ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«ВЕБ-САЙТ ДЛЯ СЕРВІСНОГО ЦЕНТРУ РЕМОНТУ СМАРТФОНІВ»

на здобуття освітнього ступеня бакалавр
за спеціальності 126 Інформаційні системи та технології
(код,найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

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

_____ Олег Чудаков
(підпис) *(ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти
група ІСД-42 Олег Чудаков
(ім'я, ПРІЗВИЩЕ)
Керівник Ольга ПОЛОНЕВИЧ
к.т.н. доцент *(ім'я, ПРІЗВИЩЕ)*
Рецензент: _____
(ім'я, ПРІЗВИЩЕ)

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна прорама Інформаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедрою ІСТ
_____ Каміла СТОРЧАК

“____” _____2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Чудакову Олегу Сергійовичу
(прізвище, ім’я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Веб-сайт для сервісного центру ремонту смартфонів
керівник кваліфікаційної роботи: Ольга ПОЛОНЕВИЧ к.т.н., доцент
(ім’я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від “24” лютого 2025 р. № 56

2. Строк подання кваліфікаційної роботи «30» травня 2025 р.

3. Вихідні дані кваліфікаційної роботи:
1. Принципи побудови веб застосунків на основі технологій Next.js та Firebase.
2. Архітектура та логіка реалізації сервісів обробки заявок та сповіщень.
3. Аналітичні та науково-технічні джерела щодо цифровизації послуг в Україні.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
1. Аналіз цифрової трансформації ринку сервісного обслуговування та огляд існуючих рішень.
2. Визначення цільової аудиторії та обґрунтування структури веб платформи.
3. Розробка та реалізація веб сайту з інтеграцією Firebase, Telegram-ботом та системою статусного сповіщення.

5.Перелік ілюстраційного матеріалу: *презентація*

6. Дата видачі завдання «24» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Аналіз цифровизації в Україні та актуальності сервісних онлайн-платформ	24.02-03.03.25	
2.	Дослідження конкурентних рішень і визначення цільової аудиторії	04.03-12.03.25	
3.	Постановка мети, завдань дослідження та обґрунтування технологій (Next.js, Firebase, тощо)	13.03-17.03.25	
4.	Проєктування структури веб сайту та побудова логіки взаємодії користувача із системою	18.03-28.03.25	
5.	Реалізація основного функціоналу: авторизація, подання заявок, статуси, особистий кабінет	29.03-15.04.25	
6.	Інтеграція з Firebase та Telegram-ботом. Реалізація панелі адміністратора	16.04-25.04.25	
7.	Тестування системи, адаптивність, налагодження функціоналу	26.04-04.05.25	
8.	Підготовка демонстраційних матеріалів і завершення звітної частини	05.05-20.05.25	
9.	Оформлення та подання бакалаврської роботи	21.05-30.05.25	

Здобувач вищої освіти _____ Олег Чудаков
(підпис) *(ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Ольга ПОЛОНЕВИЧ
(підпис) *(ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 66 стор., 40 рис., 8 джерел.

Мета роботи – розробка повнофункціонального веб сайту Remontika, що забезпечує онлайн-оформлення, обробку та відстеження заявок на ремонт мобільних пристроїв, із підтримкою сповіщень через месенджери та адміністративної панелі.

Об'єкт дослідження – цифрова взаємодія користувача з сервісними платформами.

Предмет дослідження – програмна реалізація вебплатформи для ремонту техніки з інтеграцією сповіщень, бази даних і адміністративної панелі.

Короткий зміст роботи. У бакалаврській роботі представлено аналіз цифровізації сервісної галузі в Україні та досліджено потреби сучасного користувача в сфері ремонту мобільних пристроїв. Здійснено огляд існуючих рішень на ринку та визначено їх обмеження. На основі цього обґрунтовано архітектуру власного веб сайту Remontika. Реалізовано авторизацію користувачів, особистий кабінет, подачу заявок, їхню обробку адміністраторами, а також систему автоматичного статусного оновлення із відправкою повідомлень через Telegram. Веб платформа створена з використанням сучасного стеку технологій: Next.js, Firebase, Tailwind CSS. Проведено тестування основного функціоналу та розроблено рекомендації для подальшого вдосконалення.

КЛЮЧОВІ СЛОВА: РЕМОНТ СМАРТФОНІВ, ВЕБ САЙТ, FIREBASE, NEXT.JS, СПОВІЩЕННЯ TELEGRAM, ОСОБИСТИЙ КАБІНЕТ, ЗАЯВКА НА РЕМОНТ.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному цифровому світі користувачі очікують максимальної зручності, швидкості та прозорості під час отримання побутових послуг, зокрема — сервісного обслуговування мобільної техніки. У зв’язку з активною цифровізацією в Україні, стрімким зростанням кількості користувачів мобільного інтернету та звичкою вирішувати більшість повсякденних задач через онлайн-сервіси, виникає потреба у створенні інтуїтивно зрозумілих і функціонально повних веб платформ. Сервіси ремонту смартфонів, які донедавна працювали здебільшого в офлайн-режимі, мають адаптуватися до нових умов — забезпечити клієнтам подачу заявки онлайн, відстеження статусу ремонту, зручну комунікацію через месенджери та можливість взаємодії без фізичної присутності.

Мета роботи – розробка повнофункціонального веб сайту Remontika, що забезпечує онлайн-оформлення, обробку та відстеження заявок на ремонт мобільних пристроїв, із підтримкою сповіщень через месенджери та адміністративної панелі.

Для досягнення мети у бакалаврській роботі виконано наступні завдання:

- Проаналізовано сучасний стан цифровізації в Україні у сфері побутових сервісів;
- досліджено приклади існуючих вітчизняних та закордонних рішень;
- визначено потреби цільової аудиторії шляхом кейс-аналізу;
- реалізовано клієнтську частину сайту з адаптивним дизайном;
- налаштовано збереження та обробку даних за допомогою Firebase;
- реалізовано систему Telegram-сповіщень;
- розроблено адміністративну панель для опрацювання заявок.

Об’єкт дослідження – процес цифрової взаємодії між клієнтом і сервісним центром.

Предмет дослідження – архітектура веб застосунку для автоматизації сервісного обслуговування.

Методи дослідження. Під час виконання роботи використовувались методи аналізу цільової аудиторії, моделювання користувацьких сценаріїв, прототипування інтерфейсу, робота з базами даних, а також інструменти веброзробки, такі як HTML, Next.js, Tailwind CSS, Firebase та Telegram API.

Наукова новизна одержаних результатів. У роботі запропоновано рішення, що об’єднує зручну подачу заявок, адміністративне управління статусами та систему автоматичних сповіщень користувачів у режимі реального часу через месенджери, що не було реалізовано в аналогічних вітчизняних проєктах.

Практична значущість. Розроблений веб сайт Remontika може бути використаний як повноцінна онлайн-платформа сервісного центру. Його структура дозволяє масштабування функціоналу, розширення каналів зв’язку з клієнтом та впровадження додаткових бізнес-логік.

1 ПЕРЕДУМОВИ, АНАЛІЗ І ПОСТАНОВКА ЗАДАЧІ ЦИФРОВОГО СЕРВІСУ З РЕМОНТУ ТЕХНІКИ

1.1 Актуальність теми

У XXI столітті цифровізація стала однією з провідних рушійних сил трансформації суспільства, економіки та державного управління. Україна, як частина глобального інформаційного простору, активно впроваджує цифрові технології в усі сфери життя, включаючи комунікацію, банківські послуги, освіту, медицину та, що особливо важливо в контексті цієї роботи, сферу сервісного обслуговування та ремонту техніки. Цифровізація в Україні набула особливої актуальності після 2014 року, коли країна обрала курс на глибоку трансформацію державних і комерційних процесів. Із запуском програм «Держава у смартфоні», порталу «Дія», електронного документообігу, а також активною підтримкою стартапів та IT-сектору, цифрові сервіси почали охоплювати все ширше коло життєвих ситуацій громадян. За даними Міністерства цифрової трансформації України, станом на 2024 рік понад 19 мільйонів українців (Рис 1.1) активно користуються мобільними додатками та онлайн-платформами для отримання адміністративних, фінансових і побутових послуг [1]. Така масовість формує очікування користувачів щодо швидкого, зручного та прозорого доступу до будь-якого сервісу в кілька кліків, зокрема й до послуг з ремонту мобільних пристроїв.

Рис.1.1 Загальна статистика українців по цифровизації

Ринок побутових цифрових послуг, до яких належать сервіси ремонту смартфонів, теж переживає активну фазу цифрової трансформації. Якщо раніше типовий клієнт змушений був особисто приходити до сервісного центру, вистоювати чергу й очікувати попередній діагноз, то сьогодні, в умовах високої мобільності та обмеженого часу, зростає попит на сервіси, які дозволяють здійснювати ці дії онлайн. Люди очікують можливості надіслати заявку через сайт або мобільний додаток, отримати підтвердження, слідкувати за процесом ремонту в режимі реального часу, а також спілкуватися з менеджером у зручному цифровому середовищі — через месенджери або SMS. Важливо підкреслити, що така потреба сформована не лише підвищенням цифрової грамотності населення, а й зміною споживчих звичок: сучасний користувач надає перевагу автоматизованому сервісу без паперової тяганини, телефонних дзвінків і тривалого очікування.

Пандемія COVID-19 стала додатковим каталізатором розвитку онлайн-сервісів у всьому світі, й Україна не стала винятком. Саме в цей період багато сервісних компаній були змушені адаптуватися до нових умов і впровадити хоча б базову онлайн-підтримку. Проте лише частина з них пішла далі, створивши повноцінні платформи для обслуговування клієнтів у цифровому форматі. Більшість же зупинилася на напівручних рішеннях — заповнення форм на сайті з подальшим телефонним дзвінком. У таких умовах поява повноцінного інтерактивного вебсайту для ремонту смартфонів із функціями онлайн-подачі заявок, відстеження статусу ремонту, інтеграції з месенджерами та адаптивним дизайном стає не просто сучасним рішенням, а відповіддю на реальні потреби споживача.

Сучасний темп життя, зростання кількості цифрових пристроїв і залежність від безперервного доступу до мобільного зв’язку створюють особливо високий попит на швидкі, якісні та передбачувані послуги ремонту. При цьому користувачі все частіше віддають перевагу тим сервісам, що дозволяють вирішити питання без фізичного контакту, у дистанційному форматі, із можливістю взаємодії через сайт і отримання миттєвих сповіщень. З огляду на ці обставини, створення вебсайту

Remontika є не лише відповіддю на глобальні тенденції цифровізації, а й стратегічно важливим кроком у побудові клієнтоорієнтованого сервісу нового покоління, що повністю відповідає вимогам часу.

Згідно з дослідженнями міжнародної компанії DataReportal, станом на початок 2024 року рівень проникнення мобільного інтернету в Україні становив понад 80% серед дорослого населення, що свідчить про надзвичайну активність використання смартфонів як основного інструменту доступу до онлайн-середовища. При цьому, за даними щорічного звіту компанії We Are Social, середньостатистичний український користувач проводить у мобільному інтернеті понад 3,5 години щодня, а загальний обсяг мобільного трафіку зростає щороку на 15–20% [2]. Таке стрімке зростання активності створює навантаження на пристрої, що безпосередньо впливає на частоту поломок та потребу в технічному обслуговуванні.

Водночас, близько 32% українських користувачів стикалися з необхідністю ремонту смартфону щонайменше один раз протягом останнього року, при цьому найбільш поширеними поломками залишаються тріщини на екрані, проблеми з зарядкою та акумулятором, а також несправності, пов’язані з оновленням програмного забезпечення або падінням пристрою у воду. Середній термін використання одного смартфона в Україні становить 2–3 роки, після чого або відбувається заміна, або виникає потреба в ремонті. Це створює стабільно високий попит на послуги сервісних центрів, причому найбільш активно до них звертаються мешканці міст із населенням понад 100 тисяч осіб.

Щодо розвитку самого ринку сервісного обслуговування мобільної техніки, експерти консалтингової компанії BRDO (Better Regulation Delivery Office) вказують, що ця галузь є однією з найбільш динамічних у сфері побутових послуг в Україні. У 2023 році в країні діяло понад 4 тисячі офіційно зареєстрованих сервісних центрів і приватних майстерень, з яких понад 70% надавали послуги саме з ремонту смартфонів. Прогнозується, що обсяг ринку в грошовому вираженні у 2024–2025 роках перевищить 3,2 млрд грн на рік, враховуючи як приватні звернення, так і корпоративне обслуговування техніки. Особливо помітним є тренд

на автоматизацію процесів, спрощення подачі заявок і інтеграцію цифрових каналів комунікації.

На тлі цих даних можна зробити висновок, що зростання мобільного трафіку, постійна фізична зношеність пристроїв та загальний рівень цифровізації формують сталу основу для розвитку сервісів ремонту смартфонів, особливо тих, що здатні запропонувати користувачам не лише якісне обслуговування, а й зручність, швидкість та прозорість усіх етапів комунікації. Таким чином, створення сучасної онлайн-платформи, яка дозволяє автоматизувати прийом і обробку заявок, а також забезпечити миттєвий цифровий зворотний зв'язок, є логічною відповіддю на тенденції ринку та реальні очікування українського споживача.

1.2 Аналіз існуючих рішень

У контексті порівняльного аналізу існуючих рішень на українському ринку ремонту смартфонів розглянемо три популярні онлайн-платформи: «Ай-Яй-Яй» (iya-yai.kiev.ua), «F1Center» (f1center.ua) та «Skeleton» (skeleton.ua). Ці сервіси обрано з огляду на їхню популярність, наявність впізнаваного бренду в межах Києва та наявність активних вебсайтів, які позиціонуються як сучасні платформи із прийому замовлень на ремонт цифрової техніки.

Сайт iya-yai.kiev.ua (Рис. 1.2) позиціонує себе як «мережа чесного сервісу» та візуально наголошує на великому досвіді: компанія працює з 2012 року, має понад 50 000 обслугованих клієнтів і понад 10 000 онлайн-відгуків. Основний інтерфейс сайту побудований за традиційною моделлю лендингу з вертикальною прокруткою. Присутні блоки з інформацією про гарантії, навчання, франшизу, а також форму замовлення дзвінка. Водночас, сайт не пропонує повноцінної системи онлайн-подачі заявки з можливістю обрати тип несправності чи відстежувати статус ремонту. Навігація базується на переході за категоріями брендів (Apple, Samsung, Xiaomi тощо), але далі користувач має або зателефонувати, або залишити заявку в спрощеній формі. Із позитивних аспектів слід відзначити адаптивний

дизайн і зручний пошук моделей, однак функціональна взаємодія обмежується лише базовим рівнем контакту.

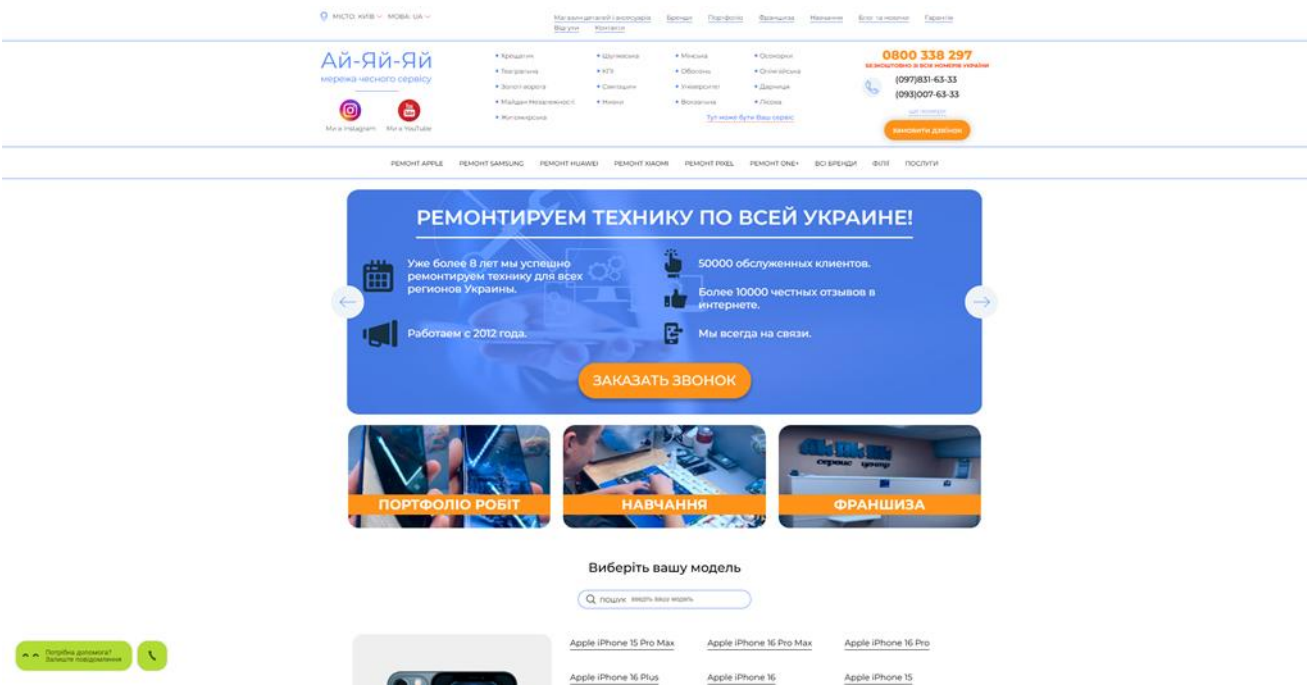

Рис.1.2 Веб-сайт «Ай-Яй-Яй».

Сайт f1center.ua (Рис 1.3) виглядає більш стримано й професійно з візуальної точки зору. Інтерфейс реалізовано з урахуванням корпоративного стилю: темна кольорова палітра, чітка структура розділів (гарантійний ремонт, післягарантійний, адресна доставка тощо), карта розташування відділень та кнопка для відстеження або оплати ремонту. Хоча сам факт наявності кнопки «відстежити» створює враження цифрової інтеграції, при спробі перейти користувача перекидає до окремої форми з номером замовлення, що передбачає лише односторонню перевірку статусу без особистого кабінету чи історії звернень. Відсутні месенджери, інтерактивні сповіщення чи візуальний прогрес бар ремонту. Форма заявки також реалізована в спрощеному вигляді без логіки вибору моделі пристрою чи типу поломки. Таким чином, сервіс є більш надійним із точки зору іміджу, однак цифрова складова сервісного досвіду досить обмежена.

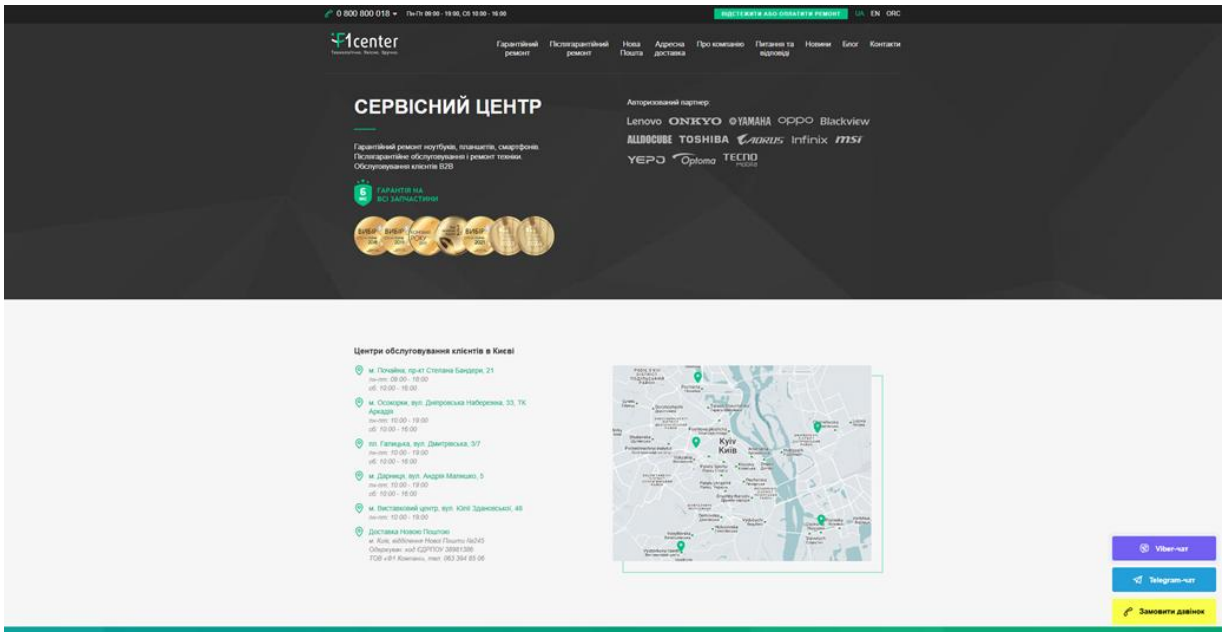

Рис.1.3 Веб-сайт «F1Center».

Сайт skeleton.ua (Рис. 1.4) суттєво відрізняється стилістикою. Його дизайн побудовано на контрастних кольорах, з агресивним копірайтингом («Зламався, але не ти!»), що націлений на молоду аудиторію. Вебсторінка має дуже лаконічну структуру: кілька коротких текстових блоків, кнопка «Заявка на ремонт» та виклик кур’єра. Усе зосереджено навколо форми замовлення, але без розширених функцій чи індивідуального кабінету. Відсутній опис етапів ремонту, немає відстеження процесу, не реалізовано інтеграцію з SMS або месенджерами. Водночас приваблює візуальна легкість та сміливість у побудові іміджу, що може бути ефективним для певного сегменту аудиторії. Однак, як платформа для постійної взаємодії з користувачем, Skeleton не надає достатнього функціоналу і виглядає швидше як онлайн-вітрина.

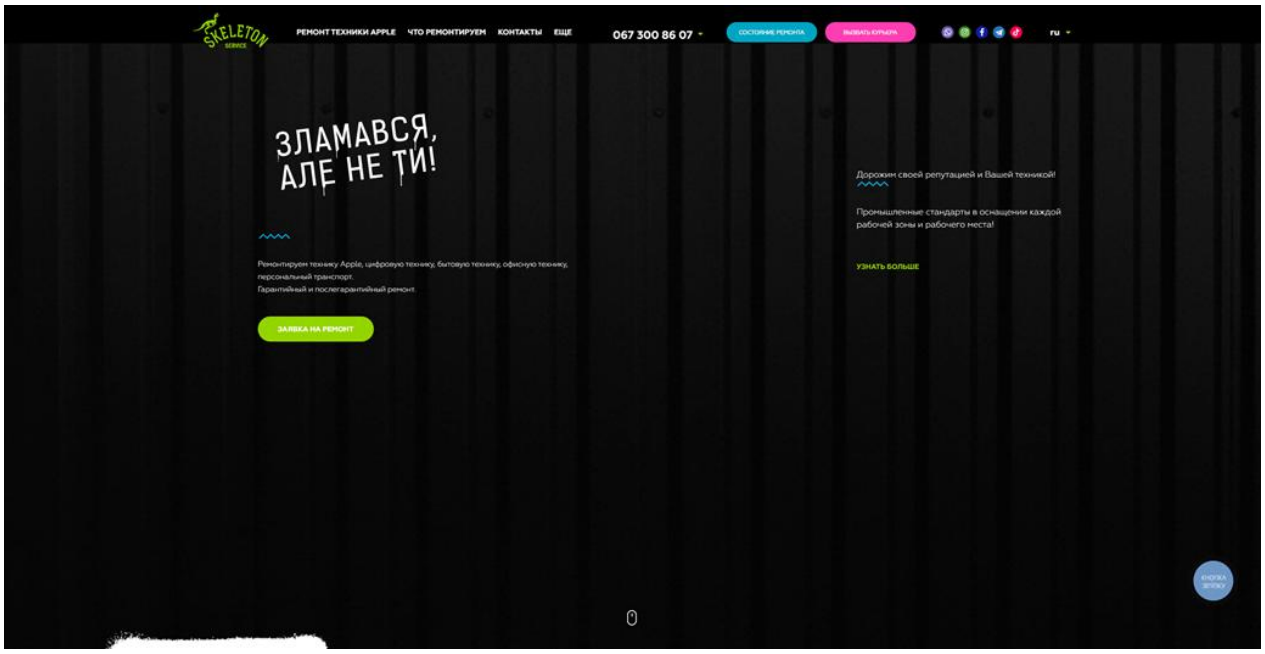

Рис.1.4 Веб-сайт «Skeleton».

Узагальнюючи, можна стверджувати, що жоден з трьох розглянутих сервісів не реалізує повного циклу цифрової взаємодії з клієнтом: подача заявки, динамічне відстеження статусу, отримання повідомлень, збереження історії звернень — все це або відсутнє, або реалізовано на базовому рівні. Таким чином, проєкт Remontika із глибокою інтеграцією Firebase, підтримкою Telegram/Viber/SMS, особистим кабінетом і повноцінною логікою заявок вигідно вирізняється на тлі українських конкурентів і відповідає сучасним очікуванням цифрового користувача.

У контексті порівняльного аналізу існуючих рішень на українському ринку доцільно розширити рамки дослідження, включивши приклади успішних закордонних платформ, які демонструють високий рівень цифровізації сервісного обслуговування техніки. Це дозволяє не лише окреслити стандарт якості, до якого варто прагнути, але й обґрунтувати конкурентні переваги локального проєкту Remontika, що у своїй реалізації орієнтується на найкращі міжнародні практики.

Однією з найбільш відомих платформ у цьому сегменті є uBreakiFix (Рис 1.5), сервісна мережа зі США, яка функціонує як в онлайні, так і через фізичні точки обслуговування. Сайт ubreakifix.com забезпечує користувачеві можливість у кілька

кліків створити заявку на ремонт, вказавши модель пристрою, тип несправності, бажаний спосіб доставки (кур'єр, самостійне занесення або доставка поштою) та отримати попередню вартість послуги. Окрім того, платформа реалізує детальне відстеження процесу ремонту в режимі реального часу та систему оцінювання сервісу після завершення обслуговування. Користувачі мають доступ до історії заявок через особистий кабінет, а також отримують повідомлення на e-mail та в мобільному застосунку. UBreakiFix використовує елементи гейміфікації в кабінеті користувача, стимулюючи залишати відгуки, що формує додаткову довіру. Цей рівень цифрової автоматизації досі не реалізований в українських реаліях.

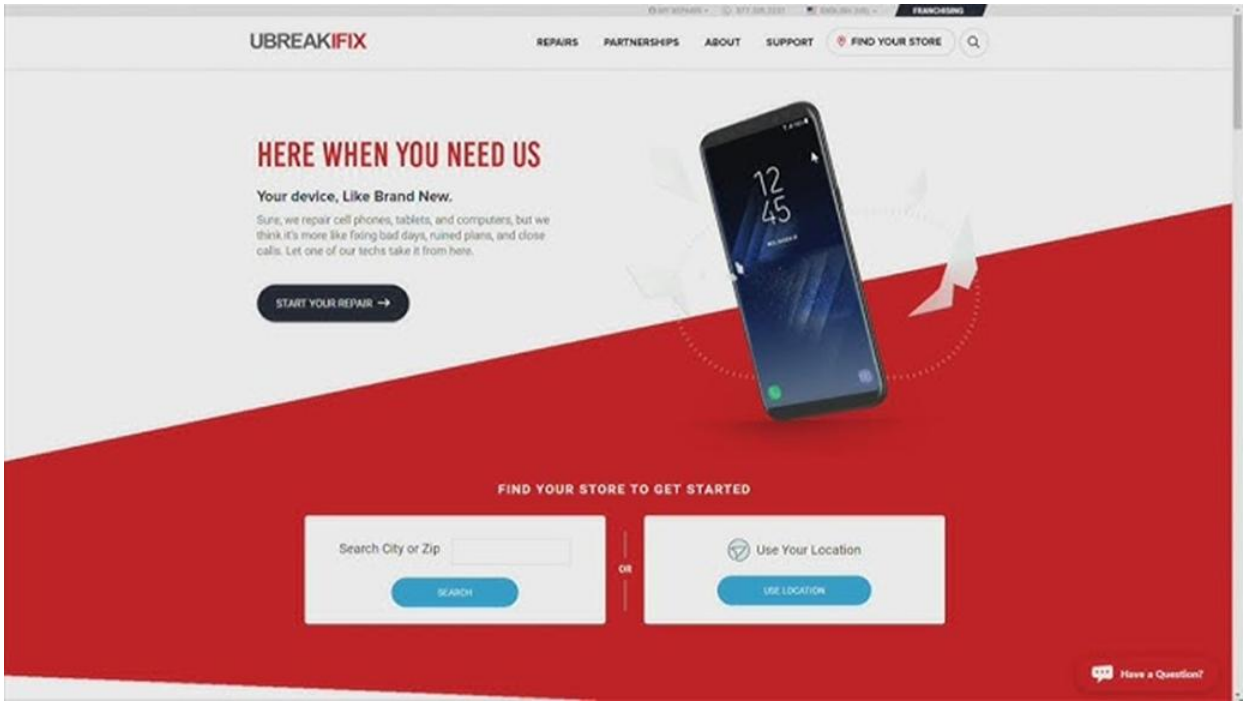

Рис.1.5 Веб-сайт uBreakiFix, сервісна мереже ремонту смартфонів.

Ще одним прикладом є iSmash (Рис. 1.6) — британський сервіс, який спеціалізується на терміновому ремонті смартфонів, планшетів і ноутбуків. На сайті ismash.com користувач може миттєво вибрати тип пристрою, зазначити пошкодження з детальним вибором (наприклад, розбитий екран, несправність зарядного порту, проблема з Face ID), обрати найближчий сервісний пункт на мапі або викликати кур'єра. Весь процес замовлення — від вибору послуги до запису на точну годину — відбувається онлайн. Важливим елементом є наявність блоку «Live

Repair Tracker», який дозволяє відстежувати статус заявки у форматі “Ready for Technician”, “Repair in Progress”, “Quality Control”, “Ready for Pickup”. Це не лише інформативно, а й створює ефект професійного та контрольованого процесу. Інтерфейс сайту адаптований для мобільних пристроїв, а UX-рішення акцентують на швидкості й мінімальній кількості кліків.

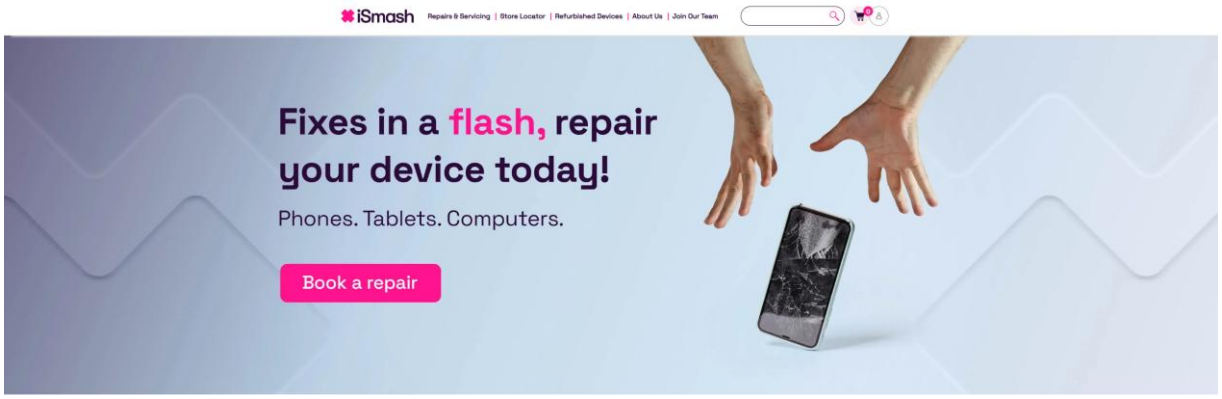

Рис.1.6 Веб-сайт iSmash – британський сервіс ремонту техніки.

Успішним прикладом із Європи є сервіс MediaMarkt Repair Service, інтегрований у загальну платформу одного з найбільших рітейлерів техніки в Німеччині. Особливістю цієї моделі є тісна прив’язка до електронної комерції: користувачі можуть оформити заявку на ремонт прямо зі сторінки купленого товару, використовуючи унікальний ID замовлення. Водночас реалізовано глибоку інтеграцію з особистим кабінетом і системою гарантійного обслуговування. MediaMarkt дозволяє оформити ремонт як фізично (в магазині), так і повністю онлайн, з викликом логістичної служби. Система сама обирає оптимальний сервісний центр у залежності від типу пристрою, регіону та завантаженості. Користувач отримує підтвердження статусу кожного етапу — e-mail, SMS або push-сповіщенням.

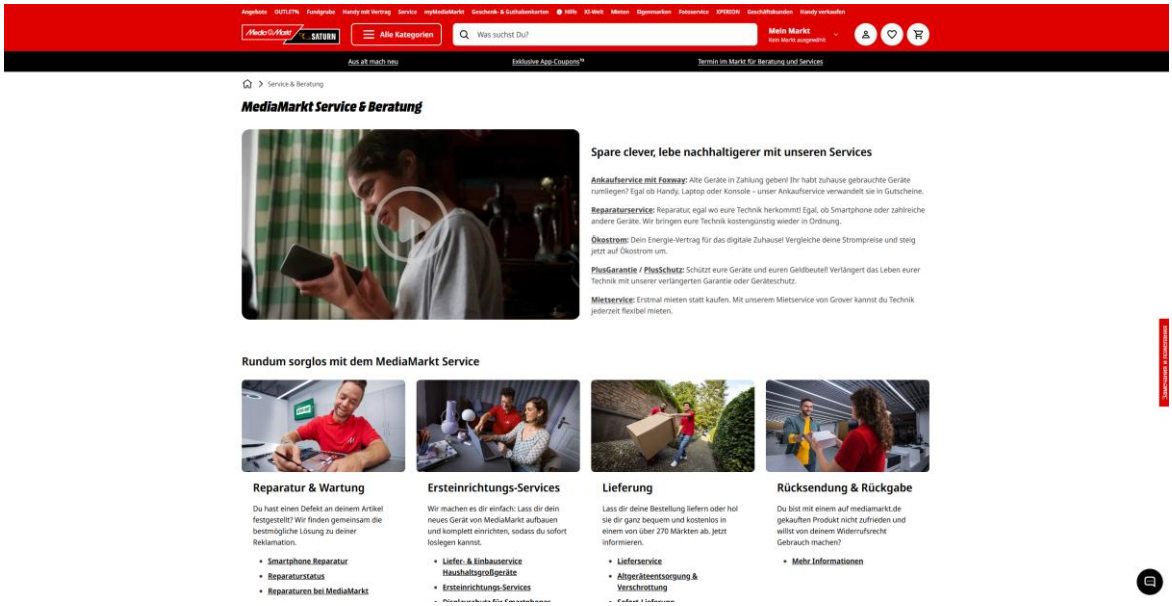

Рис.1.7 Сервіс ремонту смартфонів одного з найбільших рітейлерів техніки в Німеччині.

Загалом, усі три розглянуті закордонні приклади демонструють високий рівень цифрової автоматизації, адаптивності, персоналізації користувацького досвіду та глибокої інтеграції з логістичними та CRM-системами. На цьому тлі проєкт Remontika, що включає в себе онлайн-заявку, статусне відстеження, інтеграцію з Firebase, повідомлення через Telegram/Viber/SMS та адаптивний дизайн, фактично імплементує подібні підходи в українські реалії, з урахуванням локальної специфіки й звичок користувачів.

1.3 Цільова аудиторія та її потреби: кейс орієнтований підхід

У сучасній практиці дослідження користувачів активно застосовується метод кейс-стаді (англ. case study) — це структурований підхід до аналізу поведінки, потреб та мотивацій цільових груп на основі моделювання типових користувацьких сценаріїв. Кожен кейс-стаді дозволяє розглянути конкретну умовну людину або групу зі схожими характеристиками, яка стикається з сервісом у реальних життєвих обставинах. На відміну від абстрактних сегментів ринку, кейс-стаді дає змогу глибше розуміти контекст користування, джерела болю (pain points), очікування та

потенційні бар’єри. Такий підхід широко використовується в UX-дизайні, маркетингу, розробці цифрових сервісів і допомагає створити продукт, орієнтований на людину.

Для веб-сайту Remontika було обрано три найбільш релевантні кейси, що разом охоплюють широке коло потенційних користувачів. Це: молодь віком 17–25 років (так звані цифрові нативи), активні професійні користувачі віком 26–54 роки (самозайняті особи, працівники малого бізнесу, службовці, спеціалісти, фрилансери) та старші користувачі віком 55+, які частково адаптувалися до цифрових послуг. Важливо зазначити, що розподіл не є виключно віковим — він побудований за стилем користування, рівнем цифрової впевненості та роллю смартфона в житті кожного сегмента.

Першу групу становлять студенти й молодь віком 17–25 років, які є активними цифровими користувачами і сприймають смартфон не просто як засіб зв’язку, а як базовий інструмент у всіх сферах життя: навчання, банкінг, комунікація, розваги, логістика, держпослуги. Для них важливими є швидкість, мінімальна кількість кроків, автоматизація дій, адаптивність до мобільного пристрою, а також наявність сучасного інтерфейсу без перевантаження. Їхні болючі точки — обмежений бюджет, відсутність часу на особисте відвідування сервісу, недовіра до телефонних консультацій, страх "нарватися на шахрайство". Молодь очікує зручного способу онлайн-подачі заявки з мобільного, інтеграції з Telegram або Viber, прозорої системи оновлень щодо статусу ремонту.

Другу групу представляють професійно активні користувачі віком 26–54 роки — це працівники сфери обслуговування, логістики, IT, адміністратори малого бізнесу, приватні підприємці, маркетологи, фрилансери. Для цієї групи смартфон є не просто засобом зв’язку, а ще й повноцінним інструментом роботи. Втрачений або несправний пристрій — це прямі втрати доходу, збої в комунікації з клієнтами, невиконані замовлення або зірвані дедлайни. Найболючіші точки — це затримки в ремонті, неможливість точно знати строк виконання, складнощі із контролем за заявками, а також брак персоналізованої підтримки. Такі користувачі очікують швидкого, передбачуваного, інтерактивного процесу: з можливістю точно

заповнити тип несправності, побачити орієнтовну вартість, вибрати зручний час, отримати інформування через месенджер або SMS, бачити всі попередні звернення в особистому кабінеті та за потреби завантажити рахунок-фактуру для звітності. Цей сегмент надзвичайно чутливий до часу, прозорості процесу та професійної комунікації.

Третю групу становлять користувачі віком 55+, які за останнє десятиліття стали активними власниками смартфонів, переважно через потребу користування державними сервісами (наприклад, застосунок «Дія»), банківськими послугами або зв’язком із рідними. Проте рівень цифрової грамотності цієї аудиторії залишається помірним або низьким. Їхні страхи — втратити дані, бути ошуканими, не розібратися в складному інтерфейсі. Ці люди переважно не користуються Telegram або іншими новітніми інструментами зв’язку, але орієнтуються на SMS або телефонний дзвінок. Їм складно зареєструватися, запам’ятовувати логіни та паролі, тому важливо, щоб сайт пропонував варіант «запису без авторизації», «замовлення зворотного дзвінка», а також максимально спрощену форму подачі заявки — без надмірних полів, з великими кнопками, чіткими інструкціями, адаптованими для зору старших користувачів. Їхнє очікування — чітка комунікація, простота, можливість отримати людське пояснення, а не лише технічні терміни.

У результаті застосування методу кейс-стаді в процесі розробки сайту Remontika вдалося чітко і ґрунтовно ідентифікувати основні сегменти цільової аудиторії, проаналізувати їхні повсякденні сценарії користування, типові проблеми та очікування від сервісу. Саме цей підхід став фундаментом для формування логіки інтерфейсу, адаптації структури форм під різні стилі взаємодії, впровадження багатоканальної комунікації (через телефон, SMS, месенджери) та побудови персоналізованого кабінету. Усе це дозволило створити сервісну модель, яка одночасно відповідає потребам молоді, професійно активних користувачів і старшого покоління.

1.4 Мета та завдання дослідження

Метою даного дослідження є розробка функціонального, адаптивного та інтуїтивно зрозумілого вебсайту сервісу ремонту смартфонів Remontika, який відповідає сучасним вимогам цифрового споживача в Україні. Цей сайт має не лише автоматизувати ключові процеси взаємодії з клієнтом — подачу заявки, сповіщення, відстеження статусу ремонту — а й забезпечити високий рівень персоналізації сервісу на основі реального аналізу потреб цільової аудиторії. Основна увага приділяється впровадженню рішень, що базуються на сучасних технологіях (Next.js, Firebase, Tailwind CSS) та методах побудови користувацького досвіду, включаючи моделювання через кейс-стаді.

Для реалізації цієї мети необхідно виконати такі завдання дослідження: здійснити аналіз актуального стану цифровізації в Україні в контексті ринку сервісного обслуговування мобільної техніки; вивчити досвід національних та міжнародних конкурентів з метою виявлення недоліків у реалізації онлайн-сервісів ремонту; провести деталізований аналіз цільової аудиторії за допомогою методу кейс-стаді, що дозволить адаптувати логіку та функціональність сайту до реальних очікувань користувачів; обґрунтувати вибір технологій та архітектурних рішень для реалізації вебплатформи; побудувати логічну структуру проєкту з урахуванням взаємодії клієнтської частини, бекенду та бази даних; розробити адаптивний інтерфейс користувача відповідно до UX/UI-принципів, враховуючи окремі сценарії для різних категорій користувачів; реалізувати механізм інтерактивного відстеження статусу ремонту, з можливістю багатоканального сповіщення через Telegram, Viber або SMS; забезпечити перевірку працездатності всіх функцій шляхом тестування, а також підготувати пропозиції щодо подальшого розвитку платформи в перспективі масштабування.

2 АРХІТЕКТУРА СИСТЕМИ ТА ТЕХНОЛОГІЇ РЕАЛІЗАЦІЇ

2.1 Огляд використовуваних технологій і інструментів

У процесі створення вебсайту Remontika ключову роль відіграють базові технології веброзробки — HTML (HyperText Markup Language) та CSS (Cascading Style Sheets). Ці мови формують основу структури та візуального оформлення вебінтерфейсу, забезпечуючи доступність, зручність використання та ефективну взаємодію з користувачем.

HTML відповідає за структурну розмітку вмісту сторінки, визначаючи елементи, такі як заголовки, параграфи, списки, форми, зображення та інші. Сучасний підхід до розмітки передбачає використання семантичних елементів, які надають змістовий контекст вмісту, полегшуючи розуміння структури сторінки як для розробників, так і для пошукових систем та асистивних технологій. Наприклад, теги <header>, <nav>, <main>, <section>, <article>, <footer> чітко вказують на призначення відповідних блоків контенту. Використання семантичної розмітки сприяє покращенню доступності для користувачів з обмеженими можливостями та підвищує ефективність SEO-оптимізації, оскільки пошукові системи краще розуміють структуру та зміст сторінки [3].

У вебсайті Remontika семантична структура застосовується для логічного поділу інформаційного простору: верхнє меню розміщено у блоці <header>, основна форма заявки — у <main>, розділ з часто ставленими питаннями — у <section>, а інформація про компанію — у <footer>. Такий підхід забезпечує зрозумілу структуру коду та полегшує навігацію для користувачів.

CSS відповідає за візуальне представлення елементів HTML, дозволяючи задавати стилі, такі як кольори, шрифти, відступи, положення елементів на сторінці, а також забезпечувати адаптивність сайту до різних розмірів екранів. CSS створює стилістичну єдність інтерфейсу, формує сприйняття бренду та покращує користувацький досвід завдяки гнучкості в оформленні інтерактивних елементів.

Використання зовнішніх таблиць стилів дозволяє централізовано керувати стилями, що спрощує підтримку та оновлення дизайну сайту [4].

Завдяки поєднанню структурної логіки HTML та візуальної пластичності CSS, вебінтерфейс Remontika отримує чітку організацію та естетичну виразність. Це дозволяє реалізувати технічні функції сайту, забезпечити зручність, привабливість та професійне враження від взаємодії з платформою, що є важливим для довіри користувачів, особливо у сфері сервісного обслуговування техніки.

Окрему увагу в межах реалізації інтерфейсної частини сайту Remontika було приділено вибору фреймворку Tailwind CSS, який став основою стилізації та компонентного структурування візуального представлення. Tailwind CSS — це утилітарний CSS-фреймворк, який забезпечує розробникові можливість створення адаптивних і візуально узгоджених інтерфейсів завдяки широкому набору готових CSS-класів (Рис. 2.1), що безпосередньо вбудовуються в HTML-розмітку. Tailwind реалізує принцип “utility-first” — кожен клас відповідає за одну чітку властивість: відступ, колір, тінь, гнучке позиціонування тощо.

```
<div class="w-full flex flex-col items-center justify-center min-h-screen bg-gray-50">
  <h1 class="text-2xl font-bold text-gray-800 mb-4">Подати заявку на ремонт</h1>

  <button class="bg-blue-500 hover:bg-blue-600 text-white font-bold py-2 px-4 rounded shadow-md">
    Відправити заявку
  </button>

  <p class="text-sm text-gray-500 mt-3">Ми зв’яжемося з вами протягом 15 хвилин</p>
</div>

```

Рис.2.1 Приклад використання утилітарних класів Tailwind CSS безпосередньо в HTML-коді.

Tailwind CSS орієнтований на швидку розробку прототипів і фінальних версій інтерфейсів завдяки прямому використанню стилів без необхідності написання індивідуальних стилей [5]. Подібний підхід дозволяє значно зменшити обсяг окремих CSS-файлів, скоротити кількість дублювань у стилях, а також уніфікувати візуальний стиль сайту на всіх його сторінках (Рис 2.2).

index.tailwind.scss

```scss
.btn {
  @apply text-white py-2 px-4 rounded;
  &-purple {
    @apply bg-purple-500;
    &:hover {
      @apply bg-purple-700;
    }
  }
}
```

index.css

```css
.btn {
  color: #fff;
  padding: .5rem 1rem;
  border-radius: .25rem;
}

.btn-purple {
  background-color: #9f7aea;
}

.btn-purple:hover {
  background-color: #6b46c1;
}
```

Рис. 2.2 Порівняння традиційного CSS та утилітарного підходу Tailwind CSS у стилізації кнопок.

Перевагою Tailwind CSS є його гнучкість і розширюваність — розробник може створити власну конфігурацію теми, кольорів, шрифтів і breakpoints у конфігураційному файлі tailwind.config.js. Завдяки цьому стало можливим швидко створити інтерфейс для Remontika, який відповідає сучасним стандартам UX/UI, зберігає візуальну єдність та легко масштабується при додаванні нових сторінок або компонентів.

Особливо важливим є те, що Tailwind CSS дозволяє значно пришвидшити прототипізацію: зміни в інтерфейсі можна вносити майже миттєво, прямо в HTML-коді, без необхідності перемикатися між HTML і CSS-файлами. Це суттєво економить час на етапах дизайнування, тестування і рефакторингу, що є важливим фактором в умовах обмежених термінів реалізації дипломного проєкту.

Tailwind CSS є ідеальним інструментом для створення масштабованих, адаптивних, компонентно-орієнтованих вебінтерфейсів і широко використовується в комерційній розробці [6]. Саме тому вибір цієї технології в межах розробки сайту Remontika є обґрунтованим з позиції продуктивності, гнучкості та відповідності сучасним галузевим стандартам.

Ключовим завданням було забезпечення високої продуктивності, адаптивності, гнучкості у маршрутизації та можливості інтеграції з зовнішніми сервісами. З огляду на ці вимоги, було прийнято рішення реалізувати клієнтську частину сайту на основі Next.js — популярного JavaScript-фреймворку для створення реактивних застосунків, побудованих поверх React.

Next.js надає низку переваг, які вирізняють його на фоні класичних SPA або традиційних шаблонізаторів. По-перше, він підтримує серверну побудову сторінок (SSR — Server-Side Rendering), що дозволяє динамічно створювати HTML-код на сервері під кожен запит користувача. Це суттєво покращує швидкість завантаження першого екрану (time to first byte), що є критично важливим для SEO та мобільного досвіду (Рис. 2.3).

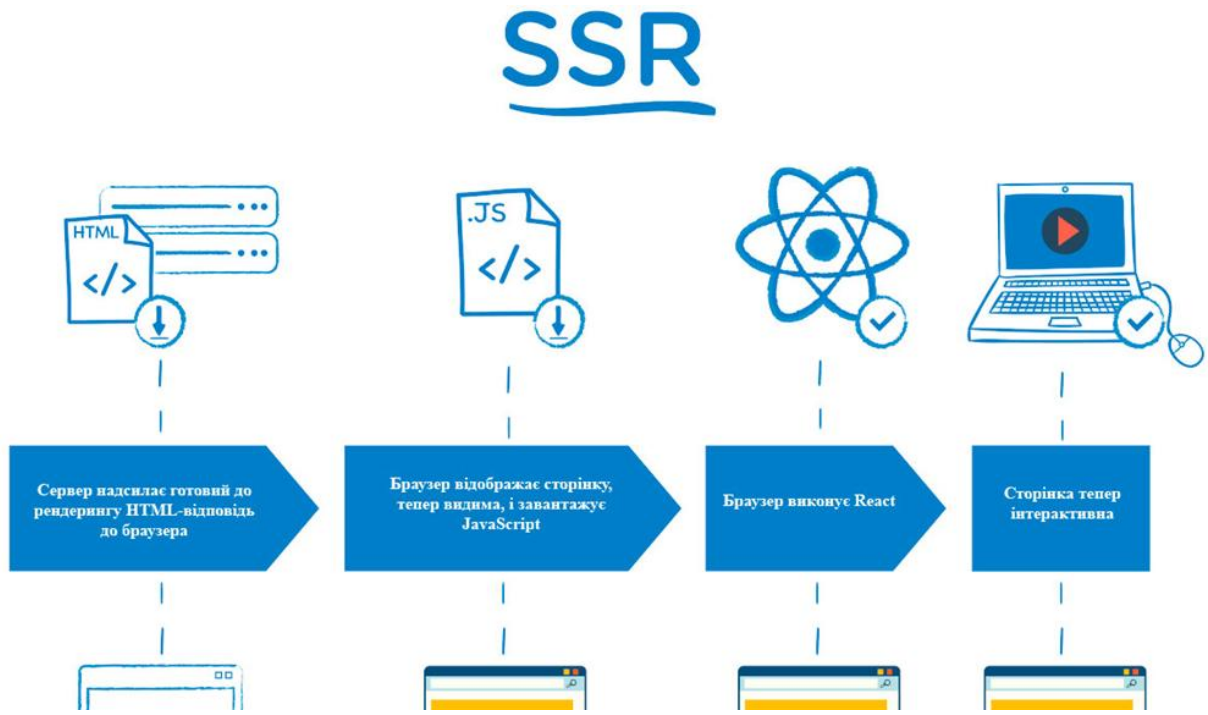

Рис.2.3 Схема SSR-процесу в Next.js.

Друга вагома перевага — це можливість статичної генерації сторінок (SSG — Static Site Generation), за якої HTML створюється на етапі компіляції і зберігається у вигляді готових файлів. Це забезпечує миттєве завантаження та дозволяє сайту масштабуватися без додаткових витрат на серверну

інфраструктуру. Завдяки цьому гібридному підходу (SSR + SSG) сторінки, які змінюються рідко (наприклад, сторінка «Про нас» або «Довідка»), можуть бути згенеровані один раз, а сторінки з динамічним контентом (наприклад, статус заявки або кабінет користувача) — формуються в реальному часі.

Next.js також має автоматичну маршрутизацію на основі файлової структури, що значно спрощує організацію проєкту. Кожен файл у папці /pages автоматично стає маршрутом, що усуває потребу в ручному налаштуванні шляхів та контролерів. Це підвищує швидкість розробки та полегшує підтримку структури застосунку.

Крім того, Next.js легко інтегрується з REST- або GraphQL-API, підтримує серверні функції (API Routes), що дозволяє реалізувати бекенд-логіку без підключення окремого серверного середовища. Наприклад, у проєкті Remontika через API Routes реалізована обробка форм подачі заявки та взаємодія з Firebase. Завдяки підтримці TypeScript, middleware, dynamic routing і можливості Lazy Loading компонентів, Next.js забезпечує високий рівень масштабованості, що критично важливо в умовах розширення функціоналу сервісу. Такич чином, Next.js у межах даного дипломного проєкту дозволяє поєднати гнучкість React із продуктивністю SSR/SSG, простотою структурування маршрутизації та інтеграційною потужністю сучасного JavaScript-стека.

У межах реалізації проекту для зберігання, обробки та розповсюдження даних було обрано платформу Firebase, розроблену компанією Google. Це хмарне рішення, що надає набір повнофункціональних сервісів для створення сучасних веб- та мобільних застосунків без необхідності самостійно налаштовувати сервери, інфраструктуру баз даних або окремі бекенд-модулі [7]. У проєкті використовуються три ключові компоненти: Firebase Hosting, Firebase Realtime Database та Firebase Storage.

Однією з основних причин вибору Firebase стало усунення потреби у створенні та підтримці власної серверної інфраструктури. У класичному підході розробник повинен налаштувати окремий сервер (наприклад, з використанням Node.js або PHP), налаштувати базу даних (MySQL, PostgreSQL), сертифікати

безпеки (SSL), доменні записи, маршрутизацію тощо. Firebase надає цю функціональність «із коробки», що дозволяє зосередитися виключно на розробці функціоналу та логіки користувацького інтерфейсу. У проєкті Remontika це стало критично важливим через бажання сконцентруватися на побудові зручного UX-середовища.

Firebase Hosting забезпечує швидкий та безпечний хостинг статичних вебзастосунків, включаючи сайти, створені на базі Next.js. Він підтримує HTTPS за замовчуванням, автоматичну CDN-доставку та миттєвий деплой через CLI-команду. Це дозволяє викладати оновлення сайту на продакшн за кілька секунд, не втрачаючи в продуктивності. У випадку Remontika — це означає, що сайт можна моментально оновити в разі зміни дизайну або логіки.

Основним джерелом динамічних даних виступає Firebase Realtime Database — некласична NoSQL-база, яка зберігає інформацію у вигляді вкладених JSON-структур. Ключовою перевагою цієї технології є механізм автоматичних оновлень у реальному часі (Realtime Sync): коли адміністратор змінює статус заявки в базі, ця зміна миттєво оновлюється на стороні клієнта без оновлення сторінки. Такий підхід дозволяє створити відчуття «живого» інтерфейсу, в якому інформація завжди актуальна, без затримок або ручного перезавантаження.

Ще один важливий сервіс — Firebase Storage, який використовується для зберігання файлів, зокрема зображень пристроїв, сканів документів чи фотографій пошкоджених смартфонів. Цей сервіс дозволяє безпечно завантажувати та отримувати файли через унікальні URL, а правила доступу можна налаштувати за допомогою Firebase Security Rules — гнучкої системи прав, яка дозволяє обмежувати доступ лише для авторизованих користувачів або відповідних ролей (наприклад, лише адміністратор має право видаляти файли).

Крім цього, важливою перевагою Firebase є гнучка система безпеки через правила доступу. Кожен запис у базі або файл у хмарі може бути захищений умовами, які враховують роль користувача (admin/user), час запиту, тип операції (read/write), або навіть специфічні дані (наприклад, userId у шляху до ресурсу). Це

дозволяє реалізувати базову авторизацію і контроль доступу без написання серверного коду.

Завдяки вищезгаданим можливостям Firebase став оптимальним бекенд-рішенням. Він не лише забезпечує масштабованість, продуктивність і простоту в реалізації, а й дозволяє підтримувати сучасну логіку інтерактивного цифрового сервісу, який реагує на дії користувача в реальному часі, залишаючись при цьому безпечним, швидким і гнучким.

У межах реалізації сервісу, окрім фронтенд-інтерфейсу сайту, також впроваджується інтеграція з месенджером Telegram через власного бота. Це рішення обумовлене бажанням підвищити зручність користувача та зробити сервіс максимально інтерактивним і доступним у режимі реального часу. Telegram є одним із найпопулярніших месенджерів в Україні, має відкритий API, а також підтримує надійний механізм відправки повідомлень, що ідеально підходить для реалізації системи сповіщень.

Суть функціональності Telegram-бота полягає у відправленні повідомлень користувачу при зміні статусу його заявки. Такий підхід дозволяє мінімізувати потребу вручну перевіряти стан звернення на сайті, адже оновлення надходитимуть безпосередньо у чат Telegram. Алгоритм взаємодії виглядає наступним чином: користувач залишає заявку на сайті – запис зберігається у Firebase Realtime Database разом із його Telegram ID (якщо він його вказав) – адміністратор змінює статус заявки через адмін-панель – тригерна функція (наприклад, Firebase Function) виявляє зміну й ініціює виклик Telegram API – користувачу надходить повідомлення з оновленим статусом та, за потреби, коментарем.

Переваги цього рішення очевидні:

- Оперативність — повідомлення доставляються миттєво, без затримок;
- Приватність і зручність — користувач отримує оновлення у звичному месенджері;
- Легкість впровадження — Telegram надає повноцінне REST API з методами sendMessage, editMessageText, getUpdates тощо, що дозволяє гнучко інтегрувати логіку бота з Firebase;

- Масштабованість — бот може бути легко доповнений іншими функціями: наприклад, запит історії заявок, зв’язок з оператором, або навіть обробка голосових звернень у майбутньому.

2.2. Архітектурна структура системи

Архітектура Remontika побудована на модульному принципі з чітким розділенням функціональних зон. Система умовно поділена на три основні складові: клієнтську частину (FrontEnd), серверно-логічну (BackEnd) частину, а також систему зберігання та обробки даних, які взаємодіють між собою через стандартизовані API-запити, події в реальному часі та хмарні сервіси Google Firebase.

Клієнтська частина (Frontend)

Це те, з чим безпосередньо взаємодіє користувач. Сайт побудований за допомогою фреймворку Next.js, що дозволяє реалізовувати як серверний рендеринг, так і клієнтське оновлення. Користувацький інтерфейс (UI) реалізований із використанням Tailwind CSS, забезпечуючи адаптивність, швидкість та візуальну цілісність. Саме через цю частину здійснюється: заповнення форми заявки, відображення статусу заявки, авторизація користувача, перегляд повідомлень або помилок, взаємодія з Telegram (через API-ключ користувача).

Серверна частина (Backend/обробка логіки)

Бекенд-процеси реалізовані двома шляхами: через API Routes у Next.js (тобто файли всередині /pages/api/), які відповідають за отримання/відправку даних — наприклад, submit.js, getStatus.js або через Firebase Cloud Functions, які можуть реагувати на події в Realtime Database чи Firestore (onWrite, onUpdate), або бути викликані з клієнта напряму. До завдань серверної частини входить: збереження заявки до бази, перевірка автентичності користувача, надсилання повідомлення в Telegram через бот, оновлення статусу заявки з боку адміністратора.

Для повнішого розуміння логіки маршрутизації сайту Remontika нижче наведено схему відповідності URL-шляхів компонентам застосунку та подальшій логіці обробки даних і запитів (Рис 2.4).

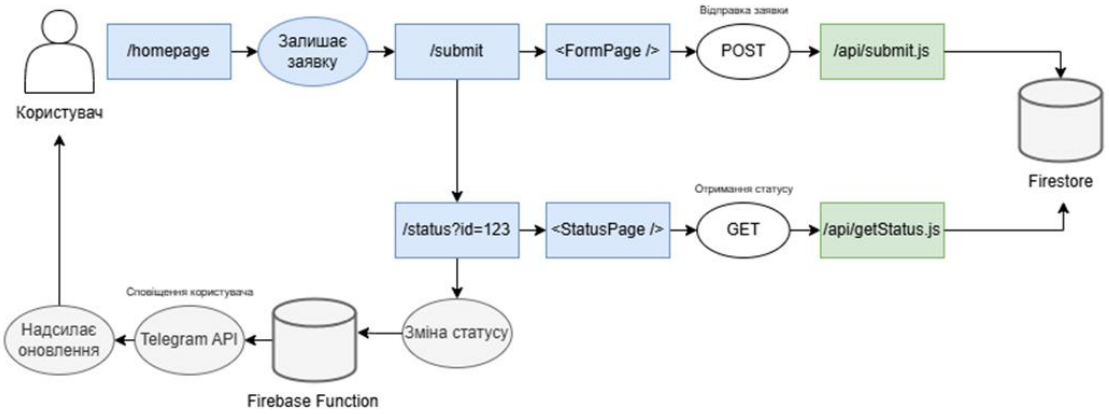

Рис. 2.4 Схема маршрутизації сторінок, API-запитів і логіки сповіщень.

Середовище зберігання та обробки даних (Firebase)

Уся інформація зберігається в екосистемі Firebase, яка включає:

- Firestore — NoSQL-база даних із колекціями типу applications, users, admins, notifications;
- Firebase Storage — зберігання фото пошкоджених пристроїв;
- Firebase Hosting — хостинг сайту Remontika;
- Firebase Authentication — базова автентифікація;
- Realtime Listener API — відстеження змін у статусах.

Усі ці логічні модулі та потоки взаємодії між ними представлені у вигляді компонентної діаграми, що зображена на рисунку 2.5. Діаграма наочно ілюструє залежності між клієнтською частиною, API-модулями, хмарними функціями та базами даних, на яких ґрунтується робота системи.

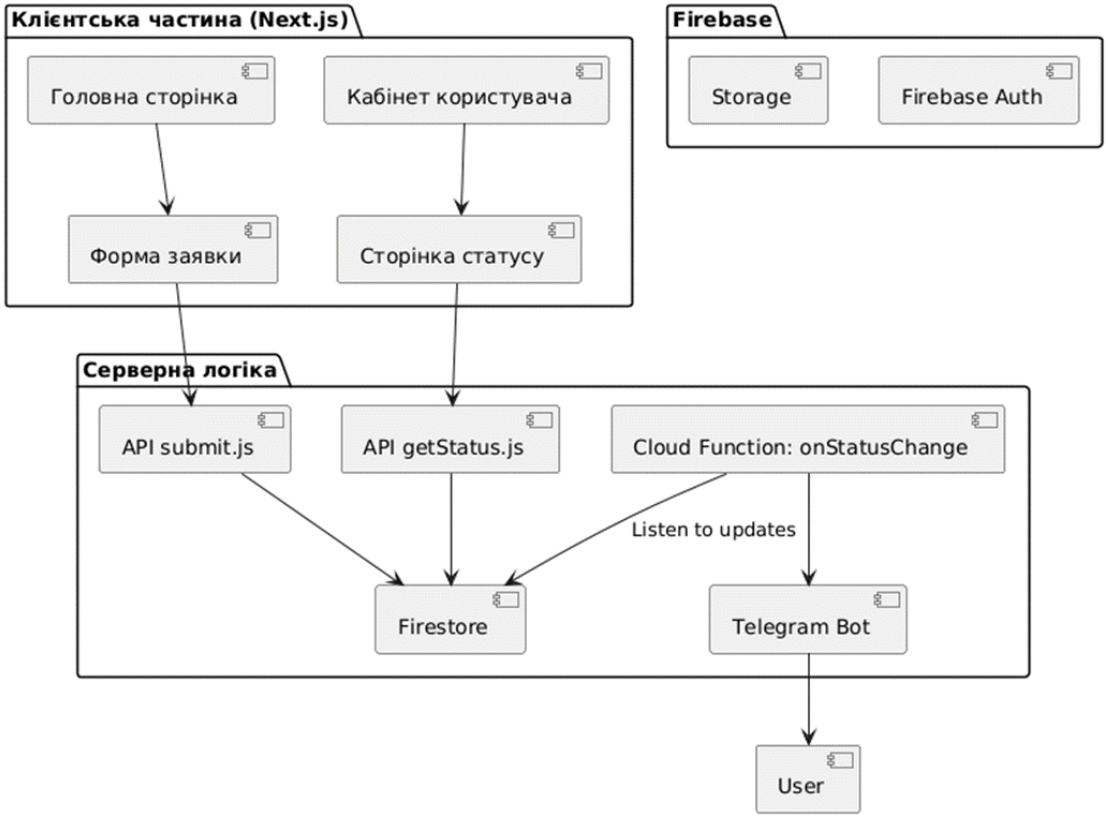

Рис.2.5 Компонентна UML-діаграма системи Remontika з розподілом на інтерфейсну, серверну та хмарну частини.

2.3 Модель даних та UML/ER-діаграми

Модель даних є фундаментом для стабільного функціонування будь-якого сучасного вебзастосунку, особливо такого, що обробляє динамічні запити користувачів та взаємодіє з зовнішніми сервісами в режимі реального часу. У межах розробки Remontika було спроєктовано логічну структуру даних, яка дозволяє зберігати, обробляти та аналізувати всю критично важливу інформацію: заявки на ремонт, користувацькі профілі, сповіщення, історію змін і навіть характеристики пристроїв. Хоча система реалізована на основі NoSQL-бази Firestore, де структура є гнучкою і документно-орієнтованою, логіка між об’єктами чітко витримана, а кожна сутність виконує окрему роль у загальній архітектурі.

Центральне місце в системі займає сутність користувача. Кожен користувач має унікальний ідентифікатор, який генерується через Firebase Authentication. Крім того, зберігаються контактні дані, такі як ім’я, електронна адреса, номер телефону та, за бажанням, Telegram ID, якщо користувач погодився на сповіщення через бот. Окремий атрибут фіксує роль користувача — звичайний клієнт або адміністратор, що дозволяє реалізувати контроль доступу до окремих функцій. Саме користувач ініціює створення заявки, що формує прямий зв’язок між двома ключовими сутностями: User та Request.

Заявка на ремонт є основною транзакційною одиницею в системі. Вона включає унікальний ідентифікатор, посилання на автора заявки (через userId), дані про пристрій, зокрема бренд, модель, а також опис технічної проблеми, яку користувач самостійно вводить у формі. Важливу роль відіграє поле статусу, яке визначає, на якій стадії перебуває обробка: створення, у роботі, завершено, або скасовано. При потребі користувач може завантажити зображення несправності, яке зберігається у Firebase Storage, а посилання на нього додається до заявки. Кожна заявка містить відмітку про дату створення та останню зміну, що необхідно для аналітики й побудови журналу подій.

У деяких реалізаціях доцільно вводити додаткову сутність «Пристрій», яка дозволяє формувати довідник моделей за брендами, що особливо актуально для майбутнього автозаповнення полів у формі та аналітики типових несправностей. Такий підхід спрощує обробку повторюваних запитів і забезпечує уніфікованість у поданні даних. Цей об'єкт, хоча й не критично необхідний, дозволяє структурувати і нормалізувати дані без втрати гнучкості NoSQL-архітектури.

Ще однією ключовою частиною структури є сповіщення, які формуються на основі змін у статусі заявки. Усі повідомлення зберігаються як окремі об’єкти Notification, де фіксується, з якої заявки та якому користувачу надіслане повідомлення, через який канал — Telegram, SMS чи інший, який був зміст тексту, коли саме це відбулося та чи пройшла доставка успішно. Це дозволяє в будь-який момент відновити лог подій та перевірити, чи отримав клієнт потрібну інформацію.

Також у разі потреби можна реалізувати повторну відправку на основі виявлених збоїв.

Окремим важливим елементом моделі є журнал дій адміністратора. У випадку, якщо адміністратор оновлює статус заявки або вносить правки, ці дії реєструються як окремі лог-записи з інформацією про тип дії, цільову заявку, час і, за потреби, коментар. Це важливо з погляду прозорості, відповідальності й безпеки, особливо якщо до роботи в системі залучено кількох операторів.

Загальна структура даних підтримує відносини один-до-багатьох: один користувач може мати кілька заявок, кожна заявка — кілька сповіщень, і кожне сповіщення пов’язане з єдиним користувачем. Якщо вводиться таблиця пристроїв — то багато заявок можуть належати одній моделі. Усе це утворює гнучку, але структуровану модель, яка чудово підходить для роботи у Firestore.

Для візуалізації цієї структури було побудовано UML-діаграму класів (Рис 2.6), де відображено кожну сутність як окремий блок з основними атрибутами, а також стрілки-асоціації, які демонструють залежності між об’єктами. Це дозволить не лише впорядкувати логіку зберігання даних та швидко доповнювати її новими сутностями, а й легко масштабувати архітектуру в майбутньому.

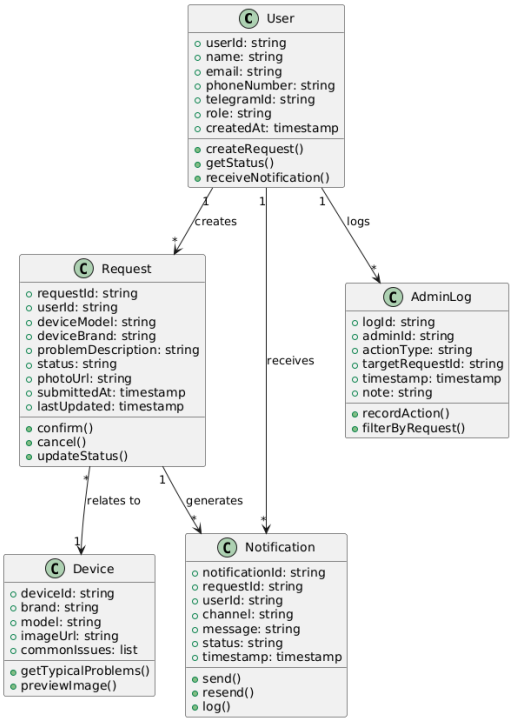

Рис.2.6 Класова UML-діаграма сутностей з методами для системи Remontika.

2.4 Опис функціональної схеми

Функціональна схема системи Remontika описує повну послідовність обробки взаємодії користувача із сайтом — від натискання кнопки подачі заявки до отримання сповіщення про її обробку. Цей підрозділ деталізує логіку бізнес-процесів системи, включаючи дії клієнта, реакції інтерфейсу, обробку на рівні API, збереження даних у базі та використання хмарних функцій для автоматизованого інформування користувача про зміну статусу. Таким чином, формується цілісний життєвий цикл кожної заявки, в якому беруть участь усі ключові модулі проєкту.

Процес починається з того, що користувач переходить на сторінку /submit, де перед ним відображається форма заявки, реалізована у вигляді React-компонента. Після заповнення обов’язкових полів — таких як тип пристрою, марка, модель, опис проблеми — користувач натискає кнопку «Відправити заявку». На цьому етапі на клієнтській частині виконується базова валідація введених даних, що реалізована за допомогою JavaScript-методів і дозволяє перевірити коректність формату номера телефону, заповненість обов’язкових полів, обмеження довжини текстових полів та тип прикріпленого файлу. У разі проходження перевірки формується HTTP-запит типу POST, який надсилається на маршрут /api/submit.js. Запит містить JSON-об’єкт з усіма введеними даними, включно з посиланням на фото (якщо воно попередньо було завантажене до Firebase Storage через окремий запит).

На стороні бекап-функції Next.js у файлі submit.js відбувається обробка цього POST-запиту. Серверна логіка перевіряє коректність структури запиту, автентичність користувача (якщо авторизований), формує об’єкт заявки та виконує запис у Firestore до колекції requests. Записується унікальний requestId, прив’язується userId, зберігається модель пристрою, опис проблеми, статус заявки («створено»), а також мітки часу для submittedAt та lastUpdated. Після успішного запису функція повертає клієнту відповідь зі статусом 200 і унікальним ID заявки, який буде використовуватись для подальшого перегляду.

Одразу після створення заявки в Firestore автоматично спрацьовує Cloud Function, яка спостерігає за колекцією requests і реагує на події створення документа. Виявивши новий запис, функція формує повідомлення для Telegram або іншого обраного каналу (наприклад, Viber), яке містить текст на кшталт: «Ваша заявка №123 успішно прийнята.» За допомогою Telegram Bot API виконується HTTP-запит на адресу https://api.telegram.org/bot<TOKEN>/sendMessage, у якому передаються ідентифікатор користувача та текст повідомлення. У разі успішної доставки, запис про сповіщення зберігається до колекції notifications.

Наступним важливим етапом є оновлення статусу заявки адміністратором, що виконується через інтерфейс адмін-панелі. Після авторизації адміністратор отримує список заявок і може змінити статус будь-якої з них на «в роботі», «завершено», «скасовано» або перейти на наступний етап статусу. Зміна статусу автоматично оновлює відповідний документ у Firestore, що, своєю чергою, знову активує Cloud Function — тепер вже на подію onUpdate. Функція порівнює попередній і новий статус, і, якщо є зміна, формує нове повідомлення користувачеві: «Заявка №123 оновлена: Статус – виконано.» Так само, як і раніше, відправляється повідомлення через Telegram API та зберігається запис до notifications.

На фронтенді ця зміна також відображається без потреби ручного оновлення сторінки. Це реалізується за допомогою механізму реального часу Firestore Listener, який дозволяє підписатися на конкретний документ заявки. Коли користувач переглядає статус своєї заявки на сторінці /status?id=123, клієнтська частина виконує не одноразовий запит, а підключається до змін у Firestore. У разі оновлення поля status або lastUpdated, нове значення миттєво підтягується в інтерфейс і виводиться на екран, без перезавантаження сторінки або додаткових дій з боку користувача.

Таким чином, увесь цикл обробки заявки відбувається в асинхронному режимі: дані миттєво передаються між компонентами інтерфейсу, API, базою даних, хмарними функціями та зовнішніми каналами комунікації. Кожна дія

користувача або адміністратора автоматично ініціює відповідну реакцію системи, що забезпечує динамічність, інформативність і високу інтерактивність сервісу.

Для наочної демонстрації цієї логіки доцільно використати UML-діаграму послідовності, яка відображає обмін повідомленнями між ключовими компонентами системи (Рис 2.7). У діаграмі відображено: користувач – клієнтське застосування – API /submit.js – Firestore – Cloud Function – Telegram API – користувач, а також цикл зміни статусу та реакцію на події через Listener.

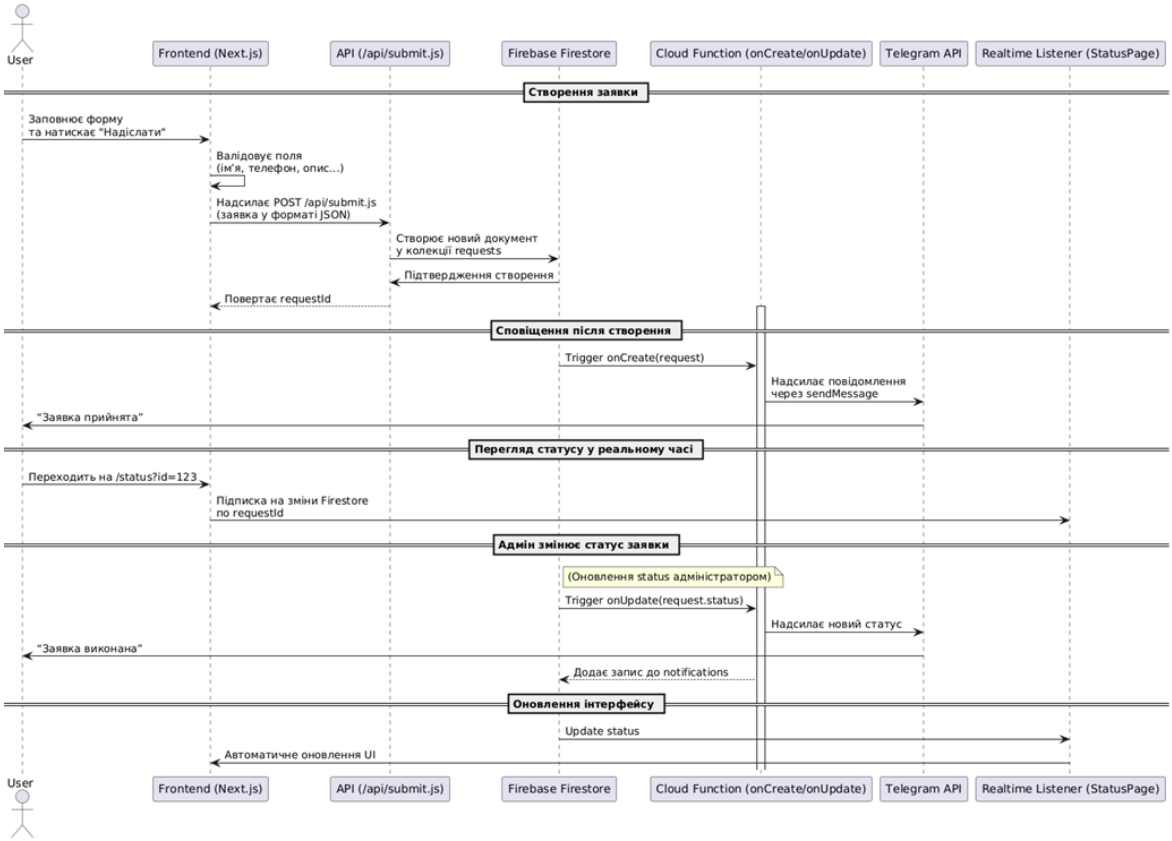

Рис. 2.7 UML-діаграма послідовності життєвого циклу заявки в системі Remontika.

3 ІНТЕРФЕЙС КОРИСТУВАЧА: ПРАКТИЧНА РЕАЛІЗАЦІЯ

Інтерфейс користувача є одним з ключових елементів будь-якого вебсервісу, що визначає не лише зовнішній вигляд застосунку, але й загальний досвід взаємодії користувача з системою. Від того, наскільки зручною, логічною та доступною є побудова інтерфейсу, безпосередньо залежить ефективність виконання основних завдань сервісу, рівень задоволеності користувача та загальне враження від цифрового продукту. У випадку платформи Remontika інтерфейс має забезпечити інтуїтивну навігацію, швидку подачу заявки, оперативний перегляд статусу ремонту, а також легкий доступ до допоміжної інформації.

Під час проєктування інтерфейсу було враховано принципи зручності використання (usability), відповідність сучасним стандартам UX/UI-дизайну, адаптивність для різних типів пристроїв, а також мінімізацію кількості кроків, які потрібні користувачу для досягнення мети. Вебсервіс структурується за модульним підходом, де кожна функціональна одиниця реалізована у вигляді окремої сторінки або компонента. Це дозволяє забезпечити як простоту навігації, так і гнучкість при масштабуванні функціоналу.

У цьому розділі буде здійснено послідовний огляд реалізованого інтерфейсу, пояснено логіку його побудови та структуру сторінок, а також продемонстровано основні екрани вебсервісу з прикріпленими візуальними прикладами. Кожен скріншот супроводжується поясненням функціонального призначення окремого елементу, що дозволяє зв'язати технічну реалізацію з реальними сценаріями використання.

3.1 Домашня інформаційна сторінка веб-сервісу Remontika

Після переходу на головну сторінку вебсервісу Remontika користувач одразу потрапляє в чітко структуровану та візуально привабливу зону знайомства з брендом (Рис 3.1). У центральній частині екрана розміщено великий заголовок з

назвою сервісу та коротким підзаголовком, який чітко окреслює сферу діяльності — ремонт смартфонів. Це дозволяє одразу сформувати враження про призначення сайту та зменшує когнітивне навантаження на нового відвідувача.

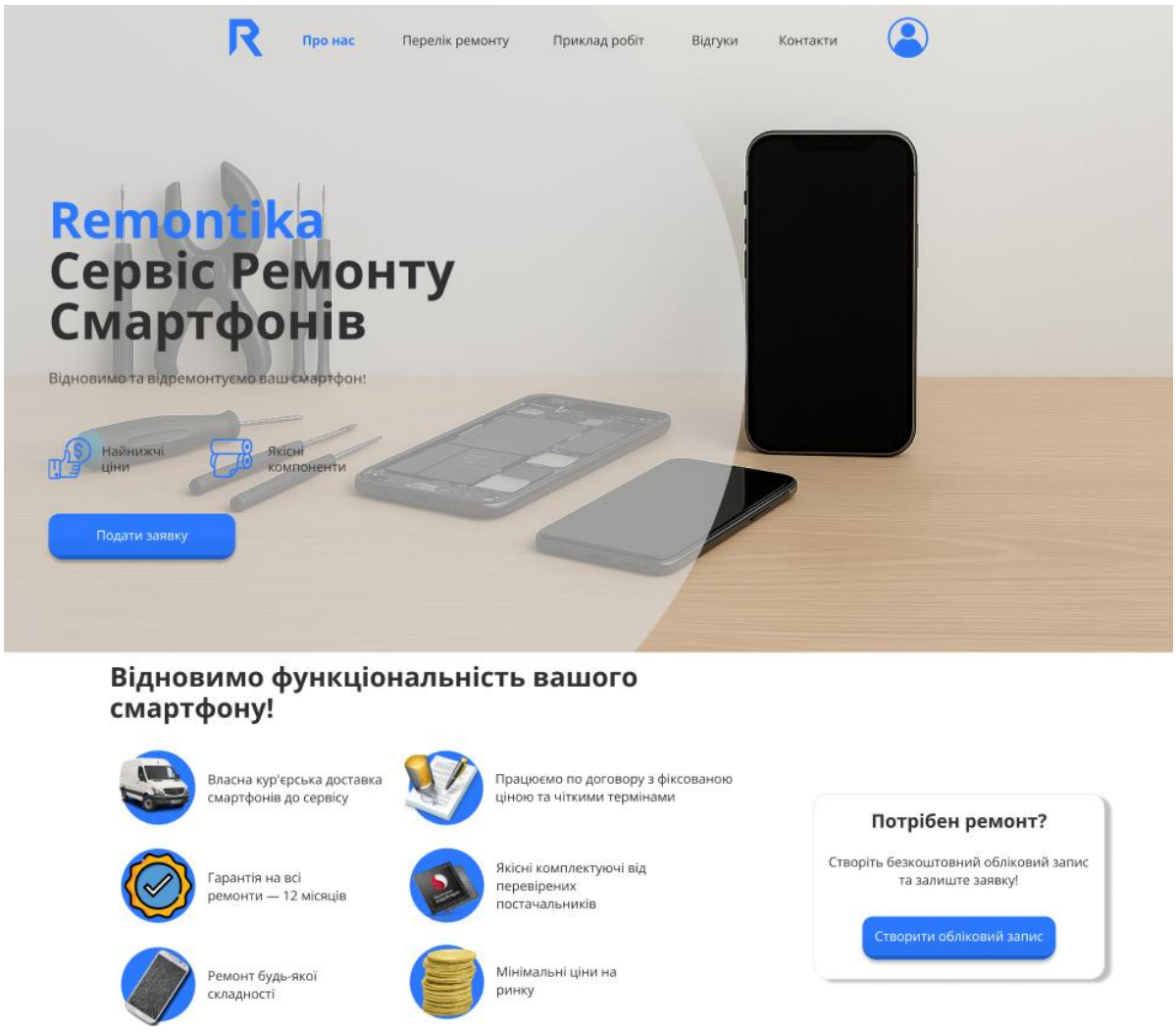

Рис.3.1 Головна сторінка вебсервісу Remontika.

У верхній частині сайту (хедер) розташоване меню навігації, що включає пункти «Про нас», «Перелік ремонту», «Приклад робіт», «Відгуки», «Контакти» та іконку входу до облікового запису. Меню виконує функціонал «якіру», при натисканні сторінка користувача гортається до певної точки. Також таке горизонтальне меню дозволяє легко орієнтуватися в логіці розміщення основних

функціональних розділів, а інтуїтивна піктограма з силуетом користувача однозначно вказує на можливість реєстрації або входу до особистого кабінету.

Візуальний акцент зроблено на заклику до дії — синя кнопка «Подати заявку», яка має помітне розміщення зліва внизу від центру екрану. Її кольорове оформлення контрастує з фоном, що збільшує ймовірність взаємодії з боку користувача. Також під кнопкою додано два інформаційні маркери — «Найнижчі ціни» та «Якісні компоненти», які слугують додатковим підсиленням довіри.

Нижче основного блоку розміщено секцію переваг, реалізовану у вигляді іконок з підписами. Вона забезпечує швидке ознайомлення з ключовими перевагами компанії: гарантія, доставка, фіксовані строки, якісні запчастини, ціни та універсальність ремонту. Праворуч розташовано окремий віджет із закликом до реєстрації. Він виконує функцію другорядного, але важливого заклику до дії — створити обліковий запис, необхідний для подачі заявки. Таким чином, головна сторінка виконує одразу кілька ключових функцій: інформує, викликає довіру, орієнтує в структурі ресурсу та веде користувача до наступного етапу — активної взаємодії з сервісом.

Наступні три скриншоти є логічним продовженням головної сторінки й охоплюють розділи переліку послуг (Рис. 3.2 – 3.4), демонстрації результатів роботи, механіки обслуговування, відгуків і відповідей на часті питання.

У нижній частині головної сторінки розміщено секцію «Що ми ремонтуємо?», яка представлена у вигляді зрозумілої візуальної сітки з шести основних категорій пристроїв: мобільні телефони, планшети, ігрові консолі, розумні годинники, ноутбуки та комп’ютери (Рис. 3.2). Наступний блок «Ми у цифрах» є елементом соціального підтвердження (social proof) та інструментом формування довіри. У ньому вказано ключові факти: одинадцятирічний досвід у сфері ремонту, більше ніж 5000 виконаних замовлень та інша інформація. Подібне числове позиціонування дає змогу користувачу швидко оцінити професійний рівень компанії.

Рис.3.2 Категорії пристроїв та статистика компанії Remontika.

Секція під назвою «Наші роботи говорять самі за себе» демонструє приклади виконаних ремонтів, представлених у форматі «до» та «після». Такі візуальні кейси дозволяють підкріпити обіцяну якість реальними результатами. Для зручності перегляду додано фільтри за категоріями пристроїв: смартфони, ноутбуки та планшети. Під цим блоком розташовано секцію «Як ми працюємо?», яка у вигляді покрокової інструкції пояснює процес взаємодії клієнта з сервісом.

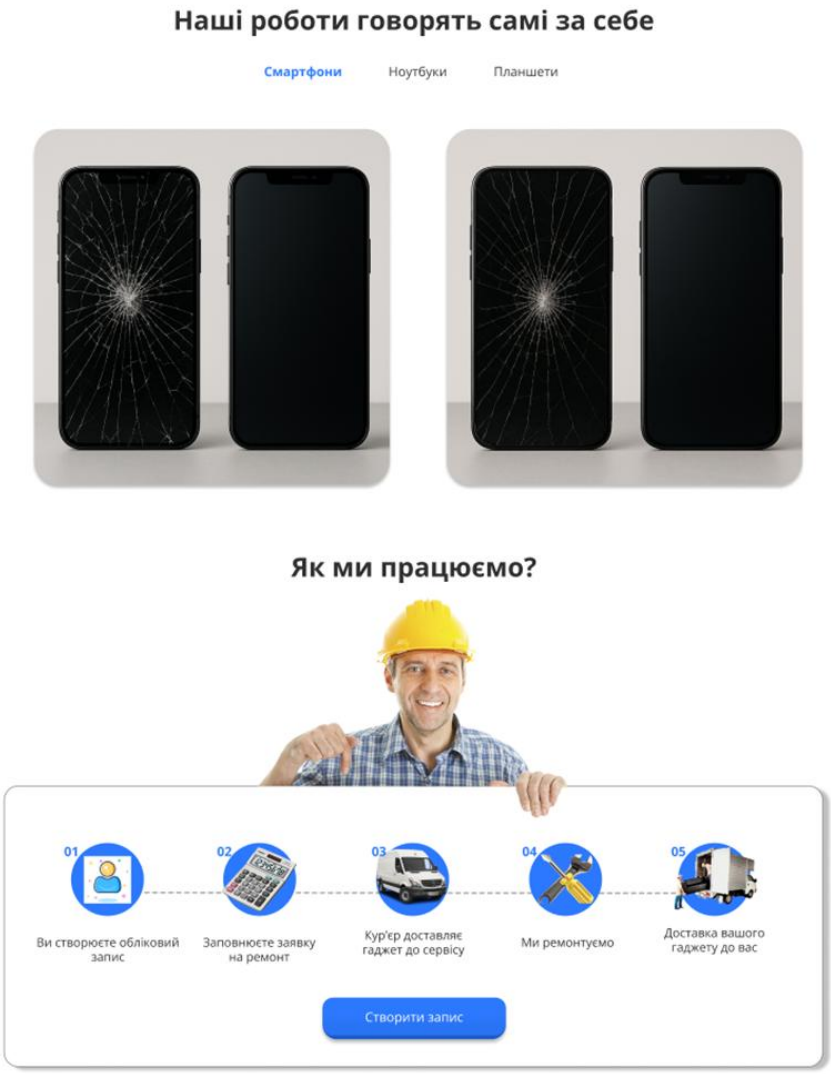

Рисунок 3.3 – Приклади виконаних ремонтів та опис моделі роботи сервісу.

Нижче подано два важливих інформаційних розділи: «Відгуки» та «FAQ». Перший реалізовано у вигляді слайдера з інтеграцією реальних коментарів клієнтів — імен, дат, текстів та оцінок. Другий блок — це «FAQ» (Часті запитання), який оформлено в стилі акордеону: користувач натискає на запитання й отримує розгорнуту відповідь.

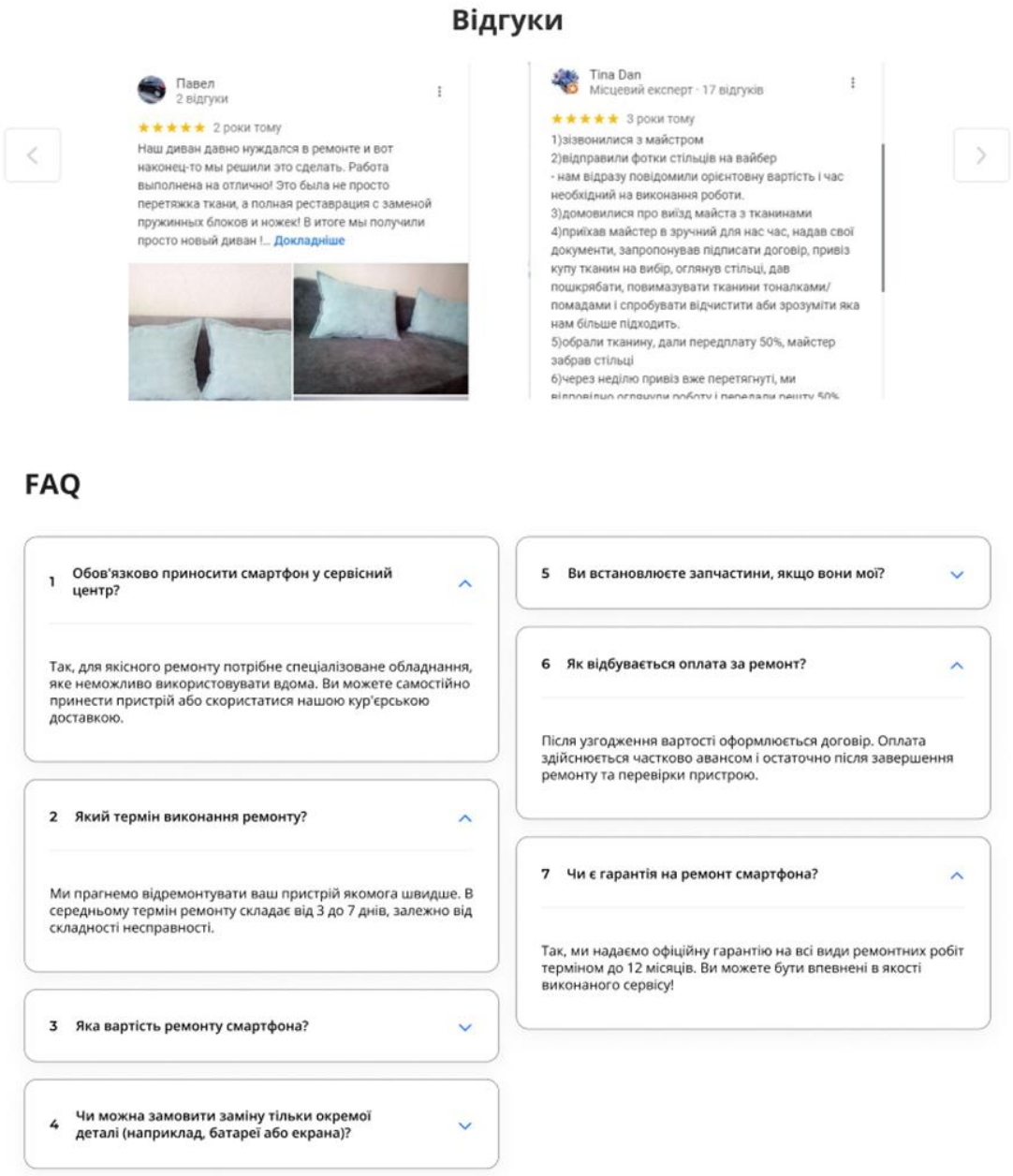

Рис.3.4 Відгуки клієнтів і секція з відповідями на часті запитання.

Завершує головну сторінку вебсервісу Remontika інформаційний блок «Контакти», що розташований у нижній частині сайту у форматі класичного футера. Він реалізований у вигляді двох основних частин: текстового віджета з контактними даними та інтерактивної мапи. У лівій частині футера відображено ключову контактну інформацію: номер телефону, фізичну адресу в Києві (вул. Солом’янська, 7, ДУІКТ), електронну пошту, а також логотипи соціальних месенджерів Telegram і Viber, які вказують на альтернативні канали зв’язку. У

правій частині блоку інтегровано карту з API Google Maps, на якій відображено точне розташування компанії. У нижній частині футера, в окремому блоці правіше від логотипу сервісу вказано авторство розробки проєкту.

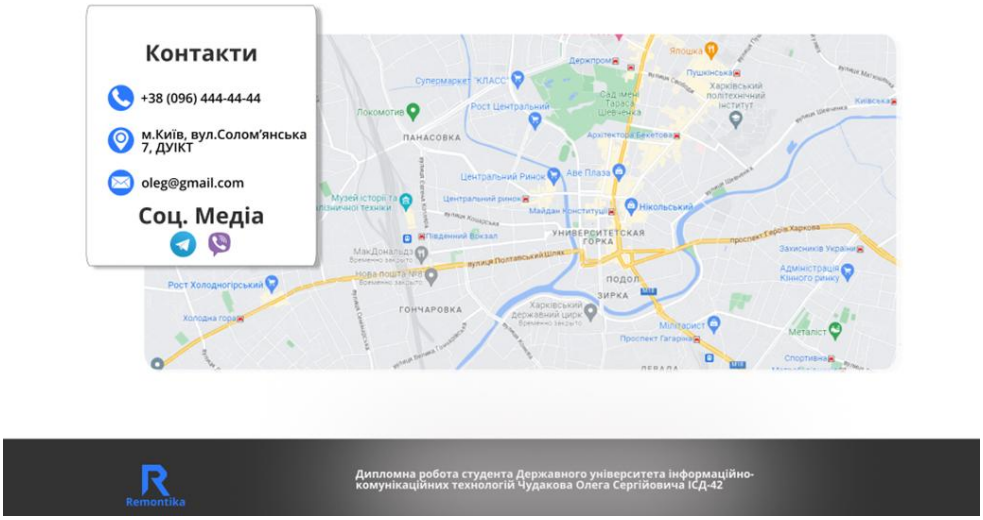

Рис. 3.5 Контактна інформація, мапа та футер сайту Remontika.

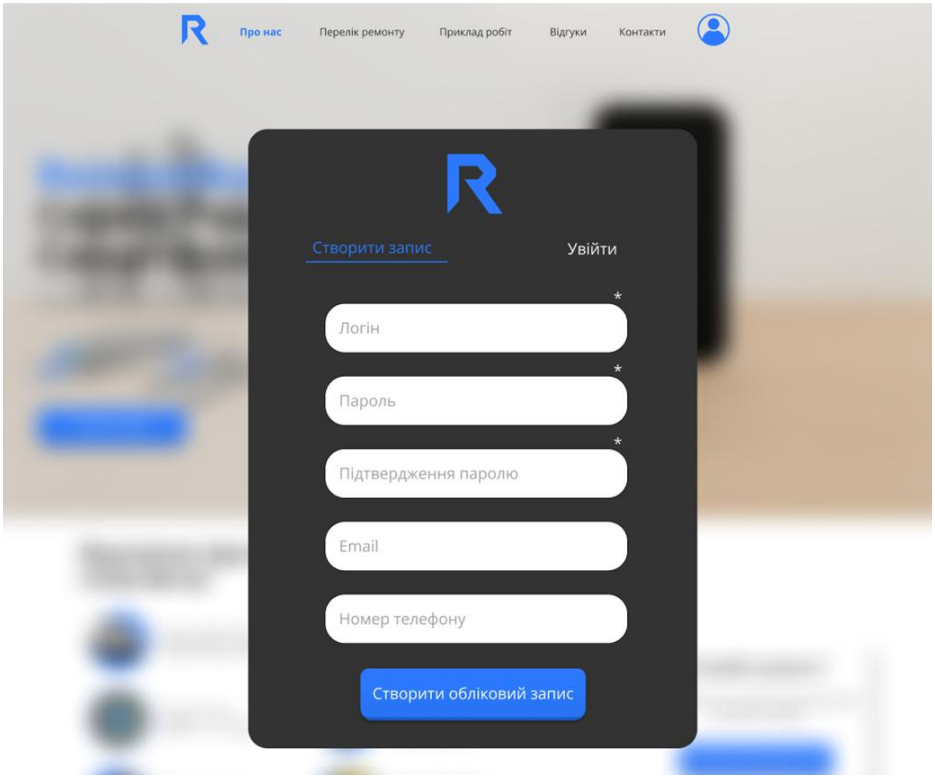

Рис.3.6 Форма створення облікового запису у модальному вікні.

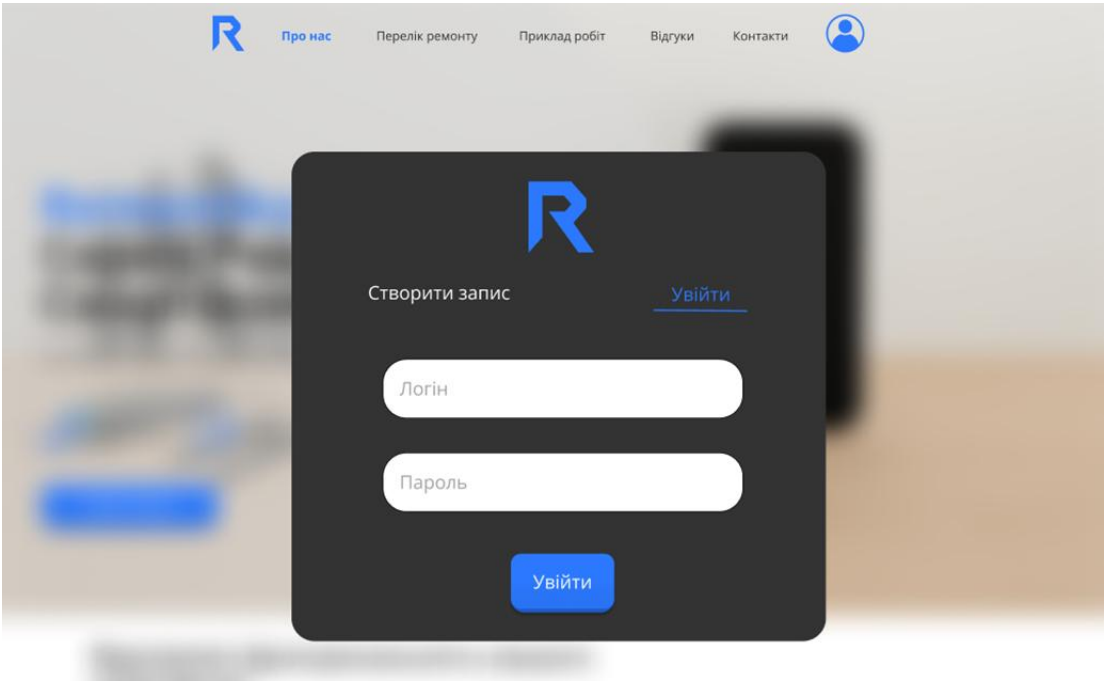

Рис.3.7 Інтерфейс авторизації для входу до персонального кабінету.

Окрему увагу в загальній структурі інтерфейсу варто приділити елементу авторизації та регістрації користувача, що реалізований у вигляді інтерактивної іконки у правому верхньому куті навігаційної панелі. При натисканні на цю іконку відкривається модальне вікно, яке дозволяє користувачеві виконати одну з двох основних дій: створити новий обліковий запис або увійти до вже існуючого профілю. У режимі створення облікового запису користувачеві пропонується заповнити шість основних полів: логін, пароль, підтвердження пароля, електронна адреса та номер телефону. Усі поля мають мінімалістичний дизайн із заокругленими краями, адаптивне масштабування і маркування обов'язковості введення. Завершується форма виразною кнопкою «Створити обліковий запис», яка виділяється насиченим синім кольором і має візуальну ієрархію першого рівня дії (primary action). При перемиканні на вкладку «Увійти» інтерфейс спрощується до двох ключових полів — логіну та паролю — з тією ж візуальною логікою та кнопкою «Увійти» нижче. Обидві форми реалізовані у вигляді модального вікна із ефектом розмиття фону, що дозволяє зосередити увагу користувача виключно на дії входу або реєстрації.

3.2 Інформаційна панель користувача

Після успішної авторизації користувач потрапляє до персоналізованої інформаційної панелі, яка виконує роль головного навігаційного і функціонального центру взаємодії з платформою (Рис 3.8).

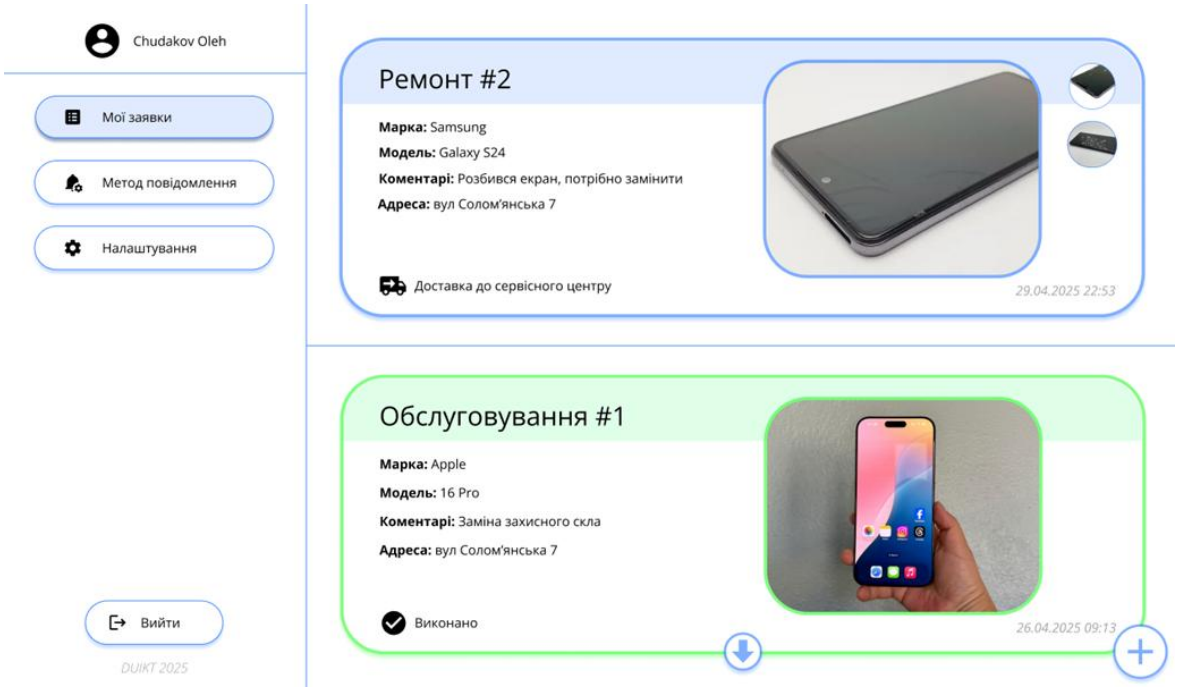

Рис.3.8 Персональна інформаційна панель користувача з активними та завершеними заявками.

Ліву частину екрана займає вертикальне меню, що дозволяє переходити між основними розділами: «Мої заявки», «Метод повідомлення» та «Налаштування» (Рис. 3.8). У нижній частині цього блоку розміщено кнопку виходу з системи та підпис із назвою навчального закладу, що відсилає до навчального походження проєкту. Центральна частина інтерфейсу зосереджена на виведенні поданих заявок користувача. Кожна заявка відображається у вигляді окремої інформаційної картки, оформленої кольоровим акцентом, який візуально відповідає статусу заявки. Наприклад, активна заявка «Ремонт #2» обрамлена світло-блакитним контуром і містить усі ключові деталі: бренд пристрою, модель, опис проблеми, а

також адресу, куди користувач замовив кур’єра. Додатково вказано статус зявки – доставка до сервісного центру, що формує очікування користувача стосовно подальших дій. Праворуч розміщено зображення пристрою та кнопки для перегляду додаткових фото. У нижньому правому куті зазначено точну дату й час подачі заявки.

Нижче розміщено заявку «Обслуговування #1», яка має зелений контур і позначена як завершена. Цей блок відображає вже виконаний запит на заміну захисного скла смартфону. Завершеність заявки підтверджується відповідною іконкою та міткою «Виконано». Як і в попередньому випадку, заявка супроводжується фотографією пристрою, яку прикрипів користувач у процесі створення заявки.

У нижньому правому куті екрана розташовано плаваючу кнопку зі знаком «плюс», яка дозволяє ініціювати створення нової заявки. Це відповідає сучасним підходам до дизайну взаємодії (FAB — Floating Action Button), забезпечуючи постійний доступ до основної функції сервісу незалежно від місця прокручування сторінки.

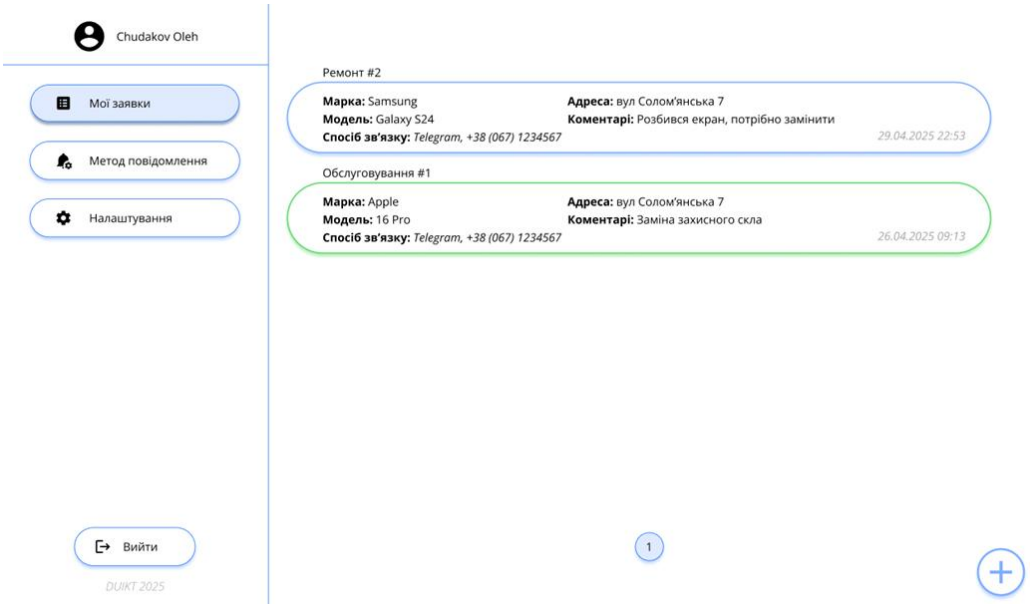

Рис.3.9 Компактний режим перегляду заявок після активації перемикача зі стрілкою.

Крім розгорнутого вигляду заявок, персональна панель користувача містить додатковий механізм навігації, реалізований у вигляді кнопки зі стрілкою вниз, що розташована по центру нижньої частини екрана (Рис. 3.9). Після активації кнопки відкривається мінімалістичне представлення всіх запитів, яке містить лише ключову інформацію: марку й модель пристрою, адресу ремонту, спосіб зв’язку, короткий опис проблеми та часову мітку.

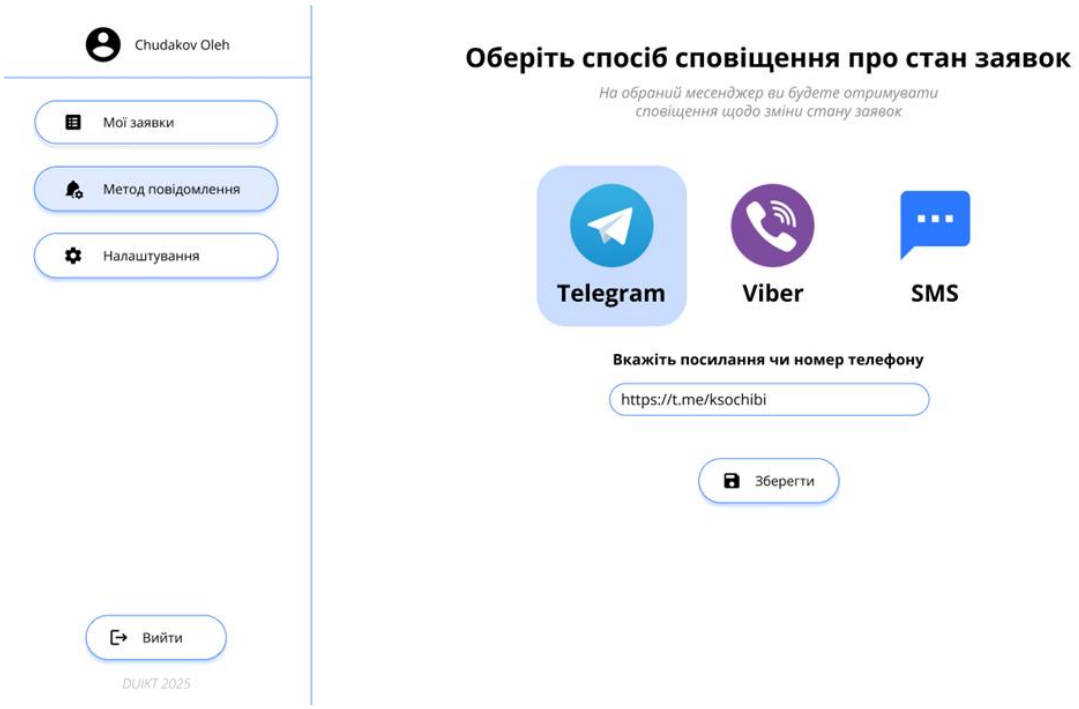

Рис.3.10 Інтерфейс вибору методу сповіщення про зміну статусу заявки.

Однією з ключових функцій системи Remontika є механізм інформування користувача про зміну статусу заявки в режимі реального часу (Рис. 3.10). Для цього в інтерфейсі особистої панелі передбачено окремий розділ — «Метод повідомлення», що дозволяє змінити канал отримання сповіщень, оскільки вибір каналу обирається користувачем під час створення заявки. Розділ реалізований у вигляді окремої сторінки з трьома основними варіантами комунікації: Telegram, Viber та SMS та полем для введення посилання чи номеру телефону.

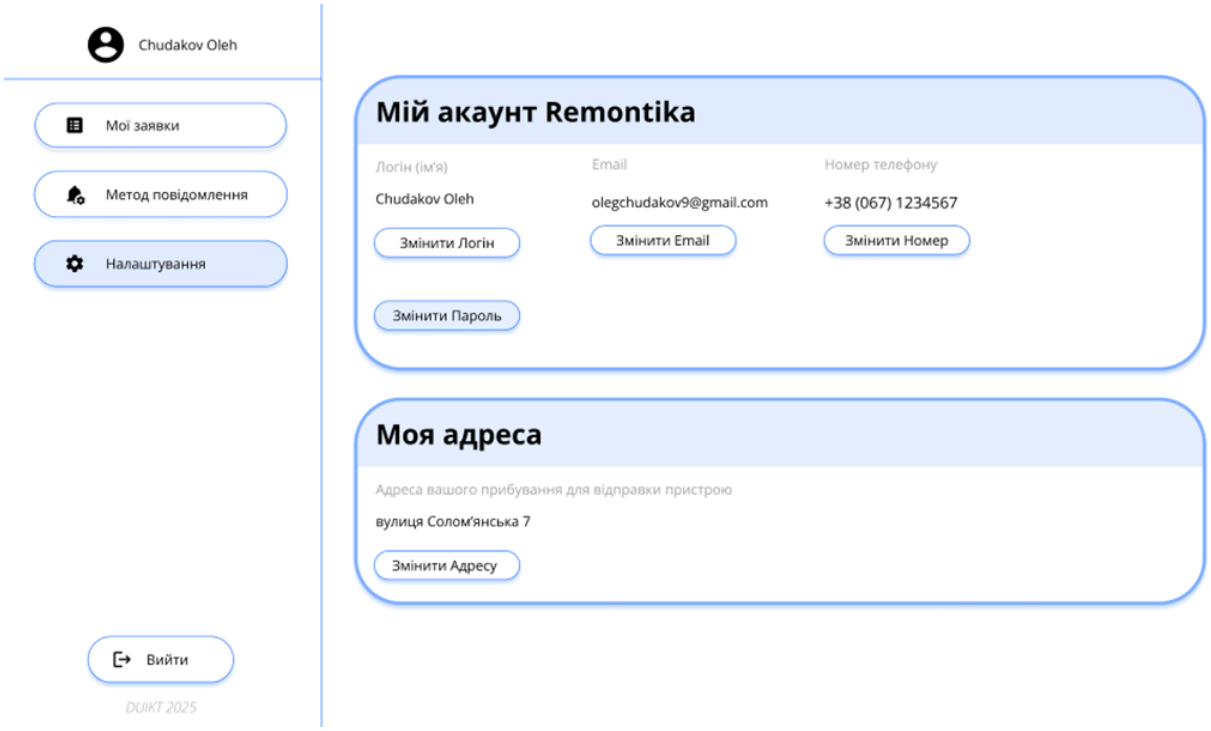

Рис.3.11 Розділ налаштувань акаунту та адреси доставки користувача.

Розділ «Налаштування» призначений для управління персональними даними користувача та конфігурації адреси, необхідної для логістичних операцій. Інтерфейс реалізовано у вигляді двох основних блоків: “Мій акаунт Remontika” та “Моя адреса”, кожен з яких має чітке візуальне відокремлення, мінімалістичний стиль та логічну структуру. У верхній частині відображається блок персональної інформації. Він містить логін користувача, електронну пошту та номер телефону. Кожне з цих полів супроводжується окремою кнопкою редагування — «Змінити Логін», «Змінити Email», «Змінити Номер», що дозволяє оперативно оновити відповідну інформацію.

Важливо відзначити, що компонування інтерфейсу розділу налаштувань відповідає принципам редагування in-place — користувач змінює лише ті поля, які необхідні, без перезавантаження сторінки. Це позитивно впливає на зручність, швидкість роботи та рівень контролю користувача над особистими параметрами.

Загалом інтерфейс інформаційної панелі є прикладом ефективної реалізації принципів персоналізації, логічного розділення функцій та наочного подання

даних. Користувач має змогу оперативно переглядати поточні заявки, відстежувати статуси, переглядати історію обслуговування та ініціювати нові запити, не виходячи за межі єдиного простору.

Процес створення заявки на ремонт

Розглянемо повний цикл створення заявки на ремонт через інтерфейс особистого кабінету користувача. На наступних рисунках (Рис. 3.12–3.17) покроково продемонстровано, як користувач взаємодіє з формою заповнення звернення, вказуючи тип пристрою, характер несправності, метод зворотного зв’язку, адресу доставки, здійснюючи фінальну перевірку введених даних, а також вигляд створеної заявки.

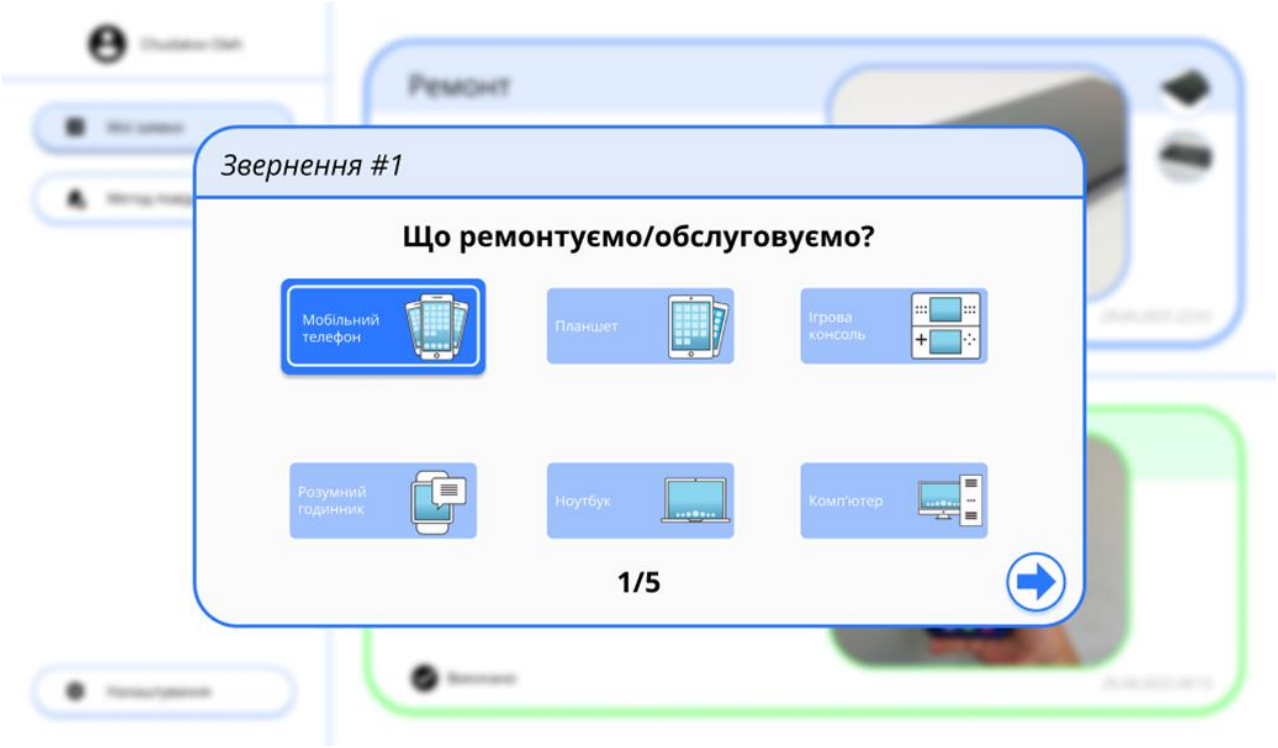

Рис.3.12 Вибір типу пристрою для ремонту або обслуговування.

Рисунок 3.12 демонструє перший крок процесу створення звернення після натискання кнопки із «плюсом». Користувачеві пропонується обрати тип пристрою, який потребує обслуговування або ремонту. Усі варіанти візуально представлені у вигляді карток із піктограмами та підписами (смартфон, планшет, ноутбук, розумний годинник, ігрова консоль тощо).

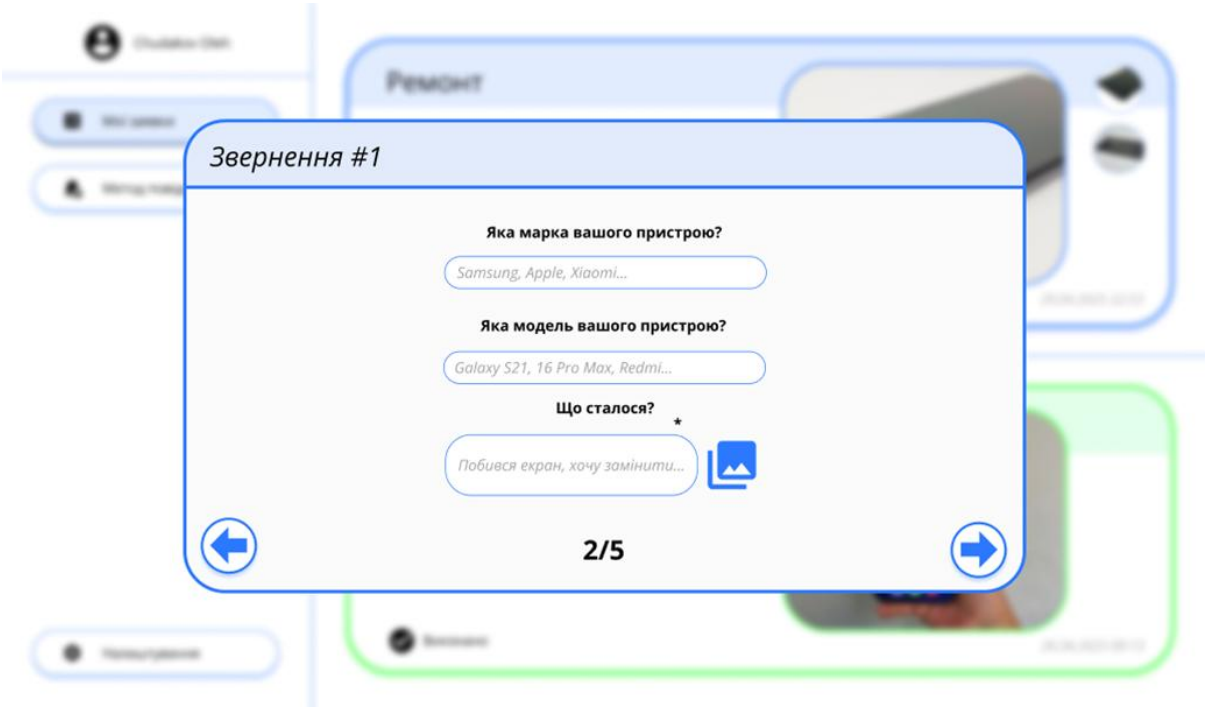

Рис.3.13 Введення марки, моделі пристрою, опису несправності та завантаження фото.

Рисунок 3.13 ілюструє другий крок. Тут користувач має вказати бренд і модель пристрою, а також описати проблему у відповідному полі — наприклад, “Побився екран, хочу замінити”. Є також можливість прикріпити фотографію пошкодженого пристрою для полегшення попередньої діагностики фахівцем.

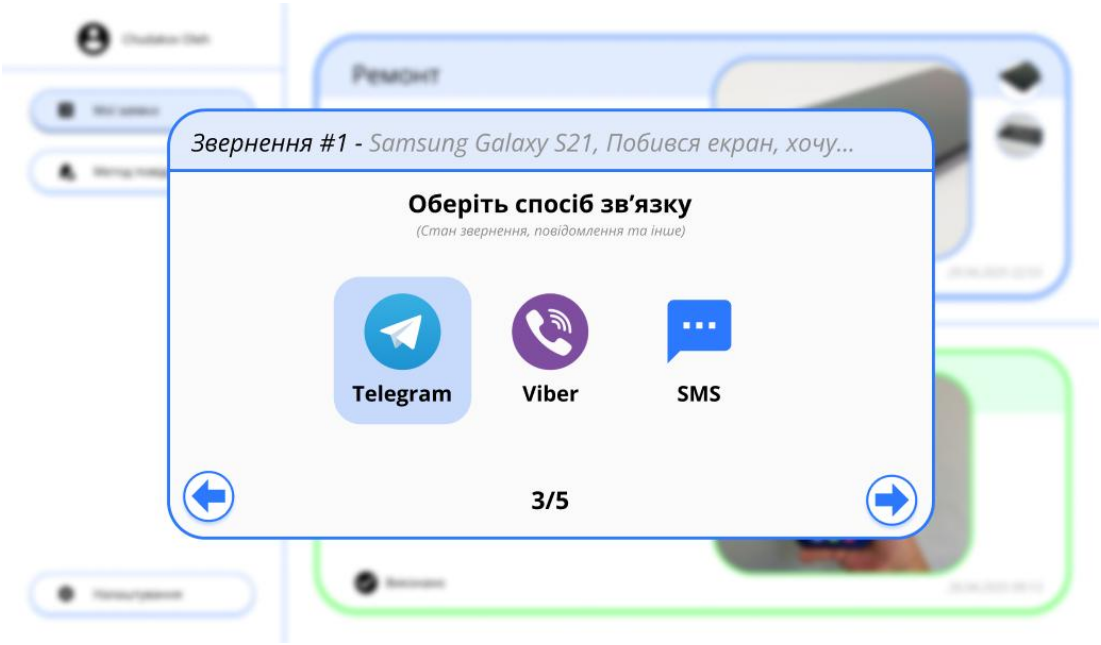

Рис.3.14 Вибір способу зворотного зв’язку.

Рисунок 3.14 зображує третій етап, де користувач обирає зручний для себе метод сповіщення — Telegram, Viber або SMS. Це важливий етап персоналізації обслуговування, адже надалі оновлення щодо стану заявки надсилатимуться через обраний канал зв’язку.

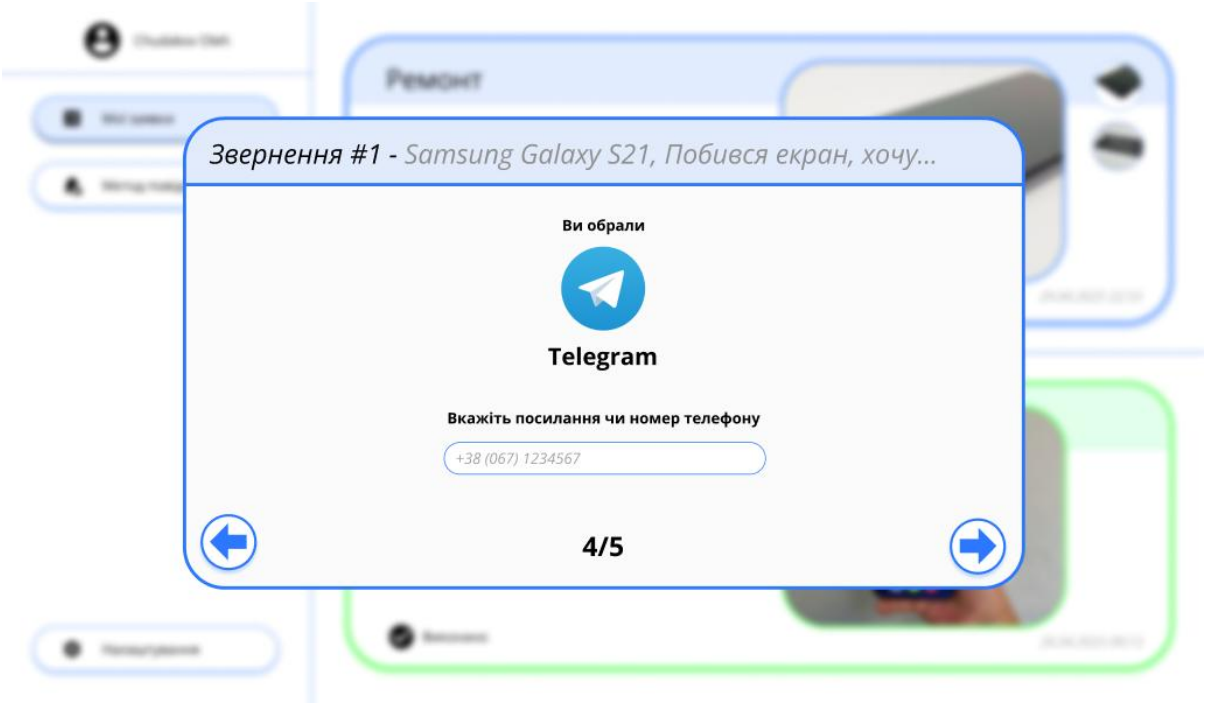

Рис.3.15 Введення контактних даних для обраного способу зв’язку.

Рисунок 3.15 деталізує вибір методу зв’язку. Наприклад, у разі вибору Telegram користувачеві пропонується ввести відповідне посилання на профіль або номер телефону, з яким пов'язаний акаунт у цьому месенджері. Таким чином система фіксує канал сповіщень для подальшого використання Firebase Cloud Function.

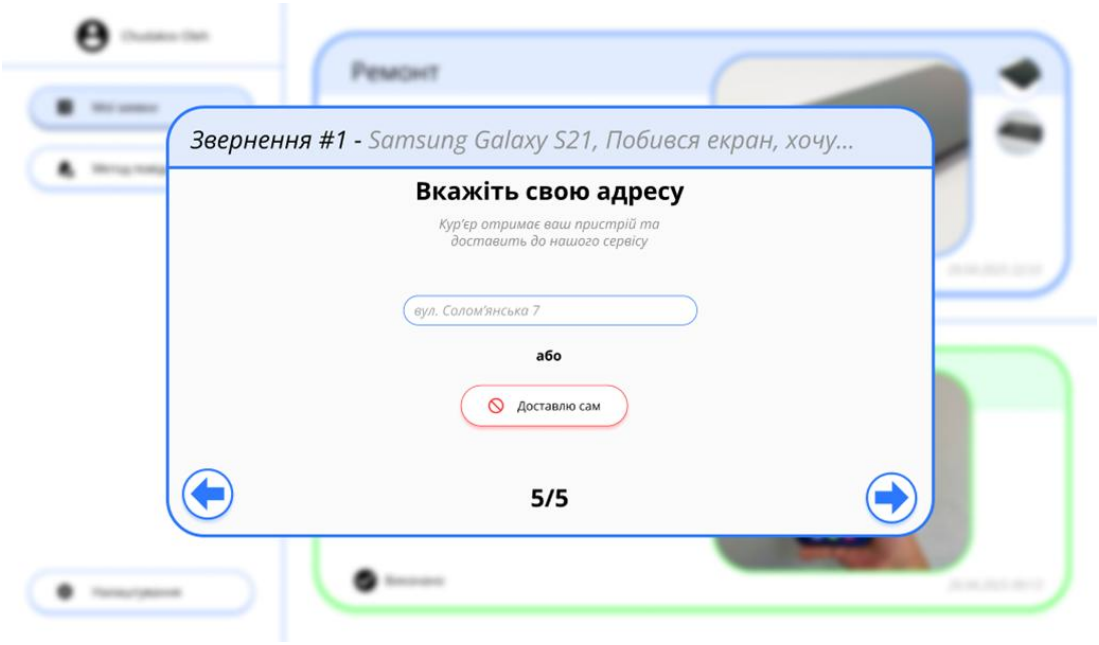

Рис.3.16 Введення адреси для виклику кур’єра або вибір самостійної доставки.

Рисунок 3.16 демонструє п’ятий крок: введення адреси доставки. Користувач може або вписати свою адресу для виклику кур'єра, або обрати опцію самостійної доставки пристрою до сервісного центру. Цей вибір фіксується в базі даних і впливає на подальшу логістику.

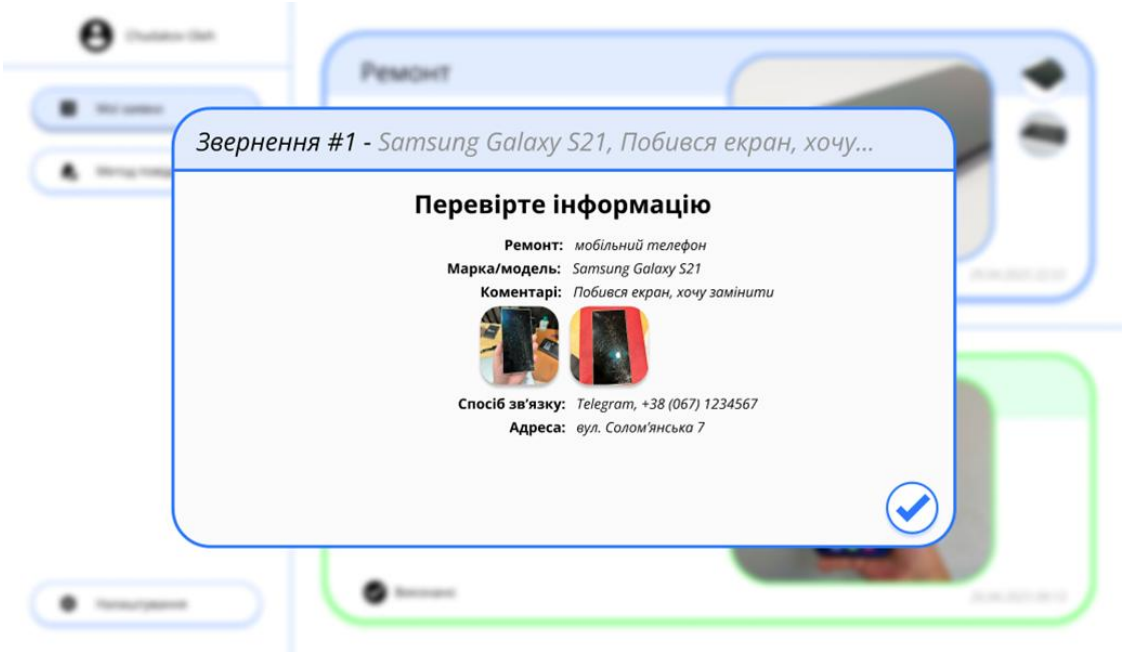

Рис.3.17 Підсумковий перегляд усіх введених даних.

Рисунок 3.17 представляє фінальний перегляд заявки. Весь введений користувачем зміст зібрано на одному екрані — тип ремонту, бренд та модель, фото пошкоджень, спосіб зв’язку, адреса.

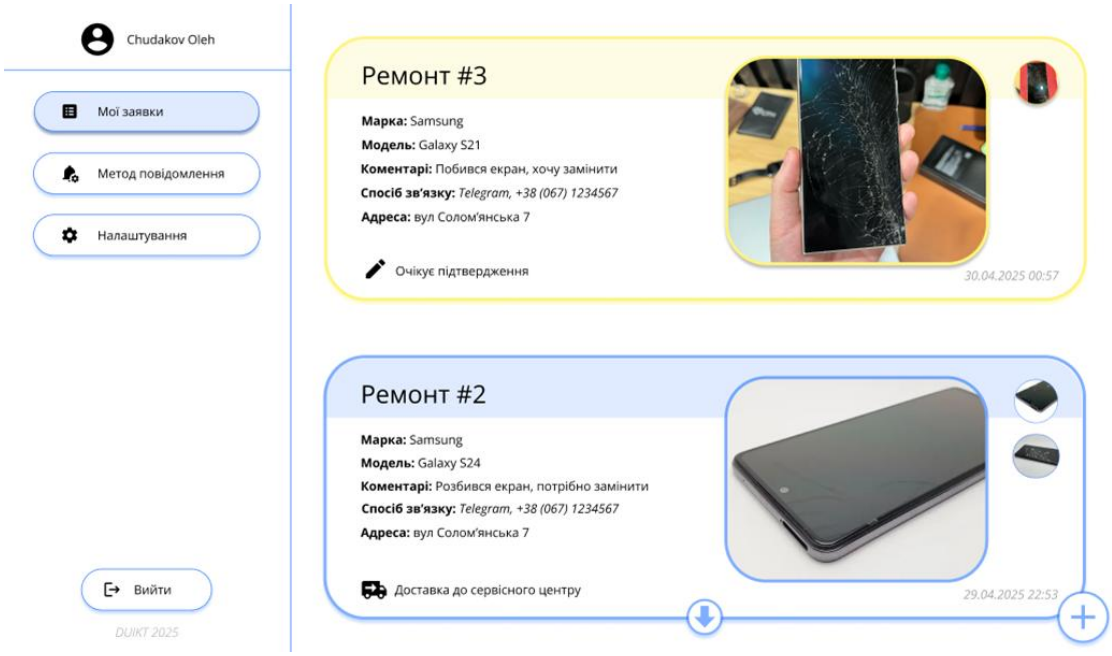

Рис.3.18 Відображення нової заявки в особистому кабінеті користувача після успішного створення.

Після створення заявки вона автоматично з’являється у персональному кабінеті користувача (Рис. 3.18). Кожне звернення представлено у вигляді окремого блоку з чіткою візуальною розміткою: зверху зазначено номер заявки, нижче вказані марка та модель пристрою, коментар із описом проблеми, обраний спосіб зв’язку, адреса доставки, а також прикріплені фотографії. Нижче знаходиться мітка із поточним статусом (наприклад, «Очікує підтвердження»), який змінюється в режимі реального часу. Таким чином, користувач має змогу не лише отримувати повідомлення у вибраному месенджері, а й оперативно відстежувати оновлення безпосередньо на сайті, що забезпечує прозорість і зручність взаємодії з сервісом.

Цей покроковий візуально-орієнтований підхід дозволяє знизити кількість помилок при введенні, підвищити зручність використання та адаптувати інтерфейс до потреб широкого кола користувачів.

3.3 Адміністративна панель

У цьому підрозділі буде детально розглянуто інтерфейс адміністративної панелі — окремої частини системи, призначеної для менеджменту заявок працівниками сервісу. Я проаналізую, як саме адміністратор опрацьовує кожну заявку від моменту її створення користувачем і до фінального етапу підтвердження доставки. Особливу увагу буде приділено логіці зміни статусів, реакції системи на кожну зміну та механізму сповіщення користувача через обраний месенджер. Цей підхід дозволяє побачити, як вся система працює як єдиний узгоджений механізм, де дії адміністратора миттєво синхронізуються з клієнтським інтерфейсом.

Рис.3.19 Початковий стан нової заявки.

Після того, як користувач створює заявку, вона автоматично потрапляє в адміністративну панель до категорії «Очікують підтвердження» (Рис. 3.19). Адміністратор бачить ключову інформацію: бренд і модель пристрою, адресу, спосіб зв’язку та прикріплені фото пошкодження. У цьому стані він має два

варіанти дій: натиснути кнопку «Наступний етап», якщо заявка виглядає коректно, або «Відхилити», якщо є підозра на спам, тестове або жартівливе звернення. Інтерфейс підсвічений жовтим кольором, щоб візуально вирізнити ще не опрацьовані замовлення.

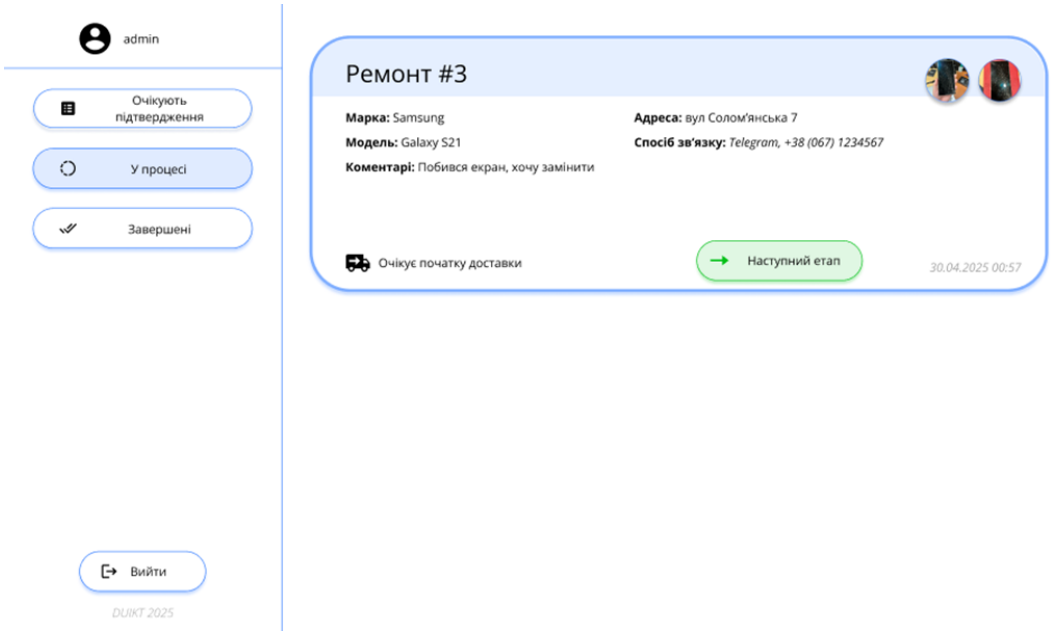

Рис.3.20 Заявка зі статусом «Очікує початку доставки».

Після підтвердження заявки вона переміщується до розділу «У процесі», змінює колір на блакитний, і система очікує, поки кур'єр отримає пристрій у клієнта (Рис. 3.20). Тут адміністратор уже не має змоги відхилити замовлення — є лише кнопка «Наступний етап», яка дозволяє оновити статус, коли пристрій фізично прибув до сервісного центру. Система автоматично надсилає повідомлення користувачу про зміну статусу.

Коли пристрій отримано, адміністратор натискає «Наступний етап», і заявка переходить до фази активного ремонту. Відповідно, фон змінюється на яскраво-блакитний із іконкою гайкового ключа. Цей статус сигналізує, що пристрій зараз знаходиться в обробці, і користувач про це повідомляється в месенджері, а також бачить зміну на сайті в реальному часі (Рис 3.21).

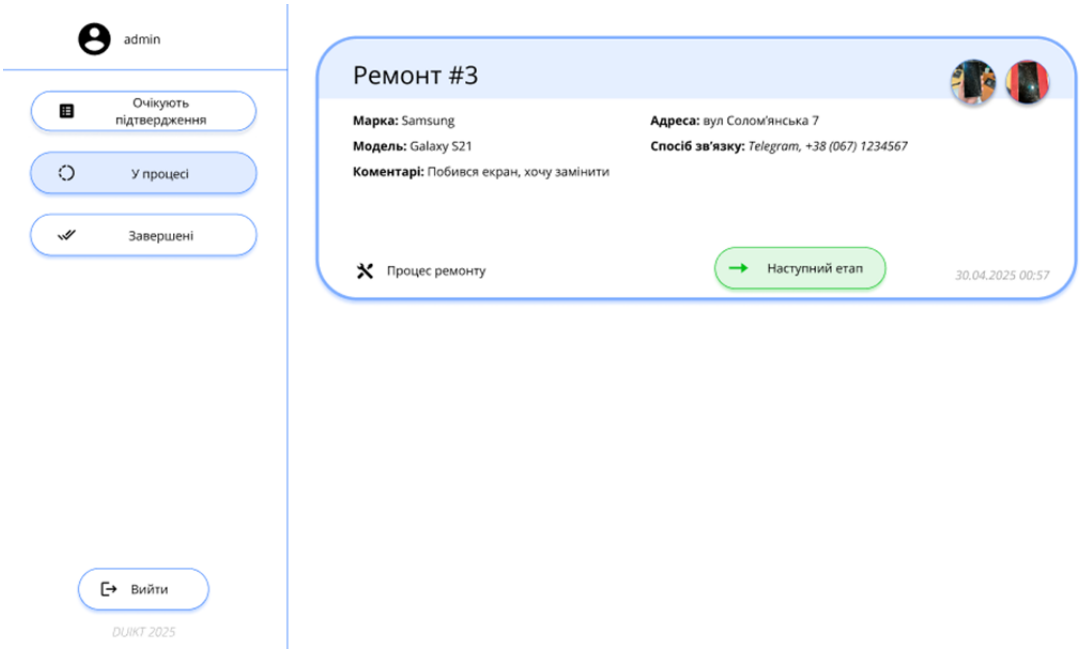

Рис.3.21 Заявка зі статусом «Процес ремонту».

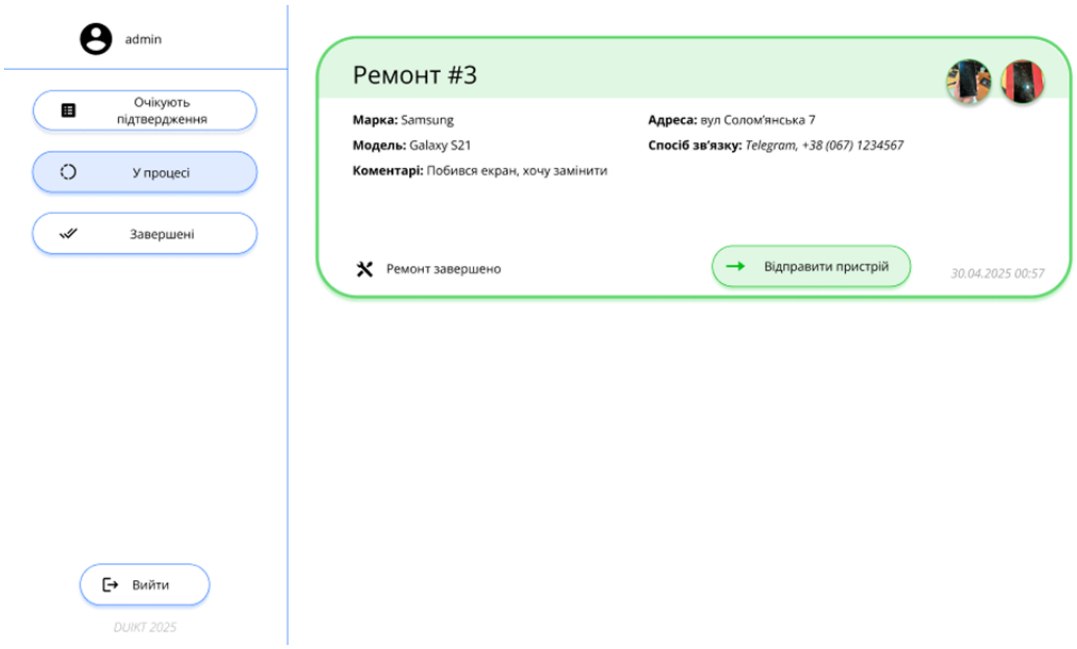

Рис.3.22 Завершення ремонту, кнопка «Відправити пристрій».

Після завершення ремонту адміністратор знову натискає кнопку «Наступний етап», і заявка переходить у стан «Ремонт завершено». Колір фону змінюється на світло-зелений (Рис. 3.22). У цьому інтерфейсі з'являється кнопка «Відправити пристрій», яка підтверджує, що пристрій готовий до доставки клієнту. Натискання цієї кнопки оновлює статус і запускає механізм сповіщення користувача.

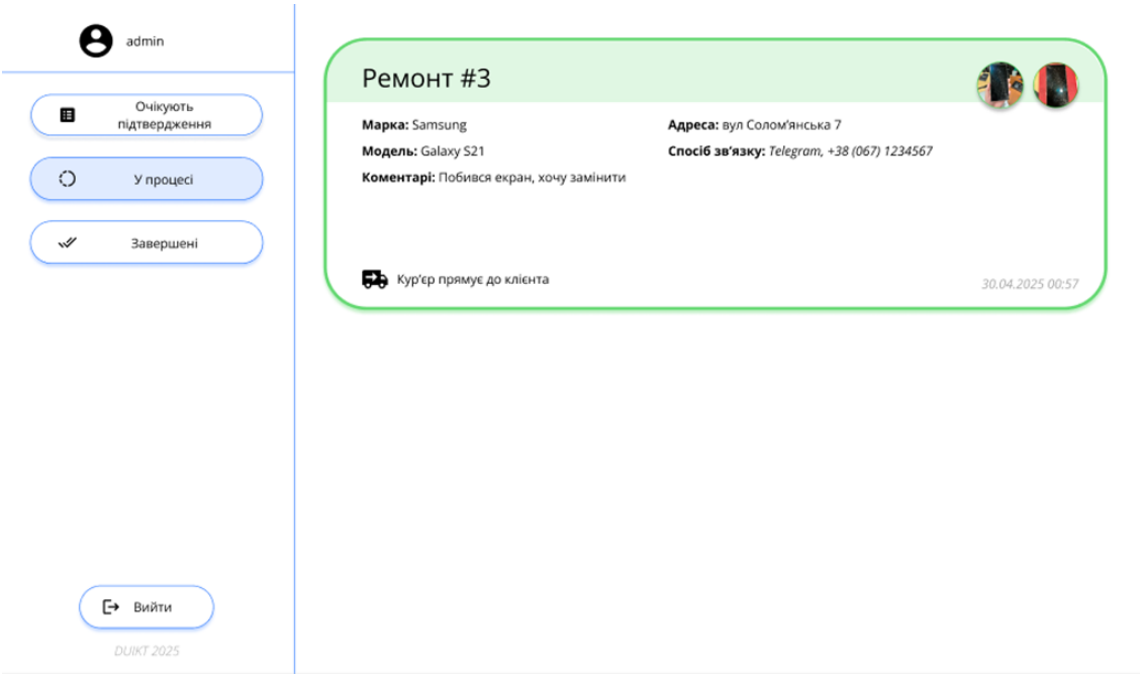

Рис.3.23 Статус «Кур’єр прямує до клієнта».

Після відправлення пристрою назад клієнту, заявка отримує статус «Кур’єр прямує до клієнта». Це останній активний етап із боку адміністратора. Інтерфейс все ще зелений, щоб підкреслити позитивний розвиток подій, і має значок доставки. У цей момент користувач отримує фінальне повідомлення в месенджері про очікувану доставку.

Рис.3.24 Всі завершені заявки у меню «Завершені».

Коли пристрій доставлений, і користувач підтвердив отримання, заявка переноситься до категорії «Завершені». У цій панелі адміністратор може бачити повний список усіх виконаних замовлень із можливістю перегляду деталей у згорнутому форматі. Це дозволяє зручно зберігати історію звернень.

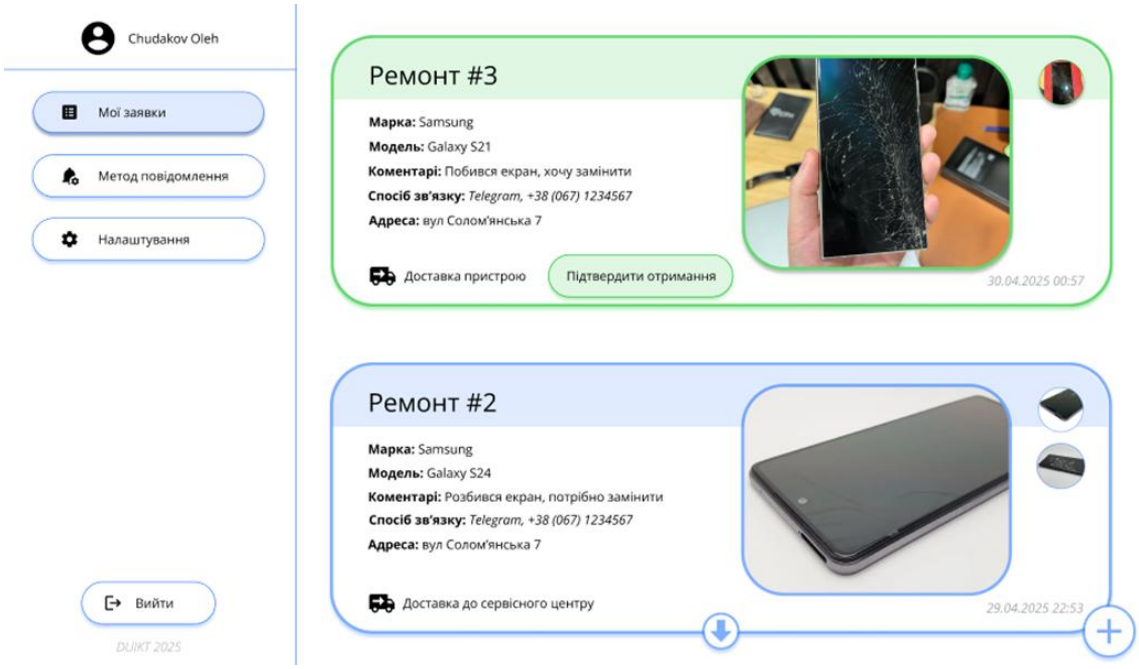

Рис.3.25 Відображення фінального стану заявки у кабінеті користувача.

Після доставки пристрою заявка в кабінеті користувача змінює статус на «Доставка пристрою» із кнопкою «Підтвердити отримання». Саме натискання цієї кнопки завершує повний цикл обробки заявки. Користувач бачить оновлений візуальний статус, а також отримує підтвердження в обраному месенджері (Telegram, Viber або SMS). Тільки після цього заявка остаточно переноситься до завершених.

Як результат, описана система забезпечує прозорий, автоматизований та зручний процес обробки звернень, де кожен крок супроводжується як візуальним індикатором, так і сповіщенням у месенджері, що значно підвищує довіру та зручність для користувача порівняно із сьогоднішнями рішеннями у бізнесі.

3.4 Інтеграція з Firebase

У якості бази даних для проєкту було обрано **Cloud Firestore**, що є частиною Firebase — платформи для розробки мобільних та вебдодатків від Google. Цей вибір зумовлений гнучкою структурою зберігання, вбудованою підтримкою реального часу та масштабованістю без потреби в обслуговуванні серверів.

Авторизація користувачів

Механізм аутентифікації реалізовано за допомогою модуля Firebase Authentication, який підтримує email/пароль. Коли користувач заповнює реєстраційну форму, дані надсилаються до Firebase Auth, який створює нового користувача в системі. Одразу після цього в базу Firestore вноситься запис з базовою інформацією: логін, email, номер телефону, а також автоматично присвоюється роль (role: "user"). У разі створення акаунта адміністратором, роль вказується як "admin" — це забезпечує розмежування функціоналу між звичайними користувачами та працівниками сервісу. У випадку авторизації вже створеного користувача, система перевіряє збіжність із вже існуючими даними, та якщо дані збігаються авторизує користувача у свій акаунт. Дані автентифікації ніколи не зберігаються локально в додатку — лише токени, які Firebase самостійно оновлює за сесіями, що відповідає сучасним вимогам безпеки.

Запис та зчитування заявок

Після авторизації користувач має змогу створити нову заявку. Цей процес запускає POST-запит до API-ендпоінта, який викликає серверну функцію, що створює новий документ у колекції requests. У документі зберігаються наступні поля: userId, deviceBrand, deviceModel, problemDescription, status, submittedAt, photoUrl та інші.

Структура Firestore наведена нижче:

/users/{userId}

/requests/{requestId}

/notifications/{notificationId}

/adminLogs/{logId}

Для зчитування даних про заявки, використовується getDocs() з умовами фільтрації, наприклад, по userId. Це дозволяє відображати лише ті заявки, що належать конкретному користувачу, не зачіпаючи записи інших.

Оновлення статусу заявки

Адміністратор взаємодіє з інтерфейсом, який через API або Cloud Function змінює статус заявки в документі requests. Після кожного оновлення поля status, Cloud Function активує тригер, що записує журнал дій у adminLogs, а також створює нове повідомлення у notifications, яке потім використовується для сповіщення користувача.

Реалізація Realtime-механізму

На стороні клієнта в компонентах React використовуються реактивні підписки через onSnapshot(). Це означає, що як тільки змінюється документ у Firestore — наприклад, оновлюється статус заявки — ці зміни миттєво з’являються в інтерфейсі користувача без необхідності ручного оновлення сторінки.

Безпека і права доступу

У проєкті налаштовано Firebase Security Rules, які жорстко обмежують доступ: Користувач може читати й писати лише власні документи /requests та /notifications; Доступ до /adminLogs та перегляду всіх requests мають лише користувачі з role == "admin".

Таким чином, чітко реалізовано розмежування прав і мінімізовано ризики витоку або маніпуляцій з боку звичайних користувачів. Ця інтеграція з Firebase забезпечує повноцінний бездокерний і серверлес підхід до побудови сучасного вебсервісу з високим ступенем безпеки, швидкості реагування та зручного адміністрування.

3.5 Система сповіщень

Перед впровадженням функціональності сповіщень користувача про зміну статусу заявки, необхідно було створити спеціального Telegram-бота, який виступає як канал доставки повідомлень.

Створення Telegram-бота

Процес створення починається зі стандартного інструмента Telegram — BotFather. Це офіційний бот для керування іншими ботами. Користувачеві достатньо знайти @BotFather у Telegram, відкрити діалог і ввести команду /newbot (Рис 3.26).

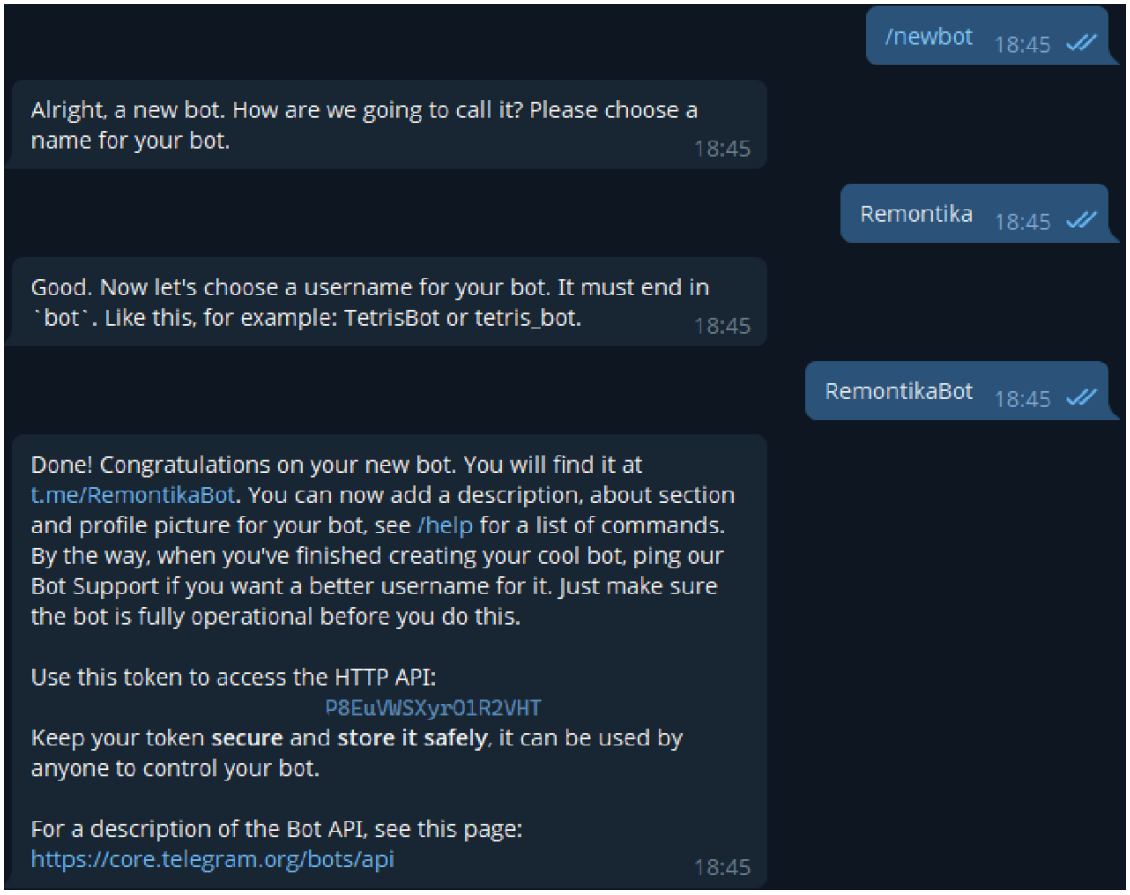

Рис.3.26 Процес створення власного боту у Telegram.

У ході створення необхідно:

- Присвоїти ім’я бота, яке буде відображатися у профілі.
- Вказати унікальний юзернейм бота, який обов’язково має закінчуватися на bot.
- Завантажити аватар, який створює впізнаваність і брендованість бота серед користувачів.

- Після завершення налаштувань, BotFather видає токен авторизації — це секретний ключ, за допомогою якого можна надсилати запити до Telegram API від імені створеного бота.

Цей токен зберігається в системних змінних середовища (наприклад, .env_local) і ніколи не потрапляє у відкритий код, що є важливим принципом безпеки.

Бот був повністю кастомізований — від імені, що відповідає бренду Remontika, до зображення профілю та привітального повідомлення, яке користувач бачить після старту взаємодії.

Після створення Telegram-бота наступним етапом стала реалізація зв’язку між змінами в базі даних і відправкою повідомлень. Основою для цього слугує сервіс Firebase Functions — це безсерверна технологія, що дозволяє запускати код у відповідь на події в екосистемі Firebase.

Реалізація логіки сповіщень

Було створено спеціальний хмарний триггер, який спрацьовує при кожному оновленні документа заявки (Request) у Cloud Firestore. Тобто, коли адміністратор змінює статус заявки (наприклад, з "Очікує підтвердження" на "У процесі ремонту"), ця зміна активує функцію Firebase, яка:

1. Зчитує оновлений документ.
2. Визначає telegramId користувача, який пов'язаний із цією заявкою.
3. Формує текст повідомлення відповідно до нового статусу (наприклад: "Ваш пристрій прийнято в ремонт. Очікуйте оновлень.").
4. Надсилає це повідомлення за допомогою запиту на Telegram Bot API через fetch або axios.

Механізм доставки

Користувач ще на етапі створення заявки вибирає бажаний спосіб отримання сповіщень — Telegram, Viber або SMS. Якщо обрано Telegram, то у форму вводиться посилання або ідентифікатор користувача (наприклад, @username або telegramId, який був отриманий під час авторизації або взаємодії з ботом).

Після цього Telegram ID зберігається в полі telegramId документа користувача (User). Всі подальші зміни заявки автоматично ініціюють надсилання повідомлення на цей ID. Користувач отримує миттєве повідомлення з оновленням статусу, навіть не заходячи на сайт.

У результаті система забезпечує повну інтеграцію між базою даних і месенджером Telegram, гарантує актуальність інформації для користувача та створює зручну й сучасну модель зворотного зв’язку.

3.6 Адаптивність і кросбраузерність інтерфейсу

Одним з ключових вимог до сучасного вебінтерфейсу є його адаптивність та кросбраузерність, що безпосередньо впливає на зручність користувача, доступність сервісу на різних пристроях і відповідність актуальним стандартам веброзробки. У проєкті Remontika ці принципи були закладені вже на етапі проектування дизайну, що дозволяє на етапі реалізації забезпечити однаково стабільну роботу на смартфонах, планшетах і десктопах незалежно від браузера.

Адаптивність: основа під мобільну аудиторію

Оскільки значна частина користувачів звертається до сервісів ремонту з мобільних пристроїв, дизайн інтерфейсу було реалізовано за принципами mobile-first. Це означає, що першочергово розробка ведеться під екрани з невеликою шириною, з поступовим масштабуванням компонентів для більших дисплеїв. Такий підхід дозволяє уникнути ситуацій, коли мобільна версія є лише урізаною копією десктопної — навпаки, вона є повноцінною та самодостатньою.

Використання Tailwind CSS стало ключовим інструментом для впровадження адаптивності. Tailwind дозволяє задавати стилі безпосередньо в HTML з урахуванням breakpoints — умов для певних розмірів екрана (sm, md, lg, xl). Це дало змогу легко адаптувати відступи, розміри шрифтів, відображення колонок та структуру блоків під різні розширення. Наприклад, на мобільних пристроях елементи розміщуються в один стовпчик, а на десктопах — у сітку з кількох колонок, без дублювання коду.

Також дизайн у Figma був створений з урахуванням адаптивності: у макетах задані респонсивні точки (480px, 768px, 1024px і 1280px), які дозволили

спроєктувати логіку перебудови компонування інтерфейсу. Це забезпечує коректну роботу інтерфейсу на всіх популярних типах екранів.

Кросбраузерність: стабільність на різних платформах

Під кросбраузерністю розуміється здатність сайту однаково відображатися та функціонувати у різних браузерах: Google Chrome, Opera, Microsoft Edge, Safari, а також менш поширених — таких як Mozilla Firefox, Brave чи мобільний Samsung Internet. Незалежно від рушія відображення (Chromium, WebKit, Gecko), сайт повинен коректно реагувати на стилі, анімації, події кліку та адаптацію.

У проєкті використовується такі підходи для забезпечення кросбраузерності:

- Використання Tailwind CSS, який генерує стандартизований CSS-код без використання застарілих властивостей.
- Відмова від нестабільних або експериментальних CSS-функцій.
- Створення UI-компонентів із перевірених бібліотек React, що підтримуються спільнотою та тестуються під різні браузери.
- Обхід специфічних ефектів, які можуть не підтримуватися старими версіями браузерів (наприклад, backdrop-фільтри чи CSS grid у IE11).

Тестування і оптимізація

Для перевірки адаптивності використовувались DevTools браузерів — зокрема симуляція пристроїв у Google Chrome (Pixel, iPhone, iPad тощо). Це дозволяє в реальному часі протестувати, як поводяться компоненти на різних розмірах екранів. Tailwind також забезпечує швидке перемикання класів залежно від ширини, що пришвидшує процес оптимізації.

Для кросбраузерного тестування будуть застосовані такі підходи:

- Вручну: перевірка сайту в Chrome, Firefox, Safari, Edge на різних ОС (Windows, Linux, macOS, Android).
- Автоматизовано: використання сервісу BrowserStack або Responsively App (безкоштовний емулятор).
- Використання утиліт типу postcss-normalize, які додають CSS-уніфікатори для поведінки елементів у всіх браузерах однаково.

Таким чином, закладена структура, використання Tailwind CSS, mobile-first-підхід і плановане тестування гарантують, що інтерфейс Remontika є повністю адаптивним та кросбраузерним. Це дозволить забезпечити стабільну роботу сервісу на будь-якому пристрої, незалежно від технічних характеристик користувача, що особливо важливо для досягнення високої якості обслуговування.

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи було створено унікальний для українського ринку цифровий сервіс — веб платформа Remontika, який не має повних аналогів серед сучасних платформ для ремонту смартфонів. Проведене дослідження підтвердило, що жоден із наявних конкурентів не пропонує інтегрованого функціоналу онлайн-реєстрації, формування заявки, відстеження статусу ремонту, інтеграції з месенджерами, миттєвих повідомлень і персонального кабінету в єдиному рішенні. Саме ця сукупність інструментів виділяє розроблену платформу Remontika серед інших — користувач може здійснити повний цикл сервісної взаємодії лише через один сайт, без необхідності телефонувати, особисто відвідувати сервісний центр або перемикатися між різними каналами комунікації.

Усі етапи сервісу — від реєстрації, створення заявки, вибору типу пристрою й опису проблеми, до автоматизованого вибору способу зв’язку, контролю адреси доставки, отримання актуальних сповіщень у Telegram, Viber чи SMS, моніторингу статусу ремонту в реальному часі, — проходять швидко, прозоро й зручно для користувача. Оформлення заявки не займає багато часу та не вимагає спеціальних знань, а система миттєво повідомляє клієнта про кожен етап: підтвердження замовлення, виїзд кур’єра, надходження пристрою до сервісу, початок ремонту, відправку відремонтованого смартфона та фінальне підтвердження отримання. Усі дії адміністратори опрацьовують в окремій панелі, що гарантує швидкість і чіткість сервісу.

Завдяки використанню сучасних технологій — Next.js, Firebase, Tailwind CSS — було реалізовано високий рівень адаптивності, багатоканальної комунікації та безпеки. Інтеграція з хмарними сервісами дала змогу відмовитися від складної інфраструктури, а впровадження моделі реального часу забезпечило актуальність даних без затримок. Адміністративна панель дозволяє ефективно керувати всіма

заявками, історією взаємодії та статусами, що є рідкістю для українських платформ такого типу.

Перспективи подальшого розвитку платформи Remontika є надзвичайно широкими й передбачають впровадження низки інноваційних рішень, які підвищать зручність і конкурентоспроможність сервісу. Зокрема, доцільно реалізувати функцію GPS-відстеження кур'єра у режимі реального часу, що дозволить користувачам бачити точне місцезнаходження свого пристрою під час доставки або повернення. Також перспективним є впровадження динамічного прогнозування часу ремонту: вже на етапі створення заявки клієнт зможе отримати розрахунок орієнтовної тривалості обслуговування на основі актуального завантаження майстерні та складності несправності. В окремих випадках, наприклад, при пошкодженні екрана, система може запропонувати замовлення відповідної запчастини безпосередньо через сайт, відобразити дату її прибуття та надати можливість скоригувати терміни відправлення пристрою у сервісний центр.

Для підвищення ефективності адміністрування платформи варто розробити модуль повноцінної аналітики: це дасть змогу збирати та аналізувати статистику заявок, тривалості виконання замовлень, попиту на окремі послуги, що в свою чергу допоможе оптимізувати ресурси сервісу та оперативно реагувати на зміни ринкових тенденцій. На основі цих даних можна також гнучко регулювати орієнтовний час очікування для користувачів залежно від навантаження.

Окремої уваги заслуговує перспектива впровадження інтелектуальної чат-бот системи на базі нейромереж, наприклад, із використанням ChatGPT. Такий бот, навчений на основі документації сервісу, типових питань користувачів і технічної інформації з ремонту пристроїв, зможе оперативно відповідати на запити клієнтів у Telegram чи на сайті, тим самим зменшуючи навантаження на персонал служби підтримки й підвищуючи якість комунікації. Це відкриває можливості для масштабування платформи без пропорційного зростання витрат на підтримку, а також створює зручний канал отримання допомоги для сучасного цифрового користувача.

Крім того, варто розглянути впровадження системи рейтингів і відгуків щодо якості обслуговування майстрів, автоматизацію розрахунку вартості ремонту у реальному часі залежно від комплектації та складності, а також інтеграцію онлайн-оплати для зручності клієнтів. Усі ці функції можуть суттєво підвищити цінність платформи Remontika для кінцевого споживача та зміцнити її позиції на ринку сервісних послуг.

У підсумку, Remontika — це унікальна для України онлайн-платформа, яка не має повних аналогів на ринку сервісного обслуговування смартфонів і справді дивує інноваційним підходом до взаємодії з клієнтом. Весь процес — від першого звернення до повернення відремонтованого пристрою — автоматизований, прозорий і максимально зручний для користувача, а сучасний функціонал не зустрічається в жодному іншому українському сервісі. Завдяки гнучкій архітектурі й потенціалу впровадження таких новацій, як GPS-відстеження, динамічна аналітика, онлайн-оплата, система рейтингів і інтелектуальна підтримка, Remontika має всі передумови не лише для подальшого масштабування, а й для того, щоб стати орієнтиром цифрової трансформації усього ринку побутових послуг.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Міністерство цифрової трансформації України. “19 млн користувачів в Дія та експорт застосунку в Колумбію і Замбію: головне з презентації Мінцифри у Вашингтоні” [Електронний ресурс]. – Режим доступу: https://itc.ua/ua/novini/19-mln-korystuvachiv-v-diya-ta-eksport-zastosunku-v-kolumbiyu-i-zambiyu-golovne-z-prezentatsiyi-mintsyfry-u-vashyngtoni
2. DataReportal. “Digital 2024: Ukraine” [Електронний ресурс]. – Режим доступу: https://datareportal.com/reports/digital-2024-ukraine
3. DreamHost Blog. “Семантика HTML5” [Електронний ресурс]. – Режим доступу: https://www.dreamhost.com/blog/uk/semantika-html5/
4. RankTracker. “CSS – Cascading Style Sheets” [Електронний ресурс]. – Режим доступу: https://www.ranktracker.com/uk/seo/glossary/css-cascading-style-sheets/
5. Tailwind CSS Documentation [Електронний ресурс]. – Режим доступу: https://tailwindcss.com/
6. GeeksforGeeks. “Tailwind CSS” [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/tailwind-css/
7. Firebase Documentation [Електронний ресурс]. – Режим доступу: https://firebase.google.com/
8. W3Schools Українською. “HTML Підручник” [Електронний ресурс]. Режим доступу: https://w3schoolsua.github.io/html/index.html#gsc.tab=0

ДЕМОНСТРАЦІЙНИЙ МАТЕРІАЛ (ПРЕЗЕНТАЦІЯ)