НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Мобільний застосунок для виявлення сонливості у водіїв для запобігання ДТП

Виконав студент IV курсу, групи ІТ-94
(шифр групи)
Курілець Валерій Андрійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., доц., Ліхоузова Т. А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації доцент, к.т.н., доц., Ліщук К. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., доц. Солдатова М.О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Курільцю Валерію Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Мобільний застосунок для виявлення сонливості у водіїв для запобігання ДТП

керівник проєкту Ліхоузова Тетяна Анатоліївна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»квітня 2023 р. №_____

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання _____

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.
4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема структурна класів програмного забезпечення

3) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2023 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв’язання задачі	20.03.2023	
3	Постановка та формалізація задачі	23.03.2023	
4	Розробка інформаційного забезпечення	14.04.2023	
5	Алгоритмізація задачі	16.04.2023	
6	Обґрунтування вибору використаних технічних засобів	27.04.2023	
7	Розробка програмного забезпечення	10.05.2023	
8	Налагодження програми	24.05.2023	
9	Виконання графічних документів	01.06.2023	
10	Оформлення пояснювальної записки	02.06.2023	
11	Подання ДП на попередній захист	02.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент _____ Валерій КУРІЛЕЦЬ
(підпис) (ініціали, прізвище)

Керівник _____ Тетяна ЛІХОУЗОВА
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 18 таблиць, 15 рисунків та 12 джерел – загалом 61 сторінка.

Дипломний проєкт присвячений розробці мобільного застосунку для виявлення сонливості у водіїв для запобігання ДТП.

Метою роботи є зменшити кількість ДТП шляхом розробки мобільного застосунку для виявлення сонливості водія та його попередження про необхідність відпочинку.

Об'єкт дослідження: мобільний застосунок для виявлення сонливості у водіїв для запобігання ДТП.

Предмет дослідження: алгоритм розпізнавання ознак сонливості, технологія обробки зображення обличчя, процеси розроблення мобільного застосунку, модифікації, аналізу, забезпечення якості, тестування, впровадження і супроводження програмного забезпечення.

У розділі "Аналіз вимог до програмного забезпечення" представлено змістовний опис і аналіз предметної області. Також виконано аналіз існуючих технологій та успішних ІТ-проєктів. Поставлено задачу та сформульовано цілі розробки.

У другому розділі "Моделювання та конструювання програмного забезпечення" розглянуто мову програмування, а також представлено структуру проекту та опис класів. Розглянуто архітектуру програмного забезпечення та проведено конструювання програмного забезпечення.

У третьому розділі "Аналіз якості та тестування програмного забезпечення" описано процеси тестування та представлено контрольний приклад.

У четвертому розділі "Впровадження та супровід програмного забезпечення" описано процес розгортання програмного забезпечення та підтримки його функціонування.

КЛЮЧОВІ СЛОВА: МОБІЛЬНИЙ ДОДАТОК, РОЗПІЗНОВАННЯ ОЗНАК СОНЛИВОСТІ, РОЗПІЗНОВАННЯ ЗАПЛЮЩЕНИХ ОЧЕЙ, ANDROID STUDIO, ML KIT. .

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 18 tables, 15 figures and 12 sources – in total 61 pages.

The purpose of the diploma project is to reduce the number of road accidents by developing a mobile application to detect driver drowsiness and warn about the need for rest.

KEYWORDS: MOBILE APP, CLOSED EYES DETECTION, ANDROID, ANDROID STUDIO, DROWSINESS DETECTION, ML KIT.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-9411.045490.01.91	Технічне завдання	12	
3	А4	КПІ.ІТ-9411.045490.02.81	Пояснювальна записка	61	
4	А4	КПІ.ІТ-9411.045490.03.12	Текст програми	5	
5	А4	КПІ.ІТ-9411.045490.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-9411.045490.05.34	Керівництво користувача	6	
7	А3	КПІ.ІТ-9411.045490.06.99	Схема структурна варіантів	1	
			використань		
8	А3	КПІ.ІТ-9411.045490.06.99	Схема структурна класів	1	
			програмного забезпечення		
9	А3	КПІ.ІТ-9411.045490.06.99	Креслення вигляду екранних форм	1	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ-9411.045490.00.90		
Розроб.		Курілець В.А.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Ліхоузова Т.А.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-94	
Н.контр.		Ліщук К.І.					
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2023 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ВИЯВЛЕННЯ СОНЛИВОСТІ У ВОДІЇВ ДЛЯ ЗАПОБІГАННЯ ДТП

Технічне завдання

КПІ.ІТ- 9411.045490.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Валерій КУРІЛЕЦЬ

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ВИЯВЛЕННЯ СОНЛИВОСТІ У ВОДІЇВ ДЛЯ ЗАПОБІГАННЯ ДТП.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного застосунку під платформу Android «Мобільний застосунок для виявлення сонливості у водіїв для запобігання ДТП» [КПІ.ІТ-9411.045490.01.91], котра використовується для виявлення сонливості та сповіщення у необхідності відпочинку та призначена для водіїв для запобігання ДТП.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки Мобільного застосунка для виявлення сонливості у водіїв для запобігання ДТП є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для мобільних пристроїв на базі Android для виявлення ознак сонливості.

Метою розробки є зменшити кількість ДТП шляхом розробки мобільного застосунку для виявлення сонливості водія та його попередження про необхідність відпочинку.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

– Виведення діалогового вікна з дозволом використовувати камеру мобільного пристрою. Зображено на рисунку 4.1.

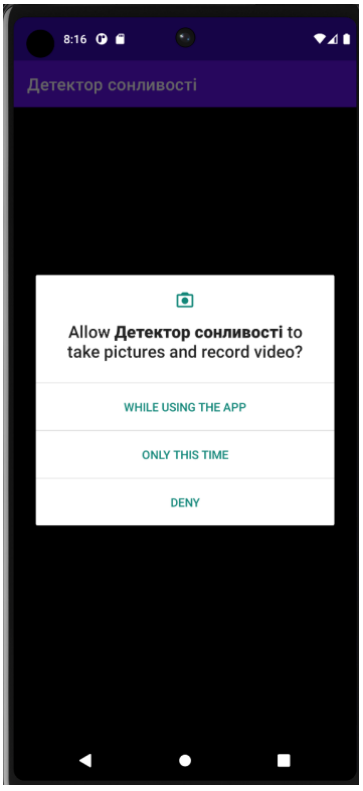

Рисунок 4.1 – Надання дозволів

4.1.2 Для користувача:

– Розпізнавання ознак сонливості на головному екрані мобільного застосунка. Зображено на рисунку 4.2.

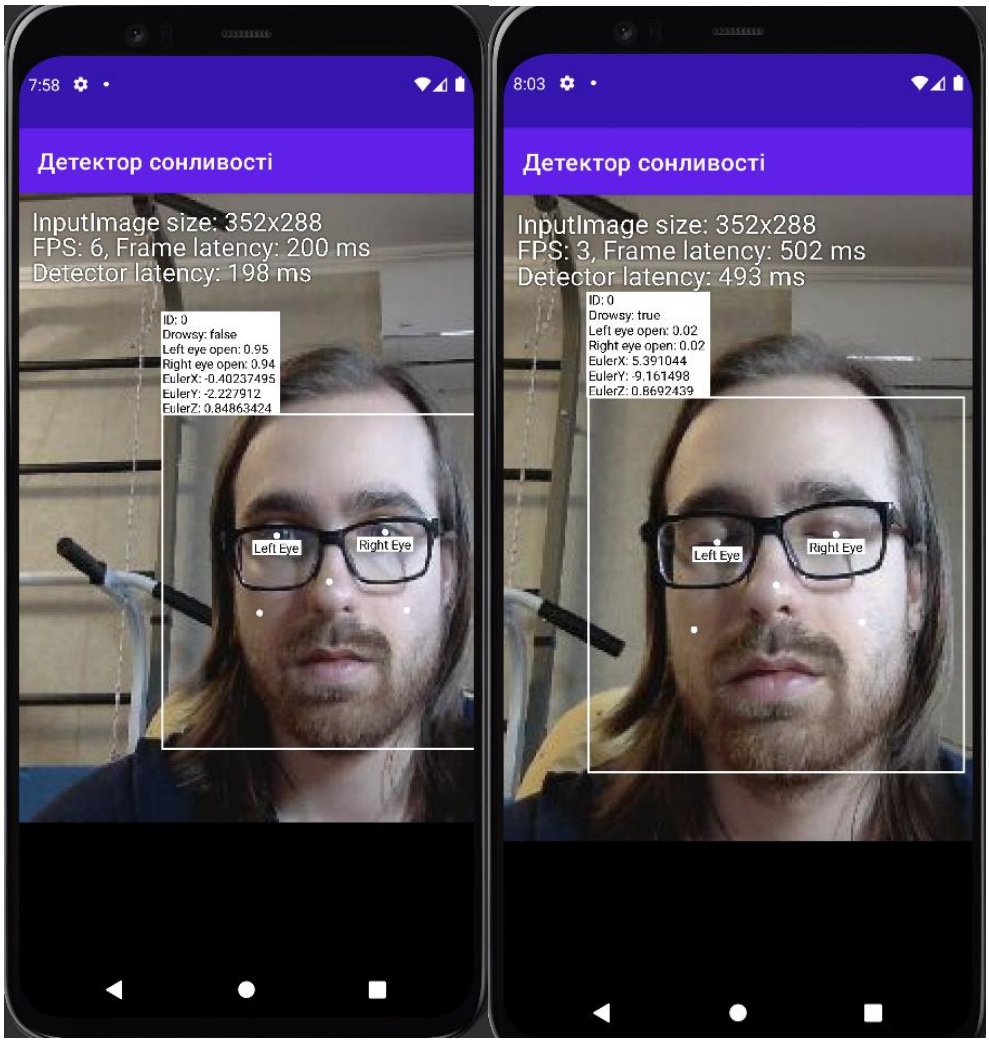

Рисунок 4.2 – Розпізнавання ознак сонливості

– Надсилання гучного аудіо-повідомлення у випадку розпізнавання ознак сонливості.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації. Застосунок повинен бути стабільним і працювати без збоїв або непередбачуваних відмов. Він не повинен виходити з ладу, коли користувач використовує його для виявлення сонливості. Якщо все ж виникає помилка або збій, застосунок повинен бути здатний відновлюватись до нормального режиму роботи.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на мобільних застосунках на базі Android.

Мінімальна конфігурація технічних засобів:

– Версія операційної системи: Android 5.0 (Lollipop) або більш пізня версія.
– Обсяг пам'яті: Мінімум 2 ГБ оперативної пам'яті (RAM).
– Процесор: Чотирьохядерний процесор з тактовою частотою 1,4 ГГц або більше.
– Обсяг пам'яті: Мінімум 2 ГБ оперативної пам'яті (RAM).
– Мінімальна роздільна здатність екрану 480x800 пікселів.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційної системи Android.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені у вигляду відеопотіку з бек-камери мобільного застосунку.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в форматі гучного аудіо повідомлення у випадку розпізнавання ознак сонливості.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Java.

4.5.4 Вимоги до середовища розробки.

Розробку виконати на платформі Android Studio.

4.5.5 Вимоги до представленню вихідних кодів .

Вихідний код програми має бути представлений у вигляді APK файлу.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Згенерувати інсталяційну версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;

– технічне завдання;

– текст програми;

– програма та методика тестування;

– керівництво користувача;

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;

– схема структурна класів програмного забезпечення;

– креслення вигляду екранних форм;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

<Брати з листа завдання>

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	10.03	
2.	Розробка технічного завдання	23.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	14.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка

до дипломного проєкту

на тему: Мобільний застосунок для виявлення сонливості у водіїв для запобігання ДТП

КПІ.ІТ-9411.045490.02.81

Київ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки
API	–	Application programming interface, прикладний програмний Інтерфейс
SDK	–	Software development kit
IT	–	Інформаційні технології
ER	–	Entity-Relation diagram
ОС	–	Операційна система
БД	–	База даних

ВСТУП

Сонливість водіїв є критичним фактором, що сприяє виникненню дорожньо-транспортних пригод у всьому світі. Аварії, спричинені втомою, можуть мати серйозні наслідки, призводячи до травм, смертей і матеріальних збитків. Системи виявлення сонливості водія відіграють вирішальну роль у зниженні цих ризиків, надаючи своєчасні попередження або втручання для запобігання аваріям, спричиненим сонливим водієм.

Різні дослідження показують, що близько 20% всіх дорожньо-транспортних пригод пов'язані з відчуттям сонливості за кермом, а на деяких дорогах - до 50% [1]. На основі звітів поліції та лікарень за 2017 рік Національна Адміністрація Безпеки Дорожнього Руху Америки (англ. NHTSA) визначила 91 000 автомобільних аварій, спричинених через втому. У результаті цих аварій люди отримали до 50 000 тілесних ушкоджень. У 2019 році 697 смертельних випадків сталися через втому водіїв. Однак NHTSA визнає, що важко визначити точну кількість аварій, травм і смертей, спричинених водіями, які відчули сонливість під час керування, і що ці цифри є заниженими [2]. Наприклад, дослідження, проведене Фондом Безпеки Дорожнього Руху Американської Автомобільної Асоціації, показало, що щороку трапляється понад 320 000 ДТП через втому, в тому числі 6400 аварій зі смертельними наслідками [3].Такі високі цифри свідчать про те, що водіння в стані сонливості є серйозною проблемою, яку необхідно вирішувати, щоб зменшити її вплив. На щастя, можна виявити сонливість водія на ранніх стадіях і подати сигнал оповіщення, щоб уникнути потенційної аварії.

Системи виявлення сонливості водія мають потенційне застосування в різних сферах, зокрема:

- системи виявлення сонливості можуть бути інтегровані в транспортні засоби як частина ADAS, підвищуючи безпеку водія та запобігаючи аваріям, спричиненим сонливістю;

- виявлення сонливості може використовуватися в системах громадського транспорту, міжміських вантажних перевезеннях та інших транспортних секторах, щоб забезпечити пильність водія і знизити ризик аварій;
- системи виявлення сонливості водіїв можуть використовуватися в наукових дослідженнях і освітніх програмах для збору даних, підвищення обізнаності про сонливе водіння і розробки ефективних контрзаходів.

Варто зазначити, що сфера виявлення сонливості водіїв постійно розвивається, а постійні дослідження та вдосконалення сприяють подальшому підвищенню точності, надійності та практичного застосування.

Зазвичай рішення для виявлення сонливості виробники автомобілів представляють на ринку у складі "Удосконаленої системи допомоги водію", так званої ADAS(англ. Advanced driver-assistance systems). Вона підрозділяється на рівні. Технологія розпізнавання сонливості найчастіше починається використовуватися у частково автоматизованих машинах, коли камери салону слідкують, щоб очі водія зберігали увагу за рухом. Згідно поясненню рівнів організацією SAE (англ. Society of Automotive Engineers) це другий рівень ADAS -- “hands off”(укр. Руки геть) [4]. У 2019-му році в Європі продано 325 000 легкових автомобілів з функціями ADAS 2-го рівня, що становить 8% від усіх проданих нових автомобілів [5]. Згідно з дослідницьким звітом компанії Canalys за 2021 рік, приблизно 33% нових автомобілів, проданих у Сполучених Штатах, Європі, Японії та Китаї, були оснащені функціями ADAS. Фірма також спрогнозувала, що до 2030 року п'ятдесят відсотків усіх автомобілів на дорогах будуть оснащені ADAS [6]. З цього можна зробити висновок, що забезпечення автомобільного ринку технологією оповіщення, якщо водій втомлений, не значне. Тому вкрай важливим є забезпечення власників авто, які не містять цієї системи – технологією виявлення сонливості, яка не є залежною від марки автомобілів, на цей

перехідний період, доки ця система не буде повністю забезпечуватися автовиробниками.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Виявлення сонливості водія - це розробка та впровадження систем або програм, які можуть відстежувати та оцінювати рівень сонливості водіїв у режимі реального часу. Ці системи використовують різні датчики та алгоритми для аналізу фізіологічних і поведінкових показників водія, щоб визначити рівень його пильності або сонливості. Виявляючи ознаки сонливості, такі як заплющення очей, кивання головою або зміни в поведінці водія, ці системи мають на меті забезпечити своєчасне попередження або втручання для запобігання аваріям, спричиненим втомою водія.

Розробка програм виявлення сонливості водія передбачає мультидисциплінарні підходи, що поєднують комп'ютерні науки, штучний інтелект, обробку сигналів та дослідження людського фактору [7, 8]. Ключовими напрямками в розробці програми є наступні.

- Інтеграція різних типів датчиків, таких як інфрачервоні камери, датчики рульового управління, датчики виїзду зі смуги руху та монітори серцевого ритму, для збору відповідних даних з тіла водія та транспортного засобу. Ці датчики забезпечують вхідні дані для алгоритмів виявлення сонливості.
- Збір різноманітних даних, включаючи рухи очей, вираз обличчя, манеру керування та фізіологічні сигнали, від водія та навколишнього середовища. Аналіз цих даних за допомогою методів машинного навчання та розпізнавання образів для побудови надійних моделей для виявлення сонливості [9].
- Розробка алгоритмів, які можуть ефективно обробляти зібрані дані і точно визначати показники сонливості. Ці алгоритми можуть використовувати різні методи, такі як обробка зображень, аналіз сигналів, глибоке навчання або поєднання цих методів для виявлення та класифікації патернів, пов'язаних із сонливістю [10].

- Розробка систем, здатних відстежувати сонливість водія в режимі реального часу, забезпечуючи швидке виявлення і своєчасне попередження. Це передбачає оптимізацію обчислювальної ефективності алгоритмів для швидкого аналізу та прийняття рішень.
- Розробка зручних інтерфейсів і механізмів зворотного зв'язку для інформування водія про стан сонливості. Це може включати слухові або візуальні сповіщення, тактильний зворотній зв'язок або навіть адаптивні системи керування транспортним засобом, які можуть допомогти водієві при виявленні сонливості.
- Проведення всебічного тестування та валідації розроблених систем за різних умов та сценаріїв водіння. Забезпечення точності, надійності та безпеки програми виявлення сонливості перед розгортанням у реальних транспортних засобах.

1.2 Змістовний опис і аналіз предметної області

Процес реалізації функції виявлення сонливості водія в програмному забезпеченні на сучасному етапі розвитку ІТ-технологій включає в себе кілька ключових етапів:

Відповідні дані збираються за допомогою різних датчиків, таких як камери, датчики кута повороту керма та фізіологічні датчики. Ці дані включають рухи очей, міміку, манеру кермування, частоту серцевих скорочень та інші показники, пов'язані з сонливістю.

Зібрані дані обробляються для видалення шуму, нормалізації сигналів і виділення релевантних ознак. На цьому етапі можуть використовуватися такі методи, як обробка зображень, фільтрація сигналів та алгоритми вилучення ознак.

Алгоритми машинного навчання та штучного інтелекту використовуються для аналізу попередньо оброблених даних і виявлення патернів, що вказують на сонливість. Для розробки надійних і точних моделей

виявлення сонливості можна використовувати різні методи, такі як глибоке навчання, машини опорних векторів або дерева рішень.

Зручний інтерфейс призначений для інформування водія про стан сонливості. Це може включати візуальні попередження, звукові попередження або тактильний зворотній зв'язок для забезпечення ефективної комунікації та швидкої реакції водія.

Недоліки поточного стану справ у розробці ІТ для виявлення сонливості водіїв:

- Існуючі рішення часто покладаються на інтеграцію конкретного обладнання в транспортні засоби, що обмежує їхню доступність, масштабованість та збільшує ціну. Це ускладнює використання технології виявлення сонливості користувачами, які не мають сумісних транспортних засобів або пристроїв на вторинному ринку.
- Інтеграція програмного забезпечення для виявлення сонливості з існуючими автомобільними системами може бути складною і вимагати співпраці з виробниками автомобілів або доступу до API. Це може перешкоджати широкому впровадженню та розгортанню таких систем.
- Досягнення високої точності у виявленні сонливості залишається складним завданням. Алгоритми можуть іноді генерувати помилкові спрацьовування, запускаючи попередження, коли водій насправді не є сонним, або помилкові негативні результати, не виявляючи сонливість в деяких випадках.
- Деякі водії можуть мати занепокоєння щодо наслідків моніторингу рівня їхньої сонливості для приватності. Забезпечення згоди користувачів і вирішення проблем конфіденційності є важливими для широкого впровадження та довіри користувачів.

Можливі шляхи покращення ситуації у сфері ІТ-розробок для виявлення сонливості водіїв:

- Розробка незалежних мобільних застосунків для виявлення сонливості дозволяє користувачам отримати доступ до технології і використовувати її незалежно від виробника автомобіля або моделі транспортного засобу. Це підвищує доступність і сприяє широкому впровадженню.
- Розробка алгоритмів, які можуть використовувати звичайні датчики, доступні в смартфонах, такі як фронтальні камери і акселерометри, зменшує залежність від апаратного забезпечення і підвищує сумісність технології виявлення сонливості на різних пристроях.
- Використання хмарних обчислень і зберігання даних дозволяє ефективніше обробляти та аналізувати дані про сонливість. Це дає змогу здійснювати моніторинг та аналіз у режимі реального часу, не покладаючись виключно на обчислювальні потужності пристрою користувача.

Постійні дослідження та співпраця між IT-розробниками, автовиробниками та науково-дослідними установами можуть сприяти вдосконаленню алгоритмів, методів збору даних та зручних інтерфейсів, усуненню поточних недоліків та підвищенню загальної ефективності систем виявлення сонливості. Це можливо завдяки розповсюдженню застосунка як проект з відкритим кодом.

1.3 Аналіз існуючих технологій та успішних IT-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації мобільного застосунку для виявлення сонливості у водіїв для запобігання ДТП. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

Відомі на сьогоднішній день алгоритмічне програмне забезпечення для виявлення сонливості водія:

- Алгоритми на основі відстеження руху очей – ці алгоритми використовують технологію відстеження руху очей для моніторингу рухів очей, таких як частота моргання, тривалість закриття очей і напрямок погляду. Аналізуючи ці патерни, алгоритми можуть виявляти ознаки сонливості. Приклади включають використання методів машинного навчання, таких як машини опорних векторів (SVM) або згорткові нейронні мережі (CNN) для класифікації поведінки очей.
- Алгоритми, які аналізують вираз обличчя, наприклад, опускання повік або зміни в активності лицьових м'язів, можуть дати уявлення про рівень сонливості. Зазвичай використовуються такі методи, як виявлення орієнтирів на обличчі, виокремлення ознак та алгоритми машинного навчання (наприклад, приховані марковські моделі або дерева рішень).
- Алгоритми можуть аналізувати фізіологічні сигнали, такі як частота серцевих скорочень, електрокардіограма (ЕКГ) або електроенцефалограма (ЕЕГ), щоб виявити сонливість. Ознаки, витягнуті з цих сигналів, використовуються для навчання класифікаторів, щоб розрізняти сонний і бадьорий стан.

Технічні рішення для реалізації мобільного застосунку для виявлення сонливості водія:

- Використовування датчиків, доступних в смартфонах, такі як фронтальна камера. Наприклад, методи комп'ютерного зору можна використовувати для аналізу рухів очей і виразу обличчя, зафіксованих камерою.
- Розробка моделі машинного навчання, використовуючи бібліотеки, такі як TensorFlow, Deeplearning4j, Weka, MLkit, PYtorch для навчання і розгортання алгоритмів виявлення сонливості. Ці бібліотеки надають інструменти для попередньої обробки даних, вилучення ознак і побудови моделей класифікації.

- Впровадження механізмів моніторингу даних датчиків у реальному часі та безперервного аналізу за допомогою фонових сервісів або виділених потоків. Це забезпечує швидке виявлення сонливості та своєчасне сповіщення водія.

Допоміжне програмне забезпечення та засоби розробки:

- OpenCV – бібліотека комп'ютерного зору з відкритим вихідним кодом, яка надає різні алгоритми обробки зображень і комп'ютерного зору. Її можна використовувати для таких завдань, як розпізнавання очей, виявлення орієнтирів на обличчі та відстеження.
- JavaFX – Фреймворк на основі Java для створення користувацьких інтерфейсів. Може використовуватися для створення зручного інтерфейсу для мобільного застосунку, включаючи візуальні сповіщення та зворотній зв'язок з водієм.
- ML Kit - це мобільний SDK (Software Development Kit), розроблений компанією Google, який надає попередньо навчені моделі машинного навчання та API для інтеграції можливостей машинного навчання в застосунки для Android та iOS. ML Kit спрощує інтеграцію функцій машинного навчання, таких як розпізнавання зображень, розпізнавання тексту, розпізнавання облич і сканування штрих-кодів, у мобільні застосунки. Він пропонує варіанти обробки як на пристрої, так і в хмарі, що дозволяє розробникам вибрати найбільш підходящий підхід на основі таких факторів, як вимоги до роботи в реальному часі та доступні ресурси пристрою.
- Android Studio - це офіційне інтегроване середовище розробки (IDE) для розробки застосунків для Android. Воно засноване на IDE IntelliJ IDEA і надає повний набір інструментів, бібліотек і функцій для розробки, тестування і налагодження Android-застосунків. Android Studio має зручний інтерфейс, редактор коду з інтелектуальним завершенням коду та рефакторингом, візуальні редактори макетів для проектування інтерфейсів застосунків, а також вбудовані

емулятори та профілі пристроїв для тестування застосунків. Вона підтримує такі мови програмування, як Java та Kotlin, і надає ряд інструментів для розробки та налагодження, які спрощують процес розробки застосунків.

- Xcode - це інтегроване середовище для розробки застосунків для iOS, подібне до Android Studio.

Готові програмні рішення:

- Google Mobile Vision API – надає попередньо навчені моделі та API для розпізнавання облич, виявлення орієнтирів і відстеження облич. Його можна інтегрувати в застосунок, щоб використовувати аналіз виразу обличчя для виявлення сонливості.
- Emotiv SDK – це SDK для доступу та обробки даних ЕЕГ з гарнітур Emotiv. Він може бути використаний для включення функції виявлення сонливості на основі ЕЕГ у мобільний застосунок.
- ML Kit Visual QuickStart App – це готове програмне рішення, яке імплементує функції розпізнавання обличчя.

Важливо зазначити, що конкретний вибір інструментів та алгоритмів може змінюватися залежно від вимог, ресурсів та сумісності з цільовою мобільною платформою.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

Водій не засинає раптово, без певних ознак. Приклади таких ознак включають [7, 8]:

- труднощі з триманням очей розплющеними;
- позіхання;
- часте моргання;
- труднощі з концентрацією уваги;
- з'їжджання зі смуги та затримка реакції на зміну руху;
- кивання головою;
- невиправдана зміна швидкості.

Коли мобільний застосунок не може аналізувати поведінку машини, як наприклад зміну швидкості та зміну руху, він цілком може аналізувати обличчя водія, якщо помістити телефон у якості камери або використати вже встановлену камеру з салону та передати зображення з неї на мобільний застосунок.

Отже, наше ПЗ повинно містити алгоритм для аналізу поведінки очей. Це завдання досягається шляхом визначення місцезнаходження та сегментації ділянки ока від решти обличчя. Потім проводиться бінаризація та маркування зображень, щоб зрозуміти, які зображення відображають виникнення сонливості, а які ні. Потім, аналізуючи моргання і їх тривалість, алгоритм може виявити сонливість, якщо очі закриті на більш тривалий час, ніж час, необхідний для моргання ока. Імплементуючи цей алгоритм за допомогою мови програмування та інтегруючи цю систему з пристроєм оповіщення, вона може бути корисною для зменшення кількості нещасних випадків, спричинених втратою свідомості.

Аналогічно можна аналізувати кивання головою та моргання, визначивши яка сама кількість цих рухів обличчям та очами є надмірною.

Серед відомих алгоритмів, алгоритми виявлення позіхання та заплющення очей відносно легко реалізувати, використовуючи лише смартфон. Ці алгоритми використовують вбудовану камеру для захоплення зображень обличчя та аналізу рис обличчя в режимі реального часу. У порівнянні з більш складними алгоритмами, які вимагають застосункового обладнання або датчиків, ці алгоритми забезпечують практичне і доступне рішення для виявлення сонливості в мобільному застосунку.

Відомі програмні архітектури та архітектурні патерни:

- MVC - це широко використовуваний архітектурний патерн, який розділяє застосунок на три взаємопов'язані компоненти: модель (дані та бізнес-логіка), представлення (користувацький інтерфейс) та контролер (обробляє вхідні дані користувача та оновлює модель і

представлення). MVC забезпечує чіткий розподіл завдань і сприяє повторному використанню коду та його підтримці.

- MVVM - це архітектурний патерн, який покращує розподіл завдань між моделлю даних, користувацьким інтерфейсом та бізнес-логікою. Він вводить рівень моделі подання, який є посередником між поданням і моделлю. MVVM сприяє зв'язуванню даних і спрощує реалізацію реактивних і керованих подіями користувацьких інтерфейсів.

У контексті мобільних застосунків клієнт-серверна архітектура передбачає клієнтський застосунок, встановлений на пристрої користувача, який взаємодіє з віддаленим сервером. Сервер виконує обчислювально інтенсивні завдання, такі як обробка машинного навчання, і надсилає результати назад клієнту. Така архітектура дозволяє розвантажити ресурсомісткі завдання і забезпечує оновлення та синхронізацію даних у режимі реального часу.

Платформи та фреймворки:

- Платформа Android є найпоширенішою платформою для розробки мобільних застосунків для виявлення сонливості водіїв. Вона надає надійне середовище для розробки, доступ до апаратних функцій, таких як камера та датчики, а також широку базу користувачів.
- Платформа iOS використовується для розробки мобільних застосунків для пристроїв Apple, включаючи iPhone та iPad. Вона пропонує багате середовище розробки, потужну інтеграцію з апаратним забезпеченням та кураторське середовище App Store.

Порівняльний аналіз.

Розглядаючи програмні архітектури та архітектурні патерни для мобільних застосунків для виявлення сонливості водіїв, зазвичай використовують патерни MVC та MVVM. MVC забезпечує структурований і модульний підхід, тоді як MVVM додає рівень абстракції і спрощує прив'язку

даних, що робить його придатним для реактивних і керованих подіями інтерфейсів.

Що стосується платформ, Android та iOS є основними платформами для розробки мобільних застосунків. Android пропонує ширший вибір пристроїв та більшу кількість користувачів, тоді як iOS забезпечує більш контрольовану та узгоджену апаратну та програмну екосистему.

Крім того, вибір між клієнт-серверною архітектурою та підходом до обробки на пристрої залежить від таких факторів, як вимоги до роботи в режимі реального часу, міркування щодо конфіденційності даних та наявність обчислювальних ресурсів. Архітектура клієнт-сервер дозволяє перекласти важку обробку на віддалений сервер, але вона вимагає надійного мережевого з'єднання. Обробка на пристрої, з іншого боку, забезпечує можливості роботи в реальному часі та конфіденційність даних, але може бути обмежена обчислювальною потужністю пристрою.

Для розробки ПЗ буде використано Android платформу та MVC архітектуру як найпоширеніші технології, які є порівняно легкими в імплементації та поширені на ринку.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Android Studio пропонує повний набір інструментів та функцій, спеціально розроблених для розробки застосунків для Android, включаючи можливості редагування коду, налагодження та тестування. У той час як Android Studio спеціально розроблена для розробки застосунків для Android, оскільки вона орієнтована на розробку застосунків для Android, який займає більшу частину ринку мобільних застосунків. Java є підходящою мовою програмування для розробки для Android.

Мова програмування Java обрана для розробки застосунків, оскільки вона широко використовується для розробки застосунків для Android і підтримується Android Studio. Java забезпечує знайоме та надійне середовище програмування для реалізації алгоритмів та інтеграції різних компонентів

програми. Swift - це мова програмування, яка використовується для розробки застосунків для iOS та macOS. Java, з іншого боку, широко використовується для розробки застосунків для Android. Обидві мови мають свої сильні сторони та підтримку екосистеми, але вибір залежить від цільової платформи (Android чи iOS).

ML Kit, мобільний SDK від Google, пропонує попередньо навчені моделі машинного навчання та API, які можна використовувати для таких завдань, як виявлення орієнтирів на обличчі або обробка зображень. ML Kit спрощує інтеграцію можливостей машинного навчання в застосунок, не вимагаючи глибоких знань або досвіду в галузі машинного навчання.

Core ML - це фреймворк машинного навчання від Apple для розробки застосунків для iOS. Він дозволяє розробникам інтегрувати моделі машинного навчання в застосунки для iOS. З іншого боку, ML Kit - це мобільний SDK для машинного навчання від Google, який підтримує платформи Android та iOS. Хоча обидва фреймворки надають схожі можливості, перевага ML Kit полягає в його крос-платформенній підтримці.

TensorFlow, фреймворк машинного навчання з відкритим вихідним кодом, можна використовувати для розробки та навчання власних моделей машинного навчання, якщо це необхідно. Він надає широкий спектр інструментів і ресурсів для побудови, навчання та розгортання моделей машинного навчання, що дозволяє використовувати більш досконалі алгоритми виявлення сонливості, якщо це необхідно.

Коли мова йде про автоматизацію збірки та управління залежностями в контексті розробки програмного забезпечення для виявлення сонливості, Gradle є одним з найпопулярніших інструментів. Хоча в цій сфері є й інші конкуренти, такі як Maven та Ant, Gradle пропонує кілька переваг, які роблять його підходящим вибором для багатьох розробників.

Gradle - це потужний інструмент автоматизації збірки, який використовує доменно-специфічну мову (DSL) на основі Groovy або Kotlin для написання сценаріїв конфігурацій збірки.

Він забезпечує гнучкий і декларативний підхід до визначення конфігурацій збірки, що полегшує управління складними проектами з великою кількістю модулів і залежностей.

Gradle пропонує ефективне управління залежностями, дозволяючи розробникам легко визначати та вирішувати залежності з локальних репозиторіїв або зовнішніх джерел, таких як Maven Central або JCenter.

Він підтримує інкрементні збірки, що означає, що він перебудовує тільки необхідні частини проекту, що призводить до скорочення часу збірки.

Gradle добре інтегрується з різними IDE, включаючи Android Studio, що робить його популярним вибором для розробки під Android.

Maven - це широко розповсюджений інструмент автоматизації збірки та управління проектами, який використовує формат конфігурації на основі XML. Він забезпечує підхід "конвенції над конфігурацією", спрощуючи налаштування проекту та стандартизуючи структуру проекту.

Maven має всеосяжне сховище залежностей, відоме як Центральний репозиторій, що дозволяє легко знаходити залежності проекту та керувати ними. Він пропонує надійне управління залежностями, керування версіями та вирішення конфліктів.

Maven підтримує широкий спектр плагінів, що забезпечує застосункову функціональність та інтеграцію з різними інструментами та фреймворками.

Ant - це інструмент автоматизації збірки, який використовує формат конфігурації на основі XML. Він забезпечує процедурний підхід до створення проектів, де розробники явно визначають завдання та їх залежності. Ant пропонує високий рівень кастомізації та гнучкості, що дозволяє розробникам створювати високоспеціалізовані процеси збірки. У ньому немає вбудованих функцій управління залежностями, тому розробникам доводиться вручну керувати залежностями проекту. Ant добре підходить для простих або застарілих проектів, але може стати більш складним у підтримці у великих, багатомодульних проектах.

У контексті дипломної програми, Gradle має кілька переваг. Його гнучкість, ефективне управління залежностями та підтримка інкрементних збірок є особливо корисними для управління складними проектами з багатьма модулями та залежностями. Крім того, інтеграція Gradle з Android Studio робить його підходящим вибором для розробки застосунків для Android.

1.3.3 Аналіз відомих програмних продуктів

Порівнюючи мобільний застосунок для виявлення сонливості за допомогою алгоритму відстеження руху очей на Android з іншими рішеннями, такими як Optalert Fatigue Risk Profiler і Eyesight Technologies DriverSense, слід враховувати кілька факторів:

Мобільний застосунок для виявлення сонливості з алгоритмом відстеження очей як і Eyesight Technologies DriverSense пропонує моніторинг рівня сонливості в режимі реального часу за допомогою технології відстеження очей.

Optalert Fatigue Risk Profiler забезпечує моніторинг втоми за допомогою натільних пристроїв, які вимірюють рухи повік і рівень пильності.

Точність дипломної програми залежить від ефективності алгоритму відстеження руху очей та якості камери пристрою.

Optalert Fatigue Risk Profiler використовує натільні пристрої, які безпосередньо вимірюють рухи повік, надаючи точні дані для аналізу втоми.

Застосунок Детектор Сонливості спеціально розроблений для платформи Android і може використовувати специфічні для Android функції та бібліотеки.

Eyesight Technologies DriverSense та Optalert Fatigue Risk Profiler може мати сумісність з тільки операційними системами автовиробника.

Застосунок можна інтегрувати в існуючі програми або системи Android за допомогою відповідних API та опцій налаштування.

Застосунок є безкоштовним та легко розповсюдженим через Google Play або github.

Optalert Fatigue Risk Profiler - це комерційне рішення, яке може вимагати придбання натільних пристроїв та ліцензій на відповідне програмне забезпечення.

Eyesight Technologies DriverSense може мати власну структуру цін, засновану на ліцензійних угодах з автовиробниками або операторами автопарків.

Для порівняння дипломного проекту з аналогами можна скористатись таблицею 1.1

Таблиця 1.1 – Порівняння з аналогами

Функціонал	Дипломний проєкт	Optalert Fatigue Risk Profiler	Eyesight Technologies DriverSense	Пояснення
Виявлення позіхання за допомогою аналізу обличчя	-	+	+	
Виявлення заплющених очей на основі відстеження очей та аналізу моргання	+	+	+	

Продовження таблиці 1.1

Система оповіщення в реальному часі	+	+	+	
Інтеграція у спеціальне обладнання, таке як наприклад окуляри Optalert, для виявлення рухів очей і закриття повік	-	+	-	
Кросплатфор-меність	+	-	-	
Режим калібровки на основі аналізу моделей	-	+	+	
Індивідуальна система оповіщення	+	-	-	Можливість налаштувати сигнал та тип самому
Інтеграція з іншими функціями програми	+	-	-	Обмежена сумісність з певними платформами або пристроями
Режим калібровки	-	+	+	

Продовження таблиці 1.1

Індивідуальна система оповіщення	+	-	-	Можливість налаштувати сигнал та тип самому
Інтеграція з іншими функціями програми	+	-	-	Обмежена сумісність з певними платформами або пристроями

1.4 Аналіз вимог до програмного забезпечення

Мобільний Java застосунок для виявлення сонливості водія має на меті виявити ознаки сонливості у водіїв та надати своєчасні попередження для забезпечення безпеки дорожнього руху.

Головними функціями програмного забезпечення є наступні.

Застосунок використовує камеру пристрою для постійного моніторингу міміки та рухів очей водія в режимі реального часу.

Алгоритм використовується для аналізу рухів очей водія та виявлення випадків заплющених очей протягом тривалого періоду, що вказує на потенційну сонливість.

При виявленні ознак сонливості застосунок запускає систему оповіщення, щоб повідомити водія. Це можуть бути звукові, вібраційні або візуальні попередження, щоб водій був уважним і міг вжити необхідних заходів для запобігання аварій.

В таблицях 1.2 - 1.5 наведені варіанти використання програмного забезпечення. Діаграма варіантів використання наведена у графічних матеріалах "Схема структурна варіантів використання".

Таблиця 1.2 - Варіант використання UC-01

Use case name	Налаштування доступів
Use case ID	UC-01
Goals	Надання необхідних дозволів для доступу до камери та функціоналу застосунку
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає налаштувати доступи
Pre-conditions	-
Flow of Events	Користувач запускає мобільний застосунок. Перед тим як він запуститься – на екрані телефона з'являється сповіщення дозволу для доступу до камери. Користувач надає доступи, переходячи до налаштувань у смартфоні або відмовляється.
Extension	-
Post-Condition	Надання доступів, перехід до UI мобільного застосунка.

Таблиця 1.3 - Варіант використання UC-02

Use case name	Моніторинг у режимі реального часу
Use case ID	UC-02
Goals	Постійна передача зображення обличчя до обробників обличчя.
Actors	Гість (незареєстрований користувач)
Trigger	Користувач успішно входить до мобільного застосунка.
Pre-conditions	-
Flow of Events	Користувач запускає мобільний застосунок.

Продовження таблиці 1.3

Extension	В випадку неможливості розпізнати обличчя вивести повідомлення.
Post-Condition	Перехід до алгоритмів аналізу обличчя.

Таблиця 1.4 - Варіант використання UC-3

Use case name	Виявлено, що очі стиснуті
Use case ID	UC-03
Goals	Попередити водія о втомі
Actors	Гість (незареєстрований користувач)
Trigger	Аналіз обличчя користувача показав, що він заплющив очі
Pre-conditions	Моніторинг у режимі реального часу
Flow of Events	Мобільний застосунок здійснює моніторинг у режимі реального часу.
Extension	В випадку неможливості розпізнати очі вивести повідомлення.
Post-Condition	Вивести повідомлення, яке попереджає водія о втомі.

Таблиця 1.5 - Варіант використання UC-4

Use case name	Повідомлення о втомі
Use case ID	UC-04
Goals	Попередити водія о втомі
Actors	Гість (незареєстрований користувач)
Trigger	Аналіз обличчя користувача показав, що він заплющив очі
Pre-conditions	Алгоритм аналізу очей повернув значення, що водій втомився
Events Flow	Здійснюються моніторинг у режимі реального часу.
Extension	-
Post-Condition	Повідомлення сигналізує о втомі

1.4.1 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблицях 1.6 – 1.10 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити у таблиці 1.11.

Таблиця 1.6 – Функціональна вимога FR-1

Назва	Надання доступу до камери
Опис	Система повинна отримати доступ до камери

Таблиця 1.7 – Функціональна вимога FR-2

Назва	Моніторинг у режимі реального часу
Опис	Система повинна отримувати зображення у режимі реального часу

Таблиця 1.8 – Функціональна вимога FR-3

Назва	Реалізація алгоритму розпізнавання заплющених очей
Опис	Система повинна розпізнати заплющенні очі на обличчі або що користувач часто моргає

Таблиця 1.9 – Функціональна вимога FR-4

Назва	Сповіщення о втомі
Опис	Система повинна передати сповіщення, якщо алгоритми розпізнали що користувач втомився

Таблиця 1.10 – Функціональна вимога FR-5

Назва	Налаштування застосунку
Опис	Система дає можливість налаштувати сигнал сповіщення.

	FR-01	FR-02	FR-03	FR-04	FR-05
UC-01	+	+	+	+	+
UC-02	+	+		+	+
UC-03	+	+	+	+	+
UC-04				+	
UC-05					+

Таблиця 1.11 – Матриця трасування вимог

1.4.2 Розроблення нефункціональних вимог

Застосунок повинен забезпечувати виявлення в реальному часі з мінімальною затримкою, щоб гарантувати своєчасні попередження та сповіщення.

Застосунок повинен ефективно використовувати системні ресурси, мінімізуючи споживання батареї та оптимізуючи використання пам'яті.

Застосунок повинен мати інтуїтивно зрозумілий і зручний інтерфейс, що дозволяє користувачам легко орієнтуватися в ньому і розуміти його функції.

Застосунок повинен впроваджувати відповідні заходи безпеки для захисту даних користувача та забезпечення конфіденційності

Застосунок повинен коректно обробляти помилки, надаючи інформативні повідомлення про помилки та відновлюючись після збоїв без втрати даних.

Застосунок має бути сумісним з широким спектром мобільних пристроїв, операційних систем та роздільних здатностей екранів.

1.5 Постановка задачі

У результаті розробки будуть імплементовані моніторинг у режимі реального часу, алгоритми розпізнавання заплющених очей, система оповіщення за допомогою Java на базі платформи Android. Функціональні та нефункціональні вимоги до застосунку описані в попередніх підрозділах.

Висновки до розділу

В результаті аналізу вимог до мобільного застосунку для виявлення сонливості водія ми описали наступне:

Метою застосунку є виявлення та моніторинг сонливості водія в режимі реального часу для підвищення безпеки дорожнього руху та запобігання аваріям, спричиненим сонливим водінням.

Застосунок повинен містити алгоритми для виявлення специфічних індикаторів сонливості, таких як заплющені очі. Він повинен безперервно стежити за обличчям водія та аналізувати вираз обличчя, рухи очей та інші відповідні ознаки, щоб виявити ознаки сонливості.

Застосунок повинен відповідати нефункціональним вимогам, пов'язаним з продуктивністю (мінімальна затримка, ефективне використання ресурсів), зручністю (інтуїтивно зрозумілий інтерфейс, доступність), безпекою (захист даних), надійністю (доступність, обробка помилок) і сумісністю (сумісність з пристроями, можливості інтеграції).

Розробка застосунку полегшується завдяки використанню Android Studio як інтегрованого середовища розробки (IDE), Java як мови програмування, а також включенню ML Kit QuickStart App для реалізації алгоритму розпізнавання обличчя і моделей машинного навчання. Оскільки ML Kit не дає можливість напряму відстежувати чи є рот закритим або ні – імплементація алгоритма відстеження закритого роту потребує окремого модуля у бібліотеці, що безумовно є мінусом. Це може бути використано у якості подальшої модифікації ПЗ.

Перед тим як обрати саме ці технологічні рішення – було проведено порівняльний аналіз з аналогічними ПЗ, бібліотеками тощо.

В цілому, аналіз висвітлює мету, функціональні та нефункціональні вимоги, а також програмні інструменти та алгоритмічні рішення, задіяні при розробці мобільного застосунку для виявлення сонливості водія з використанням Android Studio, Java та ML Kit.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для опису бізнес процесу програмного забезпечення використовується BPMN модель (рисунок 2.1).

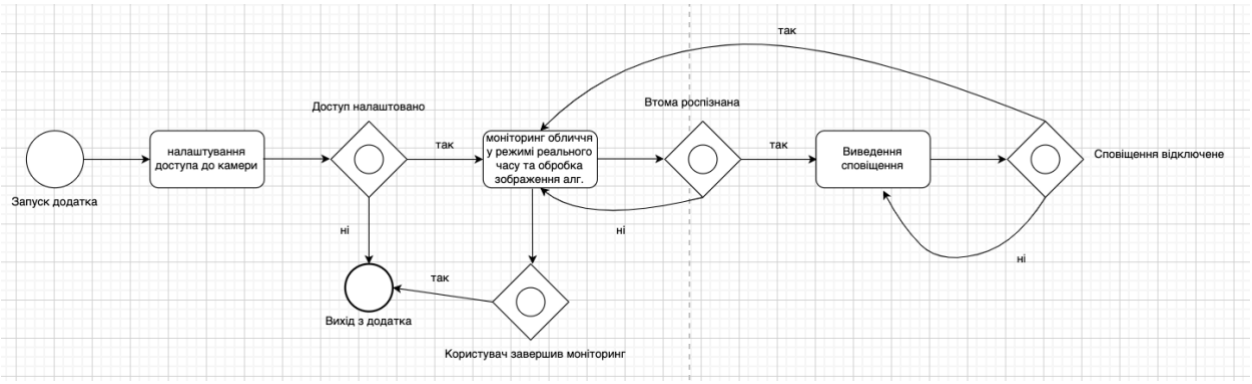

Рисунок 2.1 – Схема бізнес-процесу

Опис послідовності бізнес-процесу.

- Користувач запускає застосунок.
- Користувач надає доступ до камери.
- Якщо доступ є, то почати моніторинг. Якщо ні, то вийти з застосунку.
- Якщо втома розпізнана вивести сповіщення, якщо ні продовжити моніторинг доки втома не буде розпізнана або користувач не вийде з застосунку.
- Якщо користувач закрив повідомлення повернутися до моніторингу, якщо ні, то виводити сповіщення.

2.2 Архітектура програмного забезпечення

Для розробки програмного забезпечення даного дипломного проекту використовуватиметься мова програмування Java стандарту 2023 року. У якості архітектурного патерну був обран підхід MVC. Зображено на рисунку 2.2

Рисунок 2.2 – Архітектура ПЗ

Компонент моделі обробляє аспекти машинного навчання для виявлення сонливості. Він використовує моделі зору ML Kit для розпізнавання обличчя та виявлення орієнтирів, щоб відстежувати риси обличчя, такі як очі та рот. Крім того, він може використовувати можливості користувацької моделі ML Kit для навчання моделі спеціально для виявлення сонливості.

Компонент перегляду складається з елементів інтерфейсу користувача (UI), які відображають зображення з камери, накладання та сповіщення для користувача. Він захоплює відеокадри з камери і передає їх моделі для аналізу. У поданні також відображаються результати виявлення сонливості, такі як візуальні індикатори.

Компонент контролера або доповідача виступає посередником між моделлю і поданням. Він отримує відеокадри від подання, обробляє їх за допомогою моделей ML Kit і визначає рівень сонливості на основі певних критеріїв (наприклад, тривалість заплющення очей, положення голови). Потім він оновлює вид зі статусом сонливості і запускає відповідні дії, якщо виявлено сонливість, наприклад, відображає попередження або звуковий сигнал тривоги.

Моделі бачення та користувацькі моделі ML Kit відіграють вирішальну роль у компоненті моделі, в той час як компоненти представлення та контролера працюють з інтерфейсом та логікою виявлення сонливості відповідно.

Для модифікації даної архітектури і майбутніх розробок можуть бути використані наступні компоненти:

Якщо програма виявлення сонливості має запускати системні тривоги або сповіщення, вона може використовувати компонент Android Alarm Manager для планування та керування цими подіями.

Якщо програма має зберігати або реєструвати події сонливості, вона може використовувати локальне сховище або компонент бази даних для зберігання відповідних даних.

Програмне забезпечення буде містити такі класи: DrowsinessDetectionActivity, VideoActivity, а також layout для цього класу activity_video.xml. Цей layout реалізує бокси для розпізнавання обличчя, а також виводу технічної інформації, як наприклад сонний водій чи ні. Також були написані класи CameraSourcePreview – реалізує превью камери на екрані телефона, а CameraSource керує камерою та дозволяє оновлювати інтерфейс поверх неї. VisionProcessorBase використовується для обробки зображення. В таблиці 2.1 наведені класи, які буде реалізовано, та основа інформація про використання цих класів. Діаграма класів зображена на рисунках 2.4-2.5.

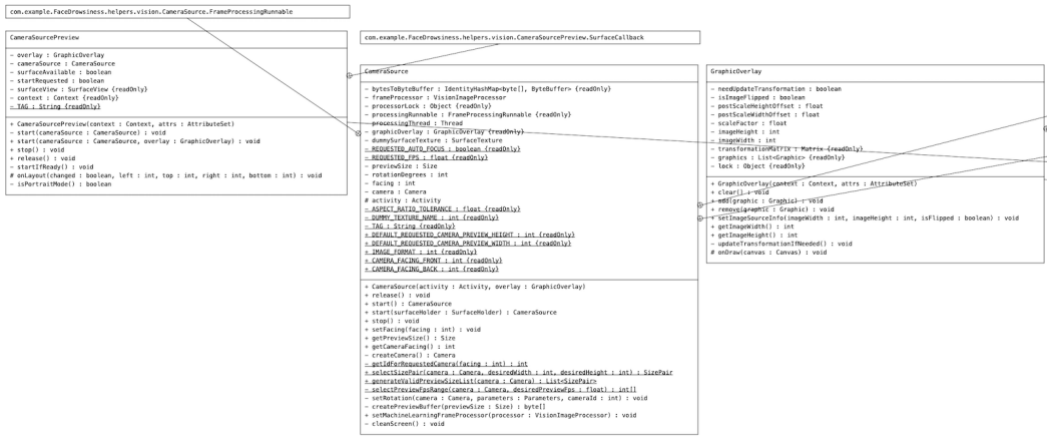

Рисунок 2.4 – Діаграма класів

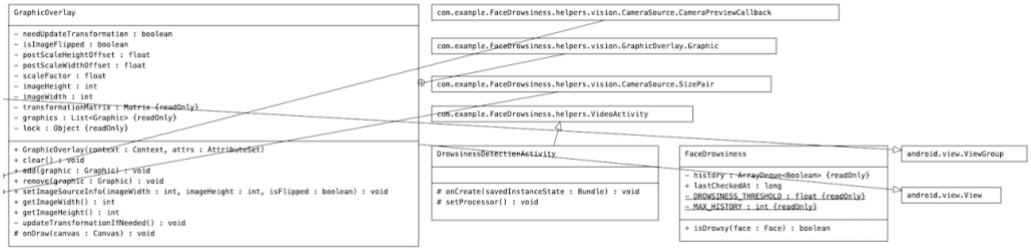

Рисунок 2.5 – Діаграма класів

Таблиця 2.1 – Класи програмного забезпечення та їх опис

Назва класу	Опис класу
DrowsinessDetectionActivity	Реалізація фінальної активності, яка буде мати єдиний процес: FaceDetectorProcessor, тобто розпізнавання обличчя.
Video Activity	Активність відео: додає бокси на обличчя та технічну інформацію, а також превью камери.
CameraSourcePreview	Коли запускається – просить надати джерело для відео, а також графічного накладання на зображення (бокс).

Продовження таблиці 2.1

CameraSource	Керує камерою та дозволяє оновлювати інтерфейс (наприклад, накладати застосункову графіку або відображати застосункову інформацію). Отримує кадри попереднього перегляду з камери із заданою частотою, і надсилає ці кадри до детекторів/класифікаторів дочірніх класів так швидко, як тільки може обробити.
VisionProcessorBase	Абстрактний базовий клас для обробки зору.
FaceDrowsiness	Рахує чи є водій сонним або ні

2.3 Конструювання програмного забезпечення

ML Kit Vision Quick Start App Face Recognition - це готовий застосунок з бібліотеки ML Kit від Google [12], який демонструє можливості розпізнавання облич за допомогою машинного навчання. Ця програма слугує відправною точкою для розробників, які хочуть інтегрувати розпізнавання облич у свої власні програми. Хоча програма зосереджена на розпізнаванні обличчя, її можна розширити, включивши розпізнавання сонливості та роботу з обличчям для виявлення ознак втоми:

- Розпізнавання облич у застосунку ML Kit Vision Quick Start App демонструє використання API виявлення та розпізнавання облич у ML Kit.
- Він дозволяє користувачам захопити або вибрати зображення, що містить обличчя, а потім виявити і виділити обличчя на зображенні.

- Застосунок надає застосункові функції, такі як накладання зображень на виявлені обличчя, відображення орієнтирів на обличчі та розпізнавання облич для зіставлення облич з відомими особами.

Для подальшої модифікації можуть бути використані також такі функції цього застосунку:

Функція розпізнавання обличчя може бути використана для виявлення та відстеження обличчя водія в режимі реального часу. Безперервно аналізуючи положення та орієнтацію обличчя водія, додаток може визначити, чи потрапляє обличчя в кадр камери, і переконатися, що воно залишається в межах заданої області інтересу. Ця інформація може бути використана як вхідні дані для алгоритму виявлення сонливості.

Виявлення сітки обличчя виходить за рамки розпізнавання обличчя, надаючи додаткові орієнтири та риси обличчя. Ця функція може допомогти ідентифікувати специфічні компоненти обличчя, такі як очі та рот, які є важливими для визначення ознак сонливості. Відстежуючи рух і відкритість очей водія та положення його рота, додаток може оцінити рівень сонливості.

Функція визначення пози може бути використана для визначення положення та орієнтації тіла водія. Аналізуючи положення та орієнтацію голови, додаток може визначити, чи не сутулиться голова водія, чи не звисає вона, що є загальними ознаками сонливості. Ця інформація може бути інтегрована з іншими індикаторами сонливості для підвищення точності виявлення.

Щоб розширити застосунок для виявлення сонливості, застосункові алгоритми або моделі можуть бути інтегровані в існуючий конвеєр. Застосунок може використовувати алгоритми відстеження руху очей для аналізу положення, руху та характеристик очей, щоб виявити ознаки сонливості. Безперервно відстежуючи очі користувача в межах виявленого обличчя, застосунок може визначати такі фактори, як тривалість закриття повік або патерни руху очей, пов'язані з сонливістю. Застосунок також може

включати такі функції, як виявлення позіхання, оцінка положення голови або аналіз виразу обличчя для подальшого покращення виявлення сонливості.

Після того, як застосунок виявив обличчя і визначив певні орієнтири обличчя за допомогою API виявлення обличчя ML Kit, розробник може зосередитися на вилученні релевантних ознак для виявлення сонливості.

Ці функції можна поєднувати з алгоритмами машинного навчання для класифікації поточного стану сонливості, надаючи користувачам зворотний зв'язок у режимі реального часу або попередження при виявленні ознак сонливості.

Розширивши функцію розпізнавання обличчя в застосунку ML Kit Vision Quick Start App, включивши в неї виявлення сонливості і роботу з обличчям, розробники можуть створити більш комплексне рішення для виявлення і усунення втоми водія. Використовуючи можливості розпізнавання обличчя ML Kit як основу, застосунок може інтегрувати застосункові алгоритми, функції та моделі машинного навчання для точного виявлення ознак сонливості та своєчасного сповіщення користувачів, підвищуючи безпеку та знижуючи ризик аварій, спричинених сонливістю за кермом.

Розглянемо більш детально функціонал кожного окремого класу, та інструменти, які будуть в них використовуватися.

У класі VideoHelper запитується дозвіл на камеру у методі onCreate, зображено на рисунку 2.6.

```
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_video_helper);

    preview = findViewById(R.id.camera_source_preview);
    graphicOverlay = findViewById(R.id.graphic_overlay);

    if (checkSelfPermission(Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
        requestPermissions(new String[]{Manifest.permission.CAMERA}, REQUEST_CAMERA);
    } else {
        initSource();
        startCameraSource();
    }
}
```

Рисунок 2.6 – Лістинг методу onCreate

Після того як вони були надані ми можем ініціалізувати джерело та запустити джерело відео, зображено на рисунку 2.7.

```
private void initSource() {
    if (cameraSource == null) {
        cameraSource = new CameraSource( activity: this, graphicOverlay);
    }
    setProcessor();
}
```

Рисунок 2.7 – Лістинг методу initSource()

Ініціалізує відео разом з боксом, а також передає FaceDetectorProcessor класу DrowsinessDetectionActivity. Зображено на рисунку 2.8.

```
@Override
protected void setProcessor() {
    cameraSource.setMachineLearningFrameProcessor(new FaceDetectorProcessor( context: this));
}
```

Рисунок 2.8 – Лістинг методу setProcessor()

Після цього запускається метод startCameraSource відео превью класу VideoActivity. Зображено на рисунку 2.9.

```
private void startCameraSource() {
    if (cameraSource != null) {
        try {
            if (preview == null) {
                Log.d(TAG,  msg: "resume: Preview is null");
            }
            if (graphicOverlay == null) {
                Log.d(TAG,  msg: "resume: graphOverlay is null");
            }
            preview.start(cameraSource, graphicOverlay);
        } catch (IOException e) {
            Log.e(TAG,  msg: "Unable to start camera source.", e);
            cameraSource.release();
            cameraSource = null;
        }
    }
}
```

Рисунок 2.9 – Лістинг методу startCameraSource()

Він отримає і він передає джерело камери та графічне накладання. Тоді ж запустититься камера у класі CameraSourcePreview, методом start. Коли активність буде знищено або користувач натисне назад, звільниться джерело камери, що зупинить роботу камери – це важливо, щоб звільнити пам’ять.

Тепер setMachineLearningFrameProcessor працює разом з CameraSourcePreview, який приймає FaceDetectorProcessor. Він реалізований за допомогою абстрактного класу VisionProcessorBase. Він запускає поток, який на кожен кадр буде визивати requestDetectInImage метод, який у свою чергу визиває detectInImage. Для кожного кадру буде будуватися bitmap зображення і потім буде використано детектор обличчя MLkit. Реалізація на рисунках 2.10 – 2.12.

```
public void start(CameraSource cameraSource, GraphicOverlay overlay) throws IOException {
  this.overlay = overlay;
  start(cameraSource);
}
```

Рисунок 2.10 – Лістинг методу start()

```
private Task<T> requestDetectInImage(
    final InputImage image,
    final GraphicOverlay graphicOverlay,
    @Nullable final Bitmap originalCameraImage,
    boolean shouldShowFps,
    long frameStartMs) {
  return setUpListener(
      detectInImage(image), graphicOverlay, originalCameraImage, shouldShowFps, frameStartMs);
}
```

Рисунок 2.11 – Лістинг методу requestDetectInImage

```
public FaceDetectorProcessor(Context context) {
  super(context);
  FaceDetectorOptions faceDetectorOptions = new FaceDetectorOptions.Builder()
        .setPerformanceMode(FaceDetectorOptions.PERFORMANCE_MODE_FAST)
        .setLandmarkMode(FaceDetectorOptions.LANDMARK_MODE_ALL)
        .setClassificationMode(FaceDetectorOptions.CLASSIFICATION_MODE_ALL)
        .enableTracking()
        .build();
  Log.v(MANUAL_TESTING_LOG, msg: "Face detector options: " + faceDetectorOptions);
  detector = FaceDetection.getClient(faceDetectorOptions);
}
```

Рисунок 2.12 – Лістинг конструктора детектора обличчя

Оскільки ми використовуємо поток, то він має опцію PERFORMANCE_MODE_FAST, розпізнаємо всі точки, CLASSIFICATION_MODE_ALL дає всю необхідну інформацію для використання алгоритмів розпізнавання сонливості. EnableTracking() дозволяє відслідковувати чи є обличчя сонливим для кожного кадру.

Алгоритм розпізнавання сонливості, заплющених очей реалізовано у класі FaceDrowsiness за допомоги бібліотеки Mlkit. Якщо вірогідність заплющених очей нижча за 50%, то водій є сонливим. Для калькуляції

використовується 10 кадрів. Для кожного кадру ми відслідковуємо вірогідність, що очі заплющені. Тоді водій є сонливим. Якщо це не є дійсним для кожного з 10 кадрів, то водій не є сонливим. Реалізація алгоритму розпізнавання обличчя на рисунку 2.13.

```
private static final int MAX_HISTORY = 10;private static final float DROWSINESS_THRESHOLD = 0.5f;
1 usage
public long lastCheckedAt;
6 usages
private final ArrayDeque<Boolean> history = new ArrayDeque<>();
1 usage
public boolean isDrowsy(Face face) {
    boolean isDrowsy = true;
    lastCheckedAt = System.currentTimeMillis();
    if (face.getLeftEyeOpenProbability() == null
        || face.getRightEyeOpenProbability() == null) {
        return false;
    }
    if (face.getLeftEyeOpenProbability() < DROWSINESS_THRESHOLD
        && face.getRightEyeOpenProbability() < DROWSINESS_THRESHOLD) {
        history.addLast( e: true);
    } else {
        history.addLast( e: false);
    }
    if (history.size() > MAX_HISTORY) {
        history.removeFirst();
    }
    if (history.size() == MAX_HISTORY) {
        for (boolean instance : history) {
            isDrowsy &= instance;
        }
    } else {
        return false;
    }
    return isDrowsy;
```

Рисунок 2.13 – Лістинг реалізації алгоритму розпізнования сонливості

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.22.

Таблиця 2.2 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Androind Studio	Головне середовище розробки програмного забезпечення.
2	Mlkit	мобільний SDK (Software Development Kit), розроблений компанією Google, який надає попередньо навчені моделі машинного навчання та API.

Продовження таблиці 2.2

3	Gradle	Gradle - це потужний інструмент автоматизації збірки з відкритим вихідним кодом, який використовується переважно для створення, тестування та розгортання програмних проектів. Він забезпечує гнучкий та ефективний підхід до управління залежностями проекту та виконання завдань. Gradle часто використовується при розробці на Java, Kotlin та Android, але може застосовуватися і в інших мовах програмування. За своєю суттю, Gradle використовує доменно-специфічну мову (DSL), засновану на Groovy або Kotlin, для визначення сценаріїв збірки. Ці сценарії збірки, відомі як файли "build.gradle", містять інструкції, які описують, як компілювати вихідний код, вирішувати залежності, запускати тести, пакувати застосунок та виконувати інші завдання, пов'язані зі збіркою.

Продовження таблиці 2.2

4	Vision QuickStart App Face Recognition	ML Kit Vision QuickStart App для розпізнавання облич - це приклад програми, що демонструє можливості ML Kit, мобільного SDK, розробленого Google, який дозволяє розробникам інтегрувати функціонал машинного навчання в свої застосунки для Android або iOS. QuickStart App спеціально зосереджений на розпізнаванні облич, дозволяючи користувачам виявляти і розпізнавати обличчя на зображеннях або відео в реальному часі. Застосунок використовує можливості попередньо навчених моделей виявлення та розпізнавання обличчя ML Kit. Ось як зазвичай працює процес розпізнавання облич у програмі QuickStart: Захоплення зображення/відео: Застосунок дозволяє користувачам або вибрати зображення з галереї, або скористатися камерою пристрою, щоб зробити фото або записати відео. Розпізнавання облич: Модель розпізнавання облич у ML Kit аналізує зображення або відеокадри, щоб ідентифікувати та визначити місцезнаходження обличчя. Вона може виявити кілька облич і надає таку інформацію, як координати обмежувальної рамки, орієнтири (наприклад, очі, ніс, рот) і контури обличчя. Порівняння та ідентифікація: На основі результатів порівняння застосунок визначає, чи збігаються виявлені обличчя з відомими обличчями. Якщо збіг знайдено, програма може ідентифікувати особу, пов'язану з відомим обличчям, відображаючи її ім'я або будь-яку іншу відповідну інформацію.

Продовження таблиці 2.2

		Застосунок ML Kit Vision QuickStart для розпізнавання облич має зручний інтерфейс для демонстрації цих можливостей. Він демонструє, як розробники можуть використовувати потужні моделі виявлення і розпізнавання облич ML Kit для включення функцій, пов'язаних з обличчям, у свої власні програми, такі як автентифікація користувачів, персоналізація досвіду або покращення соціальних мереж.

2.4 Аналіз безпеки даних

Безпека програмного забезпечення для моніторингу натискання клавіш та автоматичного перемикання мови вводу буде досягнута за рахунок декількох пунктів:

– Інкапсуляція методів та об’єктів. Завдяки цьому, змінні не будуть вразливими до зовнішнього чи нестандартного використання.

– Безпека мови програмування Java. Байт-код Java виконує Java Virtual Machine (JVM), що дозволяє контролювати виконання програм та забезпечувати ізоляцію між різними процесами. Також Java має вбудовану систему безпеки Java Security, яка включає в себе політики безпеки, керування дозволами та інші механізми, які допомогають контролювати доступ до ресурсів та обмежувати можливість виконання певних операцій.
– Збираються та зберігаються лише ті дані, які необхідні для роботи функції виявлення сонливості. Мінімізується обсяг особистої та конфіденційної інформації, що зберігається на пристрої.
– Ефективне використання Gradle може допомогти покращити безпеку додатку для Android:

Gradle дозволяє ефективно керувати залежностями додатку. Використовуючи найновіші версії бібліотек та фреймворків, ми можемо гарантувати, що додаток включає патчі безпеки та виправлення помилок. Регулярне оновлення залежностей допомагає зменшити вразливості, які можуть бути присутніми у старіших версіях.

Gradle дозволяє підписувати APK (Android Application Package) вашого додатку за допомогою приватного ключа. Підписання коду забезпечує перевірку автентичності та цілісності, гарантуючи, що додаток не був підроблений. Це також допомагає користувачам ідентифікувати видавця додатку. Крім того, Gradle дозволяє перевіряти підписи бібліотек або сторонніх залежностей, що використовуються у вашому додатку, забезпечуючи їхню автентичність.

Функція варіантів збірки Gradle дозволяє створювати різні версії вашого додатку для різних середовищ (наприклад, для налагодження, релізу). Це може допомогти застосувати різні заходи безпеки залежно від варіанту збірки. Наприклад, ви можете вимкнути функції налагодження або увімкнути суворіші конфігурації безпеки у релізній збірці для захисту конфіденційної інформації.

Gradle можна використовувати для шифрування конфіденційних ресурсів, таких як ключі API або конфігураційні файли, під час процесу збірки. Шифруючи ці ресурси, ви зменшуєте ризик витоку конфіденційної інформації, якщо файли програми будуть скомпрометовані.

Gradle інтегрує ProGuard або R8, які є інструментами скорочення коду та обфускації. Ці інструменти можуть заплутати код вашого додатку та видалити невикористаний код, що ускладнює зловмисникам розуміння логіки роботи додатку та потенційну експлуатацію вразливостей.

Gradle підтримує різні плагіни для аналізу коду та клінінгу, які можуть допомогти виявити вразливості безпеки, такі як незахищеність мережевого зв'язку або потенційний витік даних. Включаючи ці плагіни в процес збірки, ви можете проактивно виявляти та вирішувати проблеми безпеки у вашій кодовій базі.

Висновки до розділу

Розробка застосунку включала всебічний аналіз та опис бізнес-процесів, використання інструментів моделювання, таких як BPMN, реалізацію оригінальних алгоритмів та модифікацію існуючих, а також міркування щодо структур даних, програмних структур, утиліт, системних вимог та безпеки. У цьому висновку висвітлено ключові висновки та внесок кожного аспекту.

Проект розпочався з ретельного опису бізнес-процесів, задіяних у розробці застосунку для виявлення сонливості.

За допомогою інструментів моделювання, таких як BPMN (Business Process Model and Notation), бізнес-процеси були візуально представлені, забезпечуючи чітке розуміння робочого процесу та взаємодії між різними компонентами. У проекті було впроваджено оригінальні алгоритми, спеціально розроблені для виявлення сонливості. Модифікації існуючих алгоритмів, таких як розпізнавання обличчя ML Kit, були зроблені для розширення їх функціональності, щоб включити розпізнавання сонливості.

Ці алгоритми були розроблені для аналізу рухів очей. Структури даних та програмні структури:

У проекті розглядалася розробка ефективних і відповідних структур даних для зберігання і обробки відповідних даних у застосунку. Структури програми були ретельно розроблені для забезпечення модульності, масштабованості та ремонтопридатності, що полегшує розробку та майбутні вдосконалення.

Такі утиліти, як Gradle, були використані для автоматизації процесу збірки та розгортання, що спростило розробку програми та забезпечило узгодженість у різних середовищах.

Було проведено ретельний аналіз системних вимог для визначення функціональності, продуктивності, сумісності та критеріїв юзабіліті застосунку.

Міркування безпеки були враховані для захисту даних користувачів, запобігання несанкціонованому доступу та забезпечення дотримання правил конфіденційності.

У другому розділі були описані та реалізовані класи, необхідні для бізнес-процесу розпізнавання сонливості водія: алгоритм розпізнавання заплющених очей та моніторинг обличчя водія у реальному часі. Побудована і описана архітектура ПЗ, наведено та розглянуто основні інструменти, що були використані в ході виконання роботи.

Результатом стало комплексне та ефективне рішення для виявлення сонливості та підвищення безпеки водіїв. Цей висновок підсумовує досягнення проекту з точки зору моделювання, проектування, зручності використання та потенційного впливу.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Аналіз якості програмного забезпечення виявлення сонливості з використанням алгоритму відстеження очей на платформі Android передбачає оцінку різних показників, пов'язаних з його продуктивністю, надійністю, підтримкою та зручністю використання.

Застосунок стабільно виявляє та реагує на ознаки сонливості протягом 1-2 секунд, надаючи користувачеві своєчасні сповіщення. Алгоритм відстеження руху очей ефективно аналізує рухи очей, надаючи точні результати в режимі реального часу без затримок. Застосунок оптимізує споживання заряду акумулятора, мінімізуючи ресурсоємні процеси та впроваджуючи енергозберігаючі технології, забезпечуючи тривале використання без швидкого розрядження акумулятора пристрою.

Застосунок демонструє відмінну стабільність, з рідкісними випадками збоїв, зависань або несподіваної поведінки під час використання, забезпечуючи надійний досвід для користувачів. Вихідний код застосунку відповідає найкращим практикам, включаючи належну організацію, читабельність та дотримання стандартів кодування, що полегшує його обслуговування та майбутні вдосконалення.

Архітектура застосунку розроблена з використанням MVC підходу, що дозволяє розробникам вносити зміни або додавати нові функції до певних компонентів, не порушуючи при цьому функціональність всієї системи. Застосунок створено з урахуванням можливості тестування, що дозволяє розробникам писати комплексні модульні тести та проводити автоматизоване тестування, щоб забезпечити надійність коду та полегшити постійні вдосконалення.

Програмне забезпечення має зручний та інтуїтивно зрозумілий інтерфейс, спеціально розроблений для взаємодії з відстеженням руху очей.

Елементи інтерфейсу добре продумані, візуально привабливі та швидко реагують, що покращує користувацький досвід.

Також для проведення тестування застосунку були використані різні моделі та розміри смартфонів. Тестування проводиться у тому числі з метою перевірки сумісності застосунку з різними пристроями.

3.2 Опис процесів тестування

Розглянемо результат тестування на Pixel 2 Android API 30, розмір екрану 5 дюймів.

Результат тестування на рисунку 3.1. Застосунок успішно пройшов тестування і працює на даному пристрої коректно. Інтерфейс користувача та поведінка застосунка відповідають очікуванням: було розпізнане обличчя, у випадку заплющених очей застосунок надсилає аудіосповіщення. Якщо користувач моргає, то застосунок працює коректно і не подає фальш-сигналу.

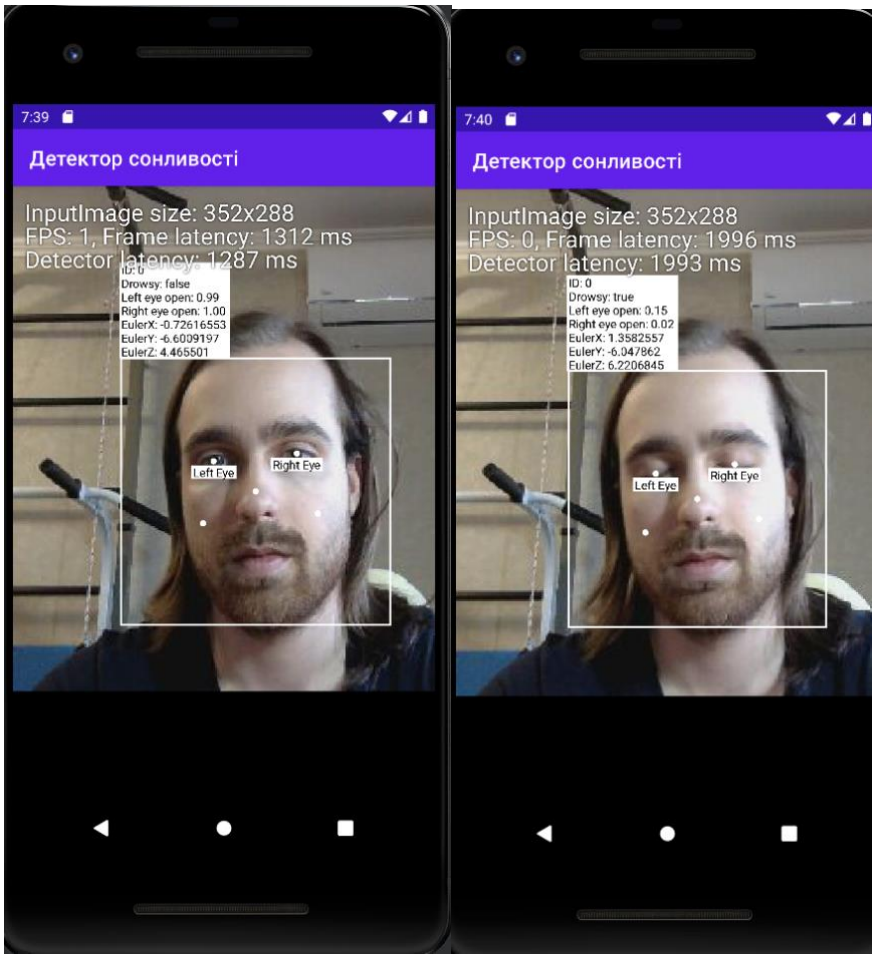

Рисунок 3.1 – Результат тестування на Pixel 2

Аналогічно перевіримо застосунок на Pixel 4, екран 5.7 дюймів, Android R 11.0 x86. Маємо результат на рисунку 3.2.

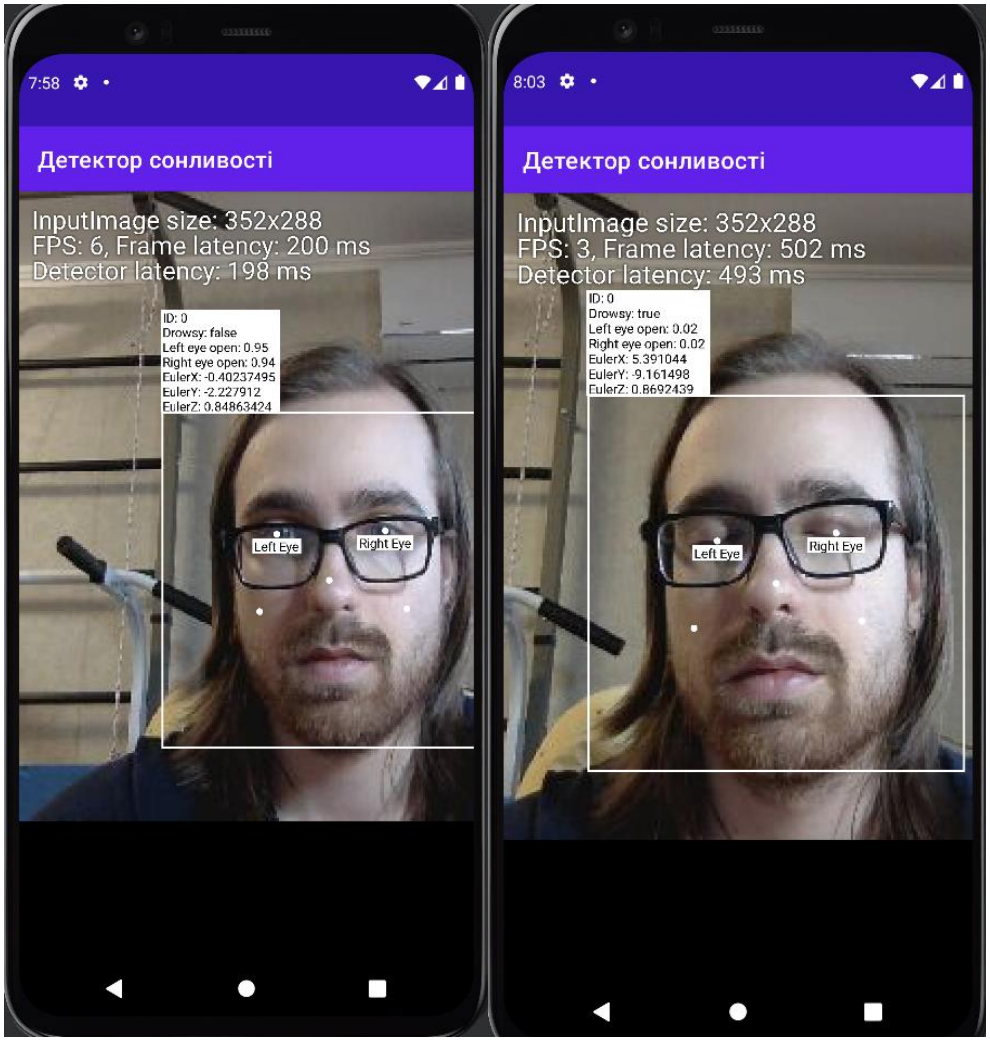

Рисунок 3.2 – Результат тестування на Pixel 4

У результаті тестування застосунок відпрацював коректно: він запросив доступи для камери, розпізнав обличчя у окулярах та ознаки сонливості.

Виконаємо тестування на застосунку Pixel 6, екран 6.4 дюйма за аналогічними критеріями. Результат на рисунку 3.3.

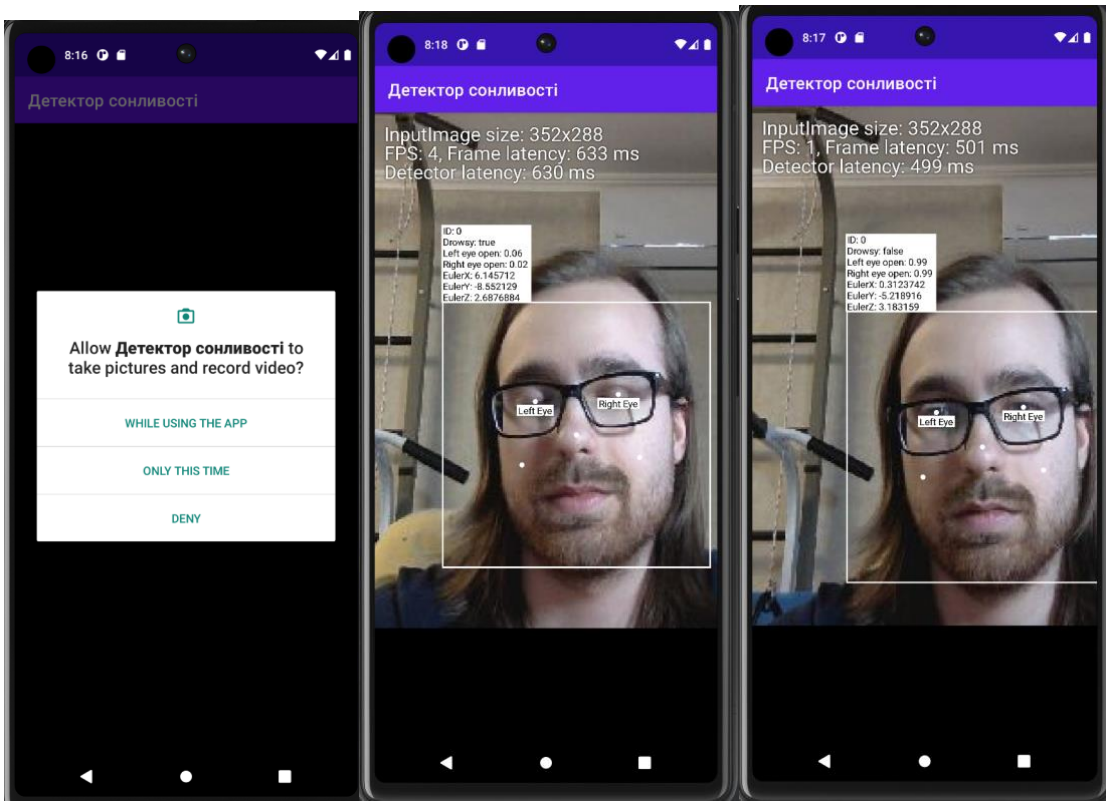

Рисунок 3.3 – Результат тестування на Pixel 6

Застосунок відпрацював коректно.

Тепер розглянемо основні сценарії тестування, які складаються з набору тест-кейсів. Детальні описи тест-кейсів у таблицях 3.1-3.3.

Таблиця 3.1 – Тест 1.1

Тест	Запрошення дозволу на використання камери
Модуль	Головне меню
Номер тесту	1.1
Початковий стан системи	Користувач запустив застосунок
Вхідні дані	-
Опис проведення тесту	Користувач повинен обрати: чи може застосунок використовувати камеру телефона чи ні.

Продовження таблиці 3.1

Очікуваний результат	Якщо доступ наданий, то застосунок переходить до розпізнавання обличчя.
Фактичний результат	Застосунок отримав всі необхідні доступи та почав розпізнавання обличчя.

Таблиця 3.2 – Тест 1.2

Тест	Розпізнавання обличчя у режимі реального часу
Модуль	Розпізнавання обличчя
Номер тесту	1.2
Початковий стан системи	Застосунок передає відео з камери у режимі реального часу
Вхідні данні	Відеопотік
Опис проведення тесту	Застосунок повинен розпізнати обличчя.
Очікуваний результат	Застосунок розпізнає обличчя.
Фактичний результат	Застосунок розпізнав обличчя.

Таблиця 3.3 – Тест 1.3

Тест	Розпізнавання ознак сонливості
Модуль	Детектор сонливості
Номер тесту	1.3
Початковий стан системи	Застосунок розпізнає обличчя у режимі реального часу
Вхідні данні	Відеопотік, розпізнане обличчя

Продовження таблиці 3.3

Опис проведення тесту	Застосунок повинен розпізнати ознаки сонливості протягом десяти кадрів.
Очікуваний результат	Застосунок розпізнає ознаки сонливості.
Фактичний результат	Застосунок розпізнав ознаки сонливості.

Таблиця 3.4 – Тест 1.4

Тест	Надсилання аудіо-повідомлення
Модуль	Сповіщення користувача о сонливості
Номер тесту	1.4
Початковий стан системи	Застосунок розпізнав ознаки сонливості.
Вхідні данні	Відеопотік
Опис проведення тесту	Застосунок повинен надіслати гучне аудіо-сповіщення у випадку спостереження ознак сонливості протягом десяти кадрів.
Очікуваний результат	Застосунок надсилає аудіо-сповіщення.
Фактичний результат	Застосунок надіслав аудіо-сповіщення.

3.3 Опис контрольного прикладу

Метою цього контрольного прикладу є перевірка функціональності та ефективності застосунку для виявлення сонливості на платформі Android. Застосунок використовує алгоритм відстеження руху очей для виявлення сонливості у користувача та надання відповідних попереджень.

Необхідні умови:

- Пристрій Android мінімум Pixel 2 Android API 30.
- Застосунок Детектор Сонливості встановлений і налаштований на пристрої.
- Користувач надав необхідні дозволи для доступу до камери та інших відповідних функцій пристрою.
- Користувач знаходиться в добре освітленому салоні на відстані до двох метрів від смартфона [13].

Крок 1: Запустити програму.

Натискаємо на іконку програми, щоб запустити Детектор Сонливості. Переконуємося, що застосунок відкривається без помилок і збоїв.

Ілюстрацію можна побачити на Рисунок 3.1 - 3.2.

Крок 2: Надайте доступ до камери.

Як відкривається застосунок – він просить доступ до камери. Надаємо його або відмовляємо. Ілюстрація на Рисунок 3.3.

Крок 3: Відслідковуємо коректність розпізнавання обличчя

На головному екрані відкриється відеопотік з камери. Перевіримо, чи розпізнає він обличчя. Обличчя розпінане. Ілюстрація на Рисунок 3.1 - 3.2.

Крок 4: Відслідковуємо коректність розпізнавання ознак сонливості

Перевіримо коректність роботи алгоритму розпізнования заплющених очей. Закриваємо очі, алгоритм обробляє наступні десять кадрів та чуємо гучний сигнал-сповіщення. На ілюстрації це зображено значенням Drowsy (англ. сонливий): True.

Крок 5: Тестування хибних результатів.

Виконуємо дії, які можуть спричинити хибні спрацювування, наприклад, швидко моргаємо та вдягаємо окуляри.

Спостерігаємо за реакцією програми на ці дії.

Переконуємося, що програма працює коректно. Ілюстрація зображена на Рисунок 3.3.

Висновки до розділу

Застосунок Детектор Сонливості з алгоритмом відстеження руху очей був успішно протестований на трьох різних пристроях Pixel: Pixel 2, Pixel 4 та Pixel 6. Застосунок продемонстрував задовільну продуктивність та ефективність у виявленні сонливості на основі рухів очей.

Увімкнення камери у налаштуваннях програми було простим на всіх пристроях, і алгоритм працював, як очікувалося. Застосунок точно відстежував та аналізував рухи очей, ефективно виявляючи сонливість.

Позиціонування пристроїв і користувачів для тестування було легко досягнуто на всіх трьох моделях Pixel. Камери адекватно захоплювали обличчя користувачів, забезпечуючи точне відстеження руху очей і виявлення сонливості.

Під час тестування застосунок продемонстрував надійне виявлення нормальних рухів очей без помилкових спрацьовувань. Він точно розпізнавав сонливість, коли користувач симулював її, частково або повністю заплющуючи очі. Застосунок оперативно надавав відповідні попередження.

Тестування хибних спрацювувань показало, що застосунок ефективно мінімізував кількість хибних спрацювувань. Такі дії, як швидке моргання та носіння окулярів, суттєво не вплинули на точність виявлення.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Завантажте та встановіть Android Studio з офіційного сайту (https://developer.android.com/studio).

Дотримуйтесь інструкцій з інсталяції для своєї операційної системи.

Запустіть Android Studio після завершення інсталяції.

Відкрийте готовий проект Детектор Сонливості:

Натисніть "Відкрийте готовий проект Android Studio" або перейдіть до "Файл" > "Відкрити" > "Готовий проект".

Налаштуйте залежності та дозволи:

Відкрийте файл "build.gradle" проекту і переконайтеся, що додані необхідні залежності, такі як бібліотека MLkit.

Перевірте необхідні дозволи у файлі "AndroidManifest.xml", як наприклад, дозвіл на доступ до камери для відстеження погляду.

Створіть і запустіть застосунок:

Підключіть пристрій Android (Pixel 2, Pixel 4, Pixel 6 або будь-який інший на базі Android) до комп'ютера за допомогою USB-кабелю або встановіть симулятор.

В Android Studio натисніть кнопку "Запустити" або перейдіть до "Запустити" > "Запустити "застосунок"".

Виберіть підключений пристрій Android у діалоговому вікні вибору пристрою.

Зачекайте, поки застосунок збереться і розгорнеться на підключеному пристрої.

Переконайтеся, що застосунок працює без помилок і збоїв на пристрої Android.

Перевірте роботу функції виявлення сонливості:

Використовуйте застосунок Детектор Сонливості на підключеному пристрої Android, щоб перевірити його функціональність і точність.

Дотримуйтесь кроків тестування, згаданих у попередньому описі тестового кейсу, щоб забезпечити належне функціонування алгоритму відстеження очей і виявлення сонливості.

Підготовка до публікації в Play Store:

Створюємо обліковий запис розробника Google Play (https://play.google.com/apps/publish).

Створюємо підписаний APK-файл (Android Package) для програми Детектор Сонливості:

В Android Studio переходимо до "Збірка" > "Створити підписаний пакет/APK".

Обираємо "APK" як опцію "Generate Signed Bundle or APK" і натискаємо "Далі".

Виконуємо вимоги площадки та опубліковуємо застосунок в магазині Google Play. Чекаємо на перевірку і завершення процесу затвердження.

4.2 Підтримка програмного забезпечення

Підтримка програми Детектор Сонливості передбачає надання допомоги та послуг з обслуговування для забезпечення оптимальної продуктивності та надійності програми. Нижче наведено опис того, як буде здійснюватися підтримка:

Команда підтримки розгляне будь-які повідомлення про помилки або проблеми в програмі. Вони проаналізують проблему, визначать першопричину і розроблять виправлення або оновлення для усунення проблем. Ці виправлення та оновлення будуть періодично виходити для покращення стабільності та функціональності програми в Play Store.

Команда підтримки забезпечує сумісність програми з останніми версіями операційної системи Android та відповідними апаратними платформами. Команда підтримки завжди готова відповісти на запитання,

проблеми та відгуки користувачів щодо програми. Відгуки користувачів будуть активно збиратися та аналізуватися, щоб визначити області для поліпшення і впровадити функції або вдосконалення, які користувачі просять.

Команда підтримки надаватиме пріоритет аспектам безпеки та конфіденційності програмного забезпечення.

Висновки до розділу

Підтримка, що надається, відіграє вирішальну роль у підтримці функціональності, продуктивності та задоволеності користувачів. Усуваючи помилки, оптимізуючі продуктивність, забезпечуючи сумісність і надаючи пріоритет безпеці та конфіденційності, команда підтримки гарантує, що програма залишається надійною, ефективною та актуальною.

Своєчасно виправляючи помилки та оновлюючи застосунок, команда підтримки вирішує будь-які проблеми, про які повідомляється, забезпечуючи безперебійну роботу користувачів. Зусилля з оптимізації продуктивності покращують швидкість відгуку та використання ресурсів застосунку, дозволяючи йому безперебійно працювати на різних пристроях.

Підтримка сумісності з останніми версіями операційної системи Android та апаратними платформами гарантує, що користувачі можуть продовжувати користуватися застосунком без перерв. Інтеграція з новими апаратними функціями або технологіями розширює можливості застосунку, дозволяючи йому залишатися на передовій технологій безпеки водіїв.

Надаючи особливого значення безпеці та конфіденційності, команда підтримки проводить регулярні аудити безпеки, впроваджує заходи для захисту даних користувачів та забезпечує дотримання правил конфіденційності. Така прихильність до безпеки даних вселяє впевненість у користувачів і сприяє створенню надійного середовища для роботи з застосунками.

В цілому, підтримка ПЗ забезпечує довговічність і надійність програми. Завдяки постійному вдосконаленню та клієнтоорієнтованому підходу,

команда підтримки дозволяє користувачам користуватися високоякісним, багатофункціональним та безпечним рішенням для виявлення сонливості.

ВИСНОВКИ

В результаті виконання дипломного проекту було спроектовано мобільний застосунок для детекції сонливості. В якості середовища розробки обрано Android Studio.

Розробка та впровадження застосунку Детектор Сонливості з використанням алгоритму відстеження очей на платформі Android з використанням бібліотеки ML Kit дозволила отримати значні практичні результати.

Застосунок успішно інтегрує передовий алгоритм відстеження руху очей з платформою Android, використовуючи можливості бібліотеки ML Kit.

Після реалізації застосунку він був протестований на пристроях з різними версіями Android, з різними розмірами екранів щоб переконатися, що застосунок акуратно відображається на різних пристроях. Завдяки широкому тестуванню та оцінці, застосунок демонструє ефективне виявлення сонливості на основі рухів очей, надаючи своєчасні попередження користувачам.

Точність програми у розрізненні нормальних рухів очей від сонливості, а також її здатність мінімізувати помилкові спрацьовування та хибні негативні результати підтверджують наукову основу алгоритму відстеження руху очей.

Реалізація алгоритму відповідає сучасному стану науково-технічних знань у галузі виявлення сонливості та комп'ютерного зору. Використовуючи бібліотеку ML Kit, застосунок використовує методи машинного навчання та попередньо навчені моделі для підвищення точності та ефективності відстеження очей, що відображає останні досягнення в цій галузі.

Потенційні сфери використання застосунку охоплюють різні галузі, включаючи транспорт, логістику та будь-яку іншу сферу, де нещасні випадки, спричинені втомою, можуть становити ризик. Завдяки широкому використанню пристроїв на базі Android, застосунок має потенціал для широкого впровадження та впливу на безпеку водіїв.

Розробка застосунку вирішує важливу соціальну проблему, зменшуючи ризики, пов'язані з керуванням автомобілем у стані сонливості, та підвищуючи безпеку дорожнього руху. Вплив застосунку виходить за межі технічних аспектів, сприяючи підвищенню добробуту та продуктивності людей, зменшенню аварійності та потенційно рятуючи життя.

Успішне впровадження застосунку Детектор Сонливості з використанням відстеження руху очей і машинного навчання закладає основу для подальших досліджень і вдосконалень у цій галузі. Наступні дослідження можуть бути спрямовані на підвищення точності, надійності та продуктивності алгоритму в реальному часі, а також на інтеграцію застосункових сенсорних даних для покращення виявлення сонливості. Як наприклад, алгоритм розпізнавання позіхання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) The royal society for the prevention of accidents driver fatigue and road accidents. A literature review and position paper february 2001. URL: https://web.archive.org/web/20170301005823/http://www.ibrarian.net/navon/paper/driver_fatigue_and_road_accidents_a_literature_re.pdf?paperid=1229744
2) National Highway Traffic Safety Administration Overview, 2021. URL: https://www.nhtsa.gov/risky-driving/drowsy-driving
3) Prevalence of motor vehicle crashes involving drowsy drivers, United States, 2009-2013. URL: https://newsroom.aaa.com/wp-content/uploads/2019/06/AAAFoundation-DrowsyDriving-Nov2014.pdf
4) Automated driving levels of driving automation are defined in new SAE international standard J3016 URL: https://web.archive.org/web/20180701034327/https://cdn.oemoffhighway.com/files/base/acbm/ooh/document/2016/03/automated_driving.pdf
5) Canalys: Overview URL: https://www.telecomtv.com/content/automotive/canalys-8-of-new-cars-in-europe-sold-with-level-2-autonomy-driving-features-36247/
6) Sleepy Foundation Overview URL: https://www.sleepfoundation.org/drowsy-driving
7) Transportation Research Part F: Traffic Psychology and Behaviour Volume 10, Issue 1, January 2007, Pages 1-10 URL: https://www.sciencedirect.com/science/article/abs/pii/S1369847806000246?via%3Dihub
8) An Evaluation of Emerging Driver Fatigue Detection Measures and Technologies – June 30, 2009 by Lawrence Barr (Author), Stephen Popkin (Author), Heidi Howarth (Author), U.S. Department of Transportation Federal Motor Carrier Safety Administration (Author)
9) Real-Time Driver's Fatigue Detection: Sensing, Analysis and Alarming by Mohsen Davoudi (Author), Mehdi Davoudi (Author), Mehdi Pazhoohesh (Author) URL: https://www.researchgate.net/publication/236155292_Real-Time_Driver's_Fatigue_Detection
10) Learning Multimodal Representations for Drowsiness Detection Kun Qian , Senior Member, IEEE, Tomoya Koike, Student Member, IEEE, Toru Nakamura, Member, IEEE, Björn W. Schuller , Fellow, IEEE, and Yoshiharu Yamamoto , Member, IEEE URL: https://d-nb.info/1251242146/34
11) ML Kit Vision Quickstart Sample App URL: https://github.com/googlesamples/mlkit/tree/master/android/vision-quickstart
12) ML Kit's face mesh detection API URL: https://developers.google.com/ml-kit/vision/face-mesh-detection

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015394093

Дата перевірки:
02.06.2023 14:25:38 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
02.06.2023 14:27:30 EEST

ID користувача:
76913

Назва документа: **IT-94_Курілець_ПЗ**

Кількість сторінок: **54** Кількість слів: **8778** Кількість символів: **67667** Розмір файлу: **3.13 MB** ID файлу: **1015058461**

12.5%
Схожість

Найбільша схожість: **9.44%** з джерелом з Бібліотеки (ID файлу: **1015058496**)

1.8% Джерела з Інтернету	70	Сторінка 56
12.4% Джерела з Бібліотеки	107	Сторінка 57

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0.07%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за **8 слів** та **0%**)

0% Вилучення з Інтернету	20	Сторінка 58
0.07% Вилученого тексту з Бібліотеки	26	Сторінка 58

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	2

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ВИЯВЛЕННЯ СОНЛИВОСТІ У ВОДІЇВ ДЛЯ ЗАПОБІГАННЯ ДТП

Текст програми

КПІ.ІТ-9411.045490.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Валерій КУРІЛЕЦЬ

Київ – 2023

Файл Drowsiness.java

```
 public class Drowsiness {
private static final float PERCENT_DROWSY= 0.45f;
private static final int MAXIMUM_FRAMES = 11;
   public long lastFRAME;
   private final ArrayDeque<Boolean> FRAMES = new ArrayDeque<>();
   public boolean Drowsy(Face faceimage) {
      boolean Drowsy = true;
      lastFRAME = System.currentTimeMillis();
      if (face.getLeftEyeOpenProbability() == null
         || face.getRightEyeOpenProbability() == null) {
         return false;
      }
      if (face.getLeftEyeOpenProbability() < MAXIMUM_FRAMES
         && face.getRightEyeOpenProbability() < MAXIMUM_FRAMES) {
         FRAMES.addLast(true);
      } else {
         FRAMES.addLast(false);
      }
      if (history.size() > MAXIMUM_FRAMES) {
         history.removeFirst();
      }
      if (history.size() == MAXIMUM_FRAMES) {
         for (boolean instance : FRAMES) {
            Drowsy &= instance;
         }
      } else {
         return false;
      }
      return Drowsy;
   }
}
```

Файл DrowsinessDetection.java

```
public class DrowsinessDetection extends VideoActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
   }
   @Override
   protected void setProcessor() {
      cameraSource.setMachineLearningFrameProcessor(new
FaceDetectorProcessor(this));
   }
}
```

Файл Main.java

```
public class MainActivity extends AppCompatActivity implements AlgoListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ArrayList<Algo> arrayList = new ArrayList<>();

        arrayList.add(new Algo(R.drawable.baseline_time_to_leave_black_48,
"Детектор сонливості", DrowsinessDetection.class));
      ));
        AlgoAdapter algoAdapter = new AlgoAdapter(arrayList, this);
        RecyclerView recyclerView = findViewById(R.id.main_recycler_view);
        recyclerView.setAdapter(algoAdapter);
        recyclerView.setLayoutManager(new GridLayoutManager(this, 2));
    }

    @Override
    public void onAlgoSelected(Algo algo) {
        Intent intent = new Intent(this, algo.activityClazz);
        intent.putExtra("name", algo.algoText);
        startActivity(intent);
    }
}

class AlgoAdapter extends RecyclerView.Adapter<AlgoViewHolder> {

    private List<Algo> algoList;
    private AlgoListener algoListener;

    public AlgoAdapter(List<Algo> algoList, AlgoListener listener) {
        this.algoList = algoList;
        this.algoListener = listener;
    }

    @NonNull
    @Override
    public AlgoViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_icons, parent,
false);
        return new AlgoViewHolder(view, algoListener);
```

```
    }

    @Override
    public void onBindViewHolder(@NonNull AlgoViewHolder holder, int
position) {
        holder.bind(algoList.get(position));
    }

    @Override
    public int getItemCount() {
        return algoList.size();
    }
}

class AlgoViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {

    private ImageView iconImageView;
    private TextView algoTextView;
    private AlgoListener algoListener;
    private Algo algo;

    public AlgoViewHolder(@NonNull View itemView, AlgoListener
algoListener) {
        super(itemView);
        itemView.setOnClickListener(this);
        this.algoListener = algoListener;

        iconImageView = itemView.findViewById(R.id.iconImageView);
        algoTextView = itemView.findViewById(R.id.algoTextView);
    }

    public void bind(Algo algo) {
        this.algo = algo;
        iconImageView.setImageResource(algo.iconResourceId);
        algoTextView.setText(algo.algoText);
    }

    @Override
    public void onClick(View v) {
        if (algoListener != null) {
            algoListener.onAlgoSelected(algo);
        }
    }
}
```

```
class Algo<T extends ImageClassificationActivity> {
    public int iconResourceId = R.drawable.ic_launcher_foreground;
    public String algoText = "";
    public Class<T> activityClazz;

    public Algo(int iconResourceId, String algoText, Class<T> activityClazz) {
        this.iconResourceId = iconResourceId;
        this.algoText = algoText;
        this.activityClazz = activityClazz;
    }
}

interface AlgoListener {
    void onAlgoSelected(Algo algo);
}
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ВИЯВЛЕННЯ СОНЛИВОСТІ У ВОДІЇВ ДЛЯ ЗАПОБІГАННЯ ДТП

Програма та методика тестування

КПІ.ІТ-9411.045490.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Валерій КУРІЛЕЦЬ

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є мобільний застосунок на базі Android, що використовує алгоритми для виявлення сонливості та розпізнавання обличчя.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка сумісності мобільного застосунку зі смартфонами Pixel 2, Pixel 4, Pixel 6.

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- системне тестування – перевіряється усе програмне забезпечення в цілому;
- мануальне тестування – тестування без використання автоматизації.
- тестування «білої скриньки» – об’єктом тестування тут є внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним;
- тестування інтерфейсу користувача на пристроях Pixel 2, Pixel 4, Pixel 6 у Android Studio.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Для перевірки якості та виявлення помилок та недоліків у функціональній частині програмного забезпечення та його зручності в користуванні застосовано ручне тестування. Планується виконати наступні види тестування, щоб перевірити працездатність та стійкість додатку:

- тестування на пристроях Pixel 2, Pixel 4, Pixel 6;
- тестування інтерфейсу користувача;
- тестування ефективності та швидкодії застосунку;
- тестування зручності використання.

Ці тести допоможуть нам впевнитися в належній працездатності та
- надійності нашого додатку, а також забезпечити зручне використання користувачами.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ВИЯВЛЕННЯ СОНЛИВОСТІ У ВОДІЇВ ДЛЯ ЗАПОБІГАННЯ ДТП

Керівництво користувача

КПІ.ІТ-9411.045490.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Валерій КУРІЛЕЦЬ

Київ – 2023

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«Детектор Сонливості» – це мобільний застосунок для виявлення сонливості у водіїв у режимі реального часу. У випадку, якщо ознаки сонливості були розпізнані – мобільний додаток відправить гучне аудіо-повідомлення. На разі він використовує алгоритм розпізнавання заплющених очей.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

- наявність мобільного пристрою з операційною системою на базі Android з версією 6.0 або вище;
- для встановлення додатку на мобільному пристрої повинно бути не менше 70 МБ вільної пам’яті.

2.2 Завантаження застосунку

На даний момент застосунок можна встановити власноруч, використовуючи відповідний apk-файл. Для цього спершу необхідно завантажити його на мобільний пристрій, а потім, використовуючи інсталятор, виконати встановлення даного додатку.

2.3 Перевірка коректної роботи

По завершенню встановлення додатка на робочому столі мобільного пристрою повинна відобразитись іконка даного застосунку. У разі, якщо дана іконка не з’явилась, то встановлення відбулось не успішно. Інакше користувач має змогу запустити додаток, клацнувши на його іконку. Після натискання повинна відобразитись початкова сторінка застосунку.

3 ВИКОНАННЯ ПРОГРАМИ

При запуску програмного застосунку користувачу буде відображено запит для надання доступу до камери. (рисунок 3.1).

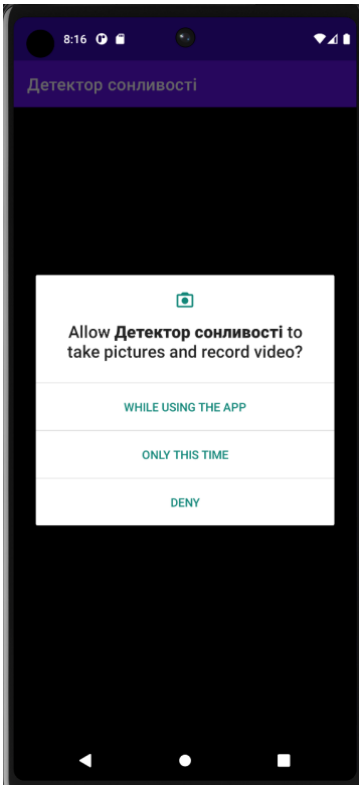

Рисунок 3.1 – Початкова сторінка додатку

Користувач повинен надати доступ. Далі відобразиться відео потік з бек-камери, який буде розпізнавати ознаки сонливості у режимі реального часу (рисунок 3.2) Треба встановити телефон перед обличчям.

Рисунок 3.2 – Постановка телефона

Якщо ознаки сонливості будуть розпізнані – мобільний застосунок відправить гучне аудіо-сповіщення. Щоб завершити роботу програми треба її закрити.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ВИЯВЛЕННЯ СОНЛИВОСТІ У ВОДІЇВ ДЛЯ ЗАПОБІГАННЯ ДТП

Графічний матеріал

КПІ.ІТ-9411.045490.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Валерій КУРІЛЕЦЬ

Київ – 2023