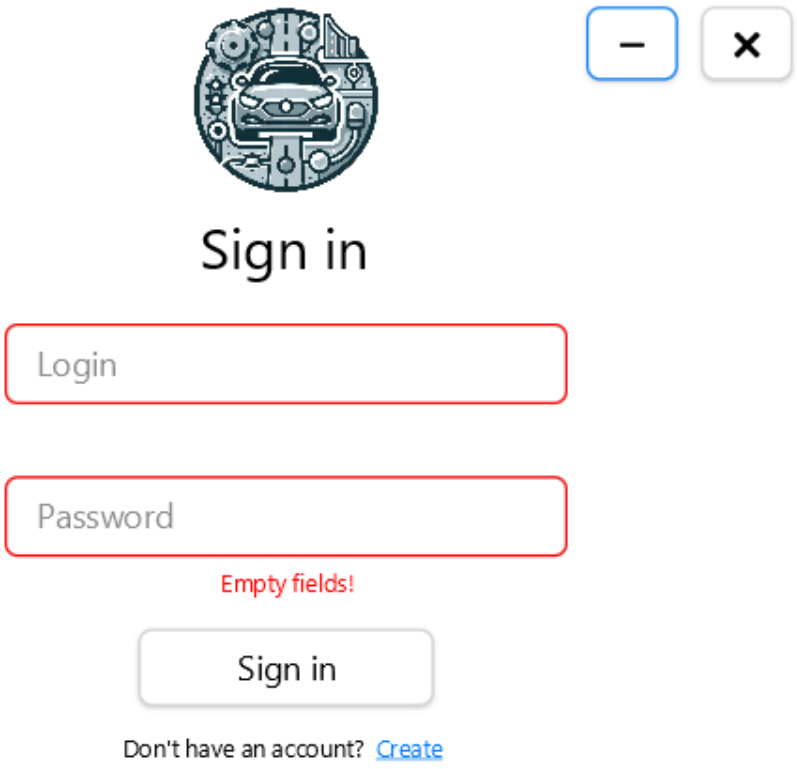


Рисунок 4.1 — Порожні поля

При введенні неіснуючого Login буде виведене повідомлення «Login is wrong!» (рис 4.2).

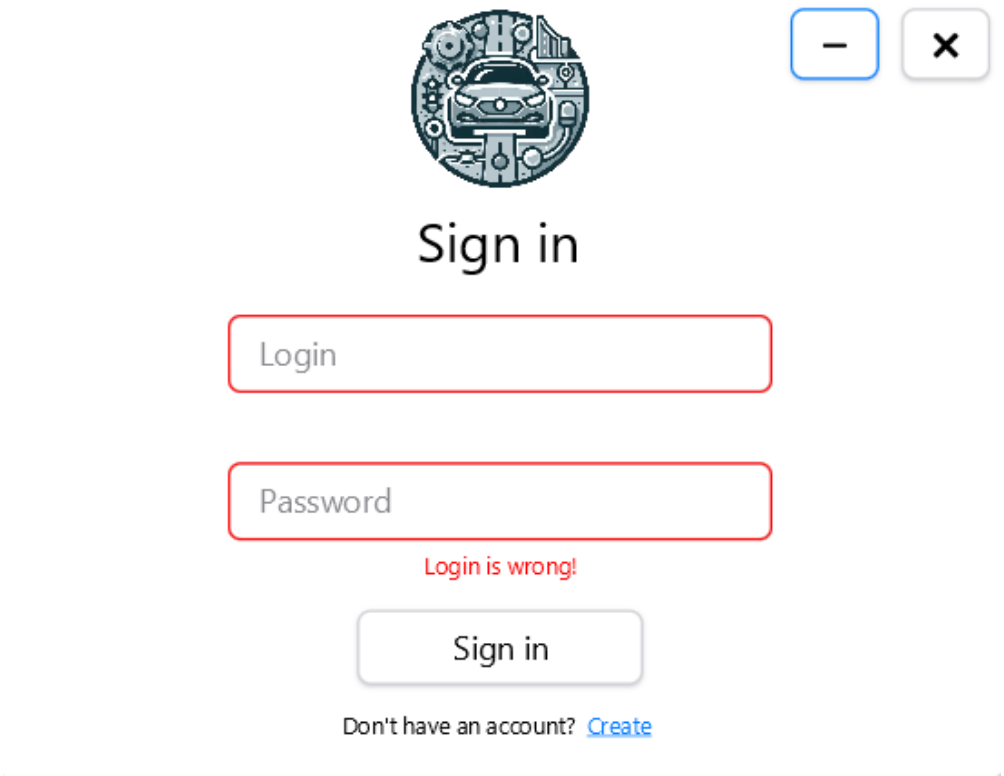


Рисунок 4.2 — Введення неіснуючого логіну

Якщо був введений неправильний пароль, буде виведене повідомлення «Password is wrong!» (рис 4.3).

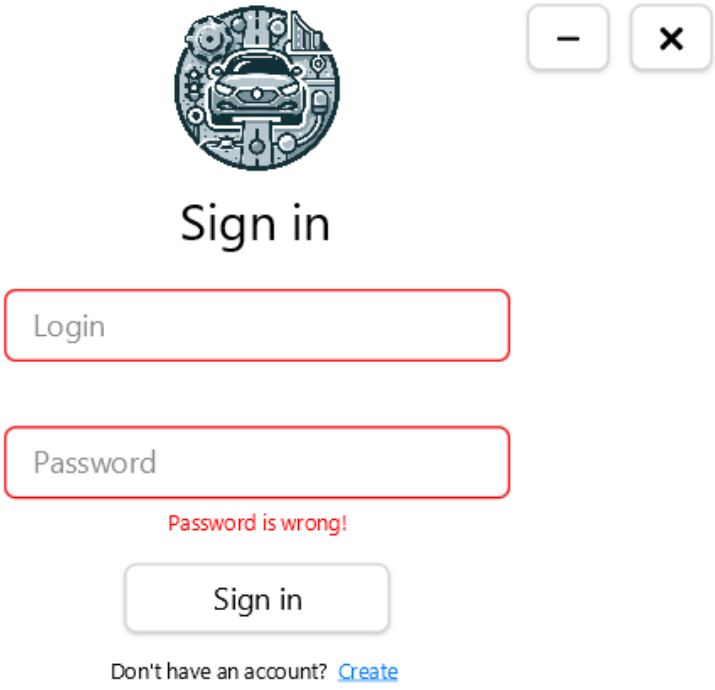


Рисунок 4.3 — Введення неправильного паролю

Якщо були введені правильні дані для авторизації, користувача перекине на головне вікно.

Наступним кроком слід перевірити реєстрацію. При спробі виконати реєстрацію з порожніми полями буде виведене повідомлення «Empty fields!» (рис. 4.4).

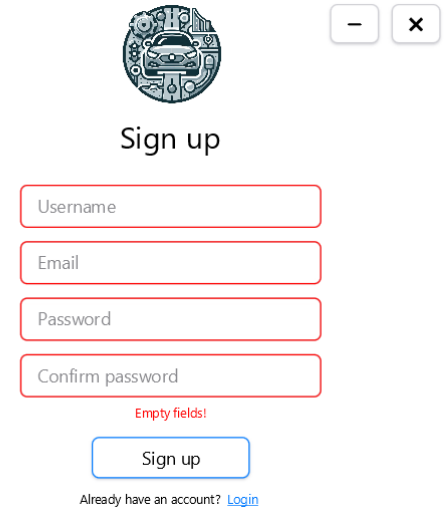


Рисунок 4.4 — Порожні поля

При спробі виконати реєстрацію з існуючим нікнеймом, буде виведене повідомлення «This username already in use!» (рис 4.5).

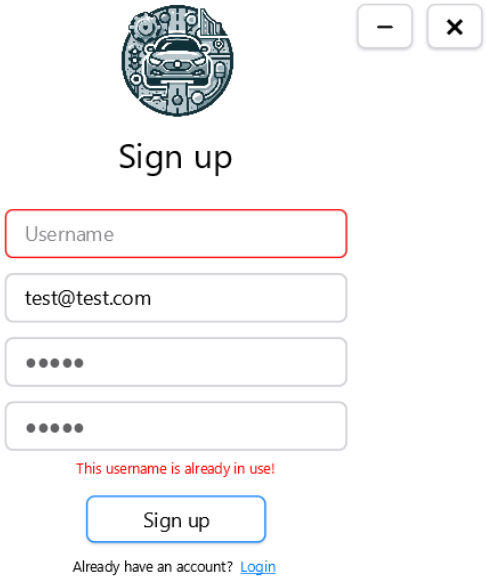


Рисунок 4.5 — Існуючий нікнейм

При спробі виконати реєстрацію з існуючим email, буде виведене повідомлення «This email already in use!» (рис 4.6).

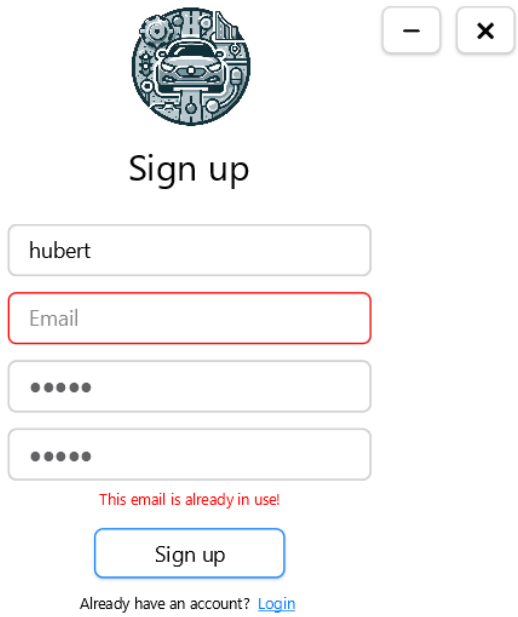


Рисунок 4.6 — Існуючий email

Якщо при спробі реєстрації дані в полях «Пароль» та «Підтвердити пароль» не збігаються, буде виведене повідомлення «The passwords don’t match!» (рис .4.7)

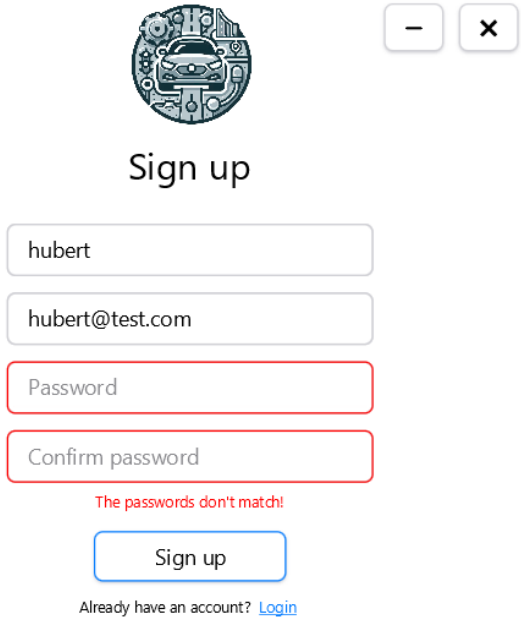


Рисунок 4.6 — Паролі не збігаються

Якщо користувач успішно зареєструвався, буде виведене повідомлення «Success!» і через декілька секунд його перекине на вікно авторизації (рис 4.8).

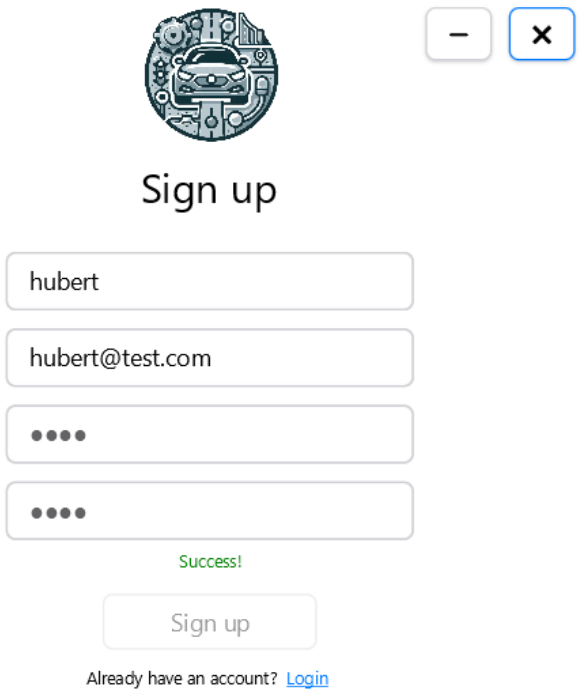


Рисунок 4.8 — Успішна реєстрація

При натисненні на кнопку «Меню» буде виведене контекстне меню, за допомогою якого можна здійснити вихід з системи (рис 4.9).

Рисунок 4.9 — Вихід з системи

У результаті тестування системи було доведено, що її функціонал працює відповідно до технічного завдання.

4.3 Розробка інструкції користувача

Для користування програмою спочатку потрібно завантажити Java Development Kit (JDK) з веб-сайту Oracle або з іншого джерела. Після завантаження слід виконати файл інсталяції. Його потрібно відкрити і дотримуватись інструкцій інсталятора. Зазвичай буде запропоновано вибрати шлях, за замовчуванням, куди буде встановлено JDK. Додатково можуть бути запропоновані опції налаштування середовища.

Наступним кроком буде встановлення .exe файлу програмного застосунку.

Файли з розширенням «.exe» — це виконувані файли для операційних систем Windows. Ці файли містять виконуваний код, який може запускати програми на комп’ютері. Після встановлення можна запустити програму.

Таблиця 4.1 демонструє мінімальні системні параметри, необхідні для роботи з усіма модулями застосунка.

Таблиця 4.1 – 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 4.2 демонструє рекомендовані системні параметри.

Таблиця 4.2 – Рекомендована конфігурація

Процесор	x64 2 ГГц
Обсяг оперативної пам’яті	2 ГБ для 64-разрядної системи

Продовження таблиці 4.2

Вільний простір диску	100 MB
Операційна система	Windows 11

4.4 Висновки

У четвертому розділі було виконано аналіз методів тестування програмного забезпечення, вибрано метод тестування «чорного ящика», проведено тестування системи для керування рухом транспорту. Також було розроблено інструкцію користувача програмного застосунку.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено програмні модулі для інтелектуального керування рухом транспорту.

Основні задачі, які були виконані:

– визначено загальний алгоритм системи;

– розроблено алгоритм симуляції руху транспорту на дорозі;

– розроблено зручний та адаптивний графічний інтерфейс системи;

– розроблено модуль для збереження даних;

– реалізовано систему моніторингу правопорушників, що перевищують швидкість;

– здійснено тестування системи згідно поставлених задач.

У першому розділі було розглянуто стан питання керування рухом транспорту. Також було проведено порівняльний аналіз аналогів з власним модулем, було доведено доцільність розробки власного програмного застосунку. Під час аналізу методів розв’язання поставленої задачі було вирішено використовувати агентні моделі. Було встановлено основні завдання, які необхідно виконати для розробки програмного застосунку.

У другому розділі було проведено аналіз вхідних даних системи, визначено етапи розробки програмних модулів. Було розроблено модель системи, що відображена за допомогою UML діаграми діяльності. Було розроблено логіку загального алгоритму системи для керування рухом транспорту, розроблено алгоритм автоматичного руху автомобіля, представлено блок-схеми алгоритмів.

У третьому розділі було проаналізовано мови програмування та інструменти для розробки. Було обґрунтовано вибір мови програмування Java для розробки програмного застосунку. Було обрано JavaFX для реалізації інтерфейсу програмного застосунку. Було розроблено інтерфейс програмного застосунку для інтелектуального керування рухом транспорту. Було розглянуто аргументи щодо вибору системи управління базами даних (СУБД), створено таблиці, виконано підключення бази даних до програмного застосунку. Було

розроблено модуль адаптивного керування світлофора та модуль моніторингу правопорушників.

У четвертому розділі було виконано аналіз методів тестування програмного забезпечення, вибрано метод тестування «чорного ящика», проведено тестування системи для керування рухом транспорту. Також було розроблено інструкцію користувача 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

ДОДАТКИ

ДОДАТОК А. Технічне завдання (обов'язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О.Н. Романюк
«12» березня 2024 р.

Технічне завдання

на бакалаврську дипломну роботу «Розробка методів і програмних засобів для інтелектуального керування рухом транспорту»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доц. каф. ПЗ Черноволик Г.О.
«12» березня 2024 р.

Виконав:
студент гр. 4ПІ-20Б О.О. Пліхта
«12» березня 2024 р.

м. Вінниця – 2024 р.

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка методів і програмних засобів для інтелектуального керування рухом транспорту».

Галузь застосування — транспортна сфера.

2. Підстава для розробки

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки

Метою роботи є підвищення безпеки на дорогах та якості керування рухом транспорту за рахунок використання методів і програмних засобів, що дозволить зменшити кількість аварій та інцидентів, покращити пропускну здатність доріг та скоротити час очікування для учасників дорожнього руху.

Призначення — можливість використання розробленої програми для системи інтелектуального керування рухом транспорту.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР:

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. Технічні вимоги

Склад комплексу *технічних засобів.*

Для розв’язку задачі буде використовуватися персональна електронно-обчислювальна машина (ПЕОМ).

Вимоги до складових *частин комплексу технічних засобів.*

Для нормальної роботи програми, необхідний персональний комп’ютер з наступною мінімальною конфігурацією:

1. Процесор x64 з частотою не менше 1 ГГц;

2. Оперативна пам’ять ємністю не менше 2 ГБ для 64-разрядної системи;

3. Вільний простір ємністю 100 МВ на запам’ятовуючому пристрої.

Вимоги до програмного *забезпечення.*

Програмний продукт розробляється на мові Java в середовищі розробки IntelliJ IDEA.

6. Конструктивні вимоги.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка;

– технічне обґрунтування доцільності розробки;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації.

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії і етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану питання та постановка задач	14.03.24 – 05.04.24
2	Розробка структури та моделі системи	06.04.24 – 15.04.24
3	Розробка алгоритмів роботи програмного застосунку	16.04.24 – 28.04.24
4	Аналіз та вибір засобів для реалізації програми	29.04.24 – 05.05.24
5	Розробка програмних модулів	06.05.24 – 20.05.24
6	Тестування роботи розробленого програмного застосунку	21.05.24 – 26.05.24
7	Оформлення матеріалів до захисту БДР	27.05.24 – 30.05.24

10. Порядок контролю і приймання

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи. Захист бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б. Протокол перевірки БДР на наявність текстових запозичень

(обов'язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка програмного забезпечення для знаходження оптимального шляху безпілотного літального апарату»

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доц. Черноволик Г.О.

Оригінальність	96%
Схожість	4%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Пліхта О. О.

Керівник роботи Черноволик Г. О.

ДОДАТОК В. Лістинг програми (довідковий)

```
//Модуль DatabaseHandler.java
package pr.traffict.database;

import javafx.util.Pair;
import org.mindrot.jbcrypt.BCrypt;
import pr.traffict.Car;
import pr.traffict.Offence;
import pr.traffict.User;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.util.ArrayList;

public class DatabaseHandler extends Configs {
   Connection dbConnection;

   public Connection getDbConnection() throws ClassNotFoundException, SQLException {
      String connectionString = "jdbc:mysql://" + dbHost + ":" + dbPort + "/" + dbName;

      Class.forName("com.mysql.cj.jdbc.Driver");

      dbConnection = DriverManager.getConnection(connectionString, dbUser, dbPass);

      return dbConnection;
   }

   public void signUpUser(User user) {
      String insert = "INSERT INTO " + Const.USER_TABLE + "(" + Const.USER_UNAME
+ ","
            + Const.USER_EMAIL + "," + Const.USER_PASSWORD + "," +
Const.USER_STATUS + ")" + "VALUES(BINARY ?,?,?,?)";

      try {
         PreparedStatement preparedStatement =
getDbConnection().prepareStatement(insert);
         preparedStatement.setString(1, user.getUsername());
         preparedStatement.setString(2, user.getEmail());
         preparedStatement.setString(3, user.getPassword());
         preparedStatement.setString(4, user.getStatus());
         preparedStatement.executeUpdate();
      } catch (SQLException | ClassNotFoundException e) {
         throw new RuntimeException(e);
      }
   }

   public boolean userExists(String username) {
```

```
        String query = "SELECT * FROM " + Const.USER_TABLE + " WHERE BINARY " +
Const.USER_UNAME + " = ?";

        try {
            PreparedStatement preparedStatement =
getDbConnection().prepareStatement(query);
            preparedStatement.setString(1, username);
            ResultSet resultSet = preparedStatement.executeQuery();

            return resultSet.next();
        } catch (SQLException | ClassNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    public boolean emailExists(String email) {
        String query = "SELECT * FROM " + Const.USER_TABLE + " WHERE BINARY " +
Const.USER_EMAIL + " = ?";

        try {
            PreparedStatement preparedStatement =
getDbConnection().prepareStatement(query);
            preparedStatement.setString(1, email);
            ResultSet resultSet = preparedStatement.executeQuery();

            return resultSet.next();
        } catch (SQLException | ClassNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    public boolean checkPassword(User user) {
        String selectQuery = "SELECT " + Const.USER_PASSWORD + " FROM " +
Const.USER_TABLE +
                " WHERE BINARY " + Const.USER_UNAME + " = ?";

        try {
            PreparedStatement preparedStatement =
getDbConnection().prepareStatement(selectQuery);
            preparedStatement.setString(1, user.getUsername());
            ResultSet resultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                String hashedPasswordFromDB = resultSet.getString(Const.USER_PASSWORD);

                if (BCrypt.checkpw(user.getPassword(), hashedPasswordFromDB)) {
                    return true;
                }
            }
        } catch (SQLException | ClassNotFoundException e) {
            throw new RuntimeException(e);
        }
```

```
        return false;
    }

    public User getUpdateUser(User user) {
        String username = user.getUsername();
        String selectQuery = "SELECT * FROM " + Const.USER_TABLE + " WHERE " +
Const.USER_UNAME + " = ?";

        try {
            PreparedStatement preparedStatement =
getDbConnection().prepareStatement(selectQuery);
            preparedStatement.setString(1, username);
            ResultSet resultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                user.setEmail(resultSet.getString(Const.USER_EMAIL));
                user.setStatus(resultSet.getString(Const.USER_STATUS));
            }
        } catch (SQLException | ClassNotFoundException e) {
            throw new RuntimeException(e);
        }

        return user;
    }

    public void registerOffence(Offence offence) {
        String insert = "INSERT INTO " + Const.OFFENCE_TABLE + "(" +
Const.OFFENCE_TYPE + ","
                + Const.OFFENCE_CAR_NUMBER + "," + Const.OFFENCE_CAR_MODEL +
"," + Const.OFFENCE_TIME + ")" + "VALUES(BINARY ?,?,?,?)";

        try {
            PreparedStatement preparedStatement =
getDbConnection().prepareStatement(insert);
            preparedStatement.setString(1, offence.getType());
            preparedStatement.setString(2, offence.getCarNumber());
            preparedStatement.setString(3, offence.getCarModel());
            preparedStatement.setString(4, offence.getTime());
            preparedStatement.executeUpdate();
        } catch (SQLException | ClassNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    public int checkOffence(Car car) {
        int counter = 0;
        String selectQuery = "SELECT COUNT(*) AS count FROM " +
Const.OFFENCE_TABLE + " WHERE BINARY " + Const.OFFENCE_CAR_NUMBER + "
= ?";

        try {
```

```
            PreparedStatement preparedStatement =
getDbConnection().prepareStatement(selectQuery);
            preparedStatement.setString(1, car.getNumber());
            ResultSet resultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                counter = resultSet.getInt("count");
            }
        } catch (SQLException | ClassNotFoundException e) {
            throw new RuntimeException(e);
        }

        return counter;
    }

    public ArrayList<Pair<String, String>> getArrayList() {
        ArrayList<Pair<String, String>> carArrayList = new ArrayList<>();

        String selectQuery = "SELECT " + Const.OFFENCE_CAR_NUMBER + ", " +
Const.OFFENCE_CAR_MODEL + " FROM " + Const.OFFENCE_TABLE;

        try {
            PreparedStatement preparedStatement =
getDbConnection().prepareStatement(selectQuery);
            ResultSet resultSet = preparedStatement.executeQuery();

            while (resultSet.next()) {
                String number = resultSet.getString(Const.OFFENCE_CAR_NUMBER);
                String model = resultSet.getString(Const.OFFENCE_CAR_MODEL);
                carArrayList.add(new Pair<>(number, model));
            }
        } catch (SQLException | ClassNotFoundException e) {
            throw new RuntimeException(e);
        }

        return carArrayList;
    }

    public ArrayList<Offence> getAllOffences() {
        ArrayList<Offence> offencesList = new ArrayList<>();

        String selectQuery = "SELECT * FROM " + Const.OFFENCE_TABLE;

        try {
            PreparedStatement preparedStatement =
getDbConnection().prepareStatement(selectQuery);
            ResultSet resultSet = preparedStatement.executeQuery();

            while (resultSet.next()) {
                String type = resultSet.getString(Const.OFFENCE_TYPE);
                String carNumber = resultSet.getString(Const.OFFENCE_CAR_NUMBER);
                String carModel = resultSet.getString(Const.OFFENCE_CAR_MODEL);
```

```
            String time = resultSet.getString(Const.OFFENCE_TIME);
            offencesList.add(new Offence(type, carNumber, carModel, time));
        }
    } catch (SQLException | ClassNotFoundException e) {
        throw new RuntimeException(e);
    }

    return offencesList;
  }

  public ArrayList<Offence> getOffencesByCarNumber(Car car) {
    ArrayList<Offence> offencesList = new ArrayList<>();

    String selectQuery = "SELECT * FROM " + Const.OFFENCE_TABLE + " WHERE
BINARY " + Const.OFFENCE_CAR_NUMBER + " = ?";

    try {
        PreparedStatement preparedStatement =
getDbConnection().prepareStatement(selectQuery);
        preparedStatement.setString(1, car.getNumber());
        ResultSet resultSet = preparedStatement.executeQuery();

        while (resultSet.next()) {
            String type = resultSet.getString(Const.OFFENCE_TYPE);
            String carNumber = resultSet.getString(Const.OFFENCE_CAR_NUMBER);
            String carModel = resultSet.getString(Const.OFFENCE_CAR_MODEL);
            String time = resultSet.getString(Const.OFFENCE_TIME);
            offencesList.add(new Offence(type, carNumber, carModel, time));
        }
    } catch (SQLException | ClassNotFoundException e) {
        throw new RuntimeException(e);
    }

    return offencesList;
  }
}
```

//Модуль Car.java

```
package pr.traffict;

import javafx.animation.AnimationTimer;
import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.fxml.FXMLLoader;
import javafx.scene.Group;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.image.ImageView;
import javafx.scene.input.MouseButton;
import javafx.scene.input.MouseEvent;
import javafx.scene.paint.Color;
```

```
import javafx.scene.shape.Rectangle;
import javafx.stage.Modality;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
import javafx.util.Duration;
import pr.traffict.controllers.CarInfoController;
import pr.traffict.controllers.MainController;

import java.io.IOException;
import java.util.Random;

public class Car {
    private String number;
    private String model;
    private Group carGroup;
    private ImageView car;
    private Rectangle impactArea;
    private Rectangle sArea;
    private double chordX;
    private double chordY;
    private int speed;
    private int direction;
    private boolean nearLight;
    private Road road;
    private boolean speedUp;

    public String getNumber() {
        return number;
    }

    public String getModel() {
        return model;
    }

    public int getSpeed() {
        return speed;
    }

    public Group getCarGroup() {
        return carGroup;
    }

    public ImageView getCar() {
        return car;
    }

    public Rectangle getImpactArea() {
        return impactArea;
    }

    public Rectangle getsArea() {
        return sArea;
```

```
    }

    public void setChordX(double chordX) {
        this.chordX = chordX;
    }

    public void setChordY(double chordY) {
        this.chordY = chordY;
    }

    public int getDirection() {
        return direction;
    }

    public boolean isNearLight() {
        return nearLight;
    }

    public boolean isSpeedUp() {
        return speedUp;
    }

    public void setSpeedUp(boolean speedUp) {
        this.speedUp = speedUp;
    }

    public Car(String number, String model, int speed, Road road) {
        this.number = number;
        this.model = model;
        this.speed = speed;
        this.road = road;
        this.direction = new Random().nextInt(4) + 1;
        this.nearLight = false;
        this.speedUp = false;
        setGR();
        this.impactArea.setFill(Color.TRANSPARENT);
        this.sArea.setFill(Color.TRANSPARENT);
        carGroup.addEventHandler(MouseEvent.MOUSE_CLICKED, event -> {
            if (event.getButton() == MouseButton.PRIMARY) {
                if (MainController.getUser().getStatus().equals("admin")) {
                    FXMLLoader loader = new
FXMLLoader(getClass().getResource("/pr/traffict/carInfo.fxml"));
                    CarInfoController controller = new CarInfoController(this);
                    loader.setController(controller);
                    Stage window = new Stage();
                    window.initStyle(StageStyle.TRANSPARENT);
                    Parent root = null;
                    try {
                        root = loader.load();
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
```

```
                Scene scene = new Scene(root, 380, 370);
                scene.setFill(null);
                window.setTitle("Car Info");
                window.setScene(scene);
                window.setResizable(false);
                window.initModality(Modality.APPLICATION_MODAL);
                window.show();
            }
        }
    });
}

private void setGR() {
    switch (direction) {
        case 1:
            this.impactArea = new Rectangle(73, 30);
            this.sArea = new Rectangle(20, 30);
            this.sArea.setTranslateX(53);
            this.car = new ImageView("file:src/main/resources/images/carGR" + (new
Random().nextInt(3) + 1) + ".png");
            this.carGroup = new Group(impactArea, car, sArea);
            break;
        case 2:
            this.impactArea = new Rectangle(73, 30);
            this.sArea = new Rectangle(20, 30);
            this.car = new ImageView("file:src/main/resources/images/carGL" + (new
Random().nextInt(3) + 1) + ".png");
            this.car.setTranslateX(20);
            this.carGroup = new Group(impactArea, car, sArea);
            break;
        case 3:
            this.impactArea = new Rectangle(30, 73);
            this.sArea = new Rectangle(30, 20);
            this.car = new ImageView("file:src/main/resources/images/carVU" + (new
Random().nextInt(3) + 1) + ".png");
            this.car.setTranslateY(20);
            this.carGroup = new Group(impactArea, car, sArea);
            break;
        case 4:
            this.impactArea = new Rectangle(30, 73);
            this.sArea = new Rectangle(30, 20);
            this.sArea.setTranslateY(53);
            this.car = new ImageView("file:src/main/resources/images/carVD" + (new
Random().nextInt(3) + 1) + ".png");
            this.carGroup = new Group(impactArea, car, sArea);
            break;
    }
}

public void setCarChord() {
    this.impactArea.setX(this.chordX);
    this.impactArea.setY(this.chordY);
```

```
        this.sArea.setX(this.chordX);
        this.sArea.setY(this.chordY);
        this.car.setX(this.chordX);
        this.car.setY(this.chordY);
    }

    public void autoMove() {
        for (Car otherCar : road.getCars()) {
            if (otherCar != Car.this &&
impactArea.getBoundsInParent().intersects(otherCar.getImpactArea().getBoundsInParent())
&& otherCar.getDirection() == direction) {
                road.deleteCar(Car.this);
                return;
            }
        }
        AnimationTimer timer = new AnimationTimer() {
            double destinationX = chordX;
            double destinationY = chordY;

            @Override
            public void handle(long now) {
                if (nearLight) return;
                switch (direction) {
                    case 1:
                        destinationX += speed;
                        if (destinationX <= Road.getRoot().getWidth() - impactArea.getWidth()) {
                            car.setX(destinationX);
                            impactArea.setX(destinationX);
                            sArea.setX(destinationX);
                            chordX = destinationX;
                            checkLight();
                            checkCollision();
                            checkNearCollision();
                            speedUp();
                        } else {
                            removeFromRoad();
                            this.stop();
                        }
                        break;
                    case 2:
                        destinationX -= speed;
                        if (destinationX >= 0) {
                            car.setX(destinationX);
                            impactArea.setX(destinationX);
                            sArea.setX(destinationX);
                            chordX = destinationX;
                            checkLight();
                            checkCollision();
                            checkNearCollision();
                            speedUp();
                        } else {
                            removeFromRoad();
```

```
                this.stop();
            }
            break;
        case 3:
            destinationY -= speed;
            if (destinationY >= 0) {
                car.setY(destinationY);
                impactArea.setY(destinationY);
                sArea.setY(destinationY);
                chordY = destinationY;
                checkLight();
                checkCollision();
                checkNearCollision();
                speedUp();
            } else {
                removeFromRoad();
                this.stop();
            }
            break;
        case 4:
            destinationY += speed;
            if (destinationY <= Road.getRoot().getHeight() - impactArea.getHeight()) {
                car.setY(destinationY);
                impactArea.setY(destinationY);
                sArea.setY(destinationY);
                chordY = destinationY;
                checkLight();
                checkCollision();
                checkNearCollision();
                speedUp();
            } else {
                removeFromRoad();
                this.stop();
            }
            break;
    }
}

private void removeFromRoad() {
    road.deleteCar(Car.this);
}

private void speedUp() {
    if(Math.random() <= 0.00001 && speed == 2) {
        speedUp = true;
        speed++;
    }
}

private void checkCollision() {
    for (Car otherCar : road.getCars()) {
        if (otherCar != Car.this &&
```

```
sArea.getBoundsInParent().intersects(otherCar.getImpactArea().getBoundsInParent()) &&
                    otherCar.getDirection() == direction && speed > 1) {
                speed = 1;
                Timeline pauseTimeline = new Timeline(new
KeyFrame(Duration.seconds(2), event -> {
                    if
(!(sArea.getBoundsInParent().intersects(otherCar.getImpactArea().getBoundsInParent()))) {
                        nearLight = false;
                        if (speed < 2)
                            speed++;
                    }
                }));
                pauseTimeline.setCycleCount(1);
                pauseTimeline.play();
                return;
            }
        }
    }

    private void checkLight() {
        for (TrafficLight otherLight : road.getLights()) {
            if
(sArea.getBoundsInParent().intersects(otherLight.getLine().getBoundsInParent()) &&
                    !otherLight.isStatus() && !nearLight) {
                nearLight = true;
                speed = 0;
                if (otherLight.getCarsNearLight().contains(Car.this)) return;
                otherLight.addNearCar(Car.this);

                Timeline pauseTimeline = new Timeline(new
KeyFrame(Duration.seconds(2), event -> {
                    if (otherLight.isStatus()) {
                        nearLight = false;
                        if (speed < 2)
                            speed++;
                        otherLight.deleteNearCar(Car.this);
                    }
                }));
                pauseTimeline.setCycleCount(Timeline.INDEFINITE);
                pauseTimeline.playFromStart();

                pauseTimeline.setOnFinished(event -> pauseTimeline.stop());
            }
        }
    }

    private void checkNearCollision() {
        for (Car otherCar : road.getCars()) {
            if (otherCar != Car.this &&

sArea.getBoundsInParent().intersects(otherCar.getImpactArea().getBoundsInParent()) &&
```

```
                    otherCar.getDirection() == direction && otherCar.isNearLight()) {
                nearLight = true;
                speed = 0;
                Timeline pauseTimeline = new Timeline(new
KeyFrame(Duration.seconds(2), event -> {
                    if (!otherCar.isNearLight()) {
                        nearLight = false;
                        if (speed < 2)
                            speed++;
                    }
                }));
                pauseTimeline.setCycleCount(Timeline.INDEFINITE);
                pauseTimeline.playFromStart();

                pauseTimeline.setOnFinished(event -> pauseTimeline.stop());
            }
        }
    }
};
timer.start();
}
}
```

//Модуль Offence.java

```
package pr.traffict;

public class Offence {
    private final String type;
    private final String carNumber;
    private final String carModel;
    private final String time;

    public Offence(String type, String carNumber, String carModel, String time) {
        this.type = type;
        this.carNumber = carNumber;
        this.carModel = carModel;
        this.time = time;
    }

    public String getType() {
        return type;
    }

    public String getCarNumber() {
        return carNumber;
    }

    public String getCarModel() {
        return carModel;
    }
```

```
    public String getTime() {
        return time;
    }
}
```

//Модуль Road.java

```
package pr.traffict;

import javafx.geometry.Point2D;
import javafx.scene.image.Image;
import javafx.scene.layout.Pane;
import javafx.scene.paint.ImagePattern;
import javafx.scene.shape.Line;
import javafx.scene.shape.Rectangle;

import java.util.ArrayList;
import java.util.Random;

public class Road {
    private final static int rootHeight = 1140;
    private final static int rootWidth = 1800;
    private static final Pane root = new Pane();
    public static Pane getRoot() {
        return root;
    }
    private final ArrayList<Car> cars;
    private final ArrayList<TrafficLight> lights;

    public ArrayList<Car> getCars() {
        return cars;
    }
    public ArrayList<TrafficLight> getLights() {
        return lights;
    }

    private final Point2D[] carSpawnCoordinates = new Point2D[]{
            new Point2D(191, 0),
            …
    };

    private static final Point2D[] trafficLightSpawnCoordinates = new Point2D[]{
            new Point2D(285, 370),
            …
    };

    private static final Point2D[] trafficLineSpawnCoordinates = new Point2D[]{
            new Point2D(232, 370),
            …
    };

    public Road(){
```

```
        root.setMinWidth(rootWidth);
        root.setMinHeight(rootHeight);
        root.setStyle("-fx-border-radius: 10px;");
        Rectangle rectangle = new Rectangle(rootWidth, rootHeight);
        Image img = new Image("file:src/main/resources/images/background.png");
        rectangle.setFill(new ImagePattern(img));
        root.getChildren().add(rectangle);
        this.cars = new ArrayList<>();
        this.lights = new ArrayList<>();
    }

    public void addNewCar(Car car){
        this.cars.add(car);
        int direction = car.getDirection();

        int start = 0, end = 0;
        end = switch (direction) {
            case 1 -> {
                start = 6;
                yield 7;
            }
            case 2 -> {
                start = 8;
                yield 9;
            }
            case 3 -> {
                start = 3;
                yield 5;
            }
            case 4 -> 2;
            default -> end;
        };

        Point2D randomCoordinate = carSpawnCoordinates[start + new Random().nextInt(end
- start + 1)];
        car.setChordX(randomCoordinate.getX());
        car.setChordY(randomCoordinate.getY());
        car.setCarChord();
        root.getChildren().add(car.getCarGroup());
    }

    public void deleteCar(Car car){
        root.getChildren().remove(car.getCarGroup());
        this.cars.remove(car);
    }

    public void addTrafficLight(TrafficLight trafficLight, int i) {
        this.lights.add(trafficLight);
        Point2D rCoordinate = trafficLightSpawnCoordinates[i];
        Point2D r2Coordinate = trafficLineSpawnCoordinates[i];
        trafficLight.setChordX(rCoordinate.getX());
        trafficLight.setChordY(rCoordinate.getY());
```

```
        trafficLight.setTrafficLightChord();
        if (i % 2 == 0){
            trafficLight.setLine(new Line(r2Coordinate.getX(), r2Coordinate.getY(),
r2Coordinate.getX() + 30, r2Coordinate.getY()));
        } else {
            trafficLight.setLine(new Line(r2Coordinate.getX(), r2Coordinate.getY(),
r2Coordinate.getX(), r2Coordinate.getY() + 30));
        }
        root.getChildren().addAll(trafficLight.getTrafficLightGroup(), trafficLight.getLine());
    }

    public void clearRoad() {
        for (Car car : cars) {
            root.getChildren().remove(car.getCarGroup());
        }
        for (TrafficLight light : lights) {
            root.getChildren().remove(light.getTrafficLightGroup());
        }
        cars.clear();
        lights.clear();
    }
}

//Модуль Scanner.java

package pr.traffict;

import javafx.animation.AnimationTimer;
import pr.traffict.database.DatabaseHandler;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class Scanner {
    private Road road;

    public Scanner(Road road) {
        this.road = road;
    }

    public void scanRoad() {
        AnimationTimer timer = new AnimationTimer() {
            @Override
            public void handle(long now) {
                for (Car car : road.getCars()) {
                    if (car.getSpeed() >= 3 && car.isSpeedUp()) {
                        LocalDateTime currentTime = LocalDateTime.now();
                        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");
                        String formattedTime = currentTime.format(formatter);
                        Offence offence = new Offence("speeding", car.getNumber(),
car.getModel(), formattedTime);
                        car.setSpeedUp(false);
```

```
                    new DatabaseHandler().registerOffence(offence);
                }
            }

            for (TrafficLight light : road.getLights()) {
                if (light.isNeedToChange()) {
                    for (TrafficLight otherLight : road.getLights()) {
                        if (light != otherLight && light.getLightsGroupDeterminer() ==
otherLight.getLightsGroupDeterminer()) {
                            otherLight.changeStatus();
                        }
                    }
                    light.setNeedToChange(false);
                }
            }
        }
    };
    timer.start();
  }
}


//Модуль Spawner.java

package pr.traffict;

import javafx.application.Platform;
import javafx.util.Pair;
import pr.traffict.database.DatabaseHandler;

import java.util.*;

public class Spawner {
    private Road road;
    private Random random;
    private ArrayList<Pair<String, String>> carArrayList;
    private Timer timer;
    private final String[] REGIONS = {"AB", "AC", "AE", "AH", "AM", "AO", "AP", "AT",
"AI", "AA", "BA", "BB", "BC", "BE", "BH", "BI", "BK", "BM", "BO", "AX", "BT", "BX",
"CA", "CB", "CE"};
    private final String[] LETTERS = {"A", "B", "E", "K", "M", "H", "O", "P", "C", "T", "У",
"X"};
    private final String[] MODELS = {"Toyota", "Honda", "Ford", "BMW", "Chevrolet",
"Tesla", "Mercedes-Benz", "Audi", "Nissan", "Volkswagen"};

    public Spawner(Road road) {
        this.road = road;
        this.random = new Random();
        this.carArrayList = new DatabaseHandler().getArrayList();
    }

    public void startSpawning() {
        timer = new Timer();
```

```
    timer.scheduleAtFixedRate(new TimerTask() {
        @Override
        public void run() {
            spawnCar();
        }
    }, 0, 1000);
}

public void spawnLights() {
    for (int i = 0; i < 24; i++) {
        TrafficLight trafficLight;
        if(i % 2 == 0) {
            trafficLight = new TrafficLight(i, true, road);
        } else {
            trafficLight = new TrafficLight(i, false, road);
        }
        road.addTrafficLight(trafficLight, i);
        trafficLight.carScan();
    }
}

public void stopSpawning() {
    timer.cancel();
}

private void spawnCar() {
    Platform.runLater(() -> {
        String number = generateCarNumber();
        String model = generateCarModel(number);
        Car car = new Car(number, model, 2, road);
        road.addNewCar(car);
        car.autoMove();

        timer.cancel();
        timer = new Timer();
        timer.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                spawnCar();
            }
        }, getRandomSpawnInterval(), getRandomSpawnInterval());
    });
}

private long getRandomSpawnInterval() {
    return random.nextInt(250) + 250;
}

private String generateCarNumber() {
    String regionCode = REGIONS[random.nextInt(REGIONS.length)];
    int lettersCode = random.nextInt(9000) + 1000;
    String rCode = LETTERS[random.nextInt(LETTERS.length)] +
```

```
LETTERS[random.nextInt(LETTERS.length)];
        return regionCode + " " + lettersCode + " " + rCode;
    }

    private String generateCarModel(String number) {
        String candidateModel = "";
        boolean found = false;
        for (Pair<String, String> pair : carArrayList) {
            if (pair.getKey().equals(number)) {
                candidateModel = pair.getValue();
                found = true;
                break;
            }
        }
        if (!found) {
            candidateModel = MODELS[random.nextInt(MODELS.length)];
            carArrayList.add(new Pair<>(number, candidateModel));
        }
        return candidateModel;
    }
}
```

//Модуль TrafficLight.java

```
package pr.traffict;

import javafx.animation.Animation;
import javafx.animation.AnimationTimer;
import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.scene.Group;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Line;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Text;
import javafx.util.Duration;

import java.util.ArrayList;

public class TrafficLight {
    private Rectangle border;
    private Circle redCircle;
    private Circle yellowCircle;
    private Circle greenCircle;
    private Text timeText;
    private Group trafficLightGroup;
    private double chordX;
    private double chordY;
    private int direction;
    private boolean status;
    private Line line;
```

```
private int time;
private int currentTime;
private int lightsGroupDeterminer;
private ArrayList<Car> carsNearLight;
private Timeline timeline;
private Timeline currenttimeline;
private boolean needToChange;
private Road road;

public Group getTrafficLightGroup() {
   return trafficLightGroup;
}

public void setChordX(double chordX) {
   this.chordX = chordX;
}

public boolean isStatus() {
   return status;
}

public void setChordY(double chordY) {
   this.chordY = chordY;
}

public void setLine(Line line) {
   this.line = line;
   this.line.setStroke(Color.WHITE);
   this.line.setOpacity(0);
}

public Line getLine() {
   return line;
}

public ArrayList<Car> getCarsNearLight() {
   return carsNearLight;
}

public void addNearCar(Car car) {
   this.carsNearLight.add(car);
}

public void deleteNearCar(Car car) {
   this.carsNearLight.remove(car);
}

public int getLightsGroupDeterminer() {
   return lightsGroupDeterminer;
}

public boolean isNeedToChange() {
```

```
        return needToChange;
    }

    public void setNeedToChange(boolean needToChange) {
        this.needToChange = needToChange;
    }

    public TrafficLight(int direction, boolean status, Road road) {
        this.border = new Rectangle(32, 72);
        this.border.setStrokeWidth(2);
        this.border.setStroke(Color.BLACK);
        this.border.setFill(Color.BLACK);
        this.timeText = new Text();
        this.timeText.setX(8);
        this.timeText.setY(41);
        this.timeText.setStroke(Color.WHITE);
        this.road = road;
        this.status = status;
        this.needToChange = false;
        this.time = 15;
        this.currentTime = this.time;
        setC();
        this.trafficLightGroup = new Group(border, redCircle, yellowCircle, greenCircle,
timeText);
        this.direction = direction;
        this.carsNearLight = new ArrayList<>();
        setR();
        setD();
        updateStatus();
    }

    public void setTrafficLightChord(){
        this.trafficLightGroup.setLayoutX(this.chordX);
        this.trafficLightGroup.setLayoutY(this.chordY);
    }

    public void changeStatus() {
        stopTimeline();
        updateStatus();
    }

    private Circle createCircle(double centerX, double centerY, Color color) {
        Circle circle = new Circle(centerX, centerY, 10);
        circle.setFill(color);
        return circle;
    }

    private void setC() {
        if (this.status) {
            this.redCircle = createCircle(16, 14, Color.TRANSPARENT);
            this.redCircle.setStroke(Color.WHITE);
            this.yellowCircle = createCircle(16, 36, Color.TRANSPARENT);
```

```
            this.yellowCircle.setStroke(Color.WHITE);
            this.greenCircle = createCircle(16, 58, Color.GREEN);
            this.greenCircle.setStroke(Color.WHITE);
        } else {
            this.redCircle = createCircle(16, 14, Color.RED);
            this.redCircle.setStroke(Color.WHITE);
            this.yellowCircle = createCircle(16, 36, Color.TRANSPARENT);
            this.yellowCircle.setStroke(Color.WHITE);
            this.greenCircle = createCircle(16, 58, Color.TRANSPARENT);
            this.greenCircle.setStroke(Color.WHITE);
        }
    }

    private void setR() {
        switch (direction) {
            case 0, 4, 8, 12, 16, 20:
                this.trafficLightGroup.setRotate(0);
                break;
            case 1, 5, 9, 13, 17, 21:
                this.trafficLightGroup.setRotate(90);
                break;
            case 2, 6, 10, 14, 18, 22:
                this.trafficLightGroup.setRotate(180);
                break;
            case 3, 7, 11, 15, 19, 23:
                this.trafficLightGroup.setRotate(270);
                break;
        }
    }

    private void setD() {
        switch (direction) {
            case 0, 1, 2, 3:
                this.lightsGroupDeterminer = 1;
                break;
            case 4, 5, 6, 7:
                this.lightsGroupDeterminer = 2;
                break;
            case 8, 9, 10, 11:
                this.lightsGroupDeterminer = 3;
                break;
            case 12, 13, 14, 15:
                this.lightsGroupDeterminer = 4;
                break;
            case 16, 17, 18, 19:
                this.lightsGroupDeterminer = 5;
                break;
            case 20, 21, 22, 23:
                this.lightsGroupDeterminer = 6;
                break;
        }
    }
```

```
    public void carScan () {
        AnimationTimer timer = new AnimationTimer() {
            @Override
            public void handle(long now) {
                if (carsNearLight.size() > 4) {
                    needToChange = true;
                    changeStatus();
                }

                if (status) {
                    carsNearLight.clear();
                }

                for (Car car : road.getCars()) {
                    if (carsNearLight.contains(car)) {
                        for (Car otherCar : road.getCars()) {
                            if (otherCar != car && !carsNearLight.contains(otherCar) &&

otherCar.getsArea().getBoundsInParent().intersects(car.getImpactArea().getBoundsInParent
()) &&
                                    otherCar.getDirection() == car.getDirection() &&
otherCar.isNearLight()) {
                                carsNearLight.add(otherCar);
                            }
                        }
                    }
                }
            }
        };
        timer.start();
    }

    private void updateStatus() {
        switchLights();
        updateTimer();
        timeline = new Timeline(new KeyFrame(Duration.seconds(time), event -> {
            switchLights();
            updateTimer();
        }));
        timeline.setCycleCount(Animation.INDEFINITE);
        timeline.play();
    }

    private void switchLights() {
        this.status = !status;
        if (this.status) {
            this.redCircle.setFill(Color.TRANSPARENT);
            this.greenCircle.setFill(Color.GREEN);
        } else {
            this.redCircle.setFill(Color.RED);
            this.greenCircle.setFill(Color.TRANSPARENT);
```

```
        }
    }

    private void updateTimer() {
        currenttimeline = new Timeline(new KeyFrame(Duration.seconds(1), event -> {
            if (currentTime > 0) {
                currentTime--;
                if (currentTime < 10) {
                    timeText.setX(12);
                } else {
                    timeText.setX(8);
                }
                if (currentTime == 0) {
                    timeText.setText("");
                } else {
                    timeText.setText(String.valueOf(currentTime));
                }
            }
        }));
        currenttimeline.setCycleCount(time);
        currenttimeline.play();
        this.currentTime = this.time;
    }

    private void stopTimeline() {
        if (timeline != null) {
            timeline.stop();
            currenttimeline.stop();
        }
    }

}


//Модуль User.java

package pr.traffict;

public class User {
    private String username;
    private String email;
    private String password;
    private String status;

    public User() {}

    public User(String username, String email, String password, String status) {
        this.username = username;
        this.email = email;
        this.password = password;
        this.status = status;
    }
```

```
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }
}
```

```
//Модуль Const.java
package pr.traffict.database;


public class Const {
    public static final String USER_TABLE = "users";
    public static final String USER_ID = "user_id";
    public static final String USER_UNAME = "username";
    public static final String USER_EMAIL = "email";
    public static final String USER_PASSWORD = "password";
    public static final String USER_STATUS = "status";


    public static final String OFFENCE_TABLE = "offences";
```

```
    public static final String OFFENCE_ID = "offence_id";
    public static final String OFFENCE_TYPE = "type";
    public static final String OFFENCE_CAR_NUMBER = "car_number";
    public static final String OFFENCE_CAR_MODEL = "car_model";
    public static final String OFFENCE_TIME = "time";
}

//Модуль Shake.java
package pr.traffict.animations;

import javafx.animation.TranslateTransition;
import javafx.scene.Node;
import javafx.util.Duration;

public class Shake {
    TranslateTransition tt;
    public Shake (Node node) {
        tt = new TranslateTransition(Duration.millis(70), node);
        tt.setFromX(0f);
        tt.setByX(10f);
        tt.setCycleCount(4);
        tt.setAutoReverse(true);
    }

    public void playAnimation() {
        tt.playFromStart();
    }
}

//Модуль CarInfoController.java
package pr.traffict.controllers;

import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
```

```
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.input.MouseEvent;
import javafx.stage.Modality;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
import pr.traffict.Car;
import pr.traffict.database.DatabaseHandler;

import java.io.IOException;

public class CarInfoController {
    private final Car car;
    @FXML
    private Button close_button;
    @FXML
    private Button trey_button;
    @FXML
    private Label number_label;
    @FXML
    private Label model_label;
    @FXML
    private Label speed_label;
    @FXML
    private Label offence_label;

    public CarInfoController (Car car) {
        this.car = car;
    }

    @FXML
    public void initialize(){
        number_label.setText(car.getNumber());
```

```
        model_label.setText(car.getModel());
        speed_label.setText(String.valueOf(car.getSpeed()));
        setOffenceLabel();
        close_button.addEventHandler(MouseEvent.MOUSE_CLICKED, new
EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent mouseEvent) {
                Stage stage = (Stage) close_button.getScene().getWindow();
                stage.close();
            }
        });

        trey_button.addEventHandler(MouseEvent.MOUSE_CLICKED, new
EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent mouseEvent) {
                Stage stage = (Stage) trey_button.getScene().getWindow();
                stage.setIconified(true);
            }
        });

        offence_label.addEventHandler(MouseEvent.MOUSE_CLICKED, new
EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent mouseEvent) {
                if (!offence_label.getText().equals(String.valueOf(0))) {
                    FXMLLoader loader = new
FXMLLoader(getClass().getResource("/pr/traffict/carOffence.fxml"));
                    CarOffenceController controller = new CarOffenceController(car);
                    loader.setController(controller);
                    Stage window = new Stage();
                    window.initStyle(StageStyle.TRANSPARENT);
                    Parent root = null;
                    try {
                        root = loader.load();
```

```
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
                Scene scene = new Scene(root, 600, 370);
                scene.setFill(null);
                window.setTitle("Offence Info");
                window.setScene(scene);
                window.setResizable(false);
                window.initModality(Modality.APPLICATION_MODAL);
                window.show();
            }
        }
    });
}

private void setOffenceLabel() {
    int offenceCount = new DatabaseHandler().checkOffence(car);
    if (offenceCount != 0)
        offence_label.setStyle("-fx-text-fill: blue; -fx-underline: true");
    offence_label.setText(String.valueOf(offenceCount));
}
}

//Модуль CarOffenceController.java
package pr.traffict.controllers;

import javafx.collections.FXCollections;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import javafx.stage.Stage;
```

```
import pr.traffict.Car;
import pr.traffict.Offence;
import pr.traffict.database.DatabaseHandler;

import java.util.ArrayList;

public class CarOffenceController {
    private final Car car;
    @FXML
    private Button close_button;
    @FXML
    private Button trey_button;
    @FXML
    private TableView<Offence> tableView;
    @FXML
    private TableColumn<Offence, String> typeColumn;
    @FXML
    private TableColumn<Offence, String> carNumberColumn;
    @FXML
    private TableColumn<Offence, String> carModelColumn;
    @FXML
    private TableColumn<Offence, String> timeColumn;

    public CarOffenceController (Car car) {
        this.car = car;
    }

    @FXML
    public void initialize(){
        ArrayList<Offence> offencesList = new
DatabaseHandler().getOffencesByCarNumber(car);

        typeColumn.setCellValueFactory(new PropertyValueFactory<>("type"));
        carNumberColumn.setCellValueFactory(new PropertyValueFactory<>("carNumber"));
        carModelColumn.setCellValueFactory(new PropertyValueFactory<>("carModel"));
```

```
        timeColumn.setCellValueFactory(new PropertyValueFactory<>("time"));

        tableView.setItems(FXCollections.observableArrayList(offencesList));

        close_button.addEventHandler(MouseEvent.MOUSE_CLICKED,                new
EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent mouseEvent) {
                Stage stage = (Stage) close_button.getScene().getWindow();
                stage.close();
            }
        });

        trey_button.addEventHandler(MouseEvent.MOUSE_CLICKED,                new
EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent mouseEvent) {
                Stage stage = (Stage) trey_button.getScene().getWindow();
                stage.setIconified(true);
            }
        });
    }
}
```

ДОДАТОК Г. Ілюстративна частина (обов’язковий)

Розробка методів і програмних засобів для інтелектуального керування рухом транспорту

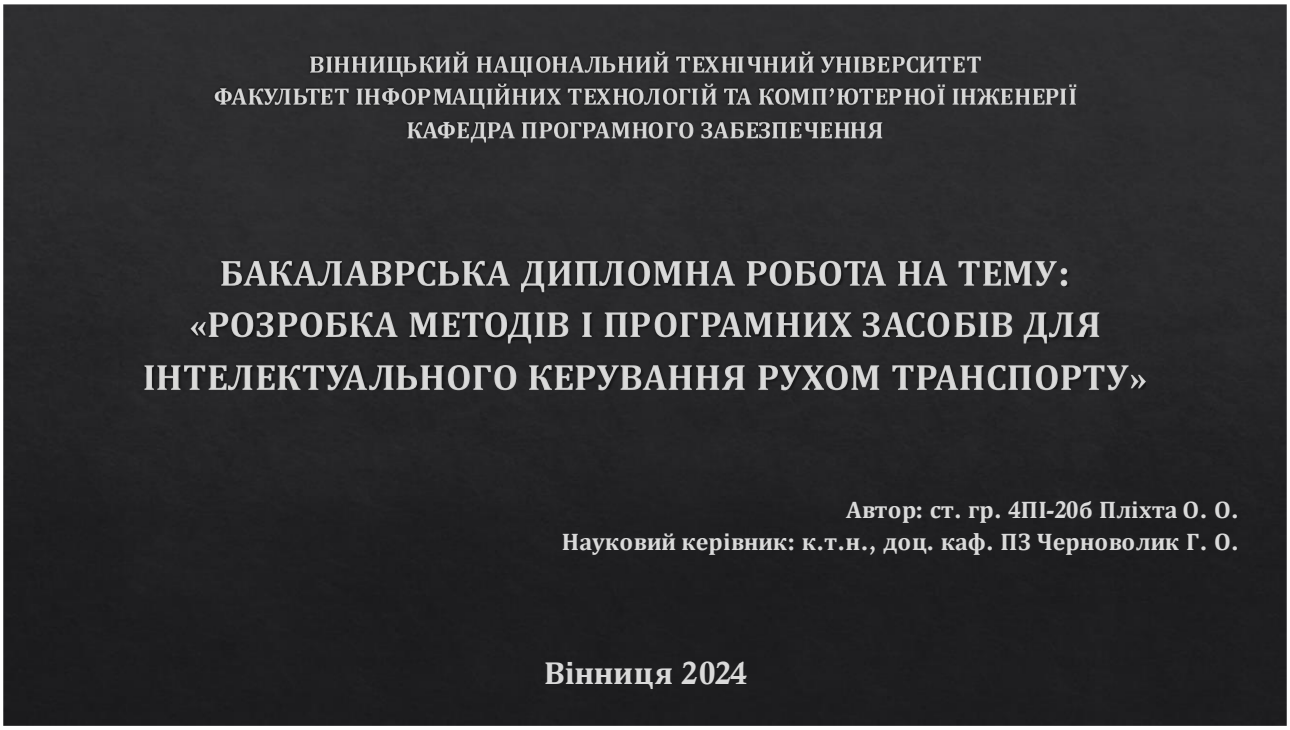


Рисунок Г.1 – Титульний слайд

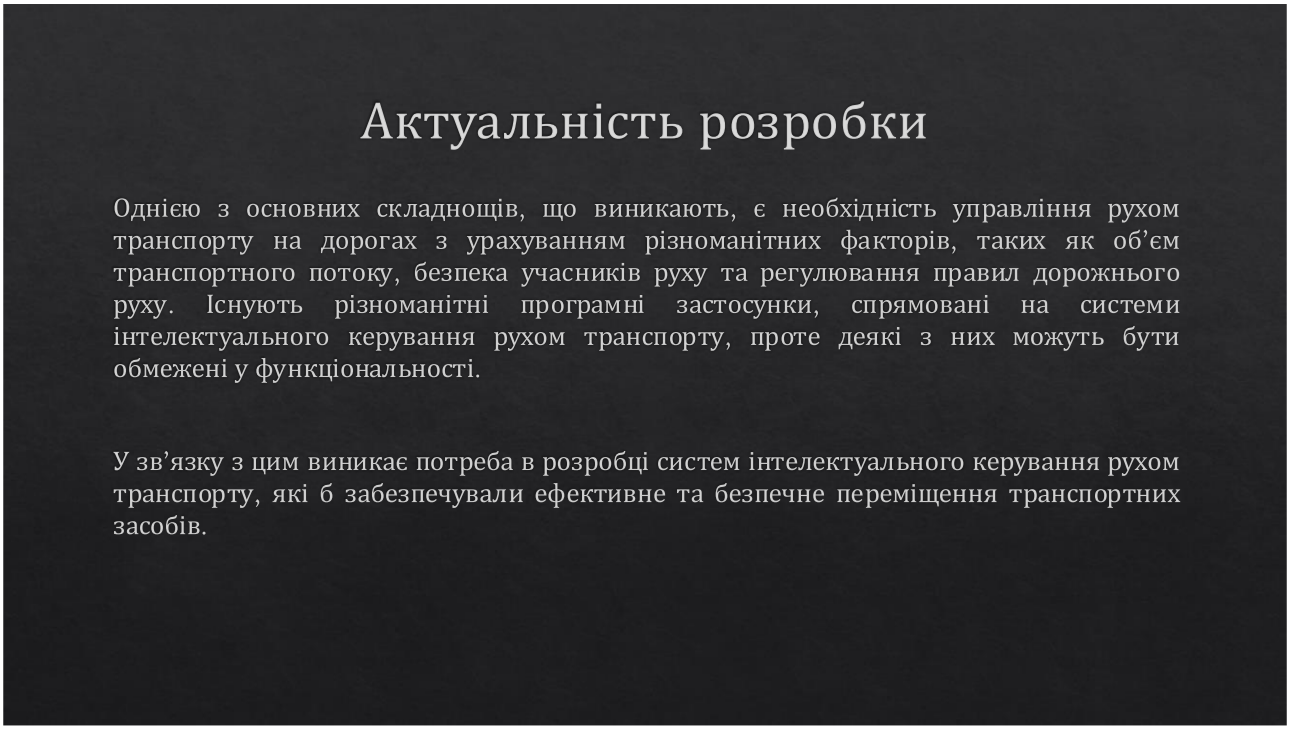


Рисунок Г.2 – Актуальність роботи

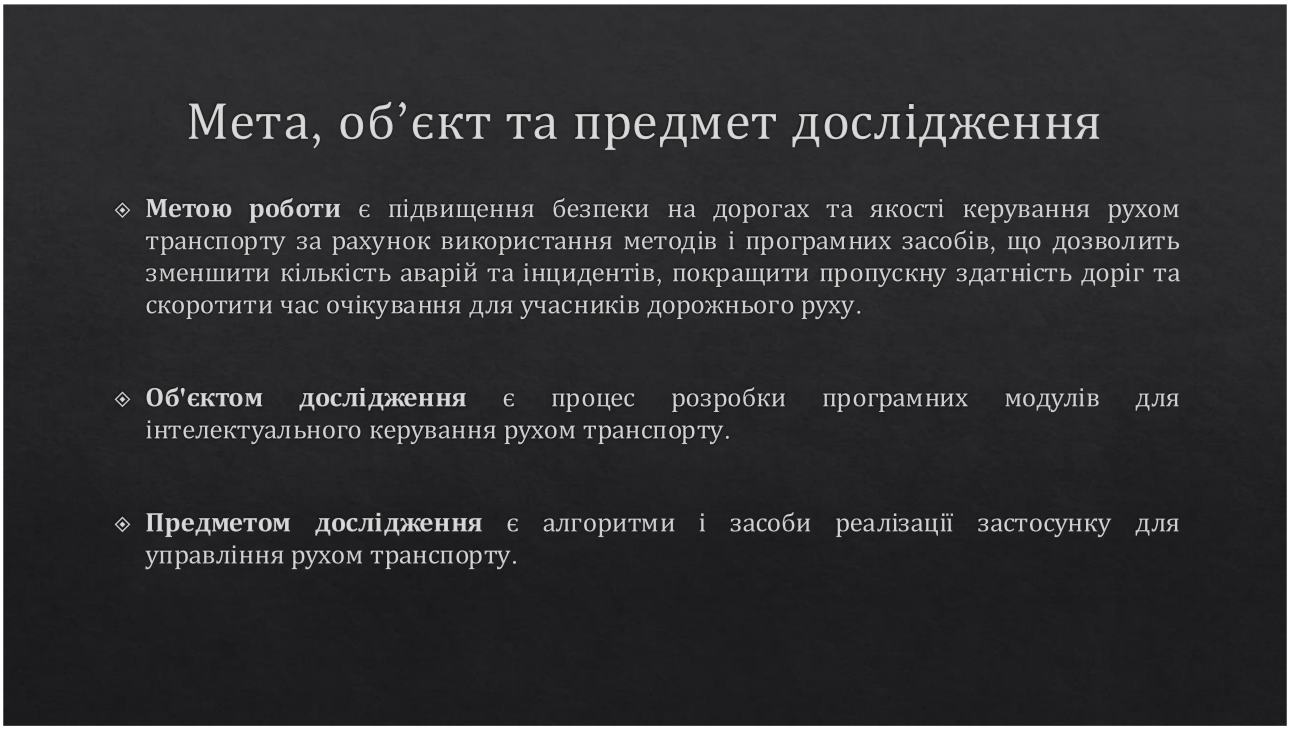


Рисунок Г.3 – Мета, об’єкт та предмет дослідження

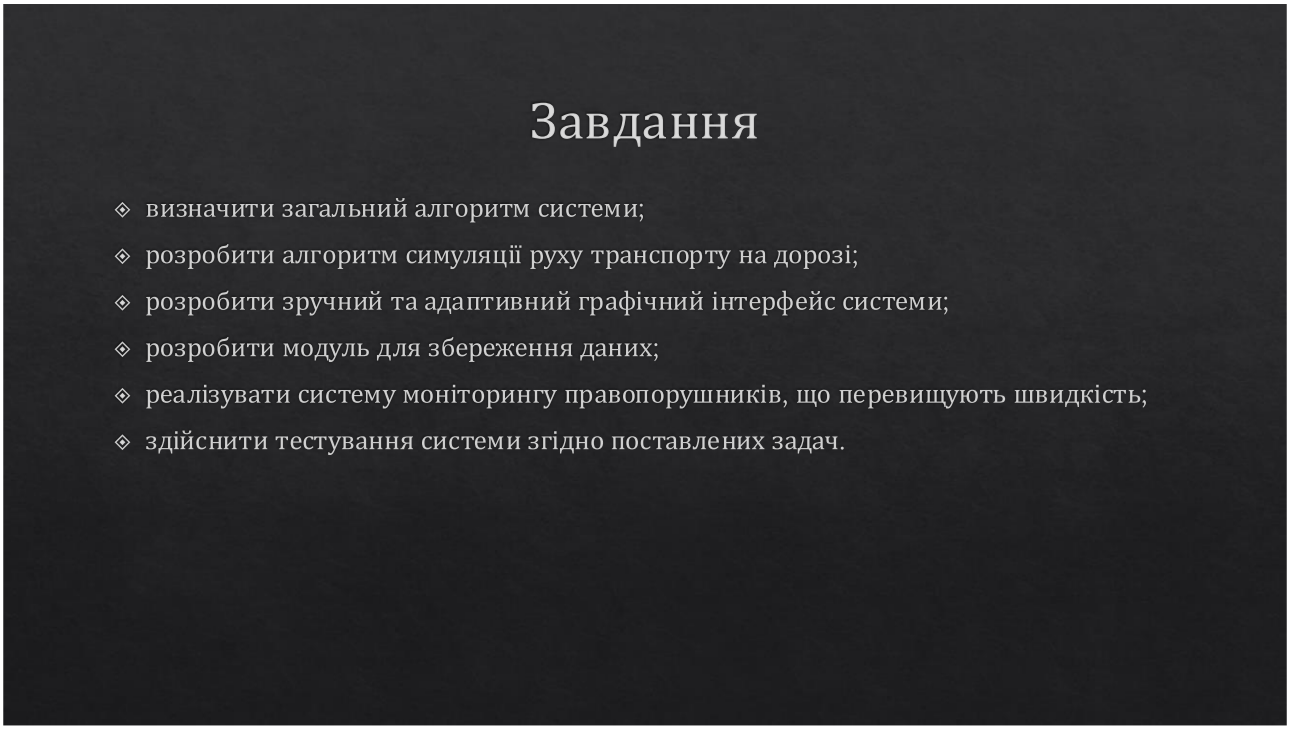


Рисунок Г.4 – Завдання

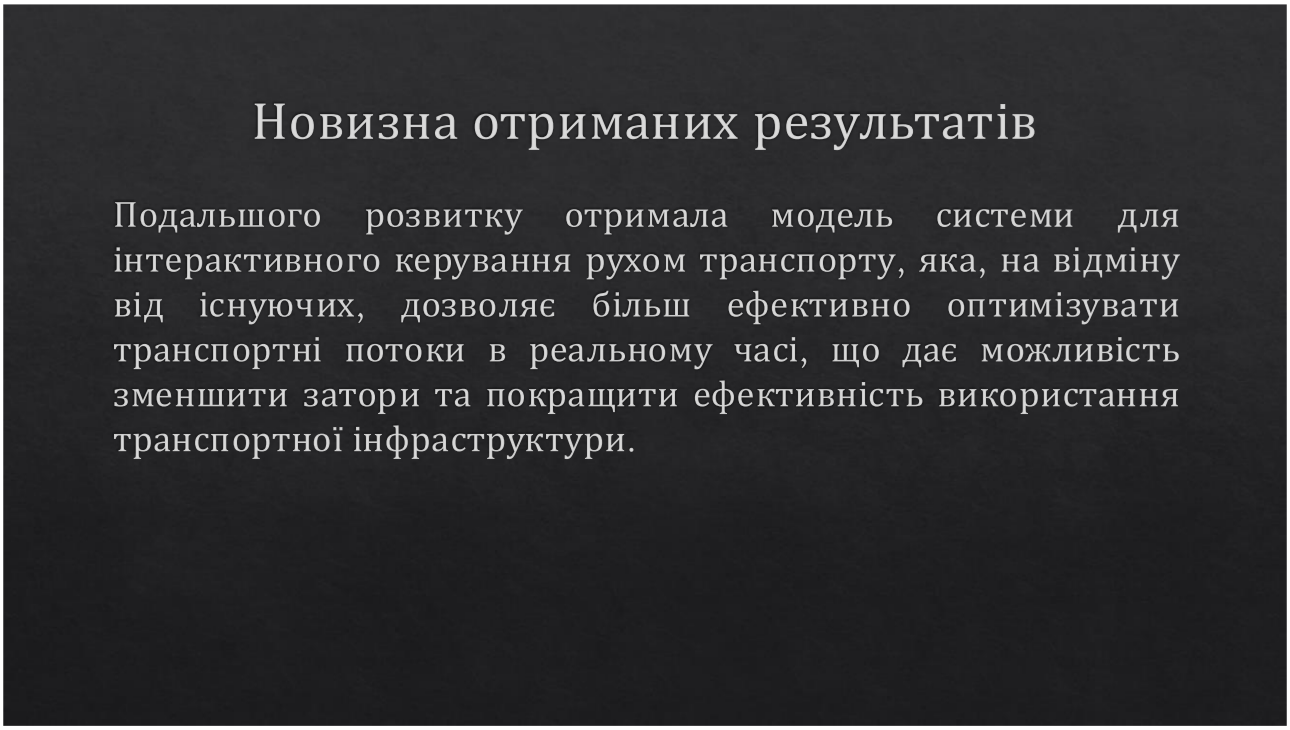


Рисунок Г.5 – Новизна отриманих результатів

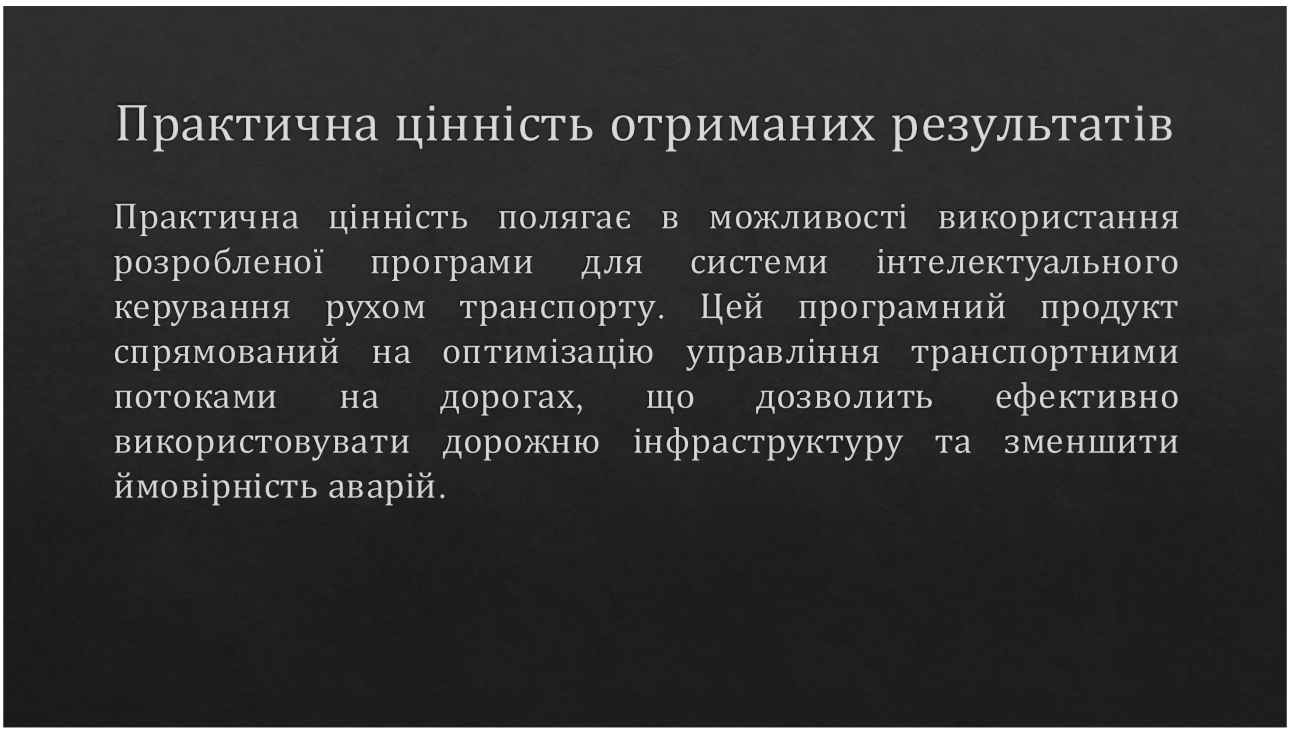


Рисунок Г.6 – Практична цінність отриманих результатів

Рисунок Г.7 – Аналіз аналогів (частина 1)

Рисунок Г.8 – Аналіз аналогів (частина 2)

Порівняльний аналіз аналогів

Критерії	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%

Рисунок Г.9 – Аналіз аналогів (частина 3)

Загальна модель системи

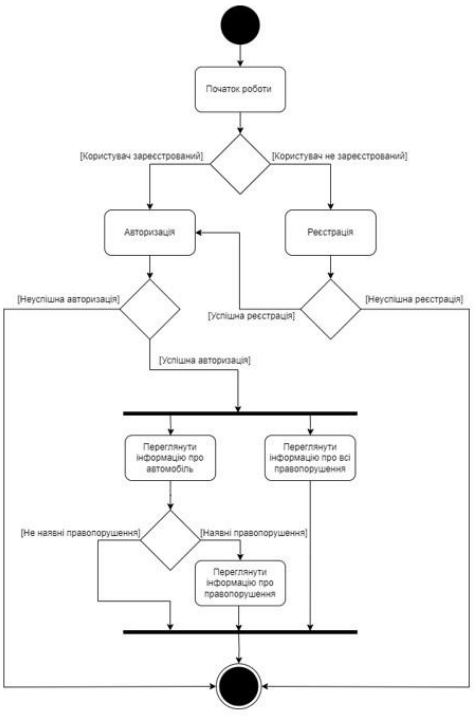
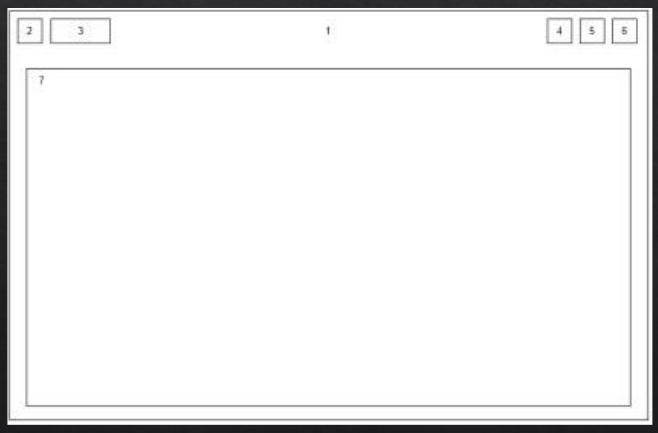


Рисунок Г.10 – Загальна модель системи

Розробка структури інтерфейсу



На схемі зображено:

- Вікно.
- Кнопка «Меню».
- Ім’я користувача.
- Кнопка «Переглянути список правопорушень».
- Кнопка «Згорнути»
- Кнопка «Закрити».
- Панель «Дорога».

Рисунок Г.11 – Розробка структури інтерфейсу

Розробка алгоритму автоматичного руху автомобіля

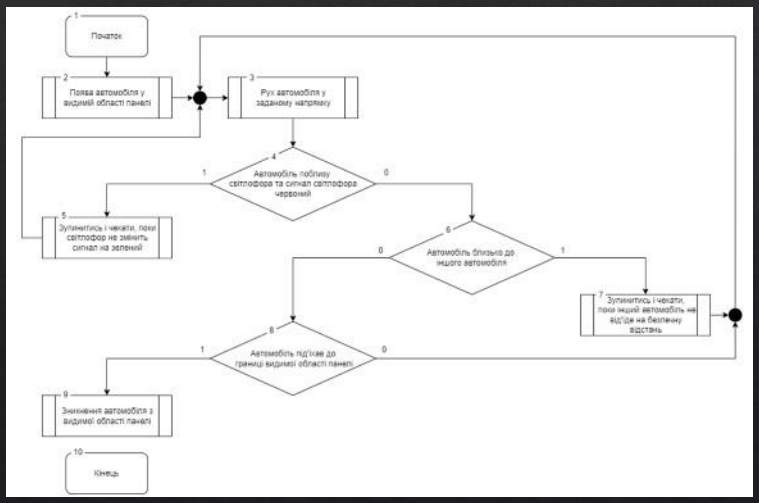


Рисунок Г.12 – Розробка алгоритму автоматичного руху автомобіля

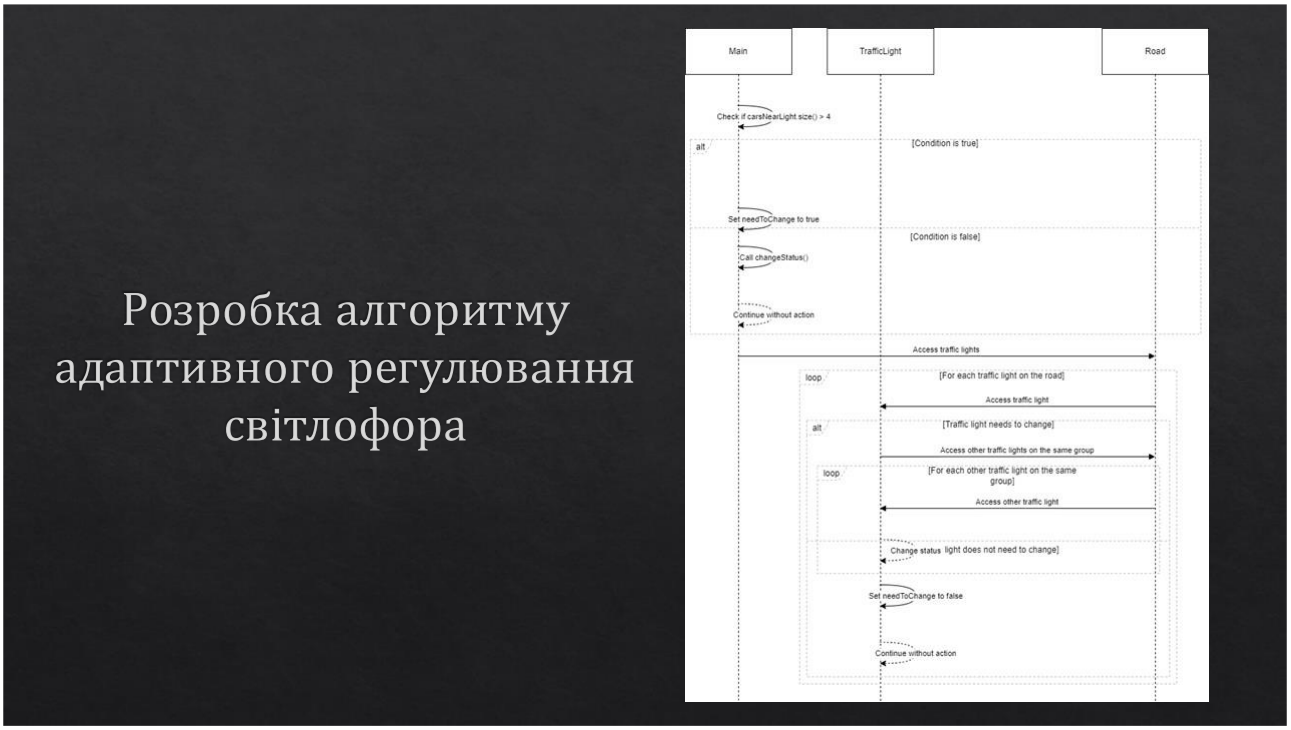


Рисунок Г.13 – Розробка алгоритму адаптивного регулювання світлофора

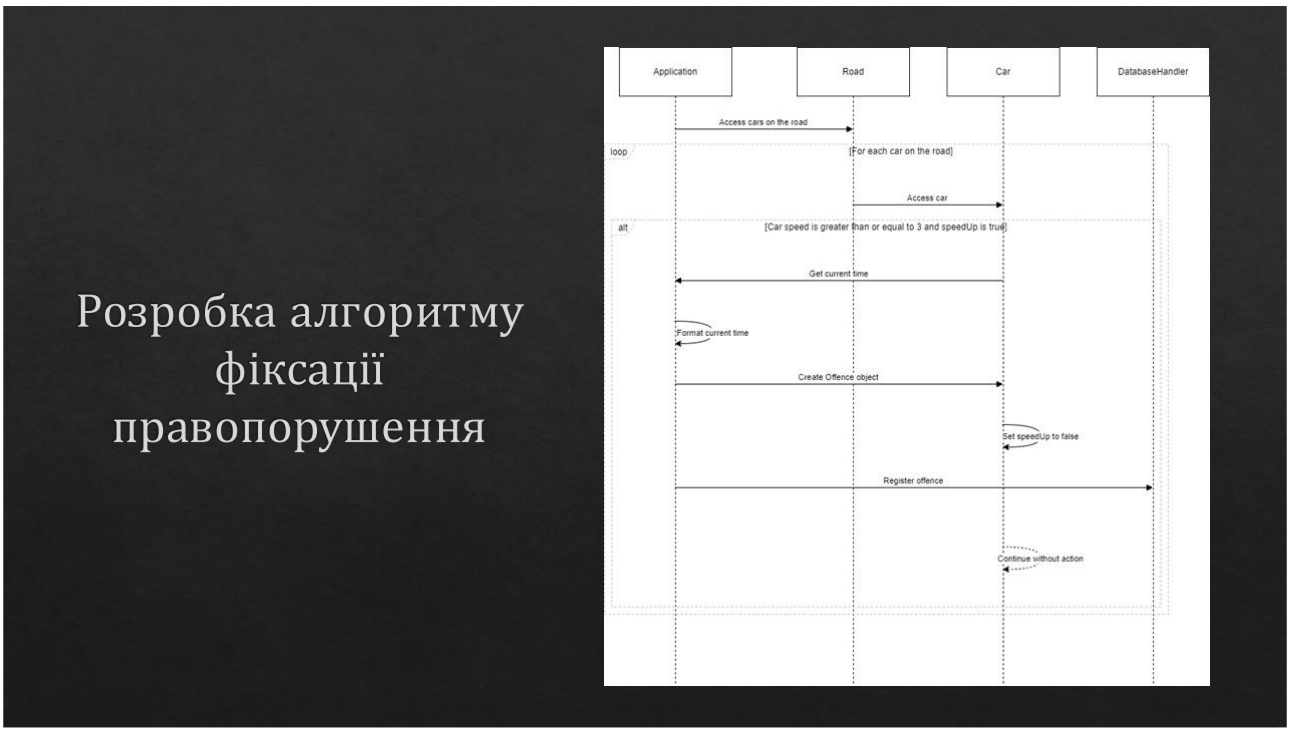


Рисунок Г.14 – Розробка алгоритму фіксації правопорушення

Рисунок Г.15 – Використані технології

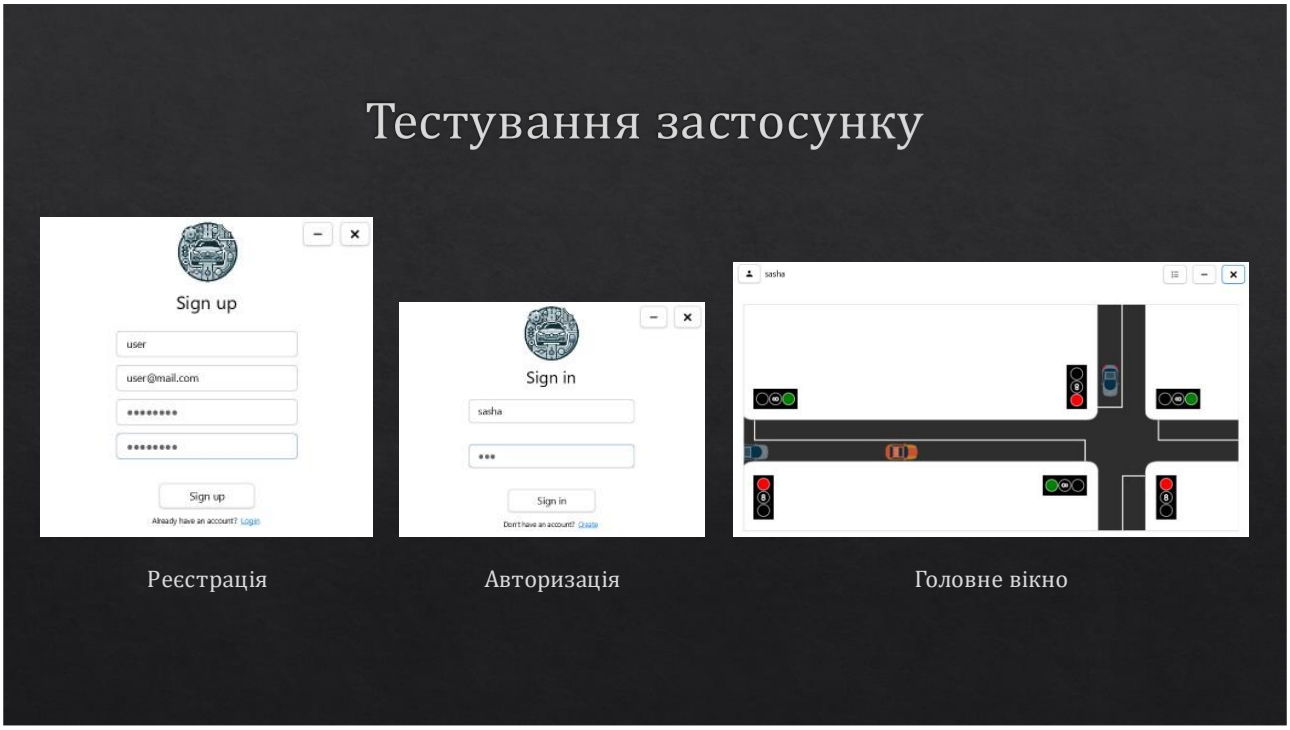
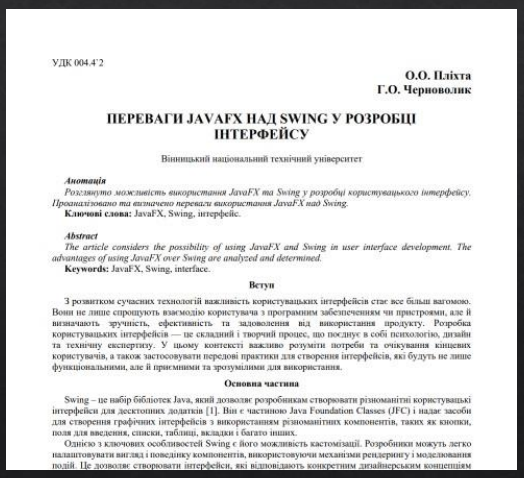


Рисунок Г.16 – Тестування застосунку

Апробація та публікація матеріалів роботи

Матеріали бакалаврської кваліфікаційної роботи доповідались на науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії «LIII Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2024)».



УДК 004.4'2

О.О. Пліхта
Г.О. Черноволик

ПЕРЕВАГИ JAVAFX НАД SWING У РОЗРОБЦІ ІНТЕРФЕЙСУ

Вінницький національний технічний університет

Анотація
Розглянуто можливість використання JavaFX та Swing у розробці користувацького інтерфейсу. Проаналізовано та визначено переваги використання JavaFX над Swing.
Ключові слова: JavaFX, Swing, інтерфейс.

Abstract
The article considers the possibility of using JavaFX and Swing in user interface development. The advantages of using JavaFX over Swing are analyzed and determined.
Keywords: JavaFX, Swing, interface.

Вступ

З розвитком сучасних технологій важливість користувацьких інтерфейсів стає все більш вагомою. Вони не лише спрощують взаємодію користувача з програмним забезпеченням чи пристроями, але й визначають зручність, ефективність та задоволення від використання продукту. Розробка користувацьких інтерфейсів — це складний і творчий процес, що поєднує в собі психологію, дизайн та технічну експертизу. У цьому контексті важливо розуміти потреби та очікування кінцевих користувачів, а також застосовувати передові практики для створення інтерфейсів, які будуть не лише функціональними, але й приємними та зрозумілими для використання.

Основна частина

Swing – це набір бібліотек Java, який дозволяє розробникам створювати різноманітні користувацькі інтерфейси для десктопних додатків [1]. Він є частиною Java Foundation Classes (JFC) і надає засоби для створення графічних інтерфейсів з використанням різноманітних компонентів, таких як кнопки, поля для введення, списки, таблиці, вкладки і багато інших.

Однією з ключових особливостей Swing є його можливість кастомізації. Розробники можуть легко налаштовувати вигляд і поведінку компонентів, використовуючи механізми рендерингу і моделювання подій. Це дозволяє створювати інтерфейси, які відповідають конкретним дизайнерським концепціям

Рисунок Г.17 – Апробація та публікація матеріалів роботи

Висновки

- Проаналізовано стан питання та виконано постановку задач.
- Розроблено структури та моделі системи.
- Розроблено алгоритмів роботи програмного застосунку.
- Проаналізовано та вибірано засоби для реалізації програми.
- Розроблено програмних модулів.
- Розроблено зручний та адаптивний графічний інтерфейс системи;
- Здійснено тестування роботи розробленого програмного застосунку.

Рисунок Г.18 – Висновки