Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *«Розробка застосунку для визначення сонливості водія за допомогою штучного інтелекту»*

Виконав: студент 4 курсу, групи СПс-41
спеціальності 121 «інженерія програмного забезпечення»
(шифр і назва спеціальності)

(підпис) Целінь А.Р. (прізвище та ініціали)

Керівник (підпис) Цуприк Г.Б. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *програмної інженерії*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
« 6 » квітня 2026 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *121 «Інженерія програмного забезпечення»*
(шифр і назва спеціальності)

студенту Целіню Артуру Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *«Розробка застосунку для визначення сонливості водія за допомогою штучного інтелекту»*

Керівник роботи Цуприк Галина Богданівна к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *6* » *квітня* 2026 року № *4/9-172*

2. Термін подання студентом завершеної роботи *22.06.2026*

3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
1. Аналіз вимог до програмної системи
2. Проєктування та розробка програмної системи
3. Тестування, впровадження та підтримка
4. Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Слайди презентації

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання *06.04.2026р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
	Розробка технічного завдання	*06.04 – 12.04*	
	Робота над першим розділом «Аналіз вимог до програмної системи»	*13.04 – 24.04*	
	Робота над другим розділом «Проєктування та розробка програмної системи»	*25.04 – 04.05*	
	Робота над третім розділом «Тестування, впровадження та підтримка»	*05.05 – 18.05*	
	Робота над четвертим розділом «Безпека життєдіяльності, основи охорони праці»	*19.05 – 24.05*	
	Оформлення пояснювальної записки та графічного матеріалу	*25.05 – 07.06*	
	Перевірка на академічний плагіат, перевірка керівником та консультантами	*08.06 – 14.06*	
	Попередній захист кваліфікаційної роботи бакалавра	*15.06 – 21.06*	
	Захист кваліфікаційної роботи бакалавра		

Студент _____ _____
(підпис) (прізвище та ініціали)

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

АНОТАЦІЯ

Розробка застосунку для визначення сонливості водія за допомогою штучного інтелекту // Кваліфікаційна робота освітнього рівня «Бакалавр» // Целінь Артур Романович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії група СПс-41 // Тернопіль, 2026 // С. 68, рис. – 12, табл. – 5, додат. – 2, бібліогр. – 38.

Ключові слова: сонливість водія, комп'ютерний зір, нейронна мережа, OpenCV, Keras, моніторинг стану водія, безпека дорожнього руху, машинне навчання.

У кваліфікаційній роботі розроблено програмну систему визначення сонливості водія, яка забезпечує автоматичний моніторинг його стану в режимі реального часу на основі аналізу відеопотоку з вебкамери. Система використовує методи комп'ютерного зору та нейронні мережі для розпізнавання стану очей водія й формування попереджувального сигналу у випадку виявлення ознак сонливості.

У першому розділі проведено аналіз предметної області, розглянуто основні причини виникнення дорожньо-транспортних пригод, пов'язаних із втомою водіїв, досліджено сучасні технології моніторингу стану людини та виконано аналіз існуючих програмних аналогів.

У другому розділі виконано проєктування архітектури системи, розроблено структуру бази даних Drownes, побудовано UML-діаграми та обґрунтовано вибір програмних засобів реалізації. Описано програмну реалізацію модулів виявлення обличчя та очей, класифікації стану очей за допомогою згорткової нейронної мережі, формування сигналу тривоги. Також розроблено графічний інтерфейс користувача.

У третьому розділі проведено тестування функціональних можливостей програмного продукту, визначено системні вимоги до його використання, описано процес розгортання та виконано верифікацію розробленої системи.

У четвертому розділі розглянуто питання охорони праці та безпеки життєдіяльності під час роботи з комп’ютерними системами.

Об’єкт дослідження – процес контролю стану водія під час керування транспортним засобом.

Предмет дослідження – методи та програмні засоби виявлення сонливості водія на основі технологій комп’ютерного зору та нейронних мереж.

Результатом роботи є програмна система, здатна в автоматичному режимі виявляти ознаки сонливості водія, своєчасно повідомляти про потенційно небезпечний стан та зберігати результати роботи для подальшого аналізу.

ABSTRACT

Development of an app to detect driver drowsiness using artificial intelligence // Bachelor's Qualification Thesis // Artur Romanovych Tselin // Ivan Puluj Ternopil National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, Group SPs-41 // Ternopil, 2026 // P. 68, Fig. – 12, Tables – 5, Appendices – 2, References – 38.

Keywords: driver drowsiness, computer vision, neural network, OpenCV, Keras, driver monitoring, road safety, machine learning.This bachelor's qualification thesis is devoted to the development of a software system for driver drowsiness detection that provides automatic real-time monitoring of a driver's condition based on the analysis of video streams obtained from a webcam. The system utilizes computer vision techniques and neural networks to recognize the driver's eye state and generate warning signals when signs of drowsiness are detected.

The first chapter presents an analysis of the subject area, examines the main causes of road traffic accidents related to driver fatigue, investigates modern human condition monitoring technologies, and reviews existing software solutions in this field.

The second chapter describes the design of the system architecture, the development of the Drownes database structure, the creation of UML diagrams, and the justification of the selected implementation technologies. It also covers the software implementation of face and eye detection modules, eye-state classification using a convolutional neural network, and the generation of warning signals. In addition, a graphical user interface was developed.

The third chapter focuses on testing the functional capabilities of the software product, defining the system requirements for its operation, describing the deployment process, and verifying the developed system.

The fourth chapter addresses occupational health and safety issues as well as life safety considerations related to working with computer systems.Object of research – the process of monitoring a driver's condition while operating a vehicle.Subject of research –

methods and software tools for driver drowsiness detection based on computer vision and neural network technologies.

The result of the work is a software system capable of automatically detecting signs of driver drowsiness, providing timely warnings about potentially dangerous conditions, and storing operational results for further analysis.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект.

НМ – нейронна мережа.

ЗНМ – згорткова нейронна мережа.

БД – база даних.

СУБД – система управління базами даних.

MS SQL Server – система керування базами даних Microsoft SQL Server.

UML (Unified Modeling Language) – уніфікована мова моделювання.

ER-діаграма (Entity-Relationship Diagram) – діаграма сутність–зв’язок.

CNN (Convolutional Neural Network) – згорткова нейронна мережа.

OpenCV (Open Source Computer Vision Library) – бібліотека комп’ютерного зору з відкритим кодом.

Keras – програмний інтерфейс для створення та навчання нейронних мереж.

Kivy – фреймворк для розробки графічних інтерфейсів користувача.

ODBC (Open Database Connectivity) – інтерфейс доступу до баз даних.

SQL (Structured Query Language) – мова структурованих запитів до баз даних.

ЗМІСТ

ВСТУП 8
1. АНАЛІЗ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ 10
1.1 Аналіз предметної області 10
1.2 Постановка завдання та цілей 12
1.3 Пошук акторів та варіантів використання 13
1.4 Опис ключових варіантів використання 15
2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ 18
2.1 Вибір процесу розробки 18
2.2 Проєктування архітектури системи 19
2.3 Побудова схеми бази даних 22
2.4 Побудова UML-діаграм класів 28
2.5 Вибір мови та середовища розробки 30
2.6 Реалізація основних класів та методів 33
2.7 Розробка інтерфейсу користувача 37
3. ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА ПІДТРИМКА 43
3.1 Тестування програмної системи 43
3.1.1 Види та план тестування 43
3.1.2 Розробка тестових сценаріїв 44
3.2 Розгортання програмної системи та системні вимоги 46
3.3 Верифікація програмної системи 47
4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ 49
4.1 Безпека життєдіяльності 49
4.2 Основи охорони праці 51

ВИСНОВКИ 53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 55
ДОДАТКИ 59
ДОДАТОК А 60
ДОДАТОК Б 63

ВСТУП

Актуальність теми дослідження зумовлена необхідністю підвищення безпеки дорожнього руху за рахунок використання сучасних технологій штучного інтелекту. Однією з ключових причин дорожньо-транспортних пригод є сонливість водія, що знижує концентрацію та швидкість реакції. Це обумовлює потребу у створенні програмного забезпечення, здатного своєчасно виявляти ознаки втоми та попереджати водія.

Метою роботи є розробка програмного застосунку на основі штучного інтелекту, який забезпечує визначення сонливості водія з використанням методів комп'ютерного зору та формує аналітичні висновки для підвищення рівня безпеки.

Для досягнення поставленої мети реалізовано систему, що обробляє відеопотік у реальному часі та аналізує поведінкові характеристики водія. Основна увага приділяється розпізнаванню ознак сонливості, таких як частота моргання, тривалість закриття очей.

Клієнтська частина застосунку забезпечує зручну взаємодію користувача із системою та відображення результатів аналізу. Інтерфейс дозволяє в режимі реального часу отримувати повідомлення про стан водія та потенційні ризики.

Основним завданням системи є проведення комплексного аналізу стану водія. Використання алгоритмів комп'ютерного зору та штучного інтелекту дозволяє оцінювати рівень сонливості на основі візуальних даних, що забезпечує високу точність і швидкість обробки інформації.

Важливим компонентом системи є модуль автоматичного формування висновків, який аналізує отримані дані та генерує рекомендації щодо необхідності відпочинку або зупинки.

У майбутньому функціональність системи може бути розширена за рахунок інтеграції з бортовими системами автомобіля та мобільними пристроями, що дозволить реалізувати автоматичне попередження водія або активацію допоміжних механізмів безпеки. Також можливе вдосконалення алгоритмів штучного інтелекту

для підвищення точності визначення сонливості, зокрема шляхом використання глибокого навчання та аналізу додаткових поведінкових і фізіологічних показників.

Висновки. У результаті виконання роботи створено програмний продукт для визначення сонливості водія з використанням штучного інтелекту та комп’ютерного зору. Його застосування сприятиме зниженню ризику аварійних ситуацій, підвищенню безпеки дорожнього руху та має потенціал до подальшого розвитку й удосконалення.

Основні положення та результати кваліфікаційної роботи доповідалися та обговорювалися на науково-технічній конференції Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2026 р.).

1. АНАЛІЗ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

У цьому розділі проводиться комплексне дослідження предметної області, пов’язаної з методами моніторингу психофізіологічного стану людини та автоматичного виявлення втоми під час керування транспортними засобами [1]. На основі аналізу наявних ринкових аналогів та існуючих систем асистування водієві (ADAS) [2] окреслено ключові проблеми, технологічні розриви та обмеження сучасних підходів, що зумовлюють необхідність розробки ефективного застосунку на базі штучного інтелекту для визначення сонливості в реальному часі.

З метою формалізації вимог до інтелектуального програмного продукту здійснюється ідентифікація майбутніх користувачів (водіїв та операторів автопарків), а також моделювання їхньої взаємодії з системою шляхом визначення та детального опису ключових варіантів використання (Use Cases), включаючи процеси калібрування камери, аналізу відеопотоку та сповіщення про небезпеку.

1.1 Аналіз предметної області

Сучасні умови інтенсивного розвитку автомобільного транспорту характеризуються значною кількістю людей, які щоденно здійснюють поїздки як у межах міст, так і на великі відстані. Особливо це стосується професійних водіїв – таксистів, водіїв вантажного транспорту та автобусів, а також осіб, що здійснюють тривалі подорожі. У таких умовах одним із ключових факторів ризику залишається втома та недосипання, які суттєво підвищують ймовірність виникнення дорожньо-транспортних пригод. Дослідження показують, що значна частина аварійних ситуацій на дорогах пов’язана саме зі зниженням уваги водія через сонливість. Сон як фізіологічний процес має циклічний характер, а рівень неспання людини змінюється від активного стану до стану дрімоти, який є критично небезпечним під час керування транспортним засобом. При цьому стан втоми та засинання є індивідуальним для кожної людини, однак існують загальні поведінкові та

фізіологічні ознаки, що дозволяють ідентифікувати зниження концентрації водіях [3].

До основних ознак розвитку втоми належать зниження уважності, підвищена дратівливість, порушення сприйняття навколишньої ситуації, коли водій частково або повністю втрачає здатність відтворити події дорожнього руху, а також поява фізичного дискомфорту, зокрема слабкості в кінцівках і відчуття важкості в потиличній частині голови. У таких станах керування транспортним засобом стає вкрай небезпечним, і єдиним ефективним способом запобігання негативним наслідкам є негайна зупинка автомобіля та короткочасний відпочинок тривалістю щонайменше 15–20 хвилин. Інші поширені методи боротьби зі сонливістю, такі як вживання кофеїну, гучна музика або провітрювання салону, не забезпечують стабільного усунення стану втоми. Для об’єктивного визначення рівня втоми можуть використовуватися прості фізіологічні показники. Зокрема, у нормальному стані частота моргання становить приблизно 15 разів на хвилину, а тривалість закриття очей є мінімальною. У разі розвитку втоми частота моргання може зростати до 60 разів на хвилину, а тривалість закриття очей збільшується, що призводить до значного сумарного часу, протягом якого очі залишаються закритими. Також суттєво зростає час реакції людини, який у нормальному стані становить приблизно 0,23–0,3 секунди, тоді як при втомі він значно погіршується. Це безпосередньо впливає на збільшення гальмівного шляху та зниження загального рівня безпеки дорожнього руху.

Виявлення сонливості водія розглядається як важлива складова сучасних систем безпеки, спрямованих на запобігання аваріям, спричиненим втратою концентрації під час керування транспортним засобом. Такі системи здійснюють безперервний моніторинг стану водія та у разі виявлення ознак втоми формують попереджувальні сигнали, що рекомендують зупинку та відпочинок. Основою роботи подібних рішень є технології комп’ютерного зору, які використовують відеопотік із вбудованих або мобільних камер для аналізу обличчя водія та його поведінкових характеристик.

Сучасні системи допомоги водієві (Advanced Driver Assistance Systems – ADAS) спрямовані на підвищення рівня безпеки дорожнього руху шляхом запобігання дорожньо-транспортним пригодам або мінімізації їх наслідків. Вони формують попереджувальні сигнали високого пріоритету, які стимулюють водія до своєчасного реагування в потенційно небезпечних ситуаціях, пов’язаних із ризиком травмування або загрози життю.

До основних технологічних компонентів ADAS-систем належать:

- система розпізнавання дорожньої розмітки, що дозволяє визначати межі смуг руху та контролювати положення транспортного засобу в межах смуги;
- система контролю виходу зі смуги руху, яка аналізує траєкторію руху автомобіля та формує попередження у разі її відхилення від допустимих меж;
- система виявлення транспортних засобів, що базується на алгоритмах комп’ютерного зору та забезпечує розпізнавання інших учасників дорожнього руху в різних умовах освітлення;
- система попередження про можливе зіткнення, яка аналізує дистанцію до об’єктів попереду та прогнозує ризик аварійної ситуації [2].

Узагальнено ADAS-рішення можуть бути поділені на дві основні категорії: мобільні програмні застосунки, що встановлюються на смартфони та використовують їхні камери для аналізу стану водія, а також апаратно-програмні комплекси, які інтегруються безпосередньо в конструкцію транспортного засобу або встановлюються додатково у вигляді зовнішніх камер, сенсорів і обчислювальних модулів.

У цьому контексті актуальною є розробка застосунку для визначення сонливості водія із застосуванням методів штучного інтелекту, який дозволить автоматизувати процес аналізу стану водія та підвищити загальний рівень безпеки дорожнього руху.

1.2 Постановка завдання та цілей

Сонливість водія є однією з найчастіших причин аварій. Якщо вчасно визначити цей стан, то це дозволить запобігти великій кількості аварій. З цієї причини було вирішено створити проєкт, за допомогою якого можна було б розпізнавати сонливість водія.

Насамперед потрібно вивчити поняття нейронної мережі та її призначення у розробці проєктів машинного навчання, провести дослідження характеристик сигналів та сучасних методів їх класифікації [4]. Проаналізувати технології розпізнавання сонливості людей. Розробити застосунок із розпізнавання сонливості людини за допомогою зображення із камери засобами машинного навчання.

Розглянути ключові характеристики для розпізнавання сонливості людини за зображенням. Для цього використати відкриту нейромережеву бібліотеку Keras [5] мови програмування Python, яка є надбудовою над фреймворком TensorFlow [6].

Система визначення сонливості водія повинна задовольняти наступні вимоги:·

- програма має бути простою і надійною у використанні;·
- вона повинна мати можливість інтеграції в наявну систему транспорту;·
- повинна мати зручний та інтуїтивно зрозумілий інтерфейс;·
- повинна бути невимогливою до програмно-апаратних засобів;·
- надавати можливість реєстрації окремих водіїв, а також підприємств та їх менеджерів;·
- реалізувати різні права доступу для користувачів системи;·
- надавати можливість перегляду статистики водіям та менеджерам підприємства.

1.3 Пошук акторів та варіантів використання

Проєктування програмного забезпечення, орієнтованого на аналіз стану людини в реальному часі, потребує чіткого визначення ролей учасників системи та формалізації сценаріїв їхньої взаємодії. У межах розробки застосунку для визначення сонливості водія із використанням технологій штучного інтелекту

доцільним є застосування UML-діаграми варіантів використання, яка дозволяє структуровано відобразити функціональні вимоги та межі системи [7].

Першим етапом побудови моделі є ідентифікація акторів, тобто зовнішніх сутностей, що взаємодіють із системою або використовують її результати [8].У межах даної програмної системи визначено таких основних акторів:

1. Водій (кінцевий користувач системи) є основним суб'єктом взаємодії із застосунком. Він використовує систему під час керування транспортним засобом. Його основна функція полягає в отриманні своєчасних повідомлень щодо поточного стану уваги та рівня сонливості, а також попереджень про необхідність зупинки транспортного засобу для відпочинку.

2. Модуль штучного інтелекту (AI-обробка) є програмним компонентом, що реалізує автоматизований аналіз відеопотоку. Його функціональність включає обробку зображень обличчя водія, визначення ключових ознак втоми та формування числової оцінки рівня сонливості на основі алгоритмів комп'ютерного зору та машинного навчання.

На основі аналізу функціональних вимог було визначено основні варіанти використання системи:

- отримання відеопотоку з камери пристрою;
- попередня обробка зображення та виділення обличчя водія;
- аналіз поведінкових ознак (частота моргання, положення очей, нахил голови);
- визначення рівня сонливості на основі AI-моделі;
- формування попереджувального сигналу у разі перевищення порогового значення втоми;
- відображення результатів аналізу в інтерфейсі користувача.

Додатково система передбачає механізм інформування водія про критичний стан, що спрямований на запобігання потенційно аварійним ситуаціям.

Таким чином, сформована модель варіантів використання дозволяє чітко визначити функціональні межі системи та є основою для подальшого проєктування

архітектури застосунку, алгоритмів аналізу зображень та логіки взаємодії користувача із системою [9].

Для наочності структури взаємодії розроблено UML-діаграму варіантів використання, наведену на рисунку 1.1.

Рисунок 1.1 – Діаграма варіантів використання

1.4 Опис ключових варіантів використання

Для формалізації процесу функціонування програмної системи визначення сонливості водія доцільно використовувати діаграми послідовності UML, які дозволяють відобразити часову взаємодію між основними компонентами системи та описати порядок виконання операцій у межах одного сценарію використання [10].Такі діаграми забезпечують можливість деталізації алгоритму роботи системи від моменту ініціації дії користувачем до формування фінального результату, що є особливо важливим для систем, побудованих на основі методів комп’ютерного зору та штучного інтелекту.

Цей прецедент є основним сценарієм для актора Користувач, що забезпечує виконання ключової функції системи – автоматичне виявлення ознак сонливості за допомогою аналізу зображення обличчя та своєчасне сповіщення користувача про потенційно небезпечний стан.

Актор: Користувач.

Передумови: Мобільний додаток встановлений на пристрої та має дозвіл на використання камери. База даних доступна для обробки результатів аналізу.

Післяумови: Виконано аналіз зображення обличчя користувача. У разі виявлення ознак сонливості сформовано попередження та подано звуковий сигнал.

Основний сценарій виконання:

1. Користувач запускає додаток.

2. Система отримує доступ до камери пристрою після підтвердження відповідних дозволів.

3. Користувач здійснює фотографування або система автоматично захоплює поточне зображення.

4. Модуль комп’ютерного зору виконує виявлення обличчя на отриманому зображенні.

5. Дані про обличчя передаються до модуля аналізу сонливості, який здійснює оцінку стану користувача за визначеними ознаками (ступінь відкриття очей, положення голови, частота моргання тощо).

6. Результат аналізу передається до бази даних або модуля прийняття рішень, де фіксується факт виявлення сонливості.

7. Система генерує та відтворює звуковий сигнал для попередження користувача про небезпечний стан.

Таким чином, розглянута послідовність дій відображає повний цикл обробки даних у системі та демонструє взаємодію між користувачем, пристроєм та аналітичним модулем штучного інтелекту.

На рисунку 1.2 наведено діаграму послідовності «Виявлення сонливості користувача»

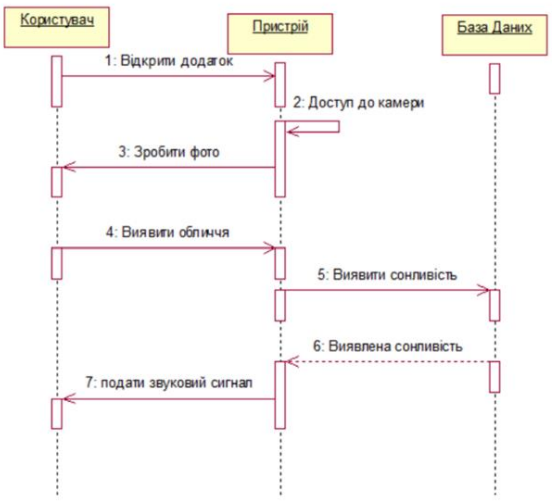

Рисунок 1.2 – Діаграма послідовності користувача

У цьому розділі було проведено аналіз предметної області, пов’язаної з виявленням сонливості водія, та обґрунтовано актуальність використання технологій штучного інтелекту для підвищення безпеки дорожнього руху. Визначено основні причини та ознаки сонливості, а також розглянуто сучасні підходи до моніторингу стану водія.

Сформульовано мету та завдання розробки програмного застосунку, визначено основних акторів системи та описано ключові варіанти використання. Для формалізації функціональних вимог побудовано UML-діаграми варіантів використання та послідовності, які відображають процес взаємодії користувача із системою та алгоритм виявлення сонливості.

Отримані результати є основою для подальшого проєктування архітектури застосунку та реалізації алгоритмів аналізу зображень.

2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

У розділі розглянуто процес проєктування та програмної реалізації системи визначення сонливості водія. Описано вибір підходу до розробки, побудову архітектури застосунку, реалізацію модулів комп'ютерного зору та штучного інтелекту, а також обґрунтовано вибір використаних технологій. Крім того, наведено особливості створення користувацького інтерфейсу та механізму формування попереджувальних повідомлень.

2.1 Вибір процесу розробки

Створення системи визначення сонливості водія є комплексним завданням, яке поєднує розробку програмного забезпечення, використання методів комп'ютерного зору та застосування моделей машинного навчання. Особливість таких проєктів полягає в тому, що кінцевий результат значною мірою залежить від якості навчання нейронної мережі та ефективності алгоритмів обробки зображень. Тому під час розробки виникає потреба неодноразово повертатися до попередніх етапів, коригувати параметри моделей та перевіряти отримані результати.

З урахуванням цих особливостей для реалізації системи було обрано. Її застосування дозволяє організувати роботу у вигляді послідовності циклів, у межах яких відбувається створення окремих функціональних компонентів, їх тестування та подальше вдосконалення. Такий підхід є доцільним для інтелектуальних систем, оскільки дає можливість поступово покращувати якість роботи алгоритмів на основі результатів експериментальних досліджень [11].

На початковому етапі було проведено аналіз предметної області та дослідження існуючих підходів до виявлення сонливості водіїв. Особливу увагу приділено методам комп'ютерного зору, які використовуються для аналізу стану людини за відеозображенням [12]. Після цього було обрано програмні засоби та бібліотеки, необхідні для реалізації проєкту.

Наступним кроком стала розробка механізму отримання зображень із камери та їх попередньої обробки. На цьому етапі реалізовано процедури виявлення обличчя користувача та підготовки даних для подальшого аналізу. Отримані результати використовувалися для перевірки коректності роботи алгоритмів і внесення необхідних змін до програмного коду.

Подальша робота була спрямована на створення та навчання моделі штучного інтелекту. У процесі навчання виконувалося багаторазове тестування нейронної мережі, оцінювання точності класифікації та підбір параметрів, які забезпечували найкращі результати розпізнавання ознак сонливості [13].

Після завершення розробки основних алгоритмів було виконано інтеграцію всіх компонентів у єдину програмну систему. На цьому етапі реалізовано користувацький інтерфейс, механізм формування попереджень та взаємодію між модулем комп’ютерного зору й модулем штучного інтелекту. Завершальним кроком стало комплексне тестування застосунку в різних умовах роботи.

Отже, використання ітераційної моделі дозволило забезпечити поетапне вдосконалення системи на всіх стадіях розробки та створити програмний продукт, здатний здійснювати автоматичне визначення сонливості водія в режимі реального часу.

2.2 Проєктування архітектури системи

Архітектура програмної системи є одним із ключових етапів проєктування, оскільки визначає структуру програмного забезпечення, взаємозв’язки між його компонентами та механізми обробки даних. Для систем реального часу, що використовують технології комп’ютерного зору та штучного інтелекту, особливо важливим є забезпечення ефективної взаємодії між модулями отримання даних, їх аналізу та формування результатів. Тому під час проєктування системи визначення сонливості водія основна увага приділялася створенню архітектури, здатної забезпечити безперервний аналіз відеопотоку та оперативне реагування на виявлення ознак втоми.

З урахуванням функціональних вимог, сформованих у першому розділі, для реалізації застосунку було обрано модульну багаторівневу архітектуру. Такий підхід передбачає розподіл програмного забезпечення на окремі функціональні компоненти, кожен із яких відповідає за виконання визначених завдань. Модульна структура дозволяє спростити процес розробки та тестування системи, а також забезпечує можливість подальшої модернізації окремих компонентів без суттєвого впливу на роботу всієї системи [14].

Основною перевагою обраного підходу є незалежність функціональних модулів. Зокрема, зміна або вдосконалення моделі машинного навчання не потребує модифікації інтерфейсу користувача чи механізму взаємодії з базою даних. Аналогічно, оновлення підсистеми зберігання даних не впливає на роботу алгоритмів комп’ютерного зору. Такий розподіл відповідальності дозволяє підвищити надійність системи та полегшує її супровід.У розробленій системі можна виділити декілька основних компонентів: модуль користувацького інтерфейсу, модуль керування застосунком, модуль комп’ютерного зору, модуль штучного інтелекту, модуль зберігання даних та модуль сповіщення користувача [15]. Загальна структура взаємодії між компонентами наведена на рисунку 2.1.

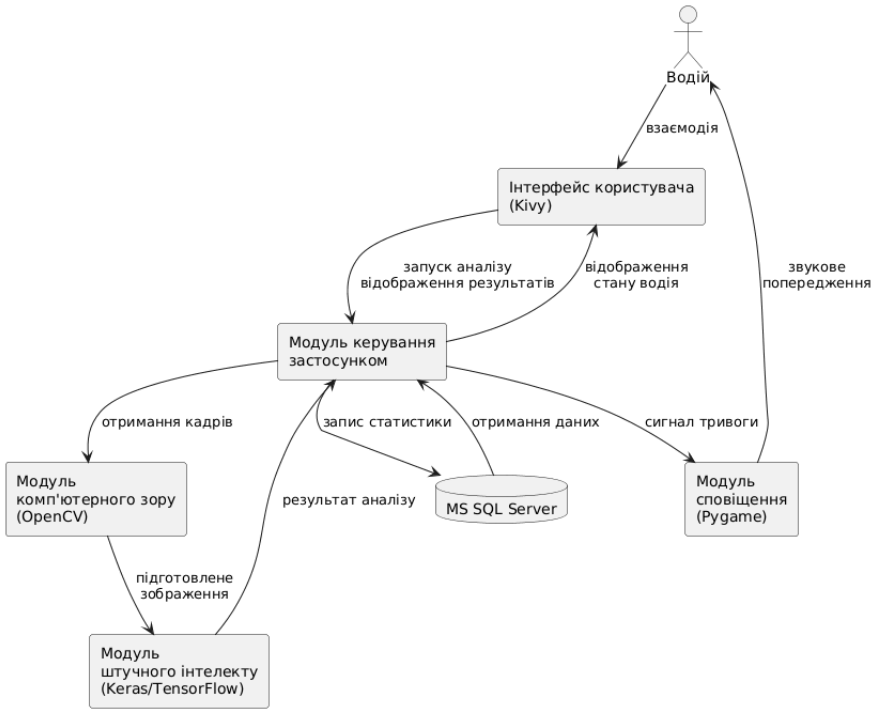

Рисунок 2.1 – Діаграма компонентів системи визначення сонливості водія

Центральним елементом архітектури є модуль керування застосунком. Його основним призначенням є координація взаємодії між усіма підсистемами. Саме цей компонент забезпечує отримання даних від камери, передачу їх до модулів аналізу, отримання результатів класифікації та виконання відповідних дій залежно від визначеного рівня сонливості.

Взаємодія користувача із системою здійснюється через графічний інтерфейс, реалізований за допомогою фреймворку Kivy [16]. Використання даного інструменту обумовлено його можливістю створення кросплатформних застосунків із сучасним інтерфейсом користувача. Через інтерфейс водій може запускати систему моніторингу, переглядати результати аналізу, отримувати інформацію про поточний стан та переглядати статистику попередніх перевірок. Крім того, інтерфейс забезпечує відображення повідомлень про виявлення ознак сонливості та рекомендацій щодо відпочинку.

Отримання та попередня обробка відеоданих виконується модулем комп’ютерного зору, який реалізований із використанням бібліотеки OpenCV [17]. Основним завданням цього компонента є захоплення кадрів із камери пристрою та їх підготовка до подальшого аналізу. На даному етапі здійснюється виявлення обличчя водія, виділення необхідних областей зображення та виконання операцій нормалізації даних. Попередня обробка дозволяє зменшити вплив зовнішніх факторів, таких як зміни освітлення або незначні зміщення камери, що позитивно впливає на точність подальшої класифікації.

Після завершення попередньої обробки дані передаються до модуля ШІ. Даний компонент є ключовим елементом системи, оскільки саме він відповідає за визначення рівня сонливості водія. Для реалізації модуля використовуються бібліотеки TensorFlow та Keras, які забезпечують створення та використання нейронних мереж для класифікації зображень. У процесі роботи модель аналізує характерні ознаки поведінки користувача, зокрема стан очей, тривалість їх закриття та інші параметри, що можуть свідчити про втому або сонливість. Результатом роботи модуля є числова оцінка стану водія та відповідний висновок щодо наявності або відсутності небезпечного рівня втоми.

Для забезпечення зберігання інформації про користувачів та результати роботи системи використовується база даних MS SQL Server. Вона дозволяє накопичувати статистичні дані, зберігати результати попередніх перевірок та формувати історію спостережень для кожного користувача. Застосування реляційної системи керування базами даних забезпечує надійне зберігання інформації та підтримує можливість подальшого розширення функціональності системи [18].

У випадку виявлення сонливості модуль штучного інтелекту передає відповідне повідомлення до підсистеми сповіщення. Для реалізації механізму звукового попередження використовується бібліотека Pygame, яка забезпечує відтворення аудіосигналів у режимі реального часу. Основним призначенням даного компонента є привернення уваги водія до потенційно небезпечного стану та стимулювання його до припинення керування транспортним засобом або відпочинку [19].

Запропонована архітектура забезпечує чітке розмежування функціональних обов’язків між компонентами системи та дозволяє ефективно реалізувати процес визначення сонливості водія в режимі реального часу. Використання модульного підходу спрощує підтримку програмного забезпечення, підвищує його масштабованість та створює передумови для подальшого вдосконалення алгоритмів аналізу стану користувача.

2.3 Побудова схеми бази даних

Програма призначена для визначення сонливості водія, що дозволить зберегти його життя, адже сонливість одна з причин більшості аварій. Система працює за допомогою нейронної мережі, яка розпізнає зображення з камери, визначає контури обличчя водія та перевіряє чи не закриті у нього очі. Після набору заданої кількості балів запускається звукова доріжка, яка повинна привернути увагу водія.

Система буде зберігати дані про водія (ім’я, адресу, телефон тощо), його машину (марка машини, тип палива тощо) та дані про роботу програми (час запуску програми, кількість набраних балів, коли спрацювала система).

Розглянемо операції з даними спроєктованої бази:

– виведення інформації за вказаним користувачем;
– виведення інформації про усіх користувачів за період;
– виведення інформації за період для заданого користувача;
– середній час спрацювання сигналу для конкретного користувача.

Враховуючи аналіз предметної області, створимо набір сутностей бази даних Drownes [20]. Задокументуємо (опишемо, розглянемо і щось інше) перелік сутностей спроєктованої бази даних у таблиці 2.1.

Таблиця 2.1 – Опис сутностей бази даних Drownes

Ім’я сутності	Опис	Псевдоніми	Особливості використання
1	2	3	4
Користувачі	Містить дані про реєстрацію	Users	До кожного користувача прив’язується id
Машини	Містить дані про машини водіїв	Car	До кожного водія прив’язується машина
Водії	Містить дані про водіїв	Drivers	Зберігається інформація про водія
Використання програми	Містить дані про всі запуски програми	Detection	Зберігає та виводить інформацію про використання програми

Визначимо для кожної сутності основні характеристики та набір атрибутів (див. Табл. 2.2).

Таблиця 2.2 – Опис атрибутів сутностей бази даних Drownes

Сутність	Атрибут	Опис	Домен	Обмеження	Значення за замовчуванням	Псевдонім	Значення NULL
1	2	3	4	5	6	7	8
Користувачі	Код користувача	Унікальний індентифікатор	Числовий тип, цілий	Первинний ключ	–	ID	Ні

Продовження таблиці 2.2

1	2	3	4	5	6	7	8
	Логін	Логін користувача	Символьний тип	–	–	login	Ні
	Пароль	Пароль для індентифікації	Символьний тип	–	–	password	Ні
Машини	ID	Унікальний індентифікатор	Числовий тип, цілий	Первинний ключ	–	ID_car	Ні
	Марка	Марка машини	Символьний тип	–	–	Marka	Ні
	Тип палива	Тип палива яке використовує машина	Символьний тип	–	–	Tup_paluva	Ні
	Водій	Водій якому належить ТЗ	Числовий тип, цілий	–	–	driver	Ні
Водій	ID	Унікальний індентифікатор	Числовий тип, цілий	Первинний ключ	–	Id_driver	Ні
	Ім’я	Ім’я користувача	Символьний тип	–	–	Name	Ні
	Адреса	Адреса проживання	Символьний тип	–	–	Address	Ні
	Номер телефону	Номер телефону	Символьний тип	–	–	Nomer	Ні
	Електрона пошта	Електрона пошта	Символьний тип	–	–	mail	Ні
Використання програми	Id	Унікальний індентифікатор	Числовий тип, цілий	Первинний ключ	–	id	Ні
	Запуск програми	Час запуску програми	Дата та час	–	–	start	Ні
	Кількість набраних балів	Кількість набраних балів	Символьний тип	–	–	score	Ні
	Виключення програми	Коли виключилася програма	Дата та час	–	–	finish	Ні
	Водій	Хто використовує програму	Числовий тип, цілий	–	–	driver	Ні

У таблиці 2.3 наведемо характеристики зв’язків сутностей бази даних.

Таблиця 2.3 – Опис зв’язків між сутностями бази даних Drownes

Ім’я сутності 1	Ім’я зв’язку	Ім’я сутності 2	Тип зв’язку
Користувач	має	Водій	1 : 1
Водій	має	Використання програми	1 : N

Продовження таблиці 2.3

1	2	3	4
Водій	Використовує	Машина	1 : N

На етапі логічного проєктування бази даних потрібно обрати модель даних. Відповідно до аналізу предметної області буде використано реляційну модель даних. Для реляційної моделі даних проєктування полягає у створенні реляційної схеми, визначенні кількості і структури відношень бази даних.

Коректність логічної моделі буде перевірятись за допомогою правил нормалізації, які дозволять переконатися в логічній цілісності і мінімальній збитковості моделі. Для аналізу концептуальної схеми використаємо нормалізацію відношень до третьої нормальної форми.

Для нормалізації спроєктованих відношень бази даних до першої нормальної передбачається усунення дублювання даних. Відношення знаходиться в першій нормальній формі тоді і тільки тоді, коли домен кожного атрибута містить лише неподільні значення, тобто значення кожного атрибута містить лише одне значення з цього домену [20]. Відношення повинне відповідати таким критеріям:

- кожна таблиця повинна мати первинний ключ: мінімальний набір атрибутів, які ідентифікують записи;
- уникнення повторень груп даних, тобто це категорії даних, що можуть зустрічатись різну кількість раз в різних записах;
- кожен атрибут повинен мати лише одне значення, а не множину значень.

Розглянемо відношення бази даних Drownes на наявність груп даних, що повторюються, та багатозначних атрибутів. Проаналізувавши їх, не було виявлено багатозначних атрибутів у відношеннях. Підсумовуючи аналіз відношень, можна сказати, що база даних Drownes відповідає першій нормальній формі.

Друга нормальна форма передбачає функціональну залежність описових атрибутів від первинного ключа. Для кожного відношення визначено простий первинний ключ. Далі потрібно перевірити чи існує функціональна залежність усіх інших атрибутів від первинного ключа у кожному відношенні. Проаналізувавши

відношення бази даних Drownes, виявлено функціональну залежність між описовими атрибутами та їх ключем. Отже, відношення спроєктованої бази даних [20] Drownes відповідають вимогам другої нормальної форми.

Третя нормальна форма передбачає усунення транзитивних залежностей між атрибутами. Розглянувши відношення бази даних Drownes, було визначено, що не існує транзитивних залежностей між описовими атрибутами і первинним ключем [20]. Тобто відношення бази даних Drownes відповідають вимогам третьої нормальної форми.

Отже, було проаналізоване кожне відношення бази даних Drownes, виключена надлишковість даних та проведена нормалізація до третьої нормальної форми.

Врахувавши аналіз предметної області було створено базу даних в системі керування MS SQL Server з іменем Drownes. Створимо спроєктовані таблиці для цієї бази даних. За допомогою наступного SQL-запиту створимо таблицю users (див. Лістинг 2.1) для збереження даних про користувачів.

Лістинг 2.1 – SQL-запит для створення таблиці users

```
CREATE TABLE [dbo]. [Users](
      [id_user] [int] IDENTITY(1,1) PRIMARY KEY,
      [login] [varchar](30) NULL,
      [password] [varchar](15) NULL)
```

Наступною створеною таблицею, є таблиця car (див. Лістинг 2.2), для виведення новин.

Лістинг 2.2 – SQL-запит для створення таблиці car

```
CREATE TABLE [dbo]. [Car](
      [id_car] [int] Primary key,
      [marka] [varchar](50) NULL,
      [rik] [varchar](50) NULL,
      [tup_paluva] [varchar](50) NULL,
      [driver] [int] NULL,
FOREIGN KEY( [driver]) REFERENCES [dbo]. [Drivers] ( [id_driver]))
```

Наступною створеною таблицею, є таблиця detection (див. Лістинг 2.3).

Лістинг 2.3 – SQL-запит для створення таблиці detection

```
CREATE TABLE [dbo]. [detection](
      [id] [int] IDENTITY(1,1) PRIMARY KEY,
      [start] [varchar](50) NULL,
      [score] [int] NULL,
      [finish] [varchar](50) NULL,
      [driver] [int] NULL,
FOREIGN KEY( [driver]) REFERENCES [dbo]. [Drivers]( [id_driver]))
```

Далі створюємо таблицю drivers (див. Лістинг 2.4).

Лістинг 2.4 – SQL-запит для створення таблиці drivers

```
create table transfers
(id int Primary Key Not NULL,
 [name] varchar(25) Not NULL,
 [img] text Not NULL,
 [content] text Not NULL)
```

Зв’язок між таблицями у системі керування базами даних SQL Server створено за допомогою SQL-запитів і діаграму зв’язків між таблицями наведено на рисунку 2.1.

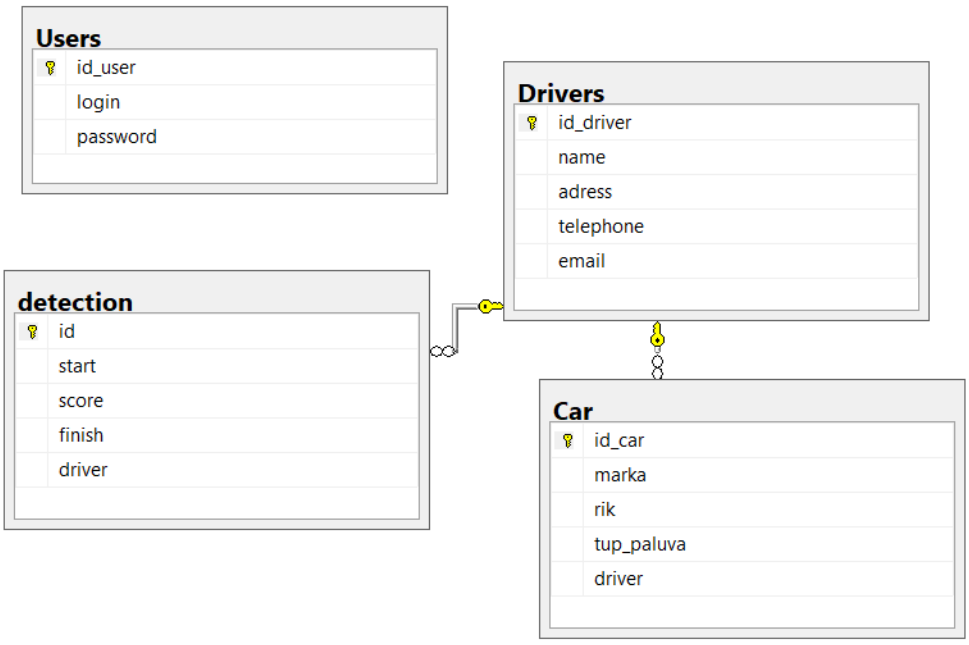

Рисунок 2.2 – Діаграма зв’язків між таблицями бази даних Drownes

2.4 Побудова UML-діаграм класів

Для деталізації внутрішньої структури програмної системи та опису взаємозв’язків між її основними об’єктами було побудовано UML-діаграму класів. На відміну від діаграм варіантів використання та послідовності, які відображають поведінку системи та взаємодію її компонентів у часі, діаграма класів дозволяє представити статичну структуру програмного забезпечення, набір його сутностей, атрибутів і зв’язків між ними [8].

Під час побудови моделі враховувалися результати аналізу предметної області, архітектура застосунку та структура бази даних Drownes. Основними сутностями системи є користувачі, водії, транспортні засоби та результати роботи програми. Крім того, до моделі включено програмні компоненти, які реалізують отримання відеопотоку, аналіз зображення та формування попереджувальних повідомлень.

Центральною сутністю системи є клас Driver, який містить персональні дані водія та використовується для зберігання інформації про користувача, що проходить моніторинг. Кожен водій пов’язаний із обліковим записом користувача системи та може використовувати один або декілька транспортних засобів.

Клас User відповідає за зберігання даних авторизації та використовується для забезпечення доступу до функціональних можливостей програмного забезпечення. Зв’язок між класами User та Driver реалізований за принципом «один до одного», оскільки кожному користувачу відповідає один профіль водія.

Інформація про транспортні засоби зберігається у класі Car. До його складу входять характеристики автомобіля, зокрема марка та тип палива. Один водій може використовувати декілька транспортних засобів, що визначає зв’язок типу «один до багатьох» між класами Driver та Car.

Результати роботи системи накопичуються у класі Detection. Даний клас містить відомості про запуск програми, кількість набраних балів сонливості та час завершення сеансу моніторингу. Кожному водію може відповідати множина записів

про використання програми, тому між класами Driver та Detection існує зв’язок типу «один до багатьох».

Окрім сутностей бази даних, до складу діаграми включено класи програмної логіки. Клас CameraManager відповідає за взаємодію із камерою пристрою та отримання відеопотоку. Клас AIModel реалізує аналіз зображення за допомогою нейронної мережі та визначає рівень сонливості користувача. Формування звукових попереджень виконує клас AlertManager, який взаємодіє з модулем штучного інтелекту та активується у випадку перевищення встановленого порогового значення.

Побудована UML-діаграма класів дозволяє наочно відобразити структуру програмної системи та є основою для подальшої реалізації програмного забезпечення засобами мови Python і системи керування базами даних MS SQL Server [15].

На рисунку 2.4 наведено UML-діаграму класів системи визначення сонливості водія.

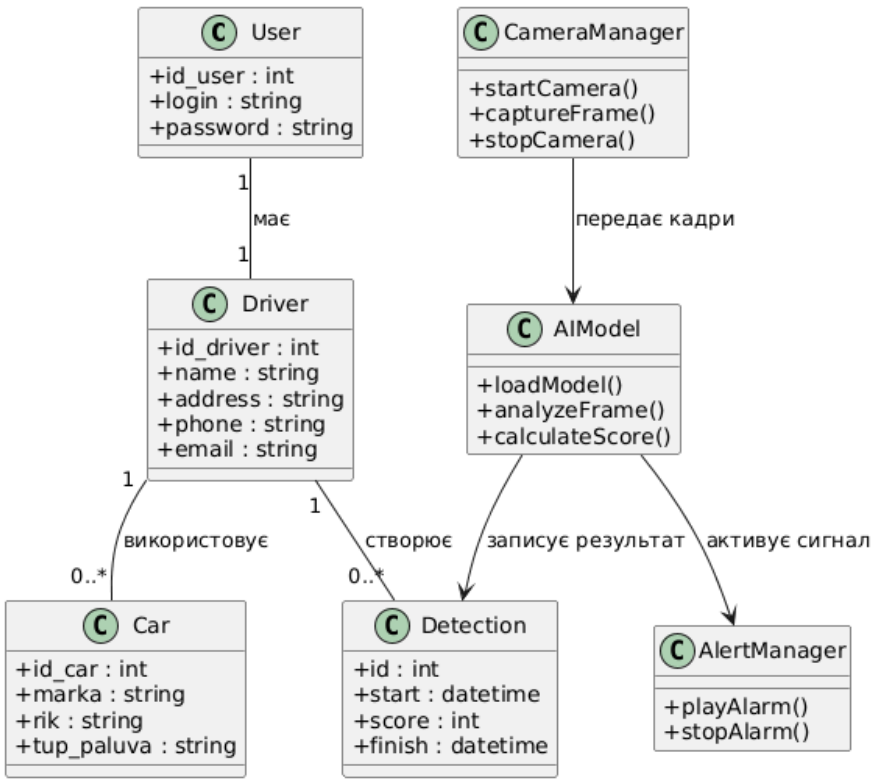

Рисунок 2.3 – Діаграма класів системи визначення сонливості водія

2.5 Вибір мови та середовища розробки

Ефективність роботи системи визначення сонливості водія значною мірою залежить від правильного вибору програмних засобів, які забезпечують обробку відеоданих у режимі реального часу, функціонування алгоритмів штучного інтелекту та взаємодію користувача із застосунком. З огляду на особливості предметної області та поставлені вимоги до програмного забезпечення було обрано технологічний стек, орієнтований на реалізацію систем комп'ютерного зору та машинного навчання.

Основною мовою програмування для створення системи стала Python. Даний вибір обумовлений широким використанням цієї мови у сфері штучного інтелекту, аналізу даних та комп'ютерного зору. Python поєднує простий синтаксис, високу швидкість розробки та наявність великої кількості спеціалізованих бібліотек, що дозволяє ефективно реалізовувати складні алгоритми обробки інформації. Важливою перевагою є також підтримка об'єктно-орієнтованого програмування, що спрощує проєктування архітектури програмного забезпечення та підвищує його масштабованість [21].

Для реалізації підсистеми комп'ютерного зору було використано бібліотеку OpenCV. Вона є одним із найпоширеніших програмних засобів для роботи із цифровими зображеннями та відеопотоками. Використання OpenCV дозволило організувати отримання кадрів із камери пристрою, виконувати їх попередню обробку та виділення об'єктів, необхідних для подальшого аналізу.

Основними перевагами даної бібліотеки є:

- висока швидкодія під час обробки відеопотоку в режимі реального часу;
- підтримка великої кількості алгоритмів комп'ютерного зору;
- можливість інтеграції з бібліотеками машинного навчання;
- кросплатформність та активна підтримка спільнотою розробників [17].

Для реалізації інтелектуального модуля визначення сонливості використано бібліотеку Keras, яка функціонує поверх платформи TensorFlow. Дана технологія забезпечує зручний інтерфейс для створення та навчання нейронних мереж різної

складності. Застосування Keras дозволило реалізувати модель штучного інтелекту, здатну аналізувати отримані зображення та визначати наявність ознак сонливості водія на основі візуальних характеристик обличчя.

Перевагами використання Keras є:

- спрощене проєктування архітектур нейронних мереж;
- підтримка сучасних методів глибокого навчання;
- можливість використання апаратного прискорення за допомогою графічних процесорів;
- інтеграція з екосистемою TensorFlow [5, 6].

Однією з основних вимог до системи є забезпечення зручної взаємодії користувача із програмним забезпеченням. Для реалізації графічного інтерфейсу було обрано фреймворк Kivy, який дозволяє створювати сучасні кросплатформні застосунки мовою Python. За допомогою даного інструменту реалізовано вікна керування програмою, елементи відображення результатів аналізу та інформаційні повідомлення для користувача.

Використання Kivy забезпечує такі переваги:

- підтримка різних операційних систем;
- можливість створення адаптивних інтерфейсів;
- інтеграція з бібліотеками комп’ютерного зору та машинного навчання;
- підтримка графічних компонентів і мультимедійних елементів [16].

Для реалізації механізму сповіщення водія використовується бібліотека Pygame. Її функціональність дозволяє відтворювати звукові сигнали в режимі реального часу та оперативно реагувати на результати роботи моделі штучного інтелекту. У разі досягнення критичного рівня сонливості система автоматично активує звукове попередження, яке привертає увагу користувача та сигналізує про необхідність відпочинку [19].

Зберігання інформації про користувачів, транспортні засоби та результати роботи системи реалізовано за допомогою системи керування базами даних Microsoft SQL Server. Використання реляційної бази даних дозволяє забезпечити

структуроване зберігання інформації, підтримку цілісності даних та ефективне виконання операцій пошуку й формування статистичних звітів.

До основних переваг MS SQL Server належать:

- підтримка механізмів забезпечення цілісності даних;
- висока продуктивність при роботі з великими обсягами інформації;
- підтримка зовнішніх ключів та реляційних зв’язків між таблицями;
- наявність засобів резервного копіювання та відновлення даних [22].

Для проєктування структури бази даних, створення таблиць та адміністрування сховища використовувався програмний комплекс SQL Server Management Studio (SSMS). Він забезпечує графічний інтерфейс для роботи з базою даних та дозволяє виконувати SQL-запити, контролювати структуру таблиць і здійснювати тестування розроблених рішень.

Безпосередня розробка програмного забезпечення здійснювалася в інтегрованому середовищі програмування PyCharm. Дане середовище є одним із найпопулярніших інструментів для створення програм мовою Python та забезпечує повний набір засобів для написання, тестування та налагодження програмного коду.

Перевагами PyCharm є:

- інтелектуальне автодоповнення коду та аналіз помилок у режимі реального часу;
- інтегровані засоби налагодження програм;
- підтримка віртуальних середовищ Python;
- інтеграція з системами контролю версій;
- зручні інструменти роботи з бібліотеками машинного навчання та комп’ютерного зору.

Таким чином, обраний технологічний стек забезпечує повну реалізацію функціональних вимог системи визначення сонливості водія. Поєднання мови програмування Python, бібліотек OpenCV, Keras та Pygame, фреймворку Kivy, системи керування базами даних MS SQL Server і середовища розробки PyCharm дозволяє створити ефективне програмне забезпечення для моніторингу стану водія та своєчасного виявлення ознак сонливості в режимі реального часу.

2.6 Реалізація основних класів та методів

Практична реалізація системи визначення сонливості водія базується на використанні методів комп'ютерного зору, згорткових нейронних мереж та технологій обробки відеопотоку в реальному часі. Основною задачею програмного забезпечення є безперервний аналіз зображення з вебкамери, визначення стану очей водія та формування попереджувального сигналу у випадку виявлення ознак засинання.

Для реалізації функціоналу використано мову програмування Python та бібліотеки OpenCV, Keras, NumPy, Kivy і PyODBC. Загальна логіка роботи системи складається з декількох послідовних етапів: ініціалізація моделей комп'ютерного зору, отримання відеопотоку, аналіз положення очей, оцінювання рівня сонливості, подача сигналу тривоги та збереження результатів роботи до бази даних MS SQL Server.

Першим етапом роботи програми є підключення необхідних бібліотек та завантаження нейронної мережі для класифікації стану очей. Реалізацію даного механізму наведено в лістингу 2.5.

Лістинг 2.5 – Ініціалізація моделей комп'ютерного зору

```
face = cv2.CascadeClassifier(
     'haar cascade files\haarcascade_frontalface_alt.xml')
     leye = cv2.CascadeClassifier(
     'haar cascade files\haarcascade_lefteye_2splits.xml')
     reye = cv2.CascadeClassifier(
     'haar cascade files\haarcascade_righteye_2splits.xml')
     model = load_model('models/cnncat2.h5')
```

У даному фрагменті коду відбувається завантаження каскадних класифікаторів Хаара, які використовуються для виявлення обличчя та очей водія на відеокадрі. Крім того, виконується завантаження попередньо навченої згорткової нейронної мережі CNN, яка класифікує стан ока як відкритий або закритий. Такий підхід дозволяє поєднати швидкість роботи алгоритмів OpenCV із високою точністю моделей машинного навчання.

Для отримання зображення з камери використовується механізм захоплення відеопотоку OpenCV, наведений у лістингу 2.6.

Лістинг 2.6 – Ініціалізація відеопотоку

```
cap = cv2.VideoCapture(0)
     while(True):
       ret, frame = cap.read()
       height, width = frame.shape [:2]
       gray = cv2.cvtColor(frame,
                 cv2.COLOR_BGR2GRAY)
```

Метод VideoCapture() забезпечує доступ до вебкамери пристрою. Під час кожної ітерації циклу отримується новий кадр відеопотоку, який додатково перетворюється у відтінки сірого. Використання чорно-білого зображення зменшує обсяг обчислень та підвищує швидкість роботи алгоритмів розпізнавання обличчя.

Після отримання кадру система виконує пошук обличчя та областей очей. Реалізацію даного механізму наведено в лістингу 2.7.

Лістинг 2.7 – Виявлення обличчя та очей

```
faces = face.detectMultiScale(
       gray,
       minNeighbors=5,
       scaleFactor=1.1,
       minSize=(25,25)
     )
     left_eye = leye.detectMultiScale(gray)
     right_eye = reye.detectMultiScale(gray)
```

Функція detectMultiScale() виконує пошук об'єктів на основі каскадних класифікаторів. Спочатку визначається область обличчя користувача, після чого окремо виконується пошук лівого та правого ока. Отримані координати використовуються для подальшого аналізу стану водія.

Наступним етапом є підготовка зображення очей до передачі нейронній мережі та виконання класифікації. Відповідний код наведено в лістингу 2.8.

Лістинг 2.8 – Аналіз стану очей за допомогою CNN

```
r_eye = cv2.resize(r_eye,(24,24))
     r_eye = r_eye/255
     r_eye = r_eye.reshape(24,24,-1)
     r_eye = np.expand_dims(r_eye,axis=0)
     rpred = model.predict(r_eye)
     rpred = np.argmax(rpred, axis=1)
```

Перед передачею до нейронної мережі зображення приводиться до розміру 24×24 пікселів та нормалізується в діапазоні від 0 до 1. Після цього модель виконує прогнозування та повертає результат класифікації. Значення «1» відповідає відкритому оку, тоді як значення «0» означає закритий стан.Для оцінювання рівня сонливості використовується спеціальний показник score, який накопичує бали залежно від тривалості закриття очей. Реалізація даного механізму наведена в лістингу 2.9.

Лістинг 2.9 – Обчислення рівня сонливості

```
if (np.logical_and(rpred [0]==0,
                lpred [0]==0)):
       score = score + 1
     else:
       score = score - 1
     if(score < 0):
       score = 0
```

Якщо обидва ока визначені як закриті, значення показника збільшується. У випадку відкриття хоча б одного ока кількість балів зменшується. Таким чином формується інтегральний показник, який відображає рівень втоми користувача протягом певного проміжку часу.

При досягненні критичного порогового значення система формує аварійне попередження. Відповідний механізм наведено в лістингу 2.10.

Лістинг 2.10 – Формування сигналу тривоги

```
if(score > 15):
       cv2.imwrite(
         os.path.join(path,'image.jpg'),
         frame
```

```
)
try:
  sound.play()
except:
  pass
```

У разі перевищення порогового значення 25 балів система вважає, що водій перебуває у стані сонливості. Поточний кадр зберігається на диск для подальшого аналізу, після чого відтворюється звуковий сигнал тривоги. Такий підхід забезпечує своєчасне привернення уваги водія до небезпечного стану.

Після завершення роботи програми результати автоматично записуються до бази даних MS SQL Server. Реалізацію підключення та збереження даних наведено в лістингу 2.11.

Лістинг 2.11 – Збереження результатів до бази даних

```
cnxn = pyodbc.connect(
     'DRIVER={ODBC Driver 17 for SQL Server};'
     'SERVER='+server+
     ';DATABASE='+database+
     ';UID='+username+';PWD='+password)
     cursor = cnxn.cursor()
     tsql = "INSERT INTO detection \
     start,score,finish,driver) \
     VALUES (...)"
     cursor.execute(tsql)
```

Для взаємодії з MS SQL Server використовується бібліотека PyODBC, яка забезпечує доступ до реляційної бази даних через драйвер ODBC. Після завершення сесії аналізу до таблиці Detection заносяться час початку роботи системи, фінальне значення показника сонливості, час завершення роботи та ідентифікатор водія. Збереження таких даних дозволяє накопичувати статистику використання системи та здійснювати подальший аналіз поведінки користувачів.

Таким чином, програмна реалізація поєднує алгоритми комп’ютерного зору OpenCV, нейронну мережу Keras та механізми роботи з базою даних MS SQL Server. Це забезпечує можливість виконання аналізу стану водія в режимі реального часу, оперативного формування попереджень про сонливість та накопичення статистичних даних для подальшого моніторингу ефективності роботи системи.

2.7 Розробка інтерфейсу користувача

Графічний інтерфейс користувача є важливою складовою програмної системи визначення сонливості водія, оскільки забезпечує взаємодію між користувачем та модулями аналізу стану водія. Під час розробки інтерфейсу основна увага приділялася простоті використання, швидкому доступу до основних функцій системи та зручності перегляду результатів роботи застосунку.

Інтерфейс реалізовано засобами фреймворку Kivy, що дозволило створити кросплатформний програмний продукт із сучасним дизайном та можливістю інтеграції алгоритмів комп'ютерного зору, машинного навчання та роботи з базою даних [18].

Структурно інтерфейс складається з декількох взаємопов'язаних вікон, кожне з яких виконує окрему функцію в процесі роботи системи.

Після запуску застосунку користувач потрапляє на сторінку авторизації, де може виконати вхід до системи під власним обліковим записом. Дане вікно містить поля для введення логіна та пароля, а також елементи навігації для переходу до реєстрації нового користувача.

Зовнішній вигляд вікна авторизації наведено на рисунку 3.1.

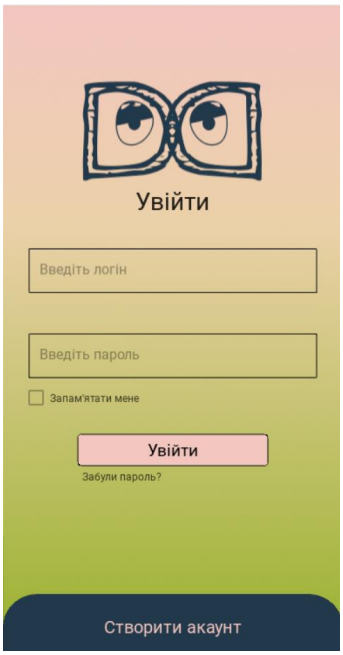

Рисунок 3.1 – Вікно авторизації додатку

Для нових користувачів передбачено механізм створення облікового запису. Після вибору відповідної функції система пропонує перейти до вікна реєстрації, де необхідно ввести персональні дані та створити обліковий запис для подальшого використання системи.

Вікно вибору реєстрації та форма створення нового користувача представлені на рисунках 3.2 та 3.3.

Рисунок 3.2 – Вікно вибору реєстрації

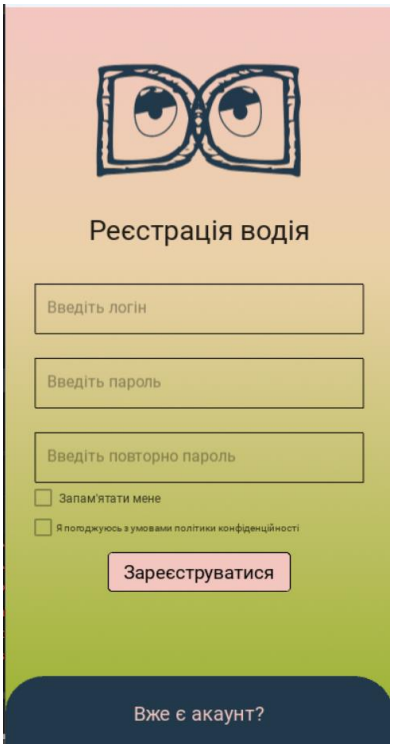

Рисунок 3.3 – Вікно реєстрації

Після успішної реєстрації користувач може виконати авторизацію у системі. Для цього використовується окреме вікно входу, в якому здійснюється перевірка облікових даних та надання доступу до функціоналу застосунку.

Вікно входу до системи показано на рисунку 3.4.

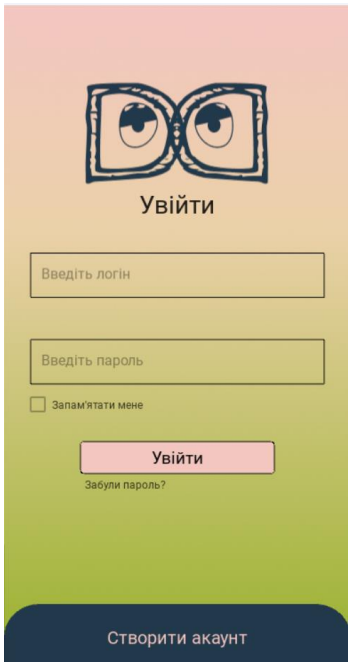

Рисунок 3.4 – Вікно входу у додаток

Після першої авторизації користувачеві пропонується заповнити анкету водія. На цьому етапі система збирає персональні дані, необхідні для подальшого збереження статистики та формування звітності щодо використання програми. До анкети входять відомості про водія, які надалі зберігаються у базі даних системи Drownes.

Форма введення даних водія наведена на рисунку 3.5.

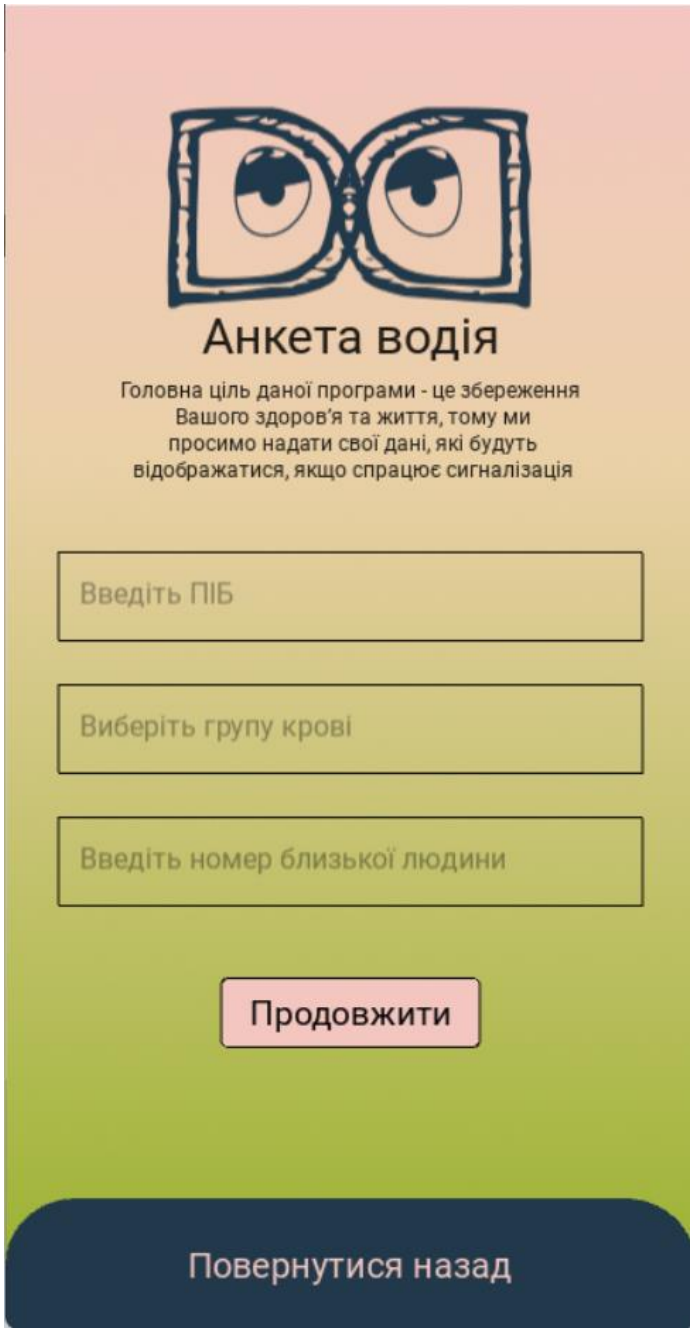

Рисунок 3.5 – Анкета водія

Після завершення налаштування профілю користувач переходить до головного вікна програми. Саме це вікно є центральним елементом взаємодії із системою та забезпечує доступ до основних функцій застосунку.

У головному меню відображаються: інформація про користувача; статистика використання системи; кнопка запуску процесу моніторингу; навігаційне меню; засоби переходу до додаткових розділів програми. Головне вікно системи наведено на рисунку 3.7.

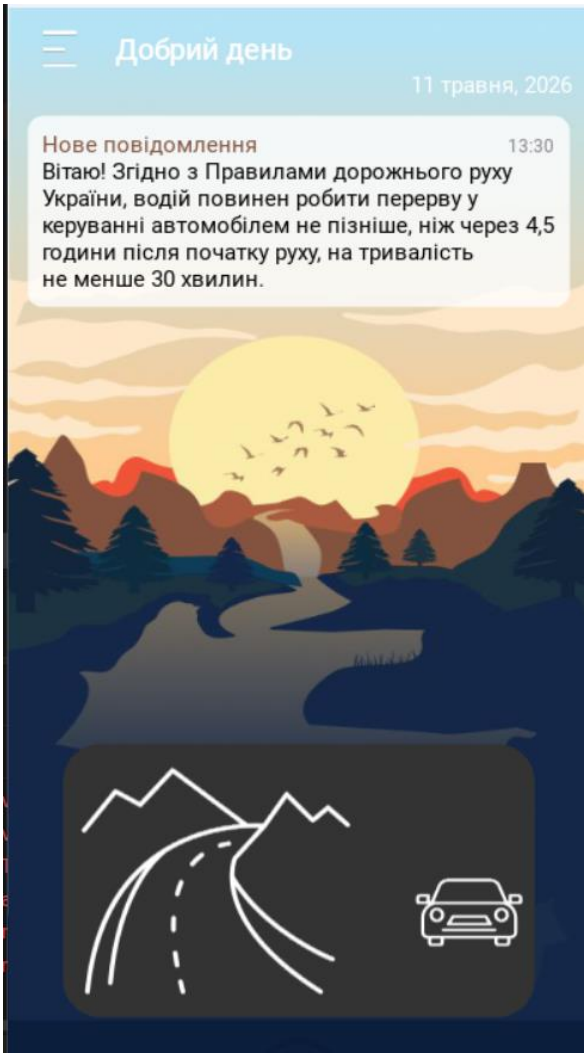

Рисунок 3.7 – Головне вікно програми

Для зручної навігації реалізовано бокове меню, яке забезпечує швидкий доступ до основних розділів застосунку. Завдяки цьому користувач може

переглядати власний профіль, статистичні дані та інші інформаційні модулі без необхідності повернення на головний екран.

Вигляд бокового меню представлено на рисунку 3.8.

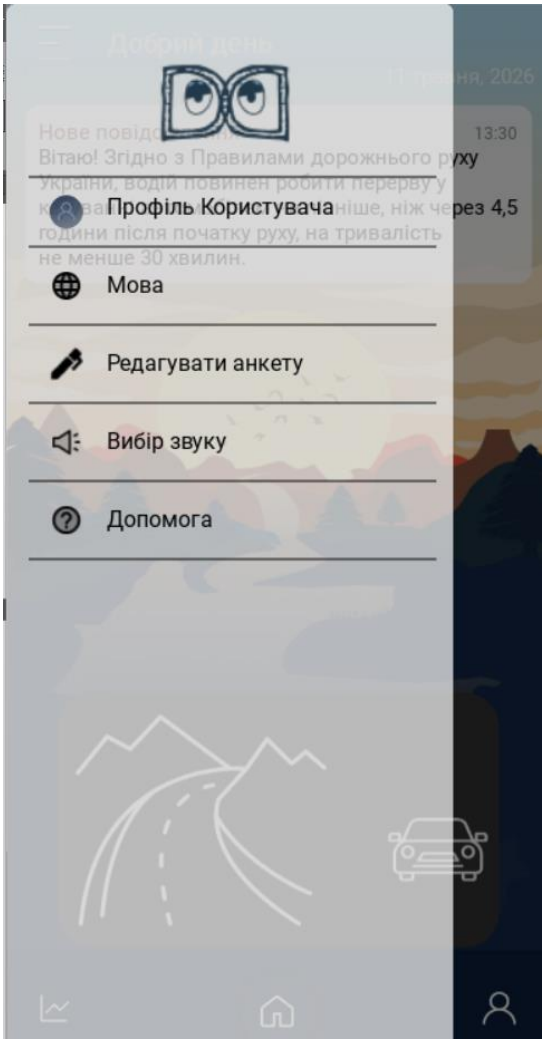

Рисунок 3.8 – Бокове меню додатка

У цьому розділі виконано проєктування та програмну реалізацію системи визначення сонливості водія. Розроблено архітектуру програмного забезпечення, спроєктовано базу даних Drownes, обґрунтовано вибір технологій та реалізовано основні програмні модулі. Також створено графічний інтерфейс користувача, який забезпечує реєстрацію, авторизацію та запуск моніторингу стану водія. У результаті отримано працездатну систему, здатну виявляти ознаки сонливості в режимі реального часу та формувати попереджувальний сигнал для підвищення безпеки дорожнього руху.

3. ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА ПІДТРИМКА

Після завершення розробки програмного комплексу визначення сонливості водія необхідно виконати оцінювання коректності його функціонування в реальних умовах експлуатації. Особливу увагу приділено перевірці роботи алгоритмів комп'ютерного зору, моделі машинного навчання, механізму формування попереджувального сигналу та взаємодії із базою даних [23].

У даному розділі наведено результати тестування розробленої системи, виконано аналіз її працездатності та перевірку правильності реалізації основних функціональних можливостей. Також розглянуто вимоги до програмного й апаратного забезпечення, необхідні для запуску системи, описано порядок її налаштування та введення в експлуатацію. Завершальним етапом є оцінка відповідності отриманого програмного продукту поставленим вимогам і завданням, визначеним на початкових етапах проєктування.

3.1 Тестування програмної системи

Перевірка працездатності розробленої системи є важливим етапом, що дозволяє оцінити коректність функціонування програмних модулів, надійність роботи алгоритмів комп'ютерного зору та стабільність взаємодії з базою даних. Основною метою тестування було підтвердження правильності виявлення стану очей водія, своєчасності спрацювання попереджувального сигналу та коректного збереження результатів роботи системи [24].

3.1.1 Види та план тестування

Для оцінювання якості програмного продукту було використано функціональне та інтеграційне тестування. Перевірка здійснювалася безпосередньо під час роботи системи в реальному часі із застосуванням вебкамери та тестових сценаріїв, що імітують різні стани водія [24].

Тестування охоплювало такі напрями:

- перевірка коректності підключення та отримання відеопотоку з камери;
- перевірка роботи алгоритму виявлення обличчя та очей за допомогою каскадів Хаара OpenCV;
- тестування нейронної мережі щодо правильності класифікації стану очей як відкритих або закритих;
- перевірка механізму накопичення балів сонливості;
- тестування спрацювання звукового сигналу при перевищенні критичного значення показника сонливості;
- оцінка стабільності роботи програми під час тривалого безперервного використання [25].

3.1.2 Розробка тестових сценаріїв

Для перевірки основних функцій програмної системи було сформовано набір тестових сценаріїв. Результати їх виконання наведено у таблиці 3.1.

Таблиця 3.1 – Текстові сценарії

Код тесту	**Опис тестового сценарію**	**Очікуваний результат**	**Фактичний результат**
ТС-01	Запуск програми та підключення до вебкамери	Відеопотік успішно відображається у вікні програми	Виконано успішно
ТС-02	Виявлення обличчя водія у кадрі	Система знаходить обличчя та відображає рамку навколо нього	Виконано успішно
ТС-03	Виявлення відкритих очей	Нейронна мережа визначає стан очей як «Open»	Виконано успішно
ТС-04	Виявлення закритих очей	Нейронна мережа визначає стан очей як «Closed»	Виконано успішно

Продовження таблиці 3.1

1	2	3	4
ТС-05	Накопичення балів сонливості при закритих очах	Значення Score збільшується при тривалому закритті очей	Виконано успішно
ТС-06	Зменшення показника сонливості після відкриття очей	Значення Score зменшується або повертається до нуля	Виконано успішно
ТС-07	Досягнення критичного значення Score	Після перевищення порога активується сигнал тривоги	Виконано успішно
ТС-08	Відтворення звукового попередження	Відтворюється файл alarm.wav	Виконано успішно
ТС-09	Візуальне попередження водія	Межі вікна підсвічуються червоним кольором	Виконано успішно
ТС-10	Робота системи при різному освітленні	Коректне виявлення обличчя та очей при достатньому освітленні	Виконано успішно
ТС-11	Завершення роботи програми клавішею Q	Програма завершується без помилок, камера звільняється	Виконано успішно

Для перевірки алгоритму комп’ютерного зору було виконано серію експериментів із різним положенням голови та різними умовами освітлення. У ході тестування система коректно виявляла обличчя та очі користувача, після чого нейронна мережа здійснювала класифікацію їхнього стану.

Окремо перевірявся механізм реагування на сонливість водія. Під час тривалого утримання очей у закритому стані відбувалося накопичення значення показника Score. Після досягнення встановленого порогового значення система автоматично запускала звуковий сигнал попередження та візуально виділяла межі відеокадру червоним кольором.

Завершальним етапом тестування стала перевірка роботи модуля взаємодії з базою даних. Після завершення роботи програми інформація про час запуску, кількість набраних балів, час завершення роботи та ідентифікатор водія успішно записувалася до таблиці Detection бази даних Drownes.

За результатами проведених випробувань усі тестові сценарії були виконані успішно. Отримані результати підтвердили правильність реалізації алгоритмів визначення сонливості, стабільність роботи програмного забезпечення та відповідність системи поставленим функціональним вимогам.

3.2 Розгортання програмної системи та системні вимоги

Розроблена система моніторингу сонливості водія не потребує високопродуктивного обладнання та може функціонувати на більшості сучасних персональних комп’ютерів і ноутбуків. Завдяки використанню бібліотек OpenCV, Keras, Kivy та MS SQL Server програмний продукт забезпечує стабільну роботу навіть на пристроях із середніми технічними характеристиками.

Для коректного функціонування системи необхідна операційна система сімейства Windows 10 або новішої версії. Також на комп’ютері має бути встановлене середовище виконання Python версії 3.8 або вище разом із необхідними програмними бібліотеками OpenCV, NumPy, Keras, TensorFlow, Kivy, Pygame та PyODBC. Для зберігання інформації про водіїв, транспортні засоби та результати роботи системи використовується система керування базами даних MS SQL Server [26].

Мінімальні апаратні характеристики робочого місця включають двоядерний процесор із тактовою частотою від 2 ГГц, 4 ГБ оперативної пам’яті та близько 2 ГБ вільного місця на накопичувачі. Обов’язковою умовою є наявність вебкамери, оскільки саме відеопотік із камери використовується для аналізу стану водія та визначення ознак сонливості.

Підготовка системи до роботи складається з кількох етапів. Спочатку встановлюється Python та всі необхідні програмні залежності. Після цього

розгортається база даних Drownes у середовищі MS SQL Server і створюються таблиці відповідно до розробленої структури. Далі налаштовуються параметри підключення до бази даних у програмному коді, зокрема адреса сервера, назва бази даних та облікові дані користувача.

Наступним кроком є розміщення файлів програми, моделі нейронної мережі, каскадів Хаара та звукового файлу сигналізації у робочому каталозі застосунку. Після запуску програма автоматично активує камеру, завантажує модель розпізнавання стану очей та переходить у режим безперервного моніторингу водія. Усі результати роботи системи після завершення сеансу записуються до бази даних, що дозволяє накопичувати статистику та виконувати подальший аналіз отриманих даних.

Таким чином, процедура розгортання не потребує складного налаштування, а обраний програмно-технічний стек забезпечує стабільну роботу системи на звичайних користувацьких комп’ютерах.

3.3 Верифікація програмної системи

Після завершення розробки було виконано перевірку програмного продукту з метою підтвердження правильності реалізації всіх передбачених функцій. Верифікація дозволила встановити, наскільки отриманий результат відповідає вимогам, визначеним під час аналізу предметної області та проєктування системи моніторингу стану водія [27].

У ході перевірки особлива увага приділялася роботі модуля комп’ютерного зору. Було встановлено, що система коректно отримує зображення з вебкамери, виконує пошук обличчя та локалізацію очей у відеопотоці. Після обробки кадрів нейронна мережа успішно визначає стан очей водія та передає результати для подальшого аналізу.

Дослідження роботи алгоритму показало, що накопичення показника сонливості відбувається відповідно до закладеної логіки. При тривалому закритті очей значення лічильника поступово збільшується, а після досягнення

встановленого порогу система автоматично формує попередження про небезпечний стан водія. Під час практичних випробувань звуковий сигнал активувався своєчасно, що підтвердило правильність функціонування механізму оповіщення.

Додатково було перевірено коректність взаємодії програмного забезпечення з базою даних. Результати тестових запусків засвідчили, що інформація про початок роботи програми, час завершення сеансу, набрану кількість балів та дані водія успішно зберігаються у базі даних Drownes. Отримані записи можуть бути використані для перегляду статистики та аналізу ефективності роботи системи.

Оцінка користувацької частини також підтвердила працездатність усіх основних екранів застосунку. Процеси реєстрації, авторизації, заповнення анкети водія, перегляду статистичних даних та запуску моніторингу виконуються без порушення логіки роботи програми. Інтерфейс забезпечує зручну взаємодію користувача із системою та надає доступ до всіх необхідних функцій.

За результатами проведеної верифікації можна зробити висновок, що розроблений програмний продукт забезпечує стабільне виконання поставлених завдань, а реалізовані програмні модулі коректно взаємодіють між собою в межах єдиної інформаційної системи контролю сонливості водія [27].

У третьому розділі було проведено перевірку працездатності розробленої системи визначення сонливості водія. Виконано тестування основних програмних модулів, проаналізовано результати їх роботи та підтверджено коректність функціонування алгоритмів комп’ютерного зору, нейронної мережі та механізму оповіщення. Також визначено системні вимоги до програмного забезпечення, описано процес його розгортання та виконано верифікацію готового продукту. Отримані результати підтвердили відповідність розробленої системи поставленим вимогам і можливість її практичного використання для моніторингу стану водія під час керування транспортним засобом.

4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

Розроблення програмного забезпечення передбачає постійну роботу спеціаліста з комп'ютерною технікою та значний час перебування за робочим місцем. Така професійна діяльність характеризується підвищеним навантаженням на організм працівника, зокрема на органи зору, опорно-руховий апарат та нервову систему, що може призводити до зниження працездатності й погіршення стану здоров'я. У даному розділі розглянуто основні ергономічні фактори ризику, які виникають під час роботи з комп'ютерними системами, а також визначено комплекс організаційних і технічних заходів щодо облаштування робочого місця програміста з метою забезпечення безпечних та комфортних умов праці.

4.1 Безпека життєдіяльності

У сучасних умовах розвитку транспортних технологій значна увага приділяється людському фактору як одній з основних причин виникнення дорожньо-транспортних пригод. Водій транспортного засобу є оператором складної системи «людина – машина – середовище», в якій безпека руху значною мірою залежить від його психофізіологічного стану. Одним із найбільш небезпечних чинників зниження працездатності водія є сонливість та втома

Працездатність людини-оператора визначається як здатність виконувати професійні обов'язки із заданою ефективністю протягом певного проміжку часу. Згідно з працями Атаманчука П.С. [28], Желібо Є.П. [29] та Запорожця О.І. [30], працездатність залежить від фізичного стану людини, рівня уваги, емоційного навантаження, умов праці та впливу факторів навколишнього середовища.

Для водія транспортного засобу основними показниками працездатності є:

- швидкість реакції на зміну дорожньої ситуації;
- концентрація уваги;
- точність керуючих дій;
- здатність приймати рішення;

- стійкість до монотонної роботи [28, 29].

Під час тривалого керування автомобілем спостерігається поступове зниження функціональних можливостей організму. Особливо небезпечним є виникнення стану сонливості, який характеризується:

- зменшенням частоти моргання;
- погіршенням концентрації уваги;
- збільшенням часу реакції;
- короткочасними епізодами засинання;
- зниженням здатності оцінювати дорожню обстановку [28, 29].

Стан сонливості є наслідком тривалого нервово-емоційного навантаження, недостатнього відпочинку, роботи в нічний час та монотонності дорожньої обстановки. Відповідно до положень безпеки життєдіяльності, людський фактор є причиною більшості аварійних ситуацій на транспорті [28].

При аналізі системи «водій – автомобіль – середовище» встановлено, що сонливість призводить до різкого зростання ризику виникнення надзвичайної ситуації. При цьому небезпека полягає не лише у повному засинанні водія, але й у так званих мікроснах тривалістю від декількох секунд до хвилини. Навіть короткочасна втрата контролю над транспортним засобом може призвести до тяжких наслідків.

З метою підвищення безпеки дорожнього руху дедалі ширше використовуються інтелектуальні системи моніторингу стану водія. У розробленій системі визначення сонливості використовується штучний інтелект та комп'ютерний зір для аналізу відеопотоку з камери. Алгоритм контролює положення очей, частоту моргання, тривалість закриття повік та інші ознаки втоми. У разі виявлення критичного стану система формує попереджувальний сигнал.

Застосування таких систем дозволяє:

- знизити ймовірність виникнення аварійних ситуацій;
- своєчасно попередити водія про небезпечний стан;
- підвищити рівень безпеки дорожнього руху;
- зменшити вплив людського фактора на виникнення ДТП.

Таким чином, контроль працездатності людини-оператора є важливою складовою забезпечення безпеки життєдіяльності. Розроблена система визначення сонливості водія на основі штучного інтелекту сприяє своєчасному виявленню ознак втоми та зниженню ризику виникнення надзвичайних ситуацій під час керування транспортним засобом [30].

4.2 Основи охорони праці

Розроблення програмного забезпечення для визначення сонливості водія здійснюється із застосуванням персонального комп'ютера. Тривала робота програміста за комп'ютером супроводжується значними статичними навантаженнями, напруженням органів зору та нервової системи. Тому організація робочого місця повинна відповідати вимогам ергономіки та охорони праці [4, 5].

Згідно з ДСанПіН 3.3.2.007-98 [32], НПАОП 0.00-7.15-18 [33], ДСТУ ISO 9241-5:2004 [34] та рекомендаціями щодо організації праці користувачів комп'ютерної техніки [31], робоче місце користувача персонального комп'ютера повинно забезпечувати комфортні та безпечні умови праці.

Робочий стіл програміста повинен мати достатню площу для розміщення монітора, клавіатури, миші та супровідної документації. Рекомендована висота робочої поверхні становить 680–800 мм [34].

Крісло оператора повинно бути регульованим за висотою та кутом нахилу спинки. Сидіння повинно забезпечувати підтримку поперекового відділу хребта та дозволяти змінювати робочу позу протягом робочого дня [34].

Монітор необхідно розташовувати на відстані 600–700 мм від очей користувача. Верхня межа екрана повинна знаходитися приблизно на рівні очей або трохи нижче. Таке розташування сприяє зменшенню навантаження на шийний відділ хребта та органи зору [31,34].

Клавіатура повинна розташовуватися на відстані 100–300 мм від переднього краю столу. Положення рук має забезпечувати кут згину в ліктьових суглобах

близько 90°. Маніпулятор типу «миша» розміщується на одній висоті з клавіатурою [34].

Особливе значення має освітлення робочого місця. Відповідно до ДБН В.2.5-28:2018 [35], для приміщень, у яких виконуються роботи з комп'ютерною технікою, нормативна освітленість робочої поверхні повинна становити 300–500 лк. Освітлення має бути рівномірним та не створювати відблисків на екрані монітора.

Параметри мікроклімату повинні відповідати вимогам ДСН 3.3.6.042-99 [36]. Для приміщень з персональними комп'ютерами рекомендуються параметри наведенні у таблиці 4.1

Таблиця 4.1 – Параметри приміщення

Показник	**Значення**
Температура повітря	22–24 °C
Відносна вологість	40–60 %
Швидкість руху повітря	до 0,1 м/с

Допустимий рівень шуму в приміщенні не повинен перевищувати 50 дБА відповідно до ДСН 3.3.6.037-99 [38].

Для запобігання професійній втомі необхідно дотримуватись раціонального режиму праці та відпочинку. При роботі з програмним кодом рекомендується через кожні 60 хвилин роботи виконувати перерву тривалістю 5–10 хвилин для відновлення функціонального стану організму та профілактики зорового стомлення [31-34].

Під час розробки системи визначення сонливості водія особливе значення має правильна організація робочого місця програміста, оскільки процес створення та тестування алгоритмів машинного навчання передбачає тривалу роботу за комп'ютером. Дотримання ергономічних вимог дозволяє зменшити втому, підвищити продуктивність праці та забезпечити збереження здоров'я працівника.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розв’язано задачу створення програмної системи для автоматизованого виявлення сонливості водія із застосуванням методів комп’ютерного зору та технологій нейронних мереж. Необхідність розробки подібних систем зумовлена зростанням кількості дорожньо-транспортних пригод, спричинених втратою концентрації уваги та засинанням водіїв під час руху. Використання сучасних засобів аналізу відеоданих дозволяє своєчасно визначати небезпечний стан людини та підвищувати рівень безпеки дорожнього руху.

У ході дослідження було проведено детальний аналіз предметної області, розглянуто фактори, що впливають на виникнення сонливості під час керування транспортними засобами, а також проаналізовано існуючі програмні та апаратні рішення, призначені для моніторингу стану водія. На основі виконаного аналізу визначено функціональні вимоги до майбутньої системи та обрано технології, які найкраще відповідають поставленим завданням.

У процесі проєктування сформовано архітектуру програмного продукту, що включає модулі отримання відеопотоку, виявлення обличчя та очей, аналізу отриманих зображень за допомогою нейронної мережі, генерації попереджувального сигналу та збереження результатів роботи. Для реалізації системи використано мову програмування Python і спеціалізовані бібліотеки OpenCV, Keras, Kivy та Pygame, які забезпечують обробку відеоданих, роботу моделей машинного навчання, побудову графічного інтерфейсу та відтворення звукових повідомлень. Для організації зберігання інформації була розроблена база даних Drownes на платформі MS SQL Server.

Практична реалізація системи дала змогу забезпечити безперервний контроль стану водія в режимі реального часу. Алгоритм роботи ґрунтується на визначенні області обличчя та очей за допомогою каскадних класифікаторів, після чого згорткова нейронна мережа виконує класифікацію стану очей. На основі послідовності отриманих результатів формується показник сонливості. У випадку

досягнення критичного значення система автоматично активує звукове попередження, що дозволяє привернути увагу водія та зменшити ризик виникнення небезпечної ситуації. Також реалізовано механізм накопичення статистичних даних про роботу програми та їх подальше збереження у базі даних.Під час тестування було перевірено працездатність усіх функціональних компонентів програмного продукту. Результати проведених випробувань підтвердили правильність роботи алгоритмів виявлення обличчя, локалізації очей, класифікації їх стану, формування тривожного сигналу та запису інформації до бази даних. Отримані результати свідчать про відповідність створеної системи поставленим вимогам і підтверджують можливість її практичного використання.

Таким чином, поставлена мета роботи була досягнута, а всі визначені завдання виконані у повному обсязі. Розроблена система забезпечує автоматичне виявлення ознак сонливості водія без використання дорогого спеціалізованого обладнання та може бути застосована як додатковий засіб підвищення безпеки керування транспортними засобами.

Перспективами подальшого розвитку проєкту є вдосконалення алгоритмів аналізу поведінки водія, розширення переліку контрольованих параметрів за рахунок аналізу положення голови, частоти моргання та позіхання, а також інтеграція програмної системи з мобільними платформами та сучасними автомобільними комплексами допомоги водію.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів спеціальності 121 – Інженерія програмного забезпечення, всіх форм навчання / уклад. Д. М. Михалик, Г. Б. Цуприк, В. М. Бревус. Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2024. 45 с. URL: https://elartu.tntu.edu.ua/handle/lib/50317 (дата звернення: 12.06.2026).

2. Система ADAS. URL: https://uamotors.com.ua/auto/163482 (дата звернення: 11.06.2026).

3. Користь сну. URL: https://zdrav.ck.gov.ua/uk/content/koryst-snu (дата звернення: 10.06.2026).

4. Olianin D., Tsupryk H., Hovorushchenko T., Bahrii-Zaiats O., Andrushchak I. Transformer Neural Networks in Industry 4.0 // Computer Information Technologies in Industry 4.0 : Proceedings of the 3rd International Workshop (CITI-2025), Ternopil, Ukraine, 11–12 June 2025. Ternopil : Ternopil Ivan Puluj National Technical University, 2025. URL: https://ceur-ws.org/Vol-4057/ (дата звернення: 12.06.2026).

5. Keras Documentation. URL: https://keras.io (дата звернення: 15.05.2026).

6. TensorFlow Documentation. URL: https://www.tensorflow.org (дата звернення: 16.05.2026).

7. Фаулер М. UML. Розробка програмного забезпечення / пер. з англ. Київ, 2017. 210 с.

8. Основи UML-проєктування розподілених систем. URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?inpopup=true&id=44416 (дата звернення: 17.05.2026).

9. Поглиблене знання діаграми варіантів використання UML. URL: https://www.mindonmap.com/uk/blog/what-is-a-uml-use-case-diagram/ (дата звернення: 18.05.2026).

10. Діаграма послідовності. URL: http://flash.retejo.info/cxefpagxo/uml/diagrama-poslidovnosti (дата звернення: 19.05.2026).

11. Ітераційна модель розробки. URL: https://stud.com.ua/174127/informatika/iteratsiyna_model (дата звернення: 20.05.2026).

12. Kotov Y., Yavorska E., Tsupryk H., Dzierżak R., Reshetnik O., Bokovets V. Evaluating interoperability and data quality in FHIR-based AI assessment pipelines // Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments 2025. Proc. SPIE. 2025. Vol. 14009. Art. 140091F. DOI: https://doi.org/10.1117/12.3100561.

13. Olianin D., Tsupryk H. LLM-based Extraction from Resumes // Advanced Technologies in Scientific Research : Collection of Scientific Papers with Proceedings of the 1st International Scientific and Practical Conference, Rotterdam, Netherlands, 20–22 August 2025. International Scientific Unity, 2025. P. 72–76.

14. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. 775 p.

15. UML-діаграми, їхні основні типи та процес розроблення. URL: https://foxminded.ua/uml-diagramy/ (дата звернення: 21.05.2026).

16. Kivy Documentation. URL: https://kivy.org/doc/stable (дата звернення: 22.05.2026).

17. OpenCV Documentation. URL: https://docs.opencv.org (дата звернення: 23.05.2026).

18. Коннолі Т., Бегг К. Системи баз даних: практичний підхід до проектування, впровадження та управління. 6-те вид. Київ : Підручники і посібники, 2014. 1440 с.

19. Pygame Documentation. URL: https://www.pygame.org/docs/ (дата звернення: 24.05.2026).

20. Берко А. Ю., Верес О. М., Пасічник В. В. Системи баз даних та знань. Книга 2. Системи управління базами даних та знань : навч. посіб. Львів : Магнолія-2016, 2012. 584 с.

21. Python Documentation. URL: https://docs.python.org/3 (дата звернення: 25.05.2026).

22. SQL Server Documentation. URL: https://learn.microsoft.com/sql (дата звернення: 26.05.2026).

23. Олянін Д., Цуприк Г. Огляд ролі трансформерних нейронних мереж у видобуванні інформації із неструктурованих даних // Measuring and Computing Devices in Technological Processes. 2025. № 2 (82). С. 360–364. DOI: https://doi.org/10.31891/2219-9365-2025-82-52.

24. Black R., van Veenendaal E., Graham D. Foundations of Software Testing: ISTQB Certification. 3rd ed. Boston : Cengage Learning EMEA, 2012. 320 p.

25. Тіан Д. Інженерія якості програмного забезпечення: тестування, забезпечення якості та кількісне поліпшення. Нью-Йорк : John Wiley & Sons, 2005. 448 с.

26. Zadok M. DevOps in Python: Infrastructure as Python. New York : Springer, 2020. 320 p.

27. Хрол Н. Д. Формальні методи верифікації програмного забезпечення // ІКТ. 2019. Вип. 10. С. 45–56. URL: https://ict.op.edu.ua/index.php/journal/article/view/10 (дата звернення: 27.05.2026).

28. Атаманчук П. С. Безпека життєдіяльності : навч. посіб. Київ : Центр учбової літератури, 2020.

29. Желібо Є. П., Зацарний В. В. Безпека життєдіяльності : підручник. Київ : Каравела, 2023.

30. Запорожець О. І. Безпека життєдіяльності. Київ : Центр учбової літератури, 2020.

31. Андрейчук Н. І. Охорона праці : навч. посіб. Львів : Видавництво Львівська політехніка, 2021.

32. Жидецький В. Ц. Охорона праці користувачів комп'ютерів. Львів : Афіша, 2020.

33. ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами.

34. НПАОП 0.00-7.15-18. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями.

35. ДСТУ ISO 9241-5:2004. Ергономічні вимоги до роботи з відеотерміналами в офісі.

36. ДБН В.2.5-28:2018. Природне і штучне освітлення.

37. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень.

38. ДСН 3.3.6.037-99. Санітарні норми виробничого шуму, ультразвуку та інфразвуку.

ДОДАТКИ

ДОДАТОК А

Тези

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет
Чернівецький національний університет
імені Юрія Федьковича
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

IX МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

24-25 квітня 2026 р.

(збірник тез конференції)

Тернопіль 2026

IX Міжнародна студентська науково - технічна конференція
"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

УДК 004.41
Целінь А. -ст. гр. СПс-41
Тернопільський національний технічний університет імені Івана Пулюя

СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНОЇ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЧНОГО ВИЗНАЧЕННЯ ПОРУШЕНЬ КОНЦЕНТРАЦІЇ УВАГИ

Науковий керівник: к.т.н., доцент Цуприк Г. Б.

Tselin A.
Ternopil Ivan Puluj National Technical University

DEVELOPMENT OF AN INTELLIGENT SOFTWARE SYSTEM FOR THE AUTOMATIC DETECTION OF ATTENTION DEFICITS

Supervisor: PhD, Associate Professor Tsupryk H. B.

Ключові слова: штучний інтелект, визначення сонливості водія, комп'ютерний зір.
Keywords: artificial intelligence, driver drowsiness detection, computer vision.

Актуальність теми дослідження зумовлена необхідністю підвищення безпеки дорожнього руху за рахунок використання сучасних технологій штучного інтелекту. Однією з ключових причин дорожньо-транспортних пригод є сонливість водія, що знижує концентрацію та швидкість реакції. Це обумовлює потребу у створенні програмного забезпечення, здатного своєчасно виявляти ознаки втоми та попереджати водія.

Метою роботи є розробка програмного застосунку на основі штучного інтелекту, який забезпечує визначення сонливості водія з використанням методів комп'ютерного зору та формує аналітичні висновки для підвищення рівня безпеки.

Для досягнення поставленої мети реалізовано систему, що обробляє відеопотік у реальному часі та аналізує поведінкові характеристики водія. Основна увага приділяється розпізнаванню ознак сонливості, таких як частота моргання, тривалість закриття очей.

Клієнтська частина застосунку забезпечує зручну взаємодію користувача із системою та відображення результатів аналізу. Інтерфейс дозволяє в режимі реального часу отримувати повідомлення про стан водія та потенційні ризики.

Основним завданням системи є проведення комплексного аналізу стану водія. Використання алгоритмів комп'ютерного зору та штучного інтелекту дозволяє оцінювати рівень сонливості на основі візуальних даних, що забезпечує високу точність і швидкість обробки інформації.

Важливим компонентом системи є модуль автоматичного формування висновків, який аналізує отримані дані та генерує рекомендації щодо необхідності відпочинку або зупинки.

У майбутньому функціональність системи може бути розширена за рахунок інтеграції з бортовими системами автомобіля та мобільними пристроями, що дозволить реалізувати автоматичне попередження водія або активацію допоміжних механізмів безпеки. Також можливе вдосконалення алгоритмів штучного інтелекту для підвищення точності визначення сонливості, зокрема шляхом використання глибокого навчання та аналізу додаткових поведінкових і фізіологічних показників.

Висновки. У результаті виконання роботи створено програмний продукт для визначення сонливості водія з використанням штучного інтелекту та комп'ютерного зору. Його застосування сприятиме зниженню ризику аварійних ситуацій, підвищенню безпеки дорожнього руху та має потенціал до подальшого розвитку й удосконалення.

Посилання на літературу:

1. Олянін, Д., Цуприк, Г. (2025) Transformer Neural Networks in Industry 4.0 / Д. Олянін, Г. Цуприк, Т. Говорущенко, О. Багрій-Заяць, І. Андрущак // Computer Information Technologies in Industry 4.0: proceedings of the 3rd International Workshop (CITI-2025), Ternopil, Ukraine, 11–12 June 2025. – Ternopil : Ternopil Ivan Puluj National Technical University, 2025 (Scopus) https://ceur-ws.org/Vol-4057/

2. Tsupryk, H., Olianin, D. (2025). Vydobuvannia danyh z tekstu vykorystovuiuchy transformerni neironni merezhi [Data extraction from text using Transformer Neural Networks]. Information Technology: Computer Science, Software Engineering and Cyber Security, 125–130, DOI: https://doi.org/10.32782/IT/2025-2-13

3. ОЛЯНІН D., & ЦУПРИК Н. (2025). Огляд ролі трансформерних нейроних мереж у видобувані інформації із неструктурованих даних. Measuring and computing devices in technological processes, 82(2), 360–364. https://doi.org/10.31891/2219-9365-2025-82-52

4. Zhao X., Wang L., Zhang Y. et al. (2024). A review of convolutional neural networks in computer vision. Artificial Intelligence Review, 57, 99. https://doi.org/10.1007/s10462-024-10721-6

ДОДАТОК Б

Код програми

```
from kivy.lang import Builder
from kivy.uix.label import Label
from kivymd.uix.boxlayout import MDBoxLayout
from kivymd.uix.screen import MDScreen
from kivymd.app import MDApp
from kivymd.theming import ThemableBehavior
import sqlite3
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.core.window import Window
import os
from kivy.uix.boxlayout import BoxLayout
import time
from kivy.app import App
from keras.models import load_model
import numpy as np
import datetime
from kivy.graphics.texture import Texture
from kivy.uix.camera import Camera
from kivy.clock import Clock
from kivymd.uix.menu import MDDropdownMenu
from kivymd.uix.list import OneLineListItem
from kivy.properties import StringProperty
from kivy.uix.floatlayout import FloatLayout
from kivy.properties import ObjectProperty
from kivymd.uix.menu import MDDropdownMenu
import pyodbc
import cv2
import os
from keras.models import load_model
import numpy as np
from kivy.uix.popup import Popup
from kivy.uix.label import Label
from kivy.uix.button import Button
from kivy.storage.jsonstore import JsonStore
from kivy.animation import Animation
from kivy.core.audio import SoundLoader


#import pymssql

kv = Builder.load_file("Design.kv")

class WindowManager(ScreenManager):
  pass

class CameraScreen(Screen):
  def fill_progress_bar(self):
    progress_bar = self.ids.progress_bar
```

```
    progress_bar.max = 100
    progress_bar.value = 0
    duration_hours = 4.5
    duration_seconds = duration_hours * 60 * 60
    progress_animation            =            Animation(value=100,
duration=duration_seconds)
    Clock.schedule_once(lambda                                    dt:
progress_animation.start(progress_bar), 0.5) # Початок анімації після
0.5 секунди


class Item(OneLineListItem):
  text = StringProperty()

class HomeScreen(Screen):
  def profile(self):
    self.manager.current = "profile"
  def drownes(self):

    sound = SoundLoader.load("alarm.wav")

    face        =        cv2.CascadeClassifier('haar        cascade
files\haarcascade_frontalface_alt.xml')
    leye        =        cv2.CascadeClassifier('haar        cascade
files\haarcascade_lefteye_2splits.xml')
    reye        =        cv2.CascadeClassifier('haar        cascade
files\haarcascade_righteye_2splits.xml')

    lbl= ['Close','Open']

    model = load_model('models/cnncat2.h5')
    path = os.getcwd()
    cap = cv2.VideoCapture(0)
    font = cv2.FONT_HERSHEY_COMPLEX_SMALL
    count=0
    score=0
    thicc=2
    rpred= [99]
    lpred= [99]

    start = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    print(start)

    while(True):
      ret, frame = cap.read()
      height,width = frame.shape [:2]

      gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

      faces                                                        =
face.detectMultiScale(gray,minNeighbors=5,scaleFactor=1.1,minSize=(2
5,25))
      left_eye = leye.detectMultiScale(gray)
```

```
      right_eye = reye.detectMultiScale(gray)

      cv2.rectangle(frame, (0,height-50) , (200,height) , (0,0,0) ,
thickness=cv2.FILLED )

      for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y) , (x+w,y+h) , (100,100,100) , 1 )

      for (x,y,w,h) in right_eye:
        r_eye=frame [y:y+h,x:x+w]
        count=count+1
        r_eye = cv2.cvtColor(r_eye,cv2.COLOR_BGR2GRAY)
        r_eye = cv2.resize(r_eye,(24,24))
        r_eye= r_eye/255
        r_eye= r_eye.reshape(24,24,-1)
        r_eye = np.expand_dims(r_eye,axis=0)
        rpred = model.predict(r_eye)
        rpred = np.argmax(rpred, axis=1)
        if(rpred [0]==1):
          lbl='Open'
        if(rpred [0]==0):
          lbl='Closed'
        break

      for (x,y,w,h) in left_eye:
        l_eye=frame [y:y+h,x:x+w]
        count=count+1
        l_eye = cv2.cvtColor(l_eye,cv2.COLOR_BGR2GRAY)
        l_eye = cv2.resize(l_eye,(24,24))
        l_eye= l_eye/255
        l_eye=l_eye.reshape(24,24,-1)
        l_eye = np.expand_dims(l_eye,axis=0)
        lpred = model.predict(l_eye)
        lpred = np.argmax(lpred, axis=1)
        if(lpred [0]==1):
          lbl='Open'
        if(lpred [0]==0):
          lbl='Closed'
        break

      if (np.logical_and (rpred [0]==0, lpred [0]==0)):
        score=score+1
        cv2.putText(frame,"Closed",(10,height-20),              font,
1,(255,255,255),1,cv2.LINE_AA)
        #print(rpred [0], lpred [0])
      #if(np.logical_or(rpred [0]==1, lpred [0]==1)):
      else:
        score=score-1
        cv2.putText(frame,"Open",(10,height-20),                font,
1,(255,255,255),1,cv2.LINE_AA)
        #print(rpred [0], lpred [0])

      if(score<0):
```

```
        score=0
      cv2.putText(frame,'Score:'+str(score),(100,height-20),    font,
1,(255,255,255),1,cv2.LINE_AA)
      if(score>25):
        #person is feeling sleepy so we beep the alarm
        cv2.imwrite(os.path.join(path,'image.jpg'),frame)
        try:
          if sound:
            sound.play()
        except: # isplaying = False
          pass
        if(thicc<16):
          thicc= thicc+2
        else:
          thicc=thicc-2
          if(thicc<2):
            thicc=2
        cv2.rectangle(frame,(0,0),(width,height),(0,0,255),thicc)
      cv2.imshow('frame',frame)

      if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    cap.release()
    cv2.destroyAllWindows()



class ProfileScreen(Screen):
  pass

class ChoiceScreen(Screen):
  pass

class RegisterDriverScreen(Screen):
  def clear(self):
    self.ids.login.text = ""
    self.ids.password.text = ""
    self.ids.repassword.text = ""

  def createaccount(self):
    input_login = self.ids.login.text
    input_password = self.ids.password.text
    input_repassword = self.ids.repassword.text

    server = r'localhost\SQLEXPRESS'
    database = 'CarDB'

    cnxn = pyodbc.connect(
      'DRIVER={ODBC Driver 17 for SQL Server};'
      f'SERVER={server};'
      f'DATABASE={database};'
      'Trusted_Connection=yes;'
    )
```

```
    cursor = cnxn.cursor()

    if input_login == "" or input_password == "" or input_repassword
== "":
      self.clear()
    elif input_password != input_repassword:
      self.clear()
      self.invalidPasswords()
    else:
      # Перевірка наявності вже існуючого запису з таким логіном
      query = "SELECT count(*) FROM Users WHERE login='" + input_login
+ "'"
      cursor.execute(query)
      data = cursor.fetchone()
      count = data [0]

      if count > 0:
        self.clear()
        self.invalidLogin()
      else:
        query = "INSERT INTO Users(login, password) VALUES('" +
input_login + "', '" + input_password + "')"
        cursor.execute(query)
        cnxn.commit()
        # Перехід до іншого екрану
        self.manager.current = "questionnairedriver"

    cnxn.close()

  def invalidPasswords(self):
    content = Label(text="Паролі не збігаються.")
    close_button = Button(text="Закрити", size_hint=(None, None),
size=(150, 50))
    popup = Popup(title="Помилка реєстрації", content=content,
size_hint=(None, None), size=(400, 200))
    close_button.bind(on_release=popup.dismiss)
    content.add_widget(close_button)
    popup.open()

  def invalidLogin(self):
    content = Label(text="Користувач з таким логіном вже існує.")
    close_button = Button(text="Закрити", size_hint=(None, None),
size=(150, 50))
    popup = Popup(title="Помилка реєстрації", content=content,
size_hint=(None, None), size=(400, 200))
    close_button.bind(on_release=popup.dismiss)
    content.add_widget(close_button)
    popup.open()


class RegisterEnterpriseScreen(Screen):
  def clear(self):
```

```
    self.ids.nameenterprice.text = ""
    self.ids.loginenterprice.text = ""
    self.ids.passwordenterprice.text = ""
    self.ids.repasswordenterprice.text = ""

  def createaccount(self):
    input_name = self.ids.nameenterprice.text
    input_login = self.ids.loginenterprice.text
    input_password = self.ids.passwordenterprice.text
    input_repassword = self.ids.repasswordenterprice.text

    server = 'web.mail.guscoll.com,1433'
    database = 'tselinart'
    username = 'atselin'
    password = '111'
    cnxn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL
Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD
='+ password)
    cursor = cnxn.cursor()

    if input_name == "" or input_login == "" or input_password == ""
or input_repassword == "":
      self.clear()
    elif input_password != input_repassword:
      self.clear()
      self.invalidPasswords()
    else:
      # Перевірка наявності вже існуючого запису з таким логіном
      query = "SELECT count(*) FROM UserEnterprice WHERE
loginEnterprice='" + input_login + "'"
      cursor.execute(query)
      data = cursor.fetchone()
      count = data [0]

      if count > 0:
        self.clear()
        self.invalidLogin()
      else:
        query = "INSERT INTO UserEnterprice(nameEnterprice,
loginEnterprice, passpwordenterprice) VALUES('" + input_name + "', '"
+ input_login + "', '" + input_password + "')"
        cursor.execute(query)
        cnxn.commit()
        # Перехід до іншого екрану
        self.manager.current = "home"

    cnxn.close()

  def invalidPasswords(self):
    content = Label(text="Паролі не збігаються.")
    close_button = Button(text="Закрити", size_hint=(None, None),
size=(150, 50))
```

```
    popup = Popup(title="Помилка реєстрації", content=content,
size_hint=(None, None), size=(400, 200))
    close_button.bind(on_release=popup.dismiss)
    content.add_widget(close_button)
    popup.open()

  def invalidLogin(self):
    content = Label(text="Користувач з таким логіном вже існує.")
    close_button = Button(text="Закрити", size_hint=(None, None),
size=(150, 50))
    popup = Popup(title="Помилка реєстрації", content=content,
size_hint=(None, None), size=(400, 200))
    close_button.bind(on_release=popup.dismiss)
    content.add_widget(close_button)
    popup.open()


class QuestionnaireEnterpriseScreen(Screen):
  def on_enter(self):
    menu_list = [
      {
        "viewclass": "Item",
        "text": "A(I)R(-)",
        "on_release": lambda x="A(I)R(-)": self.set_item(x)
      },
      {
        "viewclass": "Item",
        "text": "A(I)R(+)",
        "on_release": lambda x="A(I)R(+)": self.set_item(x)
      },
      {
        "viewclass": "Item",
        "text": "B(III)R(-)",
        "on_release": lambda x="B(III)R(-)": self.set_item(x)
      },
      {
        "viewclass": "Item",
        "text": "B(III)R(+)",
        "on_release": lambda x="B(III)R(+)": self.set_item(x)
      },
      {
        "viewclass": "Item",
        "text": "AB(IV)R(-)",
        "on_release": lambda x="AB(IV)R(-)": self.set_item(x)
      },
      {
        "viewclass": "Item",
        "text": "AB(IV)R(+)",
        "on_release": lambda x="AB(IV)R(+)": self.set_item(x)
      },
      {
        "viewclass": "Item",
        "text": "0(I)R(-)",
```

```
        "on_release": lambda x="0(I)R(-)": self.set_item(x)
      },
      {
        "viewclass": "Item",
        "text": "0(I)R(+)",
        "on_release": lambda x="0(I)R(+)": self.set_item(x)
      }
    ]
    self.menu = MDDropdownMenu(
      caller=self.ids.dropItem,
      items=menu_list,
      width_mult=4
    )
    self.menu.bind()

  def set_item(self, text_item):
    self.ids.dropItem.text = text_item
    self.menu.dismiss()

class QuestionnaireDriverScreen(Screen):
  def on_enter(self):
    menu_list = [
      {
        "viewclass": "Item",
        "text": "A(I)R(-)",
        "on_release": lambda x="A(I)R(-)": self.set_item(x)
      },
      {
        "viewclass": "Item",
        "text": "A(I)R(+)",
        "on_release": lambda x="A(I)R(+)": self.set_item(x)
      },
      {
        "viewclass": "Item",
        "text": "B(III)R(-)",
        "on_release": lambda x="B(III)R(-)": self.set_item(x)
      },
      {
        "viewclass": "Item",
        "text": "B(III)R(+)",
        "on_release": lambda x="B(III)R(+)": self.set_item(x)
      },
      {
        "viewclass": "Item",
        "text": "AB(IV)R(-)",
        "on_release": lambda x="AB(IV)R(-)": self.set_item(x)
      },
      {
        "viewclass": "Item",
        "text": "AB(IV)R(+)",
        "on_release": lambda x="AB(IV)R(+)": self.set_item(x)
      },
      {
```

```
        "viewclass": "Item",
        "text": "0(I)R(-)",
        "on_release": lambda x="0(I)R(-)": self.set_item(x)
      },
      {
        "viewclass": "Item",
        "text": "0(I)R(+)",
        "on_release": lambda x="0(I)R(+)": self.set_item(x)
      }
    ]
    self.menu = MDDropdownMenu(
      caller=self.ids.dropItem,
      items=menu_list,
      width_mult=4
    )
    self.menu.bind()

  def set_item(self, text_item):
    self.ids.dropItem.text = text_item
    self.menu.dismiss()

class EnterScreen(Screen):
  def __init__(self, **kwargs):
    super(EnterScreen, self).__init__(**kwargs)
    self.store = JsonStore('credentials.json') # JsonStore для
збереження логіна та паролю

  def connect(self):
    input_login = self.manager.get_screen('enter').ids ['login'].text
    input_password = self.manager.get_screen('enter').ids
['password'].text
    remember_me = self.manager.get_screen('enter').ids
['remember_me'].active

    server = r'localhost\SQLEXPRESS'
    database = 'CarDB'

    cnxn = pyodbc.connect(
      'DRIVER={ODBC Driver 17 for SQL Server};'
      f'SERVER={server};'
      f'DATABASE={database};'
      'Trusted_Connection=yes;'
    )

    cursor = cnxn.cursor()

    sql = "SELECT count(*) FROM Users where login='" + str(input_login)
+ "' and password='" + str(input_password) + "'"
    cursor.execute(sql)
    data = cursor.fetchone()
    count = data [0]

    if count == 0:
```

```
      self.invalidLogin()   #   Виклик   функції   для   відображення
повідомлення про невірний логін або пароль
      #self.clear()
    else:
      if remember_me:
        self.saveCredentials(input_login, input_password)
      query = "SELECT * FROM users WHERE login='" + str(input_login)
+ "' and password='" + str(input_password) + "'"
      cursor.execute(query)
      cnxn.commit()
      self.manager.current = "home"

    cnxn.close()

  def saveCredentials(self, login, password):
    self.store.put('credentials',  login=login,  password=password)  #
Збереження логіна та паролю

  def loadCredentials(self):
    if 'credentials' in self.store:
      credentials = self.store.get('credentials')
      login = credentials ['login']
      password = credentials ['password']
      self.manager.get_screen('enter').ids ['login'].text = login
      self.manager.get_screen('enter').ids    ['password'].text    =
password

  def invalidLogin(self):
    content = Label(text="Невірний логін або пароль.")
    close_button  =  Button(text="Закрити",  size_hint=(None,  None),
size=(150, 50))
    popup  =  Popup(title="Помилка  авторизації",  content=content,
size_hint=(None, None), size=(400, 200))
    close_button.bind(on_release=popup.dismiss)
    content.add_widget(close_button)
    popup.open()

class StatisticScreen(Screen):
  pass
class MainApp(MDApp):
  def build(self):
    Window.size = (390, 750)
    self.theme_text_color = "White"
    return WindowManager()

MainApp().run()
```