

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ПРОДАЖУ АВТІВОК
НА ЄВРОПЕЙСЬКОМУ РИНКУ
MOBILE APPLICATION DEVELOPMENT FOR SELLING CARS IN THE
EUROPEAN MARKET

Виконав: студент(ка) 4 курсу, групи КНТ-129сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Колпаков Д.О.

(прізвище та ініціали)

Керівник Казурова А.Є.

(прізвище та ініціали)

Рецензент Голуб Т.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Колпакова Дмитра Олексійовича
 (прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка мобільного застосунку для продажу автовок на європейському ринку. Mobile Application Development for Selling Cars in the European Market.

керівник проєкту (роботи) Казурова Аліна Євгенівна, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ 27 ” квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) 07 червня 2022 року
 3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2 Вибір та обґрунтування структури системи, яка проектується. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Казурова А.Є., доцент		
Нормоконтроль	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання «04» квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4, 5
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис)

Колпаков Д.О.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Казурова А.Є

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 114 с., 39 рис., 2 табл., 3 дод., 15 джерел.

МОБІЛЬНИЙ ДОДАТОК, ПРОДАЖ, АВТОМОБІЛЬ, АВТОСАЛОН,
REACT NATIVE

Об'єкт дослідження – програмні засоби для зручного продажу.

Предмет дослідження – пріоритети та досвід сучасних торговельних платформ та ринку.

Мета роботи – створення мобільного додатку для автоматизації та поліпшення продажу автомобілів на Європейському ринку з використанням нових технологій для створення зображень.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, технології React Native та Augmented Reality, мови програмування JavaScript, Java, Swift, персональний комп'ютер MacBook Pro 13-inch. 2020 з процесором Apple Silicon M1 під управлінням операційної системи MacOS Monterey.

Результати. Розроблено мобільний кросплатформовий додаток для продажу авто по Європі з технологіями Augmented Reality та Optical character recognition з локалізацією та вибіркою для різних країн.

Висновки. Розроблено мобільний кросплатформовий додаток, який автоматизує, прискорює та спрощує процес продажу та створення оголошень з авто.

Галузь використання – продаж використаних автомобілів та від офіційних дилерів на Європейському ринку.

ABSTRACT

Explanatory note to the bachelor's thesis: 114 pages, 39 figures, 2 tables, 3 appendixes, 15 sources.

MOBILE APP, SALE, CAR, CAR SHOW, REACT NATIVE

The object of research is software for easy sale.

The subject of research is the priorities and experience of modern trading platforms and the market.

The aim of the work is to create a mobile application for automation and improve car sales in the European market using new imaging technologies.

Materials, methods and technical means: structural and object-oriented programming, React Native and Augmented Reality technologies, JavaScript, Java, Swift programming languages, MacBook Pro 13-inch personal computer. 2020 with Apple Silicon M1 processor running MacOS Monterey operating system.

Results. Developed a mobile cross-platform application for car sales in Europe with Augmented Reality and Optical character recognition technologies with localization and sampling for different countries.

Conclusions. Developed a mobile cross-platform application that automates, speeds up and simplifies the process of selling and creating ads from cars.

Field of use – sale of used cars and from official dealers in the European market.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок.....	8
Вступ	9
1 Аналіз предметної області	10
1.1 Основні поняття	10
1.2 Порівняльний аналіз існуючих рішень для продажу автівок	12
1.3 Постановка завдання до роботи	14
1.4 Висновок.....	15
2 Вибір та обґрунтування структури системи, яка проектується	16
2.1 Вибір інструментарію розробки програмного продукту	16
2.2 Структурна схема програмного продукту	19
2.3 Висновок.....	24
3 Основні рішення щодо реалізації компонентів системи.....	25
3.1 Алгоритм завантаження до API через черги.....	25
3.2 Процес поетапної зйомки авто.....	27
3.3 Використання AR	28
3.4 Висновок.....	30
4 Керівництво програміста	31
4.1 Призначення й умови використання.....	31
4.2 Характеристики програмного продукту.....	31
4.3 Звернення до програмного продукту.....	32
4.4 Висновок.....	33
5 Керівництво оператора.....	34
5.1 Призначення й умови виконання програмного продукту	34
5.2 Виконання програмного продукту	34
5.3 Висновок.....	45
Висновки.....	46
Перелік джерел посилання.....	48
Додаток А Технічне завдання.....	49

Додаток Б Лістинги файлів роботи	54
Додаток В Слайди презентації	106

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
IDE	– Integrated Development Environment;
AR	– Augmented Reality;
WEB	– веб;
B3	– веб-застосунок;
IC	– інформаційна система;
ПП	– програмний продукт;

ВСТУП

Інформаційні технології почали бурхливо розвиватися з початком 21 століття, охоплюючи всі галузі людської діяльності: від вивчення космосу до повсякденної роботи. Завдяки обчислювальній потужності електронних пристроїв розвиток як людини, і суспільства прискорюється в геометричній прогресії.

Найпоширенішим сегментом інформаційних технологій є мобільний ринок. З кожним роком зростають не лише нові моделі пристроїв, а й кількість програм. Менш ніж за 20 років людство перейшло від старих, незручних мобільних телефонів до технологій, що стоять поряд із деякими настільними системами. Розміри екрана для виробників більше не є проблемою. Кожне нове покоління смартфонів має ядро, покращений процесор та оперативну пам'ять.

Вибрана була одна з найширших сфер додатків на телефон – продаж, конкретніше, автомобілів. На жаль зараз, особливо на європейському, не так багато застосунків для продажу чи перегляду автівок та мало у яких зручно працювати у ролі продавця.

Для вирішення цієї проблеми було розроблено програму для зручного перегляду списку автомобілів, створення нових авто на продаж та керування їх актуальними даними. Адже набагато простіше відкрити додаток, який тобі підкаже як і з якого ракурсу зробити фото автомобіля, ніж робити це вручну, особливо коли в додатку є не тільки функція вибору ракурсу, але й доповненої реальності.

Розроблений додаток містить в собі лист з фільтрами та сортуванням для пошуку авто, AR, черга фото з ракурсами, адаптивна черга для загрузки фото до API, інструменти для збору інформації про діяльність користувачів для того щоб показувати завжди актуальну і нову інформацію про те що їх цікавить.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття

Проект – це тимчасове підприємство, яке спрямоване на організацію особистої події, метою якої є надання послуг, можливостей, створення унікального продукту та створення умов обміну знаннями та ідеями. Кожен з проектів має свою характеристику, за допомогою якої можна чітко визначити чи є організована подія проектом. Серед основних характеристик виділяють:

- тимчасовість;
- унікальність продукту проекту;
- послідовність організації проекту [1].

Характеристика тимчасовості проекту вказує на те чи має організована подія чіткі дати початку та кінця події. У разі якщо дана подія не має визначеного терміну проведення, то її не можна називати проектом.

Характеристика унікальності продукту проекту вказує на унікальність результату, розробки або дослідження проекту. Інакше таку подію можна назвати «серійним виробництвом» [1].

Характеристика послідовності організації проекту вказує на строгий розвиток та проведення проекту згідно з етапами, що були чітко сформульовані та описані у створених на початковому етапі специфікаціях проекту [1].

Не зважаючи на унікальність продукту кожного з проектів, усі вони мають спільний підхід до реалізації, а саме:

- організовується та виконується чітким складом людей, кожен з яких є представником певних громад та юридичних осіб, що являються ініціаторами, співорганізаторами та спонсорами проекту;
- має чіткі обмеження по ресурсам та фінансуванню, що виділені для реалізації продукту проекту;

– планується, організовується та керується представниками організації-ініціатора, враховуючи ідеї та думки співорганізаторів та спонсорів проекту [1].

Міжнародний проект – це проект, в організації та керуванні якого беруть участі представники з двох чи більш країн і які залучають велику кількість спеціалістів до участі з усього світу.

Автомобіль – самохідна колісна машина, яка приводиться в рух встановленим у неї двигуном і призначена для перевезення людей, вантажу, буксирування транспортних засобів, виконання спеціальних робіт та перевезення спеціального устаткування без рейковими дорогами. Пересувається переважно суходолом.

Мотоцикл – двоколісний механічний транспортний засіб з боковим причепом або без нього, що відрізняється вертикальною посадкою водія та прямим (безредукторним) управлінням переднім рульовим колесом. До мотоциклів прирівнюються класичні двоколісні мотоцикли, моторолери, триколісні (трицикли), чотириколісні (квадроцикли), снігоходи та інші механічні транспортні засоби, дозволена максимальна маса яких не перевищує 400 кг. Окрім кількості коліс, мотоцикли також розрізняються за своєю конструкцією та розмірами: мопеди, мокіки (мають невеликий розмір двигуна, зазвичай до 50 см³), моторолери або скутери (закритий кузовом двигун розташований під сидінням водія та майданчики для ніг), і власне самі мотоцикли різних типів: класичні, крузери, турери, спортивні, шосейні, кросові, ендуро, чопери. Мотоцикл є однією з найбільш доступних форм автотранспорту.

Електромобіль – автомобіль, що приводиться в рух одним або декількома електродвигунами з живленням від акумуляторів (BEV) або паливних елементів тощо, а не двигуном внутрішнього згоряння. Електромобіль слід відрізняти від автомобілів з двигуном внутрішнього згоряння і електричною передачею та від тролейбусів. Підвидами електромобіля вважаються електрокар (транспортний засіб для перевезення

вантажів на закритих територіях) і електробус (автобус з акумуляторною тягою)

Термін електромобільність охоплює повністю електричні транспортні засоби, а також гібридні електричні транспортні засоби та автомобілі, які використовують технологію водневих паливних елементів.

1.2 Порівняльний аналіз існуючих рішень для продажу автівок

1.2.1 Характеристика аналога OLX

OLX – це платформа онлайн-оголошень, яка об'єднує людей для купівлі, продажу або обміну товарами та послугами. Оголошення класифікуються за такими категоріями, як «Дитячий світ», «Нерухомість», «Транспорт», «Запчастини для транспорту», «Робота», «Тварини», «Будинок та сад», «Електроніка», «Бізнес та послуги», «Мода та стиль», «Хоббі, відпочинок та спорт», «Віддам задарма» та «Обмін». Станом на 2018 рік налічується 35 штаб-квартир інтернет-компанії OLX, розташованих у таких містах, як Буенос-Айрес, Київ, Познань, Ташкент, Бухарест, Софія, Пекін, Алма-Ата, Мумбаї та інші. Щомісяця сайти OLX відвідують понад 330 мільйонів людей із 40+ країн світу.

Сайт надає платформу для онлайн купівлі/продажу товарів та послуг як приватним особам, так і представникам бізнесу. Користувачі OLX можуть розміщувати свої оголошення, попередньо зареєструвавшись за допомогою мобільного телефону, електронної пошти або увійти через облікові записи соціальних мереж. У формі подачі оголошення необхідно додати опис пропозиції, фотографії та свої контактні дані. Також є можливість відгукнутися на опубліковане оголошення, зв'язавшись з його автором через онлайн-чат на сайті або за номером телефону.

За умовчанням оголошення на OLX відображаються за датою їх розміщення. Для просування оголошень існує ряд платних послуг: «VIP-

оголошення», «Топ-оголошення», «Підняти вгору списку», «Лого в списку оголошень на 30 днів».

Крім веб-сайтів розроблені програми OLX для платформ iOS та Android. Функціональність програми дозволяє користувачам переглядати свій профіль, додавати оголошення, завантажувати фотографії прямо зі смартфона, писати та відповідати на повідомлення, подзвонити продавцю та інше.

1.2.2 Характеристика аналога Auto.ria

AUTO.RIA – український бренд автомобільної інтернет-торгівлі, один з проєктів RIA.com Marketplaces.

З 2014 року сайт слугує майданчиком для розміщення оголошень від приватних осіб і компаній про купівлю-продаж нових автомобілів, та автомобілів із пробігом. Офіси знаходяться у Таллінні, Вінниці та Києві.

В жовтні 2016 року за даними SimilarWeb, сайт займав 6-е місце за відвідуваністю серед усіх автомобільних сайтів світу в категорії «Купівля авто» і 11-е місце в категорії «Автомобілі та транспорт».

Сайт функціонує як віртуальний торговельний майданчик для автомобілів, де у відкритому доступі розміщуються оголошення про продаж і характеристики автомобілів. Також тут зібрано інформацію про автомобільні компанії України.

1.2.3 Порівняльна характеристика

На сьогоднішній день вже розроблено веб та мобільні сервіси для надання можливості створення, перегляду та продажу автівок. У табл. 1.1 наведено одні з доступних сервісів, котрими користується достатньо багато користувачів.

Таблиця 1.1 – Порівняльна характеристика сервісів з продажу авто

Характеристика	OLX	OTOMOTO	AUTO.RIA
Зручні фільтри	–	+	+
Локалізація інтерфейсу на різні мови	+/-	–	–
Різні зручні можливості для створення фото	–	–	–
Доступно в різних країнах	+	–	–
Моніторинг цін	–	+	+
AR	–	–	–
Мобільна версія	+	+	+

Кожна з вище зазначених платформ веде свою політику доступу до власних інструментів, але жоден з них не надає повних можливостей для роботи с фото. Це надає підставу для створення інструмента, що надає можливість того що можна робити в інших додатках та зручну роботу с фотодокументами.

1.3 Постановка завдання до роботи

В даній роботі досліджується сфера продажу авто та зручність користування додатками націленими на продаж автівок.

Після дослідження потреб користувачів зробив висновки, що завжди є багато мобільних застосунків в яких достатньо функціоналу, але в різних додатках він різний. Тому було вирішено що додаток повинен вміщувати у себе всі можливості своїх аналогів та додатково мати функціонал який ще не існує у конкурентів.

Для виконання поставленої задачі необхідно виконати наступні завдання:

- створити мобільний додаток для всіх платформ;
- створити методи для відстеження дій користувача;

- створити віртуальну чергу для загрузки зображень з хмарними технологіями;
- створити менеджер для фото;
- створити модулі для підтримки доповненої реальності.

1.4 Висновок

В цьому розділі було досліджено предметну область, визначено основні поняття і терміни, пов'язані з предметною областю, дано характеристики застосунків аналогів.

Було проаналізовано основні методи вирішення завдання та наведено порівняльний аналіз цих методів, за результатом якого було виявлено перевагу синтаксичного методу як для розробки, так і для розширення можливостей інструменту.

Проведено порівняльний аналіз додатків, призначених для продажу та перегляду автівок, виявлено переваги та недоліки кожного із них. Було сформульовано і поставлено завдання до робіт.

2. ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЕКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного продукту

Додаток був написаний на мові JavaScript з використанням фреймворка React Native.

Перше що треба для початку розробки – оточення (NodeJs). Node або Node.js – програмна платформа, заснована на движку V8 (трансляючому JavaScript в машинний код), перетворююча JavaScript з вузькоспеціалізованої мови в мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з об'єктами введення-виведення через свій API, написаний на C++, підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з JavaScript-коду. Node.js застосовується переважно на сервері, виконуючи роль веб-сервера, але є можливість розробляти на Node.js і десктопні віконні застосування (за допомогою NW.js, AppJS або Electron для Linux, Windows і macOS) і навіть програмувати мікроконтролери (наприклад, tessel, low.js і espruino). У основі Node.js лежить подієво-орієнтоване і асинхронне (чи реактивне) програмування з неблокуючим введенням/виведенням.

Flutter – це програмний каркас із відкритим кодом для створення додатків для платформ Android та iOS, а також на веб, розроблений компанією Google. Він є основним способом створення додатків для Google Fuchsia. Весь графічний інтерфейс Google Fuchsia створено за допомогою Flutter.

React Native – це фреймворк для створення кроссплатформених застосунків на мові JavaScript. Він дозволяє писати додатки для IOS, Android і навіть VR (на React VR можна створювати додатки для шоломів і окулярів віртуальної реальності").

Творець React Native – Facebook, лідер в front-end, компанія, яка вкладає величезні ресурси в розвиток своїх технологій. Facebook активно

розвиває React і React Native, створює навколо них цілу інфраструктуру і потужне ІТ-співтовариство.

React Native дозволяє створювати мобільні застосування, використовуючи при цьому тільки JavaScript з такою ж структурою, що і у React. Це дає можливість складати багатофункціональний мобільний UI із застосуванням декларативних компонентів.

У табл. 2.1 порівняння двох фреймворків для розробки кросплатформених застосунків.

Таблиця 2.1 – Порівняльна характеристика фреймворків

Параметр	React Native	Flutter
Початковий випуск	Травень 2015	Травень 2017
Розробники	Facebook	Google
Мова	JavaScript	Dart
Продуктивність додатків	Close to native	Fast. 60 fps
IDE підтримка	Range of IDE's and tools with JS support	Android Studio, Visual Studio Code, IntelliJ IDEA
Рідна зовнішність	Lower. Dependency on third-party apps	Better. Has access to device core functionalities
Гаряче перезавантаження	Так	Так
Час виходу на ринок	Повільніше ніж Flutter	Швидше
Повторне використання коду	90% коду багаторазового використання	50-90% коду багаторазового використання
Програми	Tesla, Discord, Instagram	KlasterMe, PostMuse Reflectiy

Хоч React Native більш популярний та в нього більше ком'юніті, він дуже поступає по швидкості розрахунків своєму супернику Flutter та особливо нативним мовам (рис. 2.1).

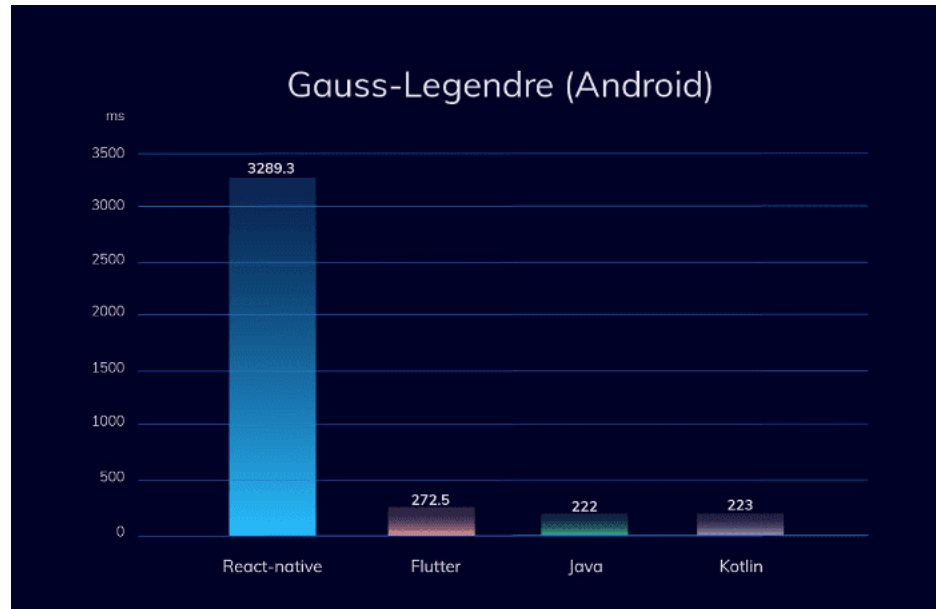


Рисунок 2.1 – Порівняння швидкості розрахунків

Xcode – інтегроване середовище розробки (IDE) програмного забезпечення для платформ macOS, iOS, watchOS і tvOS, розроблена корпорацією Apple. Перша версія випущена в 2003 році. Стабільні версії поширюються безкоштовно через Mac App Store.

Android Studio – інтегроване середовище розробки (IDE) для роботи з платформою Android, анонсована 16 травня 2013 року на конференції Google I/O.

Ця IDE знаходилася у вільному доступі починаючи з версії 0.1, опублікованою в травні 2013, а потім перейшла в стадію бета-тестування, починаючи з версії 0.8, яка була випущена в червні 2014 року. Перша стабільна версія 1.0 була випущена в грудні 2014 року, тоді ж припинилася підтримка плагіну Android Development Tools (ADT) для Eclipse.

Android Studio, заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains, – офіційний засіб розробки Android додатків. Це

середовище розробки доступне для Windows, macOS і GNU/Linux. 17 травня 2017, на щорічній конференції Google I/O, Google анонсував підтримку мови Kotlin, використовуваної в Android Studio, як офіційної мови програмування для платформи Android на додаток до Java і 3++.

2.2 Структурна схема програмного продукту

Файлова система програми має наступну структуру.

У Android знаходяться усі файли і модулі що належать до андроиду, в ios – усі файли і модулі відносяться до ios. У теці src знаходиться увесь javascript код який безпосередньо розбитий на такі розділи: components, assets (json анімації, зображення, шрифти), screens, utils, redux (actions, reducers, sagas), constants. Сам додаток React Native базується на архітектурі мостів (Bridges) між javascript потоком і Native UI потоком.

React Native – це приклад рішення, яке можна назвати "незалежним". В результаті виявляється, що головна мета фреймворка полягає в тому, щоб дозволити розробникам писати JavaScript-код, що використовує механізми React. Завдання React Native полягає в тому, щоб забезпечити перетворення дерева React-компонентів в щось таке, що виявиться працездатним на різних мобільних платформах. Це означає наступне:

- коректне формування інтерфейсу;
- забезпечення доступу до нативних можливостей різних платформ.

За спілкування між двома потоками відповідає так званий міст (bridge) – це ядро React Native. Міст дозволяє потокам спілкуватися найкращим, оптимізованим способом. Він служить посередником, який регулює запити і дані, що входять, від двох потоків. Такий підхід дозволяє їм спілкуватися асинхронно, що призводить до стабільної роботи, тому що потоки ніколи не зможуть заблокувати друг-друга.

У кожному React Native-додатку паралельно виконуються три наступних потоку:

– Metro JS. Це – потік, в якому здійснюється читання і компіляція JavaScript-коду. Тут же виконується основна частина логіки додатка. Бандлер Metro комбінує увесь JS-код в єдиний файл, відповідає за обробку JSX- і TS-конструкцій. Потім отриманий код відправляють движку JavaScriptCore, засобами якого цей код може бути запущений;

– Native Thread. Тут робиться виконання нативного коду. Цей потік взаємодіє з JavaScript-потокком тоді, коли треба оновити інтерфейс або звернутися до нативних функцій. Цей потік можна розділити на дві частини. Перша, Native UI, відповідає за використання нативних засобів формування інтерфейсу. Друга, Native Modules, надає доступ до особливих можливостей платформи, на якій працює додаток. Вона готова до роботи після запуску додатка. Тобто, наприклад, якщо React Native використовує Bluetooth-модуль, він завжди буде активним, навіть у тому випадку, якщо він, насправді, не використовується;

– Shadow Thread. У ньому виконується перерахунок макету додатка. Тут використовується движок Yoga, що є власною розробкою Facebook. У цьому потоці виконуються обчислення, пов'язані з формуванням інтерфейсу додатка. Результати цих обчислень відправляють потоку, відповідальному за виведення інтерфейсу.

Для організації взаємодії між потоками JS і Native використовується модуль Bridge (міст). Цей модуль написаний на C++, в його основі лежить асинхронна черга. Коли міст отримує дані від однієї із сторін, він серіалізує ці дані, перетворюючи в строковий вид, і передає їх через чергу. Коли дані прибувають в пункт призначення, вони десеріалізуються.

Це означає, що усі потоки покладаються на асинхронні JSON-повідомлення, що передаються через міст. Кожна із сторін відправляє ці повідомлення, чекаючи (але не маючи гарантії) того, що колись в майбутньому на ці повідомлення буде отримана відповідь. При такій схемі обміну даними існує ризик перевантаження каналу зв'язку.

Ось приклад, який зазвичай приводять, щоб проілюструвати те, як подібна схема обміну даними здатна викликати проблеми з продуктивністю додатка. Припустимо, що користувач додатка виконує прокрутку величезного списку. Коли в нативному середовищі відбувається подія `onScroll`, інформація асинхронно передається в JavaScript-оточення. Але нативні механізми не чекають до тих пір, поки JavaScript-частина додатка зробить свою справу і їм про це відзвітує. Через це виникає затримка, що виражається в появі в списку, перед виведенням його вмісту, порожнього простору.

Аналогічно, перед тим, як результати перерахунку макету дійдуть до екрану, системі належить виконати декілька сеансів обміну даними. Так, перш ніж дані макету потрапляють в нативне середовище, їх треба обробити в Yoga. А це, зрозуміло, означає, що такі дані теж доводиться передавати через міст.

Як видно, відправка JSON-даних з використанням асинхронних механізмів, а також серіалізація і десеріалізація цих даних, викликають проблеми з продуктивністю. Цей механізм недосконалий.

В ході того, що перепроєктувало архітектури React Native виконується поступова відмова від функцій моста, на зміну яким приходить новий механізм, званий JavaScript Interface (JSI).

Застосування JSI відкриває можливості для деяких чудових поліпшень.

Перше таке поліпшення полягає в тому, що JS-бандл більше не покладається на JSC. Іншими словами, движок JSC тепер легко можна замінити на щось інше, цілком можливо, що відрізняється більш високою продуктивністю. Наприклад – на движок V8.

Друге поліпшення – це те, що лежить в основі нової архітектури React Native. Воно полягає в тому, що, завдяки використанню JSI, в JavaScript можна зберігати посилання на C++-об'єкти (Host Objects) і викликати методи цих об'єктів. В результаті виявляється, що JavaScript – частина додатка і нативні механізми знатимуть один про одного значно більше, чим раніше.

Іншими словами, JSI дозволяє забезпечити повну взаємодію між усіма потоками. Завдяки використанню концепції спільного володіння (shared

ownership), JavaScript-код може запускати нативні методи безпосередньо з JS-потоків. При цьому немає необхідності в тому, щоб використати JSON для передачі даних. Це дозволяє позбавитися від проблем, характерних для використання моста, пов'язаних з можливим переповнюванням черги і з асинхронною передачею даних.

Нова архітектура, дозволяє здійснювати безпосереднє управління нативними модулями. Це означає, що нативні модулі можна використати тоді, тоді вони потрібні, а не ініціалізувати їх все при запуску додатка. Це призводить до серйозного збільшення швидкості запуску додатків.

Цей новий механізм може відкрити дорогу до розробки нових проектів, здатних на те, що було недоступне старим RN-застосуванням. Річ у тому, що тепер в нашому розпорядженні опинилася потужність C++. А це означає, що на базі React Native тепер можна буде створювати значно більше різновидів додатків, чим раніше.

Генератори можуть породжувати (yield) безліч значень одно за іншим, в міру необхідності. Генератори відмінно працюють з перебираними об'єктами і дозволяють легко створювати потоки даних.

Дивлячись з боку кожної окремої платформи ми отримаємо два застосування. Перше на Objective C для iOS і друге на Java для андроїд.

Для стейт менеджменту в своєму додатку я вибрав Redux-Saga.

Redux-Saga – це бібліотека, яка покликана спростити і поліпшити побічні ефекти (тобто такі дії, як асинхронні операції, наприклад, завантаження даних, і "брудні" дії, такі, як доступ до браузерного кешу), зробити легкими в тестуванні і краще справлятися з помилками.

Можна представити це так, що saga – це як окремий потік у вашому застосуванні, який відповідає за побічні ефекти. redux-saga – це мидлвар redux, що означає, що цей потік може запускатися, зупинятися і відмінятися з основного застосування за допомогою звичайних дій redux, воно має доступ до повного стану redux застосування і також може диспатчить дії redux.

Бібліотека використовує концепцію ES6, під назвою генератори, для того, щоб зробити ці асинхронні потоки легкими для читання, написання і тестування. Тим самим, ці асинхронні потоки виглядають, як ваш стандартний синхронний JavaScript-код (накшталт `async/await`, але генератори мають декілька прекрасних можливостей, необхідних нам). Звичайні функції повертають тільки одно-єдине значення(чи нічого).

Redux – бібліотека для JavaScript з відкритим початковим кодом, призначена для управління станом додатка. Найчастіше використовується в зв'язці з React або Angular для розробки клієнтської частини. Містить ряд інструментів, що дозволяють значно спростити передачу даних сховища через контекст.

AsyncStorage – це зашифрована, асинхронна, постійна система зберігання "ключ-значення", глобальна для додатка. Його слід використати замість LocalStorage.

Рекомендується використати абстракцію поверх AsyncStorage замість безпосередньо AsyncStorage для чого-небудь, окрім легкого використання, оскільки вона працює глобально.

У iOS AsyncStorage підтримується власним кодом, який зберігає невеликі значення в серіалізованому словнику, а великі значення – в окремих файлах. У Android AsyncStorage використовуватиме або RocksDB, або SQLite залежно від того, що доступно.

Код JavaScript AsyncStorage – це фасад, який надає зрозумілий API JavaScript, реальні об'єкти помилок і немульті-функції. Кожен метод в API повертає об'єкт Promise.

Усі компоненти і екрани додатка побудовані за принципом MVC.

Для двосторонньої взаємодії потрібні динамічні сторінки. MVC – ключ до розуміння розробки динамічних застосувань, тому розробникові треба знати цю модель.

MVC розшифровується як модель-представлення-контролер (від англ. model-view-controller). Це спосіб організації коду, який припускає виділення

блоків, що відповідають за рішення різних завдань. Один блок відповідає за ці застосування, інший відповідає за зовнішній вигляд, а третій контролює роботу додатка. Компоненти MVC:

- модель – цей компонент відповідає за дані, а також визначає структуру додатка. Наприклад, якщо ви створюєте To-Do додаток, код компонента `model` визначатиме список завдань і окремі завдання;

- подання – цей компонент відповідає за взаємодію з користувачем. Тобто код компонента `view` визначає зовнішній вигляд додатка і способи його використання;

- контролер – цей компонент відповідає за зв'язок між `model` і `view`. Код компонента `controller` визначає, як сайт реагує на дії користувача. По суті, це мозок MVC-дodatка.

2.3 Висновок

В результаті виконання роботи по даному розділу було визначено чітку структуру та описано модулі, що повинні бути розроблені для реалізації автоматизованого мобільного додатку. Підібрано необхідний інструментарій, описано принципи їх роботи та процедури, що вони виконують для реалізації поставленої задачі.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Алгоритм завантаження до API через черги

Черги повідомлень (Message Queue) рис. 3.1 – форма асинхронної комунікації між сервісами. Тому, перш ніж говорити про них, покажемо на спрощеному, трохи штучному прикладі різницю між синхронною та асинхронною взаємодією.

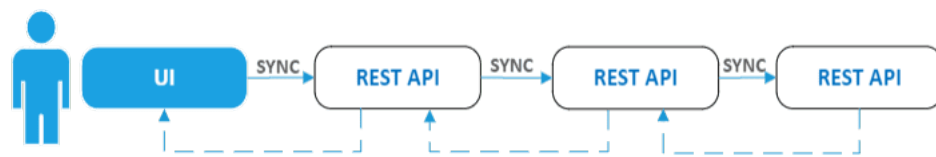


Рисунок 3.1 – Черги повідомлень

Припустимо, ви розробляєте сайт книгарні і у вас є сервіс, до якого звертається користувач, наприклад, відправка рецензії на прочитану книгу. При натисканні кнопки «Надіслати» викликається деякий API, який, у свою чергу, може звернутися до інших API.

При синхронній взаємодії всі запити в цьому ланцюжку викликів виконуються строго один за одним, а при виконанні останнього запиту відповіді передаються у зворотному напрямку. У результаті користувач змушений кілька секунд чекати повідомлення про публікацію свого відгуку, хоча його не цікавлять особливості серверної обробки і він цілком обґрунтовано хоче побачити повідомлення відразу після натискання кнопки. Звичайно, час очікування багато в чому визначатиметься потужністю обладнання, але при пікових навантаженнях воно може стати серйозною проблемою.

Ще один недолік такої схеми – обробка збоїв. Якщо на одному з кроків виникне виняток, воно повернеться назад, і користувач отримає повідомлення

про помилку з проханням повторно надіслати рецензію. Навряд чи когось потішить отримання подібного повідомлення після тривалого очікування.

Описану схему можна змінити, додавши асинхронні дзвінки. Достатньо викликати в асинхронному режимі перший REST API і паралельно повернути користувачеві повідомлення про те, що його рецензія прийнята та буде розміщена, наприклад, протягом доби. Зрештою сайт не блокується, а виклики всіх наступних API відбуваються незалежно від користувача.

Але така схема також має істотний недолік (рис. 3.2): у разі збою в одному з API інформація, введена користувачем, буде втрачена. Якщо у першому прикладі у разі помилок достатньо повторно надіслати рецензію, тут її необхідно заповнити знову.

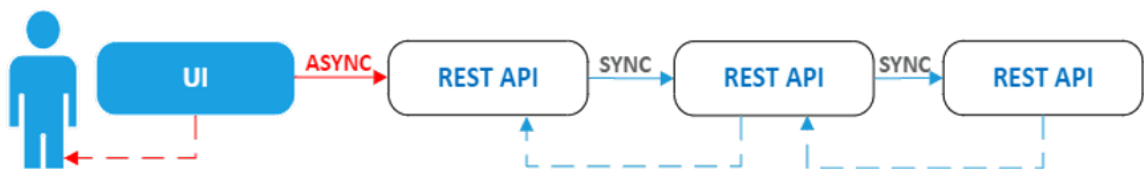


Рисунок 3.2 – Помилки черг

Для користувача у додатку було використано цю архітектуру щоб при одночасному завантаженні 30 - 40 фото, що є можливим через AR, користувач не помічав ніякої різниці у взаємодії з фото та додатком. Таким чином користувач зможе замінити ці фото, редагувати їх порядок, типи та переглядати без ніякого блокування інтерфейсу. Також це розумна черга котра буде автоматично помічати якщо будуть якісь помилки приходити від API, наприклад автоматично додавати поля яких не вистачає, додавати у кінець черги фото які були з помилками та після успішного завершення наступних ітерацій повертатися до незавершених запитів, виправляти та надсилати повторно. Тому користувач будет приємно здивований, що йому треба лише слідувати підказкам у додатку, як краще та вірно робити фото, а про все інше попілкується система.

3.2 Процес пошагової зйомки авто

У додатку реалізована одна з основних її можливостей, пошагова зйомка автівки.

Для легкої реалізації цього функціоналу були використано Figma – для макетів та акселерометр.

Figma – це векторний онлайн-сервіс розробки інтерфейсів та прототипування з можливістю організації спільної роботи, що розробляється однойменною компанією. Працює у двох форматах: у браузері та як клієнтський додаток на десктопі користувача. Зберігає онлайн-версії файлів, з якими працював користувач.

Акселерометр. За допомогою його вимірюють прискорення, що здається. Тобто, якщо говорити про акселерометр в телефоні, то його обов'язки входить допомога ПЗ у визначенні положення пристрою в просторі. Трапляється, що цей пристрій часто плутають з гіроскопом, але це не так, ці прилади чудово доповнюють один одного і в окремих випадках можуть бути взаємозамінними. Головна їхня відмінність полягає в принципі роботи, спільне їх використання дозволяє досягти найточніших вимірів. Синонімом акселерометра у мобільному пристрої є G-сенсор.

В додатку є багато ракурсів авто. Наприклад: вид спереду, передні та задні колеса, під лівим кутом, вид спереду під правим кутом, вид сзади, тощо.

Для реалізації цього функціоналу першим чином було прийнято рішення створити макети у Figma під кожний ракурс, з поясненням де що повинно знаходитись и модуль для перевірки чи вірно розміщення об'єктів у кадрі. Також був розроблен модуль для відстежування нахилу телефона, якщо телефон сильно нахилений ви не зможете зробити фото.

На android ці файли зберігаються у форматі xml під кожну мову. На iOS – strings.

3.3 Використання AR

ARCore – це інструмент для розробки програмного забезпечення, розроблений Google, який дозволяє створювати додатки доповненої реальності.

ARCore використовує три ключові технології для "впровадження" віртуального контенту в реальне середовище.

Відстеження руху: воно дозволяє смартфону зрозуміти своє становище у реальному світі.

Розуміння навколишнього середовища: воно дозволяє смартфону визначати розмір та розташування всіх типів поверхонь (вертикальних, горизонтальних та кутових).

Оцінка освітленості: це дозволяє смартфон оцінити поточні умови освітлення навколишнього середовища.

Коли ви переміщаєте свій телефон, ARCore запам'ятовує оточення і будує власний світ, де він може розміщувати віртуальні об'єкти. Він також використовує технологію відстеження руху для визначення того, як деякі об'єкти рухаються з огляду на рухи вашої камери.

Ось чому, коли ви розміщуєте віртуальний об'єкт у своїй кімнаті, виходите та повертаєтеся, об'єкт все ще там. ARCore побудував свій віртуальний світ, де він пам'ятає, де знаходиться кожна деталь.

Коли ви розміщуєте в цьому просторі об'єкт, ARCore обчислює його положення по відношенню до інших об'єктів, а коли ви повертаєтеся до того ж місця, об'єкт з'являється знову.

Для реалізації AR було обрано технологію OpenGL.

GLSL містить ті самі оператори, що і оператори в C та C ++, за винятком покажчики. Побітові оператори були додані у версії 1.30.

Шейдери GLSL не є окремими програмами; їм потрібна програма, яка використовує API OpenGL, який доступний на багатьох різних платформах

(наприклад, Linux, macOS, Windows). Існують мовні прив'язки C., C ++, C #, JavaScript, Дельфи, Java і багато іншого.

Самі шейдери GLSL – це просто набір струни які передаються драйверу постачальника обладнання для компіляції з програми за допомогою точок входу API OpenGL. Можна створювати шейдери на льоту зсередини програми або зчитування як текстові файли, але має бути надіслано драйверу у вигляді рядка.

Набір API, що використовуються для компіляції, зв'язування та передачі параметрів програмам GLSL, вказані в трьох розширеннях OpenGL і стали частиною основного OpenGL станом на OpenGL версії 2.0. API розширено за допомогою геометричних шейдерів у OpenGL 3.2, тесселяційних шейдерів у OpenGL 4.0 та обчислюють шейдери у OpenGL 4.3. Ці API OpenGL містяться в розширеннях:

- ARB вершинний шейдер;
- Шейдер фрагмента ARB;
- Об'єкти шейдера ARB;
- ARB геометричний шейдер 4;
- ARB тесселяційний шейдер;
- ARB обчислювальний шейдер.

Також можна використовувати шейдери GLSL Вулкан, і є поширеним способом використання шейдерів у Vulkan.

Зараз ARCore SDK доступний для:

- Android;
- Android NDK;
- Unity для Android;
- Unity для iOS;
- iOS;
- Unreal.

3.4 Висновок

В даному розділі було розроблено схему алгоритму виконання програми, розглянуто та описано основні рішення та процеси, що були створені для вирішення задачі.

Було описано всі етапи існування даних з моменту їх вводу, існування, обробки на всіх етапах, та до моменту як вони потрапляють до API та користувача.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови використання

Розроблений додаток призначений для вирішення проблеми продажу, перегляду та управління автівками на ринку та зручним використанням фото менеджера.

Для роботи з розробленим додатком система повинна відповідати мінімальним системним вимогам:

- операційна система macOS;
- чотирьох ядерний центральний процесор;
- 8 ГБ оперативної пам'яті;
- 10 ГБ вільного місця на жорсткому диску;
- доступ до мережі Інтернет;
- наявність останніх версій Xcode та Android Studio.

4.2 Характеристики програмного продукту

Розроблюваний ПП складається з наступних модулів:

- авторизація. Цей модуль відповідає за авторизацію користувача. У кожного користувача і його компанії до котрої він належить є свій регіон. При авторизації обраний регіон повинен співпадати з дійсним. Також є можливість вибору мови. Для зберігання даних використовується KeyChain;
- модуль навігації. У цьому модулі представлені всі екрани котрі є у додатку, їх параметри та властивості;
- модуль черги. Він відповідає за всі процеси та допоміжні модулі які потрібні для черг. У ньому можна знайти для повторного використання модуля для відстеження стану додатку (відкритий, закритий), стан підключення до інтернету та тип підключення;
- модуль камери. Це звичайний модуль камери с додатковим функціоналом написаний на Java з інтерфейсом для JavaScript;

- модуль AR. Модуль на Java з використанням OpenGL для відображення віртуальних об'єктів;
- центральний API модуль. Обертка над звичайними запитами до API для більш зручного використання та автоматизації щодо заміни токенів та обробки помилок;
- модуль повідомлень. Розроблений базовими механізмами з власними анімаціями;
- модуль стеження. Цей модуль відповідає за збір інформації про дії користувача та їх аналіз.

4.3 Звернення до програмного продукту

Для звернення до розроблюваної версії ВЗ необхідно мати налагоджений інструмент для взаємодії з сервісом GitHub та скопувати у необхідну директорію репозиторій з сервера зазначеного сервісу.

Треба налагодити React Native до вашої системи командами:

- <https://apps.apple.com/ru/app/xcode/id497799835?mt=12>;
- <https://reactnative.dev/docs/environment-setup>.

Далі сконфігурувати сертифікати та провіжн на сайті “<http://developer.apple.com>”. Перейти в директорію проекту та запустити “yarn install”. Після установки залежностей потрібно доставити Pods в ios директиві командою “pod install” або “pode –repo-update”.

Після цього потрібно налаштувати сервіс для забезпечення перекладів цими командами:

- yarn run poeditor-rn;
- yarn run poeditor-android.

Потрібно зайти до директиви перекладів у src/i18n та запустити команду для актуалізації локального словника “curl-X POST <https://api.poeditor.com/v2/projects/upload>-Fapi_token=<токен>-Fid=<проект

```
>"-Fupdating='terms_translations'-Flanguage='en'-Ffile=@'en.json'-Foverwrite='0'-  
F sync_terms='0'-F tags=["RN"]”.
```

4.4 Висновок

В даному розділі було проведено опис призначення та визначено умови використання розробленого додатку, визначено мінімальні системні вимоги для функціонування додатку на комп'ютері. Було надано інструкції, за якими визначається можливості доступу до проекту. Було описано як без перешкод встановити за розгорнути проект.

5 КЕРІВНИЦТВО ОПЕРАТОР

5.1 Призначення й умови виконання програмного продукту

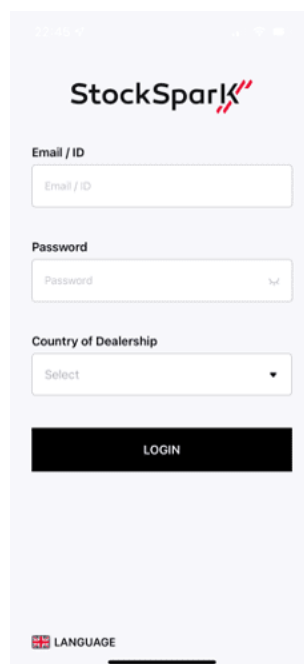
Для роботи з розробленим додатком система повинна відповідати мінімальним системним вимогам:

- операційна система Android чи iOS;
- версія Android не нижче 6 та версія iOS не нижче 11;
- підтримка AR опціонально;
- 50 Мб вільного місця на мобільному;
- доступ до мережі Інтернет.

5.2 Виконання програмного продукту

Після завантаження додатку користувач побачить сплеш скрін під час якого будуть завантажуватись основні дані.

Після завантаження даних користувач почне роботу з додатком. Спершу буде сторінка авторизації, зображена на рис. 5.1.



StockSpark

Email / ID

Email / ID

Password

Password

Country of Dealership

Select

LOGIN

LANGUAGE

Рисунок 5.1 – Авторизація

Після натискання на Логін користувачу треба погодитись з політикою компанії, зображено на рис. 5.2.

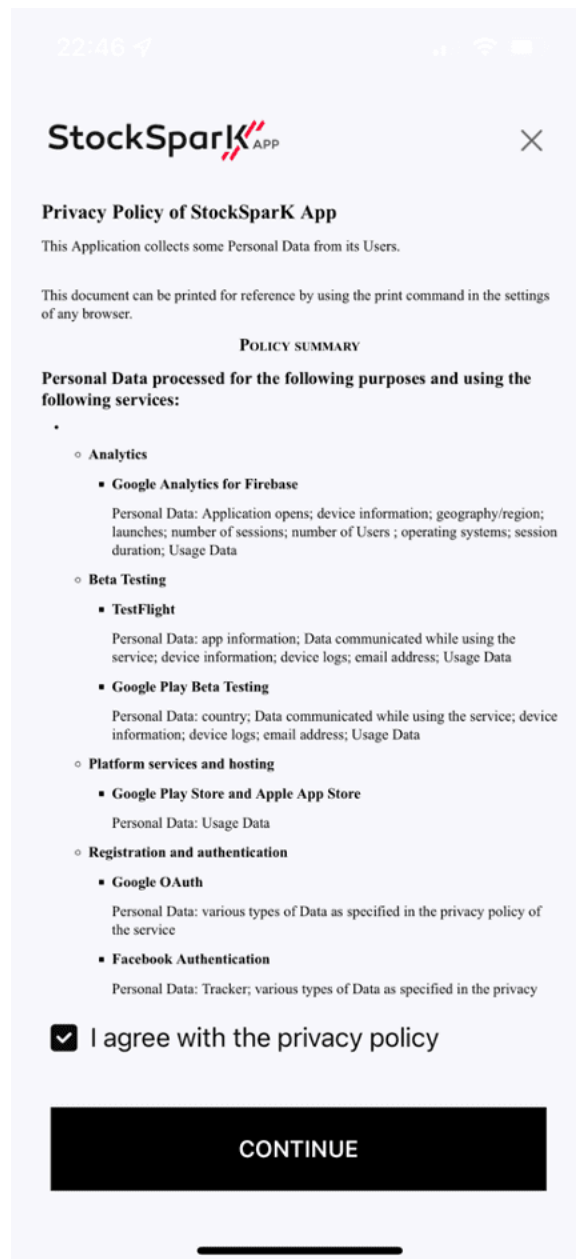


Рисунок 5.2 – Політика компанії

Після вдалої авторизації користувач буде перенаправлен на головну сторінку на якій є головна навігація та список автівок відсортованих за замовчуванням, зображену на рис. 5.3.

По всьому додатку справа зверху можна знайти кнопку яка дозволяє відкрити бокове меню, зображено на рис. 5.4.

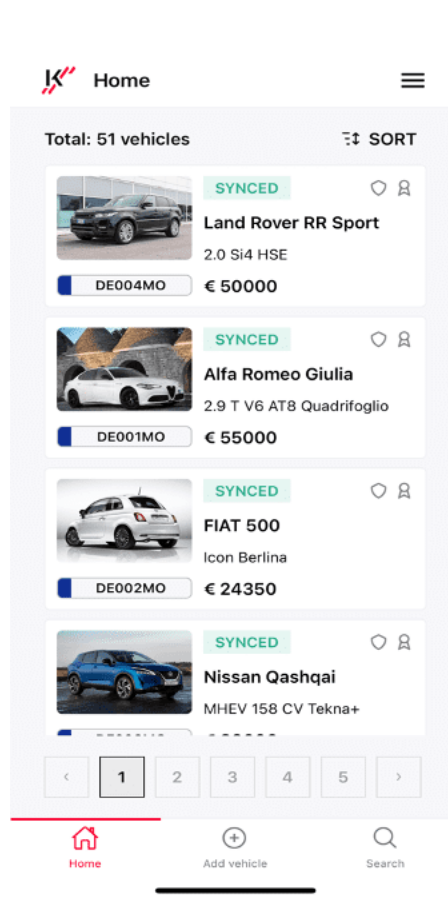


Рисунок 5.3

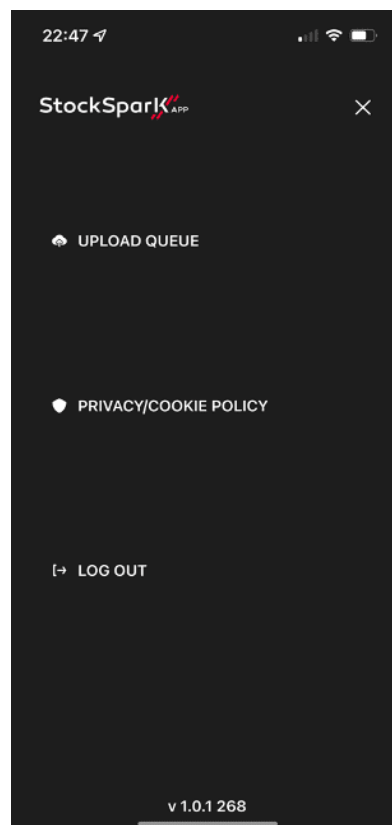


Рисунок 5.4 – Меню

Після відкриття транспорту ми бачимо головну інформацію про транспорт, зображено на рис. 5.5 та рис. 5.6.

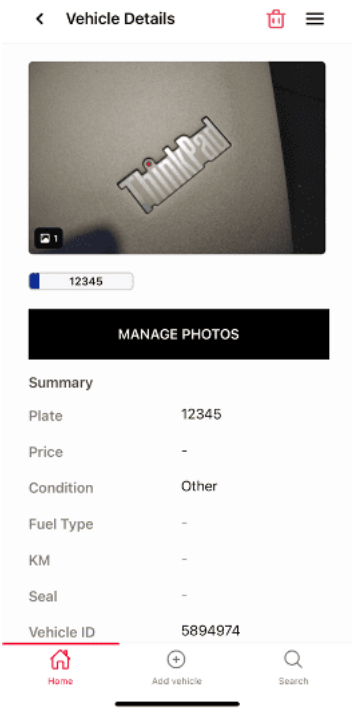


Рисунок 5.5 – Головна сторінка

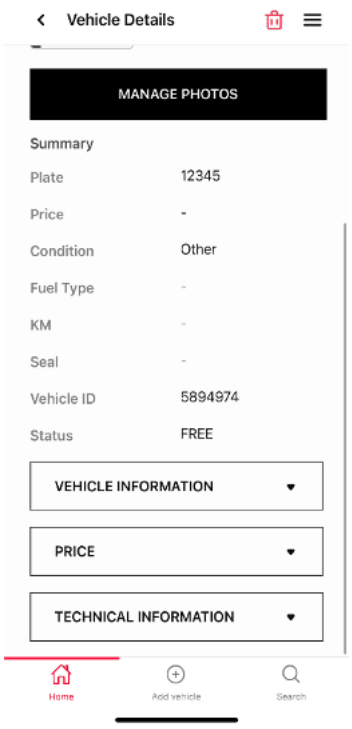


Рисунок 5.6 – Додаткова інформація

Далі у нас іде галерея з усіма можливими функціями (рис. 5.7, рис. 5.8, рис. 5.9).

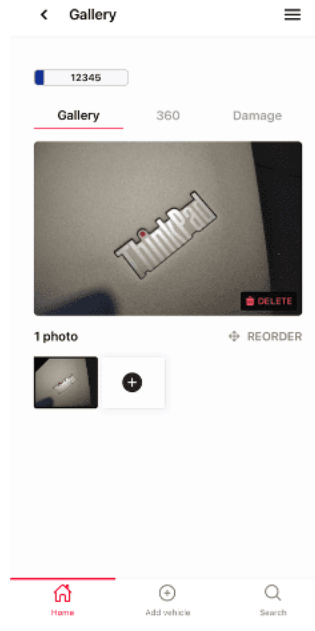


Рисунок 5.7

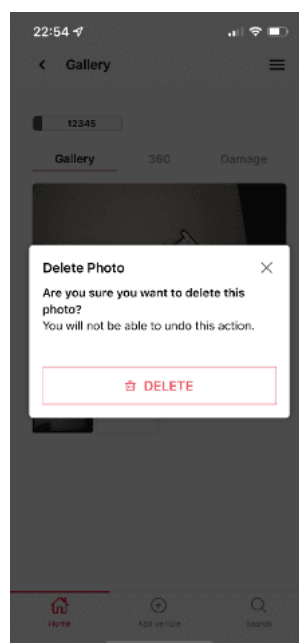


Рисунок 5.8

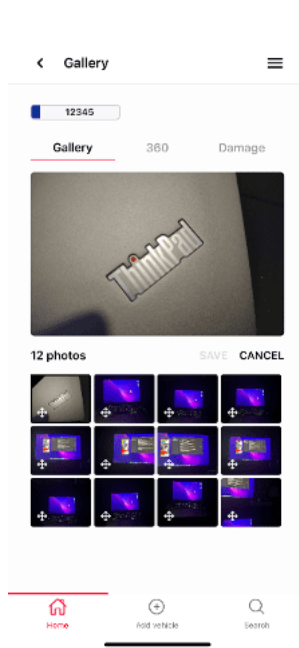


Рисунок 5.9

Додавання звичайних фото існує 2 варіанти:

- завантажувати по 1 фото;
- натиснути старт, слідувати прикладу та інструкціям і зробити всі фото разом.

Демонстрація на рисунках 5.10, 5.11, 5.12.

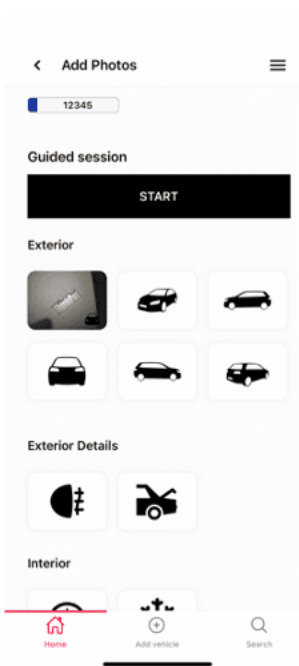


Рисунок 5.10

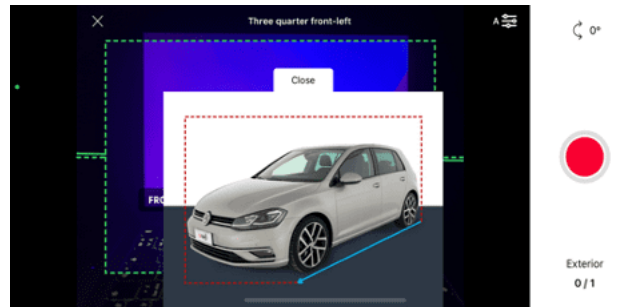


Рисунок 5.11

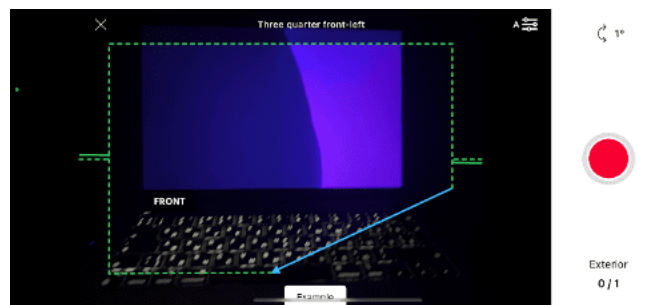


Рисунок 5.12

Після успішного завершення сесії фото автоматично стануть в чергу на завантаження і користувач буде бачити які вже завантажені, а які ні (рис. 5.13).

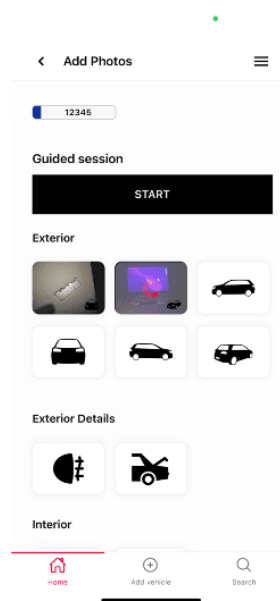


Рисунок 5.13 – Завантаження фото

Сторінку з усіма завантаженими ракурсами, зображено на рис. 5.14.

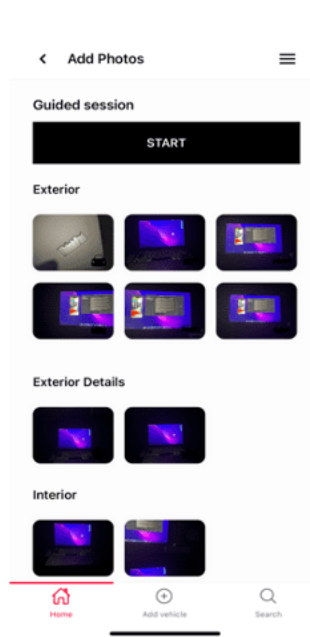


Рисунок 5.14

Також чергу на виконання можна переглянути на спеціальному екрані, зображеному на рис. 5.15.

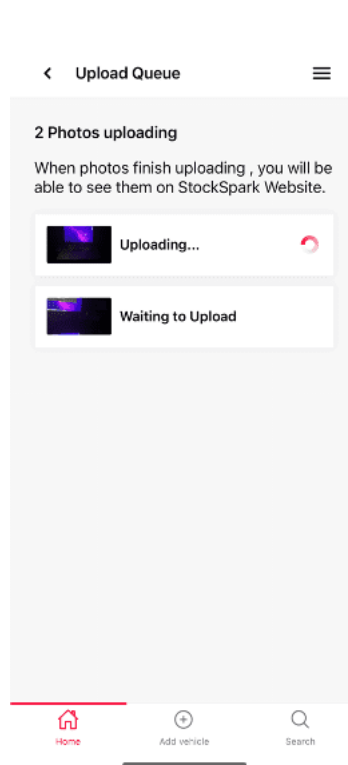


Рисунок 5.15

Під час завантаження та після користувач може бачити фото у галереї, змінювати їх, видаляти та змінювати порядок (рис. 5.16).

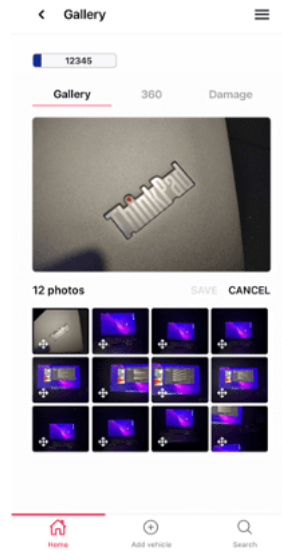


Рисунок 5.16

Крім інструкцій та макетів з ракурсами користувач може скористатися AR функціоналом для завантаження фото його авто (рис 5.17, 5.18, 5.19).

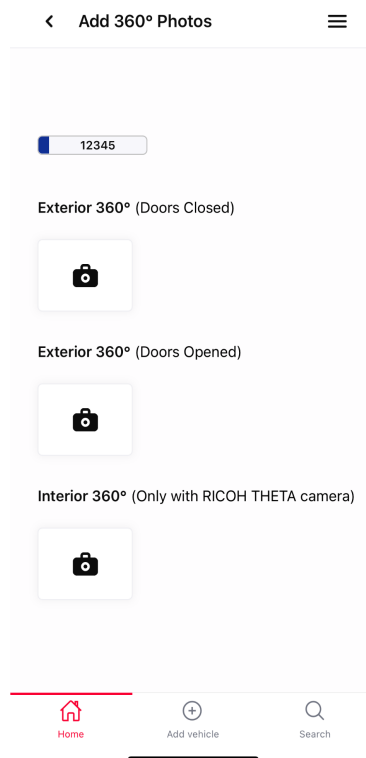


Рисунок 5.17 – Менеджер фото

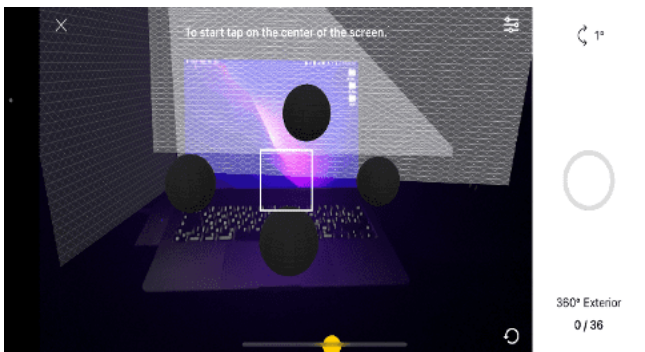


Рисунок 5.18 – AR

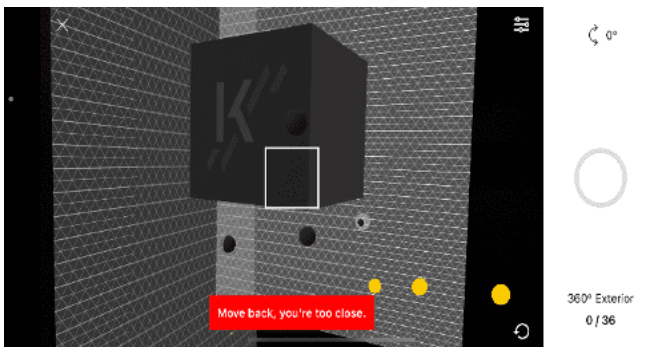


Рисунок 5.19 – AR

Також одна із можливостей це зробити панораму на камеру телефону та завантажити у додаток. Це зображення буде зчитане та ви зможете крутити телефон по сторонах та воно буде працювати як в VR окулярах але на екрані вашого телефону (рис. 5.20, 5.21).

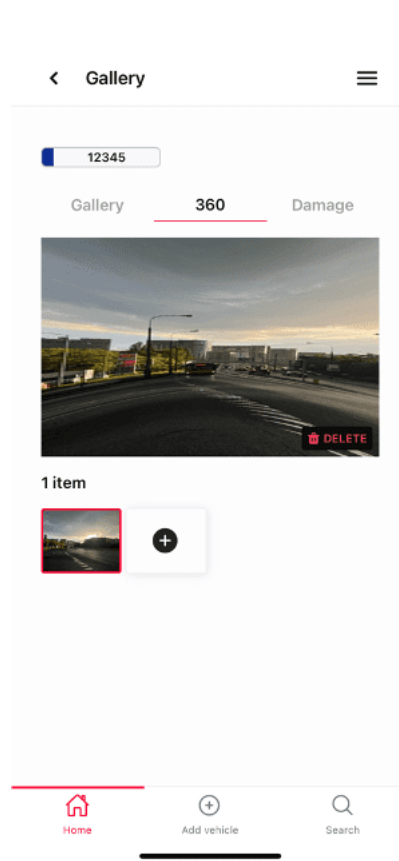


Рисунок 5.20 – Інтер'єр 360

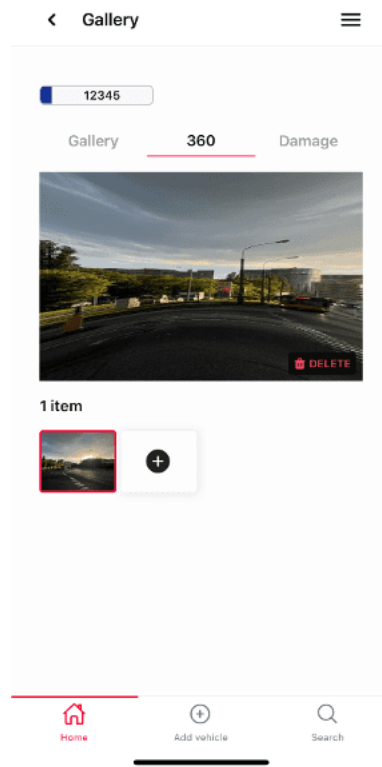


Рисунок 5.21 – Інтер'єр 360

5.3 Висновок

В даному розділі були наведені необхідні апаратні вимоги для функціонування програми та інструкцію організації процесу роботи з нею. Також описано усі доступні розділи інтерфейсу, їх компоненти та принципи їх роботи. Було описано процес створення та наведено приклади їх редагування користувачем, пояснено функціонування модуля який використовується для створення фото та AR фото. Було пояснено механізму роботи з додатком чергою та новими інструментами.

ВИСНОВКИ

Під час виконання роботи було виконано дослідження предметної області, розглянуто та визначено основні поняття, які пов'язані з обраною предметною областю. Також були проаналізовані та досліджені існуючі методи, кожен з яких можливо застосовувати для вирішення поставленої задачі, було визначено їх переваги та недоліки. Був проведений порівняльний аналіз досліджених методів, які можуть бути використані для вирішення задачі автоматизації роботи з ринком автомобілів.

Було сформульовано і визначено завдання до роботи.

Метою роботи є створення програмної системи для продажу та огляду авто на Європейському ринку.

В даній роботі досліджується сфера продажу авто та зручність користування додатками націленими на продаж автівок.

Для виконання поставленої мети було виконано наступні завдання:

- створити мобільний додаток для всіх платформ;
- створити методи для відстеження дій користувача;
- створити віртуальну чергу для загрузки зображень з хмарними технологіями;
- створити менеджер для фото;
- створити модулі для підтримки доповненої реальності.

Також в роботі були розглянуті різні мови програмування та середовища розробки, які дозволяють розробляти програмні продукти.

Було проведено їх аналіз, виділено основні переваги і недоліки та складено порівняльну характеристику. В ході цієї роботи було встановлено, що для програмного продукту найбільш ефективною мовою програмування є JavaScript, тому що вона більш проста і працює з достатньою швидкістю.

Також найбільш ефективним середовищем розробки було обрано середовище React Native, тому що воно є офіційним для мови програмування

JavaScript, не потребує великих обсягів апаратних ресурсів, а також дає можливість без перешкод розробити додаток відразу для всіх платформ.

Для роботи було розроблено структурну схему програмного продукту. Встановлено, що для розробки розумної черги для непомітного та швидкого завантаження, краще використовувати аналог серверної архітектури – мікросервіси.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бхаргава А. Грокаем Алгоритми / А. Бхаргава. – СПб: Пітер, 2019. – 290 с.
2. Фленаган Ф. JavaScript. / Ф. Девід. – Сіетл: Діалектика-Вільямс, 2021. – 720 с.
3. Резіг Д. JavaScript ніндзя / Д. Регіз. – Каліфорнія: Вільямс, 2018. – 544 с.
4. Фаулер М. Рефакторінг кода на JavaScript / М. Фаулер. – Бостон: Вільямс, 2019 – 464 с.
5. Тіленс М. Т. React в дії / М. Т. Тіленс. – СПб: Пітер, 2019. – 368с.
6. Дабіт Н. React Native in Action / Н. Дабіт. – Manning, 2019. – 320 с.
7. Petzold C. Code: The Hidden Language of Computer Hardware and Software / C. Petzold. – М: Microsoft Press, 2000. – 400 с.
8. Osmani A. Javascript Design Patterns / A. Osmani. – O'Reilly Media, 2021. – 254 с.
9. Althoff C. The Self-Taught Programmer: The Definitive Guide to Programming Professionally / C. Althoff. - O'Reilly Media, 2016. – 301 с.
10. Баас Л. Архитектура программного обеспечения на практике / Л. Баас. – СПб: Пітер, 2006 – 576 с.
11. Freeman E. Head First Design Patterns / E. Freeman. - O'Reilly Media and Associates, 2004 – 692 с.
12. Visual Studio Code – Code Editing. Redefined [Eletronic Resource]. – Access mode: <https://code.visualstudio.com>.
13. React – JavaScript бібліотека для створення користувацьких інтерфейсів [Електрон. ресурс]. – Режим доступу: <https://uk.reactjs.org>.
14. React Redux: Official React bindings for Redux [Electronics Resource]. – Access mode: <https://react-redux.js.org>.
15. Redux Form – The best way to manage your form state in Redux [Electronics Resource]. – Access mode: <https://redux-form.com>.

ДОДАТОК А
Технічне завдання

Вступ

В результаті виконання роботи була досліджена проблема автоматизації процесу створення, ведення та продажу автомобілів на Європейському ринку, яку було вирішено шляхом розробки кроссплатформного мобільного застосунку з новітніми технологіями.

Додаток має можливість не тільки перегляду та завантаження нових авто, але й розумну хмарну чергу для швидкого та зручного завантаження на сервер та технології AR для найкращих ракурсів для фото і перегляду авто у режимі 360 градусів.

A.1 Мета розробки і її призначення

Об'єкт дослідження – програмні засоби для зручного продажу.

Предмет дослідження – пріоритети та досвід сучасних торговельних платформ та ринку.

Мета роботи – створення мобільного додатку для автоматизації та поліпшення продажу автомобілів на Європейському ринку з використанням нових технологій для створення зображень.

Актуальність полягає в тому, що більшість компаній не надають належної уваги темі фотографій для приємного користування сервісом, як користувачів, так і продавців. Тому у моєму додатку для продажу авто по Європі є технології Augmented Reality та Optical character recognition з локалізацією та вибіркою для різних країн, щоб користувач міг якнайшвидше знайти те що йому треба, зробити фото під кращими ракурсами та переглядати все це у панорамі.

A.2 Стислий зміст основних теоретичних посилок до розробки на основі проведених пошуків

В даній роботі досліджується сфера продажу авто та зручність користування додатками націленими на продаж автівок.

Після дослідження потреб користувачів зробив висновки, що завжди є багато мобільних застосунків в яких достатньо функціоналу, але в різних додатках він різний. Тому було вирішено що додаток повинен вміщувати у себе всі можливості своїх аналогів та додатково мати функціонал який ще не існує у конкурентів.

Для виконання поставленої задачі необхідно виконати наступні завдання:

- створити мобільний додаток для всіх платформ;
- створити методи для відстеження дій користувача;
- створити віртуальну чергу для завантаження зображень з хмарними технологіями;
- створити менеджер для фото;
- створити модулі для підтримки доповненої реальності

A.3 Підстави для розробки

Підставою для виконання розробки є завдання на дипломний проект на тему “Розробка мобільного застосунку для продажу автовок на Європейському ринку”, затверджене наказом по НУ «Запорізька політехніка» № 102 від 27 квітня 2022 року.

A.4 Основні вимоги до програми, що розробляється, початковим даним і результату

A.4.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик:

- повинен бути реалізований функціонал для авторизації користувачів;
- повинна бути підтримка декількох мов для Європейського ринку;

- повинен бути реалізований метод для автоматичного та швидкого завантаження через хмарну технологію;
- повинен бути реалізований метод для перегляду в режимі панорами;
- повинен бути реалізований метод для користування AR;
- повинен бути реалізований метод для використання камери з підстановкою макетів у різних форматах для обоих платформ.

A.4.2 Вимоги до надійності

Додаток повинен вміти приховувати дані користувача під час авторизації та під час всієї роботи з відновленням сесії та під час завантаження даних через хмару.

Додаток повинен нормально функціонувати без помилок, особливо під час завантаження контенту до серверу, навіть якщо нестабільне з'єднання.

Додаток повинен збирати статистику користувачів та автоматично обробляти та зберігати помилки у додатку в єдиному просторі.

A.4.3 Вимоги до складу та параметрів технічних засобів

Вимоги до складу та параметрів технічних засобів:

- операційна система MacOS;
- 8 Гб оперативної пам'яті;
- 10 Гб вільного місця на жорсткому диску;
- доступ до мережі Інтернет.

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;

- керівництво програміста;
- керівництво оператора;
- опис програми.

A.6 Основні обмеження на установку і використання програмного продукту

Розроблений програмний продукт потребує інсталяції з електронного магазину вашого пристрою. Для того щоб користуватися усіма функціями додатку ваш смартфон повинен підтримувати технологію AR.

ДОДАТОК Б
Лістинги файлів роботи

Б.1. Файл vehicle-saga.tsx

```
import {
  put,
  all,
  select,
  call,
  takeLatest,
  delay,
} from 'redux-saga/effects';
import { PermissionsAndroid } from 'react-native';
import { Image } from 'react-native-compressor';
import FastImage from 'react-native-fast-image';
import _ from 'lodash';
import i18n from 'i18next';
import RNFS from 'react-native-fs';

import * as api from 'src/utls/api';
import * as actions from 'src/redux/actions';
import * as endpoints from 'src/constants/endpoints';
import * as DEFAULT_DATA from 'src/constants/defaultData';
import { getSortedSyncQForUploading, getUploadPhotoItem } from 'src/redux/selectors';
import { StackActions } from '@react-navigation/native';
import { navigationRef } from 'src/navigation/AppNavigator';

import { sendEvent } from 'src/utls/analytics';
import * as EVENTS from 'src/constants/events';

const authSelector = state => state.auth;
const syncQLoadingSelector = state => state.syncQ.loading;
const currentVehicleSelector = state => state.selectedVehicle.vehicle;
const countryIdSelector = state => state.settings.countryId;
const globalSearchSelector = state => state.globalSearch.result;
const filteredSearchSelector = state => state.filteredSearch.result;
const syncQArraySelector = state => getSortedSyncQForUploading(state);
```

```

function* getConfig() {
  try {
    const auth: any = yield select(authSelector);
    let countryId: string = yield select(countryIdSelector);
    if (countryId === 'en') countryId = 'uk'

    const { companyId } = auth;
    const { data } = yield call(api.get, endpoints.CONFIG(countryId));
    let imageShots = [];
    data?.map(item => item.companyId === companyId && (imageShots =
item?.galleryWizard.imageShots));
    yield put(actions.setConfigAction(imageShots));
  } catch (e) {

  }
}

function* createVehicle({ payload }) {
  try {
    yield put(actions.setLoaderAction(` ${i18n.t('addVehicle.creatingVehicle')}...`));

    const { companyId, dealerId } : { companyId: string, dealerId: string } = yield
select(authSelector);
    let countryId : string = yield select(countryIdSelector);
    if (countryId === 'en') countryId = 'uk';
    const { vin, plate } = payload;

    const { data: vehicle } = yield call(api.post, endpoints.CREATE_VEHICLE(countryId),
DEFAULT_DATA.addVehicleData(vin, plate, companyId, dealerId));
    yield delay(750);
    yield put(actions.stopLoaderAction());
    yield delay(350);
    yield put(actions.openVehicleAction({ ...vehicle, gallery: [] }));
    yield delay(350);
  }
}

```

```

    yield put(actions.addNotificationAction({ status: 'OK', title:
i18n.t('notifications.createdVehicle') }));

    yield sendEvent(EVENTS.CREATED_VEHICLE, { countryId, companyId, dealerId, vin,
plate });
  } catch (e) {
    yield put(actions.stopLoaderAction());
    console.log(e);
  }
};

```

```

function* damageGallery({ payload }) {
  try {
    let countryId: string = yield select(countryIdSelector);
    if (countryId === 'en') countryId = 'uk'
    const { vehicleId } = payload;

    const { data } = yield call(api.get, endpoints.DAMAGE_GALLERY(countryId, vehicleId));
    yield put(actions.setSelectedVehicleDamageGalleryAction(data));
  } catch (e) {
    console.log(e);
  }
};

```

```

function* setMainImage({ payload }) {
  try {
    const { vehicleId, imageId } = payload;
    const main = { index: 0, main: true };
    let countryId: string = yield select(countryIdSelector);
    if (countryId === 'en') countryId = 'uk'

    const globalSearch: any = yield select(globalSearchSelector);
    const filteredSearch: any = yield select(filteredSearchSelector);

    yield call(api.put, endpoints.CHANGE_ORDER(countryId, vehicleId, imageId), main);

    const vehicle = yield select(currentVehicleSelector);

```

```

const gallery = [];
const mainImage = vehicle?.gallery?.filter(_ => _.id === imageId)[0];
gallery.push({ ...mainImage, main: true, index: 0 });
vehicle?.gallery?.map(_ => _.id !== imageId ? gallery.push({ ..._, main: false, index:
gallery.length }) : null);
yield put(actions.resetSelectedVehicleGalleryAction(gallery));

if (globalSearch && globalSearch?.vehicles && globalSearch?.vehicles?.length > 0) {
  let usedVehicle = null;
  const filtered = globalSearch?.vehicles?.filter(_ => _.vehicleId === vehicleId);
  filtered?.length > 0 && (usedVehicle = filtered[0]);
  if (usedVehicle) {
    usedVehicle.gallery = [...gallery];
    const pureVehicles = globalSearch?.vehicles?.map(_ => _.vehicleId !== vehicleId ? _ :
usedVehicle);
    yield put(actions.resetGalleryForGlobalVehicleAction(pureVehicles));
  }
};

if (filteredSearch && filteredSearch?.vehicles && filteredSearch?.vehicles?.length > 0) {
  let usedVehicle = null;
  const filtered = filteredSearch?.vehicles?.filter(_ => _.vehicleId === vehicleId);
  filtered?.length > 0 && (usedVehicle = filtered[0]);
  if (usedVehicle) {
    usedVehicle.gallery = [...gallery];
    const pureVehicles = filteredSearch?.vehicles?.map(_ => _.vehicleId !== vehicleId ? _ :
usedVehicle);
    yield put(actions.resetGalleryForFilteredVehicleAction(pureVehicles));
  }
};

yield sendEvent(EVENTS.SET_MAIN_IMAGE, { vehicleId, imageId });
} catch (e) {
  console.log(e);
} finally {

```

```

    yield put(actions.stopLoaderAction());
  }
};

function* deleteDamageImages({ payload }) {
  try {
    yield delay(450);
    yield put(actions.setLoaderAction(`${i18n.t('gallery.processingDeleting')}...`));
    const { vehicleId, images } = payload;
    let countryId: string = yield select(countryIdSelector);
    if (countryId === 'en') countryId = 'uk'

    for (let i = 0; i < images.length; i++) {
      try {
        yield call(api.del, endpoints.DELETE_DAMAGE_IMAGE(countryId, vehicleId,
images[i]));
      } catch (e) {
        console.log(e);
      }
    }
  };

  yield put(actions.clearDeleteImagesFromDamageGalleryAction(images));
  yield delay(750);
  yield put(actions.stopLoaderAction());
  yield delay(150);
  yield put(actions.addNotificationAction({ status: 'OK', title: images?.length > 1 ?
i18n.t('notifications.deletedMultipleDamagePhotos', { count: images?.length }) :
i18n.t('notifications.deletedSingleDamagePhoto') }));
  yield sendEvent(images?.length > 1 ? EVENTS.DELETE_MULTIPLE_DAMAGE_IMAGE :
EVENTS.DELETE_SINGLE_DAMAGE_IMAGE, { countryId, vehicleId });
  } catch (e) {
    console.log(e);
  } finally {
    yield delay(1000);
    yield put(actions.stopLoaderAction());
  }
}

```

```

}
};

```

```

function* deleteImages({ payload }) {
  try {
    const { vehicleId, images } = payload;
    let countryId: string = yield select(countryIdSelector);
    if (countryId === 'en') countryId = 'uk'
    const currentVehicle = yield select(currentVehicleSelector);
    const globalSearch: any = yield select(globalSearchSelector);
    const filteredSearch: any = yield select(filteredSearchSelector);

    yield delay(450);
    if (images?.length > 1) yield
    put(actions.setLoaderAction(` ${i18n.t('gallery.processingDeleting')} ...`));

    const gallery = currentVehicle?.gallery?.filter(_ => !images.includes(_.id));
    const sortedGallery = gallery?.sort((a, b) => {
      if (a.index > b.index) return 1;
      if (b.index > a.index) return -1;

      return 0
    });

    if (sortedGallery && sortedGallery.length > 0 && sortedGallery[0].index !== 0) yield
    put(actions.setMainImageAction({ vehicleId, imageId: sortedGallery[0].id }));

    const pureGallery = sortedGallery?.map((_, index) => ({ ..._, index, main: index === 0 ? true :
    false }));

    yield put(actions.resetSelectedVehicleGalleryAction(pureGallery));

    if (globalSearch && globalSearch?.vehicles && globalSearch?.vehicles?.length > 0) {
      let usedVehicle = null;
      const filtered = globalSearch?.vehicles?.filter(_ => _.vehicleId === vehicleId);

```

```

    filtered?.length > 0 && (usedVehicle = filtered[0]);
    if (usedVehicle) {
      usedVehicle.gallery = [...pureGallery];
      const pureVehicles = globalSearch?.vehicles?.map(_ => _.vehicleId !== vehicleId ? _ :
usedVehicle);
      yield put(actions.resetGalleryForGlobalVehicleAction(pureVehicles));
    }
  };

  if (filteredSearch && filteredSearch?.vehicles && filteredSearch?.vehicles?.length > 0) {
    let usedVehicle = null;
    const filtered = filteredSearch?.vehicles?.filter(_ => _.vehicleId === vehicleId);
    filtered?.length > 0 && (usedVehicle = filtered[0]);
    if (usedVehicle) {
      usedVehicle.gallery = [...pureGallery];
      const pureVehicles = filteredSearch?.vehicles?.map(_ => _.vehicleId !== vehicleId ? _ :
usedVehicle);
      yield put(actions.resetGalleryForFilteredVehicleAction(pureVehicles));
    }
  };

  for (let i = 0; i < images.length; i++) {
    try {
      yield call(api.del, endpoints.DELETE_IMAGE(countryId, vehicleId, images[i]));
    } catch (e) {
      console.log(e);
    }
  };
  yield put(actions.stopLoaderAction());
  yield delay(150);
  yield put(actions.addNotificationAction({ status: 'OK', title: images?.length > 1 ?
i18n.t('notifications.deletedMultiplePhotos', { count: images?.length }) :
i18n.t('notifications.deletedSinglePhoto') }));
  yield sendEvent(images?.length > 1 ? EVENTS.DELETE_MULTIPLE_IMAGE :
EVENTS.DELETE_SINGLE_IMAGE, { countryId, vehicleId });

```

```

    payload && payload?.callback && (payload?.callback());
  } catch (e) {
    console.log(e);
  } finally {
    yield put(actions.stopLoaderAction());
  }
};

```

```

function* deleteThreeSixty({ payload }) {
  try {
    yield delay(450);
    yield put(actions.setLoaderAction(`${i18n.t('gallery.processingDeleting')}...`));

    const { vehicleId, type } = payload;
    let countryId: string = yield select(countryIdSelector);
    if (countryId === 'en') countryId = 'uk'

    let endpoint;

    switch (type) {
      case 'closed':
        {
          endpoint = endpoints.DELETE_ALL_THREE_SIXTY_CLOSED(countryId, vehicleId);
          break;
        }
      case 'opened':
        {
          endpoint = endpoints.DELETE_ALL_THREE_SIXTY_OPENED(countryId, vehicleId);
          break;
        }
      default: endpoint = endpoints.DELETE_ALL_THREE_SIXTY_INTERIOR(countryId,
vehicleId);
    }

    const res = yield call(api.del, endpoint);
  }
}

```

```

if (res) {
  switch (type) {
    case 'closed':
      {
        yield put(actions.clearLocalThreeSixtyClosed());
        break;
      };
    case 'opened':
      {
        yield put(actions.clearLocalThreeSixtyOpened());
        break;
      }
    default: yield put(actions.clearLocalThreeSixtyInterior());
  };
}
yield delay(750);
yield put(actions.stopLoaderAction());
yield delay(150);
if (res) {
  switch (type) {
    case 'closed':
      {
        yield put(actions.addNotificationAction({ status: 'OK', title:
i18n.t('notifications.deletedClosedDoors') }));
        yield sendEvent(EVENTS.DELETE_THREE_SIXTY_CLOSED_EXTERIOR, {
countryId, vehicleId });
        break;
      };
    case 'opened':
      {
        yield put(actions.addNotificationAction({ status: 'OK', title:
i18n.t('notifications.deletedOpenedDoors') }));
        yield sendEvent(EVENTS.DELETE_THREE_SIXTY_OPENED_EXTERIOR, {
countryId, vehicleId });
        break;
      };
    }
  }
}

```

```

    }
    default: {
      yield put(actions.addNotificationAction({ status: 'OK', title:
i18n.t('notifications.deletedInterior') }));
      yield sendEvent(EVENTS.DELETE_THREE_SIXTY_INTERIOR, { countryId, vehicleId
    });
    }
  };
}
} catch (e) {
  console.log(e);
} finally {
  yield delay(1000);
  yield put(actions.stopLoaderAction());
}
}

```

```

function* checkIsPhotoUploaded(syncQObject, endpoint, countryId, isFailed) {
  if (isFailed) {
    const result: Object = yield call(api.get, endpoint(countryId, syncQObject.vehicleId));

    if (result?.data && result?.data?.length > 0) {
      const foundPhoto = result?.data.find(item => item.index === syncQObject.index);

      if (foundPhoto?.id) {
        return { included: true, photo: foundPhoto };
      }
    }
  }
  return { included: false, photo: {} };
}

```

```

function* uploadSimplePhoto(syncQObject) {
  let countryId: string = yield select(countryIdSelector);
  if (countryId === 'en') countryId = 'uk'

```

```

const currentVehicle: Object = yield select(currentVehicleSelector);

const globalSearch: any = yield select(globalSearchSelector);
const filteredSearch: any = yield select(filteredSearchSelector);

const syncQObjectForStatus: Object = yield select(state => getUploadPhotoItem(state,
syncQObject));

let isFailed = syncQObjectForStatus.status === 'failed';

yield put(actions.updateStatusToImageAction({ vehicleId: syncQObject.vehicleId, pose:
syncQObject.pose, status: 'uploading' }));
yield put(actions.setLoadingSyncQAction(true));

const isPhotoUploaded: Object = yield checkIsPhotoUploaded(syncQObject,
endpoints.GALLERY, countryId, isFailed);

const isMain = syncQObject.index ? "false" : "true";
const compressed = yield Image.compress(syncQObject.uri, { maxWidth: 720, quality: 0.8, });

const formData = new FormData();
// formData.append('file', { uri: syncQObject.uri, name: 'file', type: 'image/jpeg' });
formData.append('file', { uri: compressed, name: 'file', type: 'image/jpeg' });
formData.append('index', `${syncQObject.index}`)
formData.append('main', isMain);
formData.append('tags', `["APP", "${syncQObject.pose}", "${syncQObject.type}"]`);

const result: Object = !isPhotoUploaded.included
? yield call(
  api.post,
  endpoints.SEND_IMAGE(countryId, syncQObject.vehicleId),
  formData,
  {
    headers: {
      'Content-Type': 'multipart/form-data'

```

```

    }
  }
)
: { data: { status: 'OK', image: isPhotoUploaded.photo } } };

if (result?.data?.status === 'OK') {
  if (currentVehicle?.vehicleId && currentVehicle.vehicleId === syncQObject.vehicleId) {
    yield put(actions.addPhotoToCurrentVehicleAfterSyncAction(result.data.image));
  }

  if (globalSearch && globalSearch?.vehicles && globalSearch?.vehicles?.length > 0) {
    let usedVehicle = null;
    const filtered = globalSearch?.vehicles?.filter(_ => _.vehicleId === syncQObject.vehicleId);
    filtered?.length > 0 && (usedVehicle = filtered[0]);
    if (usedVehicle) {
      usedVehicle.gallery = [...usedVehicle.gallery, { ...result.data.image, main:
usedVehicle.gallery.length ? false : true }];
      const pureVehicles = globalSearch?.vehicles?.map(_ => _.vehicleId !==
syncQObject.vehicleId ? _ : usedVehicle);
      yield put(actions.resetGalleryForGlobalVehicleAction(pureVehicles));
    }
  }

  if (filteredSearch && filteredSearch?.vehicles && filteredSearch?.vehicles?.length > 0) {
    let usedVehicle = null;
    const filtered = filteredSearch?.vehicles?.filter(_ => _.vehicleId === syncQObject.vehicleId);
    filtered?.length > 0 && (usedVehicle = filtered[0]);
    if (usedVehicle) {
      usedVehicle.gallery = [...usedVehicle.gallery, { ...result.data.image, main:
usedVehicle.gallery.length ? false : true }];
      const pureVehicles = filteredSearch?.vehicles?.map(_ => _.vehicleId !==
syncQObject.vehicleId ? _ : usedVehicle);
      yield put(actions.resetGalleryForFilteredVehicleAction(pureVehicles));
    }
  }
}

```

```

yield sendEvent(EVENTS.UPLOAD_GALLERY_PHOTO, result.data.image);

yield put(actions.removeSyncQAfterSuccessAction({ vehicleId: syncQObject.vehicleId, pose:
syncQObject.pose }));
}
}

function* uploadThreeSixtyExteriorPhoto(syncQObject) {
  let countryId: string = yield select(countryIdSelector);
  if (countryId === 'en') countryId = 'uk'
  const currentVehicle: Object = yield select(currentVehicleSelector);

  const syncQObjectForStatus: Object = yield select(state => getUploadPhotoItem(state,
syncQObject));

  let isFailed = syncQObjectForStatus.status === 'failed';

  yield put(actions.updateStatusToImageAction({ vehicleId: syncQObject.vehicleId, pose:
syncQObject.pose, status: 'uploading' }));
  yield put(actions.setLoadingSyncQAction(true));
  console.log('syncQObject', syncQObject);

  const compressed = yield Image.compress(syncQObject.uri, { maxWidth: 720, quality: 0.8, });

  const formData = new FormData();
  // formData.append('file', { uri: syncQObject.uri, name: 'file', type: 'image/jpeg' });
  formData.append('file', { uri: compressed, name: 'file', type: 'image/jpeg' });
  formData.append('index', `${syncQObject.index}`)
  formData.append('tags', `["APP", "${syncQObject.poseCode}", "${syncQObject.type}"]`);

  let result: Object;

  if (syncQObject.type === 'exteriorClosed') {

```

```

const isPhotoUploaded: Object = yield checkIsPhotoUploaded(syncQObject,
endpoints.THREE_SIXTY_CLOSED, countryId, true);

result = !isPhotoUploaded.included
  ? yield call(
    api.post,
    endpoints.SEND_THREE_SIXTY_EXTERIOR_CLOSED_IMAGE(countryId,
syncQObject.vehicleId),
    formData,
    {
      headers: {
        'Content-Type': 'multipart/form-data'
      }
    }
  )
  : { data: { status: 'OK', image: isPhotoUploaded.photo } };
} else {
  const isPhotoUploaded: Object = yield checkIsPhotoUploaded(syncQObject,
endpoints.THREE_SIXTY_OPENED, countryId, true);

result = !isPhotoUploaded.included
  ? yield call(
    api.post,
    endpoints.SEND_THREE_SIXTY_EXTERIOR_OPENED_IMAGE(countryId,
syncQObject.vehicleId),
    formData,
    {
      headers: {
        'Content-Type': 'multipart/form-data'
      }
    }
  )
  : { data: { status: 'OK', image: isPhotoUploaded.photo } };
}
console.log('result', result?.data);

```

```

if (result && result?.data && result?.data?.image && result?.data?.image?.url) {
  FastImage.preload([
    {
      uri: result?.data?.image?.url,
    }
  ])
}

if (result?.data?.status === 'OK' || result?.data?.status === 200) {
  if (currentVehicle?.vehicleId && currentVehicle.vehicleId === syncQObject.vehicleId) {
    if (syncQObject.type === 'exteriorClosed') {
      yield
      put(actions.updateThreeSixtyExteriorClosedPhotoToCurrentVehicleAfterSyncAction(result.data
        .image));
      yield sendEvent(EVENTS.UPLOAD_THREESIXTY_EXTERIOR_CLOSED_PHOTO,
        result.data.image);
    } else {
      yield sendEvent(EVENTS.UPLOAD_THREESIXTY_EXTERIOR_OPENED_PHOTO,
        result.data.image);
      yield
      put(actions.updateThreeSixtyExteriorOpenedPhotoToCurrentVehicleAfterSyncAction(result.dat
        a.image));
    }
  }
  yield put(actions.removeSyncQAfterSuccessAction({ vehicleId: syncQObject.vehicleId, pose:
    syncQObject.pose }));
}
}

function* uploadThreeSixtyInteriorPhoto(syncQObject) {
  let countryId: string = yield select(countryIdSelector);
  if (countryId === 'en') countryId = 'uk'
  const currentVehicle: Object = yield select(currentVehicleSelector);

```

```

const syncQObjectForStatus: Object = yield select(state => getUploadPhotoItem(state,
syncQObject));

let isFailed = syncQObjectForStatus.status === 'failed';

yield put(actions.updateStatusToImageAction({ vehicleId: syncQObject.vehicleId, pose:
syncQObject.pose, status: 'uploading' }));
yield put(actions.setLoadingSyncQAction(true));

const isPhotoUploaded: Object = yield checkIsPhotoUploaded(syncQObject,
endpoints.THREE_SIXTY_INTERIOR, countryId, isFailed);
const compressed = yield Image.compress(syncQObject.uri, { maxWidth: 720, quality: 0.8, });

const formData = new FormData();
// formData.append('file', { uri: syncQObject.uri, name: 'file', type: 'image/jpeg' });
formData.append('file', { uri: compressed, name: 'file', type: 'image/jpeg' });
formData.append('index', `${syncQObject.index}`)
formData.append('tags', `["APP", "${syncQObject.pose}", "${syncQObject.type}"]`);

const result: Object = !isPhotoUploaded.included
? yield call(
  api.post,
  endpoints.SEND_THREE_SIXTY_INTERIOR_IMAGE(countryId, syncQObject.vehicleId),
  formData,
  {
    headers: {
      'Content-Type': 'multipart/form-data'
    }
  }
)
: { data: { status: 'OK', image: isPhotoUploaded.photo } };

console.log('result', result);

if (result?.data?.status === 'OK' || result?.data?.status === 200) {

```

```

    if (currentVehicle?.vehicleId && currentVehicle.vehicleId === syncQObject.vehicleId) {
      yield
      put(actions.updateThreeSixtyInteriorPhotoToCurrentVehicleAfterSyncAction(result.data.image)
    );
    }

    yield sendEvent(EVENTS.UPLOAD_THREESIXTY_INTERIOR_PHOTO,
result.data.image);

    yield put(actions.removeSyncQAfterSuccessAction({ vehicleId: syncQObject.vehicleId, pose:
syncQObject.pose }));
  }
}

function* uploadDamagePhoto(syncQObject) {
  let countryId: string = yield select(countryIdSelector);
  if (countryId === 'en') countryId = 'uk'
  const currentVehicle: Object = yield select(currentVehicleSelector);

  const syncQObjectForStatus: Object = yield select(state => getUploadPhotoItem(state,
syncQObject));

  let isFailed = syncQObjectForStatus.status === 'failed';

  yield put(actions.updateStatusToImageAction({ vehicleId: syncQObject.vehicleId, pose:
syncQObject.pose, status: 'uploading' }));
  yield put(actions.setLoadingSyncQAction(true));

  const isPhotoUploaded: Object = yield checkIsPhotoUploaded(syncQObject,
endpoints.DAMAGE_GALLERY, countryId, isFailed);
  const compressed = yield Image.compress(syncQObject.uri, { maxWidth: 720, quality: 0.8, });

  const formData = new FormData();
  // formData.append('file', { uri: syncQObject.uri, name: 'file', type: 'image/jpeg' });
  formData.append('file', { uri: compressed, name: 'file', type: 'image/jpeg' });

```

```

formData.append('index', `${syncQObject.index}`)
formData.append('tags', `["APP", "${syncQObject.type}"]`);

const result: Object = !isPhotoUploaded.included
  ? yield call(
    api.post,
    endpoints.SEND_DAMAGE_IMAGE(countryId, syncQObject.vehicleId),
    formData,
    {
      headers: {
        'Content-Type': 'multipart/form-data'
      }
    }
  )
  : { data: { status: 'OK', image: isPhotoUploaded.photo } };

console.log('result', result);

if (result?.data?.status === 'OK' || result?.data?.status === 200) {
  if (currentVehicle?.vehicleId && currentVehicle.vehicleId === syncQObject.vehicleId) {
    yield put(actions.updateDamagePhotoToCurrentVehicleAfterSyncAction(result.data.image));
  }

  yield sendEvent(EVENTS.UPLOAD_DAMAGE_PHOTO, result.data.image);

  yield put(actions.removeSyncQAfterSuccessAction({ vehicleId: syncQObject.vehicleId, pose:
syncQObject.pose }));
}
}

function* syncLogic() {
  const syncQArray: Array<Object> = yield select(syncQArraySelector);
  if (syncQArray.length) {
    const syncQObject = syncQArray[0];
    if (syncQObject?.pose === 'threeSixtyInterior') {

```

```

    yield uploadThreeSixtyInteriorPhoto(syncQObject)
  } else if (syncQObject?.type === 'exteriorClosed' || syncQObject?.type === 'exteriorOpened') {
    yield uploadThreeSixtyExteriorPhoto(syncQObject);
  } else if (syncQObject?.type === 'DAMAGE') {
    yield uploadDamagePhoto(syncQObject);
  } else {
    yield uploadSimplePhoto(syncQObject);
  }
}
}
}

```

```

function* syncFailed(e) {
  try {
    if (e?.response?.status) {
      const syncQArray: Array<Object> = yield select(syncQArraySelector);
      if (syncQArray.length) {
        const syncQObject = syncQArray[0];

        yield sendEvent(EVENTS.UPLOAD_PHOTO_ERROR, syncQObject);
      }
      yield delay(15000);
      yield syncLogic();
    } else {
      const syncQArray: Array<Object> = yield select(syncQArraySelector);
      if (syncQArray.length) {
        const syncQObject = syncQArray[0];
        yield put(actions.updateStatusToImageFailedAction({ vehicleId: syncQObject.vehicleId,
pose: syncQObject.pose }));
      }
    }
  } catch (error) {
    const syncQArray: Array<Object> = yield select(syncQArraySelector);
    if (syncQArray.length) {
      const syncQObject = syncQArray[0];

```

```

    if (error?.response?.status) {
      yield sendEvent(EVENTS.UPLOAD_PHOTO_ERROR, syncQObject);
    }

    yield put(actions.updateStatusToImageFailedAction({ vehicleId: syncQObject.vehicleId,
pose: syncQObject.pose }));
  }
}
}

function* sync({ payload }) {
  if (payload) {
    try {
      yield syncLogic();
    } catch (e) {
      console.log('e', e);
      yield syncFailed(e);
    }
  } else {
    yield put(actions.setLoadingSyncQAction(false));
  }
};

function* getThreeSixty() {
  try {
    let countryId: string = yield select(countryIdSelector);
    if (countryId === 'en') countryId = 'uk'
    const { vehicleId } = yield select(currentVehicleSelector);

    const { data: closed } = yield call(api.get, endpoints.THREE_SIXTY_CLOSED(countryId,
vehicleId));
    const { data: opened } = yield call(api.get, endpoints.THREE_SIXTY_OPENED(countryId,
vehicleId));
    const { data: interior } = yield call(api.get, endpoints.THREE_SIXTY_INTERIOR(countryId,
vehicleId));
  }
}

```

```

[closed, opened, interior]?.map(type => {
  if (type.length > 0) {
    FastImage.preload([...type?.map(_ => ({ uri: _.url }))] );
  }
});
yield put(actions.setThirtySixtyAction({ closed, opened, interior }));
// console.log('closed, opened, interior', closed, opened, interior);

// try {
//   const granted = yield PermissionsAndroid.requestMultiple([
//     PermissionsAndroid.PERMISSIONS.WRITE_EXTERNAL_STORAGE,
//     PermissionsAndroid.PERMISSIONS.READ_EXTERNAL_STORAGE,]);
//   const readGranted = yield
PermissionsAndroid.check(PermissionsAndroid.PERMISSIONS.READ_EXTERNAL_STORA
GE);
//   const writeGranted = yield
PermissionsAndroid.check(PermissionsAndroid.PERMISSIONS.WRITE_EXTERNAL_STORA
GE);
//   const hash = {};
//   if (readGranted && writeGranted) {
//     try {

//       const absoluteStorageLocation = `${RNFS.DocumentDirectoryPath}/images/`;
//       const isPathSet = yield RNFS.exists(absoluteStorageLocation);
//       if (!isPathSet) yield RNFS.mkdir(absoluteStorageLocation);
//       const typesArr = ['closed', 'opened', 'interior'];

//       [closed, opened, interior]?.map((type, typeIdx) => {
//         if (type.length > 0) {
//           type?.map((img, index) => {
//             const storageLocation = RNFS.DocumentDirectoryPath
+ '/images/' + `${typesArr[typeIdx]}_${vehicleId}_${index}.jpg`;
//             console.log('storageLocation', storageLocation);
//             RNFS.downloadFile({ fromUrl: img?.url, toFile: storageLocation }).promise.then(res
=> {

```

```

//      console.log('res.statusCode', res.statusCode);
//      if (res.statusCode === 200) {
//          hash[img?.url] = storageLocation;
//      }
//      });
//  })
//  }
//  });
//  } catch (e) {
//      console.log('Downloading error', e);
//  }
//  } else {
//      return false
//  }
//  // yield delay(2000);
//  // console.log('hash', hash);
//  // yield put(actions.setHashAction(hash));
//  } catch (err) {
//      console.warn(err);
//  }

// RNFS.downloadFile()

} catch (e) {
    console.log(e);
}
};

function* getPermissions() {
    try {
        yield put(actions.setLoaderAction(`${i18n.t('gallery.loading')}...`));
        let countryId: string = yield select(countryIdSelector);
        if (countryId === 'en') countryId = 'uk'
    }
}

```

```

const { data: companies } = yield call(api.get, endpoints.COMPANY(countryId));
const permissions = [];
companies?.map(company => company?.moduleActivations?.map(module =>
permissions.push(module?.name)));
yield put(actions.setPermissionsAction(_.uniq(permissions)));
} catch (e) {
  console.log(e);
} finally {
  yield delay(750);
  yield put(actions.stopLoaderAction());
}
};

function* reorder({ payload: images }) {
  try {
    let countryId: string = yield select(countryIdSelector);
    if (countryId === 'en') countryId = 'uk'
    const { vehicleId } = yield select(currentVehicleSelector);
    const globalSearch: any = yield select(globalSearchSelector);
    const filteredSearch: any = yield select(filteredSearchSelector);

    for (let i = 0; i < images.length; i++) {
      const { index, main } = images[i];

      yield delay(100);
      yield call(api.put, endpoints.REORDER_GALLERY(countryId, vehicleId, images[i].id), {
index, main });
    };

    const { data: gallery } = yield call(api.get, endpoints.GET_DEFAULT_GALLERY(countryId,
vehicleId));
    yield put(actions.resetSelectedVehicleGalleryAction(gallery));

    if (globalSearch && globalSearch?.vehicles && globalSearch?.vehicles?.length > 0) {

```

```

let usedVehicle = null;
const filtered = globalSearch?.vehicles?.filter(_ => _.vehicleId === vehicleId);
filtered?.length > 0 && (usedVehicle = filtered[0]);
if (usedVehicle) {
  usedVehicle.gallery = [...gallery];
  const pureVehicles = globalSearch?.vehicles?.map(_ => _.vehicleId !== vehicleId ? _ :
usedVehicle);
  yield put(actions.resetGalleryForGlobalVehicleAction(pureVehicles));
}
};

if (filteredSearch && filteredSearch?.vehicles && filteredSearch?.vehicles?.length > 0) {
  let usedVehicle = null;
  const filtered = filteredSearch?.vehicles?.filter(_ => _.vehicleId === vehicleId);
  filtered?.length > 0 && (usedVehicle = filtered[0]);
  if (usedVehicle) {
    usedVehicle.gallery = [...gallery];
    const pureVehicles = filteredSearch?.vehicles?.map(_ => _.vehicleId !== vehicleId ? _ :
usedVehicle);
    yield put(actions.resetGalleryForFilteredVehicleAction(pureVehicles));
  }
};
yield put(actions.stopLoaderAction());
yield delay(350);
yield put(actions.addNotificationAction({ status: 'OK', title: i18n.t('notifications.reorderSaved')
})));
yield sendEvent(EVENTS.REORDER_IMAGES, { countryId, vehicleId });
} catch (e) {
  console.log(e);
} finally {
  yield put(actions.stopLoaderAction());
}
};

function* reorderDamage({ payload: images }) {

```

```

try {
  let countryId: string = yield select(countryIdSelector);
  if (countryId === 'en') countryId = 'uk'
  const { vehicleId } = yield select(currentVehicleSelector);

  for (let i = 0; i < images.length; i++) {
    const { index } = images[i];

    yield delay(100);
    yield call(api.put, endpoints.REORDER_DAMAGE_GALLERY(countryId, vehicleId,
images[i].id), { index, main: false });
  };

  const { data: damageGallery } = yield call(api.get,
endpoints.DAMAGE_GALLERY(countryId, vehicleId));
  yield put(actions.setSelectedVehicleDamageGalleryAction(damageGallery));
  yield sendEvent(EVENTS.REORDER_DAMAGE_IMAGES, { countryId, vehicleId });

} catch (e) {
  console.log(e);
}
};

function* openVehicle({ payload: vehicle }) {
  try {
    yield put(actions.setLoaderAction(118n.t('vehicleDetails.loadingVehicle')));

    const { vehicleId } = vehicle;
    FastImage.preload([...vehicle?.gallery?.map(_ => ({ uri: _.url } ))]);

    yield put(actions.getConfigAction());
    yield put(actions.setSelectedVehicleAction(vehicle));
    yield put(actions.getSelectedVehicleDamageGalleryAction({ vehicleId }));
    yield put(actions.getThreeSixtyAction());
    yield delay(750);
  }
}

```

```

yield navigationRef.current?.navigate('VehicleDetails');
yield sendEvent(EVENTS.OPENED_VEHICLE_DETAILS, { vehicleId });
} catch (e) {
  console.log(e);
} finally {
  console.log('HERE')
  yield delay(250);
  yield put(actions.stopLoaderAction());
}
}

```

```
function* deleteVehicle({ payload: vehicle }) {
```

```

  try {
    yield put(actions.setLoaderAction(int('gallery.processingDeleting')));
    const { vehicleId } = vehicle;

```

```

    let countryId: string = yield select(countryIdSelector);

```

```

    if (countryId === 'en') countryId = 'uk';

```

```

    const globalSearch: any = yield select(globalSearchSelector);

```

```

    const filteredSearch: any = yield select(filteredSearchSelector);

```

```

    yield call(api.del, endpoints.DELETE_VEHICLE(countryId, vehicleId));

```

```

    if (globalSearch && globalSearch?.vehicles && globalSearch?.vehicles?.length > 0) {

```

```

      const totalPages = (globalSearch?.vehicles.length === 1) ? globalSearch?.totalPages - 1 :

```

```

      globalSearch?.totalPages

```

```

      const totalVehicles = globalSearch?.totalVehicles - 1

```

```

      const vehicles = globalSearch?.vehicles?.filter(_ => _.vehicleId !== vehicleId);

```

```

      yield put(actions.deleteVehicleFromListAction({ totalPages, totalVehicles, vehicles }));

```

```

    }

```

```

    if (filteredSearch && filteredSearch?.vehicles && filteredSearch?.vehicles?.length > 0) {

```

```

      const totalPages = (filteredSearch?.vehicles.length === 1) ? filteredSearch?.totalPages - 1 :

```

```

      filteredSearch?.totalPages

```

```

const totalVehicles = filteredSearch?.totalVehicles - 1
const vehicles = filteredSearch?.vehicles?.filter(_ => _.vehicleId !== vehicleId);
yield put(actions.deleteVehicleFromFilteredListAction({ totalPages, totalVehicles, vehicles
})));
}
yield put(actions.stopLoaderAction());
yield navigationRef.current?.goBack();
yield delay(500);
yield put(actions.addNotificationAction({ title:
i18n.t('notifications.deleteVehicleSuccessfully'), status: 'OK' }));
// yield put(actions.addNotificationAction({ title: `Deleted ${vehicle?.numberPlate || 'Vehicle'}
Sucessefully`, status: 'OK' }));
} catch (e) {
  console.log(e);
} finally {
  yield delay(250);
  yield put(actions.stopLoaderAction());
}
}

```

```

export default function* activitiesSagas() {
  yield all([
    takeLatest([actions.startSyncAction, actions.stopSyncAction], sync),
    takeLatest(actions.getConfigAction, getConfig),
    takeLatest(actions.setMainImageAction, setMainImage),
    takeLatest(actions.createVehicleAction, createVehicle),
    takeLatest(actions.getThreeSixtyAction, getThreeSixty),
    takeLatest(actions.deleteImagesFromVehicleAction, deleteImages),
    takeLatest(actions.getSelectedVehicleDamageGalleryAction, damageGallery),
    takeLatest(actions.deleteThreeSixtyAction, deleteThreeSixty),
    takeLatest(actions.deleteDamageImagesAction, deleteDamageImages),
    takeLatest(actions.getPermissionsAction, getPermissions),
    takeLatest(actions.reorderGalleryAction, reorder),
    takeLatest(actions.reoderDamageGalleryAction, reorderDamage),

```

```

    takeLatest(actions.openVehicleAction, openVehicle),
    takeLatest(actions.deleteVehicleAction, deleteVehicle),
  ]);
};

```

Б.2. Файл Gallery/index.tsx

```

import React, { useEffect, useMemo, useState } from 'react';
import { connect } from 'react-redux';

import Gallery from './Gallery';
import * as actions from 'src/redux/actions';
import { getVehicleWithsyncQ } from 'src/redux/selectors';
import { useTranslation } from 'react-i18next';
import FastImage from 'react-native-fast-image';
import { reorderThreeSixtyArrayForViewer } from 'src/utls/syncQUtls';

const GalleryScreen = (props: any) => {
  const {
    hash,
    navigation,
    vehicle,
    setLoader,
    reorderGallery,
    permissions,
    getPermissions,
    setMainImage,
    deleteImages,
    deleteDamageImages,
    openPopup,
    closePopup,
    deleteThreeSixty,
    reorderDamageGallery,
  } = props;
  const { t } = useTranslation();

  const [idx, _idx] = useState(0);
  const [category, _category] = useState('gallery');
  const [images, _images] = useState(vehicle?.gallery || []);
  const [enabledSelection, _enabledSelection] = useState(false);
  const [selectedImages, _selectedImages] = useState([]);
  const [pureImages, _pureImages] = useState(vehicle?.gallery || []);

```

```

const [isPanorama, _isPanorama] = useState(false);
const [is360Viewer, _is360Viewer] = useState(false);
const [pureClosed, _pureClosed] = useState([]);
const [pureOpened, _pureOpened] = useState([]);
const [isScrollEnabled, setIsScrollEnabled] = useState(true);

const [isEnabledReordering, _isEnabledReordering] = useState(false);
const [reorderedImages, _reorderedImages] = useState([]);

useEffect(() => {
  getPermissions();
}, []);

useEffect(() => {
  // _images(vehicle?.gallery || []);
  // _idx(0);
  _enabledSelection(false);
  _selectedImages([]);
  changeCategory(category);
}, [vehicle]);

useEffect(() => {
  _is360Viewer(category === 'threeSixty' && !images[idx]?.tags?.includes('threeSixtyInterior')
? true : false);
  _isPanorama((images[idx]?.tags?.includes('threeSixtyInterior')));
}, [idx, images]);

const changeCategory = (c: string) => {
  _is360Viewer(false);
  _isPanorama(false);
  _category(c);
  _isEnabledReordering(false);
  _reorderedImages([]);
  if (c === 'threeSixty' && vehicle) {
    const interior = [];
    const closed = [];
    const opened = [];
    if (vehicle[c] && vehicle[c]?.interior && vehicle[c]?.interior.length > 0)
interior.push(vehicle[c]?.interior[vehicle[c]?.interior.length - 1]);
    if (vehicle[c] && vehicle[c]?.closed && vehicle[c]?.closed.length > 0)
closed.push(vehicle[c]?.closed[0]);
    if (vehicle[c] && vehicle[c]?.opened && vehicle[c]?.opened.length > 0)
opened.push(vehicle[c]?.opened[0]);
    vehicle[c] && vehicle[c]?.closed && vehicle[c]?.closed?.length > 0 &&
(_pureClosed(reoderThreeSixtyArrayForViewer([...vehicle[c]?.closed])));
  }
}

```

```

    vehicle[c] && vehicle[c]?.opened && vehicle[c]?.opened?.length > 0 &&
    (_pureOpened(reoderThreeSixtyArrayForViewer([...vehicle[c]?.opened])));
    _images([...interior, ...closed, ...opened]);

    } else {
      _images(vehicle && vehicle[c] ? vehicle[c]?.map((_, order) => ({ ..._, order })) : []);
    };
    category !== c && (_idx(0));
    _enabledSelection(false);
    _selectedImages([]);
  };

const onSelection = (id) => {
  // alert('Selection disabled for now :(');
  // return;
  _enabledSelection(!enabledSelection);
  _selectedImages([id]);
};

const cancelSelection = () => {
  _enabledSelection(false);
  _selectedImages([]);
};

const selectMini = (i: number) => {
  _isPanorama(false);
  _is360Viewer(false);
  i !== idx && (_idx(i))
}

const selectImage = (id: string) => {
  selectedImages.includes(id) ? _selectedImages([...selectedImages?.filter(_ => _ !== id)]) :
  _selectedImages([...selectedImages, id]);
}

const addPress = () => {
  switch (category) {
    case 'gallery': {
      navigation.navigate('Poses');
      break;
    }
    case 'damageGallery': {
      navigation.navigate('DamageManage');
      break;
    }
    case 'threeSixty': {

```

```

        navigation.navigate('360Manage');
        break;
    }
    default: alert('Will be soon');
};
};

const setMain = () => {
    setLoader(`${t('login.processing')}...`);
    setMainImage({ vehicleId: vehicle?.vehicleId, imageId: images[idx]?.id });
    _idx(0);
};

const deleteAnyThreeSixty = () => {
    const { vehicleId } = vehicle;

    const action = () => {
        switch (true) {
            case images[idx]?.tags?.includes('exteriorClosed') || images[idx].type ===
'CLOSED_DOORS_360':
                {
                    deleteThreeSixty({ vehicleId, type: 'closed' });
                    break;
                }
            case images[idx]?.tags?.includes('exteriorOpened') || images[idx].type ===
'OPENED_DOORS_360':
                {
                    deleteThreeSixty({ vehicleId, type: 'opened' });
                    break;
                }
            default: {
                deleteThreeSixty({ vehicleId, type: 'interior' });
            }
        }
    };

    openPopup({
        isVisible: true,
        data: {
            type: 'delete',
            title: t('gallery.delete360Title'),
            subTitle: t('gallery.delete360SubTitle'),
            description: t('gallery.deleteDescription'),
            action,
        },
    })
}

```

```
};
```

```
const deleteDamage = (imageId) => {
  let imagesToDelete: Array<string>;
  imageId ? (imagesToDelete = [imageId]) : imagesToDelete = [...selectedImages];

  openPopup({
    isVisible: true,
    data: {
      type: 'delete',
      title: t('gallery.deletePhotoTitle'),
      subTitle: t('gallery.deletePhotoSubTitle'),
      description: t('gallery.deleteDescription'),
      action: () => deleteDamageImages({ vehicleId: vehicle?.vehicleId, images: imagesToDelete,
callback: closePopup() }),
    },
  })
};
```

```
const deleteOne = (imageId) => {
  if (category === 'damageGallery') {
    deleteDamage(imageId);
    return;
  };

  openPopup({
    isVisible: true,
    data: {
      type: 'delete',
      title: t('gallery.deletePhotoTitle'),
      subTitle: t('gallery.deletePhotoSubTitle'),
      description: t('gallery.deleteDescription'),
      action: () => {
        deleteImages({ vehicleId: vehicle?.vehicleId, images: [imageId], callback: closePopup() });
        setTimeout(() => _idx(0));
      },
    },
  })
};
```

```
const deleteMultiple = () => {
  if (category === 'damageGallery') {
    deleteDamage();
    return;
  };
};
```

```

openPopup({
  isVisible: true,
  data: {
    type: 'delete',
    title: t('gallery.deletePhotosTitle'),
    subTitle: t('gallery.deletePhotosSubTitle'),
    description: t('gallery.deleteDescription'),
    action: () => {
      deleteImages({ vehicleId: vehicle?.vehicleId, images: [...selectedImages], callback:
closePopup() });
      setTimeout(() => _idx(0));
    },
  },
});
};
const onDragStart = () => setIsScrollEnabled(false);

const onDragRelease = (order: Array<any>) => {
  const oldId = reorderedImages[_idx]?.id;
  let newIdx;
  order?.map((img, i) => img?.id === oldId ? (newIdx = i) : null);
  // console.log('BEFORE', reorderedImages);
  // console.log('order', order);
  _idx(newIdx);
  _reorderedImages(order);
  setIsScrollEnabled(true);
}

const startReordering = () => {
  _reorderedImages([...images.filter(el => el?.id)]);
  _isEnabledReordering(true);
};

const cancelReordering = () => {
  _isEnabledReordering(false);
  _reorderedImages([...images?.filter(el => el?.id)]);
};

const saveReordering = () => {
  const reordered = reorderedImages?.filter((el, i) => i !== el?.order);
  if (reordered.length === 0) return;

  setLoader(`${t('gallery.reordering')}...`);
  const reorderArray = reorderedImages?.map((_, index) => ({ ..._, index, main: index === 0 ?
true : false }));
  category === 'gallery' ? reorderGallery(reorderArray) : reorderDamageGallery(reorderArray);

```

```

    setIsScrollEnabled(true);
  };

  const goBack = () => navigation.goBack();

  const enabledThreeSixty = useMemo(() => permissions?.includes('motork360'), [permissions]);

  return (
    <Gallery
      goBack={goBack}
      setMain={setMain}
      addPress={addPress}
      deleteOne={deleteOne}
      selectMini={selectMini}
      onSelection={onSelection}
      selectImage={selectImage}
      onDragStart={onDragStart}
      onDragRelease={onDragRelease}
      changeCategory={changeCategory}
      deleteMultiple={deleteMultiple}
      deleteAnyThreeSixty={deleteAnyThreeSixty}
      cancelSelection={cancelSelection}

      startReordering={startReordering}
      cancelReordering={cancelReordering}
      saveReordering={saveReordering}
      isEnableReordering={isEnableReordering}
      reorderedImages={reorderedImages}

      idx={idx}
      hash={hash}
      images={images}
      vehicle={vehicle}
      category={category}
      pureClosed={pureClosed}
      pureOpened={pureOpened}
      isPanorama={isPanorama}
      pureImages={pureImages}
      is360Viewer={is360Viewer}
      selectedImages={selectedImages}
      isScrollEnabled={isScrollEnabled}
      enabledSelection={enabledSelection}
      enabledThreeSixty={enabledThreeSixty}
    />
  );
};

```

```

const mapStateToProps = (state) => ({
  permissions: state.permissions.permissions,
  vehicle: getVehicleWithsyncQ(state),
  hash: state.hash.hash,
});

const mapDispatchToProps = (dispatch) => ({
  setMainImage: (payload) => dispatch(actions.setMainImageAction(payload)),
  deleteImages: (payload) => dispatch(actions.deleteImagesFromVehicleAction(payload)),
  openPopup: (payload) => dispatch(actions.openPopupAction(payload)),
  closePopup: () => dispatch(actions.closePopupAction()),
  deleteThreeSixty: (payload) => dispatch(actions.deleteThreeSixtyAction(payload)),
  deleteDamageImages: (payload) => dispatch(actions.deleteDamageImagesAction(payload)),

  setLoader: (title) => dispatch(actions.setLoaderAction(title)),
  getPermissions: () => dispatch(actions.getPermissionsAction()),
  reorderGallery: (imgs) => dispatch(actions.reorderGalleryAction(imgs)),
  reorderDamageGallery: (imgs) => dispatch(actions.reoderDamageGalleryAction(imgs)),
});

export default connect(mapStateToProps, mapDispatchToProps)(GalleryScreen);

```

Б.3. Файл Gallery/Module.tsx

```

import React, { useState } from 'react';
import {
  View,
  Text,
  Image,
  Platform,
  Dimensions,
  ScrollView,
  TouchableOpacity,
} from 'react-native';
import FastImage from 'react-native-fast-image';
import LinearGradient from 'react-native-linear-gradient';
import PanoramaView from "@lightbase/react-native-panorama-view";
import Icon from 'react-native-vector-icons/Ionicons';
// import CheckBox from '@react-native-community/checkbox';
import Triangle from 'react-native-triangle';
import { useTranslation } from 'react-i18next';
import _ from 'lodash';

```

```

import CheckBox from 'src/components/Custom/CheckBox';

import { DraggableGrid } from './Drag';

import EmptyImage from './VehicleDetails/EmptyImage';
import PlateNumber from 'src/components/PlateNumber';
import DefaultHeader from 'src/navigation/components/DefaultHeader';
import Image360Viewer from 'src/components/Vizualizer360';
import Spinner from 'src/components/Custom/Spinner';

import { CATEGORIES, CATEGORIES_WITHOUT_THREE_SIXTY } from
'src/constants/gallery';
import addTestIdentifiers from 'src/utls/addTestIdentifiers';

import styles from './styles';

const Gallery = (props) => {
  const {
    idx,
    hash,
    images,
    vehicle,
    category,
    isPanorama,
    pureClosed,
    pureOpened,
    is360Viewer,
    selectedImages,
    isScrollEnabled,
    enabledSelection,
    enabledThreeSixty,

    startReordering,
    cancelReordering,
    saveReordering,
    isEnableReordering,
    reorderedImages,
    cancelSelection,

    goBack,
    setMain,
    addPress,
    deleteOne,
    selectMini,
  } = props;

```

```

onSelection,
selectImage,
onDragStart,
onDragRelease,
changeCategory,
deleteMultiple,
deleteAnyThirtySixty,
} : {
  idx: number,
  images: Array<any>,
  vehicle: Object,
  category: string,
  isPanorama: boolean,
  pureClosed: Array<any>,
  pureOpened: Array<any>,
  is360Viewer: boolean,
  selectedImages: Array<string>,
  isScrollEnabled: boolean,
  enabledSelection: boolean,
  enabledThirtySixty: boolean,

  startReordering: () => Function,
  cancelReordering: () => Function,
  saveReordering: () => Function,
  isEnabledReordering: Function,
  reorderedImages: Array<any>,

  goBack: Function,
  setMain: Function,
  addPress: Function,
  deleteOne: Function,
  selectMini: Function,
  onSelection: Function,
  selectImage: Function,
  onDragStart: Function,
  onDragRelease: Function,
  changeCategory: Function,
  deleteMultiple: Function,
  deleteAnyThirtySixty: Function,
} = props;

const { t } = useTranslation();

// const renderThirtySixtyPreview = () => {
//   const { thirtySixty } = vehicle;
//   const threeSimages = [];

```

```

// for (let type in threeSixty) {
//   if (threeSixty[type] && threeSixty[type].length > 0) threeSimages.push({ uri:
threeSixty[type]?.uri, type });
// }
// };

const threeSixtyBorder = () => {
  return (
    <LinearGradient { ...addTestIdifiers('360RotateNote') } colors={['rgba(0, 0, 0, 0.5)',
'rgba(0, 0, 0, 0.0001)']} style={styles.gradient}>
      <FastImage source={require('src/assets/images/drag_to_rotate.png')}
style={styles.dragIcon}/>
    </LinearGradient>
  );
};

const _selector = () => (
  <View style={styles.selectorRow}>
    {
      (enabledThreeSixty ? CATEGORIES :
CATEGORIES_WITHOUT_THREE_SIXTY)?.map(_ => (
        <TouchableOpacity
          {...addTestIdifiers(`${_.category} TabButton`)}
          activeOpacity={0.8}
          disabled={category === _.category}
          onPress={() => changeCategory(_.category)}
          style={[styles.selector, category === _.category && (styles.selectedSelector)]}
        >
          <Text style={[styles.selectorLabel, category === _.category &&
(styles.selectedSelectorLabel)]}> { t(`gallery.${_.category}`) } </Text>
        </TouchableOpacity>
      ))
    }
  </View>
);

const panoramaImage = () => {
  if (!images[idx]?.id) {
    <View>
      { loading() }
      <FastImage source={{ uri: images[idx]?.url }} style={[styles.bigImage]}/>
    </View>
  };
};

return (

```

```

<View style={styles.bigImage}>
  { renderThressSixtyDeleteWatermark() }
  <PanoramaView
    style={{ flex: 1, zIndex: 999999999}}
    dimensions={{ width: Dimensions.get('window').width * .85, height: 215, }}
    enableTouchTracking={true}
    imageUrl={images[idx]?.url}
    inputType="mono"
  />
</View>
);
};

const renderThressSixtyDeleteWatermark = () => (
  <TouchableOpacity
    {...addTestIdIdentifiers('deleteOnePhotoButton')}
    activeOpacity={0.8}
    onPress={deleteAnyThreeSixty}
    style={styles.deleteImageView}
  >
    <Image source={require('src/assets/images/gallery_trash.png')}
    style={styles.deleteImageIcon}/>
    <Text style={styles.deleteImageLabel}>{t('gallery.delete')}</Text>
  </TouchableOpacity>
);

const selectedImage = () => {
  if (images?.filter(img => img?.id)?.length === 0) return <EmptyImage hideCount={true}/>
  const isImageSyncQed = !images[idx]?.id;

  const renderMainWatermark = () => !enabledSelection && !isEnabledReordering &&
  !isImageSyncQed && category === 'gallery' && !images[idx]?.main && (
    <TouchableOpacity
      {...addTestIdIdentifiers('mainImageButton')}
      activeOpacity={0.8}
      disabled={!images[idx]?.id || images[idx]?.main}
      onPress={setMain}
      style={styles.mainImageView}
    >
      { /* <Image source={require('src/assets/images/gallery_main_image_watermark.png')}
      style={styles.mainImageWatermark}/> */ }
      <Image source={require('src/assets/images/main_image_flag_icon.png')}
      style={styles.mainImageWatermark}/>
      <Text style={styles.mainImageLabel}>{ images[idx]?.main && category === 'gallery' ?
      t('gallery.mainImage') : t('gallery.setMainImage')} </Text>
    </TouchableOpacity>
  )
);

```

```

);

const renderDeleteWatermark = () => !enabledSelection && !isEnableReordering &&
!isImageSyncQed && (
  <TouchableOpacity
    {...addTestIdIdentifiers('deleteOnePhotoButton')}
    activeOpacity={0.8}
    onPress={() => deleteOne(images[idx]?.id)}
    style={styles.deleteImageView}
  >
    <Image source={require('src/assets/images/gallery_trash.png')}
style={styles.deleteImageIcon}/>
    <Text style={styles.deleteImageLabel}>{t('gallery.delete')}</Text>
  </TouchableOpacity>
);

return (
  <View>
    <FastImage source={{ uri: isEnableReordering ? reorderedImages[idx]?.url :
images[idx]?.url }} style={[styles.bigImage, (isImageSyncQed && styles.syncQedPhoto) ]}/>
    { renderMainWatermark() }
    { renderDeleteWatermark() }
  </View>
);
};

const renderSyncLoader = (img) => {
  // if (is360Viewer && images[idx]) {
    if (img?.tags?.includes('exteriorClosed') || img?.tags?.includes('exteriorOpened')) {
      const currentImages = img?.tags?.includes('exteriorClosed') ? pureClosed : pureOpened;

      if (!currentImages || currentImages.length === 0) return null;
      const isUploaded = _.every(currentImages, (image: any) => _.has(image, 'id'));
      const progressLength = currentImages?.filter(img => img.id).length;
      const finishedProgress = currentImages.length;
      const progress = Math.floor((progressLength / finishedProgress) * 100);

      if (!isUploaded) {
        return (
          <View style={styles.loaderLayout}>
            <Spinner progress={progress} size={25}/>
          </View>
        );
      }
    }

    return null;
  }
};

```

```

};

if (!img?.id) {
  return (
    <View style={styles.loaderLayout}>
      <Spinner size={15}/>
    </View>
  );
}
}

const _bottomReorderingRow = () => (
  <View style={styles.photosHeader}>
    <Text {...addTestIdentifiers('photosCounter', true)} style={styles.photosCount}>{
images?.length } {`${t('gallery.photo')}${images?.length > 0 ? t('gallery.s') : ''}`</Text>
    <View style={{ flexDirection: 'row' }}>
      <TouchableOpacity
        {...addTestIdentifiers('saveReorderButton')}
        disabled={reorderedImages?.filter((el, i) => i !== el?.order).length === 0}
        activeOpacity={0.7}
        onPress={saveReordering}
        style={styles.selectTouch}
      >
        <Text style={[styles.activeSelectTouchLabel, (reorderedImages?.filter((el, i) => i !==
el?.order).length === 0) && (styles.inactiveSelectTouchLabel)]}>{t('gallery.save')}</Text>
      </TouchableOpacity>
      <TouchableOpacity
        {...addTestIdentifiers('cancelReorderButton')}
        onPress={cancelReordering}
        style={{ marginLeft: 15 }}
      >
        <Text style={[styles.activeSelectTouchLabel]}>{t('gallery.cancel')}</Text>
      </TouchableOpacity>
    </View>
  </View>
);

const deleteButtonRow = () => enabledSelection && (
  <TouchableOpacity
    {...addTestIdentifiers('deleteMultipleDeleteButton')}
    disabled={selectedImages?.length === 0}
    activeOpacity={0.7}
    onPress={deleteMultiple}
    style={styles.selectTouch}
  >

```

```

    <Text style={[styles.activeSelectTouchLabel, { marginRight: 15, color: '#EB4B5C', },
(selectedImages.length === 0) &&
(styles.inactiveSelectTouchLabel)]}>{t('gallery.delete')}

```

```

    </View>
  </View>
);

const renderItem = (_, i) => (
  <View>
    <TouchableOpacity
      {...addTestIdentifiers('imageButton')}
      // disabled={!enabledSelection && idx === i}
      disabled={true}
      activeOpacity={0.8}
      // onPress={enabledSelection ? () => _.id && selectImage(_.id) : () => selectMini(i)}
      // style={idx === i && (styles.selectedMiniImage)}
    >
      {_.main && category === 'gallery' && (
        <View style={styles.triangle}>
          <Triangle
            width={21}
            height={21}
            color={'black'}
            direction={'up-left'}
          />
        </View>
      )}
    </TouchableOpacity>
    <FastImage source={{ uri: _.url }} style={[styles.miniImage, idx === i &&
      (styles.selectedMiniImage), _.main && category === 'gallery' &&
      (styles.blackBorderMiniImage) ]}/>
    { renderSyncLoader(_) }
    { renderReorderingIcon() }
  </View>
);

const renderReorderingIcon = () => (
  <Image source={require('src/assets/images/reordering_icon_for_images.png')} style={{ width:
    16, height: 16, position: 'absolute', left: 7, bottom: 10, zIndex: 10 }}/>
);

```

```

const draggableImages = () => {
  if (reorderedImages.length === 0) return null;
  // return null;
  return (
    <DraggableGrid
      style={styles.allImagesContainer}
      enabledSelection={enabledSelection}
      numColumns={4}
      selectImage={selectImage}
      selectMini={selectMini}
      renderItem={renderItem}
      data={reorderedImages}
      // synQPhotos={}
      onDragStart={onDragStart}
      onDragRelease={onDragRelease}
      renderPlus={() => null}
    />
  );
};

```

```

const _allImages = () => (
  <View style={styles.allImagesContainer}>
    { images?.length > 0 && images?.filter(el => enabledSelection ? el?.id : true)?.map((_, i) =>
      (
        <TouchableOpacity
          {...addTestIdentifiers('imageButton')}
          // disabled={!enabledSelection && idx === i}
          disabled={category === 'threeSixty' && idx === i}
          activeOpacity={0.8}
          onPress={enabledSelection ? () => _.id && selectImage(_.id) : () => selectMini(i)}
          onLongPress={() => _.id && category !== 'threeSixty' && !enabledSelection ?
onSelection(_.id) : null}
        >
          { _.main && category === 'gallery' && (
            <View style={styles.triangle}>
              <Triangle
                width={21}
                height={21}
                color={'black'}
                direction={'up-left'}
              />
            </View>
          )
        }
      ) /* {

```

```

enabledSelection && _.id && (
  <View {...addTestIdentifiers('checkbox')} style={styles.checkboxView}>
    <CheckBox
      value={selectedImages.includes(_.id)}
      // onChange={() => selectImage (_.id)}
      lineWidth={1.0}
      style={styles.checkbox}
      boxType='square'
      // tintColor={Platform.OS === 'ios' ? '#DCDCDC' : { true: '#DCDCDC', false:
'#DCDCDC' }}
      tintColor={Platform.OS === 'ios' ? '#C8C8CD' : { true: '#C8C8CD', false:
'#C8C8CD' }}
      onFillColor='#ffffff'
      onTintColor='#DCDCDC'
      onCheckColor='#090911'
      onAnimationType='one-stroke'
      offAnimationType='one-stroke'
    />
  </View>
)
} */}
{
  enabledSelection && _.id && (
    <CheckBox
      value={selectedImages.includes(_.id)}
      containerStyles={styles.checkboxView}
    />
  )
}
  <FastImage source={{ uri: _.url }} style={[styles.miniImage, idx === i &&
(styles.selectedMiniImage), _.main && category === 'gallery' &&
(styles.blackBorderMiniImage), !_.id && (styles.syncQedPhoto)]/>
    { renderSyncLoader(_) }
  </TouchableOpacity>
)
)}
{ renderPlusButton() }
</View>
);

const renderPlusButton = () => {
  if (category === 'threeSixty') {
    return images.length < 3 && (
      <TouchableOpacity
        {...addTestIdentifiers('addButton')}
        activeOpacity={0.8}

```

```

        onPress={addPress}
        style={styles.addView}
      >
        { /* <Icon name='add-circle' color='#9F9F9F' size={24}/> */ }
        <Icon name='add-circle' color='#231F20' size={30}/>
      </TouchableOpacity>
    );
  }

  return !enabledSelection && (
    <TouchableOpacity
      {...addTestIdIdentifiers('addButton')}
      activeOpacity={0.8}
      onPress={addPress}
      style={styles.addView}
    >
      { /* <Icon name='add-circle' color='#9F9F9F' size={24}/> */ }
      <Icon name='add-circle' color='#231F20' size={30}/>
    </TouchableOpacity>
  )
  };

  const _deleteBottom = () => false && enabledSelection && selectedImages.length > 0 && (
    <TouchableOpacity
      {...addTestIdIdentifiers('deleteMultiplePhotoButton')}
      activeOpacity={0.8}
      onPress={deleteMultiple}
      style={styles.deleteTouch}
    >
      <Text style={styles.deleteButtonLabel}>{t('gallery.delete')}</Text>
    </TouchableOpacity>
  );

  const loading = (progress) => (
    <View style={styles.loadingView}>
      <Spinner progress={progress} size={45}/>
    </View>
  );

  const bigViewController = () => {
    if (is360Viewer && images[idx]) {
      let currentImages = images[idx]?.tags?.includes('exteriorClosed') ? pureClosed : pureOpened;
      if (!currentImages || currentImages.length === 0) {
        // if (images[idx]?.type === 'CLOSED_DOORS_360') currentImages = pureClosed;
        // if (images[idx]?.type === 'OPENED_DOORS_360') currentImages = pureOpened;
        // if (!currentImages || currentImages.length === 0) return null;

```

```

// return null;
return (
  <View>
    { renderThressSixtyDeleteWatermark() }
    <EmptyImage broken={true} hideCount={true}/>
  </View>
)
};

const isUploaded = _.every(currentImages, (image: any) => _.has(image, 'id'));

if (isUploaded) {
  return (
    <View {...addTestIdIdentifiers('selectedImageView')}>
      { /* { threeSixtyBorder() } */ }
      { renderThressSixtyDeleteWatermark() }
      <Image360Viewer
        srcset={currentImages}
        width={Dimensions.get('window').width * .85}
        height={215}
        // hash={hash}
      />
    </View>
  );
} else {
  const progressLength = currentImages?.filter(img => img.id).length;
  const finishedProgress = currentImages.length;
  const progress = Math.floor((progressLength / finishedProgress) * 100);

  return (
    <View>
      { loading(progress) }
      <FastImage source={{ uri: currentImages[0]?.url }} style={[styles.bigImage]}/>
    </View>
  );
};
};

return isPanorama ? panoramaImage() : selectedImage();
};

const calculateHeight = (length) => {
  const { height } = Dimensions.get('window');
  const imagePlusMargin = 68;
  const basicValue = 100;
  const additionalHeight = Math.ceil(length / 4) * imagePlusMargin;

```

```

    // const additionalHeightInPercentage = Math.ceil(additionalHeight /
Dimensions.get('window').height) * 100;
    console.log('additionalHeight', additionalHeight);
    // console.log('additionalHeightInPercentage', additionalHeightInPercentage);
    // console.log('basicValue + additionalHeightInPercentage;', basicValue +
additionalHeightInPercentage);
    return (height * .8) + additionalHeight;
};

const Container = category === 'threeSixty' ? View : ScrollView;

return (
  <
    <DefaultHeader title={t('gallery.gallery')} goBack={goBack}/>
    <Container
      style={
        [
          styles.scrollView,
          (category === 'threeSixty' && styles.flex)
        ]
      }
      // contentContainerStyle={isEnabledReordering ? { height: `${100 + (reorderedImages.length
* 1.5)}%` } : null}
      contentContainerStyle={isEnabledReordering ? { height:
calculateHeight(reorderedImages.length) } : null}
      // contentContainerStyle={{ borderWidth: 1, borderColor: 'red', height:
Dimensions.get('window').height }}
      scrollEnabled={isScrollEnabled}
    >
      <View style={styles.container}>
        <View style={styles.content}>
          { !vehicle?.make?.name && !vehicle?.model?.name ? null : <Text
{...addTestIdentifiers('vehicleName', true)} style={styles.makeAndModelLabel}>{
vehicle?.make?.name || " } { vehicle?.model?.name || " }</Text> }
          { !vehicle?.version?.name ? null : <Text {...addTestIdentifiers('vehicleDescription', true)}
style={styles.vehicleDescription}>{ vehicle?.version?.name || " }</Text> }
          <View style={styles.plateView}>
            <PlateNumber number={vehicle?.numberPlate}/>
          </View>
          { _selector() }
          { bigViewController() }
          { isEnabledReordering ? _bottomReorderingRow() : _bottomRow() }
          { /* { _allImages() } */ }
          { !isEnabledReordering ? _allImages() : draggableImages() }
          { _deleteBottom() }
        </View>
      </View>
    </Container>
  </

```

```

    </>
  );
};

```

```
export default Gallery;
```

Б.4. Файл `utils/api.tsx`

```

import { getGenericPassword } from 'react-native-keychain';
import * as effects from 'redux-saga/effects';

import api from './axios';
import * as actions from 'src/redux/actions';
import * as endpoints from 'src/constants/endpoints';
import { authInstance, setAuthHeader, resetAuthHeader } from 'src/utils/axios';

const countryIdSelector = state => state.settings.countryId;

export function* get(endpoint, options) {
  try {
    return yield effects.call(api.get, endpoint, options || {});
  } catch (error) {
    if (!error?.response) throw error;
    if (error?.response?.status === 401) {
      try {
        let countryId : string = yield effects.select(countryIdSelector);
        if (countryId === 'en') countryId = 'uk';

        const { username, password } = yield getGenericPassword();

        if (!username || !password) yield effects.put(actions.logoutAction());

        const { data } = yield effects.call(authInstance.post, endpoints.AUTH(countryId),
        endpoints.AUTH_REQUEST(username, password));
        const { access_token: access, refresh_token: refresh } = data;
        setAuthHeader(access);
        const { data: company } = yield effects.call(api.get, endpoints.COMPANY(countryId));
        const { id: companyId } = company[0];

        const { data: dealer } = yield effects.call(api.get, endpoints.DEALER(countryId,
        companyId));
        const { id: dealerId } = dealer[0];
        yield effects.put(actions.setTokensAction({ access, refresh, companyId, dealerId }));
        return yield effects.call(api.get, endpoint, options || {});
      } catch (e) {

```

```

        console.log(e);
        throw e;
    }
}
}
};

export function* del(endpoint, data, options) {
    try {
        return yield effects.call(api.delete, endpoint, data || {}, options || {});
    } catch (error) {
        if (!error?.response) throw error;
        if (error?.response?.status === 401) {
            try {
                let countryId : string = yield effects.select(countryIdSelector);
                if (countryId === 'en') countryId = 'uk';

                const { username, password } = yield getGenericPassword();

                if (!username || !password) yield effects.put(actions.logoutAction());

                const { data } = yield effects.call(authInstance.post, endpoints.AUTH(countryId),
                endpoints.AUTH_REQUEST(username, password));
                const { access_token: access, refresh_token: refresh } = data;
                setAuthHeader(access);
                const { data: company } = yield effects.call(api.get, endpoints.COMPANY(countryId));
                const { id: companyId } = company[0];

                const { data: dealer } = yield effects.call(api.get, endpoints.DEALER(countryId,
                companyId));
                const { id: dealerId } = dealer[0];
                yield effects.put(actions.setTokensAction({ access, refresh, companyId, dealerId }));
                return yield effects.call(api.delete, endpoint, data || {}, options || {});
            } catch (e) {
                console.log(e);
                throw e;
            }
        }
    }
}

export function* post(endpoint, data, options) {
    try {
        return yield effects.call(api.post, endpoint, data, options || {});
    } catch (error) {

```

```

console.log('check', error)
if (!error?.response) throw error;
if (error?.response?.status === 401) {
  try {
    let countryId : string = yield effects.select(countryIdSelector);
    if (countryId === 'en') countryId = 'uk';
    const { username, password } = yield getGenericPassword();

    if (!username || !password) yield effects.put(actions.logoutAction());

    const { data: authData } = yield effects.call(authInstance.post, endpoints.AUTH(countryId),
endpoints.AUTH_REQUEST(username, password));
    const { access_token: access, refresh_token: refresh } = authData;
    setAuthHeader(access);
    const { data: company } = yield effects.call(api.get, endpoints.COMPANY(countryId));
    const { id: companyId } = company[0];

    const { data: dealer } = yield effects.call(api.get, endpoints.DEALER(countryId,
companyId));
    const { id: dealerId } = dealer[0];
    yield effects.put(actions.setTokensAction({ access, refresh, companyId, dealerId }));
    return yield effects.call(api.post, endpoint, data, options || {});
  } catch (e) {
    console.log(e);
    throw e;
  }
} else {
  throw error;
}
}
}

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ПРОДАЖУ АВТІВОК НА ЄВРОПЕЙСЬКОМУ РИНКУ

Виконав:
Студент групи КНТ-129сп

Д.О. Колпаков

Керівник:
Доцент, к.т.н

А.Є. Казурова

Рисунок В.1 – Слайд №1

Мета, об'єкт, предмет дослідження

Об'єкт дослідження – програмні засоби для зручного продажу.

Предмет дослідження - пріоритети та досвід сучасних торговельних платформ та ринку.

Мета роботи – створення мобільного додатку для автоматизації та поліпшення продажу автомобілів на Європейському ринку з використанням нових технологій для створення зображень.

Галузь використання – продаж використаних автомобілів та від офіційних дилерів на Європейському ринку.

Рисунок В.2 – Слайд №2

Постановка завдання до роботи

Для виконання поставленої мети потрібно виконати наступні завдання:

- створити функціонал для авторизації користувачів;
- створити метод для підтримки декількох мов для Європейського ринку;
- створити метод для автоматичного та швидкого завантаження через хмарну технологію;
- створити метод для перегляду в режимі панорами
- створити метод для користування AR;
- створити метод для використання камери з підстановкою макетів у різних форматах для обоих платформ.

Рисунок В.3 – Слайд №3

Огляд існуючих фреймворків

	Flutter	React Native
<u>Мова програмування</u>	Dart	JavaScript – основний Java, Swift - додаткові
Ким створено	Google	Facebook
Перший реліз	2017 рік	2015 рік
Готові бібліотеки	Не багато бібліотек та не гнучкі	Дуже велика кількість готових рішень та велике <u>ком'юніті</u>
Екосистема	Розроблено тільки декілька пакетів	<u>Багато доступних пакетів для використання</u>
Ключова архітектура	Flutter <u>BLoC</u>	Flux та Redux

Рисунок В.4 – Слайд №4

Порівняння мов аналогів

Dart

Плюси	Мінуси
Програма з відкритим кодом	Dart базується на парадигмі (класах) одного об'єкта.
Підтримується Google і легко розгортається на Google Cloud Platform	Dart — відносно нова мова програмування для програмістів, і вона рідко використовується на ринку
Dart приблизно в два рази швидше, ніж JavaScript	Ви не можете перейменувати функцію в Dart без написання нового оператора присвоєння.
Dart — це безпечна для типу мова програмування, яку можна компілювати за допомогою компіляторів AOT і JIT	Інтернет-ресурси Dart обмежені, і знайти відповіді на труднощі важко.
Dart — це гнучка мова програмування, яку можна використовувати в різних завданнях	
Dart — це мова сценаріїв, подібна до Javascript, і її легко вивчити, якщо ви вже знаєте Javascript	

Рисунок В.5 – Слайд №5

Порівняння мов аналогів

JavaScript

Плюси	Мінуси
JavaScript — це мова сценаріїв, яка використовується як у веб-додатках, так і в мобільних додатках.	У JavaScript є бібліотеки, які не дуже досконалі.
JavaScript можна використовувати як для переднього, так і для бекенда, що дозволяє йому працювати на будь-якому пристрої.	Через динамічний характер мови програмісти схильні до значних помилок.
Спільнота JavaScript є важливою, і багато чудових фреймворків доступні в Інтернеті.	Нові рамки надходять через регулярні проміжки часу, що призводить до постійних модифікацій.
Оскільки JavaScript взаємодіє з іншими мовами, його можна використовувати в різних програмах.	Безпека на стороні клієнта дуже недостатня та потрібно робити все в ручну.
JavaScript — це швидка, гнучка та легка мова сценаріїв.	Цілий веб-сайт може бути зупинений через одну помилку.
	Немає підтримки багатопоточності.

Рисунок В.6 – Слайд №6

Огляд існуючих середовищ розробки

Критерії порівняння	Visual Studio Code	WebStorm
Кросплатформність	+	+
Доступність	++	+/-
Наявність автодоповнення	+	+
Наявність автоматичного закриття дужок, бракетів	+	+
Підтримка репозиторіїв	+	+
Наявність «plugin manager»	+	-
Простота встановлення	+	+
Підтримка аналізу коду	+	+
Підтримка рефакторингу коду	+	+
Зручний користувацький інтерфейс	++	+/-

Рисунок В.7 – Слайд №7

Схема архітектури

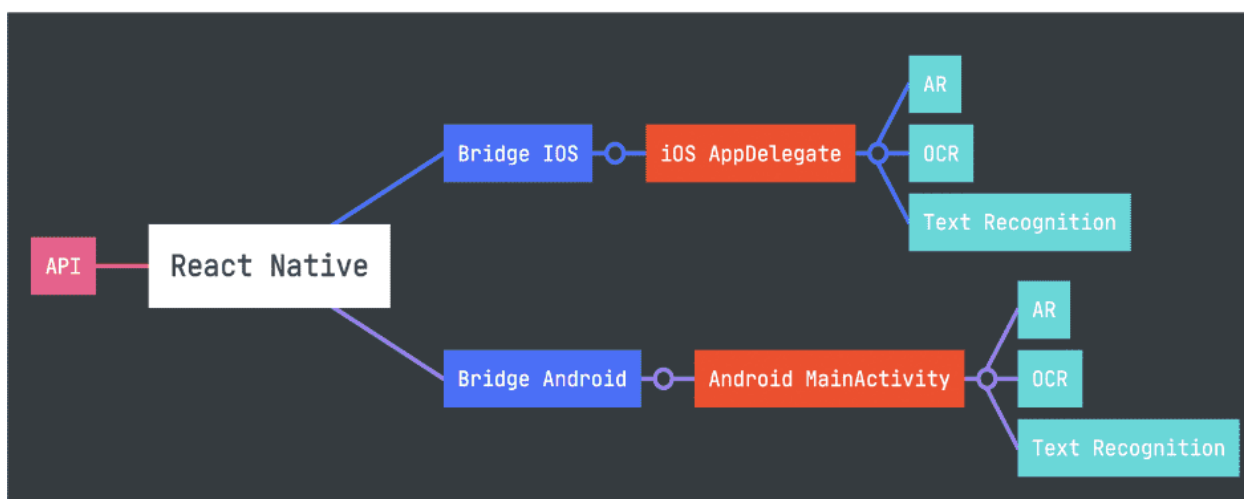


Рисунок В.8 – Слайд №8

Схема алгоритму завантаження зображень

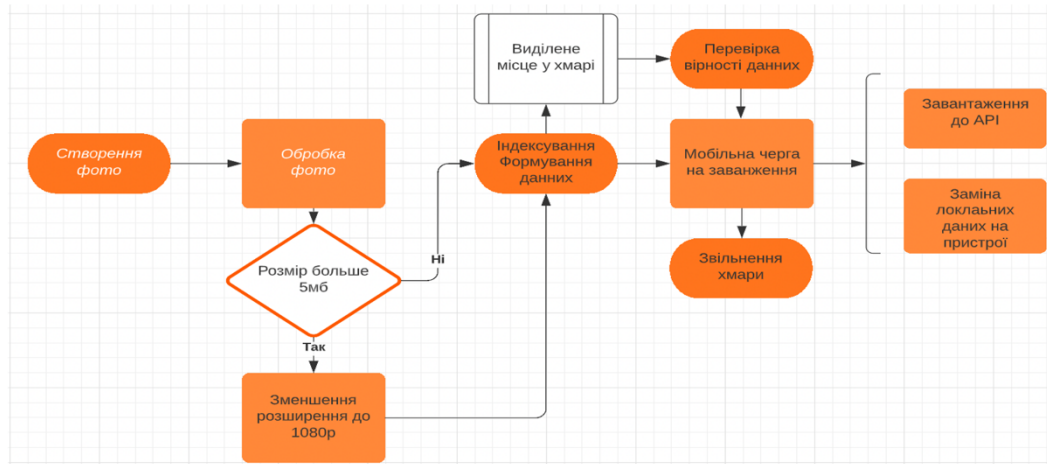


Рисунок В.9 – Слайд №9

Алгоритм асинхронної черги

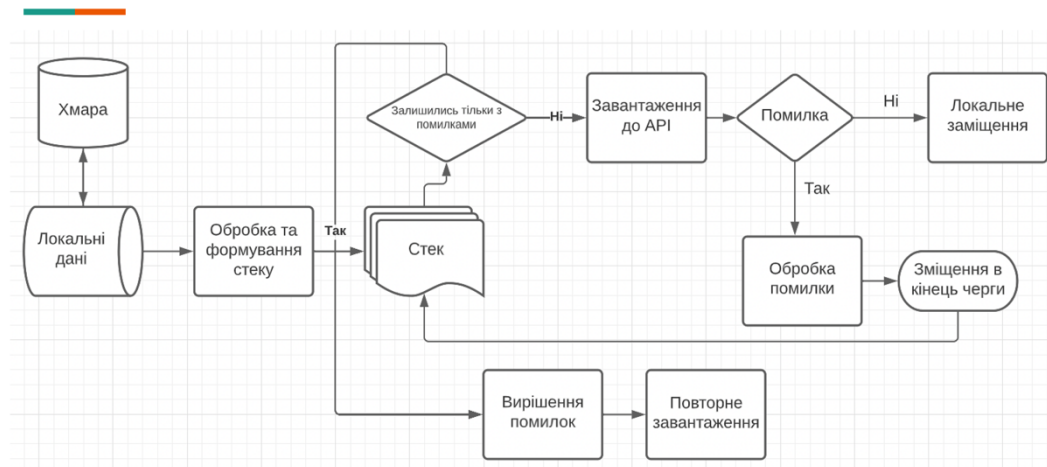


Рисунок В.10 – Слайд №10

Графічний інтерфейс додатку

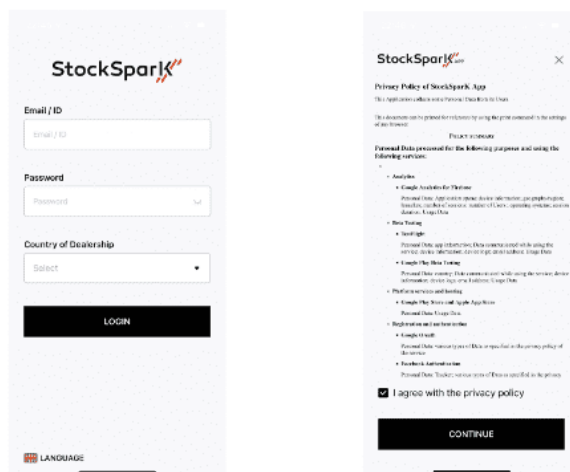


Рисунок В.11 – Слайд №11

Графічний інтерфейс додатку

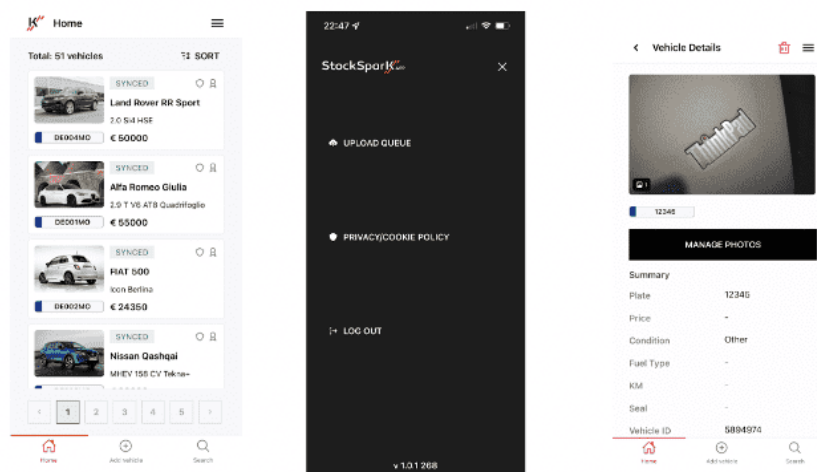


Рисунок В.12 – Слайд №12

Графічний інтерфейс додатку

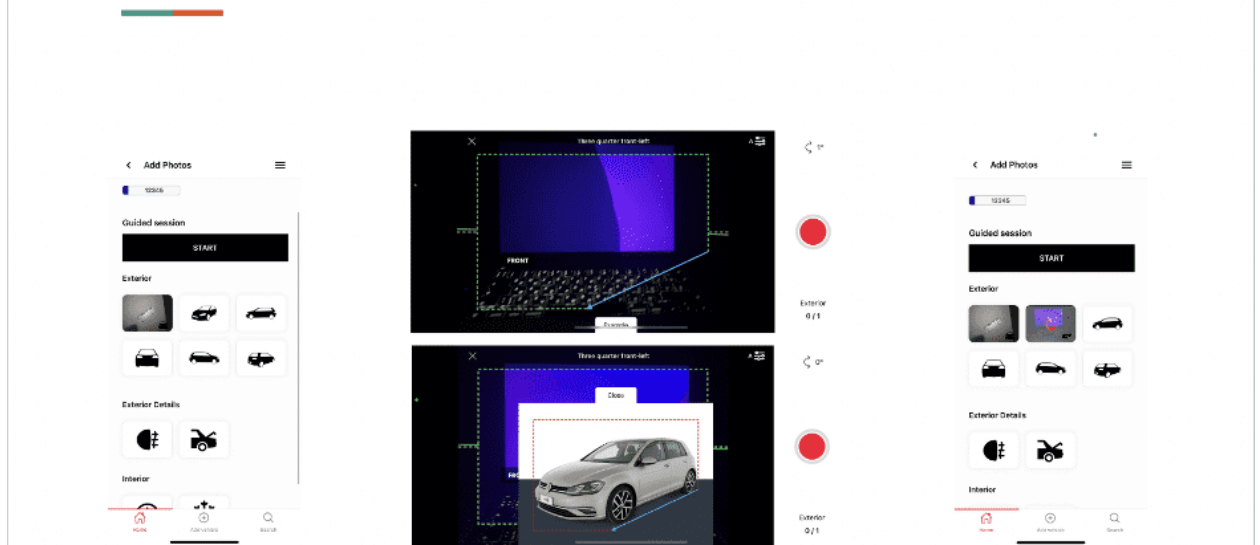


Рисунок В.13 – Слайд №13

Графічний інтерфейс додатку(AR)

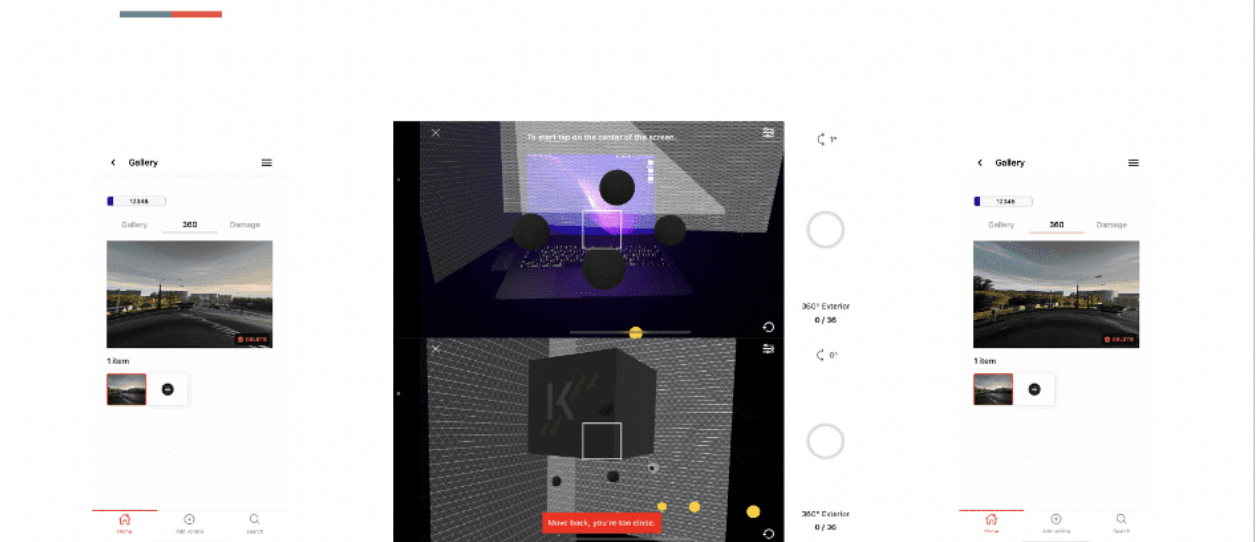


Рисунок В.14 – Слайд №14

Висновки

Під час виконання роботи було виконано дослідження предметної області, розглянуто та визначено основні поняття, які пов'язані з обраною предметною областю. Також були проаналізовані та досліджені існуючі методи, кожен з яких можливо застосовувати для вирішення поставленої задачі, було визначено їх переваги та недоліки. Був проведений порівняльний аналіз досліджених методів, які можуть бути використані для вирішення задачі автоматизації роботи з ринком автомобілів.

Для роботи було розроблено структурну схему програмного продукту. Встановлено, що для розробки розумної черги для непомітного та швидкого завантаження, краще використовувати аналог серверної архітектури - мікросервіси.

В результаті роботи було розроблено мобільний кросплатформовий додаток для продажу авто по Європі з технологіями Augmented Reality та Optical character recognition з локалізацією та вибіркою для різних країн.

Рисунок В.15 – Слайд №15