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. Front-end.

(тема)

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

Овсянніков А.В.
(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 Овсяннікову Антону Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система керування вантажними перевезеннями. Front-end.
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 Front-end частину програмної системи для контролю керування вантажними перевезеннями
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, впровадження, 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	05.06.2024	*виконано*
10	Здача роботи у електронний архів	10.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	10.06.2024	*виконано*

Дата видачі завдання 08 травня 2024р.

Студент _____ Овсянніков А.В.
(підпис)

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

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра: 95 с., 70 рис., 10 джерел.

REACT, SASS, TS, MATERIALUI V.5, ZUSTAND, ВОДІЙ, ЗАМОВНИК, ДОСТАВКА, ПОДОРОЖ.

Об’єкт розробки – front-end частина програмної системи для керуванням вантажними перевезеннями.

Мета розробки – створення програмної системи для керуванням вантажними перевезеннями, реалізація необхідної функціональності для контролю за якістю вантажних перевезень, вирішення конфліктних ситуацій виникнувших під час перевезення та аналіз статистики стосовно перевезень.

Як метод вирішення цієї задачі, було обрано середовище для розробки VS Code, мови програмування – TS на базі використання бібліотекою React з компіляцією через збиральник Vite. Для стилізації була обрана технологія MaterialUI v.5 у поєднанні з препроцессором SASS та як менеджер стану був обраний сучасний продукт Zustand.

У результаті розробки була створена front-end частина програмної системи для керуванням вантажними перевезеннями з можливістю контролю перевезення вантажу, вирішення конфліктних ситуацій між замовником та водієм та аналізом статистики перевезень.

REACT, SASS, TS, MATERIALUI V.5, ZUSTAND, DRIVER, CUSTOMER, DELIVERY, TRIP.

The object of development is the front-end part of a software system for freight transportation management.

The purpose of the pre-certification practice is to create a software system for managing freight transportation, implement the necessary functionality to control the

quality of freight transportation, resolve conflict situations that arise during transportation, and analyze transportation statistics.

As a method of solving this problem, the VS Code development environment was chosen, the programming language was TS based on the use of the React library with compilation through the Vite builder. The MaterialUI v.5 technology in combination with the SASS preprocessor was chosen for styling, and the modern Zustand product was chosen as a state manager.

As a result of the development, the front-end part of the software system for managing freight transportation was created with the ability to control cargo transportation, resolve conflicts between the customer and the driver, and analyze transportation statistics.

Я, Овсянніков Антон Вікторович, 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. Front-end.», 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

Головною метою створення програмної система «Керуванням вантажними перевезеннями. Front-end.» було надати користувачу можливість підвищіти рівень контролю за вантажними перевезеннями, швидке та ефективне вирішення можливих конфліктних ситуації між водієм та клієнтом під час вантажного перевезення та аналіз статистики стосовно перевезень.

Було розроблено front-end частину системи «Керуванням вантажними перевезеннями», яка розроблена з метою надання користувачам зручних інструментів для контролю та моніторингу стану вантажного перевезення. Ця система також забезпечує користувачів актуальною інформацією щодо змін під час вантажного перевезення за допомогою взаємодії через браузер.

Архітектурною основою системи є змішана «клієнт-серверна» архітектура, яка дозволяє розділити функціональність системи між клієнтською та серверною частинами. Так як під час виконанні практики, відповідальність полягає у розробленні front-end частини, то для цього була обрана «feature-sliced design», яка дозволяє розділити частини функціональної системи на дрібні шматочки, що у майбутньому дозволяє збирати елементи та компоненти системи значно швидше, та рентабельно розділяти бізнес-логіку та UI компоненти системи.

Програмна система також має трьохшарову архітектуру, яка включає рівні представлення, логіки та даних.

Загалом, система є насиченим Інтернет-застосунком, що враховує різноманітні фактори, такі як навантаження на систему, особливості предметної області та залежності від апаратного обладнання, інших програмних систем та API.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Люди завжди стикалися з проблемною очікування, будь то пошта, або якісь інші речі. Ця проблема була в переважному випадку в усіх аспектах людства, як такового. Ситуації, коли щось потрібно терміново передати, стають звичними, але доставка, яка у приватному випадку могла б зайняти лише кілька годин, іноді затягується на дні, а то й тижні. Саме із-за цього і створюється потреба у більш швидкому способі доставки посилок, або вантажів. Однією з можливих способів вирішення цієї проблеми і є доставка через водія, як посередника між тим хто відправив та тим хто отримує. Саме такий підхід до вирішення цієї проблеми забезпечує швидкі строки доставки, проте супроводжується певними ризиками та, можливо, не завжди прийнятними цінами для користувачів.

Одним з переважних та ключових аспектів створення ефективної системи доставки без ризиків та з гарною швидкістю виконання є повсюдне впровадження нових технологій та телекомунікації, які надають значного більше можливостей для відстеження посилок та вантажів. Сучасні системи GPS-моніторингу, які використовуються навіть у морському флоті, забезпечують чітку роботу та точність надання геопозицій. Це значно підвищує ефективність у різних галузях, включаючи цивільний сектор.

GPS-технології дозволяють відстежувати місцезнаходження вантажу в режимі реального часу, що надає можливість не тільки оптимізувати маршрути перевезення, але й забезпечити прозорість процесу доставки для клієнтів.

Використання таких систем значно знижує ризики втратити або затримати вантаж, та дозволяє швидко реагувати на непередбачувані ситуації, котрі можуть статися під час перевезення. Завдяки цьому, клієнти можуть відслідковувати свій вантаж на всіх етапах доставки, що підвищує рівень довіри до компанії та задоволення від послуг.

Значне зростання конкуренції та постійні зміни умов ринку змушують підприємства, котрі вже маніполюзували цей сегмент ринку шукати шляхи для

зниження витрат і оптимізації логістичних процесів. Це передбачає ефективне використання ресурсів, мінімізацію часу та витрат на перевезення, а також підвищення задоволенності клієнтів. В умовах жорсткої конкуренції, компанії повинні знаходити інноваційні рішення, щоб залишатися актуальними та забезпечувати високий рівень обслуговування.

Ефективне управління логістикою включає в себе аналіз поточних процесів, виявлення вузьких місць і впровадження інноваційних рішень. Використання сучасних технологій, таких як системи управління транспортом (TMS), дозволяє автоматизувати процеси планування і відстеження перевезень, що сприяє скороченню витрат і підвищенню ефективності. TMS-системи забезпечують оптимізацію маршрутів, управління автопарком, контроль за виконанням завдань та багато іншого. Це дає можливість логістичним компаніям оперативно реагувати на зміну умов ринку і швидко адаптуватися до нових вимог.

Зі зростанням обсягів перевезень і розширенням географії поставок стає важливим ефективне управління ризиками, такими як затримки в доставці, транспортні аварії, зміни умов і регуляційне середовище. Використання аналітики даних може допомогти передбачити такі ризики і розробити стратегії для їх управління. Аналітика даних дозволяє прогнозувати можливі проблеми, що можуть виникнути в ланцюзі поставок, і завчасно вжити заходів для їх уникнення.

Аналітика даних дозволяє виявляти закономірності і тренди, що можуть вказувати на потенційні проблеми в ланцюзі поставок. Це дозволяє компаніям розробляти превентивні заходи і плани дій у випадку виникнення непередбачуваних ситуацій. Наприклад, аналіз історичних даних може допомогти передбачити сезонні піки попиту і заздалегідь підготуватися до них, збільшуючи запаси або залучаючи додаткові ресурси. Це знижує ризики втрат та затримок, підвищуючи загальну ефективність логістичних операцій.

Доставка через водія є одним із способів швидкої передачі посилок, що забезпечує виконання замовлення в короткі терміни. Основні переваги такого підходу включають в себе швидкість доставки, можливість гнучкого планування маршрутів і мінімізацію формальностей, пов’язаних з перевезенням. Доставка

через водія особливо актуальна в умовах міської інфраструктури, де важливо швидко і оперативно доставляти вантажі в різні частини міста.

Однак, цей метод має і певні ризики. По-перше, безпека вантажу може бути під загрозою, особливо якщо доставка здійснюється не через офіційні канали. Водій може не мати достатнього досвіду або надійності, що може призвести до пошкодження або втрати вантажу. По-друге, вартість такої послуги може бути значно вищою, ніж стандартні методи доставки, що робить її менш доступною для широкого кола клієнтів. Крім того, непередбачувані обставини, такі як дорожні затори або технічні несправності автомобіля, можуть затримати доставку.

Розвиток технологій і зміна ринкових умов створюють нові виклики і можливості для логістичних компаній. Одним із ключових завдань є інтеграція різних систем і технологій для створення єдиної ефективної платформи управління доставками. Це включає в себе не тільки впровадження нових технологій, але й навчання персоналу, розробку нових бізнес-моделей і вдосконалення процесів взаємодії з клієнтами.

Інтеграція різних технологій, таких як IoT, блокчейн та штучний інтелект, дозволяє створити більш ефективні та надійні системи управління ланцюгами поставок. Наприклад, використання IoT дозволяє відстежувати стан вантажу в режимі реального часу, що забезпечує більш високий рівень контролю за якістю доставки. Блокчейн-технології можуть забезпечити прозорість та незмінність даних про переміщення вантажу, що знижує ризики шахрайства та помилок.

У сучасному світі, де швидкість і ефективність доставки відіграють вирішальну роль, впровадження нових технологій і оптимізація логістичних процесів стають критично важливими. Доставка через водія може стати ефективним рішенням для термінових перевезень, проте для успішної реалізації цієї моделі необхідно враховувати всі ризики і розробляти стратегії їх управління. Використання сучасних технологій і аналітики даних дозволяє забезпечити високий рівень прозорості та надійності доставки, що в кінцевому підсумку підвищує задоволеність клієнтів і конкурентоспроможність компаній на ринку.

Таким чином, розвиток логістичних технологій та впровадження інноваційних рішень є ключовими факторами успіху в умовах сучасного ринку. Компанії, що здатні ефективно використовувати новітні досягнення науки і техніки, мають більше шансів на успіх та збагачення в умовах жорсткої конкуренції.

1.2 Виявлення проблем та актуалізація рішень

Проаналізувавши ринок споживачів були визначені актуальні конкуренти, котрі на даний момент будуть суттєво сильнишими ніж ми на старті впровадження нашого системного продукту. Конкурентами на даний момент являются Uber Freight та Convoy, котрі працюють за суміжною схемою «водій посередник». В них є суттєві переваги та недоліки, котрі були перевірені безпосередньо мною під час аналізу ринку. Під час аналізу були використані: офіційні коментарі, відгуки на сервіси та спілкування зі споивачами котрі користуються одним з продуктів наших конкурентів.

З переваг що були зазначені в обох продуктах конкурентів були:

- швидке знаходження вантажів та водіїв;
- зручність використання та інтуїтивно зрозумілий інтерфейс;
- можливість відстеження статусу перевезення у реальному часі.

Але також були відзначені й мінуси, котрі зустрічаються в обох конкурентів водночас, а саме:

- високий відсоток комісії та приховані витрати;
- обмежений вибір водіїв та можливості оптимізації маршруту;
- недостатня прозорість та надійність у виконанні замовлення.

В свою чергу наша система буде вирізнятися наступними перевагами:

- покращена прозорість та надійність: Ми забезпечимо надійний рівень захищеності та доставки, що дозволяє клієнтам з впевненістю користуватися нашим продуктом, не перейматися із-за якості доставлення вантажу та більш чітко планувати свої логістичні процеси;

- низька комісія та прозорість витрат: Наші витрати будуть прозорі та нижчі за продукт конкурентів, що забезпечує економічну вигоду для клієнтів і в той же самий час буде суттєвою перевагою при виборі сервісу на ринку;
- гнучкість та адаптивність: Ми надаємо клієнтам гнучко та ефективно обирати водіїв, що надає переваги нашому продукту, як логістичній складовій в умовах ринку, перед нашими конкурентами.

Ринок, на який орієнтується програмна система, в основному націлена на сегмент логістики та транспортних послуг. Цей ринок включає в себе широкий спектр учасників, таких як: вантажні компанії, логістичні оператори, вантажні брокери, дистриб'ютори та інші потенційні користувачі, котрі потребують перевезення вантажів.

Модель котру використовує наша система, це модель «водій-посередник», котра виникла як альтернатива традиційним методам доставки вантажів. Під традиційними методами доставки вантажів ми розуміємо такі методи доставки вантажів, як пошта та спеціальні логістичні компанії. Але все ж таки виникла альтернатива і це саме нова модель «водій-посередник», тому виникає питання «Чому?».

Є декілька причин стосовно цього методу доставки, одним з них є зростання онлайн-торгівлі. Так як зростання онлайн-торгівлі є ключовим фактором, що зумовив появу стратегії доставки через посередників. З розвитком електронної комерції значно зросла кількість невеликих замовлень, які потребують швидкої та гнучкої доставки.

По-перше, онлайн-покупки призводять до збільшення попиту на швидкі та невеликі відправлення. Клієнти, які здійснюють покупки онлайн, часто очікують швидкої доставки, особливо у випадку невеликих товарів або легких вантажів. Це ставить перед бізнесом завдання забезпечити ефективну та швидку доставку, щоб задовольнити потреби своїх клієнтів.

По-друге, електронна комерція також призводить до зростання потреби у гнучкій доставці. Клієнти часто очікують можливості вибору часу і місця доставки,

а також можливості відстежувати своє замовлення в режимі реального часу. Це створює попит на доставку, яка може бути здійснена швидко, точно і відповідно до вимог клієнта.

Другою причиною появи цієї стратегії було збільшення культури спільного споживання, а саме люди стали більше ділитися ресурсами один з одним, що сприяло розвитку платформ, які з'єднують водіїв з клієнтами, які потребують доставки. Зростаюча популярність ринкової економіки відіграє значну роль у появі стратегії посередницької доставки водіїв. Ця тенденція відображає загальний рух суспільства до ефективного використання ресурсів і сприяє розвитку платформ, які з'єднують водіїв з клієнтами, що потребують доставки.

По-перше, зростання популярності ринкової економіки означає, що люди більш схильні ділитися ресурсами та надавати послуги один одному. Це створює попит на нові та більш ефективні методи доставки, які відповідають потребам спільного використання ресурсів.

По-друге, ця тенденція сприяла розвитку платформ, які з'єднують водіїв з клієнтами, що потребують доставки. Такі платформи забезпечують зручний і простий спосіб замовлення та отримання послуг доставки, а також надійний механізм отримання замовлень водіями.

Крім того, шерингова економіка створює попит на більш екологічні та ефективні методи доставки. Використання водіїв-посередників дозволяє зменшити кількість порожніх рейсів, а також допомагає оптимізувати маршрути, що сприяє скороченню викидів CO2.

Також, останньою за списком, але не за знаковістю є поширення смартфонів і мобільних технологій, що відіграло важливу роль у розвитку стратегії доставки за участі «водія-посередника». Ці технології зробили процес координації та управління доставкою більш зручним та ефективним.

Перш за все, смартфони дозволяють водіям і кур'єрам легко координувати свої дії під час виконання замовлень. Завдяки спеціалізованим додаткам водії можуть отримувати детальну інформацію про маршрути, замовлення та вимоги

клієнтів безпосередньо на свій смартфон. Це дозволяє їм ефективно планувати маршрути, уникати заторів і забезпечувати своєчасну доставку.

Крім того, мобільні технології дозволяють відстежувати замовлення в режимі реального часу. Водії можуть бачити поточний статус замовлення, включаючи інформацію про прибуття, виконання та підтвердження доставки. Це сприяє підвищенню рівня обслуговування та задоволеності клієнтів.

Крім того, мобільні технології дозволяють водіям отримувати оплату безпосередньо через мобільні додатки. Це забезпечує швидку та безпечну оплату послуг, спрощує процес фінансових транзакцій та підвищує зручність як для водіїв, так і для клієнтів.

Таким чином, усі вище зазначені у цій частині причини появи альтернативного методу доставки «водій-посередник» сходились на вирішенні майже однієї глобальної проблеми користувача, котра полягала у найшвидшій з можливих доставці вантажів, з гнучким логістичним налаштуванням, або зв’язку з водієм для подальшої координації та унікальним вирішення можливих бюрократичних моментів, таких як переведення платні та інше.

Модель «водій-посередник» може стати серйозним конкурентом для поштових служб, особливо в сегментах ринку невеликих і термінових відправлень.

Зростаюча популярність онлайн-покупок призводить до великого обсягу дрібних замовлень, з якими поштові служби не завжди можуть ефективно впоратися. Крім того, незручність поштових відділень та обмежена гнучкість графіку роботи можуть відштовхнути клієнтів. За таких обставин модель водій-посередник забезпечує швидку та гнучку доставку, може працювати 24/7 та пропонує персоналізований підхід до клієнтів, що робить її привабливою альтернативою для тих, хто шукає ефективні та зручні способи доставки.

Однак важливо зазначити, що поштові служби все ще мають низку переваг, які можуть дати їм конкурентну перевагу над моделлю «водій-посередник»:

– розгалужена мережа: Поштові служби мають розгалужену мережу відділень по всій країні, що дозволяє їм доставляти товари в будь-яке

місце. Це особливо важливо для клієнтів, які потребують доставки у невідомі або віддалені місця;

- надійність: Поштові служби мають багаторічний досвід доставки товарів і гарантують безпечну та надійну доставку. Клієнти можуть бути впевнені, що їхній вантаж буде доставлений вчасно і без пошкоджень;
- додаткові послуги: Поштові служби пропонують додаткові послуги, такі як страхування вантажів, відстеження замовлень і можливість повернення товарів. Це робить їхню пропозицію більш привабливою для клієнтів, які цінують додатковий сервіс і захист своїх товарів.

Враховуючи ці переваги, поштові служби можуть зберегти свою популярність серед клієнтів, які цінують широке покриття, надійність і додаткові послуги. Однак їм також необхідно адаптуватися до мінливих ринкових умов і конкурувати з моделями доставки, які пропонують більшу гнучкість і персоналізований підхід до клієнтів.

Очікується, що в майбутньому конкуренція між моделлю водія-посередника і поштовою службою посилиться, оскільки обидві моделі пропонують схожі послуги - доставку товарів. Переможцями цієї конкуренції стануть ті компанії, які можуть запропонувати найкращі ціни, найвищу якість послуг та найзручніший клієнтський досвід.

Модель «водій-посередник» має кілька переваг перед поштовим сервісом. По-перше, вона може забезпечити швидку та гнучку доставку, оскільки використовує водіїв, які перебувають у безпосередній близькості до місця призначення. Це дозволяє скоротити час доставки і знизити транспортні витрати. Крім того, «водії-посередники» можуть забезпечити персоналізований підхід до клієнтів і вирішити конкретні проблеми або запити, що може покращити загальний досвід користувачів.

Однак модель «водія-посередника» також має свої виклики. Наприклад, необхідна ефективна координація між водіями та клієнтами, щоб уникнути затримок доставки та конфліктів. Також важливо мати надійну систему контролю

якості та безпеки, оскільки «водії-посередники» можуть працювати на різних рівнях професіоналізму та надавати різні рівні послуг.

Тому, щоб успішно конкурувати з поштовими службами, компанії, що використовують модель водій-посередник, повинні створювати найкращі умови як для водіїв, так і для клієнтів, забезпечуючи ефективну координацію, високу якість обслуговування та конкурентоспроможні ціни.

1.3 Постановка задачі

В індустрії доставки існує певна кількість проблем, які створюють бар'єри для клієнтів і можуть вплинути на їхню задоволеність послугою.

Однією з головних проблем є відсутність гнучкості та швидкості обслуговування, що є характерною рисою для більшої частини традиційних поштових послуг. Ця проблема стає особливо актуальною в умовах зростання популярності інтернет-магазинів та інтернет-магазинів розташованих у соціальних мережах, котрі загалом спеціалізуються на маленьких пакунках, котрі треба доставити максимально швидко та ефективно, так як клієнт очікує швидкої та ефективної доставки своїх замовлень.

Додатковими факторами, які обмежують ефективність традиційних поштових послуг, є незручність поштових відділень та їх обмежені години роботи, що може призвести до незадоволення клієнтів.

Наша програмна система має на меті вирішити ці проблеми, пропонуючи альтернативу у вигляді моделі «водія-посередника». Основні завдання, які вирішує наш додаток, включають:

- забезпечення гнучкості та швидкості доставки: Наша система дозволяє клієнтам замовляти доставку в будь-який зручний для них час і отримувати її в найкоротші терміни завдяки швидкому підбору доступних «водіїв-посередників» у їхньому регіоні;
- зручність і доступність: Наша система пропонує зручний та інтуїтивно зрозумілий інтерфейс для замовлення та відстеження відправлень через

мобільний додаток, що дозволяє клієнтам уникати черг у поштових відділеннях та здійснювати доставку у зручний для них час;

- надійність і безпека: Наша система забезпечує контроль якості та безпеки доставки, включаючи відстеження вантажу в режимі реального часу та можливість зв'язку з «водієм-посередником». Це гарантує клієнтам, що їхній вантаж знаходиться в безпеці;
- персоналізований підхід та додаткові послуги: Наша система дозволяє клієнтам користуватися різноманітними додатковими послугами, такими як страхування вантажів, що робить процес доставки більш персоналізованим та зручним.

Тому у висновку постановки завдання існування нашої програмної системи, можемо виділити наступні пункти:

- надання ефективного та зручного сервісу доставки вантажів;
- відповідність до сучасних потреб клієнтів;
- гарантована якість доставки та низька собівартість у порівнянні з конкурентами;
- гарантована своєчасна координація між клієнтами та водієм, що дозволяє більш ергономічною корегувати плани клієнта на поточний період часу;
- гнучкість та ефективність у використанні, що робить з програмної системи серйозного конкуренту не тільки для схожих за галуззю та напрямком конкурентів, а й для поштових відділень.

Враховуючи усі ці переваги, представники будь якого віку будуть задоволені під час користування цим продуктом, із-за його низької собівартості на доставку вантажів, високої гнучкості та зручного й інтуїтивно зрозумілого інтерфейсу.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Керування вантажними перевезеннями - це веб-застосування, розроблене з метою оптимізації та контролю за вантажними перевезеннями. Система надає зручні інструменти для моніторингу, аналізу та управління процесом перевезення вантажів.

Метою проекту є створення веб-застосування для керування вантажними перевезеннями, що забезпечує контроль якості перевезень, вирішення конфліктних ситуацій та аналіз статистики. Система має спростити процес управління вантажними перевезеннями та забезпечити користувачів актуальною інформацією про їх статус.

Для досягнення поставлених цілей були створені наступні функціональні вимоги:

- вхід до власного кабінету: Система повинна надавати можливість входу користувачам з різними ролями. Це включає адміністраторів, головних помічників та помічників. Кожна роль матиме свої права та обмеження, що забезпечить безпеку та ефективність роботи системи;
- перевірка скарг на перевезення: Система повинна надавати можливість переглядати скарги на перевезення користувачу. Це допоможе оперативно виявляти та вирішувати проблеми, що виникають під час перевезення вантажів. Користувачі зможуть переглядати деталі скарг, аналізувати їх та приймати відповідні рішення;
- відповідь на скарги: Система повинна надавати можливість спілкуватися по скарзі з заявником у реальному часі для вирішення проблемних питань. Це забезпечить оперативний зворотний зв'язок та допоможе швидше вирішувати конфліктні ситуації. Комунікація буде здійснюватися через вбудований чат або електронну пошту;
- редагування та видалення скарг на перевезення: Важливим аспектом функціоналу веб-застосування є можливість редагування та видалення скарг. Це дозволяє користувачам коригувати інформацію у разі змін чи

помилок, а також видаляти нерелевантні або вже вирішені скарги. Така гнучкість у роботі зі скаргами забезпечує актуальність та точність даних у системі, що, в свою чергу, сприяє більш ефективному управлінню та прийняттю рішень;

- можливість внесення корективів у перевезення: Система повинна надавати можливість редагувати налаштування перевезення. Це включає зміну маршруту, графіку перевезення, контактної інформації та інших параметрів. Така гнучкість дозволить швидко адаптуватися до змін та забезпечити оптимальні умови перевезення;
- аналіз статистики: Система повинна надавати користувачу актуальну інформацію стосовно кількості перевезень, кількості різних скарг та замовлень по тим, чи іншим метрикам. Це допоможе аналізувати ефективність перевезень, виявляти проблемні зони та приймати обґрунтовані рішення для покращення логістичних процесів.

Окрім функціональних вимог, були визначені наступні нефункціональні вимоги:

- безпека: Система повинна забезпечити цілісність та доступність даних, повну конфіденційність особистої інформації. Це включає захист даних від несанкціонованого доступу, використання сучасних методів шифрування та регулярні перевірки на вразливості. Безпека є ключовим аспектом, оскільки система працює з конфіденційною інформацією про перевезення;
- швидкодія: Веб-застосування повинно працювати швидко та коректно відображатися в усіх браузерах. Це включає оптимізацію коду, використання швидких серверів та зменшення часу завантаження сторінок. Система повинна витримувати великі обсяги даних та залишатися швидкодіючою при великій кількості запитів від користувачів;

– зручність інтерфейсу: Інтерфейс користувача повинен бути зручним у використанні, інтуїтивно зрозумілим та адаптивним до всіх розмірів девайсів. Це забезпечить комфортне користування системою на різних пристроях, включаючи настільні комп'ютери, планшети та смартфони. Адаптивний дизайн та зручна навігація сприятимуть ефективному використанню системи.

Цей веб-застосунок повинен підтримувати:

– Chromium 50.0 та вище;
– Android с версією системи від 5.0 и вище.

Впровадження веб-застосування для керування вантажними перевезеннями дозволить суттєво покращити ефективність та якість логістичних операцій. Зокрема, система сприятиме зниженню витрат завдяки оптимізації маршрутів, що зменшить витрати на пальне та інші ресурси. Крім того, автоматизований моніторинг і аналіз перевезень дозволить мінімізувати час простою та підвищити продуктивність використання ресурсів.

Підвищення рівня обслуговування клієнтів стане можливим завдяки інтеграції зручних інструментів для моніторингу та комунікації. Оперативне вирішення скарг та швидка відповідь на запити клієнтів дозволять підвищити лояльність та довіру до компанії.

Важливим аспектом системи є можливість редагування та видалення скарг, що забезпечує актуальність і точність даних у системі. Користувачі зможуть коригувати інформацію у разі змін або помилок, а також видаляти нерелевантні або вже вирішені скарги. Це сприяє більш ефективному управлінню процесами та прийняттю обґрунтованих рішень.

На завершення, розроблені функціональні та нефункціональні вимоги до веб-застосування гарантують його ефективність, безпеку та зручність у використанні. Система стане потужним інструментом для компаній, що займаються вантажоперевезеннями, надаючи їм можливість ефективно контролювати процеси, оперативно вирішувати проблеми та забезпечувати високий рівень обслуговування клієнтів.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ

3.1 UML проектування ПЗ

Впродовж планування програмної системи були визначені три основні типи користувачів: адміністратор (Admin), головний помічник (HeadSupport) та помічник (Support). Кожна з цих ролей має різний рівень доступу та функціональні можливості в системі, що дозволяє ефективно розподіляти обов'язки та забезпечувати належний рівень безпеки. Адміністратор має найбільш широкий набір прав, включаючи можливість повного управління користувачами, налаштуваннями системи та доступом до всіх даних. Головний помічник займається більш спеціалізованими завданнями, зосередженими на управлінні роботою помічників та взаємодії з критичними даними. Помічник, у свою чергу, виконує повсякденні операції, обмежені в межах своєї ролі, такі як обробка заявок або надання технічної підтримки.

Залежно від ролі користувача, система надає їм відповідні дозволи на маніпулювання інформацією через зручний інтерфейс, підключений до API. Адміністратори можуть читати, редагувати або видаляти будь-які дані, що є в системі, а також управляти доступом інших користувачів. Головні помічники мають обмежений доступ до редагування та видалення даних, але зберігають можливість читання важливої інформації та керування певними аспектами системи. Помічники, маючи найнижчий рівень доступу, можуть в основному читати дані та здійснювати базові операції, необхідні для їхньої роботи. Така структура дозволів забезпечує гнучкість та безпеку системи, дозволяючи кожному користувачеві виконувати свої завдання максимально ефективно.

За для більш чіткого розуміння можливостей кожного типу користувачів були побудовані Use Case діаграми. На рисунку 3.1 представлена діаграма, призначена для адміністратора системи. Ця діаграма візуалізує важливі метрики та дані, які допомагають адміністратору стежити за станом системи, аналізувати продуктивність і приймати обґрунтовані рішення.

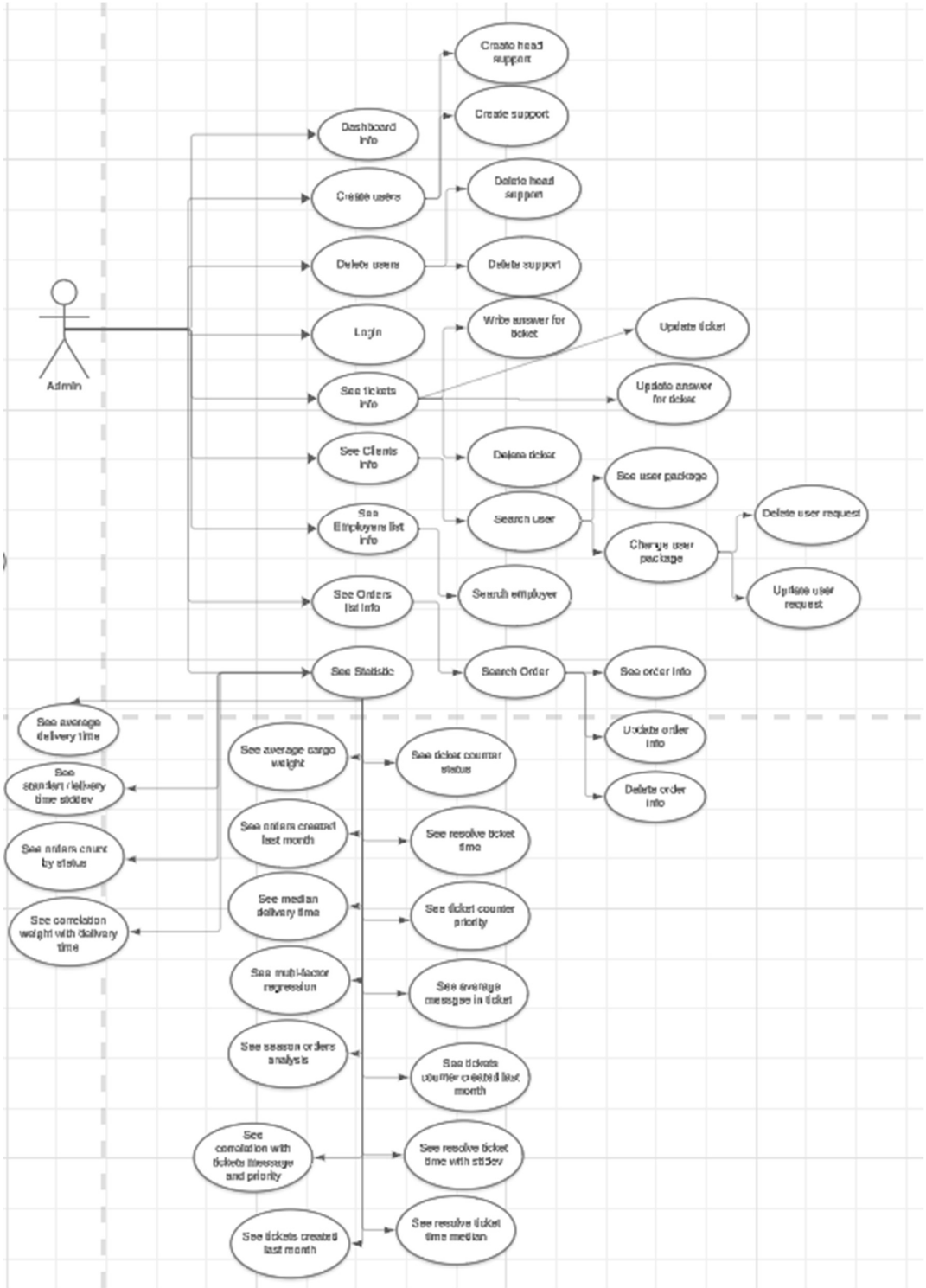


Рисунок 3.1 – Use Case діаграма для адміністратора

З діаграмми стає зрозуміло, що адміністратор може робити наступні дії:

– входити у систему за допомогою поштової скриньки та паролю;

– додавати head support;

- додавати support;
- видаляти head support;
- видаляти support;
- передивлятися загальну інформацію на сторінці Dashboard;
- передивлятися звернення до службі підтримки від користувачів;
- відповідати на звернення до службі підтримки від користувачів;
- виправляти звернення до службі підтримки від користувачів;
- видаляти звернення до службі підтримки від користувачів;
- передивлятися інформацію стосовно клієнтів;
- зробити пошук по клієнтам;
- передивлятися інформацію стосовно посилки клієнта;
- вносити зміни за проханням користувача, а саме видаляти;
- вносити зміни за проханням користувача, а саме змінювати налаштування посилки;
- передивлятися список робітників;
- передивлятися список замовлень;
- шукати замовлення;
- передивлятися детальну інформацію про замовлення;
- передивлятися статистику по середньому часу доставки;
- передивлятися стандартний час доставки з відхиленням;
- передивлятися кількість замовлень за статусами;
- передивлятися кореляцію між вагою та часом доставки;
- передивлятися середню вагу доставки;
- передивлятися кількість доставок за останній місяць;
- передивлятися медіану часу доставки;
- передивлятися аналітику за сезонами по доставках;
- передивлятися корреляцію між кількістю повідомлень та пріорітетом скарги;
- передивлятися по дням скарги створенні за останній місяць;

- передивлятися кількість скарг за статусами;
- передивлятися час за котрий скарга опрацьовується;
- передивлятися кількість скарг за пріорітетом;
- передивлятися середню кількість повідомлень у скаргах;
- передивлятися кількість скарг створених по днях за останній місяць;
- передивлятися середню кількість часу на вирішення тікета з урахуванням похибки;
- передивлятися медіану часу вирішення скарги;
- передивлятися мультифакторну регрессію доставки.

На рисунку 3.2 відображена діаграму для головного помічника

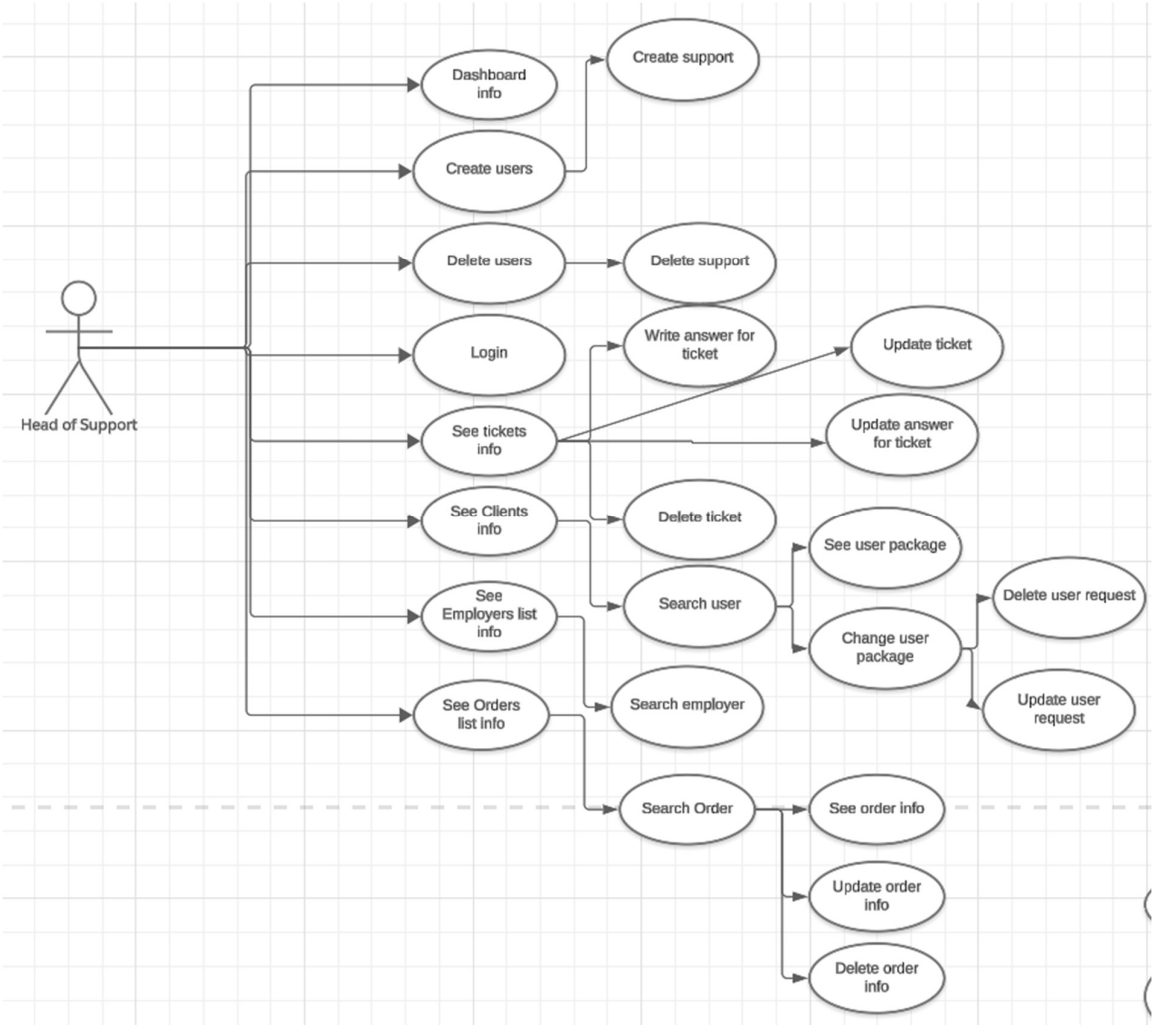


Рисунок 3.2 – Use Case діаграма для головного помічника

З діаграмми стає зрозуміло, що адміністратор може робити наступні дії:

- входити у систему за допомогою поштової скриньки та паролю;
- додавати support;
- видаляти support;
- передивлятися загальну інформацію на сторінці Dashboard;
- передивлятися звернення до службі підтримки від користувачів;
- відповідати на звернення до службі підтримки від користувачів;
- виправляти звернення до службі підтримки від користувачів;
- видаляти звернення до службі підтримки від користувачів;
- передивлятися інформацію стосовно клієнтів;
- зробити пошук по клієнтам;
- передивлятися інформацію стосовно посилки клієнта;
- вносити зміни за проханням користувача, а саме видаляти;
- вносити зміни за проханням користувача, а саме змінювати налаштування посилки;
- передивлятися список робітників;
- передивлятися список замовлень;
- шукати замовлення;
- передивлятися детальну інформацію про замовлення.

На рисунку 3.3 відображена діаграму для помічника

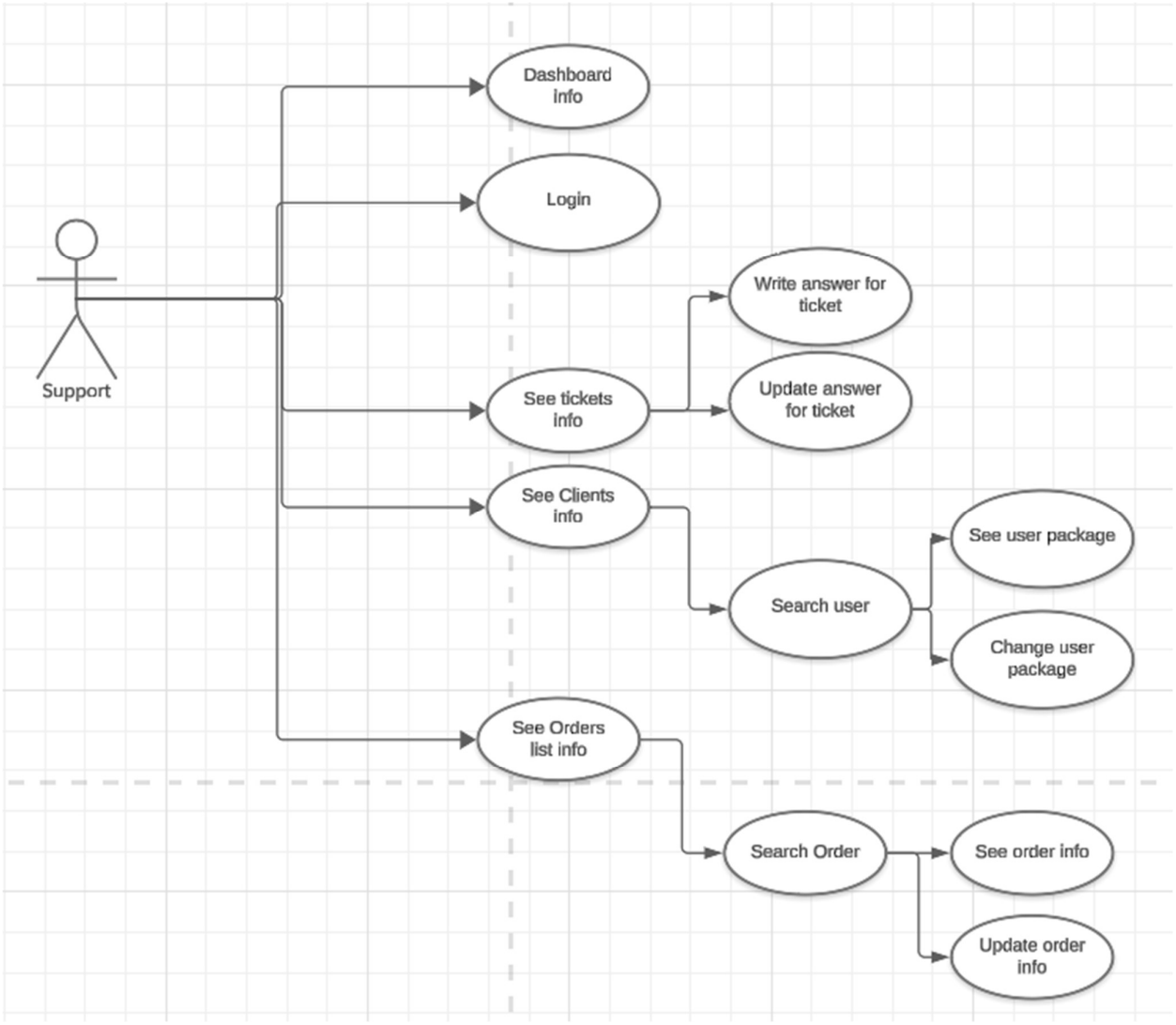


Рисунок 3.3 – Use Case діаграма для помічника

З діаграмми стає зрозуміло, що адміністратор може робити наступні дії:

– входити у систему за допомогою поштової скриньки та паролю;
– передивлятися загальну інформацію на сторінці Dashboard;
– передивлятися звернення до службі підтримки від користувачів;
– відповідати на звернення до службі підтримки від користувачів;
– виправляти звернення до службі підтримки від користувачів;
– передивлятися інформацію стосовно клієнтів;
– зробити пошук по клієнтам;
– передивлятися інформацію стосовно посилки клієнта;

- вносити зміни за проханням користувача, а саме змінювати налаштування посилки;
- передивлятися список замовлень;
- шукати замовлення;
- передивлятися детальну інформацію про замовлення.

З урахуванням представлених діаграм, можливо зрозуміти, що навантаження на працівників є цілком збалансованим та ергономічним згідно й їх роллю у системі.

Як приклад розглянемо адміністратора та звичайного помічника, в цьому випадку ми бачимо, що відповідальності на адміністраторі лежить більше, тому в його доступі є повний склад усього функціоналу програми,що тим чи іншим засобом має можливість вплинути на функціонування системи, в той же час із-за недосвідченості, помічнику не дозволяється робити якихось радикальних дій і його зона обов’язків в основному знаходиться у вирішенні складних та надзвичайних ситуацій котрі мали можливість трапитися під час виконання замовлення.

3.2 Проектування архітектури ПЗ

У якості front-end частини буде застосовуватися React в поєднанні з TypeScript(TS) та збірником Vite.

React[1] - це JavaScript-бібліотека для розробки користувацьких інтерфейсів, розроблена компанією Facebook. Вона дозволяє створювати ефективні та динамічні веб-додатки, використовуючи компонентну структуру, віртуальний DOM, односторонній потік даних та JSX - розширення синтаксису JavaScript для опису інтерфейсу. Завдяки віртуальному DOM, React може ефективно оновлювати веб-сторінки, підвищуючи продуктивність додатків. Крім того, React надає життєвий цикл компонента, який дозволяє виконувати дії на різних етапах життєвого циклу компонента. Його можна використовувати як у веб-браузері, так і на сервері, що робить його популярним інструментом для розробки універсальних веб-додатків.

Зазвичай використовується вибір між JavaScript та TypeScript[2]. Так як в нашому випадку ми будуємо надійне та здібне до маштабування у подальшому програмну системи ми обрали TypeScript. Основна відмінність TypeScript від JavaScript в тому, що TypeScript це типізовано його версія, котра за допомогою загальних та особистих типів даних описати йо ми очікуємо у відповідь, або в якості аргументів функції. Ця версія є більш сприйнятливою в нашому випадку.

Також ми використовуємо Vite[3]. Vite - це інструмент для розробки веб-додатків, який відрізняється швидкою швидкістю розробки та завантаження завдяки використанню ES-модулів і миттєвому перезавантаженню сторінок під час розробки. Він підтримує TypeScript і надає розширену систему плагінів для налаштування робочого процесу. Vite сприяє швидкому створенню сучасних веб-додатків за допомогою сучасних технологій і забезпечує зручний та ефективний робочий процес для розробників.

У якості стерилізації була обрана технологія MaterialUI v.5[4], версія котрої є актуальною на даний момент та більш збалансованою між кількістю можливостей та потрібних речей і препроцессор SASS[5] для написання більш зручного та масштабованого файлу стилів у форматі CSS.

MaterialUI v5 - це бібліотека компонентів користувацького інтерфейсу для React, заснована на дизайн-системі Material Design від Google. Версія 5 пропонує багато нових функцій і поліпшень, включно з гнучкою системою тем, новими компонентами та можливостями анімації, оновленим API і підтримкою більш сучасних підходів до розроблення веб-додатків. MaterialUI дає змогу розробникам створювати стильні та сучасні користувацькі інтерфейси з мінімальними зусиллями завдяки готовим компонентам і простому налаштуванню.

SASS (Syntactically Awesome Style Sheets) - это метаязык расширения для CSS[6], который предоставляет разработчикам большую гибкость и функциональность при написании стилей для веб-приложений. SASS позволяет использовать переменные, вложенные селекторы, миксины, условия и другие конструкции, которые значительно упрощают процесс создания и управления стилями. Он компилируется в обычный CSS, что обеспечивает совместимость со

всеми современными браузерами. SASS помогает уменьшить количество кода CSS и облегчает его обслуживание, делая разработку стильного и организованного дизайна более эффективной.

Для взаємодії з сервером, була обрана бібліотека Axios[7], котра надає можливості швидко та ефективно спілкуватися клієнтській частині з серверною частиною.

В якості менеджеру стану був обран сучасний продукту Zustand[8], котрий допомагає зберігати стан і на даний момент є дуже впевненою та гарною альтернативою Redux із-за своєї маленької кількості коду для налаштування та використання.

В нашому проекті ми використовуємо архітектурний підхід feature slice, що дозволяє розділити функціональність системи на дрібні шматочки, що полегшує розробку та тестування коду.

Цей підхід був обраний із-за того, що React сам по собі не має якоїсь чіткої структури, або чітких архітектурних уподобань і загалом усе зосереджено на розбитті великих компонентів на більш дрібні. Великі компоненти такія умовний header сторінки, або навігаційна панель це ті самі великі шматки з бізнес логікою, котрі використовують маленькі складові, такі як логотип, навігаційний лист та інші. В малих шматочках нема бізнес логіки, тому їх можливо використати у декількох місцях одночасно, що дає нашій архітектурі більш гнучкий та ефективний підхід до реалізації.

На рисунку 3.4 зображено діаграму компонентів API-частини програмної системи.

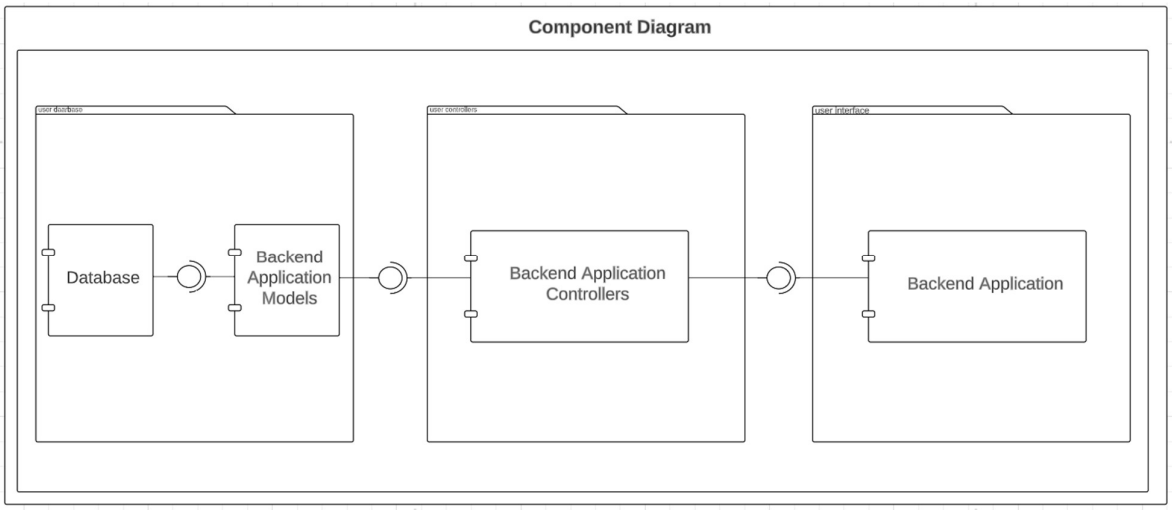


Рисунок 3.4 – Діаграма компонентів API-частини системи

На рисунку 3.5 зображено Use Case діаграму (діаграми прецедентів) обробки запиту у API.

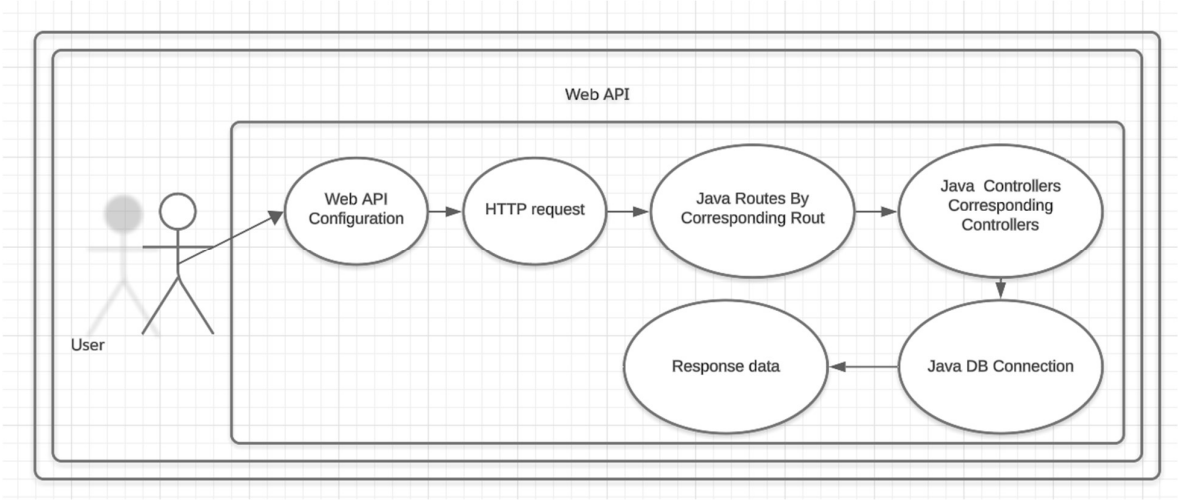


Рисунок 3.5 – Діаграма прецедентів

З діаграм ми бачимо, що взаємодія між користувачем та API відбувається через веб-браузер, у свою чергу API запит обробляється Java Routes та надходить до потрібного контроллеру на стороні серверу, після чого один з Java Controller виконує зміни даних, попередньо приєднавшись до бази даних.

3.3 Приклади найцікавіших алгоритмів та методів

Найцікавіший метод, котрий був використаний у front-end частині, це метод «Facade Pattern».

Фасадний патерн (Facade Pattern) є структурним патерном, який надає спрощений інтерфейс до більш складної системи або набору класів. У контексті фронтенд-розробки з використанням бібліотеки стану Zustand і бібліотеки для HTTP-запитів Axios, фасадний метод допомагає абстрагувати складні операції, що спрощує роботу з API та управління станом.

Розглянемо приклад реалізації цього паттерну на прикладі AuthService.

Спочатку був створений класс, котрий вміщує в собі усі потрібні методи для авторизації. На рисунку 3.6 зображено приклад AuthService.

```
export default class AuthService {
  static async login(
    userData: SignInFormInputs
  ): Promise<AxiosResponse<TokenType>> {
    return $api
      .post<TokenType>("/auth/authenticate", userData)
      .then((res): TokenType => res.data);
  }

  static logout(): void {
    clearLocalStorageUserData();
  }
}
```

Рисунок 3.6 – Приклад AuthService

Після створення AuthService, додається менеджер стану за допомогою Zustand. На рисунку 3.7 зображено приклад AuthStore.

```
const useAuthStore = create<useAuthStoreType>((set) => ({
  isAuth: !!localStorage.getItem("token"),
  userInfo: null,

  setCurrentUser: async () => {…
  },
  clearState: () => {…
  },
  login: async (userData: SignInFormInputs) => {…
  },
  logout: () => {…
  },
}));

export default useAuthStore;
```

Рисунок 3.7 – Приклад AuthStore

Тепер, після створення «фасаду », ми спокійно можемо взаємодіяти з АПІ серверу не переймаючись, що щось буде зламане, або змінений стан буде зламаний діями розробника. На рисунку 3.8 зображено приклад використання функції login, для авторизації користувача.

```
const login = useAuthStore((state: useAuthStoreType) => state.login);
```

Рисунок 3.8 – Приклад фунції login з AuthStore

Загалом, використання фасадного патерну у поєднанні з Zustand та Axios дозволяє створити більш структурований, модульний та зручний у підтримці код. Такий підхід сприяє підвищенню якості програмного забезпечення та покращенню продуктивності команди розробників, дозволяючи зосередитися на розробці нових функцій і покращення користувацького досвіду.

3.4 Створення UI / UX або іншого дизайну системи

В цьому програмному застосунку одразу і як архітектуру і як дизайн систему був обран підхід «feature-slice design» [9], котрий несе у собі ідею розбиття складних компонентів на більш дрібні. У контексті цього підходу, дизайн системи буде побудований навколо невеликих фрагментів або "фіч", кожен з яких відповідає конкретному функціоналу або елементу користувальницького інтерфейсу.

Головні принципи такого підходу включають модульність, простоту, консистентність, адаптивність та тестовість. Це дозволяє більш ефективно організувати та керувати різноманітністю компонентів та функціоналу системи, забезпечуючи зручний і продуктивний користувацький досвід на різних пристроях та екранах. Кожен "фрагмент" дизайну може бути розроблений, протестований і оптимізований окремо, що сприяє швидкому розвитку та покращенню продукту в цілому.

Так як ми використовуємо для стилізації загалом MaterialUI v.5 , то треба розглянути, як він може бути корисним в цій програмній системі. Взаємодія MaterialUI v.5 з архітектурою feature-slice design відбувається на рівні розробки компонентів користувацького інтерфейсу. MaterialUI надає багатий набір готових компонентів, які відповідають принципам дизайну Material Design[10] від Google. З використанням цих компонентів розробники можуть швидко створювати стильні та сучасні інтерфейси з мінімальними зусиллями.

У контексті feature-slice design, MaterialUI компоненти можуть бути організовані відповідно до функціональної частини системи, яку вони відображають. Кожен «фрагмент» або «фіча» може мати свій набір компонентів, які відповідають конкретним вимогам і можливостям цієї частини системи.

MaterialUI дає змогу використовувати гнучкі теми та стилізацію, що дає змогу адаптувати вигляд компонентів до конкретних потреб проекту. Це дає змогу забезпечити консистентність і візуальну гармонію всього інтерфейсу, незалежно від того, як багато «фіч» включено в систему.

Крім того, MaterialUI версії 5 надає розробникам нові потужні функції, які значно розширюють можливості для створення сучасних та динамічних користувацьких інтерфейсів. Однією з ключових нововведень є підтримка анімації, що дозволяє додавати плавні переходи та візуальні ефекти до компонентів, роблячи взаємодію з додатком більш приємною та інтуїтивною. Також оновлена система тем забезпечує легке налаштування зовнішнього вигляду додатка, дозволяючи швидко змінювати кольорову гаму, шрифти та інші стилістичні елементи для створення унікального дизайну. Підтримка нових компонентів, таких як вдосконалені форми, таблиці та інші елементи інтерфейсу, дає змогу використовувати готові рішення, що значно скорочує час на розробку та тестування.

Покращення, внесені в MaterialUI v.5, також сприяють більш ефективній взаємодії з архітектурою feature-slice design. Цей підхід до розробки програмного забезпечення базується на принципах модульності та простоти, дозволяючи розробникам розділяти код на окремі, незалежні функціональні блоки. MaterialUI, з його новими можливостями, інтегрується з цією архітектурою, надаючи засоби для створення добре структурованих та легко підтримуваних проектів. Розробники можуть використовувати переваги нових компонентів та тем, щоб швидко та ефективно впроваджувати функціональність, дотримуючись при цьому кращих практик проектування інтерфейсу користувача. Це сприяє створенню масштабованих додатків, які легко адаптуються до змінних вимог та забезпечують високу якість користувацького досвіду.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Для здійснення запитів до АПІ серверу в додатку було вирішено використовувати фасадний метод. Цей підхід дозволяє абстрагувати логіку авторизації від інших частин додатку та надає зручний інтерфейс для взаємодії з процесом авторизації. Для зберігання токенів авторизації було обрано використання localStorage, оскільки це простий та зручний спосіб зберігання даних на клієнтському боці і дозволяє дуже легко замінювати токен, котрий вже зіпсувався. На рисунку 4.1 показане використання одна зі складових паттерну «фасад», а саме контроллер авторизації.

```
export default class AuthService {
  static async login(
    userData: SignInFormInputs
  ): Promise<AxiosResponse<TokenType>> {
    return $api
      .post<TokenType>("/auth/authenticate", userData)
      .then((res): TokenType => res.data);
  }

  static logout(): void {
    clearLocalStorageUserData();
  }
}
```

Рисунок 4.1 – Використання AuthService

Після створення та налаштування сервісу авторизації, були впроваджені функції, котрі несуть в собі функціонал зміни стану localStorage. На рисунку 4.2 зовнішний вигляд цих функцій.

```
export const setLocalStorageUserData = (userData: TokenType) => {
  localStorage.setItem("token", userData.accessToken);
  localStorage.setItem("r_token", userData.refreshToken);
};
export const clearLocalStorageUserData = () => {
  localStorage.clear();
};
```

Рисунок 4.2 – Функцій для контролю стану localStorage

Після впровадження усіх необхідних частин коду, треба перейти на вікно авторизації користувача. На рисунку 4.3 зображено вікто авторизації користувача.

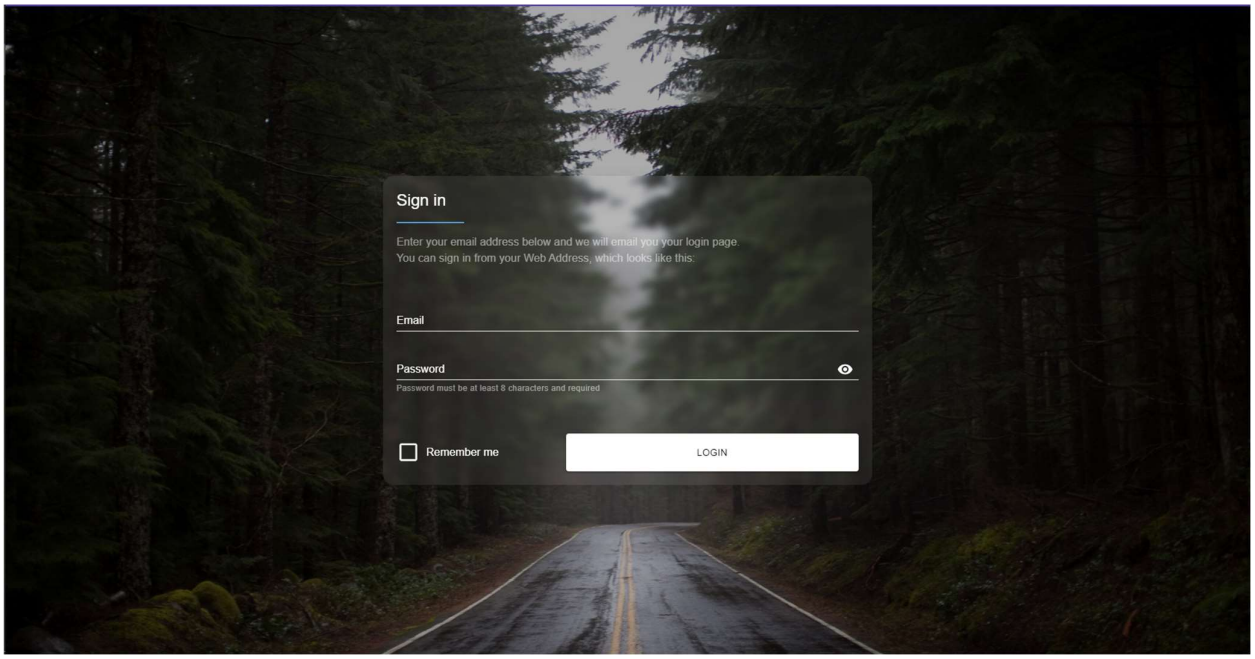


Рисунок 4.3 – Вікно авторизації

На рисунку 4.4 ми бачимо результат виконання запиту авторизації з використанням паттерну «фасад»

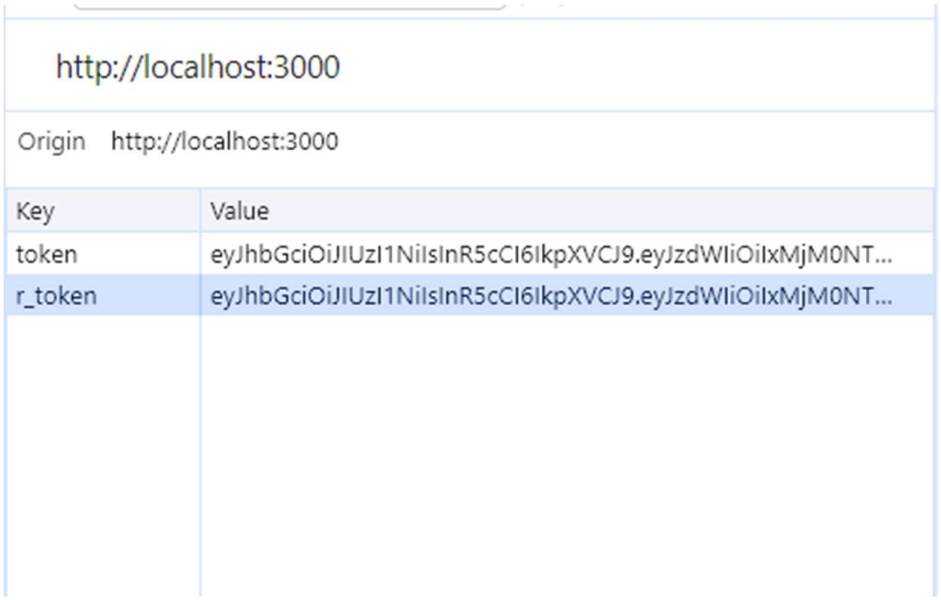


Рисунок 4.4 – Результат виконання авторизації

Для забезпечення ефективного управління станом додатку було прийнято рішення використовувати бібліотеку Zustand. Ця бібліотека надає простий та потужний спосіб для створення та управління глобальним станом додатку. Використання Zustand дозволяє зберігати стан додатку в одному централізованому місці, спрощує його модифікацію та підтримку, а також підвищує продуктивність розробки.

На рисунку 4.5 зображено приклад використання Zustand.

```
const useAuthStore = create<useAuthStoreType>((set) => ({
  isAuth: !!localStorage.getItem("token"),
  userInfo: null,

  setCurrentUser: async () => {…
  },
  clearState: () => {…
  },
  login: async (userData: SignInFormInputs) => {…
  },
  logout: () => {…
  },
}));

export default useAuthStore;
```

Рисунок 4.5 – Використання Zustand

Для здійснення взаємодії з сервером та виконання HTTP-запитів було прийнято рішення використовувати бібліотеку Axios. Axios є потужною та популярною бібліотекою, яка надає зручний інтерфейс для виконання різноманітних типів запитів до сервера. Використання Axios дозволяє забезпечити надійну та ефективну взаємодію з сервером, обробку помилок та забезпечення безпеки даних. На рисунку 4.6 зображено приклад використання Axios з базовими налаштуваннями, такими як interceptor. Цей interceptor запроваджує чіпляння до кожного запиту токену авторизації, що значно полегшує розробку.

```
const $api = axios.create({
  withCredentials: true,
  baseURL: import.meta.env.VITE_API_BASE_URL,
});

$api.interceptors.request.use((config) => {
  config.headers.Authorization = `Bearer ${localStorage.getItem("token")}`;
  return config;
});
```

Рисунок 4.6 – Використання axios з базовими налаштуваннями

Для ефективного управління доступом користувачів до різних частин додатку було реалізовано систему ролей. Ролі визначають набір дозволених шляхів для кожного типу користувача. Даний підхід забезпечує безпеку та гнучкість в керуванні правами доступу до функціональності додатку.

У компоненті AuthorizedLayout реалізовано перевірку доступу користувача до певних шляхів на основі його ролі. Під час завантаження сторінки, перевіряється чи має користувач доступ до поточного шляху. Якщо користувач не авторизований або не має доступу, він автоматично перенаправляється на сторінку входу

У функції hasAccess визначено правила доступу для кожної ролі. Вона приймає поточний шляхів, набір дозволених шляхів для кожної ролі та роль користувача. Функція перевіряє, чи є поточний шлях серед дозволених для вказаної ролі користувача, з використанням регулярних виразів для порівняння.

Для кожної ролі (ADMIN, HEAD_OF_SUPPORT, SUPPORT) визначено свій набір дозволених шляхів, які включаються в структуру rolePermission. Кожний шлях представляє собою об'єкт з властивостями routeLink, routeTitle та routeIcon, що відповідають шляху, назві та іконці відповідно.

Цей підхід дозволяє гнучко керувати доступом користувачів до різних частин додатку, забезпечуючи безпеку та конфіденційність даних. На рисунку 4.7 зображено приклад використання системи контролю безпеки.

```
export const rolePermission = {
  ADMIN: {
    routes: [ ···
    ],
    permission: [ ···
    ],
  },
  HEAD_OF_SUPPORT: { ···
  },
  SUPPORT: { ···
  },
};
```

Рисунок 4.7 – Використання массиву ролей з базовими налаштуваннями шляхів та дозволів на певні дії

На рисунку 4.8 зображена реалізації функції, яка безпосередньо вираховує, які дозволи є у користувача

```
export const hasAccess = (
  pathname: string,
  permissionArr: any,
  userRole: string
) => {
  return permissionArr[userRole].routes.some((route: RoleRoutesType) => {
    const regex = new RegExp(`^${route.routeLink}(/|$)`);
    return regex.test(pathname);
  });
};
```

Рисунок 4.8 – Приклад функції hasAccess для перевірки доступу до тих чи інших шляхів

На рисунку 4.9 зображений повний вигляд системи role_base з використанням її, коли користувач перемикається на інші сторінки.

```
useEffect(() => {
  const checkAuth = async () => {
    let isAccess: boolean = false;
    const user = await setCurrentUser().catch((error) => null);

    if (user) {
      isAccess =
        hasAccess(pathname, rolePermission, user.role[0]) ||
        pathname === "/profile";
    }

    if (!isAuth || !isAccess) {
      clearLocalStorageUserData();
      clearState();

      navigate("/sign-in");
    }
  };

  checkAuth();
}, []);
```

Рисунок 4.9 – Повний вигляд системи role_base методу, котрий спрацьовує під час відвідуванням поточного користувача сторінки

У програмній системі для реалізації сторінок було прийнято використання бібліотеки MUI (Material-UI) для створення зручного та ефективного інтерфейсу взаємодії користувача зі сторінкою. Використання MUI дозволило швидко та адаптивно реалізувати різноманітні елементи та функціональність на сторінці. . На рисунку 4.10 зображено приклад реалізації одного з компонентів системи, такого як бокова панель користувача.

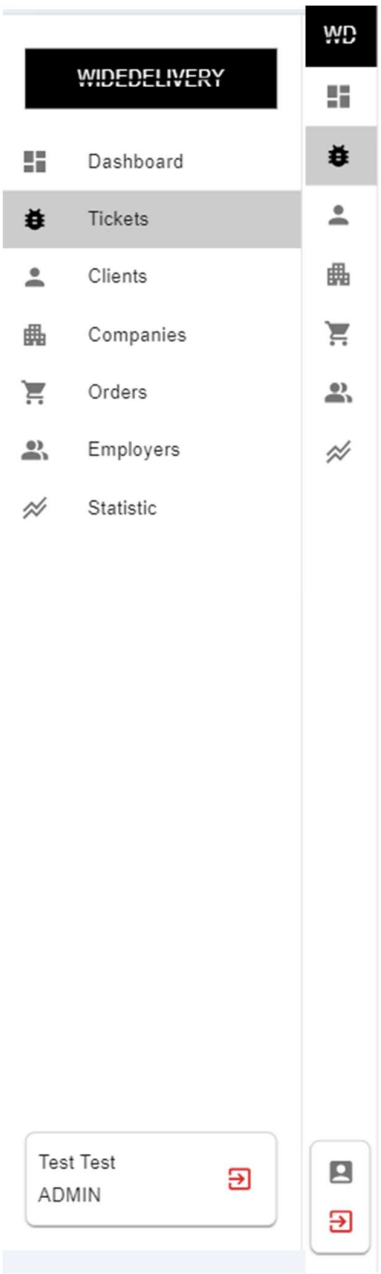


Рисунок 4.10 – Вигляд бокової панелі користувача на великих та малих екранах

На рисунку 4.11 зображен загальний зовнішній вигляд вікна скарг, де реалізован пошук по скаргам за декількома критеріями та картки самих скарг, при натискання на котрі користувача переносить на сторінку конкретної скарги.

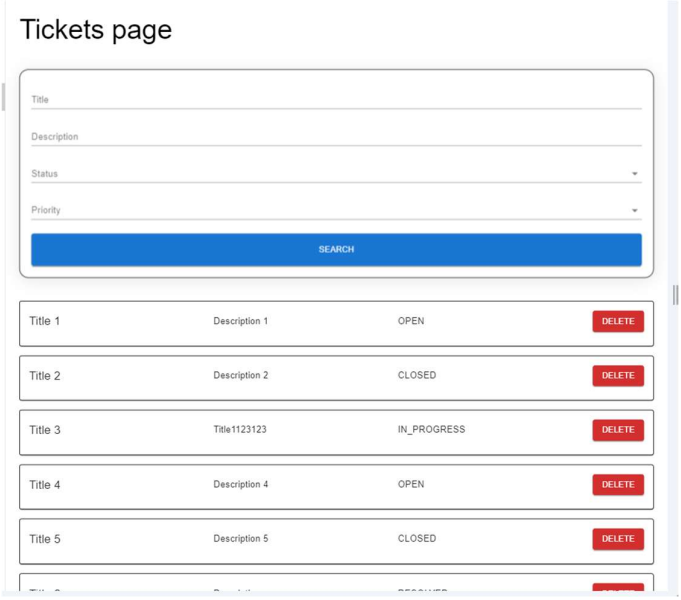


Рисунок 4.11 – Вигляд основного контенту сторінки на великих екранах

На рисунку 4.12 зображен загальний зовнішній вигляд вікна скарг, але на екранах більш меньшого розміру і з урахуванням можливості адаптації загальної частини сторінки до відповідного розширення пристрою.

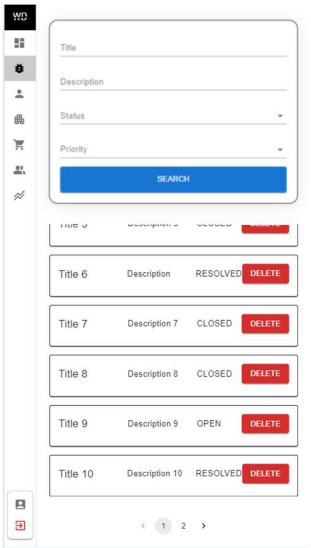


Рисунок 3.12 – Вигляд основного контенту сторінки на малих екранах

Загалом, прийняті програмні рішення демонструють ефективне використання сучасних інструментів та бібліотек для створення зручного, адаптивного та функціонального інтерфейсу користувача. Використання фасадного патерну в поєднанні з бібліотеками Axios та Zustand дозволяє абстрагувати складні операції, спрощуючи роботу з API та управління станом додатку. Додатково, застосування методів role-based доступу та layout компонентів забезпечує контроль за доступом до різних частин додатку, підвищуючи безпеку та зручність використання системи.

Використання бібліотеки MUI (Material-UI) у розробці інтерфейсу значно прискорює процес розробки, дозволяючи створювати стильні та функціональні компоненти з мінімальними зусиллями. Компоненти MUI забезпечують високу адаптивність інтерфейсу, що дозволяє користувачам комфортно взаємодіяти з системою на різних пристроях та екранах.

Завдяки обраним програмним рішенням вдалося створити надійну та зручну систему керування вантажними перевезеннями, яка відповідає всім функціональним та нефункціональним вимогам проекту. Система забезпечує ефективний контроль перевезень, зручний інтерфейс для користувачів та надійну безпеку даних.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування програмного забезпечення є ключовим етапом у забезпеченні якості продукту. У процесі розробки веб-застосунку для керування вантажними перевезеннями було проведено ретельне мануальне тестування для перевірки коректності роботи функціоналу, відповідності вимогам та забезпечення стабільності системи.

Перед початком тестування було підготовлено тестове середовище, яке включало розгортання застосунку на тестовому сервері, налаштування тестових облікових записів з різними ролями (адміністратор, головний помічник, помічник) та створення набору тестових даних, таких як користувачі, скарги, замовлення тощо.

Цей етап включав встановлення всіх необхідних залежностей, налаштування бази даних та конфігурацію серверного оточення. Було створено окреме середовище, яке відображало реальні умови роботи системи, щоб забезпечити максимально реалістичне тестування.

Для тестування різних рівнів доступу та ролей у системі були створені тестові облікові записи для адміністратора, головного помічника та помічника підтримки. Це дозволило перевірити коректність роботи функціоналу для кожного типу користувача.

Були створені фіктивні дані для користувачів, скарг та замовлень. Це включало введення реалістичних даних, таких як імена користувачів, деталі скарг та інформацію про замовлення. Це забезпечило можливість проведення тестів у умовах, наближених до реальних.

Мануальне тестування було розділено на кілька етапів: тестування автентифікації та авторизації, тестування функціоналу управління скаргами, тестування інтерфейсу користувача та адаптивності.

Етап 1. Тестування автентифікації та авторизації.

Вхід користувача. Перевірка можливості входу користувача з різними ролями. Було перевірено вхід користувача з роллю адміністратора, керівника

підтримки та співробітника підтримки. Це включало введення коректних та некоректних облікових даних, перевірку реакції системи на різні випадки.

Перевірка обмеження доступу. Було перевірено, чи система правильно обмежує доступ до сторінок, які потребують авторизації. Це включало спробу доступу до захищених сторінок без входу в систему, а також перевірку доступу для користувачів з різними ролями до сторінок, які мають обмеження доступу.

Перевірка коректності збереження токенів. Було перевірено, чи зберігається токен авторизації в localStorage після успішного входу, та чи автоматично виходить система при видаленні токена.

Етап 2: Тестування функціоналу управління скаргами.

Редагування існуючої скарги. Перевірка можливості редагування деталей скарги та збереження змін після редагування.

Видалення скарги. Перевірка можливості видалення скарги зі списку та відображення оновленого списку після видалення.

Зміна статусу квитка. Перевірка можливості зміни статусу скарги та коректного відображення статусу в загальному списку скарг.

Фільтрація за статусом. Перевірка можливості фільтрації скарг за назвою, статусом, описом та пріорітетом та коректності відображення відфільтрованих даних.

Етап 3: Тестування інтерфейсу користувача та адаптивності

Перевірка відповідності дизайну. Перевірка, що всі елементи інтерфейсу відповідають затвердженому дизайну та коректного відображення стилів та кольорів.

Адаптивність інтерфейсу. Перевірка коректного відображення інтерфейсу на різних пристроях (десктоп, планшет, мобільний) та коректної роботи елементів управління (кнопки, поля вводу) на різних розмірах екранів.

Перевірка окремих компонентів. Перевірка коректного відображення заголовка сторінок, роботи пошукового блоку та форми пошуку скарг, а також відображення списку скарг та роботи пагінації.

Після проведення мануального тестування було виявлено та виправлено низку дрібних помилок, що дозволило підвищити загальну стабільність та функціональність системи. Мануальне тестування підтвердило відповідність розробленого програмного забезпечення вимогам та забезпечило високу якість кінцевого продукту.

Для підвищення продуктивності та зменшення затримок в роботі веб-застосунку, важливо оптимізувати компоненти. Це дозволяє зменшити час завантаження сторінок, покращити взаємодію користувачів з інтерфейсом та знизити навантаження на сервер. Нижче наведено детальний опис прийнятих заходів для оптимізації компонентів в нашому застосунку.

React.memo був використаний для запобігання непотрібному повторному рендерингу компонентів. Це дозволяє компоненти повторно рендеритися лише тоді, коли їх пропси змінюються. Це особливо корисно для компонентів, які отримують пропси, що рідко змінюються. Наприклад, компоненти, які відображають інформацію про користувачів чи компанії, можуть бути обгорнуті у React.memo, щоб уникнути зайвого рендерингу при оновленні інших частин інтерфейсу.

Для зменшення часу завантаження головної сторінки, компоненти, які не використовуються на перших етапах завантаження, були розбиті та завантажуються динамічно. Для цього використовувалась функція React.lazy. Це дозволяє завантажувати компоненти лише тоді, коли вони дійсно потрібні. Наприклад, компоненти, які відображають деталі квитків чи замовлень, завантажуються лише при переході користувача на відповідні сторінки.

Для запобігання створенню нових екземплярів функцій при кожному рендері, використовувався хук useCallback. Це дозволяє зберігати функції між рендерами, що зменшує кількість обчислень та підвищує продуктивність. Аналогічно, для мемоізації значень використовувався хук useMemo, що дозволяє зберігати результати обчислень між рендерами, що також сприяє зменшенню навантаження на систему.

ВИСНОВКИ

Під час створення програмної системи «Керування вантажними перевезеннями. Front-end.» було створено програмний застосунок для управління вантажними перевезеннями. Перед початком роботи було проведено аналіз предметної області в сегменті доставки вантажів та вантажних перевезень, що дозволило визначити основні завдання системи.

В результаті аналізу були визначені основні та необхідні функції програмного комплексу, які були заплановані на етапі постановки завдань і безпосередньо пов'язані з предметною областю вантажних перевезень. Серед цих функцій можна виділити кілька ключових аспектів, які визначають ефективність і функціональність системи.

Одна з найважливіших функцій будь-якої системи – це забезпечення безпечного та надійного входу користувачів. Користувачі системи мають можливість входити в систему за допомогою своїх облікових записів. Для цього використовуються сучасні методи автентифікації та авторизації, які гарантують безпеку особистих даних та доступ до конфіденційної інформації тільки для авторизованих користувачів.

У сфері вантажних перевезень конфліктні ситуації можуть виникати досить часто. Це можуть бути питання, пов'язані з затримками, пошкодженням вантажу, невідповідністю термінів доставки та іншими проблемами. Програмний комплекс дозволяє вирішувати такі ситуації в режимі реального часу, що значно підвищує ефективність роботи та задоволеність клієнтів. Користувачі мають можливість оперативно комунікувати з клієнтом, обмінюватися інформацією та приймати спільні рішення для вирішення проблем.

Окрім оперативного управління перевезеннями, система також надає можливість аналізувати інформацію про минулі перевезення. Це дозволяє виявляти закономірності, аналізувати ефективність роботи, визначати проблемні зони та приймати обґрунтовані рішення для покращення процесу в майбутньому.

Аналітичні інструменти, вбудовані в систему, дозволяють створювати графіки та інші форми візуалізації даних, що значно спрощує процес аналізу.

Розробка програмної системи «Керування вантажними перевезеннями. Front-end.» дозволила створити ефективний інструмент для управління вантажними перевезеннями, який відповідає сучасним вимогам та викликам ринку. Впроваджені функції дозволяють забезпечити зручність, безпеку та ефективність роботи, що сприяє підвищенню рівня задоволеності клієнтів та оптимізації процесів перевезення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. React documentation — URL: https://react.dev/learn (дата звернення 11.05.2024).

2. TypeScript documentation — URL: https://www.typescriptlang.org/docs/ (дата звернення 11.05.2024).

3. Vite documentation — URL: https://vitejs.dev/guide/ (дата звернення 07.05.2024).

4. MaterialUI v.5 documentation — URL: https://mui.com/material-ui/getting-started/ (дата звернення 10.05.2024).

5. SASS documentation — URL: https://sass-lang.com/documentation/ (дата звернення 10.05.2024).

6. CSS documentation — URL: https://developer.mozilla.org/en-US/docs/Web/CSS/Reference (дата звернення 11.05.2024).

7. Axios documentation — URL: https://axios-http.com/docs/intro (дата звернення 12.05.2024).

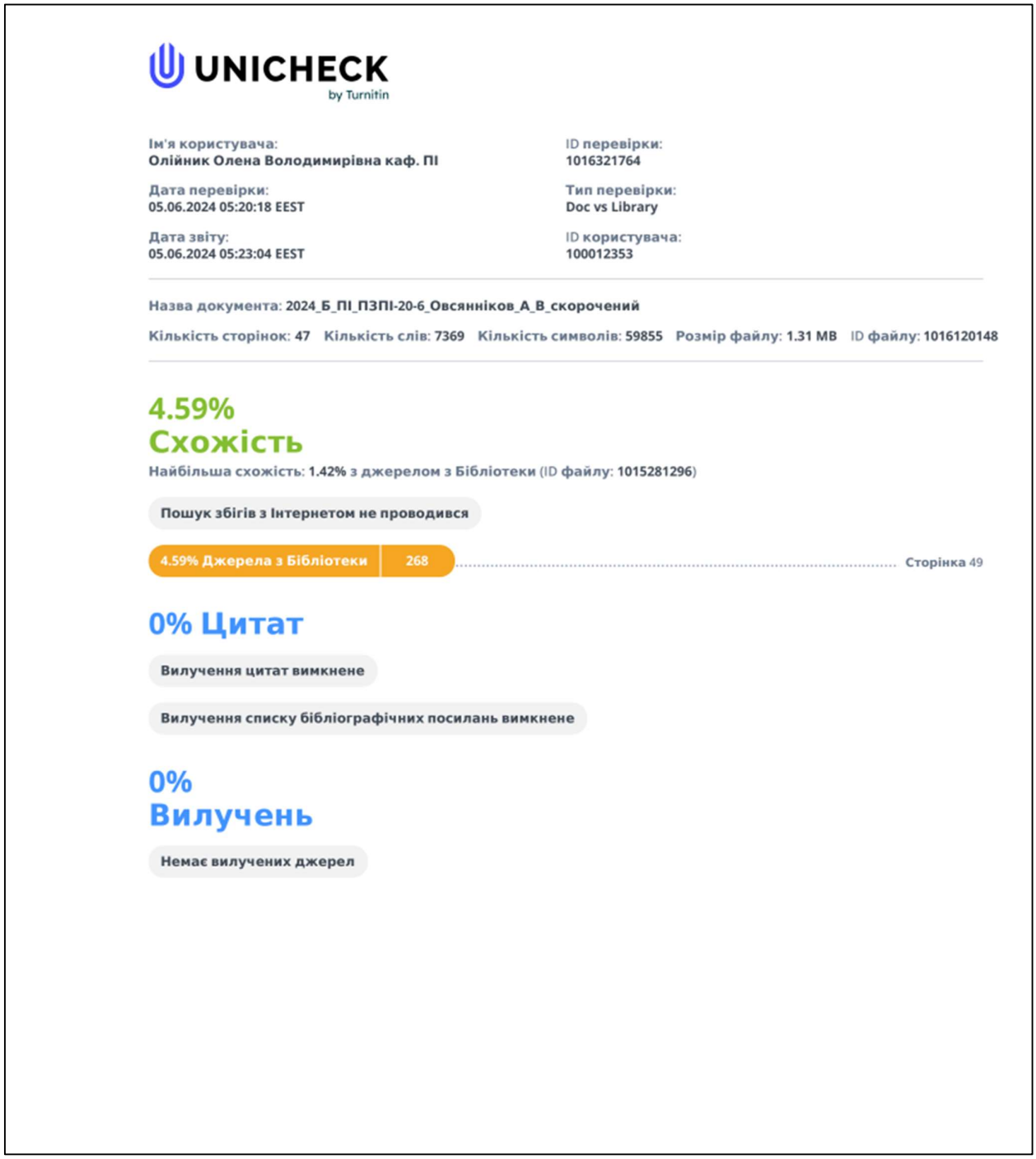
8. Zustand documentation — URL: https://docs.pmnd.rs/zustand/getting-started/introduction (дата звернення 12.05.2024).

9. Feature-slice design in React tutorial — URL: https://feature-sliced.design/docs/get-started/tutorial (дата звернення 11.05.2024).

10. Material Design documentation — URL: https://m3.material.io/get-started (дата звернення 10.05.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016321764

Дата перевірки:
05.06.2024 05:20:18 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
05.06.2024 05:23:04 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ-20-6_Овсянніков_А_В_скорочений

Кількість сторінок: 47 Кількість слів: 7369 Кількість символів: 59855 Розмір файлу: 1.31 MB ID файлу: 1016120148

4.59%
Схожість

Найбільша схожість: 1.42% з джерелом з Бібліотеки (ID файлу: 1015281296)

Пошук збігів з Інтернетом не проводився

4.59% Джерела з Бібліотеки 268 Сторінка 49

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

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

ДОДАТОК Б

Слайди презентації

Рисунок Б.1 – Слайд 1

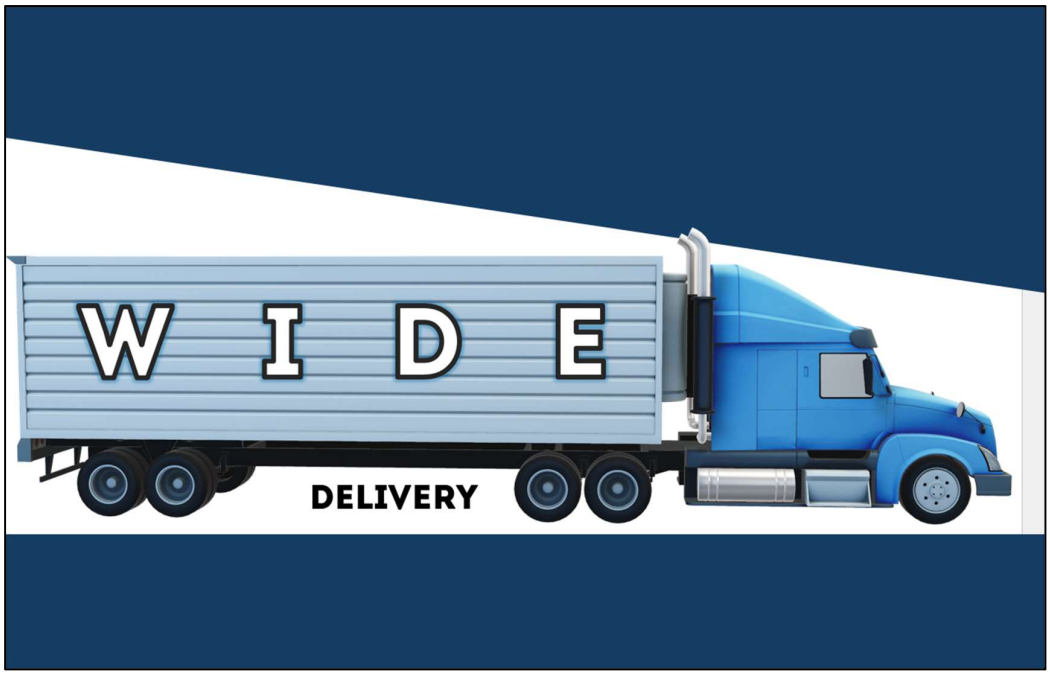


Рисунок Б.2 – Слайд 2

Рисунок Б.3 – Слайд 3

Рисунок Б.4 – Слайд 4

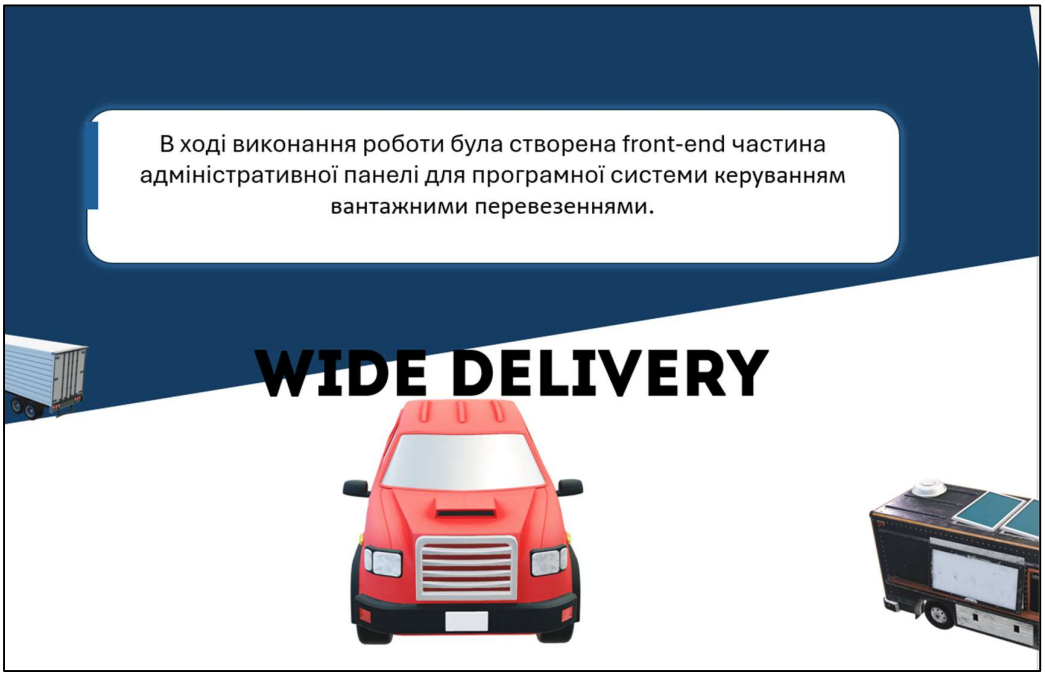


Рисунок Б.5 – Слайд 5

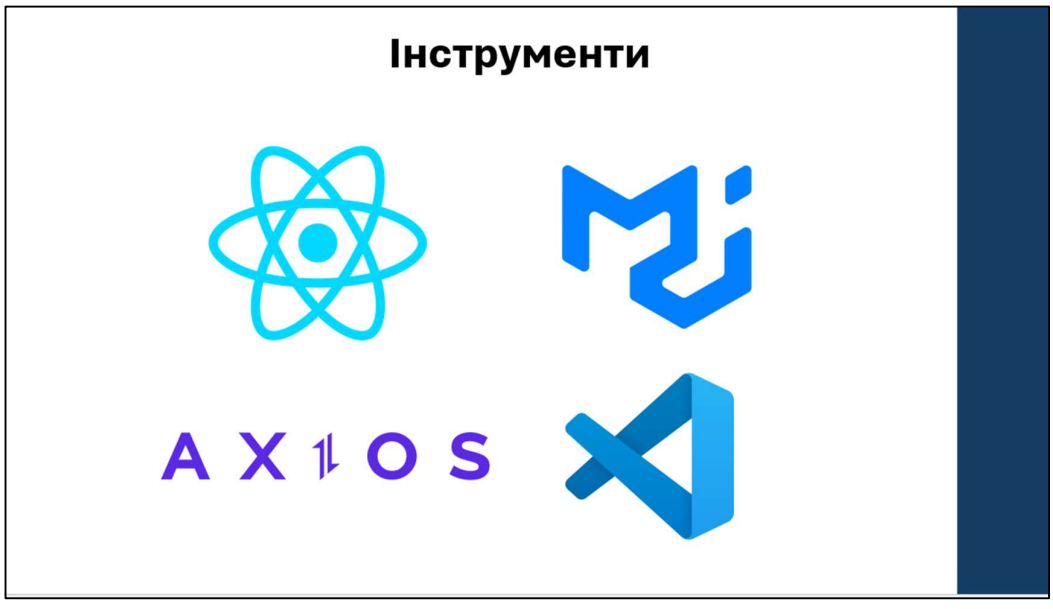


Рисунок Б.6 – Слайд 6

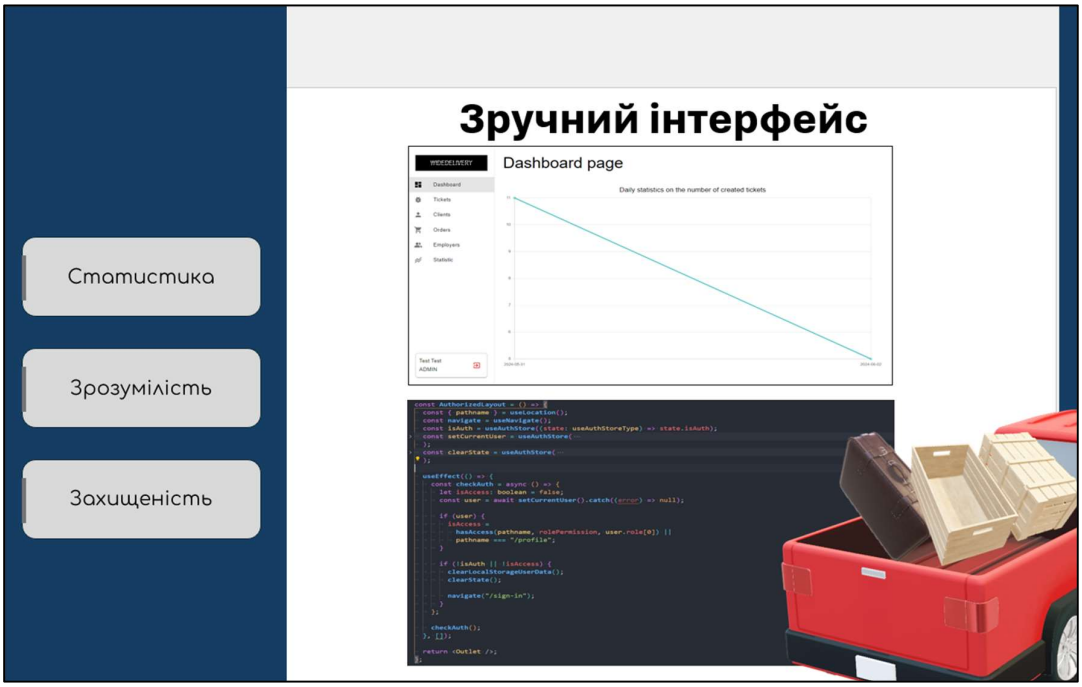


Рисунок Б.7 – Слайд 7

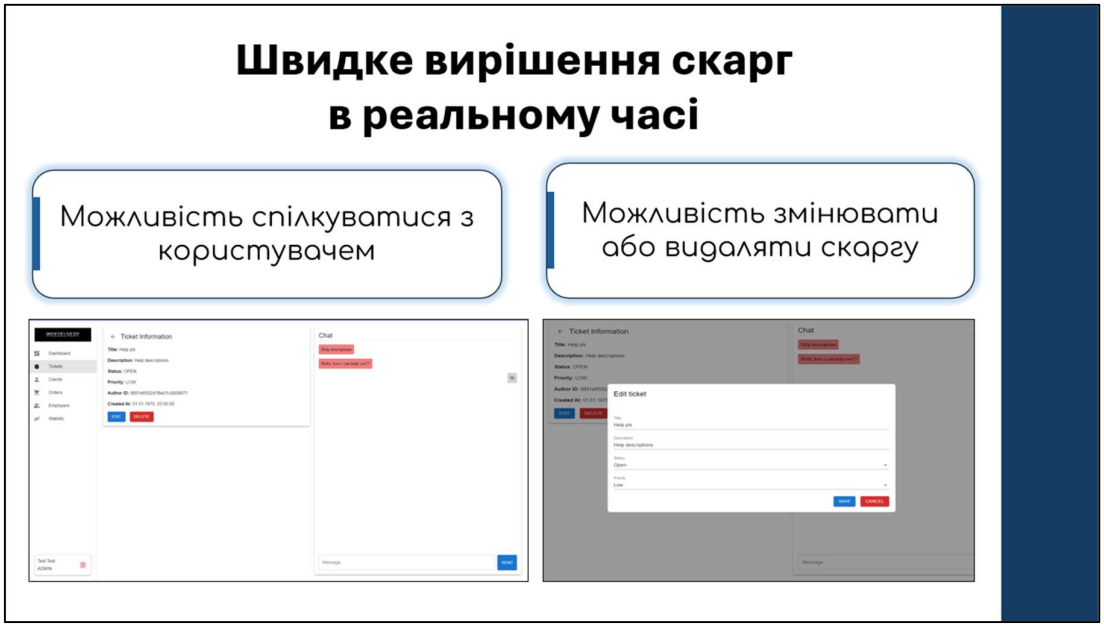


Рисунок Б.8 – Слайд 8

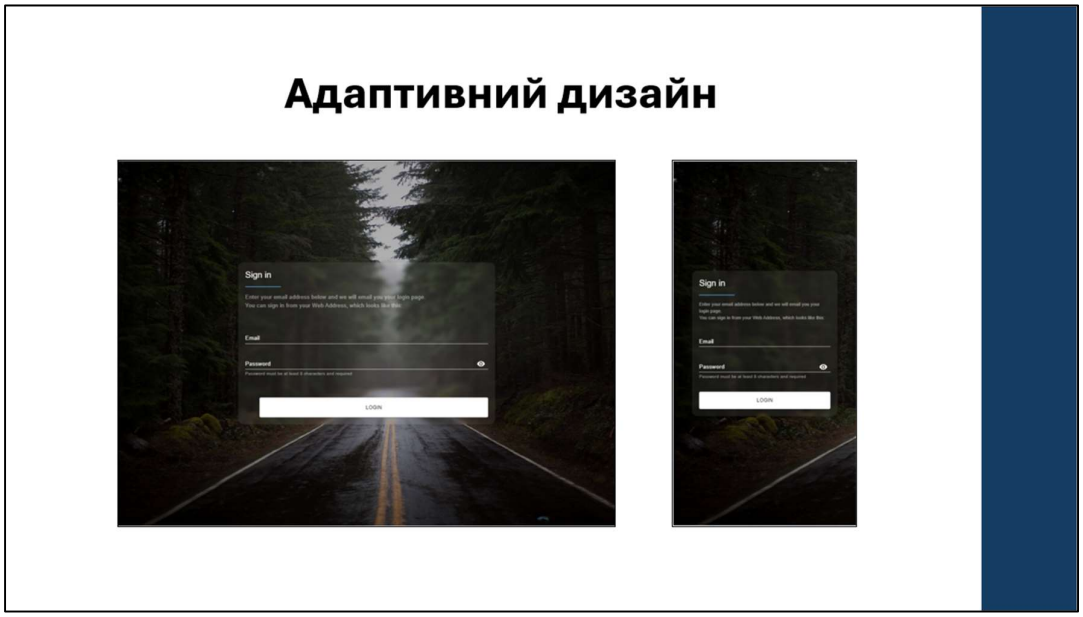


Рисунок Б.9 – Слайд 9

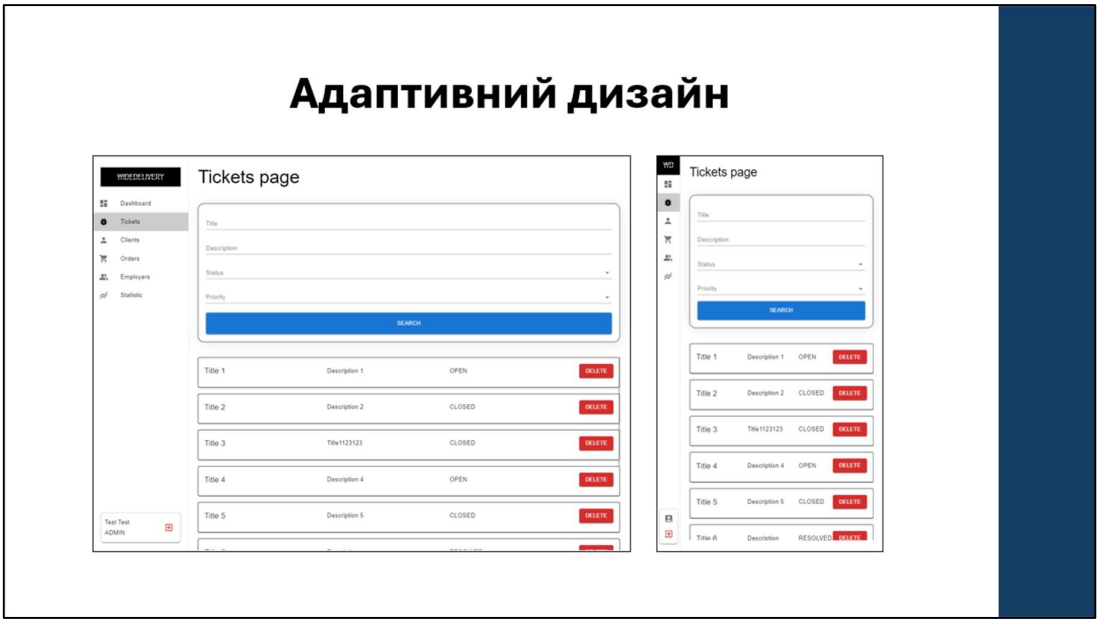


Рисунок Б.10 – Слайд 10

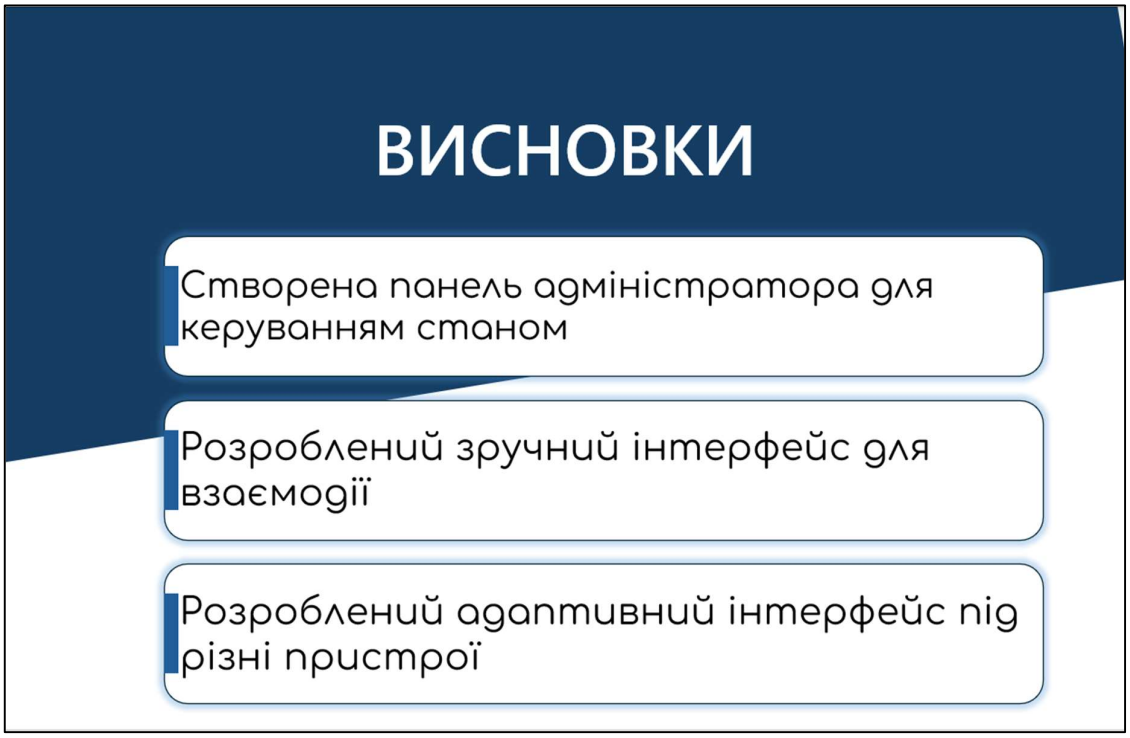
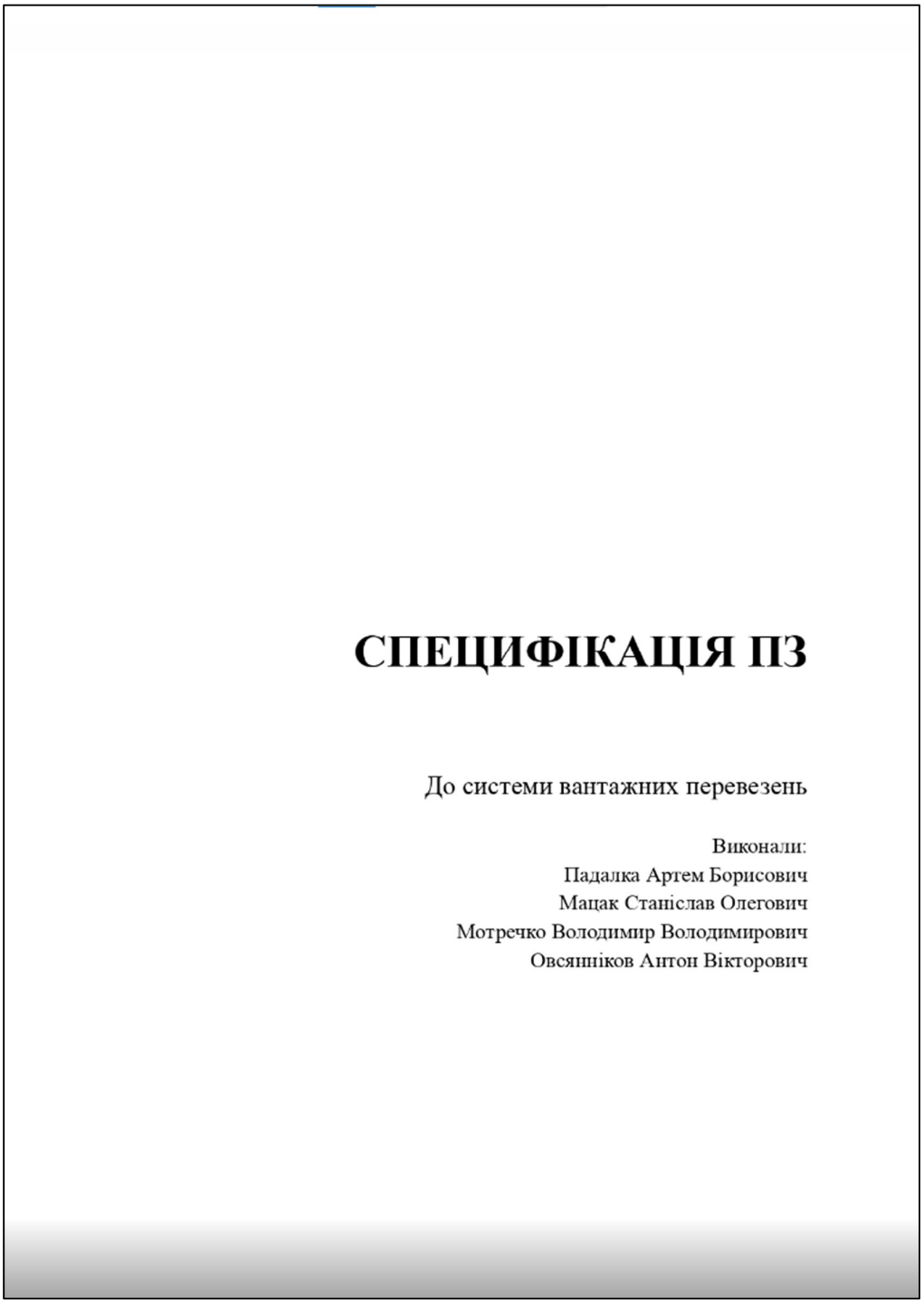


Рисунок Б.11 – Слайд 11

Рисунок Б.12 – Слайд 12

ДОДАТОК В

SRS-документ



СПЕЦИФІКАЦІЯ ПЗ

До системи вантажних перевезень

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

Рисунок В.1 – Сторінка 1 SRS-документу

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 SRS-документу

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 SRS-документу

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 SRS-документу

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 SRS-документу

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 SRS-документу

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 SRS-документу

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 SRS-документу

- 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 SRS-документу

- 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 SRS-документу

- 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 SRS-документу

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 SRS-документу

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 SRS-документу

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 SRS-документу

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.15 – Сторінка 15 SRS-документу

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.16 – Сторінка 16 SRS-документу

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.17 – Сторінка 17 SRS-документу

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.18 – Сторінка 18 SRS-документу

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.19 – Сторінка 19 SRS-документу

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.20 – Сторінка 20 SRS-документу

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.21 – Сторінка 21 SRS-документу

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.22 – Сторінка 22 SRS-документу

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.23 – Сторінка 23 SRS-документу

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.24 – Сторінка 24 SRS-документу

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.25 – Сторінка 25 SRS-документу

- 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.26 – Сторінка 26 SRS-документу

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.27 – Сторінка 27 SRS-документу

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.28 – Сторінка 28 SRS-документу

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.29 – Сторінка 29 SRS-документу

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.30 – Сторінка 30 SRS-документу

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.31 – Сторінка 31 SRS-документу

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.32 – Сторінка 32 SRS-документу

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.33 – Сторінка 33 SRS-документу

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.34 – Сторінка 34 SRS-документу

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.35 – Сторінка 35 SRS-документу

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.36 – Сторінка 36 SRS-документу

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.37 – Сторінка 37 SRS-документу

ДОДАТОК Ґ

Приклад програмного коду (AuthLayout)

```
import { Navigate, Outlet, useLocation, useNavigate } from "react-router-dom";
import useAuthStore from "@/store/AuthStore";
import { useAuthStoreType } from "@/store/AuthStore/types";
import { useEffect } from "react";
import { clearLocalStorageUserData } from "@/utils/utils";
import { hasAccess } from "@/utils/utils";
import { rolePermission } from "@/constants";

const AuthorizedLayout = () => {
  const { pathname } = useLocation();
  const navigate = useNavigate();
  const isAuth = useAuthStore((state: useAuthStoreType) => state.isAuth);
  const setCurrentUser = useAuthStore(
    (state: useAuthStoreType) => state.setCurrentUser
  );
  const clearState = useAuthStore(
    (state: useAuthStoreType) => state.clearState
  );

  useEffect(() => {
    const checkAuth = async () => {
      let isAccess: boolean = false;
      const user = await setCurrentUser().catch((error) => null);

      if (user) {
        isAccess =
          hasAccess(pathname, rolePermission, user.role[0]) ||
          pathname === "/profile";
      }

      if (!isAuth || !isAccess) {
        clearLocalStorageUserData();
        clearState();

        navigate("/sign-in");
      }
    };

    checkAuth();
  }, []);
```

```
  return <Outlet />;
};

export default AuthorizedLayout;
```