Хмельницький національний університет

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Вовчка Віталія Олександровича

«Вебсистема управління службою таксі»

Назва теми

На здобуття рівня вищої освіти бакалавр

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.200248.01.06.ПЗ

Виконала студентка IV курсу групи ІПЗ-20-1 _____ Віталій ВОВЧОК
Підпис Ім'я, прізвище

Керівник канд. тех. наук, доцент _____ Оксана ЯШИНА
Науковий ступінь, звання Підпис Ім'я, прізвище

Нормоконтролер канд. техн. наук, доцент _____ Юрій ФОРКУН
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ім'я, прізвище

11 червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
Л. П. Бедратюк _____
02 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Вовчка Віталія Олександровича
Прізвище, ім'я, по батькові студента

1. Тема роботи Вебсистема управління службою таксі

Керівник роботи Яшина Оксана Миколаївна, канд. техн. наук, доцен
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 08.01.2024 р. №6-КП
2. Строк подання студентом роботи на кафедру 01.06.24 р.
3. Вихідні дані до роботи Методичні матеріали до кваліфікаційної роботи

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація, тестування

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Три креслення: 1.Діаграма послідовності використання вебсистеми;
2.Схема архітектури вебсистеми
3.Діаграма варіантів використання

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Форкун Ю. В., доцент кафедри ІПЗ	30.06.24	06.06.24
Антиплагіат	Форкун Ю. В., доцент кафедри ІПЗ	30.06.24	06.06.24

7. Дата видачі завдання «02» січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Ознайомлення з тематикою кваліфікаційної роботи (КвР), визначення та узгодження індивідуальних тем КвР	01.12 – 31.12.2023	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2024	
3 Проектування програмного забезпечення	01.02 – 28.02.2024	
4 Програмна реалізація з використанням відповідних засобів розробки	01.03 – 10.04.2024	
5 Тестування програмного забезпечення	11.04 – 30.04.2024	
6 Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05 – 25.05.2024	
7 Попередній захист КвР	травень 2024 (згідно графіка)	
8 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 – 30.05.2024	
9 Здача КвР на кафедру; підготовка КвР для розміщення у репозиторії ХНУ; підготовка до захисту та захист КвР	з 01.06.2024	

Студент _____ Віталій ВОВЧОК
Підпис Ім'я, прізвище

Керівник роботи _____ Оксана ЯШИНА
Підпис Ім'я, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи «Вебсистема управління службою таксі».

Автор роботи: Вовчок Віталій Олександрович.

Керівник роботи: Яшина Оксана Миколаївна.

Пояснювальна записка: 83 с., 44рис., 3 табл., 3 дод., 31 джерело.

Графічна частина: 3 креслення ф. А3.

Мета кваліфікаційної роботи: розробка вебсистеми управління службою таксі.

У кваліфікаційні роботі показано створення вебсистеми управління службою таксі, аналіз конкурентів та предметної області, визначено вимоги до створеної вебсистеми, розроблена архітектура, створена структура бази даних та структура застосунку. Для розробки вебсистеми для управління службою таксі було використано мову програмування JavaScript та такі фреймворки, як ReactJS для створення адміністративної панелі керування водіями та службою таксі та React Native застосунок для водіїв, для стилізації адміністрованої панелі було використано UI бібліотеку для ReactJS, Material UI. Для ефективної роботи служби таксі було використано Google Maps API.

Використання розробленої вебсистеми дозволяє автоматизувати роботу служби таксі, зменшити час подачі таксі для клієнтів, збільшити кількість замовлень для водіїв, збільшити прибутки служби таксі та полегшити роботу для операторів та водіїв служби таксі.

11.06.2021
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.200248.01.06.ПЗ	Пояснювальна записка	83		
2	А4		Завдання на	2		
			кваліфікаційну			
			роботу			
3	А4		Анотація	1		
			Графічні документи			
4	А3	КвРІПЗ.200248.01.06.Е8	Діаграма послідовності	1		
			використання			
			вебсистеми			
5	А3	КвРІПЗ.200248.01.06.Е8	Схема архітектури	1		
			вебсистеми			
6	А3	КвРІПЗ.200248.01.06.Е8	Діаграма варіантів	1		
			використання			

Дат	Підпи	№ докум.№	Арк	Підп	Дат
Зм.Виконал		Вовчок В. О.			10.06
Керівник		Яшина О.М			10.06
Н. контр.		Форкун Ю.В.			10.06
Зав. каф.		Бедратюк Л.П.			10.06

Зав. каф. КвРІПЗ.200248.01.06.ПЗ

Вебсистема управління службою таксі

Відомість документів

Літ.	Консул	Аркушів
	Н.	1

ХНУ, ІПЗ-20-1

ЗМІСТ

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.200248.01.06.ПЗ		
Виконала		Вовчок В. О.		10.06	Вебситема керування службою таксі	Літ.	Арк. 6 / Аркушів 83
Керівник		Яшина О. М.		10.06			
Н. контр.		Форкун Ю. В.		10.06	Відомість документів	ХНУ, ІПЗ-20-1	
Зав. каф.		Бедратюк Л.П.		10.06			

ВСТУП

Сьогодні ринок транспортних послуг досить різко змінюється завдяки прогресу в галузі цифрових технологій та інновацій. Майже кожного дня з'являються нові програмні рішення, які допомагають автоматизувати, пришвидшити та зробити більш зручною роботу ту, яку раніше люди повинні були виконувати вручну, але зараз цю роботу виконує програма. Розробка вебсистеми управління службою таксі, яка відповідала б сучасним вимогам швидкості та адаптивності – це складне завдання, для реалізації цього завдання неможливо обійтися без глибокого розуміння технологічних та комерційних аспектів, не достатньо лише написати якісний код, потрібно дуже уважно слідкувати за тенденціями ринку і побудувати програмне рішення, яке б було актуальне на момент виходу на ринок. Тому тема кваліфікаційної роботи – це «Вебсистема управління службою таксі» обрана не випадково, вона відповідає багатьом вимогам ринку та інноваційним транспортним рішення. Метою кваліфікаційної роботи є створення вебсистеми управління службою такс,і що надасть можливість оперативно обслуговувати клієнтів, швидко реагувати на зміни ринку транспортних послуг, забезпечити зручне управління персоналом та підтримувати гнучкість у налаштуванні маршрутів та тарифів.

Щоб досягти поставлених цілей, у проекті використано сучасні технології, такі як React Native для розробки мобільних застосунків та ReactJS для розробки панелі керування водіями та замовленнями. Основна робота буде пов'язана з використанням Google Maps API, так як використання карти є невід'ємною частиною кваліфікаційної роботи. Такий проект вимагає не тільки технічних компетентностей, але і знання бізнес-процесів на ринку транспортних послуг, а також компетентність в адаптації продукту до кон'юнктури ринку. Отже, кваліфікаційна робота допоможе, як студентській науці про інформаційні системи, так і реальному впровадженню цифрових рішень.

Кінцевий продукт який буде створено під час виконання кваліфікаційної роботи на тему «Вебсистема управління службою таксі», має перетворитись не просто на електронну платформу для замовлення поїздок та моніторингу маршрутів, але й стати всебічним комунікаційним інструментом, який забезпечує зв'язок між водіями, диспетчерами та клієнтами. Цей проект передбачає створення функціонального аналогу таких популярних сервісів як «Uber», «Lyft», «Uklon», та «Bolt» і має на меті оптимізувати процеси управління та замовлення таксі за допомогою сучасних технологічних рішень. Для досягнення цієї мети, під час виконання проєкту потрібно виконати наступні завдання:

– Провести огляд ринку транспортних послуг, зокрема існуючих веб-платформ для організації таксі, щоб визначити потреби та очікування, як водіїв так і користувачів;

– Ідентифікувати основні функції, які має виконувати система, забезпечуючи при цьому легкість використання, ефективність управління та високий рівень задоволення користувачів;

– Проаналізувати та вибрати сучасні технології та методики, які найкраще підходять для розробки вебсистем;

– Вибір архітектурних рішень та розробка функціональної моделі системи, яка б відповідала всім технічним та комерційним вимогам;

– Забезпечення якості розробленої системи через тестування та підготовку детальної документації, що описує процеси реалізації проекту;

Слід підкреслити, що успішна реалізація цього проекту вимагає не тільки технічної кмітливості, але й глибокого розуміння динаміки ринку транспортних послуг. Проект застосовує передові технології, забезпечуючи високий рівень інтеграції та масштабування.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Аналіз предметної області у кваліфікаційній роботі включає комплексний огляд кількох фундаментальних процесів:

– вивчення актуальної динаміки ринку;

– оцінка організаційної структури в даній сфері;

– детальний аналіз застосовуваного технічного та програмного забезпечення;

Розгляд цих елементів допомагає забезпечити ґрунтовну підготовку до розробки нового програмного продукту, підкреслюючи важливість використання існуючих рішень та визначення проблем, які потрібно вирішити.

Забезпечення успішного початку проекту вимагає не лише розуміння поточних трендів і потреб ринку, але й глибокий аналіз внутрішніх процесів компаній, що діють у цій галузі ринку. Такий аналіз має включати як стратегічні, так і оперативні аспекти діяльності компанії, що дозволяє виявити основні пункти для впровадження програмного продукту та оптимізації. Критично важливим є також дослідження технічної інфраструктури та програмного забезпечення, яке наразі використовується в предметній області. Це включає аналіз існуючих систем, платформ, інструментів та їх відповідність сучасним вимогам безпеки, швидкості обробки даних та користувацького досвіду. Визначення технологічних недопрацювань і потенційних можливостей для інновацій є критичним для формування основи нової розробки.

У процесі проектування нового продукту особлива увага приділяється розробці архітектури проекту та визначенню користувацьких вимог, що повинні бути інтегровані для вдосконалення існуючих рішень. Важливою частиною є формування логіки роботи програми, що включає створення

структури диспетчерської панелі, мобільного застосунку, баз даних та ефективних механізмів взаємодії між клієнтом та сервером. Користувачі системи, використовуючи диспетчерську панель та додатком, зможуть взаємодіяти з базою даних та системою управління, що є критичним для забезпечення гнучкості та доступності сервісу. Подальші розділи 2 і 3 детально описуватимуть процеси, засновані на цьому аналіз і викладаючи всі технічні та організаційні аспекти розробки.

На сьогоднішній день існує багато різних застосунків для таксі, ще більше служб використовують старі методи коли клієнт повинен сам зателефонувати в службу таксі за вказаним номером, також клієнт повинен чекати певний час на відповідь оператора, потім користувач повинен витрачати час на те, щоб правильно вказати свою адресу або пояснити своє місце знаходження, якщо клієнт погано орієнтується на місцевості або оператор не досвідчений можуть виникнути труднощі щоб вказати правильно стартову і кінцеву точку також в замовленні можуть бути проміжні точки що також впливає на точність замовлення.

Потім після підтвердження замовлення оператором, клієнт повинен ще невизначену кількість часу очікувати на підтвердження свого замовлення водієм, клієнт в такій ситуації до останнього залишається в невизначеності чи буде чи не буде прийняте замовлення і ніяк не може вплинути на цей процес. Ця організація роботи в службі таксі не дуже підходить і не є зручною для клієнта адже займає досить багато часу і має багато етапів на якому клієнт може невірно вказати свою адресу або додаткові дані для замовлення, оператор може невірно повідомити водія про замовлення або оператор може не правильно зрозуміти клієнта, а водій в свою чергу може не вірно зрозуміти оператора і не правильно виконати замовлення.

Також втрачається комунікація між клієнтом та водієм, що є важливим недоліком, як клієнту так і водію важко комунікувати між собою і у ситуаціях коли виникають труднощі з виконанням замовленням, також немає можливості оцінити роботу водія і так само водій не може оцінити пасажира. З наведених

фактів стає зрозуміло що водій, клієнт та диспетчер мають труднощі, які можуть бути виправлені лише з вдосконаленням програмного забезпеченням та введенням в роботу в самій службі таксі.

Виходячи з цієї всієї ситуації ми можемо побачити що поступово зростає необхідність автоматизації роботи та впровадження програмних рішень в роботу компаній, які надають транспортні послуги. Саме програмні рішення можуть скоротити кількість помилок які виникають через людський фактор і впровадження програмних рішень можуть значно підвищити ефективність служб таксі, а це означає що буде підвищення прибутковості, що є важливим фактором для будь-якої сфери діяльності.

Тому зараз програмні рішення є широко популярними в компаніях, які надають транспортні послуги. Усі нові компанії, які починають заходити на ринок одразу потребують програмних рішень і практично усі нові компанії на своєму початку перед виходом на ринок будують ефективні програмні рішення щоб швидше та краще обслуговувати своїх клієнтів та забезпечувати своїх працівників ефективними рішеннями під час виконання їхньої роботи і таким чином працівники будуть виконувати свою роботу краще і ефективніше, що буде позитивно сприяти росту компанії, яка змогла застосувати в своїй роботі якісні програмні рішення.

Сьогодні компаніям може подобатися багато програних рішень. Не зважаючи на те, що програмні продукти для таксі розробляються з однією метою і мають багато спільного, як у архітектурних рішеннях так і в шаблонах та технологіях, вони часто використовують різні принципи функціонування, які можуть найкращим чином забезпечити потрібну функціональність.

Розглянемо типи вебсистем та архітектурні рішення, які можуть бути розглянуті компанією, при виході на ринок таксі послуг існує дуже багато рішень та підходів які застосовують по усьому світі.

Найпопулярнішими рішеннями на даний момент є принцип, коли клієнт має клієнтський застосунок, а водій в свою чергу має водійський застосунок і таким чином клієнт робить замовлення конфігуруючи так, як йому потрібно.

Тобто клієнт може додати стартову та кінцеву точку, клієнт при необхідності може додати проміжні точки, також користувачу не потрібно визначати своє місце знаходження. Адже застосунок по геолокації телефону визначає де знаходиться він і таким чином це суттєво додає зручності для клієнта. Також користувач може самостійно ставити маркер на карті без прив'язки до вулиці. Важливою опцією являється те, що користувач може сам додавати додаткові дані такі, як: «Тиша в салоні», «Англомовний водій», «Дитяче крісло» для замовлення клієнт може додавати опис свого місце знаходження, щоб водій міг краще орієнтуватися на місцевості.

Для клієнтів також доступна функція вибору класу авто, тобто клієнт може вибрати різні класи машин від «Стандартного» до «Бізнес» чи «Мікроавтобус». Є можливість встановлювати замовлення на конкретний час. В свою чергу замовлення потрапляє в загальний ефір де водій може за власними вибором взяти замовлення в роботу або проігнорувати. При взятті замовлення водій може перевірити, яка відстань до замовлення який рейтинг у клієнта, яка ціна замовлення та скільки буде коштувати кілометр замовлення і він може так прорахувати чи буде така поїздка вигідною для нього. Водій та клієнт можуть відмінити замовлення, що теж є дуже зручною опцією.

Такі додатки користуються широкою популярністю і підходять для великих служб таксі які мають достатньо водіїв яких можуть забезпечити великою кількістю клієнтів, також такі додатки коштують дорого в розробці і так само дорого коштують в обслуговувані йдуть великі затрати на використанні різних API, які використовуються додатками, тому це не є рішенням для компаній які мають обмежений бюджет.

Наступним варіантом, який часто зустрічається на ринку є змішана система, яка має частину від минулих часів таксі де є диспетчер і є система вже з новітніх рішень в області транспортних перевезень і базується на тому, що клієнт не має свого застосунку йому потрібно зателефонувати на конкретний номер служби і викликати таксі. У даній системі з'являється диспетчер, який буде приймати замовлення та обслуговувати клієнта і далі після підтвердження

замовлення клієнтом диспетчер відправляє це замовлення до водіїв і далі диспетчер повідомляє клієнта чи було замовлення чи не було взято водієм. Тут є багато незручностей та проблем, які витікають з першим рішенням, але такий підхід є зручним та ефективним для маленьких служб таксі, які не мають великих фінансів і не можуть собі дозволити використовувати такі ресурси які є в великих служб таксі. Тому таке рішення ефективно підходить для початківців у даній сфері і тому було обранл подібну реалізацію для кваліфікаційної роботи.

У цьому проекті розробка програмного продукту починається зі складання технічного завдання, що допомагає уникнути потенційних труднощів у процесі розробки. Далі команда створює прототип користувацького інтерфейсу, за яким відбувається розробка дизайну. Програмісти реалізують функціонал сайту, а завершальним етапом є комплексне тестування готового продукту.

Для аналізу та структурування ключових процесів, що включають множину потоків даних у вебсистемі управління службою таксі, застосовується методологія IDEF-0. Цей підхід функціонального моделювання є особливо корисним для графічного представлення складних процесів та взаємодій у системі. Що допомагає в наглядному форматі показати та зрозуміти ієрархічну структуру процесів, ефективно видавлюючи зв'язки між різними компонентами системи, включаючи їх вхідні та вихідні потоки даних.

Основною перевагою застосування IDEF-0 є можливість створення контекстних діаграм, які спрощують розуміння обсягу процесів та деталізувати залежності всередині системи. Це надзвичайно значуще для адміністраторів та розробників, оскільки це дає можливість чітко контролювати та оптимізувати кожен аспект функціонування платформи. Для реалізації візуалізації та створення цих діаграм використовується платформа Draw.io. Це безкоштовний онлайн-ресурс, який дозволяє користувачам легко створювати, редагувати та ділитися професійними діаграмами та візуальними моделями. Draw.io підтримує широкий спектр типів діаграм, включаючи, але не обмежуючись,

діаграмами потоків, UML діаграмами, діаграмами ER (Entity-Relationship) та багатьма іншими, що робить ідеальним інструментом для комплексного моделювання та аналізу процесів у кваліфікаційній роботі. Завдяки своїй інтуїтивно зрозумілій інтерфейсній платформі та можливості співпраці в реальному часі, Draw.io стимулює ефективність комунікації між учасниками проекту та забезпечує легкий доступ до важливих даних та інформації, що сприяє більш злагодженій та інформованій робочій атмосфері.

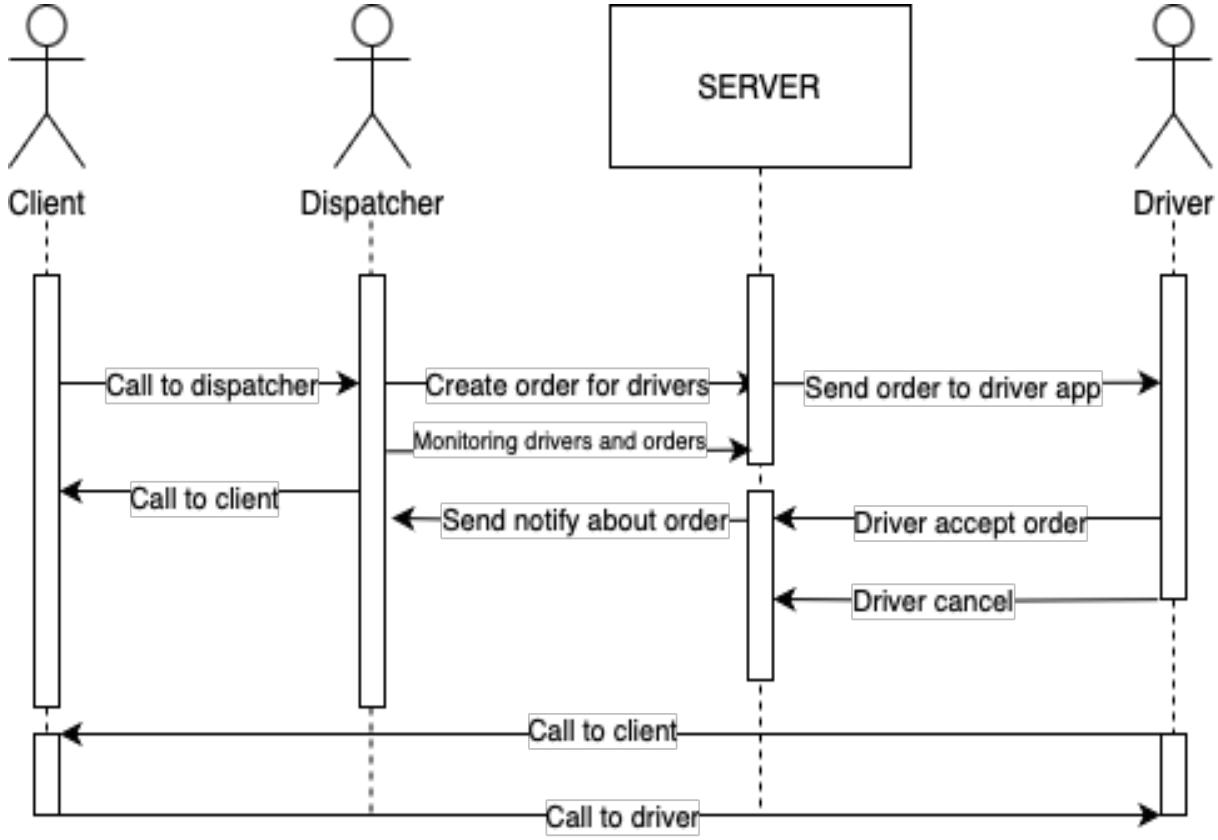


Рисунок 1.1 – Діаграма послідовності

На рисунку 1.1 зображено Діаграму послідовності, на даній діаграмі зображено, як буде відбуватися взаємодія між клієнтом, водієм, диспетчером та самим програмним продуктом. Діаграма дає зрозуміти як буде відбуватися робота і як будуть взаємодіяти між собою головні частини продукту. Потоки демонструють надходження даних та як будуть ці дані оброблятися між собою. Адміністрація також відіграє ключову роль у функціонуванні системи, оскільки

відповідає за моніторинг даних, що розміщені на платформі, взаємодію з клієнтами та обробку замовлень. Саме здатність втілити цей процес є основною причиною для розробки програмного забезпечення. Крім того, вихідні дані, які включають специфічні набори інформації, є необхідними для формування замовлень та забезпечення оптимального досвіду користувачів при використанні системи. У процесі розробки даного програмного продукту етап спілкування з замовником має важливе значення, оскільки воно передує створенню технічного завдання.

Це сприяє уникненню несподіваних проблем на більш пізніх стадіях розробки. Після отримання всіх необхідних даних та побажань від замовника, складається технічне завдання. Це завдання є основою для подальшої роботи і детально описує всі вимоги до проекту. На наступному етапі команда розробників створює прототип інтерфейсу користувача, що дозволяє візуалізувати майбутній продукт і узгодити з замовником. Далі програмісти переходять до реалізації функціональних можливостей вебсистеми для служби таксі, враховуючи всі деталі та технічні вимоги. Кожен модуль і компонент ретельно перевіряється на відповідність початковим вимогам та працездатність. Завершальним етапом є комплексне тестування всього продукту. Це тестування включає різні види перевірок: функціональні, інтеграційні, навантажувальні, і забезпечує високу якість та надійність програмного забезпечення. Після успішного проходження всіх тестів продукт готується до запуску і впровадження в експлуатацію. Після запуску продовжується моніторинг і підтримку продукту, оперативно реагуючи на будь-які проблеми чи запити від користувачів. Це включає оновлення, виправлення помилок та покращення функціоналу на основі зворотного зв'язку. Важливим аспектом є також підготовка документації для ефективного використання системи.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

В рамках аналізу наявного програмно-технічного забезпечення проведено дослідження вже існуючих рішень на ринку сервісів таксі. Такий підхід дозволяє а адаптувати та використати кращі практики від провідних компаній у даній галузі, які вже показали свою ефективність та надійність. Аналіз популярних рішень буде почато з огляду таких гігантів індустрії, як «Uber», «Lyft», «Uklon» та «Bolt». Це дозволить вивчити ключові аспекти їх архітектури та шаблонів розробки, що в свою чергу допоможе в оптимізації та поліпшенні розроблюваної системи. Аналіз буде розпочати із зарубіжних компаній, які являються першими у данйі індустрії і є провідниками тенденцій.

1) Uber – це американська транснаціональна компанія з технологій мобільних послуг, яка розробила однойменний мобільний застосунок для пошуку, виклику та оплати таксі або приватних водіїв. Компанію було засновано в далекому 2009 році, яка революціонізувала сектор пасажирських перевезень, впровадивши модель, яка дозволяє користувачам замовляти таксі через мобільний застосунок. Цей сервіс не лише змінив спосіб, яким люди думають про урбаністичні перевезення, але й став причиною появи терміну «шерингова економіка». Основою успіху Uber є його високо інтегрована технологічна платформа. Застосунок Uber використовує складні алгоритми маршрутизації і ціноутворення, що базуються на часі, відстані та попиті.

Платформа включає функції такі як: GPS-слідкування, яке дозволяє користувачам в реальному часі бачити місцезнаходження автомобілів. Динамічне ціноутворення (surge pricing), що змінює вартість поїздок в залежності від попиту та пропозиції. Системи безпеки, включаючи перевірку водіїв, можливість ділитися маршрутом із контактами для забезпечення

додаткової безпеки. На рисунку 1.2 та 1.3 буде продемонстровано фото клієнтського та водійського застосунку, яким користуються водії та клієнти =

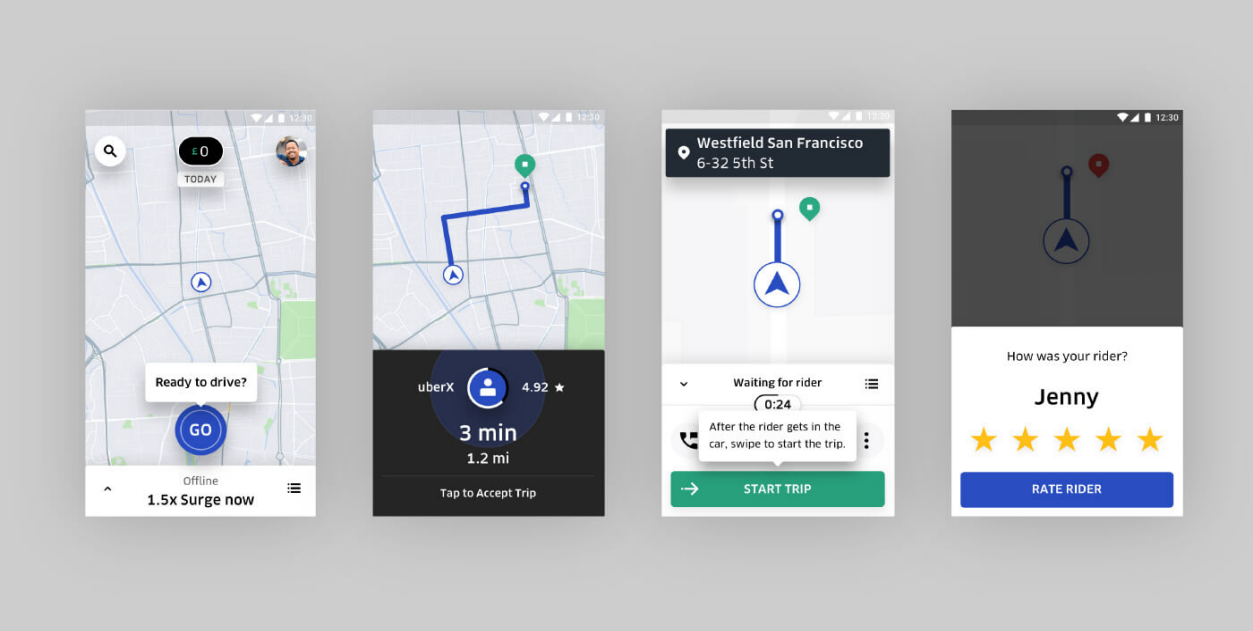


Рисунок 1.2 – Водійський застосунок Uber

Водійський застосунок Uber призначений для використання водіями, які зареєстровані в системі Uber. Цей застосунок надає інформацію про доступні поїздки, дозволяє водіям приймати або відхиляти запити на поїздки, а також слідкувати за своїм маршрутом через інтегровану GPS-навігацію. Водії можуть використовувати застосунок для відстеження своїх доходів, перегляду історії поїздок та отримання важливих повідомлень від Uber. Застосунок також включає функції безпеки, такі як кнопка екстреної допомоги та можливість поділитися маршрутом з друзями . Оновлення додатка регулярно випускаються, щоб вдосконалити його функціональність та забезпечити кращий досвід для водіїв. Водійський застосунок Uber використовує шифрування даних для забезпечення конфіденційності інформації водія та пасажира, а також використовує автоматизовані алгоритми для оптимізації маршрутів і мінімізації часу очікування.

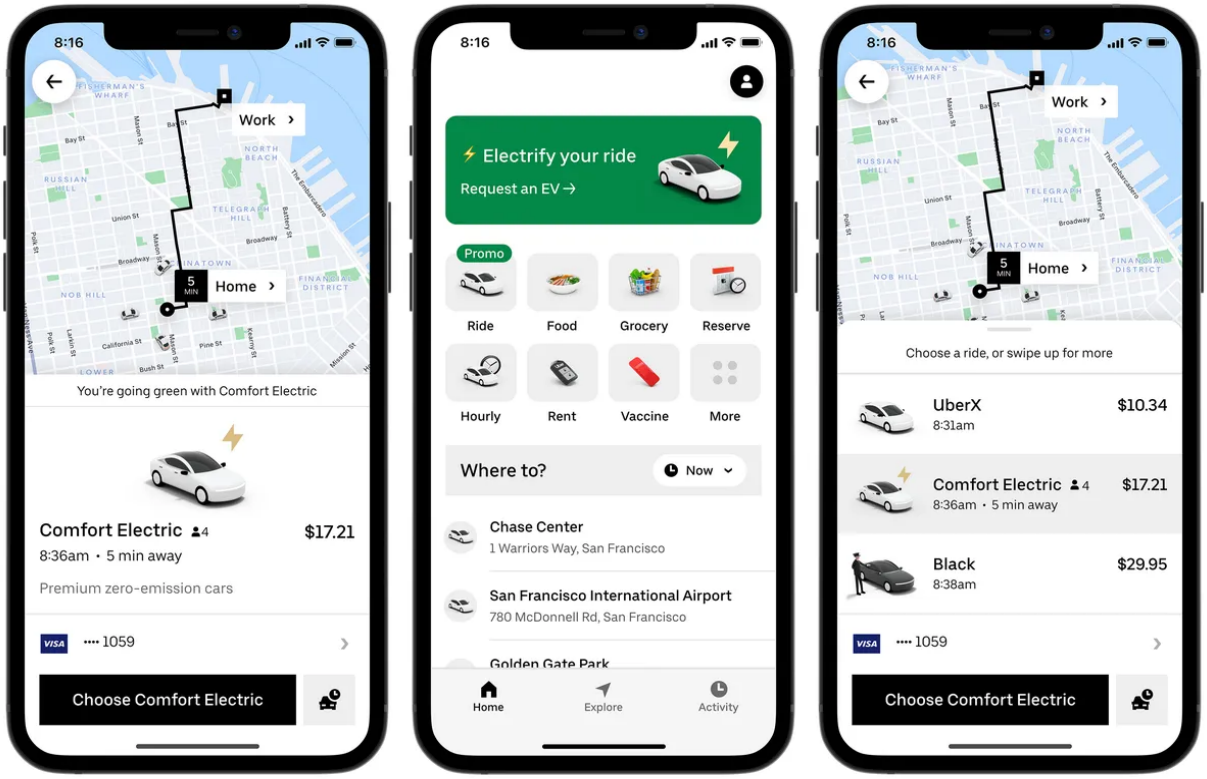


Рисунок 1.3 – Клієнтський застосунок Uber

Клієнтський застосунок Uber дозволяє користувачам з легкістю замовляти поїздки зі своїх смартфонів. Він використовує розташування користувача для знаходження найближчих доступних водіїв. Користувачі можуть вводити пункт призначення та отримувати оцінку вартості поїздки перед її замовленням. Після замовлення, користувачі можуть відслідковувати прибуття свого таксі в реальному часі, переглядати профілі водіїв, їхні оцінки та відгуки від інших користувачів. Застосунок також дозволяє вибирати різні опції поїздки, такі як економ, преміум або групові поїздки, залежно від потреб та переваг користувача. Після завершення поїздки, клієнти можуть оцінити свій досвід та залишити відгук, що допомагає покращити сервіс. У застосунку передбачена можливість оплати як через збережені кредитні картки, так і через інші методи платежу, включно з готівкою в деяких регіонах. Клієнтський застосунок Uber також включає безпекові функції, такі як можливість поділитися деталями поїздки з довіреними контактами та прямий доступ до

служб екстреної допомоги через застосунок. Це створює додатковий рівень впевненості та безпеки для користувачів. Оновлення додатка регулярно випускаються, щоб забезпечити високу якість послуг і відповідність сучасним технологічним стандартам.

2) Lyft – це американська компанія, яка пропонує послуги пасажирських перевезень, працюючи за принципом спільного використання автомобілів. Заснована в 2012 році, Lyft стала одним із основних конкурентів Uber на ринку США. Подібно до Uber, Lyft дозволяє користувачам замовляти поїздки за допомогою мобільного застосунку, який з'єднує їх із водіями-партнерами, що використовують власні автомобілі для перевезення пасажирів. На рисунках 1.4 та 1.5 буде зображено водійський та клієнтський застосунок

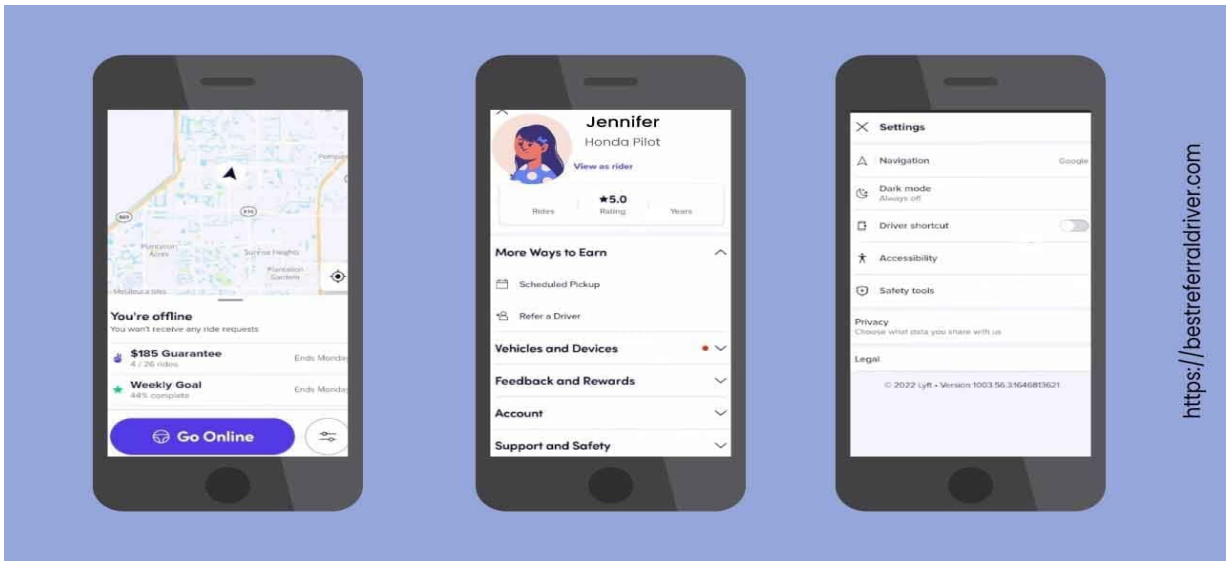


Рисунок 1.4 – Водійський застосунок Lyft

Водійський застосунок Lyft, подібно до Uber, забезпечує водіям інтерфейс для прийому та управління поїздками. Однак, відмінності між цими додатками починаються з прозорості доходів, де Lyft надає більш чіткі вказівки про можливий заробіток до прийняття поїздки. Це створює довіру серед водіїв, які можуть більш обдумано приймати рішення про поїздки.

Lyft також часто пропонує бонуси та стимули за активність, які можуть додатково підвищити заробітки водіїв, на відміну від Uber, де такі пропозиції можуть з'являтися менш часто. Lyft підтримує свою спільноту водіїв краще,

надаючи різні програми допомоги та підтримки, що може відчуватися більш вагомо порівняно з досвідом у Uber. Ще однією відмінною рисою є більш гнучкі налаштування додатка Lyft для вибору поїздок, що відповідають маршрутам чи графіку водія, тоді як в Uber система може бути схильною до автоматичного відправлення замовлень без врахування особистих переваг.

Клієнтський застосунок Lyft, як і його конкурент Uber, дозволяє користувачам замовляти поїздки через їхні смартфони. Застосунок має інтуїтивно зрозумілий інтерфейс, де користувачі можуть ввести своє місцезнаходження та пункт призначення, отримати оцінку вартості поїздки та вибрати тип автомобіля для поїздки. Особливістю Lyft є можливість побачити фотографію водія та марку автомобіля, що додає додаткової впевненості пасажирам. Lyft забезпечує високий рівень безпеки завдяки вбудованим функціям, таким як можливість ділитися деталями поїздки з контактами та екстрена кнопка, яка сполучає з службами безпеки.

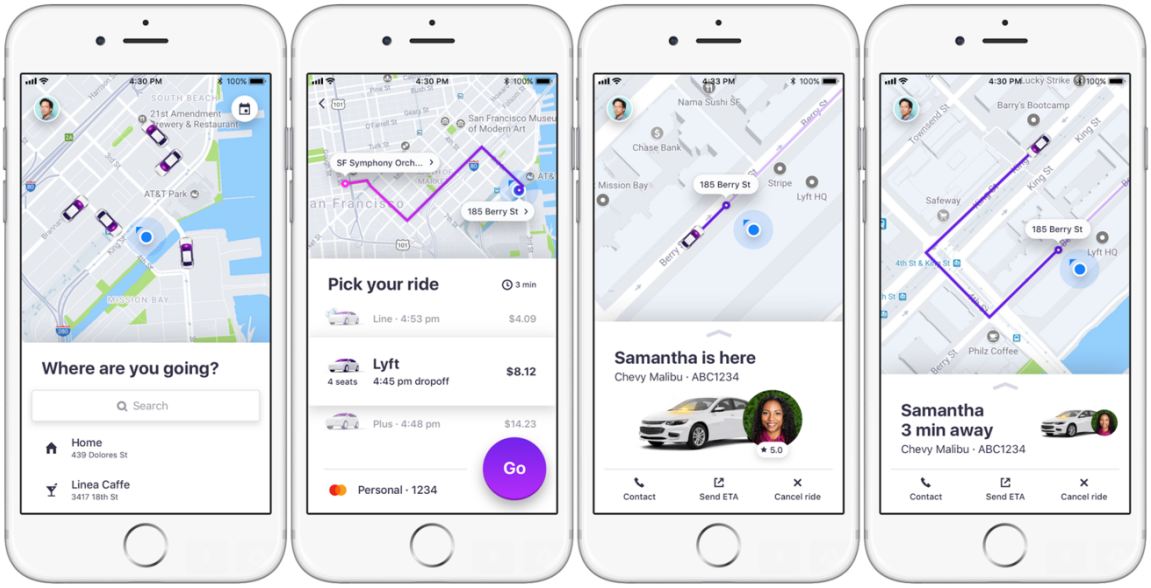


Рисунок 1.5 – Клієнтський застосунок Lyft

Також, перед початком поїздки, застосунок відображає інформацію про водія, включаючи його рейтинг і відгуки від інших користувачів. Проте, одним

із недоліків застосунку може бути його доступність, оскільки Lyft не працює у такій великій кількості міст і країн, як Uber. Це може обмежувати користувачів у деяких регіонах. Іноді, користувачі можуть зіткнутися з високими цінами під час пікових годин, коли активується динамічне ціноутворення. Загалом, клієнтський застосунок Lyft пропонує зручні та безпечні послуги для замовлення таксі, хоча і має деякі обмеження порівняно з глобальним діапазоном та доступністю Uber.

3) Bolt почав свою діяльність в Україні в 2016 році, спочатку під назвою Taxify, і швидко став одним з головних гравців на ринку послуг таксі в країні. Станом на сьогодні, Bolt діє у багатьох великих містах України, включаючи Київ, Львів, Одесу, Харків, Дніпро, та інші. Компанія надає широкий спектр послуг, від звичайних поїздок до преміум-поїздок, а також послуги доставки їжі Bolt Food. Bolt швидко завоював популярність в Україні завдяки своїм конкурентним цінам, які часто нижчі, ніж у основних конкурентів.

Це стало можливим завдяки низькій комісії, яку компанія стягує з водіїв, що дозволяє їм заробляти більше та пропонувати нижчі тарифи клієнтам. Одним з його недоліків є те що він погано інтегрований в усі міста України а доставка Bolt Food взагалі є тільки в окремих містах. Bolt не обмежується лише перевезенням пасажирів. В Україні компанія також запустила сервіс доставки їжі Bolt Food, який конкурує з іншими місцевими та міжнародними сервісами доставки. Крім того, Bolt розглядає можливості запуску прокату електросамокатів в українських містах, що вже успішно функціонує в інших країнах. Тому якісна побудова архітектури коду може в майбутньому дати можливість легкого впровадження і інших особливостей в існуючу систему. На рисунках 1.6 та 1.7 буде зображено візуальну частину клієнтського та водійського застосунківта описаний їхній функціонал та порівняно з іншими платформами.

Рисунок 1.6 – Водійський застосунок Bolt

Водійський застосунок Bolt в Україні дозволяє водія м легко приймати замовлення, відстежувати свої доходи та виконувати поїздки. Застосунок пропонує зручний інтерфейс, що інтегрований з системою навігації, що полегшує водіям планування маршрутів. Однією з переваг Bolt є нижча комісія для водіїв порівняно з Uber, що може зробити Bolt більш привабливим для водіїв в Україні.

Крім того, Bolt часто пропонує бонусні програми для водіїв, що дозволяє їм заробляти більше. На відміну від Uber і Lyft, Bolt в Україні також розвиває інші транспортні сервіси, такі як доставка їжі та електричні самокати, що надає додаткові можливості для заробітку. Проте, Uber і Lyft мають більш розгалужену міжнародну мережу та ресурси, що дозволяє їм впроваджувати більш передові технології та програми лояльності. Водночас, наявність Lyft в Україні обмежена, тому більшість порівнянь для українських водіїв відбувається між Bolt і Uber. У плані технологій та користувацького досвіду, всі три компанії пропонують схожі основні функції, але Bolt вирізняється своєю локалізацією та адаптацією до українського ринку, що може забезпечити йому конкурентну перевагу в певних регіонах країни.

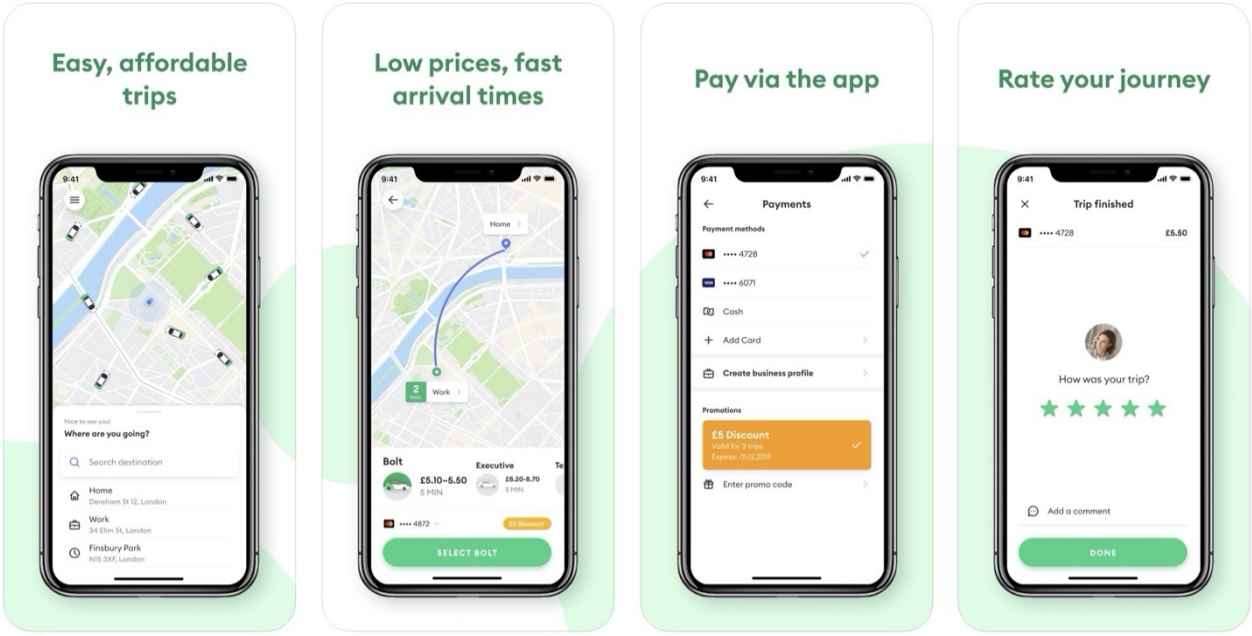


Рисунок 1.7 – Клієнтський застосунок Bolt

Bolt Клієнтський застосунок Bolt в Україні забезпечує користувачам зручний спосіб замовлення таксі та інших транспортних послуг. Інтерфейс додатка дозволяє користувачам вводити пункт призначення, вибирати тип транспорту та переглядати вартість поїздки перед її замовленням. Відмінною особливістю Bolt є конкурентні ціни та часті акції та знижки, що робить його привабливим вибором для українських користувачів.

На відміну від Uber, Bolt також пропонує послуги доставки їжі та оренди електросамокатів у деяких містах, що робить його більш універсальним додатком. Порівняно з Lyft, який має обмежену присутність у країні, Bolt має ширшу географічну доступність в Україні, що дозволяє йому обслуговувати більше клієнтів у різних регіонах. Клієнтський застосунок Bolt має вбудовані функції безпеки, такі як можливість ділитися деталями поїздки з довіреними контактами. Також, застосунок забезпечує високий рівень користувацького досвіду завдяки швидкому відгуку системи та легкості навігації. Все це разом робить Bolt сильним конкурентом на ринку мобільних перевезень в Україні. Також в Україні широко поширена транспотрна компанія Uklon, яка на мою

суб’єктивну думку є номером 1 в Україні і надає найбільш якісні послуги і має найбільше поширень по усій країні і якісно забезпечує свої послуги і тому саме на базі цієї компанії будуть будватися рішення для побудови моєї кваліфікаційної роботи. Uklon кожного року добре розвивається він постійно додає новий функціонал та робить зручний сервіс як для водіїв так і для клієнтів. Uklon — це українська компанія, яка надає послуги таксі і вважається одним з лідерів на ринку мобільних перевезень в Україні.

Заснована в 2010 році, Uklon була однією з перших служб, що запропонувала повноцінний мобільний застосунок для замовлення таксі в країні, і швидко завоювала популярність серед українських користувачів. Інновації та функціональність Uklon вирізняється своєю інноваційністю і зручністю. Компанія постійно впроваджує нові технологічні рішення для покращення користувацького досвіду. Клієнти можуть замовляти поїздки через мобільний застосунок або веб-сайт, вибираючи різні категорії авто від економ до преміум класу, а також використовувати опцію замовлення вантажівок чи кур'єрських послуг.

Особливості Однією з унікальних особливостей Uklon є можливість вибору певних водіїв або автомобілів, які користувачі вже визначили як уподобані. Також сервіс дозволяє заздалегідь планувати поїздки, роблячи його популярним серед користувачів, які цінують планування і надійність. Безпека і надійність Uklon серйозно ставиться до питань безпеки. Водії проходять ретельну перевірку перед тим, як їм дозволяється працювати через платформу. Клієнти можуть переглядати рейтинги водіїв і відгуки інших користувачів перед замовленням поїздки, а також поділитися деталями поїздки з довіреними контактами.

Розвиток і конкуренція навідміну від міжнародних конкурентів, як-от Uber і Bolt, Uklon зосереджується виключно на ринку України, що дозволяє компанії глибше зрозуміти місцеві особливості та потреби своїх користувачів. Завдяки цьому Uklon зміг здобути велику частку ринку, пропонуючи сервіси, які добре адаптовані до українського контексту. Компанія продовжує

інвестувати у розширення своїх послуг, удосконалення технологій та покращення якості обслуговування, що робить її сильним гравцем на українському ринку мобільних перевезень. На рисунках 1.8 та 1.9 буде зображено водійський та клієнтський застосунок і порівння з іншими.

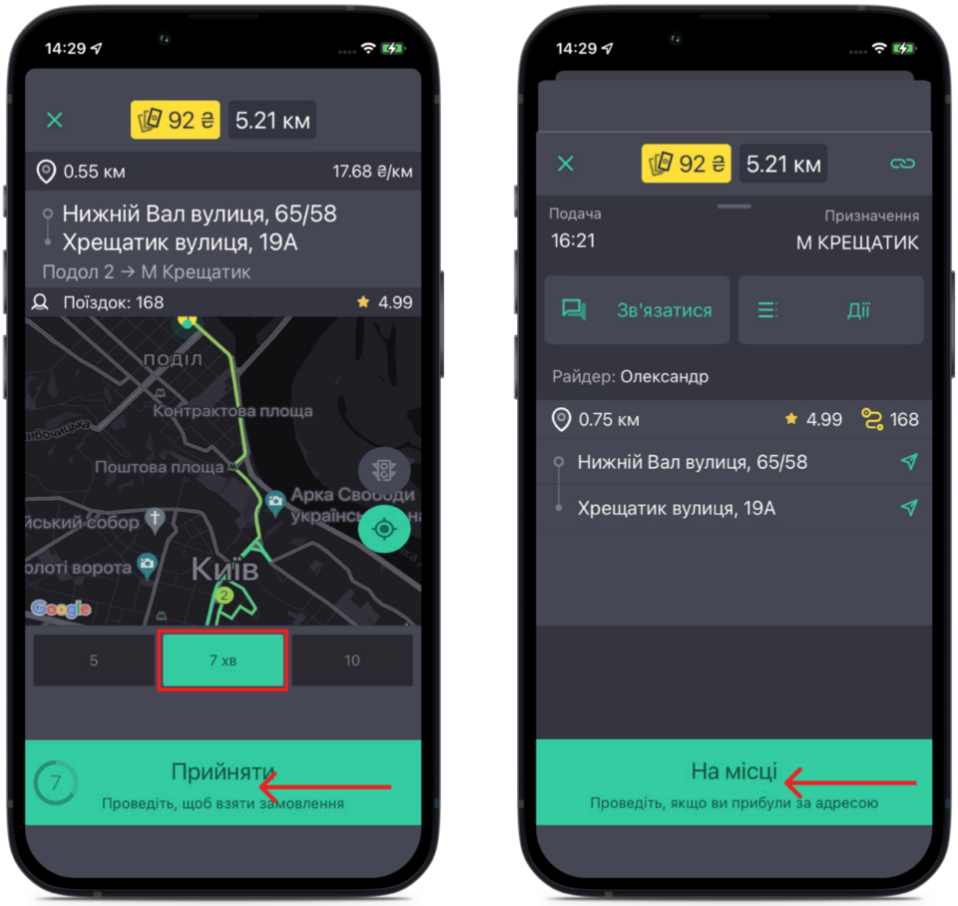


Рисунок 1.8 – Водійський застосунок Uklon

Застосунок для водіїв Uklon в Україні є зручним інструментом, який дозволяє водіям ефективно управляти своїми поїздками, відстежувати доходи та отримувати замовлення в реальному часі. Цей застосунок забезпечує водіям доступ до широкого спектру замовлень, включаючи не лише стандартні поїздки, але й замовлення на доставку та інші логістичні послуги. Водії можуть вибирати замовлення, які найкраще відповідають їхнім графікам та перевагам маршруту.

Особливістю застосунку є система оцінювання, де користувачі можуть залишати відгуки про поїздки, що стимулює водіїв підтримувати високий рівень обслуговування. Також, Uklon надає можливість водіям отримувати бонуси за виконання певної кількості поїздок або за роботу в пікові години, що збільшує їх потенційний дохід.

Застосунок також включає функції для ведення обліку фінансів, що дозволяє водіям слідкувати за своїми за]\робітками і витратами на пальне чи автомобільне обслуговування. Водіям пропонуються детальні маршрути та навігація до місця зустрічі з клієнтом, а також до пункту призначення, що спрощує процес виконання поїздок.

Важливою перевагою є і те, що Uklon активно співпрацює з водіями, пропонуючи навчальні матеріали та семінари для підвищення кваліфікації, що сприяє підвищенню якості обслуговування. Незважаючи на всі переваги, водії можуть зіткнутися з викликами, такими як велика конкуренція на платформі, що може впливати на кількість доступних замовлень. Загалом, застосунок Uklon для водіїв в Україні надає ефективні інструменти для управління роботою, допомагає оптимізувати заробітки та підтримувати високу якість послуг. Клієнтський застосунок Uklon дозволяє користувачам в Україні легко замовляти таксі та інші транспортні послуги прямо зі свого смартфона. Застосунок пропонує різноманітні опції транспорту від стандартних до преміум класів автомобілів, а також можливість замовлення вантажівок та кур'єрських послуг. Особливістю Uklon є його гнучкість у виборі способів оплати, включаючи готівку, кредитні картки та мобільні платежі.

Інтерфейс застосунку зручний і інтуїтивно зрозумілий, дозволяючи користувачам вибирати пункт відправлення та призначення, оцінювати вартість поїздки і слідкувати за прибуттям таксі в режимі реального часу. Uklon також дозволяє зберігати улюблені адреси та маршрути, що робить подальші замовлення ще швидшими і зручнішими. Застосунок надає високий рівень безпеки, дозволяючи користувачам переглядати інформацію про водія та

транспортний засіб перед початком поїздки, а також має можливість ділитися деталями поїздки з друзями або сім'єю.

Крім того, Uklon пропонує систему рейтингів і відгуків, яка дозволяє користувачам відгукуватися про свої поїздки, що сприяє підтримці високих стандартів обслуговування місцевих умов і потреб, забезпечуючи надійний та зручний спосіб пересування по місту.

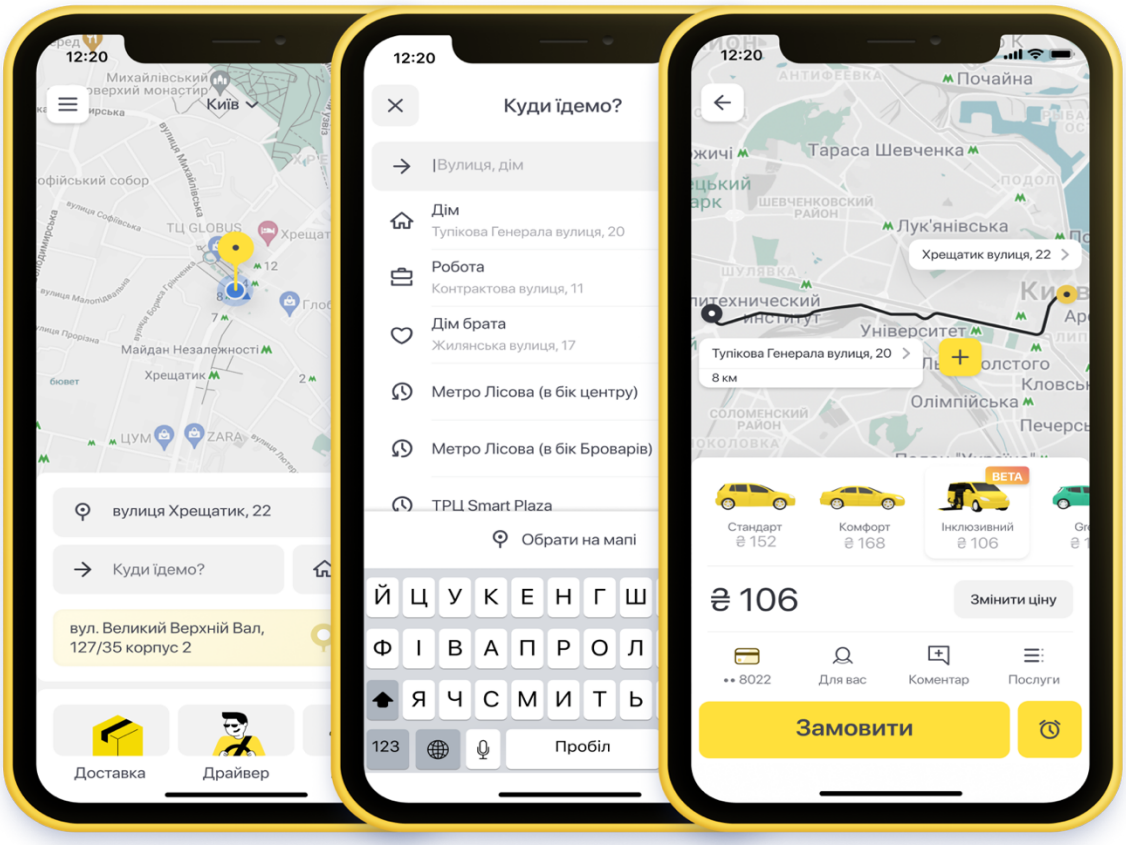


Рисунок 1.9 – Клієнтський застосунок Uklon

Uber, Lyft, Bolt і Uklon — чотири ключові гравці на ринку мобільних перевезень, кожен з яких має свої унікальні особливості та стратегії роботи в різних регіонах світу, включаючи Україну. Хоча Uber і Lyft домінують на американському ринку, Bolt та Uklon мають сильні позиції в Європі та інших регіонах. Кожен з цих сервісів пропонує зручність, безпеку та доступність для своїх користувачів.

У контексті України, Uklon виділяється серед інших завдяки кільком ключовим перевагам, які роблять його найкращим вибором для українських

користувачів: Локальна адаптація: Uklon був розроблений українською командою з урахуванням специфіки місцевого ринку, що дозволяє сервісу краще задовольняти потреби місцевих користувачів.

Широкий спектр послуг: Від замовлення таксі до доставки та логістичних послуг, Uklon пропонує більший діапазон послуг порівняно з іншими міжнародними платформами. Конкурентні ціни: Завдяки оптимізації витрат та високій ефективності, Uklon забезпечує одні з найнижчих цін на перевезення в країні. Підтримка спільноти: Uklon активно співпрацює з місцевими водіями та користувачами, надаючи підтримку і відгукуючись на їхні потреби, що зміцнює довіру і лояльність. Технологічна інновація: Компанія постійно впроваджує нові технологічні рішення для поліпшення досвіду користувачів і підвищення ефективності своїх сервісів.

Загалом, Uklon не тільки конкурує з міжнародними брендами на рівних, але і часто перевершує їх на українському ринку, надаючи більш тісний зв'язок з клієнтами та враховуючи місцеві особливості та потреби. Це робить Uklon ідеальним вибором для тих, хто шукає надійний, доступний та зручний сервіс перевезень в Україні. Самі базуючись на усіх цихфакторах була розроблена візуальна та функціональна компонента, на якій буде базуватися кваліфікаційна робота. Дуже важливо на початку розробки вебсистеми для служби таксі правильно оцінити існуючі рішення і зрозуміти, яким вимогам повинен відповідати створений застосунок та диспетчерська панель. Потрібно дуже уважно приділити увагу до деталей та виділити головні особливості та інтегрувати програмні рішення у свою розробку для залучення більшої кількості клієнтів.

1.3 Визначення функціональних вимог до вебсистеми управління службою таксі.

Розроблюване програмне забезпечення має виконувати ряд критичних функцій, які забезпечують якісне та ефективне обслуговування користувачів. Серед ключових функцій — здатність користувачів переглядати доступні автомобілі та водіїв у реальному часі, бронювання поїздок, а також забезпечення безпеки користувацьких даних, включно з платіжною інформацією, після реєстрації або входу в систему.

Додатково, система повинна надавати функції пошуку на сайті, можливість вибору конкретних параметрів поїздки, а також забезпечувати зв'язок з адміністратором або диспетчером служби. Ці функції дозволять користувачам ефективно взаємодіяти з системою, підвищуючи рівень задоволеності та лояльності до служби, а також сприяючи підвищенню її конкурентоспроможності на ринку мобільних перевезень.

При першому відвідуванні застосунку для водія потенційний водій служби таксі має мати змогу оглядати доступні послуги та переходити по різним сторінкам застосунку для отримання додаткової інформації, а диспетчер також має з легкістю отримувати доступ до усього функціоналу те легко в ньому орієнтуватися. Цей процес вимагає взаємодії з вебсайтом з віддаленою базою даних для зберігання та обробки користувацьких даних.

Розроблюваний застосунок складатиметься з головної сторінки, де відображатимуться доступні водії або таксі, сторінки для управління замовленнями, де користувачі можуть додавати, модифікувати або скасовувати замовлення, та сторінки для завершення замовлення. В застосунку повинна буде бути змога поповнити свій баланс водію щоб могти брати замовлення через платіжну систему. Застосунок має бути кросбраузерним та кросплатформеним, щоб забезпечити його доступність та функціональність на різних пристроях і веб-браузерах.

2 ПРОЕКТУВАННЯ ВЕБСИСТЕМИ СЛУЖБОЮ ТАКСІ

2.1 Проектування архітектури та структури системи.

Архітектурне проектування вебсистеми управління службою таксі значною мірою базується на клієнт-серверній моделі, яка є стандартом для сучасних веб-застосунків, зокрема для таких сервісів, як «Uber», «Lyft», «Uklon», і «Bolt». Ця архітектура передбачає, що взаємодія між користувачем (клієнтом) у даному випадку це буде водій, клієнт та диспетчер та системою (сервером) відбувається через інтернет, з використанням веб-браузера або мобільного застосунку, як клієнтської платформи.

Клієнтська частина вебсистеми забезпечує користувачам графічний інтерфейс, створений з використанням технологій, таких як HTML, CSS, JavaScript, які є основою для таких фремворків та бібліотек, як ReactJS та React Native, які дозволяють їм легко відправляти запити на сервер. Завдяки цьому користувачі можуть з легкістю вибирати водіїв, бронювати поїздки та переглядати статус своїх замовлень. Сервер, у свою чергу, обробляє ці запити, виконує логіку застосунку, зберігає дані у базі даних та відправляє відповіді назад до клієнта. Однією з ключових переваг клієнт-серверної архітектури є можливість відокремлення користувацького інтерфейсу від бізнес-логіки, що дозволяє зменшити завантаження на користувача та оптимізувати обробку даних. Це також поліпшує масштабованість та управління ресурсами, оскільки сервер може ефективно обробляти запити від великої кількості користувачів одночасно.

Ще одним критичним аспектом є використання мережевих протоколів, таких як HTTP, TCP, та UDP, для забезпечення надійної передачі даних між клієнтом і сервером. Для реалізації реального часу обміну даними та подій у системі управління таксі, застосовуються WebSockets. Ця технологія дозволяє забезпечити двосторонній комунікаційний канал між клієнтом і сервером, що є ідеальним для миттєвого оновлення статусів поїздок, координат водіїв, а також для швидкого відгуку на дії користувачів. Розглядаючи серверну частину,

часто розробляється з використанням мов програмування, таких як Java, Python, PHP, JavaScript або C#, що дозволяє виконувати комплексну обробку даних і забезпечувати високий рівень безпеки та персоналізації сервісу. Сервер не тільки займається логікою обробки даних, але й контролює доступ до ресурсів, встановлюючи різні рівні доступу для різних користувачів. Таким чином, клієнт-серверна архітектура у вебсистемі управління службою таксі демонструє свою ефективність та надійність, надаючи міцну основу для розробки високопродуктивних, безпечних та масштабованих веб-застосунків, що забезпечують чудовий досвід користувача. Обмін даними між сервером та клієнтом зображено на рисунку 2.0.

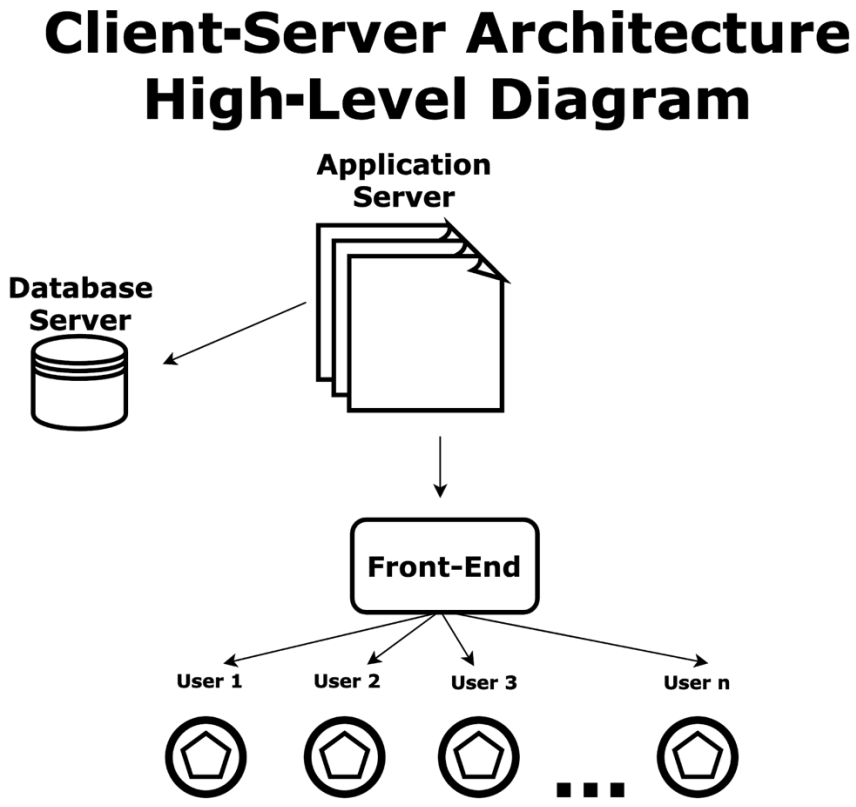


Рисунок 2.0 – Принцип роботи клієнт-серверної архітектури

Обмін даними між клієнтом і сервером є фундаментальним аспектом архітектури клієнт-сервер у розробці вебсистеми для управління службою таксі. Цей процес включає три основні етапи: надсилання запиту від клієнта

(наприклад, водіїв або пасажирів через мобільний застосунок) до сервера, обробка цього запиту на сервері, і повернення відповіді від сервера до клієнта. Протоколи грають критичну роль у забезпеченні цього обміну, встановлюючи формат та механізми передачі даних, що включає забезпечення безпеки комунікації. Зокрема, в контексті вебсистеми для управління таксі, протокол HTTP (Hypertext Transfer Protocol) є найпоширенішим засобом для обміну даними між клієнтом та сервером. HTTP є особливо підходящим для такого роду застосунків, оскільки підтримує широкий спектр форматів даних і є легким у впровадженні, дозволяючи розробникам створювати гнучкі та доступні рішення, які легко інтегрувати з різними веб-технологіями та мобільними платформами.

Використання API у системі аналогічне до інтерфейсу користувача, але замість взаємодії людей з програмою, API дозволяє програмам 'спілкуватися'. Наприклад, додатки використовують API для виконання авторизації користувачів, отримання інформації про доступні поїздки та водіїв, а також для управління замовленнями. REST (Representational State Transfer) є архітектурним стилем, що домінує в розробці веб-застосунківта API. REST використовує стандартні HTTP-методи, такі як GET для отримання даних, POST для створення нових записів, PUT для оновлення існуючих даних та DELETE для їх видалення.

Ці методи забезпечують простоту і зрозумілість системи, а також допомагають уникнути зайвої складності при її розробці. Особливо важливою є здатність REST використовувати кешування, що дозволяє ефективніше використовувати ресурси сервера і забезпечувати швидшу відповідь користувачам. Це вкрай необхідно для вебсистеми управління службою таксі, де швидкість відгуку системи може безпосередньо впливати на задоволеність користувачів. Таким чином, інтеграція продуманих API з використанням REST архітектури в проекті спрощує процес розробки. Також було необхідно додати Socket для створення безперебійного з’єднання з сервером для того щоб водій міг в реальному часі отримувати замовлення з вебсистеми.

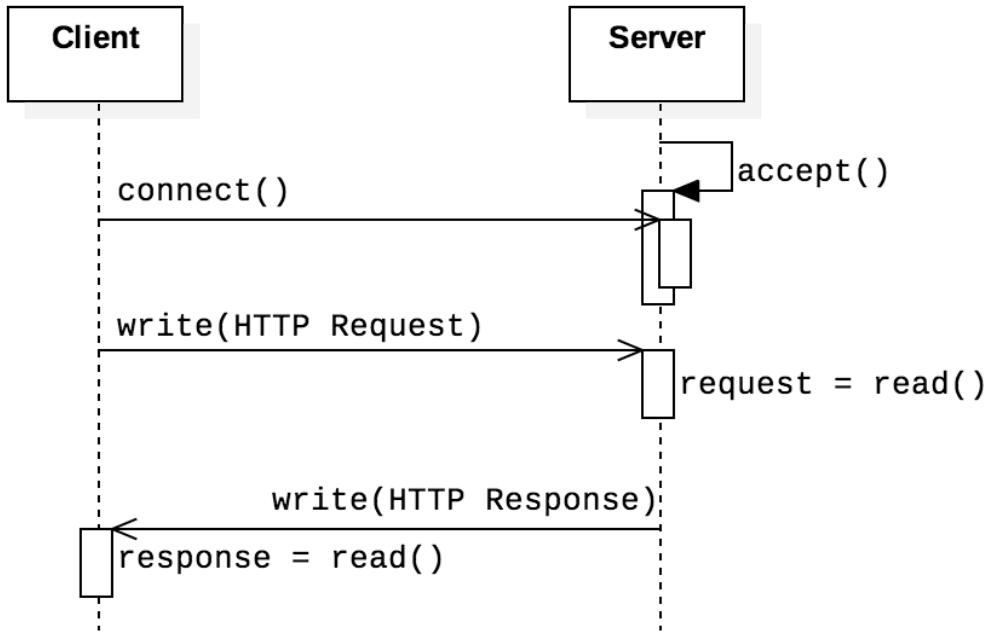


Рисунок 2.1 – Діаграма роботи Socket

Далі необхідно було вибрати між монолітною та мікросервісною архітектурою, це важливий етап, який в майбутньому дасть можливість для ефективного масштабування вебсистеми.

Однією з основних переваг монолітної архітектури є її простота у розробці та розгортанні, оскільки всі компоненти зосереджені в одному монолітному сервісі, що спрощує взаємодію між ними, а також процеси тестування та налагодження. Розгортання такого застосунку на сервері зазвичай відбувається швидко і легко. Крім того, монолітні додатки мають високу продуктивність та менше затримок, оскільки всі компоненти працюють в одному процесі та взаємодіють безпосередньо один з одним.

Ця архітектура також спрощує управління додатком і моніторинг, особливо коли потрібно вносити зміни в систему. Проте монолітна архітектура має свої недоліки, особливо при роботі з великими проектами. Головний недолік полягає в тому, що великий моноліт може бути складним для розуміння та підтримки, а внесення змін може бути ускладненим через тісне зв'язування компонентів коду. Такі обмеження не є критичними для менших проектів, де монолітна структура залишається вигідною.

Мікросервісна архітектура передбачає поділ системи на множину малих, самостійних компонентів, які комунікують між собою через API (інтерфейси програмного забезпечення). Кожен з цих компонентів може бути розроблений, випробуваний і впроваджений незалежно від інших, що сприяє підвищенню масштабованості, доступності та надійності загальної системи. Одна з основних переваг мікросервісної архітектури полягає у її гнучкості та масштабованості.

Завдяки можливості незалежного розгортання кожного сервісу, система може бути адаптована до змінюваних вимог та обтяження, оптимізуючи використання серверних ресурсів. Крім того, кожен мікросервіс може бути легко модифікований або оновлений без впливу на роботу інших частин системи, що спрощує підтримку та оновлення.

Модульність також сприяє легшій повторній використовуваності компонентів в інших проектах. Додатково, мікросервісна архітектура дозволяє використовувати різні мови програмування та технології для кожного сервісу, що надає розробникам можливість вибирати найбільш підходящі інструменти для кожного конкретного завдання. Це створює умови для оптимального вибору технологічних рішень, забезпечуючи кращу продуктивність та ефективність системи. Однак, існують і виклики, пов'язані з мікросервісною архітектурою. Зокрема, складність управління множиною сервісів, їх розгортання та координація можуть ускладнити процес налагодження. Також, збільшена кількість міжсервісних взаємодій може призвести до зростання навантаження на мережу та бази даних, оскільки кожен сервіс взаємодіє з цими компонентами окремо. Загалом, мікросервісна архітектура є відмінним вибором для складних і масштабних проектів, де потрібна висока гнучкість і масштабованість. У той же час, для менших проектів монолітна архітектура може бути більш ефективною через свою простоту та легкість впровадження.

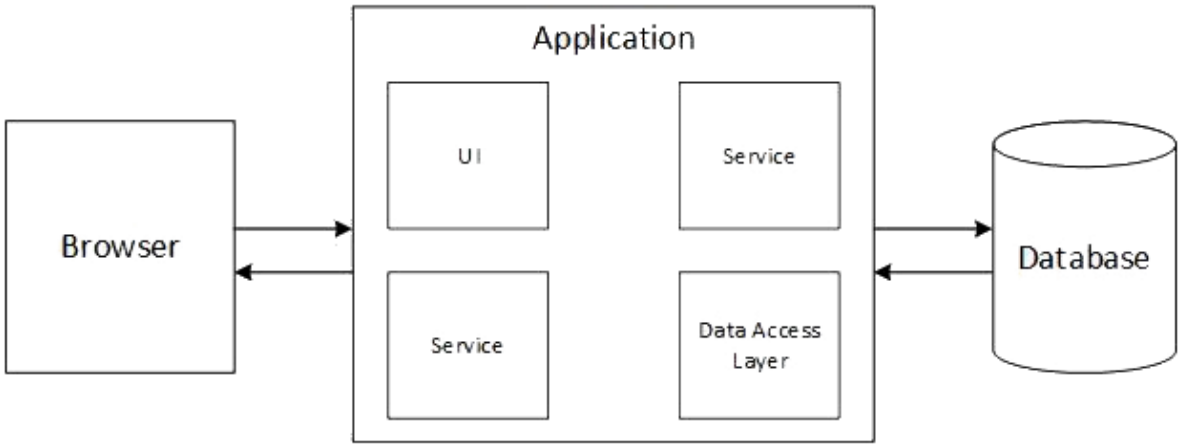


Рисунок 2.2 – Схема архітектури вебсистеми

Враховуючи переваги та недоліки обговорених технологічних підходів, можна стверджувати, що для розробки онлайн-магазину оптимальним варіантом стане монолітна архітектура. Ця архітектура ефективно справляється з основними завданнями, такими як передача даних від сервера до клієнта та обробка вхідних даних з клієнтської сторони, не передбачає виконання складних операцій з даними, що може бути важливо для забезпечення стабільності та високої продуктивності системи. Організація архітектури по рівнях також забезпечує зручність у реалізації та управлінні різними аспектами функціональності додатка.

2.2 Проектування логічної моделі бази даних.

Вибір типу бази даних та проектування логічної моделт є ключовим аспектом у процесі розробки програмного забезпечення, особливо для вебсистеми управління службою таксі так як структура бази даних буде дуже важливою, бо буде зберігатися велика кількість даних з різними типами зв’язків і тому треба це враховувати особливо, що це платформа, що будується на

технологіях NodeJS та ExpressJS і тому було вирішено вибрати MongoDB, як основну базу даних.

Розглянемо переваги MongoDB та бібліотеки Mongoose, які стали вирішальними для цього проекту. MongoDB — це NoSQL база даних, що зберігає дані у форматі документів, що дозволяє гнучко керувати структурою даних. Вона відмінно підходить для застосунків, які потребують швидкого збереження, зміни та обробки великих обсягів неструктурованих даних. Ось деякі із ключових переваг, які роблять MongoDB ідеальним вибором для даної платформи:

– Гнучкість документно-орієнтованого зберігання: MongoDB дозволяє зберігати дані у форматі JSON-подібних документів, що робить базу даних надзвичайно гнучкою для розробників, оскільки не потрібно заздалегідь визначати схему.

– Швидкість та масштабованість MongoDB ефективно функціонує з великими об'ємами даних та високою інтенсивністю запитів, що є критично важливим для динамічних веб-застосунків;

– Бібліотека Mongoose надає широкі можливості моделювання даних для роботи з MongoDB у середовищі Node.js, що забезпечує валідацію даних, кастомізацію запитів та вбудовані методи обробки даних. Це спрощує розробку та зменшує кількість помилок;

– Інтеграція з Node.js та ExpressJS: MongoDB легко інтегрується з Node.js через свою натуральну підтримку JavaScript, що дозволяє розробникам використовувати одну мову програмування на всіх етапах розробки, тим самим підвищуючи продуктивність розробки;

Після вибору бази даних було почато розробку та створення моделей для вебсистеми, було побудовано багато таблиць та зв’язків, у таблицях 1.1 – 1.3 буде описано атрибути кожної сутності:

Таблиця 1.1 – Опис атрибутів сутності «Order»

Атрибут	Опис
1	2
_id	Унікальний ідентифікатор створеного замовлення.
clientName	Ім’я клієнта.
clientPhoneNumber	Номер телефону клієнта.
pickupLocation	Назва стартової точки в замовленні.
destinationLocation	Назва кінцевої точки в замовленні.
waypoints	Проміжні точки в замовленні.
status	Статус замовлення.
price	Остаточна ціна замовлення по завершенню маршруту.
initialPrice	Початкова ціна замовлення при його створенні.
typeOrder	Тип замовлення.
additionalOptions	Додаткові опції.
locationRoute	Координати маршруту замовлення.
coordinatesRoute	Координати маршруту замовлення.
comments	Коментар до замовлення.
distance	Загальна довжина маршруту.
paymentMethod	Метод оплати який вибрав клієнт.
priceKilometers	Ціна за кілометр.
startWaiting	Початок очікування.
endWaiting	Кінець очікування.
stage	Етап замовлення.
commission	Комісія замовлення.
freeWaiting	Кількість часу для очікування.

Таблиця 1.2 – Опис атрибутів сутності «Dispatcher»

Атрибут	Опис
1	2
_id	Унікальний ідентифікато користувача вебсистемою.
email	Емейл диспетчера або адміна.
password	Пароль.
role	Роль в вебсистемі.
city	Місто в якому буде працювати користувач.
zoneVisible	Чи будуть йому видимі зони.

Таблиця 1.3 – Опис атрибутів сутності «Driver»

Атрибут	Опис
1	2
_id	Унікальний ідентифікатор водія.
nickname	Унікальний позивний водія.
priority	Пріоритет взяття замовлення водієм.
secondName	Прізвище водія.
middleName	По батькові водія.
phoneNumber	Номер телефону водія.
nameCar	Назва автомобіля водія.
modelCar	Модель автомобіля водія.
colorCar	Колір автомобіля.
numberCar	Номер автомобіля водія.
dateRegister	Дата реєстрації водія.
status	Статус водія в системі
balance	Баланс водія
statusOnline	Статус чи водій в мережі
moreInformation	Опис до водія
password	Пароль водія

Між двома документами в MongoDB може бути встановлений зв’язок. Це дає змогу за одним документом знаходити інші документи, пов’язані з ним. Розрізняють наступні 3 типи зв’язків:

– «один-до-одного». У цьому зв’язку один документ першої колекції може бути пов’язаний лише з одним документом другої колекції і навпаки;

– «один-до-багатьох» (найпоширеніший тип зв’язку в базі даних). Такий зв’язок означає, що кожен документ першої колекції може бути пов’язаний з декількома документами другої колекції. Це також означає, що кожен документ другої колекції може бути пов’язаний лише з одним документом першої колекції;

– «багато-до-багатьох». Означає, що кожен документ першої колекції може бути пов’язаний з більш ніж одним документом другої колекції. Це також означає, що кожен документ другої колекції може бути пов’язаний з декількома документами першої колекції.

Нормалізація у контексті MongoDB – це процес організації документів у колекціях, який включає створення документів і встановлення зв’язків між

ними відповідно до правил, що забезпечують захист даних і роблять базу даних більш гнучкою, усуваючи надмірність і неузгоджені залежності. Існує кілька правил нормалізації баз даних:

– ненормалізована форма або нульова нормальна форма;
– перша нормальна форма; друга нормальна форма;
– третя нормальна форма; нормальна форма Бойса-Кодда;
– четверта нормальна форма;
– п’ята нормальна форма;
– доменно-ключова нормальна форма;
– шоста нормальна форма.

База даних вважається нормалізованою, якщо вона перебуває як мінімум у третій нормальній формі. У реальному світі нормалізація до третьої нормальної форми є звичайною, стандартною практикою, тому що третя нормальна форма усуває достатню кількість аномалій, водночас продуктивність бази даних, а також зручність її використання не знижується, що не можна сказати про всі наступні форми. Вимога першої нормальної форми полягає в тому, щоб: у колекції не було документів, що повторюються; у полі зберігалися дані одного типу; були відсутні масиви та списки в будь-якому вигляді. Щоб база даних перебувала в другій нормальній формі, необхідно, щоб: її колекції вже перебували в першій нормальній формі; усі колекції мали свій ключ; усі непервинні поля документа залежали від повного ключа (у разі якщо складний).

Якщо ключ складний, тобто складається з декількох полів, то всі інші непервинні поля мають залежати від усього ключа, тобто від усіх полів у цьому ключі. Якщо якийсь атрибут (поле) залежить тільки від одного поля в ключі, значить, база даних не перебуває в другій нормальній формі. Щоб нормалізувати базу даних до третьої нормальної форми, необхідно зробити так, щоб у документах були відсутні непервинні поля, які залежать від інших непервинних полів.

Також щоб побудувати базу даних правильно потрібно конкретно розуміти завдання, які поставлені перед вебсистемою, для кращого розуміння

необхідно будувати UML діаграми, це дасть можливість більш глибоко оцінити проект та побудувати ефективну базу даних.

Ця інтеграція повинна бути здійснена таким чином, щоб забезпечити швидке та ефективне взаємодія даних між користувачем і сервером, зокрема використання REST API для комунікації між клієнтською та серверною частинами. В цілому, проектування функціональної архітектури вебсистеми управління таксі вимагає детального планування та глибокого розуміння взаємодій між різними компонентами системи, що є вирішальним для її успішної реалізації та подальшої експлуатації. Так як вебсистема має багато ключових елементів які містять дуже багато логіку використання діаграм є головним рішенням для побудови ефектиної системи.

У контексті розробки вебсистеми управління таксі, аутентифікація відіграє ключову роль у забезпеченні захисту конфіденційної інформації та доступу до системних ресурсів що зображена на рисунку 2.3.

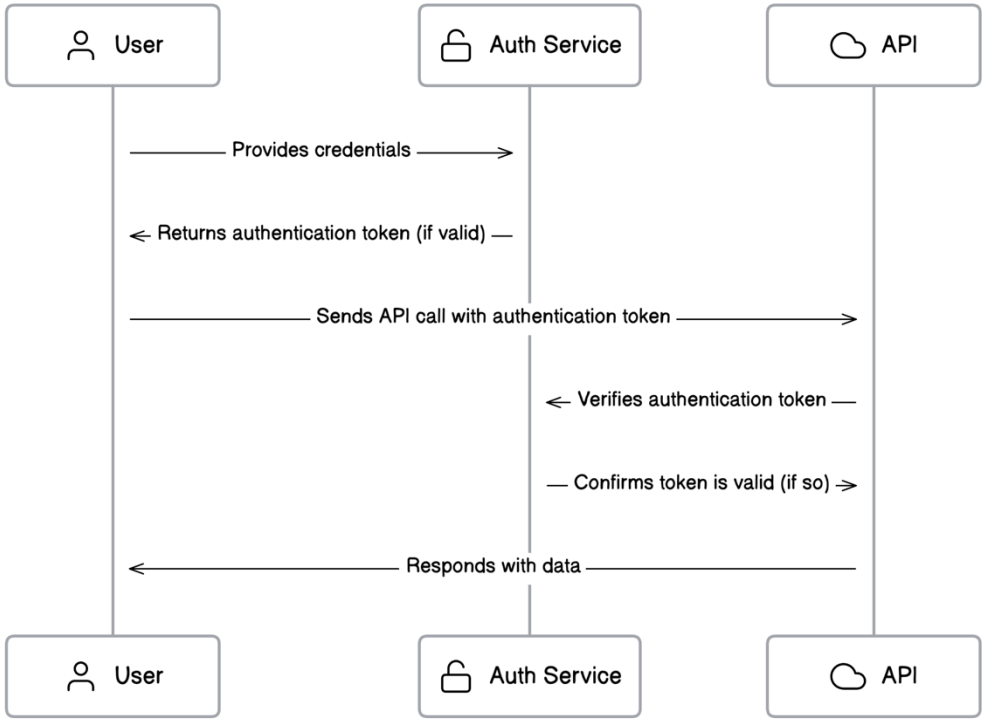


Рисунок 2.3 – Функціональна діаграма аунтифікації

2.3 Проектування інтерфейсу користувача.

Для вебсистеми управління службою таксі необхідно було побудувати інтерфейс необхідної інформації, не перевантажуючи користувача зайвими деталями. Важливо використовувати зрозумілі та зручні для сприйняття кольори, а також округлі форми для графічних елементів, що сприяє кращій візуальній взаємодії та покращує загальне враження від інтерфейсу користувацької панелі для вебсистеми управління службою таксі повинен бути як простим, так і функціональним, забезпечуючи легкий доступ до усієї системи.

Щодо функціональності, важливо розробити інтерфейс таким чином, аби інтерфейс відповідав всім потребам користувачів — від адміністраторів до диспетчерів. Це включає інтуїтивно зрозуміле розміщення елементів керування та інформації, необхідної для ефективного управління замовленнями, моніторингу руху транспорту та взаємодії з клієнтами. Співпраця з замовником, який має доступ до важливих даних і знань про бізнес-процеси служби таксі, є фундаментальною для розробки такої системи. Замовник може надати цінну інформацію, яка допоможе адаптувати систему під конкретні потреби служби, включно з типами запитів, які найчастіше виникають, особливостями роботи диспетчерів, та уподобаннями користувачів. Ця співпраця дозволить не тільки адаптувати інтерфейс під конкретний бізнес, але й забезпечити, що функціональність системи буде повноцінно відповідати всім вимогам, що значно підвищує шанси на успіх проекту в реальних умовах.

Стилізація для Адміністративної панелі яка буде на початку заточена для двох типів користувачів: “Диспетчер” та “Адміністратор” стилі будуть використовувати в кольорах які були основними в бізнесі замовника в першу чергу буде розглядатися використання швидких рішень які були направлені на створення простого інтерфейсу який буде направлений на зручність і вже потім на візуальну складову, для диспетчера буде головним це створення інтуїтивно

дієвого функціоналу який дасть можливість швидко приступити до робити і допускати мінімум помилок під час роботи. Для адміністратора буле головним це побудова якісного інтерфейсу щоб можна зручно слідкувати за роботою системи і бачити як працюють водії. У розробці дизайну для застосунку яким будуть користуватися водіїв було вирішено будувати одночасно і привабливим та функціонально зручний інтерфейс і таким чином вийшло зробити так щоб водій мав одразу доступ до карти та ефіру із усіма замовленнями. Також для водія було передбачено зміну мови в застосунку та використання інших інтерфейсних рішень, які покращують роботу виділяючи основні функціональні можливості застосунку і роблячи інтерфейс для водія привабливим та інтуїтивно зрозумілим бо не завжди служби таксі вкладають багато зусиль в додатки для водіїв.

2.4 Аналіз та вибір технологій і методів реалізації системи

На сучасному етапі розвитку платформи Node.js, вона пропонує численні ефективні рішення для розробки серверних застосунків, орієнтованих на специфічні завдання. З огляду на те, що розробка серверної частини виключно на базовому JavaScript може бути витратною з точки зору ресурсів, було вивчено альтернативні можливості.

Серед них виділяється фреймворк Express.js, який є легким та адаптивним веб-фреймворком для Node.js, спроектованим для ефективного та швидкого створення веб-застосунків. Express.js має значні переваги, включаючи простоту використання та високу гнучкість. Що дозволяє розробникам швидко налаштовувати серверні додатки без необхідності детального вивчення внутрішніх механізмів платформи.

За допомогою Express.js можливо розширити функціональні можливості сервера, використовуючи широкий спектр плагінів та middleware, що

забезпечує сумісність з різноманітними базами даних та веб-сервісами. Крім того, швидкість є ключовою перевагою цього фреймворку: Express.js відомий своєю мінімальною кількістю внутрішніх залежностей та ефективною обробкою запитів, що робить його одним із найшвидших веб-фреймворків для Node.js на ринку. Фреймворк Express.js забезпечує простоту додавання нових функцій до серверного застосунку та спрощує розробку складних веб-застосунків, що легко масштабуються.

Ці характеристики забезпечують оптимальну реалізацію функціональності платформи. Використання Express.js часто супроводжується застосуванням ORM для покращення процесів розробки. Зокрема, для обробки запитів та подій у проекті застосовується механізм Event Loop. Цей механізм, який широко використовується у середовищах JavaScript, таких як Node.js, дозволяє додаткам продовжувати роботу до тих пір, поки у черзі є події для обробки, забезпечуючи неблокуюче виконання і дозволяючи застосунку виконувати інші задачі в періоди очікування подій. Event Loop ефективно управляє асинхронними операціями, такими як мережеві запити або введення/виведення даних, зменшуючи час очікування користувача і підвищуючи продуктивність застосунку.

Цей підхід ідеально інтегрується з розробкою клієнтської частини на React Native та React JS для мобільного застосунку та адміністративної панелі відповідно. Адміністративна панель, розроблена з використанням React JS, React Hooks, Google Maps API, Redux Toolkit, react-router-dom, та MaterialUI, підтримує два типи користувачів: Адмін» і «Диспетчер», забезпечуючи їм необхідні інструменти для ефективного управління. Серверна частина, розроблена на NodeJS, ExpressJS, та MongoDB, додатково оптимізує взаємодію з даними, ефективно використовуючи ресурси та процесорний час, що зображено на рисунку 2.4

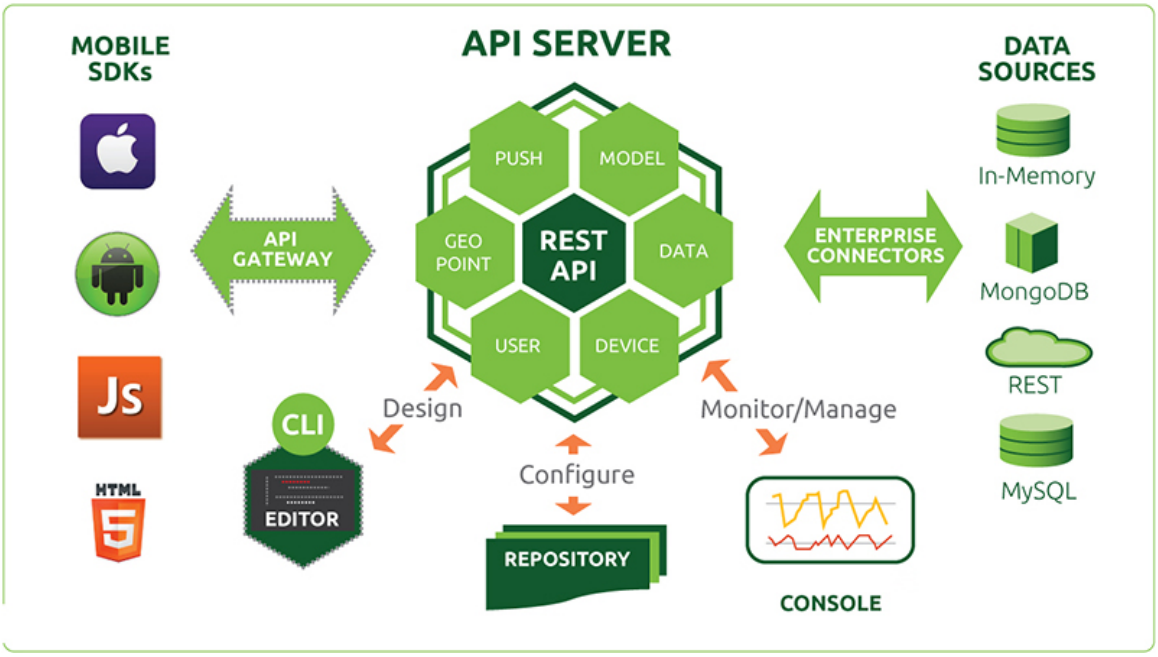


Рисунок 2.4 – Схема роботи вебсистеми

На даній схемі показано, як обрані рішення та технології інтегруються і взаємодіють у процесі функціонування вебсистеми. Використання стеку MERN (MongoDB, Express.js, React, Node.js) забезпечує можливість швидкого та ефективного створення функціоналу без значних затрат часу. Популярність та ефективність цих технологій значно полегшує вирішення будь-яких проблем, що можуть виникнути під час розробки. Мова програмування JavaScript є спільною для всіх компонентів стеку MERN, що спрощує процес розробки. MongoDB, як база даних NoSQL, надає гнучку схему зберігання даних, що ідеально підходить для динамічних застосунків. Express.js працює на серверному боці та забезпечує легкий і гнучкий фреймворк для створення RESTful API. React використовується для побудови користувацьких інтерфейсів, дозволяючи розробляти динамічні та інтерактивні фронтенд-застосунки. Node.js забезпечує серверну частину, дозволяючи обробляти високі навантаження та підтримувати асинхронні операції.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Реалізація бази даних

Для початку було вирішено побудувати серверну частину коду, щоб її побудувати потрібно створити базовий проект NodeJS, для цього потрібно ввести команду npm init -y, далі поставити необхідні бібліотеки для управління базою даних mongoose та бібліотеки для NodeJS, а саме express, мінімалістичний та гнучкий фреймворк для веб-застосунків, побудованих на NodeJS, що надає широкий набір функціональності і nodemon який перезапускає сервер при зміні файлів, npm install express mongoose npm install --save-dev nodemon. Для після цього створюється файл index.js із простим кодом для підняття серверу і також потрібно створити базу даних, ці кроки дадуть можливість побудувати простий функціонал на який далі буде можливість додавати все складніший код і функціонал:

Додавання в код бібліотек необхідний для роботи серверу і роботи з базою даних:

```
const express = require("express");
const mongoose = require("mongoose");
```

Ініціалізація express

```
const PORT = process.env.PORT || 8888;
const app = express();
```

Підключення бази даних, викликається mongoose і передається згенероване посилання для підключення до бази даних, також використовуються додаткові параметри для успішного підключення. Важливо також пам’ятати, що це повинно бути асинхронно, щоб підключення до бази даних відбулося успішно.

```
await mongoose.connect(process.env.MONGO_URL, {
     useNewUrlParser: true,
     useUnifiedTopology: true,
});
```

Далі на наступних рисунках буде показано, як створити свою базу даних в MongoDB:

Потрібно зайти на головну сторінку і вибрати кнопку «New project», щоб створити новий проект, що зображено на рисунку 2.5

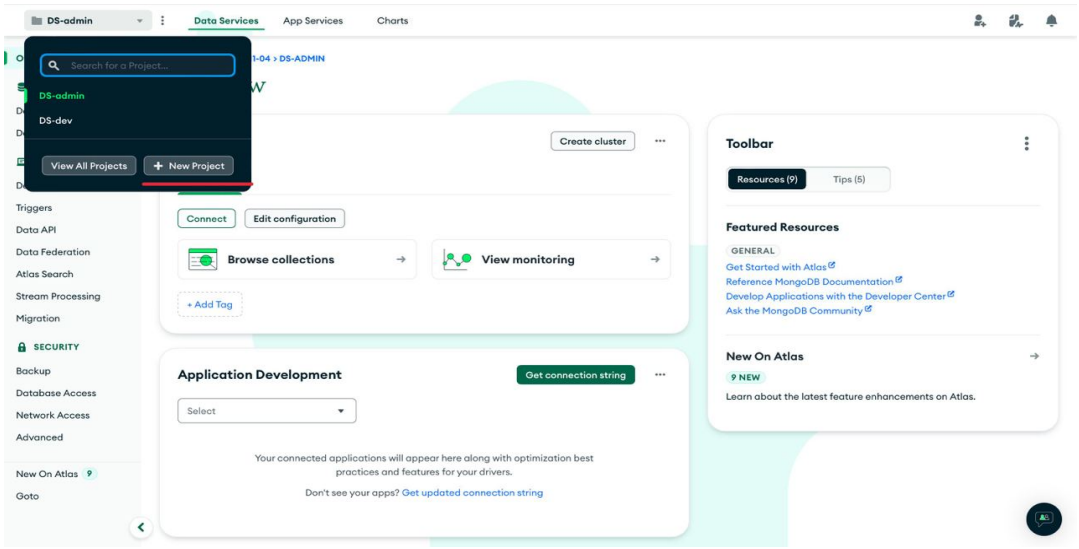


Рисунок 2.5 – Створення нового проекту

Далі потрібно буде ввести назву проекту та заповнити інші необхідні дані, що зображено на рисунку 2.6.

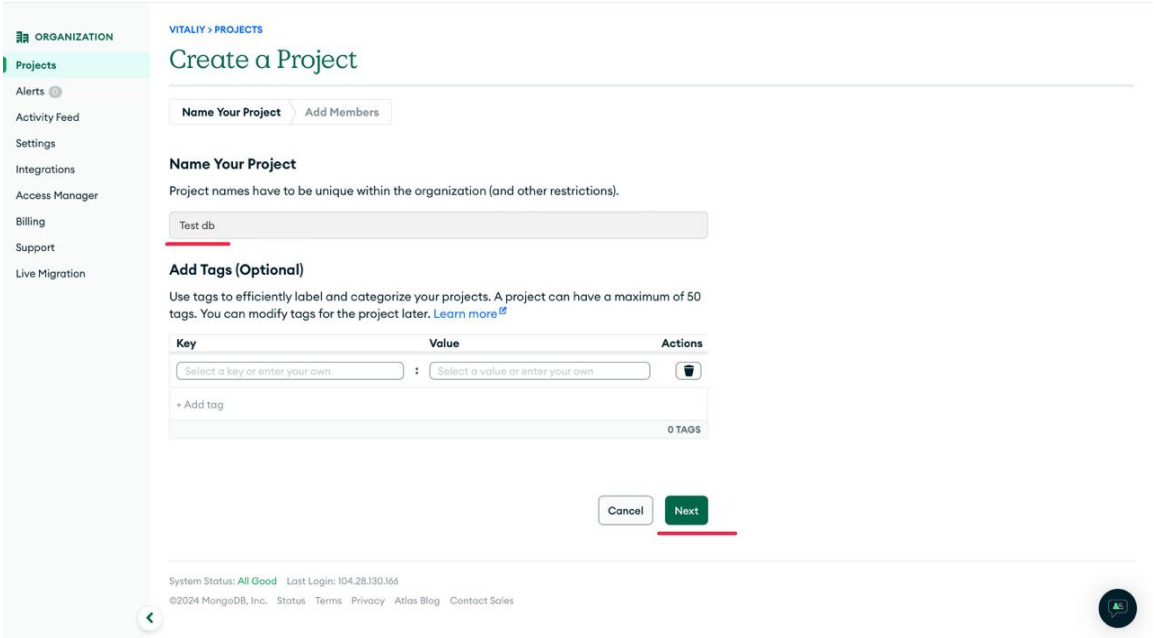


Рисунок 2.6 – Додавання назви для бази даних

Далі при необхідності можна додати інших учасників для бази даних, на рисунку 2.7 якщо база даних буде містити тільки одного учасника, то можна пропустити цей пункт.

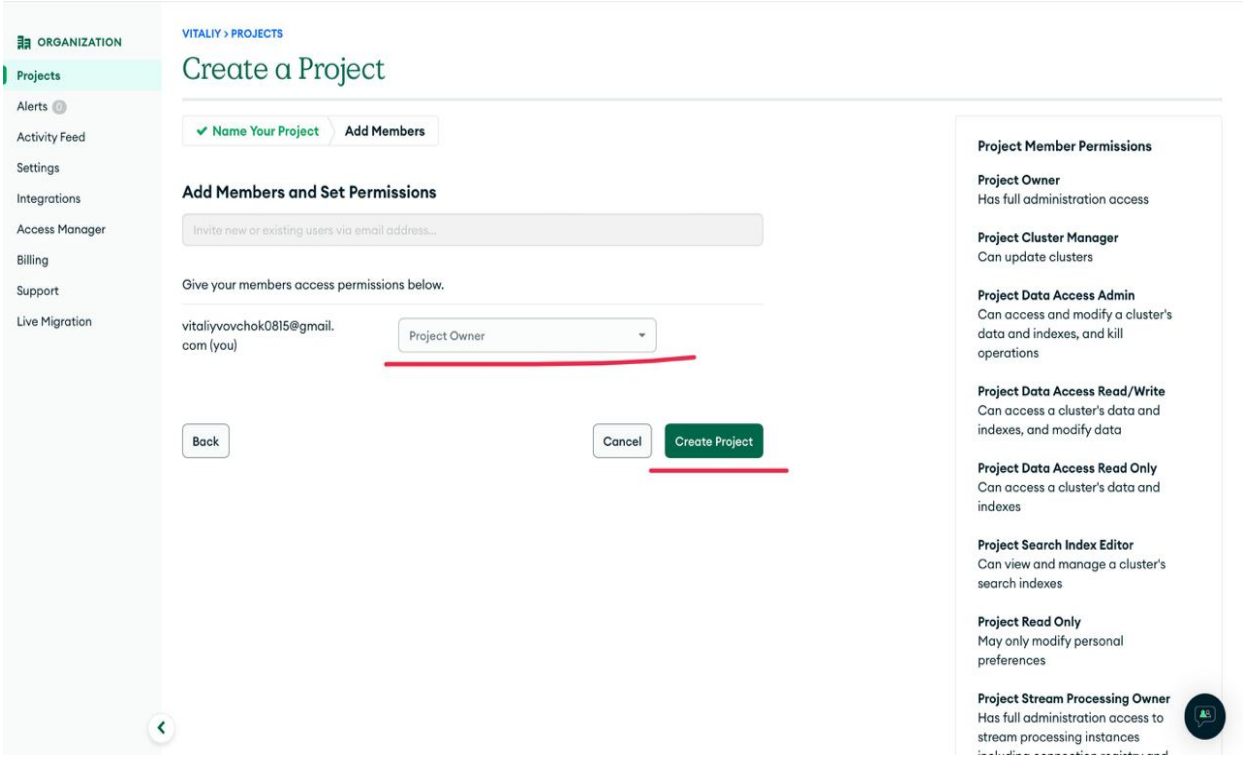


Рисунок 2.7 – Додавання учасників до бази даних

Далі після натискання кнопки «Create Project», необхідно натиснути на кнопку «+Create», для створення кластеру

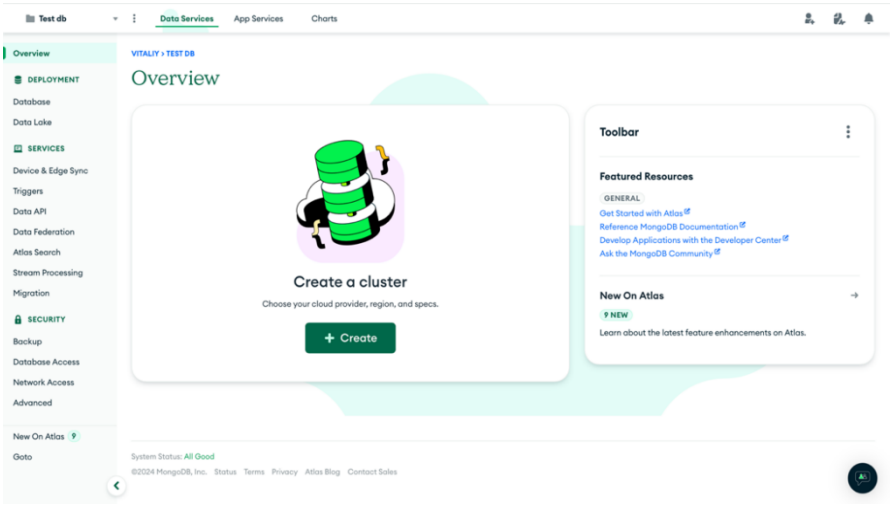


Рисунок 2.8 – створення бази даних

Далі потрібно вибрати тарифний план який буде задовольняти вимоги серверу та країну в якій буде знаходитися сервер і також інші конфігурації.

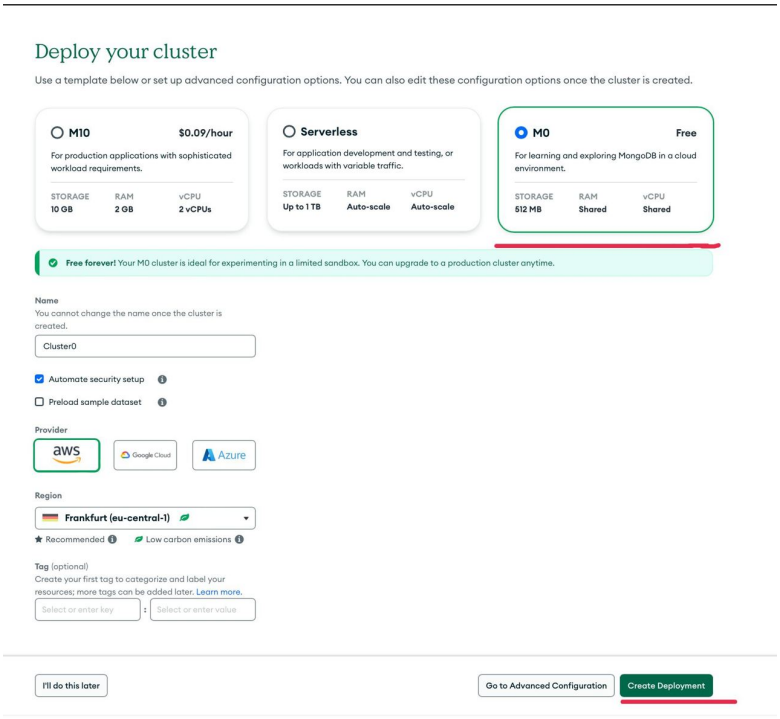


Рисунок 2.9 – конфігураці бази даних

Далі після конфігурації потріьно скопіювати згенерований пароль та логін щоб потімпідключитися до бази даних далі потрібно буде вибрати потрібний метед підключення до бахи даних.

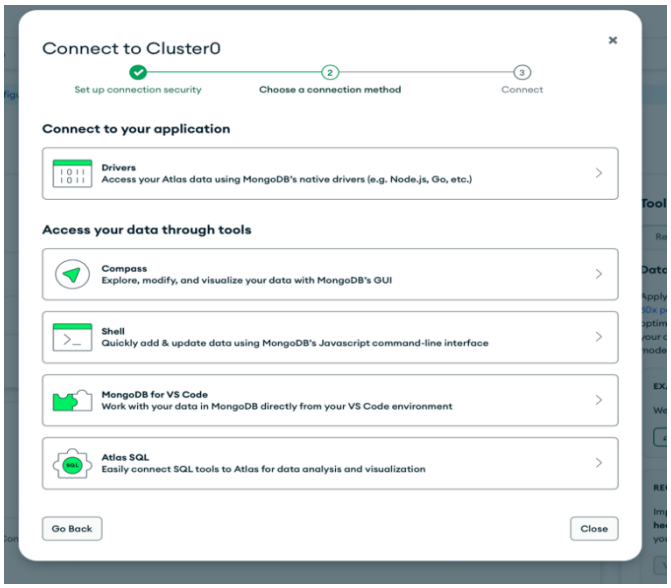


Рисунок 3.0 – підключення до бази даних

В самому кінці буде запропоновано посилання по якому через NodeJS можна буде підключитися до бази даних прямо з сервера mongodb+srv://<user>:<password>@cluster0.px76ecj.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0.

Після того, як було створено базу даних потрібно додати це посилання в код для який відповідає за підключення до бази і після успішного підключення можна починати налаштування моделей для роботи із даними.

Модель водія

```
const {Schema, model} = require("mongoose");

const Driver = new Schema({
    nickname: {type: String, required: true, unique: true},
    priority: {type: Number, required: true},
    name: {type: String, required: true},
    secondName: {type: String, required: true},
    middleName: {type: String},
    phoneNumber: {type: String, required: true, unique: true},
    email: {type: String, required: true, unique: true},
    nameCar: {type: String, required: true},
    modelCar: {type: String, required: true},
    colorCar: {type: String, required: true},
    numberCar: {type: String, required: true},
    dateRegister: {type: String, required: true},
    status: {type: String, required: true},
    balance: {type: Number},
    statusOnline: {type: Boolean},
    lastDateStatusOnline: {type: String},
    dateLastOrder: {type: String},
    moreInformation: {type: String},
    carPhoto: {type: String},
    driverLicensePhoto: {type: String},
    carInsurancePhoto: {type: String},
    password: {type: String, required: true},
    currentOrders: { type: Schema.Types.ObjectId, ref: 'Order' },
})

module.exports = model('Driver', Driver)
```

Для створення моделі потрібно створити файл і додати в нього підключення бібліотеки та далі потрібно вказати, які поля буде містити модель, для позначення типу даних використовується ключ type, якому вказуються який буде тип даних приймати конкретно це поле. Також для позначення чи це поле обов’язкове використовують required значенням true або false позначають чи воно обов’язкове.

Також через ref, та спеціальні налаштування type, будують зв’язки з базою даних і в кінці для того щоб модель можна було використати в коді її експортують.

```
const OrderModel = require('../models/orderModel');

await DriverModel.findOne({email});

await DriverModel.findByIdAndUpdate(driverData.id, {statusOnline:
driverData.statusOnline}, {new: true});
```

Для того щоб використати модель для роботи із даними її необхідно експортувати і далі вже викликати різні методи та відповідно для них передавати дані, щоб вносити зміну в базу даних.

Ще однієї перевагою MongoDB є те що в мережі є досить багато відео та інших матеріалів для побудови та набуття навичок роботи і MongoDB та NodeJS.

Для перегляду даних в MongoDB можна використовувати або застосунок MongoDB Compass або офіційний сайт.

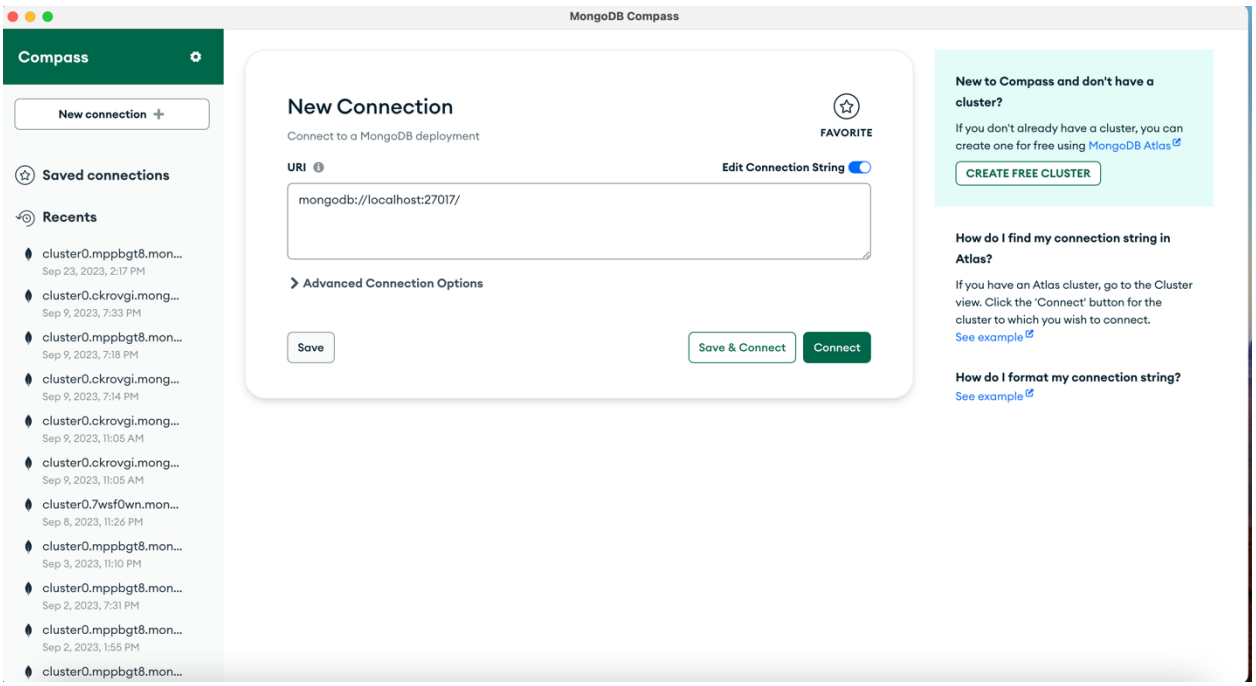


Рисунок 3.1 – MongoDB Compass

На рисунку 3.1 зображено застосунок яким зручно користуватися так як в ньому відкритий широкий функціонал для розробки.

3.2 Реалізація модулів системи

Після налаштування основних елементів пов’язаних із серверною частинию, наступним етапом було перехід до створення клієнтської частини, а саме адмін панелі. Для створення проекту на ReactJS потрібно запустити такі команди:

1) npx create-react-app my-app
2) cd my-app
3) npm start

Після запуску потрібно встановити ще бібліотеку для Material UI так, як саме на основі цієї бібліотеки будуть будуватися стилі в проекті.

npm install @mui/material @emotion/react @emotion/styled. Окрім стилів треба, ще буде розпочати роботу з Google Maps щоб побудувати цей важливий модуль необхтдно, встановити бібліотеку для роботами з картами в React. npm i -S @react-google-maps/api, також необхідно встановити необхідні бібліотеки для React Native npm i react-native-maps npm i react-native-maps-directions. Щоб можна було повноцінно використовувати карти, необхідно сконфігорувати Google Maps. Для цього потрібно буде перейти на Google Cloud Console на рисунку 3.2 буде зображено створення нового проекту.

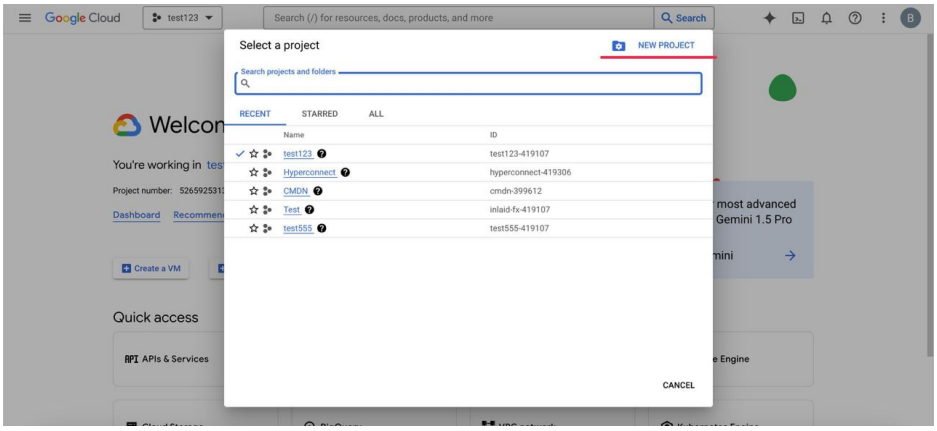


Рисунок 3.2 – Створення нового проекту

Використання Google Maps буде дуже необхідне для розробки так як карти будуть використовуватися в адміністративній панелі для прорахунку маршруту та визначення його вартості, а в застосунку вони будуть використовуватися для інформативності водія, розуміння де знаходиться клієнт і де знаходяться стартова, проміжні та кінцева точка. Щоб реалізувати цей функціонал було вибрано саме Google Maps, є інші рішення як Mapbox або інші безкоштовні альтернативи, але в усіх них немає того набору функцій які є в Google Maps.

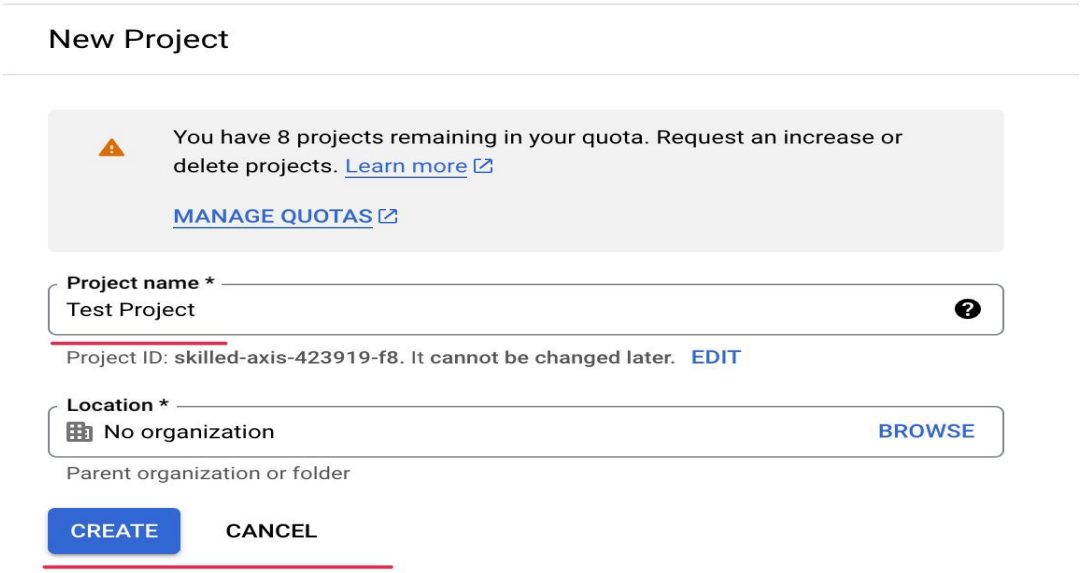


Рисунок 3.3 – Додавання назви проекту

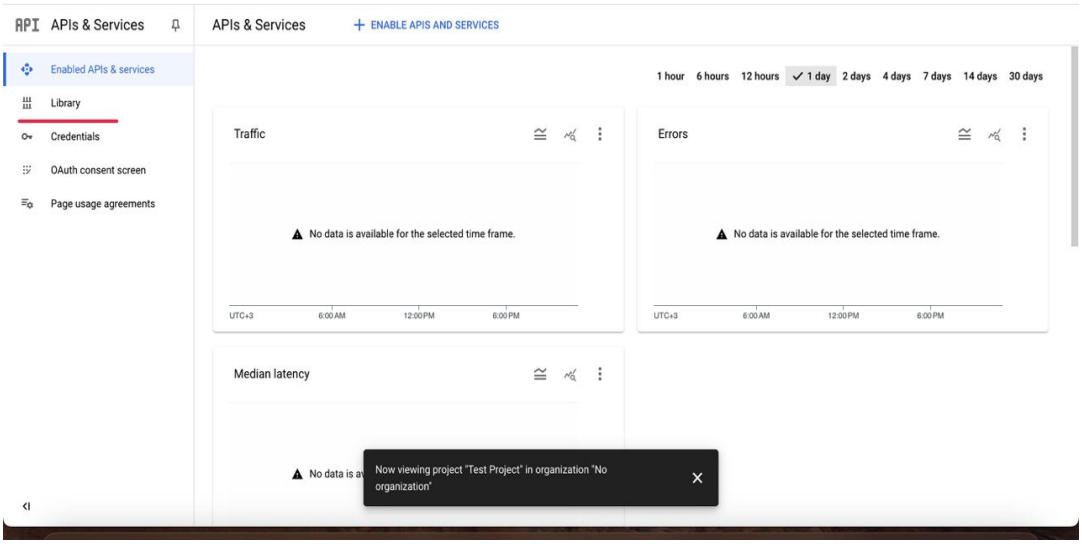


Рисунок 3.4 – Інтерфейс Google Cloud Console

На рисунку 3.3 та 3.4 було зображено як створюється проекті і далі після його створення потрібно буде відкрити Library та вибрати Maps SDK for iOS, Maps SDK for Android, Maps JavaScript API, це дасть можливість відобразити карти на вебсайті та у застосунку для водія. Далі необхідно буде підключити Geocoding API це дасть можливість отримувати координати місця вводячи назву вулиці коли буде клієнт давати адреси, щоб відображати їх на карті. Далі для можливості додавати візуальні елементи на карту необхідно додати Places API, Places API (New), далі для відображення маршрутів в застосунку на карті необхідно використати Routes API, Directions API. Як застосувати ці бібліотеки буде показано на рисунку 3.5.

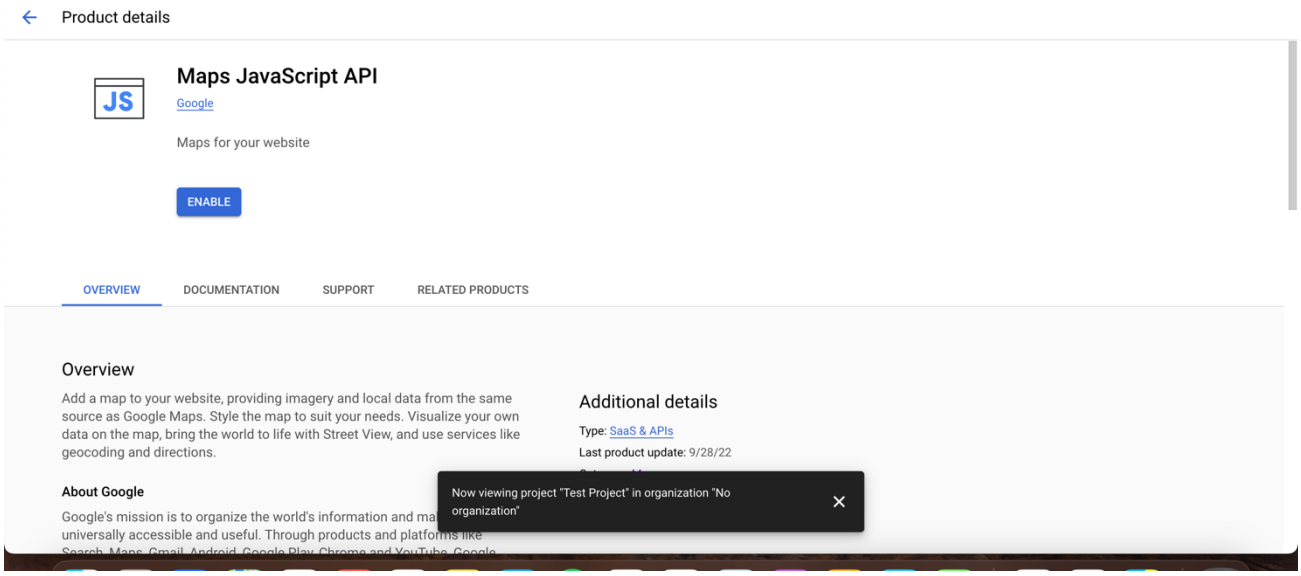


Рисунок 3.5 – Підключення бібліотеки

Після вибору необхідної бібліотеки, необхідно натиснути на кнопку «Enable», далі після додавання бібліотеки потрібно перейти на сторінку Credentials створити API ключ і зайти в налаштування апі ключа що зображено рисунку 3.6. За допомогою цих бібліотек буде можливість відкривати новий функціонал у використанні Google Maps, але необхідно враховувати що ці бібліотеки можуть бути платними.

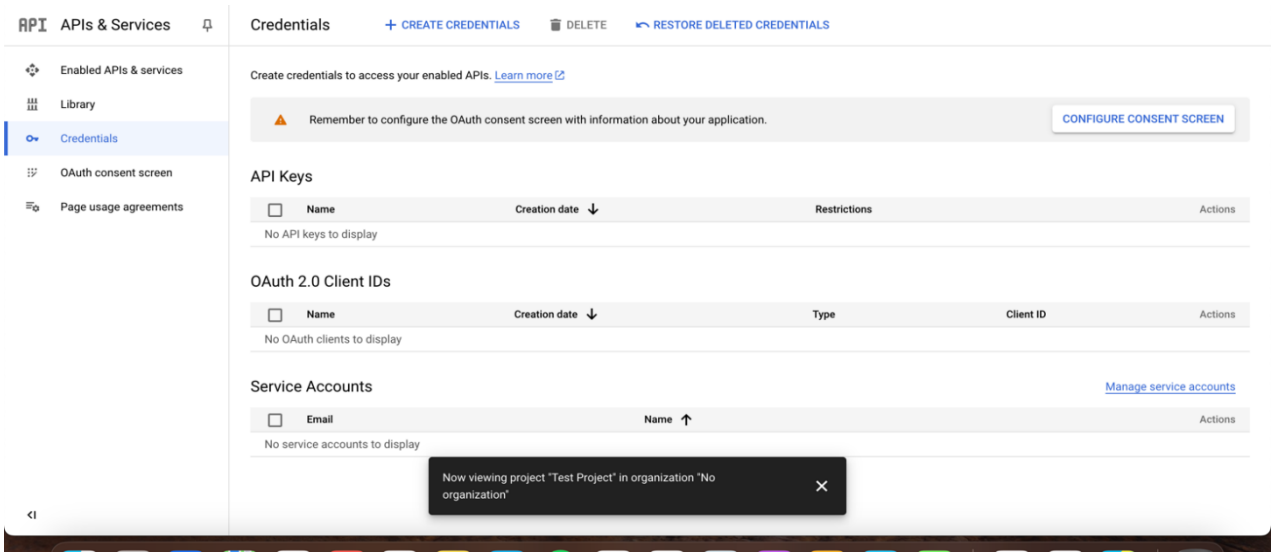


Рисунок 3.6 – Створення Google Maps API Key

Далі потрібно зайти в налаштування апі ключа і там вибрати необхідні конфігурації Set an application restriction, тут необхідно вибрати Websites, далі треба додати адресу вебсайта у поле Website restrictions у даному випадку це буде http://localhost:3000/. Далі необхідно перейти до API restrictions і вибрати там пункт Restrict key після чого треба буде вибрати бібліотеки які ми підключали раніше і додати їх в Selected APIs і після цього натиснути на кнопку «Save». Це буде зображено на рисунках 3.7 та 3.8.

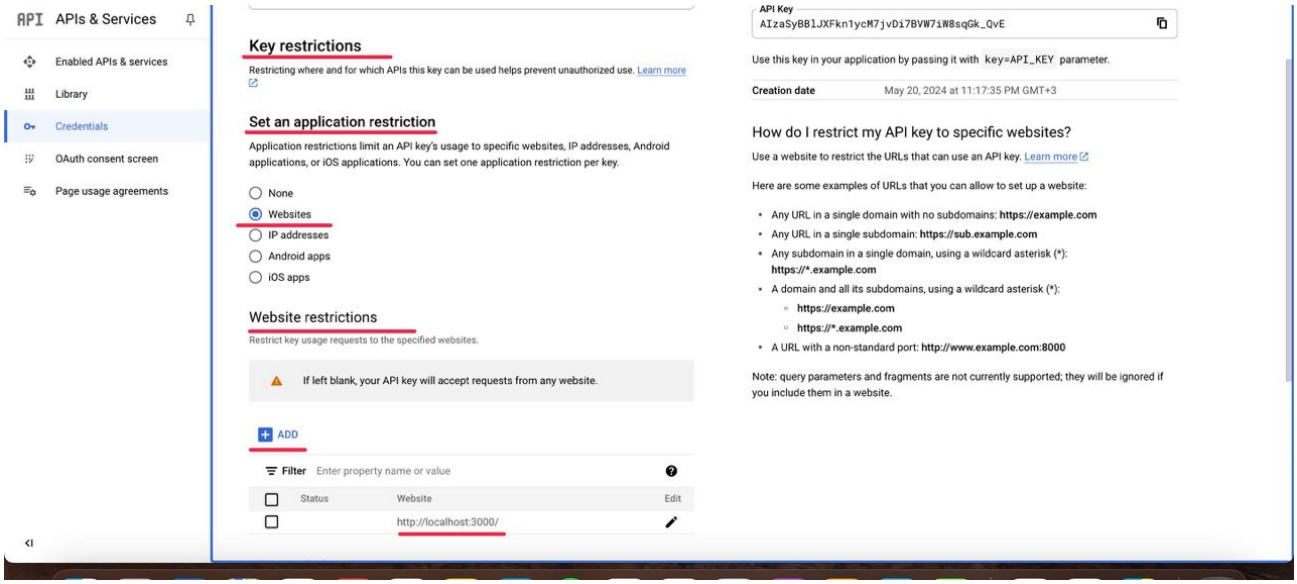


Рисунок 3.7 – Налаштування API Key

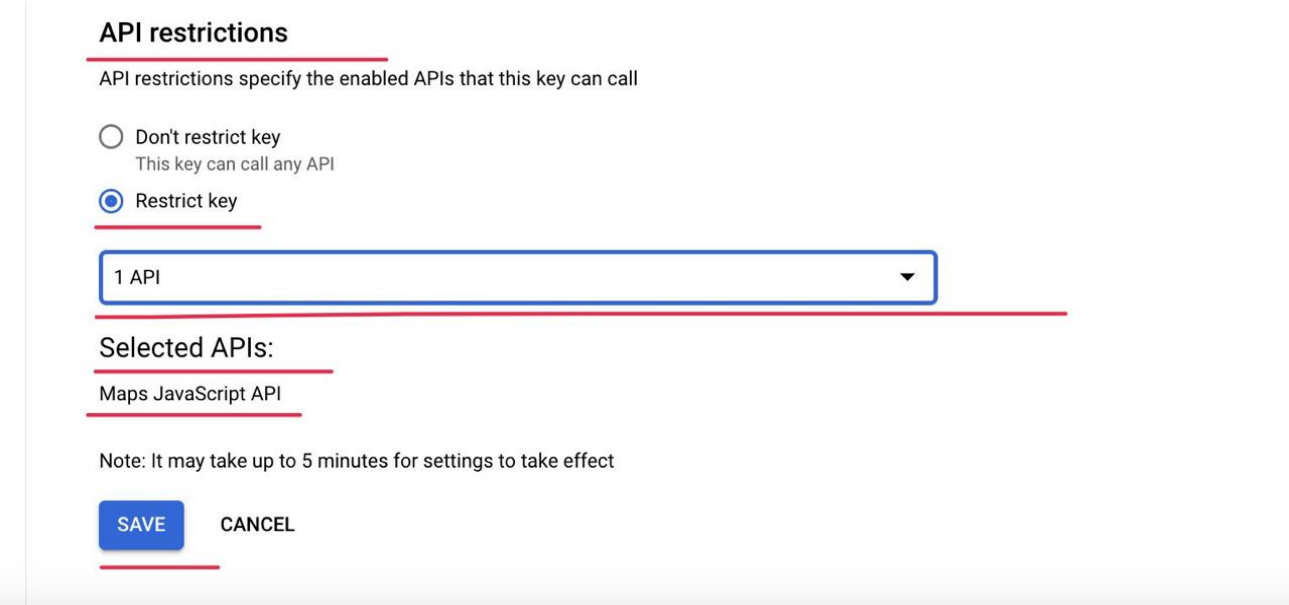


Рисунок 3.8 – Налаштування API Key

Дали після цих дій можна вже використовувати бібліотеки для React @react-google-maps/api та use-places-autocomplete, а для React Native react-native-maps, react-native-maps-directions, react-native-geolocation-service. В цьому коді зображено використання пошуку адреси з обмеженнями по країні та по площі яка має входити в пошук.

```
      import { Autocomplete } from '@react-google-maps/api';
function onLoad(autocomplete) {
    setSearchResult(autocomplete);
  }

  function onPlaceChanged() {
    if (searchResult != null) {
      const place = searchResult.getPlace();
      let lat = place.geometry.location.lat();
      let lng = place.geometry.location.lng();
      console.log('place', place);
      console.log(
        "place.formatted_address.replace(/,\\s*(Ukraine|Україна|Украина).*/,
'')","
        place.formatted_address.replace(/,\s*(Ukraine|Україна|Украина).*/, '')
      );
      dispatcher(setCenter({ lat, lng }));
      onSelect(
        { lat, lng },
        place.formatted_address.replace(/,\s*(Ukraine|Україна|Украина).*/, ''),
        id
      );
    } else {
      alert('Please enter text');
```

```
    }

<Autocomplete
          onPlaceChanged={onPlaceChanged}
          onLoad={onLoad}
          restrictions={{ country: 'ua' }}
          bounds={{
            north: data?.points?.north !== undefined ? data?.points?.north : 0,
            east: data?.points?.east !== undefined ? data?.points?.east : 0,
            south: data?.points?.south !== undefined ? data?.points?.south : 0,
            west: data?.points?.west !== undefined ? data?.points?.west : 0,
          }}
          options={{ strictBounds: !isAllCities }}
        >
          <TextField
            margin="dense"
            required
            fullWidth
            name={placeholder}
            label={placeholder}
            placeholder={placeholder}
            type="text"
            id={placeholder}
            autoComplete={placeholder}
          />
        </Autocomplete>
```

Ще важливим модулем було побудови системи, яка б могла в реальному часі здійснювати обмін даними і для цього було вибрано Socket, для того щоб це реалізувати для початку необхідно поствити на систему усі необхідні бібліотеки для сервера npm install socket.io та для клієнтської частини npm install socket.io-client.

Для на сервері непотрібно ініціалізувати Socket.io для цього неоюхідно імпортувати бібліотеку та поставити прослуховування на події підключення створення замовлення та інші події які буде додано

```
  io.on('connection', socket => {
    console.log('Новий клієнт підключився');

    // Отримання всіх замовлень із статусом "pending" з бази даних
    orderModel.find({status: 'pending'}, (err, pendingOrders) => {
        if (err) {
            console.error('Помилка при отриманні замовлень:', err);
            return;
        }
        const reversedOrders = pendingOrders.reverse();
        // Відправка всіх замовлень до водія при його підключенні
        socket.emit('all_orders', reversedOrders);
    });

    socket.on('create_order', async newOrder => {
        try {
```

```
            // Додавання нового замовлення до масиву замовлень
            const createdOrder = await orderService.create(newOrder);
            // Повідомлення всіх підключених водіїв про нове замовлення
            io.emit('new_order', createdOrder);
        } catch (error) {
            console.error('Помилка при створенні замовлення:', error);
        }
    });
```

Далі необхідно на клієнтській частині прослуховувати так само відповіді від серверу в реаліному часі і для того щоб мати можливість отримати дані по всьому проекту було вирішено побудувати систему за допомогою React Context.

```
      import React, { createContext, useContext, useEffect, useRef } from
'react';
import socketIOClient from 'socket.io-client';

const SocketContext = createContext();

export const useSocket = () => {
  return useContext(SocketContext);
};

export const SocketProvider = ({ children }) => {
  const socketRef = useRef(null);

  useEffect(() => {
    socketRef.current = socketIOClient(`${process.env.REACT_APP_SOCKET_URL}`);

    socketRef.current.on('connect', () => {
      console.log('Connected to the server');
    });

    return () => {
      socketRef.current.disconnect();
    };
  }, []);

  return (
    <SocketContext.Provider value={socketRef}>
      {children}
    </SocketContext.Provider>
  );
};
```

Ще одним важливим модулем було побудова в застосунку для водія можливості швидкого входу через Android Floating Bubble. Щоб це реалізувати було npm і react-native-floating-bubble, цей функціонал дозволяв встановити швидкий вхід в застосунок поверх інших застосунків щоб водій при отриманні нового замовлення міг одразу його взяти. Також в застосунку було передбачено, що водій може відключити цей функціонал.

```
    componentDidMount() {
  this.checkLanguage();
  this.checkNavigator();

  DeviceEventEmitter.addListener('floating-bubble-press', () => {
    reopenApp();
    hideFloatingBubble().then(() => console.log('Floating Bubble Removed'));
  });

  AppState.addEventListener('change', this._handleAppStateChange);
}

_handleAppStateChange = async nextAppState => {
  try {
    const storedValue = await AsyncStorage.getItem('checkboxState');
    if (storedValue !== null) {
      await initialize().then(() =>
        console.log('Initialized the bubble manage'),
      );
    }
    const {login} = this.props;

    if (
      this.state.appState.match(/inactive|background/) &&
      nextAppState === 'active' &&
      login
    ) {
      ScreenLock.acquire();
      PartialWakeLock.release();
      const value = await AsyncStorage.getItem('checkboxState');
      if (JSON.parse(value) === true) {
        await hideFloatingBubble().then(() =>
          console.log('Floating Bubble Removed'),
        );
      } else {
        await hideFloatingBubble().then(() =>
          console.log('Floating Bubble Removed'),
        );
      }
    }
    if (
      this.state.appState === 'active' &&
      nextAppState.match(/inactive|background/) &&
      login
    ) {
      ScreenLock.release();
      PartialWakeLock.acquire();
      const value = await AsyncStorage.getItem('checkboxState');
      if (JSON.parse(value) === true) {
        await showFloatingBubble(10, 10).then(() =>
          console.log('Floating Bubble Added'),
        );
      } else {
        await hideFloatingBubble().then(() =>
          console.log('Floating Bubble Removed'),
        );
      }
    }
    this.setState({appState: nextAppState});
  } catch (error) {
    console.error('Error loading checkbox state:', error);
  }
};
```

3.3 Інструкція користувача

Інструкція користувача призначена для надання допомоги користувачам у використанні певного продукту, послуги або програмного забезпечення. Більшість таких керівництв містять як текстові інструкції, так і відповідні зображення. У випадку комп'ютерних програм зазвичай включаються скріншоти інтерфейсу користувача, а керівництва для обладнання часто містять зрозумілі, спрощені схеми. Мова, що використовується в керівництві, адаптована до цільової аудиторії, при цьому технічні терміни зведені до мінімуму або ретельно пояснені.

Опис роботи повинен містити наступні дані:

– загальні відомості про розроблену вебисистему;

– вхідні та вихідні дані;

– функціональне призначення вебсистеми;

– послідовність дій користувача, які забезпечують виконання програми на усіх етапах;

Вебсистема керування службою розроблена у вигляді веб-сайту за допомогою мови програмування JavaScript у середовищі розробки Webstorm. Для роботи із системою користувач повинен мати досвід користування комп’ютером, а також пристрій із встановленим браузером та стабільним підключенням до мережі Інтернет.

Вебсистема керування службою таксі розроблена для зручного та ефективного створення замовлень для водіїв, які будуть отримувати ці замовлення в реальному часі. У системі реалізовані наступні функціональні можливості:

– додавання водіїв у систему;

– створення замовлень в реальному часі;

– авторизації та реєстрації користувача;

– налаштування особистого кабінету водія;

– керування системою від імені адміністратора: блокування та видалення аккаунтів водіїв.

Першим що побачить користувач перед собою це вікно входу в систему

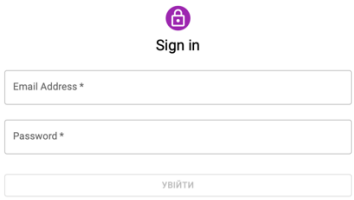


Рисунок 3.9 – Сторінка входу в вебсистему

Залежно від ролі користувача його перекине або на сторінки для диспетчера або на сторінки для адміністратора вебсистеми.

Якщо користувач має роль диспетчер то його перекине на першу сторінку створення замовлення. В користувача буде справа карта, де буде можливість побачити зони та побудований маршрут. Для того щоб побудувати маршрут користувачу потрібно ввести стартову і кінцеву точки, вибрати тип замовлення далі натиснути на кнопку Створити маршрут і після опрацювання маршрута потрібно вказати номер телефона клієнта, далі після натискання кнопки Створити замовлення замовлення буде відправлено до водіїв. Це зображено на рисунку 4.0

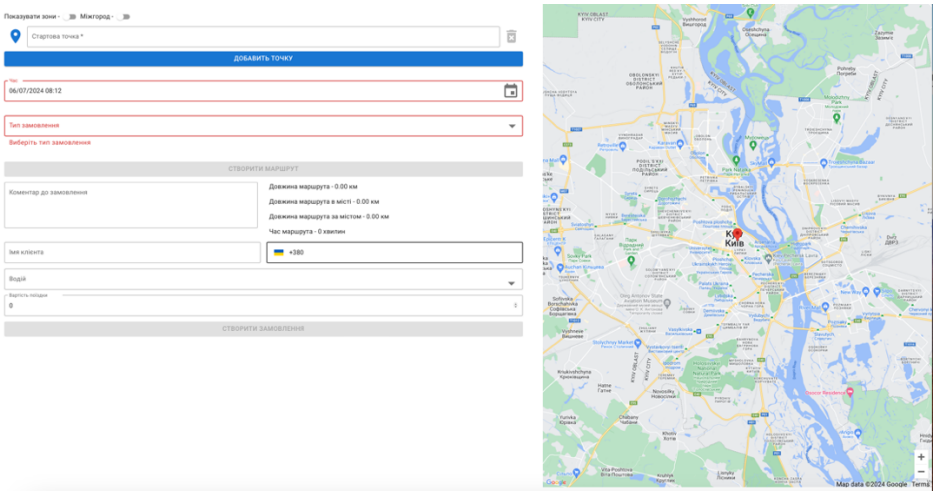


Рисунок 4.0 – Екран створення замовлення

На рисунку 4.1 зображено як виглядають зони на карті та побудований маршрут.

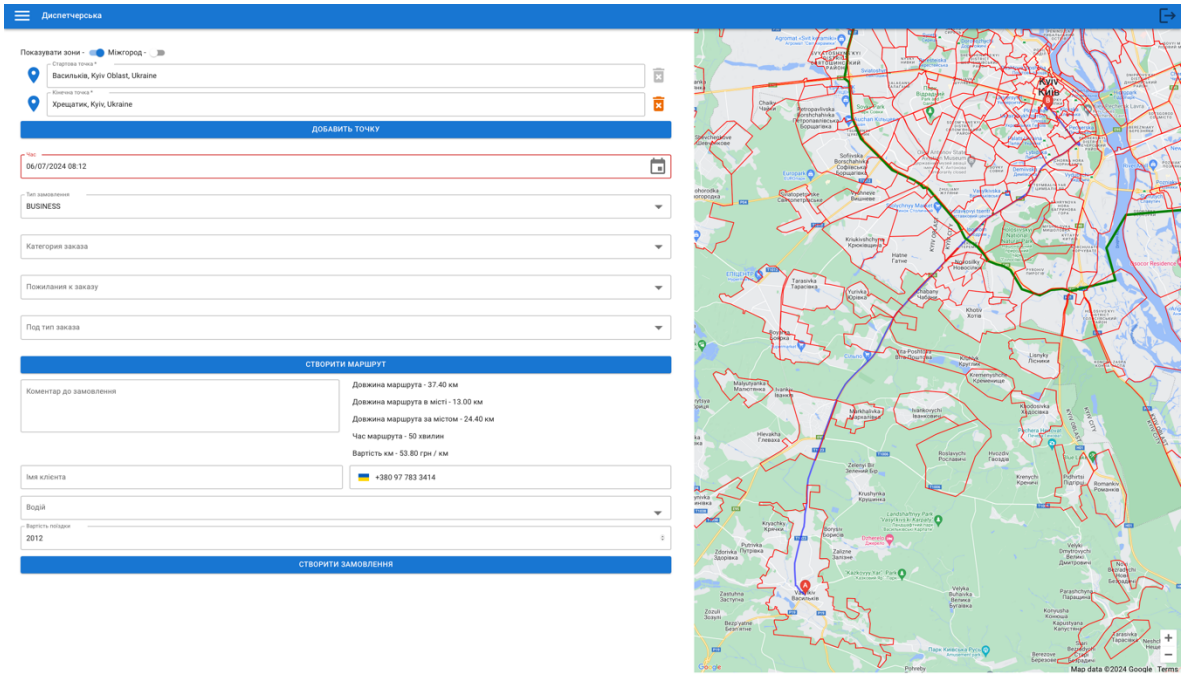


Рисунок 4.1 – Побудова маршруту

Після створення маршруту в диспетчера з’являється можливість переглянути замовлення у списку замовлень, на цьому списку відображаються усі замовлення і їхні статуси та необхідна інформація

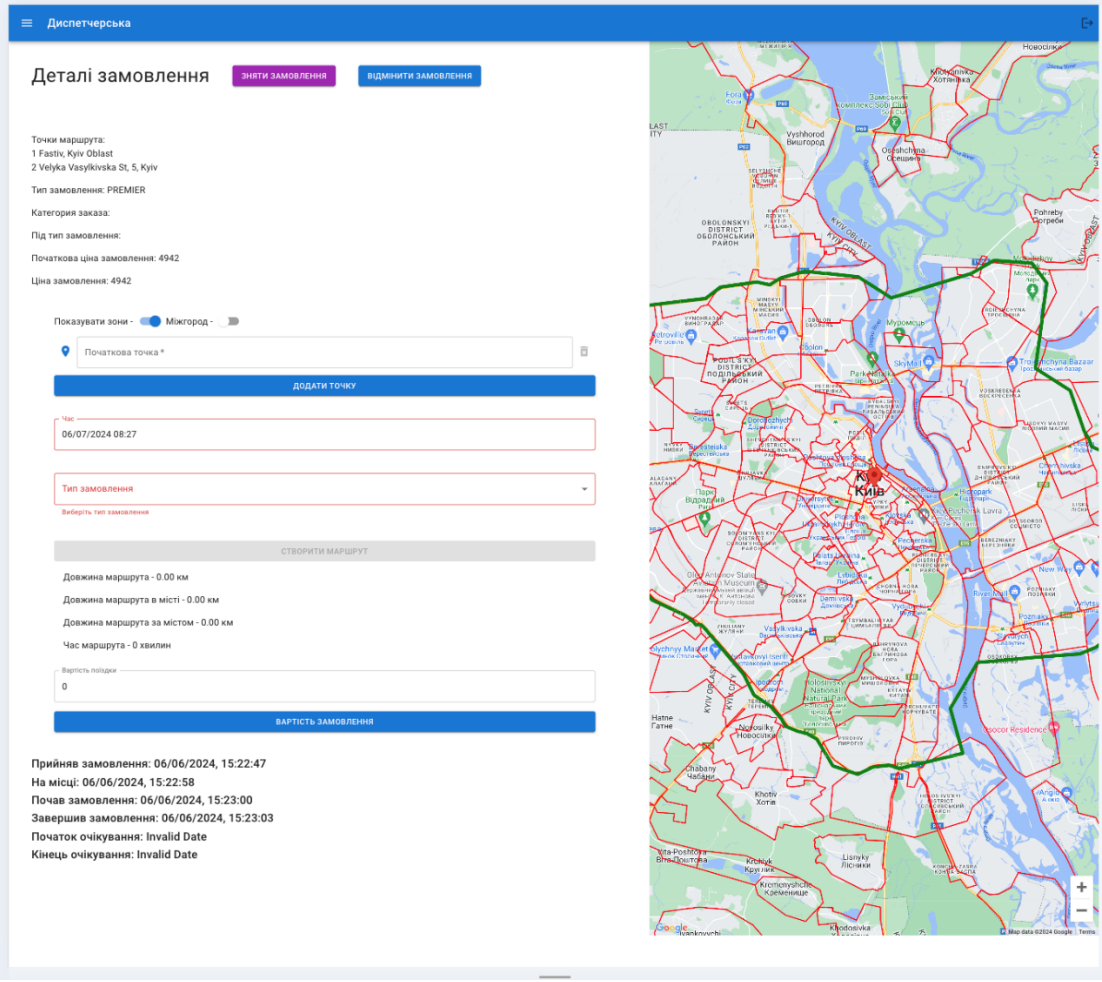
Водій	Етап	Статус	Номер клієнта	Початковий адрес	Ціна	Редагувати
	не взятий	в очікування	+380 95 352 7342	Obolonskyi Ave, 1Б, Kyiv	1129	
HROPOTOVA	замовлення закінчене	закінчений	+380 97 783 3414	Fastiv, Kyiv Oblast	4942	
	не взятий	в очікування	+380 97 783 3414	Fastiv, Kyiv Oblast	4942	
	не взятий	в очікування	+380 97 783 3414	Fastiv, Kyiv Oblast	4942	
	не взятий	в очікування	+380977833414	Horenychi, Kyiv Oblast	2465	
	не взятий	в очікування	+380963632564	Sportyvna Square, 1a, Kyiv	1190	
	не взятий	в очікування	+380973635254	Tsentralna St, 8a, Irpin, Kyivs'ka oblast	1298	
1	замовлення закінчене	закінчений	+380977834142	Avtovokzal, Kyiv	595	
	не взятий	в очікування	+380977834142	Antonovycha St, 48, Kyiv	595	
	не взятий	в очікування	+380977834142	Antonovycha St, 48, Kyiv	1375	

Rows per page: 10 1–10 of 76

Рисунок 4.2 – Сторінка списку замовлень

У списку замовлень диспетчер має можливість редагувати замовлення натиснувши на кнопку редагувати і тоді користувача перекине на сторінку для

редагування замовлення. Також на сторінці редагування замовлення можна відміняти або знімати замовлення з водія. Якщо замовлення відмінити то воно буде повністю видалено для водіїв і буде позначено як не активне. А зняття замовлення означає, що воно буде знято з водія і повернено назад в ефір щоб це замовлення міг взяти на себе інший водій.



4.3 – Сторінка редагування замовлення

В інструкції було зображено створення та редагування замовлення для водія від ролі диспетчера. Було показано сторінки та їхня функціональна складова.

3.4 Вимоги до апаратно-програмних засобів

Для стабільної роботи вебсистеми для керування службою таксі, вимоги до функціонування відповідають мінімальним системним вимогам будь-якого браузера. Google Chrome працюватиме на комп'ютерах з процесором Pentium 4 або новішої версії, що охоплює більшість комп'ютерів, випущених з 2001 року.

Комп'ютер повинен мати приблизно 100 МБ вільного місця на жорсткому диску та 128 МБ оперативної пам'яті. Найстарішою версією Windows, яку підтримує Chrome, є Windows XP зі встановленим Service Pack 2. Chrome також працює на комп'ютерах зі встановленою Windows Vista або Windows 7. Для використання браузера Chrome на Android необхідно мати версію операційної системи Android 7.0 Nougat або новішу.

Щоб запустити застосунок з картами, пристрій повинен мати наступні мінімальні характеристики: процесор з тактовою частотою не менше 1.5 ГГц, 2 ГБ оперативної пам'яті, мінімум 300 МБ вільного місця на внутрішньому накопичувачі, а також підтримка GPS та акселерометра.

3.5 Тестування вебсистеми

Тестування програмного забезпечення часто можна визначити як процес перевірки та підтвердження, що програмне забезпечення або додаток функціонує без помилок і відповідає встановленим технічним вимогам.

Тестування програмного забезпечення часто визначають як процес перевірки та підтвердження його коректності, відповідності технічним вимогам, визначеним під час проектування та розробки, а також здатності ефективно задовольняти потреби користувачів, обробляючи всі виняткові та граничні випадки. Тестування програмного забезпечення є методом оцінки

функціональності програмного забезпечення. Цей процес полягає в перевірці, чи відповідає програмне забезпечення очікуваним вимогам і чи не містить воно помилок.

Основною метою тестування програмного забезпечення є виявлення помилок, дефектів або відсутніх функціональних можливостей порівняно з початковими вимогами. Тестування спрямоване на оцінку специфікацій, функціональності та продуктивності програмного забезпечення або додатка. Тестування програмного забезпечення зазвичай поділяють на чотири рівні:

Модульне тестування - на цьому рівні тестуються окремі модулі або компоненти програмного забезпечення. Метою є підтвердження, що кожен модуль працює так, як було задумано;

Інтеграційне тестування - на цьому рівні окремі модулі об'єднуються і тестуються як група. Мета цього рівня тестування - виявити несправності у взаємодії між інтегрованими модулями;

Системне тестування - на цьому рівні тестується повна, інтегрована система або програмне забезпечення. Мета цього тесту - оцінити відповідність системи заданим вимогам;

Приймальне тестування - на цьому рівні система тестується на прийнятність. Мета цього тесту - оцінити відповідність системи бізнес-вимогам і визначити, чи є вона прийнятною для поставки.

Оскільки розроблюваний програмний продукт складається з модулів, основним видом тестування в рамках кваліфікаційної роботи буде модульне тестування.

Переваги модульного тестування:

Раннє виявлення проблем дозволяє розробникам знаходити та усувати неполадки на початкових етапах розробки, перш ніж вони перетворяться на більш серйозні та складні для виправлення

Покращена якість коду - допомагає гарантувати, що кожна одиниця коду працює за призначенням і відповідає вимогам, покращуючи загальну якість програмного забезпечення.

Підвищення впевненості - дає розробникам впевненість у своєму коді, оскільки вони можуть підтвердити, що кожна частина програмного забезпечення функціонує належним чином.

Швидша розробка - дозволяє розробникам працювати швидше та ефективніше, оскільки вони можуть перевіряти зміни в коді без необхідності чекати, поки буде протестована вся система.

Полегшення рефакторингу - дозволяє розробникам безпечно вносити зміни до коду, оскільки вони можуть переконатися, що їхні зміни не порушують існуючу функціональність.

Скорочення часу та витрат - може скоротити час і витрати, необхідні для подальшого тестування, оскільки воно допомагає виявити і виправити проблеми на ранніх стадіях процесу розробки.

Для тестування серверної частини, а саме перевірки запитів було використано Postman

Postman — це популярний інструмент для тестування API, який дозволяє розробникам надсилати HTTP-запити та отримувати відповіді від веб-серверів. Ось як можна використовувати Postman для тестування API:

На рисунку 4.4 зображено тестування PUT запита для оновлення даних про замовлення. Тестування через Postman дає можливість на ранньому етапі розробки API знайти та виправити помилки.

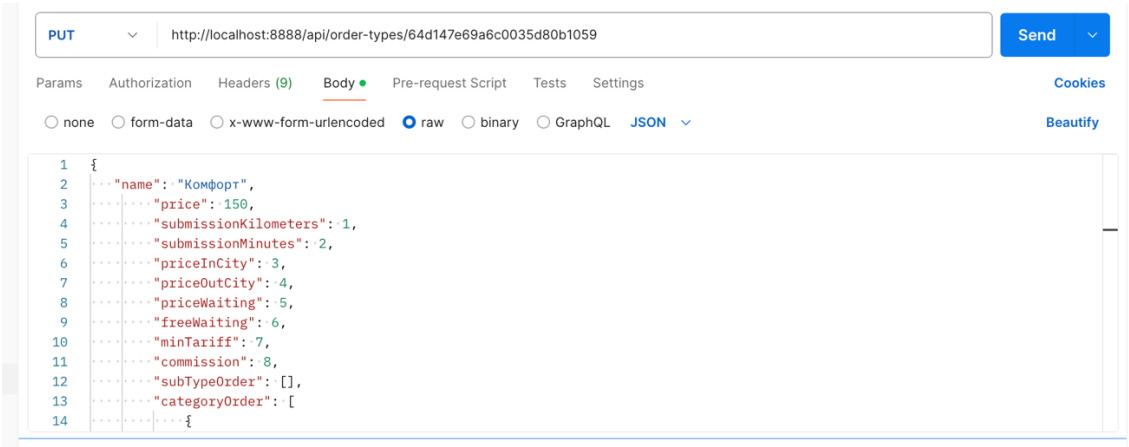


Рисунок 4.4 – Тестування запитів за допомогою Postman

ВИСНОВКИ

Ця робота мала на меті дослідження, розробку та аналіз архітектурного рішення для вебсистеми управління службою таксі. Застосунок, який буде використовуватися водіями, створений з використанням технологій React Native, JavaScript, React Hooks, Redux Toolkit, Google Maps API. Крім того, було розроблено адміністративну панель, що містить два типи користувачів: «Адмін» та «Диспетчер».

Адмін панель реалізована на основі React JS, JavaScript, React Hooks, Google Maps API, Redux Toolkit, react-router-dom, MaterialUI. Серверна частина побудована з використанням NodeJS, ExpressJS, MongoDB. Застосунок для водіїв використовує React Native Navigation, React Native Maps, React Native Floating Bubble, React Native Sound, функціональні компоненти, хуки, Redux Toolkit, WebSocket, React Native Gesture Handler, React Native Geolocation Service, React Native Maps Directions, React Native Tab View та React i18next. У реальному часі водіям надходять замовлення, які вони можуть приймати. Також водії можуть входити в застосунок, змінювати свій пароль та переглядати персональні дані.

У першому розділі проведено аналіз та вивчення об'єкта дослідження, що стосується розробки застосунку для таксі на платформі React Native. Виявлено достатню кількість інформації та прикладів, які були використані для підтримки розробки проекту.

Другий розділ присвячено вибору архітектурного рішення та зразків проектування для застосунку. Описано обрані архітектурні рішення та шаблони проектування, що використовуються у розробці та проектування бази даних. Детально розповідається про кожне рішення та переваги.

Третій розділ присвячено детальному проектуванню модулів та структурних елементів застосунку. Описано функціональні та структурні вимоги до модулів, а також здійснено декомпозицію проекту. Вказано

залежності між модулями та створено опис інтерфейсів об’єктної моделі. Висновки з цього дослідження та розробки вебсистеми управління службою таксі дозволять створити надійну та ефективну платформу для користувачів, зручну для водіїв та адміністраторів.

Використання сучасних технологій, таких як ReactJS, NodeJS, MongoDB та Google Maps API, забезпечує високу продуктивність, масштабованість та безпеку системи. Це рішення є конкурентоспроможним аналогом таких сервісів, як Uber, Lyft, Uklon, Bolt, і надає можливість ефективного управління службою таксі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. A Complete Beginner's Guide to React Native. URL: https://reactnative.dev/docs/getting-started (дата звернення: 17.01.2024)

2. Understanding the Basics of React. URL: https://react.dev/learn (дата звернення: 20.01.2024)

3. Node.js Introduction. URL: https://nodejs.org/docs/latest/api/ (дата звернення: 20.01.2024)

4. How To Get Started with MongoDB. URL: https://learn.mongodb.com/pages/learners-journey (дата звернення: 20.01.2024)

5. Integration of Google Maps with React. URL: https://medium.com/scalereal/integration-of-google-maps-with-react-part-1-86c075ab452a (дата звернення: 20.01.2024)

6. How to implement google map in react. URL: https://medium.com/@deepbag/how-to-implement-google-map-in-react299fa5a8e51a (дата звернення: 21.01.2024)

7. Virtual machines anyone can set up in seconds. URL: https://www.digitalocean.com/products/droplets (дата звернення: 21.01.2024)

8. Exploring the Power of Material-UI (MUI) for React Applications. URL: https://pratha001.medium.com/exploring-the-power-of-material-ui-mui-for-react-applications-c4a233c490b6 (дата звернення: 22.01.2024)

9. Building Real-Time Applications with Socket.io, React.js, Node.js, and Express.js: A Comprehensive Guide. URL: https://towardsdev.com/building-real-time-applications-with-socket-io-adc86da2f9f1 (дата звернення: 23.01.2024)

10. Socket get started. URL: https://socket.io/get-started/chat (дата звернення: 17.01.2024)

11. All you need to know about React Hooks. URL: https://codechronicle.medium.com/all-you-need-to-know-about-react-hooks-d2e7bfa3bc67 (дата звернення: 23.01.2024)

12. The complete guide to WebSockets with React. URL: https://ably.com/blog/websockets-react-tutorial (дата звернення: 24.01.2024)

13. How to Draw Directions Route on a Map in React Native. URL: https://medium.com/@burcuozdmr/how-to-draw-directions-route-on-a-map-in-react-native-a52f406cf5 (дата звернення: 23.01.2024)

14. How to implement WebSockets in React Native. URL: https://blog.logrocket.com/how-to-implement-websockets-in-react-native/ (дата звернення: 23.01.2024)

15. Getting started React Native Navigation URL: https://reactnavigation.org/docs/getting-started (дата звернення: 23.01.2024)

16. Getting started React Router Dom URL: https://reactrouter.com/en/main/start/overview (дата звернення: 23.01.2024)

17. Writing middleware for use in Express apps. URL: https://expressjs.com/en/guide/writing-middleware.html (дата звернення: 24.01.2024)

18. MongoDB get started. URL: https://www.w3schools.com/nodejs/nodejs_mongodb.asp (дата звернення: 24.01.2024)

19. Bolt. URL: https://bolt.eu/uk-ua/food/ (дата звернення: 13.01.2024)

20. WebStorm features. URL: https://www.jetbrains.com/phpstorm/features/ (дата звернення: 15.01.2024)

21. VS Code features. URL: https://blog.cloudanalogy.com/10-features-of-vs-code-every-developer-should-know/ (дата звернення: 15.01.2024)

22. Benefits of VS Code. URL: https://code.visualstudio.com/docs/editor/whyvscode (дата звернення: 17.01.2024)

23. Advantages of JavaScript. URL: https://codeinstitute.net/global/blog/advantages-of-javascript/ (дата звернення: 18.01.2024)

24. Front-end розробка. URL: https://brainlab.com.ua/uk/blog-uk/front-end-rozrobka (дата звернення: 24.01.2024)

25. Overview of React.js. URL: https://www.patterns.dev/react/ (дата звернення: 24.01.2024)

26. Exploring React Native Sound: A Comprehensive Guide. URL: https://www.linkedin.com/pulse/exploring-react-native-sound-comprehensive-guide-pawan-kumar-mggdf (дата звернення: 24.01.2024)

27. Route-Controller-Service structure for ExpressJS. URL: https://devtut.github.io/nodejs/route-controller-service-structure-for-expressjs.html (дата звернення: 24.01.2024)

28. How to Connect MongoDB with Node.js: A Comprehensive Guide. URL: https://medium.com/@zadafiya/how-to-connect-mongodb-with-node-js-a-comprehensive-guide-cdf4d099ae9b (дата звернення: 24.01.2024)

29. How to get started with the MERN stack. URL: https://www.mongodb.com/resources/languages/mern-stack-tutorial (дата звернення: 17.01.2024)

30. DigitalOcean Droplet Quick Start Guide. URL: https://openvpn.net/as-docs/digitalocean.html (дата звернення: 24.01.2024)

31. Foundations of Docker. URL: https://docs.docker.com/guides/ (дата звернення: 24.01.2024

Додаток А

(обов'язковий)

ПРЕЗЕНТАЦІЯ ДЛЯ ДОПОВІДІ

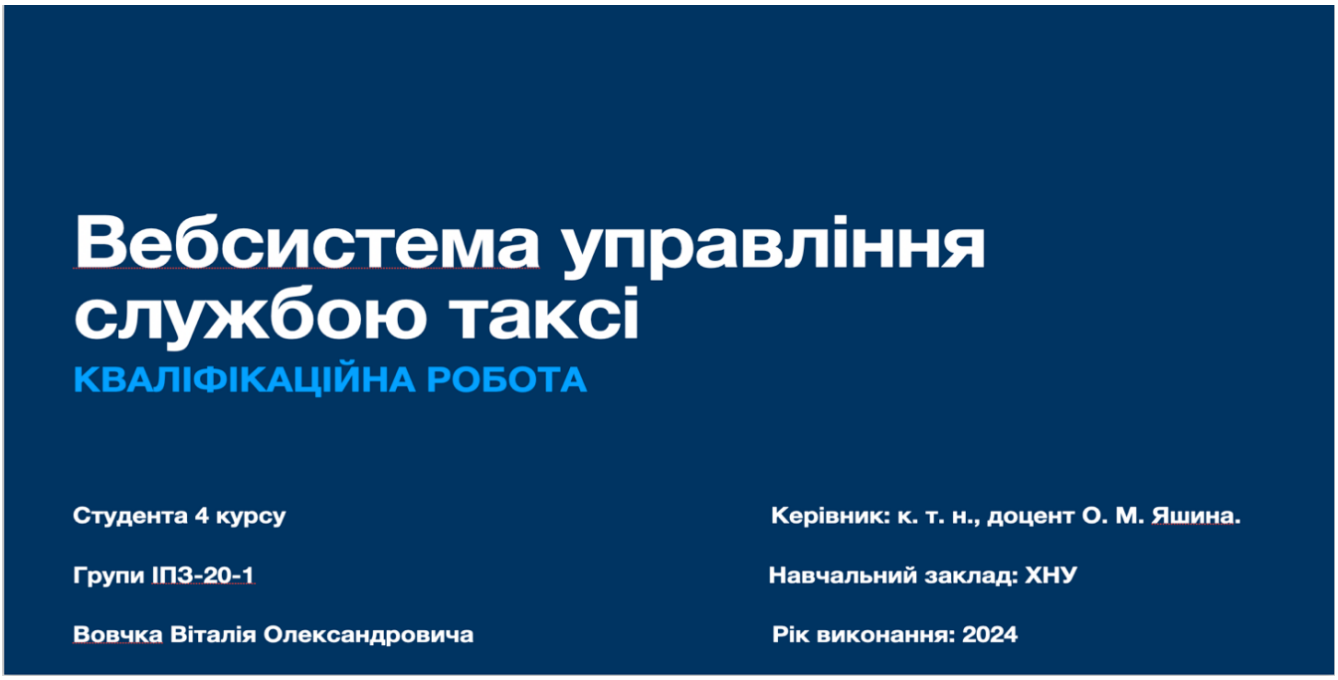


Рисунок А.1 – Заголовний слайд

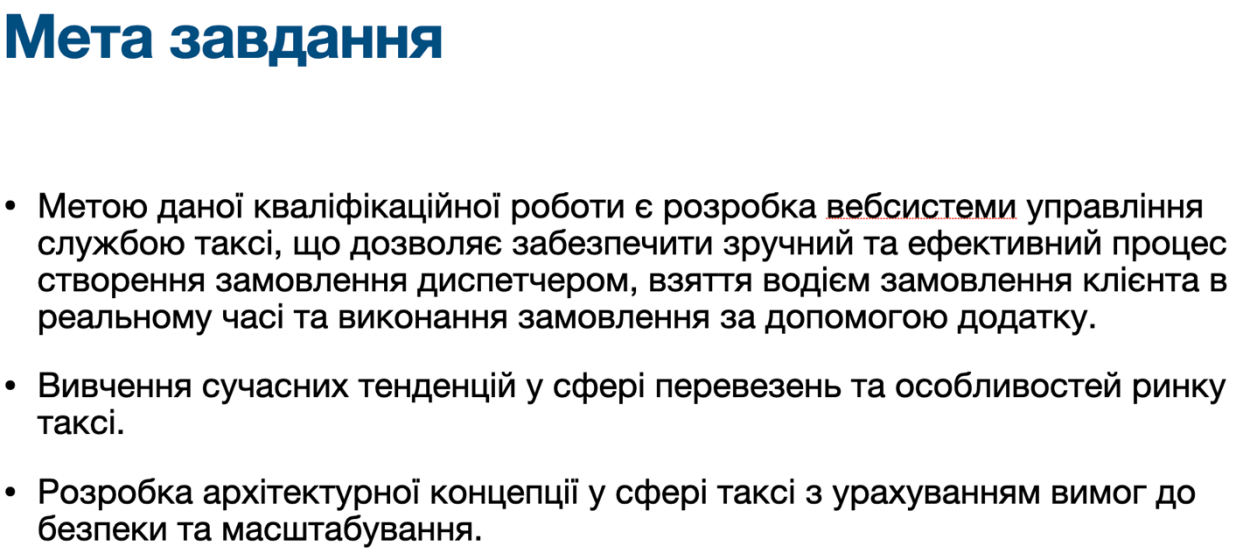
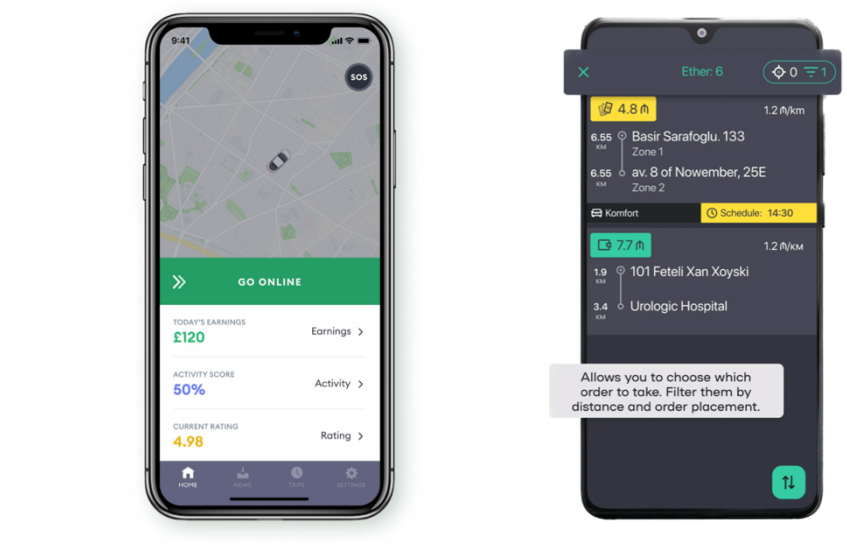


Рисунок А.2 – Мета завдання

Аналіз існуючих рішень



Bolt Driver App Uklon Driver App

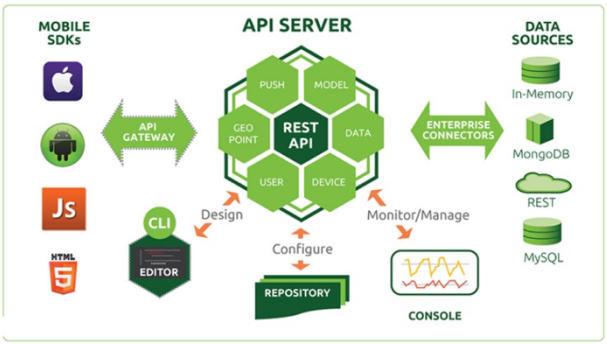
Рисунок А.3 – Аналіз існуючих рішень

Вибір засобів розробки

- JavaScript – це мова програмування, яка використовується у веб-розробці для створення інтерактивних веб-сайтів та додатків.
- ReactJS – це JavaScript-бібліотека для створення інтерфейсів користувача.
- React Native – це фреймворк, розроблений Facebook, для розробки кросплатформних мобільних додатків
- NodeJS – це платформа з відкритим кодом для виконання JavaScript коду на стороні сервера.
- ExpressJS – фреймворк веб-додатків для NodeJS.

Рисунок А.4 – Вибір засобів розробки

Вибір архітектури вебсистеми



- Під час розробки веб системи був використаний стек технологій MERN (MongoDB, ExpressJS, ReactJS, NodeJS), даний підхід передбачає використання монолітної архітектури де компоненти додатку (логіка обробки запитів, доступ до баз даних, клієнтський інтерфейс тощо) зібрані в єдиний програмний пакет. Цей підхід спрощує розгортання і запуск додатку.

Рисунок А.5 – Вибір архітектури вебсистеми

Розробка серверної частини

- Для розробки серверної частини було застосовано такі технології, як NodeJS та ExpressJS, ці технології є дуже популярними і завдяки цьому спрощується розробка. Метою кваліфікаційної роботи було забезпечення безперебійного з’єднання сервера з додатком і для цього було використано технологію socket.io-server, яка дозволяє в реальному часі здійснювати обмін даними.
- Базу даних було обрано MongoDB, вона легко інтегрується з NodeJS та ExpressJS, що забезпечує простоту та швидкість розробки.

Рисунок А.6 – Розробки серверної частини

Розробка клієнтської частини

- Клієнтську частину було написано на ReactJS.
- Для створення маршрутів було використано Google Maps API, це дозволило точно вираховувати та будувати маршрути отримувати необхідні дані і багато корисної інформації про маршрут, яку можна використати при формуванні ціни для замовлення.
- Для швидкого створення інтерфейсу було використано UI бібліотеку MaterialUI, це дозволило будувати швидко інтерфейс користувача.
- Для безперебійного з'єднання було використано socket.io-client
- Для керування запитами до API та керування станом було використано бібліотеку Redux Toolkit

Рисунок А.7 – Розробка клієнтської частини

Розробка додатку для водія

- Для розробки додатку для водія було використано React Native.
- У дадтку було додано платіжну систему Fondy, для поповнення балансу водія.
- Для зменшення ціни за використання карт було інтегровано openrouteservice
- Для безперебійного з'єднання було використано socket.io-client
- Для керування запитами до API та керування станом було використано бібліотеку Redux Toolkit
- Для зручності водія була додана статистика, історія замовлень, вибір навігатора, та вибір замовлень, які можуть йому показуватися в загальному списку доступних замовлень.

Рисунок А.8 – Розробна додатку для водія

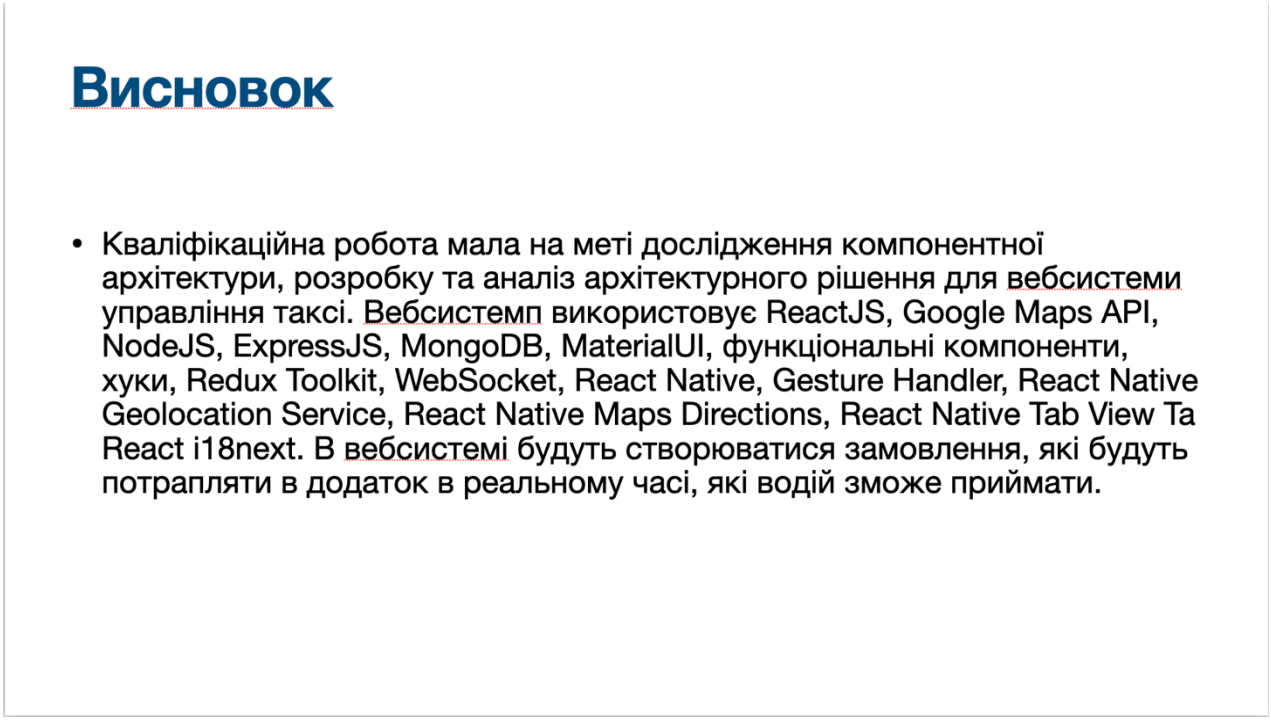


Рисунок А.9 - Висновок

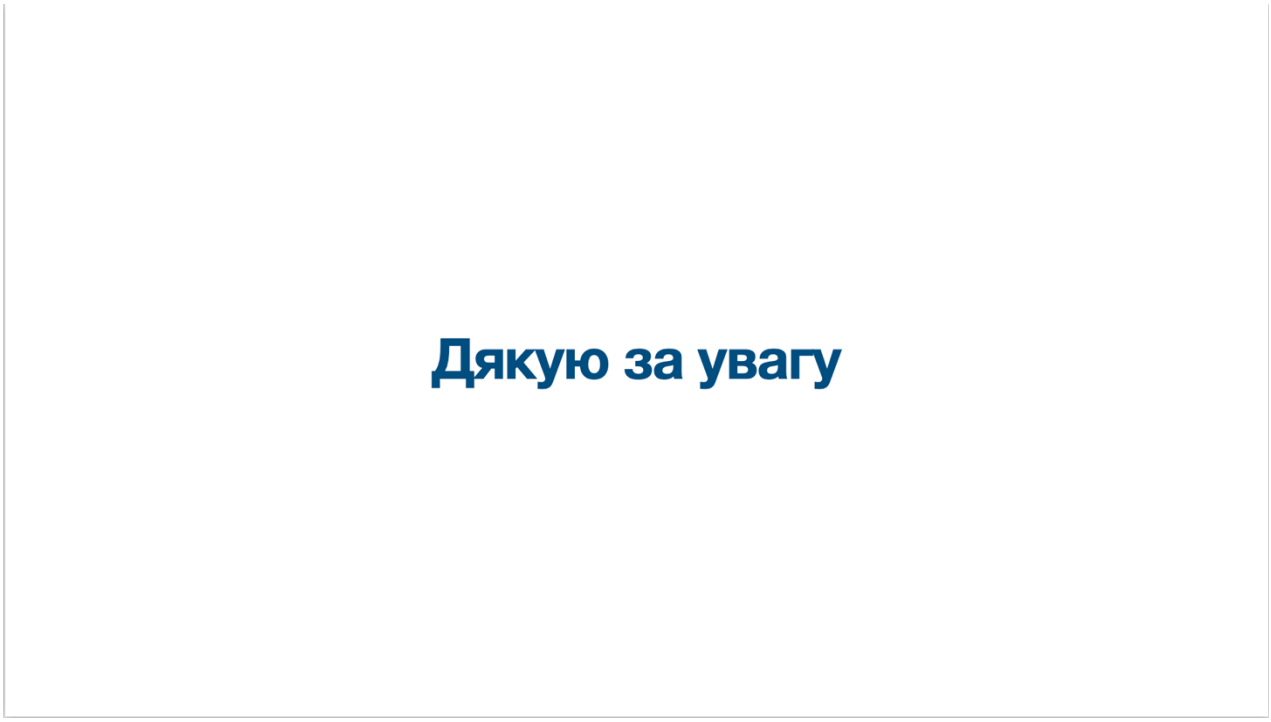


Рисунок А.10 – Дякую за увагу!

Додаток Б

(обов'язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту розробки вебсистеми керування службою таксі.

1 Підстава для розробки

Підставою для розробки є «Завдання на кваліфікаційну роботу», затверджене завідувачем кафедри інженерії програмного забезпечення.

Найменування розробки: Вебсистема керування службою таксі.

2 Призначення розробки

Призначення цієї розробки полягає в створенні вебсистеми управління службою таксі, що буде аналогом відомих сервісів таких як Uber, Lyft, Uklon та Bolt. Основна мета – забезпечити ефективне та зручне рішення для керування службою таксі, яке включатиме мобільний застосунок для клієнтів та адміністративну панель для управління.

Система використовує передові технології, такі як React Native, JavaScript, React Hooks, Redux Toolkit та Google Maps API, для забезпечення надійної роботи мобільного додатка. Адміністративна панель, розроблена на основі React JS, забезпечує функціональність для двох типів користувачів: адміністраторів та диспетчерів. Вона побудована з використанням таких технологій як JavaScript, React Hooks, Google Maps API, Redux Toolkit, react-router-dom та MaterialUI. Це дозволяє адміністраторам керувати водіями та замовленнями, а диспетчерам – координувати роботу служби в режимі реального часу. Розробка спрямована на підвищення ефективності роботи служби таксі, поліпшення якості обслуговування клієнтів та забезпечення швидкої і зручної взаємодії між усіма учасниками процесу. У вебсистемі буде можливість створювати водіїв, створбвати тарифні плані та додавани міста та зони з тарифами, буде передбачено використання платіжної системи для поповнення балансу для водія.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Вебсистема керування службою таксі повинна забезпечувати можливістю:

– додавання водіїв у систему;

– створення замовлень в реальному часі;

– авторизації та реєстрації користувача;

– налаштування особистого кабінету водія;

– керування системою від імені адміністратора: блокування та видалення аккаунтів водіїв.

Вимоги до інтерфейсу:

– сучасний та інтуїтивно зрозумілий дизайн інтерфейсу;

– якісний контент та відсутність надлишкової інформації.

3.2 Вимоги до надійності

Розроблюване ПЗ повинно мати:

− захист від несанкціонованого доступу до даних;

– надання різних ролей та прав користувачів у вебсистемі;

− контроль інформації, що вводиться, та блокування некоректних дій користувачів під час роботи зі вебистемою.

3.3 Вимоги до складу та параметрів технічних засобів

Мінімальні вимоги для функціонування системи повинні відповідати вимогам будь-якого браузера. Нижче наведено приклад мінімальних вимог для браузера Google Chrome:

– стабільне з’єднання з Інтернетом;

– операційна система: Windows 11, 10, 8, 8.1, 7, MacOS;

– розрядність: x86 (32-bit) або x64 (64-bit);

– процесор: Pentium 4 з SSE2;

– вінчестер: 350 Mb;

– відеоадаптер: 3D адаптер nVidia, Intel, AMD / ATI;

– оперативна пам’ять: 512 Mb;

– будь-яка аудіокарта;

– роздільна здатність екрану: SVGA 800x600;

– контролери: клавіатура, миша;

– телефон із мінімальною пам’яттю 2ГБ;

– телефон тактовою частотою не менше 1.5 ГГц;

– телефон з вільною пам’яттю не менше ніж 300 МБ;

– наявність в телефоні працюючого GPS.

3.4 Вимоги до інформаційної та програмної сумісності

Для створення серверної частини будуть використовуватися такі технології:

– мова програмування JavaScript;

– платформа NodeJS;

– фреймворк ExpressJS;

– монолітна архітектура;

Для створення клієнтської частини використовуватимуться такі технології:

– бібліотека ReactJS;

– кросплатформенний фреймворе React Native;

– мова розмітки HTML;

– таблиці стилів CSS;

Для роботи з базою даних буде MongoDB і середовище розробки MongoDB Compass.

3.5 Вимоги до транспортування та зберігання

Програма та її документація надаються у цифровому форматі. Умови використання програмного забезпечення співпадають з умовами експлуатації сервера, на якому воно буде розміщене.

3.6 Спеціальні вимоги

Інтерфейс програми повинен бути дружнім до користувача середньої кваліфікації (з точки зору комп'ютерної грамотності). Мова програмування вибирається виконавцем.

4 Вимоги до програмної документації

В ході розробки програми повинні бути підготовлені:

– технічне завдання;

– опис програми;

– текст програми з коментарями та поясненнями;

– методика випробувань;

– відомості про функціонування програми;

– керівництво користувача;

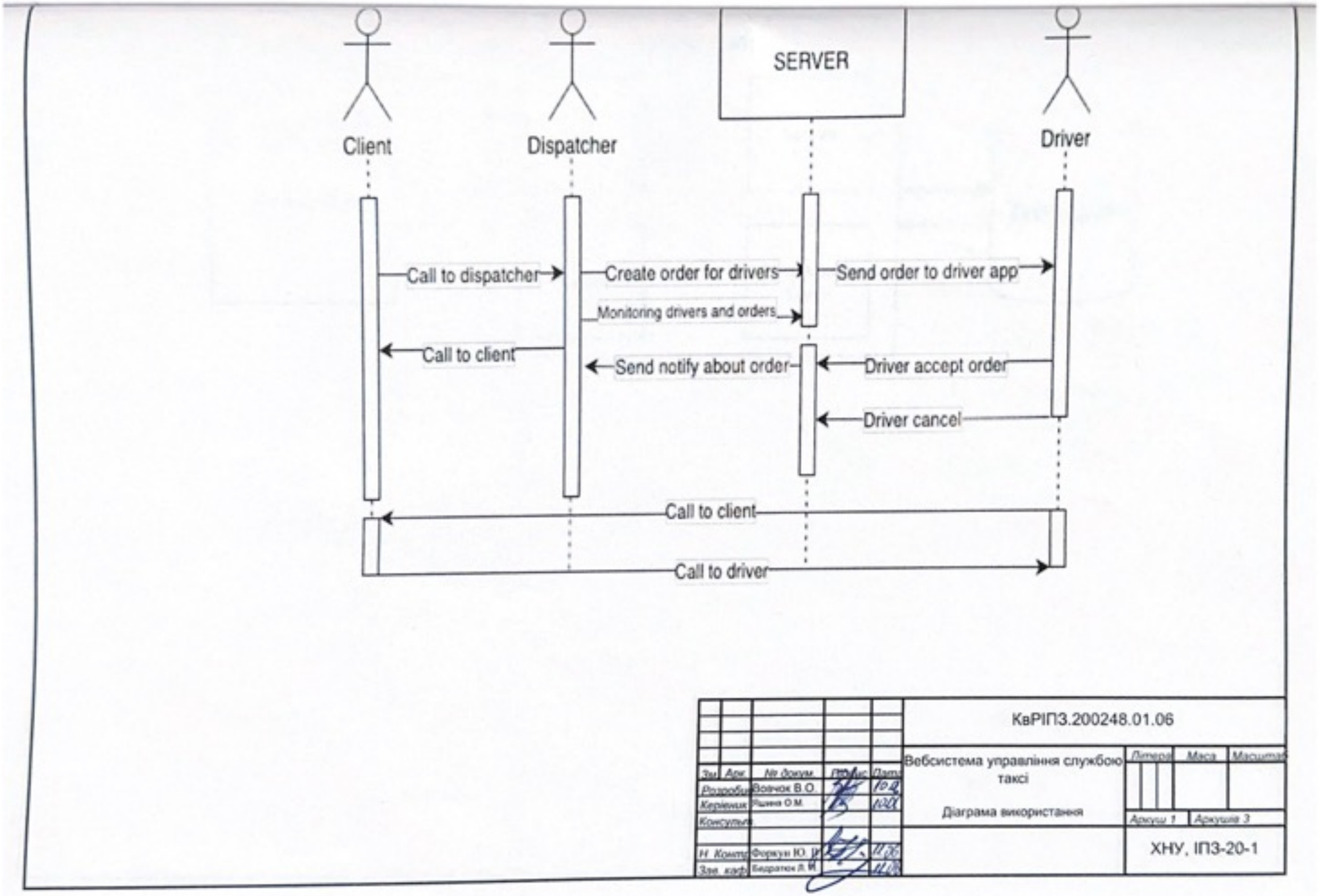
– керівництво програміста.

5 Порядок контролю та приймання

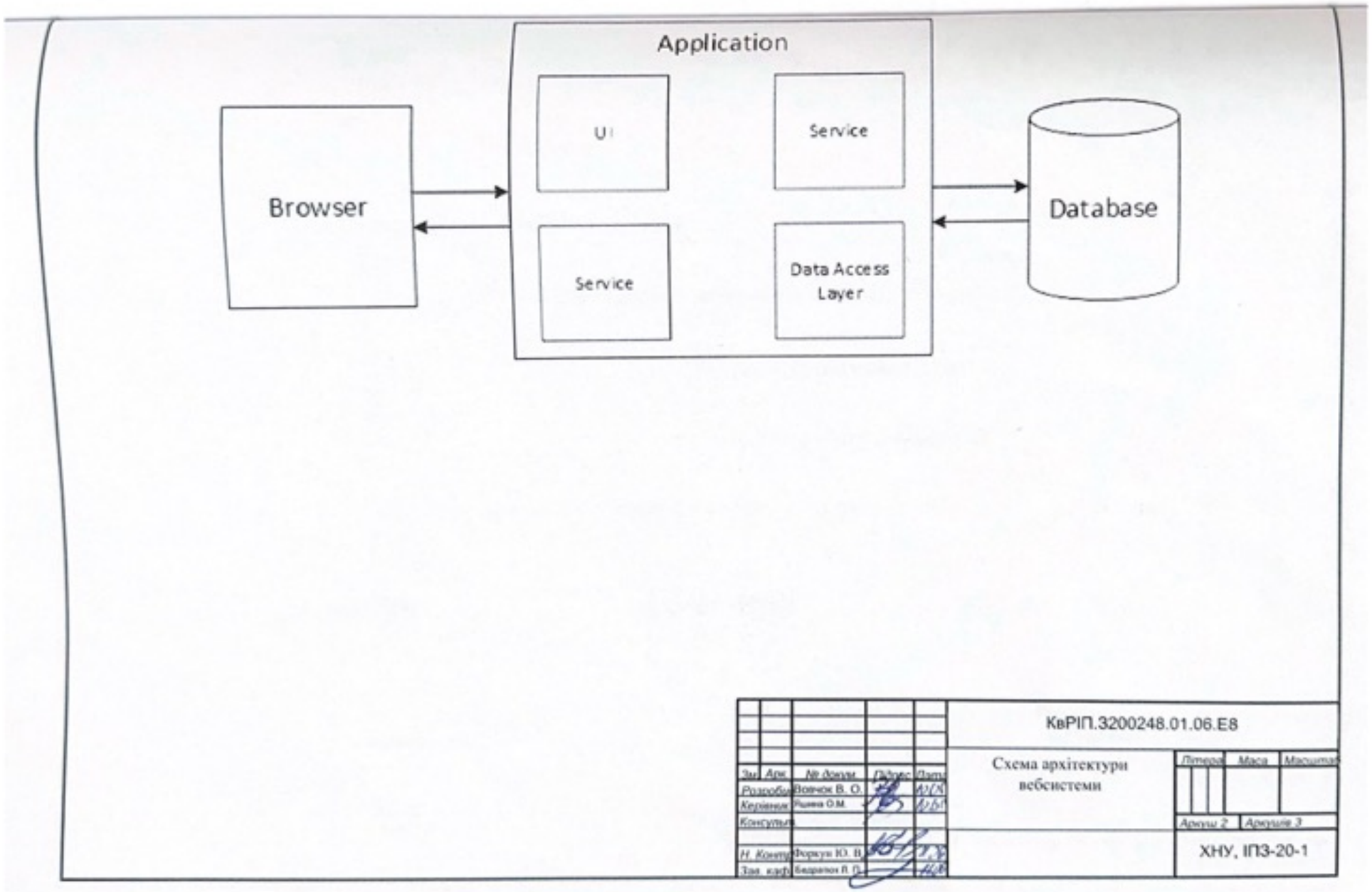
Контроль здійснюється кінцевими користувачами системи, які підключаються на етапі тестування.

Прийом системи відбувається після її повного розгортання, встановлення на хостинг та налаштування для нормального функціонування. Після завершення розробки системи необхідно провести тестування на захист від некоректного введення даних.

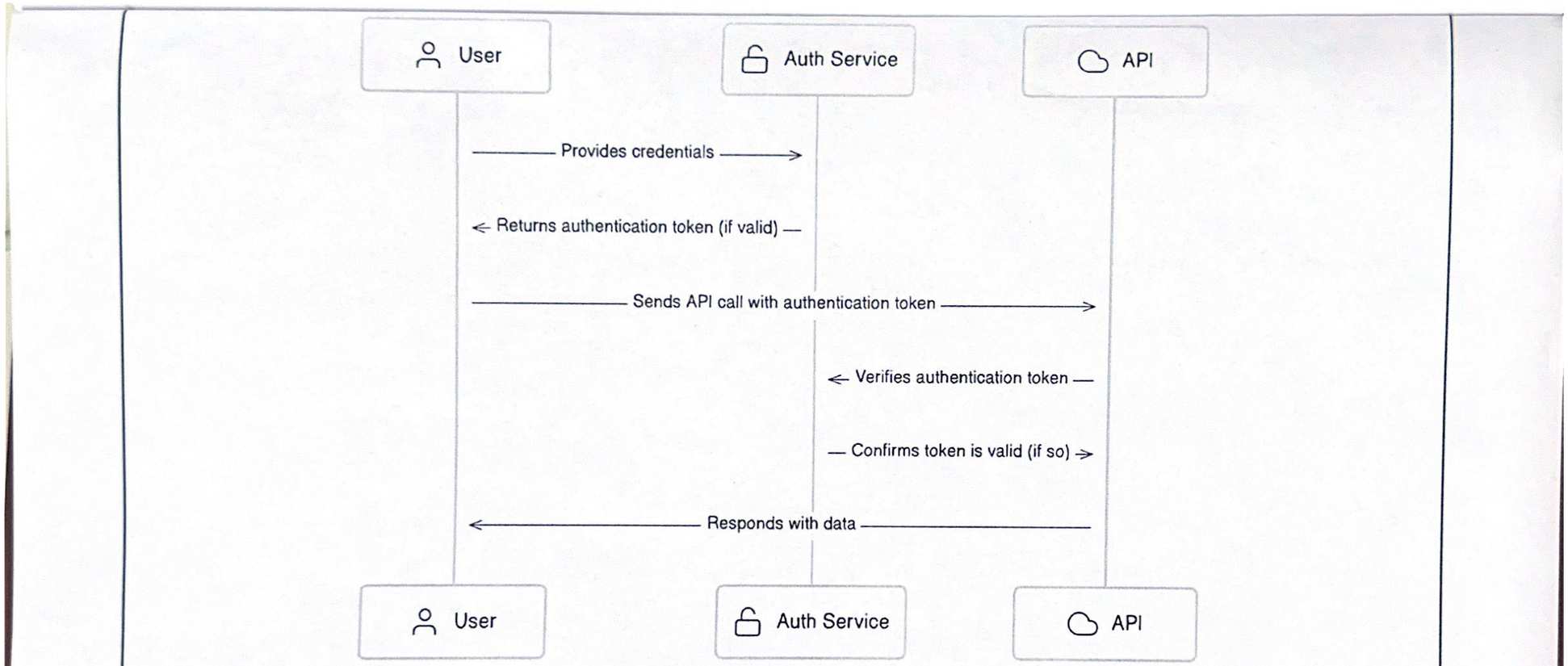
ГРАФІЧНА ЧАСТИНА



						КвРІПЗ.200248.01.06	
Зм.	Арк.	№ докум.	Підпис	Дата	Вебсистема управління службою таксі	Літера	Маса	Масштаб
Розробив		Вовчок В.О.		10.12				
Керівник		Рьяна О.М.		10.12	Діаграма використання	Аркуш 1	Аркушів 3	
Консульт.								
Н. Контр.		Форкун Ю.І.		11.12		ХНУ, ІПЗ-20-1		
Зав. каф.		Бедратюк Л.М.		11.12				



					КвРІП.3200248.01.06.Е8			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розробив		Вовчок В. О.			Схема архітектури вебсистеми	Літера	Маса	Масштаб
Керівник		Рашев О.М.						
Консульт.						Аркуш 2	Аркушів 3	
Н. Контр.		Форкун Ю. В.				ХНУ, ІПЗ-20-1		
Зав. каф.		Бедратюк Л. Д.						



					КвРІПЗ.200251.01.06.Е8		
Зм.	Арк.	№ докум.	Підпис	Дата	Діаграма варіантів використання	Літера	Маса	Масштаб
Розробив		Вовчок В.О.						
Керівник		Яшина О. М.				Аркуш 3	Аркушів 3	
Консульт.								
Н. Контр.		Форкун Ю.В.				ХНУ, ІПЗ-20-1		
Зав. каф.		Бедратюк Л. П.						

СУПРОВІДНІ ДОКУМЕНТИ

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти

Вовчок В. О.

Прізвище, ініціали

факультет ІТ, 4 курс, група ІПЗ-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності в Хмельницькому національному університеті», згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та/або Anti-Plagiarism) і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

06.01.2024

дата

підпис

Mon Jun 10 07:14:36 EEST 2024, Форкун Юрій Вікторович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

	Документ		Сумарний збіг по Базі Даних	
ID: 129202 Назва: БКР_Вебсистема управління службою таксі_Вовчок_Яшина Додано в БД: 2024-06-10 Автора: Вовчок В. Керівники: Яшина О.М., канд. техн. наук, доцент Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	68759	1028	1557 (2%)	23 (2%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK
by Turnitin

Ім'я користувача:
ІПЗ

ID перевірки:
1016333773

Дата перевірки:
07.06.2024 22:54:34 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
08.06.2024 05:43:25 EEST

ID користувача:
100012953

Назва документа: БКР_ВЕБСИСТЕМА_УПРАВЛІННЯ_СЛУЖБОЮ_ТАКСІ_Вовчок_Яшина

Кількість сторінок: 66 Кількість слів: 11871 Кількість символів: 94170 Розмір файлу: 2.78 MB ID файлу: 1016133979

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.1%
Схожість

Найбільша схожість: 1.1% з джерелом з Бібліотеки (ID файлу: 1016121455)

4.36% Джерела з Інтернету	576	Сторінка 68
2.09% Джерела з Бібліотеки	144	Сторінка 71

0.03% Цитат

Цитати	1	Сторінка 72

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	15
Підозріле форматування	23 сторінки

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Вовчок Віталій Олександрович

Тема Вебсисетема управління службою таксі

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 83

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі досліджено і проаналізовано предметну область транспортних послуг , визначено усі вимоги яким повинна відповідати вебсистема керування службою таксі Був проведений аналіз існуючих рішень на ринку, розглянуто їх переваги та недоліки, та доведено актуальність розробки нового програмного забезпечення .

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота повністю відповідає поставленому завданню. Було виконано усі етапи розробки.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи .У першому розділі розглянуто існуючі рішення які використовують служби таксі. У другому розділі розглянуто проектування вебсистеми керування службою таксі. У третьому розділі розроблено програмну частину вебсистеми та проведено її тестування.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною, оскільки на сьогодні в Україні служби таксі користуються популярністю, але не надають достатньої кількості функціональних можливостей. Також було застосовано новітні технології для побудови програмного продукту та актуальні архітектурні рішення.

5. Негативні сторони роботи У роботі не реалізована кросплатормність, не реалізована адаптація стилів диспетчерської під мобільні розширення. Немає доданої діючої платіжної системи.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Пояснювальна записка структурована, послідовна, чітка, що дозволяє чітко зрозуміти викладений матеріал у рамках теми кваліфікаційної роботи. Графічний матеріал дає можливість наочно побачити деталі проектування системи.

8. Інші зауваження У вебсистемі недостатня кількість інформативності для користувача, який тільки починає працювати з готовим програмним продуктом.

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ Лисенко Сергій Миколайович, доктор технічних наук, професор кафедри комп'ютерної інженерії та інформаційних систем ХНЕУ ім. С. Кузнеця.

"10" червня 2024 р. _____
(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів:

Назва кваліфікаційної роботи: «Вебсистема управління службою таксі»
Автор: Вовчок Віталій Олександрович
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Яшина Оксана Миколаївна, кандидат технічних наук, доцент

Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	
3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені у роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системою перевірки на плагіат Unicheck виявлено схожість з деякими документами у частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів, у рамках основних написів, у назвах публікацій переліку джерел посилання;

2) в якості запозичень системою Unicheck було зафіксовано деякі послідовності вихідного коду і посилання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) запозичення, виявлені в тексті роботи, є фрагментарними.

Максимальний обсяг запозичень, визначений системою Anti-Plagiarism, складає 1.0%. Обсяг запозичень, визначений системою Unicheck виявлення збігів ідентичності/схожості, складає 5.1% і адресується до 31 джерел з Інтернету, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Оксана ЯШИНА