ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«Розробка дизайну та реалізація вебсайту для перегляду та покупки речей ручної роботи онлайн з використанням технологій, таких як Figma, Angular та MongoDB»**

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

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

_____ Іван ЛІСЕЦЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав: здобувач вищої освіти гр. КНД-41
Іван ЛІСЕЦЬКИЙ
(Ім'я, ПРІЗВИЩЕ)

Керівник: Олена ШИКУЛА
Науковий ступінь, вчене звання *(Ім'я, ПРІЗВИЩЕ)*

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

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти «Бакалавр»

Спеціальність 122 «Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Лісецький Іван Юрійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка дизайну та реалізація вебсайту для перегляду та покупки речей ручної роботи онлайн з використанням технологій, таких як Figma, Angular та MongoDB

керівник кваліфікаційної роботи Олена ШИКУЛА д. ф-м. н., професор
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» 02.2024 р. №36

2. Строк подання кваліфікаційної роботи «31» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, дизайн у Figma, проєкт зі застосуванням Angular, підключена база даних MongoDB

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

1. Розробка дизайну та його верстка
2. Робота з сервісами та технічною частиною сайту
3. Імплементація бази даних

5. Перелік графічного матеріалу:

1. Мета роботи
2. Об'єкт роботи та предмет дослідження
3. Вибір технологій

4. Що таке Figma?
5. Використання Figma у кваліфікаційній роботі
6. Що таке Angular?
7. Роутингова система
8. Сервіси
9. Моделі для бази даних
10. Що таке MongoDB?
11. Використання MongoDB у кваліфікаційній роботі
12. Висновок

6. Дата видачі завдання «23» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Літературний огляд теми	24.02-20.03.24	Виконано
2	Дослідження та аналіз інструментів для дизайну та розробки вебсайтів (Figma, Angular, MongoDB)	21.03-04.04.24	Виконано
3	Розробка концепції дизайну вебсайту за допомогою Figma	05.04-13.04.24	Виконано
4	Реалізація фронтенд частини вебсайту з використанням Angular	13.04-29.04.24	Виконано
5	Реалізація бекенд частини вебсайту з використанням MongoDB	30.04-05.05.24	Виконано
6	Інтеграція фронтенд та бекенд частин, тестування та виправлення помилок	05.05-10.05.24	Виконано
7	Написання дипломної роботи	10.05-30.05.24	Виконано

Здобувач вищої освіти _____ Іван ЛІСЕЦЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

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

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 57 сторінок, 52 рисунків, 11 джерел.

Мета роботи – розробка функціонального вебсайту для перегляду та покупки речей ручної роботи онлайн, використовуючи технології Figma, Angular та MongoDB, а також аналіз їх ефективності та взаємодії у рамках веброзробки.

Об'єкт дослідження – процес розробки вебсайтів для електронної комерції.

Предмет дослідження – методи та засоби розробки дизайну, фронтенд-розробки та управління базами даних, зокрема технології Figma, Angular та MongoDB.

Короткий зміст роботи: у даній роботі досліджується та реалізується процес розробки дизайну та створення вебсайту для перегляду та покупки речей ручної роботи онлайн.

У ході роботи детально розглядаються методи створення користувацького інтерфейсу, функціоналу вебсайту та інтеграції з базою даних. Зокрема, досліджується використання сучасних технологій: Angular для динамічного оновлення контенту та управління станом додатку, а також MongoDB для зберігання інформації про товари та користувачів. Окрім цього, вивчаються підходи до дизайну інтерфейсу у Figma, що дозволяє створювати естетично привабливі та функціональні макети.

КЛЮЧОВІ СЛОВА: РОЗРОБКА ВЕБСАЙТУ, FIGMA, ANGULAR, MONGODB, TAILWINDCSS, ФРОНТЕНД, БЕКЕНД.

ABSTRACT

Text part of the qualification work for the bachelor's degree: 57 pages, 52 pictures, 11 sources.

Aim of the work – development of a functional website for viewing and purchasing handmade items online using Figma, Angular, and MongoDB technologies, as well as analyzing their effectiveness and interaction within the framework of web development.

Object of research – the process of developing websites for e-commerce.

Subject of research – methods and tools for designing, frontend development, and database management, specifically using Figma, Angular, and MongoDB technologies.

Summary of the work: this thesis explores and implements the process of designing and creating a website for viewing and purchasing handmade items online.

The work examines methods for creating user interfaces, website functionality, and database integration. Specifically, it investigates the use of modern technologies: Angular for dynamic content updates and application state management, and MongoDB for storing information about products and users. Additionally, it studies interface design approaches in Figma, which allow for the creation of aesthetically pleasing and functional layouts.

KEYWORDS: WEBSITE DEVELOPMENT, FIGMA, ANGULAR, MONGODB, FRONTEND, TAILWINDCSS, BACKEND.

ЗМІСТ

Стор.

ВСТУП

У сучасному світі електронної комерції важливо створювати вебсайти, що не лише відповідають естетичним та функціональним вимогам, але й забезпечують зручний і безпечний досвід для користувачів. Актуальним це питання стає для платформ, де користувачі можуть переглядати та купувати речі ручної роботи, які потребують особливої уваги до деталей та унікальності.

Ця кваліфікаційна робота присвячена розробці дизайну та реалізації вебсайту для перегляду та покупки речей ручної роботи онлайн. Основною метою роботи є створення інтуїтивно зрозумілого та естетично привабливого інтерфейсу користувача, а також надійної бази для функціонування вебсайту з використанням таких технологій, як “Figma”, “Angular” та “MongoDB”.

У першому розділі роботи розглядається проєктування дизайну та архітектури вебсайту. Цей етап включає аналіз вимог до дизайну та функціоналу, розробку інтерфейсу користувача в середовищі “Figma”, а також вибір архітектури вебсайту. Важливим аспектом є детальне опрацювання вимог, що дозволяє забезпечити високу якість кінцевого продукту.

Другий розділ присвячено безпосередній розробці вебсайту. Цей етап включає налаштування середовища розробки, верстку з використанням “Tailwind CSS”, а також реалізацію технічної частини вебсайту. Особлива увага приділяється забезпеченню продуктивності вебсайту, що є ключовим фактором для сучасних платформ.

Третій розділ роботи зосереджений на налаштуванні та управлінні базою даних. Розглядається імплементація бази даних у вебсайт та конфігурація “MongoDB” на сервері. Це дозволяє забезпечити надійне зберігання та швидкий доступ до даних користувачів і товарів.

Таким чином, ця кваліфікаційна робота надає комплексний підхід до розробки сучасного вебсайту для речей ручної роботи, поєднуючи креативні дизайнерські рішення та потужні технологічні інструменти для створення ефективної та зручної онлайн-платформи.

1 ПРОЄКТУВАННЯ ДИЗАЙНУ ТА АРХІТЕКТУРИ ВЕБСАЙТУ

1.1 Аналіз вимог до дизайну та функціоналу

Першим кроком у розробці вебсайту є визначення цільової аудиторії. Для платформи, де користувачі можуть переглядати та купувати речі ручної роботи, цільовою аудиторією є користувачі, які шукають унікальні речі ручної роботи. Їм потрібен інтуїтивно зрозумілий інтерфейс для пошуку, перегляду та покупки товарів. Також додавання можливості залишити свою ідею ручної роботи, яку потім роботодавці сайту зможуть розглянути та зробити особисто на замовлення, а в майбутньому основним товаром вебсайту, дасть можливість огортати більшу цільову аудиторію.

Другим кроком – є визначення функціональних вимог вебсайту, що дозволить окреслити основні можливості платформи. Основний функціонал складається з:

1. Реєстрації та авторизації користувача.
2. Переклад сайту на дві мови (українська та англійська).
3. Додавання, видалення та перегляд товарів у кошику.
4. Каруселі.
5. Переміщення різніми сторінкам платформи.
6. Сортування товарів за різними критеріями (ціною, рейтингом та матеріалом).
7. Можливості залишити своє замовлення (фото, назви, опису та матеріалу).
8. Перегляду сторінки товару (фото, ціни, рейтингу, матеріалу, назви, опису та коментарів).
9. Можливості залишити коментар (оцінку та відгук).
10. Підвантаження товарів та користувачів з бази даних.

Третій крок – це врахування дизайнерських вимог. Цей крок найважливіший для приваблення користувачів до своєї платформи. Не зважаючи який функціонал буде доданий, якщо вебсайт не буде інтуїтивно зрозумілим та візуально

привабливим, клієнти не будуть їм користуватися через поганий користувацький досвід, і оберуть платформи-конкурентів.

1.2 Розробка дизайну в середовищі “Figma"

Перший етап розробки дизайну – це визначення кількості сторінок та їх наповнення:

1. Головна – сторінка з асортиментом вебсайту.

2. Сторінка товару – місце розташування усієї інформації про товар, яку неможливо відобразити на головній сторінці.

3. Детальніше – сторінка меню з можливістю переходу на інші сторінки категорії “Детальніше”.

4. Відділення – інформаційна сторінка про місце розташування можливих відділів.

5. Контакти – сторінка з контактною інформацією.

6. Доставка – інформаційна сторінка про можливі сервіси доставки.

7. Гарантія – інформаційна сторінка щодо повернення та гарантійного обслуговування товару.

8. Про нас – інформаційна сторінка про компанію, яка відповідає за товари на платформі (в нашому випадку – титульна сторінка кваліфікаційної роботи).

Наступний крок – це створення детального візуального дизайну, який складається зі шрифтів (рис. 1.1) та кольорової схеми:

1. #F0E2CF (фон)

2. #FAD67F (основний)

3. #666561 (основний технічний)

4. #F3C585 (вторинний)

5. #FFE9B1 (вторинний при наведенні миші)

1. #504535 (вторинний технічний)

1. #504535 (вторинний технічний при наведенні миші)

1. #504535 (вторинний технічний при натисканні кнопки миші)

7. #191610 (темний)

Рис. 1.1 Шрифти

Тепер, використовуючи створені приклади, настає етап створення компонентів-констант кожної сторінки, якими є колонтитули, меню та карусель. На верхньому колонтитулі має бути лого з назвою платформи, конфігурації вебсайту (переклад) та користувацькі елементи (кошик та вхід/реєстрація акаунту). У нижньому колонтитулі – копірайт. Для каруселі потрібен шаблон з розмірами та декілька зображень з варіантами на кожну мову (рис. 1.2).

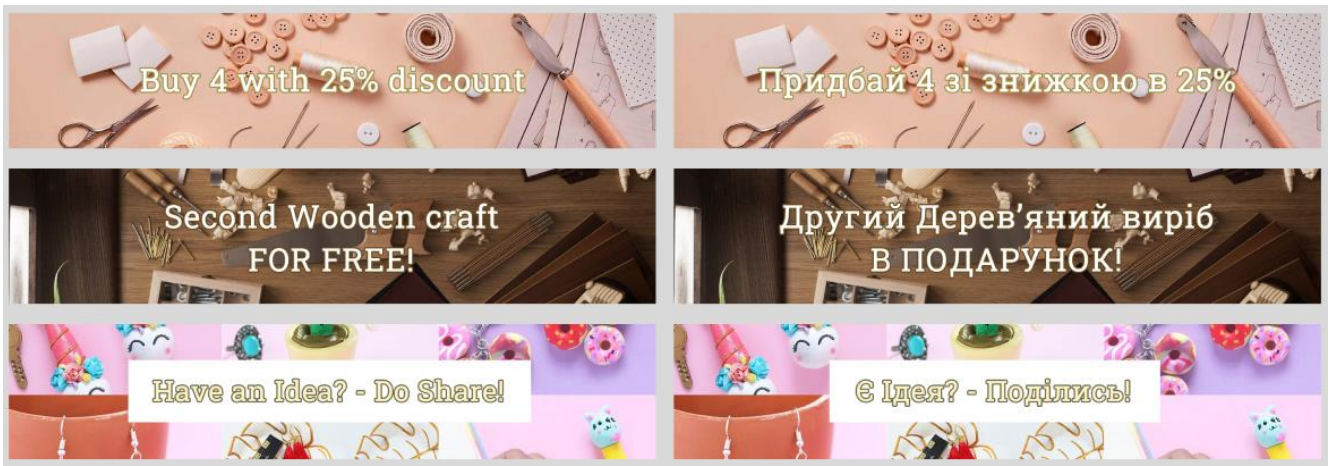


Рис. 1.2 Обрані зображення для каруселі в обох варіантах

Останній компонент-константа – меню, на якому мають бути кнопки для переміщення між сторінками. Для естетичного вигляду створюються кнопки для обирання матеріалу, які також переміщають на сторінку, але одну й ту саму (головну сторінку).

Після створення кожного компонента-константи утворюється макет для кожної сторінки, але щоб платформа була адаптивна для всіх екранів стаціонарних комп’ютерів та більше, треба створити контейнер, в якому будуть знаходитись усі компоненти, та обмежити його ширину до 1440 пікселів. Таким чином вийде дизайн сторінки без наповнення, котре залежить від сторінки (рис. 1.3).

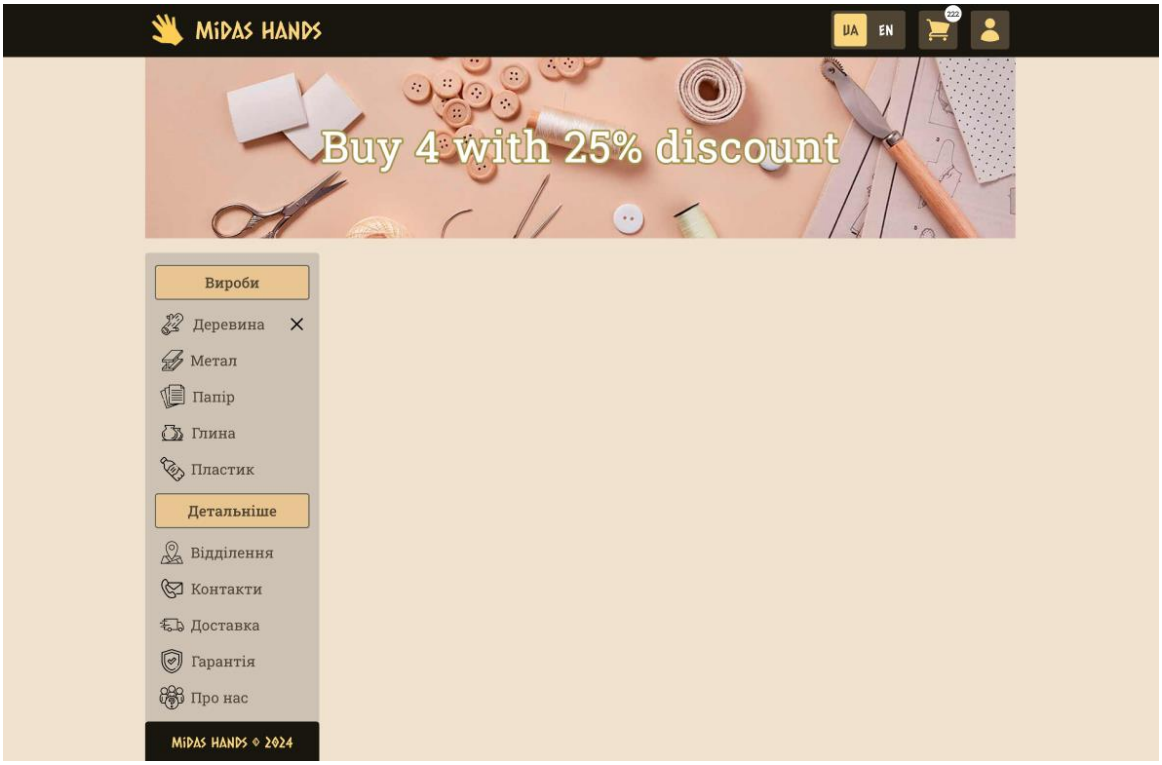


Рис. 1.3 Дизайн статичної частини вебсайту

Нарешті після отримання статичної частини вебсайту, а також розуміння загального дизайну, є можливим створення наповнення для кожної сторінки:

1. Головна сторінка - це сторінка з товарами, тож потрібно створити макет для них. Він складається з зображення, кнопки для додавання та видалення товару з кошика, назви товару, ціни та рейтингу. Для кращої оптимізації додається обмеження товарів на сторінці, а також кнопка “Показати Ще”. Залишається

сортування за ціною з рейтингом і кнопка для замовлення, після чого головна сторінка завершена (рис. 1.4).

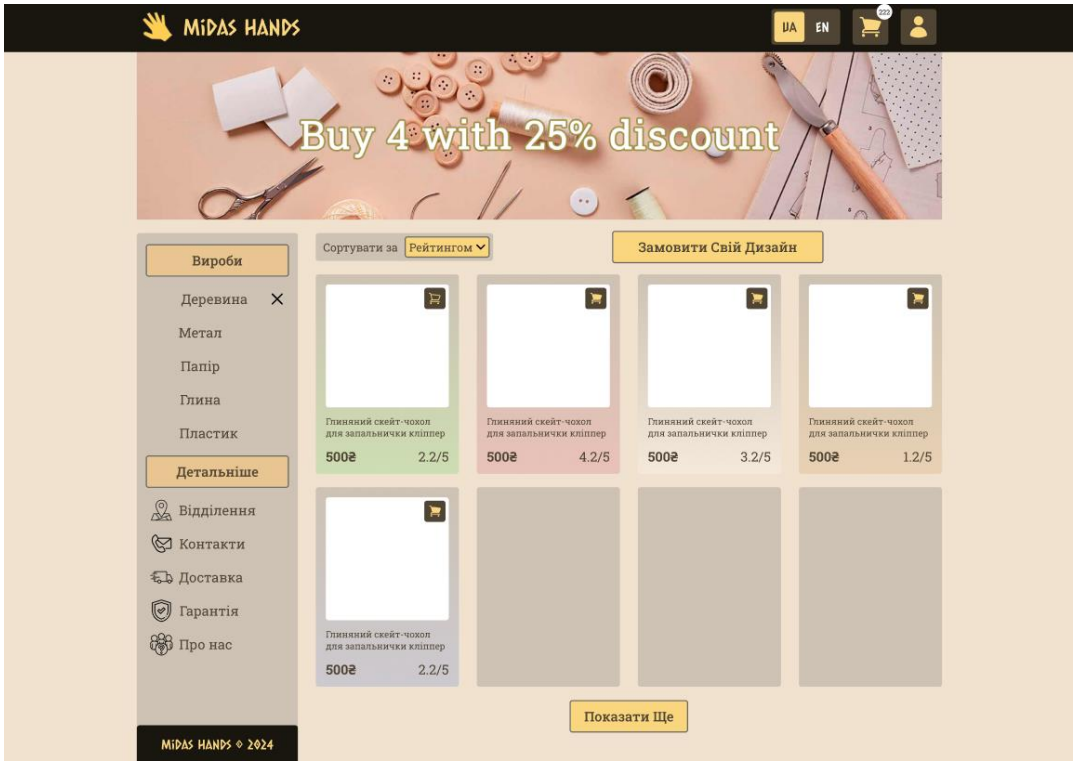


Рис 1.4 Готовий дизайн головної сторінки

2. Сторінка товару має мати усю інформацію товару з головної сторінки, а також опис та коментарі. Ще, для покращення користувацького досвіду, треба додати кнопку повернення назад і заголовок “Сторінка Товару”, після чого сторінка готова (рис. 1.5).

3. Сторінка “Детальніше” – це меню, для переходу на інші інформаційні сторінки, тож на ній мають бути кнопки для кожної з них. Для покращення користувацького досвіду треба додати зображення та текст відповідної сторінки. Також, оскільки замовлення є важливою частиною платформи, обов’язкове додання кнопки для замовлення з головного меню, але для естетичного вигляду, вона має розміри інших кнопок. Після враховування цих критеріїв, сторінка з інформаційним меню готова (рис. 1.6).

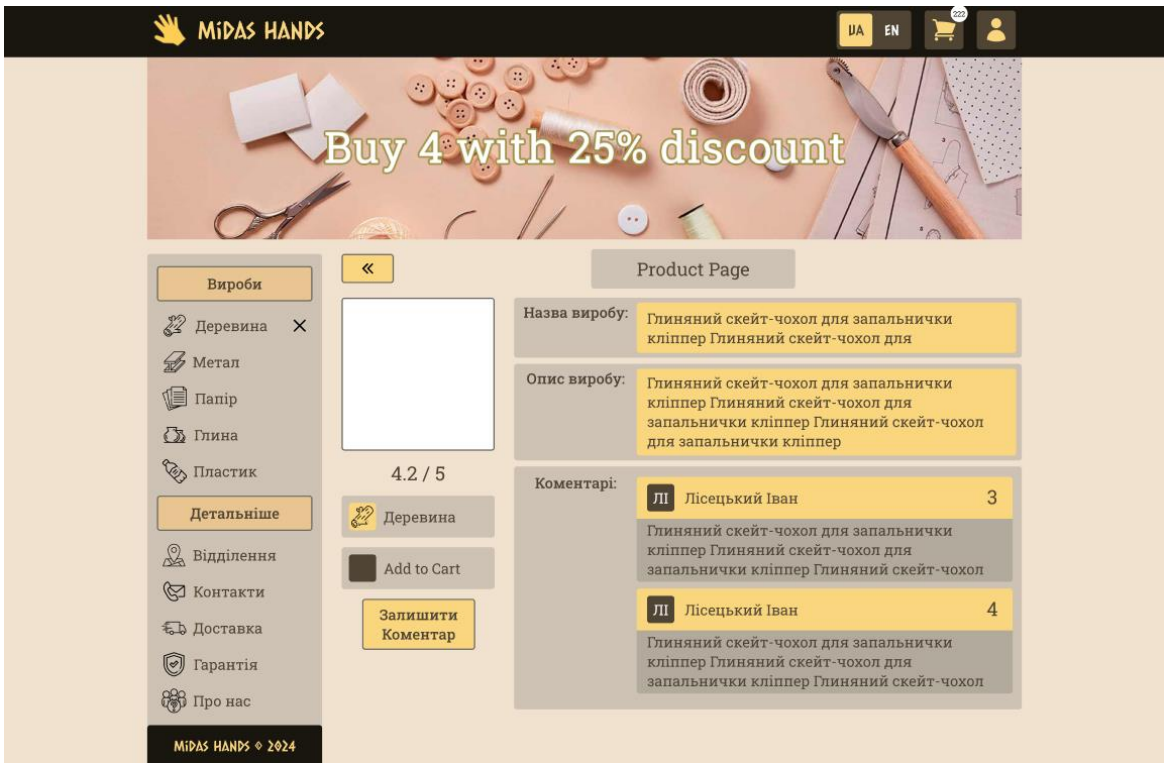


Рис 1.5 Готовий дизайн сторінки товару

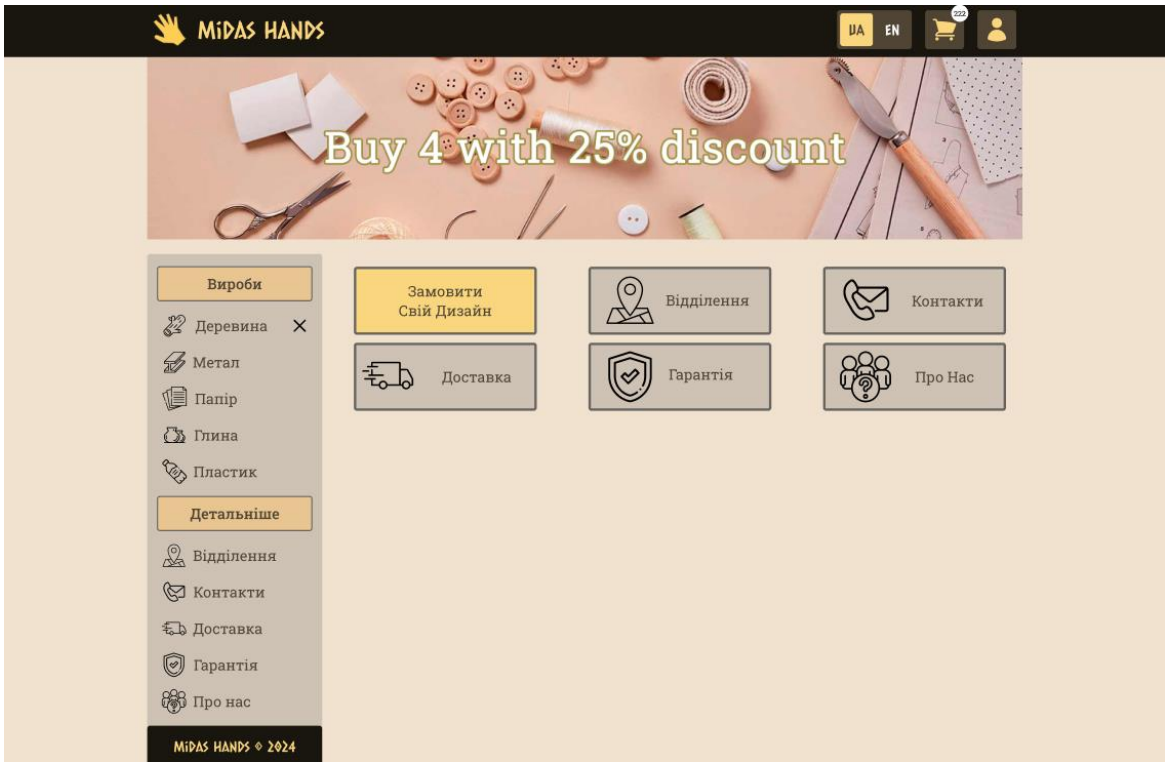


Рис 1.6 Готовий дизайн сторінки “Детальніше”

4. Сторінка з відділеннями повинна також, як і сторінка товару, мати кнопку повернення назад, а також заголовок для кращого користувацького досвіду. Окрім

цього ця сторінка – інформаційна, тож створення дизайну для цієї інформації – обов'язкове. Розділення кожного відділення зробить дизайн естетичним, а додання зображення з “Google maps”, назви закладу, адреси та можливого номеру телефону, зробить дизайн інтуїтивно зрозумілим. Останнім додаємо на текст посилання до відповідної адреси на “Google maps”, таким чином сторінка з відділеннями готова (рис 1.7).

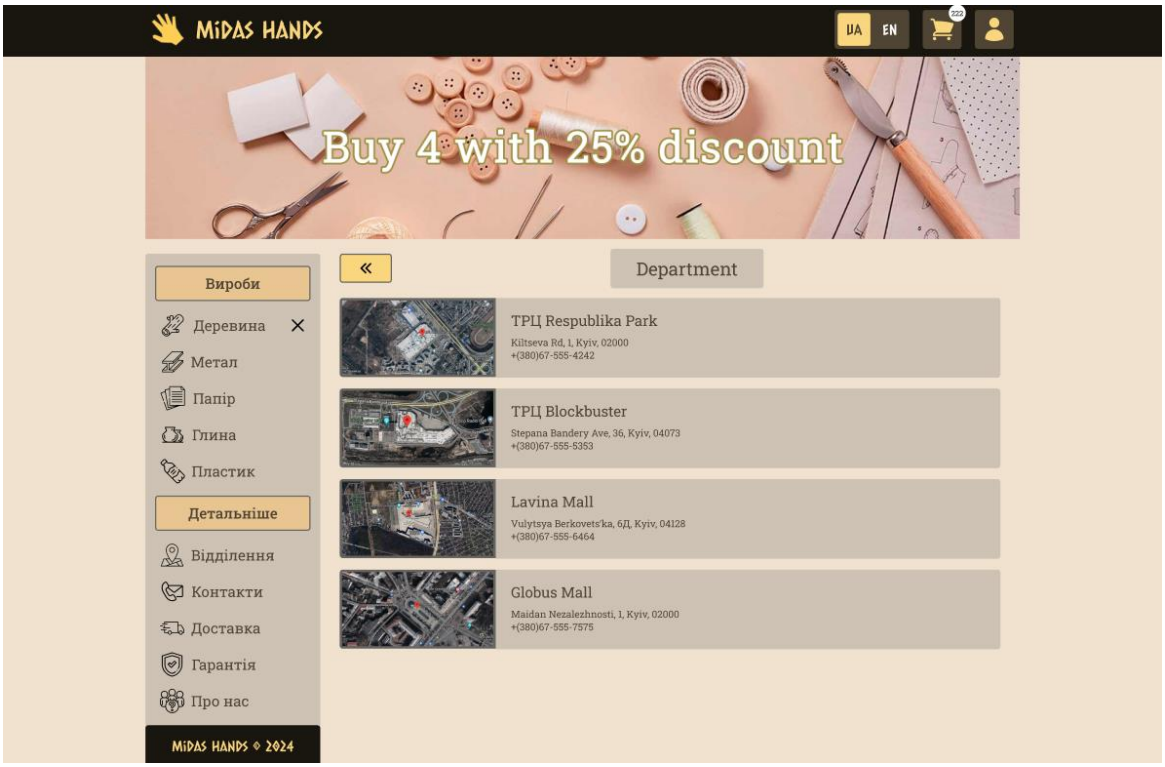


Рис 1.7 Готовий дизайн сторінки з відділеннями

5. Сторінка з контактами створюється за схожою схемою створення сторінки з відділеннями: додається кнопка повернення назад, а також заголовок. Окрім цього, ця сторінка також інформаційна, але має соціальні мережі замість відділень. Для інтуїтивного дизайну кожна мережа повинна мати лого цієї соціальної мережі, її назву та, при наведенні миші, показувати, що на неї можна натиснути. При необхідності також додається текст з інформацією щодо прийнятного часу відповіді. Таким чином сторінка з контактами завершена (рис. 1.8).

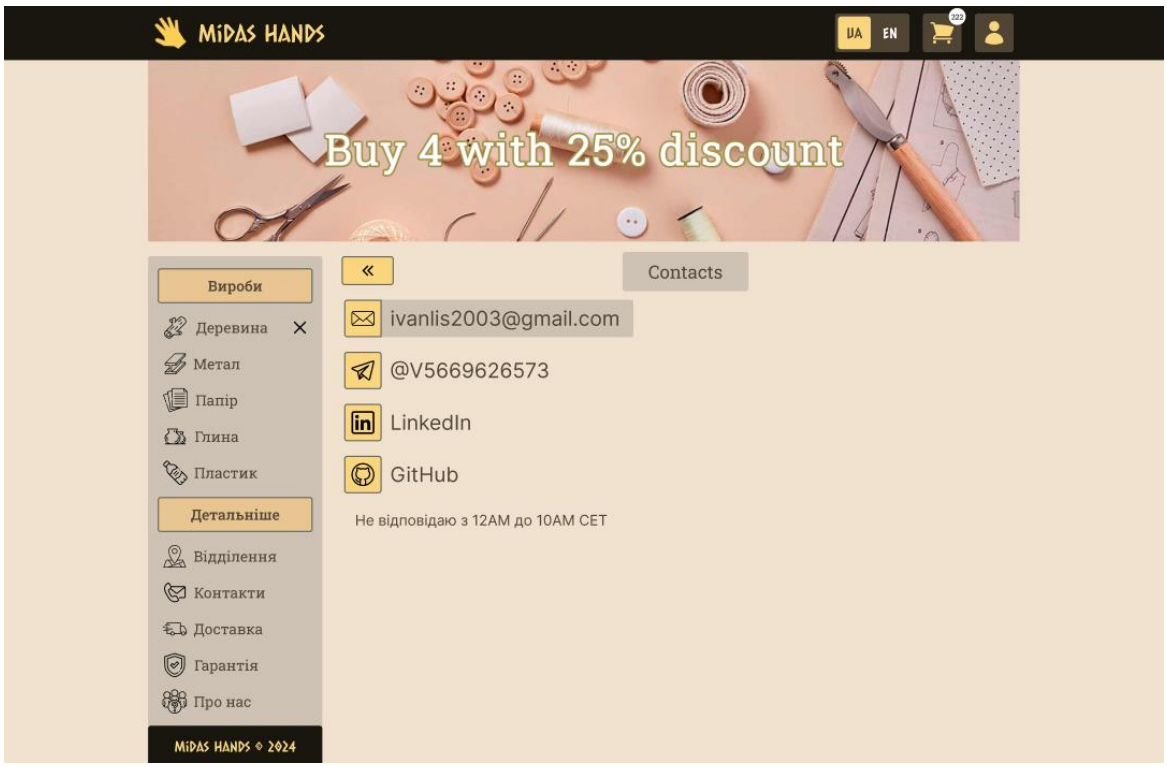


Рис 1.8 Готовий дизайн сторінки з контактами

6. Сторінка з доставкою – це носій можливих сервісів доставки. Окрім кнопки повернення назад і заголовку, створюються кнопки для кожного можливого сервісу. Для інтуїтивного дизайну вони складаються з лого сервісу та його назви, а для покращення користувацького досвіду, вони мають гіперлінк на сайт відповідного сервісу доставки. Врахувавши ці критерії – дизайн готовий (рис. 1.9).

7. Інформаційна сторінка щодо гарантії має такі ж самі кнопку повернення назад та заголовок, як і усі інші інформаційні сторінки. Також ця сторінка, хоча й повністю залежить від бажань замовника, має складатися лише з тексту. Не зважаючи на це, її також можна покращити. Оскільки при потребі в гарантійних послугах клієнт має писати у службу підтримки (контакти), створення кнопки для переходу на сторінку з контактами покращить користувацький досвід та зробить цей процес інтуїтивно зрозумілим. Після об'єднання усіх критеріїв – дизайн сторінки про гарантію готовий (рис. 1.10).

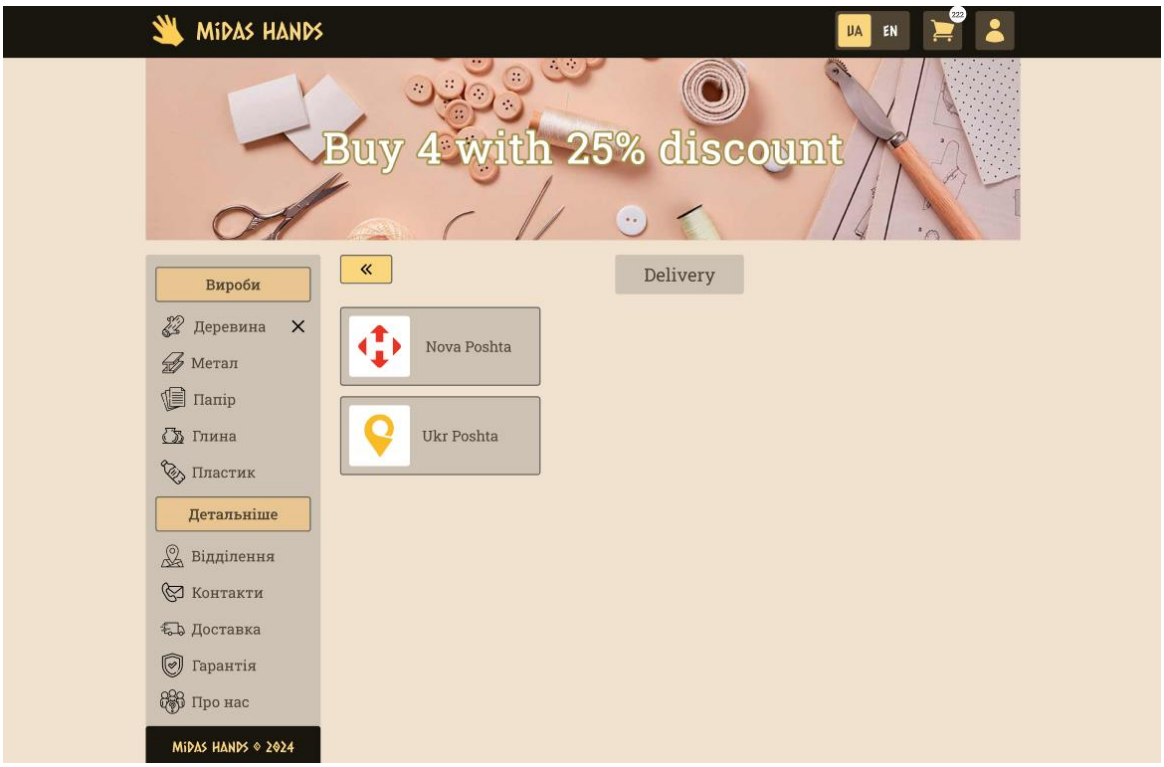


Рис 1.9 Готовий дизайн сторінки з сервісами доставки

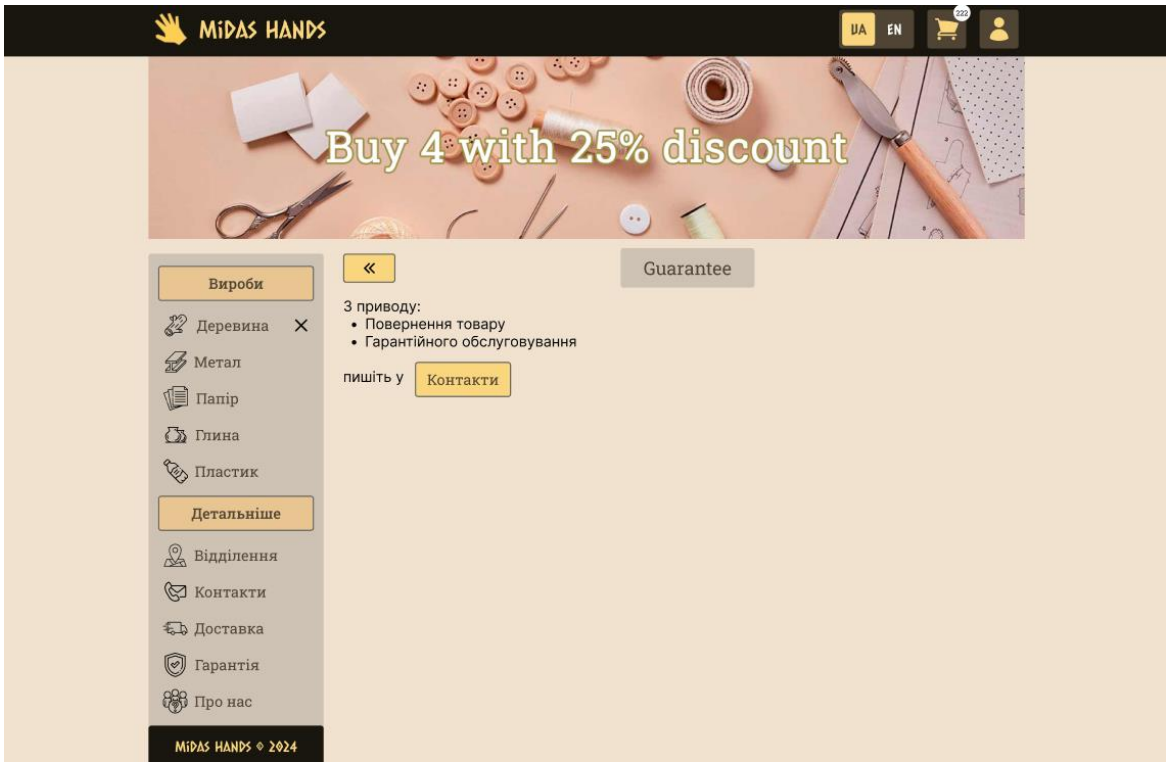


Рис 1.10 Готовий дизайн сторінки про гарантію

8. Остання сторінка – “Про нас”. Нічного незвичайного для її створення не потрібно, така ж сама кнопка повернення назад та заголовок, а також текст зі

зображеннями. Оскільки ця сторінка так само повністю залежить від бажань замовника, її наповнення має лише попадати під тему сторінки. У випадку кваліфікаційного проєкту, її наповнення – титульна сторінка (рис. 1.11).

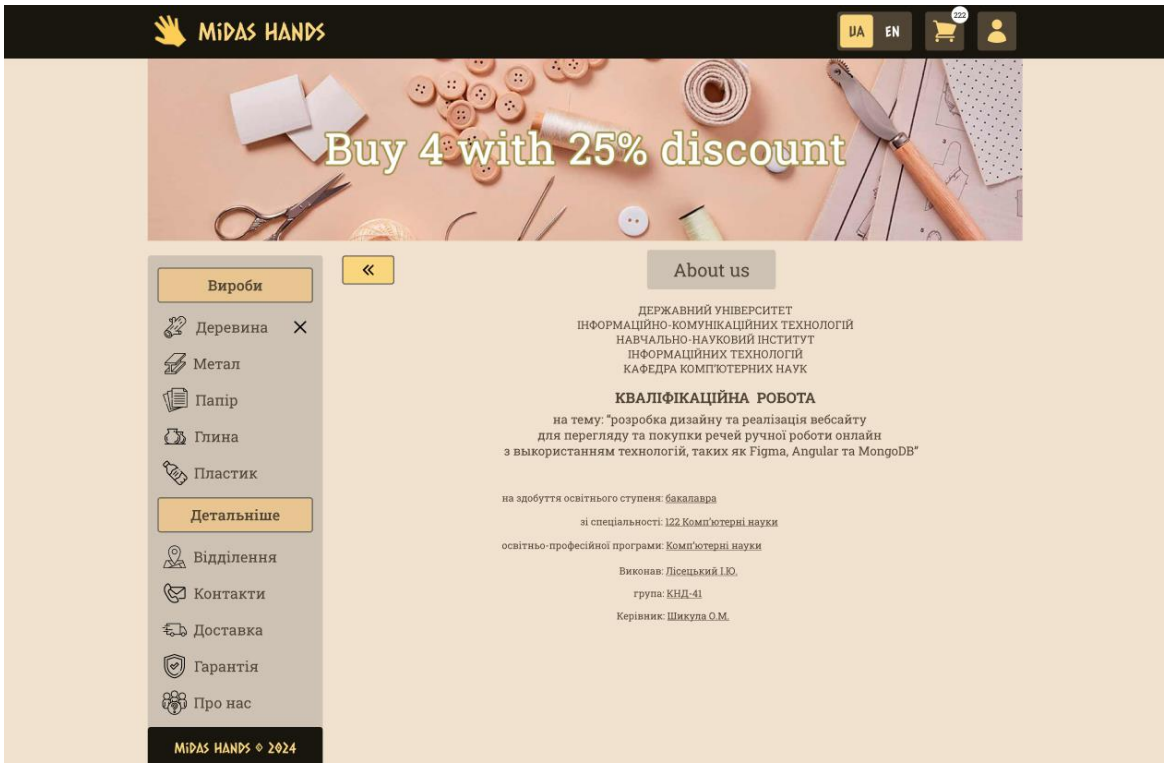


Рис 1.11 Готовий дизайн сторінки “Про нас”

Нарешті, коли усі сторінки зроблені, залишаються лише 4 компоненти:

1. Компонент авторизації – це компонент з можливістю вибору між реєстрацією та авторизацією (кількості полів введення інформації). Для авторизації потрібні поля для вводу електронної пошти та паролю, а для реєстрації додаються поля для вводу імені та прізвища. Обов’язковою частиною компонента є кнопка для відправлення введеної інформації, а також для перемикання між авторизацією та реєстрацією. Склавши усі критерії разом – компонент авторизації готовий (рис. 1.12).

2. Компонент кошика зобов’язаний зображати додані до нього товари. Також він обов’язково повинен мати кнопку для їх придбання. Для покращення користувацького досвіду від кошика, треба додати заголовок компонента, повну ціну та кнопку для його закриття, а для макета товару треба додати кнопку

видалення з кошика, та вибір кількості товарів. Таким чином кошик буде готовий (рис. 1.13).

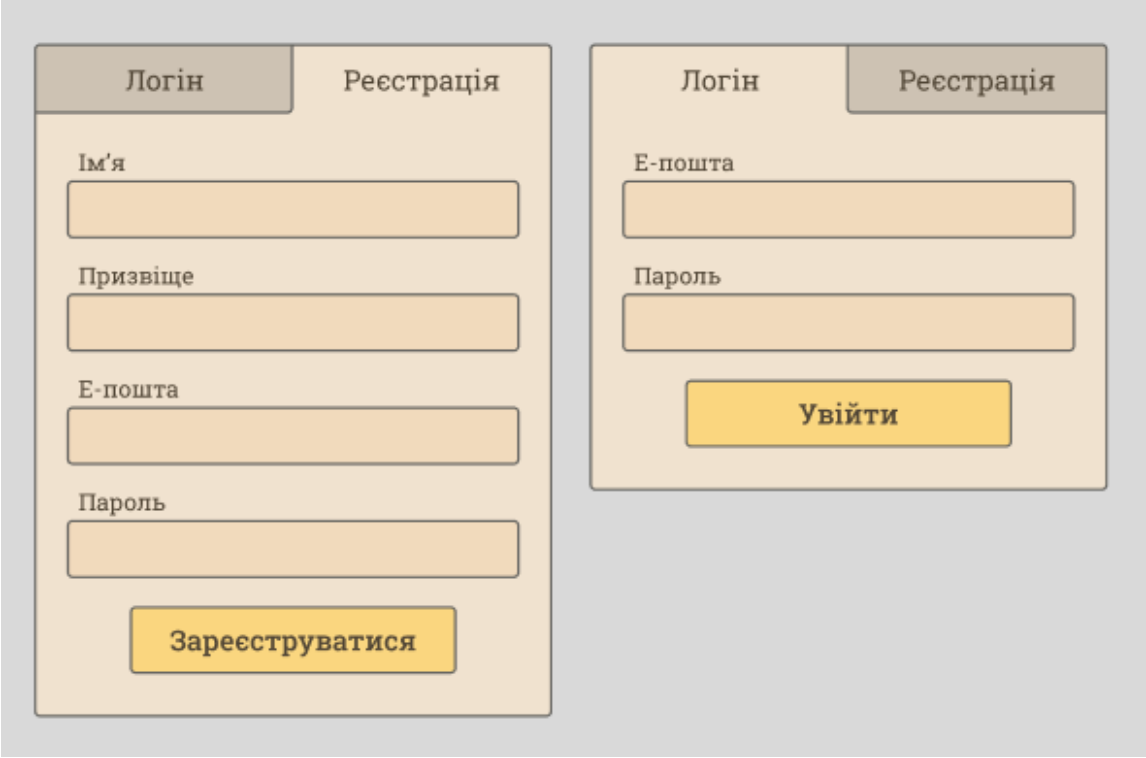


Рис 1.12 Готовий дизайн компонента авторизації в обох режимах

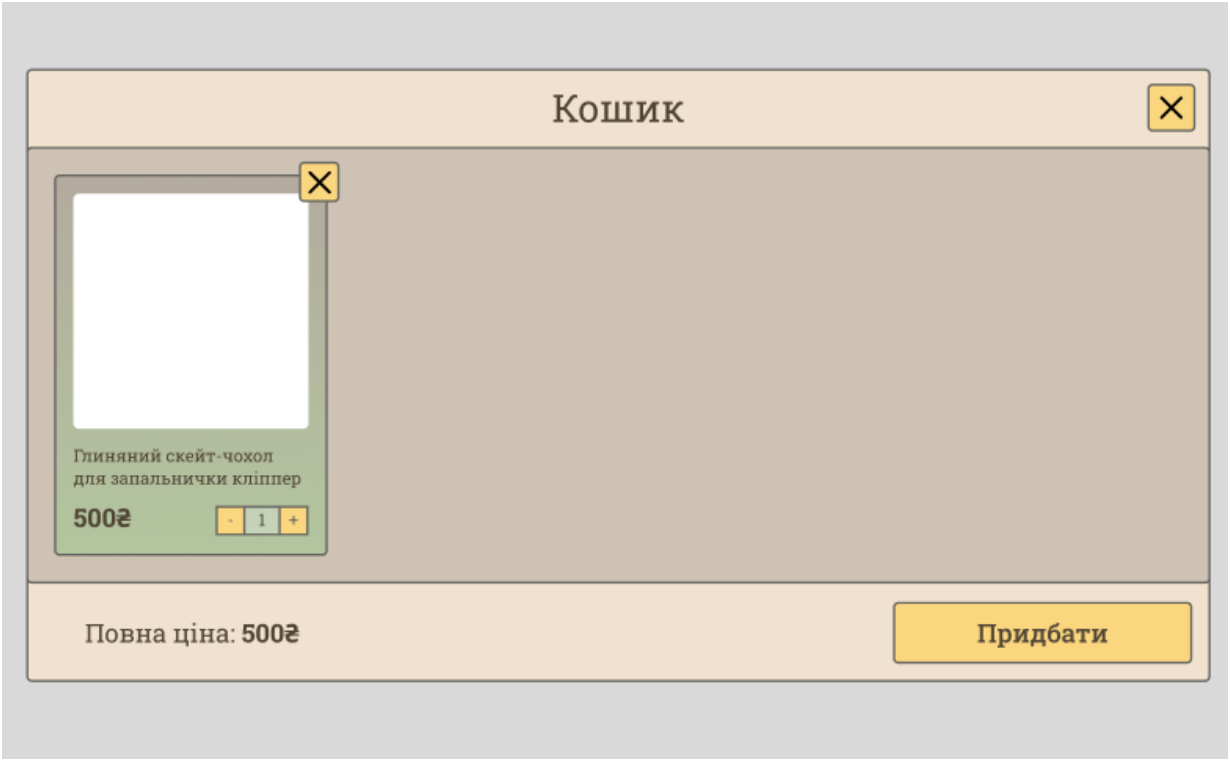


Рис 1.13 Готовий дизайн компонента кошика

3. Компонент замовлення – це важливий компонент цієї платформи, мета якого отримати ідею бажаного клієнтом товару ручної роботи, якого не існує на вебсайті. Таким чином клієнти матимуть можливість не тільки купити ще неіснуючий товар, а й додати його до списку у майбутньому. Вважаючи ці фактори, обов'язково треба додати кнопку для відправки замовлення, можливість завантажити зображення, можливість обрати матеріал, а також додати поля для назви, опису товару та поле електронної пошти, для відправки замовлення і потенціального отримання відповіді щодо нього. Для покращення інтуїтивності, треба додати на кнопки вибору матеріалу відповідні зображення, а для покращення користувацького досвіду – заголовок компонента, кнопку його закриття та прибрати поле з електронною поштою, якщо користувач авторизований. Врахувавши увесь потрібний функціонал та додавши вже перевірений на інших компонентах дизайн – компонент замовлення буде завершено (рис. 1.14).

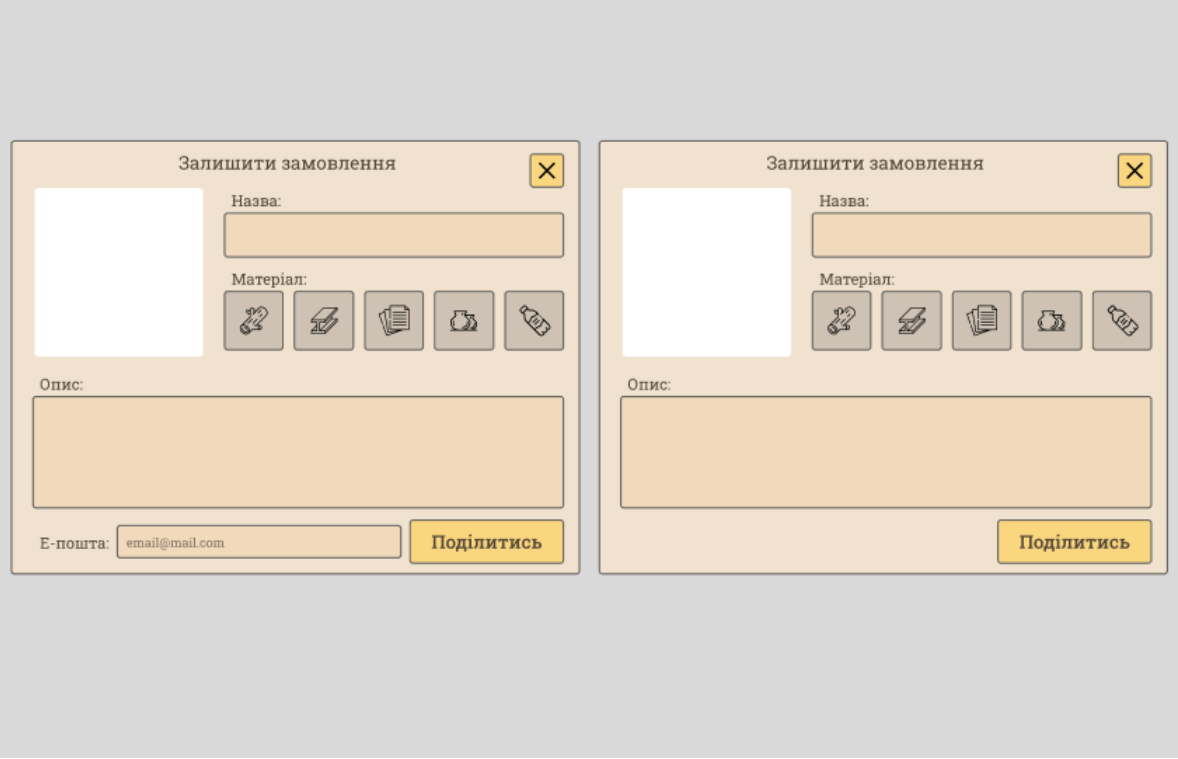


Рис 1.14 Готовий дизайн компонента замовлення в обох режимах

4. Компонент коментаря – це єдина можливість впливу на рейтинг товару. Оскільки коментар можна залишити лише бувши авторизованим, поля імені та прізвища створювати не потрібно. Обов'язковою частиною цього компоненту є

кнопка для збереження коментаря, можливість залишити відгук та оцінку (від одного до п’яти). Для покращення користувацького досвіду треба також додати кнопку закриття компонента. На цьому дизайн компонента завершений (рис. 1.15).

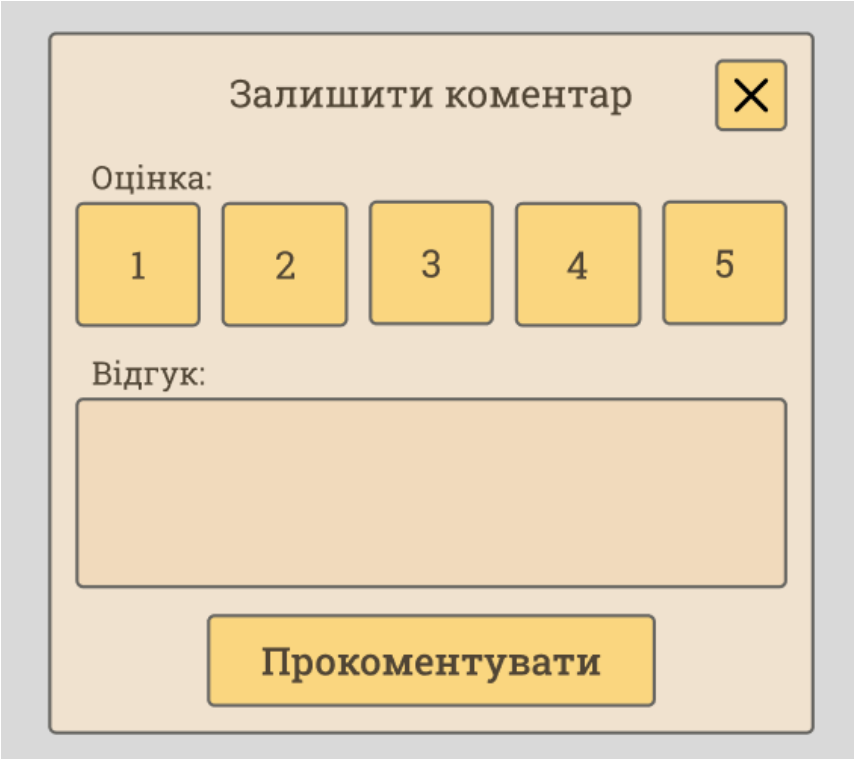


Рис 1.15 Готовий дизайн компонента коментаря

1.3 Вибір архітектури вебсайту

Вибір архітектури вебсайту – це головний етап до початку верстки, який визначає структуру та технології, які будуть використовуватися для розробки та підтримки проєкту. Для його реалізації було обрано “Angular”, “Tailwind CSS” та “TypeScript”.

“Angular” є потужним фреймворком для розробки вебдодатків. Він використовує компонентну архітектуру, що дозволяє розділяти додаток на незалежні частини. Це сприяє зручній розробці, тестуванню та повторному використанню коду. “Angular” забезпечує модульність, дозволяючи організовувати код у логічні блоки, що полегшує управління проєктом. Використання декларативного шаблонного синтаксису робить створення користувацького інтерфейсу зрозумілим та підтримуваним.

“Tailwind CSS”, як утилітарний фреймворк для створення стилів, дозволяє швидко та ефективно розробляти інтерфейси. Завдяки утилітарним класам, розробники можуть створювати стилі одразу у “HTML” без написання “CSS” коду, що зменшує час розробки. “Tailwind CSS” також забезпечує конфігурованість, що дозволяє налаштовувати теми та шрифти відповідно до потреб проєкту. “Tailwind CSS”, в комбінації з “Angular”, дає великий простір для створення анімацій та технічних реалізацій стилів завдяки директиві “ngClass”.

“TypeScript” є основною мовою фреймворку “Angular”. Він додає статичну типізацію до “JavaScript”, що підвищує якість та надійність коду. Використання “TypeScript” дозволяє виявляти помилки на етапі компіляції, що зменшує кількість багів у коді. Також він підтримує сучасні можливості “JavaScript”, забезпечуючи зручний та продуктивний процес розробки не забираючи наявний функціонал.

Комбінація цих технологій є найефективнішою, чим забезпечує створення масштабованого та продуктивного вебсайту. “Angular” надає потужний каркас для верстки, “Tailwind CSS” дозволяє швидко створювати розроблені інтерфейси, а “TypeScript” забезпечує надійність та підтримre коду.

2 РОЗРОБКА ТА ІМПЛЕМЕНТАЦІЯ ВЕБСАЙТУ

2.1 Налаштування середовища розробки

Налаштування середовища розробки є критичним етапом у розробці вебсайту, оскільки воно забезпечує стабільну та ефективну платформу для кодування та тестування. Для проєкту, що використовує “Angular”, “Tailwind CSS” та “MongoDB”, цей процес включає кілька основних кроків.

Для початку треба завантажити з офіційного сайту “NodeJS” — програмну платформу, засновану на движку “V8” від “Google”, що дозволяє запускати код типу “JavaScript” поза браузером. Вона розроблена для створення високопродуктивних серверних застосунків, використовуючи асинхронну, неблокуючу модель вводу-виводу. Це дозволяє обробляти велику кількість одночасних з'єднань з високою пропускною здатністю. “Node.js” відомий своєю здатністю працювати з однопотоковою архітектурою, використовуючи події для управління асинхронними операціями, такими як зчитування файлів, запити до баз даних чи мережеві з'єднання. Це робить його особливо придатним для розробки реальних застосунків, чатів, інструментів командного рядка, “API” та інших серверних програм. Він має велику та активну спільноту, яка постійно розширює екосистему модулів і пакетів, доступних через менеджер пакетів “npm”.

Далі у командному рядку:

1. node -v – перевірка встановлення та версії “NodeJS”.

2. npm -v – перевірка встановлення та версії ”Node Package Manager”.

3. npm install -g @angular/cli – встановлення “Angular CLI” з глобальним флагом.

4. ng new bachelor-diploma – створення нового “Angular” проєкту.

5. cd bachelor-diploma – перехід у створену директорію проєкту.

6. npm install tailwindcss postcss autoprefixer – встановлення “Tailwind CSS” та інших зв’язаних з ним пакетів.

7. npx tailwindcss init – ініціалізація файлів конфігурації “Tailwind CSS”.

8. cd src/app – перехід у директорію “app”.

9. ng generate component components/about – створення директорії для компонентів та генерація “Детальніше” компонента у ній.

10. cd components – перехід у директорію “components”.

11. ng generate component about-contacts – компонент “Контакти” згенерований.

12. ng generate component about-delivery – компонент “Доставка” згенерований.

13. ng generate component about-department – компонент “Відділення” згенерований.

14. ng generate component about-guarantee – компонент “Гарантію” згенерований.

15. ng generate component about-us – компонент “Про нас” згенерований.

16. ng generate component carousel – компонент “Карусель” згенерований.

17. ng generate component cart – компонент “Кошик” згенерований.

18. ng generate component comment – компонент “Коментар” згенерований.

19. ng generate component header – компонент “Верхній колонтитул” згенерований.

20. ng generate component login – компонент “Авторизація” згенерований.

21. ng generate component menu – компонент “Меню” згенерований.

22. ng generate component order – компонент “Замовлення” згенерований.

23. ng generate component shop – компонент “Магазин” згенерований.

24. ng generate component shop-item – компонент “Товар” згенерований.

25. ng generate component shop-item-page – компонент “Сторінка товару” згенерований.

26. ng generate component blank – компонент “Порожній” згенерований.

27. cd ../ - перехід у директорію “app”.

28. ng generate service services/cart – створення директорії для сервісів та генерація “Кошик” сервісу у ній.

29. cd services – перехід у директорію “services”.

30. ng generate service item-page – сервіс “Сторінка товару” згенерований.
31. ng generate service language – сервіс “Мова” згенерований.
32. ng generate service login – сервіс “Авторизація” згенерований.
33. ng generate service order – сервіс “Замовлення” згенерований.
34. ng generate service product – сервіс “Товар” згенерований.
35. ng generate service shop – сервіс “Магазин” згенерований.

На цьому етапі проєкт майже повністю налаштований. Для початку треба закінчити налаштування “Tailwind CSS”: надати доступ до стилів у файлі “angular.json” (рис. 2.1) та оновити файл “src/styles.css” основою, компонентами та утилітами (рис. 2.2).

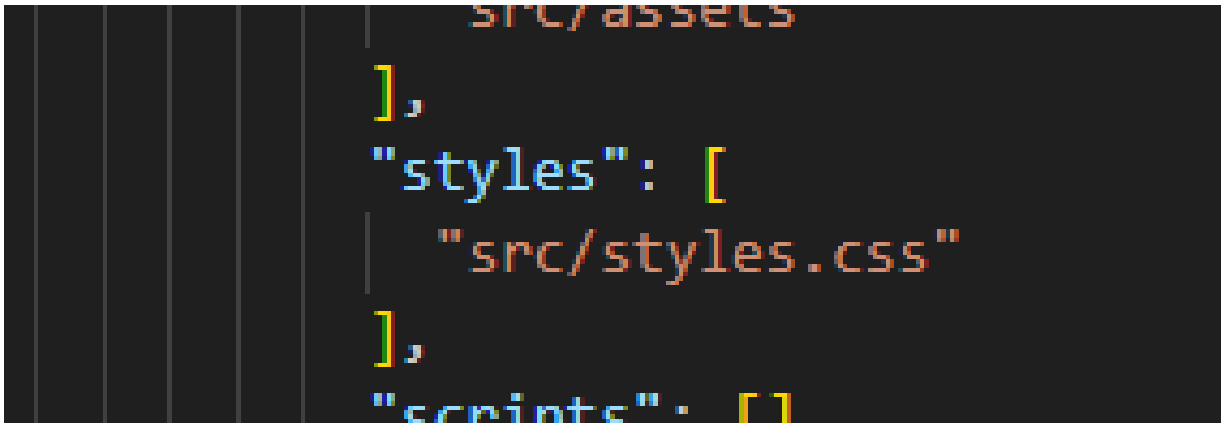


Рис. 2.1 Файл “angular.json” з наданим доступом

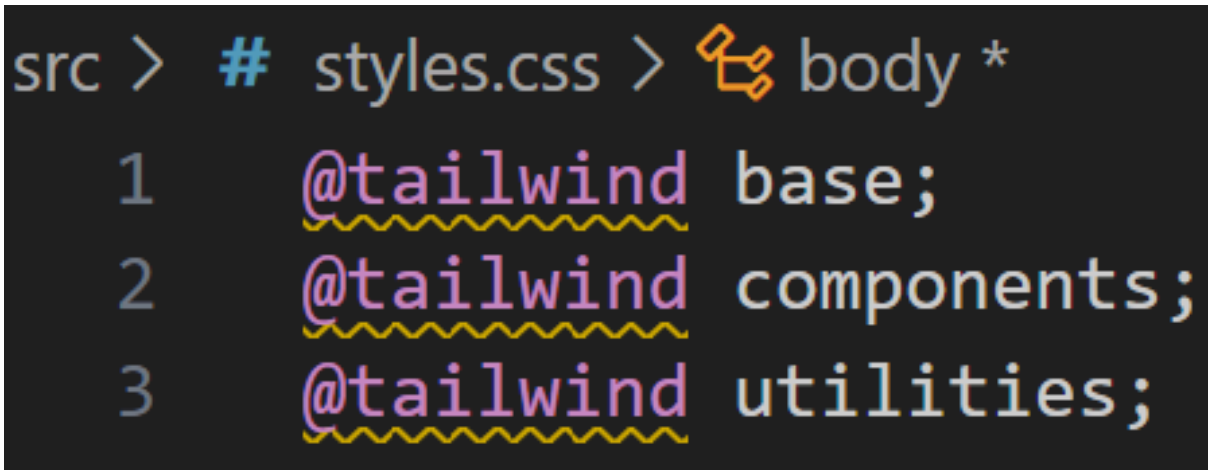


Рис. 2.2 Оновлений вигляд файлу “src/styles.css”

І нарешті залишилось лише створити моделі для товарів та користувачів, як приклад елементів, які будуть отримані та надіслані на базу даних. Для того, щоб

проєкт був структурований, створюється окрема директорія у “src/app” під назвою “models”. У ній необхідно створити два файли “product.ts” та “user.ts”.

2.2 Верстка з використанням “Tailwind CSS”

Оскільки до цього у Figma був розроблений дизайн, етап верстки спростовується до декількох етапів, які є копіюванням вже спланованого проєкту.

В першу чергу потрібно підготуватися до верстки: перенести шрифти, зображення та колірну палітру. Хоча проєкт охоплює верстку з використанням “Tailwind CSS”, користуватися глобальним файлом типу “CSS” (src/styles.css) є гарною практикою для імплементування шрифтів та зображень. Для кожного виду шрифтів зі схеми створюються стилі до відповідного текстового елементу (рис. 2.3).

```
h1 {
  font-family: "Caesar Dressing", system-ui;
  font-size: 40px;
  color: #FAD67F;
}
h2 {
  font-family: "Caesar Dressing", system-ui;
  font-size: 24px;
  color: #FAD67F;
}
h3 {
  font-size: 24px;
}
h4 {
  font-size: 24px;
  font-weight: bold;
}
h5 {
  font-size: 20px;
}
h6 {
  font-size: 16px;
}
button {
  font-size: 22px;
  font-weight: 500;
  user-select: none;
}
```

Рис. 2.3 Стилі текстових елементів

Для використання зображень треба їх зберегти, додати до директорії проєкту, а потім створити клас для кожного зображення. Щоб зберегти, потрібно в “Figma” обирати зображення, натиснути на “Перегляд”, а далі змінити формат на “SVG” (Scalable Vector Graphics), бо вага файлів є головною причиною довгого

завантаження вебсайту, а цей формат має менші розміри в порівнянні з растровими зображеннями. Після цього залишається лише натиснути “Експортувати” (рис. 2.4), а потім додати завантажені зображення в директорію “src/assets”. Далі треба створити свій клас для кожного з них (рис. 2.5).

Рис. 2.4 Експортування зображення у “SVG” форматі

```
.logo {background-image: url(assets/logo.svg)}
.full-cart {background-image: url(assets/full-cart.svg)}
.empty-cart {background-image: url(assets/empty-cart.svg)}
.user {background-image: url(assets/user.svg)}
.cross {background-image: url(assets/cross.svg)}
.arrow {background-image: url(assets/arrow.svg)}
.back-arrow {background-image: url(assets/back-arrow.svg)}
.carousel-0-ua {background-image: url(assets/carousel/carousel-0-ua.svg)}
.carousel-0-en {background-image: url(assets/carousel/carousel-0-en.svg)}
.carousel-1-ua {background-image: url(assets/carousel/carousel-1-ua.svg)}
.carousel-1-en {background-image: url(assets/carousel/carousel-1-en.svg)}
.carousel-2-ua {background-image: url(assets/carousel/carousel-2-ua.svg)}
.carousel-2-en {background-image: url(assets/carousel/carousel-2-en.svg)}
.email {background-image: url(assets/contacts/email.svg)}
.telegram {background-image: url(assets/contacts/telegram.svg)}
.linkedin {background-image: url(assets/contacts/linkedin.svg)}
.github {background-image: url(assets/contacts/github.svg)}
.craft-0 {background-image: url(assets/menu/wood.svg)}
.craft-1 {background-image: url(assets/menu/metal.svg)}
.craft-2 {background-image: url(assets/menu/paper.svg)}
.craft-3 {background-image: url(assets/menu/clay.svg)}
.craft-4 {background-image: url(assets/menu/plastic.svg)}
.detail-0 {background-image: url(assets/menu/department.svg)}
.detail-1 {background-image: url(assets/menu/contact.svg)}
.detail-2 {background-image: url(assets/menu/delivery.svg)}
.detail-3 {background-image: url(assets/menu/guarantee.svg)}
.detail-4 {background-image: url(assets/menu/about-us.svg)}
.delivery-0 {background-image: url(assets/delivery/delivery-0.svg)}
.delivery-1 {background-image: url(assets/delivery/delivery-1.svg)}
.department-0 {background-image: url(assets/department/department-0.svg)}
.department-1 {background-image: url(assets/department/department-1.svg)}
.department-2 {background-image: url(assets/department/department-2.svg)}
.department-3 {background-image: url(assets/department/department-3.svg)}
```

Рис. 2.5 Класи для кожного зображення

Нарешті для колірної палітри треба в файлі “tailwind.config.ts” додати, до об’єкта з розширеннями, ключ “colors”, який також є об’єктом, в середині якого ключем є назва кольору, а полем – значення кольору (рис. 2.6).

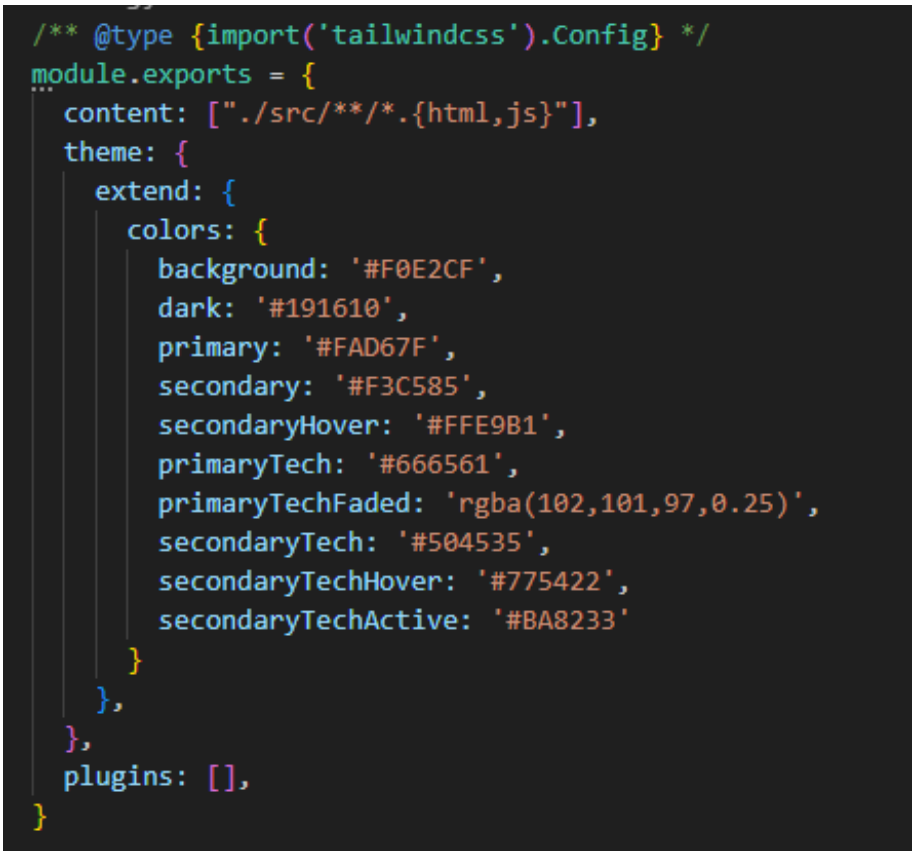


Рис. 2.6 Колірна палітра у файлі “tailwind.config.ts”

Останній етап підготовки – це встановлення на кожний елемент у тілі сайту флекс-бокс та тривалість переходу (рис. 2.7).

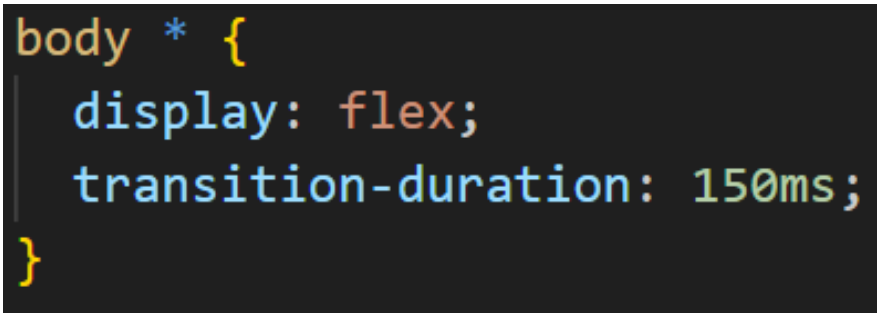


Рис. 2.7 Встановлення стилів для усіх елементів тіла

Закінчивши з підготовкою, залишається верстка за прикладом розробленого дизайну. Для початку треба зверстати компоненти-константи: основний, колонтитули, макет каруселі та меню.

Вся сторінка складається з контейнера, який вирівнює всі елементи по центру вертикально. Верхній елемент — це шапка сайту, яка займає всю ширину контейнера. Під нею розташований інший контейнер з фіксованою шириною (1440 пікселів), в якому знаходяться карусель та горизонтальний контейнер з меню та роутингом.

Верхній колонтитул складається з основного контейнера, який займає всю ширину екрана і має фіксовану висоту (90 пікселів), з темним фоном (bg-dark). Всередині знаходиться внутрішній контейнер обмежений фіксованою шириною (1440 пікселів), який використовується для розміщення двох контейнерів з обох країв: лівий контейнер містить логотип (54x54 пікселів) та назву (h1), а правий - перемикача мови, кошика та авторизації. Обидва контейнери вирівняні вертикально і мають невеликі відступи між елементами для естетичного вигляду.

Карусель знаходиться нижче верхнього колонтитула та займає всю ширину контейнера (1440 пікселів), а в висоту (300 пікселів). Замість зображення, на перший час краще поставити заглушку у вигляді сірого фону.

Меню складається з вертикального контейнера з заокругленим верхнім краєм та фоновим кольором. Всередині є ще один вертикальний контейнер зі встановленою шириною та відступами, першим елементом котрого є кнопка з фіксованою висотою, закругленими кутами та рамкою, яка змінює свій фон при наведенні й активації. Після неї йде перелік елементів, кожен з яких складається з іконки та тексту. Іконки змінюють свій вигляд при наведенні. Далі йде ще одна кнопка з такими ж властивостями, як і перша. Після неї розміщений ще один перелік елементів з іконками та текстом, аналогічний попередньому. В кінці компонента розташований нижній колонтитул з темним фоном та текстом (h2).

На цьому верстка компонентів-констант завершена (рис. 2.8) та придатна для подальшого додання функціонала.

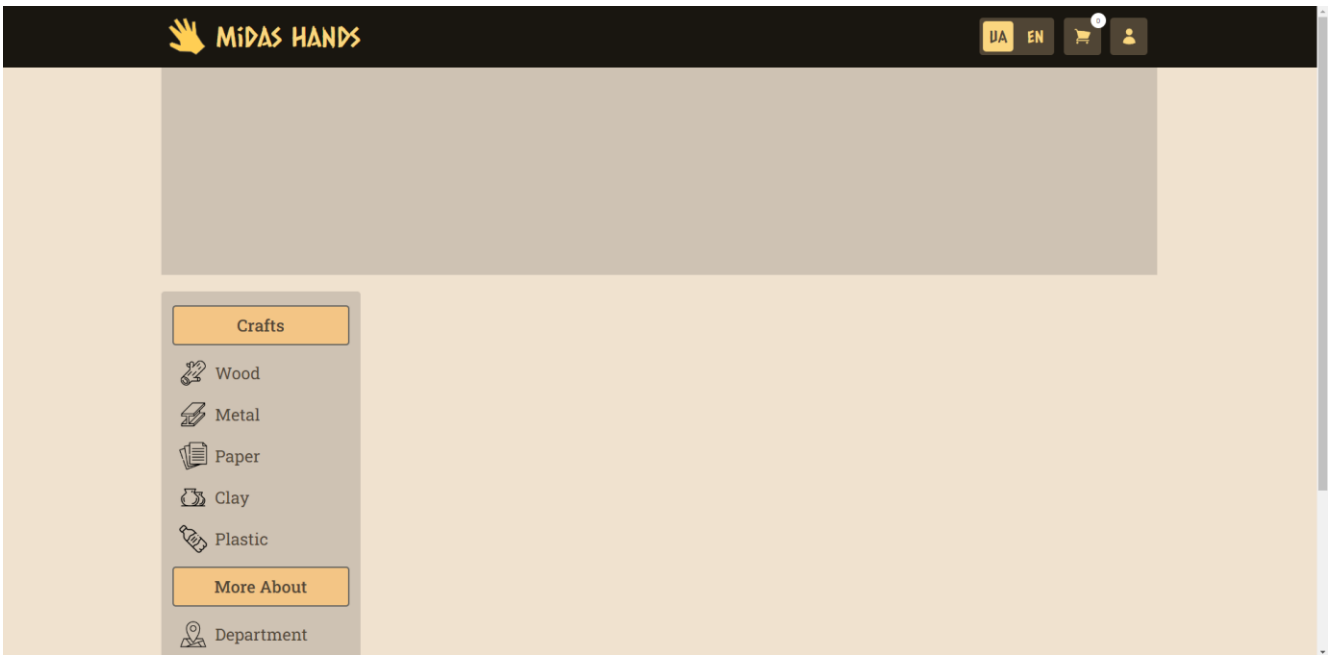


Рис. 2.8 Завершена верстка компонентів-констант

Але на цьому верстка не завершена. Для подальшої розробки роутингових компонентів треба замінити роутинг на той компонент, який зараз розробляється.

Почнемо з головної сторінки. Вона містить контейнер, який вирівнює елементи по горизонтальних краях. Перший елемент — це панель з відступами, висотою та закругленими кутами, яка містить текст (h5) і блок для сортування. Блок для сортування має кнопку з текстом, яка змінює колір фону при наведенні та натисканні. Поруч з панеллю розташована кнопка (для відправлення замовлення) великого розміру, яка змінює свій вигляд при наведенні натисканні. Нижче розташований контейнер з товарами, які розміщуються в кілька рядів і мають відступи між собою. В кінці сторінки знаходиться контейнер з кнопкою по центру, яка має фіксовану ширину, висоту та закруглені кути. Кнопка змінює свій вигляд при наведенні та натисканні.

Також важливим компонентом головної сторінки є компонент товару – це вертикальний контейнер з фіксованими шириною і висотою, відступами з усіх боків та закругленими кутами. Верхня частина контейнера містить відносний блок з фіксованими розмірами, в якому розміщене зображення та кнопка. Зображення займає всю ширину та висоту блоку, має закруглені кути та рамку, яка змінюється при наведенні та натисканні. В правому верхньому куті зображення знаходиться

кнопка, яка має фіксовані розміри, закруглені кути та змінює свій колір при наведенні та натисканні. В середині кнопки знаходиться іконка з кошика. Під зображенням розташований заголовок з текстом (h6), а ще нижче розташований блок з текстом ціни (h4) та текстом рейтингу (h3), які вирівняні по краях горизонталі та мають текст, що можна виділяти. Об’єднавши обидва зверстані компоненти – головна сторінка готова (рис. 2.9).

Рис. 2.9 Завершена верстка головної сторінки

Компонент “Сторінка товару” містить верхній контейнер, де зліва знаходиться кнопка зі стрілкою, по центру — заголовок сторінки з текстом (h3), а справа — порожній простір для вирівняння. Нижче розташовані два основні контейнери: перший контейнер (зліва) містить вертикально вирівняний блок, в якому знаходиться зображення товару з закругленими кутами та рамкою. Під зображенням розташований блок з ціною (h4) та рейтингом (h3). Ще нижче знаходиться інформаційний блок про матеріал товару з іконкою та текстом (h3). Потім йде кнопка кошика з текстом (h3), яка змінює свій вигляд при наведенні та натисканні. В кінці цього контейнера розташована велика кнопка "Залишити Коментар". другий контейнер (справа) містить вертикально вирівняні блоки з

інформацією про товар. Перший блок містить назву товару, де текст (h3) вирівняний по горизонталі, а знизу — текстове поле з фоном і текстом (h3). Другий блок містить опис товару з аналогічною структурою. Третій блок присвячений коментарям, де кожен коментар представлений у вигляді вертикального контейнера. У кожному коментарі є заголовок (h3) з іменем автора, його ініціали (h5) та рейтинг (h3). Під заголовком розміщений текст коментаря (h3) з фоном. На цьому верстка сторінки товару закінчена (рис. 2.10)

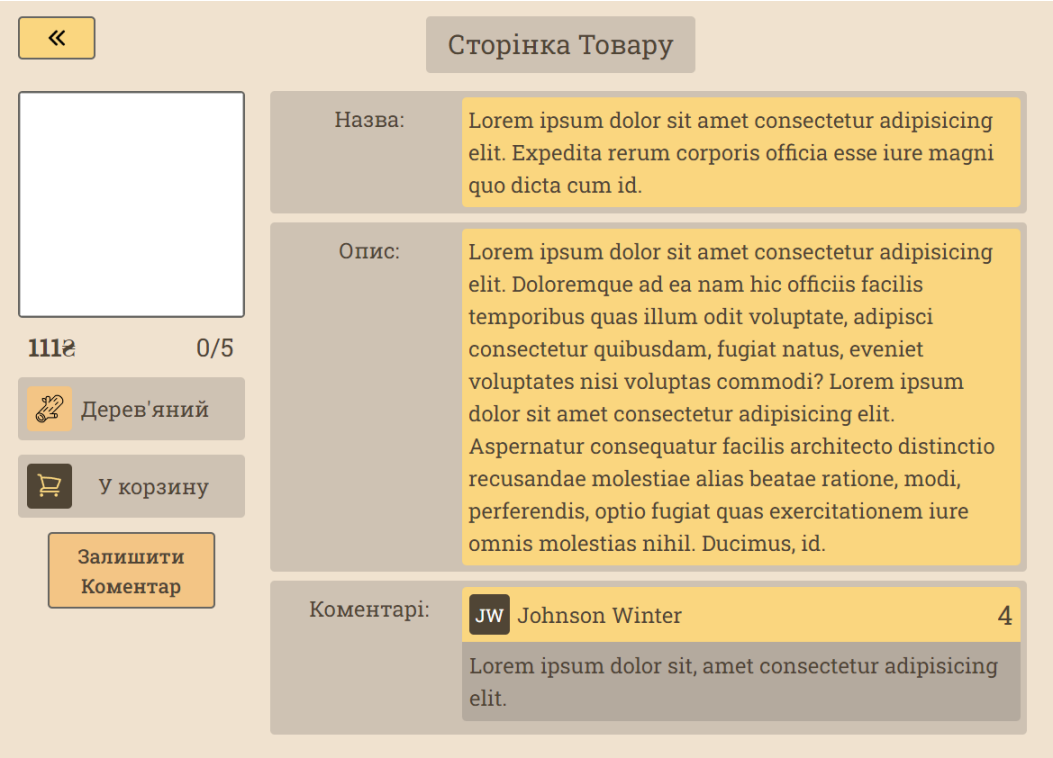


Рис. 2.10 Завершена верстка сторінки товару

Компонент “Детальніше” містить вертикальний контейнер з правим відступом. Усередині цього контейнера є два блоки: перший блок містить три вертикально вирівняні елементи з відступами знизу. Перший елемент — це кнопка з текстом "Замовити Свій Дизайн", яка має відступи всередині, закруглені кути, рамку та змінює свій колір при наведенні та натисканні. Другий та третій елементи — це кнопки з іконками та текстом "Відділення" і "Контакти", які також мають відступи всередині, закруглені кути, рамку і змінюють свій фон при взаємодії. Другий блок містить три горизонтально вирівняні елементи, кожен з яких має відступи всередині, закруглені кути, рамку, іконку та текст. Ці елементи також

змінюють свій фон при наведенні та натисканні. Перший елемент містить текст "Доставка", другий — "Гарантія", третій — "Про Нас". Оскільки в цьому компоненті нічого окрім 6 кнопок немає – сторінка “Детальніше” готова (рис. 2.11).

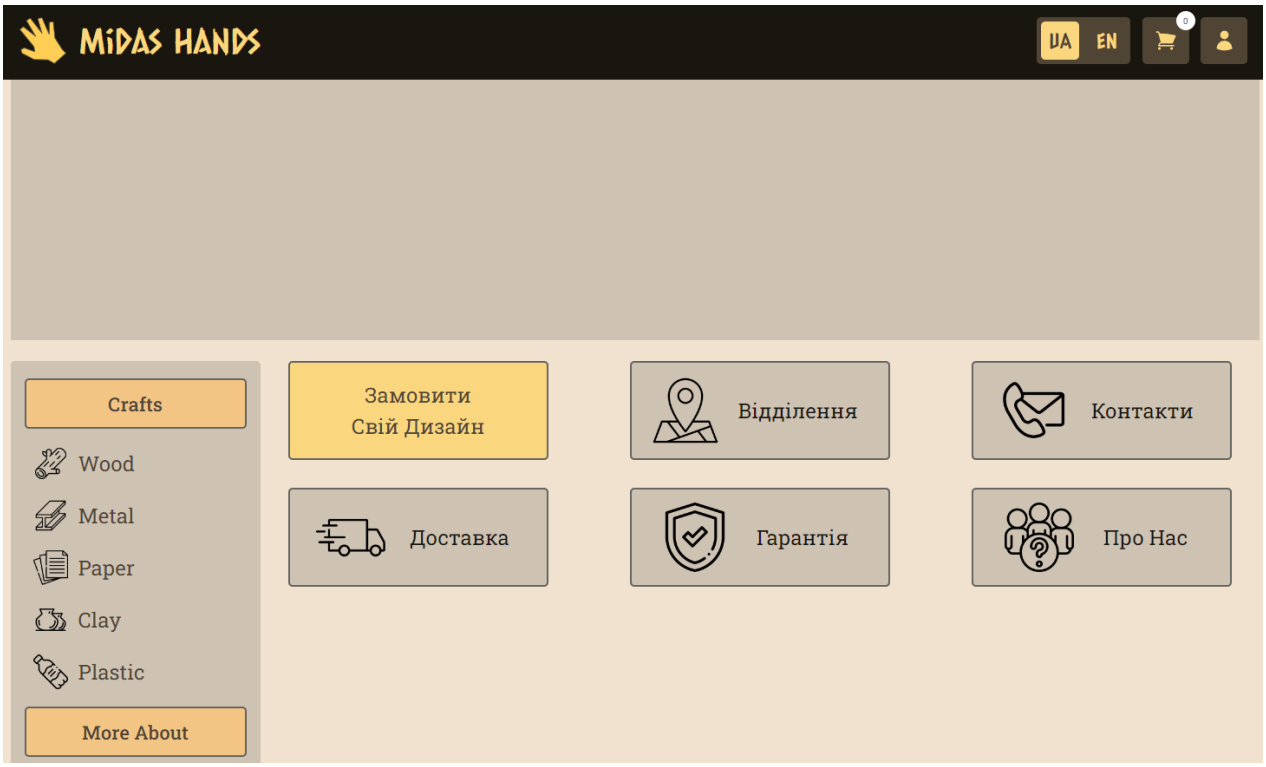


Рис. 2.11 Завершена верстка сторінки “Детальніше”

Наступний компонент – сторінка з можливими відділеннями. На ній є контейнер, що вирівнює елементи по горизонталі. Зліва знаходиться кнопка зі стрілкою, яка має фіксовані розміри, закруглені кути, рамку та змінює свій колір при наведенні та натисканні. В центрі розташований заголовок "Відділення" з великим текстом (h3), закругленими кутами та відступами. Справа залишено порожній простір для вирівняння. Нижче розміщені вертикальні блоки з інформацією про відділення, оскільки кількість блоків залежить від побажань замовника – нехай їх буде чотири. Кожен блок має фіксовану висоту, закруглені кути та відступи зверху. Фон блоку напівпрозорий. Всередині кожного блоку зліва розташоване зображення відділення з фіксованими розмірами, яке займає всю висоту блоку. Справа від зображення розміщений вертикальний текстовий блок з назвою відділення (h3), адресою (h6) та номером телефону (h6). Текстовий блок має відступи зліва, вирівняний по вертикалі та відправляє за посиланням, до

місцезнаходження обраного відділення, на “Google maps”. На цьому верстку сторінки з відділеннями – завершено (рис. 2.12).

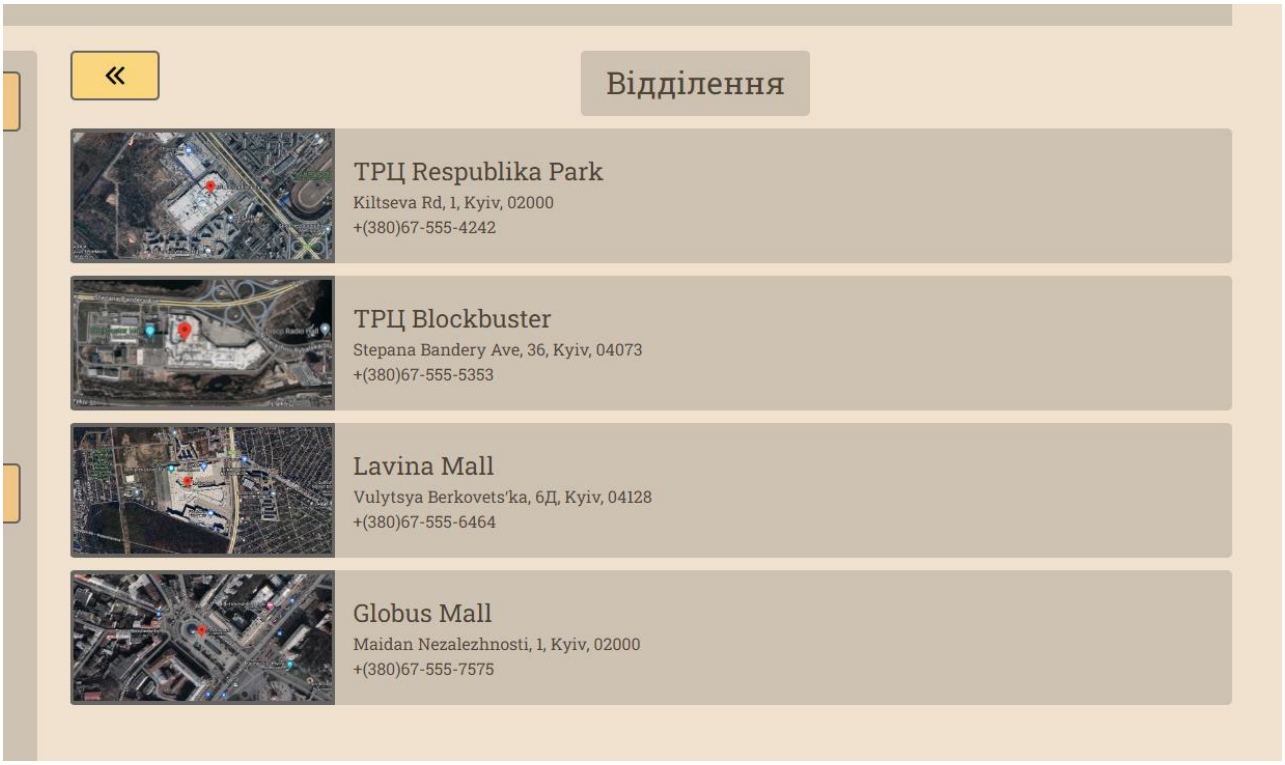


Рис. 2.12 Завершена верстка сторінки з можливими відділеннями

Нехай наступною сторінкою буде компонент з контактними даними. На сторінці є контейнер, що вирівнює елементи по горизонталі. Ліворуч розташована кнопка зі стрілкою, яка має фіксовані розміри, закруглені кути, рамку та змінює свій колір при наведенні та натисканні. В центрі розміщений заголовок "Контакти" з великим текстом (h3), закругленими кутами та відступами. Справа залишено порожній простір для вирівняння. Під цим блоком розташовані кілька посилань на відповідну соціальну мережу, кожне з яких складається з двох частин: ліворуч знаходиться квадрат з іконкою, праворуч – текст (h3). Квадрат має фіксовані розміри, закруглені кути, рамку і змінює свій колір при наведенні та натисканні. При наведенні курсора на посилання, правий кінець квадрата вирівнюється по горизонталі, а текстовий блок змінює свій фон. Текстовий блок посилання має відступи та закруглені кути. Тексти "Електронна пошта", "Telegram", "LinkedIn", "GitHub" вирівняні вертикально та горизонтально в межах блоку. Внизу сторінки

розміщений додатковий текст (h3). Текст вирівняний по центру та має кольорове виділення. На цьому компонент готовий (рис. 2.13).

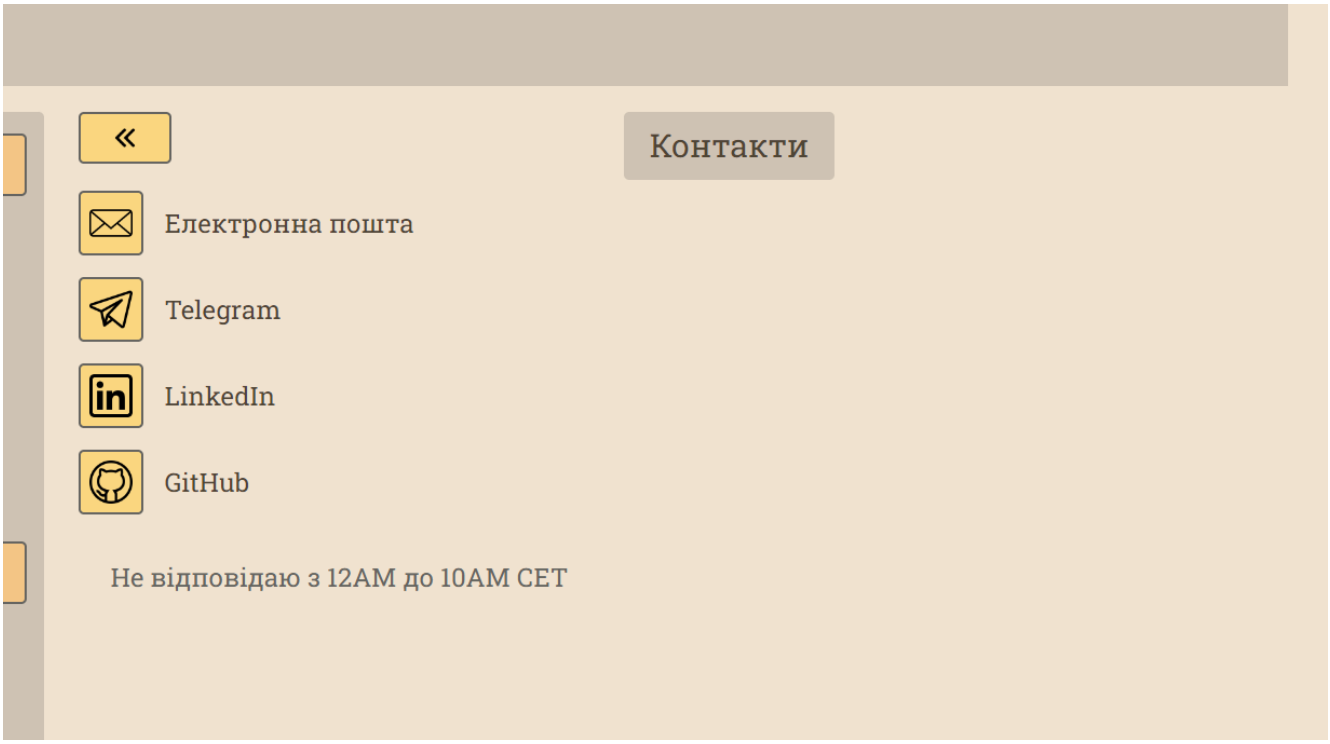


Рис. 2.13 Завершена верстка сторінки з контактами

Верстка сторінки з можливими сервісами доставки не велика: складається з горизонтального контейнера з трьома основними елементами. Ліворуч розташована кнопка з фіксованими розмірами, закругленими кутами, рамкою і стрілкою, яка змінює свій колір при наведенні та натисканні. В центрі знаходиться заголовок "Доставка" з великим текстом (h3), закругленими кутами та відступами. Справа залишено порожній простір для незвичайного вирівняння елементів. Нижче розміщені два посилання у вигляді карток, які пересилають на сайт відповідного сервісу доставки. Кожна картка має фіксовану ширину та висоту, відступи, рамку, закруглені кути та змінює свій колір при наведенні та натисканні. Всередині кожної картки є три елементи: ліворуч розташований квадрат з іконкою, що має фіксовані розміри, закруглені кути та білий фон; в центрі карток розміщений заголовок (h3) з назвою служби доставки; праворуч залишено порожній простір для центрування двох минулих елементів по лівому краю та центру відповідно. Весь контент кожної картки вирівняний по горизонталі та має приємний інтерфейс з естетичною

колірною гамою. На цьому компонент з можливими сервісами доставки – готовий (рис. 2.14).

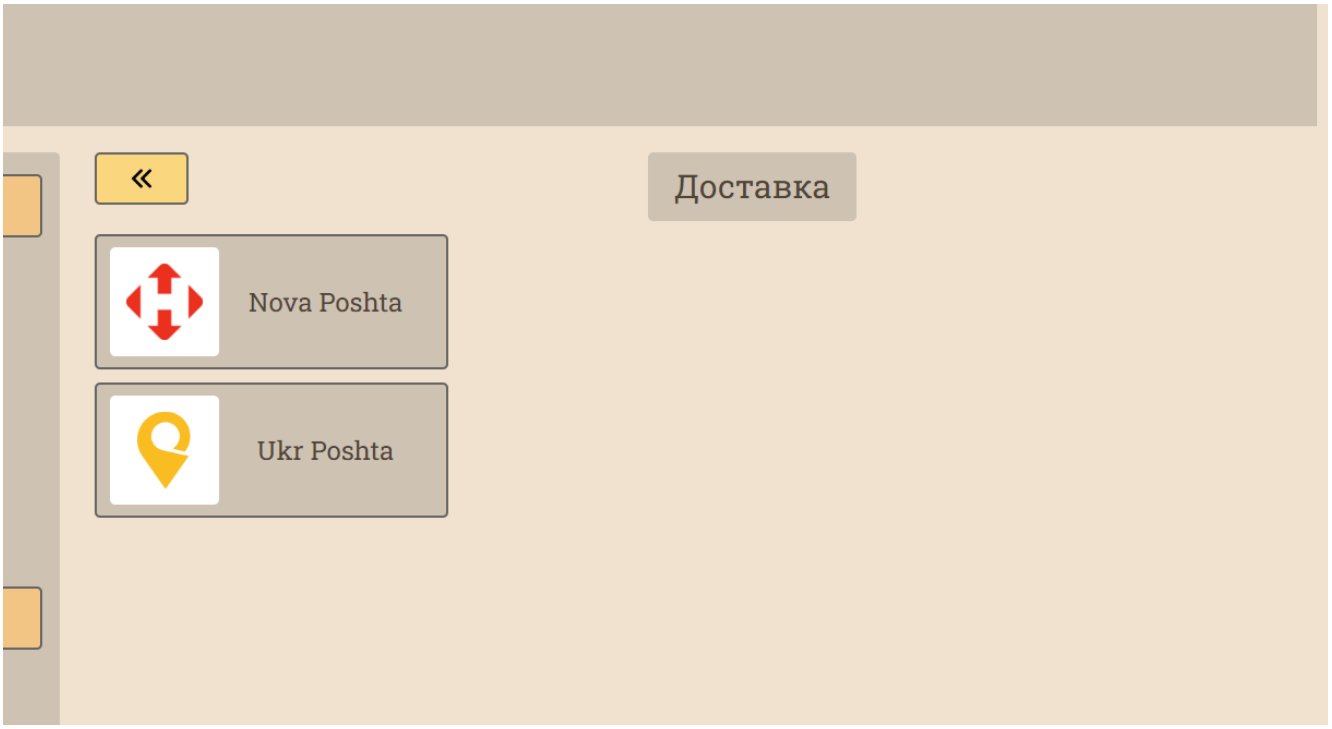


Рис. 2.14 Завершена верстка сторінки з сервісами доставки

Компонент “Гарантія” – найменший для верстки. На сторінці представлений горизонтальний контейнер з трьома основними елементами: зліва знаходиться кнопка з фіксованими розмірами, закругленими кутами, рамкою та іконкою стрілки, що змінює свій колір при наведенні та натисканні. Посередині розташований заголовок "Гарантія" великим шрифтом (h3), з закругленими кутами та відступами, справа залишено порожній простір для центрування минулих елементів по лівому краю та центу відповідно. Нижче знаходиться текстовий заголовок "З приводу:" (h3), а під ним розміщені два пункти, кожен з яких складається з невеликого круглого індикатора з фіксованими розмірами й тексту поруч (h3). Ці елементи вирівняні горизонтально. Внизу сторінки розташована кнопка, яка закликає користувача перейти на сторінку "Контакти". Кнопка має фіксовані розміри, закруглені кути, рамку і змінює свій колір та текстовий стиль при наведенні та натисканні. Загальний стиль компонування елементів є чистим і упорядкованим, з

використанням приємної колірної гами та гармонійного відступу між елементами. На цьому верстка завершена (рис. 2.15).

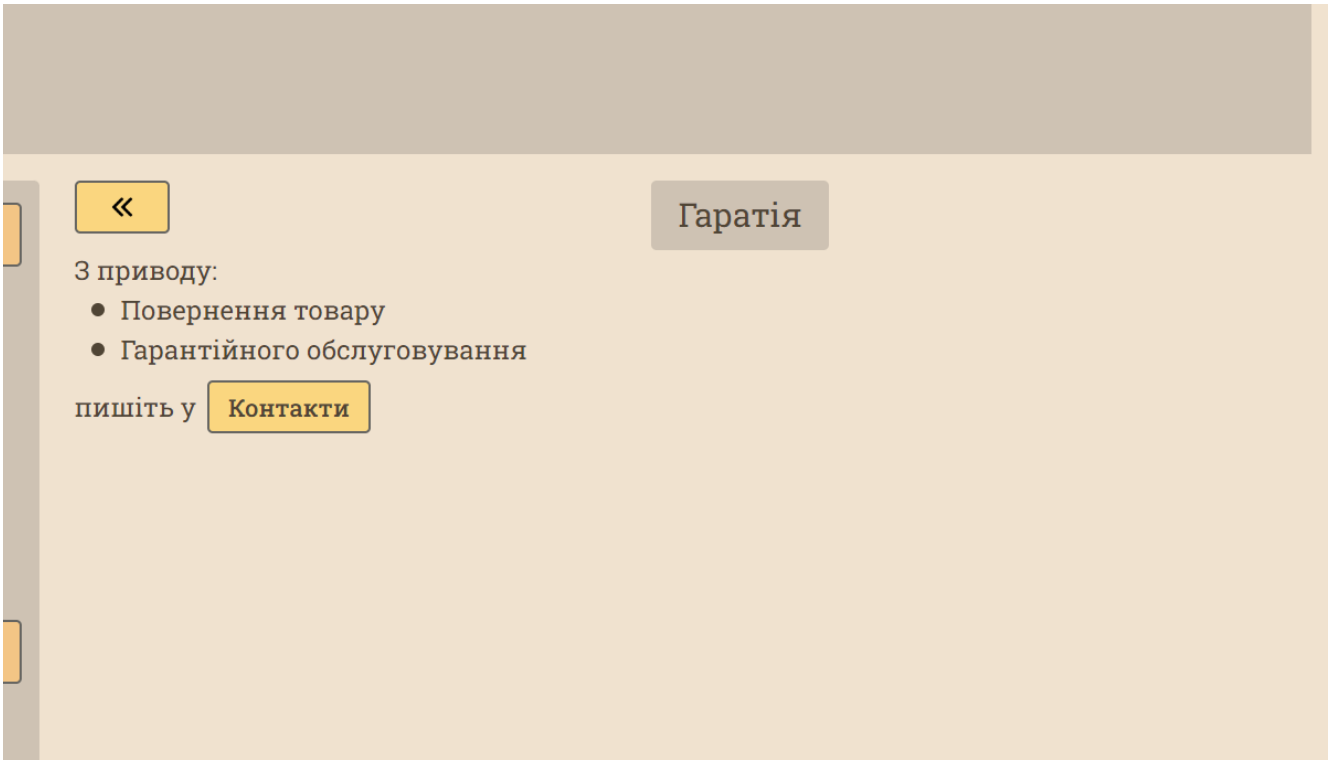


Рис. 2.15 Завершена верстка сторінки “Гарантія”

Остання сторінка – компонент “Про нас”. У верхній частині розташований горизонтальний контейнер з трьома основними елементами. Зліва знаходиться кнопка з іконкою стрілки, закругленими кутами, рамкою та можливістю змінювати колір при наведенні та натисканні. Посередині розміщений заголовок "Про нас" (h3) з закругленими кутами та відступами. Справа залишено порожній простір для рівняння компонентів. Нижче вміщено центральний контейнер з текстовим контентом, вирівняним по центру сторінки та фіксованою шириною (700 пікселів). Цей контейнер містить кілька вертикально вирівняних блоків. Перший блок включає кілька рядків тексту (p – 18 пікселів) з назвами університету, інституту та кафедри. Кожен рядок окремо вирівняний по центру. Далі розташований заголовок "КВАЛІФІКАЦІЙНА РОБОТА" (h4), також вирівняний по центру. Ще нижче розташовні три рядки тексту (h5), що описують тему роботи. Під ними розміщено декілька пунктів з текстом (h6), кожен з яких вирівняний горизонтально і містить підкреслений текст поруч із заголовком (span). Загальний стиль компонування

елементів є чистим і упорядкованим, з чіткими відступами між елементами та приємною колірною гамою, яка підкреслює основні частини тексту. Нічого окрім контейнерів та тексту для верстки використано не було (рис. 2.16).

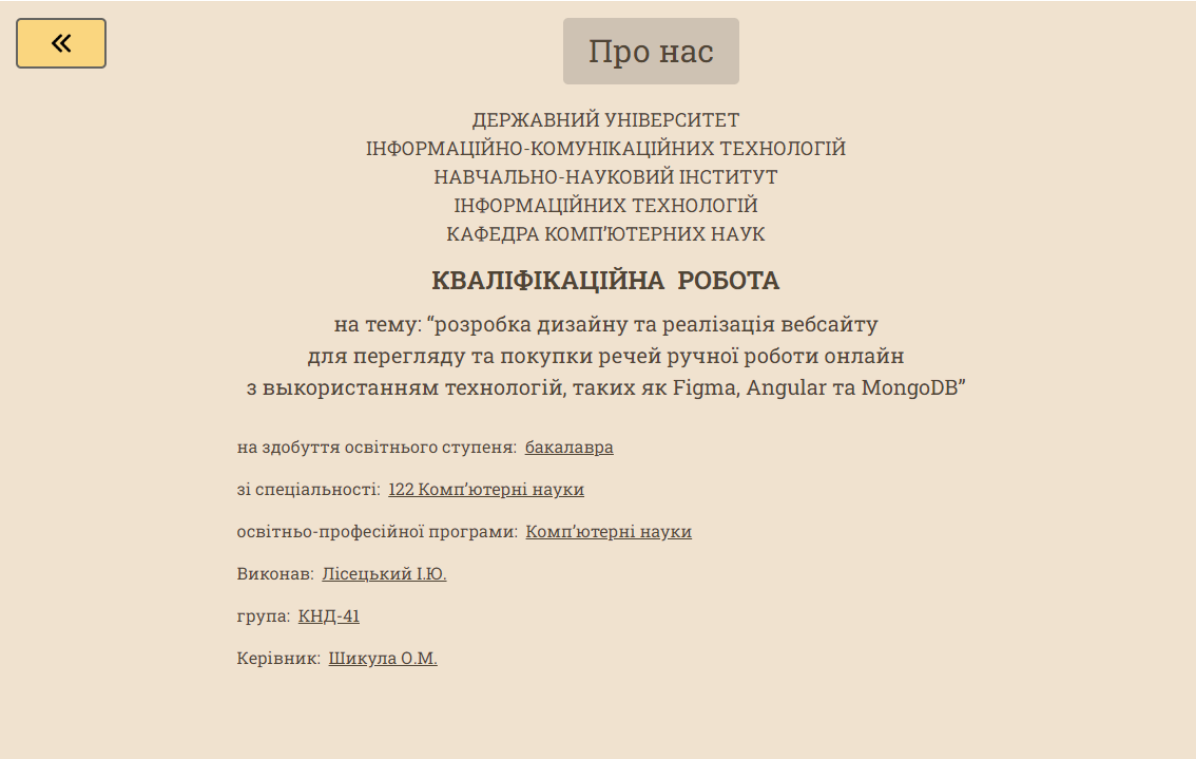


Рис. 2.16 Завершена верстка сторінки "Про нас"

Нарешті верстка сторінок завершена, але ще залишається п'ять компонентів, чотири з яких будуть з'являтися при натисканні відповідної кнопки, а п'ятий – це фон (blank.component), який візуально та технічно забороняє користувачу натискати на контент за межами одного з чотирьох активних компонентів.

Почнемо саме з цього компонента. Для початку його треба додати у головний компонент додатка (app.component) та зробити відносний стиль. Сам фоновий компонент – це фіксований елемент, що займає всю ширину та висоту екрана. Він має фоновий колір (primaryTechFaded) і розташований на верхньому шарі з індексом 12 (z-index). Таким чином уся сторінка платформи стає неклікабельною та набуває прозорий сірий колір (рис. 2.17).

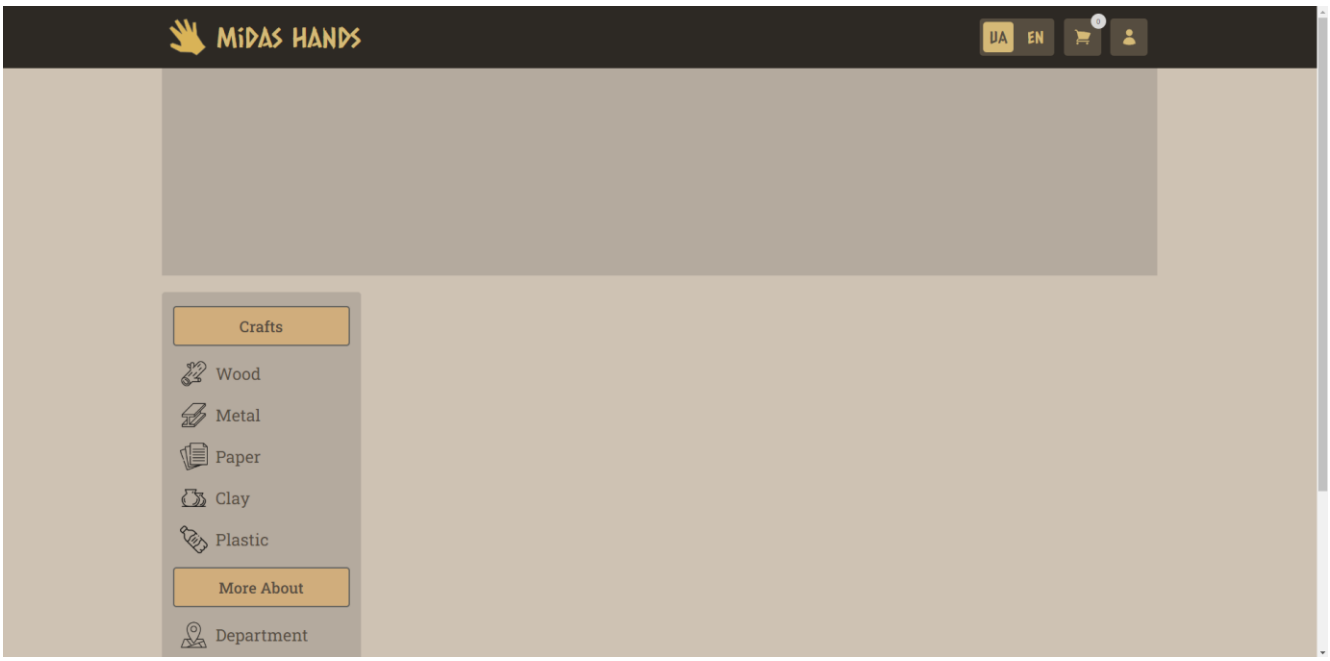


Рис. 2.17 Завершена верстка фонового компонента

Нехай наступним компонентом буде кошик. Його також потрібно спочатку додати до головного компоненту додатка та надати відносний стиль. Сам компонент розташований посередині екрану з фіксованим позиціюванням та індексом 17 (z-index). Основний контейнер має фіксовану ширину (1150 пікселів) і містить елементи, розташовані вертикально, з фоном (bg-background) та обведенням. Заголовок "Кошик" (h3) розміщений по центру, а кнопка для закриття вікна з іконкою хреста знаходиться у верхньому правому куті, має фіксоване позиціювання та змінює колір при наведенні та натисканні. Секція з товарами має горизонтальну прокрутку, фіксовану висоту (356 пікселів) на випадок, якщо кошик буде пустий, ширину вміщення контенту (w-fit), а також містить контейнер з макетом картки товарів. Кожна картка має фіксовану висоту (356 пікселів), ширину (256 пікселів), відносне позиціювання, округлення та відступ справа. Також вона включає кнопку для видалення товару, що змінює колір при наведенні та натисканні, зображення товару, назву (h6), ціну (h4) та елементи управління кількістю: кнопки для зменшення та збільшення кількості товару, а також текст кількості (h6) між ними. Внизу модального вікна розташована секція, зліва котрої контейнер зі загальною ціною, яка складається з тексту (h3) та самої ціни (h4) а

справа – кнопка для покупки, котра змінює колір при наведенні та натисканні. На цьому верстку кошика завершено (рис. 2.18).

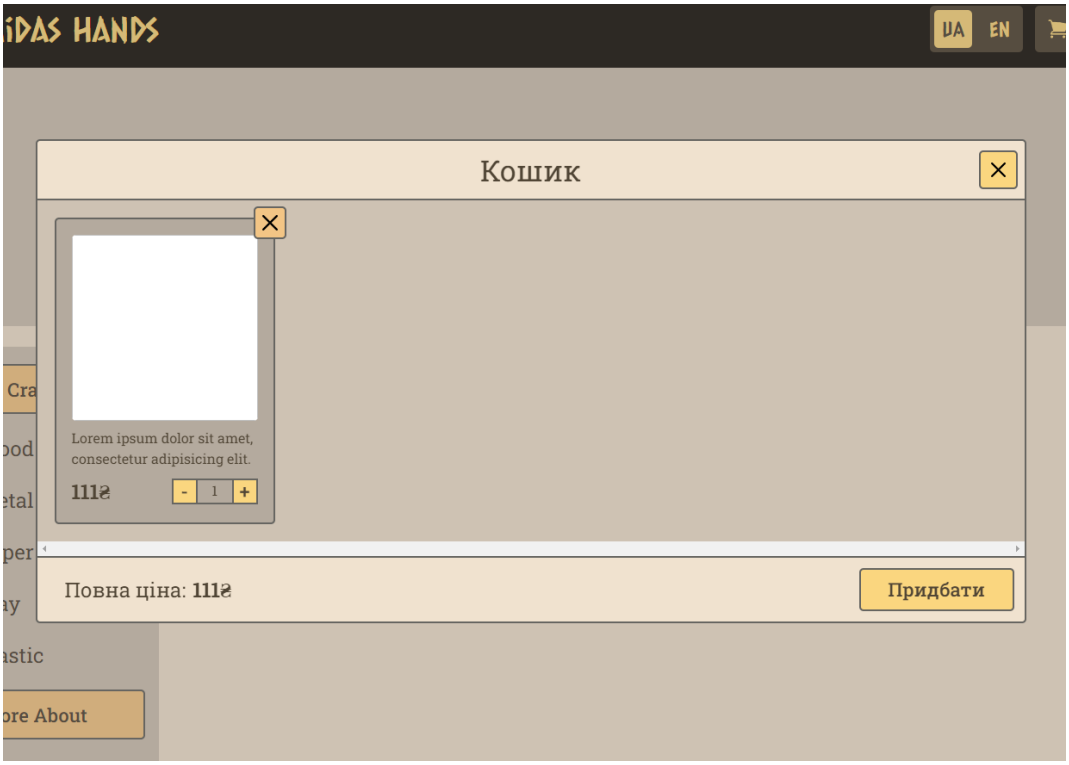


Рис. 2.18 Завершена верстка кошика

Далі верстка компонента аутентифікації. Для початку в головному компоненті додатка треба закоментувати кошик та створити компонент аутентифікації з відносним стилем. Оскільки зараз етап верстки, без функціонала, а в дизайні цей компонент має два варіанти, потрібно зверстати саме той, що має усі поля вводу. Верстка починається з контейнера з внутрішньою структурою у вигляді колонки, фіксованою шириною (444 пікселів) та висотою (530 пікселів). Він має рамку, фоновий колір (bg-background) та закруглені кути. Вгорі справа є кнопка з хрестиком, що має абсолютний статус, закруглені краї, рамки, фіксовану ширину (38 пікселів), висоту (38 пікселів) та має фоновий колір (bg-primaryTech), який змінюється при наведенні та натисканні. У середині контейнера є заголовки "Логін" та "Реєстрація" (h3). Кожен має половину максимальної ширини, рамку та фоновий колір (bg-primaryTech). Заголовок "Логін" має додаткову рамку праворуч та знизу, а також закруглення кутів справа-знизу. Під заголовками знаходиться форма з текстом (h3) та полями для введення даних: "Ім'я", "Прізвище", "Електронна пошта"

та "Пароль". Кожне поле має максимальну ширину та фіксовану висоту (50 пікселів), а також рамки. Фон полів має відтінок первинного технічного кольору, але з прозорістю двадцять п'ять відсотків. Знизу форми знаходиться кнопка "Зареєструватися" з фіксованою шириною (280 пікселів), висотою (58 пікселів) та закругленими краями. Також вона має рамку та фоновий колір (bg-primaryTech), який змінюється при наведенні та натисканні. Таким чином вийде повністю зверстаний компонент аутентифікації (рис. 2.19).

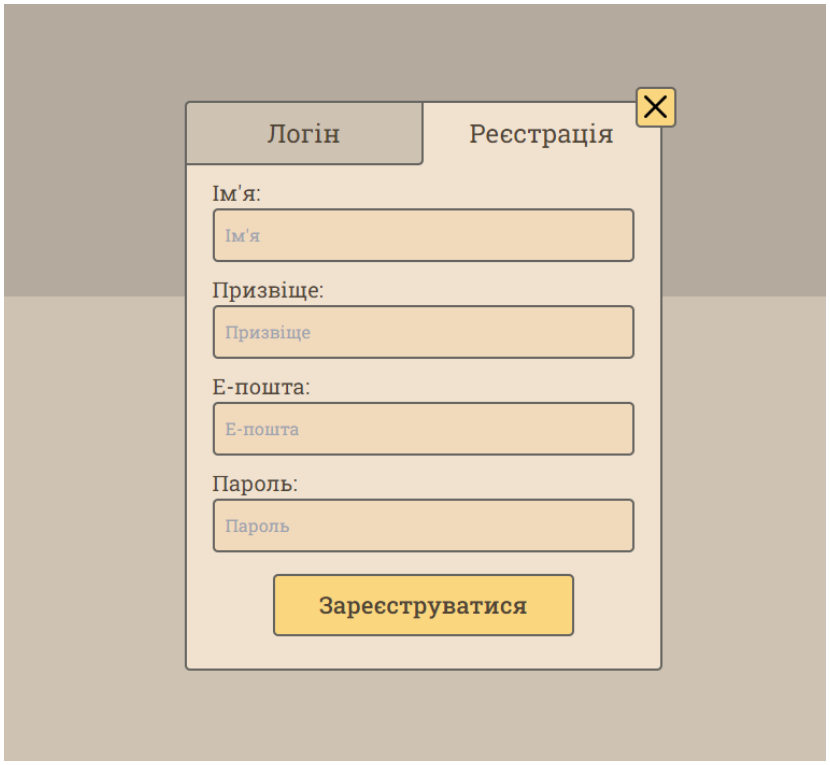


Рис. 2.19 Завершена верстка компонента аутентифікації

Наступним буде компонент замовлення. Спочатку, в головному компоненті додатка, треба закоментувати минулий об'єкт верстки, а потім, так само, додати компонент замовлення з відносним позиціюванням. Він створений з форми, що має фіксовану ширину (750 пікселів), індекс 20 (z-index), рамку, фоновий колір (bg-background), округлені кути та внутрішні відступи (22 пікселів). В правому верхньому куті форми є кнопка закриття у вигляді хрестика, яка має абсолютне позиціювання, фіксовану ширину (38 пікселів), висоту (38 пікселів), округлені краї, рами та фоновий колір (bg-primary), який змінюється при наведенні та натисканні. Заголовок форми "Залишити замовлення" (h3) розташований по центру і має

нижній відступ (22 пікселів). В середині форми є контейнер зі зображенням та іншим контейнером з напрямком колонки. Зображення має фіксовану ширину (220 пікселів), висоту (220 пікселів), правий відступ (22 пікселів), рамку та білий фон з округленими кутами. Контейнер, поруч зі зображенням, містить тексти (h5), текстове поле та кнопки матеріалів. Текстове поле має максимальну ширину, фіксовану висоту (65 пікселів), внутрішній відступ (12 пікселів з боків та 5 пікселів зверху та знизу), рамку, округлені кути та фоновий колір (bg-primaryTechFaded). Далі йде блок вибору матеріалу. Він містить п'ять кнопок, кожна з яких являє собою квадрат з фіксованою шириною (78 пікселів), висотою (78 пікселів), внутрішнім відступом (19 пікселів), рамкою, фоновим кольором (bg-secondary) і округленими кутами. Також кожен квадрат містить зображення матеріалу з максимальною шириною та висотою. Під блоком з вибором матеріалу розміщений текст (h5) та текстовий блок для введення опису замовлення, який має максимальну ширину, фіксовану висоту (140 пікселів), внутрішні відступи (12 пікселів з боків та 8 пікселів зверху та знизу), рамку, фоновий колір (bg-primaryTechFaded) та округлені кути. Нижче розташований блок з текстом (h5), текстовим полем для введення електронної пошти та кнопкою відправки замовлення. Поле має ширину (390 пікселів), висоту (40 пікселів), внутрішній відступ (10 пікселів), рамку, фоновий колір (bg-primaryTechFaded) та округлені кути. Кнопка відправки має ширину (220 пікселів), висоту (50 пікселів), рамку, округлені кути та фоновий колір (bg-primary), який змінюється при наведенні та натисканні. Таким чином компонент для залишення замовлення завершений (рис. 2.20).

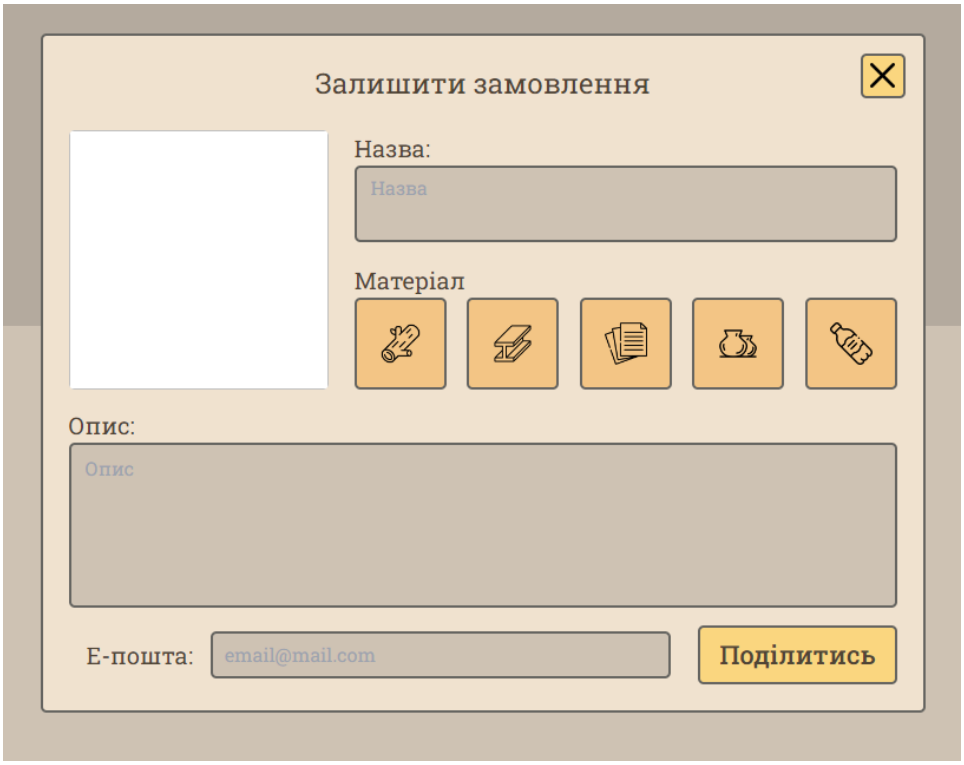


Рис. 2.20 Завершена верстка компонента замовлення

Останній компонент для верстки – залишення коментаря. Цей компонент є вікном, розташованим у центрі екрана, з фіксованим позиціюванням, яке має ширину (480 пікселів) та висоті (440 пікселів). Воно має округлені кути, фоновий колір (bg-background), рамку та внутрішні відступи (16 пікселів). У верхньому правому куті вікна розміщено кнопку закриття з шириною (45 пікселів), висотою (45 пікселів), внутрішніми відступами (12 пікселів). Вона також має округлені кути, рамку та фоновий колір (bg-primary), який змінюється при наведенні та натисканні. В середині кнопки знаходиться іконка у вигляді хрестика, що центрується і не повторюється. У верхній частині вікна розташований заголовок "Залишити коментар" (h3). Нижче знаходиться заголовок блоку - "Оцінка" (h6) та блок з полями для оцінки. У ньому розташовані п'ять квадратних кнопок з шириною (75 пікселів), висотою (75 пікселів), рамкою, фоновим кольором (bg-primary) та округленими кутами. Кожна кнопка розміщується з рівними проміжками одна від одної та має всередині цифру від 1 до 5. Далі розміщено форму з текстом (h6) та текстовою областю для введення відгуку. Ця область має максимальну ширину, фіксовану висоту (120 пікселів), внутрішні відступи (12 пікселів з боків та 8 пікселів зверху та

знизу), рамку, фоновий колір (bg-primaryTechFaded) та округлені кути. Нижче форми розташована кнопка відправки відгуку з фіксованою шириною (280 пікселів) та висотою (50 пікселів). Вона має рамку, округлені кути та фоновий колір (bg-primary), який змінюється при наведенні та натисканні. Після верстки усіх елементів – компонент “Коментар” готовий (рис. 2.21).

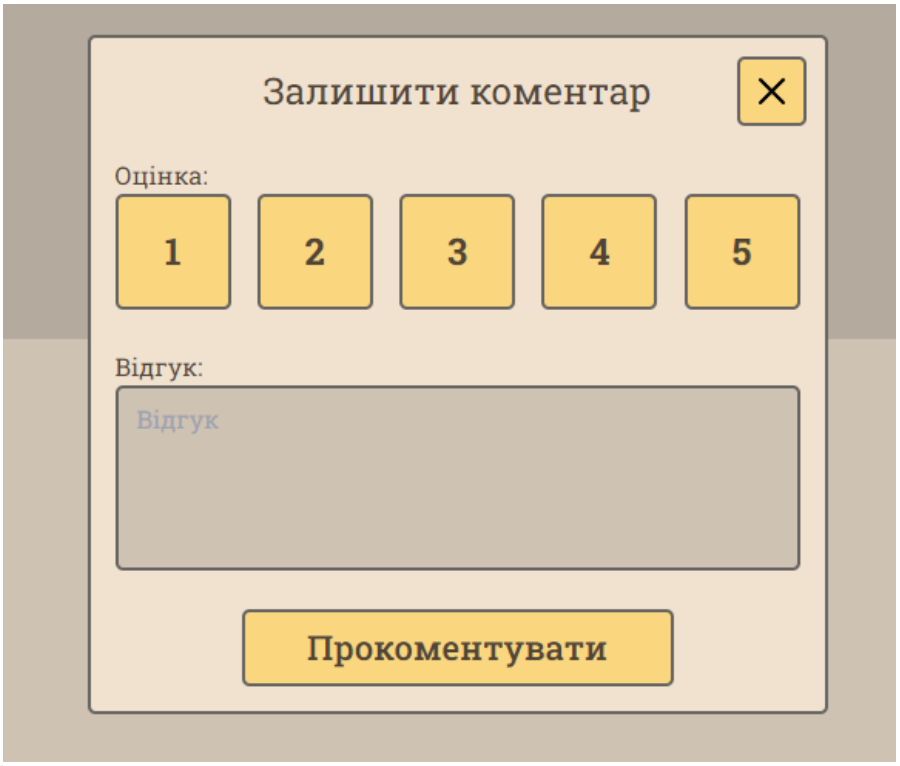


Рис. 2.21 Завершена верстка компонента залишення коментаря

2.3 Реалізація технічної частини вебсайту

Розробка технічної частини вебсайту можна розбити на декілька пунктів: роутингова система, робота з сервісами, анімації, поява зі зникненням компонентів та копіювання елементів.

Для початку треба створити роутингову систему, щоб далі можна було тестувати кожну сторінку окремо. Перший етап – створення роутингового елементу (router-outlet) у головному компоненті додатка, а саме після компоненту меню, в тому ж контейнері. Далі, щоб забезпечити навігацію між різними компонентами через маршрути, у файлі роутингу (app.routes.ts) для масиву routes визначені маршрути до різних компонентів. Кожен маршрут вказує шлях (URL) та компонент, який відображається при переході за цим шляхом. Наприклад, порожній шлях “/”

відображає компонент магазину, а шлях “/item-page” - компонент сторінки товару. Є також маршрути для компонента "Детальніше" (/about) та його підрозділів: “/about/contacts”, “/about/delivery”, “/about/department”, “/about/guarantee” та “/about/us”, які відображають відповідні компоненти (рис. 2.22).

```
import { Routes } from '@angular/router';
import { ShopComponent } from './components/shop/shop.component';
import { AboutComponent } from './components/about/about.component';
import { AboutContactsComponent } from './components/about-contacts/about-contacts.component';
import { AboutDeliveryComponent } from './components/about-delivery/about-delivery.component';
import { AboutDepartmentComponent } from './components/about-department/about-department.component';
import { AboutGuaranteeComponent } from './components/about-guarantee/about-guarantee.component';
import { AboutUsComponent } from './components/about-us/about-us.component';
import { ShopItemPageComponent } from './components/shop-item-page/shop-item-page.component';

export const routes: Routes = [
  { path: '', component: ShopComponent},
  { path: 'item-page', component: ShopItemPageComponent},
  { path: 'about', component: AboutComponent},
  { path: 'about/contacts', component: AboutContactsComponent},
  { path: 'about/delivery', component: AboutDeliveryComponent},
  { path: 'about/department', component: AboutDepartmentComponent},
  { path: 'about/guarantee', component: AboutGuaranteeComponent},
  { path: 'about/us', component: AboutUsComponent},
];
```

Рис. 2.22 Роутингова система файлу “app.routes.ts”

Далі потрібно додати переміщення по маршрутах у кожен компонент, що потребує цього. Наприклад компонент “Гарантія” має два методи для обробки кліків. Метод “backClicked()”, який однаковий для всіх підрозділів маршруту “/about”, повертає користувача на сторінку “Детальніше” після натискання кнопки назад. Другий метод – “contactsClicked()” перенаправляє користувача на сторінку “/about/contacts” (рис. 2.23). А щоб метод викликався по натиканню на кнопку, потрібно додати до цього елементу атрибут “click”.

```
export class AboutGuaranteeComponent {

  constructor(private router: Router) {}

  backClicked() {
    this.router.navigate(['/about'])
  }
  contactsClicked() {
    this.router.navigate(['/about/contacts'])
  }
}
```

Рис. 2.23 Файл “about-guarantee.component.ts” з роутинговими методами

Компонент “Товар магазину” має метод “onItemClick()”, який перенаправляє користувача на сторінку товару (/item-page), а у компоненті “Детальніше” є метод “onCardClick(i:number)”, який виконує навігацію до відповідного маршруту в масиві.

У компоненті меню є методи для обробки кліків:

1. “onMaterialClick()” перенаправляє на головну сторінку (/).

2. “onDetailClick(i:number)” перенаправляє на відповідний підрозділ маршруту “/about”.

3. “onCraftClick()” перенаправляє на головну сторінку (/).

4. “onAboutClick()” перенаправляє на сторінку “/about”.

Тепер, коли роутингова система готова, можна налаштувати кожен сервіс. І для початку – мови (language.service.ts) (рис. 2.24).

Сервіс мови забезпечує підтримку багатомовного інтерфейсу для додатка. Він має дві мовні версії текстів: українську та англійську, зберігаючи їх в окремих об'єктах “uaText” та “enText” відповідно. Тексти розподілені по розділах додатка, де кожен розділ – це окремий компонент вебсайту.

Сервіс має статус, який визначає, яка мова використовується в цей момент. Цей статус зберігається у змінній status та ініціалізується як “false”, що означає використання української мови за замовчуванням. Для керування статусом використовується “BehaviorSubject”, що дозволяє відстежувати зміни статусу та реагувати на них у компонентах, які підписані на це сповіщення.

Метод “setStatus()” змінює поточний статус на протилежний оновлює статус у “BehaviorSubject”, що дозволяє усім підписаним компонентам отримати нове значення.

Метод “getLanguageText()” повертає відповідний набір текстів на основі поточного статусу. Якщо статус “true”, повертаються тексти англійською мовою, інакше - українською. Це дозволяє компонентам додатка отримувати тексти потрібною мовою одразу після її зміни.

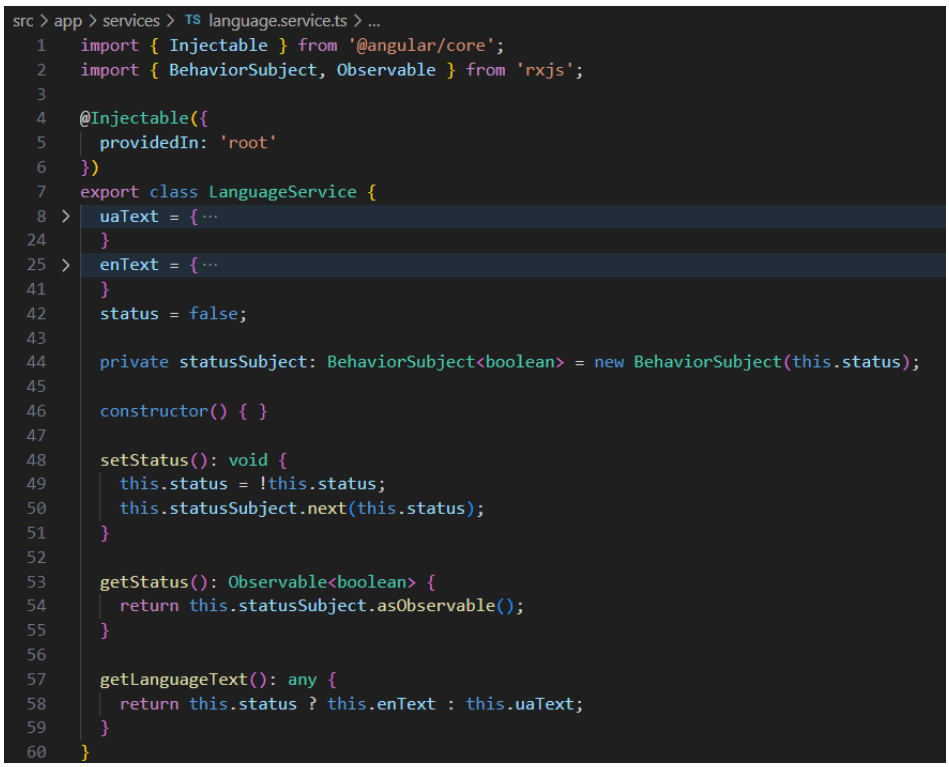


Рис. 2.24 Код сервісу мови

Сервіс кошика (cart.service.ts) (рис. 2.25) управляє станом кошика у додатку. Він містить інформацію про статус кошика, його вміст, а також кількість доданих елементів. Статус кошика (cartStatus) ініціалізується як “false” та вказує, чи повинен на вебсайті відображатися компонент кошика. Робить він це завдяки директиві “Angular” – “ngIf”. Вміст кошика зберігається у масиві “cartContent”. А кількість елементів у ньому зберігається у змінній “cartNumber”. Для керування статусом кошика також використовується “BehaviorSubject”, який зберігає поточний статус і дозволяє підписуватися на його зміни. Аналогічно управляється кількість елементів у кошику, що дозволяє компонентам відстежувати зміни при додаванні, чи видаленні елемента з нього.

Метод “setStatus(bool: boolean)” змінює статус кошика на передане значення й оновлює “BehaviorSubject”, щоб усі підписані компоненти отримали новий статус. А метод “getStatus()” повертає “Observable”, який можна використовувати для підписки на зміни статусу кошика.

Метод “setContent(product: Product)” додає або видаляє товар з кошика. Якщо товар з таким же ідентифікатором вже є в кошику, він буде видалений, і кількість

елементів оновиться. Якщо товару в кошику немає – він додається, і кількість елементів знову оновлюється. Метод “setContentEmpty()” очищає весь кошик і оновлює кількість елементів.

Метод “getContentLenght()” повертає “Observable”, який дозволяє підписуватися на зміни у кількості елементів у кошику. Метод “getContent()” повертає поточний вміст кошика у вигляді масиву товарів.

Метод “setAmount (i: number, operation: boolean)” дозволяє змінювати кількість певного товару в кошику, збільшуючи або зменшуючи її залежно від переданої логічної операції. Метод “getAmount(i: number)” повертає кількість певного товару в кошику.

```
export class CartService {
  cartStatus:boolean = false;
  cartContent:Product[] = [];
  cartNumber:number = 0;

  private cartStatusSubject: BehaviorSubject<boolean> = new BehaviorSubject(this.cartStatus);
  private cartNumberSubject: BehaviorSubject<number> = new BehaviorSubject(this.cartNumber);

  constructor() {}

  setStatus(bool:boolean): void {
    this.cartStatus = bool;
    this.cartStatusSubject.next(this.cartStatus);
  }
  getStatus(): Observable<boolean> {
    return this.cartStatusSubject.asObservable();
  }

  setContent(product:Product) {
    for (let i = 0; i < this.cartContent.length; i++) {
      if (this.cartContent[i].id == product.id) {
        this.cartContent[i].amount = 1;
        this.cartContent.splice(i,1);
        this.cartNumber = this.cartContent.length;
        this.cartNumberSubject.next(this.cartNumber);
        return;
      }
    }
    this.cartContent.push(product)
    this.cartNumber = this.cartContent.length;
    this.cartNumberSubject.next(this.cartNumber);
  }
  setContentEmpty() {
    this.cartContent = [];
    this.cartNumber = this.cartContent.length;
    this.cartNumberSubject.next(this.cartNumber);
  }
  getContentLenght(): Observable<number> {
    return this.cartNumberSubject.asObservable();
  }
  getContent() {
    return this.cartContent;
  }

  setAmount(i:number,operation:boolean) {
    if(operation) {
      this.cartContent[i].amount++;
    } else {
      this.cartContent[i].amount--;
    }
  }
  getAmount(i:number) {
    return this.cartContent[i].amount;
  }
}
```

Рис. 2.25 Код сервісу кошика

Сервіс сторінки товару (item-page.service.ts) (рис. 2.26) відповідає за управління станом та інформацією, пов'язаною зі сторінкою окремого товару в додатку. Він зберігає інформацію про поточний товар, його статус та статус коментарів.

Змінна “item” зберігає об'єкт типу “Product”, що містить дані про товар, котрі будуть задані у моделі товару (models/product.ts). Стан товару зберігається у змінній “status”, яка ініціалізується як “false” та вказує, чи товар активний для перегляду. Стан коментарів зберігається у змінній “commentStatus”, також ініціалізується як “false” та вказує, чи коментарі активні. Для керування цими станами використовуються “BehaviorSubject”, які зберігають поточні стани та дозволяють підписуватися на їх зміни.

Метод “setItem(product: Product)” встановлює новий товар, змінюючи значення змінної “item” на переданий об'єкт типу “Product”. При цьому статус товару змінюється на “true”, що вказує на те, що товар активний для перегляду. Значення “BehaviorSubject” для статусу також оновлюється, щоб усі підписані компоненти отримали новий статус.

Метод “getItem()” повертає поточний товар та встановлює статус на “false”, що вказує на те, що товар більше не активний для перегляду.

Метод “getStatus()” повертає “Observable”, який дозволяє підписуватися на зміни статусу товару, що дозволяє компонентам динамічно реагувати на ці зміни.

Метод “setCommentStatus(bool: boolean)” встановлює новий статус для компонента коментарів, змінюючи значення змінної “commentStatus” на передане значення. Значення “BehaviorSubject” для статусу коментарів також оновлюється, щоб усі підписані компоненти отримали новий статус.

Метод “getCommentStatus()” повертає “Observable”, який дозволяє підписуватися на зміни статусу коментарів, забезпечуючи динамічне оновлення інтерфейсу при зміні статусу коментарів.

```
export class ItemPageService {
  item:Product = {···
  };
  status:boolean = false;
  commentStatus:boolean = false;

  private statusSubject:BehaviorSubject<boolean> = new BehaviorSubject(this.status);
  private commentStatusSubject:BehaviorSubject<boolean> = new BehaviorSubject(this.commentStatus);

  description = '';
  comments = [
    {user:'',text:'',rate:0}
  ];

  constructor() { }

  setItem(product:Product) {
    this.item = product;
    this.status = true;
    this.statusSubject.next(this.status);
  }
  getItem() {
    this.status = false;
    return this.item;
  }

  getStatus():Observable<boolean> {
    return this.statusSubject.asObservable();
  }

  setCommentStatus(bool:boolean) {
    this.commentStatus = bool;
    this.commentStatusSubject.next(this.commentStatus);
  }
  getCommentStatus():Observable<boolean> {
    return this.commentStatusSubject.asObservable();
  }
}
```

Рис. 2.26 Код сервісу сторінки товару

Сервіс авторизації (login.service.ts) (рис. 2.27) відповідає за управління станом авторизації користувача в додатку.

Змінна “loginStatus” вказує на те, чи показувати користувачеві інтерфейс авторизації, тоді як змінна “isLogged” вказує, чи користувач авторизований.

Для керування станом обох авторизацій використовуються “BehaviorSubject” які зберігають поточні стани та дозволяють підписуватися на їх зміни.

Метод “getAll()” поки пропускаємо, оскільки він має робити запит до бази даних.

Метод “setStatus(bool: boolean)” встановлює новий статус компонента, змінюючи значення змінної “loginStatus” на передане значення й оновлюючи “BehaviorSubject”, щоб усі підписані компоненти отримали новий статус.

Метод “getStatusObs()” повертає “Observable”, який дозволяє підписуватися на зміни статусу компонента.

Метод “setLog(bool: boolean)” встановлює новий стан авторизації користувача, змінюючи значення змінної “isLogged” на передане й оновлюючи “BehaviorSubject”, щоб усі підписані компоненти отримали новий стан.

Метод “getLogObs()” повертає “Observable”, який дозволяє підписуватися на зміни стану авторизації користувача.

Метод “getLog()” повертає поточний стан авторизації користувача, дозволяючи компонентам перевіряти, чи користувач авторизований.

Метод “setUser(name: string, lastname: string)” встановлює особисті дані користувача, такі як ім'я та прізвище, і обчислює ініціали користувача, зберігаючи їх у змінній “userInitials”.

Метод “getUser()” приймає ідентифікатор користувача і повертає об'єкт користувача зі зразкових даних, якщо користувач з таким ідентифікатором існує.

Метод “getInitials()” повертає ініціали поточного користувача, що дозволяє відображати їх в інтерфейсі.

```
export class LoginService {
  private loginStatus:boolean = false;
  private isLogged:boolean = false;
  private userName:string = '';
  private userLastname:string = '';
  private userInitials:string = '';

  private loginStatusSubject: BehaviorSubject<boolean> = new BehaviorSubject(this.loginStatus);
  private isLoggedSubject: BehaviorSubject<boolean> = new BehaviorSubject(this.isLogged);

  constructor() { }

  getAll() {
    return SampleUsers;
  }

  setStatus(bool:boolean): void {
    this.loginStatus = bool;
    this.loginStatusSubject.next(this.loginStatus);
  }
  getStatusObs(): Observable<boolean> {
    return this.loginStatusSubject.asObservable();
  }

  setLog(bool:boolean): void {
    this.isLogged = bool;
    this.isLoggedSubject.next(this.isLogged);
  }
  getLogObs(): Observable<boolean> {
    return this.isLoggedSubject.asObservable();
  }
  getLog() {
    return this.isLogged;
  }

  setUser(name:string, lastname:string): void {
    this.userName = name;
    this.userLastname = lastname;
    this.userInitials = name[0].toUpperCase() + lastname[0].toUpperCase();
  }
  getUser(id:number) {
    let sampleUsers = this.getAll();
    for (const user of sampleUsers) {
      if (id = user.id) {
        return user;
      }
    }
    return;
  }

  getInitials() {
    return this.userInitials;
  }
}
```

Рис. 2.27 Код сервісу авторизації

Сервіс замовлення (order.service.ts) (рис. 2.28) відповідає за управління станом замовлення у додатку. Він зберігає інформацію про статус компонента та надає можливість іншим компонентам підписуватися на зміни цього статусу. Зберігається він у змінній “orderStatus”, яка ініціалізується як “false”, що вказує на те, що компонент не активний.

Для керування статусом компонента використовується “BehaviorSubject”. Ця реактивна змінна зберігає поточний стан і дозволяє компонентам підписуватися на його зміни.

Метод “setStatus(bool: boolean)” дозволяє змінювати статус компонента, приймаючи булеве значення. Він змінює значення змінної “orderStatus” на протилежне й оновлює “BehaviorSubject”, щоб усі підписані компоненти отримали новий статус.

Метод “getStatus()” повертає “Observable”, який дозволяє підписуватися на зміни статусу компонента. Це означає, що інші компоненти можуть динамічно реагувати на зміни стану, коли статус змінюється.

src > app > services > TS order.service.ts > ...

```
1  import { Injectable } from '@angular/core';
2  import { BehaviorSubject, Observable } from 'rxjs';
3
4  @Injectable({
5    providedIn: 'root'
6  })
7  export class OrderService {
8    orderStatus:boolean = false;
9
10   orderStatusSubject:BehaviorSubject<boolean> = new BehaviorSubject(this.orderStatus);
11
12   constructor() { }
13
14   setStatus(bool:boolean) {
15     this.orderStatus = bool;
16     this.orderStatusSubject.next(this.orderStatus);
17   }
18   getStatus():Observable<boolean> {
19     return this.orderStatusSubject.asObservable();
20   }
21 }
```

Рис. 2.28 Код сервісу замовлення

Сервіс магазину (shop.service.ts) (рис. 2.29) відповідає за управління станом магазину, включаючи налаштування сторінок, сортування та фільтрацію товарів. Він зберігає інформацію про поточний стан магазину та надає можливість іншим компонентам підписуватися на зміни цього стану.

Параметр “pageLimit” визначає максимальну кількість товарів, які можуть бути відображені на одній сторінці, і має значення “8” за замовчуванням. Параметр “pages” зберігає поточну кількість сторінок, починаючи з “1”.

Сортування товарів відбувається за двома критеріями: “sortItem” визначає предмет сортування товарів (за рейтингом, або ціною), а “sortFrom” визначає порядок сортування (за зростанням, або спаданням). “sortMaterial” зберігає вибір матеріалу для фільтрації товарів, і початково встановлюється на значення “all”, що означає відображення товарів з усіх матеріалів.

Стан магазину, який вказує, чи слід оновлювати відображення товарів, зберігається у змінній “shopStatus”. Для керування цим станом використовується “BehaviorSubject”, що зберігає поточний стан магазину і дозволяє підписуватися на його зміни.

Метод “setPages(bool: boolean)” приймає булеве значення, яке визначає, чи потрібно збільшити кількість сторінок. Якщо значення “true”, кількість сторінок збільшується на один, інакше встановлює значення “1”. Після зміни кількості сторінок викликається метод “setStatus()”, щоб оновити стан магазину.

Метод “getPages()” повертає поточну кількість сторінок.

Метод “getLimit()” повертає максимальну кількість товарів, які можуть бути відображені на одній сторінці.

Метод “setSort(item: boolean, from: boolean)” приймає два булевих значення, які визначають параметри сортування, і викликає метод “setStatus()” для оновлення стану магазину.

Метод “getSort()” повертає масив з двох значень, які представляють поточні параметри сортування.

Метод “setMaterial(material: string)” приймає рядкове значення, яке визначає матеріал для фільтрації товарів, і викликає метод “setStatus” для оновлення стану магазину.

Метод “getMaterial()” повертає поточний вибір матеріалу для фільтрації товарів.

Метод “setStatus()” оновлює “BehaviorSubject”, щоб усі підписані компоненти отримали новий стан магазину.

Метод “getStatus()” повертає “Observable”, який дозволяє підписуватися на зміни стану магазину, забезпечуючи динамічне оновлення інтерфейсу при зміні параметрів сортування, фільтрації або кількості сторінок.

```
export class ShopService {
  pageLimit:number = 8;
  pages:number = 1;
  sortItem:boolean = false;
  sortFrom:boolean = false;
  sortMaterial:string = 'all';
  shopStatus:boolean = false;
  shopStatusSubject:BehaviorSubject<boolean> = new BehaviorSubject(this.shopStatus);

  constructor() { }

  setPages(bool:boolean) {
    if (bool) {
      this.pages++;
    } else {
      this.pages = 1;
    }
    this.setStatus();
  }
  getPages() {
    return this.pages;
  }
  getLimit() {
    return this.pageLimit;
  }
  setSort(item:boolean, from:boolean) {
    this.sortItem = item;
    this.sortFrom = from;
    this.setStatus();
  }
  getSort() {
    return [this.sortItem, this.sortFrom];
  }
  setMaterial(material:string) {
    this.sortMaterial = material;
    this.setStatus();
  }
  getMaterial() {
    return this.sortMaterial;
  }
  setStatus() {
    this.shopStatusSubject.next(this.shopStatus);
  }
  getStatus():Observable<boolean> {
    return this.shopStatusSubject.asObservable();
  }
}
```

Рис. 2.29 Код сервісу авторизації

Останній сервіс – це сервіс товару (product.service.ts), але оскільки він працює з базою даних, його розгляд буде пізніше.

Імплементація технічної частини компонентів полягає у декількох аспектах, перший з яких – відправлення та отримання інформації з сервісів. Цей функціонал мають майже всі компоненти, тому розглянемо його на прикладі сервісу зміни мови.

Для відправки сигналу про зміну мови треба в компоненті верхнього колонтитула до перемикача мови додати атрибут “click”, котрий має викликати метод “onLangClick()”. Він, у свою чергу, визиває метод “setStatus()” мовного сервісу. Але щоб він спрацював, його спочатку треба імпортувати та створити значення в конструкторі (рис. 2.30).

```
import { CommonModule } from '@angular/common';
import { Component } from '@angular/core';
import { LanguageService } from '../../services/language.service';

@Component({
  selector: 'app-header',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './header.component.html'
})
export class HeaderComponent {

  constructor(private languageService: LanguageService) {}

  onLangClick() {
    this.languageService.setStatus();
  }
```

Рис. 2.30 Під’єднання та використання сервісу мови в коді

Тепер якщо користувач натисне на кнопку зміни мови – сервіс зчитає це та передасть нове значення далі. Залишилося додати зчитування цього нового значення у всі компоненти, котрі мають текст для перекладу. Оскільки цей процес однаковий для кожного з них, розглянемо лише один з компонентів. Нехай це буде компонент “Про нас”.

Для початку треба створити масив на таку кількість елементів типу рядок, скільки існую текстових елементів у компонента, далі замінити увесь текст на інтерполяцію, в котру передається змінна масиву з індексом, котрий на одиницю менший за відповідний текстовий рядок (якщо це другий рядок, то індекс -

одиниця). Наступний крок – імпортування та додавання до конструктора сервісу мови. Після цього треба, при створенні компонента (ngOnInit), підписатися на метод “getStatus()”, який є спостережувачем, та змінювати в ньому масив на новий, котрий повертається методом “getLanguageText()”. Але нам потрібен не весь об’єкт, а саме текстовий масив компонента, що робиться через крапку та ключ (рис. 2.31).

```
import { LanguageService } from '../../services/language.service';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-about-us',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './about-us.component.html'
})
export class AboutUsComponent {
  text = ['','','','','','','','','','','','','','','','','','','','','','',''];

  constructor(private languageService:LanguageService) {}

  ngOnInit():void {
    this.languageService.getStatus().subscribe(status => {
      this.text = this.languageService.getLanguageText().aboutUsText;
    })
  }
```

Рис. 2.31 Код технічної частини отримання тексту через сервіс мови

Наступною технічною частиною є анімації, які зроблені з використання директиви “ngClass” фреймворку “Angular”. Оскільки майже кожен компонент їх використовує – розглянемо функціонал на прикладі каруселі.

Для початку треба створити змінну “animated”, яка буде відповідати за те, чи перелистується в цей момент карусель, чи ні. Далі треба створити змінну “carouselNum”, котра буде статичною та відповідатиме за кількість слайдів до повторення циклу. Останньою змінною є масив з двома елементами, які матимуть значенням номера слайда.

Після цього маємо створити циклічний метод, котрий буде змінювати значення змінної “animated” на “true”, після чого запускатиме таймер (setTimeout) на шістсот мілісекунд після якого змінна стає назад “false”, а змінні масиву, які відповідають за номер слайда, збільшуються на один, або зануляються, якщо одна

з них дорівнює максимальній кількості слайдів мінус один. Після цього запускається ще один таймер, але вже на чотири секунди, після яких метод зациклюється (рис. 2.32).

Тепер можна додати стилі для анімації. Для цього треба переверстати компонент каруселі на два абсолютні компоненти, якщо змінна “animated” дорівнює “false”, перший елемент знаходиться по центру каруселі, а другий – за полем бачення. Коли змінна набуває значення “true”, обидва елементи набувають стилів, котрі переміщають перший елемент в протилежну сторону від другого, а другий – на місце першого зі швидкістю шістсот мілісекунд. Далі, за функціоналом методу, змінна “animated” стає “false”, а обидва елементи набувають нових стилів, які моменталь переміщають їх на свої місця (рис. 2.33).

Залишилося лише додати клас картинки, але через те, що він динамічний, робиться це через директиву “class” фреймворку “Angular”, який дорівнює методу “setCarouselImage(i: boolean)”, де значення “true” – це перший елемент, а “false” – другий. Цей метод повертає текст класу, котрий встановлено для картинки, з використанням номера слайда елемента (якщо номер слайда дорівнює одиниці, повертається клас “carousel-1”).

```
export class CarouselComponent {
  animated:boolean = false;
  carouselNum:number = 3;
  carousels = [0,1]

  carouselCycle() {
    this.animated = true;
    setTimeout(() => {
      this.animated = false;
      if (this.carousels[0] == this.carouselNum-1) {
        this.carousels[0] = 0;
        this.carousels[1]++;
      } else if (this.carousels[0] == 0) {
        this.carousels[0]++;
        this.carousels[1] = 0;
      } else {
        this.carousels[0]++;
        this.carousels[1]++;
      }
      setTimeout(() => {
      this.carouselCycle();
      }, 4000);
    }, 600);
  }
  setCarouselImage(i:boolean) {
    if (i) {
      return `carousel-${this.carousels[0]}`;
    } else {
      return `carousel-${this.carousels[1]}`;
    }
  }
}
```

Рис. 2.32 Код технічної частини каруселі

```
<div class="relative w-full h-[300px] rounded-b-[5px] overflow-hidden">
  <div [ngClass]="{
    'duration-[600ms]': animated,
    'duration-0': !animated,
    'left-[100%]': animated,
    'left-0': !animated,
  }" class="w-full h-[300px] absolute ease-in-out" [class]="setCarouselImage(0)"></div>
  <div [ngClass]="{
    'duration-[600ms]': animated,
    'duration-0': !animated,
    'right-0': animated,
    'right-[100%]': !animated,
  }" class="w-full h-[300px] absolute ease-in-out" [class]="setCarouselImage(1)"></div>
</div>
```

Рис. 2.33 Переверстана під анімації карусель

Іншою технічною частиною є поява та зникнення компонентів. Ця технологія використовує булева директива “ngIf” фреймворку “Angular”. Його використовують п’ять компонентів: “blank.component”, “order.component”, “comment.component”, “login.component” та “cart.component”. Але оскільки функціонує директива в кожному з них однаково, розглянемо лише один з них. Нехай це буде компонент “Коментар”.

Для початку треба створити булеву змінну всередині головного компонента додатка, яка буде відповідати за статус компонента коментаря. Щоб змінна стала “true”, треба натиснути на кнопку в компоненті сторінки товару. Оскільки це інший компонент, відстежити натискання кнопки можна лише через сервіс. У сервісу сторінки товару є потрібні методи, тож імпортуємо його, додаємо в конструктор та при створенні компонента (ngOnInit) підписуємося на метод “getCommentStatus()”. Тепер коли спостерігач буде помічати зміну змінної всередині сервісу, локальна булева змінна буде прирівнюватися до нового значення (рис. 2.34).

Далі визивається метод “setCommentStatus(bool: boolean)”, при натисканні кнопки всередині сторінки товару, та передає значення “true”, так само і для кнопки з хрестиком всередині компонента коментаря, але тут передається значення “false”. В обидва компоненти треба імпортувати сервіс сторінки товару та оголосити його в конструкторі.

Залишилося лише додати директиву “ngIf” до компонента коментаря та передати туди локальну булеву змінну (2.35).

```
        ShopItemComponent,
        CartComponent,
        LoginComponent,
        BlankComponent,
        CommentComponent,
        OrderComponent
    ]
})
export class AppComponent {
  commentStatus:boolean = false;

  constructor(private itemPageService:ItemPageService) {}

  ngOnInit():void {
    this.itemPageService.getCommentStatus().subscribe(status => {
      this.commentStatus = status;
    })
  }
}
```

Рис. 2.34 Підписка на сервіс та прирівняння до отриманого значення

```
<div class="w-full flex-col items-center">
  <app-blank *ngIf="commentStatus" class="relative"></app-blank>
  <app-comment *ngIf="commentStatus" class="relative"></app-comment>
  <app-header class="w-full"></app-header>
  <div class="w-[1440px] flex-col">
    <app-carousel class="select-none"></app-carousel>
    <div class="mt-[24px] select-none w-full">
      <app-menu></app-menu>
      <router-outlet class="ml-[32px]"></router-outlet>
    </div>
  </div>
</div>
```

Рис. 2.35 Використання директиви з локальною булевою змінною

Останньою технічною частиною є копіювання елементі. Для цього використовується директива “ngFor”, яка використовується для створення шаблонів шляхом ітерації по колекції (масиву або іншій ітеративній структурі) та

відображення її елементів у шаблоні. Розглянемо приклад використання у компоненті “Відділення”.

Для початку треба створити масив з шаблонами. Оскільки в кожному шаблоні має бути посилання, назва ТРЦ та номер телефону, змінною шаблону треба обрати об’єкт (рис. 2.36).

```
departments = [
  {href:'https://maps.app.goo.gl/gHTQg2t3TgBJsEKj6',name:'Respublika Park',phone:'+(380)67-555-4242'},
  {href:'https://maps.app.goo.gl/mLt3goWb65bERM1K7',name:'Blockbuster',phone:'+(380)67-555-5353'},
  {href:'https://maps.app.goo.gl/YbgYJNfqPzEExHsYA',name:'Lavina Mall',phone:'+(380)67-555-6464'},
  {href:'https://maps.app.goo.gl/1K3gWaBs9NhSTAM66',name:'Globus Mall',phone:'+(380)67-555-7575'}
]
```

Рис. 2.36 Масив з об’єктами-шаблонами

Далі треба видалити усі, окрім одного, елементи відділень, а єдине, що залишилось – переверстати (рис. 2.37):

1. Додати директорію “ngFor” до елемента відділення, з назвою шаблону “department” та індексом (номер створеного шаблона).

2. Замінити клас зі зображенням на директиву “class”, котра буде повертати клас з текстом “department-” та ітератором в кінці.

3. Додати посилання використавши директорію “href” та прирівнявши її до посилання шаблона (department.href).

4. Замінити тексти на інтерполяцію зі значенням імені шаблона (department. name), елемент масиву тексту для перекладу з індексом ітератора плюс один та телефону шаблона (department.phone)

```
<div *ngFor="let department of departments; index as i" class="h-[130px] w-full bg-primaryTechFaded rounded-[5px] mt-[12px]">
  <div class="w-[255px] h-full bg-center bg-cover bg-norepeat" [class]="'department-'+i"></div>
  <a [href]="department.href" class="h-full flex-col justify-center cursor-pointer ml-[18px]">
    <h3>{{ department.name }}</h3>
    <h6>{{ text[i+1] }}</h6>
    <h6>{{ department.phone }}</h6>
  </a>
</div>
```

Рис. 2.37 Переверстаний елемент відділення

3 НАЛАШТУВАННЯ ТА УПРАВЛІННЯ БАЗОЮ ДАНИХ

3.1 Імплементація бази даних у вебсайт

Для підключення бази даних до вебсайту, початку треба наповнити моделі товару та користувача, оскільки без строгої типізації нічого не скомпілюється, а дані не будуть надіслані.

Модель товару повинна мати десять обов’язкових полів:

1. Поле ідентифікатору товару.
2. Поле кількості товару.
3. Поле зображення товару.
4. Поле назви товару українською мовою.
5. Поле назви товару англійською мовою.
6. Поле опису товару українською мовою.
7. Поле опису товару англійською мовою.
8. Поле ціни товару.
9. Поле матеріалу товару.
10. Масив коментарів до товару типу об’єкт з іменем та прізвищем користувача, який залишив коментар, відгук та рейтинг до товару.

Модель користувача повинна мати п’ять обов’язкових полів:

1. Поле ідентифікатору користувача.
2. Поле імені користувача.
3. Поле прізвища користувача.
4. Поле електронної пошти користувача.
5. Поле паролю користувача.

Створивши моделі, з’являється можливість закінчити сервіс товару (product.service.ts), котрий відповідає за отримання, сортування та фільтрацію. Основні методи сервісу забезпечують взаємодію з базою даних, що містить товари та дані про них.

Метод “getAll()” робить запит на базу даних та повертає масив всіх доступних товарів. Це базовий метод, який використовується іншими методами сервісу для отримання початкового списку товарів.

Метод “getProduct(id:number)” приймає ідентифікатор товару як аргумент і повертає відповідний товар з отриманого методом “getAll()” масиву. Якщо товар з таким ідентифікатором не знайдено, метод повертає помилку.

Метод “getSortedProducts(sortMaterial: string, sortItem: boolean, sortFrom: boolean, pageLenght: number)” відповідає за сортування товарів для чого приймає чотири параметри: матеріал, за яким слід фільтрувати товари, параметри сортування (за ціною або рейтингом) і напрямок сортування (від найменшого до найбільшого, або навпаки). Він використовує інший допоміжний метод “getMaterialProducts(material: string)” для попереднього фільтрування товарів за матеріалом. Після цього він сортує товари за ціною або рейтингом, додаючи вже відсортовані до масиву “sortedProducts” поки не досягне заданої змінною “pageLenght” кількості.

Метод “getAvgRate(product: Product)” обчислює середній рейтинг товару на основі коментарів. Він проходить через кожен коментар переданої змінної, підсумовує їхні рейтинги та обчислює середнє значення. Якщо товар не має коментарів, метод повертає “0”.

Метод “getMaterialProducts(material: string)” фільтрує товари за матеріалом. Якщо текст заданої змінної не дорівнює "all", метод проходить через всі товари та додає до масиву тільки ті, що відповідають заданому матеріалу. Якщо він дорівнює "all", метод повертає всі товари без фільтрації.

Сервіс товару, в компоненті магазину, відіграє важливу роль у забезпеченні отримання та обробки даних про товари. Спочатку його потрібно імпортувати та оголосити у конструкторі компонента магазину. Далі, всередині конструктора, локальна змінна “products” прирівнюється до значення, яке повертає метод сервісу товару “getSortedProducts(sortMaterial: string, sortItem: boolean, sortFrom: boolean, pageLenght: number)”, з поточними налаштуваннями сортування та фільтрації, що

зберігаються у сервісі магазина. Після цього список відображається в інтерфейсі магазину завдяки директиві “ngFor”.

Наступною, при створенні компонента, додається підписка на метод сервісу магазину “getStatus()”, всередині якої змінна “products” так само прирівнюється до значення, яке повертає метод сервісу товару “getSortedProducts(sortMaterial: string, sortItem: boolean, sortFrom: boolean, pageLenght: number)”, але вже з новими налаштуваннями.

Статус сервісу товару оновлюється кожен раз, коли всередині компонента магазину натискається кнопка "Завантажити більше" (збільшення лімітів відображення товарів), або обираються нові параметри сортування. Також він оновлюється, коли в компоненті меню натискається кнопка з одним з п'яти матеріалом. Це дозволяє компоненту динамічно оновлювати відображений список товарів відповідно до нових критеріїв сортування.

3.2 Встановлення та конфігурація MongoDB на сервері

Для встановлення та конфігурації “MongoDB” на сервері, а також під'єднання до сервісів товарів та авторизації, слід виконати декілька ключових кроків.

Для початку потрібно запустити сервер “MongoDB” та налаштувати його для автоматичного запуску при завантаженні системи. Конфігурація сервера може включати зміну налаштувань файлу конфігурації для забезпечення необхідного рівня безпеки та продуктивності. Це може включати налаштування шляхів до даних, портів, а також параметрів мережевих з'єднань.

Наступним кроком є створення бази даних користувачів та товарів. У “MongoDB” потрібно створити окремі бази даних для обох моделей з відповідними правами доступу. Це забезпечить безпеку даних та ізоляцію між сервісами.

Після цього необхідно налаштувати підключення з бекенд-сервісів до “MongoDB”. У кожному з сервісів: товарів та авторизації, слід додати залежність на драйвер “MongoDB” для мови програмування, яку використовують ці сервіси. Потім у їх коді необхідно налаштувати підключення до “MongoDB”, вказавши посилання з'єднання, базу даних, ім'я користувача та пароль.

Далі необхідно доповнити логіку сервісів товарів та авторизації для взаємодії з “MongoDB”. Це включає написання запитів для зберігання, оновлення, видалення та отримання даних з “MongoDB”. Для сервісу товарів це може включати запити для отримання інформації про всі товари, а для сервісу авторизації - запити для перевірки облікових даних користувачів та створення нових користувачів.

ВИСНОВКИ

У даній кваліфікаційній роботі було розглянуто процес розробки дизайну та реалізації вебсайту для перегляду та покупки речей ручної роботи онлайн. Для цього були використані сучасні технології, такі як “Figma” для прототипування та дизайну, “Angular” для фронтенд-розробки, та “MongoDB” для управління даними.

В ході роботи було проведено аналіз ринку аналогічних продуктів, що дозволило визначити ключові функції та вимоги до вебсайту. Це допомогло створити чітке технічне завдання та план проєкту.

За допомогою “Figma” були створені прототипи основних сторінок та компонентів сайту, орієнтовані на забезпечення зручності використання, привабливого вигляду та інтуїтивно зрозумілого інтерфейсу. Використовуючи “Angular”, було реалізовано інтерактивний інтерфейс користувача з особливою увагою до оптимізації продуктивності та забезпечення адаптивності дизайну для декількох типів моніторів. Для зберігання даних про товари та користувачів була розроблена та впроваджена база даних на основі “MongoDB”, що забезпечило надійне та ефективне управління даними.

Всі компоненти вебсайту були інтегровані в єдину систему, після чого проведено ретельне тестування. Виявлені помилки були виправлені, а функціональність підтверджена шляхом ручного та автоматичного тестування.

Таким чином, у рамках цієї роботи було створено повнофункціональний вебсайт, який відповідає сучасним стандартам веброзробки. Проєкт продемонстрував ефективність використання разом “Figma”, “Angular”, “Tailwind CSS” та “MongoDB” у створенні складних вебзастосунків, і цей досвід може бути корисним для подальших досліджень та розробок у галузі вебтехнологій.

ПЕРЕЛІК ПОСИЛАНЬ

1. Figma [Електронний ресурс] / Figma - Режим доступу: https://www.figma.com
2. Tailwind CSS documentation [Електронний ресурс] / Tailwind CSS - Режим доступу: https://tailwindcss.com/docs/installation
3. Type Compatibility [Електронний ресурс] / TypeScript - Режим доступу: https://www.typescriptlang.org/docs/handbook/type-compatibility.html
4. Angular documentation [Електронний ресурс] / Angular - Режим доступу: https://angular.dev/overview
5. Angular routing system [Електронний ресурс] / Medium - Режим доступу: https://medium.com/@jstify.community/angular-routing-d25992afe8c7
6. Angular Directives: Enhancing User Interfaces with Ease [Електронний ресурс] / Medium - Режим доступу: https://medium.com/@ayushgrwl365/angular-directives-enhancing-user-interfaces-with-ease-bb99d74e69cd
7. Download NodeJS [Електронний ресурс] / NodeJS - Режим доступу: https://nodejs.org
8. Node.js v22.2.0 documentation [Електронний ресурс] / NodeJS - Режим доступу: https://nodejs.org/docs/latest/api/module.html
9. MongoDB [Електронний ресурс] / MongoDB - Режим доступу: https://www.mongodb.com/
10. MongoDB documentation [Електронний ресурс] / MongoDB - Режим доступу: https://www.mongodb.com/docs/
11. MongoDB Початок навчання [Електронний ресурс] / W3schoolsUA - Режим доступу: https://w3schoolsua.github.io/mongodb/mongodb_get_started.html#gsc.tab=0

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Презентація

до кваліфікаційної роботи на здобуття освітнього ступеня бакалавра

НА ТЕМУ: **«РОЗРОБКА ДИЗАЙНУ ТА РЕАЛІЗАЦІЯ ВЕБСАЙТУ ДЛЯ ПЕРЕГЛЯДУ ТА ПОКУПКИ РЕЧЕЙ РУЧНОЇ РОБОТИ ОНЛАЙН З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ, ТАКИХ ЯК FIGMA, ANGULAR ТА MONGODB»**

Виконав: студент Лісецький І. Ю.

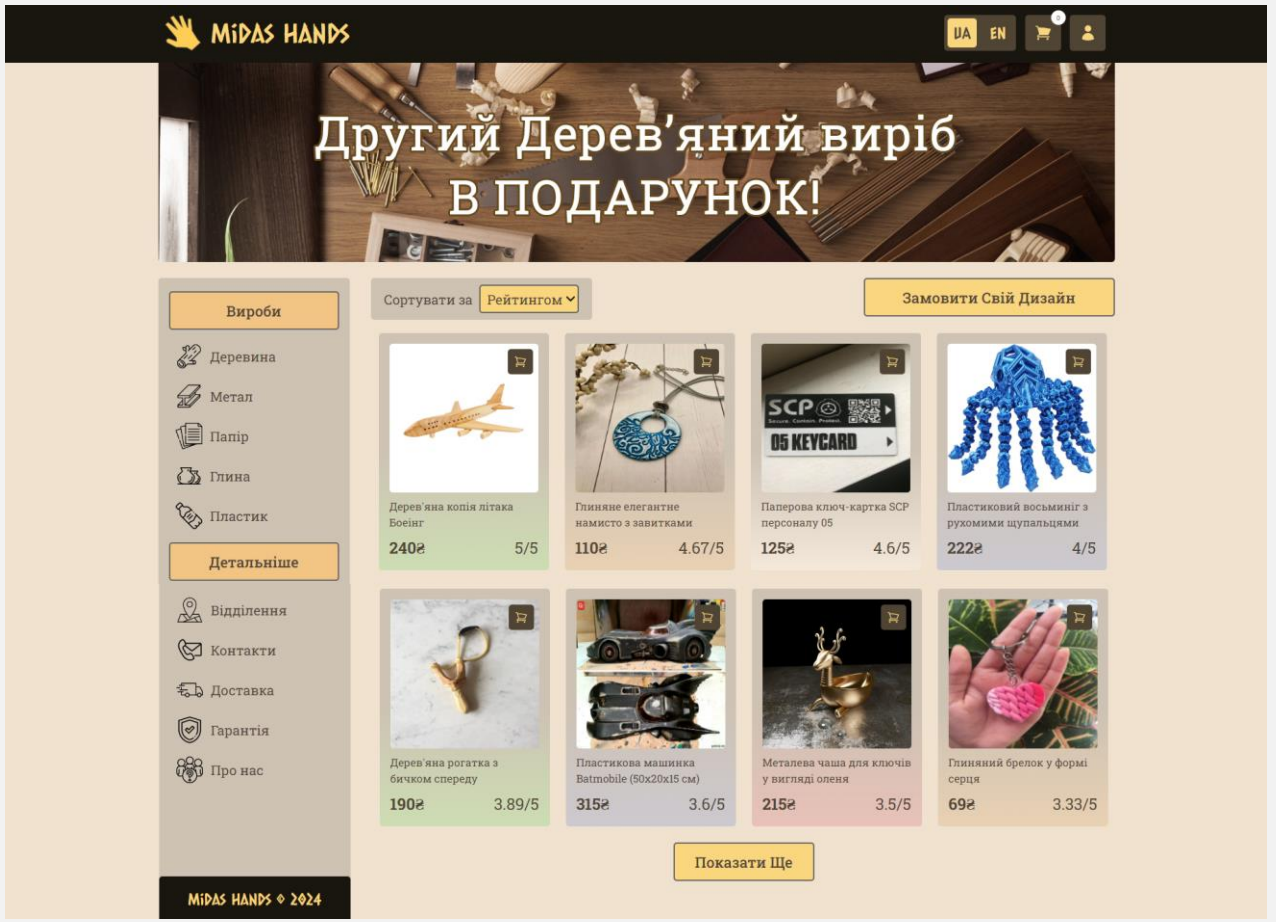
Керівник: Шикула О. М.

ПЛАН

1. Мета роботи
2. Об’єкт роботи та предмет досліждення
3. Вибір технологій
4. Що таке Figma?
5. Використання Figma у кваліфікаційній роботі
6. Що таке Angular?
7. Роутингова система
8. Сервіси
9. Моделі для бази даних
10. Що таке MongoDB?
11. Використання MongoDB у кваліфікаційній роботі
12. Висновок

МЕТА РОБОТИ

Розробка функціонального вебсайту для перегляду та покупки речей ручної роботи онлайн, використовуючи технології Figma, Angular та MongoDB, а також аналіз їх ефективності та взаємодії у рамках веброзробки.



ОБ’ЄКТ РОБОТИ

Процес розробки вебсайтів для електронної комерції включає в себе повний цикл створення:
- **Планування**: Визначення цільової аудиторії, формулювання вимог до функціоналу сайту.
- **Проектування**: Розробка архітектури сайту, створення макетів та прототипів.
- **Розробка**: Включає як фронтенд, так і бекенд розробку.
- **Тестування**: Забезпечення якості шляхом виявлення та виправлення помилок.

ПРЕДМЕТ ДОСЛІЖДЕННЯ

Методи та засоби, які використовуються для розробки вебсайтів, зокрема дизайн (Figma), фронтенд-розробка (Angular) та управління базами даних (MongoDB), є сучасними компонентами у створенні ефективних платформ для електронної комерції.

ВИБІР ТЕХНОЛОГІЙ

Figma

- Всі файли зберігаються в хмарі, тому немає потреби турбуватися про резервне копіювання або збереження змін.

- Працює на будь-якій операційній системі, що має браузер, що робить його універсальним інструментом для дизайнерів на різних платформах.

- Вбудовані інструменти для створення інтерактивних прототипів та анімацій дозволяють легко тестувати ідеї без необхідності використання додаткових програм.

Angular

- Повний набір інструментів для розробки великих і складних додатків, включаючи рендерінг на стороні сервера, двосторонній зв'язок даних, та управління станом.

- Angular написаний на TypeScript, що додає статичну типізацію, покращену інтеграцію IDE і кращу підтримку великого коду.

- Модульна архітектура дозволяє розбивати додатки на логічно відокремлені частини, що спрощує розробку, тестування та підтримку.

MongoDB

- Підтримка горизонтального масштабування (sharding), що дозволяє розподіляти дані на кілька серверів і забезпечувати високу продуктивність при великих обсягах даних.

- Оптимізована для швидкого зчитування та запису, MongoDB добре підходить для додатків, які вимагають високої продуктивності.

- Вбудована підтримка реплікації забезпечує високу доступність даних та відмовостійкість.

ЩО ТАКЕ FIGMA?

це хмарний інструмент для дизайну та прототипування інтерфейсів користувача. Відзначається своєю функціональністю, яка дозволяє декільком користувачам працювати над одним проектом в режимі реального часу. Деякі з ключових особливостей Figma включають:

- **Колаборація в режимі реального часу**: Можливість працювати разом з колегами над одним проектом без необхідності відправляти файли один одному.

- **Кросплатформеність**: Оскільки це хмарний інструмент, Figma працює на будь-якому пристрої з доступом до Інтернету.

- **Прототипування**: Інструменти для створення інтерактивних прототипів, що дозволяє тестувати та демонструвати функціональність дизайну.

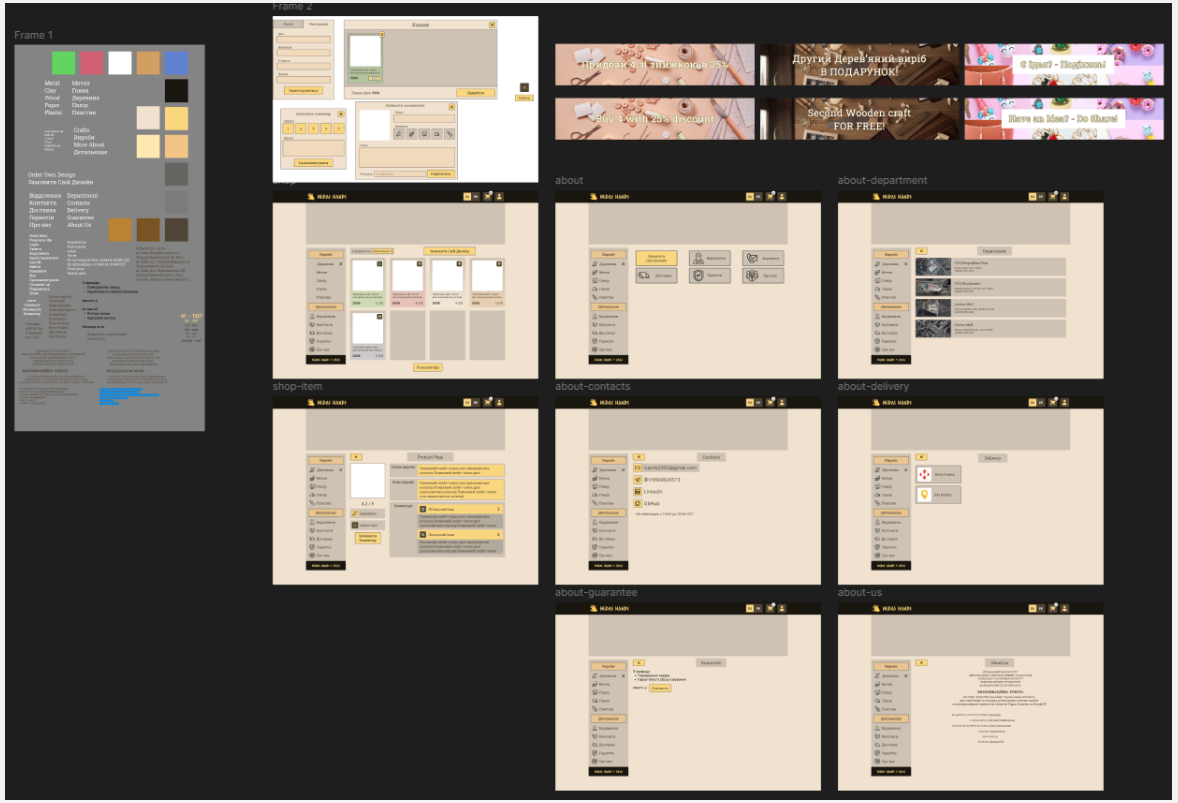
- **Компоненти та стилі**: Можливість створювати та використовувати повторювані елементи (компоненти), а також налаштовувати стилі для швидкої зміни зовнішнього вигляду інтерфейсу.

ВИКОРИСТАННЯ FIGMA У КВАЛІФІКАЦІЙНІЙ РОБОТІ

У проекті були застосовані наступні шрифти:
- Caesar Dressing (для тексту без перекладу)
- Roboto Slab (для усього іншого тексту)

Для проекту була визначена кольорова палітра:
- #504535 (текст)
- #F0E2CF (фон)
- #FAD67F (основний)
- #666561 (основний технічний)
- #F3C585 (вторинний)
- #191610 (темний)
- 5 кольорів, кожен для свого матеріалу

А також був розроблений дизайн для кожної сторінки та кожного компоненту для кожного технічного випадку.



ЩО ТАКЕ ANGULAR?

це фреймворк для розробки веб-додатків на мові TypeScript, розроблений та підтримуваний компанією Google. Angular надає розробникам набір інструментів для створення динамічних та адаптивних веб-додатків. Основні характеристики включають:

- **Компонентний підхід**: Використання компонентів для модульної розробки, що сприяє кращій організації коду та повторному використанню.

- **Двостороння прив'язка даних (Two-way data binding)**: Механізм автоматичного синхронізування даних між моделлю та представленням, що спрощує роботу з інтерфейсом.

- **Dependency Injection**: Система впровадження залежностей, яка спрощує тестування та управління залежностями.

- **Routing**: Вбудовані можливості для створення маршрутизації в односторінкових додатках (SPA), що забезпечує швидку та ефективну навігацію.

РОУТИНГОВА СИСТЕМА

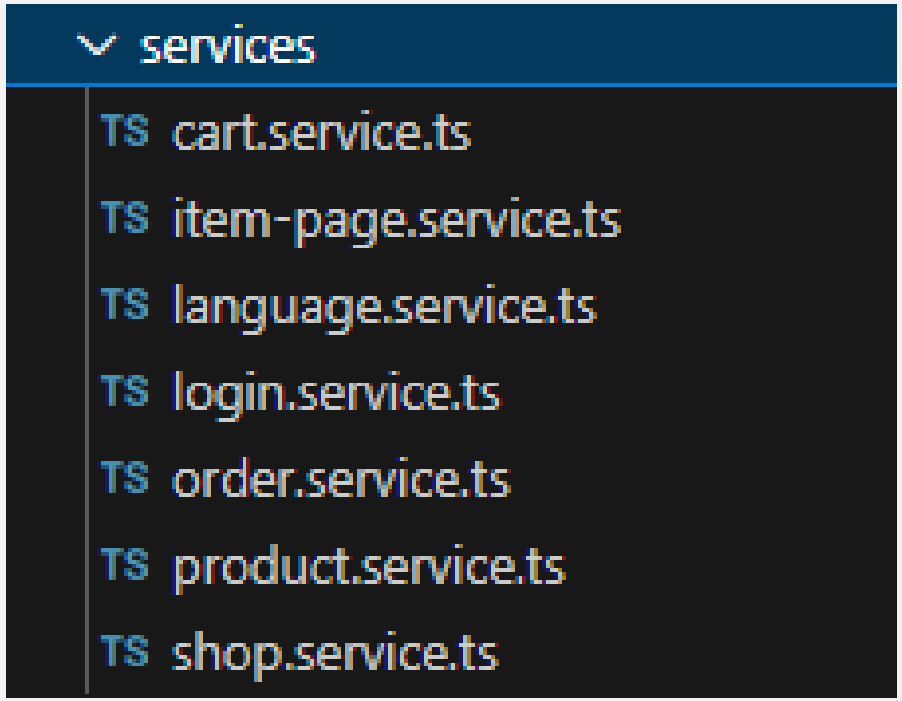
У проекті за допомогою фреймворку **Angular** була створена роутингова система, для створення багатосторінкового сайту:
- / (основна сторінка / сторінка з товарами)
- /item-page (схема сторінки товару)
- /about (сторінка меню для переходу на наступні 5 сторінок)
- /about/contacts (сторінка з контактними даними)
- /about/delivery (сторінка з сервісами доставки)
- /about/department (сторінка з місцями розташування відділів)
- /about/guarantee (сторінка з інформацією щодо повернення та гарантійного обслуговування)
- /about/us (сторінка з інформацією про проект)

```
import { Routes } from '@angular/router';
import { ShopComponent } from './components/shop/shop.component';
import { AboutComponent } from './components/about/about.component';
import { AboutContactsComponent } from './components/about-contacts/about-contacts.component';
import { AboutDeliveryComponent } from './components/about-delivery/about-delivery.component';
import { AboutDepartmentComponent } from './components/about-department/about-department.component';
import { AboutGuaranteeComponent } from './components/about-guarantee/about-guarantee.component';
import { AboutUsComponent } from './components/about-us/about-us.component';
import { ShopItemPageComponent } from './components/shop-item-page/shop-item-page.component';

export const routes: Routes = [
  { path: '', component: ShopComponent},
  { path: 'item-page', component: ShopItemPageComponent},
  { path: 'about', component: AboutComponent},
  { path: 'about/contacts', component: AboutContactsComponent},
  { path: 'about/delivery', component: AboutDeliveryComponent},
  { path: 'about/department', component: AboutDepartmentComponent},
  { path: 'about/guarantee', component: AboutGuaranteeComponent},
  { path: 'about/us', component: AboutUsComponent},
];
```

СЕРВІСИ

У проекті були використані сервіси для передачі та збереження даних між компонентами.
- cart.service:
Потрібен для додавання, видалення та перевірки товарів кошика.
- item-page.service:
Оскільки сторінка для товарів одна, а товарів багато, ми створюємо сервіс для отримання даних о товарі, який має бути зображений.
- language.service:
Потрібен для збереження тексту на обох мовах, а також передачі цього тексту у кожен компонент після зміни мови.
- login.service:
Цей сервіс відповідає за реєстрацію та перевірки юзера при вході в акаунт.
- order.service:
Відповідає лише за появу/зникнення компоненту для замовлення.
- product.service:
Відповідає за отримання товарів з бази даних та їх сортування.
- shop.service:
Отримує та віддає дані, які встановлюються у головній сторінці.



МОДЕЛІ ДЛЯ БАЗИ ДАНИХ

Для інтеграції бази даних та вебсайту були зроблені моделі отримуваємої інформації:

- **Product:**
Модель товару з усією інформацією, що має бути зображена у компонентах: **shop.component**, **item-page.component** та **cart.component**. Єдине значення, яке не зображається – це ідентифікатор товару, який потрібен для простішого діставання та передачі товарів між ціма компонентами.

- **User:**
Модуль користувача, який зберігає усю потрібну інформацію для перевірки входу в акаунт та реєстрації.

```
export class Product {
  amount!:number;
  id!:number;
  image!:string;
  titleUA!:string;
  titleEN!:string;
  descriptionUA!:string;
  descriptionEN!:string;
  price!:number;
  material!:string;
  comments!:Array<{ name: string, lastname: string, comment: string, rate: number }>;
}
```

```
export class User {
  id!:number;
  name!: string;
  lastname!: string;
  email!: string;
  password!: string;
}
```

ЩО ТАКЕ MONGODB?

це документно-орієнтована база даних NoSQL, яка використовує JSON-подібні документи для зберігання даних. Основні особливості MongoDB включають:

- **Гнучка структура даних**: Можливість зберігати документи з різними структурами в одній колекції, що дозволяє легко адаптуватися до змін в даних.

- **Масштабованість**: Підтримка горизонтального масштабування (шардування), що дозволяє обробляти великі обсяги даних і високі навантаження.

- **Висока продуктивність**: Оптимізована для швидкого читання і запису даних, що робить її придатною для використання в реальних часоподібних додатках.

- **Індексація**: Підтримка різних типів індексів для підвищення продуктивності запитів.

ВИКОРИСТАННЯ MONGODB У КВАЛІФІКАЦІЙНІЙ РОБОТІ

1. Спочатку був встановлений і налаштований MongoDB сервер. Для локальної розробки був запущений на стандартному порту 3000.

2. У серверній частині на Node.js з використанням Express було налаштовано підключення до бази даних myapp через лінк mongodb://localhost:3000/bachelor-project за допомогою бібліотеки mongoose.

3. Далі були створені схеми для моделей User та Product.

4. Для забезпечення взаємодії між клієнтською частиною на Angular та базою даних MongoDB були розроблені RESTful API. Для моделі User був створений маршрут /users, а для моделі Product – маршрут /products.

5. За допомогою Express були реалізовані обробники запитів. Для маршруту GET /users обробник отримував всіх користувачів з бази даних і повертав їх у форматі JSON. Для маршруту POST /users обробник отримував дані нового користувача з запиту, зберігав їх у базі. Аналогічно був реалізован обробник для маршруту GET /products для отримання всіх продуктів.

ВИСНОВОК

У результаті проведеної роботи було досягнуто основної мети - розробка сучасного вебсайту з використанням новітніх технологій. Вибір об'єкта дослідження та предмету роботи був обґрунтований необхідністю створення зручного, функціонального та надійного веб-додатку. Аналіз доступних технологій призвів до вибору Figma, Angular та MongoDB, як основних інструментів для реалізації проекту. Figma дозволила створити інтерактивний дизайн інтерфейсу користувача, що значно спростило процес верстки. Angular забезпечив високу продуктивність та модульність фронтенду, тоді як MongoDB надала гнучкість і масштабованість у роботі з базою даних. Інтеграція цих технологій у кваліфікаційну роботу дозволила досягти високої якості кінцевого продукту, який відповідає сучасним вимогам і стандартам розробки вебсайтів.