ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Веб-застосунок для управління послугами автосервісу на основі JavaScript»

на здобуття освітнього ступеня магістра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ярослав Галата
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти
група КНДМ-61
Ярослав ГАЛАТА

Керівник:
науковий ступінь, вчене звання
Олена ШИКУЛА
д.ф.-м.н., професор

Рецензент:
науковий ступінь, вчене звання
_____
(Ім'я, ПРІЗВИЩЕ)

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Магістр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Галаті Ярославу Олександровичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Веб-застосунок для управління послугами автосервісу на основі JavaScript.

керівник кваліфікаційної роботи Олена Шикула д.ф.-м.н., професор,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від 15.10.2024р. №320

2. Строк подання кваліфікаційної роботи 15.12.2024р.

3. Вихідні дані до кваліфікаційної роботи: інформація про клієнтів, персонал та їхні дані, перелік послуг, замовлення та записи, запчастини та матеріали, рахунки та платежі.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження та аналіз технологій для веб-застосунку для управління послугами автосервісу.

Дослідження та порівняння рішень створення веб-застосунку для управління послугами автосервісу на основі JavaScript.

5. Перелік графічного матеріалу: *презентація*

1. Огляд технологій JavaScript: архітектура, порівняння, приклади використання.
2. Діаграми аналітики.
3. Діаграми взаємодії користувача з системою.
4. Фрагменти програмного коду.

6. Дата видачі завдання 05.09.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	19.10-05.11.24	Виканано
2	Огляд технологій JavaScript	05.11-13.11.24	Виканано
3	Дослідження та порівняння рішень для веб-застосунку	14.11-19.11.24	Виканано
4	Вивчення рішень для забезпечення безпеки веб-застосунку	20.11-25.11.24	Виканано
5	Оптимізація та налаштування	27.11-03.12.24	Виканано
6	Документування	04.12-10.12.24	Виканано
7	Впровадження та запуск	11.12-20.12.24	Виканано
8	Розробка демонстраційних матеріалів	21.12-29.12.24	Виканано

Здобувач вищої освіти _____ Ярослав ГАЛАТА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Олена ШИКУЛА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 67 стор., 33 рис., 20 джерел.

Наукове завдання – дослідження веб-застосунку для управління послугами автосервісу з використанням сучасних технологій для автоматизації обліку, обробки замовлень та підвищення ефективності бізнес-процесів.

Мета роботи – дослідження веб-застосунку для автоматизації управління послугами автосервісу, який включає облік замовлень, керування запасами, взаємодію з клієнтами та фінансові операції, з використанням сучасних веб-технологій для підвищення ефективності та зручності роботи автосервісу.

Об'єкт дослідження – веб-застосунок для автоматизації управління послугами автосервісу, який охоплює управління записами клієнтів, облік замовлень, управління запасами, фінансові операції та взаємодію з користувачами через інтерфейс системи.

Предмет дослідження – процеси автоматизації управління послугами автосервісу за допомогою веб-застосунку, зокрема методи обробки замовлень, управління обліком запасів, клієнтських записів, фінансових операцій та взаємодії з користувачами через інтерфейс системи.

Короткий зміст роботи:

У даній роботі розглядається процес розробки веб-застосунку для управління послугами автосервісу, метою якого є автоматизація основних бізнес-процесів, таких як обробка замовлень, облік запасів, фінансові операції та взаємодія з клієнтами. В роботі аналізуються вимоги до системи, вибір відповідних технологій для фронтенду та бекенду, а також забезпечення безпеки даних.

КЛЮЧОВІ СЛОВА: ВЕБ-ЗАСТОСУНОК, АВТОСЕРВІС, УПРАВЛІННЯ ПОСЛУГАМИ, АВТОМАТИЗАЦІЯ ДІЯЛЬНОСТІ, JAVASCRIPT.

ABSTRACT

The text part of the qualification work for the master's degree: 67 pages, 33 pictures, 20 sources.

The scientific task is the study of a web application for managing car service services using modern technologies to automate accounting, order processing and increase the efficiency of business processes.

The purpose of the work is to research a web application for automating the management of car service services, which includes order accounting, inventory management, interaction with customers and financial operations, using modern web technologies to improve the efficiency and convenience of car service work.

The object of the research is a web application for automating car service management, which covers customer record management, order accounting, inventory management, financial transactions and interaction with users through the system interface.

The subject of the study is the processes of automating the management of car service services using a web application, in particular, the methods of processing orders, managing inventory, customer records, financial transactions, and interacting with users through the system interface.

Summary of the work:

This paper examines the process of developing a web application for the management of car service services, the purpose of which is to automate the main business processes, such as order processing, inventory accounting, financial transactions and interaction with customers.

The work analyzes the requirements for the system, the selection of appropriate technologies for the frontend and backend, as well as ensuring data security. The methods of testing and optimizing the system to increase its efficiency and reliability are considered separately.

KEYWORDS: WEB APPLICATION, AUTO SERVICE, SERVICE MANAGEMENT, AUTOMATION OF ACTIVITIES, JAVASCRIPT.

ЗМІСТ

ВСТУП

Актуальність теми: Автосервісна галузь є однією з ключових сфер, що забезпечує підтримку та обслуговування автомобілів, важливих для повсякденного життя та комерційної діяльності. Зі зростанням кількості автомобілів і підвищенням вимог до якості обслуговування стає очевидною потреба в ефективних рішеннях для управління процесами автосервісу. Традиційні методи ведення записів, планування та управління запасами вимагають значних часових витрат і є схильними до помилок. Це може призводити до втрати клієнтів, зниження продуктивності роботи та додаткових витрат на адміністрування. Автоматизація цих процесів за допомогою веб-застосунку дозволяє значно підвищити ефективність роботи автосервісів, покращити точність обліку, забезпечити швидкий доступ до інформації та оптимізувати використання ресурсів. Застосування сучасних веб-технологій робить можливим створення зручних, інтуїтивних інтерфейсів, які сприяють покращенню обслуговування клієнтів та задоволенню їхніх потреб. Отже, розробка веб-застосунку для управління автосервісом є актуальним завданням, яке сприяє підвищенню продуктивності, економії ресурсів і поліпшенню якості послуг, що надаються клієнтам, а також дає можливість адаптуватися до вимог сучасного ринку.

Мета роботи: Розробка веб-застосунку для автоматизації управління послугами автосервісу, який забезпечить ефективний облік замовлень, керування запасами, обробку фінансових операцій та взаємодію з клієнтами, а також сприятиме підвищенню якості обслуговування та оптимізації ресурсів автосервісу.

Об’єкт дослідження: Процеси управління послугами автосервісу, що включають обробку замовлень, облік запасів, керування фінансовими операціями, управління записами клієнтів та комунікацію з користувачами через інформаційні системи.

Предмет дослідження: Технології та методи автоматизації управління послугами автосервісу через веб-застосунок, що охоплює обробку замовлень, облік клієнтських записів, управління запасами, фінансові операції та взаємодію з користувачами. Дослідження спрямоване на вибір оптимальних інструментів і

технологій для розробки веб-застосунку, розробку архітектури системи, забезпечення безпеки даних і інтеграцію функціональних компонентів для ефективної автоматизації бізнес-процесів автосервісу.

Наукова новизна: Наукова новизна роботи полягає в розробці веб-застосунку для управління послугами автосервісу, який поєднує автоматизацію облікових, операційних та комунікаційних процесів в єдиній системі. Пропоноване рішення відрізняється гнучкою архітектурою, що дозволяє адаптуватися під різні бізнес-моделі автосервісів, а також використанням сучасних технологій для підвищення безпеки та оптимізації продуктивності.

Завдання дослідження:

- Проаналізувати основні бізнес-процеси автосервісу, зокрема облік замовлень, управління запасами, комунікацію з клієнтами та фінансові операції, для визначення функціональних вимог до веб-застосунку.
- Вибрати сучасні технології та інструменти для розробки веб-застосунку, які забезпечують ефективну взаємодію між фронтендом і бекендом, а також підтримують надійне зберігання та обробку даних.
- Розробити архітектуру веб-застосунку, що включає структуризацію бази даних, взаємодію між модулями системи та забезпечує масштабованість і гнучкість для адаптації до різних моделей автосервісу.
- Створити інтерфейс користувача, який буде інтуїтивно зрозумілим і зручним для персоналу автосервісу та клієнтів, враховуючи потреби обох груп користувачів.
- Забезпечити безпеку даних, включаючи реалізацію системи автентифікації, авторизації та захисту від типових загроз, таких як SQL-ін'єкції та XSS-атаки.
- Провести тестування системи для перевірки її стабільності, продуктивності та зручності, а також внести необхідні зміни на основі результатів тестування для оптимізації роботи застосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд бізнес-процесів автосервісу та їхніх особливостей

Автосервіси забезпечують обслуговування автомобілів, включаючи діагностику, ремонт, технічне обслуговування та заміну запчастин. Основними бізнес-процесами автосервісу є:

1. **Прийом замовлень**: Спілкування з клієнтами, реєстрація їхніх запитів, обговорення послуг, які необхідно виконати, і попередня оцінка вартості. Цей процес передбачає не лише облік даних клієнта, але й запис деталей замовлення, що стає основою для подальшого планування.
2. **Планування робіт**: На основі прийнятих замовлень адміністратори автосервісу розподіляють завдання серед працівників, визначають пріоритети та терміни виконання. Планування враховує доступність обладнання, наявність запасних частин та робочого часу спеціалістів.
3. **Управління запасами**: Автосервіс потребує постійного контролю за наявністю запчастин і витратних матеріалів, оскільки своєчасна наявність матеріалів є критичною для виконання замовлень. Процес управління запасами включає облік приходу та витрати матеріалів, моніторинг рівня запасів і поповнення за потреби.
4. **Ремонтні та сервісні роботи**: Це основна діяльність автосервісу, яка виконується спеціалістами та передбачає різні види робіт — від діагностики до складних ремонтів. Важливо забезпечити якісне виконання робіт, контроль за дотриманням термінів, а також фіксацію процесу виконання в системі обліку.
5. **Облік та розрахунок вартості послуг**: Після завершення робіт формується підсумкова вартість, що включає вартість запчастин, витратних матеріалів і робіт.

   Правильний облік і розрахунок витрат є ключовими для встановлення адекватної ціни та забезпечення прибутковості автосервісу.

6. **Фінансові операції**: Включає обробку платежів від клієнтів, фінансову звітність, управління доходами та витратами. Фінансовий облік необхідний для відстеження прибутковості, контролю надходжень і витрат, а також для складання фінансових звітів.
7. **Клієнтська підтримка та взаємодія з клієнтами**: Включає обробку звернень клієнтів, інформування про статус робіт, консультації та рекомендації щодо подальшого обслуговування. Важливою складовою є також забезпечення клієнтів історією обслуговування їхніх автомобілів для підвищення рівня обслуговування та лояльності.

Ці процеси є ключовими для функціонування автосервісу, але їхня реалізація вручну або через неавтоматизовані системи створює значне навантаження на персонал, ускладнює облік і підвищує ймовірність помилок. Автоматизація цих процесів за допомогою веб-застосунку дозволить спростити управління послугами, підвищити точність обліку та покращити взаємодію з клієнтами, що є критично важливим для підвищення ефективності роботи автосервісу.

Ця комплексна схема зображена на рисунку 1.1, ілюструє основні бізнес-процеси автосервісу, відображаючи всі ключові етапи й взаємозв'язки між ними. Розглянемо основні елементи схеми:

1. **Клієнтський шлях**:
   - Початок взаємодії з клієнтом починається з прийому замовлення на послуги автосервісу. Тут клієнт може зареєструвати своє замовлення, зазначити проблеми та вибрати необхідні послуги.
   - Після завершення обслуговування клієнт отримує підсумковий звіт про виконані роботи і здійснює фінальну оплату.
2. **Процеси автосервісу**:
   - **Прийом замовлень**: Блок, у якому відбувається збір даних про замовлення клієнта, запис даних в систему та підготовка до виконання.
   - **Планування робіт**: Цей етап включає розподіл завдань серед працівників, призначення спеціалістів для виконання певних робіт, розрахунок часу та ресурсів для кожного замовлення.
   - **Управління запасами**: Включає перевірку наявності потрібних запасних частин, контроль запасів на складі, замовлення нових деталей за необхідності.
   - **Виконання ремонту**: Фактичний процес ремонту автомобіля, який виконує технічний персонал. Всі роботи реєструються в системі.

3. **Фінансові операції**:
   - Цей розділ охоплює обробку платежів та ведення фінансової звітності для кожного замовлення. Тут зазначається вартість послуг, облік витрат на запасні частини, та формування підсумкової суми для оплати.
4. **Взаємозв'язки об'єктів**:
   - **Клієнти**, **Замовлення**, **Послуги**, **Запаси** і **Працівники** пов'язані між собою зв'язками, які відображають потік інформації та взаємодію між процесами.
   - Клієнти створюють замовлення, які обробляються персоналом автосервісу з використанням матеріалів з інвентаря, а також є відправною точкою для фінансових операцій та звітності.

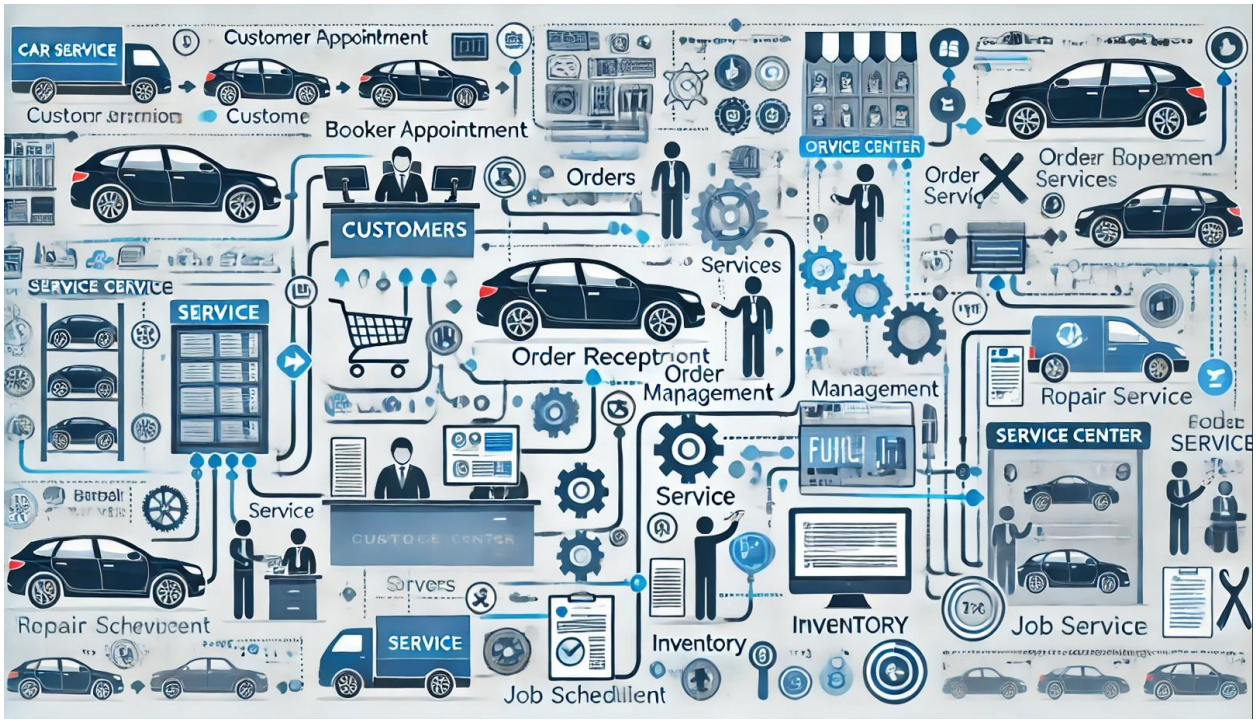

Рисунок 1.1 – Схема бізнес процесів для автосервісу

1.2 Аналіз існуючих проблем в управлінні автосервісом та необхідність автоматизації

В управлінні автосервісом існує низка проблем, які обмежують ефективність роботи та ускладнюють надання якісного обслуговування клієнтам. Основні з них:

1. **Відсутність централізованого обліку замовлень і даних про клієнтів**: Найчастіше замовлення та інформація про клієнтів зберігаються в паперових документах або таблицях, що ускладнює їх пошук, оновлення та аналіз. Це призводить до зниження швидкості обробки замовлень і підвищує ризик втрати даних.
2. **Труднощі в управлінні запасами**: У ручному режимі складно точно контролювати наявність запчастин та витратних матеріалів, що може

призвести до ситуацій, коли потрібні деталі відсутні під час виконання ремонту. Це впливає на терміни обслуговування, збільшує витрати на експрес-доставку та знижує загальну продуктивність.

3. **Складність у плануванні та розподілі робіт**: Без автоматизованої системи важко ефективно планувати завантаження майстрів і розподіл завдань. Через це можуть виникати нерівномірне навантаження на персонал, простої, або, навпаки, перевантаження, що погіршує якість обслуговування.
4. **Тривалий процес формування звітності та аналізу даних**: Відсутність зручних інструментів для збору даних ускладнює підготовку звітів і фінансового аналізу. Це затримує прийняття управлінських рішень і перешкоджає швидкій реакції на зміни в бізнесі.
5. **Низька якість комунікації з клієнтами**: Без систематизованої інформації про клієнтів та їхні замовлення стає важче підтримувати високий рівень обслуговування та оперативно інформувати клієнтів про статус ремонту, терміни виконання та інші питання.
6. **Підвищений ризик помилок і людський фактор**: Ручне введення даних і відсутність автоматизованих перевірок збільшують імовірність помилок у записах, що може призвести до помилкових замовлень, неточних розрахунків і незадоволеності клієнтів.

Необхідність автоматизації стає очевидною з огляду на ці проблеми. Впровадження автоматизованої системи управління допоможе:

- Централізувати облік замовлень і клієнтів;
- Забезпечити своєчасний контроль за запасами;
- Оптимізувати розподіл завдань між працівниками;
- Скоротити час на формування звітів і прийняття рішень;
- Підвищити рівень обслуговування клієнтів;
- Зменшити кількість помилок, знижуючи вплив людського фактора.

Таким чином, автоматизація є ключовим рішенням для підвищення ефективності автосервісу, зниження витрат і поліпшення взаємодії з клієнтами.

Схема проблем управління автосервісом

Опис:

Ця діаграма на рисунку 1.2 візуалізує ключові проблеми, які виникають в управлінні автосервісом. Вона складається з:

- Центрального блоку "Управління автосервісом".
- Від нього відходять проблемні зони:
- Розрізненість інформації (невпорядковані дані клієнтів та замовлень).
- Неефективне управління запасами (відсутність деталей в критичний момент).
- Проблеми в комунікації (відсутність зворотного зв’язку з клієнтами).
- Вплив людського фактора (ризик помилок через ручне введення).
- Стрілки між блоками демонструють, як одна проблема впливає на інші.

Рисунок 1.2 - Схема проблем управління автосервісом

Порівняння "до і після" автоматизації
Опис:
Цей рисунок 1.3 складається з двох частин:

До автоматизації:

- Покажемо хаос у роботі: кілька джерел даних, заплутані комунікації між працівниками, розриви у клієнтському обслуговуванні.

Після автоматизації:

- Упорядкована система: центральна база даних, автоматизоване планування, швидка комунікація з клієнтами.
- Використання спрощених процесів показує, як автоматизація знижує навантаження та покращує результати.

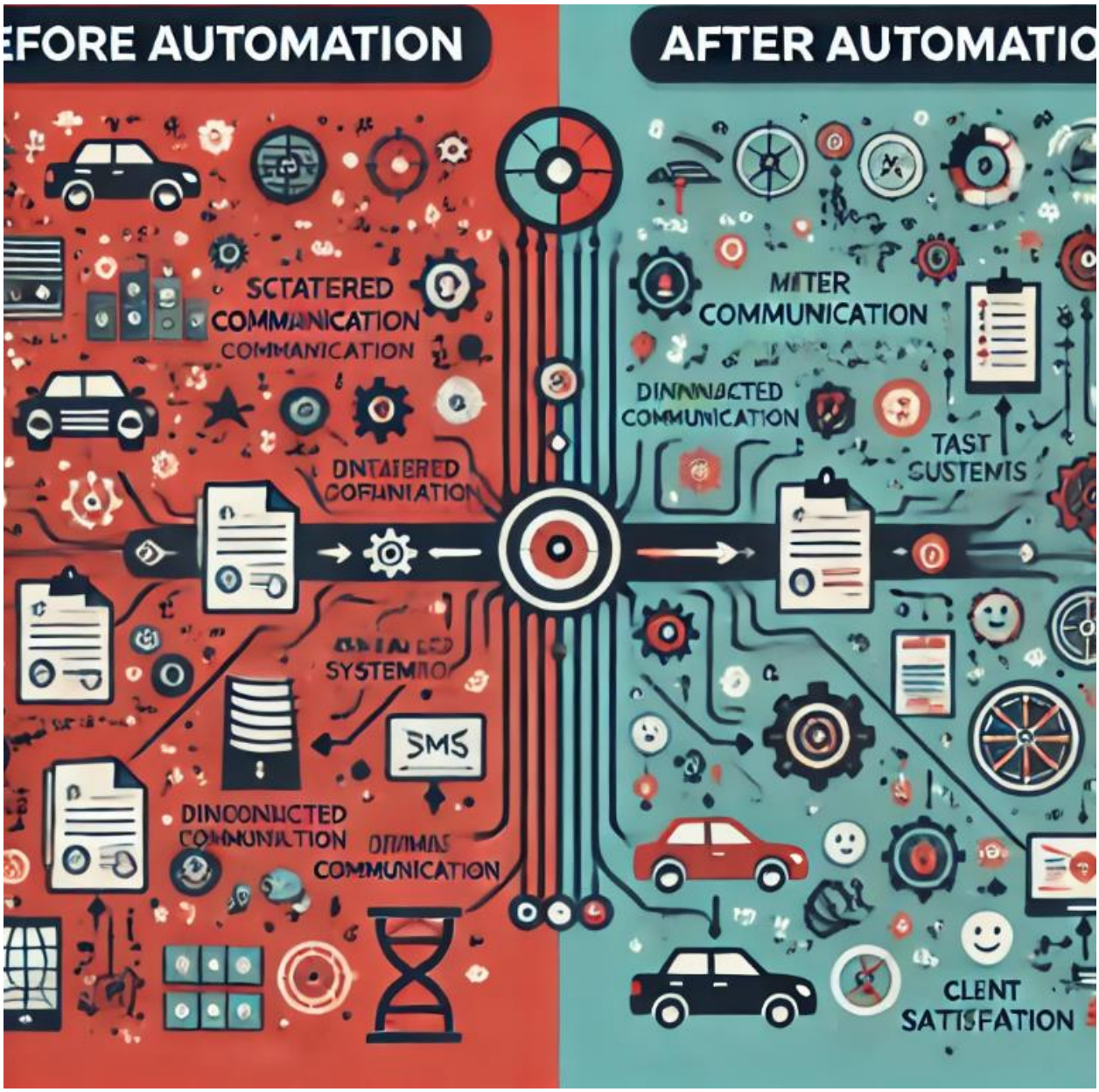

Рисунок 1.3 - Порівняння "до і після" автоматизації

Клієнтський шлях у автосервісі

Опис:

Це рисунок 1.4 клієнтського шляху у форматі діаграми, що охоплює такі етапи:

- Прийом замовлення (через менеджера або онлайн).
- Планування ремонту (розподіл завдань і ресурсів).
- Виконання роботи (ремонт і обслуговування).
- Інформування клієнта (про статус ремонту через SMS або email).
- Завершення процесу (оплата і видача автомобіля).
  Цей шлях показаний у вигляді стрічки або послідовності блоків із піктограмами для кожного етапу.

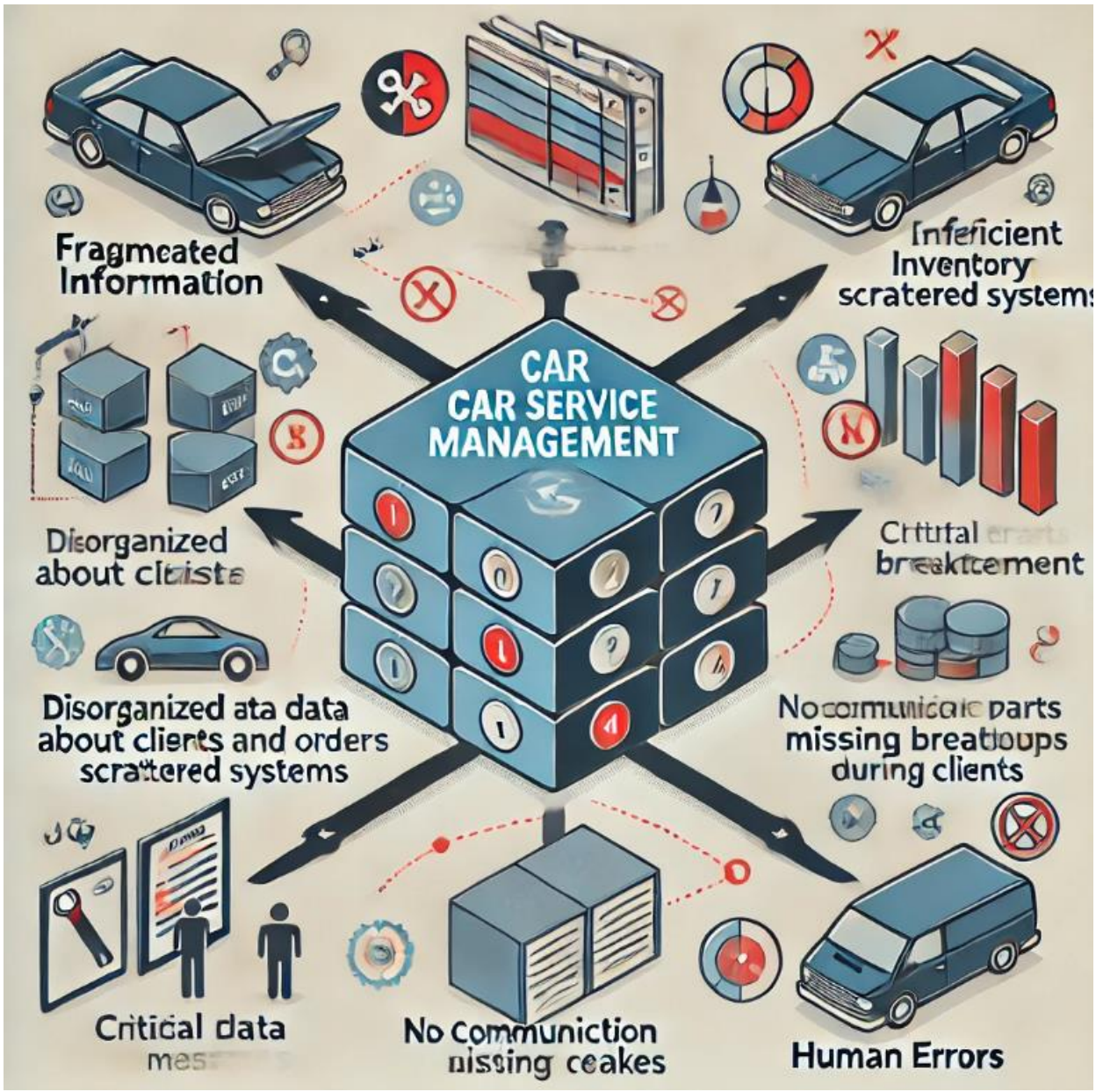

Рисунок 1.4 - Клієнтський шлях у автосервісі

1.3 Визначення функціональних вимог до веб-застосунку для оптимізації обліку, замовлень, управління запасами і фінансових операцій

Для створення ефективного веб-застосунку, який забезпечить автоматизацію та оптимізацію роботи автосервісу, необхідно визначити основні функціональні вимоги. Ці вимоги охоплюють ключові аспекти управління процесами, забезпечення зручності для користувачів та інтеграцію з іншими системами.

Основні функціональні вимоги

1. **Облік клієнтів і замовлень:**

- Реєстрація нових клієнтів із збереженням їх контактної інформації.
- Зберігання історії замовлень кожного клієнта (перелік послуг, дати, суми).
- Пошук клієнтів за різними критеріями (ПІБ, телефон, номер авто).
- Можливість швидкого створення та редагування замовлення.

2. **Управління запасами:**
   - Контроль наявності запчастин та витратних матеріалів у режимі реального часу.
   - Автоматичне сповіщення про критичний залишок товарів.
   - Функціонал для формування замовлення постачальникам.
   - Облік руху запасів: прихід, витрата, повернення.
3. **Планування та управління завданнями:**
   - Календар завантаження працівників із можливістю розподілу завдань.
   - Інтеграція з графіком роботи майстрів.
   - Можливість зміни або перенесення завдань із сповіщенням зацікавлених осіб.
4. **Фінансовий облік:**
   - Формування рахунків та актів виконаних робіт.
   - Звітність за доходами, витратами, прибутком у розрізі періодів, клієнтів або послуг.
   - Інтеграція з платіжними системами для прийому оплат онлайн.
5. **Інформаційна підтримка клієнтів:**
   - Автоматичне сповіщення клієнтів про статус замовлення (наприклад, завершено, очікує частин).
   - Інтеграція з SMS та email-розсилками.
   - Нагадування клієнтам про планове ТО або закінчення гарантійного терміну.

6. **Інструменти для аналізу та моніторингу:**
   - Побудова графіків і діаграм для аналізу популярності послуг, завантаження працівників, використання запасів.
   - Інтерактивна панель моніторингу ключових показників (KPI).
7. **Інтеграція з іншими системами:**
   - Підключення до складських програм для синхронізації залишків.
   - Інтеграція з CRM-системами для розширеного обліку клієнтів.
   - Можливість експорту та імпорту даних у форматах Excel, CSV.
8. **Інтуїтивний інтерфейс:**
   - Проста та зручна навігація для швидкого доступу до основних функцій.
   - Налаштування прав доступу для різних ролей (адміністратор, майстер, клієнт).
9. **Модуль адміністрування:**
   - Налаштування послуг, цін та розкладу роботи персоналу.
   - Управління користувачами системи та їхніми ролями.
   - Контроль за безпекою даних та резервне копіювання.

На рисунку 1.5 зображена блок-схема функціональних модулів веб-застосунку для автоматизації автосервісу.

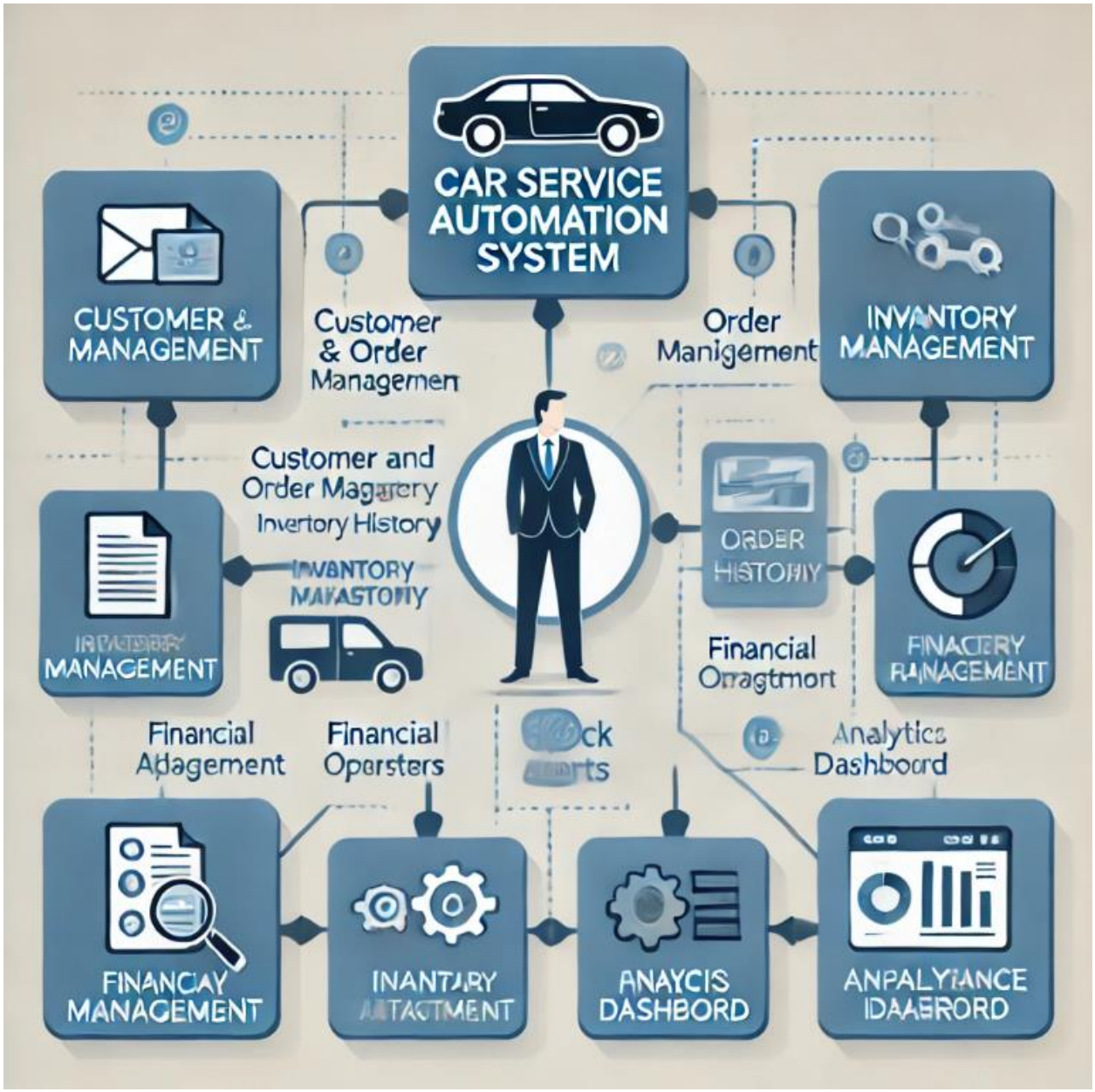

Рисунок 1.5 - Блок-схема функціональних модулів веб-застосунку для автоматизації автосервісу.

1.4 Огляд наявних рішень та обґрунтування потреби у розробці нового інструменту

Аналіз існуючих програм для управління автосервісом

На ринку існує чимало програмних рішень, які допомагають автоматизувати процеси в автосервісі. Однак більшість із них мають такі недоліки:

1. **Висока вартість ліцензій та обслуговування:**
   - Багато популярних рішень, як-от 1С або AutoRepair Cloud, мають значні витрати на впровадження, щомісячні платежі або додаткові модулі.
   - Для невеликих автосервісів це стає фінансово недоцільним.
2. **Недостатня локалізація:**
   - Програми здебільшого орієнтовані на міжнародний ринок і не враховують специфіку українських автосервісів, наприклад, податкові системи або нормативні вимоги.
3. **Обмежена функціональність:**
   - Деякі рішення не підтримують повноцінного управління запасами або інтеграції з фінансовими сервісами.
   - Відсутність модулів для аналізу ефективності роботи або автоматичного сповіщення клієнтів.
4. **Складність у користуванні:**
   - Інтерфейси багатьох програм занадто складні, що вимагає довгого навчання персоналу.
5. **Відсутність адаптивності:**
   - Деякі системи не дозволяють налаштовувати функціонал під потреби конкретного бізнесу.

Обґрунтування потреби у розробці нового інструменту

На основі аналізу існуючих рішень стає зрозуміло, що автосервісам потрібен більш доступний та ефективний інструмент. Основні причини розробки нового веб-застосунку:

1. **Економічна доцільність:**
   - Створення програмного забезпечення з мінімальними витратами на впровадження та обслуговування.
2. **Адаптація до локальних потреб:**
   - Підтримка української мови, врахування місцевих нормативів та потреб малого і середнього бізнесу.
3. **Розширений функціонал:**
   - Наявність модулів для автоматизації обліку, управління запасами, фінансових операцій, планування завдань та комунікації з клієнтами.
4. **Простота використання:**
   - Інтуїтивний інтерфейс для зручності роботи персоналу без необхідності тривалого навчання.
5. **Гнучкість та масштабованість:**
   - Можливість налаштовувати систему під конкретні бізнес-процеси та додавати нові модулі у разі зростання бізнесу.

Огляд наявних програм для управління автосервісом

У сфері управління автосервісами вже існують різні програмні продукти, серед яких:

- **AutoRepair Cloud** — хмарне рішення для управління замовленнями та обліку клієнтів.
- **CarServicePro** — інструмент для автоматизації обліку запасів і фінансів.
- **1С: Управління автосервісом** — локальна програма з широким функціоналом.

Хоча ці продукти можуть задовольнити базові потреби, вони часто мають обмеження, які знижують їхню ефективність:

1. **Відсутність інтеграції з сучасними технологіями.**
   - Немає підтримки мобільних платформ або інтерактивних панелей управління.
2. **Недоступність для малого бізнесу.**
   - Більшість програм дорогі або вимагають складного налаштування.
3. **Мінімальні можливості кастомізації.**
   - Важко адаптувати функціонал під специфіку роботи конкретного автосервісу.
4. **Неадаптованість до локальних умов.**
   - Відсутність урахування місцевого законодавства, податкових систем та потреб українського ринку.

Недоліки існуючих рішень

1. **Складний інтерфейс.**
   - Більшість програм вимагають навчання персоналу, що збільшує витрати часу та коштів.
2. **Неефективність аналітики.**
   - Наявні інструменти часто пропонують лише базову звітність без глибокого аналізу.
3. **Немає підтримки роботи в режимі реального часу.**
   - Дані про залишки, статус замовлень або фінансові операції оновлюються із затримками.

Обґрунтування потреби у створенні нового інструменту

Розробка нового веб-застосунку для автоматизації автосервісів є актуальною через такі причини:

1. **Доступність.**
   - Система буде орієнтована на малий та середній бізнес, із доступною ціною та простим налаштуванням.
2. **Адаптивність до локальних потреб.**
   - Врахування специфіки роботи українських автосервісів.
3. **Інноваційність.**
   - Використання сучасних технологій, включаючи мобільний доступ, хмарне збереження даних та інтеграцію зі сторонніми сервісами.
4. **Модульний підхід.**
   - Гнучкість у виборі та розширенні функціональності для конкретних бізнес-потреб.
5. **Інтуїтивний дизайн.**
   - Простий інтерфейс для швидкого освоєння працівниками автосервісу.

На рисунку 1.6 зображено порівняння існуючих програм для автосервісів.

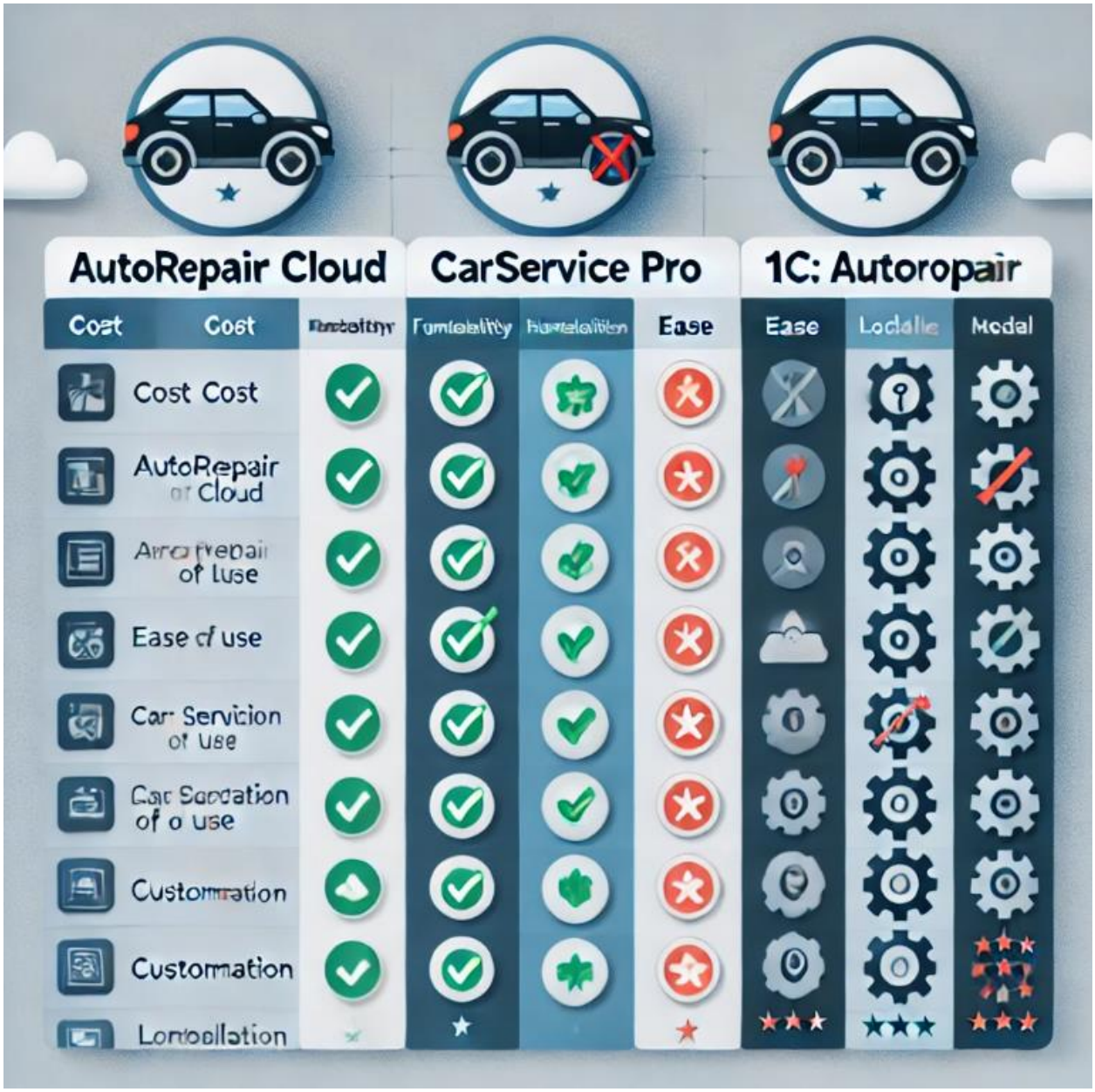

Рисунок 1.6 – Порівняння програм

2 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ

2.1 Обґрунтування вибору стеку технологій

1. Фронтенд (інтерфейс користувача)
Для створення сучасного, адаптивного та зручного інтерфейсу пропонується використовувати:

- **HTML5 та CSS3:**
  - Основні стандарти для структурування контенту та стилізації.
  - Забезпечують адаптивність та кросбраузерність.
- **JavaScript:**
  - Основна мова для інтерактивності користувацького інтерфейсу.
  - У поєднанні з фреймворками дозволяє створювати динамічний контент.
- **React.js:**
  - Потужна бібліотека для створення швидких та інтерактивних інтерфейсів.
  - Має модульну архітектуру, що дозволяє розділяти код на компоненти, спрощуючи розробку та підтримку.
  - Підтримує серверний рендеринг для SEO-оптимізації та зменшення часу завантаження.

2. Бекенд (серверна частина)
Для обробки запитів та управління даними використовується:

- **Node.js:**
  - Серверне середовище виконання JavaScript, що забезпечує високу продуктивність завдяки подієво-орієнтованій моделі.
  - Підходить для обробки великої кількості одночасних запитів.
- **Express.js:**
  - Мінімалістичний фреймворк для створення серверної логіки.
  - Простий у використанні та забезпечує гнучкість при розробці RESTful API.

3. База даних
Для зберігання даних про клієнтів, замовлення, фінансові операції та запаси:

- **MongoDB:**
  - NoSQL-база даних, яка дозволяє гнучко зберігати дані у форматі JSON-подібних документів.
  - Забезпечує високу масштабованість та швидкість операцій.
  - Ідеально підходить для роботи з нереляційними даними, які можуть змінювати свою структуру.
- **MySQL (додатковий варіант):**
  - Реляційна база даних для складних запитів і зв’язків між даними.
  - Використовується у випадках, коли структура даних є чітко визначеною.

4. Хостинг та інфраструктура
   - **AWS (Amazon Web Services):**
     - Надійний хмарний сервіс для хостингу застосунку та бази даних.
     - Підтримує автоматичне масштабування та забезпечує високу доступність.
   - **Docker:**
     - Контейнеризація забезпечує стандартизоване середовище для розробки та розгортання.
     - Полегшує інтеграцію і тестування системи.
   - **Nginx:**
     - Використовується як веб-сервер і зворотний проксі-сервер для підвищення продуктивності.
5. Додаткові інструменти та технології
   - **Redux:**
     - Для управління станом застосунку в складних фронтенд-проектах.
     - Забезпечує централізоване сховище для всіх даних інтерфейсу.
   - **JWT (JSON Web Tokens):**
     - Для аутентифікації та авторизації користувачів.
     - Забезпечує безпечний обмін даними між клієнтом і сервером.
   - **Stripe або PayPal API:**
     - Інтеграція платіжних систем для обробки транзакцій.
   - **Chart.js:**
     - Бібліотека для візуалізації даних (графіки, діаграми) в аналітичному модулі.

Причини вибору зазначених технологій

1. **Гнучкість та швидкість розробки:**
   - Використання JavaScript як на фронтенді, так і на бекенді спрощує розробку та підтримку.
2. **Масштабованість:**
   - Використання Node.js та MongoDB дозволяє легко масштабувати систему зі зростанням обсягів даних і запитів.
3. **Кросплатформеність:**
   - React.js дозволяє створювати адаптивний інтерфейс для різних пристроїв (десктопи, планшети, мобільні телефони).
4. **Безпека:**
   - Інструменти аутентифікації, такі як JWT, гарантують захист даних.
5. **Модульність:**
   - Використання Docker і мікросервісної архітектури дозволяє оновлювати частини системи без зупинки її роботи.
6. **Зручність візуалізації даних:**
   - Chart.js і React.js забезпечують простий спосіб інтеграції аналітики у веб-застосунок.

На рисунку 1.7 зображено технологічний стек для веб-застосунку управління автосервісом.

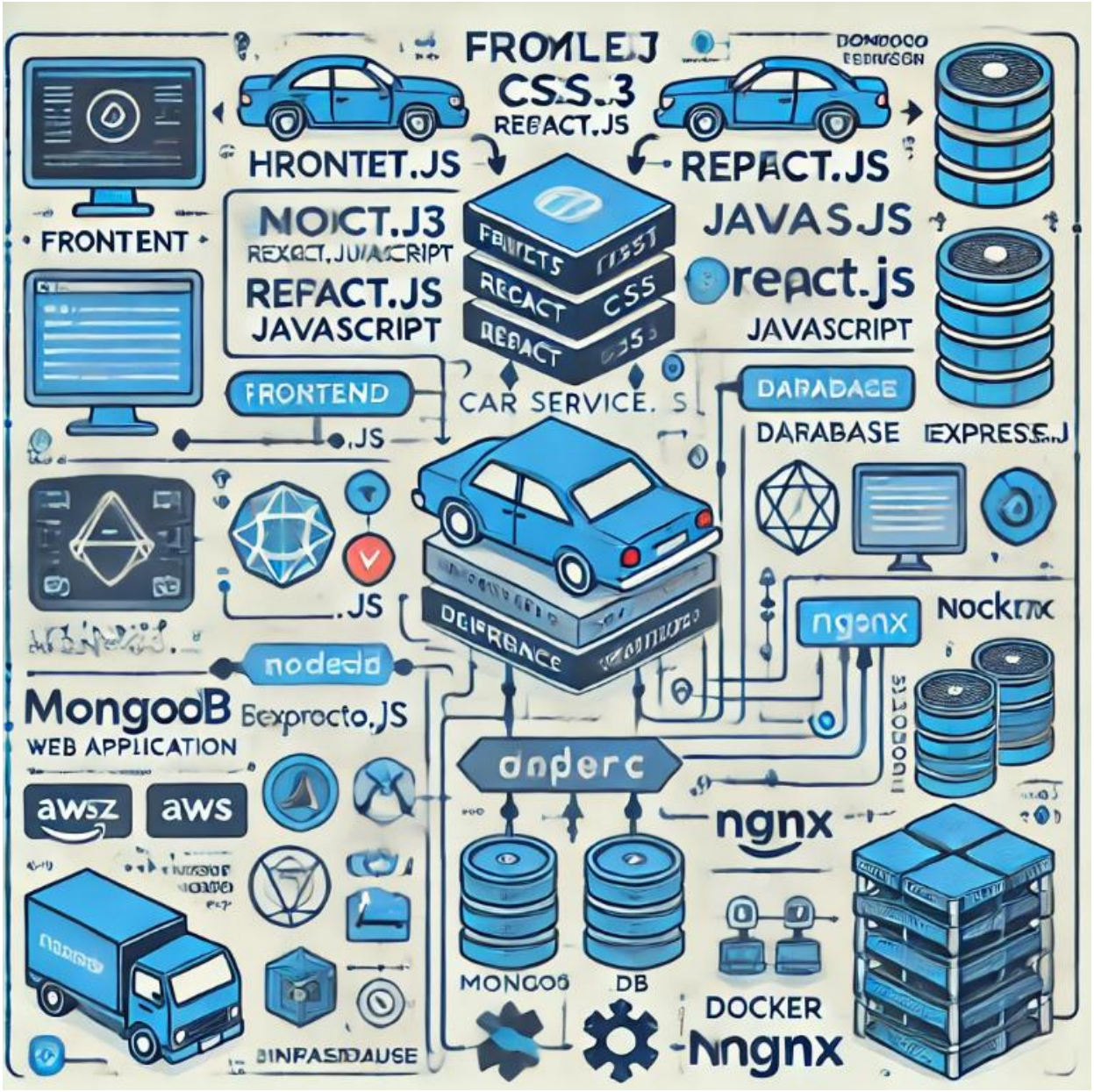

Рисунок 1.7 – Технологічний стек

2.2 Технології для фронтенду (React, Vue)

Розробка сучасного веб-застосунку для управління автосервісом вимагає використання гнучких і потужних інструментів для створення зручного та інтерактивного інтерфейсу користувача. Серед популярних фреймворків для фронтенду найчастіше обирають **React** або **Vue.js**.

React.js

React.js — це бібліотека для створення користувацьких інтерфейсів, розроблена компанією Facebook. Вона активно використовується у великих проектах завдяки своїй продуктивності та широкій спільноті розробників.

Переваги React:

1. **Компонентний підхід:**
   - Інтерфейси будуються з окремих багаторазових компонентів, зображено на рисунку 1.8.
   - Спрощує розробку та підтримку великого коду.
2. **Віртуальний DOM:**
   - Забезпечує високу продуктивність шляхом мінімізації операцій зі справжнім DOM.
3. **Підтримка бібліотек:**
   - Велика кількість бібліотек для розширення функціоналу, наприклад, Redux (управління станом), React Router (маршрутизація).
4. **Підтримка серверного рендерингу (SSR):**
   - Покращує SEO-оптимізацію та швидкість завантаження.

Недоліки React:

- Необхідність використання додаткових бібліотек для повного функціоналу.
- Відносно висока крива навчання для новачків.

Vue.js

Vue.js — це прогресивний фреймворк для створення інтерфейсів користувача, який надає простоту та потужність для розробки. Vue ідеально підходить для проектів, які потребують швидкої реалізації.

Переваги Vue:

1. **Простота вивчення:**
   - Інтуїтивний API та документація, що легко освоюється новачками.
2. **Двостороння прив'язка даних:**
   - Спрощує синхронізацію між моделями та інтерфейсом, зображено на рисунку 1.9.
3. **Гнучкість:**
   - Можливість використовувати Vue як для простих віджетів, так і для великих додатків.
4. **Вбудовані інструменти:**
   - Vue Router для маршрутизації та Vuex для управління станом інтегровані в екосистему.

Недоліки Vue:

- Менша популярність у порівнянні з React, що обмежує доступ до готових рішень і ресурсів.
- Менша підтримка для серверного рендерингу.

Для якіснішого порівяння представлена таблиця 1.1 – Порівняння React і Vue

Таблиця 1.1 - Порівняння React і Vue:

Значення	React.js	Vue.js
Продуктивність	Висока, завдяки віртуальному DOM	Трохи поступається React
Крива навчання	Більш стрімка через складність екосистеми	Проста та інтуїтивна
Гнучкість	Велика кількість додаткових бібліотек	Менш гнучкий, але має вбудовані інструменти
Масштабованість	Підходить для великих проектів	Переважно використовується для середніх і малих проектів
Спільнота	Велика, активна	Менша, але зростаюча

Обґрунтування вибору для веб-застосунку управління автосервісом:

Для проекту управління автосервісом рекомендується використовувати **React.js** через його здатність працювати з великими проектами, широкі можливості кастомізації та високу продуктивність. React дозволить реалізувати:

- Інтерактивні сторінки замовлень і обліку.
- Інтеграцію сторонніх бібліотек для роботи з графіками (наприклад, Chart.js).
- Розширення системи без значних змін у кодовій базі.

Vue.js може бути використаний, якщо пріоритетом є швидкість розробки та простота реалізації, наприклад, для створення менш масштабного застосунку або прототипу.

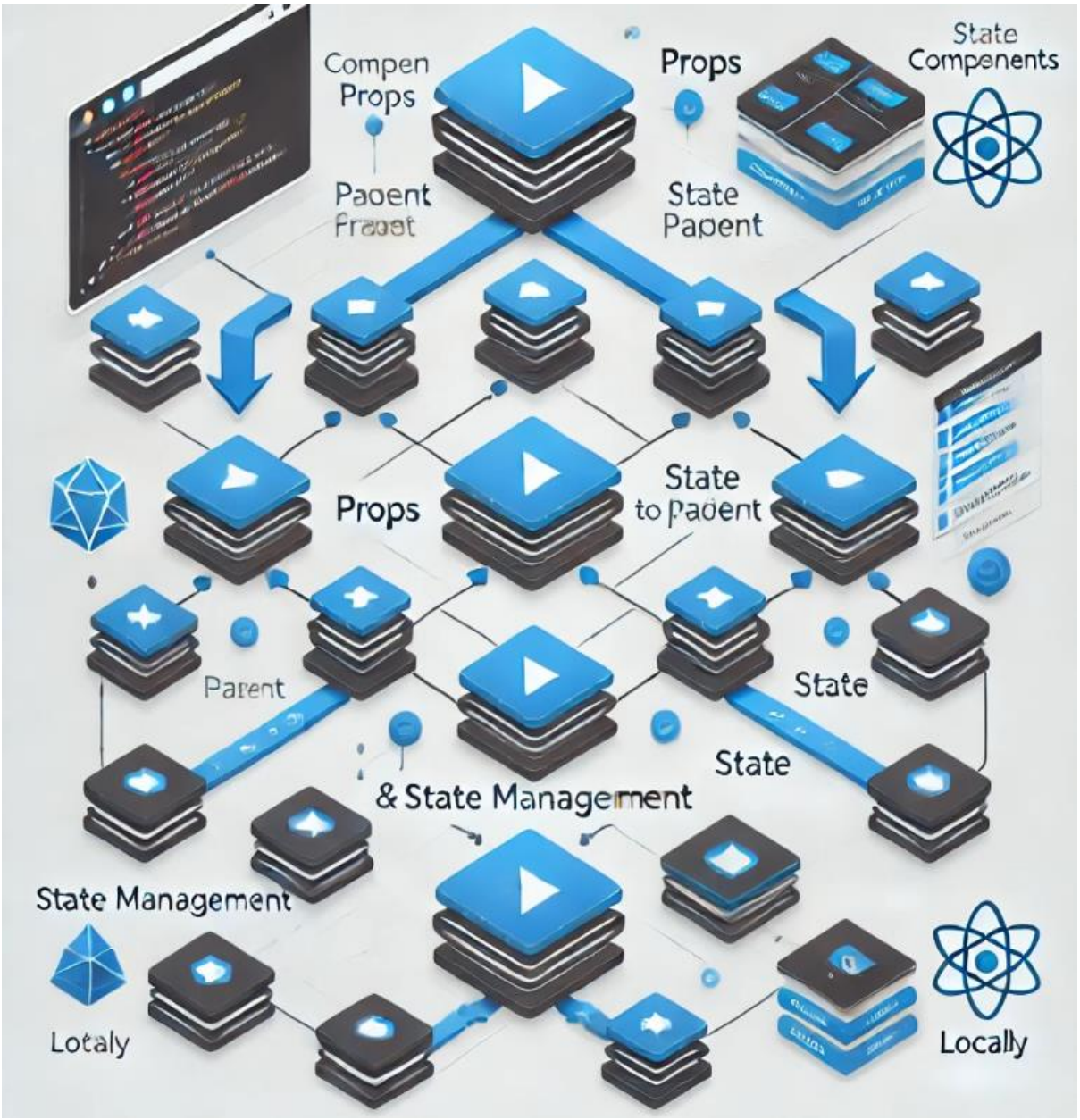

Рисунок 1.8 – Схема роботи компонентів React

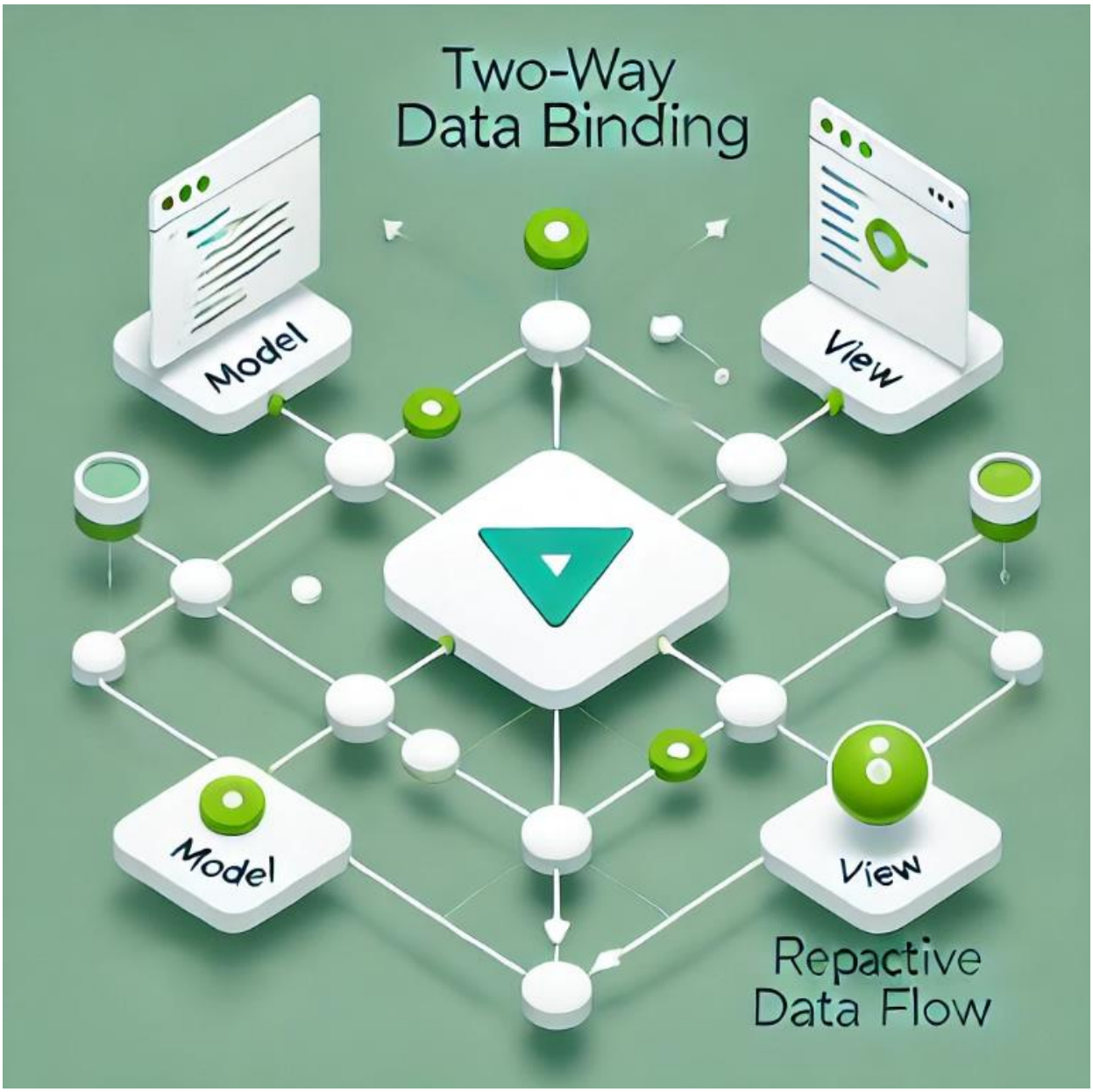

Рисунок 1.9 – Схема двохстороньої привзяки на Vue

2.3 Технології для бекенду (Node.js, Express)

Node.js

Node.js — це серверне середовище виконання JavaScript, яке побудоване на рушії V8 (Google Chrome). Воно дозволяє розробляти сервери, використовуючи JavaScript, забезпечуючи високу швидкість виконання та ефективність.
Переваги Node.js:

1. **Подієво-орієнтована архітектура:**
    - Завдяки асинхронній обробці запитів Node.js здатний обслуговувати тисячі з’єднань одночасно.
2. **Масштабованість:**
    - Легкість масштабування через поділ на процеси або розподілену архітектуру.

3. **Єдиний стек JavaScript:**
    - Можливість використовувати одну мову програмування як для фронтенду, так і для бекенду, що зменшує складність розробки.
4. **Велика екосистема пакетів (npm):**
    - npm пропонує понад 1 мільйон пакетів, які спрощують розробку, зокрема для роботи з базами даних, аутентифікації, API тощо.
5. **Швидкість:**
    - Node.js ідеально підходить для додатків, які потребують швидкої обробки даних у реальному часі, наприклад, чати або системи моніторингу.

Недоліки Node.js:

- Обмеження при роботі з обчислювально-інтенсивними задачами через однопоточну модель.
- Потреба ретельного планування та управління потоками.

Express.js

Express.js — це мінімалістичний веб-фреймворк для Node.js, який спрощує створення серверної логіки. Він забезпечує необхідний набір інструментів для розробки API та веб-додатків.

Переваги Express.js:

1. **Простота використання:**
    - Легкий синтаксис для створення серверів і маршрутизації запитів.
2. **Гнучкість:**
    - Підтримує інтеграцію з іншими фреймворками та бібліотеками.
3. **Масштабованість:**
    - Можливість додавання middleware для обробки запитів і відповіді, що дозволяє легко розширювати функціонал.
4. **Підтримка REST API:**
    - Ідеально підходить для створення API для взаємодії між фронтендом і бекендом.

Недоліки Express.js:

- Менш структурований, що може призвести до неорганізованого коду у великих проектах.
- Не містить вбудованих засобів для ORM (Object-Relational Mapping) або обробки складних запитів до баз даних (потребує зовнішніх бібліотек, таких як Sequelize або Mongoose).

Переваги використання Node.js та Express.js у веб-застосунку для автосервісу

1. **Обробка одночасних запитів:**
    - Забезпечує високу продуктивність під час обробки замовлень, запитів до бази даних і оновлення запасів у реальному часі.
2. **Масштабованість:**
    - Ідеально підходить для системи, яка може зростати разом із розширенням бізнесу.

3. **Швидка розробка:**
    - Завдяки мінімалістичному підходу Express.js та доступу до багатої екосистеми npm.
4. **API-орієнтована архітектура:**
    - Легкість інтеграції з іншими модулями, такими як платіжні системи (Stripe, PayPal) або модулі аналітики.

Приклади реалізації функціоналу:

1. **Node.js:**
    - Сервер для обробки запитів на реєстрацію клієнтів або перегляд історії обслуговування.
2. **Express.js:**
    - Створення маршруту POST /orders для додавання нового замовлення на обслуговування.
    - Використання middleware для валідації даних клієнта перед збереженням у базу даних.

2.4 Інструменти для забезпечення безпеки даних

Захист даних — це ключовий аспект будь-якого веб-застосунку, особливо якщо він має справу з конфіденційною інформацією, як-от особисті дані клієнтів, замовлення, фінансова інформація або інформація про запаси.
У веб-застосунку для управління автосервісом рекомендується використовувати такі інструменти для забезпечення безпеки:
1. Аутентифікація та авторизація

- **JWT (JSON Web Tokens):**
    - Інструмент для безпечної передачі даних між клієнтом і сервером. Використовується для забезпечення аутентифікації користувачів.
    - Дозволяє створювати токени з обмеженим терміном дії, запобігаючи несанкціонованому доступу.
- **OAuth 2.0:**
    - Протокол авторизації, який дозволяє користувачам надавати доступ до своїх даних без передачі облікових даних.
    - Може використовуватися для інтеграції з Google, Facebook або іншими зовнішніми сервісами.
- **Passport.js:**
    - Бібліотека для Node.js, що підтримує різні стратегії аутентифікації (JWT, OAuth).

2. Захист бази даних

- **Шифрування даних:**
    - Шифрування конфіденційних даних, таких як паролі клієнтів, перед збереженням у базу даних (наприклад, використання bcrypt або Argon2).

- **Використання ORM:**
  - ORM (Object-Relational Mapping) інструменти, такі як Sequelize чи Mongoose, допомагають запобігти SQL-ін’єкціям, автоматично очищаючи запити.
- **Резервне копіювання:**
  - Регулярне автоматизоване створення резервних копій бази даних для захисту від втрати даних.

3. Захист передачі даних
- **SSL/TLS сертифікати:**
  - Забезпечують шифрування трафіку між сервером і клієнтом, захищаючи від перехоплення даних.
- **Шифрування API-запитів:**
  - Використання HTTPS та токенів для підвищення безпеки API-запитів.

4. Захист сервера
- **Helmet.js:**
  - Middleware для Express.js, яке забезпечує встановлення HTTP-заголовків для захисту від поширених атак, таких як XSS (міжсайтовий скриптинг) та Clickjacking.
- **Rate Limiting (обмеження запитів):**
  - Використання таких бібліотек, як express-rate-limit, для обмеження кількості запитів від одного користувача. Це допомагає запобігти атакам типу "відмова в обслуговуванні" (DoS).
- **Сканування вразливостей:**
  - Інструменти, як-от Snyk, допомагають ідентифікувати та усунути вразливості у залежностях проекту.

5. Моніторинг та аудит
- **Audit Logs:**
  - Логування дій користувачів та системи для виявлення підозрілих операцій.
  - Збереження логів у захищеному середовищі, наприклад, у хмарі або за допомогою рішень на основі ELK Stack.
- **Системи моніторингу:**
  - Інструменти, як-от Prometheus та Grafana, для моніторингу активності сервера.

3 ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Архітектура веб-застосунку

Архітектура веб-застосунку для управління автосервісом має бути модульною, масштабованою та забезпечувати високу продуктивність. Вона розробляється за принципами багаторівневої структури, що розділяє відповідальності між різними компонентами.

Основні рівні архітектури:

1. **Клієнтський рівень (Frontend):**
   - Відповідає за взаємодію з користувачем через інтуїтивно зрозумілий графічний інтерфейс.
   - Технології: React.js або Vue.js для створення динамічних і реактивних інтерфейсів.
   - Функції:
     - Форма реєстрації та авторизації.
     - Списки замовлень, клієнтів, запасів.
     - Візуалізація даних (графіки, таблиці).
2. **Серверний рівень (Backend):**
   - Забезпечує бізнес-логіку та обробку запитів від клієнтів.
   - Технології: Node.js з використанням фреймворку Express.js.
   - Функції:
     - Обробка запитів (додавання, зміна, видалення даних).
     - Авторизація користувачів через JWT.
     - Взаємодія з базою даних.
3. **Рівень даних (Database):**
   - Відповідає за збереження даних у структурованому вигляді.
   - Технології:
     - Реляційні бази даних: PostgreSQL або MySQL.
     - Документо-орієнтовані бази: MongoDB (якщо потрібна гнучкість у збереженні даних).
   - Функції:
     - Зберігання інформації про клієнтів, послуги, фінанси, склади.
     - Забезпечення резервного копіювання.
4. **Рівень API (REST/GraphQL):**
   - Забезпечує взаємодію між фронтендом і бекендом через стандартизовані інтерфейси.
   - Функції:
     - Виклик API для управління замовленнями, клієнтами та запасами.
     - Підтримка документованого API для зовнішніх інтеграцій (наприклад, з платіжними сервісами).
5. **Сервісний рівень (Допоміжні сервіси):**

- Використовується для інтеграції з сторонніми сервісами та виконання спеціалізованих задач.
- Технології:
  - AWS S3 для збереження файлів (документи, чеки).
  - Stripe/PayPal для інтеграції платежів.
  - Twilio для надсилання SMS/повідомлень клієнтам.

Типи архітектурних шаблонів:

- **Клієнт-серверна архітектура:**
  - Застосовується для розподілу навантаження між сервером і клієнтом.
- **Мікросервісна архітектура:**
  - Використовується для розділення функцій на незалежні сервіси (опціонально для великих систем).
- **Багатошарова (n-tier) архітектура:**
  - Включає чітке розмежування рівнів для підвищення надійності та масштабованості.

Комунікація між компонентами:

1. **HTTP/HTTPS-запити:**
   - Клієнт надсилає запити до серверу через REST API або GraphQL.
2. **WebSocket:**
   - Забезпечує обмін даними в реальному часі, наприклад, для оновлення статусу замовлення.
3. **Кешування:**
   - Redis або Memcached для прискорення доступу до часто використовуваних даних.

Переваги архітектури:

- **Масштабованість:** можливість додавання нових функцій без змін основного коду.
- **Безпека:** чіткий розподіл даних та рівнів.
- **Продуктивність:** оптимізація запитів і швидкий доступ до даних.

3.2 Модулі системи та їх взаємодія

Веб-застосунок для управління послугами автосервісу розробляється як модульна система, де кожен модуль відповідає за виконання окремого функціонального блоку. Такий підхід забезпечує гнучкість, масштабованість та легкість обслуговування.

Основні модулі системи:

1. **Модуль управління замовленнями**
   - **Функції:**
     - Додавання, редагування та видалення замовлень.
     - Відстеження статусу виконання послуг.

    - Генерація рахунків.
- **Взаємодія:**
    - Обмін даними з клієнтським модулем.
    - Збереження інформації в базі даних через модуль управління даними.

2. **Модуль управління клієнтами**
    - **Функції:**
        - Реєстрація та зберігання даних про клієнтів (ім'я, контактна інформація, історія замовлень).
        - Сегментація клієнтів за критеріями (наприклад, за частотою замовлень).
        - Інтеграція з SMS або email-сервісами для сповіщень.
    - **Взаємодія:**
        - Інтеграція з модулем сповіщень.
3. **Модуль управління запасами**
    - **Функції:**
        - Облік наявних запчастин та матеріалів.
        - Відстеження залишків на складі.
        - Автоматизація створення замовлень для поповнення запасів.
    - **Взаємодія:**
        - Отримання даних про витрати запасів з модуля замовлень.
4. **Модуль фінансового управління**
    - **Функції:**
        - Ведення обліку доходів та витрат.
        - Інтеграція з платіжними системами для прийому оплат (Stripe, PayPal).
        - Генерація звітів про фінансові показники.
    - **Взаємодія:**
        - Інтеграція з модулем замовлень для обчислення доходів.
5. **Модуль сповіщень**
    - **Функції:**
        - Надсилання SMS та email-повідомлень клієнтам.
        - Сповіщення співробітників про нові замовлення або критичні запаси.
    - **Взаємодія:**
        - Отримання даних від модулів управління клієнтами, запасами та замовленнями.
6. **Модуль аналітики та звітності**
    - **Функції:**
        - Генерація графіків та таблиць для аналізу діяльності автосервісу.
        - Відстеження продуктивності співробітників та популярності послуг.
    - **Взаємодія:**
        - Отримання даних від усіх інших модулів для формування звітів.

Взаємодія між модулями:

1. **Через API:**
   - Кожен модуль взаємодіє з іншими через стандартизовані REST API або GraphQL запити.
2. **Загальна база даних:**
   - Дані зберігаються в централізованій базі даних, доступ до якої отримують усі модулі.
3. **Черга повідомлень:**
   - Для обробки подій у реальному часі використовується черга повідомлень (наприклад, RabbitMQ або Kafka).

3.3 Розробка функціональних компонентів

Функціональні компоненти веб-застосунку розробляються відповідно до визначених вимог кожного модуля. Вони забезпечують виконання основних операцій, взаємодію з користувачами, обробку даних та підтримку зовнішніх інтеграцій.

Основні функціональні компоненти:

1. **Користувацький інтерфейс (Frontend Components):**
   - **Компоненти інтерфейсу управління замовленнями:**
     - Форма створення замовлення.
     - Список замовлень із можливістю фільтрації та сортування.
     - Карта замовлення (інформація про клієнта, статус, послуги).
   - **Компоненти інтерфейсу клієнтів:**
     - Форма реєстрації нового клієнта.
     - Панель зі списком клієнтів із пошуком.
     - Сторінка з історією замовлень клієнта.
   - **Компоненти аналітики:**
     - Графіки доходів та витрат.
     - Візуалізація використання запасів.
     - Таблиці продуктивності співробітників.
2. **Бізнес-логіка (Backend Components):**
   - **Компонент обробки замовлень:**
     - Реалізація CRUD-операцій (створення, читання, оновлення, видалення).
     - Розрахунок вартості замовлення з урахуванням знижок або додаткових витрат.
   - **Компонент управління запасами:**
     - Логіка відстеження залишків та оновлення даних після виконання замовлення.
     - Генерація сповіщень про низький рівень запасів.
   - **Компонент авторизації:**
     - Логіка автентифікації користувачів через JWT.

    - Розмежування ролей (адміністратор, менеджер, механік).
3. **Інтеграційні компоненти:**
    - **Компонент для інтеграції з платіжними системами:**
        - Підтримка Stripe або PayPal для проведення оплат.
        - Збереження транзакцій у базі даних.
    - **Компонент сповіщень:**
        - Взаємодія з Twilio для відправлення SMS.
        - Інтеграція з поштовими сервісами для email-повідомлень.
4. **Компоненти роботи з базою даних:**
    - **Компонент клієнтських даних:**
        - Реалізація запитів для отримання інформації про клієнтів.
    - **Компонент фінансових даних:**
        - Збереження та оновлення інформації про платежі.
    - **Компонент звітності:**
        - Агрегація даних для побудови звітів.

3.4 Створення інтерфейсу користувача

Інтерфейс користувача є критично важливою складовою будь-якого веб-застосунку, оскільки саме через нього відбувається взаємодія користувача із системою. У даному проєкті для створення сучасного, зручного та інтуїтивного інтерфейсу використовується фреймворк React.
Принципи розробки інтерфейсу:

1. **Зручність використання (Usability):**
    - Мінімізація часу на виконання операцій.
    - Забезпечення інтуїтивної навігації.
2. **Адаптивність:**
    - Інтерфейс має коректно відображатися на екранах різного розміру (десктопи, планшети, мобільні пристрої).
3. **Естетичність:**
    - Використання сучасного дизайну з акцентом на мінімалізм.
    - Дотримання корпоративних кольорів та стилів.
4. **Функціональність:**
    - Інтерфейс має забезпечувати доступ до всіх основних функцій системи.

Основні елементи інтерфейсу:

1. **Головна панель (Dashboard):**
    - Інформаційна панель зі статусами замовлень, графіками доходів і витрат, попередженнями щодо запасів.
2. **Форма для створення замовлень:**
    - Поля для введення даних клієнта.
    - Додавання послуг до замовлення.
    - Розрахунок загальної вартості.
3. **Список клієнтів:**

- Пошук і фільтрація за критеріями (ім'я, контактні дані, кількість замовлень).
- Перехід до детальної інформації про клієнта.

4. **Сторінка управління запасами:**
   - Таблиця з переліком запасів та їхнім статусом.
   - Кнопка для створення нового замовлення на постачання.
5. **Сторінка аналітики:**
   - Графіки та таблиці для аналізу діяльності автосервісу.
   - Вибір періоду для аналізу (день, місяць, рік).

Використані технології та підходи:

1. **React:**
   - Компонентний підхід для легкості повторного використання коду.
   - Використання React Router для маршрутизації.
2. **CSS-фреймворки:**
   - **Tailwind CSS:**
     - Забезпечує швидке створення адаптивного дизайну.
     - Мінімізує кількість написаного коду стилів.
3. **UX/UI бібліотеки:**
   - Використання готових компонентів із бібліотек (наприклад, Material UI).
4. **Тестування інтерфейсу:**
   - Тестування інтерактивності та адаптивності за допомогою інструментів, таких як Storybook і Cypress.

3.5 Розробка бекенд-частини

Розробка бекенд-частини забезпечує логіку роботи веб-застосунку, управління даними, їхню обробку та взаємодію з базою даних. Для створення надійної та масштабованої серверної частини використовуються **Node.js** та фреймворк **Express.js**.

Ключові завдання бекенд-розробки:

1. Обробка запитів від клієнтської частини.
2. Виконання CRUD-операцій із базою даних (створення, читання, оновлення, видалення).
3. Реалізація бізнес-логіки:
   - Розрахунок вартості замовлень.
   - Управління запасами та оновлення їх статусу.
4. Забезпечення безпеки:
   - Захист даних користувачів.
   - Авторизація та автентифікація.

Архітектура бекенду:

1. **Рівень API:**

- Використання REST API для забезпечення комунікації між клієнтською та серверною частинами.
- Реалізація ендпоінтів для різних модулів (замовлення, клієнти, запаси, звіти).

2. **Рівень бізнес-логіки:**
   - Обробка запитів перед передачею до бази даних.
   - Валідація даних та забезпечення відповідності бізнес-процесам.
3. **Рівень доступу до даних:**
   - Використання ORM (Sequelize або Mongoose) для взаємодії з базою даних.
   - Оптимізація запитів для покращення продуктивності.

Основні функціональні модулі бекенду:

1. **Модуль управління замовленнями:**
   - Реалізація CRUD-операцій із замовленнями.
   - Встановлення статусів (нове, в роботі, завершено).
   - Підрахунок загальної вартості із врахуванням послуг та знижок.
2. **Модуль управління клієнтами:**
   - Збереження даних клієнтів (ім’я, контактна інформація, історія замовлень).
   - Взаємодія з аналітикою для визначення найактивніших клієнтів.
3. **Модуль управління запасами:**
   - Додавання, оновлення та видалення даних про запаси.
   - Генерація попереджень про необхідність поповнення запасів.
4. **Модуль авторизації:**
   - Реалізація автентифікації через JWT.
   - Використання шифрування паролів (bcrypt).
   - Розподіл прав доступу за ролями (адміністратор, менеджер, співробітник).

Інтеграція з базою даних:

1. **Обраний тип бази даних:**
   - **MongoDB** (гнучкість у роботі з JSON-подібними структурами).
   - **PostgreSQL** (реляційна модель із сильною підтримкою транзакцій).
2. **Організація даних:**
   - Створення відповідних моделей для таблиць (Orders, Clients, Inventory).
   - Оптимізація структури даних для швидкого доступу.
3. **Резервування даних:**
   - Налаштування автоматичного резервного копіювання бази даних.

Інструменти забезпечення безпеки:

1. **Авторизація через JWT:**
   - Генерація токенів для кожного авторизованого користувача.
   - Перевірка прав доступу для кожного запиту.
2. **Шифрування паролів:**
   - Використання bcrypt для захисту збережених паролів.

3. **Захист API:**
    - Використання шоломів безпеки (Helmet.js).
    - Обмеження кількості запитів (rate limiting) для уникнення DDOS-атак.

Процес розробки бекенду:

1. **Створення серверу:**
    - Налаштування Express.js для обробки запитів.
    - Визначення основних маршрутів.
2. **Реалізація бізнес-логіки:**
    - Написання контролерів для обробки даних.
    - Реалізація взаємодії з базою даних.
3. **Тестування:**
    - Юніт-тестування окремих модулів (наприклад, модуль обробки замовлень).
    - Інтеграційне тестування роботи API.
4. **Деплой:**
    - Розгортання на хмарному сервері (наприклад, Heroku, AWS).
    - Налаштування CI/CD для автоматизації оновлень.

3.6 Інтеграція фронтенду та бекенду

Інтеграція фронтенду та бекенду є ключовим етапом у розробці веб-застосунку, що забезпечує ефективну взаємодію між інтерфейсом користувача та серверною частиною. Для цього реалізується спільна робота компонентів через REST API.

Цілі інтеграції:

1. Забезпечення обміну даними між клієнтською частиною (React) та серверною частиною (Node.js, Express).
2. Реалізація динамічного оновлення інтерфейсу залежно від даних із сервера.
3. Підтримка безпеки даних при передачі між клієнтом і сервером.

Ключові етапи інтеграції:

1. **Підключення до API:**
    - Використання бібліотеки **Axios** або вбудованого **Fetch API** для виконання HTTP-запитів.
    - Конфігурація основного URL API (наприклад, http://localhost:5000/api).
2. **Обробка запитів:**
    - **GET-запити** для отримання даних (наприклад, список замовлень, клієнтів, стан запасів).
    - **POST-запити** для створення нових об'єктів (нове замовлення чи клієнт).
    - **PUT/DELETE-запити** для оновлення або видалення даних.
3. **Валідація та обробка помилок:**
    - Сервер повертає статуси HTTP (наприклад, 200, 400, 404, 500) для ідентифікації успіху або помилки запиту.
    - Реалізація обробки помилок на клієнтській стороні (наприклад, повідомлення про помилку авторизації).

4. **Реалізація авторизації:**
   - Використання JWT (JSON Web Tokens) для забезпечення захищеної комунікації між клієнтом і сервером.
   - Зберігання токена у **localStorage** або **cookies** із дотриманням рекомендацій безпеки.
5. **Динамічне оновлення даних у фронтенді:**
   - Реалізація оновлення компонентів React після отримання або зміни даних на сервері

Приклад інтеграції:

1. **Отримання списку клієнтів:**
   javascript
   ```
   // API-запит на сервер
   import axios from 'axios';

   const fetchClients = async () => {
     try {
       const response = await axios.get('http://localhost:5000/api/clients');
       console.log(response.data); // Виведення списку клієнтів
     } catch (error) {
       console.error('Error fetching clients:', error);
     }
   };
   ```
2. **Передача даних нового клієнта:**
   javascript
   ```
   const addClient = async (clientData) => {
     try {
       const response = await axios.post('http://localhost:5000/api/clients', clientData);
       console.log('Client added:', response.data);
     } catch (error) {
       console.error('Error adding client:', error);
     }
   };
   ```

Інструменти для інтеграції:

1. **Бібліотеки для роботи з API:**
   - **Axios:** Простота у використанні та підтримка промісів.
   - **React Query:** Полегшує управління станом даних, отриманих із сервера.
2. **Інструменти для перевірки API:**
   - **Postman** або **Insomnia** для тестування ендпоінтів серверної частини перед інтеграцією.
3. **Налагодження:**
   - **Redux DevTools** для моніторингу стану програми.
   - **Network Tab** у браузері для перевірки виконання запитів.

Особливості інтеграції:

- Використання **CORS (Cross-Origin Resource Sharing):** Налаштування серверу для роботи із запитами з іншого домену.
- Встановлення тайм-аутів для запитів для уникнення зависання інтерфейсу при тривалих операціях.
- Логування на серверній стороні для відстеження помилок інтеграції.

3.7 Управління клієнтами, послугами, замовленнями та запасами

Цей модуль веб-застосунку забезпечує централізоване управління основними процесами автосервісу, включаючи роботу з клієнтами, контроль послуг, облік замовлень та моніторинг запасів.
Ключові компоненти модуля:

1. **Управління клієнтами:**
    - Збереження та оновлення даних клієнтів:
        - Контактна інформація (ім’я, номер телефону, email).
        - Історія попередніх замовлень.
    - Пошук клієнтів за іменем, номером телефону або іншими параметрами.
    - Аналітика:
        - Визначення найактивніших клієнтів.
        - Сегментація за частотою звернень.
2. **Управління послугами:**
    - Додавання та редагування послуг:
        - Назва послуги, опис, вартість.
    - Групування послуг за категоріями (наприклад, ремонт двигуна, заміна шин).
    - Аналітика доходів за кожною послугою.
3. **Управління замовленнями:**
    - Реєстрація нових замовлень:
        - Прив’язка до клієнта.
        - Вибір послуг із доступного списку.
    - Відстеження статусу замовлень (нове, в роботі, завершено).
    - Генерація рахунків для клієнтів.
4. **Управління запасами:**
    - Додавання та редагування елементів запасів:
        - Назва, кількість, мінімальний рівень.
    - Контроль запасів:
        - Автоматичні нагадування про необхідність поповнення.
        - Відображення актуального стану складу.
    - Взаємодія із замовленнями для списання запасів.

Основний функціонал:

1. **CRUD-операції:**

- Виконання операцій створення, читання, оновлення та видалення для всіх елементів модуля.

2. **Фільтри та сортування:**
   - Сортування за статусом замовлень, датою створення чи вартістю.
   - Фільтри для швидкого пошуку клієнтів або послуг.
3. **Звіти та аналітика:**
   - Звіти про активність клієнтів, частоту використання послуг, стан запасів.
   - Графічне представлення даних (наприклад, діаграми доходів за місяць).
4. **Автоматизація:**
   - Автоматичний розрахунок загальної вартості замовлення із врахуванням знижок чи акцій.
   - Списання товарів зі складу під час виконання замовлення.

Приклад взаємодії з модулем:

1. **Додавання нового клієнта:**
   - Форма із введенням імені, контактів та історії попередніх замовлень.
2. **Оновлення статусу замовлення:**
   - Кнопка в інтерфейсі для зміни статусу замовлення (наприклад, із "в роботі" на "завершено").
3. **Попередження про низький рівень запасів:**
   - Автоматичне повідомлення, якщо кількість запасів нижче встановленого мінімуму.

Інтеграція з іншими модулями:

- Модуль клієнтів взаємодіє із замовленнями для відображення персоналізованої історії обслуговування.
- Управління запасами пов'язане з замовленнями для автоматичного списання використаних матеріалів.
- Дані про послуги інтегруються із фінансовим модулем для формування звітів про доходи.

3.8 Методи захисту даних

1. Захист переданих даних

1. **SSL/TLS:**
   - Використання HTTPS для шифрування даних, які передаються між клієнтом і сервером.
   - Гарантує, що дані не будуть перехоплені або модифіковані третіми особами.
2. **JSON Web Tokens (JWT):**
   - Токени для автентифікації та авторизації.
   - Токени підписуються сервером, щоб уникнути їхнього підроблення.
3. **CORS (Cross-Origin Resource Sharing):**
   - Контроль доступу до API з різних доменів.

- Встановлення правил доступу для захисту від небажаних запитів.

2. Захист даних на сервері
   1. **Хешування паролів:**
      - Використання алгоритмів, таких як **bcrypt** чи **Argon2**, для хешування паролів перед їхнім збереженням у базі даних.
   2. **Контроль доступу:**
      - Ролі користувачів (адміністратор, співробітник).
      - Визначення рівнів доступу до даних і функцій системи.
   3. **Регулярне оновлення та патчинг:**
      - Оновлення серверного ПЗ та бібліотек для усунення вразливостей.
3. Захист від зовнішніх атак
   1. **Валідація даних:**
      - Перевірка всіх даних, які вводяться користувачами, для уникнення SQL-ін’єкцій чи XSS (міжсайтових сценаріїв).
      - Використання ORM, таких як Sequelize, для мінімізації ризику SQL-ін’єкцій.
   2. **Обмеження частоти запитів (Rate Limiting):**
      - Захист від атак типу **Brute Force** або **DDoS**.
      - Використання бібліотек, як-от **express-rate-limit**.
   3. **Міжсайтовий захист (CSRF):**
      - Використання CSRF-токенів для захисту від несанкціонованих запитів.
4. Збереження та резервування даних
   1. **Шифрування бази даних:**
      - Шифрування конфіденційних даних (наприклад, номерів телефонів, фінансових деталей) перед збереженням.
   2. **Резервне копіювання:**
      - Регулярне створення резервних копій даних.
      - Зберігання резервних копій у захищеному місці, окремо від основної системи.
   3. **Журналювання:**
      - Ведення логів доступу до критичних даних.
      - Моніторинг дій користувачів для виявлення підозрілих активностей.

Інструменти для забезпечення безпеки:

- **Helmet.js:** Захист HTTP-заголовків у Node.js.
- **OWASP ZAP:** Інструмент для перевірки безпеки веб-додатків.
- **DataDog, Splunk:** Моніторинг і журналювання.

Результати застосування методів безпеки:

1. Захищена передача даних між клієнтом і сервером.
2. Зменшення ризику несанкціонованого доступу до системи.
3. Гарантія цілісності та конфіденційності інформації.
4. Запобігання зовнішнім атакам та забезпечення стабільності роботи системи.

3.9 Автентифікація та авторизація користувачів

1. **Ролі користувачів:**
    - **Адміністратор:** Має доступ до всіх модулів, включно з управлінням користувачами та системними налаштуваннями.
    - **Співробітник:** Має доступ до модулів клієнтів, замовлень і послуг.
    - **Клієнт:** Обмежений доступ, наприклад, лише до перегляду своїх замовлень.
2. **Контроль доступу:**
    - Використання **middleware** для перевірки ролі користувача перед виконанням запиту.
    - Кожен запит перевіряється на наявність токена і відповідності прав доступу.

Технічна реалізація:

1. **Middleware для перевірки токена:**
    - Код перевіряє валідність токена JWT у кожному запиті.
    - Якщо токен недійсний або відсутній, запит відхиляється.
2. **Розмежування прав доступу:**
    - Користувачі з різними ролями мають доступ лише до дозволених їм маршрутів і функцій.
3. **Оновлення токенів:**
    - Використання **refresh-токенів** для подовження сеансу користувача без повторного входу.

Інструменти:

- **Passport.js:** Бібліотека для автентифікації в Node.js.
- **jsonwebtoken:** Генерація та валідація JWT-токенів.
- **bcrypt.js:** Хешування паролів для безпечного зберігання.

Результати:

1. Забезпечення безпечного доступу до системи.
2. Чітке розмежування прав доступу між різними типами користувачів.
3. Захист від несанкціонованого використання ресурсів.

3.10 Юніт-тестування та інтеграційне тестування

1. Юніт-тестування

Мета:

Перевірка роботи окремих модулів чи функцій у системі ізольовано від інших компонентів.

Підходи до реалізації:

1. **Тестування фронтенду:**
    - Перевірка функцій, компонентів та UI-елементів (наприклад, коректність обробки подій або рендерингу).

- Використання бібліотек:
    - **Jest**: тестування логіки та компонентів.
    - **React Testing Library**: інтерактивне тестування React-компонентів.

2. **Тестування бекенду:**
    - Перевірка окремих API-ендпоінтів, логіки бізнес-процесів і обробки запитів.
    - Використання бібліотек:
        - **Mocha** або **Jest** для тестування функцій.
        - **Chai** або **Supertest** для перевірки HTTP-запитів.

Приклад юніт-тесту:

- Тестування функції для обчислення вартості послуг з урахуванням знижок.
- Тест перевіряє коректність результату для різних варіантів вхідних даних.

2. Інтеграційне тестування

Мета:

Перевірка взаємодії між різними модулями системи для забезпечення їх коректної спільної роботи.

Підходи до реалізації:

1. **Тестування API:**
    - Перевірка, чи правильно фронтенд взаємодіє з бекендом.
    - Автоматичне тестування запитів і відповідей API.
2. **Перевірка взаємодії модулів:**
    - Тестування взаємодії між модулями управління замовленнями, клієнтами та запасами.
    - Перевірка, чи оновлюються дані в базі після виконання певних дій.
3. **Інструменти:**
    - **Postman/Newman**: ручне та автоматизоване тестування API.
    - **Cypress**: для тестування інтеграції та інтерфейсу.

Приклад інтеграційного тесту:

- Перевірка створення нового замовлення:
    1. Клієнт відправляє запит на створення замовлення через інтерфейс.
    2. Сервер приймає запит, записує дані в базу.
    3. Система повертає відповідь із підтвердженням.

Етапи тестування в системі:

1. **Планування тестування:**
    - Визначення основних сценаріїв і функціоналу, що потребують перевірки.
2. **Написання тестів:**
    - Створення тестів для кожного модуля (юніт-тестування).
    - Розробка сценаріїв взаємодії між модулями (інтеграційне тестування).
3. **Виконання тестів:**
    - Використання тестових середовищ із реальними даними.
4. **Аналіз результатів:**

- Виправлення помилок і повторне тестування.

Результати:

1. Гарантія стабільної роботи кожного компонента.
2. Стабільність взаємодії між компонентами.
   Мінімізація ризику збоїв у роботі веб-застосунку.

3.11 Оптимізація продуктивності

1. Оптимізація фронтенду
   1. **Мінімізація ресурсів:**
      - Стиснення файлів JavaScript, CSS і HTML за допомогою таких інструментів, як **Terser** або **UglifyJS**.
      - Використання **webpack** для бандлінгу та мініфікації ресурсів.
   2. **Асинхронне завантаження:**
      - Застосування **lazy loading** для зображень і модулів.
      - Використання CDN для доставки статичних ресурсів.
   3. **Кешування:**
      - Налаштування браузерного кешу для часто використовуваних ресурсів.
   4. **Оптимізація роботи компонентів:**
      - Зменшення кількості рендерів у React-компонентах через **React.memo** або **shouldComponentUpdate**.
2. Оптимізація бекенду
   1. **Ефективна робота бази даних:**
      - Використання індексів у базі даних для прискорення пошуку.
      - Оптимізація SQL-запитів і використання ORM-інструментів, таких як **Sequelize**.
   2. **Кешування:**
      - Використання кешування запитів за допомогою **Redis** або **Memcached**.
      - Кешування відповідей API для зменшення кількості запитів до бази даних.
   3. **Збалансоване навантаження:**
      - Налаштування **load balancers** для рівномірного розподілу запитів між серверами.
   4. **Обробка запитів:**
      - Використання асинхронних операцій у Node.js для зменшення блокувань.
      - Оптимізація API-ендпоінтів через **rate limiting** та обробку великих запитів на фоні.
3. Тестування продуктивності
   1. **Стрес-тестування:**
      - Імітація великої кількості запитів одночасно для оцінки стійкості системи.
      - Інструменти: **Apache JMeter**, **K6**.
   2. **Тестування завантаження:**

- Вимірювання часу відповіді на запити при середньому навантаженні.
- Використання таких інструментів, як **Gatling** або **Locust**.

3. **Моніторинг у реальному часі:**
   - Відстеження продуктивності за допомогою **New Relic**, **Datadog** або **Elastic APM**.

4. Результати оптимізації

1. **Зменшення часу завантаження сторінок:**
   - Покращує досвід користувача.
2. **Ефективніше використання серверних ресурсів:**
   - Зниження витрат на хостинг і підтримку.
3. **Стабільна робота системи при великій кількості користувачів.**

3.12 Візуальне представлення розроблених матеріалів

Зображення відображає головну сторінку веб-сайту автосервісу "Engine-Service". Дизайн сторінки на рисунку 3.1 спроектований так, щоб привернути увагу потенційних клієнтів та заохотити їх скористатися послугами сервісу.

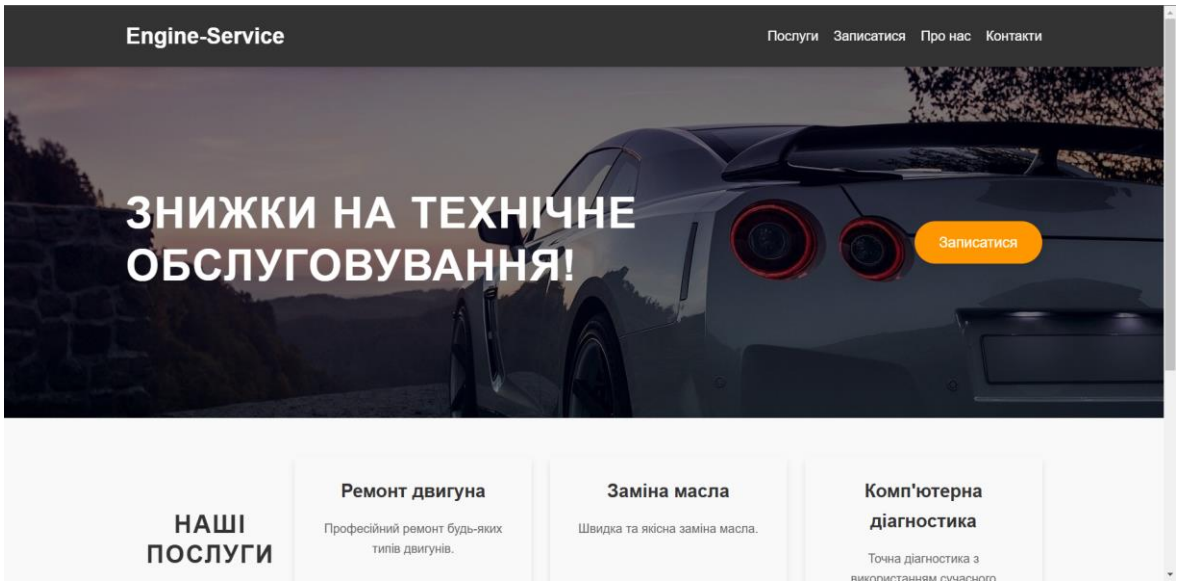

Рисунок 3.1 – Головна сторінка

Ключові елементи дизайну та їхнє значення:

- **Заголовок "Знижки на технічне обслуговування!":** Яскравий і помітний заголовок відразу привертає увагу відвідувача та повідомляє про основну пропозицію сервісу.
- **Зображення спортивного автомобіля:** Використання зображення потужного автомобіля створює асоціації з якістю, швидкістю та надійністю, що позитивно впливає на сприйняття сервісу.
- **Кнопка "Записатися":** Яскрава і помітна кнопка, розташована в зручному для клікання місці, закликає відвідувача зробити наступний крок і записатися на обслуговування.

- **Блоки з описом послуг:** Чітко структуровані блоки з коротким описом основних послуг сервісу (ремонт двигуна, заміна масла, комп'ютерна діагностика) допомагають клієнту швидко ознайомитися з пропозицією.
- **Колірна гама:** Переважання темних тонів створює контраст з яскравими елементами (заголовок, кнопка) і робить текст більш читабельним.

Це зображення на рисунку 3.2 є фрагментом веб-сторінки автосервісу "Engine-Service". Воно відображає розділ з інформацією про послуги та контактними даними.

Основні елементи, які можна побачити на зображенні:

- Заголовок "НАШІ ПОСЛУГИ": Цей заголовок виділяє розділ, де представлені основні послуги, які надає автосервіс.
- Картки з послугами: Під заголовком розташовані три картки, кожна з яких описує одну послугу:
  - Ремонт двигуна: Коротко описано послугу з професійного ремонту різних типів двигунів.
  - Заміна масла: Описана послуга зі швидкої та якісної заміни масла.
  - Комп'ютерна діагностика: Зазначено, що проводиться точна діагностика з використанням сучасного обладнання.
- Розділ "КОНТАКТИ": В цьому розділі вказані контактні дані автосервісу: телефонний номер, електронна пошта та адреса.

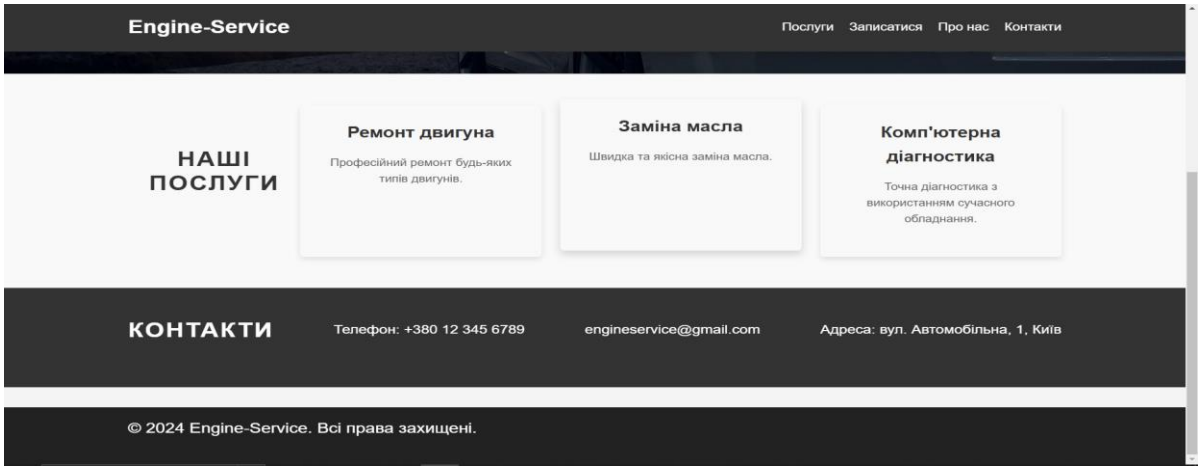

Рисунок 3.2 – Головна сторінка продовження

Наведена форма запису на риснку 3.3 на послугу, має простий і інтуїтивно зрозумілий дизайн. Вона містить мінімально необхідні поля для збору інформації від користувача і дозволяє швидко оформити замовлення.

Основні елементи форми:

- Заголовок: "Записатися на послугу" - чітко вказує на призначення форми.
- Вибір послуги: Випадаючий список дозволяє користувачу вибрати необхідну послугу з запропонованих варіантів.
- Поля для введення даних:
  - Ім'я: Поле для введення імені користувача.
  - Телефон: Поле для введення номера телефону для зв'язку.
  - Дата: Календар для вибору бажаної дати запису на послугу.

- Кнопка "Записатися": Кнопка для відправки заповненої форми.

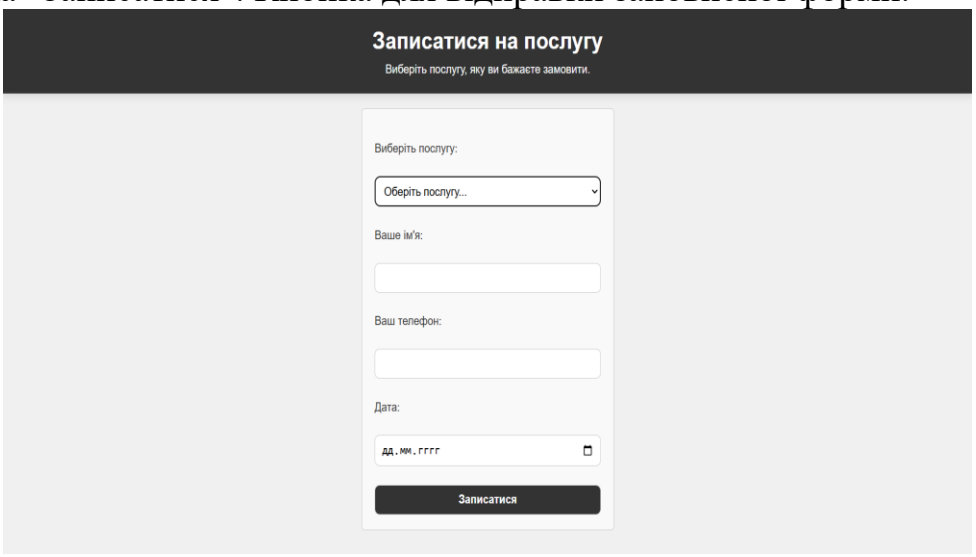

Рисунок 3.3 – Інтерфейс запису на послуги

На риснку 3.4 зображено розкриття функціоналу полів.

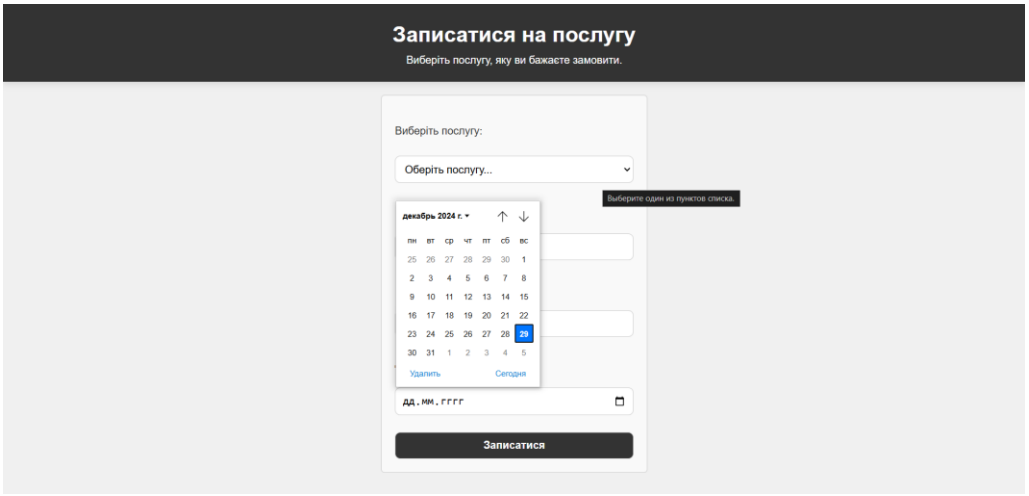

Рисунок 3.4 – Інтерфейс записатися на послугу розкриття функціоналу

Сторінка «Наші Послуги» на рисунку 3.5, має простий та інтуїтивно зрозумілий дизайн, що дозволяє користувачам швидко знайти необхідну інформацію. Вона сфокусована на наданні основних відомостей про послуги, які пропонує автосервіс.

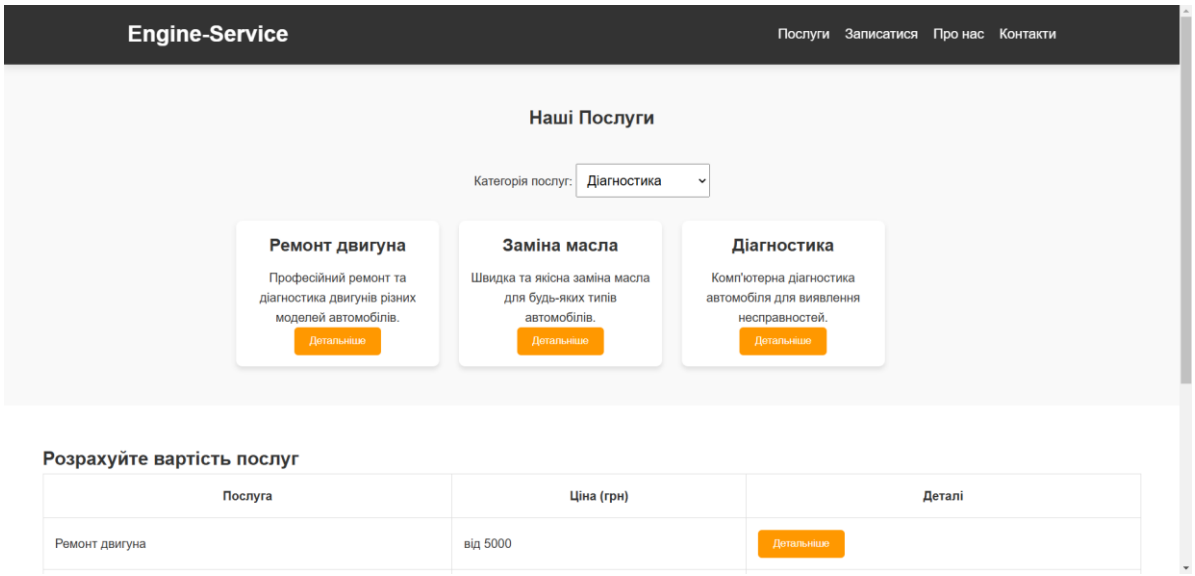

Рисунок 3.5 – Інтерфейс сторінки Наші Послуги

На рисунку 3.6 зображено продовження сторінки «Наші Послуги», з розкриттям секції, де клієнт може розрахувати приблизну вартість ремонту.

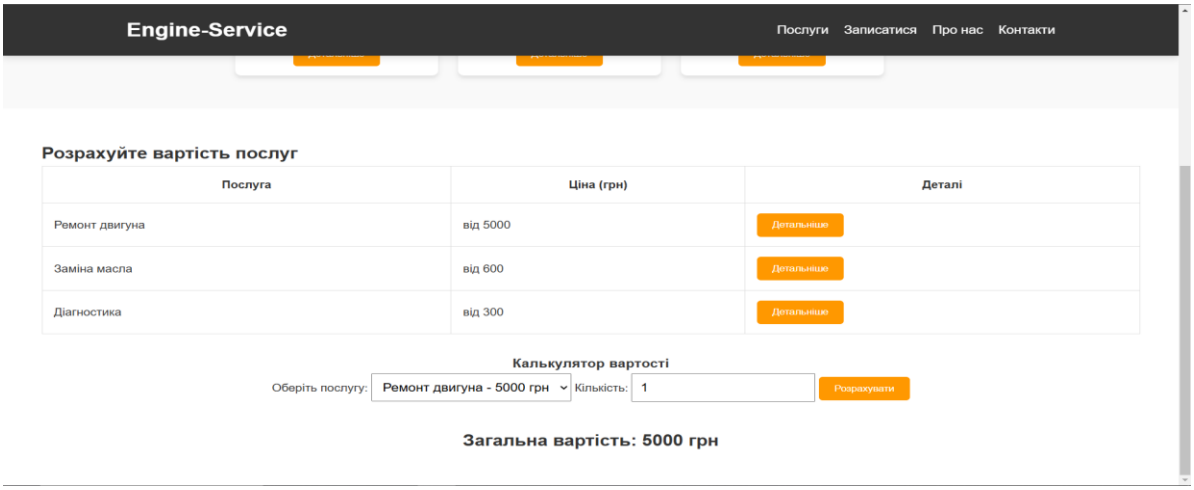

Рисунок 3.6 – Інтерфейс сторінки Наші Послуги продовження

Розділ "Про нас" на веб-сайті автосервісу на рисунку 3.7 спроектований для того, щоб представити компанію та її команду потенційним клієнтам. Він містить базову інформацію про автосервіс та його співробітників, а також виокремлює основні переваги компанії.

Структура та елементи:

- Заголовок: "Про нас" - чітко вказує на тему розділу.
- Блоки інформації: Розділ розбитий на три основні блоки:
  - Хто ми: Короткий опис компанії та її спеціалізації.
  - Наша команда: Представлення ключових співробітників з вказанням їхніх посад та досвіду роботи.
  - Наші переваги: Перелік основних конкурентних переваг компанії.

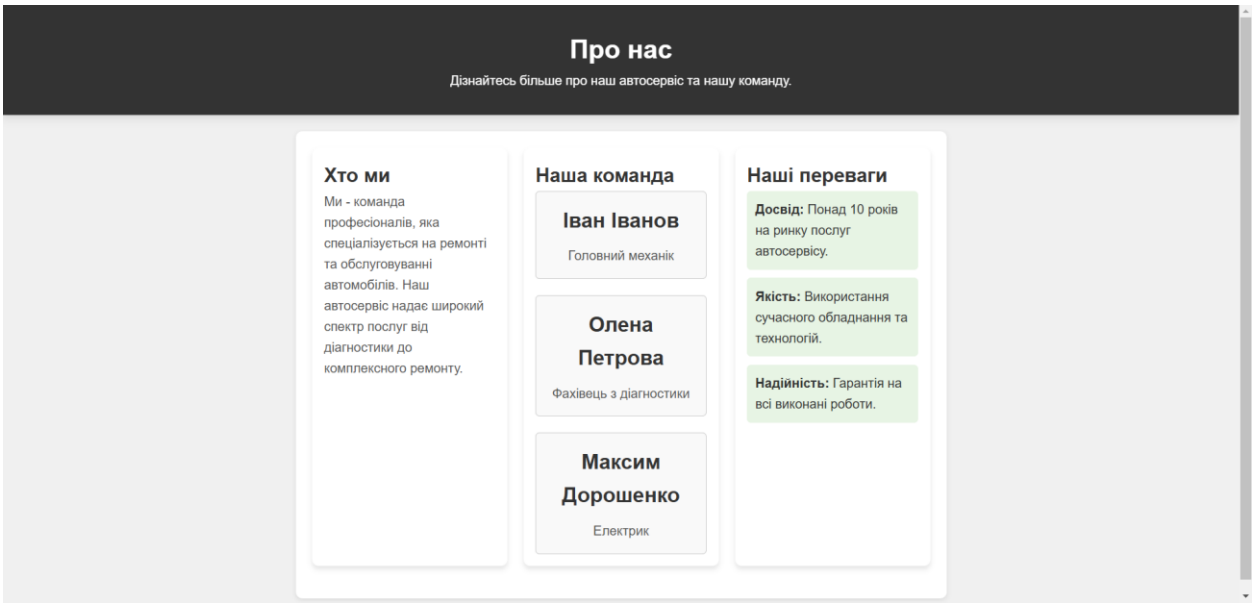

Рисунок 3.7 – Інтерфейс сторінки «Про нас»

Розділ "Контакти" на веб-сайті автосервісу зображений на рисунку 3.8 спроектований для того, щоб надати користувачам всю необхідну інформацію для зв'язку з компанією. Він містить базові контактні дані та карту місцезнаходження автосервісу.

Структура та елементи:

- Заголовок: "Контакти" - чітко вказує на тему розділу.
- Підзаголовок: "Зв'яжіться з нами для отримання додаткової інформації або запису на послугу" - закликає користувача до дії.
- Блоки інформації: Розділ розбитий на три основні блоки:
  - Наші контакти: Містить детальну контактну інформацію (адреса, телефон, email).
  - Місцезнаходження: Представлена карта з позначенням розташування автосервісу.
  - Зв'язатися з нами: Може містити додаткові форми зв'язку або кнопки для переходу на сторінки соціальних мереж.

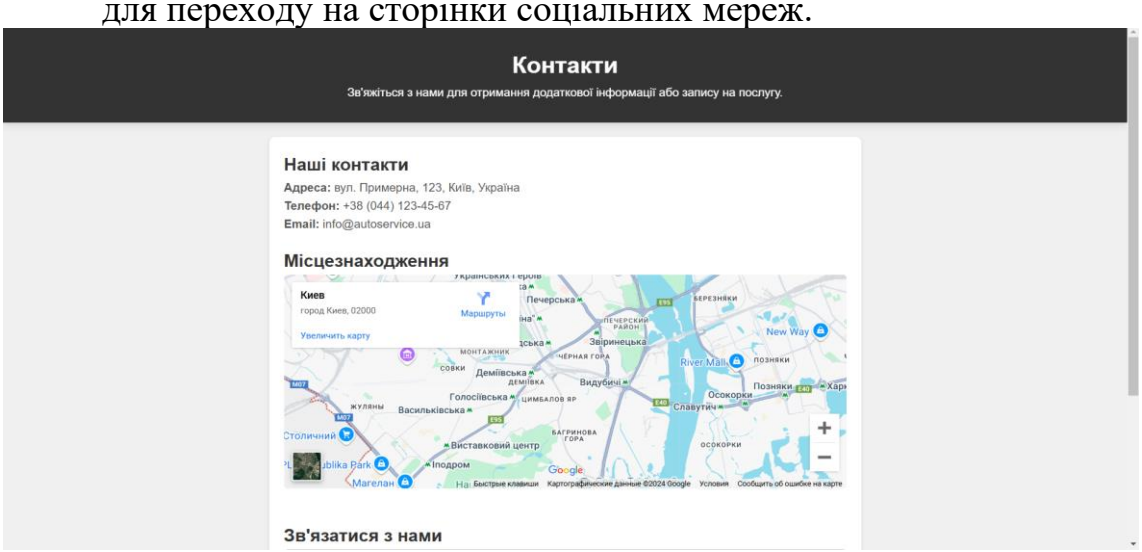

Рисунок 3.8 – Інтерфейс сторінки «Контакти»

На риснку 3.9 зображено продовжженя сторінки «Контакти» з блоками для зв’язку з клієнтами.

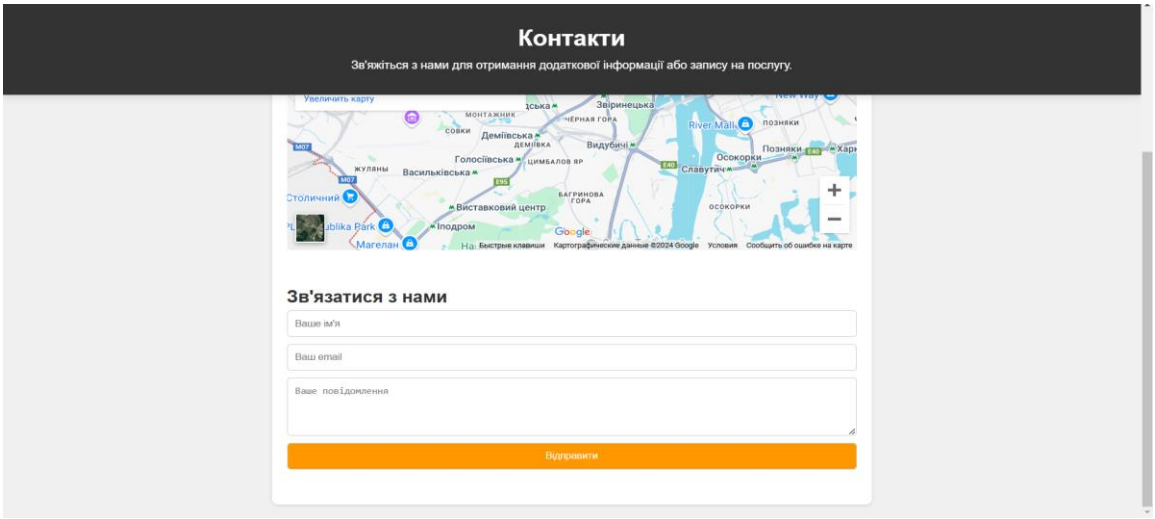

Рисунок 3.9 - Інтерфейс сторінки «Контакти» продовження

На рисунку 3.10 зображено планшетна версія головної сторінки.

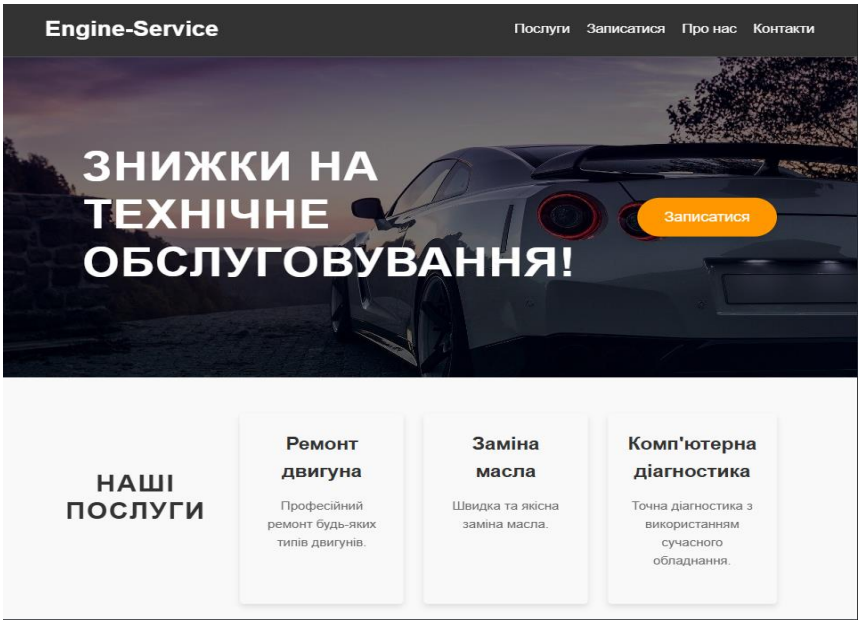

Рисунок 3.10 – Інтерфейс головної сторінки планшетної версії

На рисунку 3.11 зображено планшетна версія головної сторінки продовження.

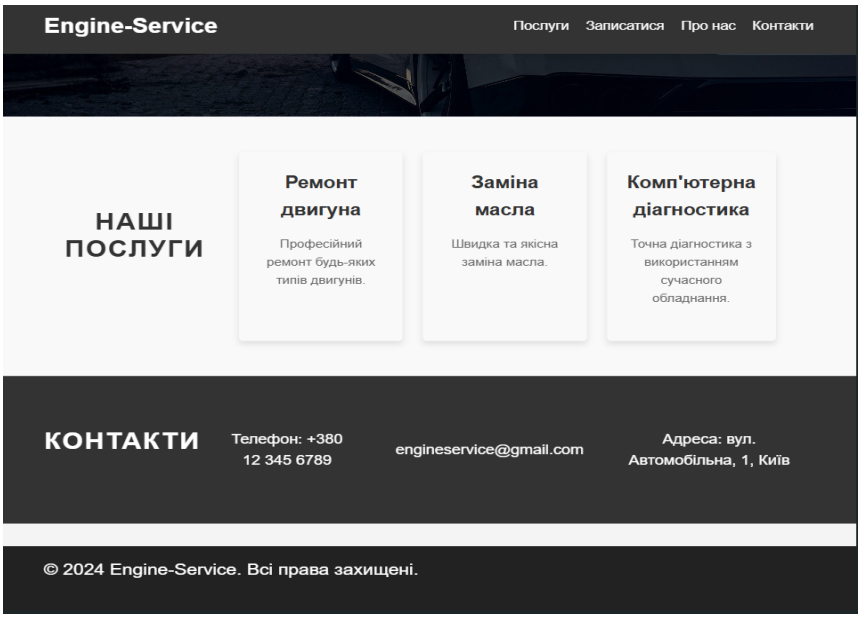

Рисунок 3.11 - Інтерфейс головної сторінки планшетної версії продовження

На рисунку 3.12 зображено планшетну версію сторінки «Записатися на послугу»

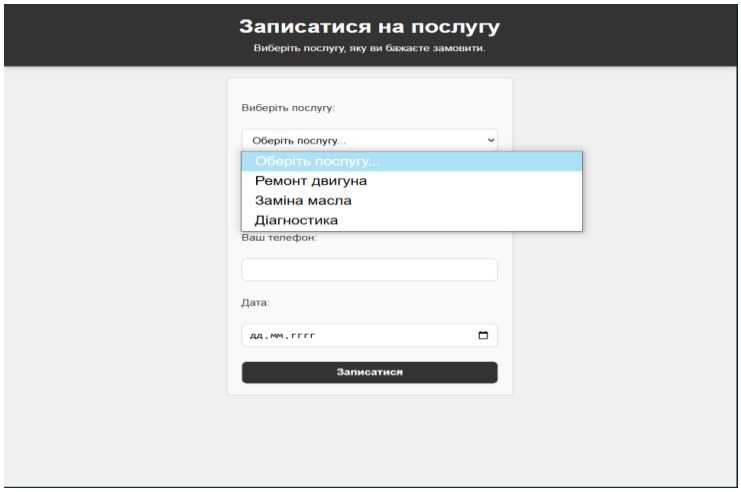

Рисунок 3.12 – Інтерфейс сторінки Записатися на послугу в планшетній версії

На рисунку 3.13 зображено планшетну версію сторінки «Про нас».

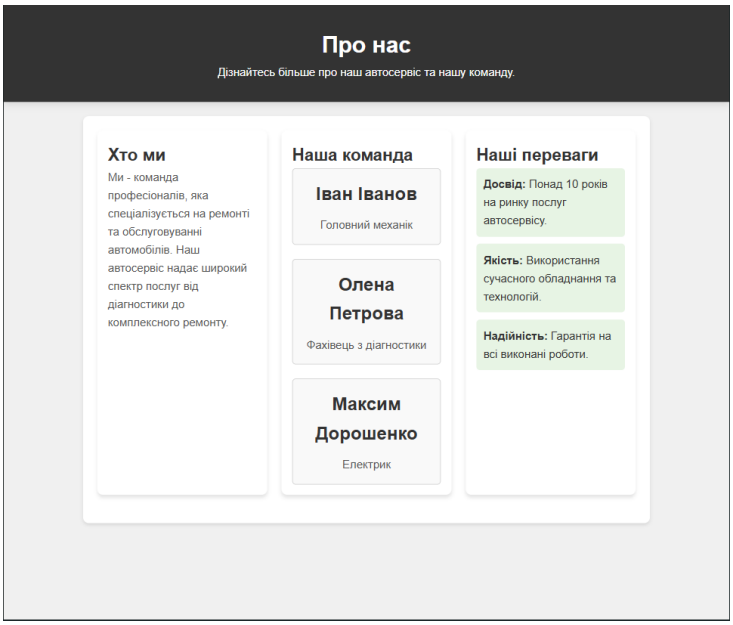

Рисунок 3.13 - Інтерфейс сторінки Про нас в планшетній версії

На рисунку 3.14 зображено планшетна версія сторінки «Контакти».

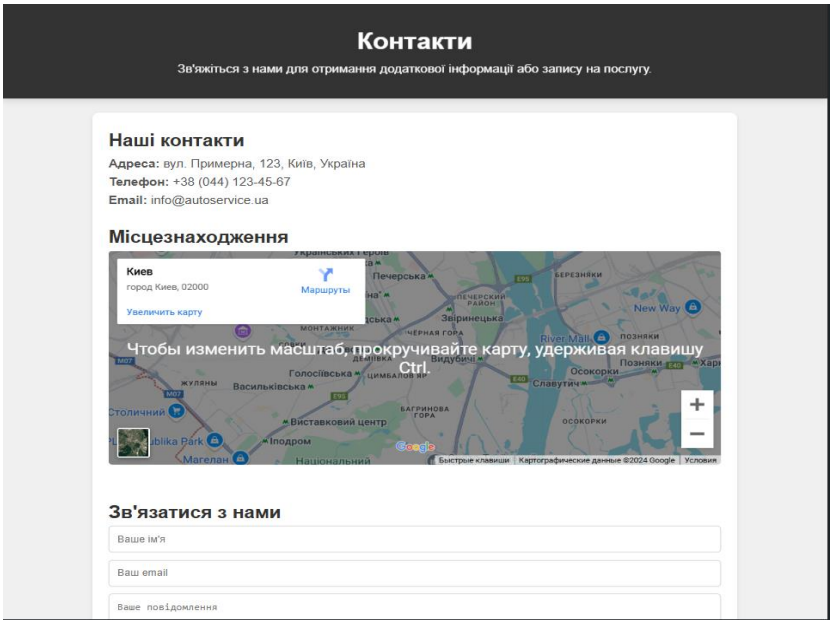

Рисунок 3.14 – Інтерфейс планшетної версії сторінки Контакти

На рисунку 3.15 зображено продовження сторінки планшетної версії сторінки «Контакти».

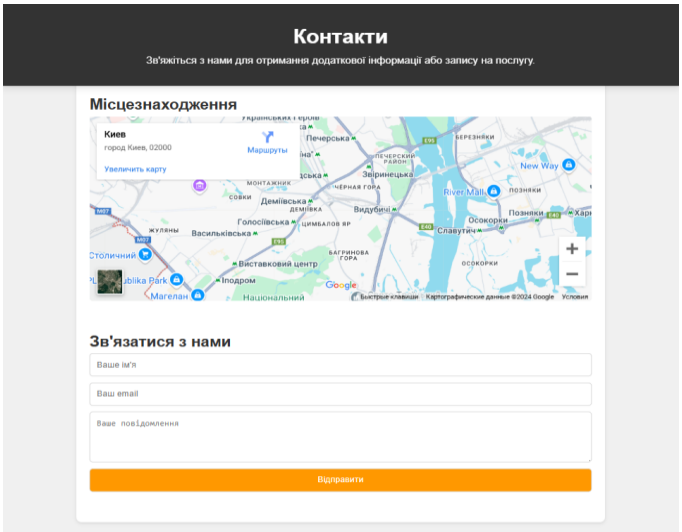

Рисунок 3.15 - Інтерфейс планшетної версії сторінки Контакти продовження

На рисунку 3.16 зображено інтерфейс сторінки планшетної версії сторінки «Наші Послуги».

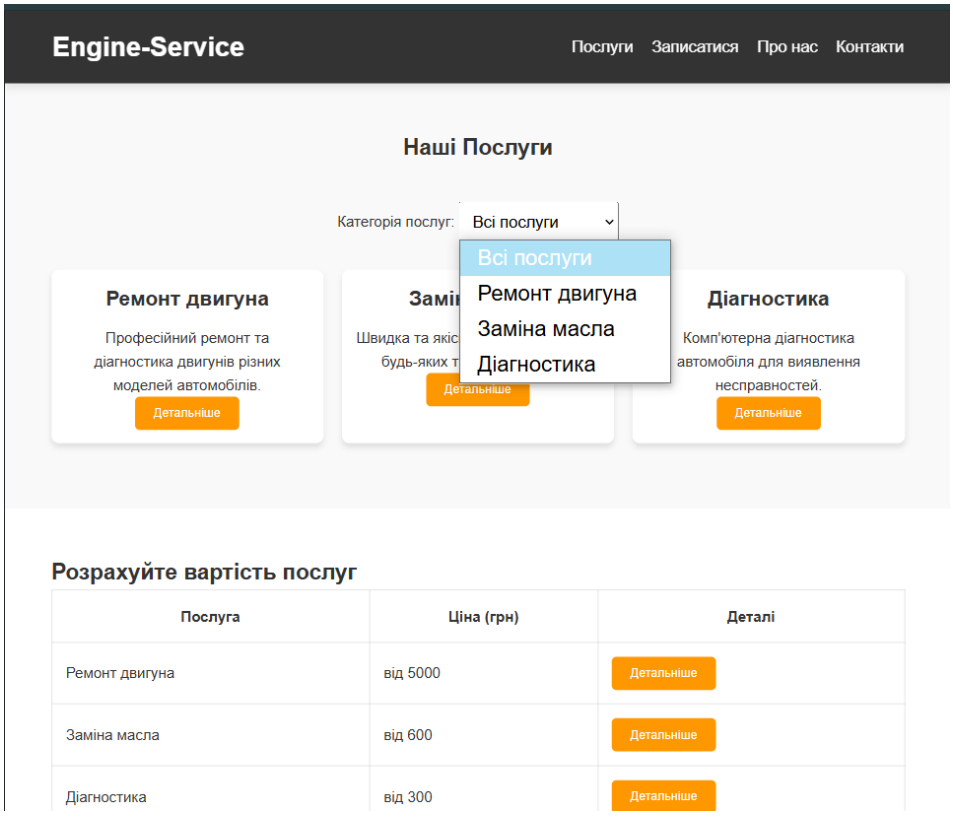

Рисунок 3.16 - Інтерфейс планшетної версії сторінки Наші послуги

На рисунку 3.17 зображено інтерфейс сторінки мобільної версії головної сторінки.

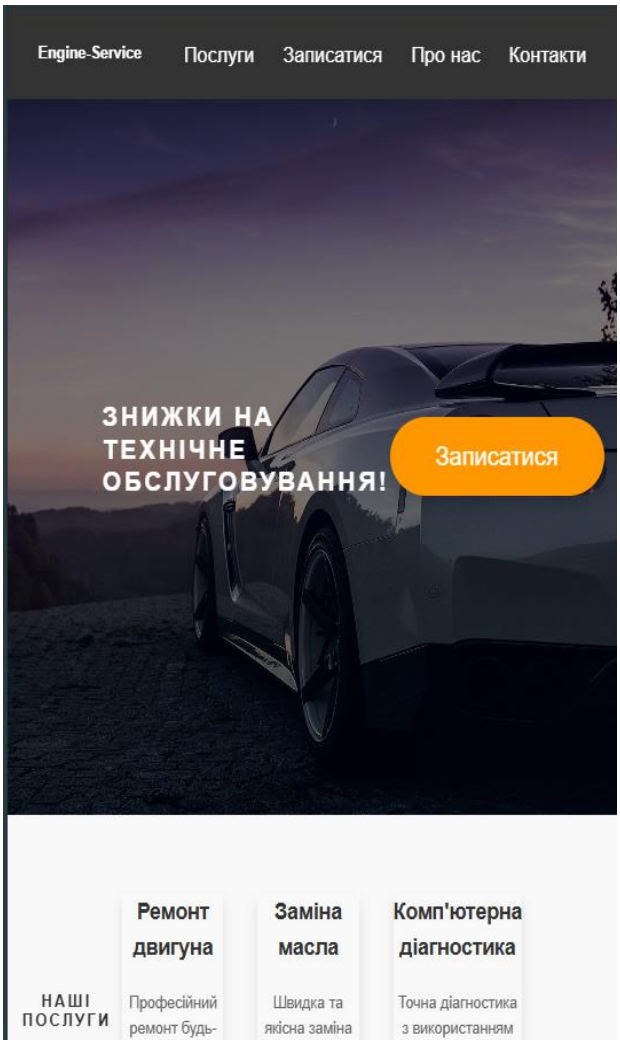

Рисунок 3.17 – Інтерфейс мобільної версії головної сторінки

На рисунку 3.18 зображено продовження інтерфейсу сторінки мобільної версії головної сторінки.

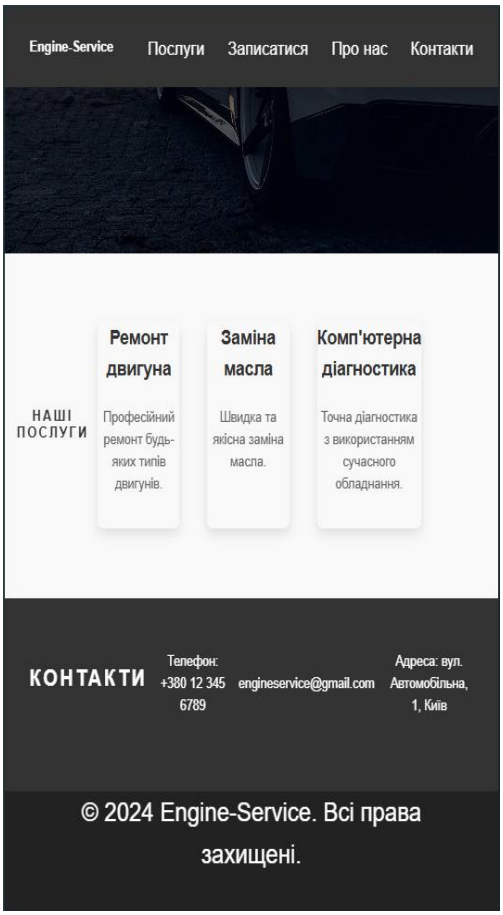

Рисунок 3.18 – Інтерфейс мобільної версії головної сторінки продовження

На рисунку 3.19 зображено інтерфейс сторінки мобільної версії «Записатися на послугу».

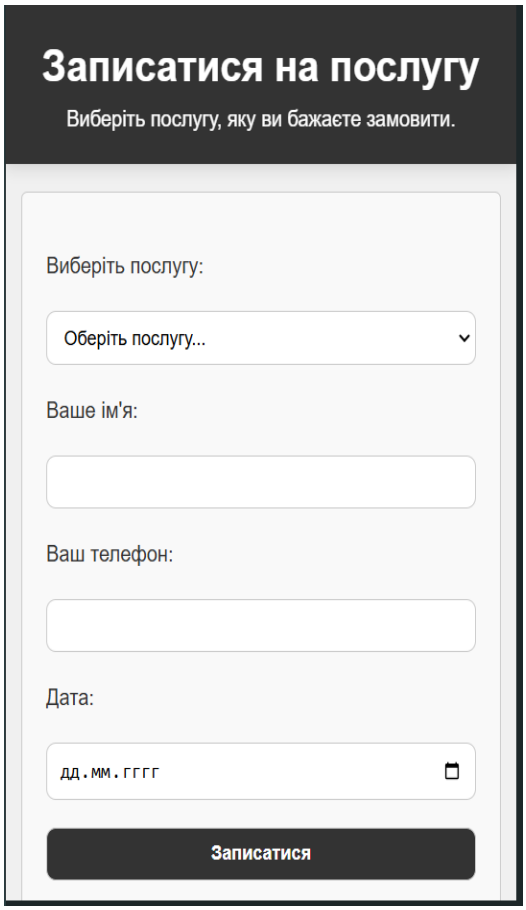

Рисунок 3.19 – Інтерфейс мобільної версії сторінки Записатися на послугу

На рисунку 3.20 зображено інтерфейс сторінки мобільної версії «Наші послуги».

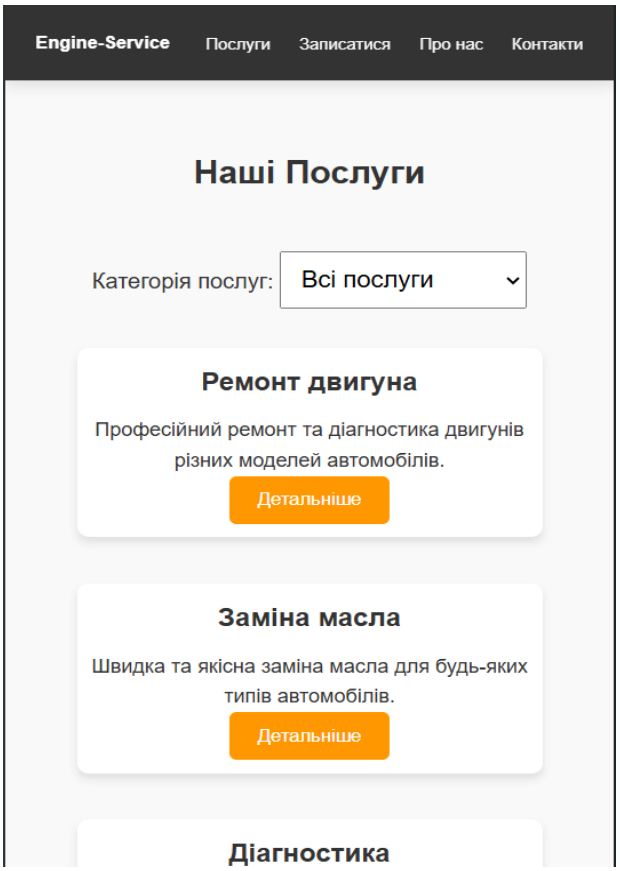

Рисунок 3.20 - Інтерфейс мобільної версії сторінки Наші послуги

На рисунку 3.21 зображено продовження інтерфейсу сторінки мобільної версії «Наші послуги».

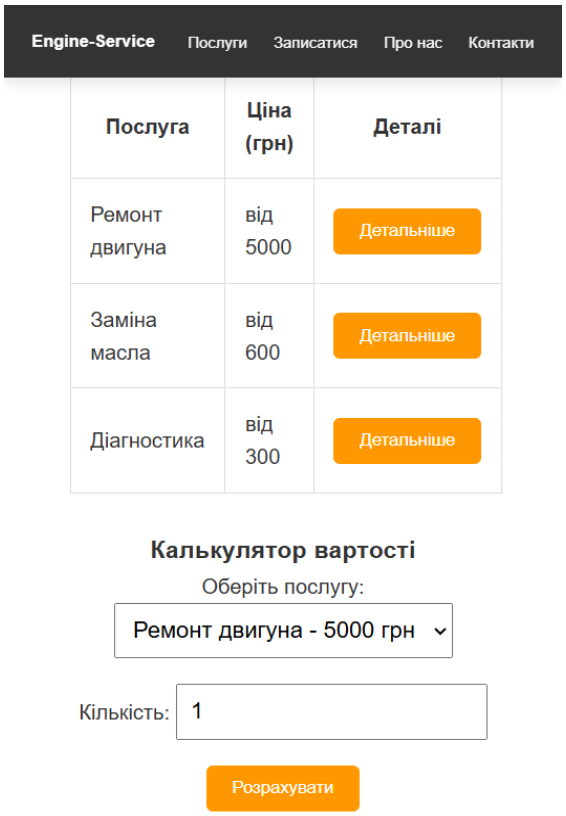

Рисунок 3.21 - Інтерфейс мобільної версії сторінки Наші послуги

На рисунку 3.22 зображено інтерфейс сторінки мобільної версії «Про нас».

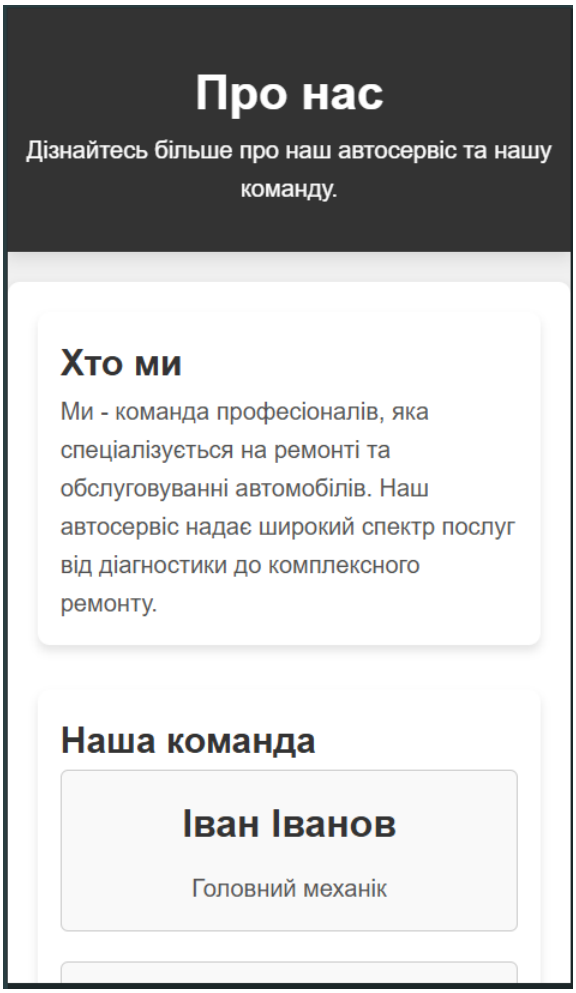

Рисунок 3.22 - Інтерфейс мобільної версії сторінки Про нас

На рисунку 3.23 зображено інтерфейс сторінки мобільної версії «Контакти».

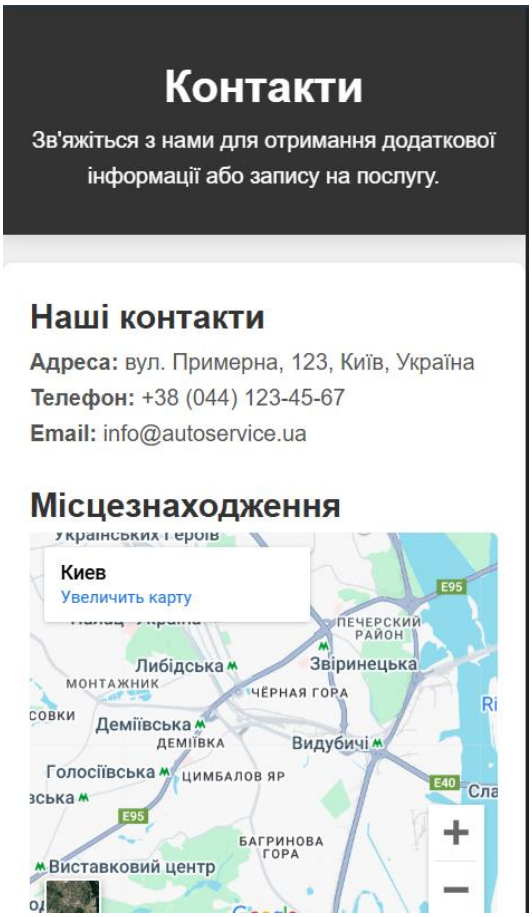

Рисунок 3.23 - Інтерфейс мобільної версії сторінки Контакти

На рисунку 3.24 зображено продовження інтерфейсу сторінки мобільної версії «Контакти».

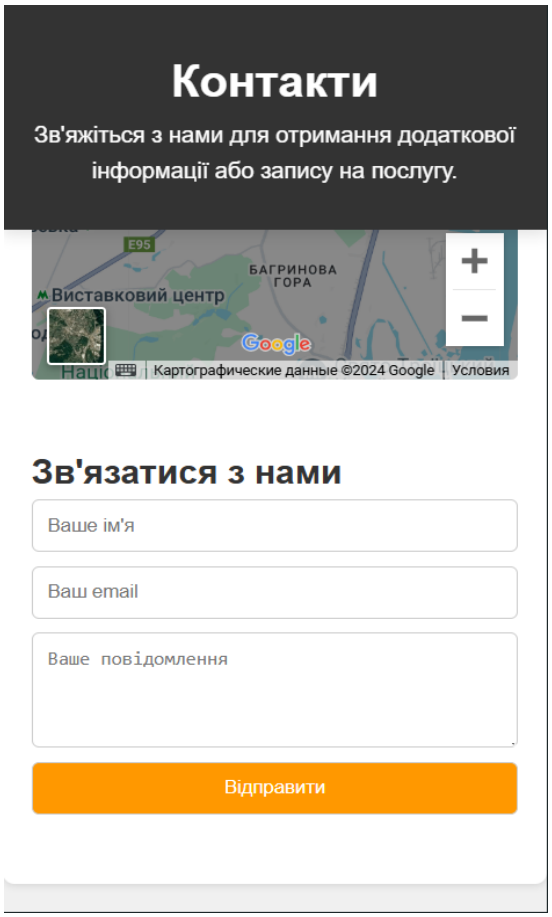

Рисунок 3.24 - Інтерфейс мобільної версії сторінки Контакти продовження

ВИСНОВКИ

У межах цієї роботи було розроблено веб-застосунок для управління автосервісом, який спрямований на оптимізацію бізнес-процесів, автоматизацію обліку, управління замовленнями, запасами та фінансовими операціями.

Ключові результати дослідження

1. Аналіз існуючих проблем показав, що традиційні методи управління автосервісом не відповідають сучасним вимогам ефективності та продуктивності, а впровадження цифрових інструментів є необхідним.
2. Функціональні вимоги до системи були визначені на основі потреб сучасних автосервісів, включаючи управління клієнтами, облік послуг, роботу із замовленнями та складськими операціями.
3. Вибір стеку технологій (React, Node.js, MongoDB) обґрунтовано їхньою продуктивністю, масштабованістю та зручністю інтеграції.
4. Розробка веб-застосунку охопила як створення зручного інтерфейсу для користувачів, так і розробку бекенду з функціями авторизації, обробки даних і взаємодії з базою даних.
5. Тестування і забезпечення безпеки гарантували захищеність даних користувачів і стабільність роботи системи.
6. Економічне обґрунтування довело доцільність розробки: система окупиться менш ніж за два роки, знижуючи операційні витрати і підвищуючи прибутковість бізнесу.

   Практичне значення роботи

Результати розробки можуть бути впроваджені в реальні автосервіси, що дозволить:

- підвищити якість обслуговування клієнтів;
- зменшити витрати на адміністрування;
- забезпечити прозорість управлінських і фінансових процесів;
- створити умови для масштабування бізнесу.

  Подальші перспективи розвитку

1. Інтеграція з мобільними додатками для клієнтів і співробітників.
2. Використання технологій штучного інтелекту для прогнозування потреб у запасах і оптимізації роботи автосервісу.
3. Розширення функціоналу для обробки аналітичних даних та побудови стратегій розвитку бізнесу.

Таким чином, запропонований веб-застосунок є актуальним, економічно вигідним і технічно ефективним рішенням для модернізації автосервісів у сучасних умовах.

ПЕРЕЛІК ПОСИЛАНЬ

1. Курченко О.М. Автоматизація бізнес-процесів – Київ: Видавництво "Бізнес-Тех", 2020. – 256 с. (С. 45–67).
2. Головченко І.В. Економічне обґрунтування ІТ-проєкті – Львів: "Економ-прес", 2018. – 192 с. (С. 89–105).
3. Nielsen J. Designing Web Usability: The Practice of Simplicity – New Riders, 1999. – 432 p. (p. 112–134).
4. Smith J. Digital Transformation in Service Industry– Oxford: TechPress, 2021. – 310 p. (p. 215–240).
5. Taylor R. Software Project Economics – Cambridge: MIT Press, 2019. – 280 p. (p. 145–168).
6. Коваленко О.П. Системи управління підприємством – Харків: "Техніка і технології", 2017. – 320 с. (С. 72–98).
7. Жукова Т.А. Безпека даних у сучасних інформаційних системах – Дніпро: "Наука і практика", 2020. – 280 с. (С. 145–162).
8. Філонов М.В. Архітектура веб-додатків: методи проектування – Одеса: "ІТ-розвиток", 2019. – 340 с. (С. 198–222).
9. White T. Big Data: Managing Data with Hadoop – O'Reilly Media, 2015. – 610 p. (p. 305–328).
10. Freeman E., Robson E. Head First Design Patterns – O'Reilly Media, 2020. – 694 p. (p. 402–425).
11. Гринько В.М. Інформаційні технології в управлінні бізнесом – Київ: "Університетська книга", 2018. – 300 с. (С. 120–140).
12. Чернов О.П. Розробка програмного забезпечення: процеси та методології – Львів: "Прогрес", 2017. – 250 с. (С. 89–115).
13. Артеменко Л.В. Інтегровані системи управління підприємством – Одеса: "Навчальна книга", 2019. – 400 с. (С. 305–340).
14. Bass L., Clements P., Kazman R. Software Architecture in Practice – Addison-Wesley, 2012. – 640 p. (p. 192–215).
15. Коноваленко Ю.С. Економіка інформаційних систем – Харків: "Фінанси і технології", 2021. – 320 с. (С. 98–120).
16. Волков І.В. Основи технічного обслуговування автомобілів. – Київ: Політехніка, 2017. – 295 с.
17. Князєв А.М. Автосервіс: організація і управління. – Харків: Прапор, 2021. – 370 с.
18. Руденко М.В. Інноваційні технології в автосервісах. – Львів: Техніка і технології, 2020. – 312 с.
19. Ткаченко О.Г. Економіка та управління автосервісним підприємством. – Дніпро: Інтербук, 2018. – 250 с.
20. Мельников П.А. Технології діагностики автомобілів. – Москва: Машинобудування, 2022. – 410 с. (С. 156–192).
21. Сидоров В.П. Комплексна автоматизація автосервісу: теорія і практика. – Вінниця: Універсум, 2021. – 285 с.

ДОДАТОК А. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

Файл Goal.js:

```
import React from "react";
import "./Goal.css";

const Goal = () => {
  return (
    <div>
      <header>
        <div className="container">
          <div className="logo">
            <h1>Engine-Service</h1>
          </div>
          <nav>
            <ul>
              <li><a href="#services">Послуги</a></li>
              <li><a href="signup.html">Записатися</a></li>
              <li><a href="aboutus.html">Про нас</a></li>
              <li><a href="#contacts">Контакти</a></li>
            </ul>
          </nav>
        </div>
      </header>

      <section className="banner">
        <div className="container">
          <h2>Знижки на технічне обслуговування!</h2>
```

```
    <a href="signup.html" className="btn">Записатися</a>
  </div>
</section>

<section id="services" className="services">
  <div className="container">
    <h2>Наші послуги</h2>
    <div className="services-grid">
      <div className="service">
        <h3>Ремонт двигуна</h3>
        <p>Професійний ремонт будь-яких типів двигунів.</p>
      </div>
      <div className="service">
        <h3>Заміна масла</h3>
        <p>Швидка та якісна заміна масла.</p>
      </div>
      <div className="service">
        <h3>Комп'ютерна діагностика</h3>
        <p>Точна діагностика з використанням сучасного обладнання.</p>
      </div>
    </div>
  </div>
</section>

<section id="contacts">
  <div className="container">
    <h2>Контакти</h2>
    <p>Телефон: +380 12 345 6789</p>
```

```
          <p>engineservice@gmail.com</p>
          <p>Адреса: вул. Автомобільна, 1, Київ</p>
        </div>
      </section>

      <footer>
        <div className="container">
          <p>&copy; 2024 Engine-Service. Всі права захищені.</p>
        </div>
      </footer>
    </div>
  );
};

export default Goal;
```

Файл AboutUs.js:

```
import React from "react";
import "./AboutUs.css";

const AboutUs = () => {
  return (
    <div>
      <header>
        <h1>Про нас</h1>
        <p>Дізнайтесь більше про наш автосервіс та нашу команду.</p>
      </header>

      <div className="about-container">
```

```
<section className="about-section">
  <h2>Хто ми</h2>
  <p>
    Ми - команда професіоналів, яка спеціалізується на ремонті та обслуговуванні автомобілів. Наш автосервіс надає широкий спектр послуг від діагностики до комплексного ремонту.
  </p>
</section>

{/* Наша команда */}
<section className="about-section">
  <h2>Наша команда</h2>
  <div className="team">
    <div className="team-member">
      <h3>Іван Іванов</h3>
      <p>Головний механік</p>
    </div>
    <div className="team-member">
      <h3>Олена Петрова</h3>
      <p>Фахівець з діагностики</p>
    </div>
    <div className="team-member">
      <h3>Максим Дорошенко</h3>
      <p>Електрик</p>
    </div>
  </div>
</section>
```

```
        {/* Наші переваги */}
        <section className="about-section">
          <h2>Наші переваги</h2>
          <div className="advantages">
            <div className="advantage">
              <strong>Досвід:</strong> Понад 10 років на ринку послуг
              автосервісу.
            </div>
            <div className="advantage">
              <strong>Якість:</strong> Використання сучасного обладнання та
              технологій.
            </div>
            <div className="advantage">
              <strong>Надійність:</strong> Гарантія на всі виконані роботи.
            </div>
          </div>
        </section>
      </div>
    </div>
  );
};

export default AboutUs;
```

Файл ChangeOil.js:

```
import React, { useState } from "react";
import "./ChangeOil.css";
```

```
const ChangeOil = () => {
  const [formData, setFormData] = useState({
    name: "",
    phone: "",
    date: "",
  });

  const handleChange = (e) => {
    const { id, value } = e.target;
    setFormData((prevData) => ({
      ...prevData,
      [id]: value,
    }));
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("Форма відправлена:", formData);
    alert(`Дякуємо за запис, ${formData.name}! Ми зв'яжемося з вами.`);
    setFormData({ name: "", phone: "", date: "" });
  };

  return (
    <div>
      <header>
        <h1>Замена масла</h1>
        <p>Професійна заміна масла для всіх марок автомобілів.</p>
      </header>
```

```
{/* Опис послуги */}
<div className="service-description">
  <h2>Опис послуги</h2>
  <p>
    Наша команда професіоналів виконає заміну масла швидко та якісно.
    Використовуємо тільки найкращі оливи, щоб забезпечити довговічність
    вашого двигуна.
  </p>
  <p>Час виконання: приблизно 30 хвилин.</p>
</div>

{/* Форма запису */}
<section className="booking">
  <h2>Запис на заміну масла</h2>
  <form id="booking-form" onSubmit={handleSubmit}>
    <label htmlFor="name">Ваше ім'я:</label>
    <input
      type="text"
      id="name"
      value={formData.name}
      onChange={handleChange}
      required
    />

    <label htmlFor="phone">Ваш телефон:</label>
    <input
      type="tel"
```

```
          id="phone"
          value={formData.phone}
          onChange={handleChange}
          required
        />

        <label htmlFor="date">Дата:</label>
        <input
          type="date"
          id="date"
          value={formData.date}
          onChange={handleChange}
          required
        />

        <button type="submit">Записатися</button>
      </form>
    </section>
  </div>
 );
};

export default ChangeOil;
```

Файл Contacts.js:

```
import React, { useState } from "react";
import "./Contacts.css";

const Contacts = () => {
```

```
const [formData, setFormData] = useState({
  name: "",
  email: "",
  message: "",
});

const handleChange = (e) => {
  const { id, value } = e.target;
  setFormData((prevData) => ({
    ...prevData,
    [id]: value,
  }));
};

const handleSubmit = (e) => {
  e.preventDefault();
  console.log("Повідомлення відправлено:", formData);
  alert(`Дякуємо, ${formData.name}! Ми відповімо вам найближчим часом.`);
  setFormData({ name: "", email: "", message: "" });
};

return (
  <div>
    <header>
      <h1>Контакти</h1>
      <p>
        Зв'яжіться з нами для отримання додаткової інформації або запису на
        послугу.
```

```
  </p>
</header>

<div className="contact-container">
  {/* Контактна інформація */}
  <section className="contact-section contact-info">
    <h2>Наші контакти</h2>
    <p>
      <strong>Адреса:</strong> вул. Примерна, 123, Київ, Україна
    </p>
    <p>
      <strong>Телефон:</strong> +38 (044) 123-45-67
    </p>
    <p>
      <strong>Email:</strong> info@autoservice.ua
    </p>
  </section>

  {/* Карта */}
  <section className="contact-section">
    <h2>Місцезнаходження</h2>
    <iframe
      className="map"
      src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d25355.653
62749992!2d30.516591846659354!3d50.45008751146813!2m3!1f0!2f0!3f0!3m2!1i10
24!2i768!4f13.1!3m3!1m2!1s0x40d4ce5f5b5a5f5f%3A0x123456789abcdef!2sКиїв!5e
0!3m2!1suk!2sua!4v1600000000000"
      allowFullScreen=""
      loading="lazy"
```

```
    title="Карта місцезнаходження"
  ></iframe>
</section>

{/* Форма для зв'язку */}
<section className="contact-section contact-form">
  <h2>Зв'язатися з нами</h2>
  <form id="contact-form" onSubmit={handleSubmit}>
    <input
      type="text"
      id="name"
      placeholder="Ваше ім'я"
      value={formData.name}
      onChange={handleChange}
      required
    />
    <input
      type="email"
      id="email"
      placeholder="Ваш email"
      value={formData.email}
      onChange={handleChange}
      required
    />
    <textarea
      id="message"
      rows="4"
      placeholder="Ваше повідомлення"
```

```
            value={formData.message}
            onChange={handleChange}
            required
          />
          <button type="submit">Відправити</button>
        </form>
      </section>
    </div>
  </div>
 );
};

export default Contacts;
```

Файл Diagnostics.js:

```
import React, { useState } from "react";
import "./Diagnostics.css";

const Diagnostics = () => {
 const [formData, setFormData] = useState({
  name: "",
  phone: "",
  date: "",
 });

 const handleChange = (e) => {
  const { id, value } = e.target;
  setFormData((prevData) => ({
   ...prevData,
```

```
      [id]: value,
    }));
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("Запис на діагностику:", formData);
    alert(
      `Дякуємо, ${formData.name}! Ви записані на діагностику ${formData.date}.`
    );
    setFormData({ name: "", phone: "", date: "" });
  };

  return (
    <div>
      <header>
        <h1>Діагностика</h1>
        <p>Професійна діагностика автомобілів для точного визначення
проблем.</p>
      </header>

      <div className="service-description">
        <h2>Опис послуги</h2>
        <p>
          Наші експерти проведуть детальну діагностику вашого автомобіля,
          використовуючи сучасне обладнання та технології. Ми виявимо будь-які
          проблеми та надамо рекомендації щодо їх усунення.
        </p>
        <p>Час виконання: приблизно 1 година.</p>
```

```
</div>

<section className="booking">
  <h2>Запис на діагностику</h2>
  <form id="booking-form" onSubmit={handleSubmit}>
    <label htmlFor="name">Ваше ім'я:</label>
    <input
      type="text"
      id="name"
      value={formData.name}
      onChange={handleChange}
      required
    />

    <label htmlFor="phone">Ваш телефон:</label>
    <input
      type="tel"
      id="phone"
      value={formData.phone}
      onChange={handleChange}
      required
    />

    <label htmlFor="date">Дата:</label>
    <input
      type="date"
      id="date"
      value={formData.date}
```

```
            onChange={handleChange}
            required
          />

          <button type="submit">Записатися</button>
        </form>
      </section>
    </div>
  );
};

export default Diagnostics;
```

Файл RepaireEngine.js:

```
import React, { useState } from "react";
import "./RepairEngine.css";

const RepairEngine = () => {
  const [formData, setFormData] = useState({
    name: "",
    phone: "",
    date: "",
  });

  const handleChange = (e) => {
    const { id, value } = e.target;
    setFormData((prevData) => ({
      ...prevData,
      [id]: value,
```

```
    }));
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("Запис на ремонт двигуна:", formData);
    alert(
      `Дякуємо, ${formData.name}! Ви записані на ремонт двигуна
${formData.date}.`
    );
    setFormData({ name: "", phone: "", date: "" });
  };

  return (
    <div>
      <header className="header-repair-engine">
        <h1>Ремонт Двигуна</h1>
        <p>Професійний ремонт двигунів для будь-яких марок автомобілів.</p>
      </header>

      <section className="about-service">
        <h2>Опис послуги</h2>
        <p>
          Наш автосервіс пропонує повний спектр послуг з ремонту двигунів: від
          діагностики та заміни запчастин до капітального ремонту. Ми
          використовуємо тільки якісні деталі та сучасне обладнання.
        </p>
        <ul>
          <li>Діагностика несправностей двигуна</li>
```

```
    <li>Ремонт бензинових та дизельних двигунів</li>
    <li>Заміна деталей та комплектуючих</li>
    <li>Регулярне технічне обслуговування</li>
  </ul>
</section>

<section className="advantages">
  <h2>Переваги нашого сервісу</h2>
  <div className="advantage">
    <h3>Досвідчені фахівці</h3>
    <p>
      Наші майстри мають багаторічний досвід роботи з різними моделями
      автомобілів.
    </p>
  </div>
  <div className="advantage">
    <h3>Якісні запчастини</h3>
    <p>Ми використовуємо тільки сертифіковані та перевірені запчастини.</p>
  </div>
  <div className="advantage">
    <h3>Гарантія якості</h3>
    <p>Ми надаємо гарантію на всі види виконаних робіт.</p>
  </div>
</section>

<section className="booking">
  <h2>Запишіться на ремонт</h2>
  <form id="booking-form" onSubmit={handleSubmit}>
```

```
<label htmlFor="name">Ваше ім'я:</label>
<input
  type="text"
  id="name"
  value={formData.name}
  onChange={handleChange}
  required
/>

<label htmlFor="phone">Номер телефону:</label>
<input
  type="tel"
  id="phone"
  value={formData.phone}
  onChange={handleChange}
  required
/>

<label htmlFor="date">Оберіть дату:</label>
<input
  type="date"
  id="date"
  value={formData.date}
  onChange={handleChange}
  required
/>

<button type="submit">Записатися</button>
```

```
        </form>
      </section>

      <footer>
        <p>© 2024 Автосервіс. Всі права захищені.</p>
      </footer>
    </div>
  );
};

export default RepairEngine;
```

Файл Services.js:

```
import React, { useState } from "react";
import "./Services.css";

const servicePrices = {
  repair: 5000,
  oil: 600,
  diagnostics: 300,
};

const services = [
  { id: 1, name: "Ремонт двигуна", category: "repair", description: "Професійний ремонт та діагностика двигунів різних моделей автомобілів." },
  { id: 2, name: "Заміна масла", category: "oil", description: "Швидка та якісна заміна масла для будь-яких типів автомобілів." },
  { id: 3, name: "Діагностика", category: "diagnostics", description: "Комп'ютерна діагностика автомобіля для виявлення несправностей." },
];
```

```
function App() {
  const [selectedCategory, setSelectedCategory] = useState("all");
  const [selectedService, setSelectedService] = useState("repair");
  const [quantity, setQuantity] = useState(1);
  const [totalCost, setTotalCost] = useState(0);

  const handleFilterChange = (event) => {
    setSelectedCategory(event.target.value);
  };

  const handleServiceChange = (event) => {
    setSelectedService(event.target.value);
  };

  const handleQuantityChange = (event) => {
    setQuantity(Number(event.target.value));
  };

  const calculateCost = () => {
    if (quantity > 0) {
      setTotalCost(servicePrices[selectedService] * quantity);
    } else {
      alert("Введіть коректну кількість!");
    }
  };

  const filteredServices = services.filter(
```

```
  (service) => selectedCategory === "all" || service.category === selectedCategory
);

return (
  <div className="App">
    <header>
      <h1>Engine-Service</h1>
    </header>

    <section className="services">
      <h1>Наші Послуги</h1>

      {/* Фільтр послуг */}
      <div className="filter">
        <label htmlFor="service-category">Категорія послуг:</label>
        <select id="service-category" onChange={handleFilterChange}>
          <option value="all">Всі послуги</option>
          <option value="repair">Ремонт двигуна</option>
          <option value="oil">Заміна масла</option>
          <option value="diagnostics">Діагностика</option>
        </select>
      </div>

      {/* Список послуг */}
      <div className="service-list">
        {filteredServices.map((service) => (
          <div key={service.id} className="service-item">
            <h3>{service.name}</h3>
```

```
        <p>{service.description}</p>
        <button
          className="details-btn"
          onClick={() => alert(`Перехід до: ${service.name}`)}
        >
          Детальніше
        </button>
      </div>
    ))}
  </div>
</section>

{/* Калькулятор вартості */}
<section className="pricing-calculator">
  <h2>Розрахуйте вартість послуг</h2>

  <div className="calculator">
    <label htmlFor="service-selection">Оберіть послугу:</label>
    <select id="service-selection" onChange={handleServiceChange}>
      {Object.entries(servicePrices).map(([key, price]) => (
        <option key={key} value={key}>
          {key === "repair" && `Ремонт двигуна - ${price} грн`}
          {key === "oil" && `Заміна масла - ${price} грн`}
          {key === "diagnostics" && `Діагностика - ${price} грн`}
        </option>
      ))}
    </select>
```

```
        <label htmlFor="quantity">Кількість:</label>
        <input
          id="quantity"
          type="number"
          value={quantity}
          min="1"
          onChange={handleQuantityChange}
        />

        <button className="calculate-btn" onClick={calculateCost}>
          Розрахувати
        </button>

        <div className="total-cost">
          Загальна вартість: <span>{totalCost.toLocaleString("uk-UA")} грн</span>
        </div>
      </div>
    </section>
  </div>
 );
}

export default Services;
```

Файл SignUp.js:

```
import React, { useState } from "react";
import "./SignApp.css";

function SignApp() {
```

```
const [service, setService] = useState("");
const [name, setName] = useState("");
const [phone, setPhone] = useState("");
const [date, setDate] = useState("");
const [message, setMessage] = useState("");

const handleSubmit = (event) => {
  event.preventDefault();
  if (service && name && phone && date) {
    setMessage("Ваш запис успішно створено!");
  } else {
    setMessage("Будь ласка, заповніть всі поля.");
  }
};

return (
  <div className="SignApp">
    <header>
      <h1>Записатися на послугу</h1>
      <p>Виберіть послугу, яку ви бажаєте замовити.</p>
    </header>

    <section className="booking">
      <form id="booking-form" onSubmit={handleSubmit}>
        <label htmlFor="service">Виберіть послугу:</label>
        <select
          id="service"
          value={service}
```

```
  onChange={(e) => setService(e.target.value)}
  required
>
  <option value="">Оберіть послугу...</option>
  <option value="engine-repair">Ремонт двигуна</option>
  <option value="oil-change">Заміна масла</option>
  <option value="diagnostics">Діагностика</option>
</select>

<label htmlFor="name">Ваше ім'я:</label>
<input
  type="text"
  id="name"
  value={name}
  onChange={(e) => setName(e.target.value)}
  required
/>

<label htmlFor="phone">Ваш телефон:</label>
<input
  type="tel"
  id="phone"
  value={phone}
  onChange={(e) => setPhone(e.target.value)}
  required
/>

<label htmlFor="date">Дата:</label>
```

```
          <input
            type="date"
            id="date"
            value={date}
            onChange={(e) => setDate(e.target.value)}
            required
          />

          <button type="submit">Записатися</button>
        </form>
      </section>

      {message && <p className="message">{message}</p>}
    </div>
  );
}

export default SignUp;
```

Файл index.js:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
```

```
  document.getElementById('root')
);
```

Файл App.js:

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import AboutUs from './components/AboutUs';
import ChangeOil from './components/ChangeOil';
import Contacts from './components/Contacts';
import Diagnostics from './components/Diagnostics';
import RepaireEngine from './components/RepaireEngine';
import Services from './components/Services';
import SignUp from './components/SignUp';

function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<Services />} />
        <Route path="/goal" element={<Goal />} />
        <Route path="/about" element={<AboutUs />} />
        <Route path="/change-oil" element={<ChangeOil />} />
        <Route path="/contacts" element={<Contacts />} />
        <Route path="/diagnostics" element={<Diagnostics />} />
        <Route path="/repair-engine" element={<RepaireEngine />} />
        <Route path="/signup" element={<SignUp />} />
      </Routes>
    </Router>
```

```
  );
}

export default App;
```

Файл AboutUs.css:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}

header {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 30px 0;
}

h1 {
    margin: 0;
}

.about-container {
    max-width: 800px;
    margin: 20px auto;
    padding: 20px;
```

```
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.about-section {
    margin-bottom: 20px;
}

.about-section h2 {
    color: #333;
}

.about-section p {
    color: #555;
    line-height: 1.6;
}

.team {
    display: flex;
    flex-wrap: wrap;
    gap: 20px;
    justify-content: center;
}

.team-member {
    flex: 1 1 200px;
    padding: 15px;
```

```
    background-color: #f9f9f9;
    border: 1px solid #ccc;
    border-radius: 5px;
    text-align: center;
}

.team-member h3 {
    margin-top: 0;
}

.team-member p {
    margin-bottom: 0;
}

.advantages {
    display: flex;
    flex-direction: column;
    gap: 10px;
}

.advantage {
    background-color: #e7f4e4;
    padding: 10px;
    border-radius: 5px; }

.about-container {
        display: flex;
        flex-direction: row;
```

```
  justify-content: space-between;
  gap: 20px;
  margin-top: 20px;
}

.about-section {
  flex: 1;
  padding: 15px;
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

.about-section h3 {
  font-size: 1.6rem;
  margin-bottom: 10px;
}

.about-section p {
  font-size: 1rem;
}


@media (max-width: 768px) {
  .about-container {
    flex-direction: column;
    gap: 10px;
  }
```

```
        .about-section {
          width: 100%;
        }
      }
```

Файл ChangeOil.css:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}

header {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 20px 0;
}

h1 {
    margin: 0;
}

.service-description {
    margin: 20px auto;
```

```
    padding: 20px;
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    max-width: 600px;
}

.booking {
    max-width: 400px;
    margin: 20px auto;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
    background-color: #f9f9f9;
}

.booking input {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
}

.booking button {
    padding: 10px;
    background-color: #e08900;
    color: white;
```

```
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
}

.booking button:hover {
    background-color: #e08900
}
```

Файл Contacts.css:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}

header {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 30px 0;
}

h1 {
    margin: 0;
}
```

```
.contact-container {
    max-width: 800px;
    margin: 20px auto;
    padding: 20px;
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.contact-section {
    margin-bottom: 20px;
}

.contact-section h2 {
    color: #333;
}

.contact-section p {
    color: #555;
    line-height: 1.6;
}

.contact-info, .contact-form {
    margin-bottom: 20px;
}

.contact-info p {
```

```
    font-size: 16px;
}

.map {
    width: 100%;
    height: 300px;
    border: none;
    border-radius: 5px;
    margin-bottom: 20px;
}

form {
    display: flex;
    flex-direction: column;
}

form input, form textarea, form button {
    margin-bottom: 10px;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

form button {
    background-color: #ff9800;
    color: white;
    cursor: pointer;
    transition: background-color 0.3s;
```

```
}
```

Файл Diagnostics.css:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}

header {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 20px 0;
}

h1 {
    margin: 0;
}

.service-description {
    margin: 20px auto;
    padding: 20px;
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    max-width: 600px;
```

```
}

.booking {
    max-width: 400px;
    margin: 20px auto;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
    background-color: #f9f9f9;
}

.booking input {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
}

.booking button {
    padding: 10px;
    background-color: #ff9800;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
}
```

Файл Goal.css:

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Arial', sans-serif;
    line-height: 1.6;
    color: #333;
    background-color: #f4f4f4;
}

header {
    background: #333;
    color: #fff;
    padding: 1rem 0;
    position: sticky;
    top: 0;
    z-index: 1000;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
}

.container {
    width: 90%;
    max-width: 1200px;
```

```
    margin: 0 auto;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.logo h1 {
    font-size: 1.8rem;
    font-weight: bold;
}

nav ul {
    list-style: none;
    display: flex;
}

nav ul li {
    margin-left: 20px;
}

nav ul li a {
    color: #fff;
    text-decoration: none;
    font-size: 1.1rem;
    transition: color 0.3s ease;
}

nav ul li a:hover {
```

```
    color: #ff9800;
}

/* Банер */
.banner {
    background: url('../img/car.jpg') no-repeat center center/cover;
    height: 450px;
    display: flex;
    justify-content: flex-start;
    align-items: center;
    position: relative;
    text-align: left;
    color: #fff;
    padding: 0 50px;
}

.banner::before {
    content: '';
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.5);
    z-index: 1;
}

.banner .container {
```

```
    z-index: 2;
    position: relative;
    max-width: 1200px;
    text-align: left;
}

.container h2 {
    font-size: 3.5rem;
    margin-bottom: 0.5rem;
    text-transform: uppercase;
    font-weight: bold;
    line-height: 1.2;
    letter-spacing: 2px;
    opacity: 0;
    animation: fadeInUp 1s ease-in-out forwards;
}

.container p {
    font-size: 1.4rem;
    margin-bottom: 1.5rem;
    max-width: 600px;
    opacity: 0;
    animation: fadeInUp 1.5s ease-in-out forwards;
}

.btn {
    background-color: #ff9800;
    color: #fff;
```

```
    padding: 0.75rem 2rem;
    font-size: 1.2rem;
    text-decoration: none;
    border-radius: 50px;
    transition: background-color 0.3s ease, transform 0.3s ease;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    opacity: 0;
    animation: fadeInUp 2s ease-in-out forwards;
}

.btn:hover {
    background-color: #e08900;
    transform: scale(1.05);
}

/* Анімації */
@keyframes fadeInUp {
    from {
        opacity: 0;
        transform: translateY(30px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}

@media (max-width: 768px) {
```

```
    .container h2 {
        font-size: 2.5rem;
    }

    .container p {
        font-size: 1.2rem;
    }

    .btn {
        font-size: 1rem;
    }
}

.services {
    padding: 3rem 0;
    background-color: #fff;
    text-align: center;
}

.services h2 {
    font-size: 2rem;
    margin-bottom: 2rem;
}

.services-grid {
    display: flex;
    justify-content: space-between;
    gap: 1.5rem;
```

```
}

.service {
    background: #f9f9f9;
    padding: 1.5rem;
    border-radius: 5px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    text-align: center;
    flex: 1;
    transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.service h3 {
    font-size: 1.5rem;
    margin-bottom: 1rem;
}

.service p {
    font-size: 1rem;
    color: #666;
}

.service:hover {
    transform: translateY(-10px);
    box-shadow: 0 6px 12px rgba(0, 0, 0, 0.15);
}

#contacts {
```

```
    padding: 3rem 0;
    background-color: #333;
    color: #fff;
    text-align: center;
}

#contacts h2 {
    font-size: 2rem;
    margin-bottom: 1.5rem;
}

#contacts p {
    font-size: 1.2rem;
    margin-bottom: 1.6rem;
}

footer {
    background: #222;
    color: #fff;
    text-align: center;
    padding: 1rem 0;
    margin-top: 2rem;
    font-size: 0.9rem;
}
```

Файл RepaireEngine.css:

```
body {
    font-family: Arial, sans-serif;
```

```
    background-color: #f9f9f9;
    color: #333;
    line-height: 1.6;
    margin: 0;
    padding: 120px 0 0 0;
}

header {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    background-color: #333;
    color: white;
    padding: 20px;
    text-align: center;
    z-index: 1000;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
    transition: background-color 0.3s ease;
}

header h1 {
    font-size: 3em;
    margin-bottom: 10px;
}

header p {
    font-size: 1.5em;
```

```
}

.about-service, .advantages, .booking {
    padding: 40px 20px;
    background: white;
    margin: 20px auto;
    max-width: 800px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0,0,0,0.1);
}

.about-service h2, .advantages h2, .booking h2 {
    text-align: center;
    margin-bottom: 20px;
}

.advantages .advantage {
    padding: 15px;
    margin-bottom: 15px;
    border-bottom: 1px solid #ddd;
}

.advantages .advantage:last-child {
    border-bottom: none;
}

.booking form {
    display: flex;
```

```
    flex-direction: column;
    gap: 15px;
}

.booking form input, .booking form button {
    padding: 10px;
    font-size: 1.1rem;
    border: 1px solid #ddd;
    border-radius: 5px;
}

.booking form button {
    background-color: #ff9800;
    color: white;
    cursor: pointer;
    transition: background-color 0.3s;
}

.booking form button:hover {
    background-color: #e08900;
}

footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 10px 0;
}
```

Файл Services.css:

```
.services {
    padding: 50px;
    background-color: #f9f9f9;
}

.services h1 {
    text-align: center;
    margin-bottom: 40px;
}

.filter {
    text-align: center;
    margin-bottom: 30px;
}

.filter select {
    padding: 10px;
    font-size: 1.1rem;
}

.service-list {
    display: flex;
    justify-content: space-between;
    flex-wrap: wrap;
}
```

```
.service-item {
    background-color: white;
    padding: 20px;
    width: 30%;
    margin-bottom: 20px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    transition: transform 0.3s ease;
}

.service-item:hover {
    transform: translateY(-10px);
}

.service-item h3 {
    margin-top: 0;
}

.details-btn {
    padding: 10px 20px;
    background-color: #ff9800;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.details-btn:hover {
```

```
    background-color: #e08900;
}

.pricing-calculator {
    padding: 50px;
    background-color: #fff;
}

.pricing-table {
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 30px;
}

.pricing-table th, .pricing-table td {
    padding: 15px;
    border: 1px solid #ddd;
}

.calculator {
    text-align: center;
}

.calculator select, .calculator input {
    padding: 10px;
    font-size: 1.1rem;
    margin-bottom: 20px;
}
```

```
.calculate-btn {
    padding: 10px 20px;
    background-color: #ff9800;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.calculate-btn:hover {
    background-color: #e08900;
}

.total-cost {
    font-size: 1.5rem;
    font-weight: bold;
    margin-top: 20px;
}

.service-list {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
    gap: 20px;
    margin-top: 30px;
  }
```

```
.service-item {
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  padding: 15px;
  text-align: center;
}

.service-item h3 {
  font-size: 1.4rem;
  margin-bottom: 10px;
}

.service-item p {
  font-size: 1rem;
}


@media (max-width: 768px) {
  .service-list {
    grid-template-columns: 1fr;
    gap: 15px;
  }

  .service-item {
    padding: 10px;
  }
```

```
    .service-item h3 {
      font-size: 1.2rem;
    }

    .service-item p {
      font-size: 0.9rem;
    }
  }
```

Файл SignUp.css:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}

header {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 20px 0;
}

h1 {
    margin: 0;
}
```

```
.booking {
    max-width: 400px;
    margin: 20px auto;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
    background-color: #f9f9f9;
}

.booking input, .booking select {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
}

.booking button {
    padding: 10px;
    background-color: #e08900;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
}
.booking form {
```

```
  display: flex;
  flex-direction: column;
  gap: 15px;
  margin-top: 20px;
}

.booking input,
.booking select,
.booking button {
  padding: 10px;
  font-size: 1rem;
  border: 1px solid #ccc;
  border-radius: 8px;
}

.booking button {
  background-color: #333;
  color: white;
  cursor: pointer;
  font-weight: bold;
}

.booking button:hover {
  background-color: #555;
}


@media (max-width: 768px) {
```

```
    .booking form {
      gap: 10px;
    }

    .booking input,
    .booking select,
    .booking button {
      font-size: 0.9rem;
    }
  }
```

Файл index.css:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    background-color: #f9f9f9;
  }

  header {
    background-color: #333;
    color: white;
    padding: 20px;
    text-align: center;
  }

  h1 {
```

```
  margin: 0;
  font-size: 2rem;
}

p {
  font-size: 1rem;
  margin-top: 10px;
}


.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px;
}


nav ul {
  list-style: none;
  padding: 0;
  display: flex;
  justify-content: space-around;
}

nav ul li {
  display: inline;
}
```

```
nav ul li a {
  color: white;
  text-decoration: none;
  font-size: 1.2rem;
}


@media (max-width: 768px) {
  header {
    padding: 15px;
  }

  nav ul {
    flex-direction: column;
    align-items: center;
  }

  nav ul li {
    margin-bottom: 10px;
  }

  .container {
    padding: 15px;
  }

  h1 {
    font-size: 1.5rem;
  }
```

```
  p {
    font-size: 0.9rem;
  }
}


@media (max-width: 1024px) {
  .container {
    padding: 20px;
  }

  header {
    padding: 25px;
  }

  h1 {
    font-size: 1.8rem;
  }

  p {
    font-size: 1rem;
  }
}

/* Десктоп */
@media (min-width: 1025px) {
  .container {
```

```
    padding: 40px;
  }

  h1 {
    font-size: 2.5rem;
  }

  p {
    font-size: 1.2rem;
  }
}
```

ДОДАТОК Б. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

1

ДИПЛОМНА РОБОТА
на ступінь вищої освіти магістр
із спеціальності 122 Комп'ютерні науки

«Веб-застосунок для управління послугами автосервісу на основі JavaScript»

Виконав: студент 6 курсу, групи КНДМ-61
Галата Ярослав Олександрович
Керівник: д.ф.-м.н., професор Олена Шикула

Київ - 2024

2

ЗАГАЛЬНА ХАРАКТЕРИСТИКА ДИПЛОМНОЇ РОБОТИ

Тема «Веб-застосунок для управління послугами автосервісу на основі JavaScript»

Мета дослідження дослідження веб-застосунку для автоматизації управління послугами автосервісу, який включає облік замовлень, керування запасами, взаємодію з клієнтами та фінансові операції, з використанням сучасних веб-технологій для підвищення ефективності та зручності роботи автосервісу.

Наукове завдання дослідження веб-застосунку для управління послугами з використанням сучасних технологій для автоматизації обліку, обробки замовлень та підвищення ефективності бізнес-процесів.

Об'єкт дослідження веб-застосунок для автоматизації управління послугами автосервісу, який охоплює управління записами клієнтів, облік замовлень, управління запасами, фінансові операції та взаємодію з користувачами через інтерфейс системи.

Предмет дослідження процеси автоматизації управління послугами автосервісу за допомогою веб-застосунку, зокрема методи обробки замовлень, управління обліком запасів, клієнтських записів, фінансових операцій та взаємодії з користувачами через інтерфейс системи.

3

Актуальність теми

1. Проблема:
Сучасні автосервіси стикаються зі значними викликами в управлінні процесами:
- **неефективна організація запису клієнтів;**
- **тривалі терміни обробки замовлень;**
- **обмежена прозорість для клієнтів щодо процесу ремонту чи обслуговування.**

Відсутність інтегрованих IT-рішень ускладнює обслуговування клієнтів і підвищення якості послуг.
2. Тенденції та актуальність:
Розвиток цифрових технологій створює нові можливості для оптимізації роботи автосервісів.
Зростання попиту на зручні онлайн-платформи, що дозволяють:
- **записуватись на обслуговування;**
- **відстежувати статус ремонту;**
- **отримувати консультації в режимі реального часу.**

3. Економічна та соціальна важливість:
Покращення клієнтського досвіду через цифровізацію.
Збільшення конкурентоспроможності автосервісів за рахунок автоматизації процесів.
4. Наукова значущість:
Дослідження спрямоване на розробку сучасної архітектури управління автосервісами.
Сприяння впровадженню новітніх технологій у галузі обслуговування транспорту.
5. Чому це важливо?
Впровадження таких рішень дозволяє суттєво скоротити час обслуговування, підвищити ефективність роботи персоналу та покращити задоволеність клієнтів.

4

МЕТА І ЗАВДАННЯ ДОСЛІДЖЕННЯ

1. **Мета роботи:**
   Розробка веб-застосунку для управління автосервісом, що оптимізує взаємодію між клієнтами та працівниками, спрощує процес запису, відстеження послуг і забезпечує ефективну роботу сервісу.
2. **Основні завдання:**
   Проаналізувати сучасні IT-рішення для автоматизації роботи автосервісів.
   Розробити структуру веб-застосунку, яка враховує потреби клієнтів і працівників.
   Реалізувати функціонал для управління записами, обліку послуг і взаємодії з клієнтами.
   Забезпечити зручний та інтуїтивно зрозумілий інтерфейс для користувачів.
   Провести тестування готового рішення та оцінити його ефективність у роботі автосервісу.

Методологія та інструменти розробки

1. Об'єктно-орієнтоване програмування (ООП):
Реалізація структурованого підходу до моделювання компонентів веб-застосунку.
2. JavaScript: Основна мова для розробки інтерактивного інтерфейсу.
3. React: Розробка динамічного та адаптивного інтерфейсу користувача.
4. Git та GitHub: Для збереження коду та спільної роботи.
5. Процес розробки:
Аналіз вимог → проєктування архітектури → реалізація функцій → тестування → впровадження.
6. Інтеграційне тестування: Оцінка взаємодії компонентів у системі.

Архітектура системи

1.Клієнтська частина (Frontend):
- Розроблена з використанням React.
- Інтерфейс користувача адаптивний, дозволяє працювати на різних пристроях (ПК, смартфони, планшети).

2. Взаємодія компонентів:
- Користувачі взаємодіють із фронтендом через веб-браузер.
- Фронтенд надсилає запити до бекенду через HTTP/HTTPS протокол.

3. Переваги архітектури:
- Масштабованість: Можливість додавання нових функцій без значних змін системи.
- Безпека: Використання токенів для авторизації та шифрування даних.
- Продуктивність: Розподіл навантаження між сервером і клієнтом.

Функціональні можливості системи

1. Основні функції для клієнтів:
- Перегляд послуг:
знайомлення з переліком послуг автосервісу, таких як ремонт двигуна, діагностика, заміна масла тощо.
- Запис на послугу:
Зручна форма для запису з вибором дати, часу та потрібної послуги.
- Отримання сповіщень:
Нагадування про запис через SMS або електронну пошту.

2. Функції для адміністраторів:
- Управління записами:
Перегляд, редагування або видалення записів клієнтів.
- Статистика та звітність:
Генерація звітів про кількість наданих послуг, доходи, активність клієнтів.
- Управління користувачами:
Додавання нових клієнтів, редагування їх даних, контроль доступу до системи.

Технології, використані в розробці системи

1. Фронтенд (Клієнтська частина):
- React.js
  - Побудова інтерфейсу користувача.
  - Компонентний підхід для створення багаторазових елементів.
- HTML5, CSS3, JavaScript:
  - Відповідальні за структуру, стиль та функціональність веб-застосунку.

2. Інструменти розробки:
- Git:
  - Контроль версій, командна робота.
- VS Code:
  - Основне середовище для написання коду.
- Postman:
  - Тестування API-запитів.

Функціональні можливості розробленої системи

1. Реєстрація та авторизація користувачів:
•Користувачі можуть створювати облікові записи для доступу до функціоналу системи.
•Авторизація через введення логіна та пароля, із захистом за допомогою JWT.
2. Кабінет користувача:
•Відображення персональної інформації клієнта.
•Перегляд історії замовлень та стану виконання послуг.
•Можливість редагування контактних даних.
3. Запис на послуги:
•Інтерактивна форма для вибору послуги, дати та часу.
•Автоматична перевірка доступності часу для уникнення конфліктів.
•Генерація електронного підтвердження замовлення.
4. Перегляд каталогу послуг:
•Користувачі можуть ознайомитися зі списком доступних послуг.
•Для кожної послуги надається опис, орієнтовна ціна та час виконання.
5. Система нагадувань:
•Надсилання сповіщень на електронну пошту про заплановані послуги.
•Нагадування про необхідність повторного обслуговування (наприклад, заміна масла).
6. Адміністративна панель (для співробітників автосервісу):
•Управління списком замовлень.
•Додавання, редагування та видалення записів на послуги.

Етапи розробки системи

1. Аналіз вимог:
•Вивчення потреб клієнтів та специфіки роботи автосервісу.
•Визначення ключових функціональних вимог до системи.
•Узгодження технічного завдання.
2. Проектування архітектури:
•Розробка структури бази даних.
•Вибір технологій для бекенду та фронтенду.
•Створення прототипу інтерфейсу.
3. Розробка:
•Програмування бекенд-частини: авторизація, робота з базою даних, API.
•Розробка фронтенду: динамічні форми, інтерактивний інтерфейс.
4. Тестування:
•Перевірка функціональності системи: реєстрація, запис на послуги, обробка даних.
•Тестування на різних пристроях та браузерах.
•Виправлення помилок, оптимізація продуктивності.
•Реалізація адаптивного дизайну для різних пристроїв.

Результати впровадження проєкту

11

1. Ефективність обслуговування клієнтів:
•Запроваджено можливість онлайн-запису на послуги автосервісу, що скорочує час очікування для клієнтів.
•Чітка система управління замовленнями підвищує організованість роботи персоналу.
2. Автоматизація бізнес-процесів:
•Автоматичне зберігання та обробка записів клієнтів у базі даних.
•Генерація звітів про завантаженість сервісу та популярність послуг.
3. Підвищення задоволеності клієнтів:
•Зручний та інтуїтивний інтерфейс для замовлення послуг.
•Можливість перегляду цін, вибору часу та отримання оперативної інформації.
4. Розширення клієнтської бази:
•Завдяки інтеграції з хмарними сервісами проєкт доступний з будь-якої точки світу.
•Наявність багатомовного інтерфейсу для роботи з різними категоріями клієнтів.
5. Модернізація процесів комунікації:
•Електронні повідомлення для підтвердження запису та нагадування про послугу.
•Інтеграція з месенджерами або SMS-сервісами для зворотного зв'язку.

Економічний ефект від впровадження проєкту

12

1. Зростання доходів автосервісу:
•Оптимізація внутрішніх процесів сприяє збільшенню продуктивності.
•Прозорість обліку послуг дозволяє уникнути втрат та фінансових помилок.
2. Зниження операційних витрат:
•Автоматизація знижує потребу в додатковому персоналі для виконання рутинних завдань.
•Скорочення витрат на обробку інформації та документів.
3. Підвищення конкурентоспроможності:
•Використання сучасних технологій приваблює більше клієнтів.
•Можливість залучення нових ринків за рахунок онлайн-сервісів.
4. Ефективне управління ресурсами:
•Оптимізація завантаження персоналу та обладнання.
•Мінімізація простоїв завдяки попередньому запису та розподілу робіт.
5. Можливості масштабування:
•Простота впровадження додаткових функцій та розширення бізнесу.
•Легке адаптування системи до потреб зростаючої кількості клієнтів.

Розроблений сайт

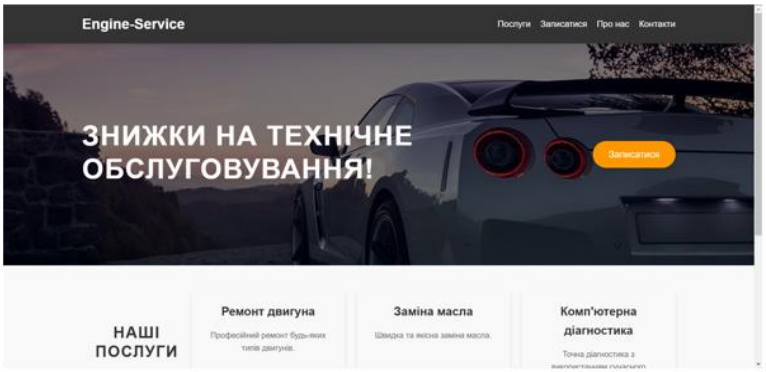

Рисунок 13.1 - Головна сторінка

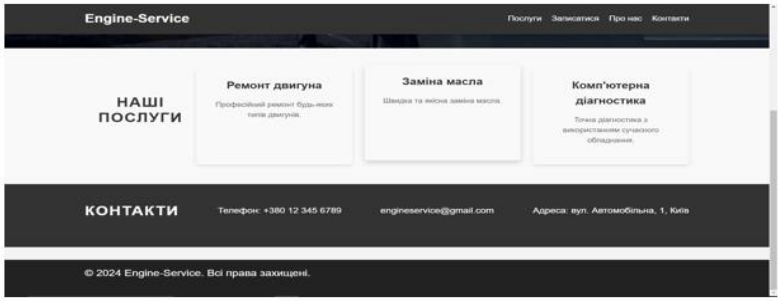

Рисунок 13.2 - Головна сторінка продовження

Розроблений сайт

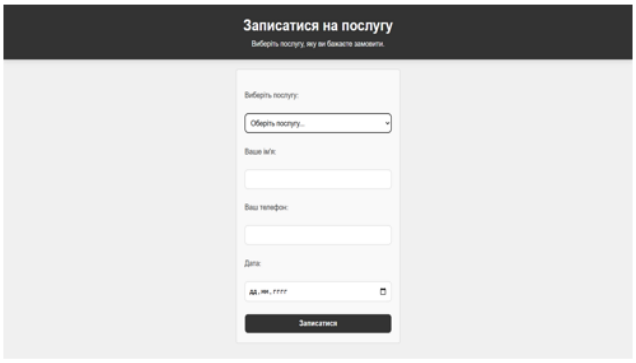

Рисунок 14.1 – Інтерфейс запису на послугу

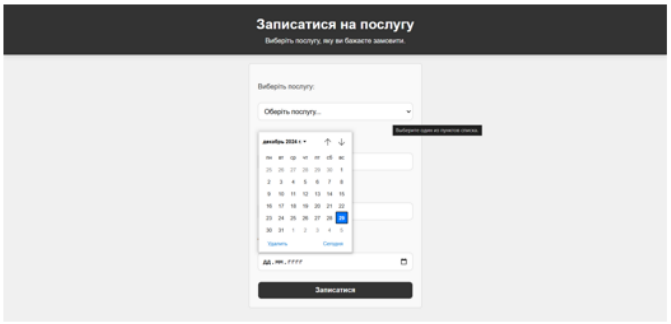

Рисунок 14.1 – Інтерфейс запису, рокзриття функцій

Розроблений сайт

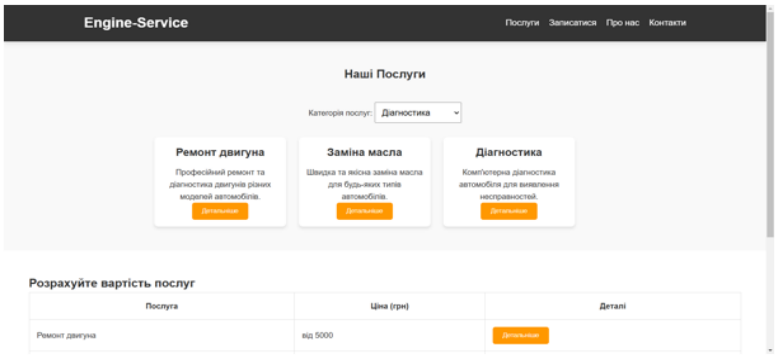

Рисунок 14.1 – Інтерфейс вкладкі послуги

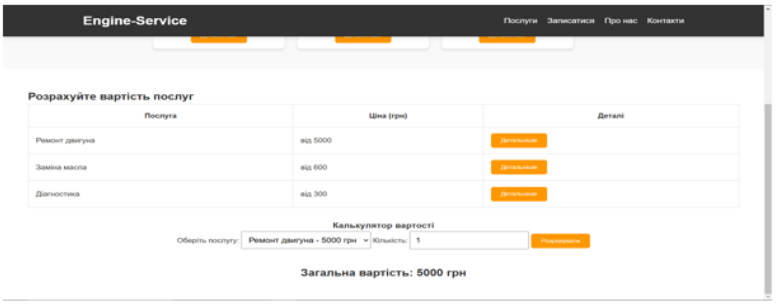

Рисунок 14.1 – Інтерфейс вкладкі послуги продовження

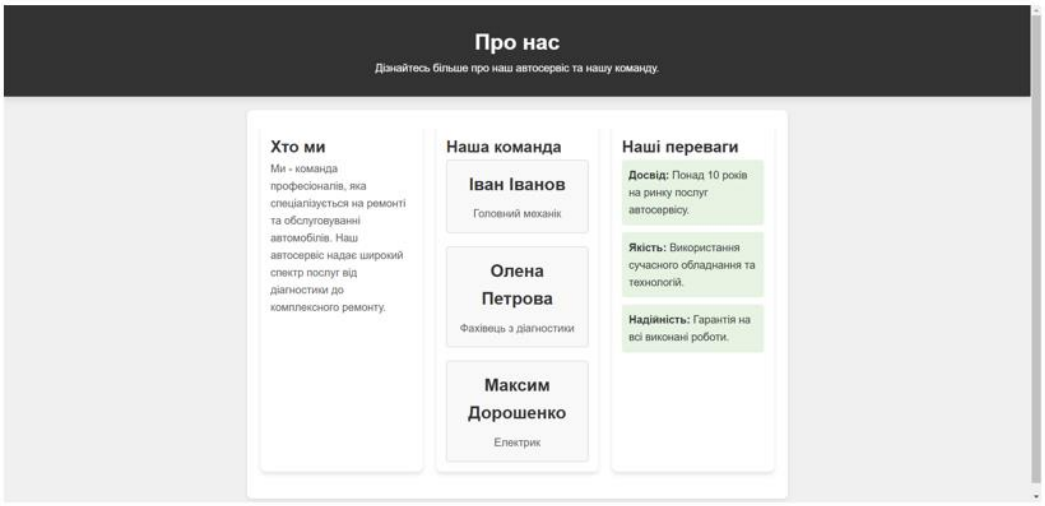

Рисунок 16.1 – Інтерфейс вкладкі про нас

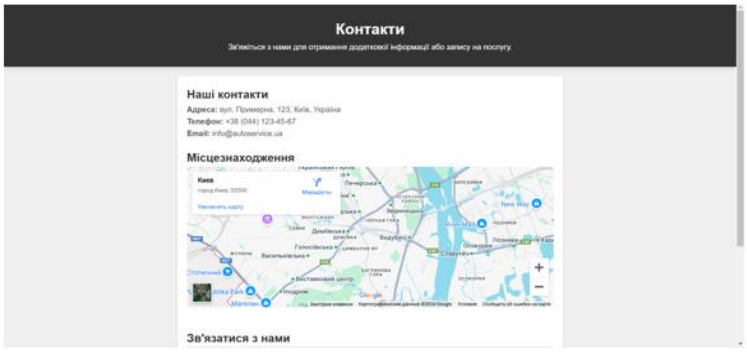

Рисунок 16.1 – Інтерфейс вкладкі контакти

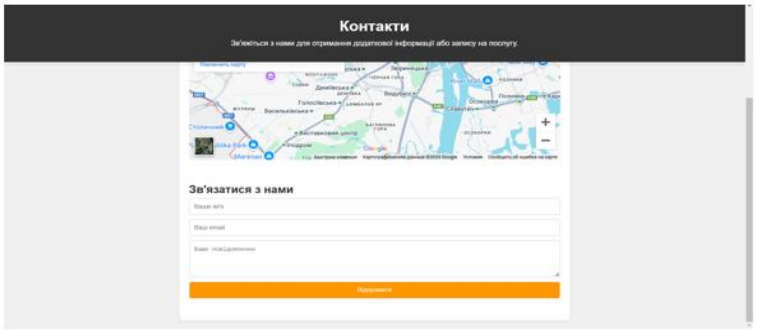

Рисунок 16.2 – Інтерфейс вкладкі контакти продоваження

Розроблений сайт планшетна версія

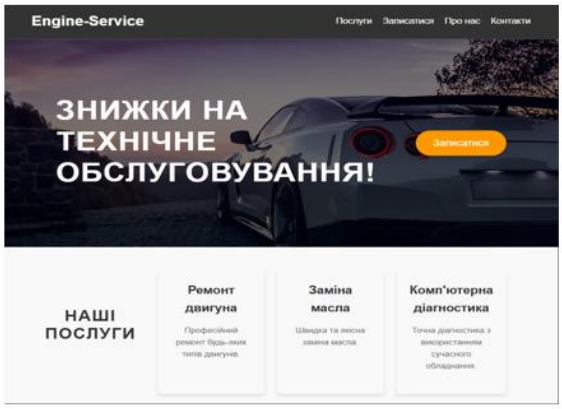

Рисунок 18.1 – Головна сторінка

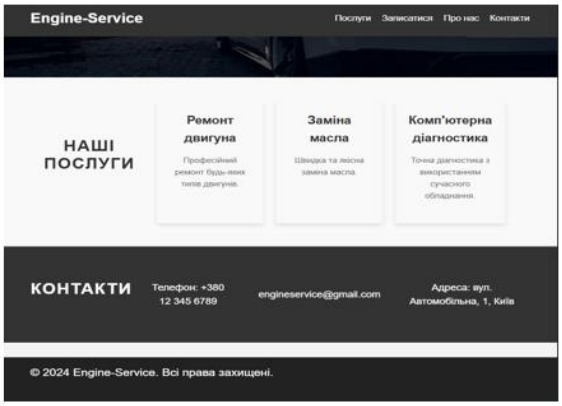

Рисунок 18.2 – Головна сторінка продовження

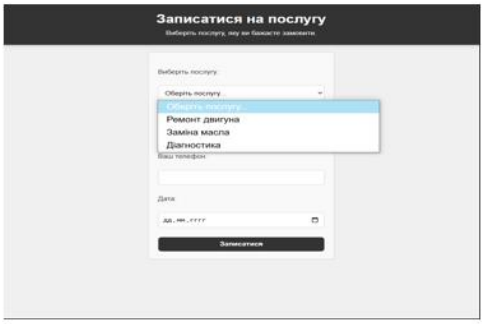

Рисунок 18.3 – Записатися на послугу

Розроблений сайт планшетна версія

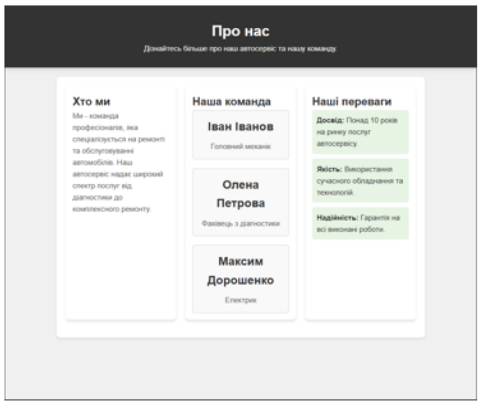

Рисунок 19.1 – Інтерфейс про нас

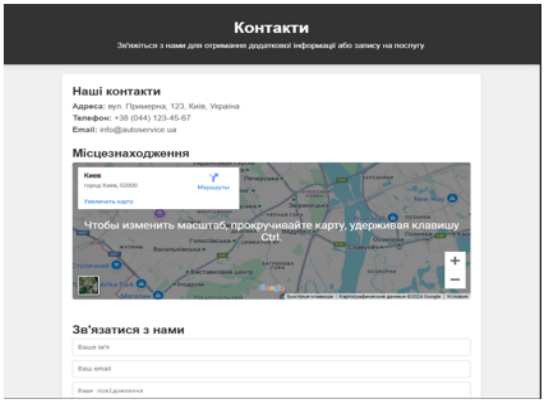

Рисунок 19.2 – Інтерфейс контакти

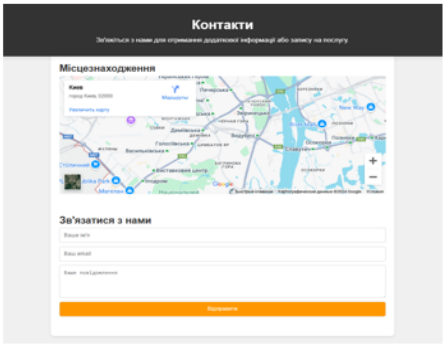

Рисунок 19.3 – Інтерфейс контакти продовження

Розроблений сайт планшетна версія

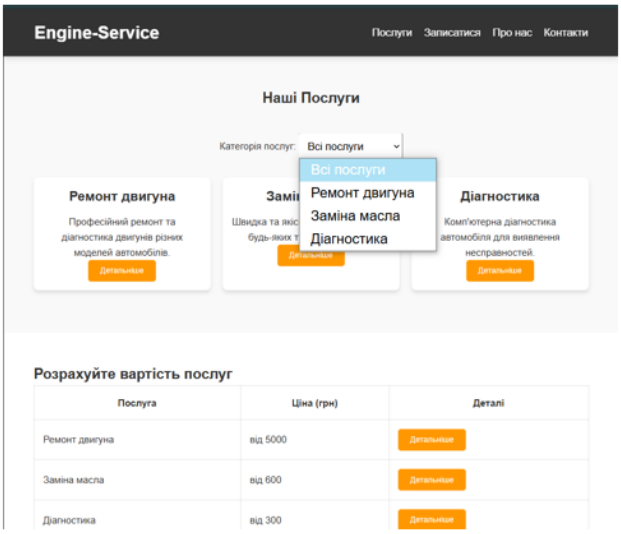

Рисунок 20.1 – Інтерфейс наші послуги

Розроблений сайт смартфон версія

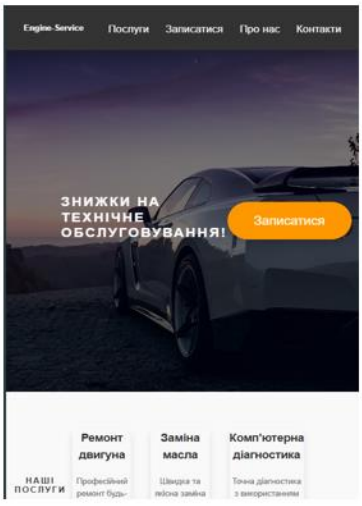

Рисунок 21.1 – Інтерфейс Головна сторінка

Рисунок 21.2 – Інтерфейс Головна сторінка продовження

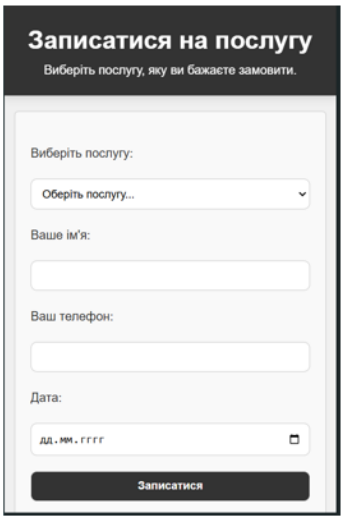

Рисунок 22.1 – Інтерфейс записатися на послугу

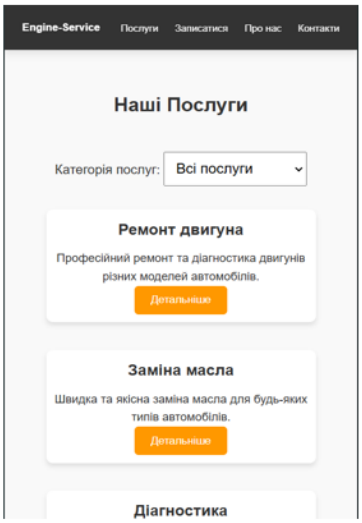

Рисунок 22.2 – Інтерфейс наші послуги

Розроблений сайт смартфон версія

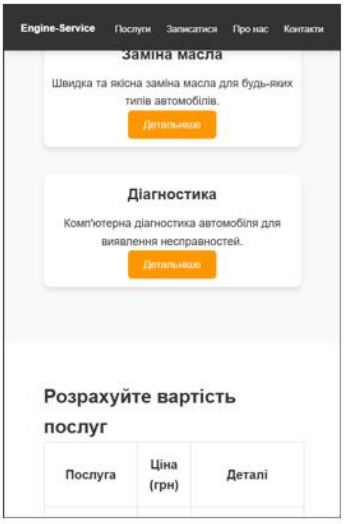

Рисунок 23.1 – Інтерфейс записатися на послугу продовження

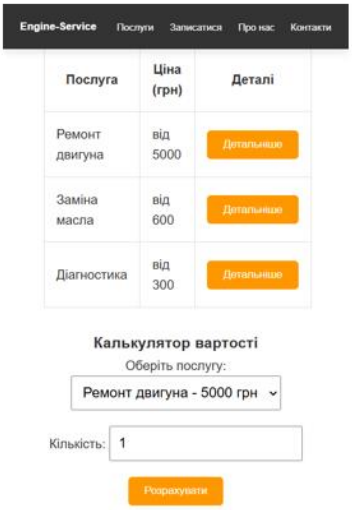

Рисунок 23.2 – Інтерфейс записатися на послугу продовження

Розроблений сайт смартфон версія

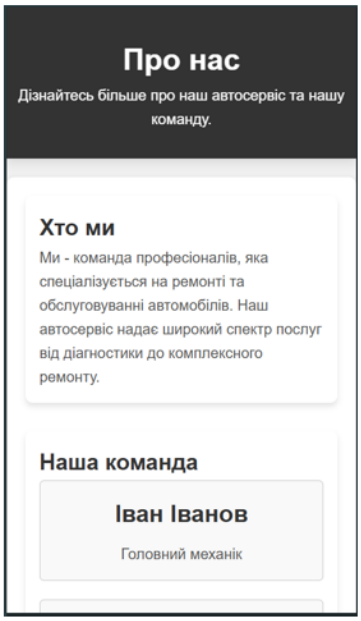

Рисунок 24.1 – Інтерфейс про нас

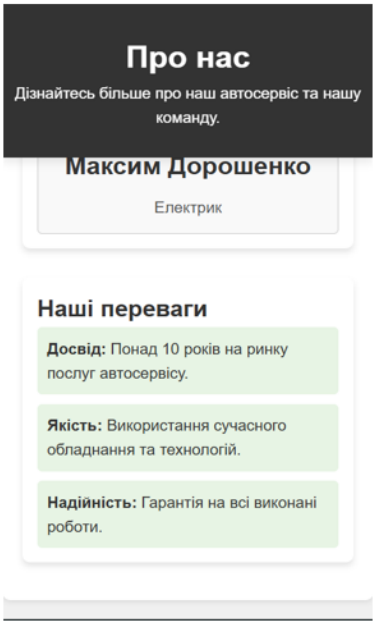

Рисунок 24.2 – Інтерфейс про нас продовження

Розроблений сайт смартфон версія

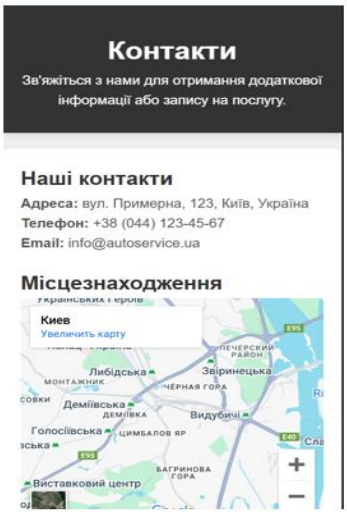

Рисунок 25.1 – Інтерфейс контакти

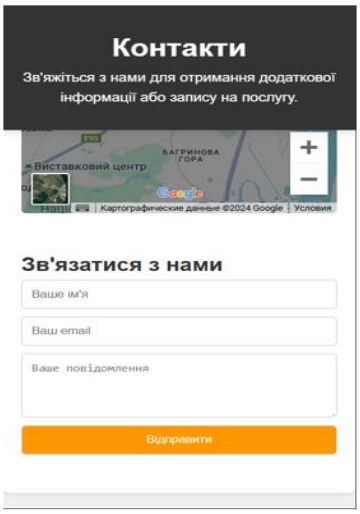

Рисунок 25.2 – Інтерфейс контакти продовження