Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система керування вантажними перевезеннями. Мобільний застосунок
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-6

Падалка А. Б.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Саманцов О.О.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Падалці Артему Борисовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система керування вантажними перевезеннями. Мобільний застосунок

Затверджена наказом по університету від 20.05. 2024р. № 471 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 07.06.2024

3. Вихідні дані до роботи Розробити програмну систему керування вантажними перевезеннями, а саме такі елементи програми: Мобільний застосунок

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	08.04.2024	*виконано*
2	Створення специфікації ПЗ	15.04.2024	*виконано*
3	Проектування ПЗ	25.04.2024	*виконано*
4	Розробка ПЗ	19.05.2024	*виконано*
5	Тестування ПЗ	23.05.2024	*виконано*
6	Оформлення пояснювальної записки	24.05.2024	*виконано*
7	Підготовка презентації та доповіді	25.05.2024	*виконано*
8	Попередній захист	26.05.2024	*виконано*
9	Нормоконтроль, рецензування	02.06.2024	*виконано*
10	Здача роботи у електронний архів	03.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	07.06.2024	*виконано*

Дата видачі завдання 08 травня 2024р.

Студент _____ Падалка А. Б.
(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Саманцов О.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра: 113с., 65 рис., 10 джерел.

ЛОГІСТИКА, ПЕРЕВЕЗЕННЯ, ANDROID, KOTLIN, JETPACK COMPOSE, GOOGLE MAPS SDK, MVVM, RETROFIT, API.

Об’єкт розробки – мобільний додаток Wide Delivery для організації вантажних перевезень.

Метою практики є проведення детального аналізу ринку вантажних перевезень в Україні, виявлення проблем існуючих рішень та проектування інноваційної платформи Wide Delivery, що сприятиме ефективній, прозорій та зручній взаємодії між клієнтами та перевізниками.

Для реалізації мобільного додатку Wide Delivery будуть використані сучасні технології та архітектурні рішення, такі як Kotlin, Jetpack Compose, Retrofit, MVVM, та Google Maps SDK.

В результаті роботи було розроблено концепцію та архітектуру мобільного додатку Wide Delivery, що має потенціал стати лідером на ринку вантажних перевезень в Україні, забезпечуючи зручний, прозорий та ефективний сервіс для всіх учасників процесу.

LOGISTICS, TRANSPORT, ANDROID, KOTLIN, JETPACK COMPOSE, GOOGLE MAPS SDK, MVVM, RETROFIT, API.

The object of development is the Wide Delivery mobile application for organising freight transport.

The aim of the practice is to conduct a detailed analysis of the freight transport market in Ukraine, identify problems with existing solutions and design an innovative Wide Delivery platform that will facilitate efficient, transparent and convenient interaction between customers and carriers.

To implement the Wide Delivery mobile application, we will use modern technologies and architectural solutions such as Kotlin, Jetpack Compose, Retrofit, MVVM, and Google Maps SDK.

As a result of the work, the concept and architecture of the Wide Delivery mobile application was developed, which has the potential to become a leader in the freight market in Ukraine, providing convenient, transparent and efficient service for all participants in the process.

Я, Падалка Артем Борисович, студент гр. ПЗПІ-20-6, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система керування вантажними перевезеннями. Mobile застосунок», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений із діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Сучасний ритм життя вимагає швидких та ефективних рішень для вирішення повсякденних задач. Однією з таких задач є перевезення вантажів. Традиційні методи логістики часто виявляються громіздкими, дорогими та не завжди відповідають потребам сучасного користувача.

Саме тому виникла потреба у створенні платформи Wide Delivery, яка пропонує інноваційний підхід до вантажних перевезень в межах міста або між містами за концепцією Uber.

Wide Delivery – це мобільний додаток, що поєднує замовників, яким необхідно перевезти вантаж, з водіями, що готові надати цю послугу. Додаток надає зручний та інтуїтивно зрозумілий інтерфейс для замовлення перевезень будь-яких габаритів та типу вантажу, пропонуючи гнучкий вибір часу та дати доставки.

Wide Delivery враховує потреби як замовників, так і водіїв. Замовники отримують можливість швидко та ефективно знайти перевізника, обрати оптимальний варіант доставки, враховуючи ціну та час, а також отримати додаткові послуги, такі як допомога в завантаженні та розвантаженні.

Водії, в свою чергу, отримують доступ до широкого кола замовлень, можливість оптимізувати свої маршрути та збільшити свій дохід.

Wide Delivery – це сучасне рішення для перевезення вантажів, що поєднує в собі гнучкість, доступність та інноваційні технології.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Сфера логістики та перевезень вантажів є одним з наріжних каменів сучасної економіки. Від ефективності та швидкості доставки товарів залежить успіх бізнесу, своєчасне задоволення потреб споживачів, а також функціонування багатьох галузей промисловості. Традиційні методи організації перевезень, що часто базуються на паперовій документації, ручному пошуку перевізників та відсутності можливості оперативного відстеження вантажу, все частіше не відповідають вимогам динамічного ринку. Вони характеризуються громіздкістю, високою вартістю та недостатньою гнучкістю, що створює перешкоди для розвитку бізнесу та економіки в цілому.

В Україні ринок вантажних перевезень представлений широким спектром учасників: від великих логістичних компаній, що працюють за традиційними схемами, до індивідуальних перевізників, які часто стикаються з проблемою нестабільного потоку замовлень та простою техніки. Клієнти, в свою чергу, зіштовхуються з низкою труднощів: пошук надійних перевізників, відсутність прозорості в ціноутворенні, складність оформлення замовлень та відсутність можливості відстежувати статус доставки в режимі реального часу.

Проте з розвитком мобільних технологій та появою платформ, що працюють за моделлю "Uber", відкрилися нові можливості для створення інноваційних рішень в сфері логістики. Такі платформи дозволяють об'єднати замовників та перевізників, надаючи зручний та прозорий сервіс, що вирішує багато проблем традиційних методів організації перевезень.

Платформа Wide Delivery, що розробляється, належить до нового покоління логістичних рішень. Це мобільний додаток, який поєднує замовників, яким необхідно перевезти вантаж, з водіями, що готові надати цю послугу. Аналогічно до платформ таксі, Wide Delivery забезпечує зручний та інтуїтивно зрозумілий інтерфейс для замовлення перевезень, пропонуючи гнучкий вибір часу та дати доставки, а також прозоре ціноутворення, що базується на відстані та типі вантажу.

Однією з ключових переваг Wide Delivery є вирішення проблеми "порожніх" поїздок вантажних автомобілів. Дозволяючи перевізникам отримувати замовлення на перевезення вантажів по шляху прямування або в межах певного радіусу, Wide Delivery сприяє ефективнішому використанню ресурсів, збільшенню доходу водіїв та зниженню вартості перевезень для клієнтів.

1.1.2 Аналіз конкурентів

Uklon та Bolt, відомі своїми платформами для виклику таксі, також розширили свої послуги, запропонувавши перевезення вантажів. Їх головними перевагами є швидкість замовлення та доступність. Велика кількість водіїв та простий інтерфейс дозволяють користувачам швидко знайти транспортний засіб та отримати послуги. До того ж, обидві платформи надають можливість відстежувати рух автомобіля та очікуваний час прибуття в режимі реального часу, що забезпечує прозорість та контроль. Uklon та Bolt також мають розгалужену систему оплати, пропонуючи безготівковий розрахунок за допомогою карткових систем та різних платіжних методів. Однак, обидві платформи орієнтуються на перевезення невеликих вантажів, таких як посилки або дрібні предмети, і не надають можливостей для перевезення великогабаритних вантажів.

Нова пошта - це логістична компанія, що пропонує широкий спектр послуг, включаючи доставку вантажів. Її головними перевагами є розгалужена мережа відділень по всій Україні, що спрощує отримання та відправлення вантажів, та широкий спектр послуг, від стандартної доставки до експрес-доставки, а також додаткові послуги, такі як упаковка, страхування та зберігання вантажів. Однак, Нова пошта орієнтована на великовантажні перевезення та не надає зручних інструментів для замовлення перевезень в межах міста.

Wide Delivery має потенціал стати конкурентом цим платформам, надаючи комплексний сервіс, що об'єднує їх переваги. Wide Delivery буде орієнтований на перевезення різноманітних вантажів, від невеликих посилок до великогабаритних вантажів, що робить його більш універсальним. До того ж, Wide Delivery буде надавати простий та інтуїтивно зрозумілий інтерфейс для замовлення перевезень,

а також прозоре ціноутворення, що базується на відстані, типі вантажу та додаткових послугах. Wide Delivery також буде надавати можливість відстежувати рух вантажу в режимі реального часу, пропонуючи різні варіанти доставки та додаткові послуги, такі як допомога в завантаженні/розвантаженні та страхування вантажу.

Wide Delivery може виділитися на ринку, надаючи комплексний сервіс, що об'єднує переваги конкурентів, а також фокусуючись на зручності та прозорості для клієнтів та водіїв. Wide Delivery може стати цінним рішенням для тих, хто шукає зручний та надійний спосіб перевезення будь-якого типу вантажу.

1.2 Виявлення проблем та актуалізація рішень

Ринок вантажних перевезень в Україні, попри свою важливість, стикається з рядом суттєвих проблем, що ускладнюють процес організації доставки вантажів та знижують його ефективність. Для клієнтів однією з головних перешкод є складність пошуку надійних перевізників. Замовники змушені витрачати дорогоцінний час на пошук компаній чи приватних осіб, що надають послуги перевезень, перевіряючи їх репутацію, порівнюючи ціни та умови доставки. Цей процес часто виявляється неефективним та трудомістким, змушуючи клієнтів витрачати час та ресурси на рутинні завдання замість того, щоб зосередитися на розвитку свого бізнесу.

На додаток до складнощів з пошуком перевізників, клієнти часто стикаються з непрозорим ціноутворенням. Відсутність єдиних стандартів та часті випадки завищення цін перевізниками створюють незручності для замовників, підриваючи довіру до ринку та змушуючи клієнтів постійно сумніватися в чесності пропонованих умов.

Ще однією проблемою є обмежені можливості контролю доставки. Традиційні методи логістики, що базуються на паперовій документації та телефонних дзвінках, часто не надають можливості відстежувати статус доставки вантажу в режимі реального часу. Це змушує клієнтів хвилюватися про стан свого

вантажу, збільшує ризики непередбачених затримок та ускладнює процес планування бізнес-процесів, що залежать від своєчасної доставки товарів.

Крім того, клієнти обмежені у виборі часу доставки, а також часто не мають можливості замовити додаткові послуги, такі як допомога в завантаженні/розвантаженні, страхування вантажу тощо. Це створює незручності та змушує клієнтів шукати альтернативні, часто дорожчі, рішення для задоволення своїх потреб.

Водії, в свою чергу, також стикаються з низкою проблем. Індивідуальні перевізники та невеликі транспортні компанії часто страждають від нестабільного потоку замовлень. Це призводить до простою техніки, зниження доходів та ускладнює процес планування робочого часу.

Відсутність інструментів для оптимізації маршрутів та планування перевезень призводить до неефективного використання ресурсів та збільшення витрат на паливо. Водії змушені витрачати час та зусилля на пошук вантажів, що відповідають їх можливостям та маршрутам, замість того, щоб зосередитися на виконанні замовлень та отриманні доходу.

Платформа Wide Delivery пропонує комплекс рішень для актуальних проблем ринку вантажних перевезень. Wide Delivery створює єдину платформу, що поєднує клієнтів та водіїв, надаючи зручний та інтуїтивно зрозумілий інтерфейс для замовлення перевезень та пошуку вантажів. Це забезпечує швидке та ефективне поєднання сторін, спрощуючи процес пошуку перевізників для клієнтів та гарантуючи водіям стабільний потік замовлень.

Wide Delivery вирішує проблему непрозорого ціноутворення шляхом використання алгоритмів автоматичного розрахунку вартості перевезень. Ціна базується на відстані, типі вантажу та додаткових послугах, що забезпечує прозорість та справедливість цін як для клієнтів, так і для водіїв.

Інтеграція з GPS-навігацією дозволяє Wide Delivery надавати клієнтам можливість відстежувати рух вантажу в режимі реального часу, підвищуючи рівень контролю та дозволяючи уникнути непередбачених затримок.

Wide Delivery також надає можливість вибору зручного часу доставки, а також замовлення додаткових послуг, таких як допомога в завантаженні/розвантаженні, страхування вантажу тощо, забезпечуючи гнучкість та зручність для клієнтів.

Водіям Wide Delivery пропонує інструменти для планування маршрутів та пошуку вантажів, що знаходяться по шляху прямування, сприяючи ефективному використанню ресурсів та збільшенню доходів. Система рейтингу та відгуків дозволяє клієнтам та водіям оцінювати один одного та залишати відгуки, що сприяє підвищенню якості послуг та формуванню довірчих відносин на платформі.

В цілому, Wide Delivery пропонує комплексний підхід до вирішення проблем ринку вантажних перевезень, створюючи зручний, прозорий та ефективний сервіс як для клієнтів, так і для водіїв.

1.3 Постановка задачі

Створення платформи Wide Delivery ставить перед розробниками амбітну задачу – створити інноваційний сервіс для організації вантажних перевезень, який не просто вирішить існуючі проблеми ринку, але й задасть нові стандарти якості, зручності та прозорості. Wide Delivery має стати справжнім проривом у сфері логістики, надаючи клієнтам та водіям інструменти, які зроблять процес доставки вантажів максимально ефективним, надійним та комфортним.

Для досягнення цієї мети необхідно розробити комплексне рішення, що охопить усі аспекти взаємодії між замовниками та перевізниками. В основі платформи лежатиме зручний та інтуїтивно зрозумілий мобільний додаток. Додаток повинен бути доступним для широкого кола користувачів, забезпечуючи простий та швидкий процес замовлення перевезень для клієнтів, а також пошуку вантажів для водіїв. Важливо, щоб інтерфейс додатку був адаптивним та зручним для використання на різних мобільних пристроях, забезпечуючи максимальний комфорт для користувачів.

Прозорість – один з ключових принципів Wide Delivery. Система повинна використовувати алгоритми автоматичного розрахунку вартості перевезень, що

базуються на об’єктивних параметрах – відстані, типі вантажу та додаткових послугах. Ціна повинна бути чітко відображена клієнту до оформлення замовлення, виключаючи будь-які приховані платежі та неприємні сюрпризи.

Для усунення невизначеності та забезпечення клієнтам повного контролю над процесом доставки, Wide Delivery повинен надавати можливість відстежувати статус доставки вантажу в режимі реального часу. Інтеграція з GPS-навігацією дозволить клієнтам бачити точне місцезнаходження свого вантажу та очікуваний час прибуття, що підвищить рівень довіри до сервісу та дозволить уникнути непередбачених затримок. Водії, в свою чергу, зможуть оперативно повідомляти клієнтів про своє місцезнаходження та час прибуття, що спростить комунікацію та зробить процес доставки більш прозорим.

Wide Delivery повинен бути гнучким сервісом, здатним задовольнити різноманітні потреби клієнтів. Платформа повинна надавати можливість вибору зручного часу доставки, а також замовлення додаткових послуг, таких як допомога в завантаженні/розвантаженні, страхування вантажу тощо. Широкий спектр доступних опцій зробить сервіс привабливим для різних категорій клієнтів та допоможе завоювати лідерські позиції на ринку.

Для водіїв Wide Delivery стане незамінним інструментом для ефективної роботи. Платформа повинна надавати водіям доступ до інструментів, що дозволять їм планувати маршрути, враховуючи наявність вантажу, що знаходиться по шляху прямування. Це допоможе ефективно використовувати ресурси, зменшити пробіг без вантажу та збільшити доходи, що є ключовим фактором для залучення та утримання водіїв на платформі.

Система рейтингу та відгуків стане важливим елементом Wide Delivery, дозволяючи клієнтам та водіям оцінювати один одного та залишати відгуки. Це сприятиме підвищенню якості послуг, формуванню довіри та створенню репутації як для водіїв, так і для клієнтів.

Безпека – фундаментальний принцип, який має лежати в основі Wide Delivery. Платформа повинна забезпечувати безпеку даних користувачів, захист від

недобросовісних дій та гарантувати клієнтам надійність перевезень та відповідальність водіїв за стан вантажу.

Важливо, щоб Wide Delivery був масштабованим рішенням, здатним обслуговувати зростаючу кількість користувачів та замовлень. Платформа повинна бути гнучкою та адаптивною до змін ринку та потреб користувачів, забезпечуючи можливість розширення функціоналу та впровадження нових можливостей в майбутньому.

1.3.1 Цільова аудиторія

Цільова аудиторія Wide Delivery - це широкий спектр людей, що потребують послуг вантажних перевезень. До неї входять як звичайні люди, так і компанії.

Серед приватних осіб це можуть бути люди, яким потрібно перевезти особисті речі, меблі, будівельні матеріали, або ж доставити покупки з інтернет-магазинів. Також, це можуть бути фрілансери та представники малого бізнесу, які займаються доставкою товарів і потребують зручного способу перевезення своїх товарів до клієнтів, до складів або до поштових відділень.

Серед компаній цільовою аудиторією Wide Delivery є малі та середні підприємства, які потребують доставки товарів, сировини або обладнання, а також великі компанії, що використовують Wide Delivery для доставки товарів клієнтам, переміщення товарно-матеріальних цінностей між складами та філіями.

Wide Delivery буде цінним рішенням для власників транспортних засобів, які мають власний транспорт і бажають отримати додатковий дохід, забезпечуючи доставку вантажів. Крім того, додаток буде корисним для фізичних осіб та компаній, що потребують перевезення вантажів в межах міста або між містами.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Постановка мети

Метою виконання кваліфікаційної роботи є створення мобільного додатку Wide Delivery, який спростить процес організації вантажних перевезень в Україні. Додаток повинен стати зручним інструментом як для клієнтів, які замовляють перевезення, так і для водіїв, які їх виконують.

Wide Delivery має вирішити низку проблем, що існують на ринку вантажних перевезень: складність пошуку надійних перевізників, непрозоре ціноутворення, відсутність можливості відстежувати статус доставки в режимі реального часу, обмежений вибір часу доставки та відсутність додаткових послуг для клієнтів, а також нестабільний потік замовлень, відсутність інструментів для оптимізації маршрутів та планування перевезень для водіїв.

Додаток повинен забезпечити простий та безпечний процес реєстрації та авторизації, щоб користувачі могли легко створювати облікові записи та входити в систему, використовуючи різні методи авторизації. Інтерфейс для клієнтів повинен бути інтуїтивно зрозумілим і дозволяти легко вказати всі необхідні деталі замовлення, такі як тип вантажу, його габарити та вага, місце відправлення та призначення, бажаний час доставки, а також обрати додаткові послуги. Wide Delivery має автоматично розраховувати вартість доставки, враховуючи доступних водіїв, їх рейтинг, час та вартість доставки, надаючи клієнту кілька варіантів виконання замовлення.

Водії також повинні мати зручний доступ до актуальних замовлень в межах вибраного радіусу пошуку або вздовж запланованого маршруту. Додаток повинен надавати їм детальну інформацію про кожне замовлення. Крім того, Wide Delivery повинен впровадити систему відстеження доставки в режимі реального часу, щоб клієнт та водій могли бачити поточне місцезнаходження вантажу на карті та отримувати сповіщення про зміну статусу замовлення. Додаток також повинен мати інтегровану систему комунікації, що дозволить клієнтам та водіям

зв'язуватися один з одним для уточнення деталей замовлення, узгодження часу та місця зустрічі.

Важливим аспектом є забезпечення простого та безпечного процесу оплати послуг. Клієнти повинні мати можливість оплачувати послуги зручним для них способом, використовуючи різні платіжні системи. Після виконання замовлення, Wide Delivery має запровадити систему рейтингу та відгуків, щоб водій та клієнт могли оцінити один одного та залишити відгуки про виконання замовлення.

Wide Delivery має стати зручним, інтуїтивно зрозумілим та надійним інструментом для замовлення та виконання вантажних перевезень. Його функціонал повинен задовольнити потреби як клієнтів, так і водіїв, роблячи процес доставки вантажу простим, прозорим та ефективним.

2.2 Загальний опис

Wide Delivery – це мобільний додаток для організації вантажних перевезень, що працює за принципом Uber. Додаток поєднує клієнтів, які потребують перевезення вантажів, з водіями, які готові надати цю послугу.

Wide Delivery спрощує процес замовлення та виконання перевезень, надаючи користувачам зручний та інтуїтивно зрозумілий інтерфейс.

Клієнти можуть легко створити замовлення, вказавши всі необхідні деталі: тип вантажу, габарити, місце відправлення та призначення, бажаний час доставки та додаткові послуги, такі як допомога в завантаженні/розвантаженні, страхування вантажу тощо. Система автоматично розраховує вартість доставки, пропонуючи клієнту кілька варіантів виконання замовлення, враховуючи доступних водіїв, їх рейтинг, час та вартість доставки.

Водії можуть переглядати доступні замовлення в межах вибраного радіусу пошуку або вздовж запланованого маршруту. Вони отримують детальну інформацію про кожне замовлення: тип вантажу, габарити, місце відправлення та призначення, час доставки, вартість тощо.

Wide Delivery надає можливість відстежувати рух вантажу в режимі реального часу, що дозволяє клієнтам та водіям бачити поточне місцезнаходження вантажу на карті та отримувати сповіщення про зміну статусу замовлення.

Вбудована система комунікації дозволяє клієнтам та службі підтримки зв'язуватися один з одним для вирішення тих чи інших проблем.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ

3.1 Опис ідеї та створення плану розробки

Wide Delivery - це інноваційний мобільний додаток для організації вантажних перевезень, що працює за принципом Uber. Він поєднує клієнтів, які потребують перевезення, з водіями, що готові надати цю послугу. Додаток спрощує процес пошуку перевізників, надає прозорі ціни та зручний інтерфейс для відстеження доставки.

Для клієнтів Wide Delivery пропонує швидкий і легкий спосіб замовити перевезення будь-якого типу вантажу, від невеликих посилок до великогабаритних товарів. Додаток автоматично підбирає доступних водіїв, розраховує вартість доставки, дозволяє відстежувати рух вантажу в режимі реального часу та забезпечує зручний спосіб оплати.

Для водіїв Wide Delivery допомагає знайти нові замовлення на перевезення та ефективно управляти своєю роботою. Додаток забезпечує зручний інтерфейс для пошуку замовлень, прийняття та відхилення пропозицій, спілкування з клієнтами, а також надає інформацію про маршрути доставки та час в дорозі.

План розробки включає в себе:

- аналіз ринку та конкурентів;
- створення прототипу;
- проектування архітектури;
- розробка UI-інтерфейсу;
- розробка API;
- тестування;
- випуск та підтримка.

Шлях до успіху будь-якого продукту починається з глибокого розуміння ринку та конкурентного середовища. Для Wide Delivery, інноваційного мобільного додатку для вантажних перевезень, цей етап був особливо важливим.

Український ринок вантажних перевезень - це динамічне середовище з великим потенціалом. З кожним роком зростає попит на швидкі, надійні та доступні послуги доставки, особливо на тлі стрімкого розвитку електронної комерції. Однак, поряд з цим, існують проблеми, які заважають ринку розкрити свій повний потенціал. Аналізуючи потреби клієнтів, ми з'ясували, що найбільшими "болючими точками" є складнощі з пошуком надійного перевізника, непрозоре ціноутворення та відсутність контролю над вантажем. Клієнти втомилися від довгих пошуків, прихованих платежів і невизначеності. Водночас, водії також стикаються з низкою проблем, таких як нестабільний потік замовлень, неефективні маршрути та відсутність зручних інструментів для управління роботою.

Саме глибоке розуміння потреб ринку та аналіз конкурентів дозволили Wide Delivery зайняти свою нішу та запропонувати користувачам дійсно унікальний продукт, здатний змінити правила гри на ринку вантажних перевезень.

Створення прототипу - це важливий етап в розробці будь-якого цифрового продукту, і Wide Delivery, інноваційний додаток для вантажних перевезень, не став винятком. Це як створення макету будинку перед початком будівництва: дозволяє уявити, як все буде працювати, виявити потенційні проблеми та внести необхідні корективи ще до того, як буде написано хоч рядок коду.

На цьому етапі ми зосередились на двох головних завданнях: визначенні основних функцій додатку та створенні інтерактивного прототипу. Спочатку ми чітко прописали, що саме повинні змогти робити користувачі - як клієнти, що замовляють перевезення, так і водії, що їх виконують. Для клієнтів було важливо забезпечити простий та зрозумілий процес замовлення перевезення, можливість відстежувати рух вантажу в режимі реального часу, зручну систему оплати та збереження історії замовлень. Водії ж повинні були отримати зручний інструмент для пошуку замовлень, прийняття або відхилення пропозицій, спілкування з клієнтами, а також перегляду інформації про маршрути та оплату.

Після того, як функціонал було чітко визначено, ми перейшли до створення wireframes - схематичних зображень екранів додатку. Wireframes допомогли нам

візуалізувати структуру додатку, розміщення елементів інтерфейсу та логіку переходу між екранами.

Наступним кроком стало створення інтерактивного прототипу за допомогою спеціального програмного забезпечення. Це дозволило нам "оживити" наш макет та протестувати його в дії. Ми змогли "пройтися" по всіх екранах додатку, "натиснути" на кнопки, "заповнити" форми і переконатися, що все працює логічно та зрозуміло.

Створення прототипу дозволило нам виявити та усунути низку потенційних проблем на ранніх етапах розробки, що значно зекономило час та ресурси в майбутньому. Ми переконалися, що інтерфейс Wide Delivery є інтуїтивно зрозумілим та зручним у використанні, а функціонал додатку відповідає потребам як клієнтів, так і водіїв. Цей етап став міцним фундаментом для подальшої розробки Wide Delivery, допомог нам рухатись в правильному напрямку та створити продукт, який дійсно буде корисним для користувачів.

Проектування архітектури - це як створення каркасу для нашого майбутнього будинку, додатку Wide Delivery. Від того, наскільки продуманим і надійним буде цей каркас, залежить стійкість, функціональність і масштабованість всього проекту.

На цьому етапі ми мали визначити ключові технологічні рішення, які стануть фундаментом Wide Delivery. Першим кроком був вибір мови програмування - і наш вибір упав на Kotlin[1]. Ця сучасна, лаконічна та безпечна мова ідеально підходить для розробки під Android, що є пріоритетною платформою для Wide Delivery.

Далі ми зосередились на виборі інструментів для створення користувацького інтерфейсу. Jetpack Compose[2] - сучасний фреймворк від Google, що дозволяє створювати гарні, гнучкі та легкі в підтримці інтерфейси - став очевидним вибором. Для взаємодії з серверною частиною додатку ми обрали Retrofit[3] - потужну бібліотеку, яка значно спрощує роботу з HTTP-запитами.

Важливим етапом був вибір архітектурного шаблону для додатку. Ми зупинилися на MVVM[4] (Model-View-ViewModel) - підході, який допомагає чітко розділити логіку роботи додатку, його зовнішній вигляд та дані, з якими він

працює. Це робить код більш структурованим, зрозумілим та легким в тестуванні та підтримці.

Оскільки Wide Delivery - це сервіс, який працює з даними користувачів, замовленнями та іншою важливою інформацією, ми приділили особливу увагу вибору бази даних та проектуванню API (Application Programming Interface). API - це своєрідний "міст", який дозволяє мобільному додатку спілкуватися з сервером, обмінюватися даними та виконувати різні операції.

Ми ретельно продумали всі API-запити, формати даних та заходи безпеки, щоб забезпечити надійний та безпечний обмін інформацією між додатком та сервером.

Проектування архітектури - це невидима, але надважлива частина розробки Wide Delivery. Це фундамент, на якому буде збудовано весь додаток. І ми впевнені, що обрані нами технології та рішення дозволять створити надійний, швидкий та масштабований сервіс, який задовольнить потреби найвибагливіших користувачів.

Розробка UI-інтерфейсу - це як створення інтер'єру в нашому домі Wide Delivery. Від того, наскільки він буде продуманим, зручним та привабливим, залежить, чи сподобається він "мешканцям" - нашим користувачам. Ми прагнули створити інтерфейс, який буде інтуїтивно зрозумілим і приємним у використанні, незалежно від досвіду користувача в сфері логістики чи мобільних додатків.

На початку шляху ми зосередились на створенні дизайну - візуального образу Wide Delivery. Наші дизайнери промальовували екрани додатку, експериментували з кольоровими схемами, шрифтами та елементами інтерфейсу. Ми прагнули знайти баланс між сучасним стилем, простотою та інформативністю. Зробивши акцент на зручності, ми ретельно продумали кожен елемент інтерфейсу. Кнопки, форми, списки, іконки - все було розміщено та оформлено таким чином, щоб користувачі інтуїтивно розуміли, як з ними взаємодіяти. Ми використали Jetpack Compose - сучасний інструмент від Google, який дозволяє створювати гнучкі та естетично привабливі інтерфейси. Material Design 3 - система дизайну від Google - допомогла нам досягти єдності стилю та високої якості візуального оформлення.

Але справжній UI - це не лише гарна картинка, а й продумана логіка взаємодії з користувачем. Ми працювали над тим, щоб кожен крок в додатку, від реєстрації до оформлення замовлення і відстеження доставки, був максимально простим та зрозумілим. Тестування usability дозволило нам побачити додаток очима користувачів, виявити незручні моменти та вдосконалити інтерфейс. Ми переконались, що Wide Delivery - це не лише функціональний, але й приємний у використанні додаток, який допоможе зробити процес організації вантажних перевезень легким та зручним.

Розробка API, Application Programming Interface - це як прокладання невидимих комунікацій для нашого додатку Wide Delivery, завдяки яким він може "спілкуватися" з зовнішнім світом - сервером, де зберігаються дані про користувачів, замовлення, водіїв та інша важлива інформація. API - це своєрідний "мова", який розуміють і мобільний додаток, і сервер. Завдяки цьому "спілкуванню" користувачі можуть бачити актуальну інформацію, робити замовлення, відстежувати доставку - і все це в режимі реального часу.

На етапі розробки API ми приділили особливу увагу деталям. Ми ретельно продумали кожен API-запит - своєрідний "дзвінок" з додатку на сервер з певним проханням, наприклад, отримати список доступних водіїв, оформити замовлення чи оновлення статус доставки. Для кожного запиту ми чітко визначили формат даних - як саме інформація буде "упакована" для передачі між додатком і сервером. Ми обрали JSON - універсальний та легкий для обробки формат, який розуміють різні програмні платформи. Безпека даних була для нас пріоритетом. Ми впровадили надійні механізми аутентифікації та авторизації, щоб переконатися, що до даних користувачів мають доступ лише авторизовані особи та пристрої. Всі чутливі дані, такі як паролі та платіжна інформація, передаються в зашифрованому вигляді, що запобігає їх перехопленню чи зміні. Ми постійно тестували API на протязі всього процесу розробки, імітуючи різні сценарії використання та навантаження. Це дозволило нам переконатися, що наш "міст" між додатком та сервером працює стабільно, надійно та безпечно.

Розробка API - це важливий етап, який часто залишається "за кадром", але від якого безпосередньо залежить функціональність, швидкість та безпека Wide Delivery. Ми доклали максимум зусиль, щоб наш API був надійним фундаментом для зручного та безпечного сервісу вантажних перевезень.

3.3 UML проектування ПЗ

Програмна система буде мати 2 типи користувачів – клієнт (Client) та водій (Driver). Кожен тип користувача матиме доступ до певних функцій системи

Побудуємо Use Case діаграму для більш чіткого розуміння головних процесів. На рисунку 3.1 зображено діаграму для Клієнта та Водія.

Можемо побачити з діаграми, клієнт матиме можливість користуватися наступними функціями:

– клієнт може зареєструватися в системі, надавши необхідну інформацію;

– зареєстрований клієнт може увійти в систему, використовуючи свої облікові дані;

– клієнт може змінювати інформацію в своєму профілі, включаючи ім’я, прізвище, спосіб оплати та стати водієм;

– клієнт може створити нове замовлення на перевезення, вказавши тип вантажу, вагу, додаткові опції, місце відправлення, бажаний час;

– клієнт може оплатити замовлення, використовуючи обраний спосіб оплати;

– клієнт може відстежувати рух свого вантажу в режимі реального часу;

– клієнт може звернутися до служби підтримки Wide Delivery для вирішення питань або отримання допомоги;

Також, на діаграмі зазначено можливості, що доступні водієві, а саме:

– водій може налаштувати свої переваги щодо бажаних замовлень, наприклад, вказати радіус пошуку, тип вантажу, який він готовий перевозити, тощо;

– водій може переглядати список доступних замовлень, що відповідають його налаштуванням;

– водій може прийняти замовлення, яке його зацікавило;

– водій може додати інформацію про свій транспортний засіб, включаючи бренд та модель машини, номер машини, тип машини(вантажопідйомність, габарити);

З цього функціоналу видно, що система Wide Delivery має на меті забезпечити комплексний та зручний сервіс для організації вантажних перевезень, задовольняючи потреби як клієнтів, так і водіїв. Wide Delivery прагне до максимальної простоти та зручності для клієнтів, забезпечуючи легкий і інтуїтивно зрозумілий процес замовлення перевезення. Клієнти можуть без зайвих зусиль вказати всі необхідні параметри, включаючи тип вантажу, його вагу, додаткові опції, місце відправлення та бажаний час доставки.

Wide Delivery також робить акцент на прозорості процесу доставки, надаючи клієнтам можливість відстежувати рух вантажу в режимі реального часу. Це дає клієнтам повний контроль над процесом доставки та усуває невизначеність.

Цікаво, що Wide Delivery також пропонує клієнтам, які мають власний транспортний засіб, можливість зареєструватися як водій та отримувати додатковий дохід. Це створює додаткові можливості для користувачів та розширює базу водіїв платформи.

Рисунок 3.1 – Use Case діаграма для клієнта і водія

Діаграма демонструє, що Wide Delivery надає широкий спектр функцій як для клієнтів, так і для водіїв, спрямованих на зручне та ефективне виконання вантажних перевезень.

3.4 Проектування архітектури ПЗ

Мобільний додаток Wide Delivery, що стане ключовим елементом платформи, буде розроблено для операційної системи Android, оскільки вона є найбільш розповсюдженою в Україні. Для забезпечення високої продуктивності, надійності та зручності використовуватимемо сучасні технології та архітектурні рішення.

В якості основної мови програмування оберемо Kotlin. Kotlin - це сучасна мова програмування, що поєднує в собі лаконічність, безпеку та високу продуктивність. Kotlin має повну сумісність з Java, що дозволить використовувати вже існуючі Java-бібліотеки та фреймворки.

Для створення привабливого та зручного інтерфейсу використовуватимемо Jetpack Compose, сучасний декларативний фреймворк від Google для побудови інтерфейсів користувача Android додатків. Jetpack Compose дозволяє описувати елементи інтерфейсу в декларативному стилі, що значно спрощує розробку та робить код більш читабельним. Jetpack Compose надає широкі можливості для створення динамічних та інтерактивних інтерфейсів, що дозволить реалізувати складні елементи, такі як інтерактивні карти з відображенням маршрутів доставки, анімовані переходи між екранами та інші візуальні ефекти.

Для взаємодії з API-частиною системи будемо використовувати Retrofit - потужну бібліотеку, що спрощує роботу з HTTP-запитами. Retrofit дозволяє описувати API-запити за допомогою анотацій, що робить код більш структурованим та легким для розуміння. Retrofit підтримує різні типи запитів (GET, POST, PUT, DELETE), а також автоматично конвертує дані з формату JSON в об'єкти Kotlin. Це значно спрощує процес обробки відповідей сервера та інтеграцію даних в мобільний додаток.

Для структурування коду та забезпечення його модульності буде використано архітектурний шаблон Model-View-ViewModel (MVVM). MVVM дозволяє розділити логіку роботи з даними та відображення інформації. Model відповідає за дані та бізнес-логіку. View відповідає за відображення інформації, отриманої від ViewModel, та за взаємодію з користувачем. ViewModel виступає посередником між Model та View, обробляючи дані, підготовлені Model, та надаючи їх View в зручному форматі. MVVM підвищує модульність коду, спрощує розробку, тестування та подальше розширення функціоналу.

Для відображення карт, маршрутів доставки та відстеження вантажу в режимі реального часу буде використано Google Maps SDK[5] - набір інструментів для розробників, що дозволяє інтегрувати карти Google Maps в мобільні додатки. SDK надає широкий функціонал для роботи з картами: відображення карт, побудова маршрутів, отримання інформації про об'єкти на карті, відстеження місцезнаходження користувача, а також багато іншого. Google Maps SDK допоможе клієнтам Wide Delivery бачити точне місцезнаходження свого вантажу, а водіям - планувати оптимальні маршрути та оперативно оновлювати інформацію про своє місцезнаходження.

Мобільний додаток буде розділено на модулі, що відповідають за різні аспекти функціоналу: модуль авторизації та реєстрації, модуль замовлення перевезень, модуль пошуку вантажів, модуль відстеження, модуль оплати, модуль комунікації. Такий підхід дозволить збільшити модульність коду, спростити розробку та тестування.

3.5 Проектування UI/UX

Користувацький інтерфейс (UI) та користувацький досвід (UX) мобільного додатку Wide Delivery повинні бути максимально простими, зрозумілими та зручними для всіх категорій користувачів, незалежно від їхнього досвіду користування мобільними додатками. При розробці UI/UX головний фокус буде спрямований на досягнення простоти та інтуїтивності, щоб кожен користувач, навіть той, хто вперше стикається з додатком для замовлення вантажних

перевезень, міг легко зорієнтуватися та виконати необхідні дії. Додаток повинен дозволяти користувачам швидко та легко виконувати необхідні дії, такі як створення замовлення, пошук водія, відстеження доставки, оплата послуг тощо. Дизайн додатку повинен бути сучасним, візуально привабливим та відповідати загальній стилістиці бренду Wide Delivery.

На скріншоті з Figma[7] (див. рис. 3.2), де розроблявся макет інтерфейсу, представлено основні екрани додатку: головний екран, екран входу та екран реєстрації. Головний екран зустрічає користувача інтерактивною картою, що займає більшу частину екрану. Ця карта дозволяє користувачам візуально визначити місце відправлення та призначення, просто натиснувши на потрібну точку на карті. Під картою розташовані поля вводу "Where to?" та "From", куди користувачі можуть ввести точні адреси або вибрати їх, натиснувши на карту.

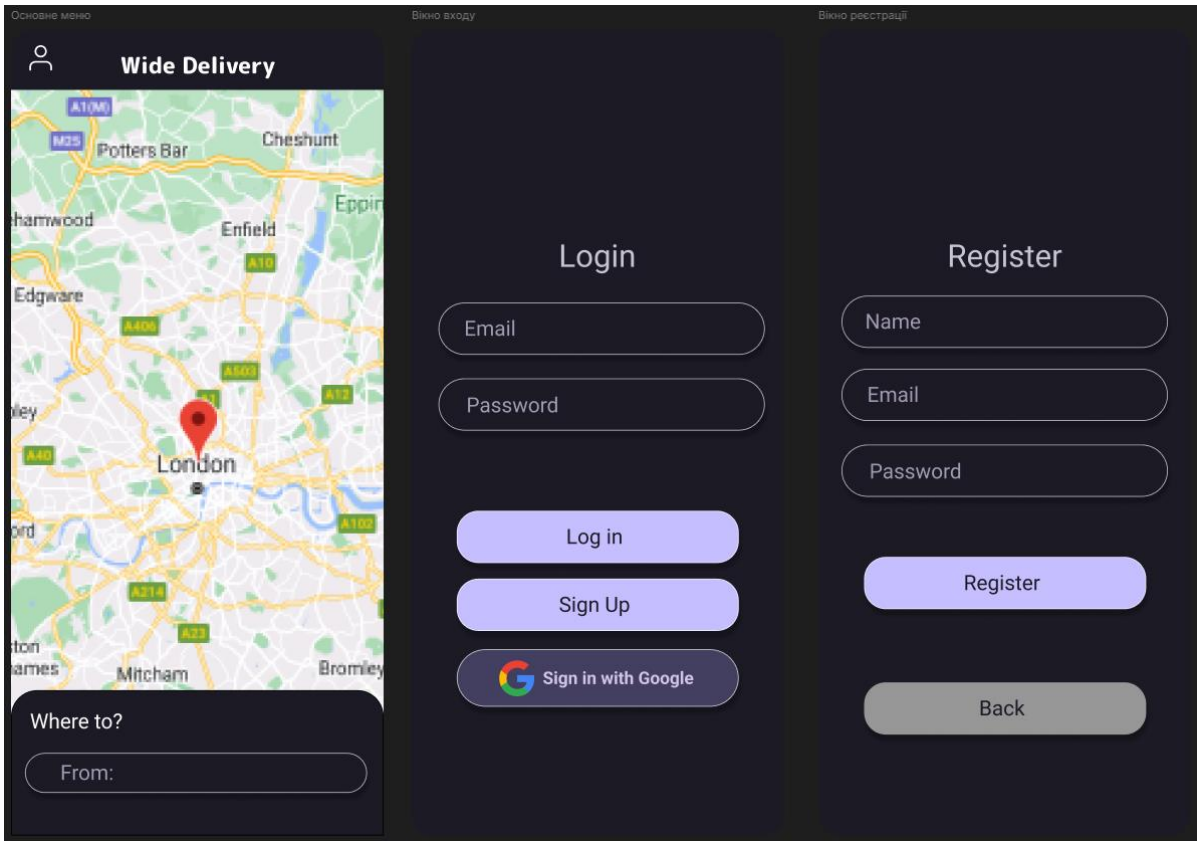


Рисунок 3.2 – Макет розроблюваного застосунку

Екран входу пропонує простий та зрозумілий спосіб входу в систему, використовуючи електронну пошту та пароль. Для тих, хто не бажає запам'ятовувати ще один логін та пароль, надається можливість входу за

допомогою облікового запису Google, що робить процес входу швидким та зручним.

Екран реєстрації дозволяє новим користувачам легко створити обліковий запис, вводячи ім'я, електронну пошту та пароль. Процес реєстрації також включає підтвердження електронної пошти для забезпечення безпеки та достовірності облікових записів.

Загальний дизайн додатку виконаний в темних тонах з акцентом на контрастні елементи. Використання темного фону зменшує навантаження на очі користувачів, особливо в умовах низької освітленості, що робить користування додатком більш комфортним. Контрастні елементи, такі як кнопки, поля вводу та текст, виділяються на фоні, що покращує читабельність та полегшує навігацію, дозволяючи користувачам швидко знаходити потрібні елементи інтерфейсу.

Під час подальшої розробки UI/UX додатку, будуть враховані принципи матеріального дизайну, розроблені компанією Google. Це забезпечить єдиний та зручний користувацький досвід на різних Android-пристроях, незалежно від їхніх характеристик та розміру екрану. Принципи матеріального дизайну включають використання простих і інтуїтивно зрозумілих анімацій, природних переходів між екранами, та чітких візуальних ієрархій, що допомагають користувачам легко орієнтуватися в додатку.

Окрім зображених на скріншоті екранів входу, реєстрації та головного екрану, будуть розроблені інші екрани та елементи інтерфейсу, що забезпечать повний функціонал Wide Delivery. Наприклад, екран створення замовлення дозволить користувачам вказати всі необхідні деталі про вантаж, такі як розміри, вага, адреси відправлення та отримання. Інтерфейс вибору типу вантажу та додаткових послуг надасть можливість вибрати спеціальні умови перевезення, наприклад, перевезення крихких або небезпечних матеріалів.

Користувачі також матимуть можливість отримати технічну підтримку безпосередньо через додаток. Для цього в інтерфейсі буде передбачена можливість переходу в Telegram-чат, де можна буде швидко зв’язатися з технічною підтримкою. Це забезпечить зручність та швидкість у вирішенні будь-яких питань

чи проблем, що можуть виникнути під час користування додатком. Користувачі зможуть отримати консультацію, допомогу з технічними аспектами та відповіді на свої запитання в режимі реального часу, що значно покращить користувацький досвід.

Розробка всіх цих екранів буде враховувати ті самі принципи простоти, зручності та привабливості, що й основні екрани, забезпечуючи послідовний та якісний користувацький досвід на всіх етапах взаємодії з додатком.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Розробка користувацьких екранів

Для реалізації мобільного додатку Wide Delivery було ретельно відібрано сучасні та ефективні технології, що дозволять створити надійний, швидкий та зручний інструмент для організації вантажних перевезень. В якості основної мови програмування було обрано Kotlin, сучасну статично типізовану мову, розроблену компанією JetBrains. Kotlin приваблює своєю лаконічністю, що дозволяє писати менше коду, досягаючи тих самих результатів, що і в Java. Kotlin також відрізняється підвищеною безпекою, запобігаючи поширеним помилкам програмування, таким як NullPointerException. Окрім цього, Kotlin демонструє високу продуктивність, що важливо для забезпечення плавної роботи мобільного додатку.

Рисунок 4.1 – Логотип мови прогрмування Kotlin

Важливою перевагою Kotlin є його повна сумісність з Java, що дозволяє використовувати вже існуючі Java-бібліотеки та фреймворки та значно пришвидшує процес розробки. Завдяки цьому, розробники Wide Delivery можуть скористатися багаторічним досвідом спільноти Java розробників та

використовувати готові рішення для різних задач, таких як робота з мережею, базами даних, графікою тощо. Вибір Kotlin сприяє створенню надійного та швидкого додатку, а також полегшує його подальшу підтримку та розвиток.

Для створення інтерфейсу користувача (UI) обрано Jetpack Compose, сучасний декларативний фреймворк від Google для побудови UI Android-додатків. Jetpack Compose революціонізує розробку інтерфейсів для Android, дозволяючи описувати елементи інтерфейсу в декларативному стилі. Це означає, що розробник зосереджується на тому, як інтерфейс повинен виглядати в залежності від поточного стану даних, а не на тому, як його потрібно оновлювати вручну. Jetpack Compose автоматично оновлює UI при зміні даних, що значно спрощує розробку та робить код більш читабельним та легким для розуміння.

Нижче наведено приклад коду, що створює елемент тексту і текстове поле. Для них надані кастомні параметри для видозмінення кольору, розміру, тощо:

```
Text(
     text = "Login",
     style = MaterialTheme.typography.headlineMedium,
     modifier = Modifier
      .padding(bottom = 16.dp)
      .align(Alignment.CenterHorizontally)

    )
    OutlinedTextField(
     value = state.email,
     onValueChange = {
      viewModel.onEmailChanged(it)
      viewModel.emailValidation(it)
     },
     label = { Text("Email") },
     modifier = Modifier.fillMaxWidth(),
     shape = RoundedCornerShape(50.dp),
     keyboardOptions = KeyboardOptions(
      keyboardType = KeyboardType.Email,
      imeAction = ImeAction.Next
     ),
     isError = !state.isEmailValid,
     supportingText = {
      if (!state.isEmailValid) {
       Text(text = state.emailErrorMessage)
      }
     }
    )
```

У результаті на екрані у користувача буде наступний вигляд елементів, що були створені вище в прикладі коду (див. рис. 4.2)

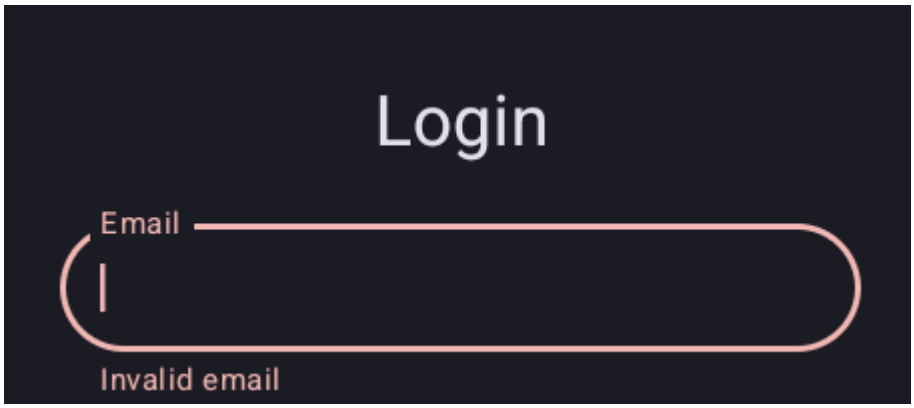


Рисунок 4.2 – Результат виконання коду

Jetpack Compose надає широкі можливості для створення динамічних та інтерактивних інтерфейсів. Завдяки цьому, розробники Wide Delivery можуть реалізувати складні елементи, такі як інтерактивна карта з відображенням маршрутів доставки, анімовані переходи між екранами та інші візуальні ефекти, що зроблять додаток привабливим та зручним для користувачів. Jetpack Compose також забезпечує високу продуктивність UI, що важливо для забезпечення плавної та приємної взаємодії з додатком.

Розглянемо створення різних екранів для користувача і водія.

Екран входу в застосунок Wide Delivery (див. рис. 4.3) розроблений з метою забезпечення простого та швидкого доступу для існуючих користувачів. Він виконаний в мінімалістичному стилі, що відповідає загальній концепції дизайну додатку. На темному фоні екрану, що зменшує навантаження на очі, розташовані контрастні елементи, які чітко виділяються та легко сприймаються.

В центрі екрану розміщено великий напис "Login", що чітко вказує на призначення екрану. Під ним розташовані два поля вводу: "Email" та "Password", куди користувач вводить свої облікові дані. Поля вводу мають чіткі межі та достатньо великий розмір, що спрощує введення даних навіть на пристроях з невеликим екраном.

Під полями вводу розміщена кнопка "Log in", виконана в контрастному кольорі, що привертає увагу та спонукає до дії. Натиснувши на цю кнопку, користувач входить в систему, якщо введені облікові дані є правильними.

Для тих, хто не бажає запам'ятовувати ще один логін та пароль, Wide Delivery пропонує альтернативний спосіб входу за допомогою облікового запису Google. Кнопка "Sign in with Google" розташована під кнопкою "Log in" і містить логотип Google, що робить її легко впізнаваною. Натиснувши на цю кнопку, користувач авторизується в системі через свій Google акаунт.

Нижче розміщена кнопка "Sign Up", яка перенаправляє користувача на екран реєстрації, якщо в нього ще немає облікового запису в Wide Delivery.

Такий дизайн екрану входу забезпечує простий та зрозумілий процес авторизації для існуючих користувачів, а також пропонує альтернативний спосіб входу за допомогою Google акаунту, що робить Wide Delivery доступним для широкої аудиторії.

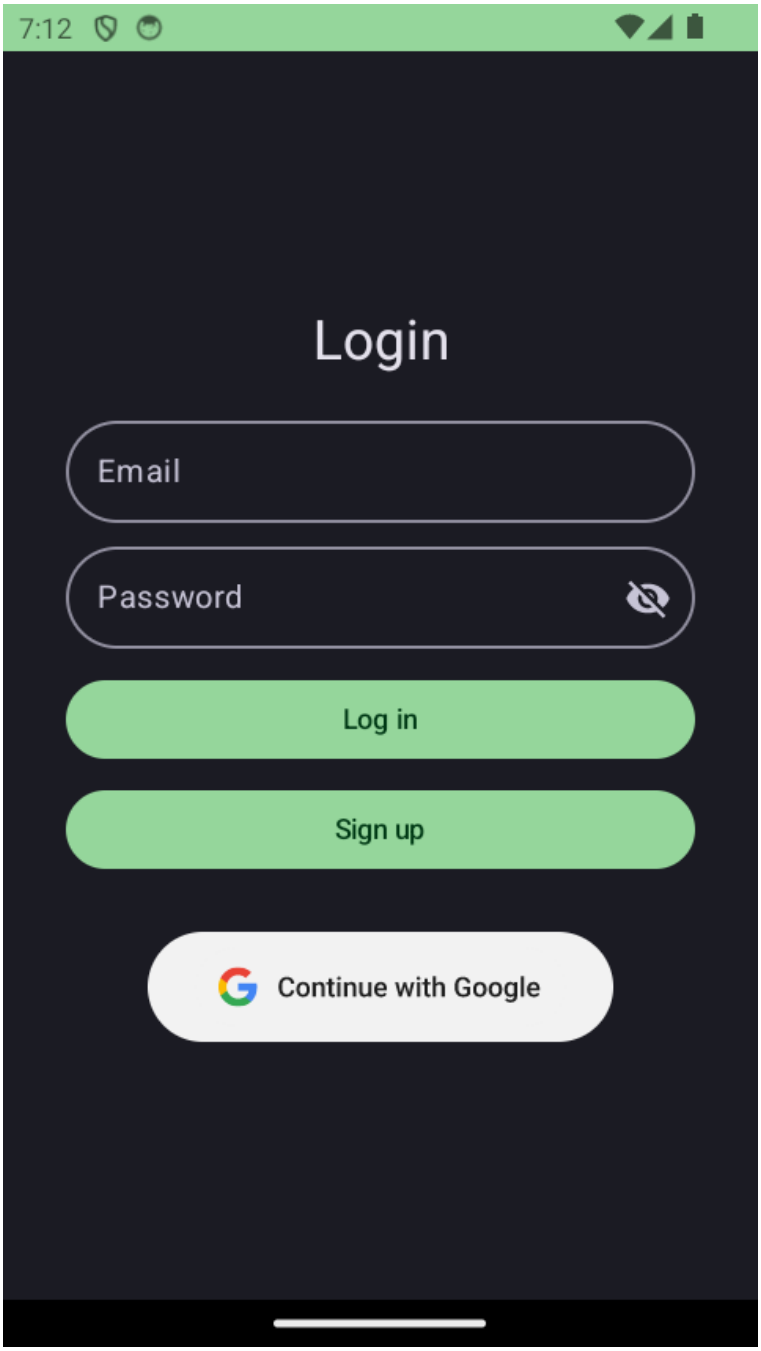


Рисунок 4.3 – Екран входу в застосунок

У разі неправильного написання пошти і/або паролю, користувачу буде виведено попередження про помилку у написанні (див. рис. 4.4).

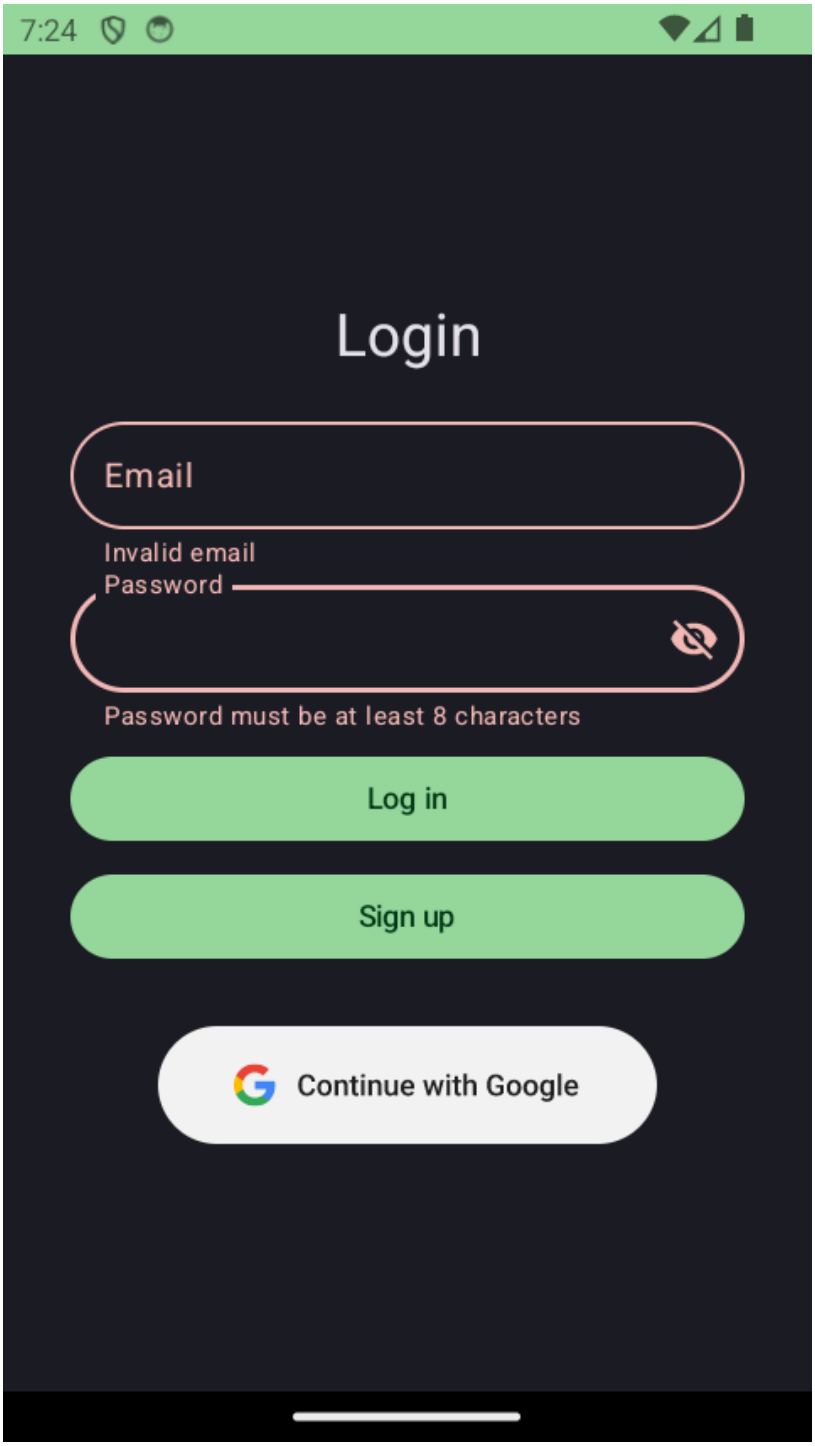


Рисунок 4.4 – Виведення попереджень про неправильність введених даних

Це зроблено за допомогою функцій isPasswordValid, що приймає значення, яке було введено в поле і повертає булеве значення, passwordValidation, що в залежності від значення, яке повертає функція isPasswordValid змінює значення state, яке впливає на стан елементів, що ми можемо побачити на екрані. Аналогічно зроблено і для валідації пошти. Код наведено нижче:

```
fun passwordValidation(password: String) {
   if (isPasswordValid(password)) {
  _state.value = state.value.copy(isPasswordValid = true,
   passwordErrorMessage = "")
        } else {
            _state.value = state.value.copy(
                isPasswordValid = false,
                passwordErrorMessage = "Password must be at least 8
   characters"
            )
        }
    }

    fun emailValidation(email: String) {
        if (isEmailValid(email)) {
            _state.value = state.value.copy(isEmailValid = true,
    emailErrorMessage = "")
        } else {
            _state.value =
                state.value.copy(isEmailValid = false,
    emailErrorMessage = "Invalid email")
        }
    }

    private fun isPasswordValid(password: String): Boolean {
        return password.length >= 8
    }

    private fun isEmailValid(email: String): Boolean {
        return
    android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()
    }
```

Екран реєстрації в додатку (див. рис. 4.5). Wide Delivery спроектований таким чином, щоб нові користувачі могли легко і швидко створити свій обліковий запис та розпочати користуватися сервісом. Як і решта екранів додатку, він виконаний в мінімалістичному стилі з темним фоном, що сприяє комфортному сприйняттю інформації.

У верхній частині екрану розміщено великий напис "Register", що одразу дає зрозуміти користувачеві призначення екрану. Під написом знаходяться три поля вводу: "Name", "Email" та "Password", куди користувач вводить необхідну інформацію для створення облікового запису. Поля вводу мають чіткі межі, достатньо великі та зрозумілі підписи, що спрощує введення інформації навіть на невеликих екранах.11.06.2024

Після заповнення всіх полів, користувач натискає кнопку "Register", розташовану в центрі екрану. Кнопка виконана в контрастному фіолетовому кольорі, що привертає увагу та спонукає до дії. Після натискання кнопки система перевіряє введені дані та створює новий обліковий запис, якщо введена інформація є коректною та відповідає вимогам системи.

У нижній частині екрану розміщена кнопка "Back", яка дозволяє користувачеві повернутися на попередній екран, якщо він передумав реєструватися або потребує додаткової інформації.

Загалом, екран реєстрації Wide Delivery пропонує простий та зрозумілий процес створення облікового запису для нових користувачів, забезпечуючи їм швидкий доступ до функціоналу додатку. Мінімалістичний дизайн та контрастні елементи роблять екран візуально привабливим та зручним для використання.

У випадку виникнення помилки під час введення даних, користувач отримає відповідне повідомлення, яке підсвічує проблемне поле червоним кольором та вказує на тип помилки. Це може бути неправильний формат електронної пошти, недостатня складність пароля або незаповнене поле. Повідомлення про помилку розміщені безпосередньо під відповідним полем, що дозволяє швидко зрозуміти проблему та виправити її.

Таким чином, екран реєстрації Wide Delivery створює сприятливі умови для швидкого та безпроблемного створення нового облікового запису, що є важливим етапом у забезпеченні позитивного користувацького досвіду. Легкість навігації, чіткі інструкції та безпека даних роблять процес реєстрації приємним та зручним для користувачів будь-якого рівня технічної підготовки.

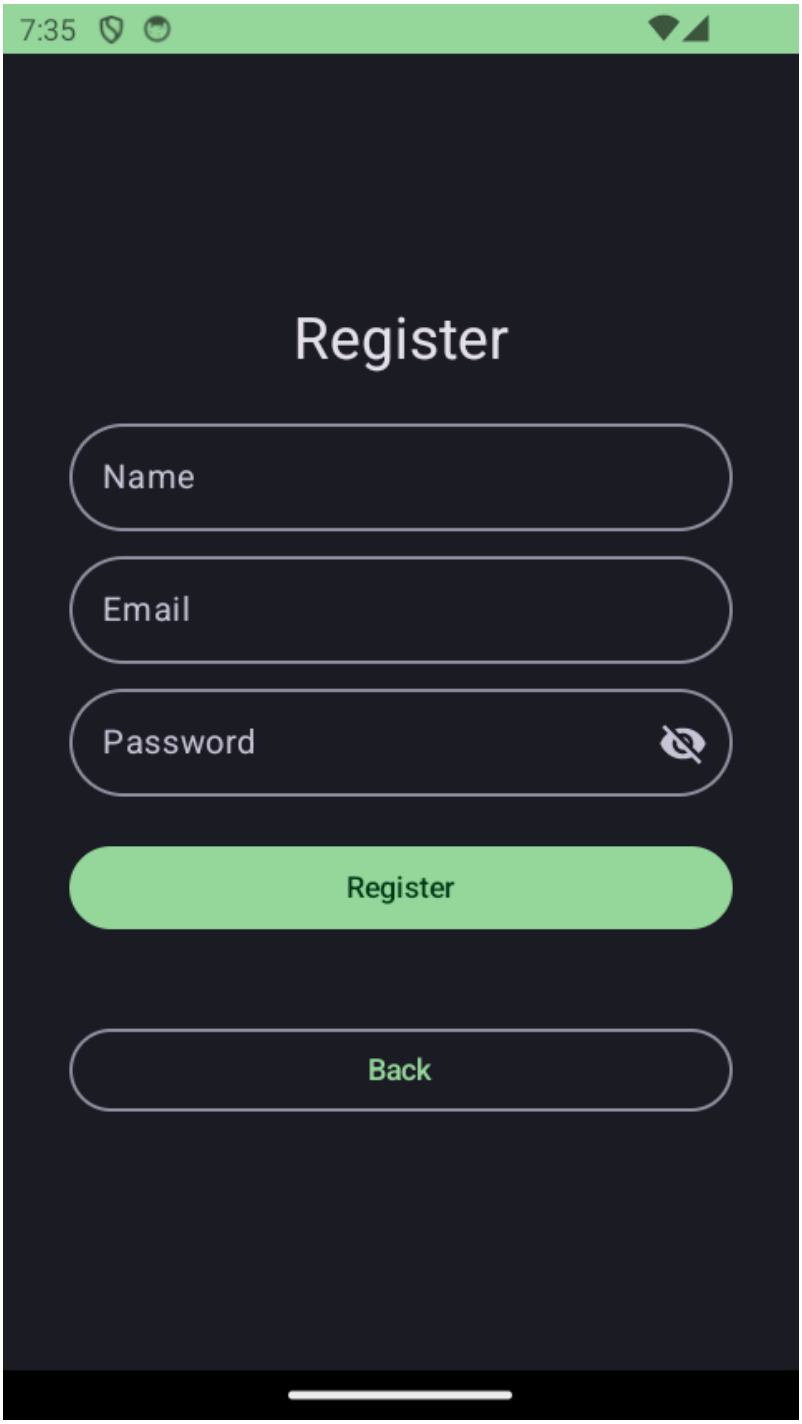


Рисунок 4.5 – Екран реєстрації користувача

Однак, розуміючи, що заповнення форм реєстрації може бути стомлюючим для користувачів, Wide Delivery пропонує альтернативний, швидкий та зручний спосіб створення облікового запису - авторизацію через Google. Замість заповнення форми реєстрації, користувач може просто натиснути на кнопку "Sign in with Google", розташовану в нижній частині екрану. Кнопка виконана в стилі Google з

його фірмовим логотипом, що робить її легко впізнаваною. Після натискання на кнопку, відкриється стандартне вікно авторизації Google, де користувач обирає свій обліковий запис Google та надає додаток Wide Delivery доступ до необхідної інформації, такої як ім'я та електронна адреса. Після успішної авторизації через Google, користувачеві не потрібно заповнювати форму реєстрації - він автоматично отримує обліковий запис в Wide Delivery та може розпочати користуватися додатком.

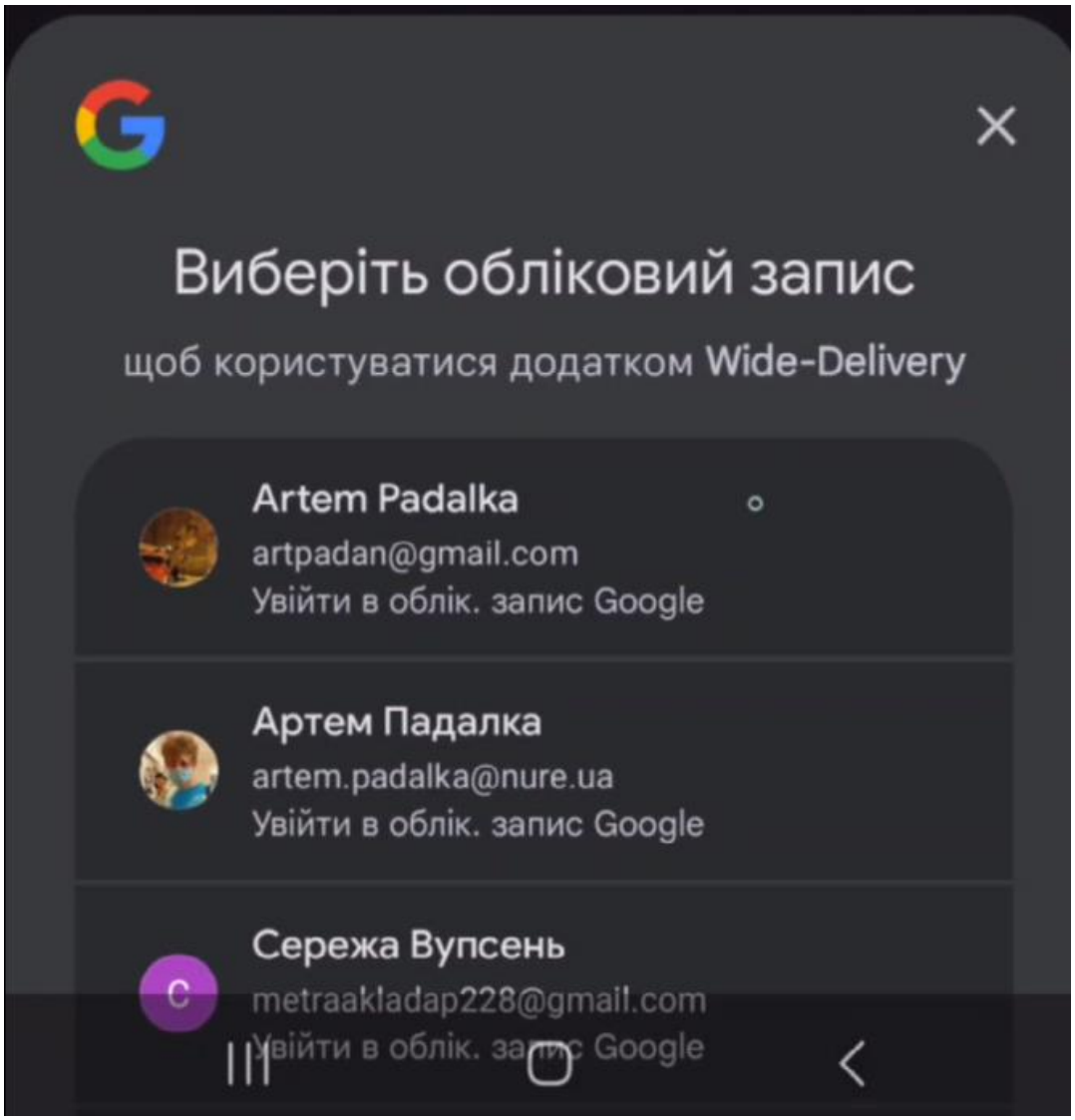


Рисунок 4.6 – Діалогове вікно з вибором аккаунту, що збережені на пристрої

Головний екран мобільного додатку Wide Delivery зустрічає користувача інтерактивною картою, яка займає майже весь простір екрану, візуально

підкреслюючи географічний характер сервісу. Ця карта - не просто статичне зображення, а потужний інструмент, що дозволяє користувачам взаємодіяти з додатком на інтуїтивно зрозумілому рівні. Користувач може переглядати карту, змінювати масштаб, переміщатися по ній, а також визначати місце відправлення та призначення вантажу, просто натиснувши на потрібні точки на карті.

Під картою розташовані важливі елементи інтерфейсу: кнопка "From" та текст "Where to?". Ці елементи призначені для введення точних адрес відправлення та призначення вантажу. Користувач може ввести адреси вручну або ж скористатися підказками, які надає додаток на основі введених даних. Після натискання на кнопку "From" користувач переходить на екран замовлення.

У верхньому лівому куті екрану розміщена іконка користувача, яка відкриває вікно профілю, де можна виконати такі операції, як: звернення до служби підтримки, вийти з застосунку, перейти на вікно водія тощо. Таке розташування меню є стандартним для Android-додатків та дозволяє зберегти чистоту головного екрану, не перевантажуючи його зайвими елементами.

Головний екран Wide Delivery (див. рис. 4.7) - це візитна картка додатку, яка створює перше враження про сервіс. Його мінімалістичний дизайн, зосереджений навколо інтерактивної карти, підкреслює простоту та зручність використання Wide Delivery. Користувач може інтуїтивно зрозуміти, як працює додаток та швидко розпочати процес замовлення вантажного перевезення.

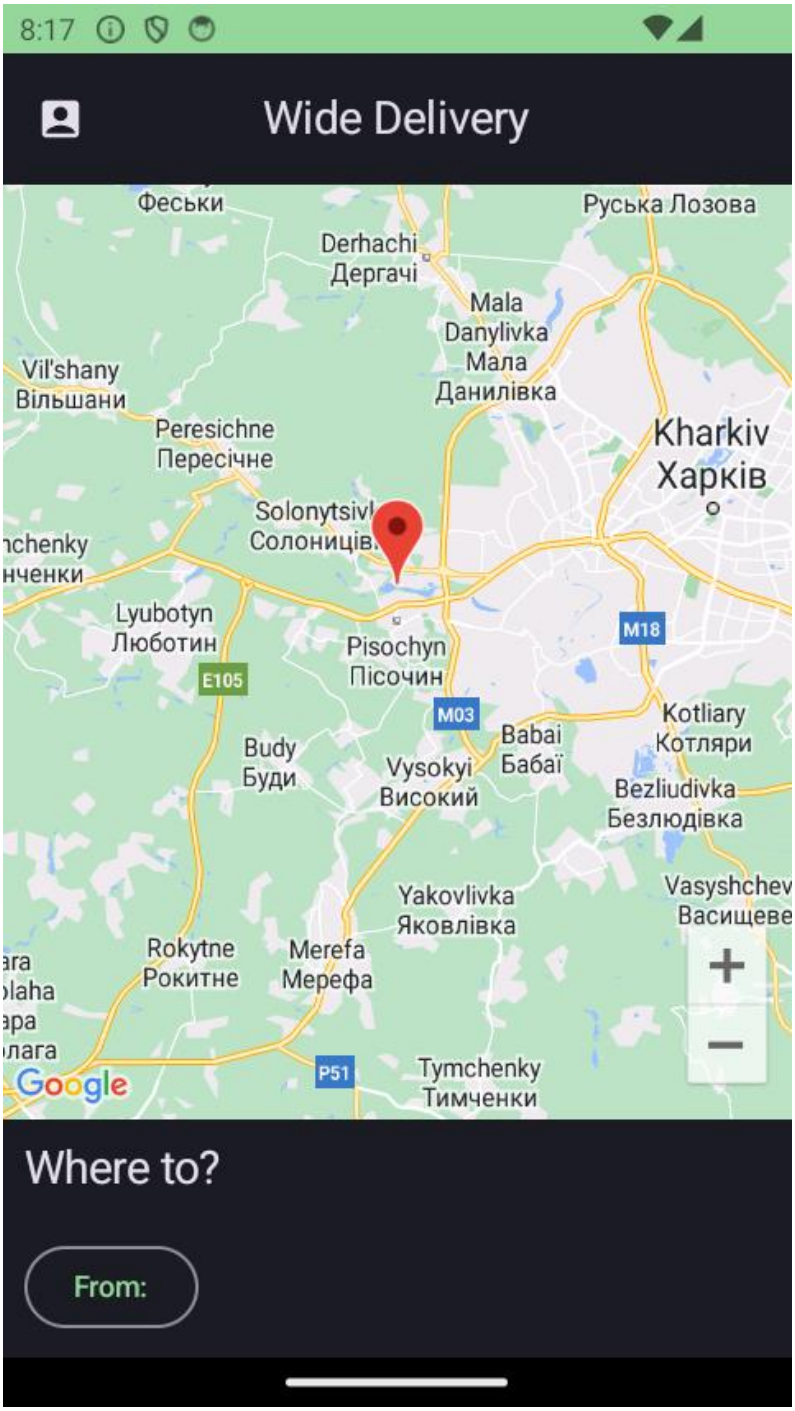


Рисунок 4.7 – Головний екран застосунку

Дуже важливою частиною застосунку є вікно профілю користувача , на якому зображено екран профілю (див. рис. 4.8) користувача мобільного додатку Wide Delivery.

Екран виконаний в лаконічному стилі з темним фоном та контрастними елементами, що відповідає загальному дизайну додатку. У верхній частині екрану, на фоні приємного зеленого кольору, розташовані іконки для швидкого доступу до

інформації про додаток, налаштувань та сповіщень. Нижче, на темному фоні, знаходиться круглий аватар користувача з зображенням, обраним під час реєстрації. Під аватаром розміщено привітання "Hello, Artem!", де ім'я користувача підставляється автоматично з його профілю.

Далі знаходиться рядок з електронною адресою користувача, що використовується для входу в додаток.

Центральну частину екрану займають чотири кнопки, кожна з яких виконана у вигляді прямокутника із закругленими кутами та має чіткий і зрозумілий напис. Перша кнопка, "Be a driver!", дозволяє користувачеві перейти до процесу реєстрації водія, якщо він бажає надавати послуги вантажного перевезення через Wide Delivery. Друга кнопка, "Back to main screen", повертає користувача на головний екран додатку з інтерактивною картою та формами для замовлення перевезення. Третя кнопка, "Delete account", надає користувачеві можливість видалити свій обліковий запис з Wide Delivery. Остання кнопка, "Sign out", дозволяє користувачеві вийти зі свого облікового запису, завершивши сеанс роботи з додатком.

Екран профілю користувача Wide Delivery надає зручний доступ до ключових функцій управління обліковим записом, дозволяючи користувачеві швидко переходити до потрібних розділів додатку та змінювати налаштування профілю.

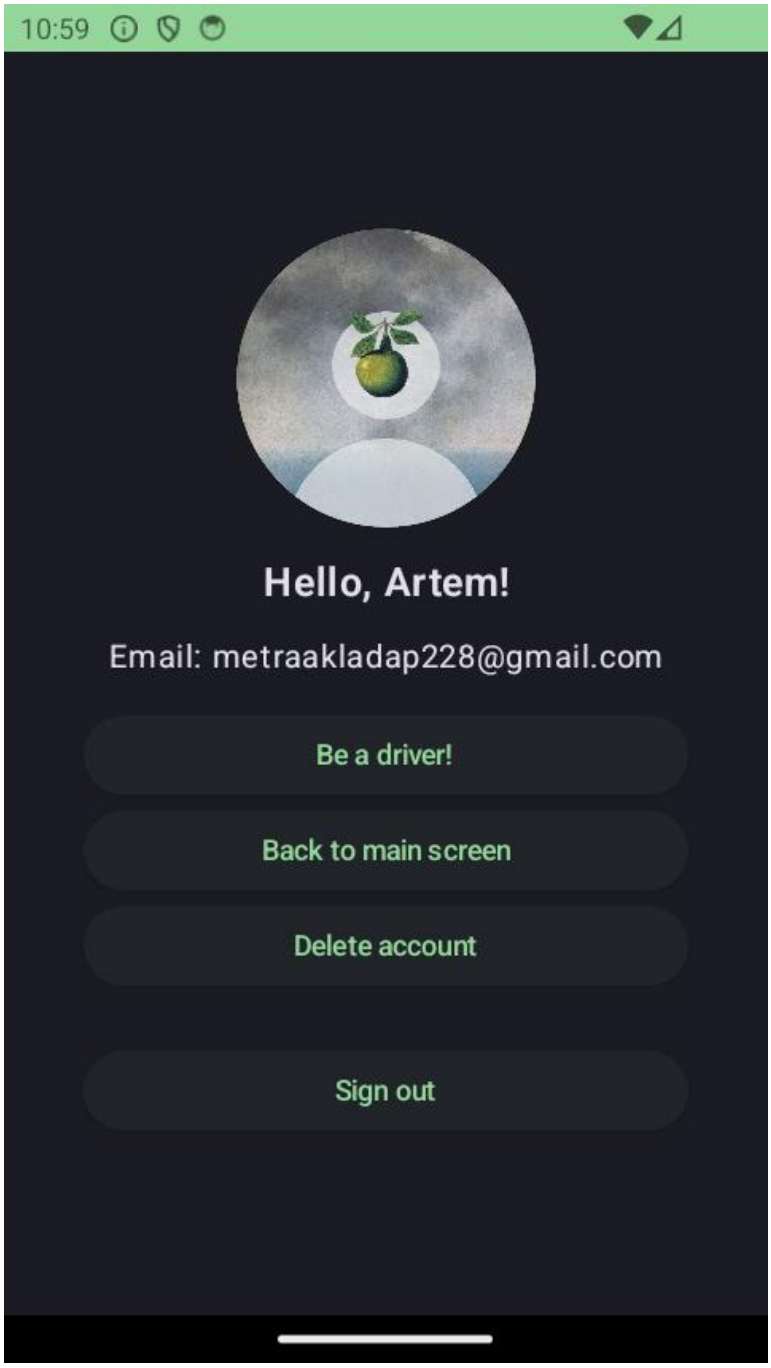


Рисунок 4.8 – Вікно профілю користувача

При створені цього вікна була задіяна мультипотоковість для для виконання асинхронних операцій, таких як отримання даних користувача з сервера. Для цього використовуються корутини, що є особливістю мови Kotlin.

У функції getUserData в ProfileViewModel ми бачимо використання viewModelScope.launch, що запускає корутину для виконання мережевого запиту в фоновому режимі. Це дозволяє не блокувати головний потік додатку та

забезпечити плавну роботу інтерфейсу, навіть якщо мережевий запит займає деякий час. Сама функція getUserData запускається під час івенту, LoadProfile. Також, використовується ще один івент для події, коли користувач натискає на кнопку Sign out. Код використання наведено нижче.

```
fun onEvent(event: ProfileEvent, navController: NavController) {
        when (event) {
            is ProfileEvent.LoadProfile -> {
                viewModelScope.launch {
                    getUserData(navController)
                }
            }

            ProfileEvent.OnSignOut -> {
                viewModelScope.launch {
                    preferenceManager.clearTokens()

navController.navigate(DestinationScreen.SignIn.route)
                }
            }
        }
    }
```

Код для кнопки Sign out, де викликається ця подія також наведено нижче.

```
ElevatedButton(
            modifier = Modifier
                .fillMaxWidth()
                .padding(end = 40.dp, start = 40.dp),
            onClick = {
                viewModel.onEvent(
                    ProfileEvent.OnSignOut,
                    navController = navController
                )
            }
        ) {
            Text(text = "Sign out")
        }
```

Завдяки архітектурі MVVM цей код легко масштабований, тож не має проблем в майбутньому додавати нові функції та обробляти нові події, зберігаючи при цьому структурованість та читабельність коду.

MVVM дозволяє чітко розділити відповідальність між різними компонентами: Model відповідає за дані, View - за відображення, а ViewModel - за

обробку логіки та взаємодію між ними. Такий підхід сприяє створенню більш модульного та гнучкого коду, що полегшує його розробку, тестування та підтримку.

У випадку з кнопкою "Sign out", View (екран профілю) відповідає лише за відображення кнопки та виклик відповідної події в ViewModel при натисканні. ViewModel, в свою чергу, обробляє цю подію, очищуючи токени авторизації та перенаправляючи користувача на екран входу. Model, в даному випадку, представлений класом PreferenceManager, що відповідає за зберігання та управління токенами.

Така чітка розподіленість відповідальності дозволяє легко змінювати та розширювати функціонал додатку, не впливаючи на інші компоненти. Наприклад, якщо в майбутньому з'явиться необхідність додати додаткові дії при виході з облікового запису, це можна зробити, просто змінивши код ViewModel, не торкаючись View або Model.

Ще один цікавий момент обробка станів екрану в функції ProfileScreen. Тут використовується observeAsState для відстеження змін стану ProfileViewModel та відображення відповідного контенту на екрані. Залежно від поточного стану (завантаження, помилка, відображення даних користувача), користувач бачить на екрані або індикатор завантаження, або повідомлення про помилку, або інформацію про свій профіль.

```
val state by viewModel.state.observeAsState(ProfileState())
when {
    state.isLoading -> {
        CircularProgressIndicator(modifier = Modifier.fillMaxSize())
    }

    state.error != null -> {
        Text(
            text = state.error ?: "Unknown Error",
            modifier = Modifier
                .fillMaxSize()
                .wrapContentSize(Alignment.Center)
        )
    }

    state.userResponse != null -> {
```

```
            ProfileContent(state.userResponse!!, navController, context,
viewModel)
        }}
```

4.2 Впровадження комунікації з сервером за допомогою API

Для взаємодії мобільного додатку Wide Delivery з серверною частиною системи було реалізовано API (Application Programming Interface) на базі протоколу REST (Representational State Transfer). Це дозволяє додатку обмінюватися даними з сервером, використовуючи стандартні HTTP-запити, такі як GET, POST, PUT та DELETE.

Для зручності роботи з API в коді додатку використовується бібліотека Retrofit. Retrofit - це потужна бібліотека, що спрощує роботу з HTTP-запитами, дозволяючи описувати API-запити за допомогою анотацій. Наприклад, для реєстрації водія використовується наступний код:

```
@POST("drivers/")
suspend fun registerDriver(
    @Header("Authorization") authHeader: String,
    @Body request: DriverRegisterRequest): Response<Unit>
```

Анотація @POST("drivers/") вказує, що цей метод відповідає за POST-запит до ендпоінту /drivers/. Анотація @Header("Authorization") вказує, що в заголовку запиту необхідно передати токен авторизації. Анотація @Body вказує, що дані запиту будуть передані в тілі запиту в форматі JSON.

Для виконання самого запиту використовується корутина (suspend функція), що дозволяє виконувати запит асинхронно, не блокуючи головний потік додатку.

```
val response =
WideDeliveryWebService.ApiService.apiService.registerDriver(
    "Bearer ${preferenceManager.getAccessToken()}",
    buildDriverRegisterRequest()
)
```

Після виконання запиту, код аналізує код відповіді сервера (response.code()) та виконує відповідні дії. Наприклад, якщо код відповіді 201 (Created), це означає, що водій успішно зареєстрований. Якщо ж код відповіді 409 (Conflict), це означає, що водій з такими даними вже існує в системі.

```
when (response.code()) {
    201 -> {
        Log.d("DriverRegistrationViewModel", "Driver registered
successfully")
        // ...
    }

    409 -> {
        Log.e("DriverRegistrationViewModel", "Driver already
registered")
        // ...
    }

    // ... обробка інших кодів відповіді
}
```

Використання Retrofit та корутин спрощує взаємодію з API, робить код більш читабельним та дозволяє ефективно обробляти мережеві запити, не впливаючи на продуктивність додатку.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розробка тестових випадків

Тестування - це як випробування на міцність для нашого додатку Wide Delivery. Ми прагнули переконатися, що він не лише гарно виглядає та має продуманий функціонал, але й працює бездоганно в будь-яких умовах. Це як перевірка будинку перед заселенням - ми хотіли бути впевнені, що в ньому комфортно та безпечно.

Процес тестування охопив всі аспекти Wide Delivery. Ми перевіряли кожен елемент користувацького інтерфейсу, переконуючись, що кнопки натискаються, форми заповнюються, а дані відображаються коректно. Наші тестувальники пройшлися по всіх можливих сценаріях використання додатку, намагаючись знайти і "зламати" його в найнесподіваніших місцях. Ми проводили функціональне тестування, щоб переконатися, що всі функції додатку - від реєстрації та оформлення замовлення до відстеження доставки та оплати - працюють коректно і відповідають заявленим вимогам.

Окрім функціонального тестування, ми також провели ретельне тестування зручності використання Wide Delivery. Адже наш додаток має бути не лише функціональним, але й інтуїтивно зрозумілим та приємним у використанні. Ми запросили представників нашої цільової аудиторії – як приватних осіб, так і представників бізнесу – щоб вони оцінили зручність навігації, зрозумілість інтерфейсу та загальне враження від взаємодії з Wide Delivery.

Особливу увагу ми приділили тестуванню ключових елементів додатку: інтерактивної карти, форми створення замовлення, системи відстеження доставки та процесу реєстрації авто. Ми прагнули переконатися, що ці елементи є зрозумілими та зручними для всіх категорій користувачів, незалежно від їх досвіду користування подібними додатками.

В таблиці 5.1 представлено повний список тест-кейсів, розроблених у процесі тестування програмного модуля.

Таблиця 5.1 – Тест-кейс №1

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №1		
Опис функції:	Реєстрація як водій і створення замовлення як клієт		
Власник тесту:	Падалка Артем Борисович		
Дата створення:	05.05.2024		
Мета тесту:	Перевірити спроможність програми виконувати основні функції		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити мобільний застосунок	Користувач відкриває мобільний застосунок	Пройдено
2	Користувач має акаунт та хоче авторизуватися	Перенаправленя на головний екран	Пройдено
Зареєструватися як водій вантажівки			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити профіль користувача	З’являється кнопка «Стати водієм»	Пройдено
2	Натиснути на кнопку «Стати водієм»	Перенаправлення на вікно реєстрації водія	Пройдено
3	Заповнити обов’язкове поле «Пошуковий радіус»	Правильно вказані дані	Пройдено
4	Заповнити обов’язкове поле «Бренд вантажівки»	Правильно вказані дані	Пройдено
5	Заповнити обов’язкове поле «Модель вантажівки»	Правильно вказані дані	Пройдено
6	Заповнити обов’язкове поле «Автомобільний номер»	Одне з обов'язкових полів було заповнено правильно та кількість символів вірна	Пройдено
7	Заповнити обов’язкове поле «Серійний номер вантажівки»	Правильно вказані дані	Пройдено
8	Заповнити обов’язкове поле «Колір вантажівки»	Правильно вказані дані	Пройдено
9	Заповнити обов’язкове поле «Вільне місце в довжину»	Вказаний коректний розмір	Пройдено
10	Заповнити обов’язкове поле «Вільне місце в ширину»	Вказаний коректний розмір	Пройдено
11	Заповнити обов’язкове поле «Вільне місце в висоту»	Вказаний коректний розмір	Пройдено
12	Вибрати чи може водій бути тим, хто переносить вантаж	Правильно вказані дані	Пройдено
13	Натиснути кнопку «Зареєструвати»	Інформація має вказані дані та очікує на підтвердження	Пройдено

Кінець таблиці 5.1

Зареєструвати замолення як клієнт			
№	Опис випадку	Очікуваний результат	Висновок
1	Настинути кнопку «Звідки їдемо»	З’являється форма для створення заявки	Пройдено
2	Заповнити обов’язкове поле «Звідки»	Вказані коректні дані	Пройдено
3	Заповнити обов’язкове поле «Куди»	Вказані коректні дані	Пройдено
4	Обрати «Тип вантажу», який необхідно перевезти	Було обрано тип вантажу	Пройдено
5	Обрате один з варіантів в обов’язковому полі «На зараз» або «На певний час»	Вибран один з варіантів	Пройдено
6	Натиснути, що потрібна допомога з вантажем	Не натиснуто	Пройдено
7	Натиснути кнопку “Створити замовлення”	Інформація має вказані дані та очікує на підтвердження	Пройдено
8	Переглянути і погодитися з оголошеною вартістю. Натиснути «Так» або «Ні»	Було погоджено оголошену вартість, натиснуто «Так»	Пройдено
9	Очікувати на прибуття водія в зазначений час	Водій прибув на зазначений час	Пройдено
Результати тестування			
Тестувальник: Падалка А. Б.	Дата прогону тесту: 05.05.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

Тестування функціональності мобільного додатку Wide Delivery проводилося систематично на кожному етапі розробки. Окрім спеціально розроблених тестових сценаріїв, описаних у таблиці 5.1, велику увагу приділено тестуванню окремих компонентів та функцій додатку.

Кожен модуль, кожна функція та навіть невеликі фрагменти коду були ретельно перевірені на коректність роботи та відповідність заданим вимогам. Такий підхід дозволив вчасно виявляти та виправляти помилки, забезпечуючи стабільну та надійну роботу Wide Delivery.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено концепцію та архітектуру інноваційної платформи Wide Delivery, що покликана вирішити актуальні проблеми ринку вантажних перевезень в Україні.

Було проведено детальний аналіз предметної області, виявлено ключові проблеми клієнтів та водіїв, а також проаналізовано існуючі рішення конкурентів. Виявлено, що існуючі платформи мають ряд недоліків, таких як орієнтація на великовантажні перевезення, обмежений функціонал, висока вартість послуг та недостатня гнучкість.

У порівнянні з ними, Wide Delivery пропонує комплексний підхід, спрямований на вирішення проблем як клієнтів, так і водіїв, надаючи зручний, прозорий та ефективний сервіс для замовлення та виконання вантажних перевезень. Платформа буде доступна у вигляді мобільного додатку для платформи Android з використанням сучасних технологій та архітектурних рішень, таких як Kotlin, Jetpack Compose, Retrofit, MVVM, та Google Maps SDK.

Використання цих технологій гарантуватиме високу продуктивність, надійність та зручність мобільного додатку. Модульна структура додатку дозволить розширювати його функціонал та адаптувати до змін ринку та потреб користувачів. Wide Delivery має потенціал стати лідером на ринку вантажних перевезень в Україні, забезпечуючи зручний, прозорий та ефективний сервіс для всіх учасників процесу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kotlin Programming Language [Електронний ресурс] – URL: https://kotlinlang.org/ (дата звернення: 11.05.2024).

2. Jetpack Compose [Електронний ресурс] – URL: https://developer.android.com/jetpack/compose (дата звернення: 10.05.2024).

3. Retrofit [Електронний ресурс] – URL: https://square.github.io/retrofit/ (дата звернення: 07.05.2024).

4. MVVM Architecture [Електронний ресурс] – URL: https://developer.android.com/jetpack/guide (дата звернення: 11.05.2024).

5. Google Maps Platform Android SDK [Електронний ресурс] – URL: https://developers.google.com/maps/documentation/android-sdk/ (дата звернення: 12.05.2024).

6. Android Developers [Електронний ресурс] – URL: https://developer.android.com/ (дата звернення: 14.05.2024).

7. Figma [Електронний ресурс] – URL: https://www.figma.com/ (дата звернення: 15.05.2024).

8. REST API Design [Електронний ресурс] – URL: https://restfulapi.net/ (дата звернення: 16.05.2024).

9. The Geography of Transport Systems [Електронний ресурс] – URL: https://transportgeography.org/ (дата звернення: 18.05.2024).

10. Logistics and Supply Chain Management [Електронний ресурс] – URL: https://www.investopedia.com/terms/l/logistics.asp (дата звернення: 19.05.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

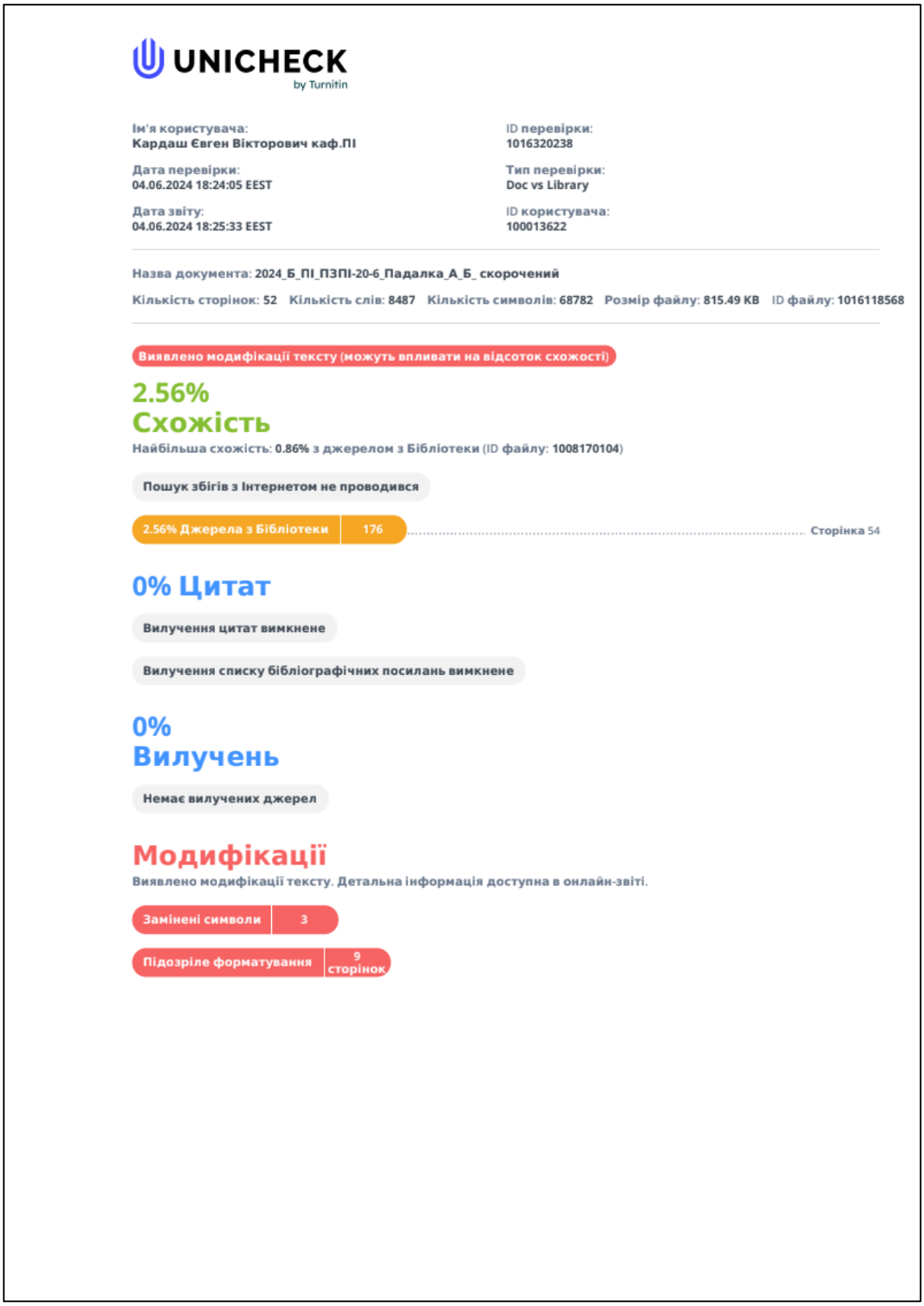


Рисунок А.1 – Результаті перевірки на унікальність тексту в базі ХНУРЕ

ДОДАТОК Б

Слайди презентації

Рисунок Б.1 – Слайд 1

Рисунок Б.2 – Слайд 2

Рисунок Б.3 – Слайд 3

Рисунок Б.4 – Слайд 4

Рисунок Б.5 – Слайд 5

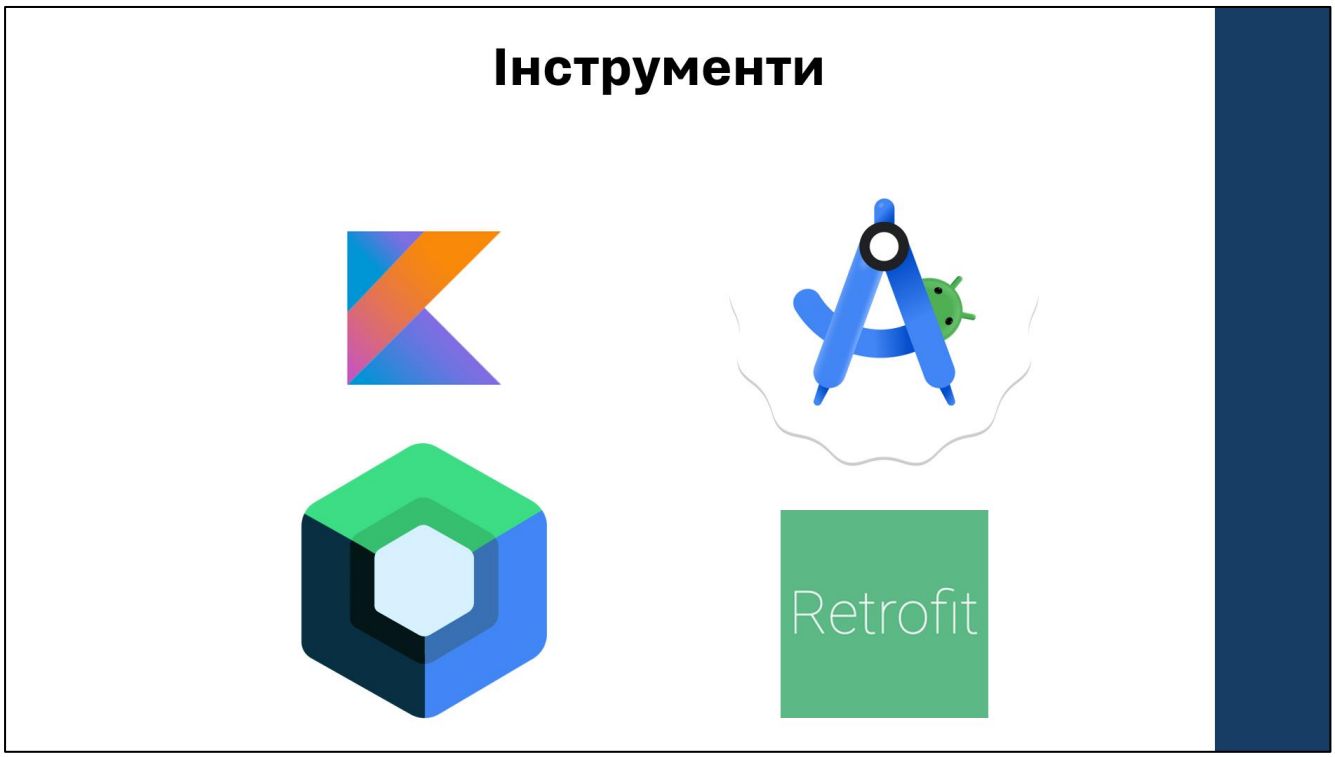


Рисунок Б.6 – Слайд 6

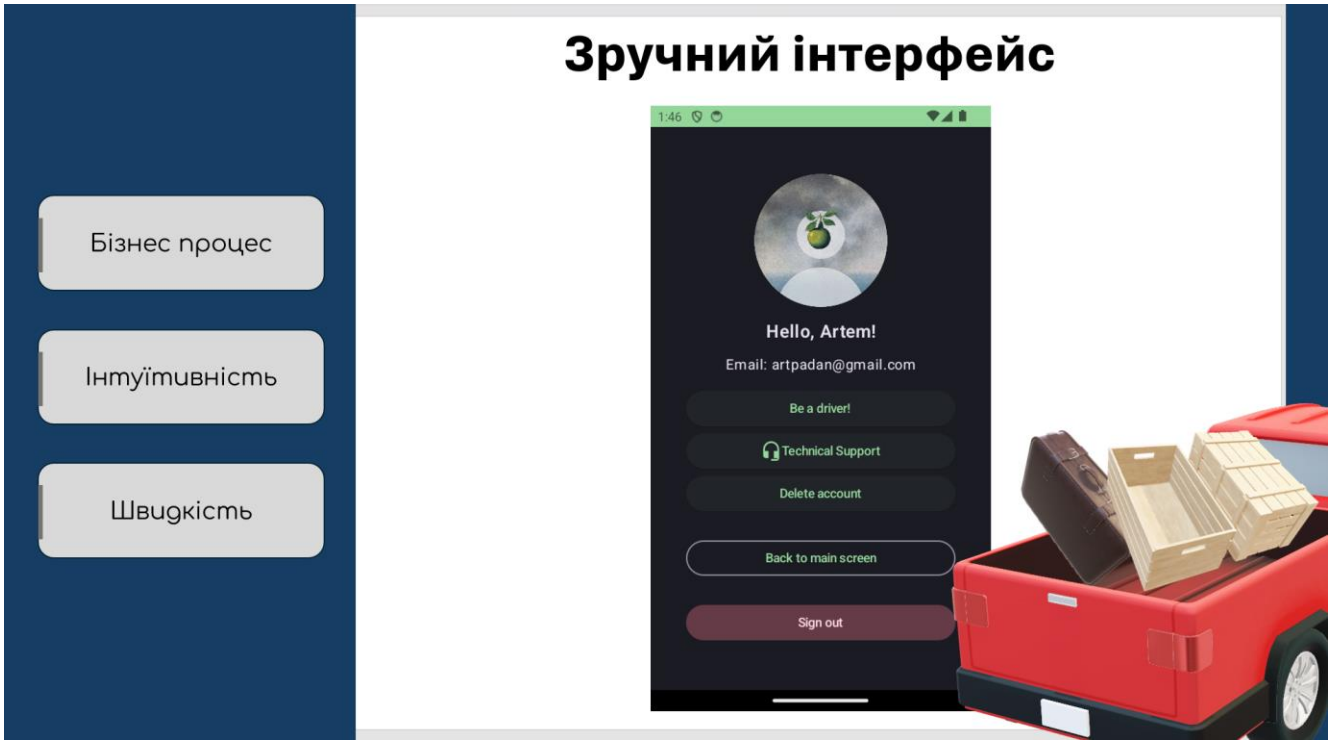


Рисунок Б.7 – Слайд 7

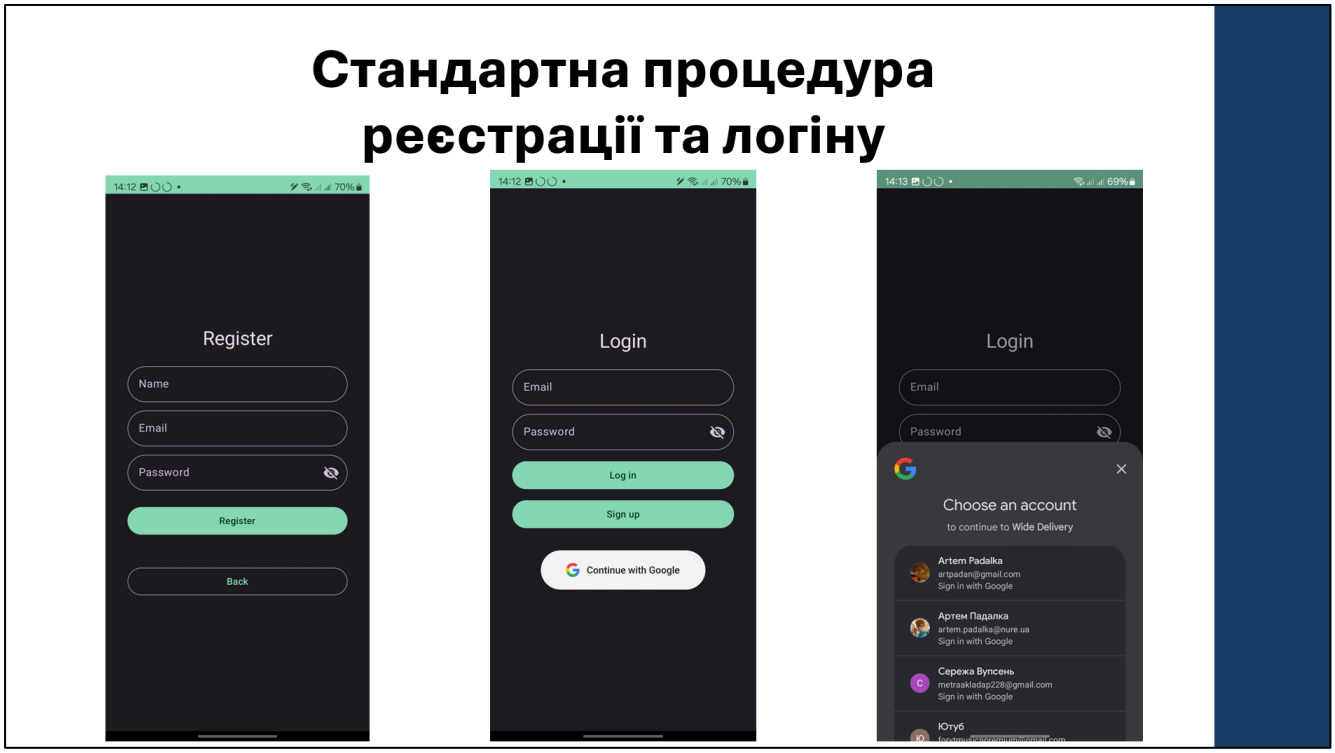


Рисунок Б.8 – Слайд 8

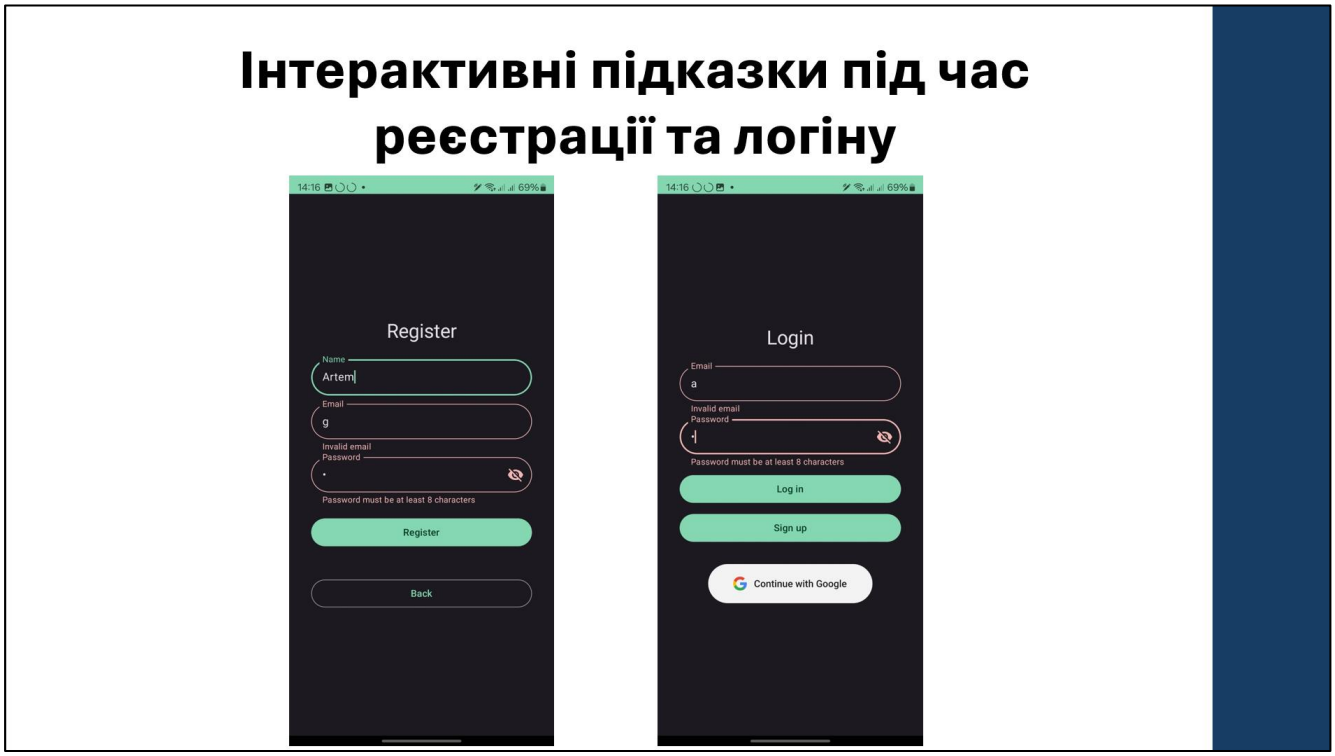


Рисунок Б.9 – Слайд 9

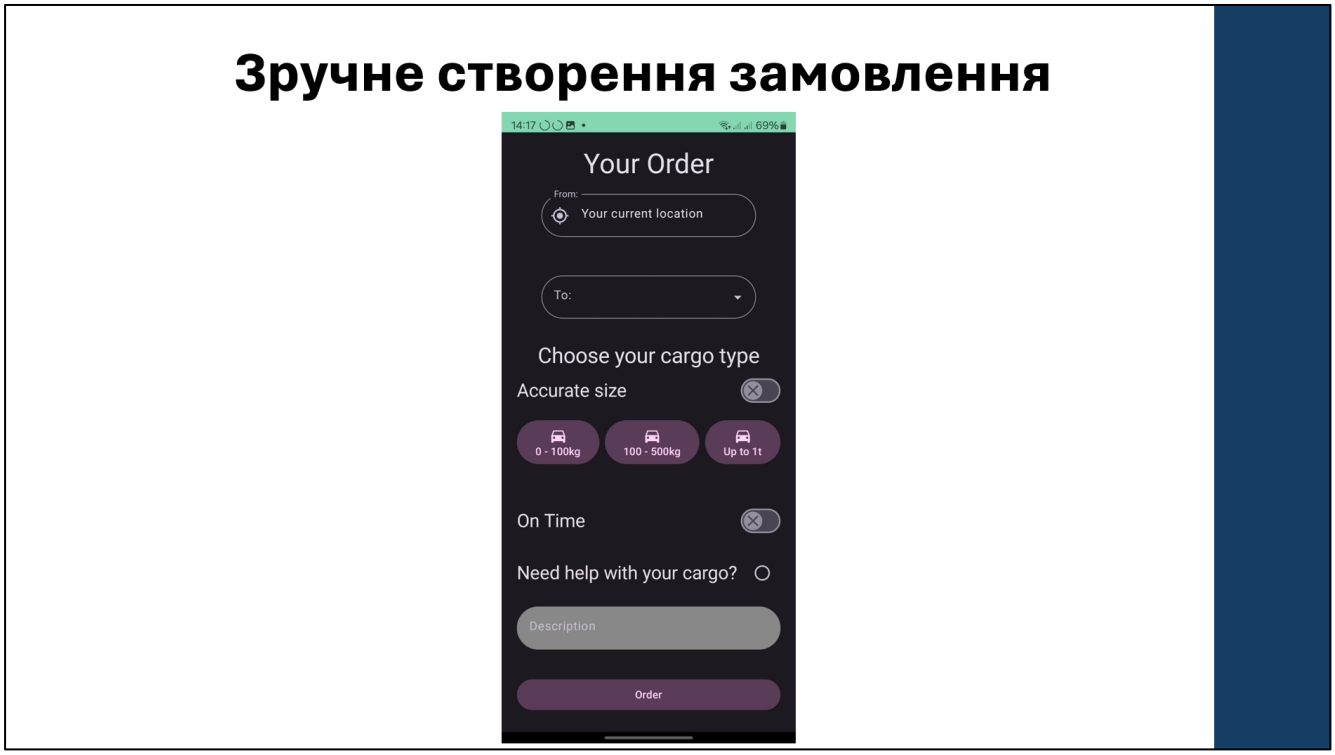


Рисунок Б.10 – Слайд 10

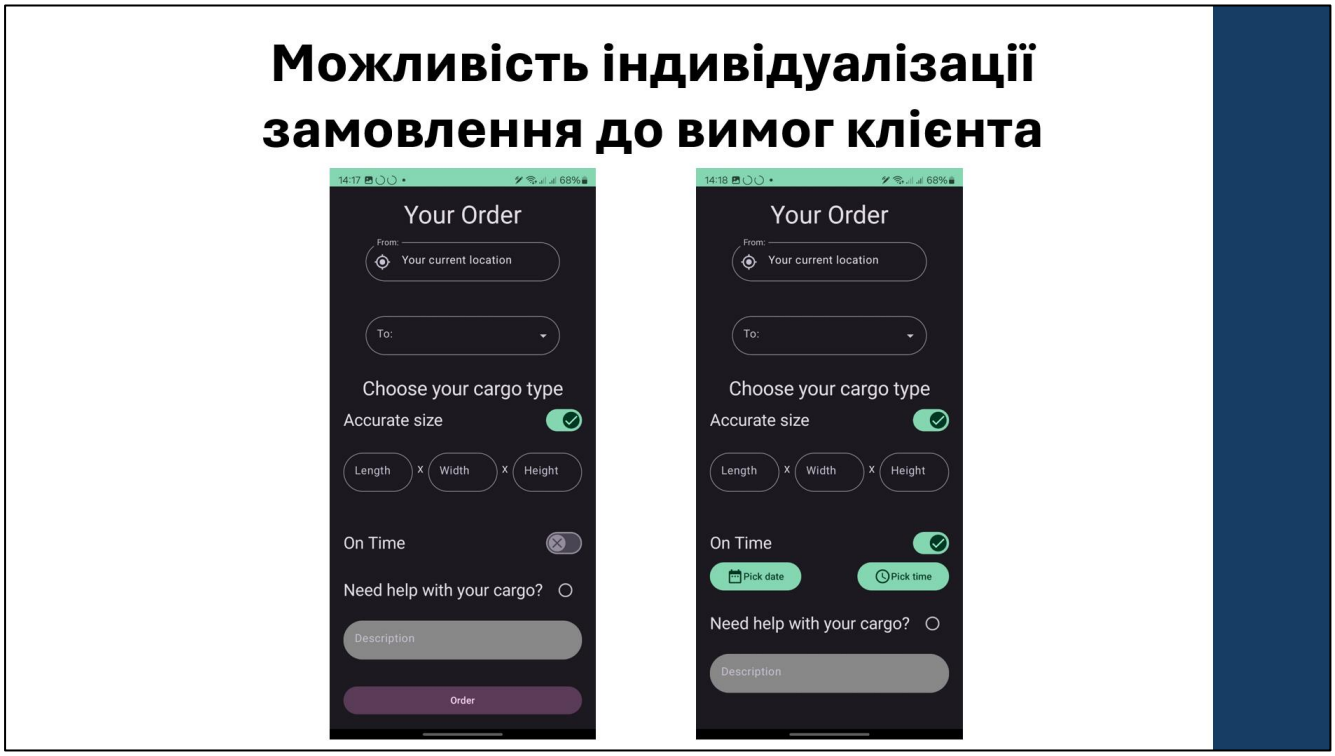


Рисунок Б.11 – Слайд 11

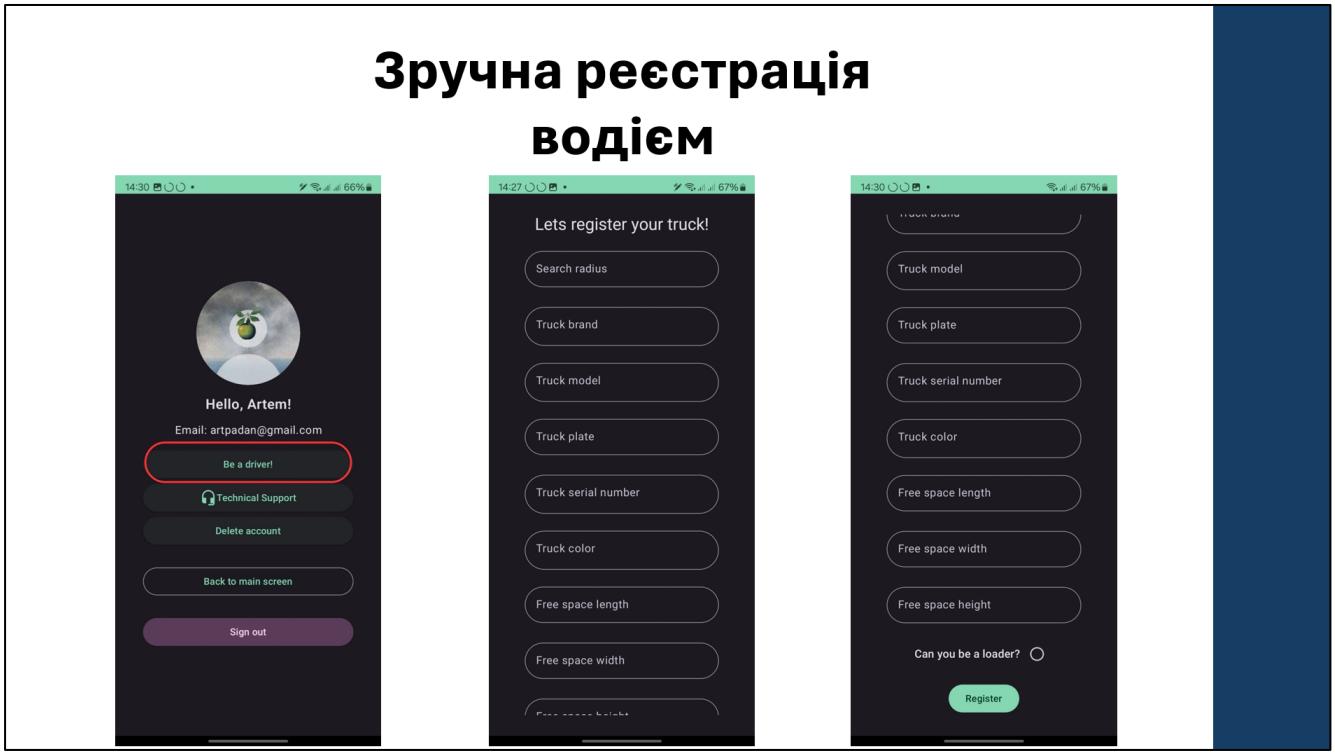


Рисунок Б.12 – Слайд 12

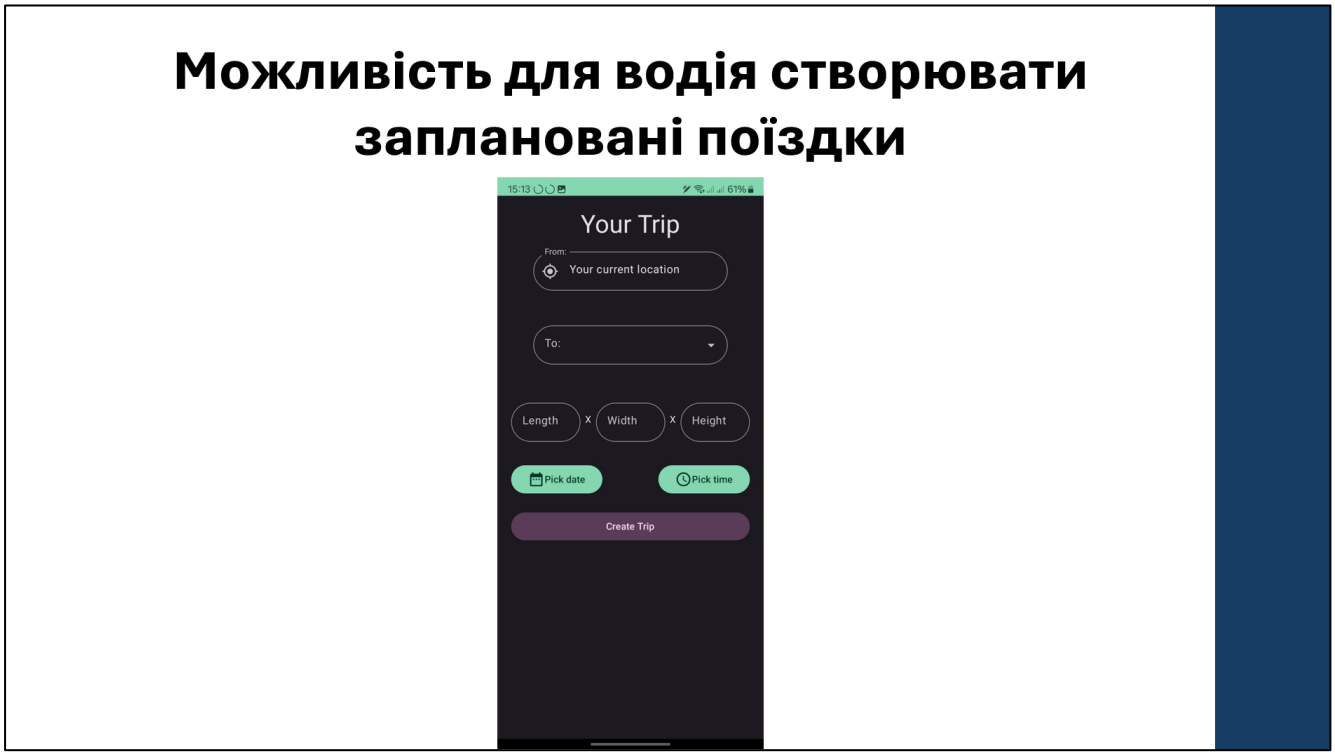


Рисунок Б.13 – Слайд 13

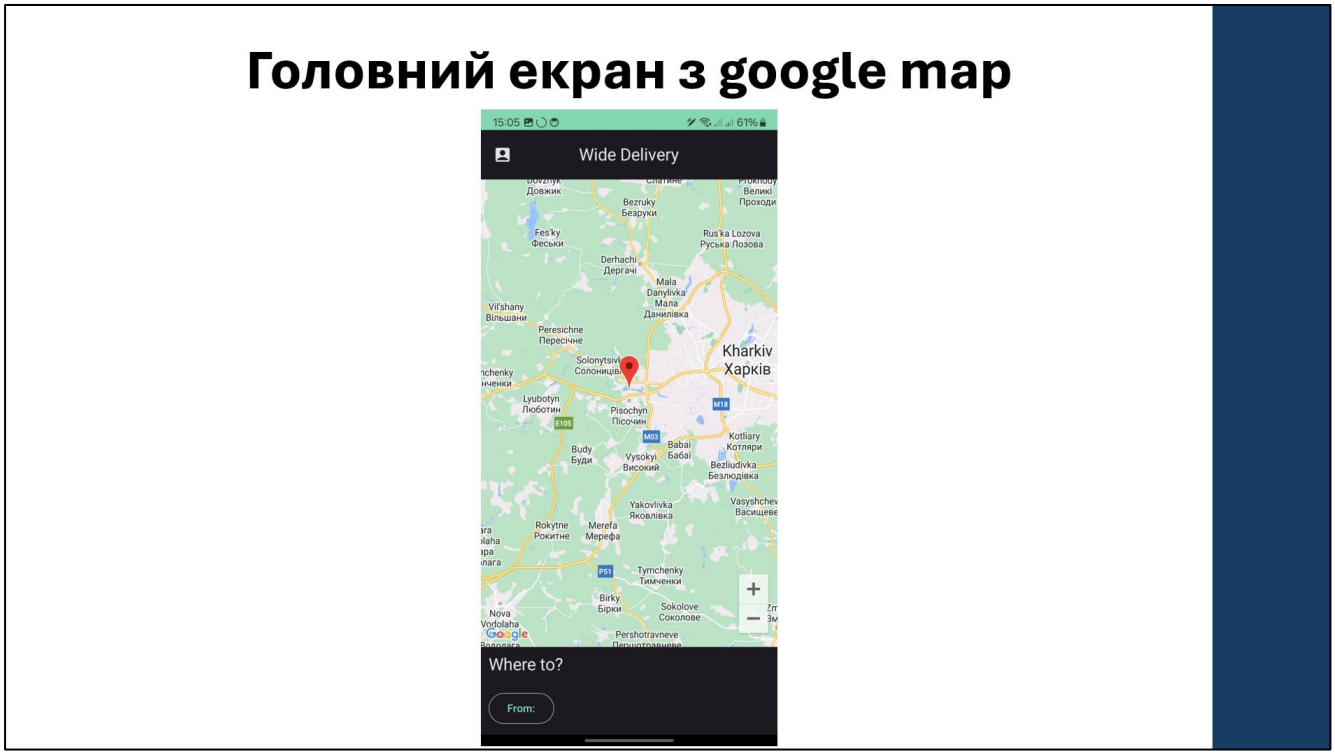


Рисунок Б.14 – Слайд 14

Рисунок Б.15 – Слайд 15

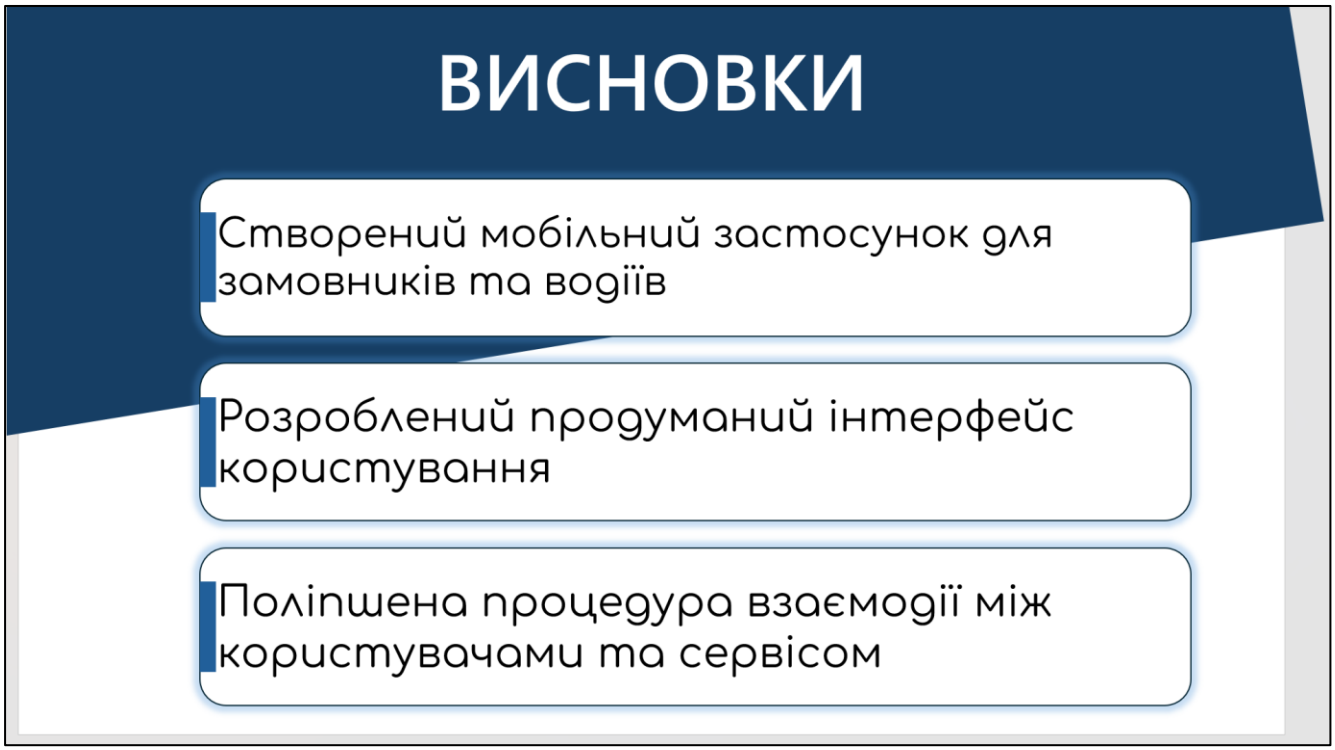
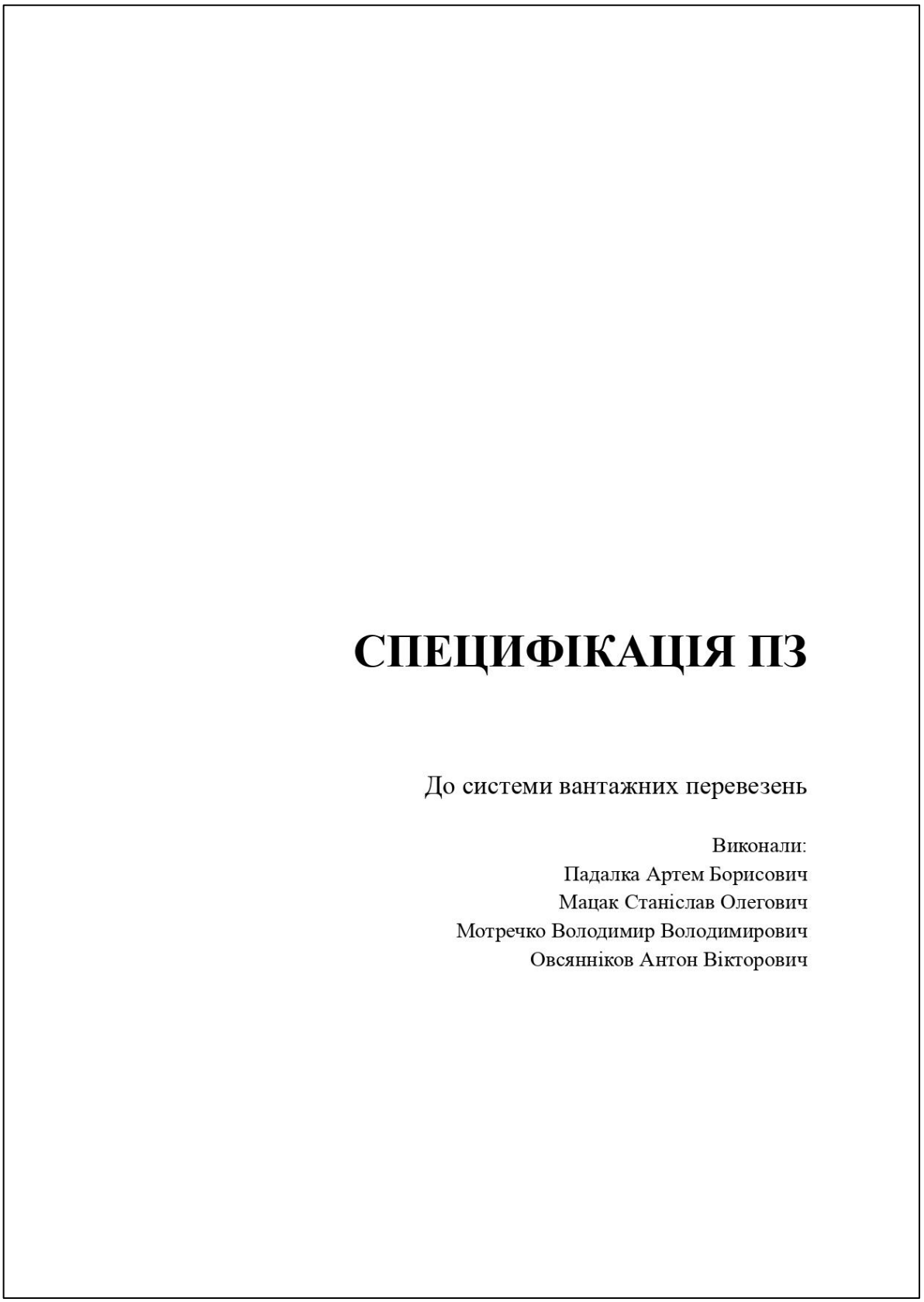


Рисунок Б.16 – Слайд 16

Рисунок Б.17 – Слайд 17

ДОДАТОК В

Специфікація програмного забезпечення



СПЕЦИФІКАЦІЯ ПЗ

До системи вантажних перевезень

Виконали:
Падалка Артем Борисович
Мацак Станіслав Олегович
Мотречко Володимир Володимирович
Овсянніков Антон Вікторович

Рисунок В.1 – Сторінка 1 специфікації пз

Специфікація ПЗ

1. Вступ

1.1 Огляд продукту

Система вантажних перевезень, що ми розробляємо, є інноваційним рішенням для організації доставки вантажів в межах міста або між містами. Концепція системи базується на моделі Uber (офлайн-послуги, доступні відразу після оплати з мобільного додатку), але з деякими відмінностями, пов'язаними з особливостями вантажних перевезень.

Система повинна надати клієнтам зручний та ефективний спосіб замовлення вантажного перевезення “не відриваючись від екрану мобільного пристрою”, де клієнт мусить вказати габарити, тип вантажу та час і місце доставки. Водії, які можуть бути як представниками компаній, що займаються вантажними перевезеннями, так і звичайними користувачами, мають можливість приймати замовлення та виконувати доставку.

Продукт спрямований на модернізацію традиційних підходів до транспортної логістики, забезпечуючи зручні, швидкі та вартісно ефективні рішення для мувінгових компаній та кінцевих користувачів.

Система дозволяє користувачам замовляти перевезення вантажів через мобільний застосунок, пропонуючи опції для негайних та запланованих доставок. Водії можуть реєструватися у системі, вказуючи свої маршрути та доступність, що дозволяє системі автоматично спарювати замовлення з відповідними водіями, оптимізуючи навантаження та мінімізуючи порожні поїздки.

Ключовим нововведенням є впровадження моделі спільної участі, де користувачі можуть ділитися ресурсом своїх транспортних засобів (зокрема, вантажним простором автомобілів), для збільшення загальної ефективності і зниження витрат. Система також надає докладну інформацію про статус доставки, включаючи трекінг у реальному часі,

Рисунок В.2 – Сторінка 2 специфікації пз

оцінки та відгуки, що сприяє підвищенню довіри та задоволеності клієнтів.

Це програмне забезпечення має на меті не тільки підвищити ефективність логістичних операцій, але й зробити процес перевезення більш прозорим та доступним для всіх учасників ринку.

Система передбачає два варіанти доставки: “онлайн” (тобто, максимально швидко) або з запланованим часом.

Для водіїв передбачено можливість вказувати радіус пошуку, якщо вони готові приймати замовлення прямо зараз, або вказувати маршрут, за яким вони будуть їхати в майбутньому. Система автоматично підбирає замовлення, що проходять в межах радіусу або вздовж маршруту, та повідомляє клієнта про можливі варіанти доставки. Після прийняття замовлення водієм, клієнт оплачує вартість доставки та очікує, що в зазначений час водій прибуде та забере (чи допоможе завантажити) необхідний товар.

1.2 Мета

Основна мета створення програмного забезпечення - забезпечити зручну та ефективну організацію доставки вантажів для клієнтів у будь-якій точці країни; пов’язати логістично складні регіони та надати клієнтам широкий вибір способів задоволення їх потреб у перевезенні - з варіацією цін, термінів, допомоги в завантаженні/розвантаженні тощо.

Окрім того, ми хочемо максимально зменшити кількість “пустих” поїздок для вантажних автомобілів малого та середнього бізнесу - тобто максимізувати їх корисний ресурс. Для цього система буде забезпечувати можливість підбору вантажів, що мають схожий маршрут доставки, та можливість об'єднання їх в одне замовлення, а також надавати водіям можливість планувати свої маршрути з урахуванням можливих замовлень на перевезення вантажів.

Ми вбачаємо наше призначення у створенні зручного, надійного та ефективного ринку вантажних перевезень, що задовольняє потреби всіх

Рисунок В.3 – Сторінка 3 специфікації пз

учасників процесу, від користувачів до водіїв та компаній, що займаються перевезеннями.

1.3 Межі

Межі системи:
- мобільний додаток для клієнтів;
- мобільний додаток для водіїв;
- веб-інтерфейс для адміністрування системи;
- серверна частина для мобільних додатків (мікросервіси);
- серверна частина для веб-інтерфейсу для адміністрування системи (мікросервіси);
- база даних, яка зберігає інформацію про клієнтів, водіїв, вантажі, замовлення та доставки.

Межі функціоналу:
- можливість замовлення вантажних перевезень клієнтами через мобільний додаток;
- можливість для водіїв приймати замовлення та виконувати доставку;
- можливість для адміністратора системи керувати роботою системи, відстеження замовлень та доставки, аналіз статистики та звітності;
- можливість для адміністратора системи коригувати замовлення та вести діалоги з клієнтами та водіями, для вирішення проблем/питань.

Межі взаємодії з іншими системами:
- інтеграція з картографічними сервісами для відображення місцезнаходження водіїв та маршрутів доставки (Google Maps);
- інтеграція з платіжними системами для оплати вартості доставки через мобільний додаток (Google Pay);
- інтеграція з системами збору та аналізу метрик та іншої статистики щодо роботи системи (Elasticsearch, Kibana, Logstash).

Рисунок В.4 – Сторінка 4 специфікації пз

1.4 Посилання

Посилання на документи, що стосуються вимог до системи вантажних перевезень:

1. ДСТУ 2609-94 Вантажні автомобільні перевезення. Терміни та визначення
2. ДСТУ ISO 9001:2015 Системи управління якістю. Вимоги (ISO 9001:2015, IDT)
3. ДСТУ 3649:2010 КОЛІСНІ ТРАНСПОРТНІ ЗАСОБИ. Вимоги щодо безпечності технічного стану та методи контролювання

Посилання на документи, що стосуються вимог до програмного забезпечення:

1. ДСТУ ISO/IEC/IEEE 12207:2018 Інженерія систем і програмних засобів. Процеси життєвого циклу програмних засобів (ISO/IEC/IEEE 12207:2017, IDT)
2. ДСТУ ISO/IEC 25010:2016 Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Моделі якості системи та програмних засобів (ISO/IEC 25010:2011, IDT)

1.5 Означення та абревіатури

АПІ (API) - Application Programming Interface, програмний інтерфейс застосунку.

GPS (GPS) - Global Positioning System, глобальна система позиціонування.

Рисунок В.5 – Сторінка 5 специфікації пз

REST (Representational State Transfer) - архітектурний стиль веб-сервісів, що використовується для взаємодії між клієнтськими та серверними додатками.

СКБД - Система керування базами даних.

WebSocket, WS - протокол, що призначений для обміну інформацією між браузером та вебсервером в режимі реального часу.

UI (User Interface) - графічний інтерфейс користувача, що використовується для взаємодії з програмним забезпеченням.

UX (User Experience) - загальний досвід користувача при взаємодії з програмним забезпеченням.

HTTPS (Hypertext Transfer Protocol Secure) - протокол передачі гіпертексту, що використовується для безпечного з'єднання між клієнтом та сервером (з'єднання мобільного застосунку із сервером встановлюється за допомогою HTTP/S).

OAuth (Open Authorization) - протокол авторизації, що використовується для надання обмеженого доступу до ресурсів користувача (авторизація через Google Sign-In працює з використанням протоколу OAuth 2.0).

Користувач - особа, що використовує мобільний застосунок (клієнт-замовник або водій).

Клієнт - особа, яка замовляє послуги з перевезення вантажів.

Водій - особа, яка виконує перевезення вантажів.

Логбук водія - електронне візуальне представлення останніх подій, що були зроблені водієм.

Замовлення - запит клієнта на перевезення вантажу.

Запланована поїздка - інформація, надана водієм, щодо його майбутньої поїздки між вказаними містами чи місцями, впродовж якої він може виконати замовлення.

Маршрут - шлях, за яким здійснюється перевезення вантажу.

Рисунок В.6 – Сторінка 6 специфікації пз

Радіус пошуку - відстань, в межах якої водій готовий прийняти замовлення.

Доставка - процес перевезення вантажу від місця відправлення до місця призначення.

Вантаж - предмети, що перевозяться.

Габарити - розміри вантажу.

Тип вантажу - характеристика вантажу, що визначає особливості його перевезення (харчові продукти, меблі, будівельні матеріали тощо).

Час доставки - час, протягом якого має бути здійснено доставку вантажу.

Оцінка - оцінка клієнтом якості наданих послуг з перевезення вантажу.

Відгук - коментар клієнта про надані послуги з перевезення вантажу.

Адміністратор - особа, що здійснює управління системою.

Модератор - особа, що представляє службу підтримки, комунікує з клієнтами, водіями та компаніями і має можливість редагувати замовлення, скасовувати оплату тощо.

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Ми бачимо Wide Delivery як екосистему, яка об'єднає клієнтів, водіїв та компанії, що займаються перевезеннями, надаючи їм інструменти для швидкої, надійної та ефективної доставки вантажів.

Наші плани розвитку сягають далеко за межі простого створення зручного додатку. Ми прагнемо стати лідером на ринку, задаючи нові стандарти якості, зручності та прозорості. Для досягнення цієї мети ми зосередимося на трьох ключових напрямках.

По-перше, ми будемо поступово розширювати географію обслуговування Wide Delivery. Наша мета - охопити всю територію України, зв'язавши навіть найвіддаленіші населені пункти. Ми прагнемо

Рисунок В.7 – Сторінка 7 специфікації пз

забезпечити доступність наших послуг для всіх, хто потребує швидкої та надійної доставки вантажів, незалежно від їх місця розташування.

По-друге, ми постійно працюємо над удосконаленням функціоналу мобільного додатку. Ми плануємо впровадити нові зручні функції, які зроблять процес замовлення та відстеження доставки ще більш інтуїтивним та ефективним. Наша команда аналізує відгуки користувачів, вивчає потреби ринку та слідкує за останніми технологічними трендами, щоб запропонувати нашим клієнтам найкращі рішення.

По-третє, ми активно працюємо над інтеграцією Wide Delivery з іншими системами. Ми розуміємо, що багато компаній вже використовують власне програмне забезпечення для управління логістикою, тому ми надамо їм можливість інтегрувати свої системи з Wide Delivery через API. Це дозволить автоматизувати процес отримання замовлень, управління доставкою та обміну даними, спрощуючи роботу та підвищуючи ефективність бізнес-процесів.

2.2 Функції програмного забезпечення

Програмне забезпечення для управління вантажними перевезеннями включає в себе широкий спектр функцій, які забезпечують ефективність та зручність використання для клієнтів, водіїв та адміністраторів системи. Нижче наведено список кожної з основних функцій:

- реєстрація та авторизація користувачів (клієнтів та водіїв);
- авторизація користувачів (адміністратор, головний помічник, помічник);
- введення інформації про вантаж та його характеристики;
- введення інформації про місце розташування вантажу та місце призначення;
- введення інформації про час доставки;
- обчислення вартості доставки;
- формування замовлення на доставку;
- пошук водіїв, які підходять під замовлення;
- приймання/відхилення замовлень водіями;

Рисунок В.8 – Сторінка 8 специфікації пз

- надсилання електронних листів з інформуванням про зміну статусу замовлення;
- відстеження статусу замовлення;
- спільна робота водіїв та клієнтів з доставкою;
- ведення статистики та звітності;
- адміністрування системи з веб-інтерфейсу (підтримка онлайн-чатів, редагування інформації про замовлення та скарги, тощо);
- інтеграція з картографічними сервісами (Google Maps/Google Streets);
- інтеграція з платіжними системами (Google Pay);
- забезпечення безпеки та конфіденційності даних користувачів;
- надання технічної підтримки користувачам.

2.3 Характеристики користувачі

Система передбачає наявність таких основних категорій користувачів: клієнт, водій, адміністратор, модератор.

Клієнти - фізичні та юридичні особи, які потребують послуг з доставки вантажів.

Водії - фізичні особи або транспортні компанії, що надають послуги з доставки вантажів.

Адміністратори - особи, що відповідають за управління системою, переглядом звітності, менеджмент персоналу

Модератори - особи, що мають доступ до інформації щодо клієнтів, водіїв, замовлення та можуть підтримувати комунікацію і вирішувати проблеми із взаємодією інших користувачів із системою.

Вимоги клієнтів:

Рисунок В.9 – Сторінка 9 специфікації пз

- просте та інтуїтивно зрозуміле користування додатком;
- можливість швидкого та зручного замовлення вантажоперевезень;
- можливість замовити допомогу в завантаженні вантажу чи його розвантаженні;
- можливість відстежувати статус замовлення в реальному часі;
- надійність та пунктуальність виконання замовлення;
- гарантії безпеки та повернення коштів у разі проблем із доставкою замовлень;
- зручні опції оплати;
- зручні варіанти авторизації (включно з Google Sign-In);
- можливість залишити відгук про виконання замовлення.
- можливість звернутись до технічної підтримки, якщо є така потреба

Вимоги водіїв:

- просте та інтуїтивно зрозуміле користування додатком;
- можливість швидкого та зручного прийняття замовлень;
- можливість планування маршрутів та вантажоперевезень, в тому числі запланованих на значний час наперед;
- можливість відстежувати статус виконання замовлення;
- надійну та своєчасну оплату за виконані перевезення;
- можливість залишити відгук про виконання замовлення.
- можливість звернутися до служби підтримки в текстовому форматі.

Вимоги адміністраторів:

- доступ до інформації про клієнтів та водіїв та можливість її редагування;
- можливість створювати, редагувати та видаляти модераторів та інших адміністраторів системи;
- доступ до інформації про замовлення та можливість їх редагування, скасування та повернення коштів;

Рисунок В.10 – Сторінка 10 специфікації пз

- можливість вести чат з клієнтами та водіями для надання допомоги та розв'язання проблем;
- можливість керувати скаргами та поверненнями коштів;
- доступ до статистики та звітів про роботу системи;
- можливість керувати доступом користувачів до системи та налаштуваннями безпеки.

Модератор має аналогічні до адміністратора вимоги щодо системи, однак він не має доступу до редагування інформації щодо персоналу, інших менеджерів та адміністраторів.

2.4 Загальні обмеження

Обмеження функціоналу

- система не передбачає перевезення небезпечних вантажів, таких як вибухонебезпечні, отруйні, радіоактивні та інші речовини, що підпадають під дію чинного законодавства щодо перевезення небезпечних вантажів;
- система не передбачає перевезення вантажів, що порушують чинне законодавство або права інтелектуальної власності;
- система не передбачає перевезення живих істот, за винятком тварин, які перевозяться разом з власниками в спеціальних контейнерах для перевезення тварин;
- система не передбачає перевезення вантажів, які перевищують максимально допустимі габарити та вагу, встановлені для перевезення на конкретному типі транспортного засобу;
- максимальні допустимі габарити вантажу визначаються параметрами транспортних засобів, що використовуються водіями-перевізниками. Користувач повинен вказати точні габарити свого вантажу під час створення замовлення, щоб система могла коректно підібрати відповідний транспортний засіб. Водій повинен

Рисунок В.11 – Сторінка 11 специфікації пз

вказати точні габарити свого транспортного засобу, для коректного пошуку замовлення під його автомобіль;

- система надає можливість користувачеві вибрати один з двох варіантів доставки: якомога швидша або запланована на певну дату в майбутньому. При виборі швидкої доставки, користувач розуміє, що вартість може бути вищою, ніж при замовленні на майбутнє. У разі вибору запланованої доставки, користувач зобов'язаний вказати точну дату і час, коли вантаж буде готовий до відправлення.

Обмеження географії:

- система не розрахована на міжнародні перевезення, географія місця призначення замовлення обмежується лише територією держави, звідки замовлення створено (початково - в межах України);
- система може не працювати в деяких районах, які є небезпечними для перевезення вантажів (регіон ведення бойових дій, тимчасово окуповані території тощо). Система повинна сповіщувати про неможливість перевезення вантажу ще на етапі створення замовлення.

Обмеження, що стосуються водіїв-перевізників:

- водії-перевізники повинні використовувати транспортні засоби, що відповідають вимогам системи щодо габаритів і типу вантажу. Система не допускає використання транспортних засобів, що не відповідають цим вимогам, для виконання замовлень на перевезення вантажів;
- водії-перевізники можуть вказувати радіус пошуку, якщо вони готові прийняти замовлення прямо зараз, або вказувати маршрут, за яким вони будуть їхати в майбутньому. Система підбирає замовлення, що проходять в межах радіусу чи вздовж маршруту (з можливими відхиленнями) і сповіщає водія-перевізника про можливі варіанти.

Рисунок В.12 – Сторінка 12 специфікації пз

Водій зобов'язаний вчасно повідомляти систему про зміни в своєму розкладі або маршруті;

- водії-перевізники можуть надавати послуги з допомоги в завантаженні та розвантаженні вантажу, але це не є обов'язковою умовою. Якщо водій-перевізник не може надати таку послугу, він повинен чітко вказати це в своєму профілі. Ця послуга обов’язково сплачується додатково до тарифу перевезення, про що клієнта буде повідомлено відразу при створенні замовлення.

Обмеження відповідальності:

- Система не несе відповідальності за вантаж під час перевезення. Відповідальність за вантаж покладається на відправника та водія-перевізника. У разі пошкодження або втрати вантажу, сторони зобов'язані самостійно вирішувати питання щодо компенсації збитків. Команда модераторів системи має обов’язок допомагати сторонам під час вирішення таких питань, однак не несе відповідальності за будь-які пошкодження;
- сторони зобов'язані дотримуватися вимог щодо безпеки перевезень, встановлених чинним законодавством. У разі порушення цих вимог, сторони несуть відповідальність відповідно до закону;
- система не гарантує надання послуг з перевезення вантажів у всіх випадках. У разі відсутності водіїв-перевізників, що готові виконати замовлення або за наявності інших обставин, що перешкоджають наданню послуг, система повідомляє користувача про неможливість виконання замовлення.

2.5 Припущення і залежності

Для того, щоб Wide Delivery успішно функціонував та розвивався, ми враховуємо низку важливих факторів. По-перше, ми виходимо з припущення, що наші користувачі мають доступ до сучасних мобільних

Рисунок В.13 – Сторінка 13 специфікації пз

пристроїв з операційною системою Android та стабільним інтернет-з'єднанням, що дозволить їм повноцінно використовувати всі можливості мобільного додатку. Ми також очікуємо, що водії, які приєднаються до платформи, мають необхідний досвід та кваліфікацію для керування вантажними автомобілями, а також відповідально ставляться до правил дорожнього руху. Крім того, ми розраховуємо на те, що всі учасники платформи - клієнти, водії та транспортні компанії - будуть діяти відповідно до чинного законодавства України, яке регулює сферу вантажних перевезень. Ми також спираємося на стабільну та безперебійну роботу сторонніх сервісів, таких як Google Maps, Google Pay та інші платформи, інтегровані в Wide Delivery.

Не менш важливим є розуміння залежностей, які можуть впливати на роботу платформи. Очевидно, що стабільне інтернет-з'єднання є критично важливим як для клієнтів, так і для водіїв. Будь-які проблеми з інтернетом можуть призвести до збоїв у роботі Wide Delivery та ускладнити процес замовлення та відстеження доставки. Точність та доступність геолокаційних даних, що надходять з мобільних пристроїв, також є ключовим фактором для коректної роботи платформи. Неточності або відсутність геолокації можуть спричинити помилки у визначенні місця розташування клієнтів та водіїв, а також ускладнити побудову оптимальних маршрутів доставки.

Безпека та надійність інтегрованих платіжних систем також є важливою складовою успіху Wide Delivery. Від них залежить можливість клієнтів зручно та безпечно оплачувати замовлення.

Не менш важливою є довіра та репутація платформи. Ми докладаємо максимум зусиль, щоб створити Wide Delivery як надійний та безпечний сервіс, який гарантує якісне виконання замовлень, захищає інтереси всіх учасників та формує позитивний досвід взаємодії.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Основна взаємодія користувача з системою Wide Delivery відбувається через мобільний додаток, який розроблено з метою забезпечення максимально зручного та інтуїтивно зрозумілого доступу до

Рисунок В.14 – Сторінка 14 специфікації пз

всіх функцій сервісу. Додаток, призначений як для клієнтів, так і для водіїв, має дозволяти виконувати всі необхідні дії безпосередньо в ньому, без необхідності звертатися до сторонніх сервісів.

Користувачі можуть авторизуватися в системі за допомогою облікового запису Google або ж створити обліковий запис, використовуючи свою електронну пошту. Клієнти легко створюють замовлення на перевезення, вказуючи всі необхідні параметри вантажу: його габарити, тип, адресу завантаження та доставки, бажаний час прибуття, а також мають можливість зазначити необхідність допомоги з завантаженням та розвантаженням. Водії, в свою чергу, мають доступ до детальної інформації про доступні замовлення, включаючи маршрут, тип вантажу, вартість доставки та інші важливі деталі.

Навігація в додатку має бути простою та інтуїтивно зрозумілою, дозволяючи швидко переміщуватися між різними екранами та розділами. Система забезпечує можливість відстежувати поточний статус замовлення. Користувачі також мають можливість редагувати свої профілі, змінюючи особисту інформацію та налаштування. У системі є можливість звертатися до служби підтримки у зручному месенджері Telegram, де їм зможуть надати допомогу Support'и.

Wide Delivery має включати систему оплати з чітко визначеними тарифами, що гарантує прозорість та передбачуваність вартості послуг.

Адміністратори системи Wide Delivery взаємодіють з нею через спеціалізований веб-інтерфейс, який забезпечує зручний та інтуїтивно зрозумілий доступ до всіх функцій управління сервісом. Авторизація здійснюється через ввід пошти та паролю, після чого адміністратори можуть керувати базою користувачів, включаючи створення, редагування та видалення облікових записів. Вони мають доступ до детальної інформації про всі замовлення, можуть переглядати та редагувати деталі замовлень, відстежувати статус доставки в реальному часі, а також аналізувати різні аспекти замовлень для підвищення ефективності операцій.

Крім того, веб-інтерфейс адміністратора надає можливості для аналізу статистики, що допомагає в ухваленні обґрунтованих рішень. Вони також відповідають за підтримку безпеки та конфіденційності даних користувачів. Для надання підтримки користувачам, адміністратори можуть використовувати вбудований чат або електронну пошту, що

Рисунок В.16 – Сторінка 16 специфікації пз

дозволяє швидко реагувати на запити та вирішувати проблеми, забезпечуючи високий рівень обслуговування.

3.1.2 Апаратні інтерфейси

Пристрої, що використовують мобільний додаток, повинні мати не менш ніж 2ГБ ОЗУ, і мати частоту процесора не нижче 1 ГГц. Мобільні пристрої повинні мати 350 Мб вільної пам'яті у постійному сховищі для встановлення додатку та його нормального функціонування.

3.1.3 Програмний інтерфейс

Мобільний додаток створюється першочергово для Android-пристроїв, отже Android версії 7.0+ є вимогою щодо ОС мобільного пристрою.

Будуть використовуватися наступні API:

- Google OAuth 2.0 API;
- Google Maps API;
- Google Pay API.

Браузери, що запускатимуть веб-застосунок адміністраторської частини мають підтримувати стандарт ECMAScript 2015 (ES6). Отже, мінімальними необхідними версіями найпопулярніших у світі браузерів є:

- Chrome 51 + або вище;
- Edge 15 або вище;
- Safari 10 або вище;
- Mozilla Firefox 54 або вище;
- Opera 38 або вище;
- Internet Explorer 11 (частково).

Веб-застосунок, призначений для адміністраторів та модераторів, може бути запущений з Windows 7 або вище, Linux, MacOS 10.5 або вище.

Рисунок В.17 – Сторінка 17 специфікації пз

3.1.4 Комунікаційний протокол

Wide Delivery використовує різноманітні комунікаційні протоколи для забезпечення ефективної та надійної взаємодії між різними компонентами системи.

Для спілкування мобільного додатку з сервером буде застосовано HTTP/HTTPS протокол. Це забезпечить безпечну передачу даних та захистить конфіденційну інформацію користувачів. Окрім того, буде впроваджено технологію WebSocket для підтримки швидкої передачі постійних пакетів даних із сервера на клієнт. WebSocket дозволить реалізувати функції реального часу, такі як відображення поточного місцезнаходження вантажу на карті та оновлення статусу замовлення.
Внутрішня комунікація між мікросервісами та основним API на стороні сервера буде здійснюватися за допомогою системи gRPC. gRPC - це сучасний високопродуктивний фреймворк для віддаленого виклику процедур (RPC), який забезпечує швидку та ефективну взаємодію між сервісами. Використання gRPC дозволить оптимізувати роботу серверної частини Wide Delivery та забезпечити її масштабованість.

REST буде використовуватися для надання зовнішнього API для інтеграції з іншими системами, такими як платіжні шлюзи та картографічні сервіси. REST API забезпечить стандартизований інтерфейс для взаємодії з Wide Delivery та дозволить стороннім розробникам легко інтегрувати свої сервіси з нашою платформою.

3.1.5 Обмеження пам'яті

Система Wide Delivery повинно бути оптимізованою для роботи з великими обсягами даних, зберігаючи при цьому швидку та стабільну роботу системи. Для цього необхідно застосовувати алгоритми стиснення даних, кешування, а також вміло використовувати пам'ять пристрою.

Рисунок В.18 – Сторінка 18 специфікації пз

Основна інформація зберігатиметься в реляційних та NoSQL базах даних, тож додаток повинен мати мінімальну кількість інформації, що зберігається безпосередньо на мобільному девайсі.

3.1.6 Операції

Основна мета: Опис операцій, які здійснюються в рамках системи, дозволяє забезпечити зручність і ефективність процесів взаємодії користувачів з додатком.

Операції системи включають:

1. **Створення замовлення:** Користувачі можуть створювати замовлення через мобільний додаток, вказуючи деталі вантажу (габарити, тип, час доставки, місце відправлення та прибуття). Система автоматично підбирає водія на основі заданих критеріїв.
2. **Оновлення статусу замовлення:** Водії можуть оновлювати статус замовлення у режимі реального часу, що дає можливість користувачам відстежувати процес доставки.
3. **Підтвердження доставки:** Після завершення доставки водій вводить статус як завершений, а клієнт отримує повідомлення про успішну доставку з можливістю залишити відгук.
4. **Скасування замовлення:** Користувач має можливість скасувати замовлення до моменту його прийняття водієм, з інформуванням водія та поверненням коштів.
5. **Оплата за допомогою інтегрованих платіжних систем:** Користувачі можуть оплачувати послуги через зручні та безпечні платіжні системи, інтегровані в мобільний додаток.

Рисунок В.19 – Сторінка 19 специфікації пз

Зауваження: Всі операції повинні супроводжуватися детальними повідомленнями про статус операції для забезпечення прозорості та контролю за процесом доставки вантажів.

3.1.7 Функції продукту

3.1.7.1 Реєстрація, авторизація (мобільний застосунок)

Вступ

Користувач повинен мати можливість авторизуватися через Google або зареєструвати обліковий запис через пошту для подальшої авторизації в додатку під своїм обліковим записом.

Вхідні дані

Користувач завантажив застосунок, вводить електронну пошту та пароль або авторизується через обліковий запис Google та натискає кнопку підтвердження.

Обробка

За наявності акаунту, перевіряється правильність введеного паролю, інакше створюється новий акаунт. Для Google Sign In зберігати пароль в БД не потрібно.

Результати

У разі правильних надісланих даних користувач отримує токен доступу до системи та токен відновлення токену доступу. Застосунок перенаправляє його на головну сторінку мобільного додатку.

Обробка помилок

Користувач не зможе зареєструватися/увійти у разі будь-якої помилки, неправильного паролю, неіснуючого акаунту, або якщо намагається пройти аутентифікацію з Google, при створеному за допомогою пошти та паролю акаунті чи навпаки. Back-end повинен повертати HTTP-статус 500 для внутрішніх помилок сервера, 403 для неправильно введених пошти чи пароля та некоректного OAuth-токену.

3.1.7.2 Створення замовлення (мобільний застосунок)

Вступ

Користувач повинен мати можливість створити замовлення після успішної аутентифікації.

Вхідні дані

Рисунок В.20 – Сторінка 20 специфікації пз

Користувач вводить інформацію про вантаж, його тип, габарити, місце призначення та час відправлення, зазначає необхідність у допомозі з завантаження чи розвантаження.

Обробка

Замовлення перевіряється на валідність, оцінюється можливість його виконання, визначається його ціна та розпочинається підбір водіїв, що можуть виконати це замовлення.

Результати

У разі коректного створення замовлення воно валідується та переходить на етап пошуку водія, що зможе виконати замовлення. Також користувач отримує інформацію про орієнтовну вартість замовлення із пропозицією щодо сплати за допомогою Google Pay.

Обробка помилок

У разі помилки створення замовлення користувач отримує інформацію про причину помилки та контактами служби підтримки, для уточнення деталей. Back-end клієнтської частини повинен повертати повідомлення, чому саме замовлення не може бути прийнято до обробки.

3.1.7.3 Оплата замовлення

Вступ

Клієнт повинен оплатити замовлення для того, щоб водій розпочав його виконання.

Вхідні дані

Замовлення успішно прийнято до обробки та клієнтський back-end визначнив вартість замовлення.

Обробка

Сторонній сервіс проводить оплату та передає результат оплати серверу. Сервер перевіряє, чи справді сторонній сервіс провів оплату, і чи запит не був виконаний зловмисником.

Результати

У разі успішної оплати розпочинається процес підбору водія.

Обробка помилок

У разі проведення оплати та списання коштів, але відсутності зміни статусу замовлення, користувач має змогу звернутись до служби підтримки. У разі помилкових відповідей від постачальника платіжних послуг, користувачу буде запропоновано ще раз оплатити замовлення.

Рисунок В.21 – Сторінка 21 специфікації пз

3.1.7.4 Перегляд статусу замовлення в режимі реального часу (мобільний застосунок)

Вступ

Клієнт повинен бути проінформованим про зміну статусу замовлення та його поточне місцерозташування.

Вхідні дані

Підтверджене замовлення, що розпочало своє виконання. Водій підтвердив відправку до місця призначення. Водій передає своє місцезнаходження за допомогою увімкненого на мобільному пристрої GPS.

Обробка

Перевірка того, що GPS збігається з визначеним маршрутом перевезення та збереження поточного місцезнаходження, оцінка очікуваного часу прибуття. Надсилання сповіщення клієнтові щодо статусу замовлення за допомогою електронної пошти.

Результати

Клієнт дізнається про статус замовлення та поточне місцезнаходження вантажу.

Обробка помилок

В разі помилки отримання місцезнаходження впродовж виначеного часу модератор служби підтримки зв’язується із водієм. Також надсилається електронний лист про помилку, що сталась під час виконання замовлення.

3.1.7.5 Підбір замовлення для водія (мобільний застосунок)

Вступ

Система підбирає замовлення для водія та пропонує прийняти його.

Вхідні дані

Водій вказує свої налаштування - радіус пошуку, або запланований маршрут, можливість бути вантажником, а також поточний GPS (надає дозвіл додатку на передачу).

Обробка

Система підбирає замовлення, що відповідають вказаним у налаштуваннях критеріях, та пропонує перелік підібраних замовлень.

Результати

Рисунок В.22 – Сторінка 22 специфікації пз

Водій бачить список замовлень та приймає їх, або відхиляє. В разі відхилення, замовлення більше не з’являється у списку та передається наступному водієві.

Обробка помилок

В разі помилок з отриманням замовлень водієві буде запропоновано звернутись до служби підтримки застосунку.

3.1.7.6 Звернення до служби підтримки (мобільний застосунок, система модерації)

Вступ

Користувач має можливість звернутись до служби підтримки, що працює за допомогою чат-боту Telegram.

Вхідні дані

Користувач мобільного застосунку (водій чи клієнт) потребує допомоги служби підтримки та натискає кнопку в мобільному застосунку, що розпочинає діалог в чат-боті Telegram.

Обробка

Користувач інтерфейсу адміністратора відповідає на скаргу в інтерактивному вигляді у виді чату.

Результати

Після завершення спілкування по скарзі, користувач інтерфейсу адміністратора змінює статус скарги до відповідного рішення, прийнятого обома сторонами. Клієнт також має можливість завершити розгляд питання, якщо його було вирішено.

3.1.7.7 Перегляд статистики на головній сторінці (система модерації)

Вступ

Користувач інтерфейсу адміністратора після логіну заходить на головну сторінку.

Вхідні дані

Користувач інтерфейсу адміністратора повинен бути зареєстрованим.

Обробка

Користувач інтерфейсу адміністратора перевіряється на аунтифікацію та після цього йому надається доступ до головної сторінки.

Результати

Після завершення логіну, користувач інтерфейсу адміністтратора може передивитися статистику по кількості скарг за останній місяць за

Рисунок В.23 – Сторінка 23 специфікації пз

кожен день, кількість замовлень за останній місяць за кожен день, та кількість скарг за статусами, кількість замовлень за статусами.

3.1.7.8 Вхід користувача до інтерфейсу адміністратора (система модерації)

Вступ

Користувач інтерфейсу адміністратора повинен увійти у свій обліковий запис.

Вхідні дані

Користувач інтерфейсу адміністратора повинен перейти на сторінку логіну та ввести пошту та пароль.

Обробка

Користувач інтерфейсу адміністратора перевіряється на аунтифікацію та після цього йому надається доступ до головної сторінки.

Результати

Після завершення логіну, користувач інтерфейсу адміністтратор має доступ до сторінок відповідно до своєї ролі у системі.

3.1.7.9 Перегляд статистики на сторінці статистики (система модерації)

Вступ

Користувач інтерфейсу адміністратора переглядає статистику.

Вхідні дані

Користувач інтерфейсу адміністратора повинен бути аутентифікований у системі та перейти на сторінку статистики.

Обробка

Користувач інтерфейсу адміністратора перевіряється на аунтифікацію та після цього йому надається доступ до сторінки статистики

Результати

Після завершення перевірки аутентифікації, користувач інтерфейсу адміністратор має доступ до сторінки статистики де може побачити статистику по скаргам та замовленням.

3.1.7.10 Зміна налаштувань замовлення (система модерації)

Вступ

Користувач інтерфейсу адміністратора за проханням клієнта змінює налаштування замовлення клієнта.

Рисунок В.24 – Сторінка 24 специфікації пз

Вхідні дані

Користувач інтерфейсу адміністратора отримує прохання від клієнта та переходить на сторінку замовлення де натискає на потрібне замовлення.

Обробка

Користувач інтерфейсу адміністратора домовляється з користувачем та змінює відповідно до потреби користувача замовлення.

Результати

Після завершення, користувач інтерфейсу адміністратора бачить успішно змінене замовлення та повідомляє про це користувачу.

3.1.7.11 Видалення замовлення (система модерації)

Вступ

Користувач інтерфейсу адміністратора за певних технічних обставин видаляє замовлення.

Вхідні дані

Користувач інтерфейсу адміністратора отримує прохання від клієнта, або із-за технічних причин повинен перейти на сторінку замовлення де натискає у потрібному замовленні кнопку видалили.

Обробка

Відправляється запит на сервер, після цього потрібне замовлення знаходиться і видаляється з бази даних та повертає результат видалення.

Результат

Після завершення, користувач інтерфейсу адміністратора бачить успішне видалення замовлення та повідомляє про це користувачу.

3.1.7.12 Видалення скарг (система модерації)

Вступ

Користувач інтерфейсу адміністратора за певних технічних обставин видаляє скаргу.

Вхідні дані

Користувач інтерфейсу адміністратора отримує прохання від клієнта, або із-за технічних причин повинен перейти на сторінку скарг де натискає у потрібній скарзі кнопку видалили.

Обробка

Відправляється запит на сервер, після цього потрібна скарга знаходиться і видаляється з бази даних та повертає результат видалення.

Результат

Рисунок В.25 – Сторінка 25 специфікації пз

Після завершення, користувач інтерфейсу адміністратора бачить успішне видалення скарги та повідомляє про це користувачу.

3.1.7.13 Переглядь інформації про користувача інтерфейсу адміністратора (система модерації)

Вступ

Користувач інтерфейсу адміністратора хоче подивитися інформацію про себе.

Вхідні дані

Користувач інтерфейсу адміністратора хоче подивитися інформацію про себе.

Обробка

Відправляється запит на сервер, після цього інформація за своїм профілем надсилається у відповідь.

Результат

Після завершення, користувач інтерфейсу адміністратора бачить інформацію про себе.

3.1.8 Припущення й залежності

При розробці системи вантажних перевезень ми робимо наступні припущення:

- користувачі мають доступ до мобільного інтернету та сумісних пристроїв на Android/IOS для взаємодії з нашим додатком;
- водії мають належні права та документи для здійснення вантажних перевезень, а також транспортні засоби, що відповідають вимогам безпеки та технічного стану;
- компанії, що займаються вантажними перевезеннями, мають необхідні ліцензії та дозволи на надання послуг з перевезення вантажів;
- водії зобов'язані дотримуватися правил дорожнього руху та інших нормативних вимог під час виконання замовлень на перевезення вантажів;

Рисунок В.26 – Сторінка 26 специфікації пз

- геолокаційні дані, що використовуються в додатку, надають точну та актуальну інформацію про місцезнаходження користувачів та водіїв, в тому числі під час виконання замовлення.
- система оплати в додатку функціонує належним чином, забезпечуючи безпечну та зручну оплату послуг з перевезення вантажів;

Залежності системи:

- для належної роботи системи необхідна взаємодія з зовнішніми сервісами та інтерфейсами, зокрема картографічними сервісами (Google Maps API, Google Street API), геолокаційними сервісами, системами оплати (Google Pay) тощо;
- система залежить від стабільної роботи мережі Інтернет, як для користувачів, так і для водіїв. Коректність відображення актуального місцезнаходження вантажу залежить від якості підключення до мобільної мережі;
- для забезпечення безпеки та якості послуг система залежить від дотримання користувачами та водіями правил дорожнього руху та інших нормативних вимог;
- система залежить від рівня довіри між користувачами та водіями, а також від якості наданих послуг з перевезення вантажів;
- система залежить від актуальності та повноти інформації про вантажі, маршрути та інші параметри, що вводяться користувачами та водіями;
- система залежить від ефективної взаємодії між користувачами, водіями, адміністраторами системи та іншими учасниками процесу перевезення вантажів.

Рисунок В.27 – Сторінка 27 специфікації пз

3.2 ФУНКЦІОНАЛЬНІ ВИМОГИ

3.2.1 FR-1 Реєстрація, авторизація

3.2.1.1 Вступ

Користувач повинен мати право авторизуватися через Google або зареєструвати обліковий запис через пошту для подальшої авторизації в додатку під своїм обліковим записом.

3.2.1.2 Вхідні дані

Користувач вводить свої дані у форму авторизації/реєстрації та натискає кнопку підтвердження. Або авторизується через обліковий запис Google

3.2.1.3 Обробка

Користувач вводить свої дані у форму авторизації/реєстрації та натискає кнопку підтвердження.

3.2.1.4 Результати

У разі правильних надісланих даних користувача перенаправляє на головну сторінку мобільного додатку, в іншому випадку отримує повідомлення про помилку.

3.2.1.5 Обробка помилок

Користувач не зможе зареєструватися/увійти у разі будь-якої помилки.

3.2.2 FR-2 Створення замовлення

3.2.2.1 Вступ

Основний процес створення замовлення зі сторони клієнта.

3.2.2.2 Вхідні дані

Користувач вводить інформацію про вантаж, його тип, габарити, місце призначення та час відправлення, зазначає необхідність у допомозі з завантаження чи розвантаження.

3.2.2.3 Обробка

Рисунок В.28 – Сторінка 28 специфікації пз

Замовлення перевіряється на валідність, оцінюється можливість його виконання, визначається його ціна та розпочинається підбір водіїв, що можуть виконати це замовлення.

3.2.2.4 Результати

У разі коректного створення замовлення воно валідується та переходить на етап пошуку водія, що зможе виконати замовлення. Також користувач отримує інформацію про орієнтовну вартість замовлення.

3.2.2.5 Обробка помилок

У разі помилки створення замовлення користувач отримує інформацію про причину помилки та контактами служби підтримки, для уточнення деталей.

3.2.3 FR-3 Оплата замовлення

3.2.3.1 Вступ

Клієнт повинен оплатити замовлення для того, щоб водій розпочав його виконання.

3.2.3.2 Вхідні дані

Водій підтвердив замовлення, створене клієнтом. Клієнт отримав сповіщення про це та кінцеву варітсть перевезення.

3.2.3.3 Обробка

Сторонній сервіс проводить оплату та передає результат оплати серверу. Сервер перевіряє, чи справді сторонній сервіс провів оплату, чи запит виконаний зловмисником.

3.2.3.4 Результати

У разі успішної оплати водій сповіщується про це та може виконувати замовлення.

3.2.3.5 Обробка помилок

Рисунок В.29 – Сторінка 29 специфікації пз

Користувач має змогу провести оплату кілька разів. Після чого замовлення буде відхилено або створено квиток у службу підтримки.

3.2.4 FR-4 Перегляд статусу замовлення в режимі реального часу

3.2.4.1 Вступ

Клієнт має можливість переглянути поточне місцезнаходження.

3.2.4.2 Вхідні дані

Підтверджене замовлення, що розпочало своє виконання. Водій підтвердив відправку до місця призначення. Водій передає своє місцезнаходження за допомогою увімкненого на мобільному пристрої GPS.

3.2.4.3 Обробка

Перевірка того, що GPS збігається з визначеним маршрутом перевезення та збереження поточного місцезнаходження, оцінка очікуваного часу прибуття.

3.2.4.4 Результати

Клієнт переглядає місцезнаходження вантажу та очікуваний час прибуття.

3.2.4.5 Обробка помилок

В разі помилки отримання місцезнаходження впродовж виначеного часу модератор служби підтримки зв’язується із водієм.

3.2.5 FR-5 Підбір замовлення для водія

3.2.5.1 Вступ

Система підбирає замовлення для водія та пропонує прийняти його.

3.2.5.2 Вхідні дані

Рисунок В.30 – Сторінка 30 специфікації пз

Водій вказує свої налаштування - радіус пошуку, або запланований маршрут, можливість бути вантажником, а також поточний GPS (надає дозвіл додатку на передачу).

3.2.5.3 Обробка

Система підбирає замовлення, що відповідають вказаним у налаштуваннях критеріях, та пропонує перелік підібраних замовлень.

3.2.5.4 Результати

Водій бачить список замовлень та приймає їх, або відхиляє. В разі відхилення, замовлення більше не з'являється у списку та передається наступному водієві.

3.2.5.5 Обробка помилок

В разі помилок з отриманням замовлень водієві буде запропоновано звернутись до служби підтримки застосунку.

3.2.6 FR-6 Звернення до служби підтримки

3.2.6.1 Вступ

Клієнт або водій має можливість звернутись до служби підтримки.

3.2.6.2 Вхідні дані

Текст звернення до служби підтримки.

3.2.6.3 Обробка

Система отримує звернення, котре було зареєстроване через телеграм - бота, ті віддає його команді технічної підтримки.

3.2.6.4 Результати

Модератор або клієнт закриває звернення тоді, коли клієнт чи водій задоволені отриманою відповіддю та допомогою.

3.2.6.5 Обробка помилок

В разі виникнення помилок, модератор сповіщує команду розробки про проблему під час звернення.

Рисунок В.31 – Сторінка 31 специфікації пз

3.2.7 FR-7 Перегляд статистики системи

3.2.7.1 Вступ

Адміністратор системи може переглядати статистику стосовно звернень та замовлень.

3.2.7.2 Вхідні дані

Відкриття сторінки статистики, з відповідним доступом

3.2.7.3 Обробка

Система перевіряє, чи має користувач достатньо прав для перегляду статистики, і якщо користувач має відповідні права, то віддає статистичні дані.

3.2.7.4 Результати

Відкриття сторінки з статистикою.

3.2.7.5 Обробка помилок

У разі виникнення помилок, користувачу буде відображене відповідне вікно, з тим причиною, чому саме він бачить цю помилку

3.2.8 FR-8 Модерація системи

3.2.8.1 Вступ

Адміністратор системи або модератор має право, у разі отримання запиту від користувача внести зміни у замовлення, виправити помилку в акаунті користувача, водія.

3.2.8.2 Вхідні дані

Рисунок В.32 – Сторінка 32 специфікації пз

Відкриття сторінки з користувачами або замовленнями

3.2.83 Обробка

Система перевіряє, чи має користувач достатньо прав для перегляду відповідної сторінки, та після процесу аутентифікації пропускає користувача далі.

3.2.8.4 Результати

Відкриття сторінки з користувачами системи або замовленнями

3.2.8.5 Обробка помилок

У разі виникнення помилок, користувачу буде відображене відповідне вікно, з тим причиною, чому саме він бачить цю помилку

3.2.8 FR-8 Детальний пошук для адміністраторів системи

3.2.8.1 Вступ

Адміністратор системи або модератор має право, у разі отримання запиту скористатися детальним пошуком у системі, з усіма можливими параметрами, для того, щоб знайти відповідне замовлення, або користувача/водія.

3.2.8.2 Вхідні дані

Відкриття сторінки з користувачами або замовленнями

3.2.83 Обробка

Рисунок В.33 – Сторінка 33 специфікації пз

Система перевіряє, чи має користувач достатньо прав для перегляду відповідної сторінки, та після процесу аутентифікації пропускає користувача далі.

3.2.7.4 Результати

Відкриття сторінки з користувачами системи або замовленнями

3.2.7.5 Обробка помилок

У разі виникнення помилок, користувачу буде відображене відповідне вікно, з тим причиною, чому саме він бачить цю помилку

3.3.1 Надійність

Надійність програмного забезпечення є критичною вимогою, оскільки система займається координацією та управлінням логістичними процесами, що впливають на щоденні операції користувачів і бізнесів. Для забезпечення високого рівня надійності програмного продукту, наступні критерії повинні бути виконані:

1. Відновлення після збоїв: Система повинна мати можливість автоматично відновлювати роботу після непередбачених збоїв. Це включає реалізацію стратегій на кшталт автоматичного перезапуску сервісів та використання транзакційної пам'яті для забезпечення цілісності даних.
2. Резервне копіювання даних: Періодичне резервне копіювання даних має бути автоматизовано. Система має забезпечувати легке відновлення даних з резервних копій у випадку їх втрати або пошкодження. Дозволяється налаштувати автоматичне резервне копіювання обраної СКБД.
3. Висока доступність: Система повинна бути розроблена з використанням архітектурних патернів, що підтримують високу доступність, включаючи класичний поділ на кластери, балансування навантаження та відмовостійкість.

Рисунок В.34 – Сторінка 34 специфікації пз

4. Швидке відновлення: Мінімізація часу відновлення системи після збою є обов'язковою. Система повинна мати здатність швидко відновлювати операції без значної втрати даних.
5. Моніторинг стану системи: Необхідно імплементувати інструменти для постійного моніторингу стану всіх компонентів системи, щоб забезпечити раннє виявлення потенційних проблем та уникнення непередбачених збоїв. Зокрема, за допомогою збору логів та метрик з Logstash до Elasticsearch.
6. Уніфікація середовища: для уникнення додаткових проблем під час розгортання компонентів системи необхідно створити проект, кожен елемент якого (сервіс) працює в Docker-контейнері, що працює однаково в оточенні розробника та production-оточенні за умови наявності однакових змінних середовища.

Ці вимоги до надійності забезпечують, що система може функціонувати ефективно і безперебійно, максимізуючи задоволеність користувачів і надійність бізнес-процесів.

3.3.2 Доступність

Система Wide Delivery має бути доступною для користувачів протягом більшої частини доби, враховуючи особливості ринку вантажних перевезень. Доступ до функціоналу платформи, такого як створення замовлень, пошук водіїв, відстеження доставки та спілкування з службою підтримки, буде забезпечено в проміжку часу з 6:30 до 24:00. Цей часовий діапазон обрано з урахуванням типових годин роботи більшості транспортних компаній та індивідуальних перевізників, а також потреб клієнтів, які, як правило, здійснюють замовлення на перевезення вантажів в денний та вечірній час.

Для забезпечення безпеки та конфіденційності даних, доступ до функціоналу Wide Delivery буде надано виключно авторизованим користувачам. Кожен користувач, незалежно від того, чи це клієнт, водій або представник транспортної компанії, повинен буде пройти процедуру реєстрації та створити особистий обліковий запис. Для входу в систему буде використана безпечна система авторизації, яка захистить облікові записи користувачів від несанкціонованого доступу.

3.3.4 Супроводжуваність

Рисунок В.35 – Сторінка 35 специфікації пз

Основна мета: Забезпечити легкість та ефективність обслуговування, оновлення, налаштування та розширення програмного продукту протягом усього життєвого циклу.

Опис атрибутів супроводжуваності:

1. **Модульність:** Система розроблена з використанням чітко визначених модулів, що дозволяє легко замінювати або оновлювати окремі компоненти без впливу на решту системи.
2. **Читабельність коду:** Програмний код має бути написаний з дотриманням стандартів кодування та коментування, що забезпечує його легкість для розуміння та подальшого супроводження розробниками.
3. **Документація:** Повна технічна документація системи має бути надана разом з продуктом. Документація повинна включати керівництва для розробників та користувачів, опис архітектури системи, а також процедури оновлення та вирішення проблем.
4. **Логування та моніторинг:** Система повинна включати розширені можливості логування та моніторингу, що дозволяє легко відслідковувати роботу програми та швидко виявляти та усувати помилки.
5. **Інтернаціоналізація:** Програмний продукт має підтримувати локалізацію для різних мов та регіональних налаштувань, що дозволяє адаптувати продукт для різних міжнародних ринків.
6. **Підтримка версій:** Всі оновлення програмного забезпечення мають супроводжуватись чітким веденням версій, що дозволяє користувачам легко переходити між версіями та відновлювати попередні конфігурації при необхідності.

Зауваження: Висока супроводжуваність забезпечує нижчі загальні витрати на володіння продуктом та збільшує задоволеність користувачів, що є критично важливим для довгострокового успіху програмного продукту.

Рисунок В.36 – Сторінка 36 специфікації пз

3.3.5 Переносимість

Проект має бути налаштований для роботи у контейнеризованому середовищі за допомогою Docker-compose, що дозволяє забезпечити легке та швидке розгортання на різних платформах: Ubuntu, Windows, і MacOS. Розгортання повинно відбуватися за допомогою однієї команди (`make start`), що спрощує процес ініціації та управління.

Конфігурація повинна включати всі необхідні сервіси та залежності, які описуються в docker-compose.yml файлі для кожного з репозиторіїв. Усі репозиторії (окремі сервіси) мають бути поєднаними в один проект. Додатково в кожному з репозиторіїв потрібно створити Makefile, що містить команду make start, яка автоматизує процес розгортання контейнерів та залежностей.

Таким чином гарантується ідентичність середовища та легкий запуск з будь-якої платформи розробки чи розгортання.

3.3.6 Продуктивність

Система Wide Delivery повинна демонструвати високу продуктивність та забезпечувати швидку реакцію на дії користувачів, навіть за умов значного навантаження. Мобільний додаток повинен бути оптимізований для роботи на пристроях з різними характеристиками та забезпечувати плавну роботу інтерфейсу без помітних затримок. Зокрема, головний екран додатку, що містить інтерактивну карту та форми для введення даних, повинен завантажуватись менше ніж за 2 секунди. Це забезпечить позитивний користувацький досвід та дозволить клієнтам швидко розпочати процес замовлення перевезення.

Серверна частина Wide Delivery повинна бути спроможна обробляти запити від великої кількості користувачів одночасно, забезпечуючи стабільну роботу платформи навіть у періоди пікового навантаження. Система має бути спроектована з урахуванням можливості масштабування, що дозволить збільшувати її потужність паралельно зі зростанням кількості користувачів та замовлень.

3.4 Вимоги бази даних

Необхідно обрати NoSQL базу даних, що відповідає PA/EC рішенню за теоремою PACELC. Висока доступність та консистентність даних є

Рисунок В.37 – Сторінка 37 специфікації пз

важливими вимогами до системи. База даних має легко масштабуватись та легко піддаватись партиціонуванню. Чудовим кандидатом для такої БД є MongoDB.

Також систему необхідно забезпечити сховищем кешованих даних для швидкого доступу до потрібної інформації. Зокрема, такою інформацією є сесії користувачів, що виконали аутентифікацію та використовують мобільний застосунок, а також дані для роботи чат-боту служби підтримки в Telegram.

Логи та метрики, зокрема щодо запитів користувачів, повинні зберігатись у пошуковому сервері Elasticsearch задля можливості швидкого доступу до напівструктурованої інформації.

3.5 Інші вимоги

Кожна зміна документу має бути обговорена командою розробників і власником проекту. Після внесення цих змін - підписана кожним із них.

Рисунок В.38 – Сторінка 38 специфікації пз

ДОДАТОК Г

Приклад програмного коду (OrderScreen)

```
package com.example.wide_delivery.ui.screens.order

import android.widget.Toast
import androidx.compose.foundation.background
import androidx.compose.foundation.border
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.heightIn
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.AccessTime
import androidx.compose.material.icons.filled.Check
import androidx.compose.material.icons.filled.Close
import androidx.compose.material.icons.filled.DateRange
import androidx.compose.material.icons.filled.DirectionsCar
import androidx.compose.material.icons.filled.GpsFixed
import androidx.compose.material3.Button
import androidx.compose.material3.DropdownMenuItem
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.ExposedDropdownMenuBox
import androidx.compose.material3.ExposedDropdownMenuDefaults
import androidx.compose.material3.FilledTonalButton
import androidx.compose.material3.Icon
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.OutlinedTextField
import androidx.compose.material3.RadioButton
import androidx.compose.material3.Switch
import androidx.compose.material3.Text
import androidx.compose.material3.TextField
import androidx.compose.material3.TextFieldDefaults
```

```
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.derivedStateOf
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.input.ImeAction
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController
import androidx.navigation.compose.rememberNavController
import com.vanpra.composematerialdialogs.MaterialDialog
import com.vanpra.composematerialdialogs.datetime.date.datepicker
import com.vanpra.composematerialdialogs.datetime.time.timepicker
import com.vanpra.composematerialdialogs.rememberMaterialDialogState
import java.time.LocalDate
import java.time.LocalTime
import java.time.format.DateTimeFormatter

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun OrderScreen(navController: NavController, viewModel: OrderViewModel) {
    val state by viewModel.state.collectAsState()
    val context = LocalContext.current



    Column(
        modifier = Modifier
            .padding(20.dp)
            .fillMaxWidth()


            .verticalScroll(rememberScrollState()),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally

    ) {
        Text(text = "Your Order", style = MaterialTheme.typography.displaySmall)
```

```
SetStartDestinationField(navController = navController, viewModel = viewModel)

SetEndDestinationField(navController = navController, viewModel = viewModel)


Text(
    text = "Choose your cargo type",
    style = MaterialTheme.typography.headlineMedium
)

AccurateSizeSwitch(viewModel = viewModel)

when (state.isAccurateSize) {
    true -> {
        Row(
            Modifier
                .fillMaxSize()
                .padding(top = 15.dp, bottom = 30.dp),
            horizontalArrangement = Arrangement.SpaceBetween,
            verticalAlignment = Alignment.CenterVertically

        ) {
            OutlinedTextField(
                value = state.cargoLength,
                onValueChange = { viewModel.onCargoLengthChanged(it) },
                label = { Text(text = "Length") },
                shape = MaterialTheme.shapes.extraLarge,
                keyboardOptions = KeyboardOptions(
                    keyboardType = KeyboardType.Number,
                    imeAction = ImeAction.Next
                ),
                modifier = Modifier.width(100.dp)
            )
            Spacer(modifier = Modifier.weight(1f))

            Text(text = "X", style = MaterialTheme.typography.bodyMedium)
            Spacer(modifier = Modifier.weight(1f))

            OutlinedTextField(
                value = state.cargoWidth,
                onValueChange = { viewModel.onCargoWidthChanged(it) },
                label = { Text(text = "Width") },
                shape = MaterialTheme.shapes.extraLarge,
                keyboardOptions = KeyboardOptions(
```

```
                    keyboardType = KeyboardType.Number,
                    imeAction = ImeAction.Next
                ),
                modifier = Modifier.width(100.dp)
            )
            Spacer(modifier = Modifier.weight(1f))
            Text(text = "X", style = MaterialTheme.typography.bodyMedium)
            Spacer(modifier = Modifier.weight(1f))
            OutlinedTextField(
                value = state.cargoHeight,
                onValueChange = { viewModel.onCargoHeightChanged(it) },
                label = { Text(text = "Height") },
                shape = MaterialTheme.shapes.extraLarge,
                keyboardOptions = KeyboardOptions(
                    keyboardType = KeyboardType.Number,
                    imeAction = ImeAction.Next
                ),
                modifier = Modifier.width(100.dp)
            )
        }
    }

    false -> {
        Row(
            modifier = Modifier
                .fillMaxSize()
                .width(100.dp)
                .padding(top = 15.dp, bottom = 30.dp),
            horizontalArrangement = Arrangement.SpaceBetween,
            verticalAlignment = Alignment.CenterVertically
        ) {
            CarTypeButton("0 - 100kg")
            Spacer(modifier = Modifier.weight(1f))
            CarTypeButton("100 - 500kg")
            Spacer(modifier = Modifier.weight(1f))
            CarTypeButton("Up to 1t")
        }
    }
}

OnTimeSwitch(viewModel = viewModel)

when (state.isOnTime) {
    true -> {
        DateTimePicker(viewModel = viewModel)
```

```
            }

            false -> viewModel.setDateTime()
        }

        Spacer(modifier = Modifier.height(20.dp))

        Row(
            modifier = Modifier,
            verticalAlignment = Alignment.CenterVertically,
            horizontalArrangement = Arrangement.SpaceBetween
        ) {
            Text(
                text = "Need help with your cargo?",
                style = MaterialTheme.typography.headlineSmall
            )
            Spacer(modifier = Modifier.weight(1f))
            RadioButton(
                selected = state.needLoader,
                onClick = { viewModel.onNeedLoaderChanged() })
        }
        Spacer(modifier = Modifier.height(20.dp))
        DescriptionField(viewModel = viewModel)
        Spacer(modifier = Modifier.height(35.dp))

        OrderButton()
    }

}


@Composable
private fun AccurateSizeSwitch(viewModel: OrderViewModel) {
    val state by viewModel.state.collectAsState()


    Row(
        modifier = Modifier
            .padding(top = 5.dp),
        verticalAlignment = Alignment.CenterVertically
    ) {
        Text(text = "Accurate size", style = MaterialTheme.typography.headlineSmall)
        Spacer(modifier = Modifier.weight(1f))
        Switch(
            checked = state.isAccurateSize,
```

```
            onCheckedChange = { viewModel.onIsAccurateSizeChanged() },
            thumbContent = {
                Icon(
                    imageVector = if (state.isAccurateSize) {
                        Icons.Default.Check
                    } else {
                        Icons.Default.Close
                    },
                    contentDescription = null
                )
            }
        )
    }
}

@Composable
private fun OnTimeSwitch(viewModel: OrderViewModel) {
    val state by viewModel.state.collectAsState()


    Row(
        modifier = Modifier
            .padding(top = 20.dp),
        verticalAlignment = Alignment.CenterVertically
    ) {
        Text(text = "On Time", style = MaterialTheme.typography.headlineSmall)
        Spacer(modifier = Modifier.weight(1f))
        Switch(
            checked = state.isOnTime,
            onCheckedChange = { viewModel.onIsOnTimeChanged() },
            thumbContent = {
                Icon(
                    imageVector = if (state.isOnTime) {
                        Icons.Default.Check
                    } else {
                        Icons.Default.Close
                    },
                    contentDescription = null
                )
            }
        )
    }
}

@OptIn(ExperimentalMaterial3Api::class)
```

```
@Composable
private fun SetStartDestinationField(
    navController: NavController, viewModel: OrderViewModel
) {
    val state by viewModel.state.collectAsState()

    Column(
        modifier = Modifier.padding(
            top = 10.dp, bottom = 10.dp
        ), verticalArrangement = Arrangement.SpaceBetween,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        OutlinedTextField(
            value = "Your current location",
            onValueChange = { viewModel.onDepartureLatitudeChanged(it) },

            label = { Text(text = "From:") },
            shape = MaterialTheme.shapes.extraLarge,
            readOnly = true,
            leadingIcon = {
                Icon(Icons.Filled.GpsFixed, contentDescription = "GPS icon")
            },
        )
    }

}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
private fun SetEndDestinationField(
    navController: NavController, viewModel: OrderViewModel
) {
    /*val state by viewModel.state.collectAsState()
    var query by remember { mutableStateOf("") }
    var expanded by remember { mutableStateOf(false) }

    Column(
        modifier = Modifier
            .padding(top = 10.dp, bottom = 10.dp)
            .fillMaxWidth(),
        verticalArrangement = Arrangement.SpaceBetween,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        OutlinedTextField(
            value = query,
```

```
            onValueChange = {
                query = it
                viewModel.onDestinationQueryChanged(it) // Trigger backend search
                expanded = true
            },
            label = { Text(text = "To:") },
            shape = MaterialTheme.shapes.extraLarge,
            modifier = Modifier
                .padding(bottom = 20.dp)
                .fillMaxWidth(),
            keyboardOptions = KeyboardOptions(
                keyboardType = KeyboardType.Text,
                imeAction = ImeAction.Done
            ),
            interactionSource = remember { MutableInteractionSource() }, // Added this
line
        )

        DropdownMenu(
            expanded = expanded,
            onDismissRequest = { },
            modifier = Modifier.fillMaxWidth()
        ) {
            state.suggestions.forEach { suggestion ->
                DropdownMenuItem(
                    text = { Text(text = suggestion.name) },
                    onClick = {
                        query = suggestion.name
                        viewModel.onDestinationSelected(suggestion)
                        expanded = false
                    })

            }
        }
    }*/

    val context = LocalContext.current
    val coffeeDrinks = arrayOf("Americano", "Cappuccino", "Espresso", "Latte", "Mocha")
    var expanded by remember { mutableStateOf(false) }
    var selectedText by remember { mutableStateOf("") }

    Box(
        modifier = Modifier
            .fillMaxWidth()
            .padding(32.dp)
```

```
    ) {
        ExposedDropdownMenuBox(
            expanded = expanded,
            onExpandedChange = {
                expanded = !expanded
            }
        ) {
            OutlinedTextField(
                value = selectedText,
                onValueChange = { selectedText = it },
                label = { Text(text = "To:") },

                trailingIcon = { ExposedDropdownMenuDefaults.TrailingIcon(expanded =
expanded) },
                shape = MaterialTheme.shapes.extraLarge,
                keyboardOptions = KeyboardOptions(
                    keyboardType = KeyboardType.Text,
                    imeAction = ImeAction.Done
                ),
                modifier = Modifier.menuAnchor()
            )

            val filteredOptions =
                coffeeDrinks.filter { it.contains(selectedText, ignoreCase = true) }
            if (filteredOptions.isNotEmpty()) {
                ExposedDropdownMenu(
                    expanded = expanded,
                    onDismissRequest = {
                        // We shouldn't hide the menu when the user enters/removes any
character
                    }
                ) {
                    filteredOptions.forEach { item ->
                        DropdownMenuItem(
                            text = { Text(text = item) },
                            onClick = {
                                selectedText = item
                                expanded = false
                                Toast.makeText(context, item,
Toast.LENGTH_SHORT).show()
                            }
                        )
                    }
                }
            }
```

```
        }
    }
}

@Composable
private fun CarTypeButton(carType: String) {

    FilledTonalButton(
        onClick = {

            when (carType) {
                "0 - 100kg" -> {
                    //TODO
                }

                "100 - 500kg" -> {
                    //TODO
                }

                "Up to 1t" -> {
                    //TODO
                }
            }
        },

        ) {
        Column(
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Icon(Icons.Filled.DirectionsCar, contentDescription = "Back")
            Text(text = carType)
        }
    }
}


@Composable
private fun DateTimePicker(viewModel: OrderViewModel) {
    val state by viewModel.state.collectAsState()

    var pickedDate by remember {
        mutableStateOf(state.pickedDate)
    }
    var pickedTime by remember {
        mutableStateOf(state.pickedTime)
```

```
}
val formattedDate by remember {
    derivedStateOf {
        DateTimeFormatter
            .ofPattern("yyyy-MM-dd")
            .format(pickedDate)
    }
}

val formattedTime by remember {
    derivedStateOf {
        DateTimeFormatter
            .ofPattern("kk:mm:ss")
            .format(pickedTime)
    }
}

val dateDialogState = rememberMaterialDialogState()
val timeDialogState = rememberMaterialDialogState()

Row(
    modifier = Modifier
        .fillMaxSize(),
    horizontalArrangement = Arrangement.SpaceBetween,
    verticalAlignment = Alignment.CenterVertically
) {

    Button(onClick = {
        dateDialogState.show()
    }) {
        Icon(Icons.Filled.DateRange, contentDescription = "Date icon")
        Text(text = "Pick date")
    }
    Spacer(modifier = Modifier.height(16.dp))

    Button(onClick = {
        timeDialogState.show()
    }) {
        Icon(Icons.Filled.AccessTime, contentDescription = "Time icon")
        Text(text = "Pick time")

    }

}
MaterialDialog(
```

```
        dialogState = dateDialogState,
        buttons = {
            positiveButton(text = "Ok") {

            }
            negativeButton(text = "Cancel")
        }
    ) {
        datepicker(
            initialDate = LocalDate.now(),
            title = "Pick a date",
            allowedDateValidator = {
                it.isAfter(LocalDate.now())
            }
        ) {
            pickedDate = it
        }
    }
    MaterialDialog(
        dialogState = timeDialogState,
        buttons = {
            positiveButton(text = "Ok") {

            }
            negativeButton(text = "Cancel")
        }
    ) {
        timepicker(
            initialTime = LocalTime.NOON,
            title = "Pick a time",
            is24HourClock = true
        ) {
            pickedTime = it
        }
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
private fun DescriptionField(viewModel: OrderViewModel) {
    val state by viewModel.state.collectAsState()
    var description by remember { mutableStateOf(state.description) }


    TextField(
```

```
        value = description,
        onValueChange = {
            description = it
            viewModel.onDescriptionChanged(it)
        },
        label = { Text(text = "Description") },
        modifier = Modifier
            .heightIn(min = 56.dp, max = 200.dp)
            .fillMaxWidth()
            .border(1.dp, Color.Gray, MaterialTheme.shapes.extraLarge)
            .background(Color.Gray, MaterialTheme.shapes.extraLarge),
        shape = MaterialTheme.shapes.extraLarge,
        colors = TextFieldDefaults.textFieldColors(
            containerColor = Color.Gray,
            focusedIndicatorColor = Color.Transparent,
            unfocusedIndicatorColor = Color.Transparent
        )
    )
}

@Composable
private fun OrderButton() {
    FilledTonalButton(
        onClick = { /*TODO*/ },
        modifier = Modifier.fillMaxWidth()
    )
    {
        Text(text = "Order")
    }
}

@Preview
@Composable
fun OrderScreenPreview() {
    val navController = rememberNavController()
    val viewModel = OrderViewModel()
    OrderScreen(navController = navController, viewModel = viewModel)
}
```