Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та
комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних систем та комп'ютерного моделювання
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи
перший (бакалаврський)
(освітньо-кваліфікаційний рівень)

на тему: **Розроблення інформаційної системи "Free truck"**
(тема роботи)

Виконав студент групи ІСТ-41
126 „Інформаційні системи та технології"
(шифр і назва напрямку підготовки спеціальності)

Кицак Т.М.
(прізвище, ініціали)

Керівник: Флуд Л.О.
(прізвище, ініціали)

Рецензент:
(прізвище, ініціали)

Львів-2023

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну

Кафедра інформаційних систем та комп'ютерного моделювання

Рівень вищої освіти перший (бакаравський)

Спеціальність 126 „Інформаційні системи та технології"

ЗАТВЕРДЖУЮ:
В.о. завідувача кафедри ІСКМ
_____ Сторожук О.Л.
„21" ____11____ 2022 року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Кицак Тарас Михайлович

(прізвище, ім'я, по батькові)

1.Тема бакалаврської роботи: «Розроблення інформаційної системи "Free truck"»

керівник роботи : доцент, Флуд Л.О., к.т.н ,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "21" листопада 2022 року, №С-521

2.Термін подання студентом проекту(роботи) 10 червня 2023р

3. Вихідні дані до проекту (роботи) Розробити програмну реалізацію та візуалізацію інформаційної системи "Free truck".

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Стан проблемної області

Інформаційне забезпечення

Програмне забезпечення

Висновки

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

Підготовка матеріалу до доповіді.

6. Дата видачі завдання 23 листопада 2022р.

КАЛЕНДАРНИЙ ПЛАН

№, з/п	Етапи бакалаврської роботи	Термін виконання етапів роботи	Примітка
1.	Огляд літератури згідно досліджуваної теми. Збір необхідних матеріалів.	15.12.22-25.01.23	вик.
2.	Розділ 1 Стан проблемної області	25.01.23-20.02.23	вик.
3.	Розділ 2 Інформаційне та математичне забезпечення	20.02.23-12.03.23	вик.
4.	Розділ 3 Програмне та технічне забезпечення	12.03.23-1.05.23	вик.
5.	Представлення пояснювальної записки на рецензування	1.05.23-20.05.23	вик.
6.	Підготовка доповіді	20.05.23-10.06.23	вик.

Студент _____ Кицак Т.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Флуд Л.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота містить 65 сторінки пояснювальної записки, 34 рисунок, 15 джерел, 1 додаток.

У дипломній роботі розроблено веб-орієнтовану інформаційну систему “Free truck”. Дану роботу було виконано з використанням засобів JavaScript та React.

Дана система дозволяє користувачам ознайомитися з усіма логістичними пропозиціями та запитами на ринку автоперевезень. Розроблена інформаційна система дає змогу користувачам знаходити найоптимальніші способи перевезень та доставки.

Ключові слова: web-система, сайт, програмний код, мова програмування, JavaScript, React.

ABSTRACT

The thesis contains 65 pages of explanatory note, 34 figures, 15 sources, 1 appendix.

The thesis developed a web-oriented information system "Free truck". This work was performed using JavaScript and React tools.

This system allows users to familiarize themselves with all logistical offers and requests in the road transport market. The developed information system allows users to find the most optimal methods of transportation and delivery.

Keywords: web system, site, program code, programming language, JavaScript, React.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити веб-орієнтовану інформаційну систему, яка повинна включати:

- Оболонку та інтерфейси створені веб-засобами;
- Веб-сторінки, які дозволяють ознайомитися з пропозиціями для вантажних перевезень;
- Супровідну документацію до проекту.

Створити структуру системи, інтуїтивно зрозумілу для користувача, з метою оптимізації пошуку та перегляду потрібної інформації.

Оформити сторінку відповідно стандартів та загальних норм проектування веб-додатків і інформативним дизайном.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ 7
ВСТУП 8
РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ 9
1.1. Огляд проблемної області 9
1.2 Переваги найму транспортної логістичної компанії 11
РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ WEB-СИСТЕМ 14
(ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ) 14
2.1 JavaScript 14
2.2. REACT 18
2.3. Коли використовувати React 26
2.4. React vs Angular vs Vue.js - Що краще 26
РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ 34
3.1. Використання React для розробки програмної частини веб-сайту компанії « Free Truck» 34
3.2. Тестування веб-сайту компанії «Free Truck» 56
ВИСНОВКИ 65
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 66
ДОДАТКИ 67

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface, інтерфейс програмування застосунків;

CI/CD - Continuous Integration / Continuous Delivery, безперервна інтеграція та розгортання програмного забезпечення;

ESLint - інструмент статичного аналізу коду JavaScript для виявлення та виправлення помилок;

FTL - Full Truck Load, перевезення повним завантаженням вантажівки;

HTML - HyperText Markup Language, мова розмітки гіпертексту;

IFTA - International Fuel Tax Agreement, міжнародна угода щодо обліку паливного податку;

JS - JavaScript, мова програмування для створення інтерактивних веб-додатків;

LTL - Less Than Truck Load, перевезення часткового завантаження вантажівки;

NPM - Node Package Manager, менеджер пакетів для середовища Node.js;

TS - TypeScript, типізована надбудова над мовою JavaScript;

UI - User Interface, користувацький інтерфейс;

UX - User Experience, досвід взаємодії користувача з системою.

ВСТУП

Вантажоперевезення, безсумнівно, є однією з найважливіших галузей у сучасному світі, що забезпечує безперебійну роботу інших секторів завдяки її першорядній позиції в ланцюжку поставок. Щодня водії перевозять усілякі делікатні, легкозаймисті та важкі матеріали на незліченну кількість кілометрів, завдяки чому заводи, продуктові магазини та інші торгові точки можуть працювати безперебійно. Важливість транспортних логістичних компаній важко переоцінити, особливо, під час війни в Україні, коли потреба в перевезеннях вантажів суттєво зросла. Логістичні компанії планують і виконують весь процес транспортування, а також здатні вивозити вантажі будь-якого розміру.

Об’єктом дослідження є створення веб-орієнтованої інформаційної системи “Free truck” засобами JavaScript та React.

Метою роботи є розробка інформаційної системи, яка буде давати змогу користувачу отримувати інформацію про вільні вантажівки серед переліку різноманітних логістичних компаній, а також довідатися про оптимальні маршрути та способи доставки.

Предметом дослідження є реалізація веб-орієнтованої інформаційної системи, використовуючи веб-засоби.

Практичне значення роботи полягає у представленні можливості користувачу швидко отримати необхідну інформацію про засоби та можливості доставки. Ознайомитися з наявними пропозиціями, та забронювати найоптимальніший варіант.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Огляд проблемної області.

Підприємства будь-якого розміру залежать від галузі вантажних перевезень. Індустрія вантажних перевезень обробляє набагато більше вантажів, ніж потяги, кораблі чи літаки, і без вантажівок вантажі ніколи не могли б подорожувати від станцій, портів і аеропортів до кінцевого пункту призначення. Якби індустрія вантажних перевезень зупинилася, економіка світу зупинилася б також. Перш за все вантажівки доставляють сировину виробникам. Наприклад, перевозять сировину від місцевих постачальників, таких як шахти, кар'єри, ферми та лісозаготівлі, до заводів, яким необхідні матеріали для виробництва продукції. Потім готова продукція переміщується на вантажівках до гуртових і роздрібних торговців або до інших транспортних каналів, щоб подорожувати кораблем, літаком або потягом до місць призначення по всьому регіону, країні або світу.

Є багато причин, чому послуги вантажних перевезень такі популярні у світі логістики, і одна з головних причин полягає в їх економічній природі та вигоді для бізнесу. Витрати та клопоти, пов'язані з організацією власної доставки, складських одиниць і купівлі вантажівок у поєднанні з поточними платіжками за пальне, водіїв та ремонт, миттєво усуваються при роботі з транспортною логістичною компанією. Користувачі сервісу хочуть мати можливість відстежувати свої замовлення в режимі реального часу з моменту їх розміщення. Все це стає можливим завдяки доступу до новітніх технологій під час найму транспортної компанії.

Малі автотранспортні підприємства часто працюють за моделлю власник-оператор, що означає, що водій вантажівки є самозайнятим. Великі автотранспортні підприємства часто наймають профспілкових водіїв. Подібно до того, як асоціації вантажних перевізників захищають інтереси вантажоперевізної галузі в цілому, профспілки працюють на захист інтересів водіїв. Міжнародне братство водіїв, наприклад, є великою профспілкою, яка може мати величезний вплив на економіку [1]. Один масштабний страйк може призвести до зупинки економіки, спричинивши затримки доставки та значне зростання цін, оскільки магазини намагаються

задовольнити споживчий попит.

Цілі транспортних компаній повинні бути зосереджені навколо необхідності відповідати очікуванням клієнтів. Потреби клієнтів включають завантаження та переміщення вантажів якомога швидше та безпечніше, а також доставку вантажів саме тоді, коли очікується, і в такому самому стані, як товари були під час завантаження на вантажівку [2]. Тарифні розклади повинні бути розроблені таким чином, щоб створити відмінну конкурентну перевагу, і найважливішою метою може бути наймання найкращих водіїв і допоміжного персоналу в галузі.

Безпека транспортування

Цілі безпеки встановлюються для благополуччя як співробітників, так і клієнтів. Зразок заяви про безпеку від Управління з охорони праці: «Якщо це небезпечно для здоров'я, ми не будемо цього робити». Безпека на робочому місці для транспортних компаній включає фронт-офіс, склад, вантажний майданчик і шосе. Безпека на дорозі включає дотримання періодів відпочинку водіїв, стандартів технічного обслуговування транспортних засобів і вимог до кваліфікації водіїв. Мета полягає в тому, щоб усунути або пом'якшити всі можливі ризики для безпеки на території компанії, усі ризики для безпеки на шосе та всі ризики для безпеки під час перебування співробітників на підприємствах клієнтів [3].

Стандарти якості доставки

Транспортні компанії володіють знаннями та досвідом, щоб доставляти товари без затримок або пошкоджень. Транспортна та логістична експертиза включає експортну та імпортну документацію, перевезення небезпечних вантажів, а також відстеження та розслідування втрат товарів під час транзиту. Мета повинна полягати в тому, щоб кожен продукт був доставлений до кінцевого пункту призначення без будь-яких пошкоджень або втрати вартості [4]. Додатковою метою транспортування є своєчасна доставка товару. Загальна мета транспортних компаній полягає в тому, щоб вантаж доставлявся вчасно та завжди.

1.2 Переваги найму транспортної логістичної компанії

Для більшості підприємств транспортування вантажів є головною проблемою. Транспорт використовується окремими особами та організаціями для різних цілей, від переміщення вантажів до доставки вантажів. У будь-якому випадку важливо забезпечити ефективне транспортування великогабаритних речей на великі відстані.

Історично вантажівки використовувалися для недорогих і безпечних перевезень великогабаритних вантажів на великі відстані. Використання вантажних автомобілів забезпечує найкраще поєднання ваги та часу з іншими видами транспортування вантажів, такими як морські та повітряні перевезення [5]. У той час як часи змінилися, а технології просунулися, ми продовжуємо залежати від вантажних перевезень для перевезення товарів і витратних матеріалів.

Транспортна логістична компанія є розумним вибором для будь-якого бізнесу, особливо під час війни в Україні, коли потреба в перевезеннях вантажів суттєво зросла. Логістичні компанії планують і виконують весь процес транспортування, а також здатні вивозити вантажі будь-якого розміру. Існує поширена думка, що вони забирають вантажі, а потім доставляють їх до кінцевого пункту призначення. Однак не все так просто.

Переваги співпраці з транспортними логістичними компаніями:

- Забезпечення економічної ефективності
- Забезпечення своєчасного прийому і доставки разом із передовими системами відстеження
- кращий підхід до управління ризиками
- багатофункціональні вантажівки, що задовольняють різні потреби та ведуть точну документацію

Співпрацюючи з логістичною компанією з вантажних перевезень, є можливість покращити свою операційну ефективність, заощаджуючи на витрати.

Логістичні компанії вантажівок одночасно обслуговують численні замовлення від різних компаній. Фактично, деякі компанії пропонують своїм постійним клієнтам знижки, надають першокласні послуги за доступними цінами, оскільки мають засоби для транспортування великої кількості вантажів і робочу силу, щоб

процес пройшов гладко.

Оскільки час – це дійсно гроші. Швидша й ефективна доставка продукції збільшує успіх бізнесу. Своєчасний прийом і доставку забезпечують передові системами відстеження [6]. Сьогодні є можливість скористатися новітніми технологіями, використовуючи програмне забезпечення для керування вантажними перевезеннями, які тепер мають просту у впровадженні систему відстеження. Це забезпечує прозорість і допомагає клієнтам знати, коли їхні замовлення/вантаж буде доставлено до потрібного пункту призначення.

Через ризики, пов’язані з транспортуванням, ланцюжок постачання постійно вразливий до збоїв. Під час транспортування можуть виникнути різні проблеми. Потенційним ризиком також може бути зміна місцевих чи міжнародних правил або викрадення конфіденційної інформації. Завжди існує ймовірність того, що ваші вироби можуть бути пошкоджені. Використовуючи транспортну логістичну компанію, є можливість значно зменшити ці ризики, та бути впевненим, що все, що піде не так, буде вирішено швидко й ефективно. Додатковою перевагою найму транспортної компанії є те, що пакунки обробляються лише під час прийому та доставки, що зменшує ймовірність пошкодження.

Стандартні послуги з перевезення вантажів нададуть вантажівки, які відповідатимуть необхідним специфікаціям для транспортування певного виду вантажів без будь-яких негативних наслідків. Галузь логістики складається з більш ніж просто вантажівок одного розміру або одного типу. Клієнт може скористатися такими послугами, як доставка FTL і LTL, транспортування бортовими вантажівками, важкі перевезення, доставка в той же день і на наступний день, інтермодальні перевезення, управління вантажами, складування та багато іншого.

Мережа транспортно-логістичних компаній розгалужена й охоплює тисячі кілометрів. Набагато дешевше найняти одну компанію для всіх транспортних потреб, ніж працювати з кількома постачальниками. Для успішного ведення бізнесу вкрай важливо завжди пам’ятати про економічну ефективність. Це єдиний спосіб розвивати свій бізнес, зберігаючи прибуток. Ведення успішного бізнесу вимагає розуміння, коли вартує виконувати певні завдання самостійно, а коли краще

найняти третю сторону або аутсорсинг. Переваги логістичної компанії вантажоперевезень численні. Логістичні компанії вантажоперевезень мають глибокі знання логістики та працюють систематично. Вони знають усі нормативні акти, включаючи звітність калькулятора Ifta, і надають прозорі законні послуги. Транспортні компанії використовують передову технологію, як-от програмне забезпечення Trucking Management, для відстеження та доставки товарів. Вони допомагають зменшити ризик і підвищити впевненість клієнтів.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ WEB-СИСТЕМ (ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ)

2.1 JavaScript

Згідно з нещодавнім звітом W3Techs, JavaScript є популярною мовою, яку використовують майже 95% усіх сайтів. JavaScript App використовується як клієнтська мова програмування, коли вам потрібно щось більше, ніж просто статичні зображення. Функції JavaScript включають мови сценаріїв, вбудовані функції, обробку подій та багато іншого [7]. Це чудовий інструмент для внесення важливих змін. Його популярність пояснюється простотою моделі, виявленням браузера користувача та дизайном з меншим навантаженням. У 2022 році з'явилися останні версії, які роблять його набагато швидшим. Також споживання пам'яті в два рази менше 2022 рік приніс багато нових функцій JavaScript.

Одна з найкращих тенденцій 2021 року щодо фреймворків JavaScript – зробити правильний вибір щодо фреймворку для свого веб-сайту.

- ✓ Angular — за підтримки Google Angular.js було оновлено та перейменовано на Angular у 2016 році. Angular 9 — остання версія, випущена в 2021 році. Зараз це один з найпопулярніших і надійних фреймворків для великих підприємств. Наприклад, YouTubeTV і NBA.com використовують Angular для своїх сайтів. TypeScript містить усі функції, необхідні для створення великих проєктів.
- ✓ Vue.js — це один із найцікавіших фреймворків JavaScript у трендах 2021 року. Хоча він не підтримується важливими клієнтами, це одна з найефективніших структур для використання. Vue.js використовували Ali Baba, Nintendo та GitLab. Vue має Vuex, великий обсяг пам'яті, чудова продуктивність і надійна програма. Серед усіх фреймворків, таких як React і Angular, Vue.js є законодавцем моди, який може виявитися корисним у 2023 році.
- ✓ React — очевидний лідер у системі JavaScript. Даний фреймворк застосовує реактивні процеси, запроваджуючи безліч індивідуальних концепцій/підходів для розробки інтерфейсу.

Кожен фреймворк має свої переваги та проблеми. Тому перед тим як обрати

оптимальний варіант слід проаналізувати кожен з цих трьох фреймворків.

TypeScript є єдиним у своєму роді серед усіх тенденцій технології JavaScript. Якщо у вас великий проект, доцільно використовувати TypeScript, оскільки це, безумовно, найголовніше у розробці програмного забезпечення [8]. Його можна використовувати, встановивши пакет NPM (Node Package Manager). Нижче наведено причини для його використання:

- TypeScript робить JavaScript не просто мовою сценаріїв — за останнє десятиліття на ринку з'явилося багато вражаючих мов, і одна з них — TypeScript. Однією з причин популярності TS є те, що деякі мови програмування не можуть використовувати нові процеси в апаратному забезпеченні, наприклад, у швидших мережевих або хмарних рішеннях. Отже, система структурної типізації TS компілює JavaScript.
- Найсучасніший — він не тільки полегшує роботу завдяки впізнаваним шаблонам кодування, але й має найсучасніші технології. Він підтримує розвиток і розвиток JavaScript. Такі функції, як ECMAScript 2015, асинхронні функції та декоратори, можуть виявитися корисними технологіями JavaScript у 2022 році.
- Швидша та простіша розробка — TS — це сучасна мова, яка в основному зосереджена на розвитку ергономіки (мінімізація втоми системи).

Перед тим як почати використовувати JavaScript у своїх майбутніх проектах, вартує бути в курсі останніх оновлень. Деякі важливі інструменти та функції, які слід проаналізувати:

- Інструменти моніторингу, аналіз журналів, аналітика (сервер, запити) — підприємства будь-якого розміру та стартапи часто використовують інструменти моніторингу та аналітики, оскільки вони пропонують ряд переваг, починаючи від повної чіткості продуктивності програмного забезпечення до можливості відстеження великої вартості сторінки. Такі інструменти можуть допомогти отримувати стабільні випуски в майбутньому, надаючи надійні послуги користувачам. Наприклад, JSLogger має чудові функції, які допомагають розпочати моніторинг активності користувачів на

сайті, одночасно керуючи вашими журналами в хмарі.

- Корисні оператори — для того, щоб зробити свій код суперсучасним або абстрактним, можна використовувати Chaining і Coalescing. Вони допомагають створити зрозумілий і стислий вираз, спрощуючи деякі загальні завдання та уникаючи важкості у коді.

JS у поєднанні з веб-компонентами — замість того, щоб використовувати якийсь великий фреймворк, можна почати використовувати веб-компоненти та JS без фреймворку. Попит на веб-компоненти надзвичайно зріс, оскільки вони набагато менші за стандартні компоненти інтерфейсу користувача та не потребують додаткового часу виконання (вони працюють виключно в браузері) [9].

Серед усіх останніх технологій JavaScript найбільш корисний і вигідний інструмент для розуміння наявних даних серед усіх останніх технологій GraphQL. Можливості GraphQL:

1. GraphQL дозволяє надсилати запит до API та отримувати передбачувані результати. Основні програми, створені командою Facebook, працюють на GraphQL і стабільніші за інші. Крім того, надають можливість виділити проблеми перед надсиланням запиту.
2. Один запит, кілька ресурсів — на відміну від будь-якого іншого інструменту, GraphQL допомагає знайти кілька посилань і збирає всі необхідні дані в одному запиті. Це також допомагає програмам працювати швидше, навіть якщо з’єднання з мережею повільне.

Щоб ефективно покращити продуктивність JS варта зосередитися на оптимізації даних та використовувати менш складні алгоритми для створення тонких структур даних; на обробці запитів і баз даних, зосередившись на покращенні обробки запитів. Варта спробувати методи/техніки оптимізації запитів; Тут можна використовувати інструменти моніторингу для реалізації кращих рішень. Додаток JS можна розгорнути на будь-якому сервері. Необхідно слідкувати за найновішими інструментами, службами, включаючи стратегії CI/CD для розгортання.

Якщо потрібна бездоганна система кодування, ESLint — найкращий варіант. Найменші помилки можуть призвести до краху всієї програми. ESLint — це одноетапне рішення всіх проблем, пов'язаних із кодуванням:

- ✓ Пошук проблем — незважаючи на те, що JavaScript має програмне забезпечення для усунення несправностей і пошуку помилок, вбудований ESLint набагато простіший і швидший [10]. Зазвичай він інтегрований у всю систему, а також є частиною текстових редакторів. Крім того, нещодавно об'єдналися TypeScript, TSLint, ESLint. Тепер ESLint перевіряє TypeScript шляхом інтеграції функцій TSLint. Це чудова новина, тобто ці проекти співпрацюватимуть і створюватимуть надзвичайно потужну базу для команди.
- ✓ Самовиправлення — важливо, щоб додаток був самодостатнім. ESLint — одна з тих програм, яка також виправляє деякі незначні проблеми під час аналізу. ESLint знає синтаксис і семантику, тому після їх виправлення у вас не виникне жодних проблем.

Безсерверна розробка відкриває хмарну розробку, і це є значним кроком для розробників інтерфейсу розробників. Безсерверний режим — це новий набір шаблонів для створення, розгортання та експлуатації рішень у хмарну епоху. Це відповідно вимагає нового мислення, щоб максимізувати переваги. Важливим аспектом архітектури є можливість забезпечити безперервне масштабування, підтримуючи велику кількість клієнтів/користувачів, подій і запитів у наданій програмі. Це може допомогти впоратися з такими викликами, не турбуючись про інфраструктуру, зосередившись більше на потребах бізнесу.

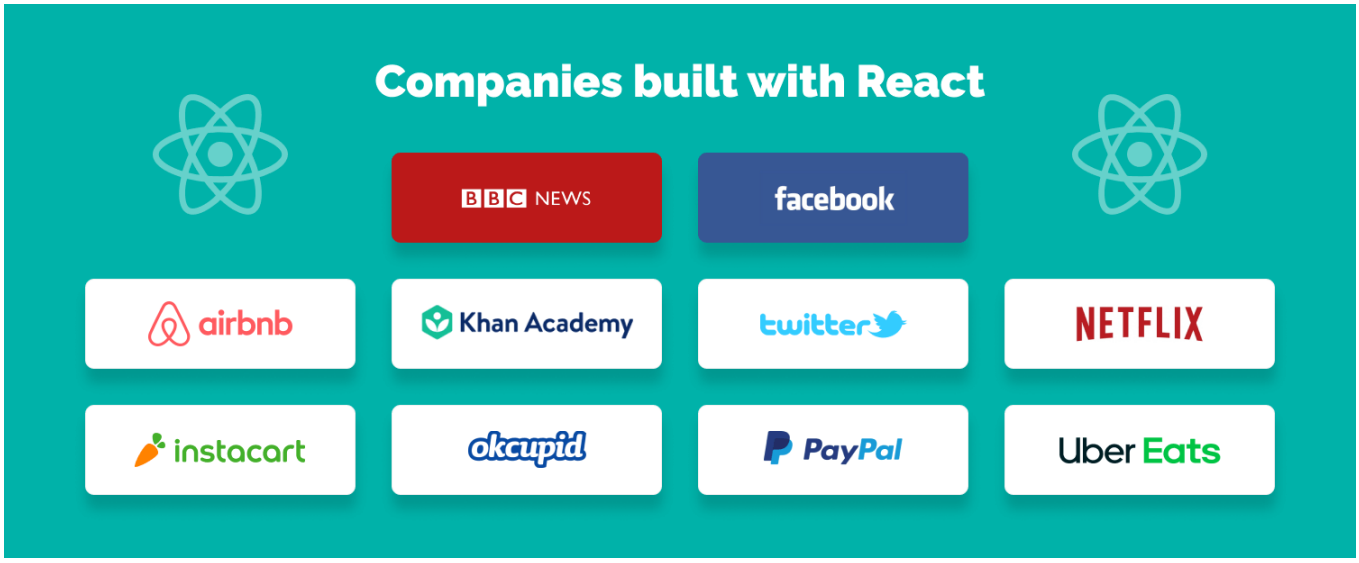
Для створення високоякісної програми достатньо скористатися єдиною кодовою базою JavaScript на безсерверній хмарній платформі. Можна використовувати JavaScript, щоб налаштувати хмарні ресурси та почати створювати й розгортати безсерверний JS. У цьому світлі є можливість модернізувати середовище за допомогою надійної та безпечної безсерверної хмарної платформи [11]. Це також питання операційних витрат хмарних обчислень, оскільки тут можна заощадити купу грошей на обслуговування в довгостроковій перспективі, перейшовши на безсерверні системи. Це, безперечно, найкращий з усіх останніх

трендів JavaScript.

2.2. REACT

Останніми роками галузь веб-розробки стабільно зростає; і оскільки це зростання продовжується, постійно з'являються нові технології, які допомагають розробникам створювати зручні веб-сайти та веб-додатки. Протягом багатьох років ми бачили, як мови веб-програмування створюють додаткові функції, більше мов програмування використовуються для створення веб-технологій, і навіть фреймворки та бібліотеки будуються на структурах існуючих технологій.

React — це бібліотека JavaScript, розроблена Facebook, яка використовується для створення інтерфейсів користувача [12]. Мета полягає в тому, щоб дозволити розробникам легко створювати швидкі користувацькі інтерфейси для веб-сайтів і програм. Основною концепцією React.js є віртуальний DOM. Це дерево на основі компонентів JavaScript, створених за допомогою React, яке імітує дерево DOM. Він виконує найменшу кількість маніпуляцій з DOM, щоб підтримувати компоненти React в актуальному стані.



По-перше, React був розгорнутий Facebook у 2011 та 2012 роках. Instagram був повністю написаний на React. За даними служби аналітики JavaScript Libscore, React зараз використовується на домашніх та інших веб-сторінках Netflix, Imgur, Bleacher Report, Feedly, Airbnb, SeatGeek, HelloSign та інших.

Чому варто використовувати React та як його встановити та використовувати? Що ж таке React? React.js, який зазвичай називають просто React, — це бібліотека

JavaScript, яка використовується для створення інтерфейсів користувача. Кожен веб-додаток React складається з багаторазово використовуваних компонентів, які складають частини інтерфейсу користувача - ми можемо мати окремий компонент для панелі навігації, один для нижнього колонтитула, інший для основного вмісту і так далі. Однією з основних причин використання React.js для веб-розробки є остаточно оптимізований інтерфейс розробки бібліотеки та мова кодування. Таким чином, спрощений API React підсилений швидкою продуктивністю для досягнення безпроблемного та швидкого робочого процесу розробки. Компоненти та концепції React прості для розуміння. На відміну від інших популярних фреймворків, таких як Vue та Angular, тут немає навали додаткових HTML-атрибутів (створених, коли JavaScript «впихають» у HTML — стандартна практика для традиційних фреймворків і бібліотечних рішень JS). У довгостроковій перспективі, помістивши JSX у JavaScript (буквально навпаки), React надає набагато чистіший, читабельніший і вичерпніший код.

Використання React.js для веб-розробки може бути надзвичайно зручним, оскільки React — це один із тих випадків, коли ви освоюєте одну технологію, щоб легко повторно використовувати її на різних платформах. І все завдяки тому, що це природна бібліотека, головною метою якої є створення окремих елементів веб-дизайну та компонентів (будь-що, від кнопок і міток до сіток та інтерактивних функцій).

До того ж популярність React зумовлена існуванням великої, давно створеної

спільноти. Поточна екосистема React настільки обширна, що дозволяє розробникам створювати рішення для настільних комп'ютерів і мобільні додатки, створювати статичні веб-сайти, обробляти серверний рендеринг та інтегрувати передові технологічні концепції (наприклад, віртуальну реальність і 360-огляди) з веб-рішеннями – і все це за допомогою подібної простої веб-розробки React настанови та філософії.

Безпроблемне повторне використання компонентів

Для чого React.js використовується у веб-розробці? Для створення окремих компонентів. І саме тому створені компоненти можна легко використовувати повторно. Після створення елемента веб-додатку React.js ми отримуємо унікальний об'єкт, який можна додати до будь-якого іншого проекту, сумісного з кодом на основі React.

Хоча більша загальна ієрархія побудована з цих компонентів (які загорнуті в компоненти вищого рівня), кожен має окрему спеціальну внутрішню логіку та принцип відтворення. Це надає чудові можливості масштабування, допомагає досягти кращої узгодженості веб-додатків React і робить подальшу підтримку та оптимізацію нескладними.

Потужності Flux і Redux

Особливий попит на React для веб-розробки також продиктований готовими можливостями Flux і Redux. Творці Facebook вперше представили програмну архітектуру на основі Flux, яка розширила стандартні компоненти React за допомогою односпрямованих можливостей потоку даних і запропонувала більш оптимальну структуру дій. Таким чином, центральний диспетчер використовується для оркестрування створених дій та оновлення сховищ. Потім він оновлює перегляди відповідно до змін. При цьому всі дані залишаються в сховищах - дублікати не генеруються, що допомагає підтримувати синхронізацію всіх даних моделі в усій програмі, не заходячи далеко.

Однак Flux — це лише архітектурний шаблон, який використовується у інтерфейсі для зручного робочого процесу розробки інтерфейсу користувача, який не можна використовувати як повноцінну бібліотеку. Ось де Redux стає зручною

реалізацією Flux. Зокрема, він пропонує єдиний об’єкт магазину для обробки всіх даних програми, що робить базові маніпуляції з керуванням даними простими та безпроблемними. Візуалізація запускається після змін магазину, тоді як перегляд продовжує синхронізуватися з пов’язаними даними.

Доступний великий набір інструментів

Серед іншого, наші фахівці пояснюють, чому ми використовуємо React JS для веб-розробки, висвітлюючи зразковий набір інструментів і стек технологій. Інструменти розробника React разом із інструментами розробки Redux пропонують надзвичайно зручні можливості. З їх допомогою можна ефективно перевіряти компоненти ієрархії на основі React (включно з пов’язаними атрибутами та станами), перевіряти дії надсилання та переглядати зміни стану негайно в розширенні (які також можна записати та використовувати як резервну копію для налагодження в майбутнє). Крім того, за допомогою інструментів керування вмістом для маркетологів веб-сайт можна створити або розширити хостингом, інструментами візуального редагування тощо.

Маючи один з 5 найкращих репозиторіїв на GitHub (із загалом понад 160 тисяч зірок), React просувається деякими з найширших спільнот і йому віддають перевагу кілька компаній зі списку Fortune 500 (зокрема Netflix, Uber, Amazon і Airbnb). В той самий час, він активно підтримується фахівцями, які працюють безпосередньо над системою програмного забезпечення Facebook, що саме по собі має бути достатньою причиною для використання React.js для веб-розробки. По суті, ви можете використовувати більше, ніж добре випробувані елементи, які лежать в основі найвидатнішого програмного рішення у світі.

Синтаксис JSX для розширеного HTML

Навіщо використовувати React для веб-розробки? З React.js можемо використовувати декларативний синтаксис HTML безпосередньо в коді JavaScript. Щоб показати інтерфейс користувача, браузери декодують тексти HTML. Вони роблять це шляхом створення дерев DOM, якими потім можна керувати за допомогою JavaScript для створення інтерактивного інтерфейсу користувача.

Маніпулювання DOM за допомогою різновидів ефективніше за допомогою JSX. Розробники можуть створювати охайний, підтримуваний код, передаючи компоненти HTML і React.js у деревовидні структури браузера. Програми React.js швидші та ефективніші завдяки JSX і Virtual DOM. Інші фреймворки та бібліотеки також можна використовувати з JSX.

Унікальні хуки React

Коли React Hooks було вперше представлено, було багато дискусій про те, чи замінить він Redux. Але Hooks — це нова функція в React.js 16.8, яка дозволяє авторам JavaScript додавати стани та інші функції до функціональних компонентів. Хуки спрощують керування логікою стану між компонентами, групують порівняльну логіку в один компонент і передають дані між компонентами, які не мають атрибутів або класів, що багато говорить на користь прийняття рішення про те, чому використовувати React для розробки веб-додатків.

Наявність цих багаторазових компонентів полегшує розробку, оскільки нам не потрібно повторювати повторюваний код. Нам просто потрібно створити його логіку та імпортувати компонент у будь-яку частину коду, де це потрібно.

React також односторінкова програма. Таким чином, замість того, щоб надсилати запит на сервер кожного разу, коли потрібно відобразити нову сторінку, вміст сторінки завантажується безпосередньо з компонентів React. Це призводить до швидшого відтворення без перезавантаження сторінки.

У більшості випадків синтаксис, який використовується для створення програм React, називається JSX (JavaScript XML), який є розширенням синтаксису JavaScript. Це дозволяє нам унікальним чином поєднувати логіку JavaScript і логіку інтерфейсу користувача. Завдяки JSX ми усуваємо потребу взаємодіяти з DOM за допомогою таких методів, як document.getElementById, querySelector та інших методів маніпулювання DOM.

Хоча використання JSX не є обов’язковим, воно полегшує розробку програм React. Ось приклад того, як ми можемо використовувати JSX у React:

```
function App() {
    const greetings = "Hello World";
        return (
            <div className="App">
            <h1> {greetings} </h1>
            </div>
        );
    }
}
```

У наведеному вище коді ми використовували функціональний компонент React для відтворення фрагмента тексту в браузері. Назва компонента App. Ми створили змінну перед функцією render(). Потім ми передали цю змінну в розмітку за допомогою фігурних дужок. Це не HTML, а синтаксис для написання коду за допомогою JSX. Далі ми розглянемо деякі причини, чому вам варто використовувати React. Багато розробників та організацій вибрали React замість інших бібліотек/фреймворків тому що:

- React легко освоїти та зрозуміти, якщо ви добре знаєте передумови. React має надійну документацію та багато безкоштовних ресурсів, створених іншими розробниками онлайн через дуже активну спільноту React.\
- Багаторазові компоненти: кожен компонент у React має власну логіку, яку можна повторно використовувати будь-де в додатку. Це зменшує потребу переписувати один і той самий фрагмент коду кілька разів.
- Можливості працевлаштування: на даний момент більший відсоток інтерфейсних веб-розробників має React як одну з необхідних навичок. Отже, розуміння того, як працює React і вміння працювати з ним, збільшує ваші шанси отримати роботу.
- Покращена продуктивність: завдяки віртуальній DOM React візуалізація сторінок може виконуватися швидше. У разі використання бібліотеки маршрутизації, як-от React Router, у нас відображатимуться різні сторінки без

перезавантаження.

- Можливість широкого розширення: React — це бібліотека, яка відображає лише інтерфейс користувача нашої програми. Розробник сам вибирає, з якими інструментами працювати, як-от бібліотеки для відтворення різних сторінок, бібліотеки дизайну тощо.

React використовували багато фронтенд-інженерів як у стартапах, так і в відомих компаніях, таких як Facebook, Netflix, Instagram, Yahoo, Uber, The New York Times тощо. Хоча всі перелічені вище компанії не створювали весь свій продукт за допомогою React, деякі їхні сторінки були створені за допомогою React. Це пов’язано з високою продуктивністю, простотою використання та масштабованістю React.

Особливості React

React має безліч чудових функцій, які й надалі роблять його популярним варіантом для розробників. Ось деякі з основних функцій React:

1. JSX: це розширення синтаксису JavaScript, яке розширює функції ES6 (ECMAScript 2015). Це дозволяє нам поєднувати логіку JavaScript і розмітку в компоненті.
2. Віртуальний DOM: це копія об’єкта DOM, який спочатку оновлює та повторно відображає наші сторінки після внесення змін; потім він порівнює поточний стан з оригінальним DOM, щоб підтримувати його синхронізацію зі змінами. Це призводить до швидшого відтворення сторінки.
3. Компоненти: програми React складаються з різних багаторазових компонентів, які мають власну відповідну логіку та інтерфейс користувача. Це робить його ефективним для масштабування програм і підтримки високої продуктивності, оскільки ви не дублюєте код так часто, як в інших фреймворках.

Плюси і мінуси React

React може бути популярним інструментом для створення нашого інтерфейсу користувача, але все ще є причини, чому деякі розробники або початківці вирішують не використовувати його.

Плюси використання React:

- ✓ React легко освоїти та зрозуміти.
- ✓ React має дуже активну спільноту, де ви можете зробити свій внесок і отримати допомогу, коли це необхідно.
- ✓ Є багато можливостей для працевлаштування для розробників React.
- ✓ React забезпечує підвищену продуктивність програми.

Мінуси використання React:

- o Початківцям, які не мають глибокого розуміння JavaScript (особливо ES6), може бути важко зрозуміти React.
- o React постачається без деяких загальних функцій, таких як керування єдиним станом і маршрутизація; доведеться встановити та навчитися використовувати для них зовнішні бібліотеки.
- o Передумови для використання React
- o Перш ніж використовувати React, потрібно добре розуміти та мати досвід роботи з JavaScript.

Попри всі плюси та мінуси React як інтерфейсна бібліотека продовжує стрімко зростати порівняно з іншими бібліотеками/фреймворками в спільноті розробників і не демонструє жодних ознак зупинки. Ми можемо знайти React на кожному сучасному веб-розробнику. З поточним впровадженням технології web3 все більшою кількістю розробників React залишається улюбленим інструментом для створення інтерфейсу децентралізованих програм (DApps). За допомогою React можна створювати різноманітні додатки, від простих веб-додатків із списками справ до ринків, інформаційних панелей тощо. React можна використовувати з багатьма технологіями, такими як Bootstrap, Tailwind CSS, Axios, Redux, Firebase та багато іншого. Ми також можемо використовувати React з Node.js та іншими серверними мовами для створення повноцінних додатків і веб-додатків, які працюють із блискавичною швидкістю.

За даними Statista, React швидко став одним із найпоширеніших фреймворків для створення сучасних веб-додатків: 42,62% респондентів повідомили, що вони використовують його.

2.3. Коли використовувати React.

Тож, коли вартує використовувати React:

Складні інтерфейси користувача

React відмінно справляється зі складними інтерфейсами користувача з легкістю. Якщо ваш веб-додаток має багато динамічних та інтерактивних елементів, таких як оновлення в реальному часі або візуалізація даних, React може допомогти створити зручну та бездоганну взаємодію з користувачем.

Масштабні програми

Компонентна архітектура React дозволяє створювати модульний і багаторазово використовуваний код, який може допомогти масштабувати веб-додаток у міру розвитку вашого бізнесу. Це робить React чудовим вибором для розробки великомасштабних програм, які вимагають зручності обслуговування, масштабованості та гнучкості.

Високопродуктивні програми

Ефективна система візуалізації React і оптимізована віртуальна DOM дозволяють обробляти великі обсяги даних і складні інтерфейси користувача без шкоди для продуктивності.

Кросплатформні програми

React можна використовувати для створення веб-, мобільних і настільних додатків, що робить його універсальним фреймворком для крос-платформної розробки. React Native, фреймворк на основі React, спеціально розроблений для розробки мобільних додатків, тоді як React Desktop дозволяє створювати настільні додатки за допомогою веб-технологій.

Можемо зробити висновок що, використання React.js досить широке та універсальне.

2.4. React vs Angular vs Vue.js - Що краще.

Які фреймворки JavaScript найкращі для компаній? Фреймворки JavaScript розвиваються надзвичайно швидкими темпами, а це означає, що сьогодні ми часто оновлюємо версії Angular, React.js та ще одного гравця на цьому ринку – Vue.js. Аналіз попиту, представлений у Google Trends за останні 5 років. Синя, червона та

жовта лінії представляють Angular, React і Vue.js відповідно (рис. 2.1).

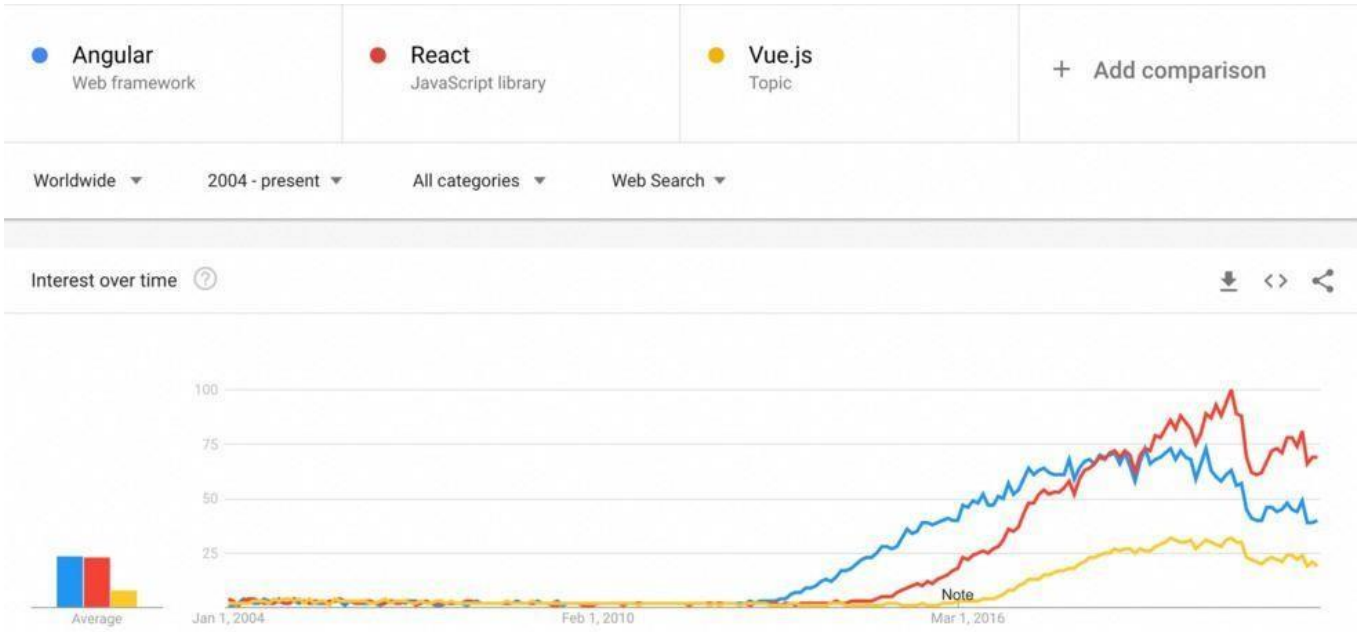


Рис. 2.1 Динаміка JavaScript-фреймворків за результатами Google Trends

З діаграми видно, що протягом 2014-2015 років була невелика різниця між кількістю запитів React і Angular. Тоді ми бачимо, що відхилення між ними збільшувалося на деякий короткий період. З середини 2017 року ці запити збалансувалися, і React почав рости та наближатися до попиту Angular. Фреймворк Vue.js ще не був дуже популярним, але трохи нарощував свою присутність на ринку фреймворків, демонструючи потенціал для подальшого зростання. В останні роки Angular і React майже збалансовані, що означає, що вони є найбільш використовуваними фронтенд-фреймворками на ринку.

Загалом, React і Angular розвиваються органічно з відносно однаковою динамікою. Якщо спробуємо спрогнозувати попит на ці фронтенд-фреймворки, то побачимо позитивну тенденцію для React, тоді як Angular має дещо знижений. Ситуація з попитом на послуги розробки Vue.js ще не ясна, але, судячи з його специфічної зручної структури, він також буде рости, можливо, трохи менше, ніж основні фреймворки.

Проаналізувавши кількість відкритих вакансій у всьому світі, які вимагають спеціальних знань певного фреймворку. Як джерело використано Indeed.com і отримано такий розподіл відповідно до понад 60 000 пропозицій про роботу (рис. 2.2).

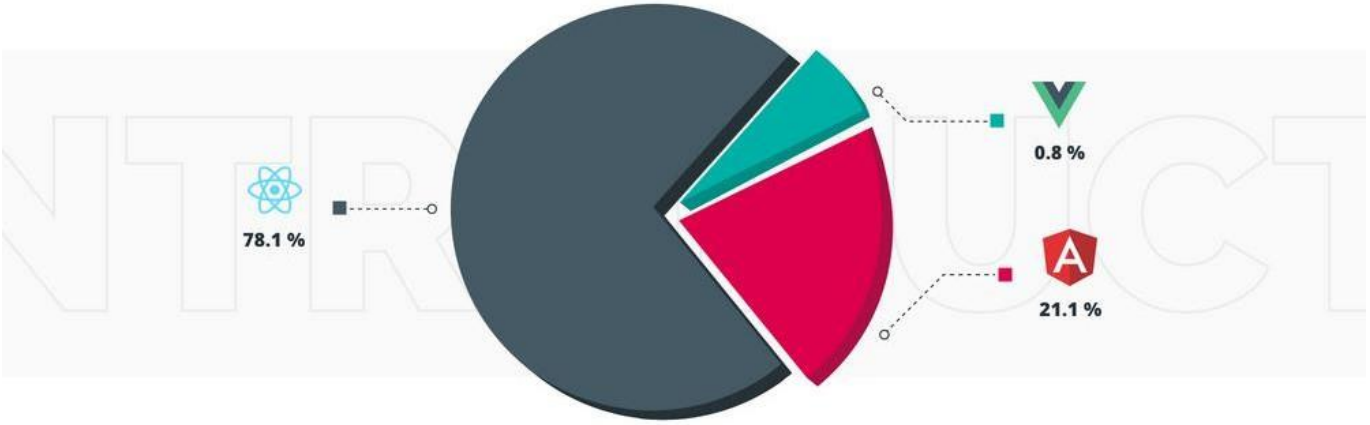


Рис. 2.2 Розподіл популярності JavaScript-фреймворків

Беручи до уваги наступні дані, проаналізуємо основні переваги та недоліки кожного фронтенд-фреймворку.

Плюси і мінуси Angular

Angular — це супергероїчний фреймворк JavaScript MVVM, заснований у 2009 році, який чудово підходить для створення високоінтерактивних веб-додатків.

Переваги Angular:

Нові функції, як-от вдосконалений RXJS, швидша компіляція (менш ніж за 3 секунди), новий запуск HttpClient.

Детальна документація, яка дозволяє отримати всю необхідну інформацію окремому забудовнику, не питаючи його колег. Однак для цього потрібно більше часу на навчання.

Двостороннє зв’язування даних, яке забезпечує унікальну поведінку програми, що мінімізує ризики можливих помилок.

MVVM (Model-View-ViewModel) дозволяє розробникам працювати окремо над одним розділом програми, використовуючи той самий набір даних.

Впровадження залежностей функцій, пов’язаних із компонентами з модулями та модульністю в цілому.

Недоліки Angular:

Складний синтаксис походить від першої версії Angular. Незважаючи на це, Angular 5 використовує TypeScript 2.4, який найменш складний для вивчення в

порівнянні.

Проблеми з міграцією можуть виникати під час переходу від старішої версії до останньої.

Компанії, які використовують Angular 5: Upwork, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather, iStockphoto, AWS, Crunchbase.

Першу версію Angular випустила Google у 2010 році. Найновішою версією, випущеною в 2019 році, є Angular 2+. Сьогодні популярні компанії, які використовують послуги розробки Angular, включають Google, Microsoft Office і Upwork.

Наприклад, Angular надає Google високофункціональну структуру для створення більших програм. Він використовується для розробки Google Cloud Console. Microsoft Office вдалося запустити дві автономні програми за допомогою Angular framework. У випадку Upwork він використовує фреймворк, щоб забезпечити адаптивний досвід для своїх учасників.

Плюси і мінуси React.js

ReactJS — це бібліотека JavaScript, відкрита Facebook у 2013 році, яка чудово підходить для створення величезних веб-додатків, у яких дані змінюються на регулярній основі.

Переваги React.js:

Легко навчатися. React набагато легше вивчити через його простоту з точки зору синтаксису. Інженерам просто потрібно пригадати свої навички написання HTML, і все. Немає необхідності глибоко вивчати TypeScript, як в Angular.

Високий рівень гнучкості та максимальна оперативність.

Віртуальна DOM (об’єктна модель документа), яка дозволяє впорядковувати документи у форматах HTML, XHTML або XML у дерево, яке краще сприймається веб-переглядачами під час аналізу різних елементів веб-програми.

У поєднанні з ES6/7 ReactJS може легко працювати з високим навантаженням.

Зв’язування даних вниз означає, що при такому типі потоку даних дочірні елементи не можуть впливати на батьківські дані.

Бібліотека JavaScript із 100% відкритим кодом, яка щодня отримує багато

оновлень і вдосконалень відповідно до внесків розробників з усього світу.

Абсолютно легкий, оскільки дані, що виконуються на стороні користувача, можуть бути легко представлені на стороні сервера одночасно.

Перехід між версіями зазвичай дуже простий, оскільки Facebook надає «codemods» для автоматизації більшої частини процесу.

Недоліки React.js:

Відсутність офіційної документації - надшвидка розробка ReactJS не залишає місця для належної документації, яка зараз дещо хаотична, оскільки багато розробників вносять її індивідуально без будь-якого систематичного підходу;

Порівнюючи Angular з React, останній не має жодної думки, тобто розробники іноді мають занадто великий вибір;

Довгий час на опанування, а це означає, що React JS вимагає глибоких знань про те, як інтегрувати інтерфейс користувача в структуру MVC.

Компанії, які користуються послугами компаній-розробників ReactJS: Facebook, Instagram, Netflix, New York Times, Yahoo, Khan Academy, Whatsapp, Codecademy, Dropbox, Airbnb, Asana, Atlassian, Intercom, Microsoft.

Великі мільярдні компанії, які використовують React, є доказом того, наскільки хороший фреймворк. Крім Facebook, більшість відомих компаній, які використовують ReactJS, це Twitter, Amazon Prime, Udemy, Expo, Codesandbox, Uber і Pinterest. Але в центрі цього — Facebook.

Здебільшого React є чудовим інструментом для вищезгаданих сервісів, оскільки вони мають високий трафік. Компанія-розробник React.js забезпечує швидший спосіб створення та підтримки мобільних додатків. Крім того, ці компанії можуть створювати програми, використовуючи менше ресурсів. Це найкраща інтерфейсна бібліотека з існуючих.

Плюси і мінуси Vue.js

Vue.js — це фреймворк JavaScript, запущений у 2013 році, який ідеально підходить для створення адаптивних інтерфейсів користувача та складних односторінкових програм.

Переваги Vue.js:

Розширений HTML. Це означає, що Vue.js має багато подібних характеристик до Angular, і це може допомогти оптимізувати обробку блоків HTML за допомогою різних компонентів.

Детальна документація. Vue.js має дуже детальну документацію, яка може пришвидшити процес навчання розробників і заощадити багато часу на розробку програми, використовуючи лише базові знання HTML і JavaScript.

Адаптивність. Він забезпечує швидкий період переходу з інших фреймворків на Vue.js через схожість з Angular і React з точки зору дизайну та архітектури.

Чудова інтеграція. Vue.js можна використовувати як для створення односторінкових програм, так і для складніших веб-інтерфейсів програм. Головне, що менші інтерактивні частини можна легко інтегрувати в існуючу інфраструктуру без негативного впливу на всю систему.

Велике масштабування. Vue.js може допомогти розробити досить великі багаторазові шаблони, які можна створювати без додаткового часу, виділеного на це відповідно до його простої структури.

Маленький розмір. Vue.js може важити близько 20 КБ, зберігаючи свою швидкість і гнучкість, що дозволяє досягти набагато кращої продуктивності в порівнянні з іншими фреймворками.

Недоліки Vue.js:

Брак ресурсів. Vue.js все ще має досить невелику частку ринку порівняно з React або Angular, що означає, що обмін знаннями в цьому фреймворку все ще знаходиться на початковій стадії.

Ризик надмірної гнучкості. Іноді у Vue.js можуть виникати проблеми під час інтеграції у великі проекти, і досі немає досвіду можливих рішень, але вони обов’язково незабаром з’являться.

Оскільки Vue.js має трохи китайського фону, багато елементів і описів все ще доступні китайською мовою. Це призводить до часткової складності на деяких етапах розробки, тим не менш, все більше матеріалів перекладається англійською мовою.

Компанії, які використовують Vue.js: Xiaomi, Alibaba, Wizz Air, Euronews,

Grammarly і Laracasts, Adobe, Behance, Codeship, Reuters.

Повноцінний фреймворк Vue.js трохи запізнився, але його використовують багато компаній. Деякі компанії, які використовують Vue, також включають Google, Apple, Trivago та Nintendo. Google зміг використати Vue для своєї сторінки кар'єри, а Nintendo включила цю структуру на свій веб-сайт. Trivago, один із найпопулярніших сайтів для пошуку готелів, включив Vue до свого журналу. З іншого боку, Apple використовувала Vue для створення своїх посібників SwiftUI. Насправді, щомісяця набагато більше компаній змінюють свої кодові бази, і близько 140 тис. починаються на платформі GitHub.

Який найкращий спосіб максимізувати переваги кожного фреймворку? Як дізнатися, коли використовувати React чи Angular? Базуючись на перевагах і недоліках кожного, проаналізуємо певні елементи, що допоможуть визначитися:

- ✓ Коли використовувати Angular

Коли порівняти популярність Angular і React серед розробників, React лише трохи популярніший за Angular. Тому, для початку порівняємо Angular з Vue.js і побачимо, коли його використовувати:

Angular більш корисний для великих складних проектів;

Angular є кращим варіантом для надійної масштабованості;

- ✓ Коли використовувати React

Що краще в битві React.js проти Angular? Популярність React порівняно з

Angular є близькою, де React трохи надається перевага. Основна причина, чому React більш популярний, якщо порівнювати React з Angular, полягає в недоліках Angular, згаданих раніше. Чудова екосистема React робить його кращим рішенням для: проектів, які можуть включати багаторазові компоненти; проектів, які хочуть мати простий інтерфейс; проектів, що вимагають найвищого рівня масштабованості; проектів зі стислими термінами.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Використання React для розробки програмної частини веб-сайту компанії « Free Truck».

Використання React для розробки програмної частини веб-сайту логістичної компанії "Free Truck" дозволяє створити веб-додаток, який є модульним, масштабованим і досить ефективним. За допомогою React, розробники можуть створювати складні інтерфейси з високою реактивністю, що забезпечує покращену користувацьку взаємодію.

Основними перевагами використання React є:

Компонентний підхід: React дозволяє розбити веб-інтерфейс на невеликі компоненти, які можуть бути повторно використані. Це спрощує розробку і підтримку коду, а також полегшує розподіл роботи між розробниками.

Віртуальний DOM: React використовує віртуальний DOM для ефективного оновлення та маніпуляції елементами інтерфейсу. Це забезпечує швидку роботу додатку і зменшує навантаження на браузер.

Одностороннє потокове оновлення даних: React працює за принципом одностороннього потоку даних, що дозволяє легко відслідковувати та керувати станом компонентів. Це полегшує виявлення та виправлення помилок.

Розширюваність: За допомогою додаткових бібліотек і інструментів, таких як Redux або React Router, можна розширити можливості React і забезпечити більшу функціональність і гнучкість[див. рис. 3.1].

Велике співтовариство розробників: React має велике та активне співтовариство розробників, яке надає безліч корисних ресурсів, документацію, пакети і підтримку.

Цей фреймворк дозволяє поєднати перевізників з клієнтами, створивши потужний інструмент для організації та керування перевезеннями.

За допомогою React, логістична компанія може створити спеціальні інтерфейси для перевізників, де вони зможуть швидко знаходити доступні

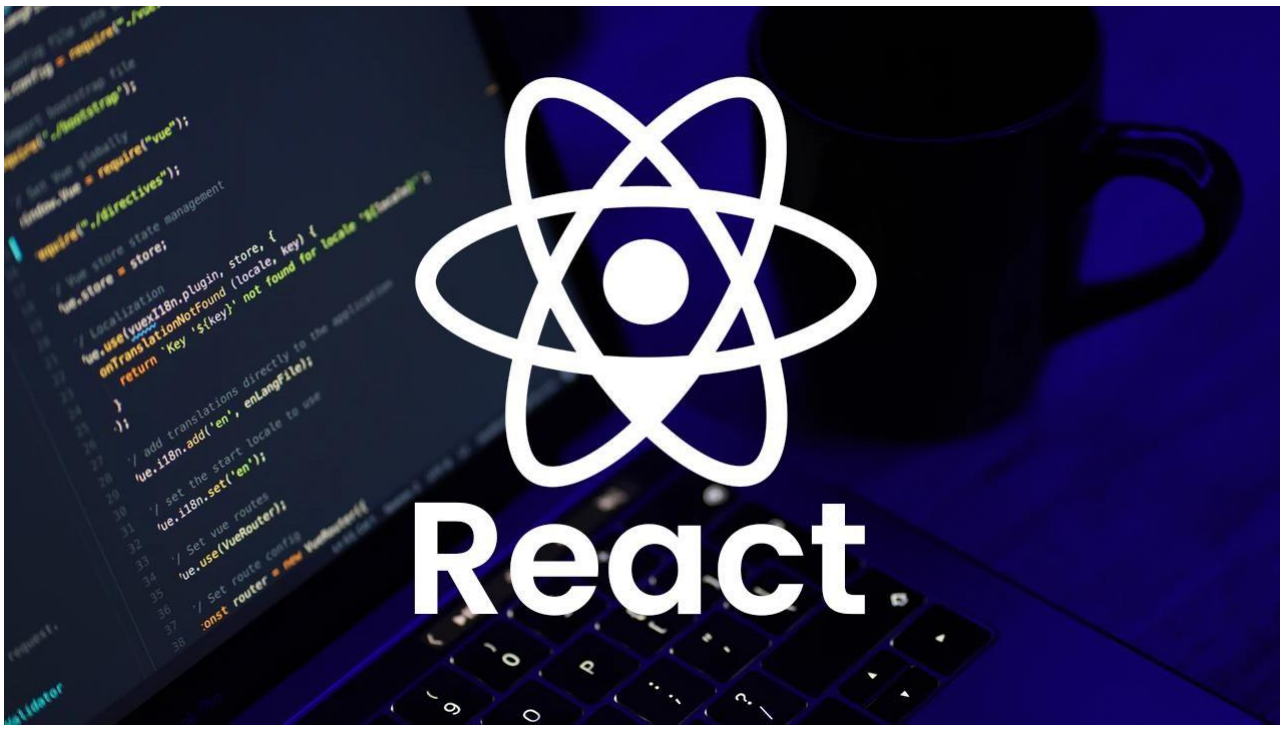


Рис. 3.1 React.

замовлення, взаємодіяти з клієнтами та керувати деталями перевезення. Розробники можуть створити динамічні списки, фільтри та пошукові функції, що допоможуть перевізникам ефективно виконувати свої обов'язки.

Крім того, React забезпечує можливість відстежувати та оновлювати стан перевезень у реальному часі. За допомогою WebSocket або інших технологій, перевізники можуть отримувати оновлення про нові замовлення, зміни у маршрутах або статусах вантажів. Це дозволяє перевізникам оперативно реагувати на зміни та покращує загальну ефективність перевезень.

Користувачам-перевізникам також надаються зручні інструменти для взаємодії з клієнтами. Вони можуть отримувати замовлення, здійснювати необхідні оновлення щодо перевезень. Такий інтерфейс допомагає перевізникам забезпечити високу якість обслуговування та задовольнення клієнтів.

Загалом, використання React для розробки програмної частини веб-сайту логістичної компанії "Free Truck" поєднує зусилля перевізників і клієнтів, створюючи потужну та зручну платформу для організації та керування перевезеннями.

Безліч переваг, роблять нашу платформу унікальною та привабливою для користувачів.

Перш за все, React надає нам можливість створювати компоненти, які можна повторно використовувати і легко управляти. Це дозволяє нам побудувати складну структуру веб-сайту з простих блоків, що спрощує розробку та підтримку коду.

Крім того, React використовує віртуальний DOM, що дозволяє нам забезпечити швидкість та ефективність роботи нашої платформи. Завдяки цьому механізму, React розраховує оптимальні зміни, які необхідно внести до відображення, замість повного перерендерингу всього дерева DOM. Це робить нашу платформу швидкою та масштабованою.

React також надає нам можливість управляти станом додатка за допомогою компонентів і хуків, таких як useState і useEffect. Це дозволяє нам реагувати на події та зміни даних у реальному часі, оновлюючи відображення відповідно до нових значень стану. Такий підхід дозволяє нам створювати інтерактивні та реактивні елементи нашої платформи.

Крім того, за допомогою React ми можемо легко інтегрувати інші бібліотеки та розширення, що дозволяє нам забезпечити додатковий функціонал та розширити можливості нашої платформи. Ми можемо використовувати багато готових компонентів, стилізаційних бібліотек та інших інструментів, що полегшують розробку та покращують вигляд нашого веб-сайту.

Усі ці особливості React доповнюють нашу програмну частину веб-сайту логістичної компанії "Free Truck" унікальними та потужними можливостями, що дозволяють нам створити сучасну, швидку та зручну платформу для наших клієнтів та перевізників.

Створення сайту включає кілька кроків, які допомагають забезпечити успішну розробку та запуск веб-проекту. Нижче описані основні етапи процесу створення сайту[див. рис. 3.2].

Рис. 3.2 Етапи розробки

Постановка цілей: Спочатку потрібно чітко визначити мету веб-сайту. Потрібнозрозуміти, яку функціональність потрібно надати користувачам, який тип веб-сайту створити і які результати потрібно досягти.

Планування та проектування: Наступним кроком є планування структури веб-сайту і проектування його вигляду. Можна створити візуальні макети, скетчі або використовувати дизайн-систему для створення концепції та розташування елементів на сторінках сайту.

Розробка фронтенду: За допомогою HTML, CSS і JavaScript створюється фронтенд веб-сайту. HTML використовується для створення структури сторінок, CSS - для стилізації та оформлення, а JavaScript – React для додавання інтерактивності та функціональності.

Тестування: Після завершення розробки важливо протестувати ваш веб-сайт, щоб переконатися, що всі функції працюють належним чином і немає помилок. Тестування можна проводити вручну або автоматизовано.

Розгортання: Після успішного тестування веб-сайт готовий до розгортання. Це означає, що ви розміщуєте ваш веб-сайт на сервері або хостинг-платформі, щоб він став доступним для користувачів через Інтернет.

Підтримка та поновлення: Після запуску веб-сайту важливо забезпечити його підтримку та здійснювати необхідні поновлення. Це може включати виправлення помилок, вдосконалення функціональності, оновлення залежностей тощо.

При розробці веб-сайту з використанням React, код зазвичай пишеться у файловій структурі проекту, яка може виглядати наступним чином[див. рис. 3.3]:

src/index.js: Цей файл є головним файлом в програмі React і відповідає за рендеринг вашого компоненту React в HTML-елемент на сторінці. У цьому файлі ви зазвичай підключаєте головний компонент за допомогою ReactDOM.render().

src/App.js: Цей файл містить головний компонент вашого додатку. Ви можете розширити його або використовувати його як контейнер для інших компонентів вашого веб-сайту.

src/components: У цьому каталозі зазвичай розміщуються всі компоненти вашого веб-сайту. Кожен компонент може бути розділений на окремі файли, які включають файл з кодом компонента та файл з стилями.

src/services: Якщо у вас є взаємодія з сервером або зовнішніми API, ви можете створити каталог "services" для розміщення файлів, які містять логіку взаємодії з цими сервісами.

src/store: Якщо ви використовуєте стейт-менеджер (наприклад, Redux) для керування станом вашого додатку, ви можете створити каталог "store", де розмістити файли, пов'язані зі станом та діями.

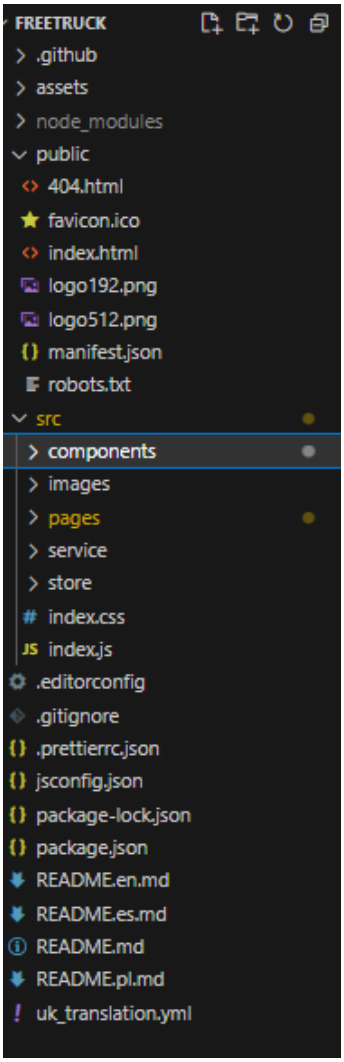


Рис. 3.3 Редактор коду Visual Studio Code з структурою.

Під час розробки веб-сайту на React підключені наступні бібліотеки та інструменти:

ReactDOM: ReactDOM є пакетом, який використовується для рендерингу React-компонентів в реальний DOM. Він взаємодіє з браузером та забезпечує оновлення та відображення компонентів на сторінці. ReactDOM робить процес рендерингу простим та ефективним, дозволяючи React-компонентам взаємодіяти з реальним DOM-деревом.

React Router: React Router є бібліотекою для маршрутизації в React-додатках. Вона дозволяє визначати маршрути та компоненти, які повинні бути відображені для кожного маршруту. За допомогою React Router можна створювати односторінкові додатки з багатьма сторінками та налаштовувати навігацію між сторінками. React Router також підтримує параметри маршруту, вкладені маршрути та історію переходів[див. рис. 3.4].

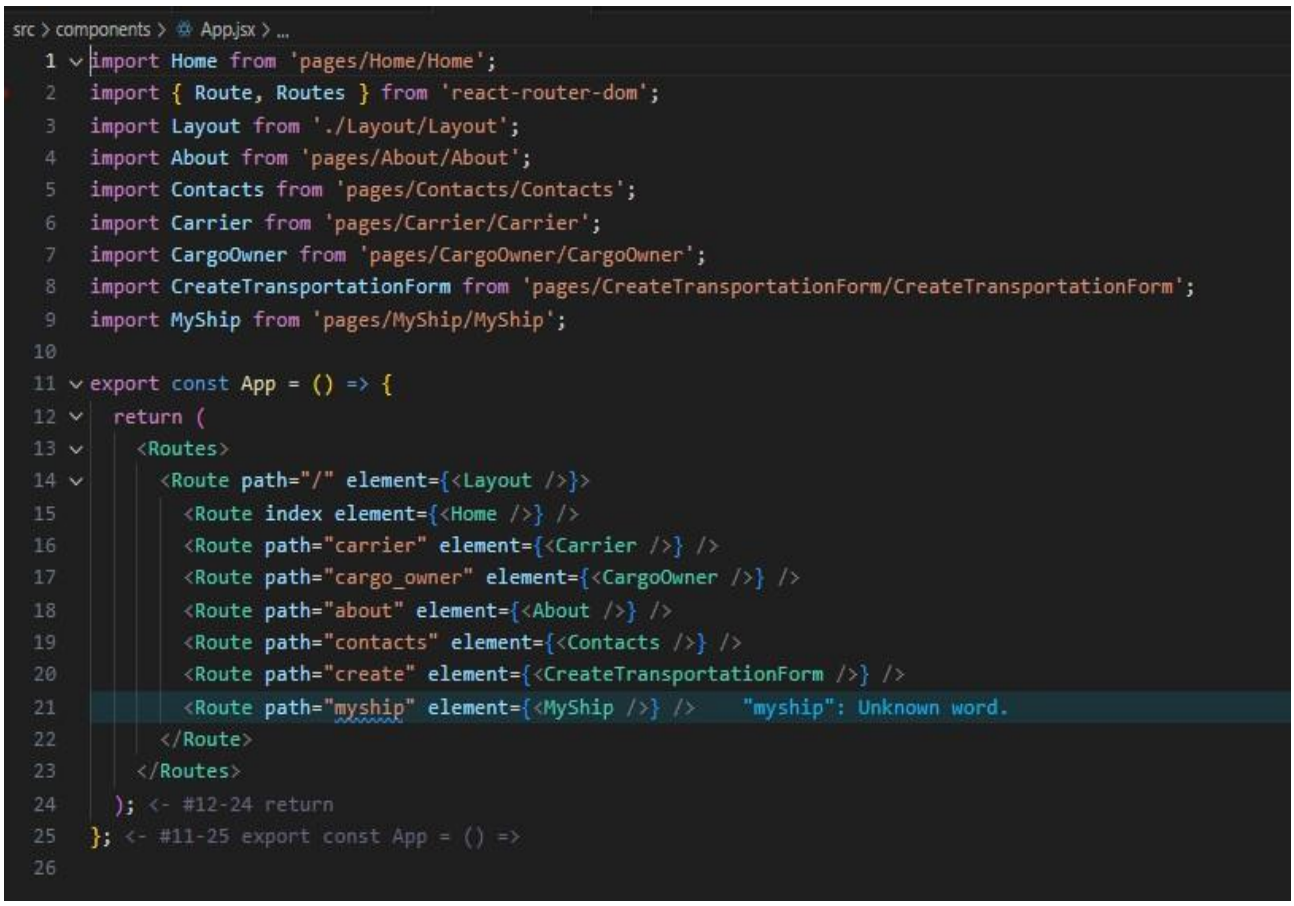
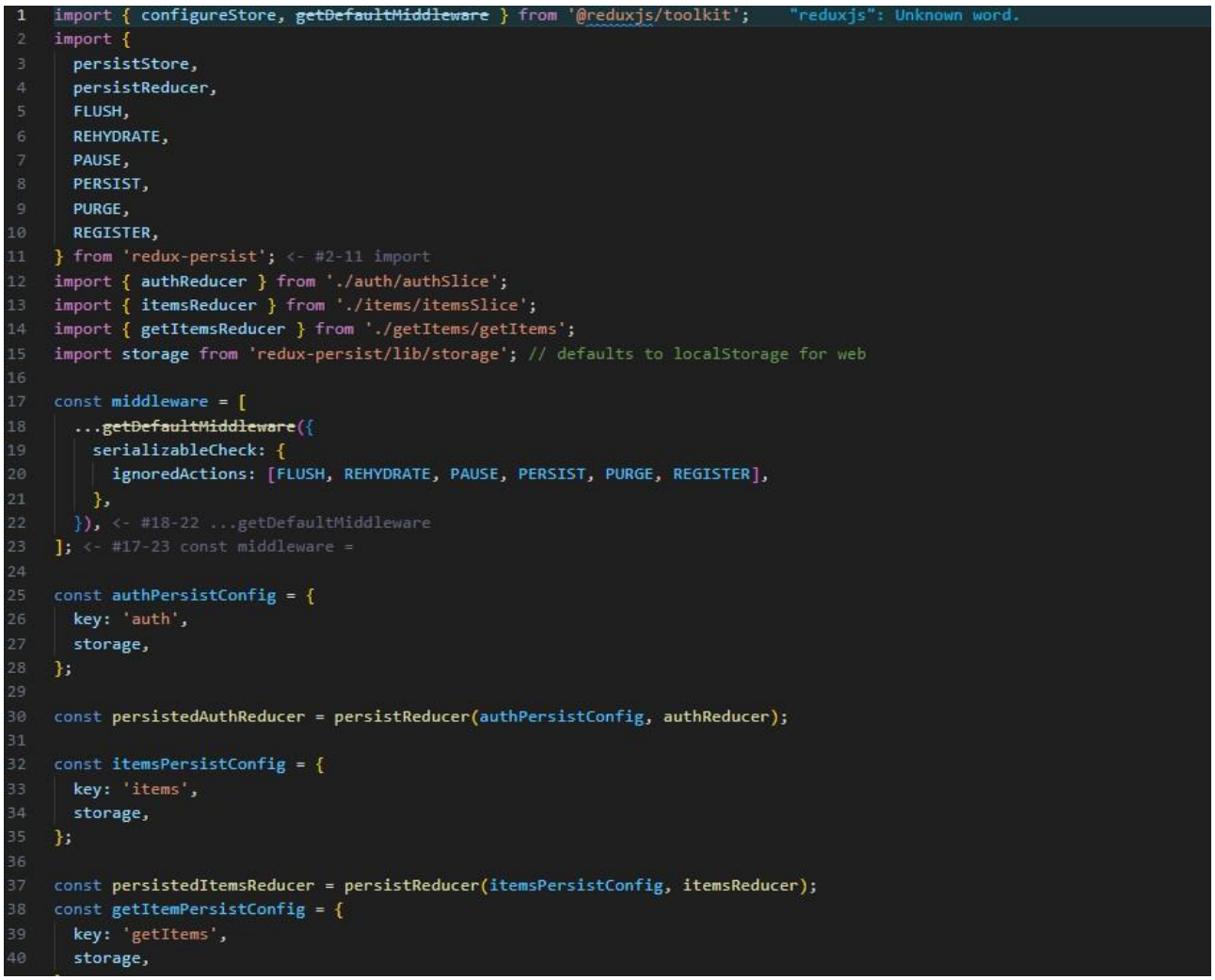


Рис. 3.4 React Router

Redux: Redux є популярною бібліотекою для керування станом додатка в React. Він використовує архітектуру Flux та допомагає організувати та керувати глобальним станом додатка. Redux використовує одне головне хранилище (store), в якому зберігається весь стан додатка. Компоненти можуть звертатись до цього хранилища для отримання стану та відправляти дії (actions), які змінюють стан. Redux спрощує керування станом, забезпечує прогнозовану поведінку та полегшує відлагодження[див. рис. 3.5].



```
import { configureStore, getDefaultMiddleware } from '@reduxjs/toolkit';    "reduxjs": Unknown word.
import {
  persistStore,
  persistReducer,
  FLUSH,
  REHYDRATE,
  PAUSE,
  PERSIST,
  PURGE,
  REGISTER,
} from 'redux-persist'; <- #2-11 import
import { authReducer } from './auth/authSlice';
import { itemsReducer } from './items/itemsSlice';
import { getItemsReducer } from './getItems/getItems';
import storage from 'redux-persist/lib/storage'; // defaults to localStorage for web

const middleware = [
  ...getDefaultMiddleware({
    serializableCheck: {
      ignoredActions: [FLUSH, REHYDRATE, PAUSE, PERSIST, PURGE, REGISTER],
    },
  }), <- #18-22 ...getDefaultMiddleware
]; <- #17-23 const middleware =

const authPersistConfig = {
  key: 'auth',
  storage,
};

const persistedAuthReducer = persistReducer(authPersistConfig, authReducer);

const itemsPersistConfig = {
  key: 'items',
  storage,
};

const persistedItemsReducer = persistReducer(itemsPersistConfig, itemsReducer);
const getItemPersistConfig = {
  key: 'getItems',
  storage,
```

Рис. 3.5 Redux

Axios: Axios є бібліотекою для виконання HTTP-запитів з React-додатком. Вона надає простий та зрозумілий інтерфейс для виконання запитів до сервера або зовнішніх API. Axios підтримує різні методи запитів, такі як GET, POST, PUT, DELETE, та дозволяє налаштовувати заголовки та передавати дані. Використання Axios полегшує взаємодію з сервером та обробку результатів запитів[див. рис. 3.6].

```
const handleOriginCityChange = async event => {
  const value = event.target.value;
  if (value === '') {
    setOriginCoordinates(null);
  }
  setOriginValue(value);
  try {
    const response = await axios.get(
      `https://api.opencagedata.com/geocode/v1/json?q=${encodeURIComponent(
        value
      )}&key=4b6e7d31f0654074aa698fd64a45063c`
    ); <- #67-71 const response = await axios.get
    const suggestions = response.data.results;
    setSuggestedOrigins(suggestions);
  } catch (error) {
    console.log('Помилка запиту геокодування:', error.message);    "Помилка": Unknown word.
    setSuggestedOrigins([]);
  }
}; <- #60-78 const handleOriginCityChange = async event =>
```

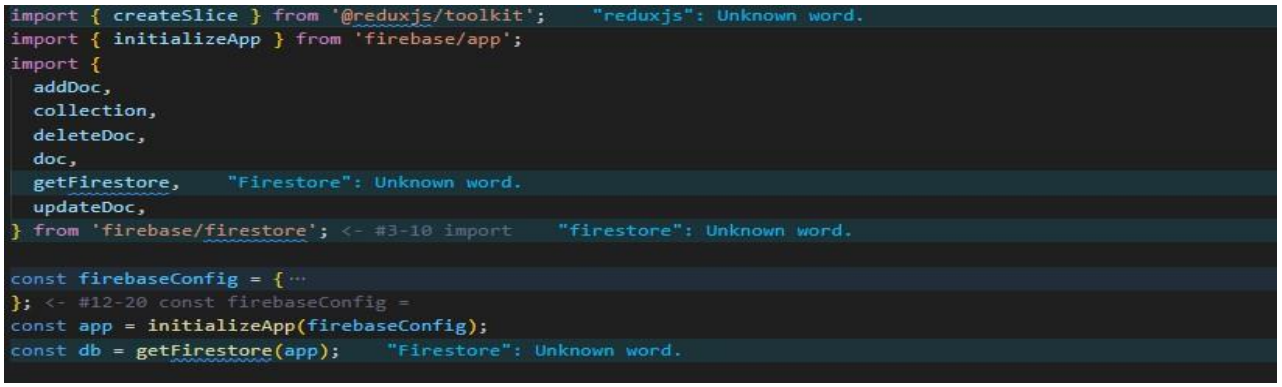
Рис. 3.6 Axios

Styled Components: Styled Components є бібліотекою, яка дозволяє стилізувати React-компоненти з використанням CSS-in-JS підходу. Замість використання зовнішніх CSS-файлів, Styled Components дозволяє вбудовувати стилі безпосередньо в код компонента. Вона пропонує зручну синтаксичну конструкцію для опису стилів та автоматично генерує унікальні CSS-класи для кожного компонента. Styled Components забезпечує легку підтримку динамічних стилів, тем та анімацій[див. рис. 3.7].

```
import styled from 'styled-components';
import SideImage from '../../images/SideBar.jpg';
import { NavLink } from 'react-router-dom';

const Container = styled.div`
  display: flex;
  flex-wrap: no-wrap;
`;

const Sidebar = styled.div`
  width: 300px;
  background-image: url(${SideImage});
  background-repeat: no-repeat;
  background-size: cover;
  background-position: 0% 50%;

  height: 98vh;
  display: flex;
  flex-direction: column;
`;
```

Рис. 3.7 Styled Components

Firebase: Firebase є платформою для розробки веб- та мобільних додатків. Вона надає набір інструментів та сервісів, які допомагають зберігати та синхронізувати дані в реальному часі, автентифікувати користувачів, реалізовувати повідомлення в режимі реального часу та багато іншого. Firebase має SDK для JavaScript, який дозволяє легко інтегрувати його функціональність в React-додатки. Використання Firebase полегшує розробку багатьох аспектів веб-сайту, таких як збереження даних, аутентифікація користувачів та взаємодія з сервером[див. рис. 3.8, 3.9, 3.10].



```
import { createSlice } from '@reduxjs/toolkit';    "reduxjs": Unknown word.
import { initializeApp } from 'firebase/app';
import {
  addDoc,
  collection,
  deleteDoc,
  doc,
  getFirestore,    "Firestore": Unknown word.
  updateDoc,
} from 'firebase/firestore'; <- #3-10 import    "firestore": Unknown word.

const firebaseConfig = {···
}; <- #12-20 const firebaseConfig =
const app = initializeApp(firebaseConfig);
const db = getFirestore(app);    "Firestore": Unknown word.

const itemsSlice = createSlice({
  name: 'items',
  initialState: { items: [] },
  reducers: {
    addItem: async (state, action) => {
      const newItem = action.payload;
      try {
        const docRef = await addDoc(collection(db, 'items'), newItem);
        newItem.id = docRef.id;
        // state.items.push(newItem);
      } catch (error) {
        console.error('Error adding item:', error);
      }
    }, <- #28-37 addItem: async (state, action) =>
    removeItem: async (state, action) => {
      const itemId = action.payload;

      try {
        await deleteDoc(doc(db, 'items', itemId.id));

        // state = state.items.filter(item => item.id !== itemId.id);
      } catch (error) {
        console.error('Error removing item:', error);
      }
    }, <- #38-48 removeItem: async (state, action) =>
    updateItem: async (state, action) => {
      const updatedItem = action.payload;
      try {
        await updateDoc(doc(db, 'items', updatedItem.id), updatedItem);
        // const index = state.items.findIndex(item => item.id === updatedItem.id);

        // if (index !== -1) {
        //   state.items[index] = updatedItem;
        // }
      } catch (error) {
```

Рис. 3.8 Firebase

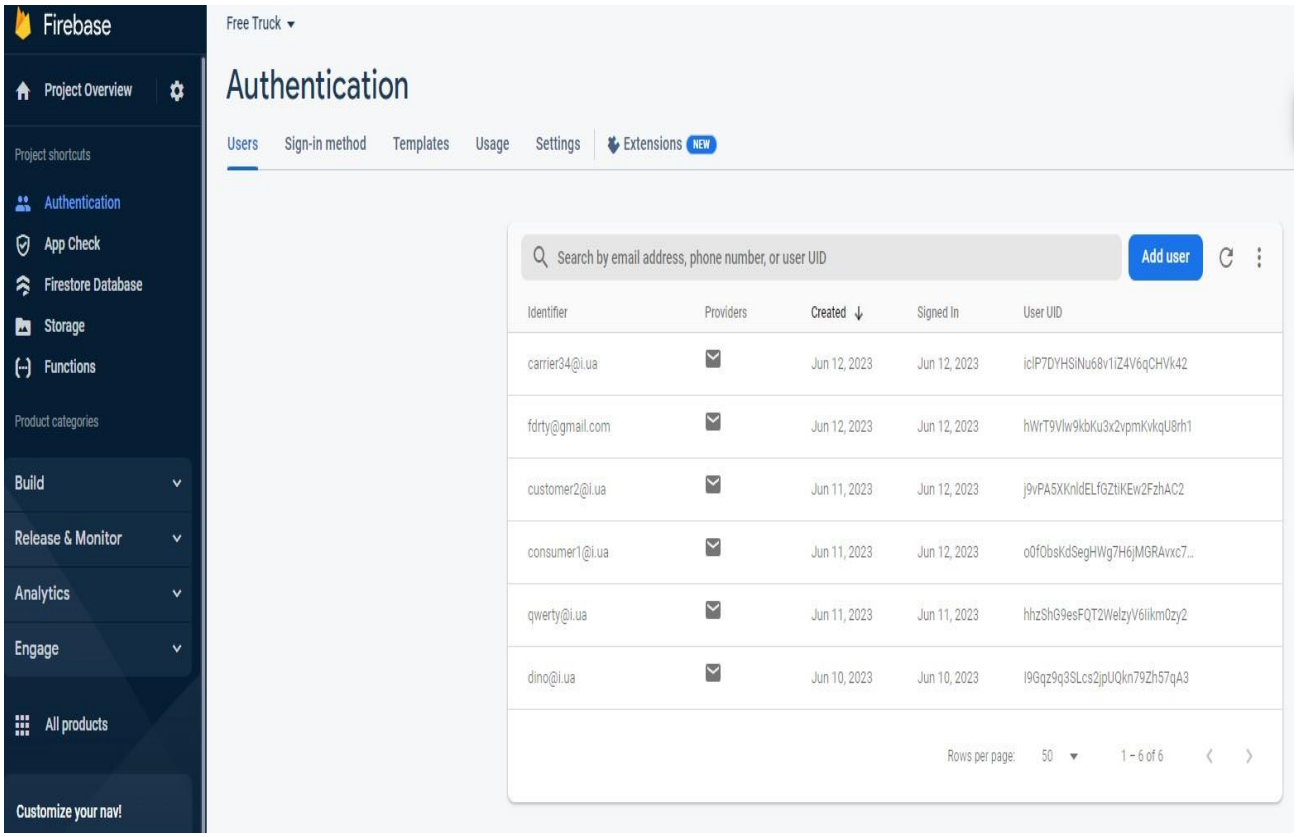


Рис. 3.9 Firebase Аутентифікація користувачів

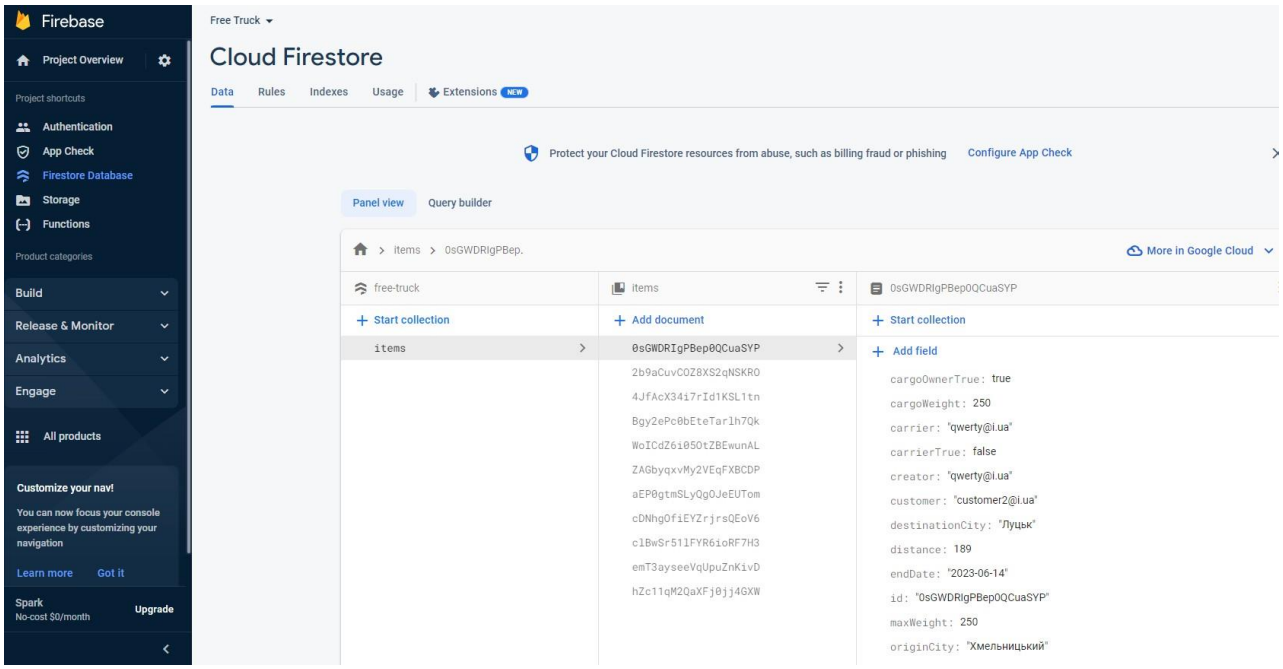


Рис. 3.10 Firebase Збереження даних на сервері

Ці бібліотеки та інструменти полегшують процес створення веб-сайтів на React та надають багато корисних можливостей. Використання цих інструментів

дозволяє писати чистий, ефективний та легко керований код, який забезпечує більшу швидкість розробки та підтримку веб-додатків.

Однією з покладених мною задач була розробка унікального компонента для автентифікації на сайті. Даний компонент розроблений на основі бібліотек React та Redux і містить набір функцій для управління процесом автентифікації користувачів.

Основні функціональні можливості компонента включають:

Відображення форми для введення електронної пошти та пароля.

Можливість реєстрації нового користувача.

Можливість входу в систему за допомогою введених даних.

Підтримка автентифікації через обліковий запис Google.

Відображення повідомлень про помилки або успішні операції.

Відображення інформації про користувача та кнопки для виходу з системи.

Можливість переключення між формою автентифікації та іншими режимами.

Компонент є гнучким та може бути використаний на будь-якій сторінці сайту, де необхідна функціональність автентифікації користувачів. Він пропонує зручність використання та надає можливість взаємодіяти з різними типами користувачів, надаючи їм можливість зареєструватися, увійти в систему або вийти з неї, використовуючи різні способи автентифікації.

Компонент є унікальним завдяки своїй реалізації на основі сучасних технологій та використання найкращих практик веб-розробки. Він може бути використаний як один з ключових елементів вашого сайту, додаючи до нього потужну функціональність автентифікації та забезпечуючи безпеку та захист конфіденційної інформації користувачів. [див. рис. 3.11, 3.12].

```
const AuthForm = () => {
  const navigate = useNavigate();
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const showAuthForm = useSelector(state => state.auth.showAuthForm);
  const loading = useSelector(state => state.auth.loading);
  const error = useSelector(state => state.auth.error);
  const successMessage = useSelector(state => state.auth.successMessage);
  const dispatch = useDispatch();

  const handleToggleAuthForm = () => {
    dispatch(setShowAuthForm(!showAuthForm));
  };

  const handleRegister = async e => {
    e.preventDefault();
    dispatch(setLoading(true));
    dispatch(setError(null));
    try {
      await dispatch(register(email, password));
    } catch (error) {
      dispatch(setError(error.message));
    }

    dispatch(setLoading(false));
  }; <- #45-56 const handleRegister = async e =>

  const handleLogin = async e => {
    e.preventDefault();
    dispatch(setLoading(true));
    dispatch(setError(null));

    try {
      await dispatch(login(email, password));
    } catch (error) {
      dispatch(setError(error.message));
    }

    dispatch(setLoading(false));
  }; <- #58-70 const handleLogin = async e =>

  const handleGoogleLogin = async () => {
    try {
      dispatch(setError(null));
      dispatch(setLoading(true));
      await dispatch(loginWithGoogle());
    } catch (error) {
      console.log('Google Login error:', error);
    }
```

Рис. 3.11 Функції роботи з аутентифікацією

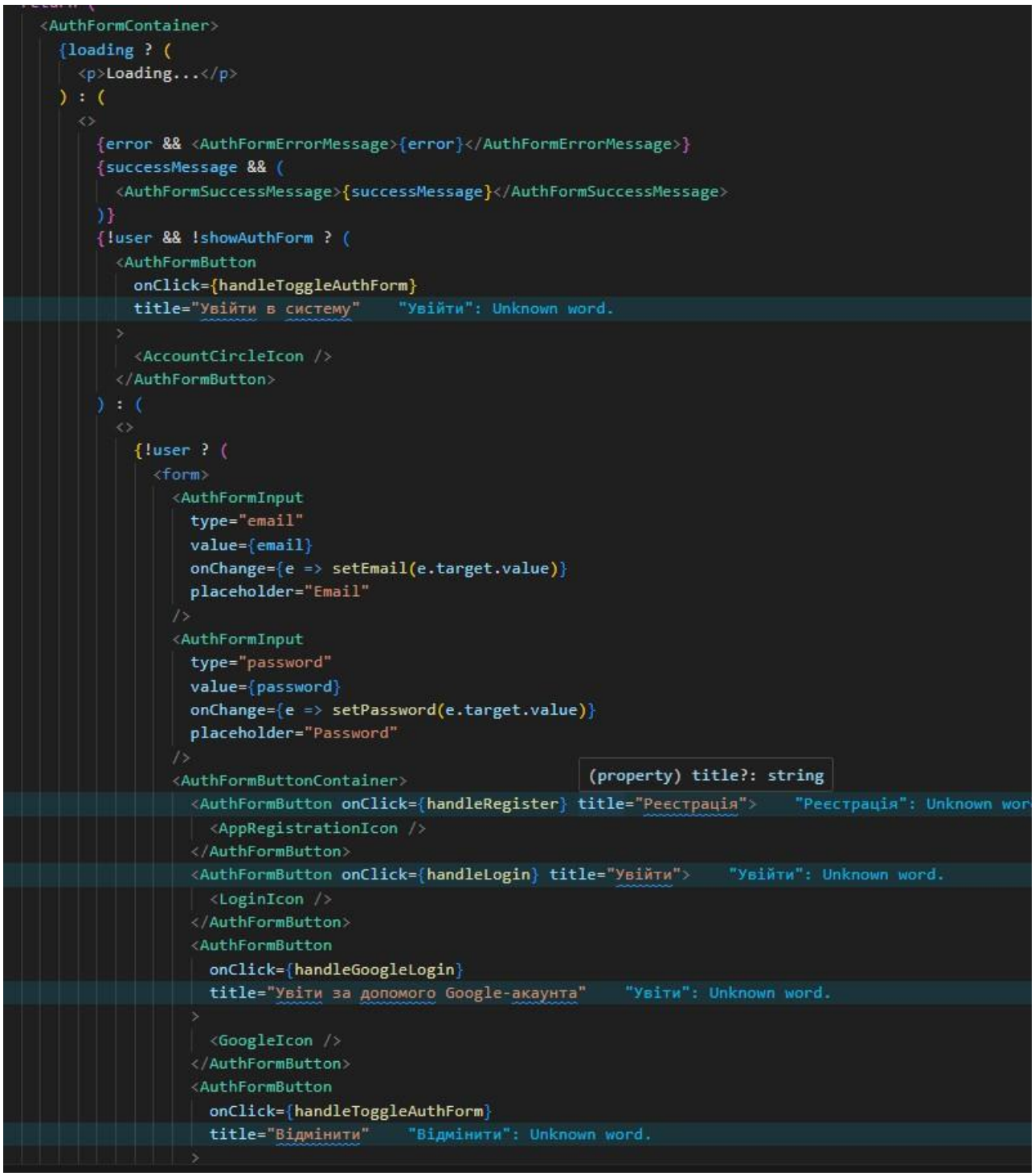


Рис. 3.12 Код відображення форми аутентифікації

Також для автентифікації та збереження даних була використана бібліотека Redux-Toolkit, яка є популярним інструментом для керування станом додатків на основі Redux в середовищі React.

Redux-Toolkit надає простий та зручний спосіб організації стану додатку, забезпечуючи централізоване збереження даних та простий доступ до них з будь-якого місця в додатку. Вона поєднує у собі декілька інших бібліотек Redux, таких як

Redux-Thunk та Redux-Saga, що дозволяє працювати з асинхронними діями та ефектами.

Основні переваги використання Redux-Toolkit включають:

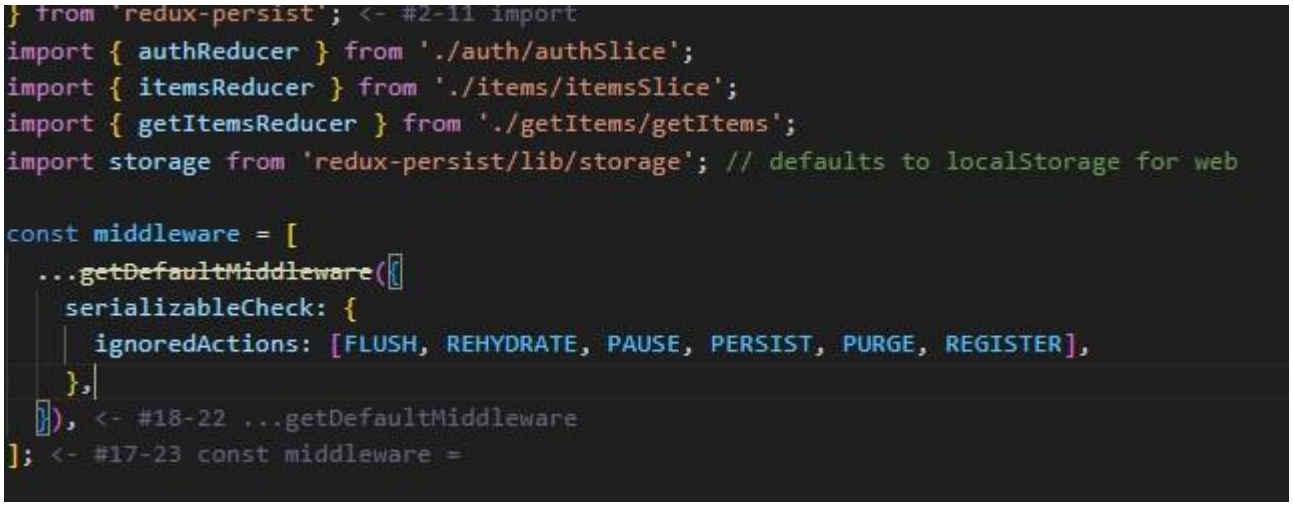
Спрощений синтаксис: Redux-Toolkit надає спрощений синтаксис для оголошення дій, редюсерів та створення централізованого стору. Це дозволяє зменшити кількість коду, необхідного для роботи з Redux.

Вбудована підтримка асинхронності: Redux-Toolkit має вбудовану підтримку для виконання асинхронних операцій за допомогою Redux-Thunk або Redux-Saga. Це дозволяє легко виконувати запити до Firebase API або інших зовнішніх сервісів.

Інструменти розробника: Redux-Toolkit надає набір корисних інструментів для розробника, таких як Redux DevTools, які спрощують налагодження та відлагодження додатку, а також надають інформацію про зміни стану додатку у реальному часі.

Модульність: Redux-Toolkit дозволяє організувати код додатку в модулі, що полегшує його розширення та підтримку. Кожен модуль може містити свій власний набір дій, редюсерів та селекторів, що полегшує розподіл роботи між розробниками та підтримку масштабованих додатків.

Завдяки використанню Redux-Toolkit, автентифікаційний компонент може ефективно керувати станом автентифікації, зберігати дані користувачів та взаємодіяти з Firebase для автентифікації та збереження даних. Це дозволяє зробити додаток більш масштабованим, керованим та ефективним з точки зору керування станом [див. рис. 3.13-3.16].



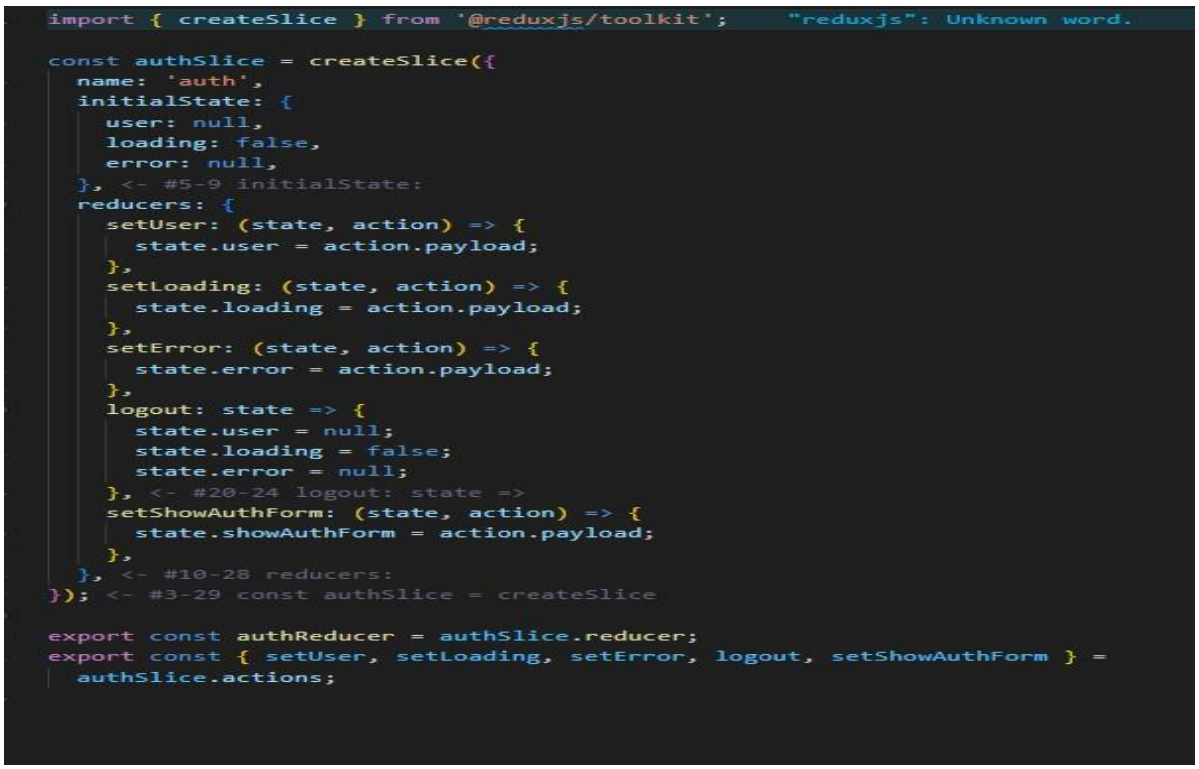
```
const authPersistConfig = {
  key: 'auth',
  storage,
};

const persistedAuthReducer = persistReducer(authPersistConfig, authReducer);

const itemsPersistConfig = {
  key: 'items',
  storage,
};

const persistedItemsReducer = persistReducer(itemsPersistConfig, itemsReducer);
const getItemPersistConfig = {
  key: 'getItems',
  storage,
};

const persistedGetItemReducer = persistReducer(
  getItemPersistConfig,
  getItemsReducer
);

export const store = configureStore({
  reducer: {
    auth: persistedAuthReducer,
    items: persistedItemsReducer,
    getItems: persistedGetItemReducer,
  }, <- #49-53 reducer:
  middleware,
  devTools: process.env.NODE_ENV === 'development',
}); <- #48-56 export const store = configureStore

export const persistor = persistStore(store);    "persistor": Unknown word.
```


```
export const store = configureStore({
  reducer: {
    auth: persistedAuthReducer,
    items: persistedItemsReducer,
    getItems: persistedGetItemReducer,
  }, <- #49-53 reducer:
  middleware,
  devTools: process.env.NODE_ENV === 'development',
}); <- #48-56 export const store = configureStore

export const persistor = persistStore(store);    "persistor": Unknown word.
```


Рис. 3.13 Головний «Стор» сайту

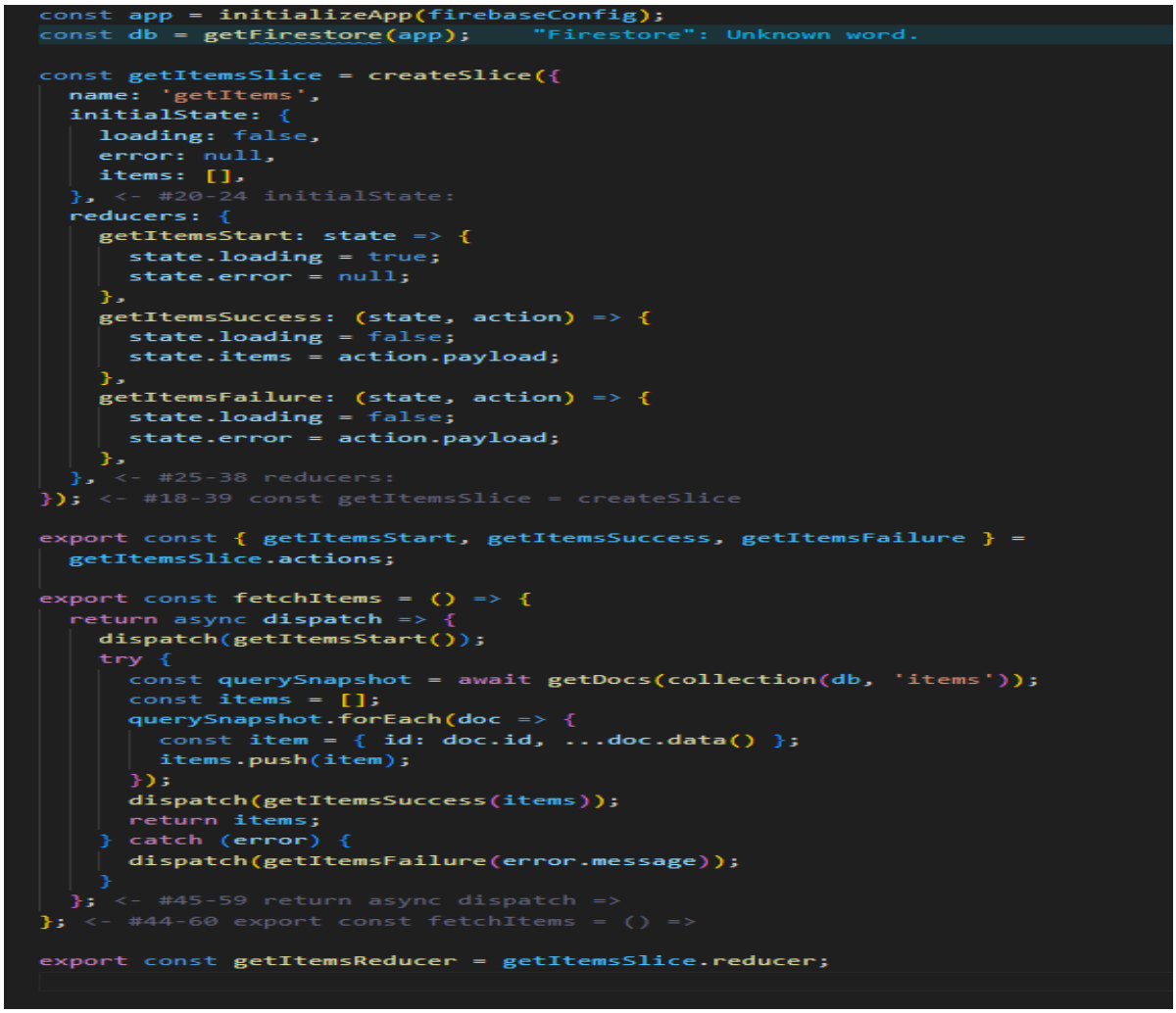
```
const itemsSlice = createSlice({
  name: 'items',
  initialState: { items: [] },
  reducers: {
    addItem: async (state, action) => {
      const newItem = action.payload;
      try {
        const docRef = await addDoc(collection(db, 'items'), newItem);
        newItem.id = docRef.id;
        // state.items.push(newItem);
      } catch (error) {
        console.error('Error adding item:', error);
      }
    }, <- #28-37 addItem: async (state, action) =>
    removeItem: async (state, action) => {
      const itemId = action.payload;

      try {
        await deleteDoc(doc(db, 'items', itemId.id));

        // state = state.items.filter(item => item.id !== itemId.id);
      } catch (error) {
        console.error('Error removing item:', error);
      }
    }, <- #38-48 removeItem: async (state, action) =>
    updateItem: async (state, action) => {
      const updatedItem = action.payload;
      try {
        await updateDoc(doc(db, 'items', updatedItem.id), updatedItem);
        // const index = state.items.findIndex(item => item.id === updatedItem

        // if (index !== -1) {
        //   state.items[index] = updatedItem;
        // }
      } catch (error) {
        console.error('Error updating item:', error);
      }
    }, <- #49-61 updateItem: async (state, action) =>
  }, <- #27-62 reducers:
}); <- #24-63 const itemsSlice = createSlice

export const { addItem, removeItem, updateItem } = itemsSlice.actions;
export const itemsReducer = itemsSlice.reducer;
```

Рис. 3.14 «Редюсери» на додавання, видалення та обновлення даних у Firestore



```
import { createSlice } from '@reduxjs/toolkit';     "reduxjs": Unknown word.

const authSlice = createSlice({
  name: 'auth',
  initialState: {
    user: null,
    loading: false,
    error: null,
  }, <- #5-9 initialState:
  reducers: {
    setUser: (state, action) => {
      state.user = action.payload;
    },
    setLoading: (state, action) => {
      state.loading = action.payload;
    },
    setError: (state, action) => {
      state.error = action.payload;
    },
    logout: state => {
      state.user = null;
      state.loading = false;
      state.error = null;
    }, <- #20-24 logout: state =>
    setShowAuthForm: (state, action) => {
      state.showAuthForm = action.payload;
    },
  }, <- #10-28 reducers:
}); <- #3-29 const authSlice = createSlice

export const authReducer = authSlice.reducer;
export const { setUser, setLoading, setError, logout, setShowAuthForm } =
  authSlice.actions;
```

Рис. 3.15 «Редюсери» операцій аутентифікації



```
const app = initializeApp(firebaseConfig);
const db = getFirestore(app);     "Firestore": Unknown word.

const getItemsSlice = createSlice({
  name: 'getItems',
  initialState: {
    loading: false,
    error: null,
    items: [],
  }, <- #20-24 initialState:
  reducers: {
    getItemsStart: state => {
      state.loading = true;
      state.error = null;
    },
    getItemsSuccess: (state, action) => {
      state.loading = false;
      state.items = action.payload;
    },
    getItemsFailure: (state, action) => {
      state.loading = false;
      state.error = action.payload;
    },
  }, <- #25-38 reducers:
}); <- #18-39 const getItemsSlice = createSlice

export const { getItemsStart, getItemsSuccess, getItemsFailure } =
  getItemsSlice.actions;

export const fetchItems = () => {
  return async dispatch => {
    dispatch(getItemsStart());
    try {
      const querySnapshot = await getDocs(collection(db, 'items'));
      const items = [];
      querySnapshot.forEach(doc => {
        const item = { id: doc.id, ...doc.data() };
        items.push(item);
      });
      dispatch(getItemsSuccess(items));
      return items;
    } catch (error) {
      dispatch(getItemsFailure(error.message));
    }
  }; <- #45-59 return async dispatch =>
}; <- #44-60 export const fetchItems = () =>

export const getItemsReducer = getItemsSlice.reducer;
```

Рис. 3.16 «Редюсери» на вибір даних з Firestore

На сайті розроблено компонент CreateTransportationForm який відповідає за створення форми для створення нового перевезення. Основна функція компонента - збирати інформацію від користувача, необхідну для створення перевезення, та відправляти цю інформацію на сервер.

Основні функції та можливості компонента:

Використання хуків useState та useEffect для збереження та оновлення стану компонента.

Використання хука useSelector для отримання стану з Redux Store.

Використання хука useDispatch для диспетчеризації дій у Redux Store.

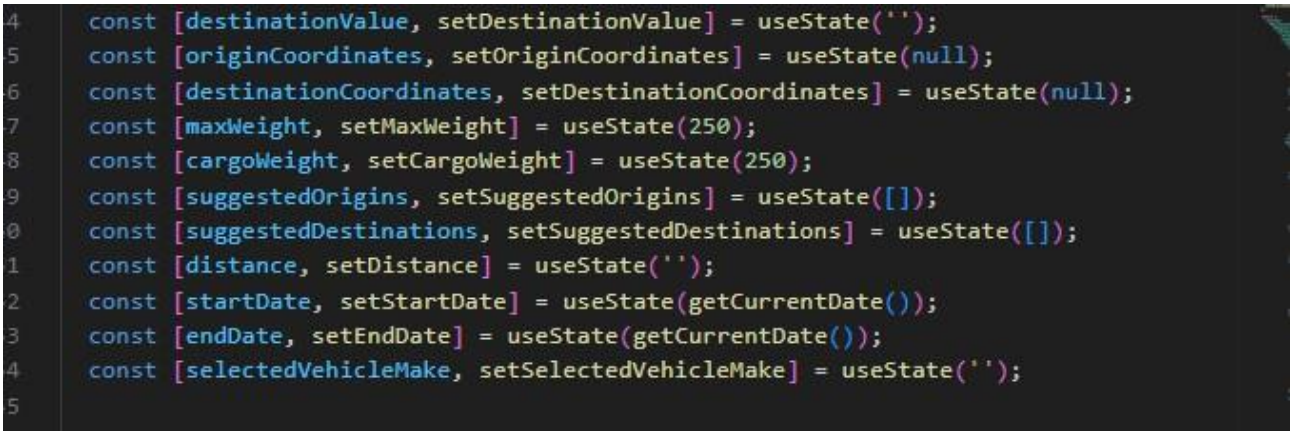
Форма містить поля для введення міста походження, міста призначення, максимальної ваги перевезення, ваги вантажу, початкової та кінцевої дат перевезення.

Поля міста походження та призначення мають функцію автодоповнення зі списку запропонованих міст.

При виборі міста з автодоповнення, отримуються координати міста та відображається відстань між містами.

Кнопка "Submit" відправляє дані форми на сервер та додає нове перевезення до списку перевезень. [див. рис. 3.17].

```
import { useDispatch, useSelector } from 'react-redux';
import { addItem } from '../../store/items/itemsSlice';
const CreateTransportationForm = () => {
  const user = useSelector(state => state.auth.user);
  const dispatch = useDispatch();
  const getCurrentDate = () => {
    const today = new Date();
    const year = today.getFullYear();
    let month = today.getMonth() + 1;
    let day = today.getDate();

    if (month < 10) {···
    }
    if (day < 10) {···
    }

    return `${year}-${month}-${day}`;
  }; <- #25-39 const getCurrentDate = () =>
  const [isActive, setIsActive] = useState(true);
  const [originCity, setOriginCity] = useState(null);
  const [destinationCity, setDestinationCity] = useState(null);
  const [originValue, setOriginValue] = useState('');
```



```
  const [destinationValue, setDestinationValue] = useState('');
  const [originCoordinates, setOriginCoordinates] = useState(null);
  const [destinationCoordinates, setDestinationCoordinates] = useState(null);
  const [maxWeight, setMaxWeight] = useState(250);
  const [cargoWeight, setCargoWeight] = useState(250);
  const [suggestedOrigins, setSuggestedOrigins] = useState([]);
  const [suggestedDestinations, setSuggestedDestinations] = useState([]);
  const [distance, setDistance] = useState('');
  const [startDate, setStartDate] = useState(getCurrentDate());
  const [endDate, setEndDate] = useState(getCurrentDate());
  const [selectedVehicleMake, setSelectedVehicleMake] = useState('');

```

Рис. 3.17 Частина компоненту CreateTransportationForm

Компонент TransportationList, який відповідає за відображення списку перевезень. Основна функція компонента - відображення інформації про кожне перевезення з переданого масиву transportations та надання можливостей для взаємодії з кожним елементом списку. Також даний компонент має логіку різноманітного реагування на сторінки з якої здійснено рендер компонента. Основні елементи та їх функціональність у компоненті:

import - імпорт необхідних залежностей та стилів для компонента.

TransportationList - головна функціональна компонента, яка отримує transportations (список перевезень) та onButton (прапорець для відображення певних кнопок) як параметри.

user - змінна, яка отримує інформацію про користувача зі стану Redux Store за допомогою хука useSelector.

dispatch - змінна, яка отримує функцію диспетчеризації дій Redux Store за допомогою хука useDispatch.

handleChangeCarrier, handleChangeCostumer, handleChangeCarrierTrue, handleChangeCostumerTrue, handleCancelTrue, handleDelete - функції, які виконують різні дії (оновлення, видалення) залежно від взаємодії з елементами списку перевезень.

return - повертає JSX-розмітку для відображення списку перевезень. Кожен елемент списку представлений компонентом TransportationItem, який містить інформацію про перевезення та різні кнопки для взаємодії.

Компонент TransportationItem має різні стилі, які залежать від певних умов (наприклад, кольори фону залежно від статусу перевезення). Він містить інформацію про номер замовлення, вагу вантажу, вантажовідправника, перевізника та початковий та кінцевий пункти маршруту.

Кнопки, які надаються для взаємодії з елементами списку, мають різні дії, такі як підтвердження перевезення, відмова від перевезення та видалення.

```
const TransportationList = ({ transportations, onButton }) => {    "transportations":
  const user = useSelector(state => state.auth.user);
  const dispatch = useDispatch();

  const handleChangeCarrier = transportations => {    "transportations": Unknown word
    const updatedItems = {
      id: transportations.id,    "transportations": Unknown word.
      ...transportations,    "transportations": Unknown word.
      carrier: user.displayName,
      carrierTrue: true,
    }; <- #23-28 const updatedItems =
    dispatch(updateItem(updatedItems));
    setTimeout(() => {
      // Оновлення сторінки    "Оновлення": Unknown word.
      window.location.reload();
    }, 2000);
  }; <- #22-34 const handleChangeCarrier = transportations =>
  const handleChangeCostumer = transportations => {    "transportations": Unknown wor
    const updatedItems = {
      id: transportations.id,
      ...transportations,
      customer: user.displayName,
      cargoOwnerTrue: true,
      cargoWeight: transportations.maxWeight,
    }; <- #36-42 const updatedItems =
    dispatch(updateItem(updatedItems));
    setTimeout(() => {
      // Оновлення сторінки    "Оновлення": Unknown word.
      window.location.reload();
    }, 2000);
```

Рис. 3.18 Фрагмент коду компоненту CreateTransportationForm

3.2. Тестування веб-сайту компанії «Free Truck»

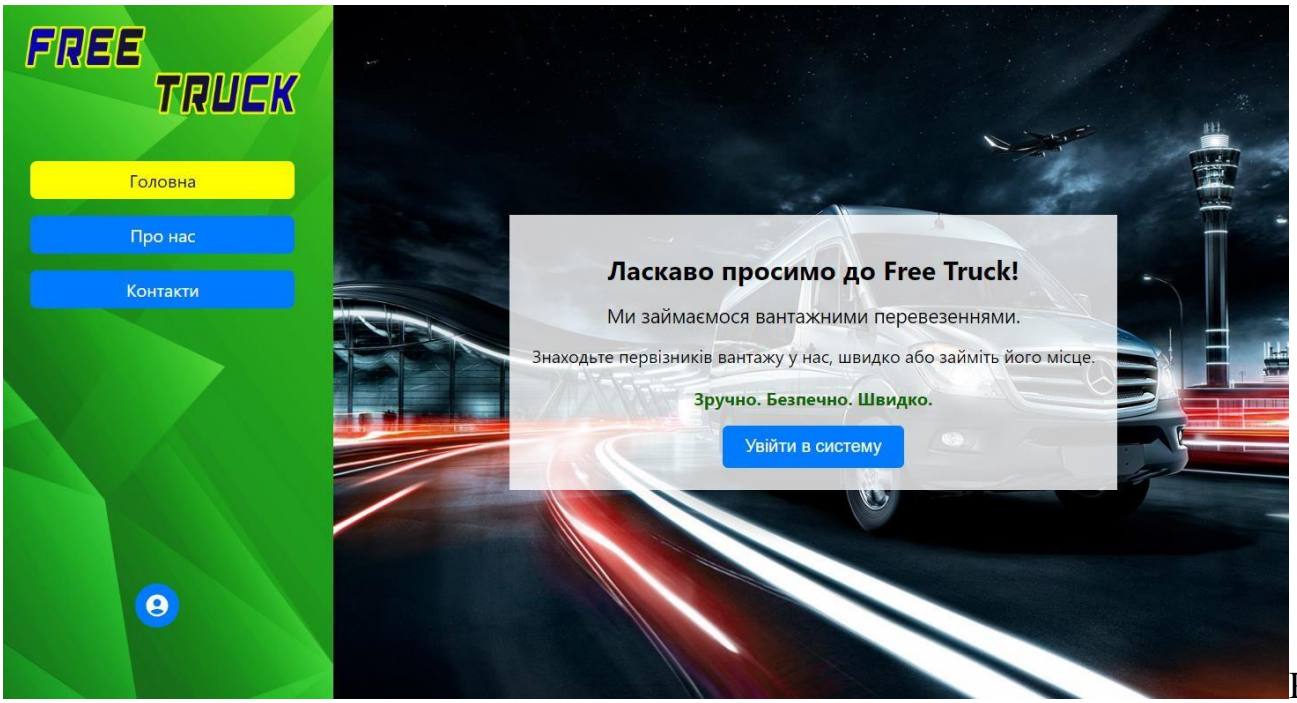


Рис. 3.19 Головна сторінка

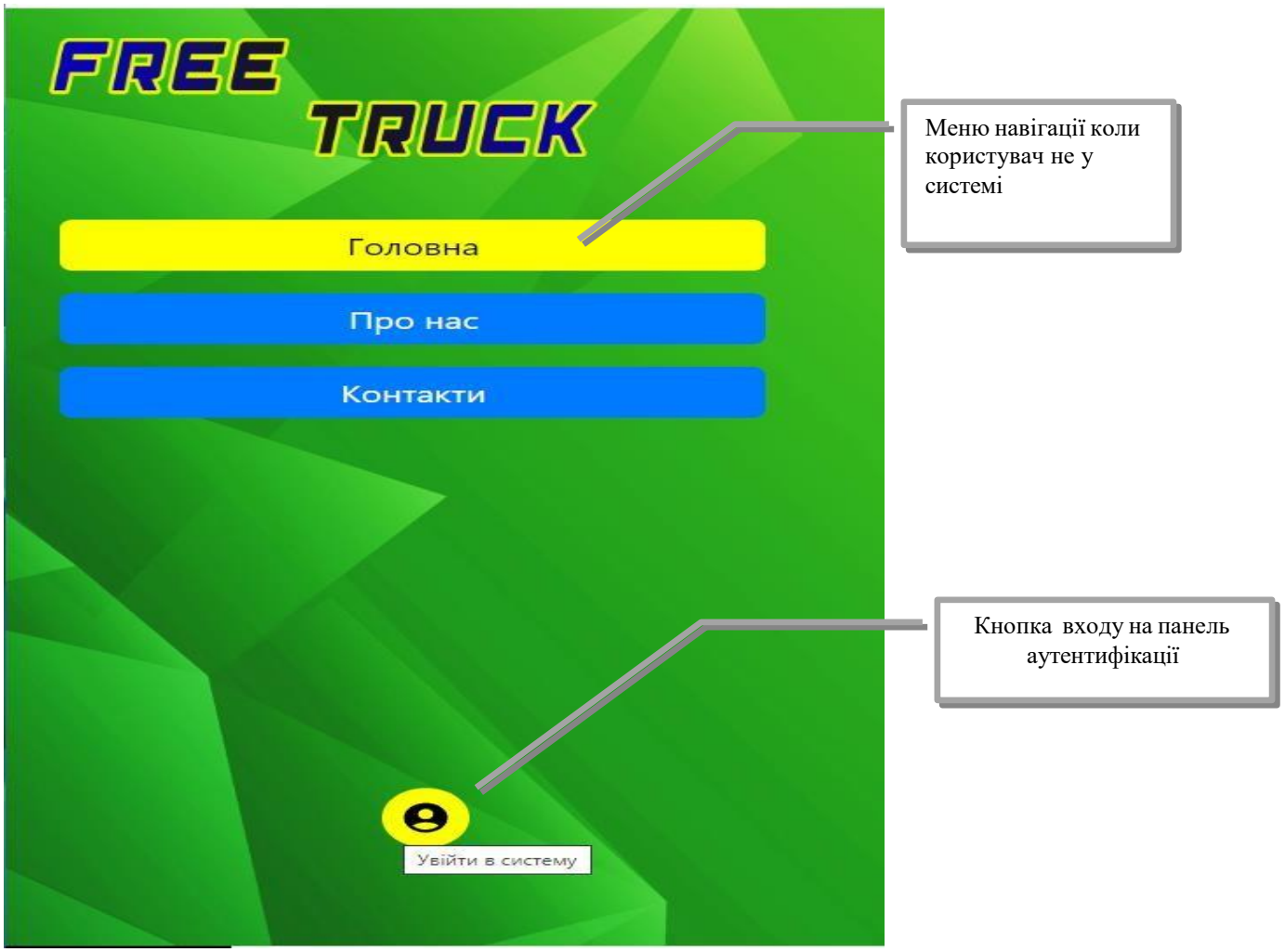


Рис. 3.20. Бічна панель керування з навігацією та кнопкою аутентифікації.

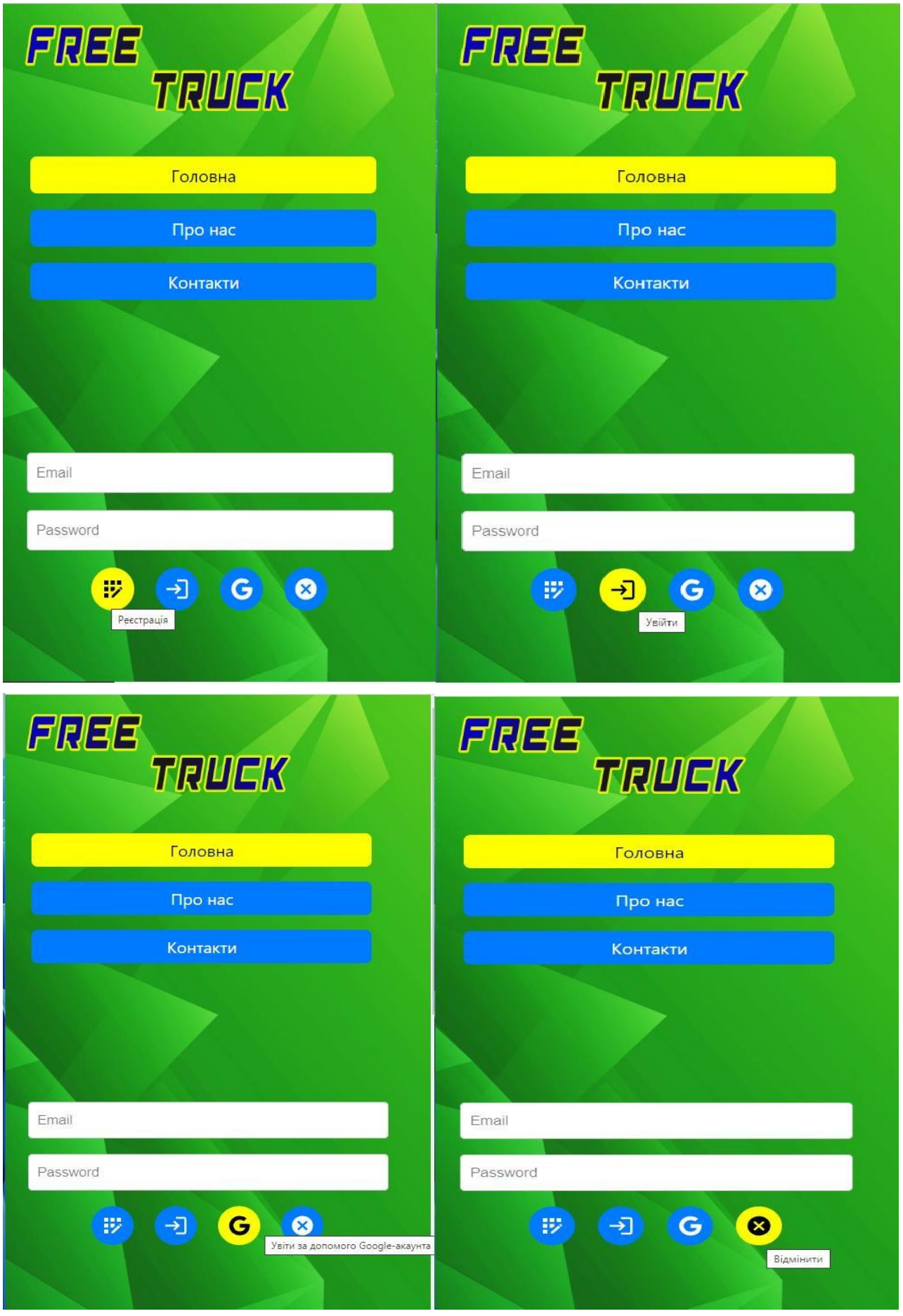


Рис. 3.21 Елементи керування панелі аутентифікації .

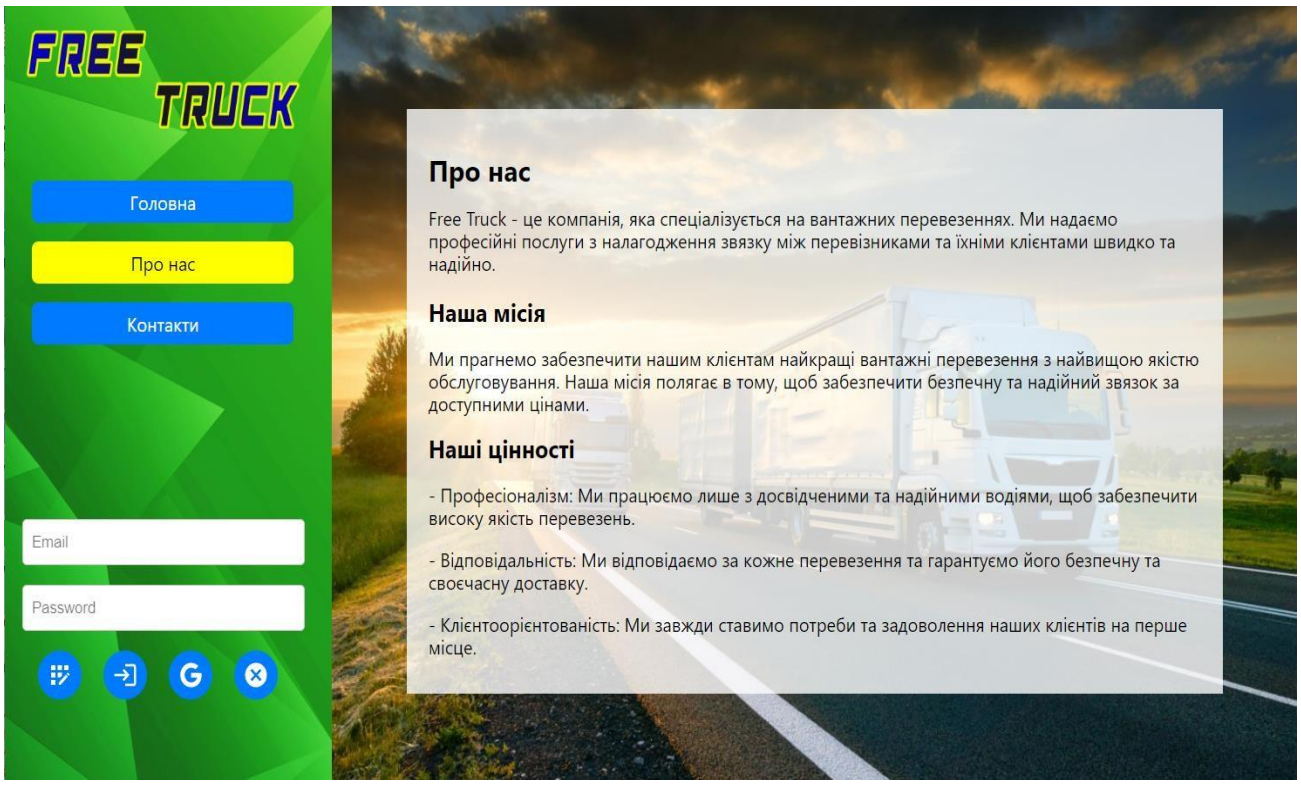


Рис. 3.22. Сторінка «Про нас».

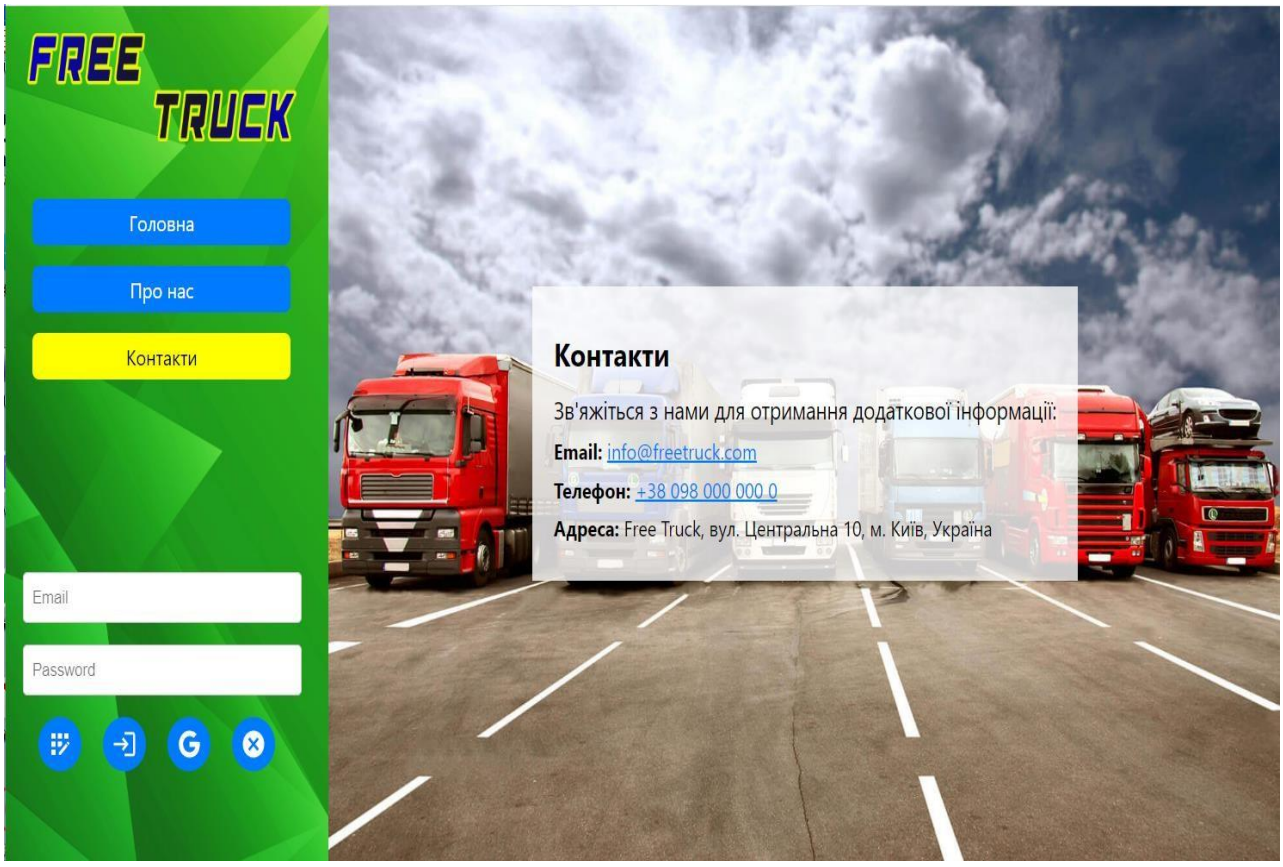


Рис. 3.23. Сторінка «Контакти».

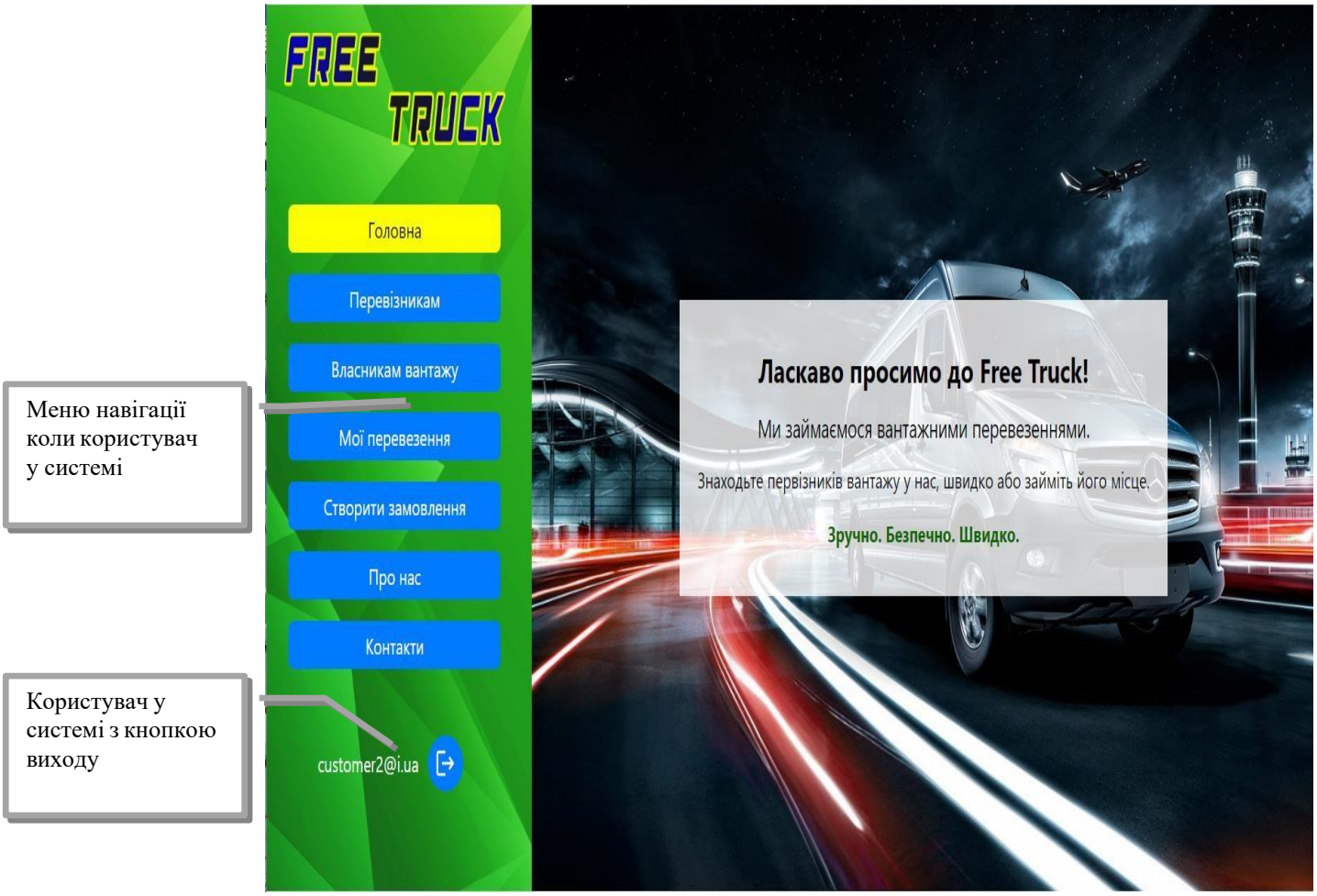


Рис. 3.24. Розширене Навігаційне меню коли користувач увійшов в систему

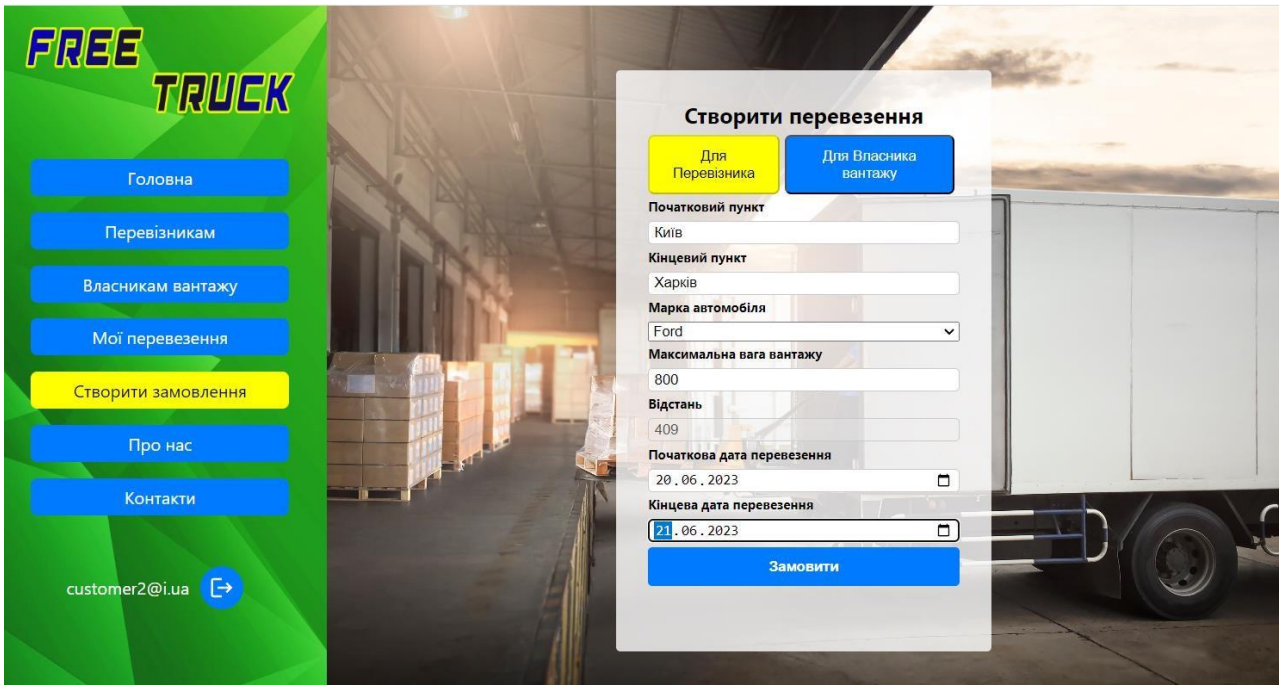


Рис. 3.25. Меню створення перевезення для перевізника

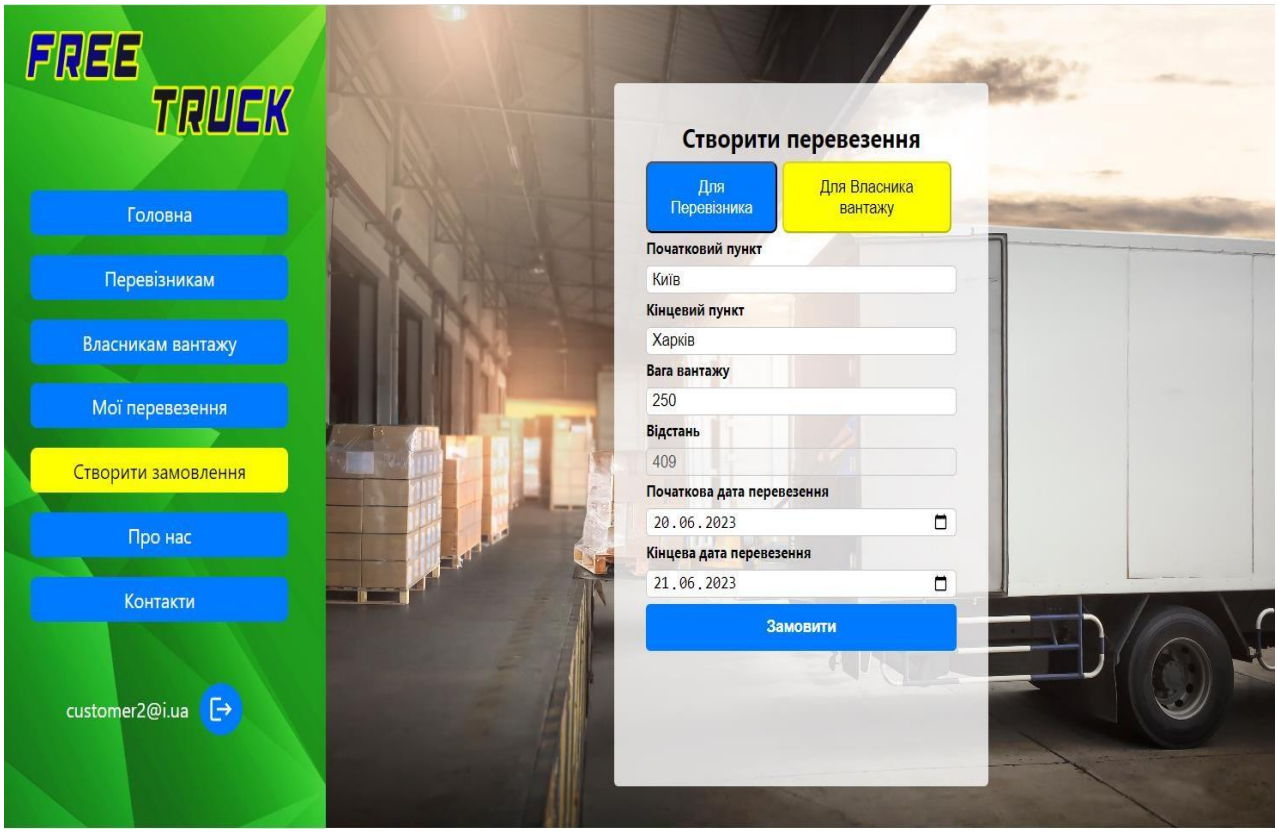


Рис. 3.26. Меню створення перевезення для Власника вантажу

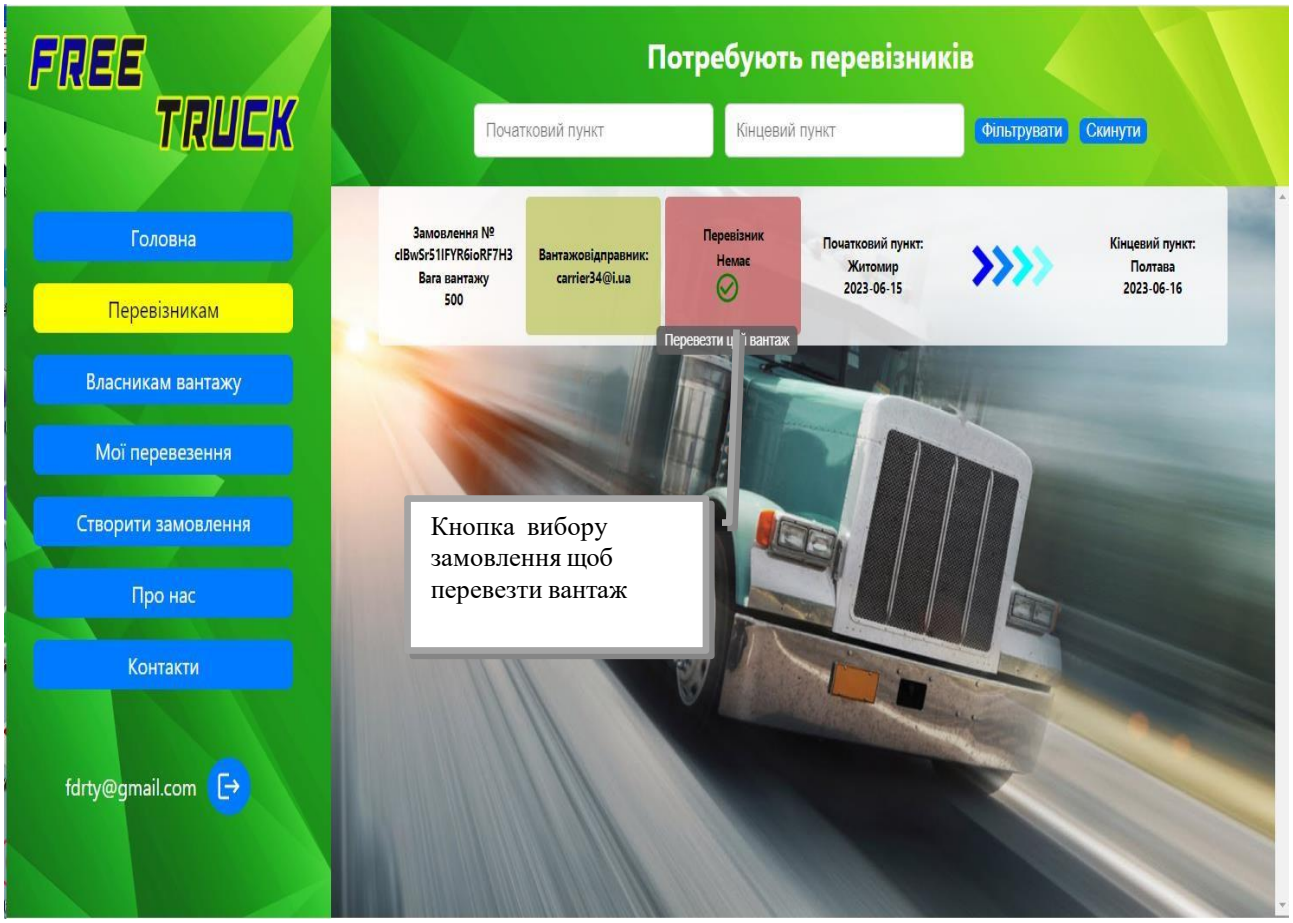


Рис. 3.27. Сторінка перегляду вантажу для перевезення з можливістю обрати

Рис. 3.28. Сторінка перегляду перевізників з можливістю обрати

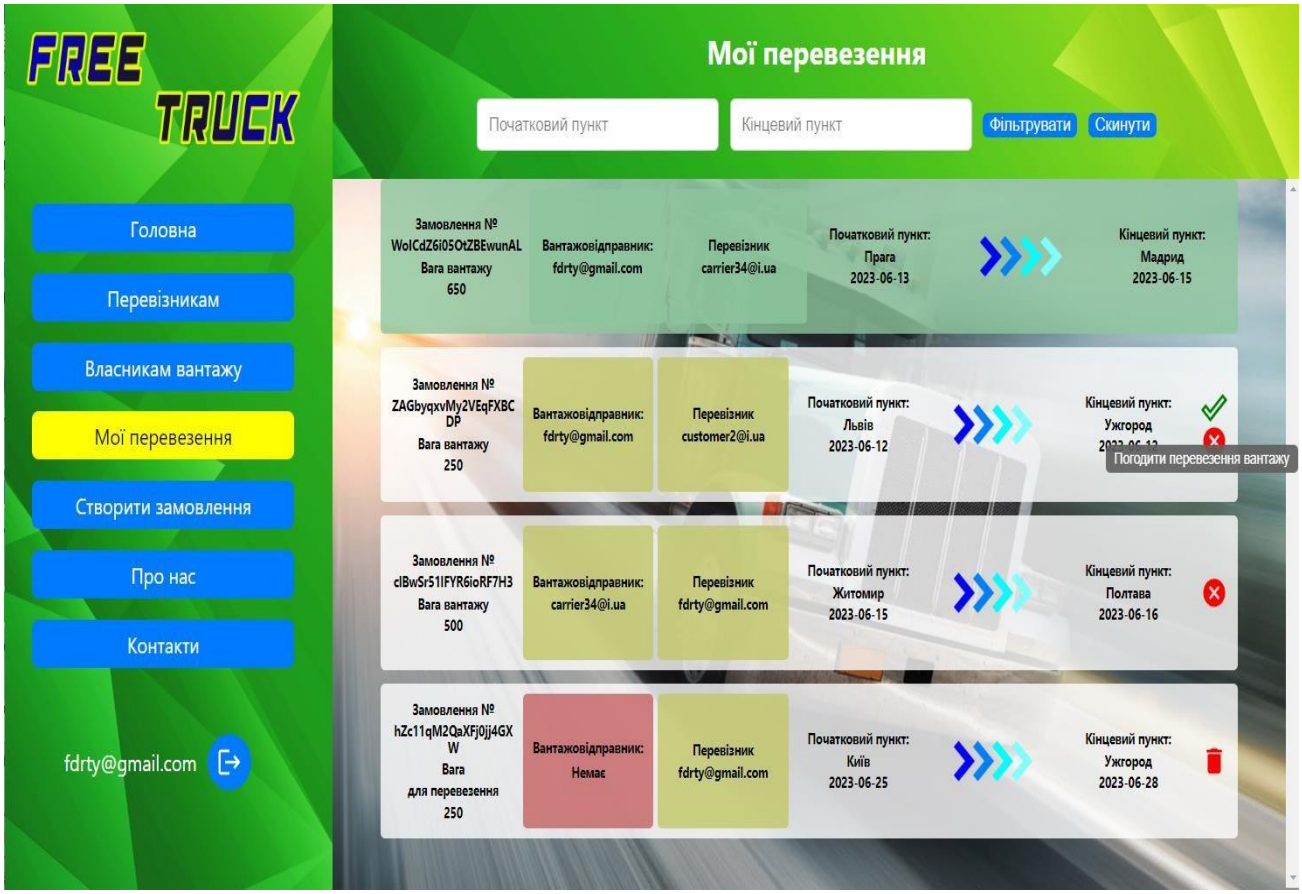


Рис. 3.29. Сторінка перегляду всіх перевезень в яких задіяний користувач

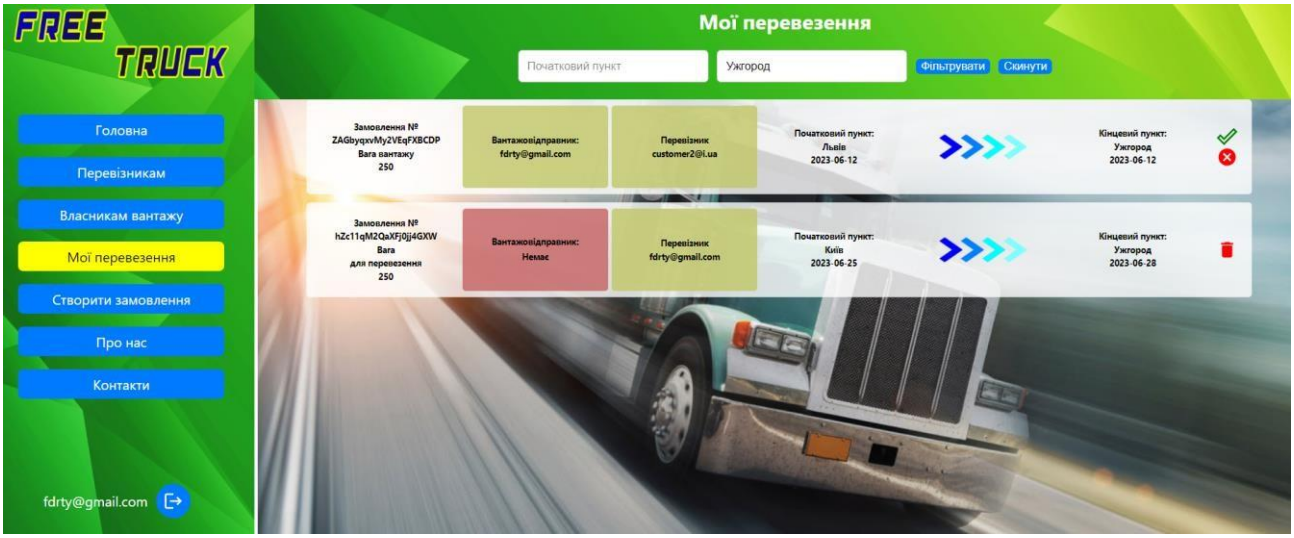


Рис. 3.30. Фільтрування перевезень по пунктах відправлення і призначення

Рис. 3.31. Різні типи перевезень та маніпуляція з ними

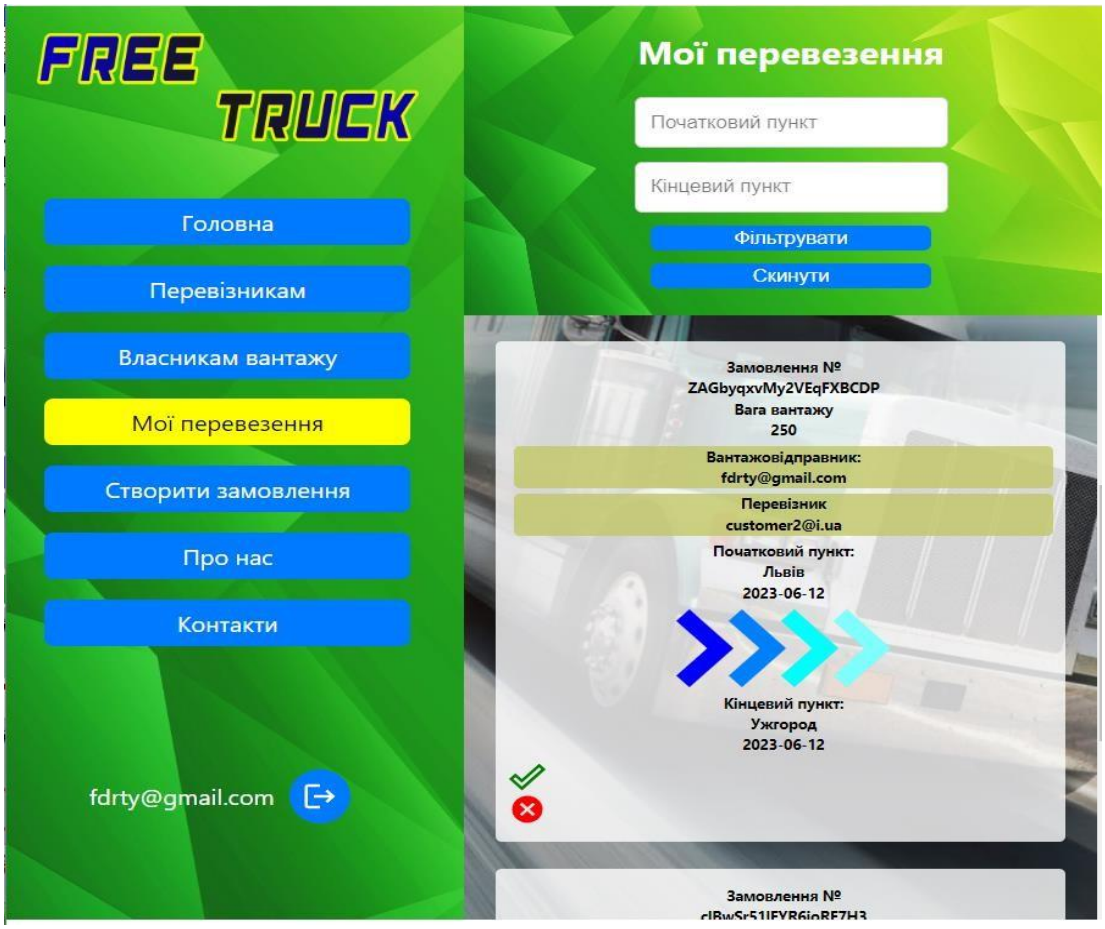


Рис. 3.32. Адаптивна верстка сайту

ВИСНОВКИ

"Free Truck" є веб-сайтом, пов'язаним з організацією та управлінням перевезеннями. Сайт надає користувачам можливість створювати та переглядати замовлення на перевезення вантажу.

Функціонал сайту дозволяє користувачам вибирати перевізника, підтверджувати або відмовлятись від перевезення, вказувати деталі щодо ваги та маршруту вантажу. Крім того, є можливість видаляти створені замовлення. Сайт також використовує Redux для управління станом даних, зокрема для зберігання інформації про користувачів та перевезення.

"Free Truck" зручна платформа для організації та взаємодії з перевезеннями вантажу. Зручний інтерфейс, який дозволяє здійснювати різні дії та отримувати необхідну інформацію. React є потужним та гнучким інструментом для розробки веб-інтерфейсів, який надає продуктивну розробку, швидкість та ефективне управління станом. Його активна спільнота та екосистема дозволяють швидко розвивати та покращувати проекти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Coyle J. J., Novack R. A., Gibson B. J., Bardi E. J. Transportation: A Global Supply Chain Perspective. – 9th ed. – Cengage Learning, 2019.
2. Rodrigue J.-P. The Geography of Transport Systems. – 5th ed. – Routledge, 2020.
3. Rushton A., Croucher P., Baker P. The Handbook of Logistics and Distribution Management. – 6th ed. – Kogan Page, 2022.
4. Bowersox D. J., Closs D. J., Cooper M. B., Bowersox J. C. Supply Chain Logistics Management. – 5th ed. – McGraw-Hill Education, 2019.
5. Wood D. F., Barone A. P., Murphy P. R., Wardlow D. L. International Logistics. – 3rd ed. AMACOM, 2020.
6. Belzer M. H. Sweatshops on Wheels: Winners and Losers in Trucking Deregulation. Oxford University Press, 2019.
7. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. [Електронний ресурс]. – Режим доступу: https://uk.legacy.reactjs.org/docs/getting-started.html (дата звернення: 02.03.2023 р.).
8. Microsoft. TypeScript Documentation. [Електронний ресурс]. – Режим доступу: https://www.typescriptlang.org/docs/ (дата звернення: 03.03.2023 р.).
9. React Suite. Introduction [Електронний ресурс]. – Режим доступу: https://rsuitejs.com/guide/introduction (дата звернення: 06.03.2023 р.).
10. ESLint Team. ESLint Documentation. [Електронний ресурс]. – Режим доступу: https://eslint.org/docs (дата звернення: 04.04.2023 р.).
11. Flanagan D. JavaScript: The Definitive Guide. – 7th ed – O'Reilly Media, 2020.
12. Wieruch, R. The Road to React: Your Journey to Master Plain Yet Pragmatic React.js.. Leanpub, 2022.
13. Griffiths, D., & Griffiths, D. React Cookbook: Recipes for Mastering the React Framework. O'Reilly Media, 2018.
14. Santana Roldán, C. React Cookbook: Create Dynamic Web Apps with React Using Redux, Webpack, Node.js, and GraphQL. Apress, 2019.
15. Mardan A., Banks A., Porcello E. React Quickly: Painless Web Apps with React, JSX, Redux, and GraphQL. – Shelter Island : Manning Publications, 2018.

ДОДАТКИ

Сторінка «index.js»

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { App } from 'components/App';
import './index.css';
import { BrowserRouter } from 'react-router-dom';
import { Provider } from 'react-redux';
import { store, persistor } from 'store/store';    "persistor": U
import { PersistGate } from 'redux-persist/integration/react';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <BrowserRouter basename="/FreeTruck">
      <Provider store={store}>
        <PersistGate loading={null} persistor={persistor}>    "pe
          <App />
        </PersistGate>
      </Provider>
    </BrowserRouter>
  </React.StrictMode>
); <- #10-20 ReactDOM.createRoot(document.getElementById('root'))
```

Компонент «App.jsx»

```
import Home from 'pages/Home/Home';
import { Route, Routes } from 'react-router-dom';
import Layout from './Layout/Layout';
import About from 'pages/About/About';
import Contacts from 'pages/Contacts/Contacts';
import Carrier from 'pages/Carrier/Carrier';
import CargoOwner from 'pages/CargoOwner/CargoOwner';
import CreateTransportationForm from 'pages/CreateTransportationForm/CreateTransporta
import MyShip from 'pages/MyShip/MyShip';

export const App = () => {
  return (
    <Routes>
      <Route path="/" element={<Layout />}>
        <Route index element={<Home />} />
        <Route path="carrier" element={<Carrier />} />
        <Route path="cargo_owner" element={<CargoOwner />} />
        <Route path="about" element={<About />} />
        <Route path="contacts" element={<Contacts />} />
        <Route path="create" element={<CreateTransportationForm />} />
        <Route path="myship" element={<MyShip />} />    "myship": Unknown word.
      </Route>
    </Routes>
  );
```

Сторінка «Home.jsx»

```jsx
import React from 'react';
import {
  Button,
  Content,
  Description,
  Highlight,
  HomeContainer,
  MainContent,
  Subtitle,
  Title,
} from './Home.styled'; <- #2-11 import
import { useDispatch, useSelector } from 'react-redux';
import { setShowAuthForm } from 'store/auth/authSlice';

const HomePage = () => {
  const user = useSelector(state => state.auth.user);
  const dispatch = useDispatch();
  const handleToggleAuthForm = () => {
    dispatch(setShowAuthForm(true));
  };
  return (
    <Content>
      <MainContent>
        <HomeContainer>
          <Title>Ласкаво просимо до Free Truck!</Title>    "Ласкаво": Unknown word.
          <Subtitle>Ми займаємося вантажними перевезеннями.</Subtitle>    "займаємо
          <Description>
            Знаходьте первізників вантажу у нас, швидко або займіть його місце.
          </Description>
          <Highlight>Зручно. Безпечно. Швидко.</Highlight>    "Зручно": Unknown wor
          {!user && (
            <Button onClick={handleToggleAuthForm}>Увійти в систему</Button>    "Ув
          )}
        </HomeContainer>
      </MainContent>
    </Content>
  ); <- #21-37 return
}; <- #15-38 const HomePage = () =>

export default HomePage;
```

Компонент «Home.styled.js»

```
import styled from 'styled-components';
import FonImage from '../../images/Home.jpg';

export const HomeContainer = styled.div`
  text-align: center;
  max-width: 80%;
  max-height: 80%;
  padding: 20px;
  background-color: rgba(255, 255, 255, 0.8);
`;

export const Title = styled.h1`
  font-size: 24px;
  font-weight: bold;
  margin-bottom: 10px;
`;

export const Subtitle = styled.p`
  font-size: 18px;
  margin-bottom: 10px;
`;

export const Description = styled.p`
  margin-bottom: 20px;
`;
export const Content = styled.div`
  height: 98vh;
  max-width: calc(100% - 300px);
  flex-grow: 1;
  background-image: url(${FonImage});
  background-repeat: no-repeat;
  background-size: cover;
  background-position: 50% 50%;
`;
export const Highlight = styled.p`
  font-size: 16px;
  font-weight: bold;
  color: #106701;
`;
export const Button = styled.button`
  padding: 10px 20px;
  font-size: 16px;
  color: #fff;
  background-color: #007bff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  &:hover {
    background-color: yellow;
    color: #000;
  }
`;
export const MainContent = styled.div`
  width: 100%;
  height: 100%;
  display: flex;
  justify-content: center;
  align-items: center;
`;
```

Сторінка «MyShip.jsx»

```jsx
import React, { useEffect, useState } from 'react';
import { Content, MainContent, Title, TitleContainer } from './MyShip.styled';
import TransportationList from '../../components/TransportationList/TransportationList';
import { useDispatch, useSelector } from 'react-redux';
import { fetchItems } from '../../store/getItems/getItems';
import TransportationFilter from '../../components/TransportationFilter/TransportationFilter';

const MyShip = () => {
  const user = useSelector(state => state.auth.user);
  const dispatch = useDispatch();
  const allTransportations = useSelector(state => state.getItems.items);    "Transportations": Unknown word.
  const [filteredTransportations, setFilteredTransportations] =    "Transportations": Unknown word.
    useState(allTransportations);    "Transportations": Unknown word.
  const [initialFilters, setInitialFilters] = useState({});

  useEffect(() => {
    dispatch(fetchItems());
  }, [dispatch]);
  useEffect(() => {
    // Фільтрувати початковий масив    "Фільтрувати": Unknown word.
    const FilteredItems = allTransportations.filter(    "Transportations": Unknown word.
      item =>
        item.carrier === user.displayName || item.customer === user.displayName
    );
    console.log(FilteredItems, user.displayName);  ×100 [ { id: 'WoICdZ6i05OtZBEwunAl', distance: 1772, creator: 'fdrty@gmai
    setFilteredTransportations(FilteredItems);
  }, [allTransportations,user.displayName]); <- #19-27 useEffect

  const handleFilterChange = filterValues => {
    const { originCity, destinationCity } = filterValues;

    // Застосувати фільтр до списку перевезень    "Застосувати": Unknown word.
    const filteredItems = allTransportations.filter(item => {
      if (
        originCity &&
        item.originCity.toLowerCase().includes(originCity.toLowerCase())
      ) {
        return true;
      }
      if (
        destinationCity &&
        item.destinationCity
          .toLowerCase()
          .includes(destinationCity.toLowerCase())
      ) {
        return true;
      }
      return false;
    }); <- #33-49 const filteredItems = allTransportations.filter
```

```jsx
    setFilteredTransportations(filteredItems);
  }; <- #29-52 const handleFilterChange = filterValues =>

  const handleResetFilters = () => {
    setFilteredTransportations(allTransportations);
    setInitialFilters({});
  };

  return (
    <Content>
      {user && (
        <>
          <TitleContainer>
            <Title>Мої перевезення</Title>    "перевезення": Unknown word.
            <TransportationFilter
              initialFilters={initialFilters}
              onFilterChange={handleFilterChange}
              onResetFilters={handleResetFilters}
            />
          </TitleContainer>

          <MainContent>
            <TransportationList
              transportations={filteredTransportations}    "transportations": Unknown word.
              onButton={true}
            />
          </MainContent>
        </>
      )} <- #61-79 <Content>
    </Content>
  ); <- #59-81 return
}; <- #8-82 const MyShip = () =>

export default MyShip;
```

Компонент «MyShip.styled.js»

```
import styled from 'styled-components';
import FonImage from '../../images/Main.jpg';
import SideImage from '../../images/SideBar.jpg';
export const Content = styled.div`
  height: 98vh;
  max-width: calc(100% - 300px);
  flex-grow: 1;
  background-image: url(${FonImage});
  background-repeat: no-repeat;
  background-size: cover;
  background-position: 50% 50%;
  overflow: hidden;
`;
export const MainContent = styled.div`
  width: 100%;
  height: 80%;
  display: flex;
  justify-content: center;
  align-items: center;
  flex: 1;
  overflow-y: scroll;
`;
export const Title = styled.h2`
  text-align: center;
`;
export const TitleContainer = styled.div`
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  width: 100%;
  background-image: url(${SideImage});
  background-repeat: no-repeat;
  background-size: cover;
  background-position: 30% 50%;
  color: #fff;
`;
```

Компонент «CreateTransportationForm.styled.js»

```
import styled from 'styled-components';
import FonImage from '../../images/BackTruck.jpg';

export const FormContainer = styled.div`
  max-width: 80%;
  height: 100%;
  margin: 0 auto;
  padding: 30px;
  background-color: rgba(255, 255, 255, 0.8);
  border-radius: 4px;
`;

export const FormTitle = styled.h2`
  text-align: center;
  font-size: 20px;
  height: 20px;
  margin: 0 0 10px 0;
`;

export const Form = styled.form`
  display: flex;
  flex-direction: column;
`;

export const FormField = styled.div`
  margin-bottom: 5px;
  display: flex;
  flex-direction: column;

  label {
    font-size: 12px;
    margin-bottom: 5px;
    height: 15px;
    font-weight: bold;
  }

  input {
    height: 10px;
    padding: 5px;
    border: 1px solid #ccc;
    border-radius: 4px;
  }
`;

export const SubmitButton = styled.button`
  padding: 10px;
  background-color: #007bff;
  color: #fff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-weight: bold;

  &:hover {
    background-color: #0056b3;
  }
`;

export const ResponsiveContainer = styled.div`
  height: 80%;
  @media (max-width: 768px) {
    ${FormContainer} {
      max-width: 100%;
    }

    ${FormField} {
      label {
        text-align: center;
      }

      input {
        width: 100%;
```

```
export const Content = styled.div`
  height: 98vh;
  max-width: calc(100% - 300px);
  flex-grow: 1;
  background-image: url(${FonImage});
  background-repeat: no-repeat;
  background-size: cover;
  background-position: 50% 50%;
`;
export const MainContent = styled.div`
  width: 100%;
  height: 95%;
  display: flex;
  justify-content: center;
  align-items: center;
`;

export const TabContainer = styled.div`
  display: flex;
  flex-direction: column;
  margin-bottom: 20px;
`;

export const TabHeader = styled.div`
  display: flex;
`;

export const TabButton = styled.button`
  margin: 0 5px 5px 0;
  display: block;
  padding: 10px;
  text-align: center;
  background-color: yellow;
  color: #000;
  border-radius: 6px;
  text-decoration: none;
  &:hover {
    background-color: yellow;
    color: #000;
  }
`;
export const TabButton1 = styled.button`
  margin: 0 5px 5px 0;
  display: block;
  padding: 10px;
  text-align: center;
  color: #fff;
  background-color: #007bff;
  border-radius: 6px;
  text-decoration: none;
  &:hover {
    background-color: yellow;
    color: #000;
  }
`;
export const List = styled.ul`
  position: absolute;
  width: 32%;
  top: 50%;
  list-style-type: none;
  padding: 0;
  margin: 0;
  font-size: 10px;
`;
```

```
export const ListItem = styled.li`
  cursor: pointer;
  padding: 3px;
  background-color: #f2f2f2;

  &:hover {
    color: #fff;
    background-color: #007bff;
  }
`;
```

Сторінка «Contacts.jsx»

```
import React from 'react';
import {
  ContactContainer,
  ContactInfo,
  Content,
  Label,
  Link,
  MainContent,
  Subtitle,
  Title,
} from './Contacts.styled'; <- #2-11 import

const Contacts = () => {
  return (
    <Content>
      <MainContent>
        {' '}
        <ContactContainer>
          <Title>Контакти</Title>     "Контакти": Unknown word.
          <Subtitle>
            Зв'яжіться з нами для отримання додаткової інформації:     "Зв'яжіться": Unknown word
          </Subtitle>
          <ContactInfo>
            <Label>Email:</Label>{' '}
            <Link href="mailto:info@freetruck.com">info@freetruck.com</Link>
          </ContactInfo>
          <ContactInfo>
            <Label>Телефон:</Label>{' '}     "Телефон": Unknown word.
            <Link href="tel:+380980000000">+38 098 000 000 0</Link>
          </ContactInfo>
          <ContactInfo>
            <Label>Адреса:</Label> Free Truck, вул. Центральна 10, м. Київ,     "Адреса": Unknown
            Україна     "Україна": Unknown word.
          </ContactInfo>
        </ContactContainer>
      </MainContent>
    </Content>
  ); <- #14-38 return
}; <- #13-39 const Contacts = () =>

export default Contacts;
```

Сторінка «Carrier.jsx»

```jsx
import React, { useEffect, useState } from 'react';
import { Content, MainContent, Title, TitleContainer } from './Carrier.styled';
import TransportationList from '../../components/TransportationList/TransportationList';
import { useDispatch, useSelector } from 'react-redux';
import { fetchItems } from '../../store/getItems/getItems';
import TransportationFilter from '../../components/TransportationFilter/TransportationFilter';

const Carrier = () => {
  const user = useSelector(state => state.auth.user);
  const dispatch = useDispatch();
  const allTransportations = useSelector(state => state.getItems.items);    "Transportations": Unknown word.
  const [filteredTransportations, setFilteredTransportations] = useState(null);    "Transportations": Unknown word.
  const [initialFilters, setInitialFilters] = useState({});

  useEffect(() => {
    dispatch(fetchItems);
  }, [dispatch]);

  useEffect(() => {
    // Фільтрувати початковий масив    "Фільтрувати": Unknown word.
    if (allTransportations) {    "Transportations": Unknown word.
      const FilteredItems = allTransportations.filter(    "Transportations": Unknown word.
        item => item.carrier === false && item.customer !== user.displayName
      );
      console.log(FilteredItems, user.displayName);  ×14 [] | 'fdrty@gmail.com' ⌛ [] | 'fdrty@gmail.com' ⌛ [] | 'fdrty@gmail.com' ⌛ [] |
      setFilteredTransportations(FilteredItems);
    }
  }, [allTransportations, user.displayName]);

  const handleFilterChange = filterValues => {
    const { originCity, destinationCity } = filterValues;

    // Застосувати фільтр до списку перевезень    "Застосувати": Unknown word.
    const filteredItems = allTransportations.filter(item => {
      if (
        originCity &&
        item.originCity.toLowerCase().includes(originCity.toLowerCase())
      ) {
        return true;
      }
      if (
        destinationCity &&
        item.destinationCity
          .toLowerCase()
          .includes(destinationCity.toLowerCase())
      ) {
        return true;
      }
      return false;
    });

    setFilteredTransportations(filteredItems);
  };

  const handleResetFilters = () => {
    setFilteredTransportations(allTransportations);
    setInitialFilters({});
  };

  return (
    <Content>
      {user && (
        <>
          <TitleContainer>
            <Title>Потребують перевізників</Title>    "Потребують": Unknown word.
            <TransportationFilter
              initialFilters={initialFilters}
              onFilterChange={handleFilterChange}
              onResetFilters={handleResetFilters}
            />
          </TitleContainer>
```

```jsx
            />
          </TitleContainer>

          <MainContent>
            {filteredTransportations && (
              <TransportationList
                transportations={filteredTransportations}    "transportations": Unknown word.
                onButton={false}
              />
            )} <-- #74-79 <MainContent>
          </MainContent>
        </>
      )} <-- #62-82 <Content>
    </Content>
  ); <-- #60-84 return
}; <-- #8-85 const Carrier = () =>

export default Carrier;
```

Сторінка «CargoOwner.jsx»

```jsx
import React, { useEffect, useState } from 'react';
import {
  Content,
  MainContent,
  Title,
  TitleContainer,
} from './CargoOwner.styled'; <- #2-7 import
import TransportationList from '../../components/TransportationList/TransportationList';
import { useDispatch, useSelector } from 'react-redux';
import { fetchItems } from '../../store/getItems/getItems';
import TransportationFilter from '../../components/TransportationFilter/TransportationFilter';

const CargoOwner = () => {
  const user = useSelector(state => state.auth.user);
  const dispatch = useDispatch();
  const allTransportations = useSelector(state => state.getItems.items);    "Transportations": Unknown word.
  const [filteredTransportations, setFilteredTransportations] =    "Transportations": Unknown word.
    useState(allTransportations);    "Transportations": Unknown word.
  const [initialFilters, setInitialFilters] = useState({});

  useEffect(() => {
    dispatch(fetchItems());
  }, [dispatch]);
  useEffect(() => {
    // Фільтрувати початковий масив    "Фільтрувати": Unknown word.
    const FilteredItems = allTransportations.filter(    "Transportations": Unknown word.
      item => item.customer === false && item.carrier !== user.displayName
    );
    console.log(FilteredItems);  ×48 [ { id: 'Bgy2ePc0bEteTarlh7Qk', cargoWeight: 1500, shipmentTrue: false, cargoOwn
    setFilteredTransportations(FilteredItems);
  }, [allTransportations, user.displayName]); <- #24-31 useEffect

  const handleFilterChange = filterValues => {
    const { originCity, destinationCity } = filterValues;

    // Застосувати фільтр до списку перевезень    "Застосувати": Unknown word.
    const filteredItems = allTransportations.filter(item => {
      if (
        originCity &&
        item.originCity.toLowerCase().includes(originCity.toLowerCase())
      ) {
        return true;
      }
      if (
        destinationCity &&
        item.destinationCity
          .toLowerCase()
          .includes(destinationCity.toLowerCase())
      ) {
        return true;
      }
      return false;
    }); <- #37-53 const filteredItems = allTransportations.filter

    setFilteredTransportations(filteredItems);
  }; <- #33-56 const handleFilterChange = filterValues =>

  const handleResetFilters = () => {
    setFilteredTransportations(allTransportations);
    setInitialFilters({});
  };

  return (
    <Content>
      {user && (
        <>
          <TitleContainer>
            <Title>Потрібно перевезти</Title>    "Потрібно": Unknown word.
            <TransportationFilter
              initialFilters={initialFilters}
              onFilterChange={handleFilterChange}
              onResetFilters={handleResetFilters}
            />
          </TitleContainer>

          <MainContent>
            <TransportationList
              transportations={filteredTransportations}    "transportations": Unknown word.
              onButton={false}
            />
          </MainContent>
        </>
      )} <- #65-83 <Content>
    </Content>
  ); <- #63-85 return
}; <- #13-86 const CargoOwner = () =>

export default CargoOwner;
```

Сторінка «CargoOwner.styled.js»

```
import styled from 'styled-components';
import FonImage from '../../images/Car1.jpg';
import SideImage from '../../images/SideBar.jpg';

export const Content = styled.div`
  height: 98vh;
  max-width: calc(100% - 300px);
  flex-grow: 1;
  background-image: url(${FonImage});
  background-repeat: no-repeat;
  background-size: cover;
  background-position: 50% 50%;
  overflow: hidden;
`;
export const MainContent = styled.div`
  width: 100%;
  height: 80%;
  display: flex;
  justify-content: center;
  align-items: center;
  flex: 1;
  overflow-y: scroll;
`;
export const Title = styled.h2`
  text-align: center;
`;
export const TitleContainer = styled.div`
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  width: 100%;
  background-image: url(${SideImage});
  background-repeat: no-repeat;
  background-size: cover;
  background-position: 30% 50%;
  color: #fff;
`;
```

Сторінка «About.jsx»

```
import React from 'react';
import {
  AboutContainer,
  Content,
  Description,
  MainContent,
  Paragraph,
  Subtitle,
  Title,
} from './About..styled.'; <-- #2-10 import
const About = () => {
  return (
    <Content>
      <MainContent>
        <AboutContainer>
          <Title>Про нас</Title>
          <Description>
            Free Truck - це компанія, яка спеціалізується на вантажних     "компанія":
            перевезеннях. Ми надаємо професійні послуги з налагодження звязку     "пере
            між перевізниками та їхніми клієнтами швидко та надійно.     "перевізниками
          </Description>
          <Subtitle>Наша місія</Subtitle>     "Наша": Unknown word.
          <Paragraph>
            Ми прагнемо забезпечити нашим клієнтам найкращі вантажні перевезення     "п
            з найвищою якістю обслуговування. Наша місія полягає в тому, щоб     "найви
            забезпечити безпечну та надійний звязок за доступними цінами.     "забезпеч
          </Paragraph>
          <Subtitle>Наші цінності</Subtitle>     "Наші": Unknown word.
          <Paragraph>
            - Професіоналізм: Ми працюємо лише з досвідченими та надійними     "Професі
            водіями, щоб забезпечити високу якість перевезень.     "водіями": Unknown w
          </Paragraph>
          <Paragraph>
            - Відповідальність: Ми відповідаємо за кожне перевезення та     "Відповідал
            гарантуємо його безпечну та своєчасну доставку.     "гарантуємо": Unknown w
          </Paragraph>
          <Paragraph>
            - Клієнтоорієнтованість: Ми завжди ставимо потреби та задоволення     "Клі
            наших клієнтів на перше місце.     "наших": Unknown word.
          </Paragraph>
        </AboutContainer>
      </MainContent>
    </Content>
  ); <-- #12-44 return
}; <-- #11-45 const About = () =>

export default About;
```

Компонент «TransportationList.jsx»

```jsx
import React from 'react';
import ArrowRight from '../../images/4-arrows-right.png';
import CheckCircleOutlineIcon from '@mui/icons-material/CheckCircleOutline';
import Tooltip from '@mui/material/Tooltip';
import CancelIcon from '@mui/icons-material/Cancel';
import DeleteIcon from '@mui/icons-material/Delete';
import DoneOutlineIcon from '@mui/icons-material/DoneOutline';
import {
  Container,
  Image,
  Rout,
  StyledLink,
  TransportationItem,
} from './TransportationList.styled'; <- #8-14 import
import { useDispatch, useSelector } from 'react-redux';
import { removeItem, updateItem } from 'store/items/itemsSlice';

const TransportationList = ({ transportations, onButton }) => {    "transportations": Unknown word.
  const user = useSelector(state => state.auth.user);
  const dispatch = useDispatch();

  const handleChangeCarrier = transportations => {    "transportations": Unknown word.
    const
      id:                                              : Unknown word.
      ...transportations,    "transportations": Unknown word.
      carrier: user.displayName,
      carrierTrue: true,
    }; <- #23-28 const updatedItems =
    dispatch(updateItem(updatedItems));
    setTimeout(() => {
      // Оновлення сторінки    "Оновлення": Unknown word.
      window.location.reload();
    }, 2000);
  }; <- #22-34 const handleChangeCarrier = transportations =>
  const handleChangeCostumer = transportations => {    "transportations": Unknown word.
    const updatedItems = {
      id: transportations.id,
      ...transportations,
      customer: user.displayName,
      cargoOwnerTrue: true,
      cargoWeight: transportations.maxWeight,
    }; <- #36-42 const updatedItems =
    dispatch(updateItem(updatedItems));
    setTimeout(() => {
      // Оновлення сторінки    "Оновлення": Unknown word.
      window.location.reload();
    }, 2000);
  }; <- #35-48 const handleChangeCostumer = transportations =>
  const handleChangeCarrierTrue = transportations => {
    const updatedItems = {
      id: transportations.id,
      ...transportations,
      carrierTrue: true,
      shipmentTrue: true,
    }; <- #50-55 const updatedItems =
    dispatch(updateItem(updatedItems));
    setTimeout(() => {
      // Оновлення сторінки    "Оновлення": Unknown word.
      window.location.reload();
    }, 2000);
  }; <- #49-61 const handleChangeCarrierTrue = transportations =>
  const handleChangeCostumerTrue = transportations => {
    const updatedItems = {
      id: transportations.id,
      ...transportations,
      cargoOwnerTrue: true,
      shipmentTrue: true,
    }; <- #63-68 const updatedItems =
    dispatch(updateItem(updatedItems));
    setTimeout(() => {
      // Оновлення сторінки    "Оновлення": Unknown word.
      window.location.reload();
```

```
  }, 2000);
}; <- #62-74 const handleChangeCostumerTrue = transportations =>
const handleCancelTrue = transportations => {
  if (
    transportations.customer === user.displayName &&
    transportations.cargoOwnerTrue
  ) {
    const updatedItems = {
      id: transportations.id,
      ...transportations,
      cargoOwnerTrue: false,
      customer: false,
    }; <- #80-85 const updatedItems =
    dispatch(updateItem(updatedItems));
    setTimeout(() => {
      // Оновлення сторінки    "Оновлення": Unknown word.
      window.location.reload();
    }, 2000);
  } else {
    const updatedItems = {
      id: transportations.id,
      ...transportations,
      carrierTrue: false,
      carrier: false,
    }; <- #92-97 const updatedItems =
    dispatch(updateItem(updatedItems));
    setTimeout(() => {
      // Оновлення сторінки
      window.location.reload();
    }, 2000);
  } <- #91-103 else
}; <- #75-104 const handleCancelTrue = transportations =>
const handleDelete = transportations => {
  const removeItems = {
    id: transportations.id,
  };
  dispatch(removeItem(removeItems));
  setTimeout(() => {
    // Оновлення сторінки
    window.location.reload();
  }, 2000);
}; <- #105-114 const handleDelete = transportations =>
return (
  <Container>
    {transportations.map(transportation => (
      <StyledLink
        to={`/transportation/${transportation.id}`}
        key={transportation.id}
      >
        <TransportationItem
          style={{
            backgroundColor:
              transportation.shipmentTrue && 'rgba(141, 198, 158, 0.8)',
          }}
        >
          <Rout>
            <div>Замовлення №</div>    "Замовлення": Unknown word.
            <div>{transportation.id}</div>
            {transportation.cargoWeight ? (
              <>
                <div>Вага вантажу </div>    "Вага": Unknown word.
                <div>{transportation.cargoWeight}</div>
              </>
            ) : (
              <>
                <div> Вага </div>    "Вага": Unknown word.
                <div>для перевезення </div>    "перевезення": Unknown word.
                <div>{transportation.maxWeight}</div>
              </>
            )} <- #131-142 <div>{transportation.id}</div>
```

```
</Rout>
<Rout
  style={{
    backgroundColor:
      transportation.shipmentTrue === false
        ? transportation.customer === false
          ? 'rgba(197, 98, 98, 0.8)'
          : 'rgba(194, 199, 101, 0.8)'
        : 'rgba(141, 198, 158, 0.8)',
  }} <- #145-152 style=
>
  <div>Вантажовідправник:</div>    "Вантажовідправник": Unknown word.
  {transportation.customer ? (
    <div> {transportation.customer}</div>
  ) : (
    <div
      style={{
        display: 'flex',
        flexDirection: 'column',
        justifyContent: 'space-between',
      }} <- #159-163 style=
    >
      <div>Немає</div>    "Немає": Unknown  any
      {user.displayName !== transportation.carrier && (
        <Tooltip title="Перевезти свій вантаж">    "Перевезти": Unknown word.
          <CheckCircleOutlineIcon
            style={{ color: 'green', cursor: 'pointer' }}
            onClick={() => handleChangeCostumer(transportation)}
          />
        </Tooltip>
      )} <- #166-173 <div>Немає</div>
    </div>
  )} <- #155-175 <div>Вантажовідправник:</div>
</Rout>
<Rout
  style={{
    backgroundColor:
      transportation.shipmentTrue === false
        ? transportation.carrier === false
          ? 'rgba(197, 98, 98, 0.8)'
          : 'rgba(194, 199, 101, 0.8)'
        : 'rgba(141, 198, 158, 0.8)',
  }} <- #178-185 style=
>
  <div>Перевізник</div>    "Перевізник": Unknown word.
  {transportation.carrier ? (
    <div>{transportation.carrier}</div>
  ) : (
    <div
      style={{
        display: 'flex',
        flexDirection: 'column',
        justifyContent: 'space-between',
      }} <- #192-196 style=
    >
      <div>Немає</div>    "Немає": Unknown word.
      {user.displayName !== transportation.customer && (
        <Tooltip title="Перевезти цей вантаж">    "Перевезти": Unknown word.
          <CheckCircleOutlineIcon
            style={{ color: 'green', cursor: 'pointer' }}
            onClick={() => handleChangeCarrier(transportation)}
          />
        </Tooltip>
      )} <- #199-206 <div>Немає</div>
    </div>
  )} <- #188-208 <div>Перевізник</div>
</Rout>
<Rout>
  <div>Початковий пункт: </div>    "Початковий": Unknown word.
  <div>{transportation.originCity}</div>
```

```
            <div> {transportation.startDate}</div>
          </Rout>
          <Rout>
            <Image src={ArrowRight} alt="right" />
          </Rout>

          <Rout>
            <div>Кінцевий пункт:</div>    "Кінцевий": Unknown word.
            <div> {transportation.destinationCity}</div>
            <div> {transportation.endDate}</div>
          </Rout>
          {onButton && !transportation.shipmentTrue && (
            <Rout style={{ width: '5%' }}>
              {transportation.carrier !== user.displayName &&
                transportation.carrierTrue && (
                  <Tooltip title="Погодити перевезення вантажу">    "Погодити": Unknown word.
                    <DoneOutlineIcon
                      style={{ color: 'green', cursor: 'pointer' }}
                      onClick={() => handleChangeCarrierTrue(transportation)}
                    />
                  </Tooltip>
                )} <- #226-234 <Rout style={{ width: '5%' }}>
              {transportation.customer !== user.displayName &&
                transportation.cargoOwnerTrue && (
                  <Tooltip title="Погодити перевізника">    "Погодити": Unknown word.
                    <DoneOutlineIcon
                      style={{ color: 'green', cursor: 'pointer' }}
                      onClick={() => handleChangeCostumerTrue(transportation)}
                    />
                  </Tooltip>
                )} <- #235-243 {transportation.customer !== user.displayName &&
              {transportation.shipmentTrue === false &&
              transportation.creator === user.displayName ? (
                <Tooltip title="Видалити своє  перевезення повністю">    "Видалити": Unknown word.
                  <DeleteIcon
                    style={{ color: 'red', cursor: 'pointer' }}
                    onClick={() => handleDelete(transportation)}
                  />
                </Tooltip>
              ) : (
                <Tooltip title="Відмовитись від перевезення">    "Відмовитись": Unknown word.
                  <CancelIcon
                    style={{ color: 'red', cursor: 'pointer' }}
                    onClick={() => handleCancelTrue(transportation)}
                  />
                </Tooltip>
              )} <- #244-259 {transportation.shipmentTrue === false &&
            </Rout>
          )} <- #224-261 </Rout>
        </TransportationItem>
      </StyledLink>
    ))} <- #117-264 <Container>
  </Container>
  ); <- #115-266 return
}; <- #18-267 const TransportationList = ({ transportations, onButton }) =>

export default TransportationList;
```