ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ТЕХНОЛОГІЙ ЦИФРОВОГО РОЗВИТКУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення SafeDrive для інтерактивного вивчення правил дорожнього руху з елементами симуляції.»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Технології цифрового розвитку»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ігор НОВАЧУК

Виконав: здобувач(ка) вищої освіти групи ТЦР-42

Ігор НОВАЧУК

Керівник: Вадим ЧИТУЛЯН
ст. викладач кафедри

Рецензент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Технологій цифрового розвитку

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Технології цифрового розвитку»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Технологій цифрового розвитку
_____ Вікторія ЖЕБКА
« ____ » _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

1. Тема кваліфікаційної роботи: «Програмне забезпечення SafeDrive для інтерактивного вивчення правил дорожнього руху з елементами симуляції.»
керівник кваліфікаційної роботи старший викладач кафедри Вадим Читулян,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «_18_» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості правил дорожнього руху України, Польщі та Німеччини, державні стандарти оформлення знаків дорожнього руху та розмітки, методики інтерактивного навчання та імітаційного моделювання, функціональна документація до бібліотек PyQt6 та Pygame та системи керування базами данихSQLite

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1.Огляд та аналіз існуючих методів інтерактивного навчання й аналогів програмного забезпечення для вивчення правил дорожнього руху.

2.Проєктування архітектури, графічного інтерфейсу та реляційної бази даних десктопного застосунку «SafeDrive».

3.Програмна реалізація та опис функціонування інтерактивного комплексу «SafeDrive» з елементами 2D-симуляції.

4.Тестування, верифікація програмного забезпечення та оцінка його продуктивності.

5. Перелік графічного матеріалу: *презентація*

1. Мета, об’єкт та предмет дослідження.
2. Задачі кваліфікаційної роботи.
3. Аналіз аналогів.
4. Програмні засоби реалізації.
5. Діаграма послідовності.
6. Діаграма класів.
7. Діаграма варіантів використання.
8. Екранні форми
9. Відео-демонстрація роботи додатку.
10. Апробація результатів дослідження.
11. Висновки.

6. Дата видачі завдання 20.02.2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	21.02.26 – 28.02.26	
2	Аналіз існуючих аналогів	01.03.26 – 03.03.26	
3	Дослідження програмних засобів	04.03.26 – 13.03.26	
4	Моделювання об’єкту проектування	14.03.26 – 19.03.26	
5	Розробка функціоналу за стосунку	20.03.26 – 28.03.26	
6	Вступ, висновки, реферат	29.03.26 – 08.04.26	
7	Розробка презентації за стосунку	09.04.26 – 30.04.26	
8	Попередній захист роботи	12.05.26	

Здобувач(ка) вищої освіти _____ Ігор Новачук
(підпис)

Керівник кваліфікаційної роботи _____ Вадим Читулян
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: _51_ стор., _2_ табл., _11_ рис., _18_ джерел.

Мета роботи – підвищення ефективності та якості вивчення правил дорожнього руху шляхом створення інтерактивного програмного комплексу з можливостями імітаційного 2D-моделювання нестандартних дорожніх ситуацій

Об'єкт дослідження – вивчення правил дорожнього руху

Предмет дослідження – методи, алгоритми та програмне забезпечення для створення багатомодульного десктопного застосунку з використанням технологій Python, PyQt6 та Pygame.

Короткий зміст роботи: В роботі проаналізовано методи імітаційного моделювання та інтерактивного навчання в контексті підготовки водіїв. Проаналізовано існуючі рішення для вивчення ПДР такі як: GreenWay, Vodiy.ua, мобільні аналоги та автосимулятори. Розроблено архітектуру десктопного застосунку та програмно реалізовано ключові функції, зокрема: безпечну авторизацію, мультикраїнний довідник ПДР України, Польщі та Німеччини, векторний каталог знаків, автоматизоване тестування та 2D-симулятор водіння з 12 сценаріями критичних ситуацій. Проведено функціональне тестування та перевірку продуктивності. В роботі використано мову Python, PyQt6 для користувацького інтерфейсу, Pygame для моделювання фізики руху та SQLite для локального збереження даних.

Сферою використання застосунку є організація самостійної підготовки кандидатів у водії та навчальний процес в автошколах при вивченні ПДР.

КЛЮЧОВІ СЛОВА: ПДР, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ, PYTHON, PYQT6, PYGAME, АВТОСИМУЛЯТОР, ІНТЕРАКТИВНЕ НАВЧАННЯ

УМОВНІ ПОЗНАЧЕННЯ

БД - База даних

ДСТУ - Державний стандарт України

ДТП - Дорожньо-транспортна пригода

ООП - Об'єктно-орієнтоване програмування

ПДР - Правила дорожнього руху

ПЗ - Програмне забезпечення

СУБД - Система керування базами даних

ТЗ - Транспортний засіб

API - Прикладний програмний інтерфейс (Application Programming Interface)

CSS - Каскадні таблиці стилів (Cascading Style Sheets)

FPS - Кількість кадрів за секунду (Frames Per Second)

GUI - Графічний інтерфейс користувача (Graphical User Interface)

HTML - Мова розмітки гіпертексту (HyperText Markup Language)

IDE - Інтегроване середовище розробки (Integrated Development Environment)

JSON - Текстовий формат обміну даними (JavaScript Object Notation)

MVC - Архітектурний шаблон «Модель-Вигляд-Контролер» (Model-View-Controller)

NPC - Об'єкт, керований алгоритмом (Non-Player Character)

QSS - Механізм стилізації інтерфейсів у Qt (Qt Style Sheets)

SQL - Мова структурованих запитів (Structured Query Language)

SQLite - Компактна вбудована реляційна

UI - Інтерфейс користувача (User Interface)

UX - Досвід користувача (User Experience)

ЗМІСТ

ВСТУП

Актуальність теми. Стан безпеки дорожнього руху є одним із критичних показників соціально-економічного розвитку країни. В Україні процес підготовки водіїв тривалий час базувався на консервативних методиках пасивного засвоєння теоретичного матеріалу. Нещодавні зміни у нормативно-правовій базі створили нову парадигму в освітній галузі, що спричинило гострий дефіцит високотехнологічного програмного забезпечення. Традиційні рішення орієнтовані на механічне заучування, а відірваність теорії від практики при виникненні критичних ситуацій може мати фатальні наслідки. Додатковим викликом є євроінтеграція, що потребує знань ПДР різних країн ЄС. Таким чином створення повноцінного інтерактивного комплексу є дуже важливим питанням, що робить дану тему актуальною.

Мета роботи – підвищення ефективності та якості вивчення правил дорожнього руху шляхом створення інтерактивного програмного комплексу з можливостями імітаційного 2D-моделювання нестандартних дорожніх ситуацій.

Для досягнення мети, у бакалаврській роботі успішно виконано наступні завдання:

- проведення системного аналізу предметної області та існуючих аналогів
- проєктування модульної архітектури системи на засадах шаблону MVC
- розробка реляційної схеми бази даних SQLite для автономного збереження сесій
- реалізація функціональних модулів для векторного малювання дорожніх знаків
- програмування інтерактивного симулятора на базі рушія Pygame
- аналіз результатів впровадження моделі симулятора та тестування системи

Об'єкт дослідження - процес автоматизованого навчання та перевірки знань Правил дорожнього руху.

Предмет дослідження - архітектурні рішення, алгоритми імітаційного моделювання та методи програмної реалізації десктопного комплексу.

Методи дослідження. Під час написання бакалаврської кваліфікаційної роботи були використані методи теоретичного дослідження, імітаційного моделювання, принципи об'єктно-орієнтованого проєктування та методи функціонального тестування.

Наукова новизна одержаних результатів. У ході дослідження описано новий підхід для самостійної підготовки водіїв на основі комбінації алгоритмів векторного рендерингу та 2D-симуляції фізики руху в єдиному автономному середовищі. Результат застосування яких показує високу ефективність засвоєння практичних навичок.

Практична значущість одержаних результатів. Запропонована модель забезпечує ефективне рішення для безпечного відпрацювання дорожніх сценаріїв та повної автономності роботи користувачів на рівні десктопного додатка.

Апробація результатів та публікації. Основні положення і результати бакалаврської роботи доповідались на науково-практичних конференціях, що проходили на базі університету.

- Новачук І.С. “Розроблення програмного забезпечення Safe Drive для інтерактивного вивчення правил дорожнього руху “ // Всеукраїнська науковотехнічна конференція “Застосування програмного забезпечення в інформаційно-комунікаційних технологіях”. Збірник тез. – К.: ДУІКТ, 2026.
- Новачук І.С. “Розроблення програмного забезпечення Safe Drive для інтерактивного вивчення правил дорожнього руху з елементами симуляції” // Всеукраїнська науково-технічна конференція “Технології цифрового розвитку: програмні системи та децентралізація”. Збірник тез. – К.: ДУІКТ, 2026.

1. АНАЛІЗ ТА ХАРАКТЕРИСТИКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1Теоретичні аспекти побудови інтерактивних систем навчання та імітаційного моделювання

Трансформація освітнього процесу в умовах глобальної цифровізації передбачає перехід від репродуктивних методів передачі знань до інтерактивних технологій, що забезпечують активне залучення користувача до середовища навчання. В інженерії програмного забезпечення під інтерактивною системою навчання розуміють комплекс апаратних та програмних засобів, які забезпечують двосторонню взаємодію між користувачем та системою, де дії користувача викликають миттєву відповідну реакцію програмного середовища.

Традиційна методика підготовки водіїв, що базується на читанні статичних текстів ПДР та пасивному перегляді ілюстрацій, має суттєвий недолік - відсутність формування динамічних образів дорожніх ситуацій. З погляду когнітивної психології, процес водіння є складним психомоторним актом, який потребує не лише знання правил, а й здатності до швидкого аналізу просторових відношень об’єктів у часі.

Теорія подвійного кодування стверджує, що інформація засвоюється ефективніше, якщо вона подається одночасно у текстовій та візуальній формах. У моєму проєкті це реалізовано через поєднання HTML-рендерингу теоретичних розділів з активною CSS-стилізацією та векторним малюванням дорожніх знаків за допомогою графічної бібліотеки QPainter. Такий підхід дозволяє створювати ієрархічні структури знань, де критично важливі елементи виділяються візуально, що прискорює їх ідентифікацію водієм.

Це метод навчання, за якого створюється спрощена модель реальності, або її окремих аспектів, що дозволяє учневі проводити експерименти та відпрацьовувати

навички без ризику для життя та майна. У контексті вивчення ПДР імітаційне моделювання є критично важливим для розуміння таких фізичних параметрів, як:

1. Інерція та гальмівний шлях ТЗ на різних типах покриття
2. Зміна дистанції та інтервалу в умовах обмеженої видимості
3. Механіка виникнення заносів та аквапланування при перевищенні безпечної швидкості.

Використання ігрового рушія Pygame дозволяє програмно реалізувати ці фізичні закони, перетворюючи “сухий” пункт правил на особистий досвід користувача у віртуальному 2D-просторі.

Впровадження елементів гри у ненавчальний контекст дозволяє підвищити мотивацію користувача. У системі SafeDrive це реалізовано через систему нарахування балів у тестах, візуалізацію прогресу та логування результатів симуляцій через реляційну базу даних SQLite. Важливим аспектом є механізм негайного розбору помилок, коли система не просто констатує факт неправильної відповіді, а надає обґрунтоване пояснення, що замикає цикл навчання.

Згідно з актуальними змінами в українському законодавстві, можливість самостійної підготовки водіїв вимагає від програмного забезпечення високого ступеня автономності та надійності. Теоретична база ПЗ має бути повною та структурованою за принципом “від простого до складного”. Саме тому архітектура SafeDrive передбачає 24 розділи теорії, що охоплюють усі аспекти ПДР, включаючи специфічні розділи про права та обов’язки учасників дорожнього руху.

Незважаючи на популярність мобільних додатків, розробка повноцінного навчального комплексу з елементами фізичної симуляції потребує значних обчислювальних ресурсів та великої площі візуалізації для коректного відображення складних дорожніх розв'язок. Десктопний формат дозволяє реалізувати плавний ігровий цикл з продуктивністб 60 FPS та використовувати векторну графіку високої чіткості, що мінімізує втому очей при тривалому навчанні.

1.2Огляд та порівняльний аналіз існуючих програмних рішень

Для обґрунтування технічної доцільності розробки програмного забезпечення “SafeDrive” необхідно провести системний аналіз наявних на ринку України інструментів для вивчення ПДР. Сучасний сегмент навчального ПЗ представлений трьома основними класами систем: інформаційні веб-портали, мобільні додатки та імітаційні тренажери. Кожен із цих класів має свої функціональні особливості та технічні обмеження.

1.2.1 Спеціалізований веб-портал “Vodiy.ua”

Це один із найбільш затребуваних ресурсів у сегменті браузерного навчання.

Функціональний склад: система надає користувачеві доступ до структурованого тексту ПДР України, ілюстрованих коментарів фахівців та бази тестових питань, адаптованих під іспити в ГСЦ МВС.

Технічний аналіз: портал побудований за клієнт-серверною архітектурою з акцентом на відображення статичного контенту. Основним недоліком є відсутність засобів динамічного моделювання ситуацій. Користувач сприймає інформацію через статичні малюнки, що не дозволяє оцінити швидкість руху об’єктів або зміну ракурсу огляду водія. Крім того, робота з ресурсом неможлива без стабільного інтернет-з’єднання, а відсутність локальної бази даних не дозволяє повноцінно зберігати персональну статистику в офлайн-режимі. Такий підхід здебільшого задовольняє базові потреби для механічного завчення білетів, однак виключно текстово-графічний формат має суттєві обмеження щодо ефективного утримання уваги учня. Отже, ресурс ефективно виконує роль електронного довідника, але не здатний сформувати у кандидата у водії просторове мислення та розуміння фізики руху транспортного засобу.

Загальний вигляд головної сторінки в режимі проходження тестів представлено на рисунку 1.1

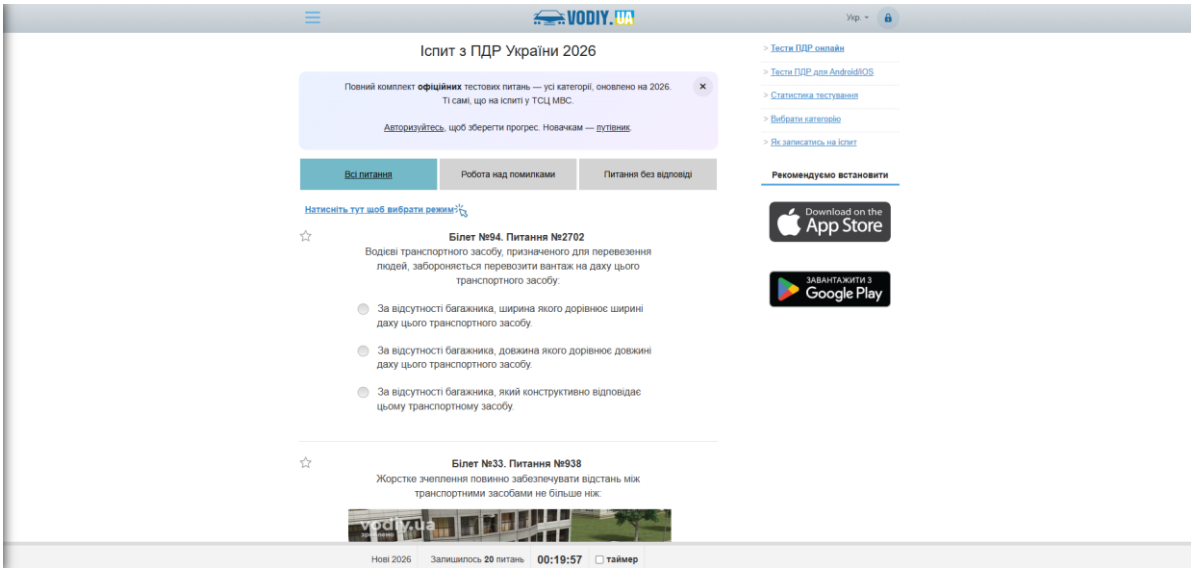


Рис. 1.1 Головне вікно сайту Vodiy.ua

1.2.2 Навчальний комплекс “GreenWay”

Система GreenWay позиціонується як професійне рішення для учнів які хочуть самостійно вивчити матеріал та автошколи.

Функціональний склад: платформа пропонує відео-лекції, розширені пояснення до тестів та систему “розумного” повторення тем, де користувач припустився помилок.

Технічний аналіз: використовується модель SaaS де повний доступ до функціоналу є платним. З погляду імітаційного моделювання, система обмежується переглядом пре-рендереного відеоконтенту. На відміну від SafeDrive, де реалізовано повноцінний фізичний рушій на базі Pygame, користувач GreenWay не може самостійно керувати об’єктом, відпрацьовувати гальмування чи маневрування у критичних сценаріях, що унеможливлює формування м’язової пам’яті.

Відображення інтерфейсу головного екрана з доступом до теоретичних та практичних модулів представлено на рисунку 1.2

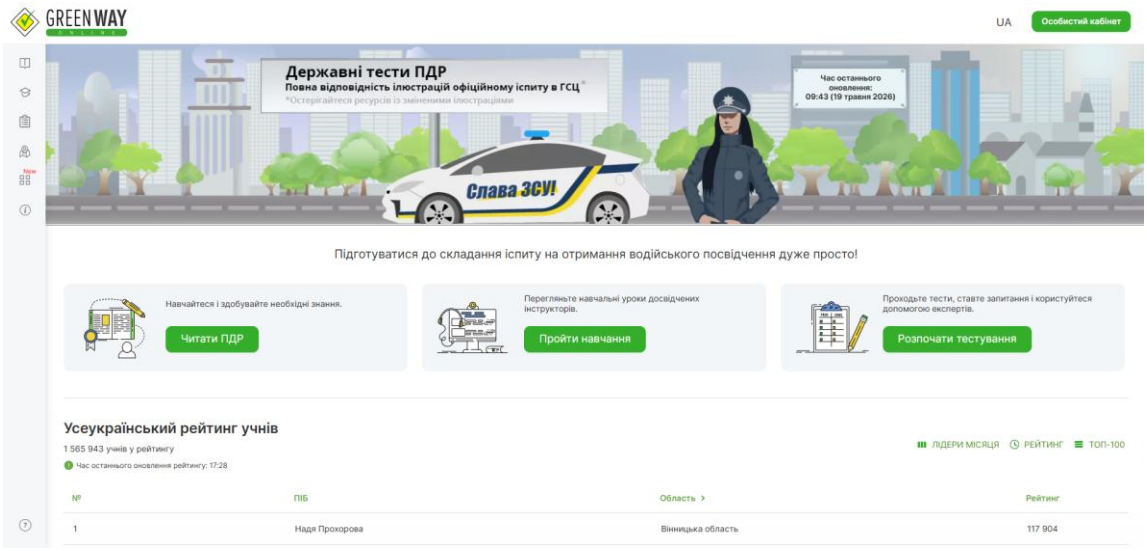


Рис.1.2 Головне вікно сайту GreenWay

1.2.3 Мобільний додаток “ПДР України 2026: Офіційні тести”

Цей тип ПЗ є найбільш масовим завдяки зручності використання на смартфонах.

Функціональний склад: додаток дозволяє проходити тести за темами та переглядати каталог знаків у режимі офлайн.

Технічний аналіз: критичним обмеженням мобільних аналогів є якість графічної підсистеми. Через потребу економії пам’яті пристрою більшість знаків та схем перехресть подані у растрових форматах низької роздільної здатності. У розробленому ПЗ SafeDrive цю проблему вирішено впровадженням векторного рендерингу через бібліотеку QPainter, що забезпечує ідеальну чіткість зображень незалежно від роздільної здатності монітора. Також такі додатки не підтримують динамічне перемикання між правилами різних країн (наприклад, Україна та Польща) у єдиному вікні, що є унікальною перевагою нашого проєкту. Отже ресурс надає всі можливості студенту для вивчення ПДР та тестування знать на основі виченої програми, але не надає розуміння фізики руху транспорту та поведінки на дорозі.

Візуалізація структури розділів тестування та панелі відстеження прогресу користувача представлена на рисунку 1.3

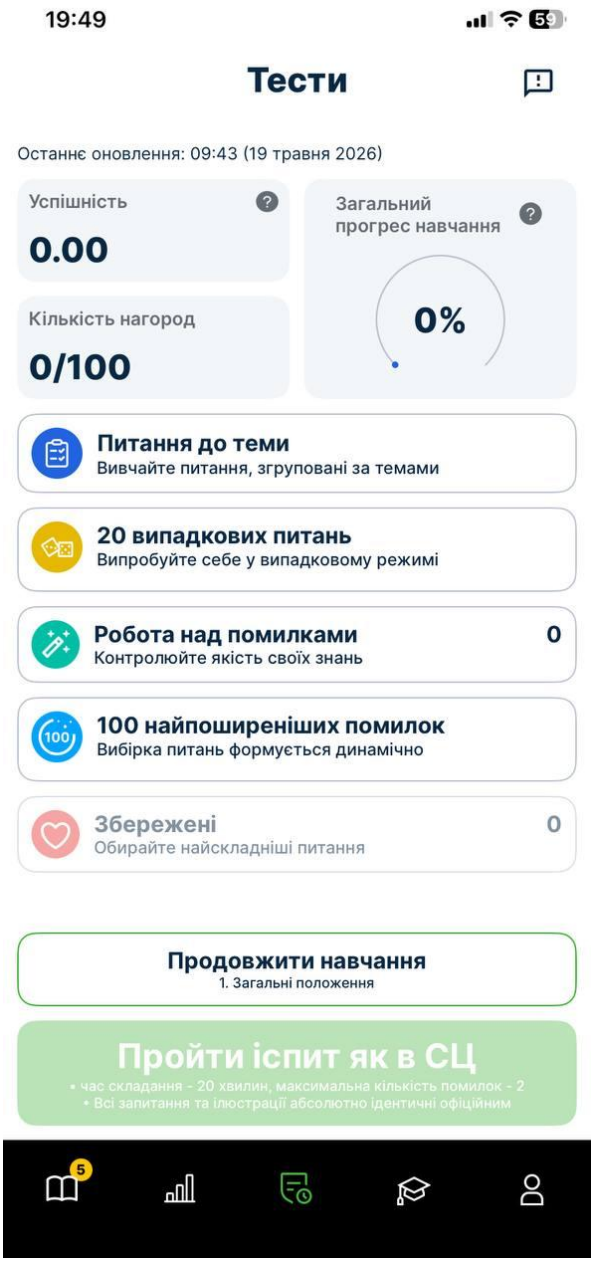


Рис 1.3 Головне меню моб. застосунку ПДР України 2026: Офіційні тести

1.2.4 Компаративний аналіз технічних характеристик аналогів та SafeDrive

Для системної оцінки технічного рівня розробленого ПЗ порівняно з існуючими рішеннями було розроблено порівняльну таблицю за дев'ятьма інженерними критеріями.

Порівняння програм аналогів з програмою SafeDrive зведені у таблиці 1.1

Таблиця 1.1

Техніко-функціональне порівняння SafeDrive з аналогами

Критерій порівняння	Green Way	Vodiy.ua	Моб. застосунки	3D-симулятори	Розроблена платформа
Теоретична база	Наявна	Наявна	Наявна	Відсутня	Наявна
Модуль тестування	Наявний	Наявний	Наявний	Відсутній	Наявний
Відстеження прогресу та статистика	Наявне	Наявне	Наявне	Відсутнє	Наявне
Інтерактивна симуляція ситуацій	Відсутня	Відсутня	Відсутня	Наявна	Наявна
Автономна робота	Відсутня	Відсутня	Часткова	Повна	Повна
Підтримка ПДР кількох країн	Відсутня	Відсутня	Відсутня	Відсутня	Наявна
Системні вимоги до пристрою	Низькі	Низькі	Не застосовується	Високі	Низькі

1.2.5 Виявлені проблеми та обґрунтування необхідності розробки

Проведений аналіз дозволив виявити суттєвий “технологічний розрив” на ринку навчального ПЗ, це існуючі системи орієнтовані на кількісне опрацювання тестів, але ігнорують якісну підготовку водія до реальних фізичних процесів на дорозі.

Розробка “SafeDrive” покликана усунути ці недоліки шляхом:

1. Впровадження імітаційного рушія - використання Pygame дозволяє програмно реалізувати математичну модель руху, що дає користувачеві досвід керування ТЗ у безпечних віртуальних умовах.
2. Забезпечення автономності та безпеки - використання SQLite та механізмів авторизації дозволяє учневі вести власний освітній профіль без ризику витоку даних чи залежності від хмарних серверів.
3. Міжнародної адаптивності - розроблений модуль country_module забезпечує швидку підготовку водіїв до специфічних вимог доріг Європи, що є надзвичайно актуальним в умовах посиленої інтеграції України в ЄС.

Таким чином, SafeDrive є технічно досконалим та соціально актуальним рішенням, яке забезпечує якісно новий рівень самостійної водійської освіти.

1.3Обґрунтування методики розробки та етапів дослідження

Процес створення інтерактивної системи “SafeDrive” базується на принципах системного інженерного підходу та ітераційній моделі життєвого циклу розробки програмного забезпечення. Для забезпечення високої якості продукту та відповідності поставленим вимогам методика дослідження та розробки розподілена на декілька взаємопов’язаних етапів.

1.3.1 Системний аналіз та збір вимог

На початковому етапі застосовується метод дескриптивного аналізу предметної галузі. Це включає вивчення нормативної бази ПДР України та країн ЄС, а також аналіз ДСТУ щодо візуального представлення дорожніх знаків та розмітки. Результатом етапу є формування функціональних вимог та нефункціональних вимог.

1.3.2 Архітектурне моделювання та проектування

Методологічною основою проектування системи обрано об’єктно-орієнтований аналіз та дизайн. Для візуалізації структури та логіки взаємодії компонентів використовується уніфікована мова моделювання UML. В межах роботи передбачено побудову таких діаграм:

1. Use Case Diagram для визначення ролей користувача та функціональних можливостей системи.
2. Class Diagram для опису структури модулів та їхніх взаємозв’язків за патерном MVC.
3. ER-діаграма для проектування реляційної схеми бази даних SQLite.

1.3.3 Методика реалізації імітаційного моделювання

Для розробки модуля симуляції використовується методика математичного моделювання фізичних процесів у 2D-просторі. Це передбачає:

-програмну реалізацію векторів сили для обчислення прискорення та гальмування ТЗ

-алгоритмічну обробку колізій на основі геометричних примітивів для фіксації ДТП

-застосування тригонометричних функцій для розрахунку кутів повороту та траєкторії руху об'єктів у реальному часі

1.3.4 Методика управління даними та безпекою

З метою забезпечення автономності системи застосовується методика локального реляційного зберігання. Для захисту персональних даних користувачів впроваджується алгоритм криптографічного хешування паролів, що виключає зберігання конфіденційної інформації у відкритому вигляді. Керування сесіями реалізується через механізм кешування даних у форматі JSON.

1.3.5 Верифікація та тестування

Для підтвердження працездатності ПЗ “SafeDrive” використовується комбінована методика тестування:

1. Модульне тестування - для перевірки окремих функцій розрахунку штрафів та коректності SQL-запитів.
2. Інтеграційне тестування - для перевірки взаємодії графічного інтерфейсу PyQt6 з фізичним рушієм Pygame.
3. UX-тестування - для оцінки зручності інтерфейсу та зрозумілості навігації при самостійному навчанні водія.

1.4Постановка задач дипломного проєктування

Проведений аналіз предметної області та існуючих засобів навчання ПДР підтвердив, що ринок потребує комплексного, автономного рішення, яке поєднує теоретичну підготовку з імітаційним моделюванням фізики руху. Виявлені недоліки аналогів, такі як залежність від мережі Інтернет та відсутність динамічних сценаріїв підготовки, зумовлюють необхідність розробки системи “SafeDrive”.

Метою роботи є розробка кросплатформеного програмного забезпечення для інтерактивного вивчення Правил дорожнього руху з вбудованим 2D-симулятором критичних ситуацій та підтримкою законодавчих баз України та країн ЄС.

Для досягнення поставленої мети та забезпечення високого технічного рівня системи в межах дипломного проєктування необхідно вирішити такий перелік науково-технічних задач:

Обґрунтування технологічного стеку: провести порівняльний аналіз інструментальних засобів розробки для забезпечення оптимальної продуктивності графічної та фізичної підсистем.

Проєктування архітектури ПЗ: розробити модульну структуру застосунку на базі архітектурного шаблону MVC для розділення логіки обробки даних, імітаційного моделювання та графічного інтерфейсу користувача.

Розробка бази даних: спроєктувати реляційну схему в СУБД SQLite для забезпечення автономного зберігання профілів, результатів тестування та логів успішності користувача.

Реалізація модуля векторної графіки: розробити алгоритми динамічного малювання дорожніх знаків за допомогою засобів бібліотеки QPainter для забезпечення ідеальної чіткості об'єктів при масштабуванні.

Програмування фізично коректного симулятора: реалізувати на базі рушія Pygame математичну модель руху ТЗ, що враховує інерцію, гальмівний шлях та умови колізій у 12 критичних сценаріях.

Впровадження системи мульти-країнності: розробити механізм динамічної адаптації контенту ПДР та стилізації інтерфейсу для підтримки правил України, Польщі та Німеччини.

Верифікація та апробація: виконати комплексне тестування функціональних модулів на відповідність вимогам ПДР та оцінити зручність UI/UX для самостійного навчання.

Виконання вищезазначених задач дозволить реалізувати надійну, функціональну та автономну систему, яка забезпечить якісно новий рівень підготовки майбутніх водіїв. Таким чином, результати проведеного аналізу предметної області та існуючих аналогів підтверджують технічну доцільність та соціальну актуальність розробки програмного комплексу SafeDrive, що стає основою для подальшого проєктування та програмної реалізації системи у наступних розділах роботи.

Крім того, розв'язання поставлених задач має на меті створення не просто дослідницького концепту, а повноцінного мінімально життєздатного продукту готового до впровадження у реальний освітній процес. Цільовою аудиторією розробленого рішення стануть як слухачі ліцензованих автошкіл, що потребують додаткового інтерактивного інструменту для закріплення матеріалу, так і особи, які, згідно з оновленим законодавством, обрали шлях повністю самостійної підготовки до складання теоретичного іспиту в територіальних Сервісних центрах МВС.

Важливим інженерним аспектом постановки задачі є також чітке окреслення меж проєкту. Враховуючи суворі вимоги до загальної доступності, кросплатформності та швидкодії на комп'ютерах початкового рівня, програмний комплекс проєктується принципово без використання технологій віртуальної реальності або важкоатлетних 3D-рушіїв. Натомість основний акцент робиться на бездоганній точності двовимірної симуляції, плавності роботи користувацького інтерфейсу та алгоритмічній надійності модуля перевірки знань. Таке свідоме архітектурне обмеження дозволяє ефективно зосередити всі ресурси розробки безпосередньо на якості навчальної бази та математичному відпрацюванні логіки дорожнього руху.

Реалізація перелічених завдань також вимагає особливої уваги до оптимізації програмного коду та раціонального використання системних ресурсів. Оскільки цільовою платформою, як зазначалося раніше, є комп'ютери початкового рівня, розроблений продукт повинен ефективно управляти оперативною пам'яттю під час завантаження графічних асетів та безперервної роботи фізичного рушія симулятора. Крім того, внутрішня архітектура локальної бази даних має бути спроєктована таким чином, щоб забезпечити миттєвий пошук потрібних теоретичних розділів, швидку генерацію випадкових тестових питань та збереження результатів без жодних затримок у роботі користувацького інтерфейсу.

Виконання цих умов дозволить створити не просто базовий навчальний тренажер, а надійний програмний інструмент, який повністю відповідає сучасним стандартам розробки автономних десктопних застосунків.

Окремим завданням, що логічно випливає із загальної мети проєкту, є забезпечення високого рівня супроводжуваності та масштабованості програмного коду. Оскільки правила дорожнього руху та екзаменаційні білети мають властивість періодично оновлюватися на законодавчому рівні, архітектура застосунку повинна передбачати можливість швидкої інтеграції нових або модифікації існуючих даних.

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА МОДЕЛЮВАННЯ СИСТЕМИ SAFEDRIVE

2.1 Аналіз та формування вимог до програмної системи

Процес проєктування системи “SafeDrive” починається з чіткого визначення вимог, які базуються на результатах аналізу предметної області, проведеного у першому розділі. Враховуючи специфіку освітнього ПЗ з елементами фізичного моделювання, вимоги поділяються на функціональні та нефункціональні.

Функціональний аспект розробки передбачає чітку ієрархію дій, де доступ до розширених можливостей, таких як ведення статистики успішності або запуск імітаційного симулятора, пов'язаний із процесом авторизації. Це дозволяє забезпечити персоналізацію навчання та автономне збереження результатів кожного сеансу роботи у локальній базі даних. Такий підхід до моделювання прецедентів дозволяє структурувати архітектуру програми та визначити межі взаємодії користувача з окремими програмними модулями.

Зокрема, графічне представлення акторів та їхніх взаємозв'язків із прецедентами відображає гнучкість розробленої архітектури. Окреслені межі системи дозволяють мінімізувати ризики логічних помилок на етапі написання коду, забезпечуючи взаємозв'язок між теоретичною базою та практичним симулятором. Це закладає надійну інженерну основу для подальшого структурного моделювання всіх компонентів програмного забезпечення.

Для чіткого визначення функціональних можливостей системи “SafeDrive” та ролей користувачів було розроблено діаграму варіантів використання. Основні взаємодії користувача з модулями теорії, тестування та симуляції представлені на рисунку 2.1

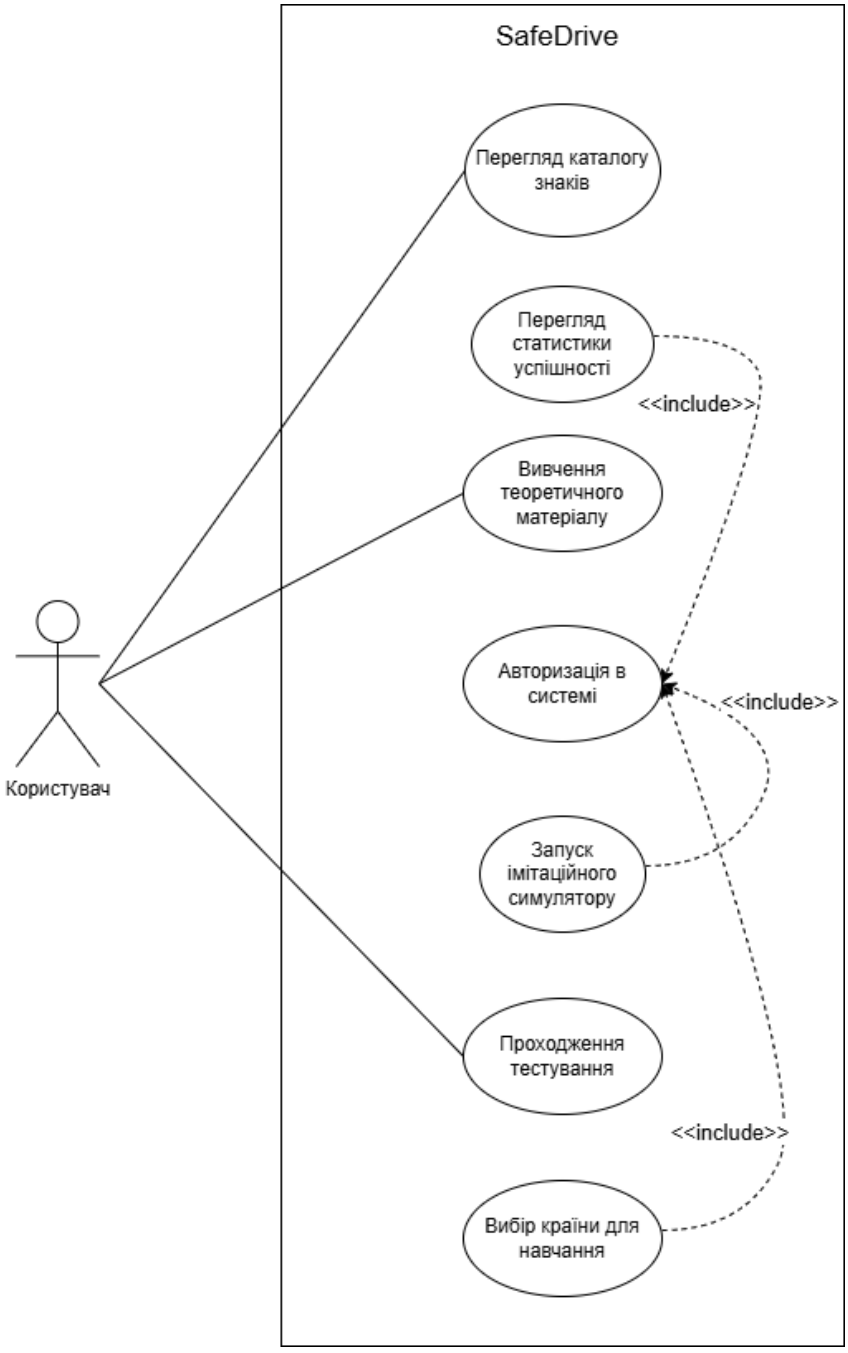


Рис. 2.1 Діаграма варіантів використання системи SafeDrive

На основі наведеної діаграми було сформовано загальну структуру графічного інтерфейсу та логіку роботи програми. Кожен із визначених прецедентів надалі реалізується у вигляді окремого функціонального модуля системи.

2.1.2 Функціональні вимоги

Функціональні вимоги визначають конкретні дії та сервіси, які програмне забезпечення повинно надавати користувачеві для досягнення цілей навчання:

1. Реєстрація та авторизація користувачів із безпечним збереженням облікових даних шляхом криптографічного хешування паролів. Система повинна забезпечувати створення індивідуальних профілів для персоналізації навчального процесу. Використання сучасних алгоритмів хешування гарантує захист конфіденційної інформації при збереженні у локальній базі даних.
2. Забезпечення доступу до інтегрованої бази знань ПДР із можливістю перемикання між правилами України, Польщі та Німеччини. Програма має надавати актуальну теоретичну інформацію, адаптовану під законодавство обраної юрисдикції, що дозволяє користувачеві готуватися до іспитів у різних країнах в межах єдиного інтерфейсу.
3. Програмний векторний рендеринг повного каталогу дорожніх знаків для збереження їхньої чіткості при масштабуванні. Для забезпечення високої якості візуалізації на моніторах з різною роздільною здатністю всі графічні об’єкти каталогу знаків повинні відмальовуватися за допомогою векторних алгоритмів.
4. Реалізація модуля тестування з генерацією питань за категоріями, підрахунком балів та веденням історії результатів. Екзаменаційна система має автоматизувати процес перевірки знань, надаючи статистику помилок та зберігаючи прогрес кожного користувача для подальшого аналізу.
5. Інтеграція двовимірного симулятора водіння з фізичною моделлю руху, зміною погодних умов та сценаріями критичних ситуацій. Імітаційне середовище повинно забезпечувати реалістичну поведінку транспортного засобу, враховуючи інерцію та гальмівний шлях, а також дозволяти відпрацьовувати навички у складних дорожніх умовах.
6. Автоматичне збереження статистики пройдених тестів та результатів симуляцій у базі даних. Програма має забезпечувати надійне фонове

логування всіх дій користувача, що дозволяє формувати об’єктивний рейтинг успішності та відстежувати динаміку навчання.

2.1.2 Нефункціональні вимоги

1. Система повинна працювати повністю автономно без підключення до Інтернету з використанням локальної бази даних. Це гарантує доступ до всіх навчальних матеріалів та функцій симулятора за будь-яких умов, забезпечуючи незалежність від мережевої інфраструктури.
2. Наявність механізму резервного копіювання та відновлення даних на випадок системних збоїв. Програмний комплекс має включати інструменти для захисту цілісності бази даних, що дозволяє уникнути втрати результатів навчання при некоректному завершенні роботи.
3. Оптимізація ігрового рушія для стабільної та плавної роботи симулятора на комп’ютерах початкового рівня. Архітектура системи повинна забезпечувати високу продуктивність із частотою оновлення кадрів не менше 60 FPS для забезпечення реалістичності керування.
4. Забезпечення кросплатформності для можливості запуску застосунку на різних операційних системах. Програмний код має бути адаптованим для функціонування у різних середовищах без втрати продуктивності чи помилок відображення інтерфейсу.
5. Створення адаптивного графічного інтерфейсу з темною темою оформлення для зниження навантаження на зір користувача. Дизайн програми повинен відповідати принципам ергономіки, забезпечуючи комфортне тривале навчання та коректне відображення елементів на екранах з різними параметрами.

2.1.3 Вимоги до апаратного та програмного середовища

Оскільки “SafeDrive” є десктопним додатком на базі Python, встановлюються наступні системні вимоги:

1. ОС - Windows 10/11, macOS або Linux дистрибутиви.

2. Процесор - 2-ядерний з частотою від 2.0 ГГц для коректної обробки фізичного рушія.
3. ОЗП - не менше 4 ГБ.
4. Відеокарта - підтримка OpenGL для прискорення рендерингу Pygame.
5. Програмне оточення - інтерпретатор Python версії 3.11 або вище з встановленими бібліотеками PyQt6 та Pygame-ce.

2.2Проєктування архітектури системи на основі патерну MVC

Для забезпечення масштабованості, легкості тестування та розділення відповідальності між компонентами системи SafeDrive було обрано архітектурний патерн MVC. Даний підхід дозволяє відокремити дані від графічного інтерфейсу та алгоритмів управління фізичною симуляцією.

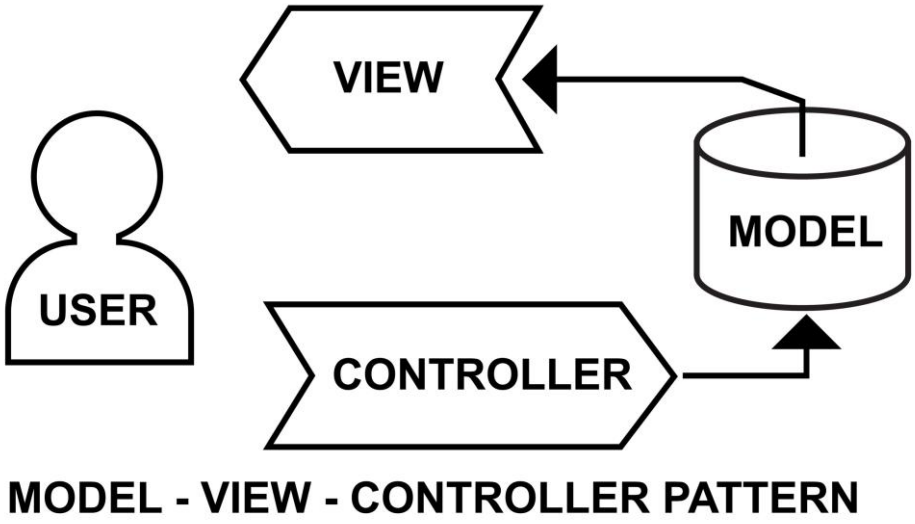


Рис. 2.2 Концептуальна схема архітектурного патерну MVC

Вибір цієї архітектури обумовлений складністю проєкту, що поєднує в собі стандартні віконні інтерфейси для навчання та динамічний ігровий рушій для симуляції водіння. Використання MVC дозволяє модифікувати логіку симулятора або структуру бази даних без необхідності переробляти графічну оболонку застосунку.

Відповідно до обраної парадигми, програмні компоненти застосунку були чітко розподілені за функціональним призначенням. Рівень моделі відповідає за збереження та обробку даних і реалізований через менеджер локальної бази даних. Візуальне представлення інкапсульовано у класах головного вікна, які забезпечують рендеринг графічного інтерфейсу користувача. Керуюча логіка зосереджена у модулях тестування та фізичної симуляції, що обробляють вхідні події та координують обмін інформацією між інтерфейсом і базою даних.

Графічне представлення структури розроблених класів та їх розподіл відповідно до архітектурних рівнів наведено на рисунку 2.3.

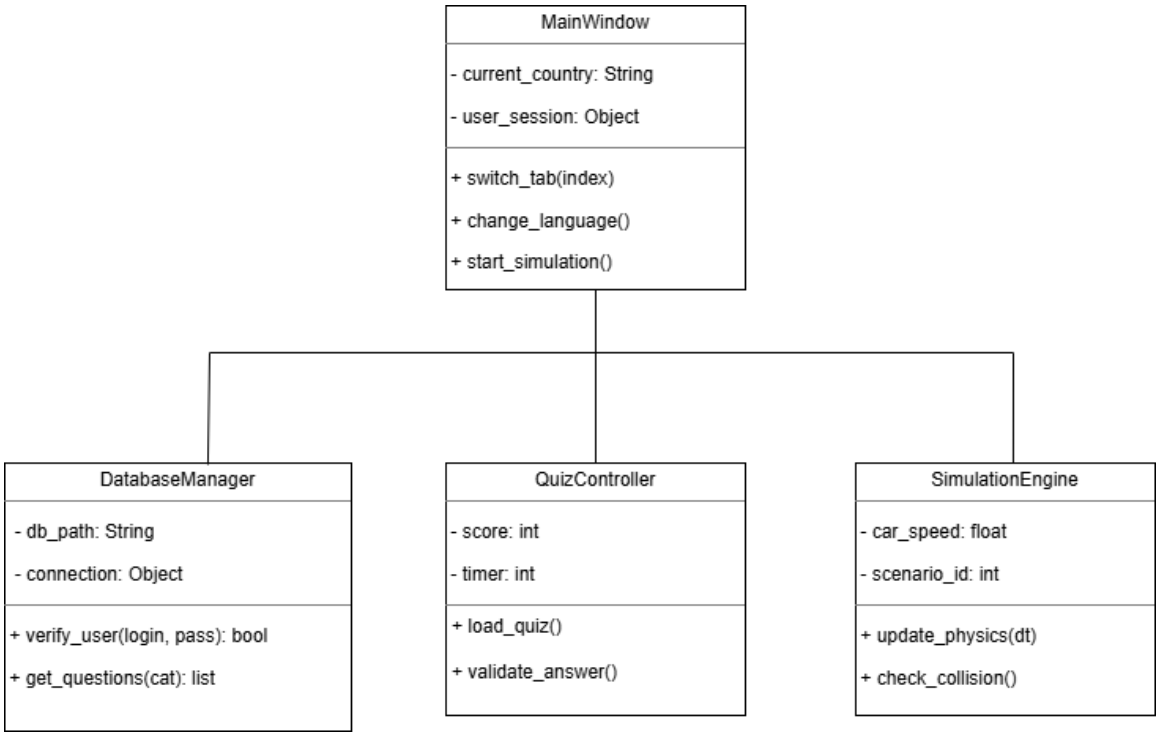


Рис. 2.3 Діаграма класів архітектури системи SafeDrive

Згідно з обраною концепцією, кожен програмний модуль виконує свою частину роботи:

1. Model - представлена класом DatabaseManager. Цей компонент повністю інкапсулює роботу з СУБД SQLite. Він відповідає за виконання SQL-запитів, верифікацію облікових даних користувачів, завантаження теоретичних матеріалів та запис результатів тестування. Важливою особливістю є те, що

інші класи не мають прямого доступу до бази даних, що підвищує безпеку та цілісність інформації.

2. View - реалізовано через клас MainWindow на базі фреймворку PyQt6. Основне завдання цього рівня - візуалізація даних та збір подій від користувача (натискання кнопок, вибір категорій). Завдяки відокремленню представлення ми маємо можливість гнучко налаштовувати інтерфейс за допомогою стилів QSS, не втручаючись у бізнес-логіку програми.
3. Controller - охоплює два ключові модулі, де QuizController керує логікою навчального процесу (відповідає за вибір випадкових питань, перевірку відповідей користувача на правильність та фінальний розрахунок балів), а SimulationEngine відповідає за динамічну частину проєкту (обробляє введення з клавіатури, обчислює вектори руху транспортного засобу, опрацьовує колізії та реалізує 12 різних сценаріїв дорожніх ситуацій на базі бібліотеки Pygame).

Така організація коду дозволяє системі бути гнучкою. Наприклад, за необхідності переходу на 3D-графіку, достатньо буде замінити лише модуль SimulationEngine, залишивши класи MainWindow та DatabaseManager без змін. Це підтверджує високу якість інженерних рішень, закладених на етапі проєктування, та спрощує подальшу підтримку та розширення функціоналу програмного комплексу.

2.2.1 Компонент Model

Модель відповідає за управління даними системи та взаємодію з базою даних. У SafeDrive цей компонент реалізовано через:

Рівень збереження даних - файл database.py, що містить методи роботи з СУБД SQLite для реєстрації користувачів, збереження результатів тестування та логів симуляційних сесій.

Рівень контенту - набір структур, що містять тексти 24 розділів ПДР та параметри дорожніх знаків.

Рівень стану - логіка розрахунку прогресу користувача, яка агрегує дані з таблиць quiz_results та sim_sessions для формування аналітичного звіту.

2.2.2 Компонент View

Цей компонент відповідає за візуалізацію даних та взаємодію з користувачем. Він не містить бізнес-логіки, а лише відображає інформацію, отриману від контролера. У системі SafeDrive представлення розділено на два підрівні:

1. Десктопний GUI - включає класи MainWindow, AuthWindow та TheoryWindow, які відповідають за вікна авторизації, перегляду теорії та налаштування симуляції
2. Графічний рендер симуляції - реалізований у модулі pygame_sim.py, що забезпечує візуалізацію дорожньої обстановки у 2D-просторі з частотою оновлення 60 FPS, включаючи відмальовування дорожнього полотна, спрайтів автомобілів та ефектів погоди.

2.2.3 Компонент Controller

Контролер виступає посередником, який обробляє вхідні дії користувача та оновлює стан Моделі або Представлення.

Логіка тестування: модуль quiz_module.py, що керує послідовністю питань, перевіряє правильність відповідей та ініціює збереження результату в БД.

Фізичний двигун: обробка подій клавіатури в симуляторі для розрахунку вектора прискорення, гальмування та обертання коліс. Контролер взаємодіє з математичною моделлю руху для визначення нових координат ТЗ та перевірки умов колізій з NPC або перешкодами.

Модуль адаптації: country_module.py, який при перемиканні країни Контролером змінює конфігурацію Моделі та View.

2.2.4 Взаємодія компонентів та управління потоками

Особливістю архітектури SafeDrive є необхідність синхронізації головного циклу PyQt6 з ігровим циклом Pygame. Для уникнення “заморожування” інтерфейсу при роботі симулятора, запуск імітаційного модуля реалізовано через окремий потік або через механізм вбудовування вікна Pygame у віджет QWindow. Така гібридна архітектура дозволяє поєднати стабільність бізнес-додатка з продуктивністю ігрового рушія.

2.3Об'єктно-орієнтоване моделювання та проєктування бази даних

Для деталізації архітектурних рішень та візуалізації внутрішніх процесів функціонування системи “SafeDrive” було застосовано мову графічного моделювання UML. Використання UML-діаграм дозволяє забезпечити однозначність трактування бізнес-логіки на етапі програмної реалізації та спрощує подальшу підтримку продукту.

Одним із найважливіших етапів такого проєктування є розробка поведінкових моделей, що відображають динамічний аспект функціонування застосунку. Це дає змогу деталізувати механізми обміну повідомленнями між графічним інтерфейсом, керуючими контролерами та підсистемою доступу до даних під час обробки цільових запитів користувача.

Для аналізу взаємодії об’єктів у часі та визначення логіки виклику методів у межах цього підходу було розроблено діаграму послідовності рис. 2.4.

Вона демонструє ключовий сценарій - процес ініціалізації сесії тестування та отримання даних із бази даних.

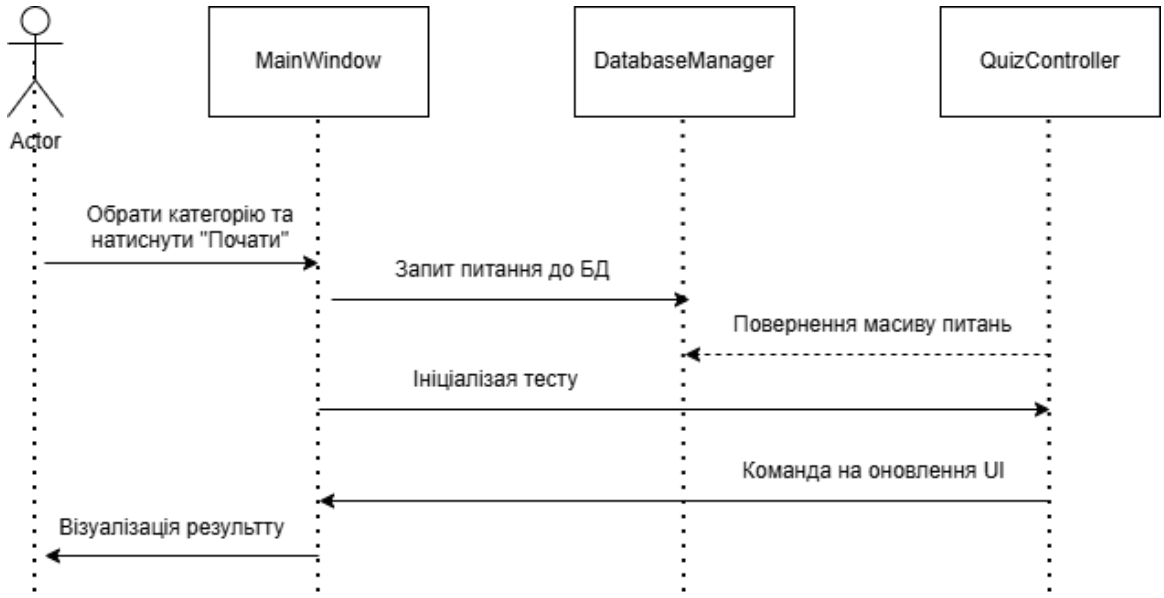


Рис. 2.4 Діаграма послідовності процесу ініціалізації тестування

Як видно з наведеної діаграми, процес розпочинається із взаємодії користувача з головним вікном програми під час вибору категорії іспиту. Далі формується запит до менеджера бази даних для вилучення необхідного масиву питань, який передається до контролера тестування. Після ініціалізації логіки сесії контролер надсилає команду на оновлення графічного інтерфейсу, що завершується візуалізацією першого завдання. Такий розподіл відповідальності між компонентами забезпечує ефективну ізоляцію бізнес-логіки від візуальної оболонки та гарантує стабільність роботи екзаменаційного модуля.

2.3.1 Моделювання функціональних можливостей

Діаграма варіантів використання визначає межі системи, ролі акторів та ключовий функціонал, який вони можуть ініціювати. У системі виділено головного користувача для якого основними прецедентами взаємодії із середовищем є:

1. Автентифікація - процес ідентифікації користувача через введення логіна та пароля з подальшою валідацією в базі даних.
2. Вивчення теорії - отримання доступу до гіпертекстового контенту 24 теоретичних розділів ПДР.

3. Перегляд каталогу знаків - взаємодія з модулем векторного рендерингу для детального вивчення дорожніх об'єктів.
4. Проходження тестування - виклик логіки генерації випадкових питань та отримання підсумкових результатів оцінювання.
5. Запуск імітаційного моделювання - вибір цільового сценарію та безпосереднє керування віртуальним транспортним засобом.
6. Аналіз прогресу - формування запитів до підсистеми збереження даних для відображення персональної статистики успішності.

2.3.2 Моделювання динаміки взаємодії

Діаграма послідовності відображає часову послідовність взаємодії об’єктів системи під час виконання конкретного процесу. Для SafeDrive найбільш показовим є процес проходження іспиту, що включає наступні кроки:

MainWindow передає керування об’єкту QuizController при натисканні відповідної кнопки інтерфейсу.

QuizController ініціює запит до Model, для випадкової вибірки 20 питань із відповідної категорії.

Після отримання даних QuizController оновлює об’єкт View, відображаючи перше питання та запускаючи таймер сесії.

Кожна відповідь користувача валідується Контролером; у разі помилки виконується запит до локальної бази знань для виведення пояснення.

По завершенні тесту QuizController надсилає фінальний бал та лог помилок до Database.py для довгострокового збереження.

2.3.3 Проектування реляційної моделі даних

З огляду на вимогу повної автономності системи, для зберігання даних обрано вбудовану СУБД SQLite. Реляційна модель бази даних safedrive.db спроектована таким чином, щоб забезпечити цілісність інформації та швидку агрегацію статистики.

Основні сутності системи та їхні зв’язки:

1. Таблиця users - зберігає облікові дані користувачів з полями user_id, username, password_hash, registration_date.
2. Таблиця quiz_results - фіксує результати кожного тестування з полями result_id, user_id, category_id, score, time_spent, date. Зв’язок із таблицею користувачів - “один-до-багатьох”.
3. Таблиця sim_sessions - зберігає логи практичних вправ у симуляторі з полями session_id, user_id, scenario_name, is_success, collision_count, duration.
4. Таблиця user_progress - агрегована таблиця для швидкого рендерингу графіків успішності в модулі прогресу.

Політики доступу - найменші привілеї; перевірка власника ресурсу на сервері, route-guards на клієнті та аудит змін.

2.3.4 Моделювання структури класів

Діаграма класів описує статичну структуру системи на рівні програмного коду Python. Ключовими класами проекту є:

1. SafeDriveApp - центральний клас PyQt6, що керує стеком віджетів.
2. SimulationEngine - клас на базі Pygame, що містить ігровий цикл run_loop(), методи обробки фізики update_physics() та малювання render_scene().
3. DatabaseManager - забезпечує низькорівневе з’єднання з SQLite через бібліотеку sqlite3, реалізуючи патерн Singleton для уникнення конфліктів доступу до файлу бази даних.
4. CountryAdapter - спеціалізований клас для динамічної підміни конфігураційних параметрів залежно від обраної країни

3 ОБҐРУНТУВАННЯ ВИБОРУ ПРОГРАМНИХ ЗАСОБІВ ТА ТЕХНОЛОГІЙ РОЗРОБКИ.

3.1Концептуальні підходи до вибору інструментальних засобів розробки

Успішна реалізація програмного комплексу “SafeDrive” залежить від правильного вибору технологічного стеку, який має забезпечити баланс між швидкістю розробки, продуктивністю графічної підсистеми та надійністю роботи з даними. Враховуючи вимоги до кросплатформеності та необхідність інтеграції ігрового рушія з класичним десктопним інтерфейсом, у даному розділі проведено аналіз та обґрунтування обраних інструментальних засобів.

3.1.2 Переваги Python для проєкту SafeDrive

Вибір Python для реалізації системи “SafeDrive” обґрунтований наступними технічними факторами:

1. Потужна підтримка GUI-фреймворків - бібліотека PyQt6, яка є обгорткою над професійним інструментарієм Qt, дозволяє створювати складні багатоланкові інтерфейси з підтримкою векторної графіки, що є критичним для відображення дорожніх знаків високої чіткості.
2. Гнучкість імітаційного моделювання - наявність бібліотеки Pygame дозволяє реалізувати фізичну модель руху автомобіля за лічені рядки коду, забезпечуючи при цьому стабільний ігровий цикл та обробку подій у реальному часі.
3. Ефективна робота з базами даних - вбудована підтримка SQLite через модуль sqlite3 забезпечує повну автономність додатка без необхідності встановлення та налаштування окремих серверів баз даних, що відповідає вимогам до офлайн-роботи SafeDrive.

4. Модульність та читабельність - синтаксис Python сприяє написанню чистого коду за принципами MVC, що спрощує підтримку таких складних модулів, як country_module.py для динамічного перемикання між законодавчими базами різних країн.

3.1.3 Висновок щодо вибору мови

Використання Python дозволяє зосередитися на інженерній логіці симулятора та методичній якості навчального контенту, а не на низькорівневих деталях реалізації інтерфейсу. Це забезпечує надійність системи та можливість її легкого масштабування, наприклад, для додавання нових сценаріїв ДТП або інтеграції з веб-сервісами у майбутньому.

3.2Обґрунтування вибору графічного фреймворку PyQt6

Для реалізації складного багатоланкового інтерфейсу системи “SafeDrive”, що включає систему навігації, інтерактивні навчальні модулі та панелі аналітики, необхідно використовувати потужний інструментарій для побудови графічних інтерфейсів. Основним критерієм вибору була підтримка об’єктно-орієнтованого підходу, стабільність на різних операційних системах та наявність засобів для роботи з векторною графікою.

3.2.1 Порівняльний аналіз GUI-бібліотек для Python

У процесі проєктування було розглянуто три найбільш розповсюджені бібліотеки для розробки десктопних застосунків на Python:

1. Tkinter - вбудована стандартна бібліотека. Вона є легкою та простою у вивченні, проте її функціонал занадто обмежений для професійних проєктів. Tkinter не має вбудованої підтримки сучасних стилів оформлення та засобів високорівневого малювання, що унеможливлює створення якісного візуального каталогу знаків.

2. Kivy - кросплатформений фреймворк, орієнтований на сенсорні інтерфейси та мобільні пристрої. Хоча він дозволяє створювати нестандартні інтерфейси, Kivy використовує нетипові для десктопних систем елементи керування, що ускладнює взаємодію користувача з класичним освітнім ПЗ та потребує значних зусиль для реалізації багатобайтних текстових документів ПДР.
3. PyQt6 - Це найпотужніша бібліотека, що є Python-обгорткою над фреймворком Qt6. Вона надає сотні готових віджетів, професійну систему компонування та потужні засоби обробки подій. PyQt6 забезпечує нативний вигляд застосунку в усіх операційних системах та підтримує сучасний стандарт відображення з високою щільністю пікселів.

3.2.2 Технічні переваги PyQt6 для проєкту SafeDrive

Вибір PyQt6 для реалізації інтерфейсу SafeDrive обґрунтований наступними факторами:

1. Механізм сигналів та слотів - ця технологія дозволяє організувати гнучку та безпечну взаємодію між різними частинами програми. Наприклад, при зміні країни в налаштуваннях (country_module.py), сигнал автоматично ініціює оновлення контенту в усіх відкритих вікнах.
2. Підтримка QSS - використання мови стилів, аналогічної CSS, дозволяє відокремити візуальне оформлення від програмної логіки. У SafeDrive це використовується для динамічної зміни колірної гами та оформлення інтерфейсу залежно від обраної країни, що підвищує автентичність навчального процесу.
3. Векторний рендеринг за допомогою QPainter - це критично важливий інструмент для модуля знаків. На відміну від растрових зображень, використання класу QPainter дозволяє програмно малювати знаки у реальному часі. Це гарантує, що при будь-якому масштабуванні вікна дорожні знаки будуть залишатися ідеально чіткими, що відповідає вимогам до якості освітнього контенту.

3.2.3 Інтеграція з іншими модулями

PyQt6 легко інтегрується з іншими компонентами системи. Зокрема, використання класу QWindow дозволяє вбудувати ігрове вікно Pygame безпосередньо в інтерфейс головного вікна програми, створюючи цілісний та нерозривний користувацький досвід. Крім того, наявність віджетів для відображення багатого тексту дозволяє коректно рендерити 24 розділи теоретичного матеріалу з ілюстраціями та таблицями.

3.3 Обґрунтування вибору ігрового рушія Pygame для модуля симуляції

Для реалізації практичного компонента системи “SafeDrive” - модуля 2D-симуляції дорожніх ситуацій - виникла потреба у використанні спеціалізованого ігрового рушія. Основними критеріями вибору були висока продуктивність рендерингу, наявність засобів обробки фізичних колізій та легкість інтеграції з основним кодом мовою Python.

3.3.1 Аналіз ігрових бібліотек для Python

Було проведено порівняння трьох найбільш актуальних рішень для розробки 2D-графіки на Python:

1. Arcade - сучасна бібліотека, орієнтована на простоту використання та підтримку апаратного прискорення через OpenGL. Проте вона має меншу спільноту та меншу кількість документації щодо реалізації специфічних фізичних сценаріїв, ніж конкуренти.
2. Panda3D - повноцінний 3D-рушій з великими можливостями. Використання Panda3D для 2D-тренажера було визнано надлишковим, оскільки це суттєво збільшило б системні вимоги додатка та складність підготовки графічних асетів.
3. Pygame - обрана бібліотека, що є найбільш стабільною та перевіреною часом. Вона забезпечує низькорівневий доступ до відеокарти через SDL, що

дозволяє досягти стабільної частоти 60 FPS навіть на малопотужних комп’ютерах.

3.3.2 Технічні можливості Pygame для реалізації SafeDrive

Вибір Pygame зумовлений наявністю вбудованих механізмів, необхідних для точного моделювання дорожнього руху:

1. Обробка подій у реальному часі - Pygame дозволяє з мінімальною затримкою зчитувати стан клавіатури, що критично важливо для відпрацювання реакції водія при екстреному гальмуванні або об’їзді перешкод у симуляторі.
2. Математика спрайтів та колізій - рушій надає зручні інструменти для виявлення зіткнень об’єктів за допомогою масок та прямокутників. У “SafeDrive” це використовується для автоматичної фіксації ДТП при контакті автомобіля користувача з NPC-трафіком або дорожнім огородженням.
3. Фізична модель руху - за допомогою засобів Pygame реалізовано розрахунок векторів швидкості та тертя. Це дозволяє симулювати зміну гальмівного шляху залежно від погодних умов, що є основою для сценаріїв “Ожеледиця” та “Аквапланування”.
4. Система шарів рендерингу - можливість розділення малювання дорожнього полотна, знаків, розмітки та автомобілів дозволяє створювати складні та наочні 2D-сцени, які легко масштабуються під вимоги конкретного навчального сценарію.

3.3.3 Сценарії імітаційного моделювання

Завдяки гнучкості Pygame у системі реалізовано 12 унікальних сценаріїв, кожен з яких базується на окремому алгоритмі поведінки навколишнього середовища. Це включає динамічну зміну прозорості спрайтів для імітації туману, генерацію частинок таких як, анімація дощу або снігу та складні траєкторії руху NPC-автомобілів на перехрестях.

Таким чином, Pygame виступає ідеальним інструментом для створення “пісочниці”, де користувач може безпечно помилятися, засвоюючи на практиці теоретичні положення ПДР.

3.4Обґрунтування вибору СУБД SQLite для збереження даних та прогресу

Для забезпечення надійної роботи системи “SafeDrive” у режимі офлайн та збереження персоналізованих даних користувачів виникла потреба в інтеграції системи управління СУБД. Основним викликом було обрання технології, яка б не потребувала встановлення додаткового серверного ПЗ на комп'ютер користувача, але при цьому підтримувала повноцінні SQL-запити для аналізу успішності.

3.4.1 Аналіз та порівняння СУБД

У ході проєктування було розглянуто кілька варіантів організації сховища даних:

1. JSON/CSV файли - найпростіший спосіб зберігання, який не потребує сторонніх бібліотек. Проте такі файли не забезпечують цілісності даних, не підтримують складні зв’язки між таблицями та мають низьку швидкість обробки при збільшенні обсягу статистики тестувань.
2. MySQL / PostgreSQL - професійні клієнт-серверні СУБД. Вони надають найвищий рівень продуктивності, але потребують розгортання окремого сервера, що суперечить концепції SafeDrive як легкої автономної навчальної програми.
3. SQLite - вбудована реляційна СУБД. На відміну від серверних рішень, SQLite зберігає всю базу даних в одному локальному файлі, що ідеально підходить для десктопних застосунків.

3.4.2 Переваги використання SQLite у проекті

Вибір SQLite обґрунтований наступними інженерними факторами:

1. Повна автономність - бібліотека для роботи з SQLite входить у стандартний пакет Python, що спрощує розгортання програми та гарантує її роботу без доступу до Інтернету.
2. ACID-відповідність - СУБД гарантує атомарність та надійність транзакцій. Це означає, що результати тесту або лог симуляції будуть збережені коректно навіть при раптовому вимкненні живлення або аварійному завершенні програми.
3. Швидкість доступу - для локального обсягу даних SQLite демонструє вищу швидкість читання та запису, ніж класичні клієнт-серверні системи, оскільки відсутні витрати часу на мережеву взаємодію.

3.4.3 Реалізація архітектури даних

У модулі database.py реалізовано логіку взаємодії з базою даних, що включає створення чотирьох основних таблиць:

1. Таблиця users - зберігання профілів із застосуванням унікальних індексів для юзернеймів.
2. Таблиця quiz_results - фіксація балів, витраченого часу та категорії питань.
3. Таблиця sim_sessions - логування результатів 12 практичних вправ симулятора.
4. Таблиця achievements - таблиця для гейміфікації процесу навчання.

Використання реляційної моделі дозволяє за допомогою одного SQL-запиту формувати складні звіти, наприклад, визначати теми ПДР, у яких користувач найчастіше допускає помилки, що є ключовим елементом адаптивного навчання у SafeDrive.

3.5Програмна реалізація ключових компонентів системи

Після вибору архітектури та технологій почався етап безпосереднього написання коду. Проєкт був розбитий на окремі незалежні модулі, це дозволило відділити роботу з базою даних від графічного інтерфейсу та фізики симулятора.

Такий підхід зробив код чистішим і значно спростив його подальшу підтримку. Розглянемо, як саме реалізовані найцікавіші частини програми.

3.5.1 Реалізація модуля взаємодії з даними

Для роботи з базою даних у файлі database.py використовуються стандартні можливості SQLite. Щоб дані користувачів були в безпеці, паролі не записуються у відкритому вигляді - вони проходять хешування за алгоритмом SHA-256. Крім того, зручним рішенням стало те, що під час реєстрації нового акаунту програма автоматично створює для нього порожній запис, де надалі зберігатиметься статистика навчання.

Фрагмент коду, що відповідає за безпечну реєстрацію користувача та ініціалізацію його прогресу, наведено нижче:

```
def register_user(username: str, password: str) -> tuple[bool, str]:
    username = username.strip()
    if not username:
        return False, "Нікнейм не може бути порожнім"
    if len(username) < 3:
        return False, "Нікнейм повинен містити мінімум 3 символи"
    if len(username) > 32:
        return False, "Нікнейм не може бути довшим за 32 символи"
    if not password:
        return False, "Пароль не може бути порожнім"
    if len(password) < 4:
        return False, "Пароль повинен містити мінімум 4 символи"

    conn = get_connection()
    try:
        now = datetime.now().isoformat()
        conn.execute(
            "INSERT INTO users (username, password, created_at) VALUES (?, ?, ?)",
            (username, _hash(password), now)
        )
        # Створюємо порожній запис прогресу
        user_id = conn.execute(
            "SELECT id FROM users WHERE username = ?", (username,)
        ).fetchone()["id"]
        conn.execute(
            "INSERT INTO user_progress (user_id, updated_at) VALUES (?, ?)",
            (user_id, now)
        )
        conn.commit()
        return True, ""
    except sqlite3.IntegrityError:
        return False, "Користувач з таким нікнеймом вже існує"
    finally:
        conn.close()
```

Щоб не робити зайвих запитів і не плодити дублікати під час оновлення прогресу. Він перевіряє: якщо запис вже є - оновлює його, якщо немає - створює новий.

3.5.2 Реалізація фізичної моделі руху

Однією з найскладніших частин стала реалізація фізики автомобіля. У модулі pygame_sim.py за це відповідає клас Car. Щоб машина рухалася плавно і реалістично, її координати перераховуються щокадру за допомогою тригонометрії залежно від кута повороту та поточної швидкості. До того ж, тут враховано вплив зовнішніх факторів, як-от занесення на льоду чи зміщення під час аквапланування.

Логіку розрахунку нових координат автомобіля під час руху та фіксації слідів від шин представлено нижче:

```
def move(self, drift_x=0.0, drift_y=0.0):
    rad = math.radians(self.angle)
    self.pos.x += math.sin(rad) * self.speed + drift_x
    self.pos.y -= math.cos(rad) * self.speed + drift_y
    if abs(self.speed) > 2.8:
        self.skids.append((self.pos.x, self.pos.y))
        if len(self.skids) > 35:
            self.skids.pop(0)
```

3.5.3 Модуль збереження прогресу з резервним копіюванням

Щоб програма працювала стабільно і користувач не втратив свою статистику при можливих збоях бази даних, у файлі progress_module.py реалізовано резервний механізм.

Його логіка дуже проста, система спочатку намагається завантажити результати з SQLite, а якщо виникає помилка або відсутнє підключення - підтягує дані з локального JSON-файлу.

Реалізацію цього відмовостійкого механізму завантаження статистики продемонстровано нижче:

```
def _load_data(self) -> dict:
    default = {
        "total_tests": 0,
        "total_correct": 0,
        "total_questions": 0,
        "categories": {},
        "theory_topics_read": 0,
        "sim_sessions": 0,
    }
    # Спочатку БД
    if self._user_id:
        try:
            db_data = get_progress(self._user_id)
            if db_data:
                default.update(db_data)
                return default
        except Exception:
            pass
    # Fallback — JSON файл
    try:
```

```
    if os.path.exists(DATA_FILE):
        with open(DATA_FILE, "r", encoding="utf-8") as f:
            data = json.load(f)
            default.update(data)
except Exception:
    pass
return default
```

3.6Інтерфейс користувача та результати роботи програми

Головною метою при розробці графічного інтерфейсу було створення сучасного, інтуїтивно зрозумілого середовища, яке не відвертає увагу від процесу навчання. Завдяки використанню фреймворку PyQt6 та каскадних таблиць стилів реалізовано адаптивний дизайн із темною темою оформлення. Це рішення значно знижує навантаження на зір під час тривалої роботи з теоретичним матеріалом.

Після успішної авторизації користувач потрапляє до головного меню Рис. 3.4, де має змогу обрати країну для вивчення ПДР та відповідний навчальний модуль.

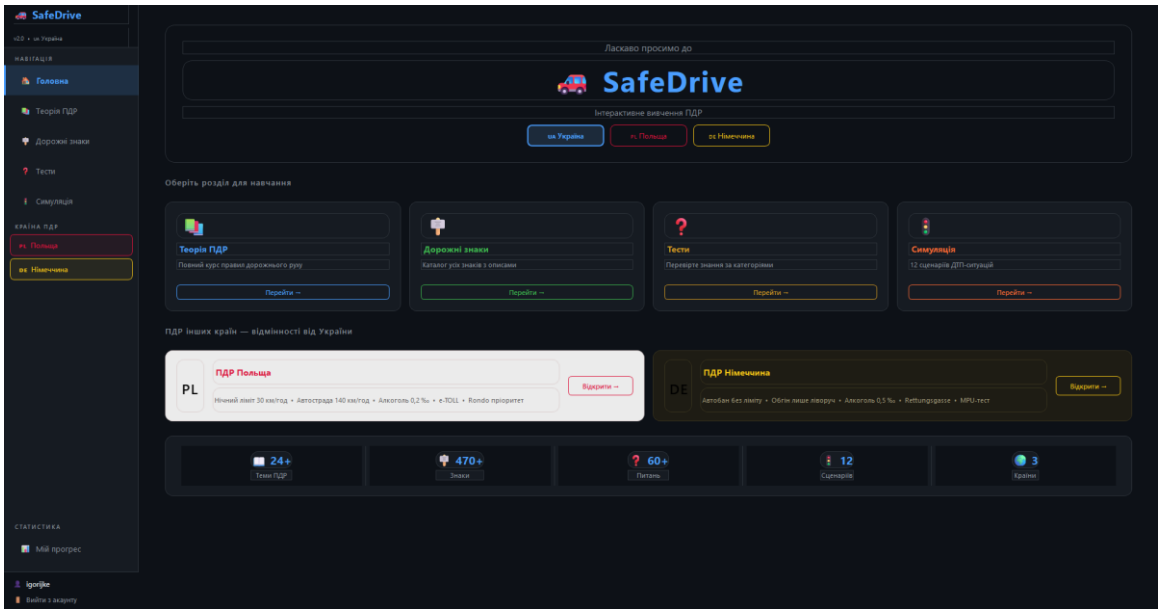


Рис. 3.4 Головне меню навчального комплексу SafeDrive

Навчальний процес поділено на кілька етапів. Наприклад, у модулі “Дорожні знаки” реалізовано векторний рендеринг зображень за допомогою класу QPainter.

Це гарантує ідеальну чіткість знаків при будь-якому масштабуванні вікна, що є критично важливим для запам'ятовування дрібних деталей Рис. 3.5.

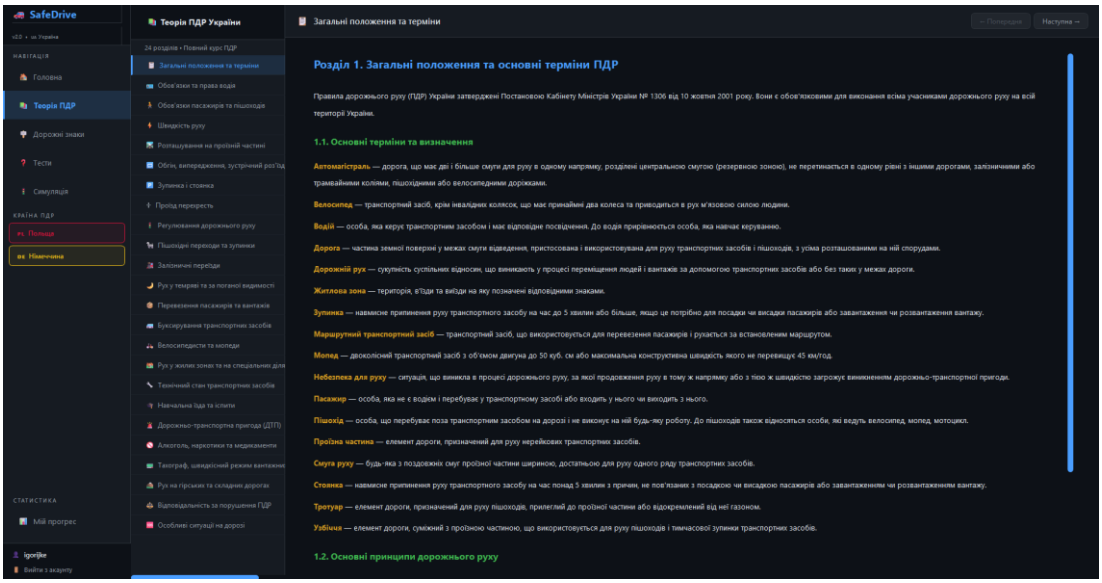


Рис. 3.5 Інтерфейс модуля вивчення дорожніх знаків

Після засвоєння теорії користувач може перевірити свої знання у модулі тестування. Завдяки інтеграції з базою даних SQLite, питання генеруються випадковим чином у межах обраної категорії. Інтерфейс модуля тестування спроєктовано з урахуванням принципів мінімалізму, щоб не відвертати увагу від навчального процесу. Програмний алгоритм контролера виключає можливість повторення питань у межах однієї екзаменаційної сесії, забезпечуючи максимальну об'єктивність оцінювання знань. Візуальне оформлення вікна тестування використовує контрастні кольорові схеми, що полегшує читання тексту та мінімізує зорову втому при тривалій роботі з навчальною програмою. Після завершення тесту система автоматично підраховує бали та зберігає результат у загальну статистику профілю користувача Рис. 3.6.

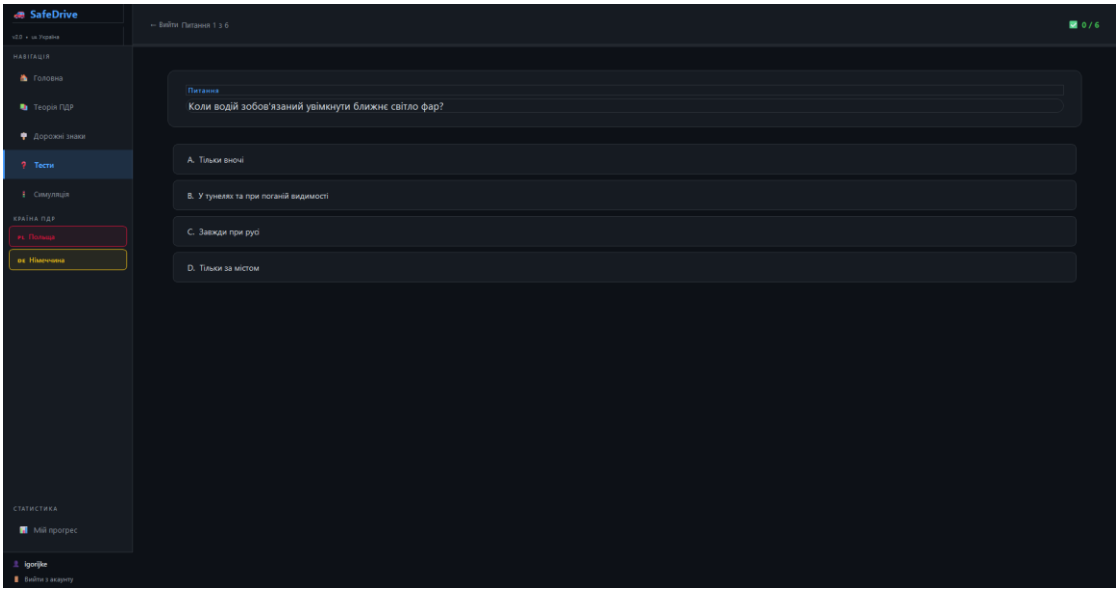


Рис. 3.6 Модуль тестування

Практичне закріплення навичок відбувається у модулі 2D-симуляції, розробленому на базі ігрового рушія Pygame. На рисунку 3.7 наведено приклад роботи одного з 12 доступних сценаріїв. Система в реальному часі відстежує дії водія та нараховує штрафні бали у разі порушення правил або виникнення ДТП.

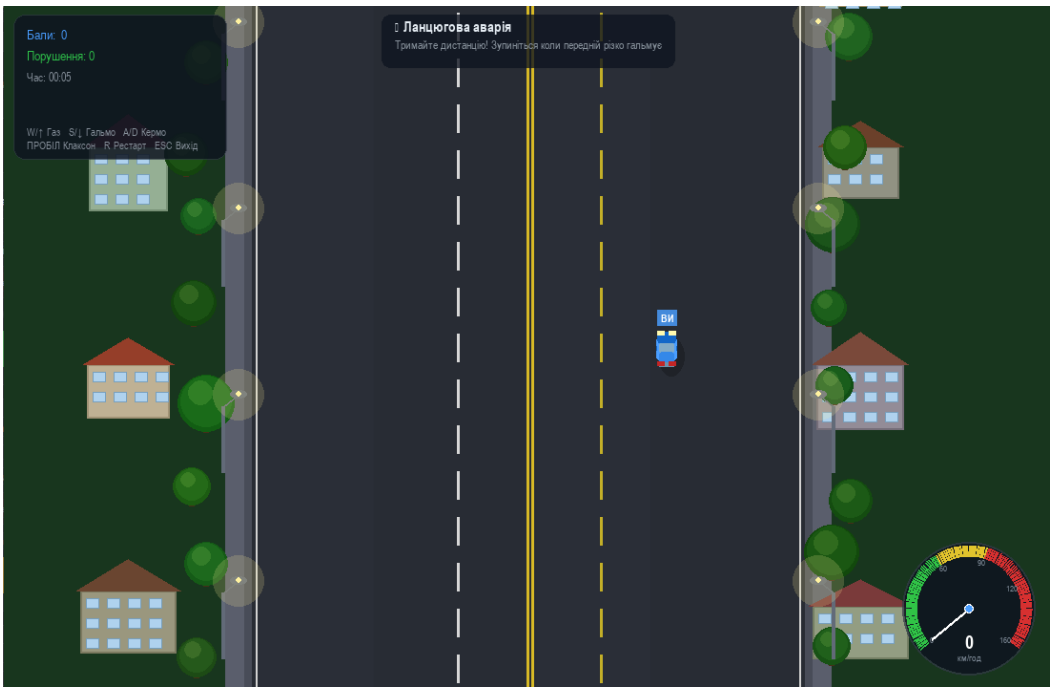


Рис. 3.7 Процес імітаційного моделювання дорожньої ситуації

Як видно з наведеного рисунка, робоча область симулятора містить не лише ігрове поле з видом зверху, але й інтерактивну інформаційну панель (HUD). На ній

у режимі реального часу відображаються поточні показники імітації: спідометр зі швидкістю транспортного засобу, лічильник нарахованих штрафних балів за помилки та загальний час проходження обраного сценарію.

Таким чином, розроблений програмний комплекс успішно поєднує теоретичну базу, систему алгоритмічного контролю знань та інтерактивний симулятор водіння, утворюючи цілісне та повнофункціональне середовище для ефективного вивчення Правил дорожнього руху. Запропонована архітектурна реалізація, заснована на патерні MVC та гнучкому графічному рушії, довела свою високу продуктивність та стабільність роботи на комп'ютерах з різними апаратними характеристиками. Це підтверджує успішне виконання поставлених завдань і робить готовий програмний продукт SafeDrive повністю придатним для використання як надійного інструменту для індивідуальної підготовки майбутніх водіїв.

3.7Ергономіка графічного інтерфейсу та забезпечення позитивного користувацького досвіду

При розробці навчального програмного забезпечення критично важливим аспектом є не лише функціональність, але й ергономіка взаємодії користувача із системою. Оскільки вивчення правил дорожнього руху вимагає високої концентрації уваги та запам'ятовування великих обсягів текстової і графічної інформації, інтерфейс комплексу SafeDrive був спроєктований з урахуванням сучасних принципів когнітивної психології та юзабіліті.

По-перше, було реалізовано концепцію «темної теми», як основної колірної схеми застосунку. Використання глибоких темних відтінків для фону в поєднанні з контрастним світлим текстом дозволяє значно зменшити навантаження на зір під час тривалих сесій навчання. Особлива увага приділялася контрастності елементів керування: акцентні кольори використовуються виключно для привернення уваги до важливих інтерактивних деталей або для індикації статусу.

По-друге, навігаційна модель застосунку побудована за парадигмою односторінкового інтерфейсу з використанням компонента QStackedWidget. Це усуває необхідність відкриття багатьох дочірніх вікон, що часто дезорієнтує користувачів-початківців. Уся навігація згрупована в лівій бічній панелі, що забезпечує миттєвий доступ до будь-якого з основних модулів системи в один клік.

Крім того, інтерфейс адаптовано для підтримки різних роздільних здатностей екрана. Завдяки системі динамічного компонування бібліотеки PyQt6 та використанню відносних розмірів замість абсолютних у пікселях, застосунок коректно відображається як на сучасних 4K-моніторах, так і на ноутбуках із базовими характеристиками дисплея. Векторний підхід до малювання дорожніх знаків ідеально доповнює цю концепцію, гарантуючи відсутність візуальних артефактів чи розмиття при масштабуванні вікна, що є поширеною проблемою серед програм-аналогів.

Важливим елементом позитивного користувацького досвіду є система миттєвого зворотного зв'язку. Під час проходження екзаменаційних тестів система не лише фіксує обраний варіант, але й одразу візуалізує результат за допомогою чіткої кольорової індикації. У разі допущення помилки користувач миттєво отримує розгорнуте пояснення, що перетворює процес тестування з простої перевірки на активну фазу навчання.

Окрему увагу було приділено типографіці та інформаційній архітектурі. Для текстових блоків обрано сучасні шрифти без зарубок, які забезпечують високу читабельність з екрана. Увесь теоретичний матеріал чітко структурований за допомогою ієрархії заголовків, маркованих списків та логічного виділення ключових термінів. Такий підхід суттєво знижує когнітивне навантаження та допомагає учневі швидко сканувати текст у пошуках потрібної інформації.

Для підтримки постійної мотивації користувачів у графічний інтерфейс інтегровано елементи наочної аналітики та відстеження прогресу. Панель особистої статистики відображає успішність проходження тестів та симуляцій у вигляді зрозумілих індикаторів і відсоткових шкал, заохочуючи учня до регулярних занять. Візуалізація досягнень дозволяє користувачеві самостійно

оцінювати свій поточний рівень підготовки та своєчасно виявляти прогалини у знаннях з конкретних розділів ПДР, що робить процес самостійного навчання більш усвідомленим і структурованим.

Окремим аспектом якісного UX є реалізація принципу запобігання помилкам та забезпечення інтуїтивного керування. Усі інтерактивні елементи, зокрема навігаційні кнопки та поля введення в модулі авторизації, мають чітко виражені візуальні стани при наведенні та активації. Водночас керування у вікні 2D-симулятора розроблено з урахуванням загальноприйнятих стандартів, що суттєво знижує поріг входження для нових користувачів. Завдяки продуманій ергономіці та логічному розміщенню елементів, програмний комплекс SafeDrive формує комфортне середовище, у якому інтерфейс залишається допоміжним інструментом і не відвертає увагу від засвоєння правил дорожнього руху.

Ще одним важливим кроком у напрямку покращення користувацького досвіду стало впровадження системи контекстних підказок та підтримка базових принципів інклюзивного дизайну. Для нових користувачів передбачено ненав'язливе ознайомлення з функціоналом під час першого запуску програми. Усі ключові елементи керування супроводжуються спливаючими текстовими підказками, які пояснюють їхнє призначення без необхідності звернення до зовнішньої документації чи окремого розділу довідки. Це дозволяє учневі постійно залишатися в контексті поточного навчального завдання.

Крім того, під час проєктування Head-Up Display панелі безпосередньо у модулі 2D-симуляції було застосовано строгий принцип візуального мінімалізму. Усі другорядні інформаційні блоки приховані, а на екран виводяться лише ті показники, що є критично важливими для прийняття рішень під час руху: поточна швидкість, лічильник порушень ПДР та таймер виконання сценарію. Такий підхід імітує фокус водія на реальній панелі приладів автомобіля і запобігає розсіюванню уваги під час виконання складних маневрів. У сукупності всі ці дизайнерські рішення створюють цілісне, безбар'єрне та максимально ефективне освітнє середовище.

4 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ SAFEDRIVE

4.1Опис структури проєкту та функціонального призначення модулів

Програмний комплекс SafeDrive має модульну структуру, що відповідає принципам MVC та чистої архітектури. Такий підхід дозволив незалежно розробляти, тестувати та оновлювати бази знань ПДР, графічний інтерфейс та фізичне ядро симуляції.

Основними компонентами системи є наступні файли:

main.py - головний файл, що ініціалізує середовище PyQt6, підключає базу даних та запускає життєвий цикл застосунку.

database.py - модуль взаємодії з СУБД SQLite, що містить логіку CRUD-операцій для профілів користувачів, збереження прогресу та результатів тестування.

main_window.py - реалізація центрального графічного інтерфейсу, навігаційної панелі та логіки перемикання між навчальними режимами за допомогою QStackedWidget.

signs_module.py - програмний рендер дорожніх знаків, що використовує векторне малювання для забезпечення абсолютної чіткості зображень.

quiz_module.py - рушій тестування, що керує вибіркою питань, підрахунком правильних відповідей та формуванням підсумкової статистики.

pygame_sim.py - ядро імітаційного моделювання, де реалізовано 12 сценаріїв дорожніх ситуацій, фізику руху автомобіля та обробку колізій у реальному часі.

country_module.py - адаптер для динамічної зміни контенту та візуальної теми залежно від обраної країни.

4.2Реалізація підсистеми управління даними та збереження результатів

Оскільки базову логіку автентифікації та безпеки було закладено на етапі проєктування системи, на етапі безпосередньої розробки головна увага приділялася алгоритмам фіксації навчального прогресу. Важливим завданням було забезпечити швидкий та безперебійний запис результатів проходження дорожніх сценаріїв без затримок у роботі інтерфейсу.

У файлі database.py для цього розроблено метод save_sim_session. Він автоматично фіксує ідентифікатор користувача, назву пройденого сценарію, кількість набраних балів, допущені порушення ПДР та загальний статус завершення:

```
def save_sim_session(user_id: int, scenario: str,
        score: int, violations: int, completed: bool):
    """Зберігає результат сесії симуляції"""
    conn = get_connection()
    try:
        conn.execute("""
            INSERT INTO sim_sessions
                (user_id, scenario, score, violations, completed, played_at)
            VALUES (?, ?, ?, ?, ?, ?)
        """, (user_id, scenario, score, violations,
              1 if completed else 0, datetime.now().isoformat()))
        conn.commit()
    finally:
        conn.close()
```

Такий підхід із використанням параметризованих запитів додатково захищає систему від потенційних SQL-ін'єкцій, а використання блоку try...finally гарантує обов'язкове закриття з'єднання з базою даних навіть у разі виникнення помилок під час виконання запиту.

4.3Алгоритм векторного рендерингу графічних об'єктів

Важливою особливістю SafeDrive є відмова від статичних растрових зображень для дорожніх знаків на користь програмного векторного малювання через клас QPainter. Це дозволяє уникнути розмиття та пікселізації при масштабуванні вікна програми, що є критичним для освітнього продукту.

Кожен тип знака має власний метод малювання. Наприклад, логіку побудови геометрії попереджувального знака за допомогою векторних контуріва представлено нижче:

```
def _draw_warn_triangle(self, p, invert=False):
    s = self.size; pad = s * 0.06; bw = s * 0.12
    path = QPainterPath()
    if not invert:
        path.moveTo(s/2, pad + bw/2)
        path.lineTo(s - pad - bw/2, s - pad - bw/2)
        path.lineTo(pad + bw/2, s - pad - bw/2)
    else:
        path.moveTo(pad + bw/2, pad + bw/2)
        path.lineTo(s - pad - bw/2, pad + bw/2)
        path.lineTo(s/2, s - pad - bw/2)
    path.closeSubpath()
    p.setPen(QPen(self.RED, bw, Qt.PenStyle.SolidLine, Qt.PenCapStyle.RoundCap, Qt.PenJoinStyle.RoundJoin))
    p.setBrush(self.WHITE)
    p.drawPath(path)
```

Завдяки цьому підходу система підтримує високу чіткість інтерфейсу на моніторах з будь-якою роздільною здатністю, включаючи 4K-дисплеї.

4.4Програмна реалізація фізичного рушія симуляції

Модуль pygame_sim.py реалізує ігровий цикл зі стабільною частотою оновлення 60 кадрів на секунду. Окрім математичного розрахунку траєкторії, важливою частиною розробки стала реалізація методу ctrl, який перетворює команди користувача з клавіатури на фізичні величини: прискорення, гальмування та кут повороту коліс.

Логіку обробки вхідних сигналів керування та симуляцію сили тертя представлено нижче:

```
def ctrl(self, keys):
    fwd = keys[pygame.K_UP]   or keys[pygame.K_w]
    bck = keys[pygame.K_DOWN] or keys[pygame.K_s]
    lt  = keys[pygame.K_LEFT] or keys[pygame.K_a]
    rt  = keys[pygame.K_RIGHT]or keys[pygame.K_d]
    if keys[pygame.K_SPACE]:
        self.horn_timer = 70
    else:
        self.horn_timer = max(0, self.horn_timer - 1)

    if fwd:
        self.speed = min(self.speed + self.accel, self.max_speed)
    elif bck:
        self.speed = max(self.speed - self.brake_f, -self.max_speed * 0.35)
    else:
        if self.speed > 0:
            self.speed = max(0.0, self.speed - self.friction)
        else:
```

```
            self.speed = min(0.0, self.speed + self.friction)

    if abs(self.speed) > 0.08:
        t = self.turn_rate * (self.speed / self.max_speed)
        if lt: self.angle -= t; self._indicator = -1
        elif rt: self.angle += t; self._indicator = 1
        else: self._indicator = 0
    else:
        self._indicator = 0
```

В межах цього модуля реалізовано 12 унікальних сценаріїв. Для кожного з них параметри accel та friction змінюються динамічно. Наприклад, у сценарії “Ожеледиця” показник тертя суттєво зменшується, що змушує алгоритм довше зберігати інерцію руху, імітуючи ковзання.

4.5Тестування та верифікація програмного продукту

Перевірка працездатності та надійності SafeDrive здійснювалася на фінальних етапах розробки програмного комплексу. Для цього було застосовано декілька взаємопов'язаних рівнів тестування: модульне, інтеграційне та загальне системне оцінювання. Головне завдання перевірки полягало у виявленні можливих логічних помилок під час роботи симулятора, тестуванні стійкості бази даних SQLite за нестандартних дій користувача та перевірці швидкодії інтерфейсу.

Перед безпосереднім виконанням підготовлених сценаріїв було налаштовано ізольоване тестове середовище, що максимально наближене до реальних умов експлуатації кінцевим користувачем. Апробація проводилася на пристроях із різними апаратними конфігураціями, включаючи ноутбуки базового рівня. Це дало змогу об'єктивно оцінити не лише правильність відпрацювання алгоритмів, але й стабільність роботи ігрового рушія Pygame за умов обмежених системних ресурсів. Також додатково перевірялася реакція програми на виняткові ситуації, зокрема спроби введення некоректних даних під час авторизації, відсутність доступу до локальної бази даних та екстрене переривання роботи симулятора. Такий всебічний підхід дозволив гарантувати високу надійність програмного продукту перед його фінальним розгортанням. Узагальнені результати проведення цих комплексних тестів зведені у таблиці 4.1

Таблиця 4.1

Результати функціонального тестування модулів системи

№	Опис сценарію	Кроки виконання	Очікуваний результат	Статус
1	Створення нового профілю	Введення коректного імені та пароля, натискання кнопки реєстрації	У базі даних з'являється новий рядок, пароль шифрується	Успішно
2	Формування білета для іспиту	Перехід до вкладки “Тести”, клік на будь-яку тему	Програма вибирає з бази даних 20 унікальних питань	Успішно
3	Збереження оцінок	Завершення тестування з допущенням кількох помилок	Результат фіксується в таблиці результатів, оновлюється графік прогресу	Успішно
4	Перевірка фізики гальмування	Розгін авто до 60 км/год у симуляторі, активація гальмування	Транспортний засіб зупиняється плавно	Успішно
5	Фіксація аварійних ситуацій	Пряме зіткнення машини гравця з NPC-трафіком на дорозі	Ігровий цикл зупиняється, з'являється вікно з описом ДТП, бали заносяться в лог	Успішно
6	Робота в автономному режимі	Видалення або перейменування файлу safedrive.db та запуск програми	Застосунок підтягує останні відомі дані про прогрес із резервного JSON-файлу	Успішно
7	Масштабування графіки знаків	Розгортання вікна на весь екран (тестування на 4K моніторі)	Векторні знаки перераховують координати	Успішно

Аналіз отриманих під час тестів даних підтвердив, що підсистема безпеки працює стабільно, паролі надійно захищені, а запити блокують спроби проведення SQL-ін'єкцій. Особливу увагу було приділено стійкості до збоїв, під час штучного відключення файлу бази даних механізм резервного копіювання успішно перемикался на локальний JSON-файл, що дозволило зберегти статистику пройдених уроків без втрат.

Для оцінки поведінки графічного двигуна Pygame було проведено стрес-тестування. На одній локації одночасно генерувався рух 20 автономних автомобілів (NPC) із паралельним увімкненням частинок дощу та снігу. Навіть за такого навантаження вбудований лічильник кадрів показав стабільні 55–60 FPS на комп'ютері середньої конфігурації, що свідчить про якісну оптимізацію ігрового циклу.

4.6Керівництво користувача та особливості розгортання системи

Програмний комплекс SafeDrive спроєктовано як готове до використання рішення, яке постачається у вигляді єдиного дистрибутиву. Це повністю позбавляє кінцевого користувача необхідності розгортати середовище розробника, встановлювати інтерпретатор мови Python чи завантажувати сторонні бібліотеки, усі необхідні компоненти, графічні ресурси та локальна база даних уже інтегровані всередину архітектури застосунку.

Для запуску системи на комп'ютерах під управлінням операційної системи Windows достатньо виконати кілька простих кроків:

1. Розпакувати архів із програмою у будь-яку зручну директорію на жорсткому диску
2. Запустити центральний виконуваний файл SafeDrive.exe
3. Під час першого старту програма автоматично перевірить цілісність вбудованої бази даних SQLite та створить локальний файл конфігурації для збереження прогресу

Графічний інтерфейс програми розроблявся з розрахунком на самостійне навчання, тому він є інтуїтивно зрозумілим і не перевантажений зайвими елементами. Перехід між теоретичним блоком, екзаменаційними тестами та автомобільним симулятором відбувається в один клік за допомогою панелі навігації в лівій стороні вікна програми. Модуль вибору країн працює швидко, під час перемикання прапорців, весь текстовий та тестовий контент оновлюється миттєво без потреби перезапускати застосунок.

Взаємодія з віртуальним автомобілем у вікні 2D-симулятора повністю перенесена на клавіатуру, що дозволяє зосередитися на дорожній обстановці:

1. Кнопки W, A, S, D або стрілки, відповідають за рух вперед, гальмування та повороти керма
2. Клавіша Пробіл дозволяє подати звуковий сигнал;
3. Кнопка R використовується для швидкого скидання і перезапуску поточного дорожнього сценарію, якщо учень потрапив у ДТП чи припустився критичної помилки
4. Клавіша ESC зупиняє роботу симулятора, дає команду на фоновий запис результатів у базу даних та повертає користувача до головного меню вибору режимів.

4.7Перспективи подальшого розвитку та масштабування системи

Незважаючи на те, що розроблений програмний комплекс SafeDrive повністю задовольняє вимоги технічного завдання та є функціонально завершеним продуктом, його модульна архітектура, заснована на патерні MVC, створює гнучкі можливості для подальшого вдосконалення та нарощування функціоналу. Масштабування системи може відбуватися у кількох стратегічних напрямах.

Першим та найбільш перспективним вектором розвитку є впровадження елементів штучного інтелекту в логіку поведінки віртуальних учасників дорожнього руху у модулі симуляції. На даному етапі трафік генерується на основі

заздалегідь прописаних алгоритмів. Використання методів машинного навчання дозволило б створити систему динамічного трафіку, де водії здатні самостійно приймати рішення, реагувати на непередбачувані дії користувача та імітувати реальні похибки людського фактора. Крім того, алгоритми глибинного навчання можуть використовуватися для аналізу індивідуального стилю керування віртуальним автомобілем. Це дасть змогу системі автоматично підлаштовувати рівень складності завдань, генеруючи саме ті дорожні ситуації, які викликають у користувача найбільші труднощі, тим самим реалізуючи концепцію повністю адаптивного персоналізованого навчання.

Другим напрямом є розширення бази даних ПДР для охоплення більшої кількості юрисдикцій. Завдяки гнучкому програмному адаптеру для динамічної підміни конфігураційних параметрів, додавання нових країн не потребує втручання в архітектуру вихідного коду застосунку. Достатньо лише сформувати відповідні інформаційні масиви з текстовим контентом, специфічними дорожніми знаками та конфігураціями тестів. Розширення географії продукту також передбачає впровадження повноцінної системи локалізації та інтернаціоналізації. Це дозволить швидко перекладати інтерфейс на різні мови, автоматично враховувати регіональні формати та метричні системи. Такий підхід відкриває широкі можливості для потенційної комерціалізації програмного забезпечення та його виходу на міжнародний ринок освітніх IT-продуктів.

Третім важливим аспектом є потенційна міграція фізичного рушія з двовимірного простору у тривимірний. Ізольованість компонента View дозволяє безболісно замінити рушій Pygame на потужнішу 3D-платформу з використанням Python-інтерфейсів. Хоча це об'єктивно підвищить системні вимоги до апаратного забезпечення комп'ютерів, перехід до 3D-графіки з імітацією вигляду «від першої особи» виведе реалістичність процесу відпрацювання м'язової пам'яті на принципово новий рівень. Логічним продовженням цього етапу стане програмна інтеграція апаратних маніпуляторів (ігрових кермів, педалей, коробок передач) та підтримка гарнітур віртуальної реальності. Це дозволить перетворити звичайний

десктопний застосунок на повноцінний апаратно-програмний тренажер для спеціалізованих класів автошкіл.

Перспективною є розробка клієнт-серверної архітектури для централізованого збору статистики та створення B2B-сегмента. Якщо застосунок буде впроваджено в офіційний навчальний процес закладів з підготовки водіїв, локальна база даних SQLite може бути доповнена модулем криптографічної синхронізації з хмарним сервером. Це дозволить викладачам через окремий вебпортал отримувати доступ до консолідованої аналітики успішності цілих груп студентів у реальному часі. Система зможе автоматично формувати звіти про готовність учнів до складання державного іспиту та виявляти прогалини у знаннях. У віддаленій перспективі така хмарна екосистема може бути інтегрована через відкриті API з державними реєстрами та базами даних Сервісних центрів МВС для автоматизованої передачі результатів внутрішнього тестування кандидатів у водії.

Ще одним стратегічним вектором розвитку є кросплатформна експансія та адаптація системи для мобільних пристроїв і вебсередовища. Хоча десктопний формат забезпечує максимальну продуктивність для розрахунку фізики в симуляторі, перенесення теоретичного та тестового модулів у формат мобільного застосунку значно розширило б доступність комплексу для широкої аудиторії. Користувачі отримали б змогу вивчати матеріали та проходити тестування у будь-якому зручному місці, синхронізуючи свій прогрес із десктопною версією через єдиний обліковий запис. Для реалізації цього завдання доцільним є перехід до мікросервісної архітектури, де розроблене ядро бази знань слугуватиме єдиним джерелом істини для всіх клієнтських платформ.

Окремо варто відзначити перспективи впровадження розширених механізмів гейміфікації та соціальної взаємодії. Наразі підсистема оцінювання працює в індивідуальному режимі, проте додавання рейтингових таблиць, складної системи досягнень та можливості проведення віртуальних змагань на знання ПДР серед учнів різних автошкіл здатне суттєво підвищити мотивацію до навчання

ВИСНОВКИ

Робота присвячена розробці автономного програмного комплексу “SafeDrive”, призначеного для автоматизації процесу вивчення Правил дорожнього руху та відпрацювання практичних навичок водіння у безпечному віртуальному середовищі. Основна мета проєкту - підвищення якості підготовки майбутніх водіїв шляхом інтеграції теоретичних знань із імітаційним 2D-моделюванням критичних дорожніх ситуацій.

1. Проведено дослідження архітектурних підходів до побудови навчальних систем та обґрунтовано вибір технологічного стеку. Це дозволило створити кросплатформену, повністю автономну систему, що відповідає сучасним вимогам до продуктивності графіки та надійності збереження даних.
2. Проаналізовано існуючі рішення конкурентів та виявлено гостру потребу в інструменті, який поєднує статичне тестування з динамічною фізичною симуляцією. На основі цього проведено проєктування архітектури за патерном MVC та побудовано систему UML-моделей, що описують логіку взаємодії компонентів.
3. Розроблено та реалізовано графічний інтерфейс застосунку з використанням фреймворку PyQt6. Впроваджено систему динамічного малювання дорожніх знаків через клас QPainter, що забезпечує векторну якість візуалізації та коректне відображення на моніторах із високою роздільною здатністю.
4. Реалізовано імітаційний модуль на базі рушія Pygame. Розроблено математичну модель фізики руху транспортного засобу, що враховує інерцію, тертя та гальмівний шлях. Програмовано 12 унікальних сценаріїв, які дозволяють користувачу відпрацьовувати реакції на складні дорожні обставини.
5. Спроєктовано та впроваджено систему мульти-країнності, яка забезпечує динамічну зміну законодавчої бази, мовних пакетів та візуальних стилів для України, Польщі та Німеччини. Це робить продукт універсальним для підготовки водіїв у межах Євросоюзу.

6. Реалізовано підсистему збереження даних на основі SQLite. Створено реляційну схему для адміністрування профілів, логування результатів тестування та відстеження прогресу користувача. Забезпечено захист персональних даних через використання криптографічного хешування паролів за стандартом SHA-256.
7. Проведено верифікацію системи, що охоплює модульне тестування алгоритмів фізики, інтеграційне тестування взаємодії PyQt6/Pygame та оцінку продуктивності забезпечуючи стабільні 60 FPS. Перевірено стійкість системи до помилкових дій користувача та валідність обробки колізій у симуляторі.
8. Система може бути використана в автошколах як допоміжний навчальний засіб, а також для індивідуальної підготовки водіїв, де потрібне глибоке розуміння ПДР та візуалізація фізики дорожнього руху без ризику реального ДТП.

Результати розробки апробовані у вигляді функціонального прототипу MVP, який демонструє повний цикл навчання: від реєстрації та вивчення теорії до проходження комплексного іспиту та практичної симуляції руху. Проєкт готовий до розгортання та подальшого масштабування в напрямку 3D-візуалізації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Python Software Foundation. Python 3.12 Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/.
2. PyQt6 Class Reference. Riverbank Computing Limited [Електронний ресурс]. – Режим доступу: https://www.riverbankcomputing.com/static/Docs/PyQt6/.
3. Pygame-ce (Community Edition) Documentation [Електронний ресурс]. – Режим доступу: https://pyga.me/docs/.
4. SQLite Documentation. SQL as Understood By SQLite [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/docs.html.
5. Python Standard Library. math - Mathematical functions [Електронний ресурс]. - Режим доступу: https://docs.python.org/3/library/math.html.
6. Python Standard Library. hashlib - Secure hashes and message digests [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/hashlib.html.
7. Python Standard Library. sqlite3 - DB-API 2.0 interface for SQLite [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/sqlite3.html.
8. Qt Group. Qt Graphics View Framework (QPainter) [Електронний ресурс]. – Режим доступу: https://doc.qt.io/qt-6/graphicsview.html.
9. Unified Modeling Language (UML) Specification. Object Management Group [Електронний ресурс]. – Режим доступу: https://www.omg.org/spec/UML/.
10. Правила дорожнього руху України. Постанова КМУ № 1306 [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/1306-2001-п.
11. ДСТУ 4100:2021 Безпека дорожнього руху. Знаки дорожні [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/rada/show/v0473854-21.
12. ISO/IEC 25010:2011 Systems and Software Engineering - Quality Model.
13. Grady Booch, James Rumbaugh, Ivar Jacobson. The Unified Modeling Language User Guide. 2nd Edition. Addison-Wesley, 2005. – p. 239–248.
14. Martin Fowler. Patterns of Enterprise Application Architecture (MVC Pattern). Addison-Wesley, 2002. – p. 85–120.

15. Robert C. Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. – p. 135–150.
16. Ian Sommerville. Software Engineering. 10th Edition. Pearson, 2015. – p. 45–70.
17. Glenford J. Myers. The Art of Software Testing. 3rd Edition. Wiley, 2011. – p. 41–65.
18. IEEE Standard 830-1998. IEEE Recommended Practice for Software Requirements Specifications.

ДОДАТОК А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Технологій цифрового розвитку

Програмне забезпечення SafeDrive для інтерактивного вивчення правил дорожнього руху з елементами симуляції

Виконав студент 4 курсу
Групи ТЦР-42
Новачук Ігор Сергійович
Керівник роботи
старший викладач кафедри ТЦР Читулян Вадим Олегович

Київ-2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – підвищення ефективності та якості вивчення правил дорожнього руху шляхом створення інтерактивного програмного комплексу з можливостями імітаційного 2D-моделювання нестандартних дорожніх ситуацій.

Об'єкт дослідження – вивчення правил дорожнього руху.

Предмет дослідження – методи, алгоритми та програмне забезпечення для створення багатомодульного десктопного застосунку з використанням технологій Python, PyQt6 та Pygame.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Провести аналіз існуючих програмних рішень для вивчення ПДР та імітаційних автосимуляторів, виявити їхні технічні та методологічні недоліки.

2. Обгрунтувати вибір технологічного стеку та спроєктувати модульну архітектуру десктопної платформи на засадах шаблону MVC.

3. Розробити підсистему імітаційного моделювання: реалізувати 2D-симулятор із фізичною моделлю руху транспортного засобу та сценаріями критичних дорожніх ситуацій.

4. Спроєктувати та сформувати локальну реляційну базу даних для автономного збереження профілів користувачів, статистики проходження тестів та логів сесій.

5. Реалізувати графічний інтерфейс із системою автоматизованого тестування, інтерактивним теоретичним довідником та модулем векторного рендерингу дорожніх знаків.

6. Провести комплексне тестування функціональних можливостей, перевірку механізмів криптографічного захисту паролів та тестування продуктивності ігрового рушія.

АНАЛІЗ АНАЛОГІВ

Критерій порівняння	Green Way	Vodiy.ua	Моб. застосунки	3D-симулятори	Розроблена платформа
Теоретична база	Наявна	Наявна	Наявна	Відсутня	**Наявна**
Модуль тестування	Наявний	Наявний	Наявний	Відсутній	**Наявний**
Відстеження прогресу та статистика	Наявне	Наявне	Наявне	Відсутнє	**Наявне**
Інтерактивна симуляція ситуацій	Відсутня	Відсутня	Відсутня	Наявна	**Наявна**
Автономна робота	Відсутня	Відсутня	Часткова	Повна	**Повна**
Підтримка ПДР кількох країн	Відсутня	Відсутня	Відсутня	Відсутня	**Наявна**
Системні вимоги до пристрою	Низькі	Низькі	Не застосовується	Високі	**Низькі**

ВИМОГИ ДО ПЛАТФОРМИ

Функціональні вимоги

1. Реєстрація та авторизація користувачів із безпечним збереженням облікових даних шляхом криптографічного хешування паролів.
2. Забезпечення доступу до інтегрованої бази знань ПДР із можливістю перемикання між правилами України, Польщі та Німеччини
3. Програмний векторний рендеринг повного каталогу дорожніх знаків для збереження їхньої чіткості при масштабуванні.
4. Реалізація модуля тестування з генерацією питань за категоріями, підрахунком балів та веденням історії результатів.
5. Інтеграція двовимірного симулятора водіння з фізичною моделлю руху, зміною погодних умов та сценаріями критичних ситуацій.
6. Автоматичне збереження статистики пройдених тестів та результатів симуляцій у базі даних.

Нефункціональні вимоги

1. Система повинна працювати повністю автономно без підключення до Інтернету з використанням локальної бази даних
2. Наявність механізму резервного копіювання та відновлення даних на випадок системних збоїв.
3. Оптимізація ігрового рушія для стабільної та плавної роботи симулятора на комп'ютерах початкового рівня.
4. Забезпечення кросплатформності для можливості запуску застосунку на різних операційних системах.
5. Створення адаптивного графічного інтерфейсу з темною темою оформлення для зниження навантаження на зір користувача.

Програмні засоби реалізації

Visual Studio Code
Є основним середовищем розробки. Саме в цьому редакторі писався та тестувався весь вихідний код програми.

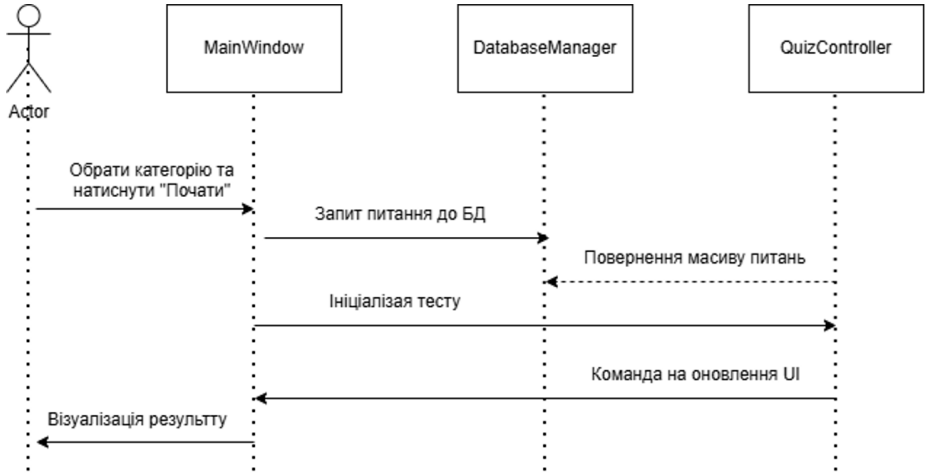
Python
Є базовою мовою програмування проєкту. Вона забезпечує роботу всієї внутрішньої логіки та об'єднує інтерфейс, базу даних і симулятор в одну систему.

PyQt6
Використовується для створення графічного інтерфейсу користувача. Завдяки йому працюють усі вікна, кнопки, меню, модулі тестування та відображаються дорожні знаки.

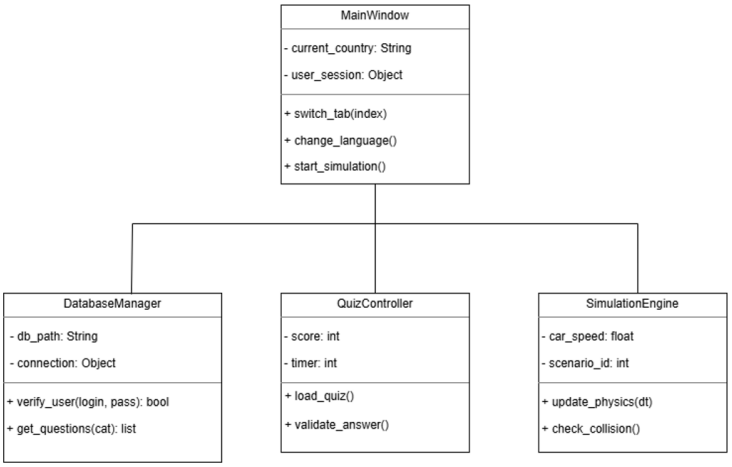
Pygame
відповідає за імітаційне моделювання. На цій бібліотеці реалізовано сам практичний 2D симулятор водіння, розрахунок фізики машини та обробку дорожніх ситуацій.

SQLite
виконує роль локальної бази даних. Вона забезпечує автономну роботу програми, зберігаючи облікові записи користувачів та статистику їхнього навчання без доступу до інтернету.

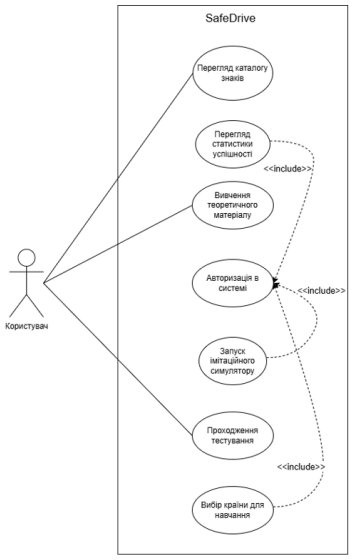
ДІАГРАМА ПОСЛІДОВНОСТІ ПРОЦЕСУ ІНІЦІАЛІЗАЦІЇ ТЕСТУВАННЯ



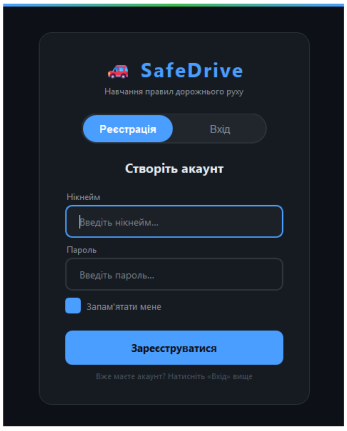
ДІАГРАМА КЛАСІВ АРХІТЕКТУРИ СИСТЕМИ



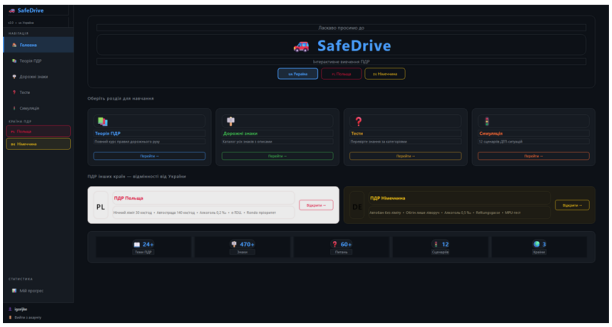
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ СИСТЕМИ



ЕКРАННІ ФОРМИ

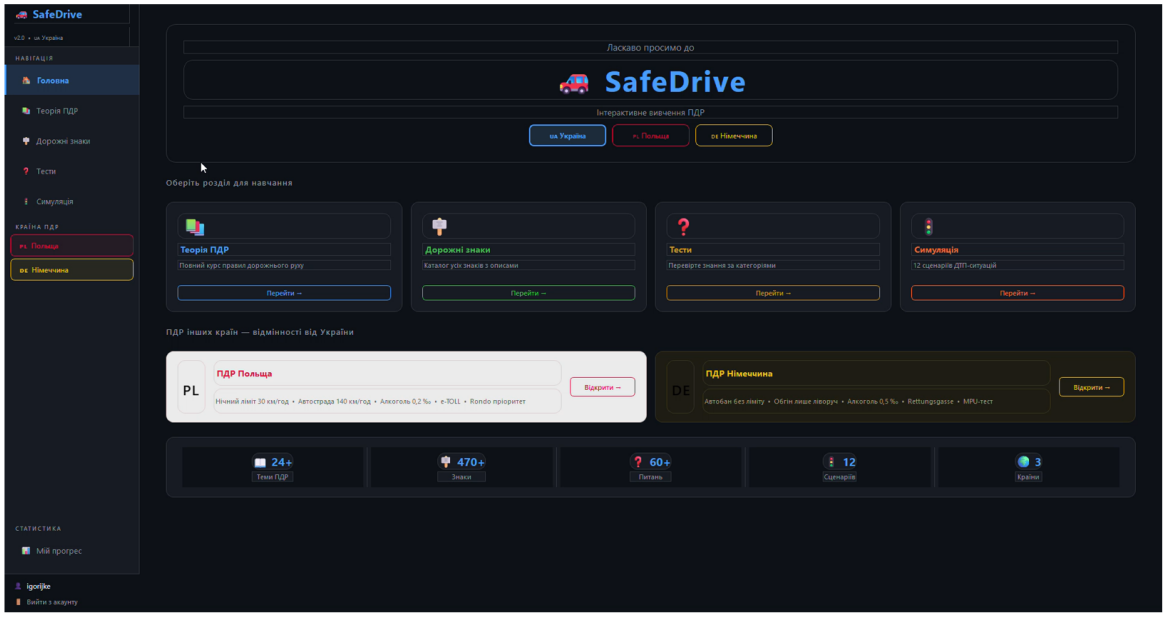


Вікно реєстрації



Вигляд вікна головного меню

Відео-демонстрація роботи додатку



АПРОБАЦІЯ

1. Новачук І.С., Читулян В.О. Розроблення програмного забезпечення Safe Drive для інтерактивного вивчення правил дорожнього руху. // VII Всеукраїнської науковотехнічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». Збірник тез. К.: ДУІКТ, 2026, (подано до друку)
2. Новачук І.С., Читулян В.О. Розроблення програмного забезпечення Safe Drive для інтерактивного вивчення правил дорожнього руху з елементами симуляції. // Всеукраїнської науково-технічної конференції «Технології цифрового розвитку: програмні системи та децентралізація». Збірник тез. К.: ДУІКТ, 2026, (подано до друку)

ВИСНОВКИ

1. Проведено аналіз існуючих рішень для підготовки водіїв. Виявлено їхні головні недоліки: постійна залежність від інтернету та відсутність практичних симуляцій.
2. Розроблено архітектуру програми на мові Python. Забезпечено чіткий розподіл завдань між інтерфейсом, імітаційним рушієм та базою знань.
3. Впроваджено механізми безпеки. Використано локальну базу даних із захищеними паролями та резервним копіюванням для повної автономності роботи.
4. Створено єдине середовище. Програма поєднує вивчення теорії, систему тестування та 2D-симулятор із 12 сценаріями дорожніх ситуацій.
5. Підтверджено інженерну ефективність. Тестування довело стабільну роботу симулятора при 60 FPS та оптимізоване споживання ресурсів комп'ютера.
6. Доведено практичну цінність розробки. Застосунок забезпечує ефективне середовище для вивчення ПДР та відпрацювання практичних навичок.