НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

«На правах рукопису»
УДК 004.4

ДО ЗАХИСТУ ДОПУЩЕНО
В.о. завідувача кафедри
_____Олександр КОВАЛЬ

"……"_....._2024 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці»
зі спеціальності 121 Інженерія програмного забезпечення
на тему: «Програмний застосунок аналізу стану водія під час керування транспортним засобом»

Виконала: студентка 2 курсу, групи ТВ-з21мп
_____Войтих Крістіна Михайлівна_____ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник: доцент, к.е.н., Гусєва І.І._____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент: _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Київ – 2024

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

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти другий, магістерський
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення інтелектуальних кібер - фізичних систем в енергетиці»

ЗАТВЕРДЖУЮ
В.о.завідувача кафедри
_____ Олександр КОВАЛЬ
(підпис)
«____» _____ 202_р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

_____Войтих Крістіни Михайлівни_____
(прізвище, ім'я, по батькові)

1. Тема дисертації: _____ Програмний застосунок аналізу стану водія під час керування транспортним засобом _____
науковий керівник дисертації_____ доцент, к.е.н., Гусєва І.І.___
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від "08"листопада 2023 року № 5202-с
2. Строк подання студентом дисертації _10 січня 2024 року
3. Об'єкт дослідження____ процес керування водія транспортним засобом у різних умовах руху_____
4. Предмет дослідження (Вихідні дані – для магістерської дисертації за ОПП): оцінка стану водія, виведення спеціальних сигналів або попереджень в разі погіршення стану під час водіння, створення звітів та надання рекомендацій водіям для покращення їхнього стану та безпеки на дорозі.
5. Перелік питань, які потрібно розробити: проаналізувати існуючі рішення; розробити методи для збору даних про водійську активність та стан; розробити математичні та статистичні методи для аналізу даних та визначення стану водія; розробити програмний застосунок; розробити інтерфейс; провести тести.
6. Орієнтований перелік ілюстративного матеріалу Огляд існуючих систем, розгляд схем архітектур, аналіз діаграм прецедентів і класів, розгляд моделей баз даних та процес тестування системи є ключовими етапами у розгляді розробленої системи.

7. Дата видачі завдання «01» листопада 2022р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.11.2022	виконано
2	Дослідження предметної області	01.11.2022 - 01.12.2022	виконано
3	Постановка вимог до проєктування системи	01.12.2022 - 15.12.2022	виконано
4	Дослідження існуючих рішень	15.12.2022 - 03.01.2023	виконано
5	Підготовка публікацій	03.01.2023 - 03.03.2023	виконано
6	Розробка програмного продукту	03.03.2023 - 20.09.2023	виконано
7	Тестування	20.09.2023 - 15.10.2023	виконано
8	Захист програмного продукту	16.10-2023 – 20.10.2023	виконано
9	Підготовка магістерської дисертації	21.10.2023 – 17.11.2023	виконано
10	Передзахист	18.12.2023 - 22.12.2023	виконано
11	Захист	15.01.2024 – 19.01.2024	виконано

Студент _____ (підпис) _____ (прізвище та ініціали)

Науковий керівник _____ (підпис) _____ (прізвище та ініціали)

РЕФЕРАТ

Актуальність теми. У світі, де автомобільна мобільність є неодмінною частиною нашого життя, кожен день водії постають перед новими викликами і вимогами. Нині, більше ніж будь-коли раніше, дорожні аварії стають нагальною проблемою, що загрожує життю і безпеці наших доріг. Ці аварії, замість того, щоб ставати все рідше, зростають у кількості, і це необхідно визнати. Значна частина цих подій є наслідком недисциплінованості водіїв, недостатньої уваги та реакційності за кермом, а також загальної неуважності. Спостереження за такими аспектами поведінки водіїв та розробка засобів для їх аналізу набувають критичного значення. Саме в цьому контексті постає необхідність в розробці програмного застосунку, який міг би систематично аналізувати стан водія під час керування транспортним засобом.

Мета роботи полягає в розробці програмного застосунку, який дозволить аналізувати стан водія під час керування транспортним засобом з метою підвищення безпеки на дорозі.

Об'єктом дослідження є процес керування транспортним засобом у різних умовах руху.

Предметом дослідження є програмне забезпечення для визначення стану водія під час керування транспортним засобом.

Методи дослідження. Теоретичні підходи, такі як аналіз, узагальнення та абстрагування, і емпіричні методи, такі як проведення експериментів, вимірювання, а також аналіз компонентів.

Практичне значення полягає у можливості впровадження системи аналізу стану водія транспортного засобу. Збір і аналіз даних про стан водія в реальному часі може слугувати основою для вдосконалення безпеки на дорозі, вдосконалення роботи автотранспорту.

Структура та обсяг дисертації. Магістерська дисертація складається зі вступу, шести розділів, висновку, переліку посилань з 25 найменувань, містить 24

рисунків, 9 таблиць. Повний обсяг магістерської дисертації складає 89 сторінки, з яких перелік посилань займає 2 сторінки, додатки займають 23 сторінку.

Ключові слова: інтерфейс програмування застосунків, дорожньо-транспортна пригода, інтерфейс користувача, GPS, React.js. стан водія.

ABSTRACT

Relevance of the topic. In a world where automotive mobility is an integral part of our daily lives, drivers face new challenges and demands every day. Currently, road accidents have become an urgent problem threatening lives and road safety more than ever before. Instead of decreasing in frequency, these accidents are on the rise, and this is a reality that needs to be acknowledged. A significant portion of these incidents results from driver indiscipline, insufficient attention, and responsiveness behind the wheel, as well as general negligence. Observing such aspects of driver behavior and developing tools for their analysis become critically important. It is in this context that the necessity arises for the development of a software application that could systematically analyze the driver's condition while operating a vehicle.

The objective of this work is to develop a software application that allows the analysis of the driver's state during the operation of a vehicle with the aim of enhancing road safety.

The research object is the process of operating a vehicle in various traffic conditions.

The subject of the research is software for determining the driver's state during the operation of a vehicle.

Research methods. Theoretical approaches such as analysis, abstraction, and generalization, and empirical methods such as conducting experiments, measurements, as well as component analysis.

Practical significance lies in the potential implementation of a system for analyzing the driver's state during the operation of a vehicle. Real-time data collection and analysis of the driver's state can serve as the basis for improving road safety and enhancing the efficiency of transportation.

Structure and scope of the dissertation. The master's dissertation consists of an

introduction, six chapters, a conclusion, a list of references with 7 titles, includes 29 figures, 9 tables. The complete volume of the master's dissertation is 117 pages, of which the list of references takes up 1 page, and the appendices occupy 23 pages.

Keywords: application programming interface, road traffic accident, user interface, GPS, React.js, driver's state.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

API - інтерфейс програмування застосунків;

ДТП - дорожньо-транспортна пригода;

UI - інтерфейс користувача;

CV - комп'ютерне зорове сприйняття;

GPS - глобальна система позиціонування;

MySQL - Система управління базами даних;

IoT - Інтернет речей;

SDK - комплект розробки програмного забезпечення;

ETA - приблизний час прибуття;

CPU - центральний процесор;

GUI - графічний інтерфейс користувача.

ВСТУП

З плином часу сучасні технології стають необхідною складовою автомобільного транспорту. Технологічний прогрес призвів до створення автомобілів, обладнаних найновішими електронними системами, які покликані покращити безпеку на дорозі та комфорт водіїв та пасажирів. Однак, незважаючи на всі технологічні досягнення, головною ланкою в системі безпеки на дорозі залишається людина - водій. Від її уваги, психофізичного стану та реакційності залежить більшість дорожніх інцидентів.

Питання безпеки на дорогах стає належним викликом для громадського здоров'я, особливо оглядаючи статистику, яка підтверджує щоденну втрату понад 3 000 життів у світі внаслідок дорожньо-транспортних подій. Ці пригоди не тільки призводять до трагічних втрат людських життів, але й приносять глобальні економічні втрати, які оцінюються у 518 мільярдів доларів США щорічно, що важить на економіки країн, які розвиваються.

У країнах, які розвиваються, витрати на наслідки дорожньо-транспортних подій досягають 100 мільярдів доларів США на рік, подвійно перевищуючи річний обсяг допомоги на розвиток. В Україні зростає кількість ДТП, з 14% збільшенням за 2021 рік, досягаючи 154 480, і призводить до втрат 2592 життів.

За статистикою, в середньому в Україні відбувається близько 500 аварій щодня, вимагаючи великих зусиль поліції та інших відомств для аналізу та визначення причин. Важливість адекватної оцінки причин аварій очевидна для визначення кримінальної відповідальності та компенсації збитків.

Необхідність ефективних заходів для зменшення цих статистичних показників дуже висока. Успішність таких заходів залежить від адекватної оцінки причин подій на дорогах. Це важливий аспект для визначення кримінальної відповідальності та відшкодування зазначених втрат.

Ця робота присвячена розробці програмного засобу, який здатний моніторити та аналізувати фізичний стан водія під час керування автомобілем. Метою роботи є створення інноваційного інструмента, який може сприяти покращенню безпеки на дорозі, вчасному виявленню ризикових ситуацій та надавати водіям рекомендації для підвищення їхньої уваги та реакційності.

Цей програмний застосунок може бути корисним для автомобільних виробників, правоохоронних органів та, безумовно, для самих водіїв, які прагнуть забезпечити більшу безпеку на дорозі. У межах даної роботи планується дослідити різні аспекти стану водія, включаючи психологічний стан, рівень втомленості, а також різні фактори, які можуть впливати на якість керування.

У першому розділі розглядається мета проведення дослідження, призначення розробленого програмного забезпечення та завдання, які воно призначено вирішувати.

Другий розділ присвячений огляду існуючих методів та аналогічних систем для аналізу стану водія під час керування транспортним засобом.

Третій розділ присвячений обґрунтуванню вибору інструментів розробки для створення програмного продукту.

Четвертий розділ включає опис архітектури системи, структури класів, алгоритму контуризації та розрахунку відстаней між об'єктами.

П'ятий розділ розглядає інтерфейс користувача та взаємодію користувача з системою.

Шостий розділ презентує концепцію стартапу, враховуючи сучасні економічні показники.

У висновках роботи оцінюються результати розробленого програмного забезпечення та набуті знання під час досліджень у предметній області.

1 ПОСТАНОВКА ЗАВДАННЯ ВИЗНАЧЕННЯ СТАНУ ВОДІЯ ПІД ЧАС КЕРУВАННЯ ТРАНСПОРТНИМ ЗАСОБОМ

1.1 Мета роботи та формулювання завдання

Мета: розробка програмного застосунку, який дозволить аналізувати стан водія під час керування транспортним засобом з метою підвищення безпеки на дорозі.

Об’єкт: процес керування транспортним засобом у різних умовах руху.

Предмет: програмне забезпечення для визначення стану водія під час керування транспортним засобом.

Завдання, які потрібно розв’язати для досягнення мети:

- провести аналіз існуючих рішень для визначення стану водія під час керування транспортним засобом;
- розробити підхід для збору даних про водійську активність та стан, такі як дані зі сенсорів автомобіля, камер, акселерометрів тощо;
- розробити алгоритм для аналізу даних та визначення стану водія, включаючи оцінку рівня утомленості, ступеня уваги;
- розробити архітектуру програмного застосунку та базу даних для зберігання даних;
- розробити інтерфейс для спілкування з користувачем та виведення результатів аналізу;
- провести тести для перевірки точності та надійності розробленого програмного застосунку.

Система складається з мобільного застосунку та серверної частини.

Мобільний застосунок - це інтерактивний інструмент, доступний на смартфонах та планшетах водіїв, який сприяє збору та передачі даних, таких як інформація з сенсорів і GPS. Цей застосунок також забезпечує взаємодію з водіями

та дозволяє їм передавати дані на сервер для подальшого аналізу та відображення результатів.

Серверна частина, у свою чергу, відповідає за зберігання, обробку і аналіз отриманих даних, забезпечуючи безпеку і надійність системи, а також можливість взаємодії з мобільними застосунками та масштабованість для обробки великої кількості даних.

Користувачі: водії транспортних засобів, дослідницькі організації та університети.

Вхідні дані: особисті дані користувача, такі як вік, стать, вага та пульс, дані GPS смартфона, швидкість автомобіля, та дані з додаткових пристроїв – електродермальна активність, прискорення, гальмівна реакція.

Вихідні дані: оцінка стану водія, виведення спеціальних сигналів або попереджень в разі погіршення стану під час водіння, створення звітів та надання рекомендацій водіям для покращення їхнього стану та безпеки на дорозі.

Серверна частина додатку створена з використанням технологій React Native, React Native Maps та React Native Maps Directions. Для зберігання даних використана Firebase Realtime Database, яка є нереляційною базою даних.

Мобільний додаток розроблений з використанням бібліотеки React Native UI Kitten. Крім того, для розгортання додатку на мобільних пристроях використано середовище розробки Android Studio.

1.2 Призначення системи

Система визначення стану водія є комплексною технологічною платформою, призначеною для аналізу та моніторингу фізичного стану водія під час керування транспортним засобом. Її основним завданням є забезпечення безпеки дорожнього руху, покращення комфорту процесу водійського керування.

Призначення системи включає в себе наступні аспекти:

- безпека на дорозі: система спрямована на забезпечення безпеки дорожнього руху шляхом вчасного виявлення ознак утомленості та інших факторів, які можуть впливати на здатність водія ефективно керувати автомобілем.
- уникнення аварій: попередження можливих аварійних ситуацій через аналіз та інтерпретацію даних про стан водія, що дозволяє системі вчасно реагувати на потенційно небезпечні умови.
- поліпшення водійського досвіду: надання водію інформації щодо його стану та порад для поліпшення зосередженості, реакції та загального комфорту водіння.
- допомога поліції та розслідування ДТП: надання поліції та слідчим інструменту для об'єктивного аналізу аварійних ситуацій та установлення причин виникнення ДТП.
- судова експертиза: допомога судовим органам в установленні обставин аварії та прийнятті обґрунтованих рішень в судових справах.
- страхові аспекти: надання страховим компаніям об'єктивних даних для точного визначення відповідальності учасників аварії та розгляду страхових випадків.
- інтеграція з автомобільними технологіями: взаємодія з автомобільними системами для покращення безпеки та ефективності керування.

Розроблене програмне забезпечення має підвищити загальний рівень безпеки на дорозі, забезпечуючи аналіз та виявлення стану водія під час керування транспортним засобом, а також сприяти оптимізації дорожнього руху та покращенню досвіду водія.

Висновки до розділу 1

Програмне забезпечення для визначення стану водія під час керування транспортним засобом представляє собою інноваційний і комплексний підхід до покращення безпеки на дорозі та оптимізації діяльності водія. Метою системи є не лише аналіз фізичного стану водія, але й активна попереджувальна реакція на потенційно небезпечні ситуації.

Мобільний застосунок та серверна частина взаємодіють, створюючи надійну технологічну платформу для збору, обробки та аналізу різноманітних даних про активність водія. Система включає в себе алгоритми для оцінки рівня утомленості, ступеня уваги та інших важливих параметрів, що впливають на безпеку та ефективність керування.

Застосунок розроблено з урахуванням інтерактивності та зручності для водіїв, надаючи їм можливість взаємодії та передавання важливих даних для подальшого аналізу. Серверна частина забезпечує надійне зберігання даних, їх обробку та аналіз.

Призначення системи охоплює не лише безпеку на дорозі, але й важливі аспекти, такі як уникнення аварій, поліпшення досвіду водія та допомога в розслідуванні ДТП. Інтеграція з автомобільними технологіями дозволить забезпечити ще більшу ефективність та безпеку управління транспортним засобом.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

2.1 Продукти компанії Zendrive

Zendrive (рис. 2.1) - це компанія, яка надає рішення для аналізу поведінки водія та безпеки на дорозі через мобільні SDK та API.

Переваги використання Zendrive:

- аналіз поведінки водія: Zendrive дозволяє моніторити та аналізувати різні аспекти поведінки водія, такі як прискорення, гальмування, швидкість та використання смартфону. Це може бути корисним для виявлення ризикової або агресивної їзди;
- управління ризиками: за допомогою аналітики Zendrive можна виявити фактори, які призводять до аварій або небезпеки на дорозі, і приймати заходи для зменшення ризику;
- безпека на дорозі: використання Zendrive сприяє безпеці на дорозі, оскільки система може виявляти небезпечні ситуації та реагувати на них;
- співпраця зі страховими компаніями: Zendrive може бути корисним для страхових компаній, які бажають аналізувати водійську поведінку та надавати знижки на страховку водіям.

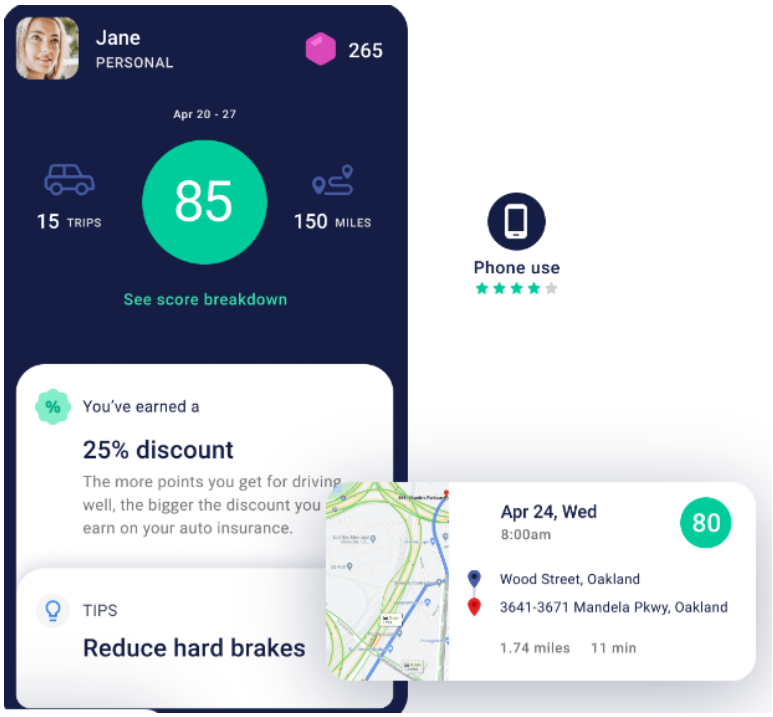


Рисунок 2.1 – Приклад інтерфейсу додатка від Zendrive

Серед недоліків Zendrive виділяються такі:

- приватність даних: використання Zendrive передбачає збір та аналіз особистих даних водія, що може порушити приватність та викликати питання щодо зберігання та захисту цих даних;
- залежність від технології: використання Zendrive вимагає певного рівня технологічних знань та інтеграції з існуючими системами, що може бути складним для деяких користувачів;
- вартість: використання послуг Zendrive може бути витратним для компаній та організацій, які бажають імплементувати їх в свої системи;
- обмеження аналізу: аналіз водійської поведінки обмежується даними, які можуть бути зібрані та оброблені системою, тож важко виявити всі аспекти поведінки водія.

Загалом, Zendrive може бути корисним інструментом для аналізу та покращення безпеки на дорозі, але варто враховувати приватність даних та інші обмеження при його використанні.

2.2 Мобільний застосунок TrueMotion Family

TrueMotion Family (рис. 2.2) - це мобільний застосунок, який спрямований на аналіз та покращення безпеки на дорозі для сімей та батьків, які бажають моніторити та поліпшити поведінку водія своїх дітей або інших членів сім'ї.

Переваги TrueMotion Family:

- моніторинг водійської активності: TrueMotion Family дозволяє вам моніторити активність водія, включаючи швидкість, прискорення, гальмування та рівень використання смартфону під час водіння. Є можливість перевіряти, як водії дотримуються безпечної поведінки на дорозі;
- безпека на дорозі: застосунок сприяє безпеці на дорозі шляхом надання статистики та рекомендацій щодо покращення поведінки водія;
- сповіщення та повідомлення: TrueMotion Family може надсилати сповіщення та повідомлення на смартфони батьків чи головного користувача, коли відбуваються небезпечні події або порушення правил дорожнього руху;
- інтерактивність: застосунок дозволяє взаємодіяти зі своєю сім'єю, обговорювати поведінку водія та встановлювати правила для безпеки на дорозі.

Недоліки TrueMotion Family:

- приватність: використання TrueMotion Family може порушувати приватність водіїв, оскільки застосунок моніторить їхню поведінку водія та місцезнаходження;
- залежність від смартфону: для ефективної роботи TrueMotion Family водіям потрібно мати смартфон з встановленим застосунком та активним з'єднанням з Інтернетом;
- вартість: деякі функції TrueMotion Family можуть вимагати щомісячних або річних підписок, що можуть бути витратними.

Рисунок 2.2 – Мобільний додаток TrueMotion Family

Загалом, TrueMotion Family може бути корисним інструментом для батьків та сімей, які бажають стежити за безпекою своїх дітей на дорозі та покращити їхню водійську поведінку, але варто бути уважним до питань приватності та витрат на користування.

2.3 Компанія Mobileye

Mobileye (рис. 2.3), яка є дочірньою компанією Intel, спеціалізується на розробці систем візійного спостереження для автомобілів. Їхні технології орієнтовані на покращення безпеки на дорогах та включають в себе різноманітні функції для спостереження та реагування на різні ситуації на дорозі.

Рисунок 2.3 – Компанія Mobileye

Переваги Mobileye:

- розпізнавання дорожніх знаків: Mobileye використовує розпізнавання дорожніх знаків для інформування водія про обмеження швидкості, знаки "стоп" та інші дорожні позначення;
- системи уникнення зіткнень: системи Mobileye можуть виявляти ризик зіткнення з іншими автомобілями, пішоходами та об'єктами, активуючи аварійне гальмування або вживаючи інші заходи для уникнення зіткнення;
- виявлення втомленості водія: Mobileye використовує алгоритми аналізу поведінки водія для виявлення ознак втомленості чи відволікання водія та висилання відповідних попереджень;
- машинне навчання: системи Mobileye використовують технології машинного навчання для покращення своєї продуктивності та адаптації до різних умов дорожнього руху.

Недоліки Mobileye:

- залежність від погодних умов: деякі системи можуть бути менш ефективними в умовах поганої видимості, таких як туман, дощ або сніг.

- вартість інтеграції: додавання систем Mobileye до автомобіля може бути дорогим заходом, зокрема враховуючи витрати на установку та обслуговування.
- обмежена автономія: системи Mobileye, хоч і високоефективні, можуть бути обмежені у високо-ступінчатому автономному водінні в порівнянні з іншими передовими системами.

Незважаючи на всі переваги, важливо враховувати, що жодна система не є ідеальною, і Mobileye не виняток. Залежність від погодних умов та обмежена автономія є деякими викликами, які можуть виникнути в процесі експлуатації. Проте, з урахуванням сталих інновацій та подальшого розвитку, система Mobileye продовжує вносити вагомий внесок у сферу автомобільної безпеки та відкривати нові можливості для ефективного використання технологій на дорогах.

2.4 Опис необхідного набору даних

Для розробки програмного застосунку аналізу стану водія під час керування транспортним засобом необхідний набір даних, який включає різноманітні параметри та показники для аналізу водійської поведінки під час керування [2], серед яких можна виділити такі:

- швидкість автомобіля: вимірювання швидкості дозволяє визначити, наскільки водій дотримується дорожніх обмежень та як реагує на зміни швидкості руху. Береться швидкість в конкретний момент часу, використано вбудований GPS смартфона для вимірювання поточної швидкості;
- пульс водія: вимірювання пульсу водія може надати інформацію про ступінь виразності стресу або емоцій під час керування. Потрібно використовувати пульсометр, вбудований у смартфон або годинник, або окремий медичний прилад для вимірювання пульсу;

- електродермальна активність (EDA): EDA вказує на зміни в потових залозах та може свідчити про ступінь напруги або нервового збудження водія. Для вимірювання активності використовуються спеціальні сенсори, електроди, які наносяться на шкіру руки, пальців чи інших ділянках тіла. Зміни в електричній провідності, які реєструються сенсорами, відображаються на моніторі або комп'ютерному програмному забезпеченні. В контексті водіння, електродермальна активність може бути використана для вивчення реакції водія на різні дорожні ситуації або стресові моменти під час керування автомобілем.
- прискорення та гальмівна реакція: дані про прискорення та гальмівну реакцію вказують на спроможність водія взаємодіяти з автомобілем та реагувати на дорожні ситуації;
- геолокаційні дані (GPS): GPS-дані надають інформацію про маршрут руху, розташування та шлях, що пройдений водієм;
- персональні характеристики водіїв: Для аналізу водійської поведінки корисно враховувати персональні характеристики водіїв, такі як вік, стаж водіння, стан здоров'я тощо.

Набір даних, який включає ці параметри, дозволить нам провести глибокий аналіз поведінки водія під час керування транспортним засобом та дослідити фактори, що впливають на безпеку на дорозі.

2.5 Дослідження стану водія з використанням допоміжних приладів

Стан водія можна дослідити за допомогою багатьох даних, які можна отримати з різних приладів та сенсорів, серед них основні – прискорення, гальмівна реакція та електродермальна активність (EDA).

Прискорення в контексті водіння визначає темп зміни швидкості автомобіля. Це важливий параметр, який впливає на безпеку на дорозі та водійську поведінку.

Під час керування автомобілем, водій може використовувати прискорення для розгону, перевищення обмежень швидкості або реакції на різні дорожні ситуації. Проте, важливо враховувати, що надмірне прискорення може призвести до небезпечної їзди та аварій.

Дані про прискорення можуть бути корисними для аналізу поведінки водія. Вони дозволяють виявити, наскільки водій дотримується правил швидкості та як він реагує на зміни руху [3]. Аналіз акселерації може виявити агресивне водіння, надмірну швидкість або нестабільність руху, що може бути показником низької безпеки на дорозі.

Для покращення безпеки на дорозі та аналізу поведінки водія, прискорення може бути важливим параметром, який враховується в програмних застосунках аналізу стану водія. Відстеження та аналіз прискорення допомагає зрозуміти, як водії взаємодіють з дорожніми умовами та іншими учасниками руху, що сприяє покращенню безпеки на дорозі.

Акселерометри (рис. 2.4) - це пристрої, призначені для вимірювання прискорення. Вони використовуються в різних пристроях, включаючи смартфони, планшети і електронні годинники. Акселерометри вимірюють прискорення в трьох вимірах: по горизонталі (ось X), вертикалі (ось Y) та вздовж осі Z. Дані з акселерометра можуть використовуватися для визначення змін швидкості та положення.

Рисунок 2.4 – Акселерометр

В більшості сучасних автомобілів є вбудовані сенсори прискорення, які вимірюють рух автомобіля. Ці сенсори зазвичай використовуються для активних систем безпеки, таких як системи антиблокування гальм (ABS) та системи контролю стабільності (ESP), показані на рисунках 2.5 та 2.6.

Рисунок 2.5 – Антиблокування гальм (ABS)

Рисунок 2.6 – Система контролю стабільності (ESP)

Позитивне прискорення - це збільшення швидкості (натиснено на педаль газу та прискорено автомобіль).

Негативна акселерація (декелерація або гальмування) - це зменшення швидкості (натиснено гальмо або відпущена педаль газу) .

Є кілька варіацій середнього прискорення. В цьому випадку швидкість змінюється, але не настільки різко, як при позитивному або негативному прискоренні. Наприклад, коли поступово натискати на газ або гальмо.

Електродермальна активність (EDA) є важливим фізіологічним показником, який вимірює активність потових залоз шкіри. Цей параметр використовується для вивчення реакцій організму на різні подразники, включаючи стрес, емоції та фізичні зусилля. У контексті водіння, вимірювання електродермальної активності може бути корисним для аналізу психофізіологічного стану водія та виявлення факторів, що впливають на безпеку на дорозі.

Основні моменти стосовно електродермальної активності в контексті водіння включають:

- стрес та емоції: електродермальна активність може свідчити про рівень стресу та емоцій водія під час керування. Збільшена активність потових залоз може бути показником тривожності або нервового збудження;
- виявлення втоми: зміни в електродермальній активності можуть також вказувати на ступінь втоми водія. Втомлений водій може бути менш уважним та менше реагувати на дорожні обставини;
- взаємодія з дорожніми умовами: електродермальна активність може відображати, як водій реагує на різні дорожні умови, такі як дощ, сніг або слизька дорога;
- співпраця з іншими системами моніторингу водія: дані щодо електродермальної активності можуть бути інтегровані з іншими системами моніторингу водія, такими як відеокамери або GPS, для більш комплексного аналізу водійської поведінки.

Електродермальна активність може допомогти розуміти, як фізіологічні фактори впливають на водійську поведінку та безпеку на дорозі. Аналіз EDA може бути корисним для розробки програмних застосунків, спрямованих на покращення безпеки на дорозі та водійського комфорту.

Електрична активність шкіри демонструє значні зміни в різних емоційних станах суб'єкта, що добре відомо з наукових досліджень. На рисунку 2.7 наведено ілюстрацію електродермального сигналу (EDA), який служить прикладом виявлення таких фізіологічних змін. Аналіз електродермальної активності може

бути корисним для визначення емоційного стану та реакцій суб'єкта у різних ситуаціях, що відкриває нові можливості для розуміння психофізіологічних аспектів водійської поведінки.

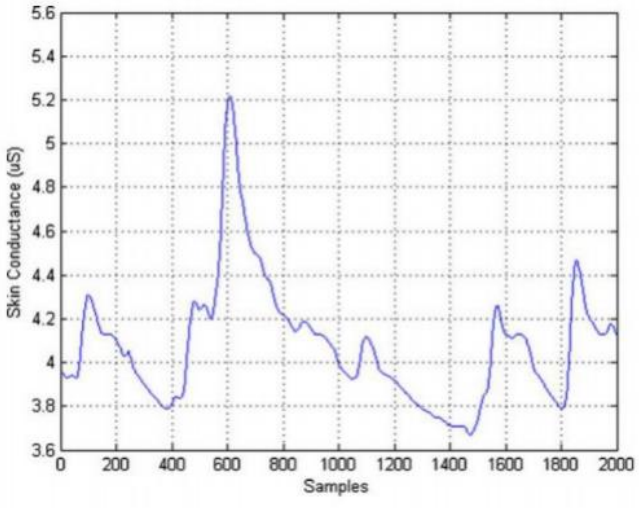


Рисунок 2.7 – Графік сигналу EDA

Гальмівна реакція є важливим аспектом водійської безпеки та пов'язана з здатністю водія адекватно реагувати на негативні дорожні ситуації та екстрені обставини. Гальмівна реакція визначає, як швидко водій може зупинити автомобіль після сприйняття небезпеки та натискання на гальмо.

Основні аспекти, пов'язані з гальмівною реакцією в контексті водіння, включають:

- час реакції: це час, який проходить від моменту сприйняття небезпеки (наприклад, виявлення перешкоди на дорозі) до моменту, коли водій розпочинає натискати на гальмо. Короткий час реакції є важливим для уникнення аварій;
- час гальмування: це час, який потрібен від моменту натискання на гальмо до моменту повної зупинки автомобіля. Він залежить від багатьох факторів, включаючи швидкість руху, стан дорожнього покриття та стан гальмівної системи автомобіля;

- відстань гальмування: це відстань, яку автомобіль подолає під час гальмування від моменту натискання на гальмо до повної зупинки. Відстань гальмування може бути визначена шляхом вимірювання відстані, яку автомобіль проїхав під час гальмування;
- вплив факторів середовища: екологічні фактори, такі як стан дорожнього покриття, погодні умови та видимість, можуть впливати на гальмівну реакцію водія. Наприклад, в дощову або сніжну погоду гальмування може займати більше часу та відстані.

Гальмівна реакція є ключовим аспектом дорожньої безпеки, і вона може бути аналізована та вдосконалена для запобігання аваріям та покращення реакції водіїв на небезпеку. Для цього можуть використовуватися технології моніторингу водія та системи екстреного гальмування.

Глобальна система позиціонування (GPS) представляє собою сучасну технологію навігації, що визначає точне місцезнаходження смартфона і надає можливість планувати маршрути та знаходити об'єкти на карті. Практично кожен сучасний телефон обладнаний GPS-модулем, який є антеною, налаштованою на отримання сигналів від супутників GPS.

Для визначення поточних координат пристрою необов'язково використовувати Інтернет, достатньо лише активувати передачу геоданих на смартфоні. Для уникнення залежності від Інтернету під час навігації рекомендується завчасно завантажити карти, які не вимагають підключення до мережі, хоча це призведе до збільшення обсягу пам'яті вашого телефону.

Важливо відзначити, що активне використання навігаційної системи може призвести до значного розряджання акумулятора, тому рекомендується вимикати передачу геоданих у випадках, коли це необхідно.

Висновки до розділу 2

Головною відмінністю між запропонованим нами програмним застосунком і схожими системами, є акцент на визначенні стану водія під час керування транспортним засобом та наданні рекомендацій для підтримки фізичного здоров'я. Хоча інші програми і системи спрямовані на аналіз типу їзди та виявлення агресивної поведінки на дорозі, запропонований програмний застосунок зосереджений на тому, як саме водій почувається під час водіння і чи має він які-небудь ознаки, які можуть вплинути на безпеку на дорозі.

Основні переваги програмного застосунку включають доступність для широкого кола користувачів, оскільки він повністю безкоштовний. Це робить його доступним для різних верств населення і допомагає залучити більше водіїв до збереження свого фізичного здоров'я та покращення безпеки на дорозі.

Програма не тільки виявляє потенційні проблеми водіїв, але й надає їм рекомендації щодо подальших дій і, в разі необхідності, вчасно звернення до лікаря. Це може бути важливим кроком у запобіганні можливих проблем зі здоров'ям та забезпеченні безпеки на дорозі.

Для ефективної роботи програмного застосунку, який аналізує стан водія, потрібен комплекс даних, включаючи різноманітні параметри та показники, що аналізують поведінку водія під час керування. Ці дані можна отримати з різних приладів та сенсорів, включаючи, але не обмежуючись, швидкість руху, пульс водія, прискорення, гальмівну реакцію та електродермальну активність.

Дослідження стану водія з використанням допоміжних приладів, таких як системи моніторингу гальмування та електродермальні сенсори (EDA), надає важливі дані для аналізу та розуміння фізіологічних та емоційних аспектів водійської поведінки. Прискорення та гальмувальна реакція можуть вказувати на рівень агресивності чи ступінь реакції водія на різні ситуації на дорозі. Електродермальна активність дозволяє визначити рівень емоційної збудженості водія, що може бути важливим фактором для безпеки на дорозі.

Ці дані можуть бути використані для розробки програмних застосунків та систем, спрямованих на покращення безпеки на дорозі та водійського комфорту. Детальний аналіз даних дозволяє виявити не лише стилі водіння, але й виявити ознаки втоми, стресу чи інших негативних впливів на водія. Застосування сучасних технологій та аналізу сенсорних даних в сфері водійської безпеки може сприяти розробці інноваційних рішень та заходів, спрямованих на запобігання нещасних випадків на дорогах.

3 АПАРАТ ВИРІШЕННЯ ПОСТАВЛЕНОГО ЗАВДАННЯ

3.1 Аналіз архітектури ПЗ

У розробці проектів на React існують різні підходи до архітектури, залежно від розміру проекту, складності завдань та вимог до масштабованості. В рамках даної роботи були використані основні архітектури та принципи для проєктування комплексних систем, а саме - архітектурний підхід Flux та Container/Presentational components. Всі вказані архітектурні підходи розроблені для вирішення різних завдань, що дозволяє їх комбінувати між собою та інтегрувати з іншими архітектурами, не порушуючи базових принципів кожної з них. Розглянемо ці архітектурні концепції більш детально.

Компонентна архітектура React додатків. Сучасна веброзробка використовує архітектуру на основі компонентів, як ефективний та масштабований підхід до створення користувацьких інтерфейсів. React, популярна бібліотека JavaScript, розроблена Facebook, відіграє ключову роль у популяризації цієї парадигми. Нижче розглядаються основні принципи, переваги та ілюстрація компонентної архітектури в React-додатках.

Принципи компонентної архітектури:

1) Будівельні блоки компонентів: React підтримує розбиття інтерфейсу користувача на невеликі та автономні компоненти, кожен з яких має конкретну функцію. Ці компоненти виступають будівельними блоками для React-застосунку.
2) Презентаційні та контейнерні компоненти: Чітке розмежування компонентів, які відповідають за відображення інформації (презентаційні), та тих, що керують логікою та станом застосунку й(контейнерні), полегшує зрозумілість та обслуговування коду.

3) Стан компонента: кожен компонент React може зберігати внутрішній стан, що визначає його поведінку та зовнішній вигляд. Зміни у стані автоматично знову викликають рендеринг компонента, забезпечуючи динамічний інтерфейс користувача.

Переваги компонентної архітектури в React:

- масштабованість та легкість розробки: модульність дозволяє ефективно масштабувати та розширювати застосунок, полегшуючи розробку.
- обслуговуваність та підтримка відлагодження: розподіл коду на компоненти спрощує обслуговування та відлагодження. Виявлення та усунення помилок стає більш простим завдяки ізольованості проблем.
- повторне використання коду: компоненти спеціально розроблені для повторного використання. Розробники можуть використовувати компоненти в різних частинах застосунку чи навіть в інших проектах, що сприяє ефективній розробці.

У React презентаційний компонент — це компонент, який лише рендерить HTML. Його єдина функція — це відображення розмітки. У застосунку, який використовує Redux, презентаційний компонент не взаємодіє зі сховищем Redux.

Презентаційний компонент приймає властивості (props) від контейнерного компонента. Контейнерний компонент вказує дані, які повинен відобразити презентаційний компонент. Контейнерний компонент також вказує поведінку. Якщо у презентаційного компонента є яка-небудь інтерактивність, наприклад, кнопка, він викликає функцію через властивість (prop), надану йому контейнерним компонентом. Контейнерний компонент відповідає за відправку дій до сховища Redux. Все це показано на рисунку 3.1.

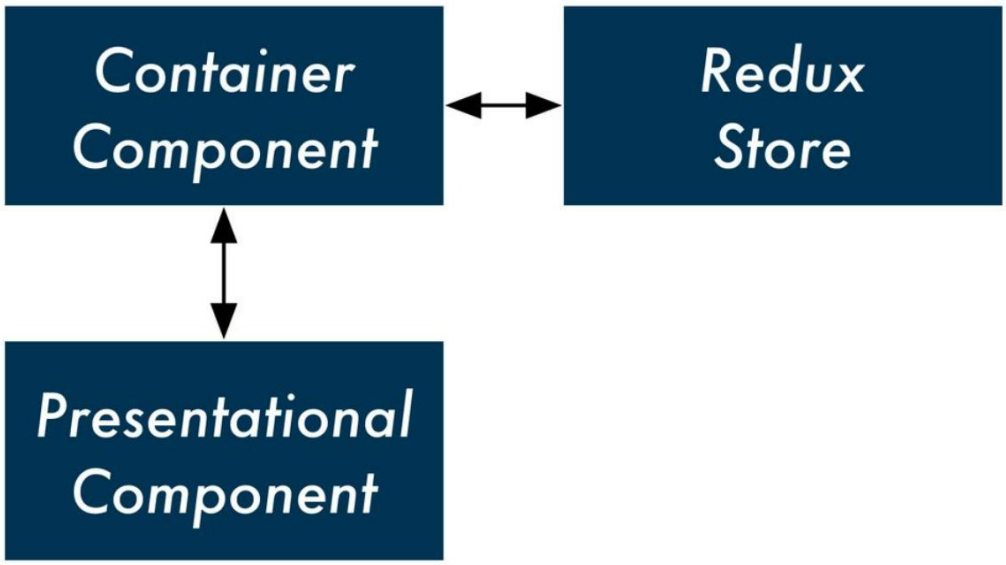


Рисунок 3.1 – Container/Presentational components архітектура

Компонентна архітектура в React представляє собою потужний та ефективний підхід до розробки сучасних вебзастосунків. Фокусуючись на модульності та ізоляції функціональностей, React дозволяє розробникам створювати чистий, зрозумілий та легко розширюваний інтерфейс користувача. Ця архітектурна парадигма значно вплинула на спосіб проектування та реалізації вебзастосунків, сприяючи успіху та широкому поширенню React серед розробників.

Архітектурний підхід Flux. Архітектурний підхід Flux набуває широкої популярності, особливо завдяки компанії Facebook та їхній бібліотеці для створення користувацьких інтерфейсів React, його адаптація для React отримала назву Redux. Однією з ключових ідей та відмінностей цього підходу є використання однонаправленого потоку даних.

У порівнянні з іншими архітектурними підходами Flux визначається тим, що дані переміщаються в одному напрямку. В багатьох шаблонах MV* потік даних є двонаправленим, де дані можуть перетікати між різними шарами в обох напрямках.

Flux складається з таких ключових елементів, як диспетчер (dispatcher), сховище даних (store), представлення (view) та дії (action). Застосунки, які

використовують цю архітектуру, реалізують взаємодію між цими компонентами для керування станом та оновлення інтерфейсу.

Схематично цей підхід можна представити у вигляді діаграми (рис. 3.2), де диспетчер координує діями, сховище даних зберігає стан застосунку, а представлення відображає інтерфейс користувача та реагує на зміни в стані.

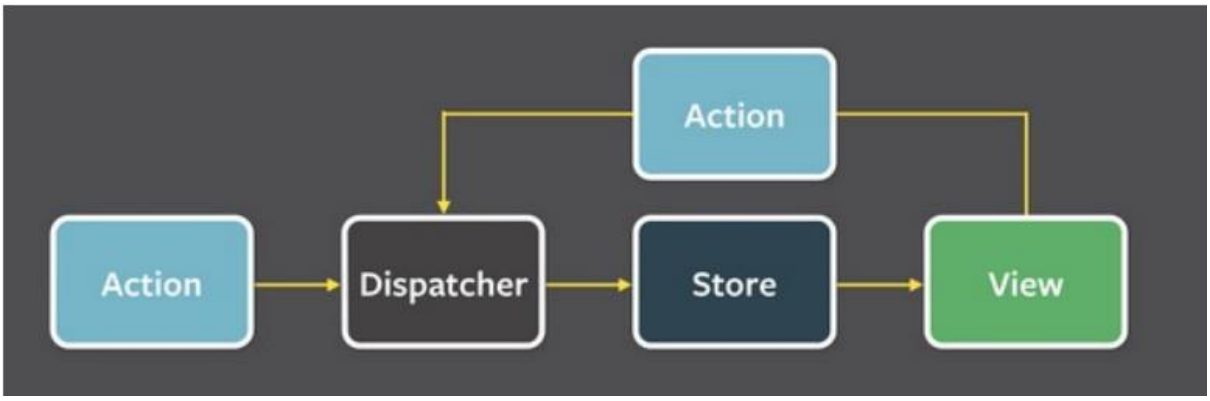


Рисунок 3.2 – Схематичне зображення патерну Flux

Action представляє собою об'єкт, який містить інформацію про подію чи зміну, що відбулася в застосунку. Ця подія може бути спричинена взаємодією користувача або іншими чинниками. Action слугує інструкцією для системи про те, що потрібно зробити.

Dispatcher є центральним елементом, який отримує Actions та розсилає їх до всіх зареєстрованих Stores. Dispatcher відповідає за управління потоком даних і гарантує, що Actions передаються правильним частинам додатку.

Store представляє собою контейнер, який утримує стан додатку та реалізує бізнес-логіку. Він реагує на Actions, змінює свій внутрішній стан і сповіщає всі підписані Views про зміни. Store є одним з головних джерел правди у Flux.

View відповідає за відображення даних користувачу та взаємодію з ним. View підписується на зміни в Store і оновлює свій інтерфейс, коли стан застосунку змінюється.

Загальний потік даних:

- користувач взаємодіє з інтерфейсом та генерує Action.

- Action надсилається до Dispatcher.
- Dispatcher передає Action всім зареєстрованим Stores.
- відповідно до бізнес-логіки, Store оновлює свій стан та сповіщає всі підписані Views.
- Views оновлюють свій інтерфейс, відображаючи зміни.

Цей однонаправлений потік даних властивий патерну Flux та допомагає уникнути складних взаємодій та забезпечити прозору архітектуру застосунку.

3.2 Технології серверної частини

Серверна частина застосунку була створена з використанням технологій React Native, React Native Maps та React Native Maps Directions. Для зберігання даних була використана MySQL Database, яка є нереляційною базою даних.

Бібліотека React Native**.** React Native - це відкрита бібліотека для розробки мобільних застосунків, що здобула широку популярність серед розробників і компаній, які прагнуть створити мобільні застосунки для різних платформ. Відома своєю швидкістю та гнучкістю**.** React Native надає засоби для розробки мобільних застосунків, які працюють як на Android, так і на iOS, використовуючи спільний код та React-підходи [6].

Однією з ключових переваг React Native є здатність працювати на різних платформах. Використовуючи React Native, розробники можуть писати один код, який може бути використаний для створення застосунків для Android і iOS, що відрізняються тільки в окремих частинах. Це значно скорчує час та зусилля, необхідні для розробки для обох платформ [4].

Основні характеристики:

- кросплатформеність: однією з головних переваг React Native є можливість розробки застосунків, які працюють як на iOS, так і на Android, використовуючи майже один і той же код. Це значно економить час та ресурси розробників.

- декларативний синтаксис: React Native використовує декларативний синтаксис, що спрощує створення інтерфейсів та управління станом застосунків. Компонентний підхід робить код легким для розуміння та модифікації.
- використання JavaScript: розробка ведеться мовою JavaScript, що є широко поширеною серед веброзробників. Це дозволяє командам легко переосвоюватися та використовувати існуючі навички.
- гаряче перезавантаження (Hot Reloading): функція, яка дозволяє розробникам бачити зміни в реальному часі без необхідності повторного запуску застосунку. Це прискорює процес розробки та відлагодження.

Однією з ключових концепцій React Native є використання компонентів для побудови інтерфейсу. Компоненти можуть бути вбудованими або створюватися розробниками, що дозволяє побудувати складні інтерфейси з невеликою кількістю коду. Це сприяє повторному використанню коду та покращує організацію проекту.

Додатковий функціонал може бути доданий за допомогою нативних модулів, написаних мовами програмування, такими як Java чи Swift. Це дає можливість використовувати специфічні для платформи можливості та бібліотеки, додавати власні модулі та оптимізувати продуктивність додатку.

Бібліотека React Native виявилася потужним інструментом для розробки мобільних застосунків, забезпечуючи ефективність та гнучкість у процесі створення. Завдяки своїй кросплатформеності та зручному інтерфейсу розробки, вона займає важливе місце в індустрії та продовжує здобувати популярність серед розробників мобільних застосунків.

Бібліотека React Native Maps. React Native Maps - це бібліотека для розробки мобільних застосунків на платформі React Native, яка спеціалізується на інтеграції та використанні картографії у застосунках для iOS та Android. Забезпечуючи простий та ефективний спосіб відображення і взаємодії з картами, ця бібліотека стала невід'ємною частиною розробки мобільних застосунків, пов'язаних з локацією та геопросторовою інформацією.

Основні компоненти бібліотеки включають в себе <MapView>, який визначає область відображення карти, і <Marker>, що відповідає за позначення конкретних точок на карті. Завдяки цим компонентам, розробники можуть легко інтегрувати функціональність карт в свої додатки та взаємодіяти з локаційними даними.

Основні можливості:

- відображення карти: React Native Maps дозволяє вставляти і відображати карти Google Maps або Mapbox;
- маркери та мітки: є можливість легко додавати маркери та мітки на карту, які відображаються в конкретних географічних точках;
- прокладання маршрутів: React Native Maps надає можливість створювати та відображати маршрути між різними точками на карті;
- геолокація та відстеження руху: бібліотека підтримує роботу з геолокацією та може відстежувати рух користувача на карті;
- події та взаємодія: є можливість додавати обробники подій для взаємодії з картою, маркерами та іншими об'єктами на карті.

React Native Maps став важливим інструментом для розробників, що створюють застосунки з акцентом на геопросторову інформацію та локаційний сервіс. Його простота використання та гнучкість роблять його популярним вибором для тих, хто прагне додати картографію до своїх мобільних застосунків.

Бібліотека React Native Maps Directions. React Native Maps Directions – це бібліотека для розробки мобільних застосунків на платформі React Native, яка дозволяє інтегрувати та використовувати функціонал маршрутизації та картографії. Цей інструмент став важливим компонентом для розробників, що шукають зручні та ефективні рішення для роботи з мапами та прокладанням маршрутів у своїх додатках.

Основні можливості:

- прокладання маршрутів: бібліотека дозволяє легко прокладати маршрути між двома або більше точками на карті;

- розрахунок відстаней: React Native Maps Directions дозволяє розраховувати відстані між точками на карті;
- вибір типу транспорту;
- підтримка різних картографічних сервісів: React Native Maps Directions підтримує різні картографічні сервіси, включаючи Google Maps та Mapbox;
- візуалізація маршруту: після розрахунку маршруту бібліотека надає можливість візуалізувати маршрут на карті, відображаючи його на маркері або лінії.

Бібліотека надає ключовий компонент – <MapViewDirections>, який дозволяє визначити початкову та кінцеву точки маршруту, а також отримувати докладні дані про маршрут. Завдяки цьому компоненту розробники можуть легко інтегрувати функціональність маршрутизації у свої застосунки.

Основні можливості бібліотеки включають в себе прокладання маршрутів для різних видів транспорту, встановлення параметрів маршруту та відображення інформації про маршрут на карті. Це робить React Native Maps Directions ідеальним інструментом для розробки додатків, пов'язаних з транспортом, туризмом чи іншими сферами, де важливо надавати користувачам точні та зручні маршрути.

Інтеграція бібліотеки в React Native додатки відбувається просто та ефективно за допомогою npm-пакетів та конфігурації відповідних компонентів. Розробники можуть легко впроваджувати функціонал маршрутизації у свої проекти, роблячи їх більш інтерактивними та зручними для користувачів.

Отже, React Native Maps Directions відкриває нові можливості для розробників, дозволяючи їм легко інтегрувати функціонал маршрутизації та картографії в свої мобільні застосунки. Його простота використання та потужні функціональність роблять його популярним вибором для тих, хто прагне надати користувачам найкращий досвід взаємодії з картами та маршрутами.

База даних MySQL. MySQL — це відкрите програмне забезпечення для управління базами даних, яке використовує мову запитів SQL (Structured Query

Language). Розроблена та підтримується корпорацією Oracle, MySQL є однією з найпоширеніших реляційних систем управління базами даних (RDBMS).

Основні характеристики:

- реляційна система управління базами даних (RDBMS): MySQL використовує модель реляційної бази даних, що дозволяє зберігати та управляти даними у вигляді табличних структур.
- мова запитів SQL: інтерфейс взаємодії з базою даних використовує мову запитів SQL, що дозволяє виконувати операції вставки, вибірки, оновлення та видалення даних.
- підтримка транзакцій: MySQL підтримує транзакційні операції, що робить його надійним рішенням для застосунків, де важлива консистентність даних.
- масштабованість: MySQL здатний працювати на різних рівнях навантаження, від невеликих вебсайтів до великих корпоративних систем.
- відкрите програмне забезпечення: MySQL доступний за ліцензією GPL (GNU General Public License), що робить його безкоштовним та відкритим для використання та модифікацій.
- спільнота користувачів: існує активна спільнота користувачів та розробників, що надає підтримку та розвиває MySQL, а також допомагає вирішувати проблеми та розробляти нові можливості.
- широкі можливості: MySQL має багато функцій, таких як індексація, забезпечення цілісності даних, підтримка зовнішніх ключів, резервне копіювання даних та багато іншого.

MySQL широко використовується веброзробниками та організаціями для зберігання та обробки даних. Вона інтегрується з різноманітними мовами програмування та фреймворками, забезпечуючи зручний доступ до даних для розробників.

3.3 Технології мобільного застосунку

Для розробки обрано бібліотеку React Native UI Kitten, яка забезпечує розробникам велику кількість готових компонентів та стилів, що спрощують створення привабливого та функціонального інтерфейсу для застосунків. Крім того, для розгортання застосунку на мобільних пристроях було використано середовище розробки Android Studio [5].

Бібліотека React Native UI Kitten. React Native UI Kitten - це популярна бібліотека для розробки інтерфейсу користувача (UI) в мобільних застосунках, побудованих з використанням React Native [5].

Основні можливості React Native UI Kitten:

- готові компоненти: бібліотека містить велику кількість готових компонентів, таких як кнопки, іконки, тексти, картки, меню, та багато інших;
- гнучкість: React Native UI Kitten дозволяє налаштовувати стилі компонентів та внесення власних змін до їх вигляду;
- використання тем: бібліотека підтримує теми, що дозволяють визначити стилі для всього додатку;
- підтримка анімацій: React Native UI Kitten підтримує анімації для компонентів, що дозволяє створювати динамічні інтерфейси для користувачів;
- інтеграція з React Navigation.

React Native UI Kitten корисна для розробки будь-якого мобільного застосунку, який потребує стильного та функціонального інтерфейсу [3].

Засіб розгортання Android Studio. Android Studio - це інтегроване середовище розробки (IDE) для створення мобільних застосунків під платформу Android. Засіб розгортання Android Studio - це процес підготовки та розгортання створеного застосунку на реальний Android-пристрій або віртуальний пристрій (емулятор) [6].

Основні кроки розгортання застосунку в Android Studio:

- збірка проекту: є можливість використовувати опцію "Build" або "Rebuild Project" в Android Studio;
- налаштування віртуального пристрою або підключення реального пристрою;
- запуск застосунку: після успішної збірки проекту та вибору пристрою для розгортання, можна запустити додаток, натиснувши кнопку "Run" або "Debug" в Android Studio;
- тестування і налагодження;
- релізна збірка: ця версія повинна бути оптимізованою для виробництва та не містити знаків для відладки;
- публікація: можна опублікувати застосунок на магазині (наприклад, Google Play). Для цього потрібно буде створити обліковий запис для розробника та завантажити релізну версію застосунку разом із необхідною інформацією та іконками.

Android Studio надає інструменти та інтерфейс, які спрощують цей процес розгортання та публікації мобільних застосунків на платформі Android [6].

3.4 Якість коду та тестування

Для тестування та перевірки якості коду використовуються інструменти Jest та ESLint. Jest використовувався для написання та виконання автоматичних тестів, що дозволило впевнитися у правильній роботі застосунку та виявити помилки під час розробки. За допомогою ESLint були встановлені стандарти стилю програмування та автоматично перевірялися правила цього стилю, що сприяло підвищенню якості та читабельності коду в проекті [4].

Jest для тестування. Jest - це популярна бібліотека для тестування JavaScript-коду, особливо в контексті розробки застосунків на React, React Native та інших сучасних JavaScript-фреймворках [3].

Особливості та можливості Jest:

- простота використання;
- автоматичне спостереження за змінами;
- підтримка асинхронного коду: Jest підтримує асинхронні функції та дозволяє тестувати код, який виконується асинхронно, з використанням засобів, таких як `async/await`;
- спрощене створення віртуального DOM: для тестування React-компонентів;
- розширення можливостей за допомогою різних плагінів та розширень для виконання різних видів тестів та генерації звітів.

Jest є популярним вибором для тестування JavaScript-коду завдяки своїм можливостям та активній підтримці спільноти [4].

ESLint для стандартизації коду. ESLint - це інструмент для автоматичної перевірки та стандартизації JavaScript-коду [7]. Основні переваги ESLint включають:

- автоматичну перевірку коду;
- є можливість налаштувати правила ESLint відповідно до потреб та встановити власні стандарти стилю програмування;
- інтеграція з редакторами коду;
- підтримка великої кількості правил: ESLint має велику кількість вбудованих правил для стандартизації коду, і надає можливість встановити плагіни та правила сторонніх розробників;
- підвищення якості коду;
- збільшення співпраці в команді.

Використання ESLint рекомендується для підвищення якості та стандартизації JavaScript-коду в проекті.

Висновки до розділу 3

Система включає у себе дві ключові складові: серверну частину, яка взаємодіє з базою даних, і мобільний застосунок, розроблений за використання реактивних технологій, таких як React. Мобільний додаток виступає основним інтерфейсом для користувачів, дозволяючи їм взаємодіяти з системою та ефективно використовувати зібрані дані.

Однією з ключових особливостей розробленого рішення є використання технології MySQL для забезпечення надійності та швидкодії бази даних. Використання Android Studio дозволило оптимально адаптувати мобільний застосунок під операційну систему Android, забезпечуючи високий рівень зручності та продуктивності для користувачів.

Окрім функцій аналізу та зберігання даних, мобільний застосунок реалізований з урахуванням сучасних стандартів UI/UX-дизайну, надаючи ефективну та зручну взаємодію з користувачем. Такий комплексний підхід дозволяє забезпечити високий рівень безпеки та ефективності водіїв, сприяючи покращенню загального стану дорожнього руху.

4 РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Архітектура системи

Архітектура програмного застосунку для визначення стану водія під час керування транспортним засобом включає в себе компоненти та їх взаємозв'язки, що забезпечують функціонування та взаємодію всіх частин системи. Реалізація мобільного застосунку являє собою систему, яка отримує дані від водія, такі як його вік, пульс, швидкість руху і дані, які отримані за допомогою допоміжних засобів та сенсорів, такі як прискорення, гальмівна реакція, електродермальна активність. На основі цих значень здійснюється аналіз стану та стилю водіння водія, та користувач отримує рекомендації щодо покращення свого стану.

Також для аналізу, всі дані зберігаються в БД.

Програмна система складається з наступних модулів (рис. 4.1):

- користувацький інтерфейс: основна частина застосунку, що відповідає за візуальну частину. Складається з різноманітних елементів, таких як кнопки, тексти, текстильні поля, списки, картки та інші, які забезпечують інтерактивність та зручність взаємодії з користувачем;
- Redux або Flux: використовується для ефективного управління станом застосунку. "Actions" генеруються при подіях користувача, а "reducers" використовуються для модифікації стану застосунку. Це дозволяє уникнути ускладнень управління станом;
- Модуль для зберігання стану застосунку - Redux Store: зберігає стан застосунку та надає механізми для його управління. Розподілення стану забезпечує одностайність та простоту у взаємодії різних компонентів;
- модуль локального сховища: використовується для зберігання невеликих обсягів даних на пристрої користувача. Це може бути важливо для зберігання тимчасових або конфіденційних даних;

- модуль відправки даних до БД: організує відправку даних до бази даних. Цей модуль може взаємодіяти з серверною частиною застосунку та забезпечувати синхронізацію даних між клієнтом та сервером;
- база даних: може бути локальною базою даних, яка використовується на пристрої користувача, або віддаленою базою даних, яка розташована на сервері.

Колективно ці компоненти створюють ефективну та організовану архітектуру, яка дозволяє розділити відповідальності та покращити керованість проектом.

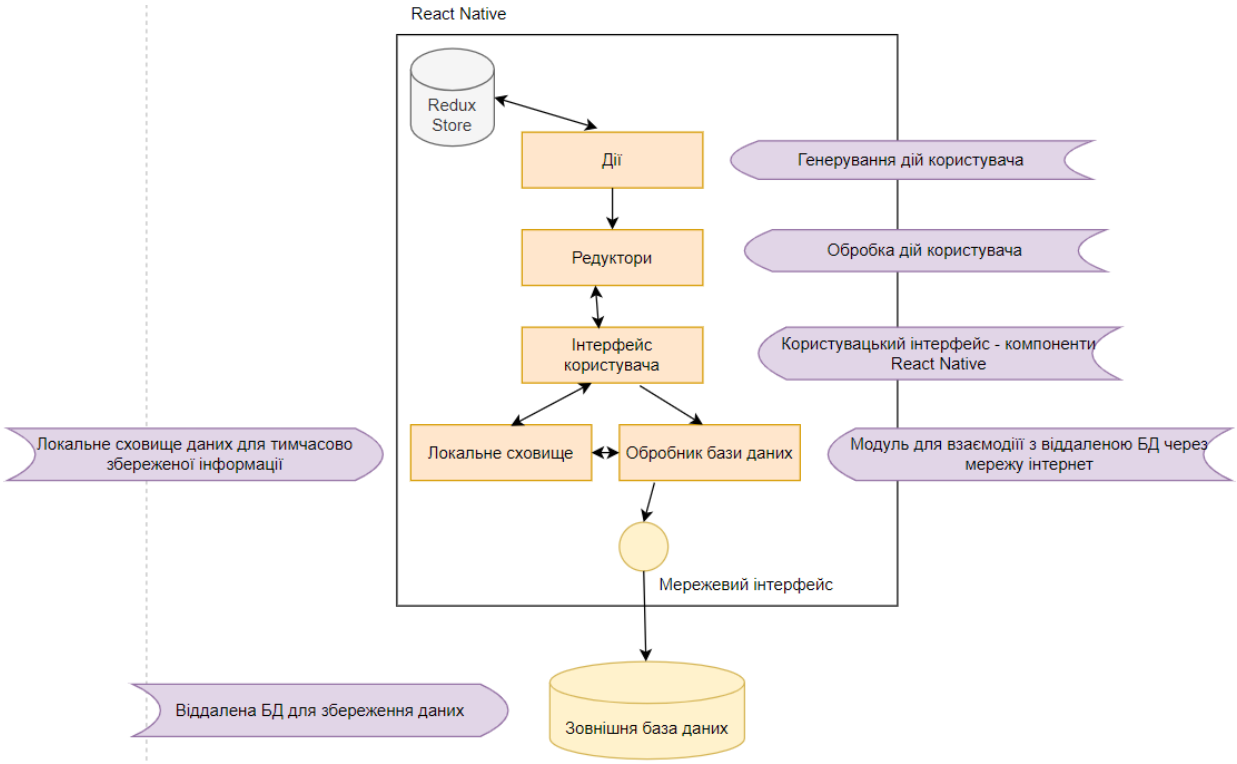


Рисунок 4.1 – Діаграма компонентів системи

У пакеті "System Interface" реалізований дизайн застосунку та вигляд елементів інтерфейсу програми. У пакеті "Source Files" містяться виконувані файли, які використовуються під час роботи програми.

4.2 Модуль збору даних

Модуль збору даних виконує важливу функцію, забезпечуючи збір та обробку даних про фізіологічний стан водія під час керування транспортним засобом. Додаткові дані отримуються з різних пристроїв та сенсорів, таких як акселерометри, GPS, інші сенсори тощо. Цей процес впроваджується за допомогою JavaScript скриптів, що зчитують заздалегідь зібрані дані та передають їх у базу даних.

У розробці, використовуючи React Native, дані отримуються через компоненти, написані на JavaScript або TypeScript. Крім того, використовуються стилі для ефективного відображення елементів інтерфейсу, що покращує користувацький досвід. Дані передаються у форматі, який може бути легко оброблений компонентами React Native.

За допомогою анотацій та обробників подій в React Native, отримані дані проходять аналіз, а результати передаються модулю аналізу та обробки для подальшого дослідження. Цей модуль виконує завдання аналізу та формує рекомендації, які можуть бути корисними для підвищення уваги та безпеки водія під час керування автомобілем.

4.3 Структура бази даних

Розроблений мобільний застосунок використовує реляційну модель бази даних MySQL. Взаємодія з базою даних відбувається через API, що дозволяє ефективно управляти та отримувати доступ до різних функціональних частин системи.

Концептуальна модель бази даних, зображена на рисунку 4.2, відображає структуру та зв'язки між різними таблицями. Ця модель служить основою для зберігання та організації інформації, що використовується мобільним застосунком для генерації статистики та рекомендацій на основі зібраних даних.

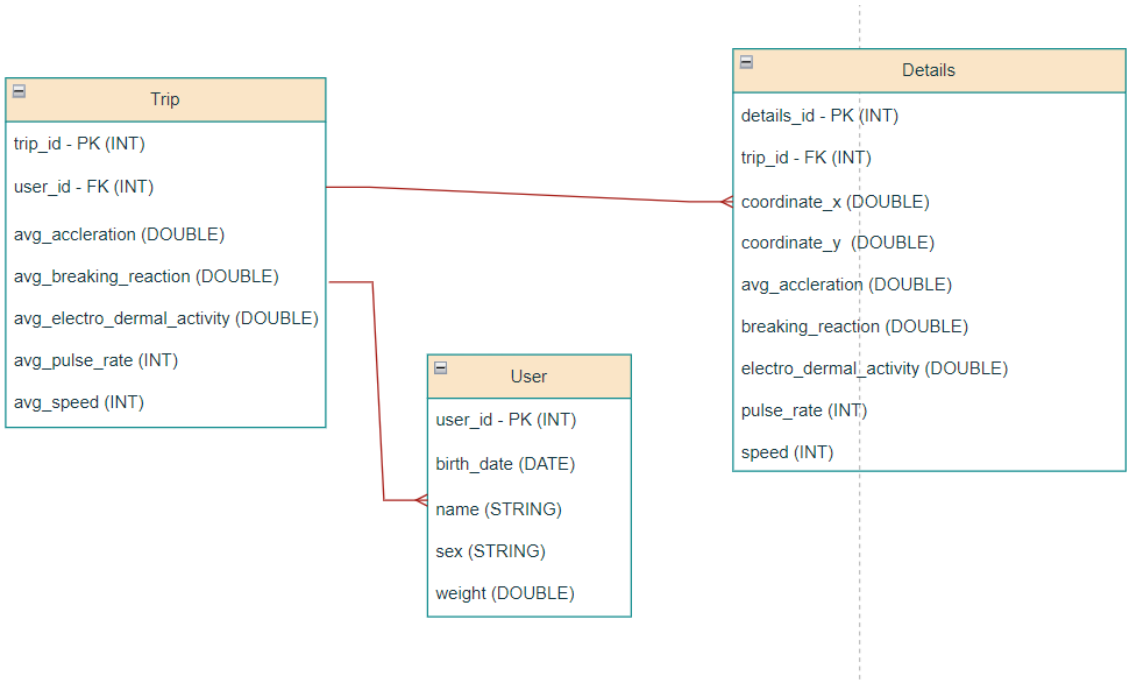


Рисунок 4.2 – Концептуальна модель бази даних системи

База даних складається з 3 таблиць.

Таблиця User зберігає дані про користувача та містить наступні поля:

- user_id;
- birth_date;
- name;
- sex;
- weight.

Таблиця Trip зберігає дані про поїздки конкретного користувача, маршрут, розраховані середні фізіологічні характеристики, дані з пристроїв та сенсорів, такі як швидкість, пульс водія, прискорення, електродермальна активність та гальмівна реакція:

- trip_id;
- user_id;
- avg_accleration;
- avg_breaking_reaction;
- avg_electro_dermal_activity;

- avg_pulse_rate;
- avg_speed.

Таблиця Details зберігає дані про поїздки в конкретний момент часу, координати по осям X та Y, а також дані про фізіологічні характеристики та дані з пристроїв у конкретний момент часу:

- details_id;
- trip_id;
- coordinate_x;
- coordinate_y;
- avg_accleration;
- breaking_reaction;
- electro_dermal_activity;
- pulse_rate;
- speed.

4.4 Алгоритм роботи програми

Програма використовує сукупність фізіологічних та технічних параметрів для аналізу стану водія з метою забезпечення безпеки на дорозі.

Спочатку відбувається збір даних, таких як швидкість, пульс водія, прискорення, електродермальна активність та гальмівна реакція. За допомогою скрипта програма заповнює цими даними БД.

Програма використовує алгоритми для аналізу зібраних даних. Швидкість порівнюється із допустимими нормами, визначаються зміни швидкості протягом коротких інтервалів часу для виявлення різкого гальмування або різкого прискорення. Пульс використовується для визначення аномальних змін, таких як підвищений пульс, що може свідчити про стрес або тривожність. Прискорення і гальмівна реакція аналізуються для виявлення різкого керування або несподіваних

рухів. Вимірювання електродермальної активності служать для визначення фізичної активності та рівня стимуляції водія.

Програма генерує рекомендації, якщо якісь із показників являються відхилені від норми, під час поїздки також позначає показник червоним, це означає, що це значення не в нормі, водій може зупинитися та перепочити.

Програма здійснює запис даних щодо стану водія та реакції водія на дорозі. Ці дані зазвичай зберігаються для подальшого аналізу та використання в контексті безпеки на дорозі.

4.5 Розробка вебзастосунку з використанням React.js

Для забезпечення зручності взаємодії з даними був створений веб-застосунок, реалізований з використанням мови програмування JavaScript. Це вирішує питання кросплатформенності та дозволяє запускати програму на будь-якому пристрої з підтримкою виходу в Інтернет.

Для створення проекту використовувалася бібліотека create-react-app, яка спрощує процес написання коду для застосунків та вирішує різноманітні завдання розробки, такі як тестування, комплектація та розгортання.

Щоб створити новий проект, достатньо було викликати команду create-react-app project_name, що призвело до створення нового проекту в поточній директорії з усіма необхідними залежностями та включеним сервером. Команда "yarn start" дозволяє запустити сервер, автоматично відкриває браузер за адресою http://localhost:3000/, слідкуючи за змінами у файлах та перебудовуючи застосунок при необхідності.

У React основною концепцією є компоненти, що виступають як фундаментальні будівельні блоки. Ці компоненти можуть отримувати різні параметри, які називаються props.

У вебзастосунках, стилі та їх значення зазвичай відповідають CSS, за винятком запису назв. У React використовується camelCase стиль, наприклад, backgroundColor замість background-color. Те саме стосується і назв подій.

Основні компоненти застосунку розміщені в папці components, конфігурації - в папці configs, сторінки - в папці components, маршрутизація - в папці routes, управління станом системи - в папці store, основні стилі - в папці styles, а допоміжні функції - в папці utils.

Було налаштовано сховище (store) за допомогою функції createStore з бібліотеки redux. У цю функцію передаються параметри reducer і enhancer, де reducer виступає як rootReducer, а enhancer використовує функцію composeWithDevTools з бібліотеки redux-devtools-extension, щоб можна було відстежувати зміни в сховищі у браузері. Також до цього процесу додається middleware (проміжний програмний засіб) sagaMiddleware за допомогою функції applyMiddleware з бібліотеки redux, і, якщо ми перебуваємо в режимі розробки, застосовується createLogger з бібліотеки redux-logger. Під час ініціалізації сховища також запускається sagaMiddleware. На рисунку 4.3 наведена організація файлів веб-клієнта.

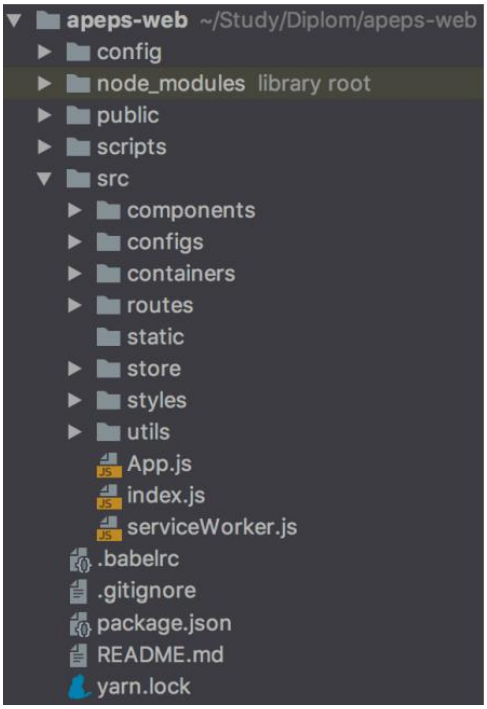


Рисунок 4.3 – Файлова структура проекту

Сховище (рис. 4.4) складається з дій (actions), які передають дані з програми до сховища. Вони є єдиним джерелом інформації для сховища, і для їх передачі необхідно викликати команду store.dispatch().

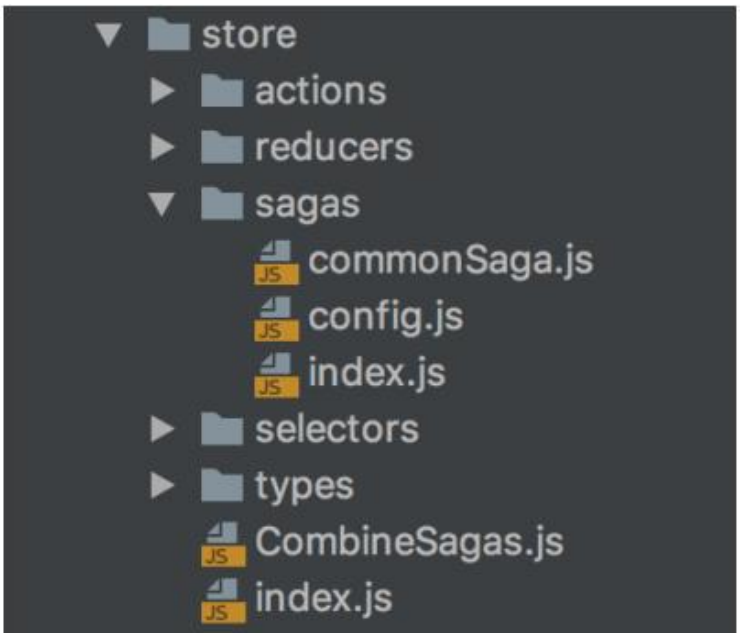


Рисунок 4.4 - Файлова структура для керування станом системи

Модуль "Selectors" включає в себе селектори, спеціально створені для мемоізації даних. У бібліотеці Reselect доступна функція createSelector, яка призначена для створення селекторів. Селектор перераховує дані лише тоді, коли один з його аргументів змінюється. Крім того, селектори можуть використовуватися в якості параметрів для інших селекторів.

У файлі config.js містяться налаштування для виконання запитів на сервер, тоді як у файлі commonSaga.js реалізована універсальна saga. Ця saga обробляє всі дії, які містять ключове слово "_REQUEST", і відправляє відповідний запит на сервер. Обробка запиту відбувається відповідно до конфігурацій, які визначені в файлі config.js.

Для забезпечення взаємодії між store та компонентами React використовується функція вищого порядку connect з бібліотеки react-redux. Функція вищого порядку - це функція, яка повертає іншу функцію і може приймати

інші функції як аргументи. Функція connect отримує в якості параметрів mapStateToProps і mapDispatchToProps.

Для того, щоб React знаходив store, необхідно оточити застосунок компонентом Provider з бібліотеки react-redux, який приймає store як властивість.

4.6 Діаграма класів

Діаграма класів, яка відображена на рисунку 4.6, надає структурний огляд використаних класів та їх взаємозв'язків.

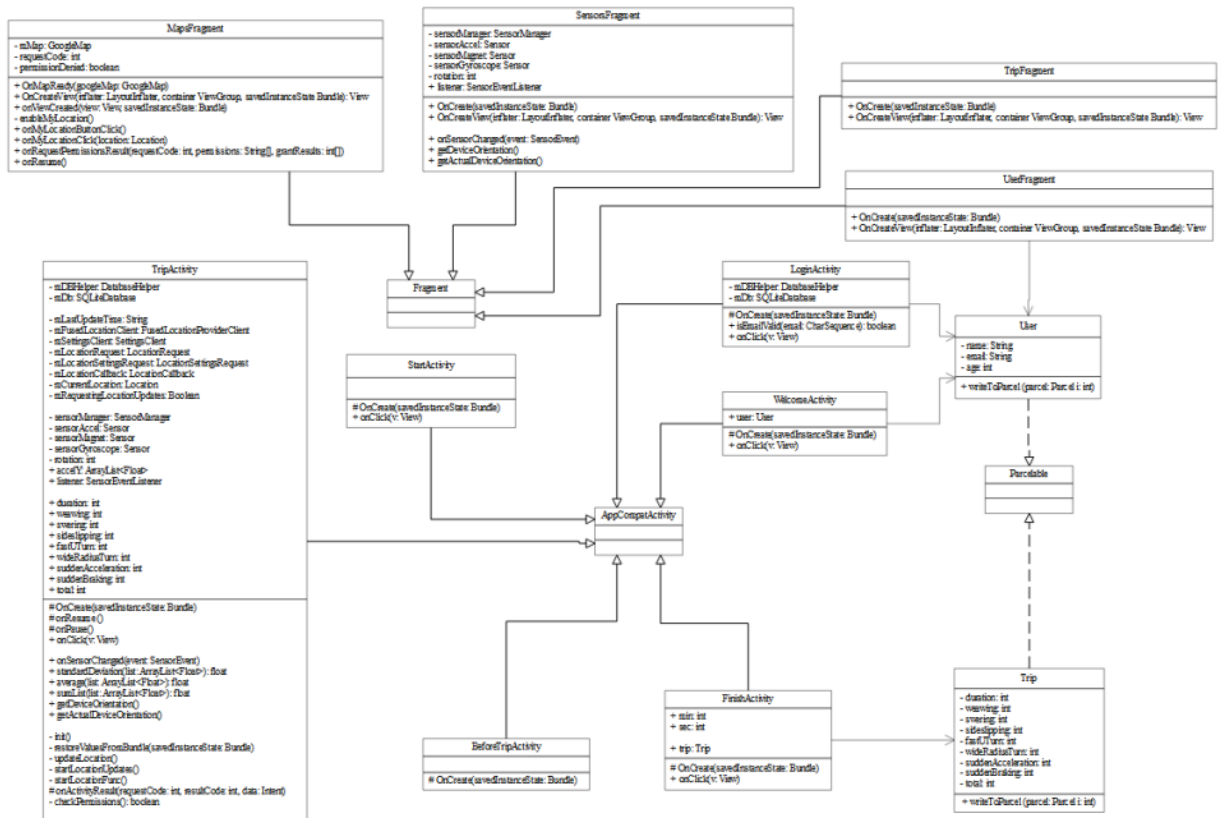


Рисунок 4.6 – Діаграма класів системи

Атрибути та характеристики користувацьких класів представлені у таблиці 4.1.

Таблиця 4.1 – Атрибути та характеристики основних класів

№	Клас	Опис	Функції класу
1	StartActivityWindow	Вхід до додатку та початок взаємодії із ним	При натисканні на відповідну кнопку відбувається перехід до наступного вікна
2	LoginActivityWindow	Екран для введення користувацьких даних	Проведення перевірки правильності введення та збереження наданих даних користувача
3	WelcomeActivityWindow	Інструкція з використання додатку	Вказівки щодо правильного збору інформації про поїздку, підкріплені ілюстраціями для кращого розуміння
4	BeforeTripActivityWindow	Екран із роздільними вкладками	Вкладки відображаються та дозволяють переходити між різними функціональними областями
5	UserFragmentWindow	Вікно, що містить інформацію про користувача	Відображення даних, введених користувачем раніше
6	SensorsFragmentWindow	Вікно, що містить показники сенсорів телефону	Демонстрація та візуалізація показників датчиків

Продовження таблиці 4.1

1	2	3	4
7	MapsFragmentWindow	Вкладка, яка включає в себе Google карту	Відображення місцезнаходження пристрою на карті
8	TripFragmentWindow	Вкладка, яка дозволяє перейти до розділу поїздки	Дозволяє перейти до розділу поїздки при натисканні відповідної кнопки
9	TripActivityWindow	Вкладка, яка дозволяє перейти до розділу поїздки	Перехід до розділу поїздки при натисканні відповідної кнопки
10	FinishActivityWindow	Вікно, в якому представлені результати поїздки	Представлення результатів поїздки та отримання висновків щодо стилю водіння
11	User	Дані про користувача	
12	Trip	Дані про поїздку	

4.6 Діаграма діяльності

Протягом свого експлуатаційного періоду об'єкт може перебувати в різних станах, які взаємодіють між собою та змінюються з плином часу. Наприклад, водій може перебувати у таких станах, як початок руху або зупинка. У кожному конкретному стані об'єкт, яким є водій, проявляє певну поведінку. Для моделювання життєвого циклу об'єкта використовується діаграма діяльності, яка наочно відображає послідовність подій та взаємодій у різних станах. На рисунку 4.5 представлена діаграма діяльності, яка ілюструє взаємодію користувача мобільного застосунку на різних етапах його використання.

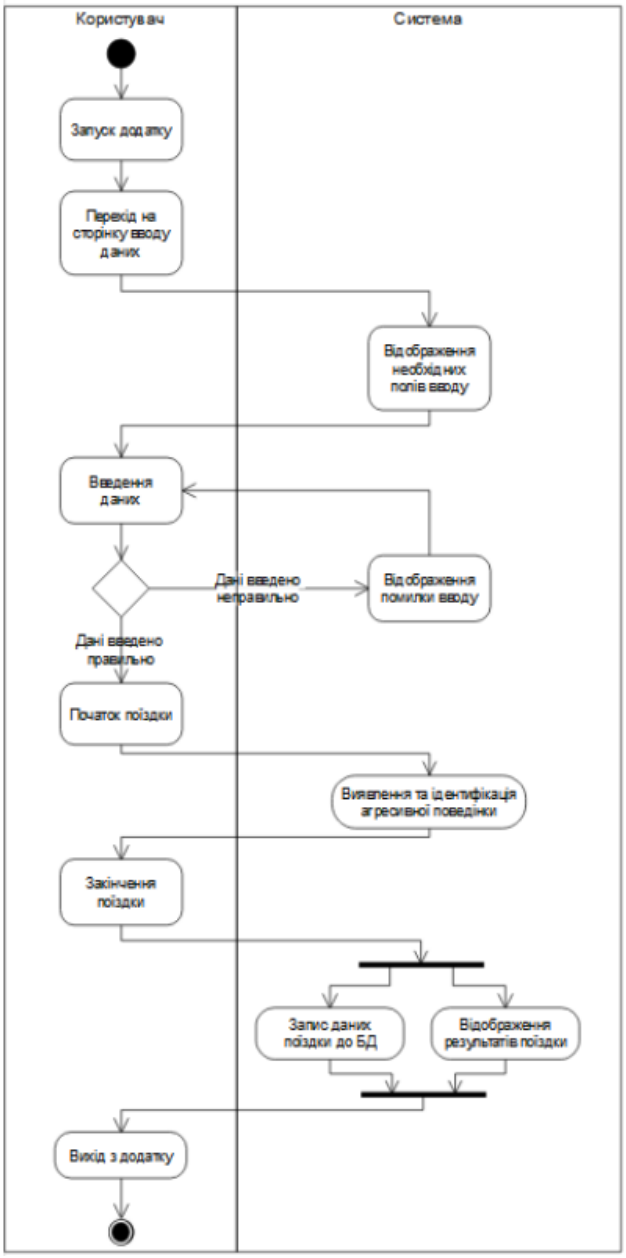


Рисунок 4.5 – Діаграма діяльності

У взаємодії з застосунком користувач виконує певні дії, які визначені системою до початку поїздки. Система пропонує конкретні інструкції, які користувач повинен виконати. У випадку введення некоректних даних система сповіщає користувача і надає можливість внести правильні дані. Цей процес повторюється, поки введені дані не будуть коректними. Лише після цього користувач може перейти до наступних етапів використання додатку, де система проводить аналіз керування транспортним засобом у реальному часі. Після

завершення поїздки водій може оцінити результати та отримати відповідну інформацію.

4.6 Діаграма прецедентів

На рисунку 4.7 діаграма варіантів використання, що надає уявлення про різноманітні можливості та сценарії використання для розробленої системи. Ця діаграма визначає, як різні актори (користувачі) можуть взаємодіяти з системою та які функції та сервіси можуть бути використані в різних ситуаціях.

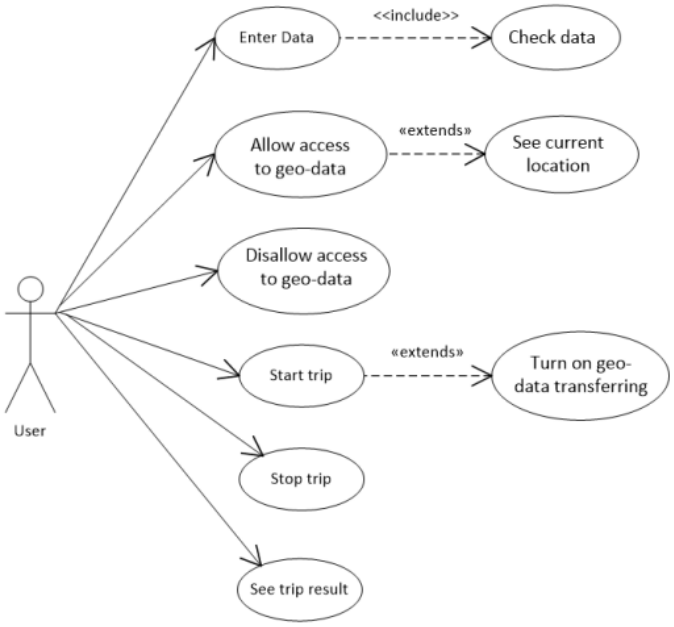


Рисунок 4.7 - Діаграма прецендентів системи

Система взаємодіє з одним актором – користувачем додатку. Під час використання застосунку користувачеві надається можливість вводу необхідних даних, вирішення питання щодо передачі геоданих, ініціація та завершення поїздки, а також перегляд результатів подорожі. У випадку надання дозволу на передачу геоданих, користувач може спостерігати за своїм місцезнаходженням на карті.

4.7 Алгоритм аналізу стану водія за кермом

Безпека на дорозі в значній мірі залежить від поведінки та стану водія під час руху. Для виявлення цієї поведінки використовується пристрій, що включає акселерометр, гіроскоп та магнітометр. Визначення стану водія полягає в фіксації відхилень параметрів датчиків на певне значення. Якщо це відхилення перевищує заданий поріг, то це вказує на аномальну поведінку та стан. Ідентифікація конкретного стану водія під час керування транспортним засобом відбувається на основі змін параметрів по відношенню до відсоткового порогу.

Схема алгоритму для аналізу стану водія під час керування транспортним засобом представлена на рисунку 4.8.

Початок процесу визначення стану розпочинається з натисканням кнопки старту поїздки. Кожні 0,5 секунди показники прискорення, електродермальної активності, швидкості руху, гальмівної реакції та пульсу водія записуються до бази даних застосунку. У цей же інтервал дані вносяться у масив, і розраховується стандартне відхилення цих даних. Якщо стандартне відхилення перевищує встановлене порогове значення, то лічильник конкретного типу для відхилення стану збільшується на одиницю. Під час натискання кнопки завершення поїздки обчислюється загальне значення проявів відхилень стану, і після цього усі дані зберігаються в базі даних та на основі них формуються рекомендації.

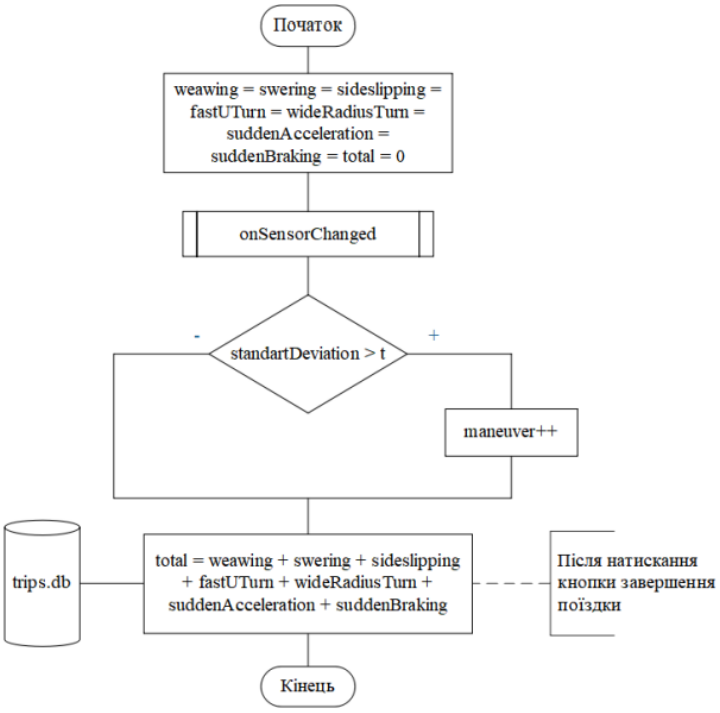


Рисунок 4.8 – Блок-схема аналізу стану водія під час поїздки

Цей підхід до моніторингу стану водія сприяє підвищенню рівня безпеки на дорозі, дозволяючи системі реагувати на потенційно небезпечні ситуації та надавати корисні рекомендації для поліпшення стану водія.

Висновки до розділу 4

Розроблена програмна система для визначення стану водія під час керування транспортним засобом є складною системою, яка включає в себе компоненти та модулі для збору та аналізу даних, відображення інтерфейсу користувача, зберігання інформації в базі даних, а також генерацію рекомендацій для водія.

Основні компоненти системи включають в себе користувацький інтерфейс на платформі React Native, ділову логіку на основі Redux або Flux, модуль зберігання стану додатку, модуль локального сховища, модуль відправки даних до бази даних і саму реляційну базу даних, в якій зберігаються дані користувача, статистика поїздок та рекомендації.

Програма збирає дані про фізіологічний стан водія, включаючи швидкість, пульс, прискорення, гальмівну реакцію та електродермальну активність. Ці дані аналізуються за допомогою алгоритму для визначення стану та стилю водіння водія. Якщо виявляються аномалії, програма генерує рекомендації для водія та може відзначати показники, що виходять за норму, червоним кольором.

Зібрані дані зберігаються в базі даних для подальшого аналізу та використання в контексті безпеки на дорозі. В результаті розробленої програми водієві надається інструмент для моніторингу та покращення його стану під час керування транспортним засобом, сприяючи загальній безпеці на дорозі.

5 РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ

5.1 Системні вимоги

Існують певні системні вимоги для програмного застосунку, який призначений для аналізу стану водія під час керування транспортним засобом. Зокрема, для апаратного забезпечення мінімальні характеристики такі:

- процесора - чотирьохядерний процесор з тактовою частотою не менше 2.0 ГГц); обсягу оперативної пам'яті - 8 ГБ;
- простору на жорсткому диску - 20 ГБ;
- графічна карта повинна підтримувати OpenGL 3.3 або вище.

Для програмного забезпечення встановлені вимоги до операційної системи, такі як сумісність з Windows 10, macOS 10.14 або вище, або Linux версією ядра не нижче 4.0, версії Java (8 або вище), та браузерів для віддаленого доступу.

Вимоги до споживача системи охоплюють базові навички користувача, процедури авторизації та управління доступом. Інтерфейс програми інтуїтивно зрозумілий та зручним для взаємодії.

Окремі вимоги стосуються зовнішніх компонентів, зокрема, сенсорів транспортного засобу для отримання даних про стан водія та можливість їх використання для аналізу поведінки водія.

Загалом, ці вимоги визначають необхідність для стабільної та ефективної роботи програмного застосунку.

5.2 Опис роботи з системою

Основним інструментом взаємодії користувача із системою є мобільний додаток. При відкритті застосунку користувач зустрічає сторінку авторизації, де йому необхідно ввести логін та пароль (рис. 5.1). Якщо користувач не має облікового запису, йому доступна можливість перейти за допомогою кнопки "Sign

Up" та зареєструвати свій обліковий запис. У цьому випадку він повинен ввести ім'я користувача, електронну адресу та обрати пароль (рис. 5.2). Якщо користувач під такою ж електронною адресою вже існує, то реєстрація буде неможлива.

Цей етап реєстрації дозволяє новим користувачам легко приєднатися до системи та отримати доступ до її функціональності. Завдяки цьому інтуїтивному інтерфейсу мобільного застосунку користувачі можуть швидко та зручно користуватися всіма можливостями системи для моніторингу та аналізу їхнього стану під час водіння.

Крім того, реалізована система валідації введених користувачем даних, яка ефективно контролює правильність введення і запобігає некоректним обліковим записам. Наприклад, система реагує на неправильний логін чи пароль, а також на введення занадто простого пароля, що містить менше 6 символів (рис. 5.3). Це допомагає забезпечити безпеку облікових записів та уникнути можливих загроз безпеки.

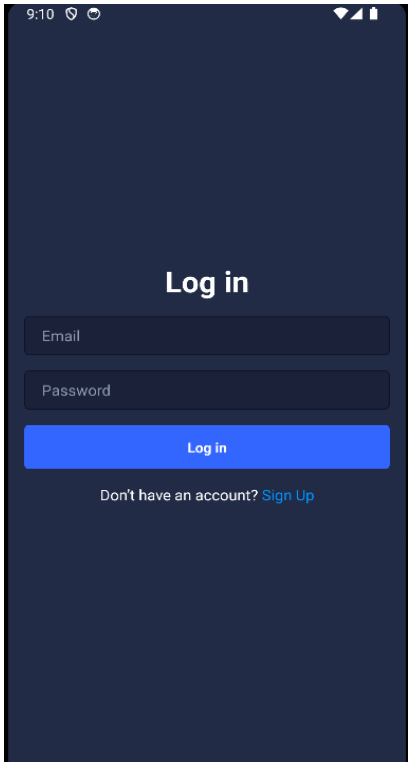


Рисунок 5.1 – Авторизація користувача

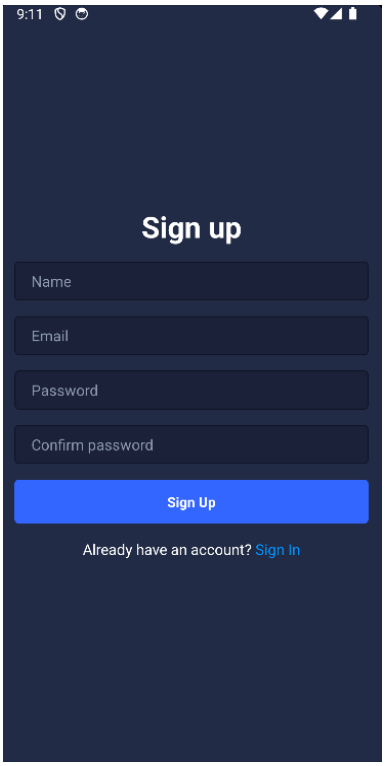


Рисунок 5.2 – Реєстрація користувача

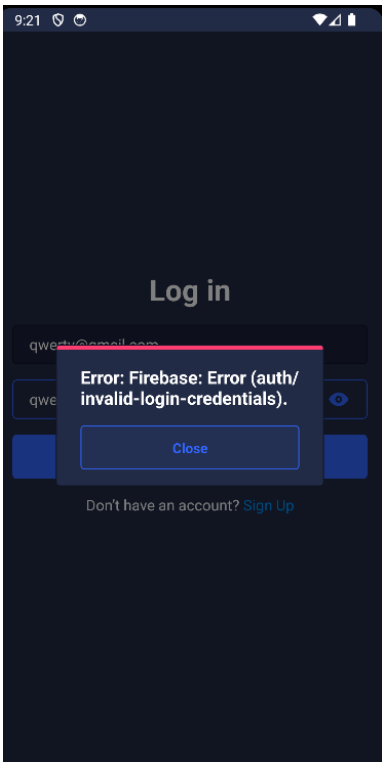


Рисунок 5.3 – Валідація введених даних користувача

Після успішної авторизації користувач потрапляє на сторінку редагування особистої інформації про водія (рис. 5.4). На цій сторінці надається можливість змінити особисті дані, такі як ім'я, дата народження, вага і стать. Ця можливість є важливою, оскільки ці особисті характеристики також впливають на фізичний та психологічний стан водія під час керування транспортним засобом.

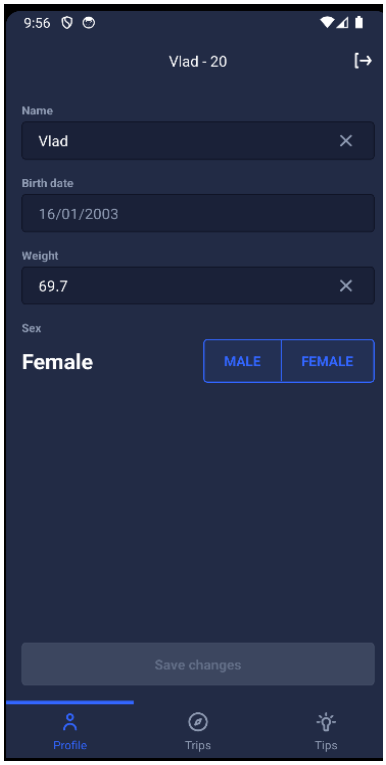


Рисунок 5.4 – Редагування особистих даних про користувача

Крім того, на нижній частині екрану розташована вкладка "Trips", де користувач може переглядати свою повну історію поїздок, а також отримувати інформацію про тривалість та пройдений шлях кожної поїздки. Тут також є можливість створити нову подорож, натискавши кнопку "Add trip" (рис. 5.5). Після цього з'являється вікно для вибору точки відправлення та точки призначення (рис. 5.6). Після введення необхідних даних та натискання кнопки "Confirm", користувач може розпочати нову поїздку.

Ця функціональність не лише забезпечує зручний перегляд історії подорожей, але й дозволяє користувачам легко додавати нові подорожі та вести облік своїх маршрутів через вбудований інтерфейс застосунку.

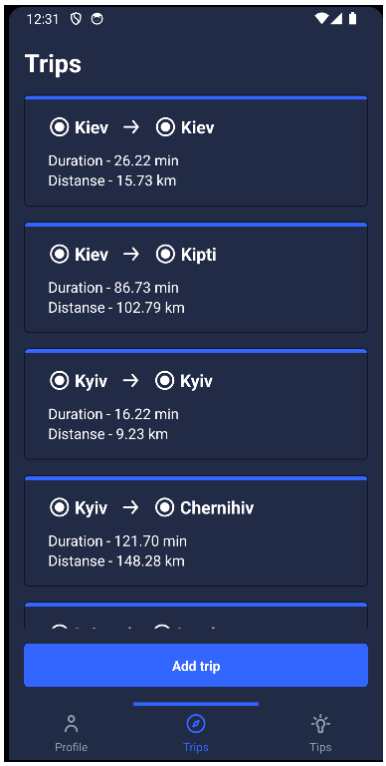


Рисунок 5.5 – Відображення подорожей

У кожній поїздці користувач має можливість детально оглянути маршрут, яким він подорожував. Для цього потрібно натискати на конкретну поїздку та обирати опцію "Show trip". Після вибору, відкривається карта, на якій відображено точну траєкторію руху, а також показники стану водія під час керування транспортним засобом. Серед цих показників входять швидкість руху, пульс водія, електродермальна активність, прискорення та гальмівна реакція (рис. 5.7).

Ця функція надає користувачам можливість глибокого аналізу своїх подорожей, визначення чинників, що впливають на їхню безпеку та комфорт, та сприяє свідомому управлінню власною поведінкою за кермом. Відображення різних показників на карті забезпечує зрозумілу та детальну візуалізацію інформації.

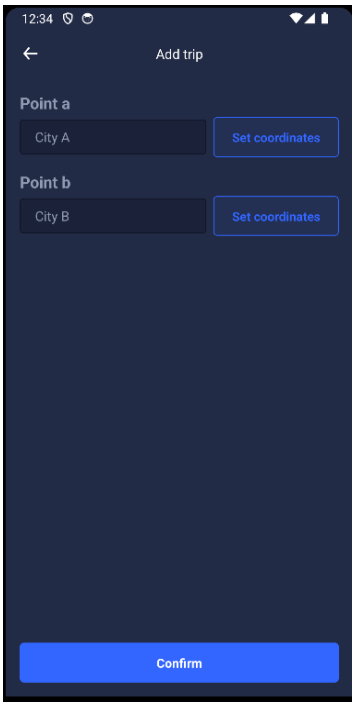


Рисунок 5.6 – Створення нової поїздки

Для кожного показника визначено нормальні значення та відхилення, враховуючи індивідуальні фізіологічні характеристики кожного водія. Якщо значення показника не входить в норму, воно підсвічується червоним кольором, щоб водій міг вчасно виявити можливі аномалії під час водіння. Програма включає такі показники та їхні норми для здорової людини:

- швидкість – 60 - 110 км/год, береться швидкість в конкретний момент часу;
- пульс – 60 - 100 ударів/секунда;
- електродермальна активність (гальванічна шкірна реакція) – 2-30 одиниць;
- прискореннння – (-2) до 3 м/с^2 - зміна швидкості автомобіля за певний час;
- гальмівна реакція. Нормальні показники для гальмівної реакції можуть значно варіюватися в залежності від багатьох факторів, включаючи фізичний стан водія, тип дороги, стан покриття, вид автомобіля,

швидкість руху та інші умови. Однак загальний час гальмівної реакції може бути приблизно в діапазоні від 1,5 до 3 секунд.

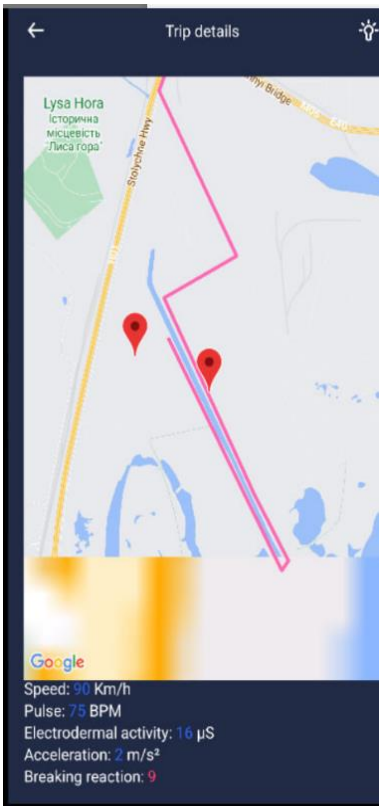


Рисунок 5.7 – Дані про водія

Ці параметри враховуються для адекватної оцінки фізичного стану водія та виявлення будь-яких відхилень, які можуть вплинути на його здатність до безпечного керування автомобілем.

На основі зібраних даних формуються індивідуальні рекомендації для кожної поїздки, які дозволяють водію визначити стан свого здоров'я та стиль водіння. Якщо алгоритм виявляє відхилення від норми, водію надсилається рекомендація звернутися до лікаря (рис. 5.8). Також, на основі аналізу минулих подорожей, користувач має можливість отримувати щоденні загальні рекомендації для покращення свого здоров'я, переходячи на вкладку "Tips" (рис. 5.9).

Ця функціональність спрямована на підтримку та покращення загального здоров'я водія, а також на сприяння формуванню безпечних та свідомих практик водіння. При наявності будь-яких суттєвих аномалій, система рекомендує

консультацію з лікарем, сприяючи дієвому виявленню та реагуванню на можливі проблеми зі здоров'ям водія.

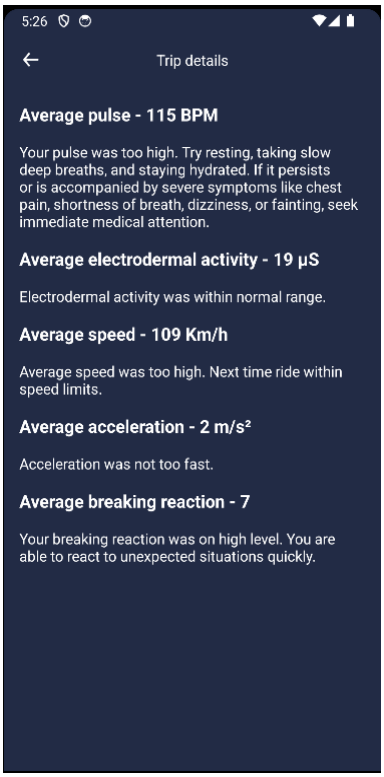


Рисунок 5.8 – Особисті рекомендаціїї

Рисунок 5.9 – Загальні рекомендації

Застосування системи особистих рекомендацій для водіїв має потенціал покращити загальний стан їхнього здоров'я та безпеку на дорозі. Інформація про фізіологічний стан та особливості стилю водіння може служити ключовим фактором для вчасної інтервенції та адаптації водіїв до можливих проблем. Це дозволяє водіям своєчасно виявляти та вирішувати можливі проблеми, а також змінювати свої водійські звички для досягнення кращих результатів в аспектах безпеки та здоров'я.

Аналіз інформації, отриманої з різних поїздок, надає водіям цінні поради та індивідуальні рекомендації, спрямовані на підтримку та покращення їхнього здоров'я. Цей підхід не лише сприяє реагуванню на конкретні проблеми, але й допомагає формувати свідомі та здорові звички водія. В цілому, така система відкриває нові можливості для водіїв у плані особистого розвитку та збереження безпеки на дорозі.

Висновки до розділу 5

Розроблений мобільний застосунок представляє собою комплексний підхід до підвищення безпеки та покращення здоров'я водіїв під час їхньої участі в дорожньому русі. Широкий функціонал застосунку дозволяє користувачам здійснювати авторизацію, реєстрацію та введення особистих даних для подальшого користування. Можливість перегляду історії поїздок надає водіям детальний огляд їхньої активності на дорозі.

Однією з ключових функцій застосунок є аналіз фізіологічних показників водія, таких як швидкість, пульс, прискорення, гальмівна реакція та інші. Цей аналіз дозволяє визначати відповідність цих показників нормі та формувати персоналізовані рекомендації для збереження та покращення здоров'я водіїв. Важливим аспектом є можливість отримання водіями індивідуальних порад на основі аналізу їхньої діяльності на дорозі.

6 РОЗРОБКА СТАРТАП-ПРОЄКТУ

6.1 Опис ідеї проекту

Ідея проекту полягає у розробці програми для визначення стану водія під час керування транспортним засобом. Це відрізняється від інших рішень, оскільки інші схожі програми і системи спрямовані на аналіз типу їзди та виявлення агресивної поведінки на дорозі, запропонований нами програмний застосунок зосереджений на тому, як саме водій почувається під час водіння і чи має він які-небудь ознаки, які можуть вплинути на безпеку на дорозі.

Його переваги включають доступність для широкого кола користувачів, оскільки він повністю безкоштовний. Програма не тільки виявляє потенційні проблеми водіїв, але й надає їм рекомендації щодо подальших дій.

Цей проект вирішує важливу проблему в галузі безпеки на дорогах, а саме - визначення стану водія під час керування транспортним засобом. Він надає цінний внесок з кількох ключових аспектів:

1) Автоматизація оцінки фізичного стану водія: програмне забезпечення використовує сучасні методи аналізу даних, щоб автоматично визначити рівень утоми, стан, швидкість реакції. Це важливо для забезпечення безпеки на дорозі та попередження можливих аварій.
2) Система попередження та реагування: програмний продукт може інтегруватися з системами безпеки автомобілів для надання реального часу попереджень водію та інших учасників дорожнього руху щодо виявлених відхилень показників фізіологічного стану.
3) Покращення безпеки на дорозі та зменшення аварій: Завдяки ранньому виявленню можливих проблем, система сприяє уникненню аварій та покращенню загальної безпеки на дорозі. Це особливо важливо у сучасному світі, де зростає кількість аварій та технологічних впливів на водіїв.

Цей проект не лише визначає нові стандарти в галузі безпеки на дорогах, але й вносить свій внесок у розвиток технологій для покращення якості стану водія та безпеки на дорозі.

Проведений аналіз існуючих аналогів підтверджує, що вони не можуть повноцінно конкурувати із розробленою системою, оскільки вирішують безліч завдань і є складними комплексними продуктами. Розроблений програмний продукт цього проекту має чітку спрямованість - визначення стану водія під час керування транспортним засобом.

Було проведено аналіз концепції проекту, розглянуті можливі сфери застосування та основні переваги, доступні для користувачів при використанні даного продукту. Отримані результати представлені в таблиці 6.1.

Таблиця 6.1 - Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створення програмного забезпечення для визначення стану водія під час керування транспортним засобом.	1. Зростання безпеки на дорогах.	Виявлення ознак утомленості, невідповідності та інших факторів, які можуть впливати на здатність водія ефективно керувати автомобілем.
	2. Уникнення аварій	Попередження можливих аварійних ситуацій через аналіз та інтерпретацію даних про стан водія.

Продовження таблиці 6.1

1	2	3
	3. Поліпшення водійського досвіду	Надання водію інформації щодо його стану та порад для поліпшення зосередженості, реакції та загального комфорту водіння.
	4. Допомога поліції та слідчим органам в розслідуваннях ДТП.	Надання поліції та слідчим інструменту для об'єктивного аналізу аварійних ситуацій та установлення причин виникнення ДТП.
	5. Для судової експертизи.	Допомога судовим органам в установленні обставин аварії та прийнятті обґрунтованих рішень в судових справах.

Продовження таблиці 6.1

1	2	3
	6. Страхові аспекти.	Надання страховим компаніям об'єктивних даних для точного визначення відповідальності учасників аварії та розгляду страхових випадків.
	7. Інтеграція з автомобільними технологіями.	Взаємодія з автомобільними системами для покращення безпеки та ефективності керування.

Для оцінки конкурентоспроможності проекту необхідно проаналізувати техніко-економічні переваги ідеї. Для досягнення цієї мети важливо порівняти розроблену систему з аналогічними продуктами на ринку, визначити переваги та недоліки розглядуваної системи і зосередитися на усуненні слабких сторін. Цей порівняльний аналіз викладено у вигляді таблиці 6.2.

Таблиця 6.2 – Визначення сильних, слабких та нейтральних характеристик

№	Характеристика	Розроблена система	Конкуренти	Результат
1	Точність визначення стану водія	Зосереджена на тому, як саме водій почувається під час водіння і чи має він	Більше спрямовані на аналіз типу їзди та виявлення агресивної	Середня

Продовження таблиці 6.2

1	2	3	4	5
		які-небудь ознаки, які можуть вплинути на безпеку на дорозі, але система нова, потребує удосконалення.	поведінки на дорозі, а отже на високу точність визначення стану водія увага не акцентується.	
2	Швидкість обробки даних	Обробка даних швидка.	Різна у різних рішеннях.	Швидка
3	Зручність в інтеграції з іншими системами	Система розроблена за допомогою нових популярних технологій, тому її буде досить легко інтегрувати з популярними сучасними рішеннями	Залежить від конкретного продукту, але, якщо брати популярні застосунки, то досить легко інтегрувати з іншими системами за рахунок використання популярних технологій. Якщо порівнювати більш старі рішення, то можлива проблема з інтеграцією	Висока

Продовження таблиці 6.2

1	2	3	4	5
4	Можливість адаптації до різних умов	Так, як система нова, то потрібно розширювати можливості її адаптування до різних умов на дорозі	Досить адаптовані	Слабка
5	Вартість розробленої системи	Не висока	Різні у різних рішеннях	Конкурентоспроможна
6	Налагодженість процесів	Так, як це нова система, треба ще налагоджувати багато процесів	Оскільки на ринку давно, більшість процесів вже налагоджено	Низька
7	Створення поїздок в реальному часі	Надає повноцінну можливість створювати поїздки в реальному часі	Надає повноцінну можливість створювати поїздки в реальному часі	Сильна
8	Надання попереджень водію в реальному часі про його стан	Присутні попередження	Загалом попередження присутні	Нейтральна

Продовження таблиці 6.2

1	2	3	4	5
9	Зміна даних користувача	Дані користувача можливо змінити	Дані можна змінити, але не у всіх	Сильна
10	Повна інформація про поїздку та історія поїздок	Інформація та історія присутні	Як правило присутня історія поїздок	Комплексна
12	Загальна функціональність	Присутня необхідна функціональність	Кожна система реалізує конкретні потреби	Нейтральна
13	Ціна підписки	Безкоштовна	В районі від 2 до 20 доларів	Сильна

Ця таблиця надає зіставлення ключових характеристик розробленої системи з конкурентами. Значення в колонці "Результат" показує наскільки система відповідає вимогам користувача та гнучка у своїй реалізації. Оскільки в системі є як сильні, так і слабкі сторони, рекомендується розглядати можливість впровадження стартап-проєкту.

6.2 Технологічний аудит ідеї проєкту

Треба оцінити наявність та досяжність технологій, які можна використовувати для впровадження проєкту та зробити висновки щодо можливостей його реалізації, залежно від технологій, які використовуються. Це проаналізовано в таблиці 6.3.

Таблиця 6.3 – Технологічна відповідність ідеї проєкту

№	Складова проєкту	Технології реалізації	Наявність технологій	Доступність технологій
1	Аналіз поведінки водія	Машинне навчання	Так	Технології доступні, але можливі витрати на обладнання та розробку
2	Визначення динаміки руху	Сенсори руху та прискорення, GPS, акселерометри, електроди, датчики гальмування, гіроскопи	Так	Легко доступні
3	Збір та обробка даних	Фреймворк Flask, СУБД MySQL	Так	Різні відкриті та комерційні рішення доступні
4	Користувацький інтерфейс	React Native, React, бібліотека React Native UI Kitten	Так	Широко використовується, доступні різні бібліотеки та фреймворки
5	Прокладання шляху	Google Maps API, бібліотека React Native Maps та React Native Maps Diections	Так	Популярні сервіси для прокладання шляху доступні

Продовження таблиці 6.3

1	2	3	4	5
6	Доступність	Інтеграція з асистивними технологіями, UI/UX дизайн	Так	Різні інструменти для покращення доступності доступні
7	Надійність	Тестування в реальних умовах, резервне копіювання	Так	Різні методи тестування та забезпечення надійності доступні
8	Залежність від транспортного засобу	Інтеграція з різними типами транспортних засобів	Так	Залежно від конкретного ТЗ та можливостей транспортних засобів
9	Кількість проведених тестувань	Unit-тестування, інтеграційне тестування	Так	Легко доступні

Важливо відзначити, що, розглядаючи технологічну здійсненність ідеї проєкту у контексті аналізу стану водія під час керування транспортним засобом, розроблені технології є ключовими для успішної реалізації. Використання сенсорів руху та прискорення, GPS, акселерометрів, електродів, сенсорів гальмування та гіроскопів надає можливість системі точно визначати різні аспекти поведінки водія та реагувати на них.

Інтеграція з вбудованими системами транспортних засобів сприяє зниженню витрат на обладнання та полегшує процес впровадження. Однак слід враховувати, що окремі технології, такі як системи розваг та інформаційні сервіси, можуть впливати на досвід водія, забезпечуючи ефективне використання часу за кермом.

Більшість технологій доступні безкоштовно, що дозволяє знизити вартість розробки та підтримки проєкту. Хоча існують платні опції, переваги, які приносить впровадження даного проєкту, вигідно переважають над можливими витратами. Таким чином, аналіз стану водія під час керування транспортним засобом, базований на використанні сучасних технологій дає можливість для успішної реалізації стартап-проекту.

6.3 Оцінка потенціалу ринку для впровадження стартап-проєкту

Аналіз ринкових можливостей та загроз визначає ключові напрями для успішного впровадження проєкту на ринку. Цей етап дозволяє ретельно розглянути потенційні можливості, які можна використати для розвитку продукту, а також ідентифікувати можливі загрози, які можуть ускладнити чи перешкодити реалізації проєкту. Враховуючи стан ринкового середовища, розуміння потреб потенційних клієнтів та аналіз конкурентних пропозицій, можна стратегічно спланувати розвиток продукту для максимізації його успішності на ринку.

У таблиці 6.4 визначено потенційні групи клієнтів, їхні характеристики та сформовано орієнтовний перелік вимог до продукту, що є ключовим етапом у розумінні споживачів та адаптації продукту до їхніх потреб.

Таблиця 6.4 – Опис цільової аудиторії стартап-проєкту

№	Потреби, які визначають ринок	Цільова аудиторія (цільові сегменти ринку)	Поведінкові особливості різних цільових груп	Вимоги споживачів до продукту
1	Безпечне керування	Водії різних класів із різним досвідом	Різниці в досвіді можуть	Системи попередження

Продовження таблиці 6.4

1	2	3	4	5
	транспортним засобом		впливати на ступінь впевненості водіїв у власних навичках	аварій, інформація про стан доріг, зручний інтерфейс взаємодії
2	Підвищення комфорту та зручностей водійського досвіду	Водії, які проводять тривалі періоди за кермом	Довгі періоди водіння можуть вимагати більше опцій для розваг та комфорту	Автоматичний режим керування, системи розваг та інформаційні сервіси
3	Ефективне використання часу за кермом	Ділові водії, які часто подорожують	Потреба в продуктивності під час поїздок може бути важливою	Системи навігації, інтеграція з робочими інструментами

Загальною тенденцією є позитивне сприйняття технологічних рішень для аналізу та покращення стану водіїв на ринку, що створює перспективи для успішного впровадження проєкту. Однак важливо враховувати і реагувати на зміни в ринковому середовищі, особливо щодо безпеки на дорозі та використання сучасних транспортних технологій.

Після ідентифікації потенційної аудиторії було проведено аналіз ринкових умов, у тому числі складення таблиць, де визначено фактори, які можуть

ускладнити впровадження проєкту на ринку, а також ті, що сприяють йому (табл. 6.5 та 6.6).

Таблиця 6.5 - Аналіз загроз

№	Фактор загроз	Опис	Можливі наслідки
1	Технічні проблеми	Проблеми зі сенсорами, GPS, або іншими технічними аспектами	Може спричинити недостатню точність або надійність даних
2	Приватність даних	Потенційні побоювання користувачів стосовно конфіденційності особистої інформації	Втрата довіри користувачів, можливі правові проблеми
3	Низька придатність для використання	Складність у використанні системи або несумісність з деякими транспортними засобами	Зменшення популярності серед водіїв та партнерів
4	Конкуренція	Поява інших схожих проєктів або послуг на ринку	Зменшення частки ринку та конфлікти із конкурентами
5	Потенційні правові обмеження	Зміни в законодавстві, пов'язані із збором та обробкою даних	Потенційні штрафи та обмеження в діяльності проєкту

Таблиця 6.6 - Фактори можливостей

№	Фактор	Вплив на ринкове впровадження проєкту
1	Зростання інтересу до безпеки на дорозі	Позитивний - сприяє усвідомленню важливості технологій для аналізу та покращення стану водіїв
2	Розвиток технологій сенсорів та машинного навчання	Позитивний - надає можливість вдосконалювати систему аналізу стану водіїв
3	Збільшення популярності та використання сучасних транспортних засобів	Позитивний - створює потребу в розвитку технологій для покращення безпеки на дорозі
4	Збільшення кількості аварій та нещасних випадків	Негативний - може стимулювати попит на рішення для зменшення аварійності та покращення умов керування
5	Збільшення рівня автоматизації транспорту	Позитивний - створює можливість для інтеграції систем у сучасні автотранспортні засоби

Подальший етап включає в себе огляд пропозицій, де визначалися основні характеристики конкурентної ситуації на ринку (табл. 6.7).

Таблиця 6.7 - Ступеневий аналіз конкуренції на ринку

№	Фактор конкуренції	Рівень впливу	Сильна сторона	Слабка сторона
1	Бренд та репутація	Високий	Наявність власного бренду	Не така висока репутація

Продовження таблиці 6.7

1	2	3	4	5
2	Якість продукту	Високий	Висока якість та надійність	Обмежений функціонал
3	Цінова стратегія	Середній	Конкурентоспроможні ціни	Високі ціни
4	Обсяг продажів	Високий	Велика кількість клієнтів	Обмежений ринок
5	Інновації	Високий	Новаторські рішення	Відсутність інновацій

Після вивчення конкурентного середовища був проведений детальний аналіз умов конкуренції в галузі за методологією М. Портера, в якому враховуються п'ять основних сил, такі як:

1) Загроза нових учасників (низька): У даній галузі високий рівень технологічної складності і великий обсяг необхідних ресурсів створюють значні бар'єри для нових учасників. Також, наявність великих гравців з високим рівнем інвестицій підвищує загрозу для потенційних конкурентів.
2) Доставка продукту чи послуги (середня): У галузі існують різні методи доставки продукту, включаючи розробку мобільних застосунків та інтеграцію з технологічними пристроями в автомобілях. Існує середній рівень стандартів доставки, але деякі компанії можуть виділятися своїми інноваціями в цьому плані.
3) Погроза заміщення продукту чи послуги (висока): З введенням нових технологій і розробкою більш продуктивних рішень існує постійна загроза заміщення. Компанії повинні постійно оновлювати свої продукти, щоб залишатися конкурентоспроможними.

4) Доступ до ресурсів (висока): Для ефективного аналізу та надання рекомендацій важливо мати доступ до великої кількості даних, технічних розробок та кваліфікованих фахівців. Забезпечення цих ресурсів може бути складним завданням.
5) Конкурентна боротьба в галузі (висока): Існує активна конкурентна боротьба між компаніями, оскільки багато з них пропонують подібні технологічні рішення. Зниження цін, покращення функціональності та інші конкурентні стратегії є поширеними в цій галузі.

Аналіз за методологією М. Портера свідчить про те, що галузь стану водія під час керування транспортним засобом є висококонкурентною, і компанії повинні постійно вдосконалювати свої продукти та послуги, забезпечуючи доступ до технологічних розробок та кваліфікованих фахівців.

В таблиці 6.8 здійснено оцінку сильних та слабких сторін стартап-проєкту.

Таблиця 6.8 – Оцінка сильних та слабких сторін стартап-проєкту

№	Фактори	Сильні сторони	Слабкі сторони
1	Технічна інфраструктура	Використання різноманітних сенсорів і сенсорів для збору точних фізіологічних даних	Можливість виникнення технічних неполадок у роботі сенсорів та апаратного забезпечення
2	Візуалізація та інтерфейс	Зручний та інтуїтивно зрозумілий інтерфейс для користувачів	Можливість виникнення проблем із сумісністю інтерфейсу на різних пристроях

Продовження таблиці 6.8

1	2	3	4
3	Аналітика та обробка даних	Високоточний аналіз фізіологічних даних для визначення стану водія	Обмежена можливість обробки великої кількості даних в реальному часі
4	Безпека та конфіденційність	Заходи безпеки для забезпечення конфіденційності особистих даних	Можливість кібератак та порушень конфіденційності
5	Рекомендації для користувача	Система видачі персоналізованих рекомендацій для поліпшення стану водія	Недостатня точність рекомендацій у випадку складних медичних станів

Результати порівняльного аналізу свідчать про те, що розроблений проєкт виявився сильнішим у порівнянні з конкурентами.

Після проведення SWOT-аналізу, в якому були виокремлені сильні (Strength), слабкі (Weak) сторони, загрози (Troubles) та можливості (Opportunities) проєкту, можна сформулювати наступні висновки та визначити можливості для успішного впровадження проєкту. Детальний SWOT-аналіз проєкту для аналізу стану водія під час керування транспортним засобом наведено нижче.

Strengths (Сильні сторони):

- Інноваційні технології: використання сучасних сенсорів та аналізу фізіологічних показників дозволяє отримувати точні та корисні дані про стан водія.
- Система рекомендацій: можливість надавати водіям рекомендації щодо їхнього фізичного стану та безпеки на дорозі.

- Доступність: програмний застосунок може бути доступний для широкого кола користувачів, що сприяє популяризації сервісу.

Weaknesses (Слабкі сторони):

- Достовірність даних: можливість точно визначити емоційний стан водія може бути обмеженою та не завжди точно відображати його фактичний стан.
- Приватність даних: збір і аналіз особистих даних водіїв може викликати проблеми щодо приватності та безпеки інформації.

Opportunities (Можливості):

- Партнерства з автовиробниками: співпраця з автовиробниками для вбудування технології в автомобілі може розширити аудиторію та покращити взаємодію з користувачами.
- Розширення функціоналу: додавання нових функцій, таких як взаємодія з іншими "розумними" системами авто, може зробити продукт більш конкурентоспроможним.

Threats (Загрози):

- Конкуренція на ринку: інші стартапи та великі компанії можуть розробляти аналогічні системи, що створює конкурентну боротьбу.
- Правові обмеження: зміни в законодавстві щодо збору та обробки особистих даних можуть вплинути на діяльність проєкту.
- Технічні виклики: проблеми з точністю та надійністю сенсорів можуть вплинути на якість зібраних даних.

SWOT-аналіз допомагає ідентифікувати потенційні переваги та виклики стартапу, щоб зрозуміти, як оптимізувати його стратегію розвитку.

Висновки до розділу 6

На основі аналізу ринкових можливостей для проєкту, спрямованого на вивчення стану водія під час керування транспортним засобом, було розроблено

концепцію проєкту та визначено його сильні та слабкі сторони. Технічний аудит виявив, що оптимальними для нашої реалізації будуть технології React для веб-інтерфейсу, MySQL для управління базою даних.

Під час аналізу ринкових можливостей ми визначили конкурентоспроможність системи та спрямували її на цільову аудиторію, якою є водії та транспортні компанії. Визначивши базову стратегію, ми обрали стратегію диференціації для надання унікальних можливостей в аналізі стану водіїв. Застосування технологій React і MySQL дозволить нам створити якісний та ефективний веб-інтерфейс, а також забезпечить надійне зберігання даних.

Для успішної реалізації проєкту визначили життєво-важливими якість вхідних даних та постійне вдосконалення функціоналу відповідно до розвитку конкурентів.

ВИСНОВКИ

Під час виконання роботи була проблема небезпечної поведінки водія та поганого стану водія під час поїздки.

1. Проведено аналіз існуючих рішень для визначення стану водія під час керування транспортним засобом. Призначення системи охоплює не лише безпеку на дорозі, але й важливі аспекти, такі як уникнення аварій, поліпшення досвіду водія та допомога в розслідуванні ДТП. Інтеграція з автомобільними технологіями дозволить забезпечити ще більшу ефективність та безпеку управління транспортним засобом;

2. Розроблено підхід для збору даних про стан водія, такі як дані зі сенсорів автомобіля, камер, акселерометрів тощо. Визначення стану водія полягає в фіксації відхилень параметрів датчиків на певне значення. Якщо це відхилення перевищує заданий поріг, то це вказує на аномальну поведінку та стан;

3. Розроблено алгоритм для аналізу даних та визначення стану водія, включаючи оцінку рівня утомленості, ступеня уваги. Ідентифікація конкретного стану водія під час керування транспортним засобом відбувається на основі змін параметрів по відношенню до відсоткового порогу;

4. Розроблено архітектуру програмного застосунку та базу даних для зберігання даних;

5. Розроблено програмне забезпечення для аналізу стану водія під час керування транспортним засобом.

Проведені тести для перевірки точності та надійності розробленого програмного застосунку.

Подальші дослідження можуть бути спрямовані на вдосконалення системи аналізу стану водія під час керування транспортним засобом. Це включає вдосконалення алгоритмів, системи попередження та рекомендацій. Оцінка стану водія може покращити безпеку на дорогах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Firebase Documentation [Електронний ресурс]. Режим доступу до ресурсу: https://firebase.google.com/docs.
2. Nader Dabit. React Native in Action. Manning Publications, 2019. 320 p.
3. Neil Smyth. Firebase Essentials - Android Edition. CreateSpace Independent Publishing Platform, 2017. 534 p.
4. Richard Kho. React Native by Example. Packt Publishing, 2017. 414 p.
5. Eric Masiello, Jacob Friedmann. Mastering React Native. Packt Publishing, 2017. 496 p.
6. Ashok Kumar. Mastering Firebase for Android Development. Packt Publishing, 2018. 394 p.
7. React Native Firebase [Електронний ресурс]. Режим доступу до ресурсу: https://rnfirebase.io/.
8. Carlos R. React Cookbook: Create dynamic web apps with React using Redux, Webpack, Node.js, and GraphQL. Packt Publishing, 2018. 580 p.
9. Gorgon Z. React Explained: Your Step-by-Step Guide to React. Amazon Digital Services LLC, 2019. 305 p.
10. Wieruch R. The Road to learn React: Your journey to master plain yet pragmatic React.js. Amazon Digital Services LLC, 2017. 202 p.
11. Human Behavior Cognition Using Smartphone Sensors / [L. Pei, R. Guinness, R. Chen та ін.]., 2013. 23 p.
12. Jiadi Yu. Sensing Vehicle Conditions for Detecting Driving Behaviors / Jiadi Yu, Yingying Chen, Xiangyu Xu., 2018. 81 p.
13. Fowler M. UML Distilled. Third edition / Martin Fowler., 2018. 178 p.
14. World Health Organisation, “The top ten causes of death.” [Електронний ресурс]. Режим доступу до ресурсу: http://www.who.int/mediacentre/factsheets/fs310/en/.

15. Fowler M. UML Distilled. A Brief Guide to the Standard Object Modeling Language. / M. Fowler //Addison-Wesley, Pearson Education, Inc., 2004. 179 p.
16. Buckett C. Dart in action / C. Buckett //Manning Publications, Shelter Island, NY, 2013. 528 p.
17. Choosing a default language [Електронний ресурс]. Режим доступу до ресурсу: https://confluence.atlassian.com/adminjiraserver071/choosing-a-defaultlanguage-802592304.html.
18. MySql [Електронний ресурс]. Режим доступу до ресурсу: http://www.mysql.ru/docs/man/.
19. Kyle Simpson. You Don't Know JS: ES6 & Beyond. O'Reilly Media, 2015. 278 p.
20. Design Patterns - MVC Pattern - Tutorialspoint [Електронний ресурс]. Режим доступу до ресурсу: https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm.
21. SQL Introduction - W3Schools [Електронний ресурс]. Режим доступу до ресурсу: https://www.diagrams.net/blog/uml-class-diagrams.
22. Eric Elliott. Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries. O'Reilly Media, 2016. 180-250 p.
23. Bill Phillips and Brian Hardy. Android Programming: The Big Nerd Ranch Guide, 1st edition. Big Nerd Ranch Inc., 2013. 221-312 p.
24. Технічна документація мови JavaScript [Електронний ресурс]. Режим доступу до ресурсу: https://uk.javascript.info/.
25. Технічна документація React.js [Електронний ресурс]. Режим доступу до ресурсу: https://devdocs.io/react/.

ДОДАТКИ

ДОДАТОК А Лістінг розробленої програми

```
import { TripDoc } from '@/firebase/types/types';
import { Card, Text } from '@ui-kitten/components';
import { View } from 'react-native';
import ArrowForwardIcon from '../icons/ArrowForwardIcon';
import RadioButtonOnIcon from '../icons/RadioButtonOnIcon';
import { Link } from 'expo-router';

export default function Index({ trip }: { trip: TripDoc }) {
  return (
    <Link asChild href={`/trip/${trip.tripId}`}>
      <Card status="primary">
        <View style={{ gap: 12 }}>
          <View style={{ flexDirection: 'row', gap: 12, alignItems: 'center' }}>
            <View
              style={{ flexDirection: 'row', gap: 4, alignItems: 'center' }}
            >
              <RadioButtonOnIcon
                style={{ width: 24, height: 24 }}
                fill="#ffffff"
              />
              <Text category="h6">{trip?.pointA?.city}</Text>
            </View>
            <View>
              <ArrowForwardIcon
                style={{ width: 24, height: 24 }}
                fill="#ffffff"
```

```
            />
          </View>
          <View
            style={{ flexDirection: 'row', gap: 4, alignItems: 'center' }}
          >
            <RadioButtonOnIcon
              style={{ width: 24, height: 24 }}
              fill="#ffffff"
            />
            <Text category="h6">{trip?.pointB?.city}</Text>
          </View>
        </View>
        {trip.duration && trip.distance && (
          <View>
            <Text category="s1">
              Duration - {trip.duration.toFixed(2)} min
            </Text>
            <Text category="s1">
              Distanse - {trip.distance.toFixed(2)} km
            </Text>
          </View>
        )}
      </View>
    </Card>
  </Link>
 );
}

import React, { useState } from 'react';
```

```
import { TouchableWithoutFeedback } from 'react-native';
import { Icon, IconProps, Input } from '@ui-kitten/components';

interface PasswordInputProps {
  value: string;
  setValue: (value: string) => void;
  placeholder?: string;
}

export default function Index(props: PasswordInputProps) {
  const { value, setValue, placeholder } = props;
  const [hidePassword, setHidePassword] = useState(true);

  const toggleSecureEntry = (): void => {
    setHidePassword(!hidePassword);
  };

  const renderTogglePasswordIcon = (props: IconProps): React.ReactElement => (
    <TouchableWithoutFeedback onPress={toggleSecureEntry}>
      <Icon {...props} name={hidePassword ? 'eye-off' : 'eye'} />
    </TouchableWithoutFeedback>
  );

  return (
    <Input
      value={value}
      placeholder={placeholder}
      accessoryRight={value ? renderTogglePasswordIcon : undefined}
      secureTextEntry={hidePassword}
```

```
      onChangeText={(nextValue) => setValue(nextValue)}
    />
  );
}

Index.defaultProps = {
  placeholder: 'Password',
};

import { Layout } from '@ui-kitten/components';
import { StyleProp, ViewStyle } from 'react-native';

interface PageLayoutProps {
  children: React.ReactNode;
  style?: StyleProp<ViewStyle>;
  level?: '1' | '2' | '3' | '4';
}

export default function Index(props: PageLayoutProps) {
  return (
    <Layout
      level={props.level || '1'}
      style={[props.style, { flex: 1, padding: 16 }]}
    >
      {props.children}
    </Layout>
  );
}
```

```
import { Icon, IconElement, IconProps } from '@ui-kitten/components';

export default function RadioButtonOnIcon(props: IconProps): IconElement {
  return <Icon {...props} name="radio-button-on" />;
}

import { Icon, IconElement, IconProps } from '@ui-kitten/components';

export default function RadioButtonOnIcon(props: IconProps): IconElement {
  return <Icon {...props} name="radio-button-on" />;
}

import { Icon, IconElement, IconProps } from '@ui-kitten/components';

export default function PersonIcon(props: IconProps): IconElement {
  return <Icon {...props} name="person-outline" />;
}

import DismissKeyboard from '@/components/DismissKeyboard';
import PageLayout from '@/components/PageLayout';
import ArrowBackIcon from '@/components/icons/ArrowBackIcon';
import { FIRESTORE_DB } from '@/firebase';
import {
  FIRESTORE_COLLECTIONS,
  FIRESTORE_SUBCOLLECTIONS,
} from '@/firebase/enums/enums';
import { Point, TripDoc } from '@/firebase/types/types';
import useStore from '@/store';
import generateFirestoreId from '@/utils/generateFirestoreId';
```

```
import {
  Button,
  Input,
  Text,
  TopNavigation,
  TopNavigationAction,
} from '@ui-kitten/components';
import { Link, router } from 'expo-router';
import {
  Timestamp,
  addDoc,
  collection,
  serverTimestamp,
} from 'firebase/firestore';
import React, { useCallback, useEffect, useState } from 'react';
import {
  SafeAreaView,
  KeyboardAvoidingView,
  Platform,
  View,
} from 'react-native';

export default function Index() {
  const { user } = useStore((state) => state);
  const { pointA, pointB, clearPointA, clearPointB } = useStore(
    (state) => state
  );
  const [tripDoc, setTripDoc] = useState<TripDoc>({} as TripDoc);
```

```
const addTripDoc = useCallback(async () => {
  const tripId = generateFirestoreId();
  const newTripDoc: TripDoc = {
    ...tripDoc,
    tripId,
    pointA: {
      ...pointA,
      city: tripDoc?.pointA?.city,
    } as Point,
    pointB: {
      ...pointB,
      city: tripDoc?.pointB?.city,
    } as Point,
    createdAt: serverTimestamp() as Timestamp,
  };
  await addDoc(
    collection(
      FIRESTORE_DB,
      FIRESTORE_COLLECTIONS.USERS,
      user?.uid as string,
      FIRESTORE_SUBCOLLECTIONS.USER_TRIPS
    ),
    newTripDoc
  );
  router.back();
}, [pointA, pointB, tripDoc, user?.uid]);

useEffect(() => {
  return () => {
```

```
    clearPointA();
    clearPointB();
  };
  // eslint-disable-next-line react-hooks/exhaustive-deps
}, []);

return (
  <SafeAreaView style={{ flex: 1 }}>
    <DismissKeyboard>
      <KeyboardAvoidingView
        behavior={Platform.OS === 'ios' ? 'padding' : 'height'}
        style={{ flex: 1 }}
      >
        <TopNavigation
          alignment="center"
          title="Add trip"
          accessoryLeft={() => (
            <TopNavigationAction icon={ArrowBackIcon} onPress={router.back} />
          )}
        />
        <PageLayout>
          <View style={{ flex: 1, gap: 16 }}>
            <View style={{ gap: 4, width: '100%' }}>
              <Text category="h6" appearance="hint">
                Point a
              </Text>
              <View
                style={{ flexDirection: 'row', gap: 8, alignItems: 'center' }}
              >
```

```
    <Input
      placeholder="City A"
      style={{ flex: 1 }}
      value={tripDoc?.pointA?.city}
      onChangeText={(text) =>
        setTripDoc((prev) => ({
          ...prev,
          pointA: { ...prev.pointA, city: text },
        }))
      }
    />
    {pointA && <Text>Point A defined</Text>}
    <Link asChild href="/(coordinates_set)/a">
      <Button appearance="outline">Set coordinates</Button>
    </Link>
  </View>
</View>
<View style={{ gap: 4, width: '100%' }}>
  <Text category="h6" appearance="hint">
    Point b
  </Text>
  <View
    style={{ flexDirection: 'row', gap: 8, alignItems: 'center' }}
  >
    <Input
      placeholder="City B"
      style={{ flex: 1 }}
      value={tripDoc?.pointB?.city}
      onChangeText={(text) =>
```

```
                setTripDoc((prev) => ({
                  ...prev,
                  pointB: { ...prev.pointB, city: text },
                }))
              }
            />
            {pointB && <Text>Point B defined</Text>}
            <Link asChild href="/(coordinates_set)/b">
              <Button appearance="outline">Set coordinates</Button>
            </Link>
          </View>
        </View>
      </View>
      <Link asChild href="/(trip_add)">
        <Button onPress={addTripDoc}>Confirm</Button>
      </Link>
    </PageLayout>
  </KeyboardAvoidingView>
 </DismissKeyboard>
</SafeAreaView>
);
}

import DismissKeyboard from '@/components/DismissKeyboard';
import PageLayout from '@/components/PageLayout';
import LogOutIcon from '@/components/icons/LogOutIcon';
import { FIRESTORE_DB } from '@/firebase';
import { FIRESTORE_COLLECTIONS } from '@/firebase/enums/enums';
import { UserDoc } from '@/firebase/types/types';
```

```
import useAuth from '@/hooks/useAuth';
import useStore from '@/store';
import shallowEqual from '@/utils/shallowEqual';
import {
  Button,
  ButtonGroup,
  Datepicker,
  Icon,
  IconProps,
  Input,
  Text,
  TopNavigation,
  TopNavigationAction,
} from '@ui-kitten/components';
import { Timestamp, doc, onSnapshot, setDoc } from 'firebase/firestore';
import React, { useCallback, useEffect, useState } from 'react';
import {
  KeyboardAvoidingView,
  Platform,
  SafeAreaView,
  TouchableWithoutFeedback,
  View,
} from 'react-native';

export default function Profile() {
  const { user } = useStore((state) => state);
  const { signOut } = useAuth();
  const [userDocConst, setUserDocConst] = useState<UserDoc>({} as UserDoc);
  const [userDoc, setUserDoc] = useState<UserDoc>({} as UserDoc);
```

```
const getUserFullYear = () => {
  if (!userDocConst?.birthDate) return 'Birth date not set';

  const date = new Date();
  return date.getFullYear() - userDocConst.birthDate.toDate().getFullYear();
};

const handleUserDocPropertiesChange = useCallback(
  (key: keyof UserDoc, value: (typeof userDoc)[keyof typeof userDoc]) => {
    setUserDoc((prev) => ({ ...prev, [key]: value }) as UserDoc);
  },
  []
);

const saveChanges = useCallback(() => {
  const newUserDoc: UserDoc = {
    ...userDoc,
    weight: Number(userDoc?.weight) || 0,
  };
  setDoc(
    doc(FIRESTORE_DB, FIRESTORE_COLLECTIONS.USERS, user?.uid as string),
    newUserDoc
  );
}, [user?.uid, userDoc]);

const renderClearNameInputIcon = (props: IconProps): React.ReactElement => (
  <TouchableWithoutFeedback
    onPress={() => handleUserDocPropertiesChange('name', '')}
```

```
  >
    <Icon {...props} name="close-outline" />
  </TouchableWithoutFeedback>
);

const renderClearWeightInputIcon = (props: IconProps): React.ReactElement => (
  <TouchableWithoutFeedback
    onPress={() => handleUserDocPropertiesChange('weight', '')}
  >
    <Icon {...props} name="close-outline" />
  </TouchableWithoutFeedback>
);

useEffect(() => {
  const unsubscribeFromUserDoc = onSnapshot(
    doc(FIRESTORE_DB, FIRESTORE_COLLECTIONS.USERS, user?.uid as string),
    (doc) => {
      setUserDoc(doc.data() as UserDoc);
      setUserDocConst(doc.data() as UserDoc);
    },
    (error) => console.log(error)
  );

  return unsubscribeFromUserDoc;
  // eslint-disable-next-line react-hooks/exhaustive-deps
}, []);

return (
  <SafeAreaView style={{ flex: 1 }}>
```

```
<DismissKeyboard>
  <KeyboardAvoidingView
    behavior={Platform.OS === 'ios' ? 'padding' : 'height'}
    style={{ flex: 1 }}
  >
    <TopNavigation
      alignment="center"
      title={`${userDocConst?.name || ''} - ${getUserFullYear()}`}
      accessoryRight={() => (
        <TopNavigationAction icon={LogOutIcon} onPress={signOut} />
      )}
    />
    <PageLayout>
      <View style={{ flex: 1, gap: 16 }}>
        <Input
          label="Name"
          value={userDoc?.name}
          onChangeText={(text) =>
            handleUserDocPropertiesChange('name', text)
          }
          accessoryRight={
            userDoc?.name ? renderClearNameInputIcon : undefined
          }
          maxLength={30}
        />
        <Datepicker
          min={new Date(1900, 1, 1)}
          max={new Date(2024, 1, 1)}
          label="Birth date"
```

```
  date={userDoc?.birthDate?.toDate()}
  onSelect={(date) =>
    handleUserDocPropertiesChange(
      'birthDate',
      Timestamp.fromDate(date)
    )
  }
/>
<Input
  label="Weight"
  value={String(userDoc?.weight || '')}
  onChangeText={(weight) =>
    handleUserDocPropertiesChange('weight', weight)
  }
  keyboardType="decimal-pad"
  placeholder="Not defined"
  accessoryRight={
    userDoc?.weight ? renderClearWeightInputIcon : undefined
  }
/>
<View style={{ gap: 4 }}>
  <Text category="label" appearance="hint">
    Sex
  </Text>
  <View
    style={{
      flexDirection: 'row',
      justifyContent: 'space-between',
      alignItems: 'center',
```

```
      }}
    >
      <Text category="h5">{userDoc.sex}</Text>
      <ButtonGroup appearance="outline">
        <Button
          onPress={() =>
            handleUserDocPropertiesChange('sex', 'Male')
          }
        >
          MALE
        </Button>
        <Button
          onPress={() =>
            handleUserDocPropertiesChange('sex', 'Female')
          }
        >
          FEMALE
        </Button>
      </ButtonGroup>
    </View>
  </View>
</View>
<Button
  disabled={shallowEqual<UserDoc>(userDocConst, userDoc)}
  onPress={saveChanges}
>
  Save changes
</Button>
</PageLayout>
```

```
        </KeyboardAvoidingView>
      </DismissKeyboard>
    </SafeAreaView>
  );
}

import PageLayout from '@/components/PageLayout';
import TripCard from '@/components/TripCard';
import { FIRESTORE_DB } from '@/firebase';
import {
  FIRESTORE_COLLECTIONS,
  FIRESTORE_SUBCOLLECTIONS,
} from '@/firebase/enums/enums';
import { TripDoc } from '@/firebase/types/types';
import useStore from '@/store';
import { Button, Text } from '@ui-kitten/components';
import { Link } from 'expo-router';
import { collection, onSnapshot, orderBy, query } from 'firebase/firestore';
import React, { useEffect, useState } from 'react';
import {
  FlatList,
  KeyboardAvoidingView,
  Platform,
  SafeAreaView,
  View,
} from 'react-native';

export default function Trips() {
  const { user } = useStore((state) => state);
```

```
const [trips, setTrips] = useState<TripDoc[]>([]);

useEffect(() => {
  const unsubscribeFromUserTrips = onSnapshot(
    query(
      collection(
        FIRESTORE_DB,
        FIRESTORE_COLLECTIONS.USERS,
        user?.uid as string,
        FIRESTORE_SUBCOLLECTIONS.USER_TRIPS
      ),
      orderBy('createdAt', 'desc')
    ),
    (data) => {
      setTrips(data.docs.map((doc) => doc.data() as TripDoc));
    }
  );

  return unsubscribeFromUserTrips;
  // eslint-disable-next-line react-hooks/exhaustive-deps
}, []);

return (
  <SafeAreaView style={{ flex: 1 }}>
    <KeyboardAvoidingView
      behavior={Platform.OS === 'ios' ? 'padding' : 'height'}
      style={{ flex: 1 }}
    >
      <PageLayout>
```

```
        <View style={{ flex: 1, gap: 16 }}>
          <Text category="h4">Trips</Text>
          <FlatList
            style={{ flex: 1 }}
            data={trips}
            contentContainerStyle={{ gap: 16 }}
            keyExtractor={(item) => item.tripId}
            renderItem={({ item }) => <TripCard trip={item} />}
          />
          <Link asChild href="/(trip_add)">
            <Button>Add trip</Button>
          </Link>
        </View>
      </PageLayout>
    </KeyboardAvoidingView>
  </SafeAreaView>
 );
}
```

ДОДАТОК Б Презентація

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

ПРОГРАМНИЙ ЗАСТОСУНОК АНАЛІЗУ СТАНУ ВОДІЯ ПІД ЧАС КЕРУВАННЯ ТРАНСПОРТНИМ ЗАСОБОМ

Виконала: студентка групи ТВ-з21мп
Войтих Крістіна Михайлівна

Керівник: к.е.н., доцент

Гусєєва Ірина Ігорівна

АКТУАЛЬНІСТЬ ТЕМИ ДОСЛІДЖЕННЯ

Перевезення пасажирів є дуже важливою частиною багатьох країн, від дій водія залежить безпека всіх учасників дорожнього руху та якість надання послуг щодо перевезення пасажирів. Нині, більше ніж будь-коли раніше, дорожні аварії стають нагальною проблемою, що загрожує життю і безпеці наших доріг. Значна частина цих подій є наслідком недисциплінованості водіїв, недостатньої уваги та реакційності за кермом, а також загальної неуважності.

Тому виникає завдання визначення та дослідження основних фізіологічних показників водія, які можуть адекватно відображати вплив зовнішніх чинників руху на його функціональний стан.

АНАЛІЗ СТАНУ ВОДІЯ ПІД ЧАС КЕРУВАННЯ ТРАНСПОРТНИМ ЗАСОБОМ

Мета: розробка та впровадження програмного застосунку, який дозволить аналізувати стан водія під час керування транспортним засобом з метою підвищення безпеки на дорозі та оптимізації водійської діяльності.

Об'єкт дослідження: процес керування водія транспортним засобом у різних умовах руху.

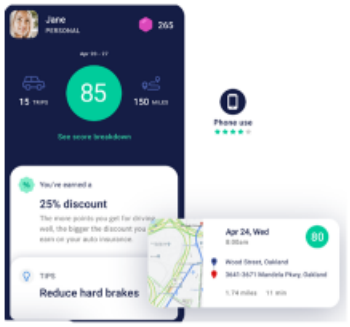
Предмет дослідження: програмне забезпечення для визначення стану водія під час керування транспортним засобом.

АНАЛІЗ СТАНУ ВОДІЯ ПІД ЧАС КЕРУВАННЯ ТРАНСПОРТНИМ ЗАСОБОМ

Часткові завдання, які потрібно розв'язати для досягнення мети:

- провести аналіз існуючих рішень для визначення стану водія під час керування транспортним засобом;
- розробити методи для збору даних про водійську активність та стан, такі як дані зі сенсорів автомобіля, камер, акселерометрів тощо;
- розробити математичні та статистичні методи для аналізу даних та визначення стану водія, включаючи оцінку рівня утомленості, ступеня уваги;
- розробити програмний застосунок, який буде включати в себе алгоритми аналізу та візуалізацію результатів;
- розробити інтерфейс для спілкування з користувачем та виведення результатів аналізу;
- провести тести для перевірки точності та надійності розробленого програмного застосунку.

ІСНУЮЧІ АНАЛОГИ



ZenDrive

НЕОБХІДНИЙ НАБІР ДАНИХ

Швидкість автомобіля
Метод вимірювання: вбудований GPS смартфона

Електродермальна активність (EDA)
Метод вимірювання: спеціальні датчики, електроди, які наносяться на шкіру руки, пальців чи інших ділянках тіла.

Пульс водія
Метод вимірювання: пульсометр, вбудований у смартфон або годинник, окремий медичний прилад для вимірювання пульсу.

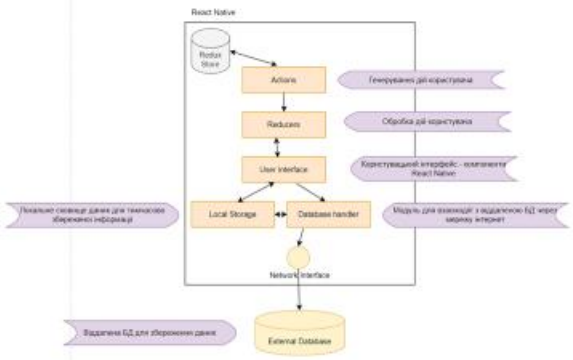
Прискорення
Метод вимірювання: акселерометр, датчики в автомобілях.

Персональні характеристики водіїв

Гальмівна реакція
Метод вимірювання: інерціальні вимірювальні системи, датчики гальмування

Геолокаційні дані GPS

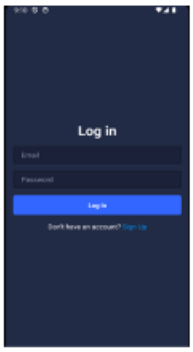
АРХІТЕКТУРА ПРОГРАМНИХ КОМПОНЕНТІВ



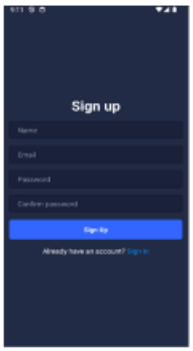
КОНЦЕПТУАЛЬНА МОДЕЛЬ БАЗИ ДАНИХ СИСТЕМИ

ЗАСОБИ РОЗРОБКИ

ІНТЕРФЕЙС ПРОГРАМНОГО ЗАСТОСУНКУ



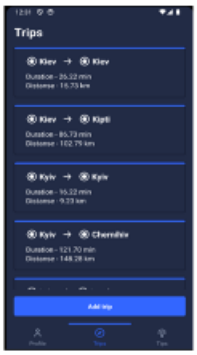
Авторизація



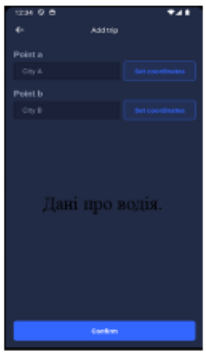
Реєстрація

Дані про користувача

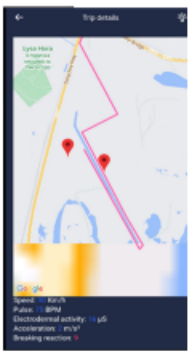
ІНТЕРФЕЙС ПРОГРАМНОГО ЗАСТОСУНКУ



Відображення подорожей

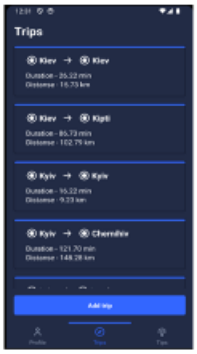


Створення нової поїздки

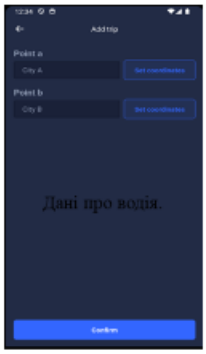


Дані про водія

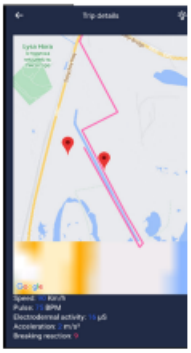
ІНТЕРФЕЙС ПРОГРАМНОГО ЗАСТОСУНКУ



Відображення подорожей



Створення нової поїздки



Дані про водія

ВИСНОВКИ

Часткові завдання, які було розв'язано для досягнення мети:

- проведено аналіз існуючих рішень для визначення стану водія під час керування транспортним засобом;
- досліджені методи для збору даних про водійську активність та стан, такі як дані зі сенсорів автомобіля, камер, акселерометрів тощо;
- розроблені математичні та статистичні методи для аналізу даних та визначення стану водія;
- розроблено програмний застосунок, який включає в себе алгоритми аналізу та візуалізацію результатів;
- розроблено інтерфейс для спілкування з користувачем та виведення результатів аналізу;
- проведено тести для перевірки точності та надійності розробленого програмного застосунку.