Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ОНЛАЙН-СЕРВІСУ ДЛЯ ПОШУКУ ПОПУТНИКІВ ТА ВОДІЇВ ЗА ЗАДАНИМ МАРШРУТОМ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-2
Головянко Є.С.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Тітова О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Головянко Єлизаветі Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення онлайн-сервісу для пошуку попутників та водіїв за заданим маршрутом

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 27 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, дані з інтернет джерел, приклади створених онлайн-сервісів з пошуку попутників та водіїв.

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналіз методів створення онлайн-сервісу з пошуку попутників та пасажирів за заданим маршрутом.
2. Проєктування системи, компонентів та бази даних.
3. Програмна реалізація онлайн-сервісу.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми розроблення онлайн-сервісу для пошуку попутників та водіїв за заданим маршрутом, постановка задачі, тестові зображення .

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-14.04.23	
3	Аналіз літератури з досліджуваної проблеми	15.04.23-17.04.23	
4	Аналіз технічних засобів	18.04.23-19.04.23	
5	Розробка методу	19.04.23-20.04.23	
6	Програмна реалізація	20.04.23-30.04.23	
7	Оформлення пояснювальної записки	01.05.23-10.05.23	
8	Перевірка на плагіат	29.05.23	
9	Рецензування	30.05.23	
10	Підготовка презентації та доповіді	30.05.23-31.05.23	
11	Занесення роботи в електронний архів	31.05.23	
12	Попередній захист кваліфікаційної роботи	05.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи ____ _____ доц. Тітова О. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 61 с., 3 табл., 20 рис., 30 джерел.

ВОДІЙ, ЗАДАНИЙ МАРШРУТ, ОНЛАЙН-СЕРВІС, ПАСАЖИР, ПОПУТНИК, INTELLIJ IDEA, MONGODB, NEXT-AUTH, NEXT.JS, NODE.JS, REACT, REACT-QUERY, ZOD.

Об’єктом роботи онлайн-сервіс з пошуку пасажирів та водіїв за обраним маршрутом.

Метою даної кваліфікаційної роботи є розробка онлайн-сервісу, що призначений для легкого та ефективного знаходження попутників для подорожей за одним напрямком, а також водіїв, які можуть забрати пасажирів по дорозі до місця їх призначення.

У процесі розробки сервісу були використані сучасні технології та фреймворки, такі як React для розробки застосунку, react-hook-form для ствоерення форм, Node.js для розробки серверної частини та MongoDB для зберігання даних. Було використано бібліотеку валідації данних ZOD, що забезпечила коректну обробку введених даних. Також був використаний інструмент для забезпечення автентифікації користувачів, такий як Next-Auth.

DRIVER, INTELLIJ IDEA, MONGODB, NEXT-AUTH, NEXT.JS, NODE.JS, ONLINE SERVICE, PASSENGER, REACT, REACT-QUERY, SELECTED ROUTE, TRAVEL COMPANION, ZOD.

The object of this work is an online service for searching passengers and drivers for a chosen route.

The aim is to develop an online service designed for easy and efficient finding of travel companions in the same direction, as well as drivers who can pick up passengers along the way to their destination.

During the development of the service, modern technologies and frameworks such as React for web application development, react-hook-form for form creation, Node.js for server-side development, and MongoDB for data storage were used. The ZOD data validation library was used to ensure proper handling of input data. Additionally, a tool for user authentication, Next-Auth, was utilized.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Пасажир – користувач системи, який використує ресурс задля пошуку водія за заданим маршрутом

Водій – користувач системи, мета якого є віднайти пасажирів та надати свою послугу з перевезення

Фреймворк – це набір загальних рішень для певної галузі програмування, надає структуру для розробки програмного продукту, забезпечує певну архітектуру, стандарти кодування, методи роботи з базою даних, інтерфейси та інші функціональні можливості

Клієнт – у технологіях веброзробки термін «клієнт» (англ. client) використовується для позначення програмного забезпечення, що знаходиться на стороні користувача та взаємодіє з сервером, щоб отримувати та відправляти дані

Вебсервер – це програмне забезпечення, яке забезпечує обробку запитів, які надходять від клієнтських пристроїв (наприклад, браузерів) через мережу Інтернет

Фронтенд – це частина веброзробки, яка займається розробкою клієнтської частини вебсайтів і застосунків

Бекенд (або серверна частина) – це частина застосунку або програми, яка забезпечує обробку запитів та взаємодію з базою даних, що знаходиться на сервері

База даних – це організована колекція даних, яка зберігається та керується певною системою управління базами даних. Вона складається з таблиць, які містять різні типи даних, такі як числа, текст, дати та інші, та забезпечує доступ до цих даних для різних програм та користувачів

Масив – це структура даних, яка дозволяє зберігати інформацію про кілька значень (елементів) під одним іменем (змінною). У масиві елементи

зберігаються в послідовному порядку і доступ до них можна отримати за допомогою індексу – числового значення, яке вказує позицію елемента в масиві

Автентифікація – це процес перевірки та підтвердження ідентифікації користувача або системи, зазвичай з метою надання доступу до ресурсів або функцій, які вимагають авторизації

Авторизація – це процес перевірки, чи має користувач доступ до певних ресурсів, які він намагається отримати. Зазвичай, користувач авторизується після успішної автентифікації, тобто після підтвердження його ідентичності

API (Application Programming Interface) – це набір готових до використання програмних інтерфейсів, структур даних та процедур, які використовуються для забезпечення взаємодії між різними програмами. API дозволяє різним застосункам та сервісам спілкуватися між собою та виконувати певні завдання, обмінюючись інформацією

API-endpoint – це адреса вебсервера, до якої можна звернутись для доступу до ресурсів або послуг, що надаються через API

ВСТУП

Транспорт та, зокрема, транспортна система беззаперечно є одною із невід'ємних частин сучасного світу. Жодна людина не може уявити дорогу додому, на роботу, школу без авто, велосипеда чи автобуса.

Усі стали заручниками засобів перевезення. Саме через це, з кожним днем створюється все більше і більше технологій, які полегшують життя людини.

Онлайн-сервіси для пошуку попутників та водіїв за заданим маршрутом стали надзвичайно популярними в останні роки. Це пов'язано зі зростанням інтернет-технологій та швидким темпом життя, що змушує людей шукати швидкі та зручні способи подорожей.

Цей сервіс дозволить пасажирам шукати водіїв, які пропонують поїздки в потрібному напрямку, та домовлятися з ними про умови подорожі. З іншого боку, водії зможуть знайти попутників, які будуть їхніми співмандатами на подорожі, або знайти нових клієнтів для своєї таксі-служби.

Актуальність даної теми полягає в тому, що все більше людей шукають зручні та ефективні способи подорожей, а також прагнуть заощадити гроші та допомогти зменшити навантаження на дороги та довкілля.

Це створює потребу в інноваційних рішеннях, які дозволяють людям шукати попутників та водіїв для спільних поїздок.

У цілому, онлайн-сервіс для пошуку попутників та водіїв за заданим маршрутом є актуальною та корисною ідеєю для тих, хто прагне заощадити гроші та час на поїздках, зменшити негативний вплив на навколишнє середовище, а також збільшити гнучкість своїх поїздок.

1 АНАЛІЗ МЕТОДІВ СТВОРЕННЯ ОНЛАЙН-СЕРВІСУ З ПОШУКУ ПОПУТНИКІВ ТА ПАСАЖИРІВ ЗА ЗАДАНИМ МАРШРУТОМ

1.1 Дослідження доцільності розробки онлайн-сервісу

Онлайн-сервіси – це вебпрограми або програмні застосунки, які забезпечують користувачам доступ до різних послуг чи інформації через Інтернет. Вони можуть надавати широкий спектр послуг – від соціальних мереж та електронної пошти до онлайн торгівельних платформ з оплатою.

Історія онлайн-сервісів сягає кінця 1960-х років, коли у США була створена перша мережа пакетної передачі даних – ARPANET. У 1970-х роках почали з’являтися перші онлайн-сервіси, такі як «CompuServe» та «The Source», що надавали можливість користувачам обмінюватися повідомленнями та отримувати доступ до різноманітної інформації.

У 1990-ті роки з появою World Wide Web та першого браузера «Mosaic» створення онлайн-сервісів стало значно простішим. У цей період з’явилися такі відомі сервіси, як «Yahoo!», «Amazon» та «eBay». У 2000-х роках з’явилися соціальні мережі, такі як «Facebook», «Twitter» та «LinkedIn», які змінили спосіб, яким користувачі спілкуються та обмінюються інформацією в Інтернеті [1, 2].

Сьогодні онлайн-сервіси є необхідною складовою сучасного світу, надаючи користувачам можливість здійснювати різноманітні дії, такі як онлайн покупки, замовлення послуг, знайомства та багато іншого. Розробка онлайн-сервісів у сфері транспортного зв’язку, таких як даний сервіс для пошуку попутників та водіїв, є однією з напрямків розвитку інтернет-технологій та сприяє зручності та ефективності пересування користувачів.

На даний момент створення онлайн-сервісів може бути досить складним процесом. Залежно від складності функціоналу та вимог щодо дизайну та користувацького досвіду, розробка може зайняти від кількох

місяців до року або більше. Для успішної розробки необхідно мати досвід у різних технологіях та платформах, а також знати основні принципи дизайну та розробки користувацького інтерфейсу.

Однак, завдяки наявності безкоштовних та платних інструментів для розробки, таких як фреймворки та сервіси хмарної інфраструктури, створення онлайн-сервісів стало більш доступним. Також існують засоби для автоматизації розробки та тестування, що допомагають зменшити витрати часу та зусиль на розробку та випуск продукту. Тому, хоча створення онлайн-сервісу може бути складним процесом, з правильними інструментами та досвідом це можливо зробити ефективно та результативно.

Останнім часом збільшилась популярність так званих мікросервісів. Мікросервіси – це підхід до розробки програмного забезпечення, при якому застосунок розбивається на невеликі інкапсульовані складові, які можна розвивати та масштабувати окремо один від одного. Онлайн-сервіси, у свою чергу, представляють собою програмне забезпечення, яке надає користувачам доступ до певних функцій або послуг через Інтернет [3].

Основна відмінність між мікросервісами та онлайн-сервісами полягає у їхній архітектурі. Мікросервіси розбиваються на невеликі складові, що дозволяє їм бути більш гнучкими та швидкими в розробці та масштабуванні. З іншого боку, онлайн-сервіси зазвичай мають більш монолітну архітектуру, де всі функції і послуги вбудовані в один застосунок.

Однією з переваг мікросервісів є їхня гнучкість та можливість масштабування окремих складових системи, не впливаючи на роботу інших частин. Це дозволяє розробникам зосередитись на конкретних функціях та послугах, що робить процес розробки та внесення змін більш ефективним та швидким. Однак, відсутність цілісної архітектури може призвести до складнощів у розгортанні та управлінні складовими.

У свою чергу, онлайн-сервіси зазвичай мають більш просту архітектуру, що спрощує менеджмент їхнього життєвого циклу [4]. Крім того, цілісність системи може забезпечити більшу стабільність та надійність

роботи. Дивлячись на усі переваги та недоліки онлайн-сервісів, саме вони були обрані для розробки аплікації з пошуку попутників та пасажирів за заданим маршрутом.

1.2 Огляд інструментарію для створення онлайн-сервісу

Процес створення онлайн-сервісу є доволі непростим. До вибіру технологій та інструментарію, що буде використовуватися, потрібно підійти із зосередженістю, щоб уникнути велику кількість проблем у подальшому розвитку програми [5].

Якщо говорити про ефективність та час написання продукту [6, 7], то беззаперечно можна назвати лідера у цьому – JavaScript. Саме це робить його однією з найбільш популярних мов програмування для створення онлайн-сервісів. Перейдемо до вибору фреймворків та основних бібліотек.

Почнемо з фреймворку React та NextJS, який використовується для створення фронтенду застосунку. React дозволяє створювати ефективний та легкий у використанні інтерфейс користувача, а NextJS допомагає забезпечити швидкий рендерінг сторінок та SEO оптимізацію.

Для забезпечення роботи з даними використовується бібліотека React-Query, яка надає зручний та ефективний інтерфейс для взаємодії з API та кешування результатів запитів, що забезпечує більш швидкий та ефективний взаємодію з базою даних.

Для стилізації компонентів використовується бібліотека Material-UI, яка дозволяє швидко та просто створювати зовнішній вигляд застосунку.

Для аутентифікації користувачів використовується бібліотека Next-Auth, яка забезпечує зручну та безпечну аутентифікацію через різні провайдери, такі як Google, Facebook, GitHub та інші.

Для створення та валідації форм використовується бібліотека react-hook-form та спеціальна бібліотека для валідації Zod.

1.2.1 React та Next.js

React та Next.js – це два інструменти, що використовуються для розробки застосунків та інших інтерактивних інтерфейсів. React – це бібліотека JavaScript, яка дозволяє веброзробникам створювати компоненти, які можна повторно використовувати та компонувати для створення складніших інтерфейсів.

React дає змогу динамічно оновлювати вміст на сторінці без необхідності перезавантаження всієї сторінки. Крім того, React використовує віртуальний DOM (Document Object Model), що дозволяє швидко оновлювати інформацію на сторінці та зменшує кількість зайвих операцій з маніпулюванням DOM.

Next.js – це фреймворк для React, що додає додаткові можливості до розробки застосунків. Next.js дозволяє створювати статичні та серверні застосунки з React, і використовує віртуальний DOM для оптимізації швидкодії.

Основні переваги Next.js включають можливість створювати швидкі та масштабовані застосунки з використанням React, автоматичне код-сплітінг для оптимізації завантаження сторінок, SSR (Server-Side Rendering) для покращення SEO, можливість побудувати архітектуру проєкту за допомогою роутінгу та інших інструментів.

Незважаючи на те, що React та Next.js мають багато переваг, у них також є деякі недоліки:

– висока складність: React та Next.js є досить складними технологіями, які потребують певного рівня знань та досвіду. Наприклад, варто знати HTML, CSS, JavaScript та знати, як працюють компоненти та props;

– невелика зручність в роботі зі статичним контентом: Next.js дозволяє зберігати статичний контент, проте цей процес може бути дещо незручним в порівнянні з іншими фреймворками;

– погана продуктивність при обробці великих даних: React та Next.js можуть стикатися з проблемами продуктивності при обробці великих даних або при запуску на повільних пристроях. Для розв’язання цієї проблеми можна використовувати додаткові бібліотеки та оптимізаційні підходи;

– вимоги до серверної частини: Next.js залежить від серверної частини, що може створювати додаткові проблеми під час розгортання та налаштування.

Хоча ці недоліки можуть затримувати розробку проєкту, здебільшого їх можна вирішити або обійти, використовуючи правильні техніки, інструменти та підходи.

У даний момент React та Next.js є однією з найпопулярніших технологій для розробки застосунків та їх комбінація з іншими інструментами може дати чудовий результат.

1.2.2 Rest API

REST API (Representational State Transfer API) – це архітектурний стиль вебсервісів, що використовується для передачі даних між клієнтом та сервером. REST API передає дані у вигляді ресурсів, що можуть бути зображені за допомогою URL-адрес. REST API зазвичай використовується з протоколом HTTP і передає дані у форматах JSON або XML. Основні переваги REST API включають:

– стандартизований протокол: REST API використовує HTTP-протокол, що є стандартом для комунікації між вебсервером та клієнтом. Це забезпечує універсальність та сумісність між різними програмними продуктами;

– простота використання: REST API легко використовувати і розуміти. Його простота дозволяє розробникам швидко створювати вебсервіси;

– незалежність від мов програмування: REST API можна використовувати з будь-якою мовою програмування, оскільки він використовує стандартні протоколи і формати даних;

– гнучкість та масштабованість: REST API дозволяє розширювати та масштабувати функціональність вебсервісу без значних змін в коді [8], що забезпечує гнучкість та ефективність розробки;

– підтримка кешування: REST API може підтримувати кешування, що дозволяє збільшити швидкість передачі даних та зменшити навантаження на сервер;

– безпека: REST API може бути захищений за допомогою протоколів безпеки, таких як SSL / TLS, що забезпечують захист даних від несанкціонованого доступу.

Узагальнюючи, REST API є простим та ефективним для створення онлайн-сервісів.

1.2.3 React-Query

React-Query – це бібліотека для кешування та управління запитами даних в React застосунках. Вона працює з будь-якими запитами, включаючи REST, GraphQL та WebSocket, і дозволяє легко управляти станом даних в застосунку.

Одною з головних переваг React-Query є його можливість кешувати запити та автоматично оновлювати дані відповідно до їхньої актуальності. Кешування зменшує кількість запитів до сервера та покращує швидкість відгуку застосунку [9].

Також бібліотека має підтримку пагінації, дозволяє використовувати оптимістичне оновлення та автоматичну перевірку на наявність помилок в запитах.

Для використання бібліотеки не потрібно займатися написанням власного коду для кешування та управління запитами, тому розробник може сконцентруватися на функціональності свого застосунку.

React-Query також має відкритий код, тому користувачі можуть внести свої внески до розвитку бібліотеки.

Однак, є кілька недоліків React-Query. Перш за все, він має дещо кручену криву навчання, оскільки його концепції можуть бути незрозумілими для початківців [10].

Також, React-Query має певні обмеження щодо налаштування запитів, що може становити проблему в окремих випадках.

React Query підтримує різні бази даних, такі як REST API, GraphQL, Firebase, React Native AsyncStorage та інші. У даному проєкті буде використовуватися MongoDB, яка є документоорієнтованою NoSQL базою даних, що зберігає дані у вигляді документів, які можуть містити різні поля та вкладені об’єкти.

MongoDB добре підходить для зберігання та обробки великих об’ємів даних, а також для розробки застосунків, які вимагають гнучкості та швидкості розробки [11].

React Query працює з будь-якими джерелами даних, які повертають проміси або функції зворотного виклику, тому його можна використовувати з будь-якою базою даних, яка підтримує такий підхід.

Для роботи з MongoDB можна використовувати офіційний драйвер для Node.js – mongodb, який надає API для підключення та взаємодії з базою даних [12].

React Query дозволяє зручно та ефективно керувати кешем даних, отриманих з бази даних, що покращує продуктивність та швидкість роботи застосунку.

Завдяки вбудованому підходу до кешування даних, React Query може ефективно зменшувати кількість запитів до сервера та оптимізувати роботу застосунку в цілому.

1.2.4 MaterialUI

MaterialUI – це бібліотека компонентів інтерфейсу користувача на основі дизайну Google Material Design. Вона містить готові компоненти, такі як кнопки, поля вводу, таблиці, меню, модальні вікна, каруселі та інші, які можна використовувати для швидкої розробки застосунків [13].

MaterialUI має безліч переваг, серед яких можна виділити:

– консистентний дизайн – всі компоненти виконані в одному стилі, що допомагає створювати зручний та логічний інтерфейс;

– розширюваність – MaterialUI дозволяє налаштовувати та розширювати стандартні компоненти за допомогою тем та стилів;

– легкість використання – MaterialUI простий та легкий в використанні, що дозволяє швидко розробляти інтерфейс без головоболів;

– налаштування анімацій – MaterialUI надає можливість додавати анімації до компонентів.

У проєкті, який використовує React та MaterialUI, будуть використовуватися готові компоненти MaterialUI для створення інтерфейсу користувача. Також MaterialUI надає можливість створювати власні теми та стилі, що дозволяє налаштовувати компоненти під вимоги проєкту (рис. 1.1).

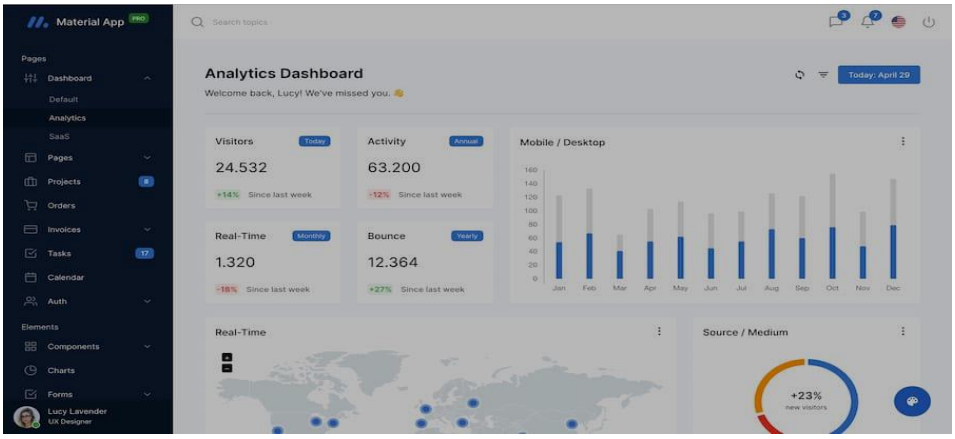

Рисунок 1.1 – Приклад застосунку, який розроблений за допомогою бібліотеки MaterialUI

Бібліотека MaterialUI працює з будь-якими фреймворками та бібліотеками JavaScript, включаючи React, Angular, Vue та інші. Для інтеграції MaterialUI з React існує спеціальний пакет react-material-ui, який дозволяє використовувати MaterialUI компоненти у проєктах на React [14].

1.3 Огляд платформ для пошуку попутників та водіїв

Даний розділ має на меті розглянути найпопулярніші платформи [15], які дозволяють користувачам знайти попутників або водіїв для подорожей. У світі існує багато платформ, які надають такі послуги, але буде розглянуто лише три основні: BlaBlaCar, CarpoolWorld та Uber.

Кожна з цих платформ має свої унікальні особливості та привабливості, які можуть залучити користувачів. Наприклад, BlaBlaCar є однією з найпопулярніших платформ у світі та має найбільшу базу користувачів, що дозволяє знайти попутників або водіїв на будь-яку відстань. CarpoolWorld, з іншого боку, є більш зосередженою на окремих регіонах та містах, що може бути зручно для тих, хто шукає попутників або водіїв у своєму місті. Враховуючи, що Uber не є спеціалізованою платформою для пошуку попутників та водіїв, вона все ж є популярною серед людей, які шукають транспорт з місця на місце [16].

1.3.1 BlaBlaCar

BlaBlaCar є однією з найбільших та найпопулярніших платформ для спільних поїздок в Європі та багатьох інших країнах. Сервіс був запущений у Франції у 2006 році та став зручним способом для подорожей у всьому світі.

Користувачі можуть шукати та бронювати місця в машинах інших користувачів, які пропонують подорожі з однієї точки до іншої. BlaBlaCar

працює на принципі взаємовигідної співпраці, коли водій отримує кошти за транспортування пасажирів, а пасажири – можливість швидко та дешево дістатися до свого пункту призначення.

Основною метою BlaBlaCar є зменшення кількості порожніх місць у автомобілях на дорогах, зменшення транспортних заторів та зниження викидів в атмосферу. Крім того, BlaBlaCar надає можливість знайомитися з новими людьми та ділитися досвідом подорожей.

Платформа BlaBlaCar має розширені функціональні можливості для пошуку і бронювання подорожей, включаючи пошук по маршруту, даті, часу, ціні та рейтингу водіїв.

Крім того, BlaBlaCar надає користувачам можливість оцінювати водіїв та пасажирів, щоб підвищити рівень безпеки та забезпечити якісну обслуговування для всіх.

BlaBlaCar працює у кількох десятках країн світу та є популярним вибором для подорожей, особливо на довгі відстані.

Платформа має свій вебсайт та мобільний застосунок для зручного використання на смартфонах та планшетах [17] (рис. 1.2).

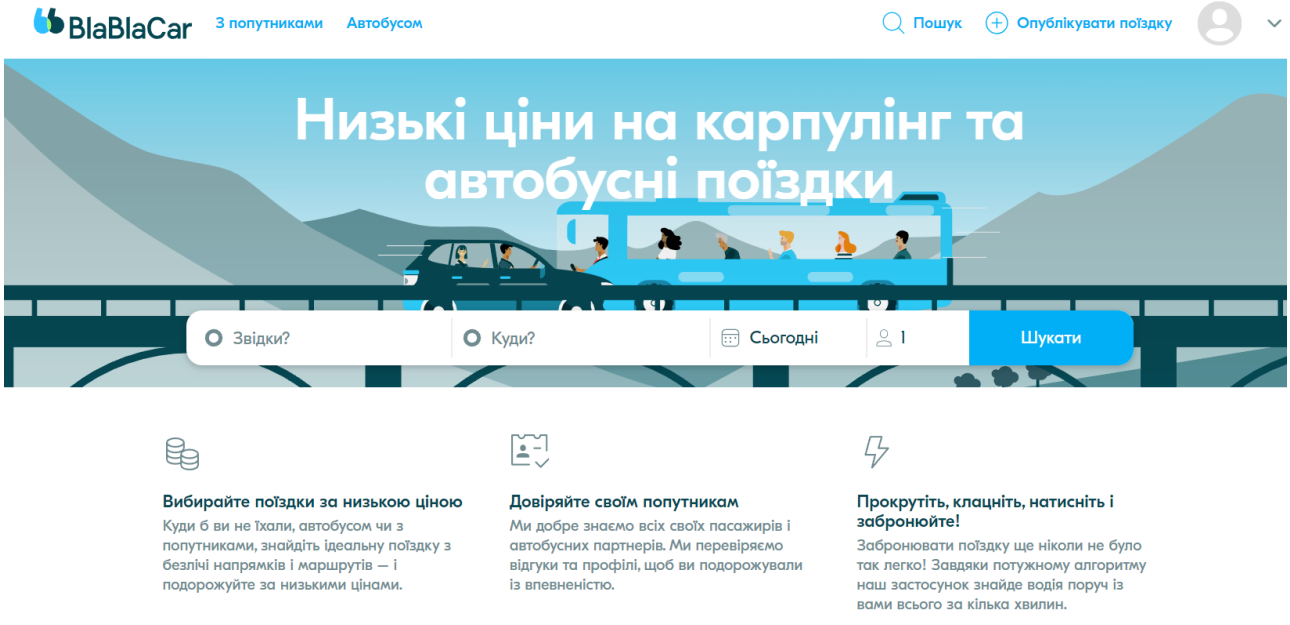

Рисунок 1.2 – Інтерфейс головної вебсторінки BlaBlaCar

BlaBlaCar є популярним сервісом для пошуку попутників та водіїв в Україні. Після запровадження карантинних обмежень у зв’язку з пандемією COVID-19, попит на подорожі зменшився, але сервіс продовжував функціонувати та навіть запустив безкоштовний сервіс для медичних працівників, які потребували транспорту для переміщення на роботу.

У зв’язку з війною в Україні, BlaBlaCar також відіграв свою роль у підтримці переселенців та військових, які потребували транспорту для переміщення до ізольованих зон, на передову лінію та назад. Багато водіїв волонтерів допомагали пересуватися безкоштовно або за символічну плату. BlaBlaCar також запустив спеціальну програму, щоб допомогти знайти безкоштовне житло волонтерам.

Загалом, BlaBlaCar зробив вагомий внесок у полегшення мобільності в Україні, як у часи миру, так і в умовах війни та пандемії.

1.3.2 CarpoolWorld

CarpoolWorld – це ще одна популярна платформа для пошуку попутників та водіїв, яка дозволяє знайти транспортне засіб для спільної поїздки. Ця платформа була заснована в 2006 році і має широку глобальну аудиторію, яка об’єднує користувачів з різних країн світу (рис. 1.3).

Основна ідея CarpoolWorld полягає в тому, щоб допомогти людям, які проживають в одному районі, спільно подорожувати до роботи, на навчання або на інші місця, що відповідають їхнім потребам. Крім того, платформа дозволяє шукати попутників для довгострокових поїздок, таких як подорожі до інших міст або країн.

Один з основних плюсів CarpoolWorld полягає в тому, що користувачі можуть знайти попутників та водіїв, які відповідають їхнім вимогам, що включає маршрут, дату та час поїздки, кількість місць, ціну та інші параметри. Крім того, платформа пропонує різні фільтри для пошуку, такі як

рейтинг користувача, вид транспорту, досвід водіння та інші фактори, які допомагають знайти найбільш підходящий варіант для поїздки.

CarpoolWorld є глобальною платформою і доступна в більшості країн світу, включаючи Україну. Проте, популярність CarpoolWorld в Україні не така велика порівняно з іншими країнами, такими як США, Іспанія, Німеччина та Франція.

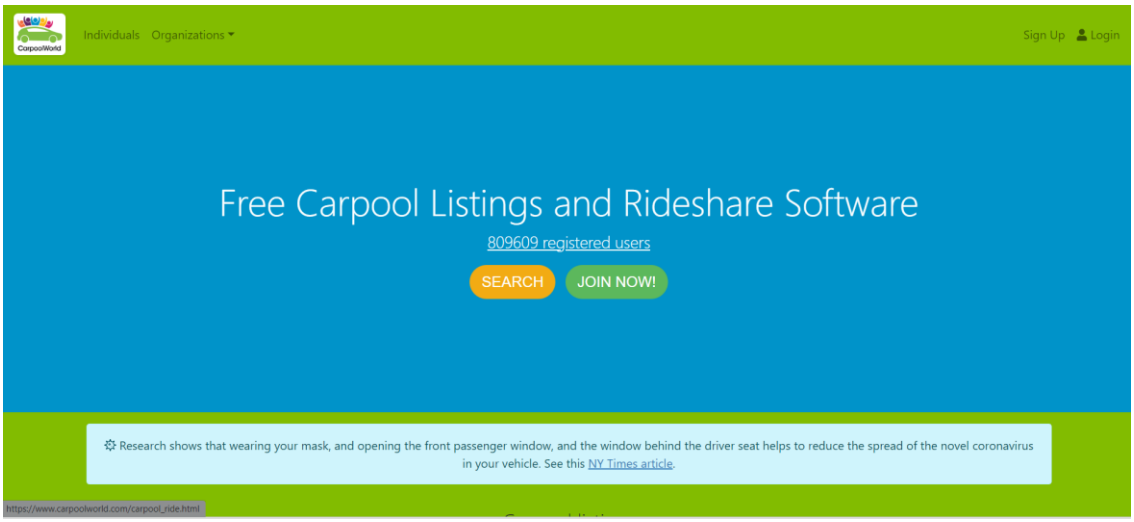

Рисунок 1.3 – Інтерфейс головної вебсторінки CarpoolWorld

CarpoolWorld пропонує користувачам різні варіанти комунікації між водіями та пасажирами, такі як електронна пошта, телефон та інші засоби зв'язку, що дозволяють зручно та швидко узгоджувати деталі поїздки. Крім того, платформа пропонує інформацію про досвід користувачів, яка допомагає зробити відповідний вибір [18].

1.3.3 Uber

Uber – це мобільний застосунок, який надає послуги по пошуку та замовленню транспорту. Застосунок дозволяє користувачам знайти доступний транспорт у своїй області та замовити його на смартфон (рис. 1.4).

Основна функціональність застосунку Uber полягає в тому, що користувач може замовити транспорт просто натиснувши на кнопку в застосунку. Після цього замовлення передається водієві, який найближчим часом з’явиться на місці зустрічі. Користувач може вибрати з різних типів транспорту, таких як економ, стандарт або преміум.

Застосунок Uber працює на базі геолокації, тобто користувач може знайти доступний транспорт у своєму районі. Також застосунок надає інформацію про водія, який здійснює поїздку, включаючи ім’я, номер автомобіля та оцінку його рейтингу [19].

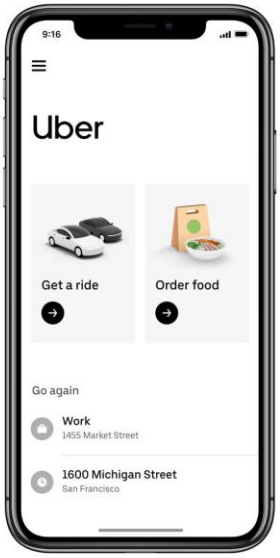

Рисунок 1.4 – Інтерфейс мобільного застосунку Uber

Хоча Uber не є спеціалізованою платформою для пошуку попутників та водіїв, вона може бути дуже зручною для коротких відстаней. Застосунок Uber дозволяє замовити автомобіль та дістатися до певного місця в будь-який час доби. Крім того, він забезпечує безпеку користувачів завдяки ідентифікації водіїв та рейтингуванню їх послуг.

Зокрема, у порівнянні з іншими платформами Uber має досить велику базу водіїв, що забезпечує можливість знайти вільний автомобіль навіть у години пік. Крім того, застосунок дозволяє користувачам вибирати тип

автомобіля, що підходить для конкретної поїздки, враховуючи кількість пасажирів та кількість багажу.

Також варто відмітити, що в застосунку Uber є можливість використовувати бонуси та промокоди для знижок на поїздки. Крім того, платіж за послуги відбувається безготівково, що забезпечує зручність та безпеку користувачів.

1.4 Постановка задачі

Таким чином, створення онлайн платформи для пошуку попутників та водіїв є актуальною задачею для полегшення транспортної мобільності та зменшення витрат на перевезення.

Об’єктом роботи онлайн-сервіс з пошуку пасажирів та водіїв за обраним маршрутом.

Метою даної кваліфікаційної роботи є розробка онлайн-сервісу, що призначений для легкого та ефективного знаходження попутників для подорожей за одним напрямком, а також водіїв, які можуть забрати пасажирів по дорозі до місця їх призначення.

Для досягнення мети необхідно вирішити такі завдання:

– розробити користувацький інтерфейс для взаємодії з платформою, з можливістю створення профілю користувача та публікації оголошень про поїздки;

– розробити схему бази даних для зберігання інформації про користувачів та їх оголошення;

– реалізувати функціональність пошуку попутників та водіїв;

– розробити систему прийняття та відхилу попутників.

2 ПРОЄКТУВАННЯ СИСТЕМИ, КОМПОНЕНТІВ ТА БАЗИ ДАНИХ

2.1 Проєктування системи

Проєктування системи – це процес розробки та створення структури, компонентів і функціональності системи з метою вирішення конкретної задачі або задоволення певних потреб. Це важлива фаза в розробці будь-якого інформаційного проєкту, оскільки вона визначає основні принципи, на яких буде базуватися система.

Процес проєктування системи зазвичай включає наступні етапи:

– аналіз вимог: визначення потреб користувачів та вимог до системи. Цей етап включає збір і аналіз вимог, встановлення функціональності та нефункціональних вимог;

– проєктування архітектури: розробка високорівневої архітектури системи, включаючи розподіл компонентів, взаємозв’язки та інтерфейси. Це визначає загальну структуру та організацію системи;

– проєктування компонентів: розробка детального опису кожного компонента системи, включаючи їх функціональність та інтерфейси. Це визначає, як кожен компонент буде виконувати свої завдання та взаємодіяти з іншими компнентами;

– проєктування бази даних: визначення структури та організації бази даних, включаючи таблиці, зв’язки та правила цілісності. Це важливий аспект для збереження та управління даними в системі;

– проєктування інтерфейсу: розробка користувацького інтерфейсу системи, зокрема його організація, навігація та взаємодія з користувачем. Важливо враховувати простоту використання та зручність для користувача;

– тестування та валідація: перевірка та валідація системи з метою переконання в її відповідності вимогам та виявлення можливих помилок або недоліків. Цей етап включає проведення функціональних, інтеграційних та

системних тестів, а також залучення користувачів для здійснення оцінки та збору фідбеку.

Проєктування системи є важким завданням, оскільки вимагає комплексного підходу та розуміння потреб користувачів, технічних обмежень, а також знання принципів дизайну та архітектури. Воно вимагає глибокого аналізу, творчості та розробки оптимальних рішень для досягнення поставлених цілей проєкту.

Крім того, проєктування системи також вимагає врахування майбутнього розвитку та масштабування системи, щоб забезпечити її ефективність та гнучкість. Також важливо забезпечити безпеку даних та захист системи від потенційних загроз [20].

2.2 Аналіз вимог

Аналіз вимог системи є ключовим етапом у розробці будь-якого проєкту. Він дозволяє зрозуміти потреби та очікування користувачів і визначити, які функціональні та нефункціональні вимоги повинна виконувати система. Нефункціональні вимоги відіграють важливу роль у визначенні якості, безпеки, ефективності та надійності системи.

Таким чином, приступаючи до аналізу вимог для даної системи пошуку попутників та водіїв, необхідно врахувати потреби користувачів, їх очікування та вимоги до функціональності.

Функціональні вимоги:

– реєстрація користувачів: система повинна мати можливість реєстрації нових користувачів, які будуть використовувати систему для пошуку попутників або водіїв;

– створення та перегляд маршрутів: користувачі повинні мати змогу створювати маршрути з усіма необхідними деталями;

– пошук попутників або водіїв: система повинна надавати можливість знаходити попутників або водіїв, які подорожують по тому ж маршруту та в той же час;

– комунікація між користувачами: система має надавати засоби комунікації між користувачами для обговорення деталей поїздки та узгодження умов;

– історія поїздок: система має зберігати історію поїздок для кожного користувача, що дозволить їм відстежувати раніше здійснені поїздки та деталі;

– список майбутніх поїздок: система має давати змогу отримати список майбутніх зарезервованих поїздок з подальшою їх відміною.

Нефункціональні вимоги:

– безпека: система повинна забезпечувати безпеку даних користувачів та їх конфіденційність, здійснюючи захист інформації та використовуючи механізми аутентифікації та авторизації;

– інтуїтивність: користувачі повинні легко орієнтуватися в системі, мати доступ до необхідних функцій та зручні інструменти для пошуку та взаємодії з іншими користувачами;

– адаптивність: система повинна бути доступною на різних пристроях, включаючи комп’ютери, планшети та мобільні телефони, забезпечуючи коректне відображення та функціонал;

– надійність: система повинна бути доступною та захищеною для користувачів у будь-який час, мінімізуючи періоди перерваності для планового обслуговування або непередбачуваних ситуацій.

Аналіз функціональних та нефункціональних вимог є важливим етапом у розробці системи пошуку попутників та водіїв. Функціональні вимоги допомагають визначити, які конкретні функції та можливості повинна мати система для задоволення потреб користувачів. Вони визначають, які дії

можуть бути виконані в системі і як вона повинна поводитись у різних ситуаціях.

Нефункціональні вимоги визначають якість та характеристики системи, такі як продуктивність, безпека, масштабованість та інші. Вони впливають на загальний досвід користувача та ефективність системи. Аналіз нефункціональних вимог допомагає врахувати ці аспекти і розробити систему, яка відповідає стандартам якості та вимогам користувачів.

В результаті аналізу отримаємо чітку картину того, яким повинен бути функціонал і характеристики даної системи. Це дозволить нам ефективно розробити та впровадити систему пошуку попутників та водіїв, що задовольняє потреби та очікування користувачів, забезпечує високу якість, безпеку та зручність використання.

2.3 Архітектура системи

Проєктування архітектури системи для даного онлайн-сервісу пошуку попутників та водіїв є важливим етапом, що включає в себе розробку структури, компонентів та взаємодії між ними. Враховуючи описані сторінки та їх функціональність, можемо розглянути загальну структуру архітектури системи.

Архітектура даної системи може бути побудована на основі клієнт-серверної моделі, де вебсайт виступає як клієнт, а сервер є основним компонентом системи, що забезпечує обробку запитів, зберігання даних та взаємодію з базою даних. Розглянемо основні компоненти архітектури.

Клієнтська частина:

– фронтенд: використання React разом з Next.js для побудови користувацького інтерфейсу. Next.js надасть можливість серверного рендерингу (SSR) та статичну генерацію (SSG) сторінок для покращення продуктивності та SEO;

– бібліотека компонентів: використання Material-UI для швидкої розробки стилізованих інтерфейсів. Material-UI надає багато готових компонентів та можливостей для стилізації;

– керування станом: використання бібліотеки React-Query для керування станом даних та управління запитами до сервера. React-Query дозволяє ефективно кешувати дані та автоматично оновлювати їх при необхідності;

– валідація даних: використання бібліотеки zod для валідації та типізації даних на клієнті. Zod надає зручні засоби для валідації форм та перевірки правильності даних.

Серверна частина:

– бекенд: використання Node.js для побудови серверної частини застосунку;

– вебфреймворк: використання Express для реалізації вебсервера та обробки запитів;

– база даних: використання MongoDB для зберігання даних. Можна використовувати офіційний драйвер MongoDB або ORM, такий як Mongoose, для спрощення роботи з базою даних.

Комунікація між клієнтом та сервером:

– REST або GraphQL API: реалізація API для взаємодії між клієнтом та сервером. Можна обрати REST або GraphQL залежно від потреб проєкту. REST надає стандартні API-endpoints для взаємодії, тоді як GraphQL дозволяє клієнту запитувати саме ті дані, які йому потрібні;

– захист API: застосування механізмів авторизації та контролю доступу до API для забезпечення безпеки та захисту даних користувачів.

Інші технології:

– типізація: використання Typescript для покращення роботи коду;

– кешування: використання механізмів кешування для збереження та та швидкого доступу;

– логування: використання системи логування, наприклад, Winston або Bunyan, для запису подій та помилок у застосунку. Це допоможе відстежувати та аналізувати проблеми в системі;

– тестування: використання фреймворку для тестування, такого як Jest або Mocha, для написання автоматизованих тестів, які перевірятимуть функціональність та стабільність системи.

У цілому, архітектура системи спроєктована таким чином, щоб забезпечити ефективну роботу, масштабованість та зручність для користувачів. Використання сучасних технологій та практик дозволяє розробити стабільну та функціональну систему, яка зможе задовольнити потреби користувачів у пошуку та бронюванні поїздок.

2.4 Опис сторінок та компонентів системи

Сторінки – це основний компонент застосунку або вебсайту, який представляє окремі екрани або візуальні інтерфейси для взаємодії з користувачем. Кожна сторінка має свій власний URL-адресу, на якому вона доступна, і вона відображає конкретний контент або функціональність.

Створення окремої сторінки для кожної функції або завдання дозволяє забезпечити логічний розподіл та структуру застосунку. Кожна сторінка може мати власну компоновку, дизайн, взаємодію з користувачем та функціональність, яка відповідає конкретному завданню або потребі користувача.

Створення сторінок дозволяє організувати інформацію та функціональність застосунку в логічну структуру, що полегшує навігацію та використання для користувачів. Користувачі можуть переходити з однієї сторінки на іншу, виконувати дії, вводити дані та отримувати результати на кожній зі сторінок.

Створення ефективних та зручних сторінок включає в себе проєктування візуального дизайну, розміщення елементів, взаємодію з користувачем, обробку даних та передачу інформації між сторінками. Компоненти, такі як кнопки, форми, таблиці, списки тощо, використовуються для створення інтерфейсу на кожній сторінці [21].

Зважаючи на опис функціональних та нефункціональних вимог проєкту, можна створити систему з наступними компонентами: «Логін», «Реєстрація», «Пошук поїздок», «Мої бронювання» та «Мої створені поїздки».

На сторінці «Логін» користувач може ввести свою електронну пошту та пароль, після чого система перевіряє введені дані та здійснює аутентифікацію користувача.

У разі успішного входу, користувач перенаправляється на сторінку пошуку поїздок. Ця сторінка створена для того, щоб користувачі могли ввійти до системи шляхом введення своїх облікових даних. Вона перевіряє достовірність введених даних і надає доступ до особистого облікового запису.

Сторінка «Реєстрація» містить поля для введення імені, прізвища, номера телефону, електронної пошти та пароля. Введені дані перевіряються на валідність та зберігаються в базі даних.

Після успішної реєстрації користувач перенаправляється на сторінку «Логін». На цій сторінці нові користувачі можуть створити обліковий запис в системі. Вона дозволяє користувачам ввести свої особисті дані та створити новий обліковий запис.

Це необхідно для ідентифікації користувача і забезпечення доступу до функціональності системи.

На сторінці «Пошук поїздок» користувач може вибрати місце відправлення, місце прибуття та дату поїздки. Після введення параметрів, система надсилає запит до сервера для отримання списку доступних поїздок. Результати відображаються на сторінці з можливістю фільтрації та

сортування. Користувач може здійснити бронювання вибраної поїздки, що призводить до оновлення списку доступних поїздок. Ця сторінка надає користувачам можливість шукати доступні поїздки за заданими критеріями, такими як місце відправлення, місце прибуття та дата.

Вона відображає список доступних поїздок, і користувачі можуть використовувати фільтри для обмеження результатів за ціною та кількістю доступних місць.

Крім того, на цій сторінці можна бронювати обрану поїздку.

Сторінка «Мої бронювання» відображає список майбутніх бронювань користувача. Для кожного бронювання відображаються деталі, такі як місце відправлення, місце прибуття, дата та час.

Ця сторінка дозволяє користувачам переглядати та керувати своїми майбутніми бронюваннями.

Попутчик може скасувати бронювання, що також оновлює список бронювань.

На сторінці «Мої створені поїздки» відображається список створених користувачем поїздок.

Кожна поїздка має свої деталі, які відображаються, такі як місце відправлення, місце прибуття, дата та час.

Користувач може переглянути деталі кожної створеної поїздки. Користувачі також можуть виконувати дії зі своїми створеними поїздками, наприклад, редагувати деталі поїздки або видаляти їх.

Сторінка «Створення поїздки» дозволяє користувачам створювати нові поїздки. Вона містить форму, в якій користувачі можуть ввести деталі поїздки, такі як місце відправлення, місце прибуття, дату, час та ціну. Користувачі можуть заповнити форму і створити нову поїздку у системі.

Кожна сторінка має свою унікальну функціональність і надає користувачам можливість виконувати певні дії та отримувати необхідну інформацію. Разом вони створюють повноцінну систему для керування поїздками, реєстрації, аутентифікації та перегляду даних.

2.5 Проєктування бази даних

База даних є центральною складовою будь-якої інформаційної системи. Вона зберігає структуровані дані, які використовуються для збереження, організації та отримання інформації. Бази даних використовуються для збереження великих обсягів даних і забезпечення швидкого та ефективного доступу до них.

У даній системі планується використовувати MongoDB. MongoDB є документ-орієнтованою базою даних, яка зберігає дані у вигляді JSON-подібних документів. Це гнучка та масштабована база даних, яка дозволяє зберігати структуровану і неструктуровану інформацію.

MongoDB дозволяє нам зберігати дані про користувачів, їх реєстраційні дані, поїздки, бронювання та інші важливі дані, необхідні для роботи даної системи. Вона підтримує швидкий доступ до даних, запити і індексацію, що дозволяє ефективно виконувати операції з базою даних.

Крім того, використання MongoDB дозволяє нам працювати з гнучкою схемою даних. Можна додавати нові поля до документів без необхідності зміни всієї структури бази даних. Це дає нам можливість легко розширювати функціональність системи і адаптуватися до змін вимог.

Загалом, база даних MongoDB допомагає нам ефективно зберігати і керувати даними в даній системі, забезпечуючи швидкий доступ та гнучкість в роботі з інформацією [22].

MongoDB є частиною стеку MERN, який є вкрай популярним на даний момент. MERN є скороченням від Mongo, Express, React, Node, що позначає стек технологій, який використовується у розробці застосунків. Кожна з цих технологій виконує важливу роль у створенні повноцінного застосунку.

MongoDB є базою даних, яка використовується для зберігання структурованих даних у вигляді документів JSON-подібного формату. Вона є основою для зберігання, отримання та керування даними у даній системі.

Express є фреймворком для розробки серверної частини застосунків на Node.js. Він допомагає встановити маршрутизацію, обробку запитів та створення API для взаємодії з базою даних та передачі даних між клієнтом і сервером.

React є бібліотекою для розробки користувацького інтерфейсу на клієнтській стороні. Він дозволяє створювати ефективні, інтерактивні та модульні користувацькі інтерфейси з використанням компонентного підходу. React робить дану систему більш динамічною та відзивчивою, забезпечуючи багатофункціональний та зручний інтерфейс для користувачів.

Node.js є середовищем виконання JavaScript, яке дозволяє запускати JavaScript-код на сервері. Він дозволяє створювати серверну частину застосунків, обробляти запити, взаємодіяти з базою даних та забезпечувати функціональність, необхідну для роботи даної системи.

Використання стеку MERN дозволяє створити повноцінний застосунок, в якому можливо взаємодіяти з базою даних, обробляти запити, створювати динамічний інтерфейс та забезпечувати потрібну функціональності для користувачів системи. MongoDB забезпечує нам гнучкість та швидкий доступ до даних, Express допомагає створити потрібні маршрути та API для обробки запитів, React дає змогу створювати інтерактивні та зручні користувацькі інтерфейси, а Node.js забезпечує зв’язок між фронтендом та базою даних, обробку запитів та виконання логіки на серверній стороні.

Використання MERN стеку має свої переваги, такі як швидкий розробка, однорідність технологій, що спрощує співпрацю розробників, розширюваність і масштабованість системи. Комбінація цих технологій дозволяє нам створити потужний та ефективний застосунок, який задовольняє потреби користувачів та забезпечує зручну та надійну роботу з даними [23, 24].

Зараз, після ознайомлення з MongoDb, давайте перейдемо до схеми бази даних даного онлайн-сервісу. У системі буде використовуватися база даних MongoDB для зберігання і управління інформацією про поїздки та

користувачів. Загалом будемо мати дві колекції (таблиці) – «travels» і «users». Кожна з них представляє певний тип даних і використовується на різних сторінках системи.

Колекція «travels» відповідає за збереження інформації про подорожі, водіїв і пасажирів. Вона містить дані про кожну окрему поїздку, такі як місце відправки (departure), час відправлення (departureTime), місце прибуття (destination), час прибуття (arrivingTime), доступні місця (availableSeats), ціну (price) і ідентифікатор водія (driverId). Крім того, у колекції «travels» зберігається масив пасажирів (passengers), який включає ідентифікатори пасажирів та їх статуси поїздки. Ця інформація використовується на сторінках пошуку поїздок, бронювання поїздок, а також на сторінках моїх бронювань та моїх створених поїздок. Детальну схему даної колекції показано у таблиці 2.1.

Таблиця 2.1 – Схема колекції Подорожі (travels)

Зміст поля	**Назва Поля**	**Тип даних**	**Нотатки**
Ідентифікатор подорожі	_id	ObjectId	
Місце відправлення	departure	string	
Час відправлення	departureTime	string	
Місце прибуття	destination	string	
Час прибуття	arrivingTime	string	
Доступні місця	availableSeats	int	
Ціна поїздки	price	string	
Пасажири	passengers	Array	Містить в собі інформацію про статус пасажирів подорожі
Ідентифікатор водія	driverId	ObjectId	

Поле Пасажири (passengers) є масивом і включає у собі певну інформацію з приводу кількості пасажирів та їхніх статусів. Огляд цього поля можна побачити у таблиці 2.2.

Таблиця 2.2 – Схема поля Пасажири (passengers)

Зміст поля	**Назва Поля**	**Тип даних**	**Нотатки**
Ідентифікатор користувача	userId	ObjectId	
Статус пасажира у поїздці	status	string	Може мати три стана – прийнятий, відмовлений, новий

Колекція «users» містить дані про користувачів системи. Вона включає особисті дані кожного користувача, такі як ім’я (name), прізвище (surname), телефон (phone), електронна пошта (email) і зашифрований пароль (password). Ця інформація використовується на сторінках реєстрації, входу в систему та профілю користувача. Крім того, ідентифікатор користувача (user_id) з колекції «users» використовується для пов’язання зі сторінками моїх бронювань та моїх створених поїздок, де показується відповідна інформація про поїздки, що належать користувачеві. Детальну схему даної колекції показано у таблиці 2.3.

Таблиця 2.3 – Схема колекції Користувачі (users)

Зміст поля	**Назва Поля**	**Тип даних**	**Нотатки**
Ідентифікатор користувача	_id	ObjectId	
Ім’я	name	string	
Прізвище	surname	string	
Номер телефону	phone	string	
E-mail	email	string	
Пароль	password	string	Зберігається у зашифрованому вигляді

Отже, дві колекції бази даних – «travels» і «users» – мають різні типи даних і використовуються на різних сторінках системи. Колекція «travels» містить інформацію про поїздки і використовується на сторінках пошуку поїздок, бронювання поїздок, моїх бронювань та моїх створених поїздок. Колекція «users» містить дані про користувачів і використовується на сторінках реєстрації, входу в систему та профілю користувача.

Ці дві колекції пов’язані між собою за допомогою ідентифікатора водія (driverId) в колекції «travels» і ідентифікатора користувача (user_id) в колекції «users». Це дозволяє встановити зв’язок між конкретною поїздкою і відповідним водієм, а також зв’язати користувача з його створеними та забронюваними поїздками.

Загалом, використання двох колекцій дозволяє ефективно зберігати та організовувати дані про подорожі, водіїв і користувачів, а також забезпечує зв’язок між ними для правильного відображення та взаємодії на сторінках системи.

2.6 Мапа сайту

Карта сайту – це візуальне представлення структури вебсайту. Вона показує ієрархічні зв’язки між різними сторінками або розділами сайту. Карта сайту допомагає організувати і систематизувати вміст, навігацію та структуру вебсайту.

Зазвичай карта сайту відображає сторінки на рівні дерева, де головна сторінка є коренем, а наступні сторінки розміщуються як гілки чи листочки. Це дозволяє візуально представити розташування сторінок і їх взаємозв’язки.

Мапа сайту допомагає розуміти загальну структуру сайту, забезпечує зручну навігацію для користувачів і полегшує роботу з розробниками і дизайнерами при створенні і підтримці вебсайту [25].

У даному проєкті також була підготовлена карту сайту, яка допоможе нам краще зорієнтуватись у структурі даної системи. Карта сайту включає всі основні сторінки та функціональні елементи, що присутні на вебсайті. Вона надає нам відображення загального зображення системи та розташування її компонентів.

Подальше представлення карти сайту дозволить нам крок за кроком розглянути кожну сторінку, її функціональність та взаємозв'язки з іншими сторінками. Це допоможе нам краще зрозуміти внутрішню логіку системи та взаємодію між її компонентами.

Давайте разом розглянемо створену карту сайту та поглибимося в деталі кожної сторінки та її функціоналу на рисунку 2.1.

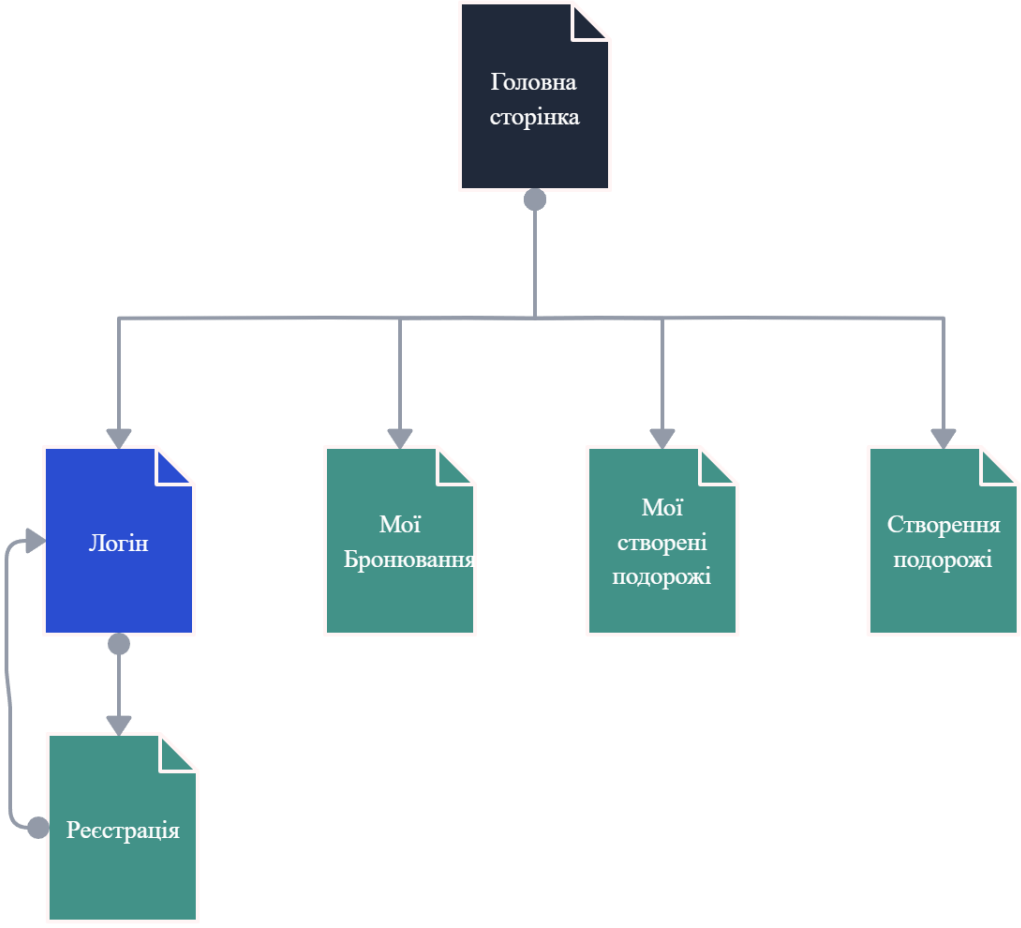

Рисунок 2.1 – Мапа сайту із основними компонентами системи

Головна сторінка служить входом до основного функціоналу сайту і надає користувачу можливість здійснювати багато різноманітних дій. Якщо користувач ще не зареєстрований, то він без великих зусиль може перейти на сторінку реєстрації та логіну до вебсайту. Створення нового аккаунту проходить доволі швидко. Після цього пасажир з легкістю може знайти поїздку увівши тільки місце відправлення та місце прибуття, а також обрати дату, щоб знайти доступні поїздки за цими критеріями. Також, на головній сторінці можуть відображатися рекомендовані поїздки або акційні пропозиції для привернення уваги користувачів.

Після виконання певних дій на головній сторінці, користувач може перейти на інші функціональні сторінки. При кліці на конкретну поїздку з результатів пошуку, користувач переходить на сторінку з детальною інформацією про поїздку, таку як місце відправлення, місце прибуття, час відправлення та прибуття, кількість доступних місць та ціна. Тут користувач може також забронювати поїздку, якщо є вільні місця.

На сторінці моїх бронювань користувач може переглянути свої майбутні бронювання, включаючи дату, час, місце відправлення та прибуття, а також скасувати його при необхідності.

На сторінці моїх створених поїздок користувач має інформацію про поїздки, які він створив. Компонент містить усі деталі поїздки.

Важливо зазначити, що структура сайту може бути налаштована відповідно до конкретних вимог та потреб користувачів, тому список сторінок та їх функціонал може бути змінений.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ОНЛАЙН-СЕРВІСУ З ПОШУКУ ПОПУТНИКІВ І ВОДІЇВ

3.1 Опис програмної реалізації

У рамках кваліфікаційної роботи був розроблений онлайн-сервіс для пошуку попутників та водіїв за заданим маршрутом. Для реалізації було обране середовище IntelliJ IDEA, а мовою програмної реалізації – Typescript. За основу програмної реалізації були взяті уже згадані раніше технології – React, Node.js, Next, React-Query та інші.

3.1.1 Створення серверної частини та API

Для створення серверної частини та API було обрано середовище Node.js, яке дозволяє створювати високопродуктивні серверні застосунки. Для забезпечення роботи з даними була використана база даних MongoDB, яка забезпечує швидку та ефективну роботу з даними. Для створення API була використана технологія REST API, яка дозволяє передавати дані між клієнтом та сервером в стандартному форматі JSON.

База даних є одним із основних складових будь-якого сервера. MongoDB є однією з найбільш популярних NoSQL баз даних, яка забезпечує гнучкість та швидкість при роботі з даними. У даному випадку, MongoDB розташована не на локальному комп’ютері, а віддалено на хмарній платформі MongoDB Atlas. Для підключення до бази використовується рядок підключення – MONGO_URI, який містить необхідні дані для аутентифікації та підключення до бази даних.

MongoDB Atlas – це повністю керований хмарний сервіс бази даних MongoDB, який дозволяє легко створювати та управляти кластерами

MongoDB в хмарному середовищі. Він забезпечує швидкий доступ до баз даних MongoDB, що дозволяє ефективно використовувати ресурси та зменшує ризик виникнення проблем з безпекою та доступністю.

Окрім бази даних, в проєкті використовується API-сервіс OpenCage, який забезпечує геокодування та зворотне геокодування.

Для використання цього сервісу також потрібен API-ключ, який зберігається як змінна оточення – OPENCAGE_API_KEY.

Аутентифікація та авторизація є важливою частиною більшості застосунків та сервісів. Аутентифікація означає перевірку ідентифікації користувача, тобто підтвердження того, що користувач є тим, за кого він себе вважає. Авторизація визначає, які дії має дозволено виконувати аутентифікованому користувачу в залежності від його ролі та прав доступу.

NextAuth – це бібліотека для Node.js, яка дозволяє швидко додати аутентифікацію та авторизацію до застосунків. Вона підтримує багато провайдерів авторизації, таких як Google, Facebook, Twitter, GitHub, а також дозволяє створювати власні провайдери. NextAuth забезпечує безпеку за допомогою JWT-токенів, які можуть бути використані для авторизації запитів до сервера та для зберігання даних про користувача на клієнтському пристрої.

Для використання NextAuth потрібно встановити його на сервері та на клієнтській стороні, налаштувати провайдерів авторизації та обробники авторизації, які дозволяють зберігати та отримувати інформацію про користувача. NextAuth також дозволяє налаштувати різні ролі та дозволи доступу, щоб забезпечити безпеку застосунку.

У створеному онлайн-сервісі авторизація відбувається наступним чином:

Крок 1. Сервер під’єднується до бази даних та робить запит до колекції Користувачів.

Крок 2. Якщо користувач бажає зайти у вже створений обліковий запис, то сервер намагається знайти його логін-дані з-поміж інших. Якщо

логін та пароль сходяться, створюється нова сесія користувача, яка зберігає у собі JWT-токен.

Крок 3. Якщо користувач бажає створити новий обліковий запис, то сервер зробить запит до бази даних зі створенням нового користувача з усією наданою інформацією. Після успішної реєстрація, користувач має можливість спробувати функціональність описану у Кроці 2 (вхід у вже створений обліковий запис).

Крок 4. При кожному запиті на будь-яку сторінку, де потрібна авторизація, JWT-токен розбирається та витягується необхідна інформація для подальшого відображення онлайн сторінок.

Дуже важливо те, що пароль є зашифрованим і доступ до бази даних ніяким чином не надає шахраям цінні дані користувача.

На рисунку 3.1 можна побачити структуру файлів авторизації.

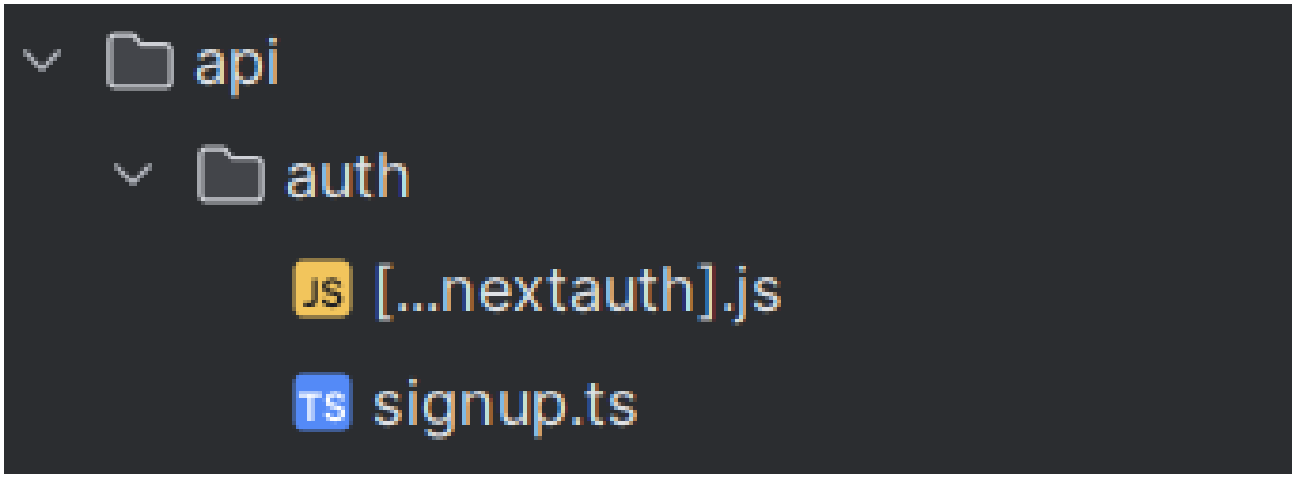

Рисунок 3.1 – Структура файлів авторизації

Файл […nexauth].js відповідає за вхід до вже створеного аккаунту користувача, в той час як signup.ts – за створення нового облікового запису.

Також, у кожному зі способів авторизації є дуже важлива валідація даних, яка запобігає створення нового аккаунту на вже зареєстровану електронну адресу, без пароля, тощо. На лістингу 3.1 можна побачити код щодо цих дій.

Лістинг 3.1 Валідація реєстрації користувача:

```
if (
  !email ||
  !email.includes("@") ||
  !password ||
  password.trim().length < 6
) {
  res.status(422).json({
    message:
      "Invalid input - password should also be at least 6 characters long.",
  });
  return;
}
```

Після успішної реєстрації та входу в обліковий запис, користувачу відкривається уся можлива функціональність онлайн-сервісу з пошуку попутників та водіїв за заданим маршрутом.

Розглянемо серверну частину реалізації даного сервісу і зокрема API-endpoint.

API (Application Programming Interface) – це інтерфейс, який дозволяє різним програмним застосункам спілкуватись між собою через запити, які надсилаються від клієнта до сервера і навпаки.

API забезпечує зручний та стандартизований спосіб взаємодії між різними застосунками та системами, що дозволяє ефективно використовувати ресурси та підвищує продуктивність програмного забезпечення та швидкість його використання.

На рисунку 3.2 можна побачити список усіх файлів, які відповідають за певний створений API-endpoint. Назва файлу відображає функціональність за яку відповідає API-endpoint.

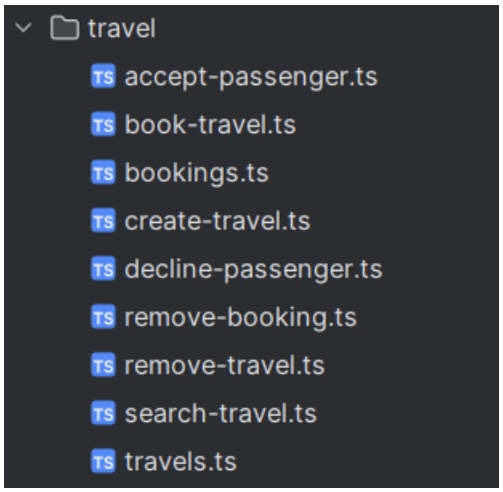

Рисунок 3.2 – Структура API-endpoint сервера

Розглянемо коротко про кожен з них у таблиці 3.1.

Таблиця 3.1 – Опис API-endpoint

Назва API-endpoint	**Назва файла**	**Опис**
1	**2**	**3**
Прийняти пасажира/попутника до поїздки	accept-passenger.ts	Відповідає за прийняття пасажира до поїздки, змінює статус замовлення з NEW до ACCEPTED.
Замовити поїздку	book-travel.ts	Відповідає за замовлення поїздки пасажиром, додає пасажира до кількості та надає можливість прийняти/відхилити пасажира із замовлення.
Замовлення	bookings.ts	Знаходить усі замовлення даного користувача.
Створити поїздку	create-travel.ts	Створює поїздку за необхідними критеріями заданими попередньо: місце відправлення/прибуття, дата та час відправлення/прибуття

Продовження таблиці 3.1

1	2	3
Відмовити пасажиру/попутнику у поїздці	decline-passenger.ts	Відповідає за відмову пасажира у поїздці, змінює статус замовлення з NEW, до DECLINED.
Видалити замовлення поїздки	remove-booking.ts	Відміняє замовлення з теперішніх замовлень користувача, водію більше не показується запит на нову поїздку
Видалити поїздку	remove-travel.ts	Видаляє повністю поїздку з усіма бронюваннями.
Знайти поїздку	search-travel.ts	Шукає поїздку за критеріями – місце відправлення/прибуття, дата та час відправлення/прибуття,
Поїздки	travels.ts	Знаходить усі поїздки/подорожі даного користувача.

Дані програмні інтерфейси активно використовуються компонентами та презентують основну функціональність вебсайту.

3.1.2 Створення компонентів сторінки

Після створення серверної частини та API, необхідно створити компоненти сторінки, щоб зробити застосунок функціональним та дружнім для користувача.

Компоненти сторінки – це невід’ємна частина застосунків та сайтів, що представляють інформацію в зручному для користувача вигляді. Компоненти

можуть бути різні: це можуть бути кнопки, поля вводу, меню, таблиці та багато іншого.

Для створення компонентів сторінки можемо використовувати бібліотеки, такі як React, який дозволяє створювати компоненти зі складових частин. У даному випадку, було використано Next.js, що є фреймворком для React, який надає додаткові можливості для розробки застосунків [26].

Next.js дозволяє створювати статичні та динамічні сторінки з мінімальними зусиллями, а також підтримує рендерінг на стороні сервера та на стороні клієнта. Це дозволяє підвищити продуктивність застосунку та забезпечити кращий досвід користувача.

Для зв’язку між компонентами та серверною частиною використовується API, що дозволяє передавати дані та отримувати відповіді з сервера. NextAuth був використаний для створення аутентифікації та авторизації, що дозволяє користувачам зареєструватися та увійти на сайт зі своїм обліковим записом.

На рисунку 3.3 можна побачити структуру створених компонентів даного онлайн-сервісу.

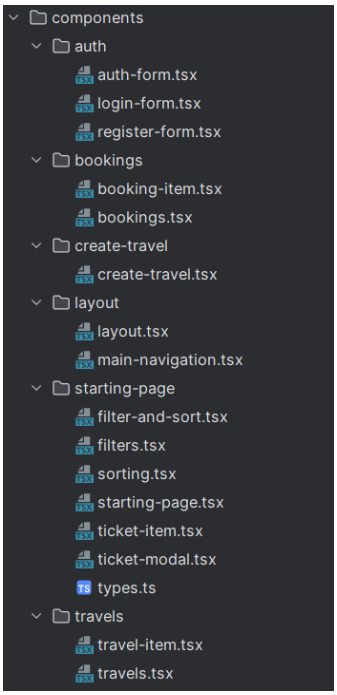

Рисунок 3.3 – Структура компонентів вебсайту

Перейдемо до опису кожного з компонентів. Компонент AuthForm представляє собою форму для аутентифікації або реєстрації користувача. В залежності від вибраного режиму (логін чи реєстрація), відображається відповідна форма. Кнопка «Create new account» або «Login with existing account» змінює режим форми. При відправленні форми відбувається аутентифікація або реєстрація за допомогою NextAuth або створення нового користувача через виклик функції createUser, яка робить запит на сервер за допомогою fetch. Коли форма відправлена, користувач буде перенаправлений на головну сторінку, якщо аутентифікація була успішною, або отримає повідомлення про помилку, якщо щось пішло не так. Компонент містить стилізацію за допомогою Material UI, яка дозволяє відображати форму з красивим дизайном.

Компонент LoginForm є формою входу користувача в систему. Він складається з двох введень, для електронної пошти та пароля, та кнопки для відправки форми. Коли користувач натискає на кнопку «Login», дані з форми перевіряються на правильність введення. Якщо дані введені некоректно, то виводиться помилка. Якщо дані введені коректно, то викликається функція «onSubmit», яка передає введені дані для подальшої обробки. Під час відправки форми кнопка стає неактивною та з’являється індикатор завантаження.

Компонент RegisterForm є формою реєстрації користувачів і містить наступні поля: ім’я, прізвище, телефон, електронна пошта, пароль та підтвердження пароля. Кожне поле є обов’язковим, а поле «електронна пошта» повинно мати правильний формат.

Для валідації використовується бібліотека zod. Щоб зробити валідацію, використовується зод-схема, що містить всі правила валідації для кожного поля. Компонент використовує React Hook Form для зберігання та обробки даних форми, включаючи управління полями за допомогою контроллера. Кнопка створення облікового запису є вимикачем із прогрес-баром, який відображається під час завантаження.

Компонент Bookings відповідає за відображення списку бронювань користувача. Перевіряється, чи користувач залогінений, і якщо так, то його електронна пошта передається в функцію getBookings, яка запитує бронювання за допомогою API. При отриманні даних про бронювання, вони зберігаються в стані компонента, і вони відображаються за допомогою компонента BookingItem, що імпортується з іншого файлу.

Компонент BookingItem – це компонент, який відповідає за відображення одного елемента бронювання (об’єкта типу Booking) на сторінці відображення списку бронювань. Він містить інформацію про маршрут (місце відправлення та місце прибуття), дати відправлення та прибуття, тривалість подорожі та статус бронювання.

Компонент приймає на вхід такі властивості:

– email – електронна адреса користувача, який зробив бронювання;

– booking – об’єкт типу Booking, який містить інформацію про бронювання;

– onCancel – функція-зворотний виклик, яка буде викликана при скасуванні бронювання.

Компонент відображає інформацію про маршрут, дати відправлення та прибуття, тривалість подорожі та статус бронювання відповідно до даних з об’єкта booking.

Також компонент містить кнопку «Cancel», яка дає можливість скасувати бронювання. Якщо бронювання скасоване, то замість кнопки «Cancel» відображається повідомлення про скасування.

Компонент CreateTravel це компонент, який містить форму створення нового маршруту подорожі. Компонент містить форму для заповнення даних, включаючи відправлення, час відправлення, прибуття, час прибуття, кількість доступних місць та ціну. Також у компоненті використовуються кастомні стилі, що дозволяють налаштовувати зовнішній вигляд компонента, а також функції для автодоповнення міст відповідно до введених даних.

MainNavigation – компонент навігаційної панелі вебсайту, який містить логотип сайту та кнопку користувача. Якщо користувач аутентифікувався, то на кнопці з'являється його ім'я та випадаюче меню з посиланнями на «My Bookings», «My Travels» та «Publish Travel», а також кнопку «Logout». Якщо користувач не аутентифікувався, то на кнопці з'являється іконка користувача та випадаюче меню з посиланням на сторінку «Login».

Компонент FilterAndSort є частиною проєкту і містить у собі фільтри та сортування для відображення списку товарів. Компонент складається з двох частин: «Filters» та «Sorting». «Filters» відповідає за встановлення фільтрів за ціною, а також кількістю місць для товарів, які можна відобразити в списку. «Sorting» дає можливість сортувати список товарів за ціною та тривалістю. Компонент є інтерактивним, тобто відображається на сторінці користувача та дозволяє вибирати фільтри та сортування для списку товарів.

Це компонент SearchTravel для пошуку подорожей у проєкті. Він складається з форми для вводу міст відправлення та прибуття, дати відправлення та кнопки пошуку. При введенні міста в поля вводу з'являється автозаповнення зі списком міст, які можна вибрати. Пошук відбувається після натискання на кнопку «Search», після чого компонент відображає список доступних подорожей, які можна фільтрувати та сортувати.

Компонент TicketItem використовується для відображення одного елемента списку квитків у проєкті. Він складається з наступних елементів:

– List Item Wrapper, який містить всі інші елементи компонента і відповідає за рамку та фон елемента;

– List Item Text, який містить інформацію про початковий та кінцевий пункти маршруту, дати відправлення та прибуття, тривалість та кількість вільних місць;

– Book Button, який дозволяє забронювати квиток;

– Ticket Modal, який відкривається при натисканні на кнопку.

Цей компонент отримує об’єкт подорожі та інформацію про статус авторизації користувача як вхідні параметри. За допомогою цих даних він відображає інформацію про квиток та дозволяє забронювати його.

Ticket Modal – є модальним вікном, яке показує інформацію про подорож. Компонент містить наступні елементи:

– зображення аватару з ініціалами користувача;

– назва маршруту та інформація про час відправлення та прибуття;

– вартість подорожі;

– кнопка «Забронювати».

Компонент Bookings є частиною проєкту і відповідає за відображення і управління подорожами користувача. Після входу в систему, якщо користувач вже забронював подорожі, вони будуть відображені в цьому компоненті. Крім того, користувач може видалити будь-яку зі своїх заброньованих подорожей. Якщо у користувача ще немає заброньованих подорожей, компонент відображає повідомлення про це, а також надає посилання на сторінку для створення нової подорожі.

Таким чином, компоненти сторінок допомагають створити зручний та легкий в інтерфейс для користувачів, спрощують процес розробки та роблять код більш масштабованим.

3.2 Сценарії роботи користувача з системою

Для досягнення максимальної ефективності та задоволення потреб користувачів, необхідно розробити чіткий та зручний інтерфейс системи, щоб користувачі могли з легкістю виконувати потрібні дії. Для цього використовуємо підхід User flow – це послідовність дій, які користувач виконує в системі з метою досягнення своїх цілей. Користувачі створеної системи матимуть можливість створювати нові подорожі, приєднуватись до

існуючих, знаходити водіїв та пасажирів, а також переглядати свої подорожі та профіль. У цьому розділі розглянемо сценарії, які користувачі можуть виконувати в системі та яким чином вони будуть досягати своїх цілей.

Детальний огляд користувацького потоку водія у системі зображено на рисунку 3.4.

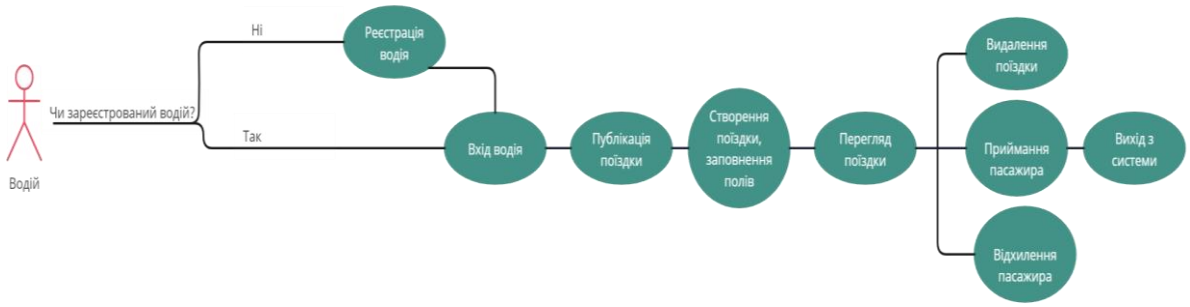

Рисунок 3.4 – Користувацький потік водія

Першою дією у сервісі для водія є створення акаунту або вхід до існуючого профілю. В залежності від того чи існує акаунт водія в системі, він буде перенаправлений на сторінку реєстраці, на якій він має заповнити всі необхідні данні. Після успішної реєстрації, водій перенаправлений на сторінку входу до системи. Після успішного входу до системи, водій переходить до кнопки «Публікація поїздки», яка дозволяє створювати нові поїздки. Водій натискає кнопку «Створити поїздку» і заповнює всі необхідні поля, які включають маршрут, дату та час відправлення, кількість місць, ціну тощо.

Після створення поїздки, водій може перейти на сторінку перегляду поїздки, де він може виконувати наступні дії: видалити поїздку, прийняти запит пасажира на поїздку, відхилити запит пасажира на поїздку. Останнім кроком для водія є вихід з системи, який здійснюється шляхом натискання кнопки «Вийти з системи».

Детальний огляд користувацького потоку пасажира у даній системі на рисунку 3.5.

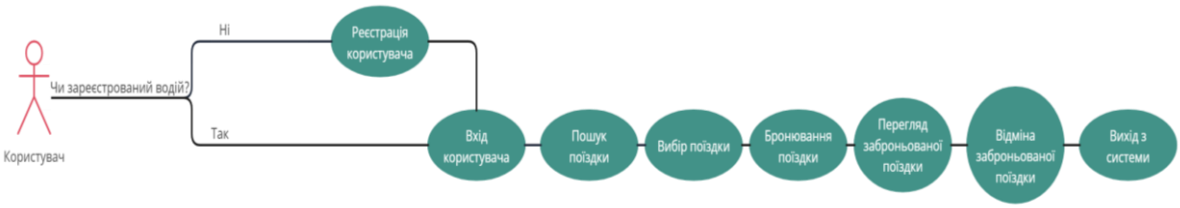

Рисунок 3.5 – Користувацький потік пасажира

Першим кроком пасажира після відкриття сервісу є або вхід або реєстрація, в залежності від того чи пасажир використовував сервіс. Якщо пасажир не зареєстрований, його перенаправляють на сторінку реєстрації. Мета реєстрації – дозволити пасажиру зареєструватися в системі та мати можливість використовувати всі її функції. Якщо пасажир вже зареєстрований, його перенаправляють на сторінку входу.

Це дозволяє користувачу увійти в систему та отримати доступ до функцій, які доступні тільки зареєстрованим клієнтам. Після входу, пасажир перенаправляється на сторінку пошуку поїздок. Де він обирає підходящу поїздку шляхом вибору дати, часу відправлення, маршруту та інших параметрів, щоб задовольнити свої потреби.

Метою цієї дії є надання пасажиру можливості знайти найбільш зручний та підходящий варіант поїздки. Після вибору потрібної поїздки, пасажир бронює її, це забезпечує йому місце в транспорті та надсилає сигнал до водія про бронювання. Після бронювання, пасажир може перейти на сторінку перегляду забронюваної поїздки, де він може переглянути данні про водія та поїздку. На сторінці перегляду забронюваної поїздки, пасажир має можливість відмінити поїздку, якщо виникли непередбачені обставини або він змінив свої плани. Останнім кроком потоку пасажира може бути вихід із системи.

Користувацький потік пасажира та водія допомагає зрозуміти послідовність дій, які користувачі будуть виконувати при взаємодії з

мобільним застосунком або вебзастосунком. Це важливий етап у процесі проєктування продукту, який дозволяє розробникам зрозуміти потреби користувачів і враховувати їх в процесі створення програмного забезпечення.

3.3 Огляд інтерфейсу

Інтерфейс – це засіб взаємодії між користувачем та системою, що дозволяє вводити дані в систему та отримувати від неї результати.

Інтерфейс може бути графічним, текстовим, голосовим або комбінованим.

Його основна мета – спростити та зробити зручним користування системою для користувача [27].

Розглянемо створений інтерфейс онлайн-сервісу з пошуку пасажирів та водіїв за заданим маршрутом.

Коли користувач заходе на сайт для пошуку попутників і водіїв, його зустрічає початкова сторінка (рис. 3.6), яка надає доступ до пошуку певної поїздки та вхід або реєстрацію облікового запису пасажира або водія.

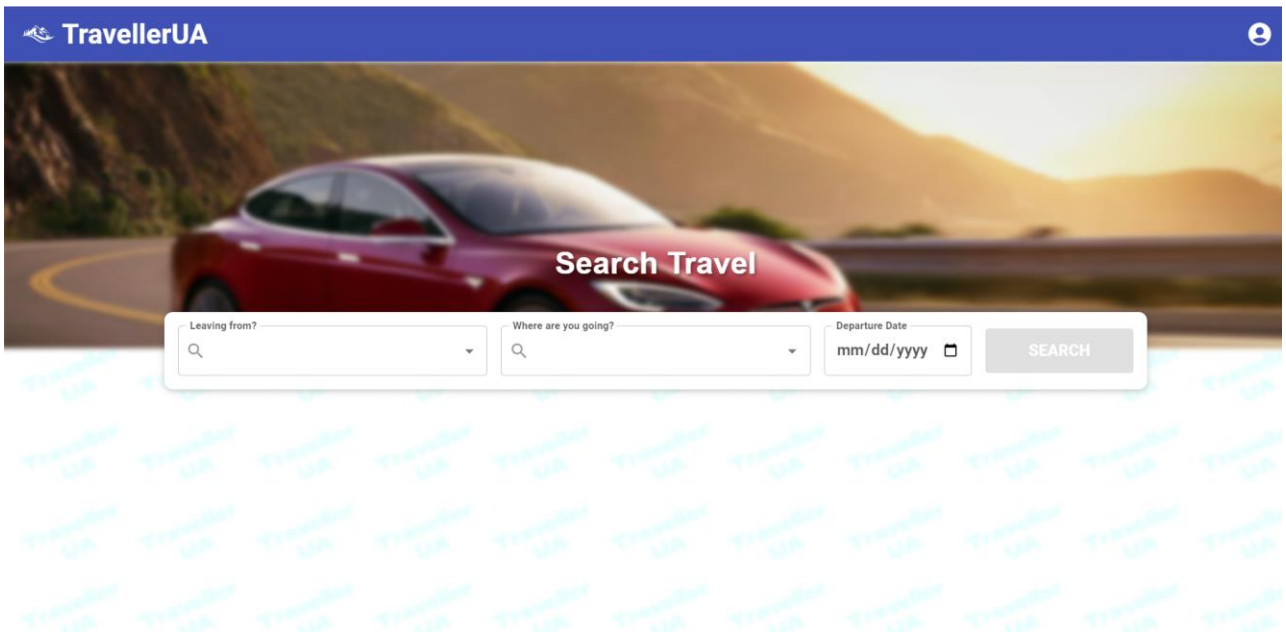

Рисунок 3.6 – Початкова сторінка користувача, що не здійснив вхід у систему

Для того щоб увійти у систему, користувач повинен натиснути на значок людини у правому верхньому углу. Для нього відкриється навігаційне меню з кнопкою «Login» (рис. 3.7), яка дозволяє перейти на сторінку входу (рис. 3.8).

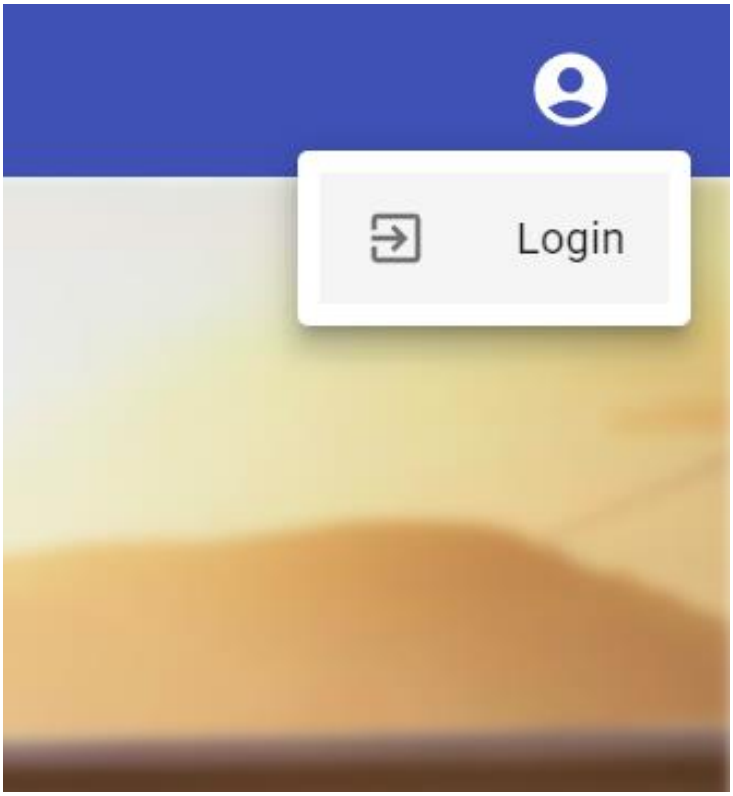

Рисунок 3.7 – Навігаційне меню користувача, що не здійснив вхід у систему

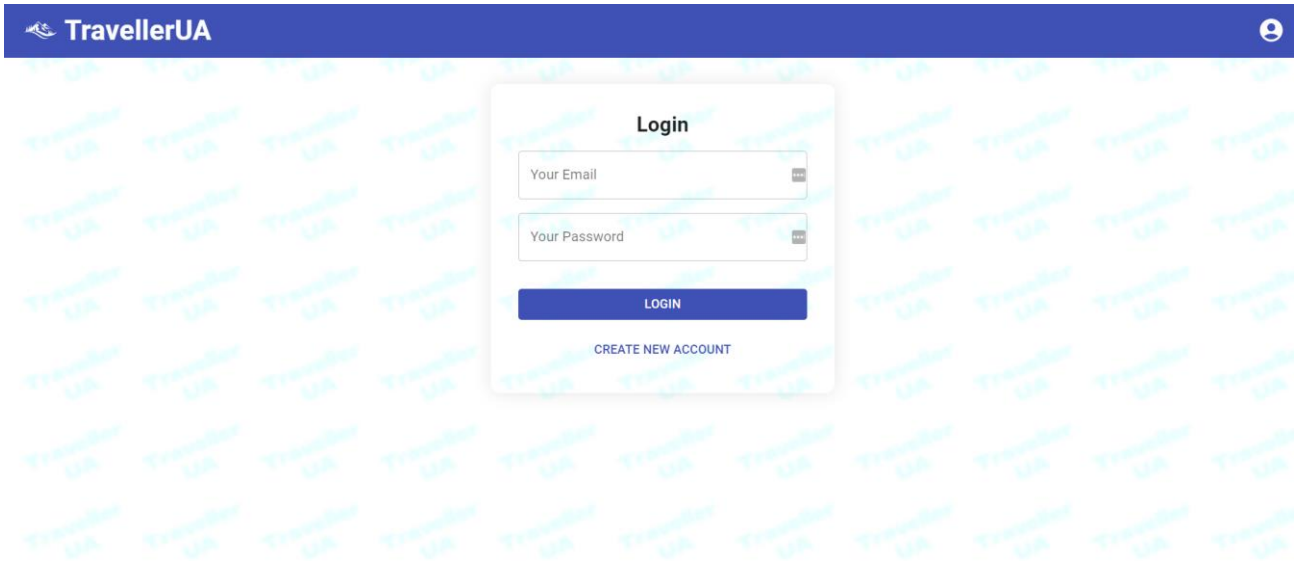

Рисунок 3.8 – Сторінка входу користувача

Після того як користувач потрапив на сторінку входу у нього з’являється дві опції – увійти у вже існуючий обліковий запис чи створити новий.

Для створення нового акаунту потрібно натиснути на кнопку «Create new account», яка перенаправе на сторінку реєстрації (рис. 3.9).

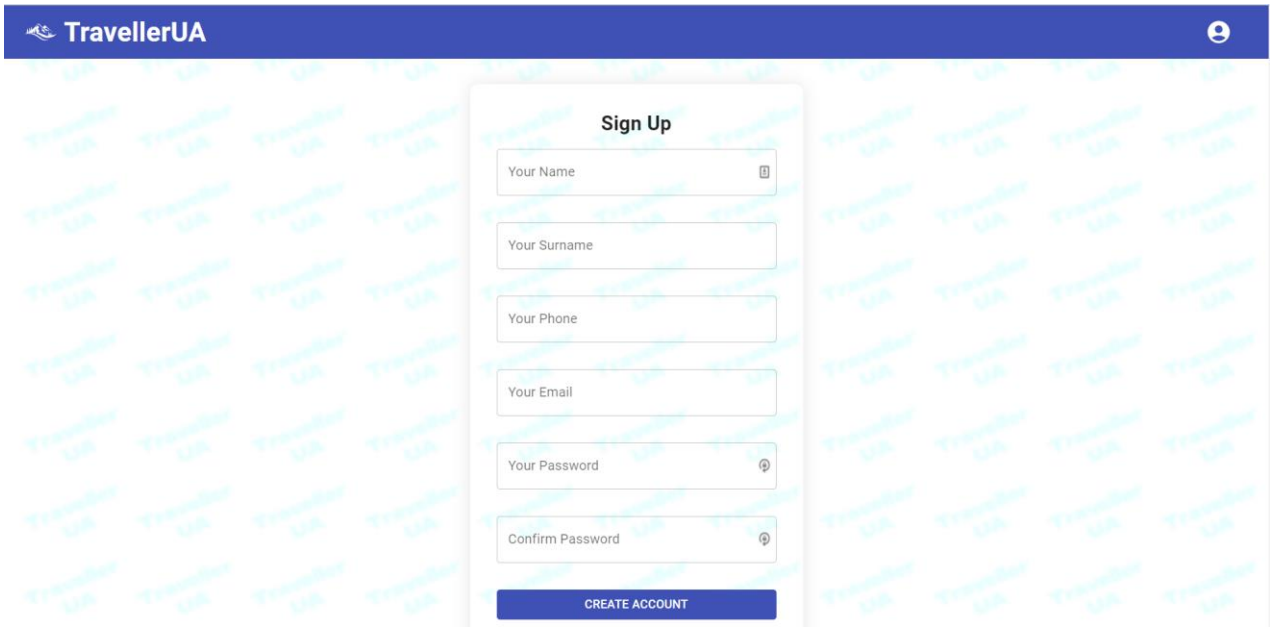

Рисунок 3.9 – Сторінка реєстрації користувача

Дана сторінка має 6 полів, які представляють собою базові реєстраційні дані, необхідні для подальшого комфортного користування даним онлайн-сервісом.

Ці поля включають у себе ім’я, прізвище, номер телефону, електронну адресу та пароль користувача, який повторюється двічі.

Після успішного заповнення даних, клієнта перенаправляє на сторінку входу, де він вводить вже знайомі йому дані.

Після успішного входу, користувачу доступний пошук поїздки, перегляд його зарезервованих поїздок, перегляд створених поїздок, створення нової поїздки та вихід з аккаунту.

Для доступу до більшості цих функцій використовується навігаційне меню, що показано на рисунку 3.10.

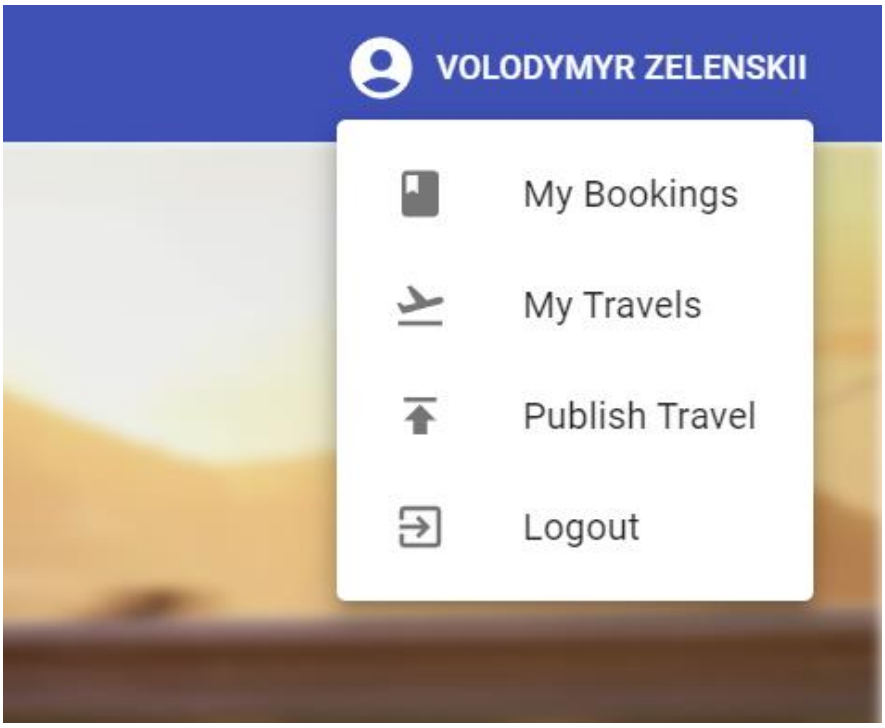

Рисунок 3.10 – Навігаційне меню користувача, що здійснив вхід у систему

Для того, щоб знайти потрібну поїздку, користувачу необхідно ввести дані місця відправлення та прибуття, та дату відправки.

На рисунку 3.11 показана сторінка результатів пошуку поїздки з міста Харків до Києва на найближчу дату.

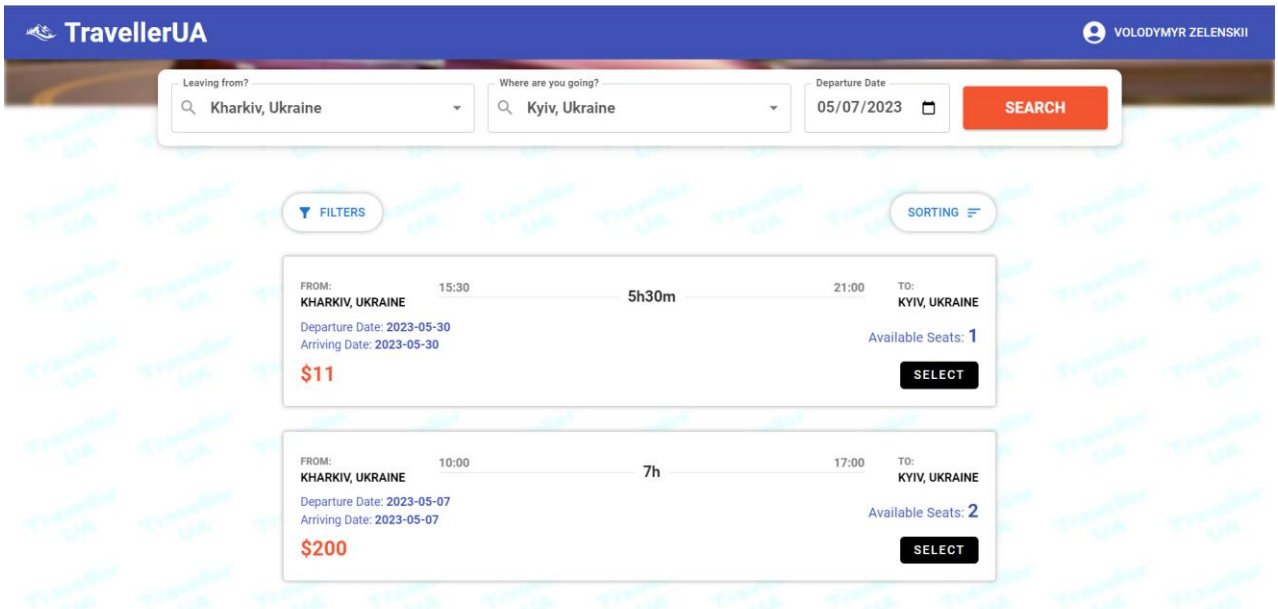

Рисунок 3.11 – Пошук поїздок з міста Харків до міста Київ

Після того як користувач обрав зручну та відповідну поїздку, йому потрібно клікнути на кнопку «Select» та передивитися усі деталі даного перевезення (рис. 3.12). Якщо йому усе підходить, то він може нажати на кнопку «Book now», яка зробить запит водію на додавання нового пасажиру у дану поїздку.

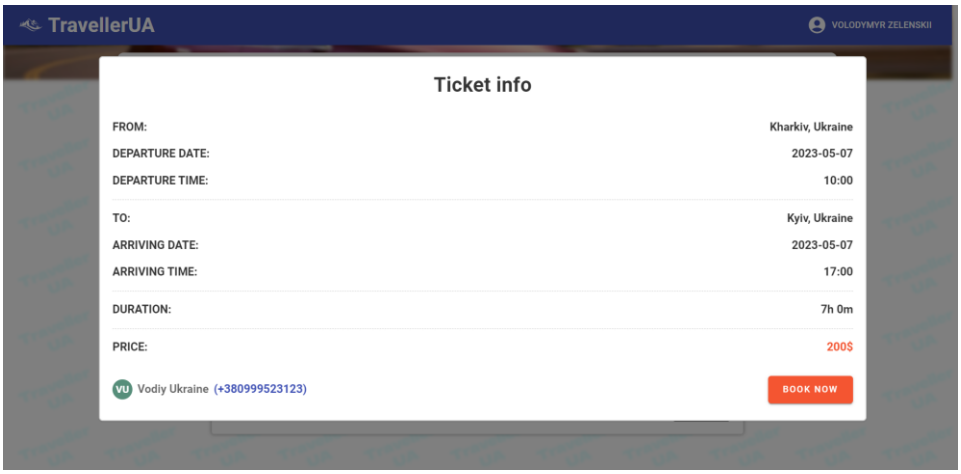

Рисунок 3.12 – Модальне вікно обраної поїздки

Для перегляду усіх своїх зарезервованих перевезень, клієнт може перейти на сторінку «Мої бронювання». Її вигляд можна побачити на рисунку 3.13. На ній знаходяться усі необхідні деталі для комфортної та швидкої поїздки. Також сторінка показує статус бронювання, який змінюється в залежності від дій водія даної подорожі.

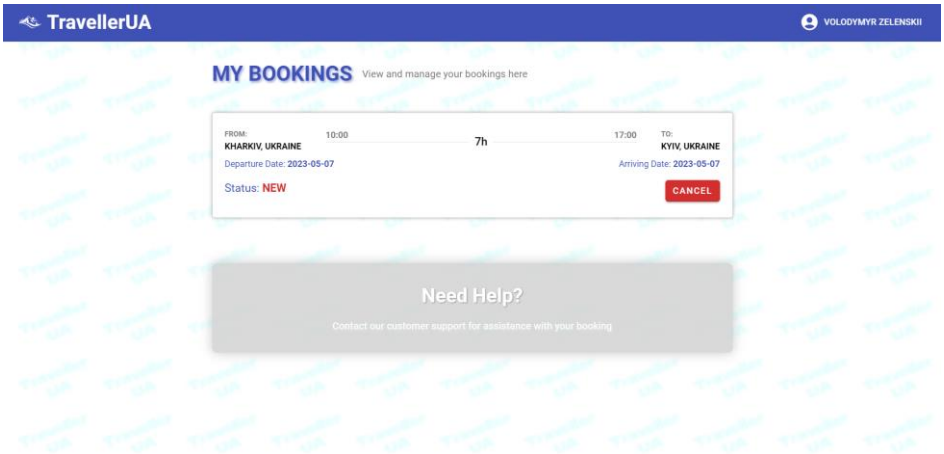

Рисунок 3.13 – Сторінка «Мої бронювання»

Водій, у свою чергу, на сторінці «Мої подорожі», бачить деталі усіх своїх раніше створених поїздок. Дані включають також список прийнятих та неприйнятих пасажирів (рис. 3.14). Після того, як водій прийме користувача до себе на борт, статус пасажира змінюється з «NEW» на «ACCEPTED». У тому випадку, якщо водій вирішить відхилити клієнта, його статус зміниться на «DECLINED» і він не буде допущений до поїздки.

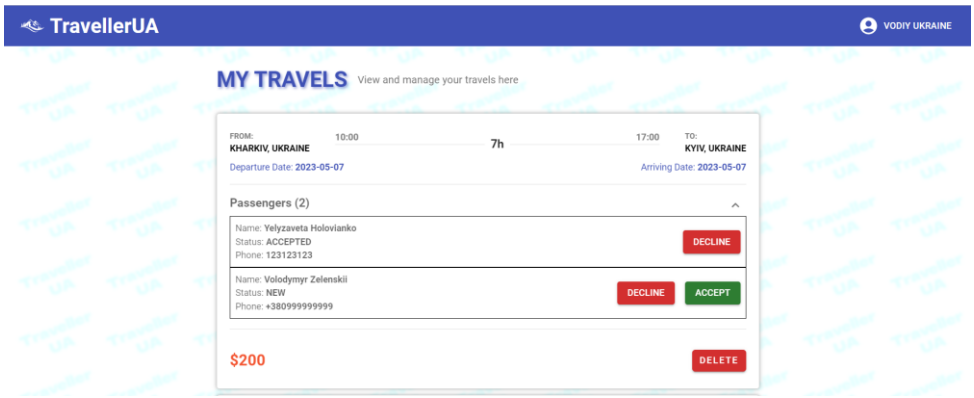

Рисунок 3.14 – Сторінка «Мої подорожі»

Будь-який користувач може створити свою власну поїздку. Для цього лише потрібно увести мінімальний набір інформації, яка дає розуміння мети та деталі подорожі. На рисунку 3.15 можна побачити сторінку «Створити поїздку», яка дозволяє вказати місце, час відправлення та прибуття, кількість вільних місць та ціну подорожі. Після того, як водій натискає кнопку «Create travel», поїздка миттєво стає доступною і будь-який користувач може зарезервувати місце у автомобілі водія [28–30].

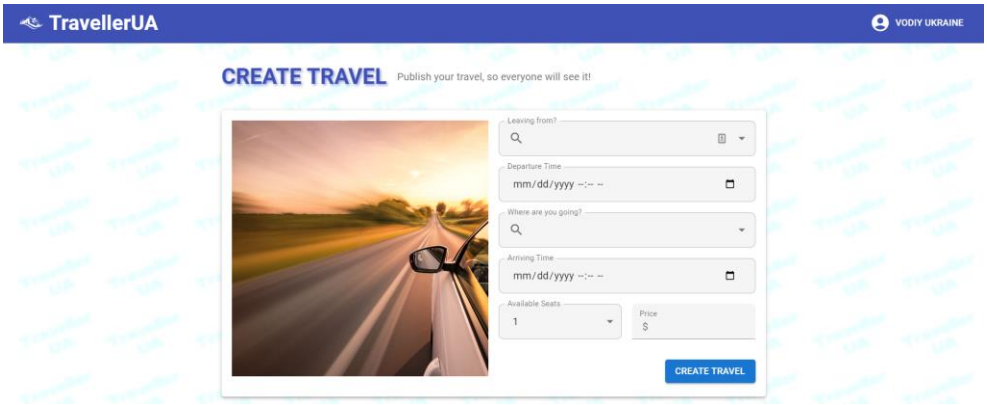

Рисунок 3.15 – Сторінка «Створити поїздку»

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено і реалізовано онлайн-сервіс з пошуку попутників та водіїв за заданим маршрутом.

Метою кваліфікаційної роботи було створення зручного та ефективного інструменту для знаходження попутників та водіїв для спільної подорожі.

Під час розробки були використані сучасні технології та фреймворки, такі як React для розробки застосунку, Node.js для розробки серверної частини та MongoDB для зберігання даних.

Роботу було успішно проаналізовано, сервіс демонструє високу швидкість та продуктивність. Він може бути використаний користувачами з будь-яким рівнем технічних знань та допоможе їм швидко знайти попутників для спільної подорожі, що зменшить витрати та зробить поїздку більш комфортною та безпечною.

У результаті, робота відповідає вимогам сучасного інтернет-сервісу та може бути успішно використана користувачами для пошуку попутників та водіїв на підставі обраного маршруту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022). Tools for Fast Metric Data Search in Structural Methods for Image Classification. *IEEE Access*, 10, pp. 124738-124746.

2. Тітов С. В., & Тітова О. В. (2014). Інформаційно-освітнє середовище навчального закладу: розвиток засобів і способів комунікаційної й інформаційної взаємодії. Вісник Харківської державної академії культури, (43), 144-150.

3. Тітов С. В., & Тітова О. В. (2015). Оцінка юзабіліті освітніх сайтів: методи і технології. Вісник Харківської державної академії культури. Серія: Соціальні комунікації, (47), 127-134.

4. Tvoroshenko I., and Maksimenko H. (2021) Research of regression and modular testing of web applications, Abstracts of IV international science conference «Science, theory and practice» (October 12-15, 2021). Tokyo, Japan, pp. 406-411.

5. Tvoroshenko, I.S., & Maksimenko, H. (2021). To the question of analysis of existing mechanisms of web application testing.

6. Sitnikov D., Titova O., Minukhin S., Kovalenko A., Titov S. (2018). Informativity of Association Rules from the Viewpoint of Information Theory. Conference: 2018 IEEE International Scientific-Practical Conference Problems of Infocommunications. Science and Technology.

7. Tvoroshenko, I. (2020). Information technologies for decision-making on the conditions of spatially distributed objects. In *I International Scientific and Practical Conference. Problems and perspectives of modern science and practice, Austria* (pp. 45-50).

8. Творошенко, І. С., Мгеброва, В. Р., & Білий, В. В. (2016). Практичні аспекти створення вихідної інформації для проведення геоінформаційного аналізу у сфері управління нерухомістю.

9. Творошенко, І. С., & Табашник, В. А. (2018). Розробка просторової моделі геоінформаційної підтримки людей з обмеженими можливостями, що пересуваються на інвалідних колясках, у місті Харків.

10. Тітов С.В., Тітова О.В., Чорна О.С. (2022). Опис нескоротних наборів ознак в приблизних множинах з використанням систем числення. *Збірник наукових праць Харківського національного університету Повітряних Сил. № 1(71)*, с. 106-110.

11. Baresi, L., & Garriga, M. (2020). Microservices: the evolution and extinction of web services?. *Microservices: Science and Engineering*, 3-28.

12. Banks, A., & Porcello, E. (2020). *Learning React: modern patterns for developing React apps*. O'Reilly Media.

13. Thakkar, M., & Thakkar, M. (2020). Next. js. Building React Apps with Server-Side Rendering: Use React, Redux, and Next to Build Full Server-Side Rendering Applications, 93-137.

14. Konshin, K. (2018). *Next. js Quick Start Guide: Server-side rendering done right*. Packt Publishing Ltd.

15. Halili, F., & Ramadani, E. (2018). Web services: a comparison of soap and rest services. *Modern Applied Science*, *12*(3), 175.

16. Quirós, C., Portela, J., & Marín, R. (2021). Differentiated models in the collaborative transport economy: A mixture analysis for Blablacar and Uber. *Technology in Society*, *67*, 101727.

17. Farajallah, M., Hammond, R. G., & Pénard, T. (2019). What drives pricing behavior in peer-to-peer markets? Evidence from the carsharing platform BlaBlaCar. *Information Economics and Policy*, 48, 15-31.

18. Shamsudin, M. F., & Ab Rahman, N. H. (2021). An Authentication of Carpooling Apps Using OTP and Fingerprint. *Applied Information Technology And Computer Science*, *2*(1), 1-10.

19. Hall, J. D., Palsson, C., & Price, J. (2018). Is Uber a substitute or complement for public transit?. *Journal of urban economics*, 108, 36-50.

20. Macdonald, A. J. (2018). *Structure and architecture*. Routledge.

21. D'Angelo, J., & Little, S. K. (1998). Successful web pages: what are they and do they exist?. *Information technology and libraries*, *17*(2), 71.

22. Makris, A., Tserpes, K., Spiliopoulos, G., Zissis, D., & Anagnostopoulos, D. (2021). MongoDB Vs PostgreSQL: A comparative study on performance aspects. *GeoInformatica*, 25, 243-268.

23. Hoque, S. (2020). *Full-Stack React Projects: Learn MERN stack development by building modern web apps using MongoDB, Express, React, and Node. js*. Packt Publishing Ltd.

24. Ofoeda, J., Boateng, R., & Effah, J. (2019). Application programming interface (API) research: A review of the past to inform the future. *International Journal of Enterprise Information Systems (IJEIS)*, 15(3), 76-95.

25. Zheng, Q., Jiao, J., Cao, Y., & Lau, R. W. (2018). Task-driven webpage saliency. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 287-302).

26. Kikuchi, K., Otani, M., Yamaguchi, K., & Simo-Serra, E. (2021, October). Modeling Visual Containment for Web Page Layout Optimization. In *Computer Graphics Forum* (Vol. 40, No. 7, pp. 33-44).

27. Beaird, J., Walker, A., & George, J. (2020). *The principles of beautiful web design*. Sitepoint.

28. Shaheen, S., Cohen, A., & Bayen, A. (2018). The benefits of carpooling.

29. Pukhovskiy, N. V., & Lepshokov, R. E. (2011, May). Real-time carpooling system. In *2011 International Conference on Collaboration Technologies and Systems (CTS)* (pp. 648-649). IEEE.

30. Ma, C., He, R., & Zhang, W. (2018). Path optimization of taxi carpooling. *PLoS One*, *13*(8), e0203221.