МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ	Економіки та бізнес-освіти
Кафедра	Економіки та цифрового бізнесу
Спеціальність	122 Комп'ютерні науки
Форма навчання	Заочна

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Горбаль Євгенії Олександрівни
(прізвище, ім'я, по батькові здобувача)

на тему Розробка інтерактивної веб-платформи для замовлення адаптивного одягу для поранених військових
(повна назва теми)

за матеріалами. _____
(повна назва бази дослідження)

науковий керівник _____ Шокотько Л.М
(наук. ступінь, вчене звання) *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 12.06.2026 р. № 15

Завідувач кафедри _____
(підпис)
к.е.н., доцент Радько В.М.
наук. ступень, вчене звання прізвище, ініціали

Кривий Ріг − 2026

ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки, молоді та спорту України
29 березня 2012 року № 384

Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ ТА БІЗНЕС-ОСВІТИ
(повне найменування вищого навчального закладу)

Кафедра економіки та цифрового бізнесу
Освітній ступінь бакалавр
Спеціальність 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри _____В.М. Радько

"30" березня 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ ЗДОБУВАЧУ

_____Горбаль Євгенії Олександрівні_____

1. Тема роботи ___Розробка інтерактивної вебплатформи для замовлення адаптивного одягу ___для поранених військових_____

науковий керівник роботи _____Шокотько Людмила Миколаївна_____,
затверджені наказом вищого навчального закладу від «23» березня 2026 р. № 194-ст

2. Строк подання здобувачем роботи 29.05.2026р.

3. Зміст кваліфікаційної бакалаврської роботи, об'єкт, предмет та мета дослідження:

Розділ 1 ___Теоретичні аспекти розробки веб-застосунку для замовлення адаптивного одягу
Розділ 2 ___Проєктування онлайн-платформи для замовлення адаптивного одягу_____
Розділ 3 ___Програмна реалізація проєкту_____

Об'єкт дослідження – _процес розробки інтерактивної веб-платформи для замовлення_____ адаптивного одягу для поранених військових_____

Предмет дослідження - __методи та засоби проєктування і програмної реалізації_____
клієнт-серверного веб-застосунку з функціоналом каталогу товарів, системи автентифікації____ користувачів та оформлення замовлень_____

Мета кваліфікаційної бакалаврської роботи – __розробка інтерактивної веб-платформи, яка забезпечує зручне онлайн-замовлення адаптивного одягу для поранених військових, шляхом реалізації повного циклу створення веб-застосунку: від проєктування інформаційної архітектури та UI/UX-дизайну до програмної реалізації клієнтської і серверної частин із підключенням хмарної бази даних_____

4. Дата видачі завдання 03.04.2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	до 17.04.2026р.	16.04.2026р.
2	Підготовка розділу 2	до 08.05.2026р.	08.05.2026р.
3	Підготовка розділу 3	до 25.05.2026р.	25.05.2026р.
4	Реєстрація завершеної кваліфікаційної роботи	до 29.05.2026р.	29.05.2026р.
5	Отримання відгуку від наукового керівника	04.06.2026р.	04.06.2026р.
6	Отримання зовнішньої рецензії	05.06.2026р.	05.06.2026р.
7	Попередній захист кваліфікаційної роботи на кафедрі	08.06.2026р.	08.06.2026р.
8	Перевірка кваліфікаційної роботи на плагіат	18.06.2026р.	18.06.2026р.
9	Допуск кафедрою кваліфікаційної роботи до захисту	18.06.2026р.	18.06.2026р.
10	Підготовка студента до захисту в ЕК	до 23.06.2026р	23.06.2026р

Завдання підготував науковий керівник _____ Шокотько Л. М.

Завдання одержав здобувач _____ Горбаль Є.О.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 61 сторінка, 25 малюнків, 2 таблиці, 26 використаних джерел, 5 додатки.

Об'єкт дослідження: процес розробки інтерактивної вебплатформи для замовлення адаптивного одягу для поранених військових.

Мета: розробка інтерактивної веб-платформи для замовлення адаптивного одягу для поранених військових шляхом реалізації повного циклу створення веб-застосунку: від проєктування архітектури та UI/UX-дизайну до програмної реалізації клієнтської і серверної частин із підключенням хмарної бази даних.

У результаті дослідження було проаналізовано сучасні веб-технології та існуючі волонтерські ініціативи у сфері забезпечення військових адаптивним одягом. Спроєктовано інформаційну модель та загальну архітектуру застосунку, розроблено UI/UX-дизайн і візуальні макети. Реалізовано серверну частину на основі Node.js та Express із підключенням хмарної бази даних MongoDB Atlas, створено систему реєстрації та автентифікації користувачів на базі JWT. Розроблено клієнтську частину з каталогом товарів, функціоналом вибору кольору і розміру, кошиком замовлень та особистим кабінетом користувача.

Результати роботи можуть бути використані для створення реального онлайн-магазину адаптивного одягу для поранених військових, а також як основа для розробки інших спеціалізованих веб-платформ із системою авторизації, каталогом товарів та оформленням замовлень.

Ключові слова: адаптивний одяг, база даних, вебзастосунок, військові, замовлення, каталог товарів, клієнт-серверна архітектура, особистий кабінет, реабілітація, BACKEND, EXPRESS, FRONTEND, HTML/CSS, JWT, MONGODB, NODE.JS, REST API

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – база даних

ЗСУ – Збройні Сили України

API (Application Programming Interface) – інтерфейс програмування застосунків, що дозволяє взаємодію між різними програмними компонентами.

CSS (Cascading Style Sheets) – мова опису стилів для оформлення вебсторінок.

DB (Database) – база даних, структуроване сховище інформації.

HTML (HyperText Markup Language) – мова розмітки гіпертекстових документів.

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту.

JSX (JavaScript XML) − JavaScript XML

JWT (JSON Web Token) − JSON Web Token.

MySQL – система управління реляційними базами даних.

NoSQL (Not Only SQL) − не лише SQL.

REST (Representational State Transfer) − передача репрезентативного стану.

SQL (Structured Query Language) – мова запитів до реляційних баз даних.

UI (User Interface) – інтерфейс користувача.

UX (User Experience) – користувацький досвід.

URL (Uniform Resource Locator) – уніфікований покажчик ресурсу.

ВСТУП

З початку повномасштабного вторгнення в Україну у 2022 році значно зросла кількість військовослужбовців, які отримують поранення та потребують тривалого лікування й реабілітації. Одним із практичних викликів у цей період є забезпечення пацієнтів зручним одягом, який можна одягати без зайвих рухів при наявності гіпсу або інших обмежень. Стандартний одяг у таких випадках завдає болю або потребує сторонньої допомоги, а спеціалізованого вітчизняного виробника адаптивного одягу для військових з функцією онлайн-замовлення на сьогодні практично немає.

Вирішення цієї проблеми потребує не лише виробництва відповідного одягу, а й зручного способу його замовлення. Веб-платформа з каталогом, можливістю вибору розміру та кольору, особистим кабінетом і кошиком дозволяє спростити цей процес як для покупця, так і для виробника. Саме тому розробка такого застосунку є актуальною і практично значущою задачею.

Мета і завдання дослідження. Метою кваліфікаційної бакалаврської роботи є розробка інтерактивної веб-платформи для замовлення адаптивного одягу для поранених військових шляхом реалізації повного циклу створення веб-застосунку: від проєктування архітектури та UI/UX-дизайну до програмної реалізації клієнтської і серверної частин із підключенням хмарної бази даних.

Для досягнення мети були поставлені такі завдання:

- проаналізувати існуючі волонтерські ініціативи та IT-рішення у сфері забезпечення військових адаптивним одягом;
- обгрунтувати вибір стека технологій для розробки платформи;
- спроєктувати інформаційну модель та загальну архітектуру застосунку;
- розробити UI/UX-дизайн та реалізувати клієнтську частину веб-застосунку;
- реалізувати серверну частину з REST API та підключити хмарну базу даних MongoDB;
- протестувати розроблений веб-сайт.

Об'єкт дослідження. Процес розробки інтерактивної веб-платформи для замовлення адаптивного одягу для поранених військових.

Предмет дослідження. Методи та засоби проєктування і програмної реалізації клієнт-серверного веб-застосунку з функціоналом каталогу товарів, системи автентифікації користувачів та оформлення замовлень у контексті забезпечення поранених військових адаптивним одягом.

Методи дослідження. У роботі використано методи аналізу та порівняння при огляді існуючих рішень, метод структурного проєктування при побудові архітектури застосунку, а також методи об'єктно-орієнтованого програмування при реалізації серверної та клієнтської частин.

Практичне значення одержаних результатів. Розроблена веб-платформа може бути використана як реальний інструмент для замовлення адаптивного одягу для поранених військових. Застосунок реалізує повний цикл роботи з користувачем: перегляд каталогу, вибір товару за розміром та кольором, оформлення замовлення через кошик, реєстрацію та авторизацію з особистим кабінетом.

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ЗАМОВЛЕННЯ АДАПТИВНОГО ОДЯГУ

1.1. Огляд існуючих волонтерських ініціатив та ІТ-рішень для забезпечення військових одягом

Повномасштабне вторгнення стало каталізатором виникнення широкого кола волонтерських ініціатив, здатних оперативно реагувати на потреби армії та суспільства. Серед актуальних запитів особливого значення набуло питання забезпечення поранених військовослужбовців спеціалізованим одягом [15].

Адаптивний одяг – це вироби на липучках, кнопках або з бічними блискавками замість традиційних застібок. Такий одяг є необхідним для поранених, оскільки стандартна форма при отриманні поранення зрізається медичним персоналом і більше не придатна до використання, а пацієнт без спеціального одягу вимушений терпіти незручності. Усвідомлення цієї потреби стало поштовхом до розвитку окремого напряму волонтерського та підприємницького виробництва.

Однією з перших і найвідоміших волонтерських ініціатив стала «Швейна рота». З погляду використання цифрових технологій вона від початку спиралася на доступні онлайн-інструменти: координація діяльності здійснювалася через чат-групи та соціальні мережі, прийом індивідуальних замовлень від поранених організовано за допомогою онлайн-форми, а сторінки в Instagram та Facebook забезпечують інформаційну підтримку та залучення нових волонтерів [2].

Водночас така модель має суттєві обмеження. Дані про потреби користувачів, наявність виробів та виконання замовлень розосереджені між різними каналами комунікації, а саме месенджерами та соціальними мережами. Відсутність єдиної інформаційної системи ускладнює накопичення статистичних даних, проведення аналітики та прогнозування виробничих потреб у довгостроковій перспективі**.** Цей приклад наочно демонструє, що ефективна

волонтерська ініціатива без спеціалізованої платформи стикається із системними труднощами масштабування та обліку.

Більш розвинений рівень цифровізації демонструє бренд «PLUS PLUS» – один із найбільш комплексних прикладів спеціалізованого e-commerce-рішення у сфері адаптивного одягу для поранених військових. Платформа функціонує як повноцінний інтернет-магазин з розвиненою структурою каталогу, де товари згруповані за категоріями. Для кожної позиції представлено декілька фотографій, опис матеріалів і конструктивних особливостей виробу та розмірну сітку. (рис. 1.1).

Рис. 1.1. Сторінка адаптивного одягу для поранених військових бренду «PLUS PLUS»

Примітка. Джерело: Розроблено з використанням [20]

Процес оформлення замовлення реалізовано відповідно до поширених практик української електронної комерції та включає додавання товарів до кошика, введення даних отримувача, вибір способу доставки та здійснення оплати [19].

Основні переваги PLUS PLUS:

– спеціалізований каталог – товари систематизовані за типом виробу та статтю, з окремим розділом для поранених військових;

– детальні картки товарів – кожна позиція містить фотографії, опис матеріалів, розмірну сітку та лічильник покупок;
– зручне оформлення замовлення – стандартний e-commerce-флоу: кошик, дані доставки, оплата онлайн або при отриманні.

Недоліки PLUS PLUS:

Інтерфейс сайту: елементи головного меню мають недостатню площу активної області; картки товарів перевантажені дрібними візуальними елементами (палітри кольорів, позначки кількості, перекреслені ціни, бейджі знижок), що ускладнює сприйняття інформації; відсутній чітко виражений елемент швидкої дії, що збільшує кількість кроків до оформлення замовлення.

Інтернет-магазин «Adaptive» (рис. 1.2) є прикладом спеціалізованої e-commerce-платформи у сфері реабілітаційного одягу. Сайт має стандартну структуру інтернет-магазину: каталог товарів, деталізовані картки продукції з описом крою, матеріалів та рекомендаціями щодо підбору розміру, кошик і механізм оформлення замовлення.

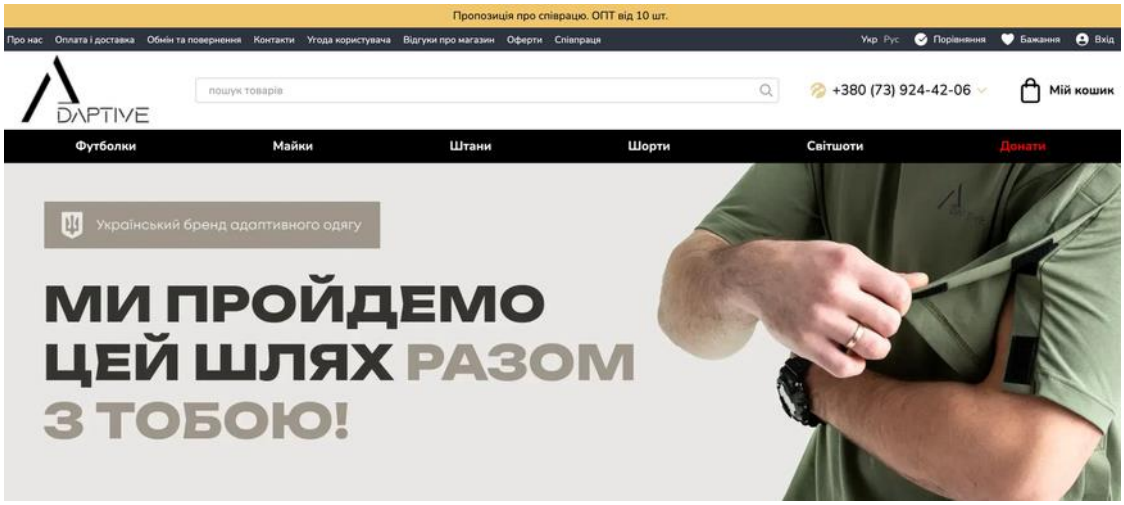

Рис. 1.2. Головна сторінка інтернет-магазину ADAPTIVE

Примітка. Джерело: Розроблено з використанням []

На головній сторінці розміщено банер із позиціонуванням бренду, навігаційне меню з категоріями (футболки, майки, штани, шорти, світшоти) та розділ «Хіти продажу» (рис. 1.3).

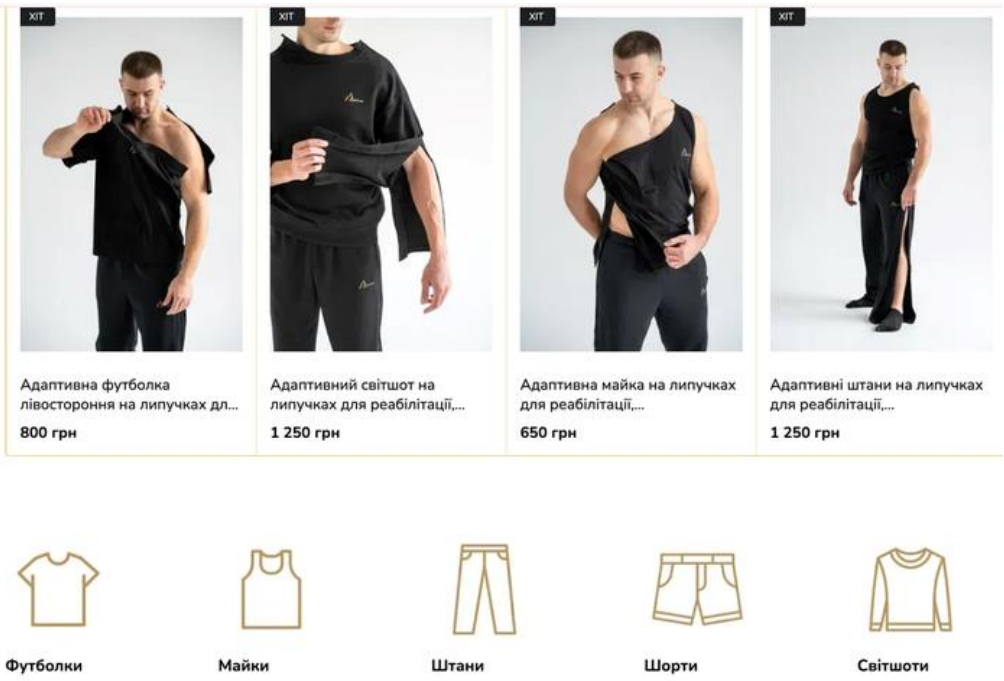

Рис. 1.3. Каталог товарів інтернет-магазину ADAPTIVE

Примітка. Джерело: Розроблено з використанням [7]

Платформа підтримує кілька варіантів оплати та забезпечує доставку по всій Україні. Окремою функціональною особливістю є наявність консультаційного сервісу, який передбачає фаховий супровід вибору продукції з урахуванням індивідуальних потреб користувача [7]. Це розширює модель взаємодії з клієнтом порівняно з іншими рішеннями.

Переваги Adaptive:

– консультаційний сервіс – фахівець допомагає підібрати виріб з урахуванням характеру поранення та типу фіксуючих пристроїв;
– повноцінний інтернет-магазин – має категорії товарів, детальні картки та кошик.

Недоліки Adaptive:

– з погляду UI/UX-дизайну сайт має певні недоліки. Використання дрібного шрифту та надмірна кількість категорій ускладнюють навігацію, що може негативно впливати на зручність взаємодії користувачів із ресурсом;

– немає особистого кабінету з відстеженням замовлень – обмежені можливості для повторних покупок та управління замовленнями.

Значно простіший підхід до цифрового представлення демонструє бренд Wheelny – перший в Україні бренд адаптивного одягу для людей на кріслах колісних. З точки зору ІТ-рішення платформа реалізована на основі конструктора односторінкових сайтів. Структура включає короткий опис продукції, фотогалерею виробів із зазначенням розмірів та матеріалів, а також контактну інформацію. Процес оформлення замовлення не є автоматизованим: користувач змушений звертатися через месенджери, що робить процес тривалим та незручним [25].

Огляд та аналіз наявних рішень показав, що в Україні платформ для забезпечення військових адаптивним одягом небагато, і більшість із них не є повноцінними інтернет-магазинами. Результати порівняльного аналізу свідчать, що:

– частина проєктів, зокрема «Швейна рота», приймає замовлення через соціальні мережі та месенджери без єдиної системи обліку;
– окремі платформи, як Wheelny, мають лише простий сайт-візитку без можливості оформлення замовлення онлайн.

PLUS PLUS та Adaptive мають повноцінний інтернет-магазин, який забезпечує користувачам можливість ознайомлення з асортиментом продукції, оформлення замовлень та здійснення покупок без необхідності додаткової комунікації з продавцем, однак їхні інтерфейси містять низку недоліків, що може ускладнювати взаємодію користувачів із платформою.

Таким чином, більшість наявних рішень або обмежені за функціоналом, або не відповідають специфічним потребам поранених військових. Це обґрунтовує доцільність розробки окремої спеціалізованої веб-платформи для замовлення адаптивного одягу для військовослужбовців із зручним каталогом, особистим кабінетом, обліком замовлень та адаптованим інтерфейсом.

1.2. Обґрунтування доцільності розробки єдиної інтерактивної вебплатформи та постановка задачі

Будь-який програмний продукт має вирішувати конкретну проблему, яка є актуальною та значущою. Платформа для замовлення адаптивного одягу для поранених військових відповідає цій вимозі, оскільки пов'язана з нагальними потребами реабілітації. За даними Міністерства охорони здоров'я, в умовах повномасштабної війни кількість людей, які потребують реабілітації, постійно зростає, а їх повернення до активного життя є одним із пріоритетів держави [4].

У грудні 2024 року Міністерство оборони України офіційно анонсувало розробку адаптивного одягу за запитом поранених військовослужбовців і медичного персоналу, щоб військові почувалися якомога зручніше під час лікування та реабілітації. Вже у листопаді 2025 року Міноборони передало до шпиталів 50 тисяч одиниць адаптивного одягу, а до кінця того ж року планувалося довести це число до 450 тисяч [3].

Ці цифри є надзвичайно важливими в контексті обґрунтування платформи. Якщо щорічна потреба вимірюється сотнями тисяч одиниць і охоплює десятки шпиталів по всій країні, обробляти такий потік замовлень через розрізнені телефонні дзвінки, форми у Viber та особисті домовленості з волонтерами є незручним та неефективним.

Спеціалізована платформа може запропонувати користувачеві більш зрозумілий інтерфейс, вузько налаштований під одну задачу, і знизити кількість кроків від вибору виробу до підтвердження замовлення.

Таким чином, постає нагальна потреба у розробці єдиної інтерактивної веб-платформи, яка централізує процес замовлення адаптивного одягу та дозволяє військовим або медичному персоналу оперативно підбирати вироби під конкретні потреби пацієнта. Метою розробки є створення веб-застосунку типу інтернет-магазину, орієнтованого на цю аудиторію.

Відповідно до мети, розроблена платформа ADAPT має реалізувати такий функціонал:

- каталог товарів – структурований перелік адаптивного одягу з розподілом за категоріями (футболки, майки, штани, шорти, світшоти), статтю та розміром, з детальними картками кожного виробу;
- сторінки товарів – детальний опис кожного виробу з фотографіями у різних ракурсах, характеристиками матеріалів, розмірною сіткою та вибором кольору;
- кошик – накопичення обраних товарів з можливістю зміни кількості, видалення позицій та перегляду підсумкової суми замовлення;
- оформлення замовлення – введення даних отримувача, вибір способу та адреси доставки, підтвердження й оплата замовлення онлайн або при отриманні;
- особистий кабінет – реєстрація та авторизація користувача з можливістю перегляду історії та статусу замовлень, збереження даних профілю;
- система авторизації – захищений вхід через JWT-токени з хешуванням паролів для забезпечення безпеки даних користувача.

Створення такого цифрового інструменту дозволить перетворити забезпечення адаптивним одягом на системну сервісну послугу, де кожен захисник зможе отримати необхідну підтримку вчасно та у повному обсязі.

1.3. Визначення стека технологій для розробки онлайн-платформи для замовлення адаптивного одягу для військових

Вебплатформа є різновидом програмного забезпечення, що функціонує через мережу Інтернет і доступне користувачеві за допомогою браузера без необхідності встановлення додаткових програм. [23]. Від вебсайту вона відрізняється насамперед характером взаємодії з користувачем: вебсайт виконує переважно інформаційну функцію, а веб-платформа передбачає активну участь користувача, наприклад: реєстрацію, авторизацію, виконання операцій із даними та персоналізований доступ залежно від ролі в системі [17].

Питання вибору стека технологій постає на самому початку проєктування будь-якої сучасної веб-платформи і фактично задає вектор усієї подальшої розробки. Від правильності вибору залежить наскільки гнучкою, стійкою та безпечною буде система впродовж усіх етапів роботи. У разі платформи для замовлення адаптивного одягу для військових ці міркування набувають особливої ваги, адже система має одночасно забезпечувати швидку обробку даних, зрозумілий інтерфейс для користувачів та надійний захист персональної інформації.

Технологічний стек традиційно розглядають як двокомпонентну екосистему, що охоплює фронтенд – клієнтську сторону, з якою безпосередньо взаємодіє користувач, і бекенд – серверну сторону, яка обробляє запити та забезпечує бізнес-логіку. Обидві частини разом з архітектурою бази даних утворюють каркас застосунку, що визначає його функціональні можливості та технічні обмеження [1].

Вибір технологічного стека завжди залежить від специфіки проєкту та поставлених завдань. Кожна предметна область висуває власний набір вимог, тому технології, ефективні для одного типу веб-ресурсів, можуть бути недостатньо придатними для іншого. З огляду на функціональні особливості платформи для замовлення адаптивного одягу, доцільним є використання простих, добре документованих і перевірених технологій, які забезпечують реалізацію реєстрації користувачів, автентифікації, обробки замовлень та взаємодії з базою даних. Для створення клієнтської частини веб-застосунку використовуються три основні технології: HTML, CSS і JavaScript.

HTML (HyperText Markup Language) – це стандартна мова розмітки для створення структури та змісту веб-сторінок. Вона визначає, які елементи присутні на сторінці, зокрема заголовки, абзаци, зображення, таблиці, форми, кнопки та гіперпосилання. Мова базується на системі тегів, за допомогою яких формуються елементи вебдокумента.

CSS3 (Cascading Style Sheets, рівень 3) є мовою таблиць стилів, яка відповідає за візуальне оформлення HTML-елементів на веб-сторінці. Вона

визначає зовнішній вигляд кожного елемента, включаючи кольори, шрифти, розміри, відступи, рамки, тіні та його розташування відносно інших елементів.

JavaScript – повноцінна мова програмування, що виконується безпосередньо в браузері користувача та надає веб-сторінкам динамічність: реагування на дії користувача, зміну вмісту без перезавантаження сторінки, відправку запитів на сервер та обробку отриманих даних.

Для забезпечення серверної логіки, зберігання даних та захисту інформації користувачів використано окремий набір технологій:

Node.js – це середовище виконання JavaScript на стороні сервера, яке дозволяє будувати серверні застосунки на тій самій мові, що й клієнтська частина. Express.js – фреймворк для Node.js, що спрощує організацію HTTP-сервера та маршрутизацію запитів.

MongoDB – документоорієнтована база даних, що зберігає дані у форматі, подібному до JSON, замість традиційних таблиць. Mongoose – бібліотека для Node.js, що надає зручний інтерфейс для роботи з MongoDB та дозволяє описувати структуру документів у вигляді схем.

bcryptjs – бібліотека для хешування паролів, що реалізує алгоритм bcrypt, розроблений спеціально для захисту облікових даних. jsonwebtoken – бібліотека для роботи з технологією JSON Web Token (JWT), яка є стандартним способом передачі інформації про автентифікацію між клієнтом і сервером у вигляді підписаного токена. Пакет cors забезпечує коректну взаємодію між клієнтською та серверною частинами, що працюють на різних портах.

Для реалізації проєкту як середовище розробки було обрано Visual Studio Code – безкоштовний редактор коду від Microsoft. Він підтримує різні мови програмування, забезпечує автодоповнення коду, перевірку синтаксису та інструменти налагодження, які дозволяють виконувати код покроково та відстежувати значення змінних безпосередньо в середовищі розробки.

Висновки до розділу 1

Проведений аналіз дозволив сформувати цілісну картину проблемного середовища та технологічної бази, необхідної для реалізації веб-платформи для замовлення адаптивного одягу для військових.

Огляд наявних волонтерських ініціатив та IT-рішень виявив суттєву нерозвиненість вітчизняного ринку цифрових сервісів у сфері забезпечення поранених військових адаптивним одягом. Встановлено, що існуючі рішення розпадаються на дві групи.

До першої належать ініціативи без власної платформи, зокрема «Швейна рота» та Wheelny, які здійснюють приймання замовлень через месенджери та соціальні мережі, що унеможливлює системний облік і автоматизацію.

До другої групи входять повноцінні інтернет-магазини PLUS PLUS та Adaptive, які, однак, мають виражені недоліки з погляду UX: перевантажені картки товарів, дрібні активні зони та відсутність особистого кабінету з відстеженням замовлень. Жодне з розглянутих рішень не реалізує повний цикл взаємодії з покупцем у зручній та спеціалізованій формі. Ця закономірність обґрунтовує практичну доцільність розробки окремої платформи.

Аналіз стека технологій підтвердив відповідність обраного набору інструментів вимогам проєкту. Поєднання HTML5, CSS3 та JavaScript на рівні клієнта із Node.js та Express.js на рівні сервера забезпечує уніфікацію мови програмування в межах усього застосунку, а MongoDB Atlas надає гнучкість схеми та хмарну масштабованість. Бібліотеки bcryptjs та jsonwebtoken реалізують захист облікових даних відповідно до сучасних стандартів.

Постановка задачі визначила шість функціональних вимог до платформи: структурований каталог товарів з розподілом за статтю та розміром, детальні картки продукції з вибором кольору та фотографіями, кошик замовлень, форма оформлення замовлення з введенням адреси доставки та вибором способу оплати, особистий кабінет із переглядом статусу замовлень, а також система авторизації на основі JWT-токенів. Визначений функціональний обсяг став вичерпною технічною специфікацією для подальшого проєктування системи.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ОНЛАЙН-ПЛАТФОРМИ ДЛЯ ДЛЯ ЗАМОВЛЕННЯ АДАПТИВНОГО ОДЯГУ

2.1. Побудова інформаційної моделі та загальної архітектури застосунку

Перед тим як перейти до безпосередньої реалізації вебзастосунку, необхідно сформувати чітке уявлення про його структуру, тобто визначити, які сутності функціонують у системі, як вони пов'язані між собою та яким чином організовано їх взаємодію на різних рівнях програмного забезпечення.

Інформаційна модель системи описує предметну область через сукупність сутностей та відносин між ними. В контексті даного проєкту виокремлюються п'ять ключових сутностей, кожна з яких відповідає реальному об'єкту предметної галузі й має власний набір атрибутів, що описують її характеристики [11]:

- користувач (User) – центральна сутність системи, що представляє фізичну особу, яка взаємодіє з платформою. Атрибути: унікальний ідентифікатор, ім'я, електронна адреса, захищений хеш пароля, номер телефону, роль у системі (адміністратор або клієнт), дата реєстрації;
- категорія (Category) – структурний елемент каталогу, що використовується для групування товарів за спільними ознаками;
- товар (Product) – одиниця асортименту магазину. Зберігає назву, опис, ціну, посилання на зображення;
- замовлення (Order) – фіксує факт покупки: містить посилання на користувача, підсумкову суму, адресу доставки, статус замовлення (нове, підтверджено, відправлено, доставлено, скасовано) та часові мітки створення й оновлення запису:
- позиція замовлення (Order Item). – сполучна сутність між замовленням і товаром. Дозволяє одному замовленню містити кілька різних позицій; зберігає кількість одиниць і ціну товару на момент покупки [14].

Зв'язки між сутностями організовано таким чином: один користувач може мати кілька замовлень; одне замовлення містить кілька позицій; один товар може входити до багатьох позицій замовлень; одна категорія включає кілька товарів. Такий підхід відповідає принципам нормалізації реляційних баз даних і дозволяє уникнути надлишкового зберігання даних (рис. 2.1) [11].

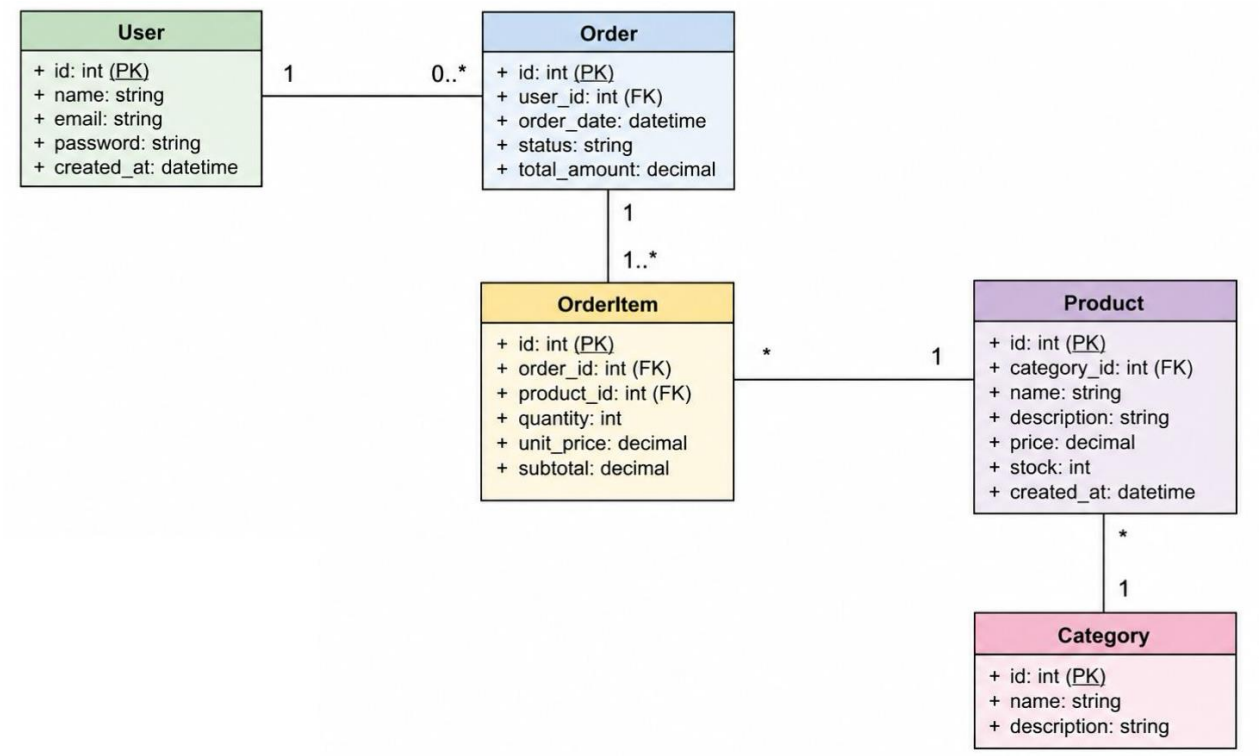

Рис 2.1. UML-діаграма класів системи інтернет-магазину

(розроблено автором)

З погляду архітектурної організації, систему побудовано за класичною трирівневою моделлю, яка є стандартним підходом у веб-розробці й забезпечує чітке розмежування відповідальності між компонентами [13].

Перший рівень – клієнтська частина. Це те, що безпосередньо бачить і з чим взаємодіє кінцевий користувач. Реалізовано засобами HTML, CSS (з адаптивною верстkою через Flexbox та Grid) і JavaScript. На цьому рівні відбувається рендеринг інтерфейсу, валідація форм на стороні клієнта та відправка HTTP-запитів до серверної частини.

Другий рівень – серверна частина. Відповідає за бізнес-логіку застосунку: обробку вхідних запитів, перевірку авторизації, формування відповідей.

Третій рівень – рівень даних. Реалізовано на основі реляційної СУБД MySQL. На цьому рівні зберігаються всі структуровані дані системи;

Така трирівнева організація суттєво спрощує масштабування та підтримку системи, у разі потреби кожен рівень може бути змінений незалежно від інших.

2.2. Моделювання алгоритмів взаємодії користувача з платформою. Проектування інтерфейсу

Ефективний інтерфейс веб-платформи – це злагоджений механізм, кожна деталь якого підпорядкована єдиній меті: провести користувача від першого знайомства з товаром до успішно оформленого замовлення з мінімальними когнітивними витратами. Для цього необхідно чітко визначити алгоритм взаємодії, що відбуваються під час відвідування сайту [18].

Загальний користувацький сценарій взаємодії з інтернет-магазином «АДАПТ» можна описати таким чином. На першому етапі відвідувач потрапляє на головну сторінку (рис. 2.2.), де його зустрічає головний екран з описом концепції бренду та CTA-кнопка «Обрати модель». Фільтрація здійснюється за категоріями (чоловічий/жіночий одяг), що винесено до верхньої навігаційної панелі. Цей перший екран є важливим, оскільки він повинен миттєво донести ціннісну пропозицію та спонукати до подальшого перегляду.

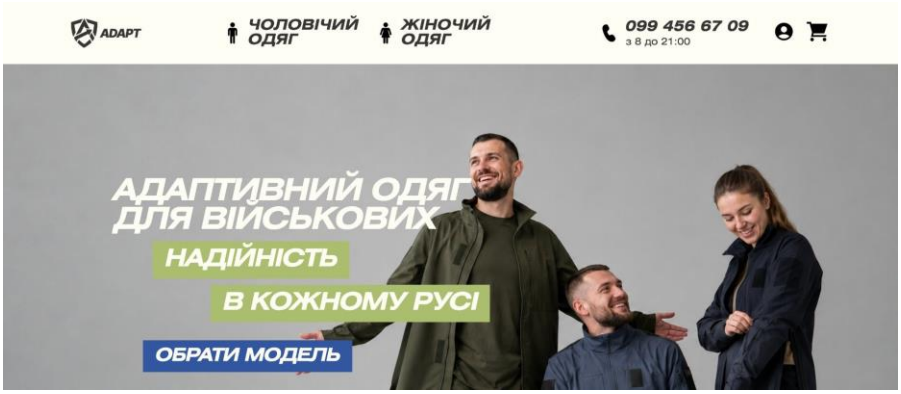

Рис. 2.2. Головна сторінка вебплатформи АДАПТ
(розроблено автором)

Другий етап: перегляд каталогу (рис. 2.3). Продукти відображаються у вигляді карток із фотографією, назвою та ціною. Картки організовано у сітку, що дозволяє одночасно бачити кілька позицій і зручно порівнювати їх.

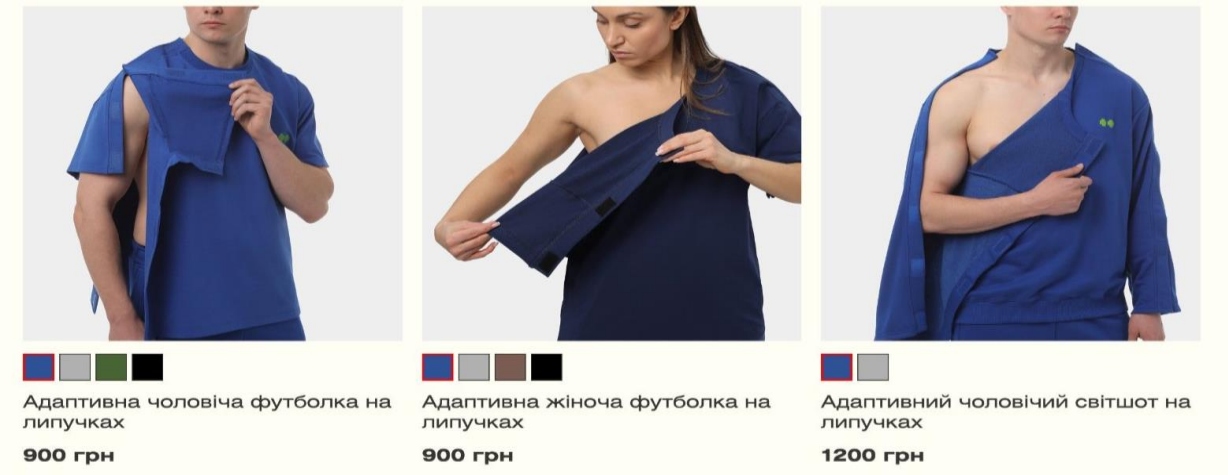

Рис. 2.3. Каталог товарів інтернет-магазину АДАПТ

Примітка. Джерело: розроблено автором з використанням [20]

Третій етап: вибір конкретного товару та перехід на сторінку детального опису. Тут користувач бачить розгорнуту інформацію про виріб, можливість вибору розміру та кнопку «Купити», що додає товар до кошику (рис. 2.4).

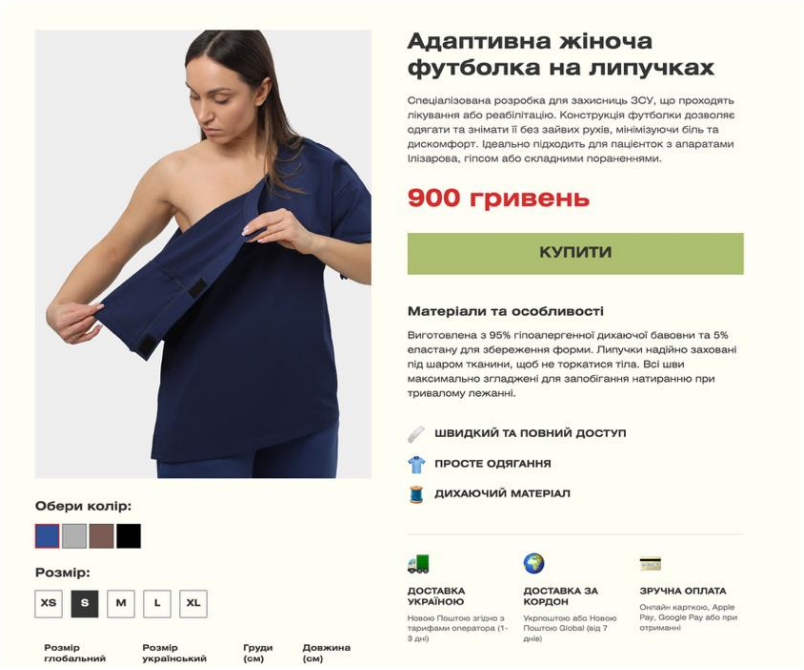

Рис 2.4. Сторінка вибору товару

Примітка. Джерело: розроблено автором з використанням [20]

Четвертий етап: перевірка авторизації. Якщо користувач не авторизований, система пропонує увійти або зареєструватися перед тим, як зберегти товар у кошику (рис. 2.5).

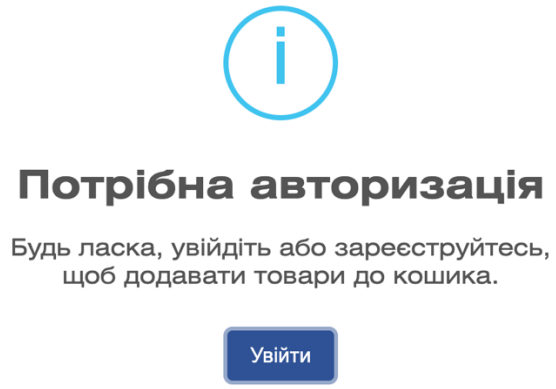

Рис 2.5. Модальне вікно повідомлення про авторизацію

(розроблено автором)

П'ятий і шостий етапи: формування кошика та оформлення замовлення. Після вибору всіх необхідних товарів користувач переходить до кошика, де бачить перелік позицій, їхні кількості та загальну суму (рис. 2.6).

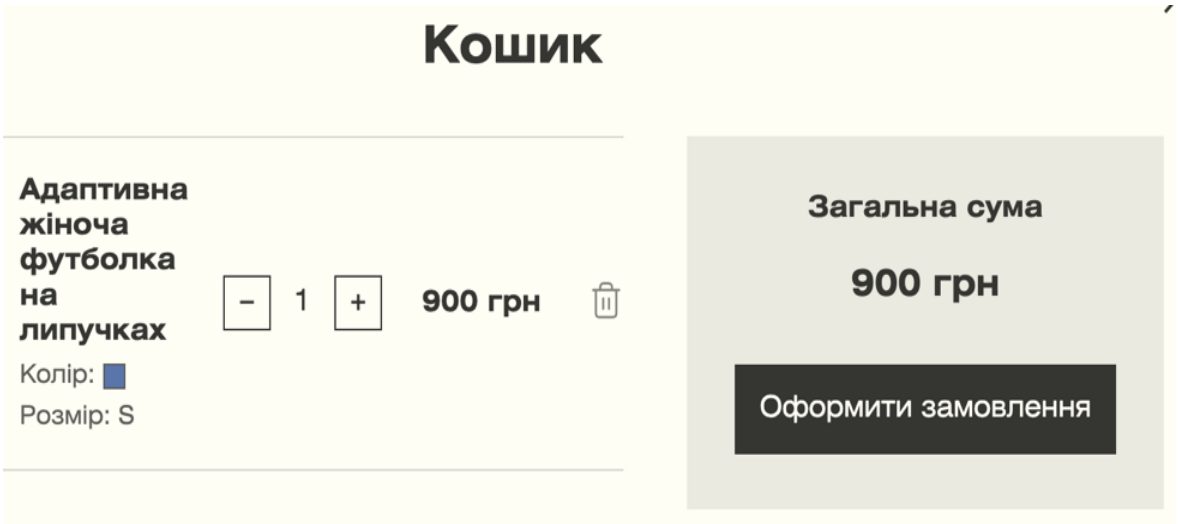

Рис 2.6. Модальне вікно кошика

(розроблено автором)

Натиснувши «Оформити замовлення», він потрапляє до форми введення адреси доставки та вибору способу оплати. Сьомий етап: підтвердження

замовлення. Після успішної оплати система надсилає email і перенаправляє до особистого кабінету для відстеження статусу (рис. 2.7).

Рис. 2.7. Блок-схема алгоритму взаємодії користувача з системою

(розроблено автором)

Проєктування інтерфейсу платформи підпорядковане загальній ідеї максимальної простоти взаємодії для цільової аудиторії – людей у лікувальних установах або їхніх рідних, які потребують швидкого оформлення замовлення. Такий підхід забезпечує узгоджений досвід незалежно від того, з якої сторінки розпочалась навігація [16].

2.3. Створення UI/UX дизайну та візуальних макетів веб-застосунку

Перш ніж переходити до опису конкретних дизайн-рішень, варто визначити саму суть понять UI та UX.

- UI (User Interface, користувацький інтерфейс) – це сукупність візуальних елементів, через які користувач взаємодіє із застосунком: кнопки, поля введення, меню, іконки, кольорова палітра, типографіка та компонування елементів на сторінці.
- UX (User Experience, досвід користувача) – це загальне враження та зручність, які отримує людина під час використання продукту: наскільки легко знайти потрібну інформацію, виконати дію чи завершити процес (наприклад, оформити замовлення).

UI/UX-дизайн – це комплексний процес проєктування інтерфейсу, спрямований на створення продукту, який є водночас естетично привабливим, інтуїтивно зрозумілим та ефективним у використанні [19].

Проєктування інтерфейсу веб-платформи «АДАПТ» спирається на низку загальновизнаних принципів UI/UX дизайну, які забезпечують зручність, передбачуваність та ефективність взаємодії користувача з системою [18]:

- принцип видимості стану системи передбачає, що користувач завжди має розуміти, що відбувається в інтерфейсі. У проєкті це реалізовано через підсвічування активного пункту навігації та відображення стану кнопок (наприклад, зміну кольору кнопки «КУПИТИ» при взаємодії з нею);
- принцип відповідності реальному світу полягає у використанні зрозумілої користувачу термінології та звичних метафор. Так, навігація поділена на

«Чоловічий одяг» та «Жіночий одяг» з відповідними піктограмами, а кнопки мають однозначні підписи («КУПИТИ», «ОБРАТИ МОДЕЛЬ», «ЗАЛИШИТИ ЗАЯВКУ»), що не потребують додаткового пояснення;

– принцип контролю та свободи забезпечує можливість легко скасувати дію або повернутися на попередній крок. У межах інтерфейсу це реалізовано через постійну доступність головної навігації, іконки кошика та профілю користувача на кожній сторінці, що дозволяє в будь-який момент перейти до іншого розділу без втрати контексту;
– принцип консистентності та стандартів означає, що однакові елементи повинні виглядати та поводитися однаково на всіх сторінках. У проєкті це досягається завдяки єдиній типографічній сітці, уніфікованій кольоровій палітрі та повторюваній структурі картки товару (зображення, варіанти кольору, назва, ціна), яка зберігається як у каталозі, так і на сторінці детального опису товару;
– принцип запобігання помилкам реалізовано через елементи вибору з обмеженим набором варіантів, наприклад, фіксований перелік кольорів і розмірів товару у вигляді кнопок, що унеможливлює введення некоректного значення, а також через таблицю розмірів, яка допомагає користувачу свідомо обрати правильний варіант ще до додавання товару в кошик;
– принцип розпізнавання, а не пригадування означає, що інформація, потрібна для виконання дії, має бути видимою, а не запам'ятовуватися користувачем. Прикладом є таблиця відповідності розмірів (рис. 2.8), розміщена безпосередньо на сторінці товару, що дозволяє користувачу прийняти рішення без необхідності звертатися до зовнішніх джерел;
– принцип естетичного та мінімалістичного дизайну реалізовано через відмову від надлишкових декоративних елементів: кожен блок сторінки (головний, каталог, секція переваг, FAQ) має чітку функціональну мету, а вільний простір використовується для візуального розділення секцій і зменшення когнітивного навантаження;

– принцип допомоги користувачу розпізнати, діагностувати та виправити помилку реалізовано через розділ «Поширені запитання» (FAQ), де користувач може самостійно знайти відповіді на типові питання щодо адаптивного одягу, догляду за виробами та умов співпраці, що знижує навантаження на службу підтримки та підвищує самостійність користувача у прийнятті рішень;

Розмір глобальний	Розмір український	Стегна (см)	Довжина (см)
XS	44	88-92	48
S	46	92-96	50
M	48	96-100	52
L	50	100-104	54
XL	52	104-108	56

Рис. 2.8. Блок вибору параметрів товару

(розроблено автором)

Перелічені принципи в сукупності формують основу евристичної оцінки інтерфейсу, запропонованої Якобом Нільсеном, і застосовуються як орієнтир при проєктуванні та подальшому тестуванні юзабіліті вебзастосунку [18].

Дизайн будь-якого інтерфейсу потребує ретельної роботи, це не лише питання естетики. Поганий UX здатен звести нанівець навіть найякіснішу функціональність, якщо користувач не може знайти потрібний товар або не розуміє, куди натиснути для оформлення замовлення, він просто покине сайт. Саме тому на етапі проєктування веб-застосунку значну увагу приділено принципам зручності та доступності інтерфейсу.

Дизайн-концепція сайту будується на поєднанні лаконічної естетики та функціональності. Головний принцип – жодних зайвих елементів, лише ті компоненти, що реально допомагають користувачу виконати його задачу. Велику частину площі сторінки відведено під якісні фотографії продукції, оскільки одяг – насамперед візуальний продукт [19].

Кольорова палітра сайту (рис. 2.9) побудована на поєднанні чотирьох базових кольорів, кожен з яких виконує власну функціональну роль в інтерфейсі.

Рис. 2.9. Кольорова палітра та типографіка веб-платформи

(розроблено автором)

Основним фоновим кольором було обрано – #FFFEF5, який використовується для світлих ділянок сторінки та забезпечує комфортне сприйняття великих масивів тексту й зображень, не створюючи різкого контрасту, властивого чистому білому кольору. Як основний колір тексту та елементів темного фону обрано колір – #353532, він замінює традиційний чорний, виглядає менш агресивно при тривалому перегляді та водночас забезпечує достатній контраст для коректного читання. Акцентним кольором бренду виступає зелений – #ACBD6F, він використовується для виділення ключових інтерактивних елементів: кнопок, активних станів навігації, позначок адаптивних товарів, і асоціюється з природністю, надійністю та спокоєм, що

відповідає концепції бренду, орієнтованого на людей з особливими потребами. Додатковим акцентним кольором є насичений синій – #3256A1, який застосовується для другорядних інтерактивних елементів (посилань, іконок, інформаційних бейджів) і створює візуальний баланс із зеленим, формуючи завершену кольорову композицію.

Вибір кольорової гами не є випадковим і ґрунтується на принципах кольорового сприйняття та психології кольору у веб–дизайні. Поєднання теплого світлого фону з темними нейтральними елементами та природними зеленими акцентами створює відчуття спокою, надійності й турботи – якостей, важливих для аудитори, що потребує адаптивного одягу. Синій колір традиційно асоціюється з довірою та стабільністю, що додатково підкреслює серйозність і професійність бренду. Окрім емоційного впливу, кольорова палітра відповідає функціональним вимогам: контрастність пар «фон – текст» (#FFFEF5 на #353532 та навпаки) перевищує мінімальний коефіцієнт 4.5:1, що відповідає рівню AA стандарту WCAG 2.1 для основного тексту, а контраст акцентних кольорів (#ACBD6F, #3256A1) із фоном достатній для виділення кнопок та інтерактивних елементів без порушення читабельності.

Типографіка сайту базується на гротесковому шрифті без засічок HeliosExt, який характеризується чіткими геометричними формами літер, високою розбірливістю на екранах різної роздільної здатності та сучасним, технологічним характером. Використання шрифту без засічок є усталеною практикою у вебдизайні, оскільки такі гарнітури краще зберігають читабельність при малих розмірах та на пристроях з невисокою щільністю пікселів [24].

Типографічна система організована за принципом чіткої ієрархії, що дозволяє користувачу швидко орієнтуватися в структурі сторінки та визначати пріоритетність інформації:

– заголовки першого рівня (H1): мають розмір 50px і використовуються для головних заголовків секцій, таких як назва сторінки або ключове повідомлення головного екрану; вони привертають основну увагу та формують першу точку контакту з контентом;

- заголовки другого рівня (H2): виконано розміром 30px і застосовуються для назв підрозділів сторінки (наприклад, заголовків блоків каталогу чи секцій «Про нас»);
- заголовки третього рівня (H3): мають розмір 24px і використовуються для назв окремих карток товарів, підпунктів та локальних акцентів усередині блоків;
- основний текст (main text): виконано розміром 16px, забезпечує комфортное читання тексту.

Окремо визначено типографічні стилі для кнопок:

- кнопки першого рівня (Button 1): виконано розміром 40px і використовуються для основних закликів до дії (CTA), таких як «Додати до кошика» чи «Оформити замовлення», їх збільшений розмір підкреслює пріоритетність дії та полегшує взаємодію;
- кнопки другого рівня (Button 2): мають розмір 24px і застосовуються для другорядних дій, наприклад, керування кількістю товару в кошику чи навігаційних елементів.

Така диференціація розмірів відповідає принципу візуальної ієрархії дій, згідно з яким найважливіші для бізнес-логіки дії повинні бути найбільш помітними та легкодоступними.

Загалом узгоджена система кольорів і типографіки формує єдину дизайн-мову проєкту: вона забезпечує не лише візуальну привабливість, а й функціональну послідовність – кожен колір та розмір шрифту несе конкретне смислове навантаження і використовується системно на всіх сторінках застосунку, що відповідає принципам системного дизайну як методології побудови масштабованих інтерфейсів [19].

Структура сторінки організована за принципом F-патерну читання. Найважливіший контент розміщено у верхньому лівому куті, оскільки саме туди спрямований погляд користувача при першому огляді сторінки. Кнопки заклику до дії (CTA) виділено контрастним кольором і розміщено у зонах максимальної уваги – під головним екраном та на картках товарів.

Прототипування інтерфейсу здійснювалося у Figma – хмарному інструменті для дизайну UI/UX, що дозволяє створювати як статичні макети, так і інтерактивні прототипи з переходами між екранами. В рамках проєкту розроблено макети для таких сторінок: головна (рис. 2.10), каталог (рис. 2.11), картка товару (рис. 2.12), кошик, оформлення замовлення, особистий кабінет, авторизація та реєстрація.

Рис. 2.10. Прототип сторінки «Головний екран»

(розроблено автором)

Рис 2.11. Прототип каталогу

(розроблено автором)

Рис 2.12. Прототип сторінки товару

(розроблено автором)

Таким чином, UI/UX-дизайн веб-платформи «АДАПТ» побудовано на поєднанні чіткої кольорово-типографічної системи та послідовної компонентної структури, орієнтованих на специфіку цільової аудиторії. Така узгодженість візуальних і функціональних рішень забезпечує не лише привабливий зовнішній вигляд інтерфейсу, а й його практичну зручність, доступність та передбачуваність для користувача на кожному етапі взаємодії із застосунком.

2.4. Проєктування бази даних

База даних (БД) – це організована сукупність структурованих даних, які зберігаються та обробляються в електронному вигляді за допомогою спеціального програмного забезпечення – системи управління базами даних (СУБД). Основне призначення бази даних полягає в забезпеченні надійного, швидкого та зручного доступу до інформації, а також у підтримці її цілісності, узгодженості та захищеності від втрати чи несанкціонованого доступу [11].

Залежно від моделі організації даних розрізняють реляційні бази даних, що зберігають інформацію у вигляді пов'язаних таблиць (наприклад, MySQL, PostgreSQL), та документоорієнтовані (NoSQL) бази даних, що зберігають дані у форматі гнучких документів, подібних до JSON (наприклад, MongoDB). Вибір конкретного типу бази даних залежить від специфіки предметної галузі, очікуваного навантаження та структури даних, з якими працює застосунок.

База даних є фундаментом будь-якої інформаційної системи – від її правильного проєктування залежить не лише коректність роботи застосунку, а й продуктивність під навантаженням. Для розробленого веб-застосунку «АДАПТ» обрано документоорієнтовану СУБД MongoDB у хмарному виконанні MongoDB Atlas, що зумовлено гнучкістю схеми даних, природною відповідністю JSON-структурам, які використовуються у Node.js застосунках, та зручністю масштабування й адміністрування через хмарну платформу [10].

Процес проєктування бази даних охоплював кілька послідовних етапів: концептуальне моделювання (визначення сутностей та їх атрибутів), логічне проєктування (перетворення концептуальної моделі на схеми колекцій Mongoose із конкретними типами полів) та визначення стратегії зберігання пов’язаних даних. У межах кластера використовується база даних adapt_shop, що містить дві основні колекції: users (користувачі) та orders (замовлення) (рис. 2.13).

users	
_id	ObjectId, PK
firstName	String
lastName	String
dob	String
phone	String, unique
password	String (bcrypt)
__v	Number

orders	
_id	ObjectId, PK
items	Array
items[].title	String
items[].colorName, colorCode, size	String
items[].price, quantity	Number
items[].image	String
date	Date

Рис. 2.13. **Схема колекцій бази даних інтернет-магазину АДАПТ**
(розроблено автором)

Структуру колекцій «Користувачі» (табл. 2.1) та «Замовлення» (табл. 2.1) представлено через опис їхніх полів, типів даних та призначення.

Таблиця 2.1

Користувачі

Поле	Тип даних / Опис
_id	ObjectId – унікальний ідентифікатор документа, генерується автоматично
firstName	String – ім’я користувача
lastName	String – прізвище користувача
dob	String – дата народження (формат YYYY-MM-DD)
phone	String – номер телефону у міжнародному форматі
password	String – хеш пароля, згенерований за алгоритмом bcrypt
__v	Number – версія документа (службове поле Mongoose)

(розроблено автором)

Таблиця 2.2

Замовлення

Поле	Тип даних / Опис
_id	ObjectId – унікальний ідентифікатор документа, генерується автоматично
items	Array – масив об’єктів-позицій замовлення
items[].id	String – ідентифікатор позиції (відповідає назві товару)
items[].title	String – назва товару
items[].colorName	String – назва кольору виробу
items[].colorCode	String – код кольору у форматі rgb()
items[].size	String – розмір виробу (S/M/L/XL тощо)
items[].price	Number – ціна товару на момент замовлення

Продовження табл. 2.2

items[].image	String – шлях/посилання на зображення товару
items[].quantity	Number – кількість одиниць товару
date	Date – дата й час оформлення замовлення
__v	Number – версія документа (службове поле Mongoose)

(розроблено автором)

Окремої уваги заслуговує поле items[].price у колекції orders. На перший погляд може здатися надлишковим зберігати ціну товару в позиції замовлення, оскільки інформація про товар може зберігатися окремо. Проте ціна на товар може змінюватися з часом, і якщо покладатися лише на актуальну ціну, то при перегляді історії замовлень система показуватиме некоректні суми. Поле price зберігає ціну товару на момент покупки та не змінюється після оформлення замовлення. Такий підхід забезпечує збереження коректної інформації про замовлення і є типовим для документоорієнтованих баз даних [14].

Структура документа orders демонструє ключову перевагу NoSQL-підходу для предметної галузі проєкту: усі позиції одного замовлення (товар, колір, розмір, кількість, ціна) зберігаються як вкладений масив items безпосередньо в документі замовлення, без необхідності виконувати додаткові запити чи об’єднання для отримання повної інформації про замовлення. Це особливо доцільно, оскільки кошик та замовлення в інтернет-магазині завжди читаються й обробляються як єдина цілісна сутність.

Для забезпечення безпеки даних поле password у колекції users зберігається виключно у вигляді хешу, отриманого за допомогою бібліотеки bcrypt, що унеможливлює відновлення оригінального пароля навіть у разі компрометації бази даних. Кожен документ автоматично отримує унікальний ідентифікатор _id типу ObjectId, який використовується для зв’язку записів – наприклад, для прив’язки замовлення до конкретного користувача через посилання на його _id.

Для оптимізації продуктивності у колекції users створено унікальний індекс на поле phone (для швидкої перевірки під час реєстрації та автентифікації), а в колекції orders передбачено індекс на поле date (для сортування замовлень за хронологією в особистому кабінеті користувача). Наявність правильно спроєктованих індексів дозволяє суттєво знизити час виконання запитів при зростанні обсягу даних навіть у документоорієнтованих СУБД [21].

У підсумку, розроблена схема бази даних є гнучкою, масштабованою та відповідає вимогам предметної галузі інтернет-магазину адаптивного одягу. Обрана модель даних враховує специфіку асортименту (вкладені атрибути товару: колір, розмір, зображення в межах позиції замовлення), забезпечує цілісність історичних даних (фіксація ціни на момент покупки) та дозволяє ефективно масштабувати систему завдяки хмарній інфраструктурі MongoDB Atlas.

Висновки до розділу 2

Проєктування онлайн-платформи охопило чотири взаємопов'язані напрями, що у сукупності сформували повну специфікацію майбутнього застосунку:

- побудову інформаційної моделі та архітектури;
- моделювання алгоритмів взаємодії;
- розробку UI/UX-дизайну;
- проєктування бази даних.

Інформаційна модель системи виокремила п'ять сутностей предметної галузі: User, Category, Product, Order та OrderItem. Архітектура застосунку реалізована за трирівневою моделлю, де клієнтський рівень відповідає за відображення та взаємодію, серверний рівень обробляє бізнес-логіку через REST API, а рівень даних забезпечує зберігання. Така організація дозволяє змінювати кожен рівень незалежно від інших.

Моделювання алгоритму взаємодії описало сім послідовних етапів сценарію роботи користувача: від знайомства з головною сторінкою через перегляд каталогу, вибір товару та перевірку авторизації до формування кошика, оформлення замовлення і підтвердження. Визначено, що перевірка авторизації є обов'язковим кроком перед збереженням товару до кошика, що забезпечує прив'язку замовлень до конкретного облікового запису.

UI/UX-дизайн веб-застосунку ADAPT спирається на систему з восьми евристичних принципів Нільсена, кожен з яких реалізований у конкретних рішеннях інтерфейсу. Кольорова палітра побудована на чотирьох кольорах. Типографічна система на основі шрифту HeliosExt включає чотири рівні заголовків та окремі стилі для кнопок першого і другого рівнів, що реалізує принцип візуальної ієрархії. Прототипування у Figma охопило вісім ключових сторінок застосунку.

Проєктування бази даних обґрунтувало вибір документоорієнтованої СУБД MongoDB Atlas на противагу реляційним альтернативам, виходячи з гнучкості схеми та природної відповідності JSON-структурам Node.js застосунків. Визначено структуру двох колекцій: users та orders. Для оптимізації запитів у колекції users створено унікальний індекс на поле phone, у колекції orders передбачено індекс на поле date.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЄКТУ

3.1. Реалізація серверної частини та налаштування бази даних

Серверна частина веб-застосунку реалізована на базі Node.js – середовища виконання JavaScript поза браузером, яке завдяки подієво-орієнтованій однопоточній моделі забезпечує ефективну обробку великої кількості одночасних HTTP-запитів без блокування потоку виконання. Вибір Node.js обумовлений уніфікацією мови програмування на обох рівнях застосунку: і клієнтська, і серверна частини написані на JavaScript, що спрощує підтримку коду [22].

Для побудови HTTP-сервера та маршрутизації запитів використовується Express.js – фреймворк, що не нав'язує жорсткої архітектурної структури, але дає розробнику всі необхідні інструменти: middleware-конвеєр, гнучку маршрутизацію та зручну роботу з об'єктами запиту та відповіді [12].

Перелік npm-пакетів, задіяних у серверній частині, визначено у файлі package.json:

```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs",
  "dependencies": {
    "bcryptjs": "^3.0.3",
```

```
    "cors": "^2.8.6",
    "express": "^5.2.1",
    "jsonwebtoken": "^9.0.3",
    "mongoose": "^9.6.3"
  }
}
```

Кожна залежність виконує чітко обмежену функцію:

- express відповідає за HTTP-сервер і маршрутизацію;
- mongoose – за взаємодію з MongoDB через схеми та моделі;
- bcryptjs – за хешування паролів;
- jsonwebtoken – за генерацію та верифікацію JWT-токенів;
- cors – за дозвіл крос-оригінальних запитів між клієнтом і сервером.

Такий принцип залежностей є усталеною практикою, що сприяє передбачуваності поведінки системи та спрощує оновлення [26].

Ініціалізація сервера відбувається шляхом підключення необхідних пакетів, задання константи порту та налаштування middleware-конвеєра. Перший з викликів: app.use(cors()) – дозволяє браузеру надсилати запити до API з іншого порту (клієнтська частина працює на порті 5500, серверна – на 3000).

Другий: app.use(express.json()) – забезпечує автоматичний парсинг JSON-тіла вхідних запитів:

```
const express = require('express');
const cors = require('cors');
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const path = require('path');

const app = express();
const PORT = 3000;
const JWT_SECRET = 'adapt_super_secret_key_2026';

app.use(cors());
app.use(express.json());
```

```
const publicPath = __dirname;
app.use(express.static(publicPath));
```

Для роздачі статичних файлів клієнтської частини використано app.use(express.static()), що дозволяє звертатися до HTML, CSS та JS безпосередньо через той самий сервер.

Підключення до хмарної бази даних MongoDB Atlas здійснюється через виклик mongoose.connect() з рядком підключення (URI), що містить облікові дані та адресу кластера:

```
const MONGO_URI =
'mongodb+srv://engorbal_db_user:vXGQELkT4mltCyDx@cluster0.dab8ucz.
mongodb.net/adapt_shop?appName=Cluster0';
mongoose.connect(MONGO_URI)
    .then(() => console.log('✅ Успішно підключено до бази!'))
    .catch((err) => console.error('❌ Помилка бази:', err));
```

Mongoose є ODM-бібліотекою (Object Document Mapper), яка надає можливість описувати структуру документів через схеми та автоматично валідувати дані перед записом. Використання хмарного кластера Atlas усуває потребу у самостійному адмініструванні серверів бази даних і забезпечує автоматичне резервне копіювання та масштабування [9].

Схеми даних описані безпосередньо у файлі server.js за допомогою конструктора mongoose.Schema:

```
const User = mongoose.model('User', new mongoose.Schema({
    firstName: String, lastName: String, dob: String, phone:
String, password: String
}));
const Order = mongoose.model('Order', new mongoose.Schema({
    userId: String,
    items: Array,
    address: String,
    payment: String,
    date: { type: Date, default: Date.now }
}));
```

Для колекції users визначено поля: firstName, lastName, dob, phone та password – усі типу String. Поле password зберігає не відкритий текст, а хеш, отриманий бібліотекою bcryptjs. Колекція orders містить масив items (товари замовлення), поля address, payment та date.

Серверна частина реалізує чотири API-маршрути, що утворюють мінімальний, але самодостатній REST-інтерфейс для роботи з користувачами та замовленнями:

```
app.post('/api/register', async (req, res) => {
    try {
        const { firstName, lastName, dob, phone, password } =
req.body;
        const hashedPassword = await bcrypt.hash(password, 10);
        await new User({ firstName, lastName, dob, phone,
password: hashedPassword }).save();
        res.status(201).json({ message: 'Реєстрація пройшла
успішно!' });
    } catch (e) { res.status(500).json({ message: 'Помилка' }); }
});
```

Маршрут POST /api/register приймає дані нового користувача, хешує пароль за допомогою bcrypt.hash() з коефіцієнтом складності 10 (що означає 2^10 ітерацій алгоритму) і зберігає документ у колекцію users. Зберігання виключно хешу замість відкритого пароля є обов'язковою практикою, що унеможливлює відновлення оригінального пароля навіть у разі витоку бази даних. Маршрут POST /api/login відповідає за автентифікацію користувача:

```
app.post('/api/login', async (req, res) => {
    try {
        const user = await User.findOne({ phone: req.body.phone
});
        if (!user || !(await bcrypt.compare(req.body.password,
user.password)))
            return res.status(400).json({ message: 'Невірний логін
або пароль' });
```

```
        const token = jwt.sign({ userId: user._id }, JWT_SECRET, {
expiresIn: '1h' });
        res.status(200).json({ token: token });
    } catch (e) { res.status(500).send(); }
});
```

Спочатку здійснюється пошук документа у колекції users за номером телефону. Якщо користувача не знайдено або порівняння хешів через bcrypt.compare() повертає false, сервер відповідає статусом 400 та повідомленням про помилку. У разі успішної перевірки генерується JWT-токен із вбудованим userId. Клієнт отримує токен у тілі відповіді та зберігає його у localStorage для подальшого використання в авторизованих запитах.

Маршрут GET /api/profile забезпечує доступ до даних авторизованого користувача.

```
app.get('/api/profile', async (req, res) => {
    try {
        const token = req.headers.authorization?.split(' ')[1];
        const decoded = jwt.verify(token, JWT_SECRET);
        const user = await User.findById(decoded.userId).select('-
password');
        res.status(200).json(user);
    } catch (e) { res.status(401).send(); }
});
```

Токен витягується з заголовка Authorization у форматі Bearer <token>, верифікується через jwt.verify() і декодується для отримання userId. На основі отриманого ідентифікатора здійснюється запит до MongoDB; метод .select('-password') виключає поле пароля з відповіді, що є стандартною практикою захисту чутливих даних навіть у межах авторизованих запитів.

Маршрут POST /api/orders реалізує збереження замовлення. Логіка маршруту передбачає необов'язкову авторизацію: якщо в заголовку присутній токен, він верифікується і замовлення прив'язується до конкретного userId; якщо токена немає, то замовлення зберігається з userId = 'Гість', що дозволяє оформлювати замовлення без реєстрації. Маршрут GET /api/my-orders повертає

список замовлень конкретного користувача, відсортованих від найновішого до найстарішого, що відображається в особистому кабінеті (див. додаток А).

Для запуску серверу використовується виклик app.listen(PORT), після якого застосунок починає прослуховувати вхідні HTTP-запити на порті 3000. Результат підключення до MongoDB та адреса запущеного сервера виводяться в консоль через console.log() і console.error().

3.2. Реалізація функціональності веб-застосунку

Функціональна логіка клієнтської частини зосереджена в єдиному файлі script.js, що підключається на кожній сторінці застосунку і відповідає за всі інтерактивні процеси: управління кошиком, роботу модальних вікон, взаємодію з REST API та захист маршрутів.

Однією з перших реалізованих функцій є зміна головного зображення товару при кліку на кольоровий свотч:

```
function changeProductImage(clickedElement, imageId) {
    const imgElement = document.getElementById(imageId);
    if (imgElement) {
        const newImageSrc = clickedElement.getAttribute('data-
image');
        imgElement.src = newImageSrc;
    }
    const swatches =
clickedElement.parentElement.getElementsByClassName('swatch');
    for (let i = 0; i < swatches.length; i++) {
        swatches[i].classList.remove('selected');
    }
    clickedElement.classList.add('selected');
}
```

Функція changeImage() використовується на сторінках каталогів (mens.html, womens.html), а changeProductImage() – на сторінках окремих товарів. Обидві функції працюють за однаковим принципом: зчитують атрибут data-

image клікнутого елемента, замінюють ним значення src основного зображення і перемикають клас selected між свотчами для візуального підсвічення активного.

Кошик замовлень реалізовано через механізм localStorage – вбудованого браузерного сховища, яке дозволяє зберігати JSON-серіалізовані дані на стороні клієнта між сесіями:

```
let cart = JSON.parse(localStorage.getItem('adapt_cart')) || [];
function saveCart() {
    localStorage.setItem('adapt_cart', JSON.stringify(cart));
    renderCart();
}
function openCartModal() {
    const cartModalElement = document.getElementById('cartModal');
    if (cartModalElement) {
        cartModalElement.classList.add('active');
        renderCart();
    }
}
function closeCartModal() {
    const cartModalElement = document.getElementById('cartModal');
    if (cartModalElement) {
        cartModalElement.classList.remove('active');
    }
}
function updateQuantity(index, delta) {
    if (cart[index] && cart[index].quantity + delta > 0) {
        cart[index].quantity += delta;
        saveCart();
    }
}
function removeFromCart(index) {
    cart.splice(index, 1);
    saveCart();
}
```

При завантаженні сторінки масив cart ініціалізується з localStorage; при кожній зміні (додаванні, видаленні, зміні кількості) оновлений масив серіалізується назад через JSON.stringify() і записується до сховища. Функція saveCart() виконує запис і одночасно оновлює відображення кошика через renderCart(), що забезпечує реактивне оновлення інтерфейсу без перезавантаження сторінки.

Функція handleAddToCart() є центральним елементом клієнтської логіки покупок. Першою дією вона перевіряє наявність JWT-токена в localStorage: якщо токен відсутній, за допомогою SweetAlert2 відображається модальне вікно з пропозицією авторизуватися, і подальше виконання функції переривається оператором return. Такий підхід перевіряє авторизацію на рівні клієнта до відправки запиту та скорочує зайвий мережевий трафік, тим самим забезпечуючи миттєвий зворотний зв'язок для користувача. Якщо ж авторизацію підтверджено, функція зчитує активний розмір та активний колір зі сторінки товару, формує об'єкт product і або додає його до масиву cart, або збільшує кількість наявної позиції, якщо товар з такими ж id, colorCode та size вже присутній у кошику (див. додаток Б).

Функція renderCart() формує HTML-розмітку кошика динамічно на підставі поточного масиву cart. Для кожного елемента генерується блок cart-item із зображенням, назвою, кольоровим індикатором, розміром, кнопками зміни кількості та кнопкою видалення. Загальна сума обчислюється як сума добутків price * quantity для кожної позиції і відображається в блоці підсумку. Завдяки тому, що renderCart() викликається після кожної операції із кошиком, інтерфейс завжди відображає актуальний стан (рис. 3.1).

Процес оформлення замовлення реалізовано через функцію checkoutOrder(). При її виклику відображається модальне вікно SweetAlert2 з вбудованою HTML-формою, що містить поле введення міста (з підказками у вигляді datalist), поле для номера відділення Нової Пошти та випадаючий список способу оплати. Валідація обов'язкових полів відбувається всередині preConfirm-колбека: якщо поля порожні, через Swal.showValidationMessage()

відображається повідомлення про помилку і форма не закривається. Після підтвердження дані разом із вмістом кошика надсилаються на сервер через fetch POST /api/orders; у разі успішної відповіді кошик очищується, а користувача перенаправляють до особистого кабінету (див. дотаток В).

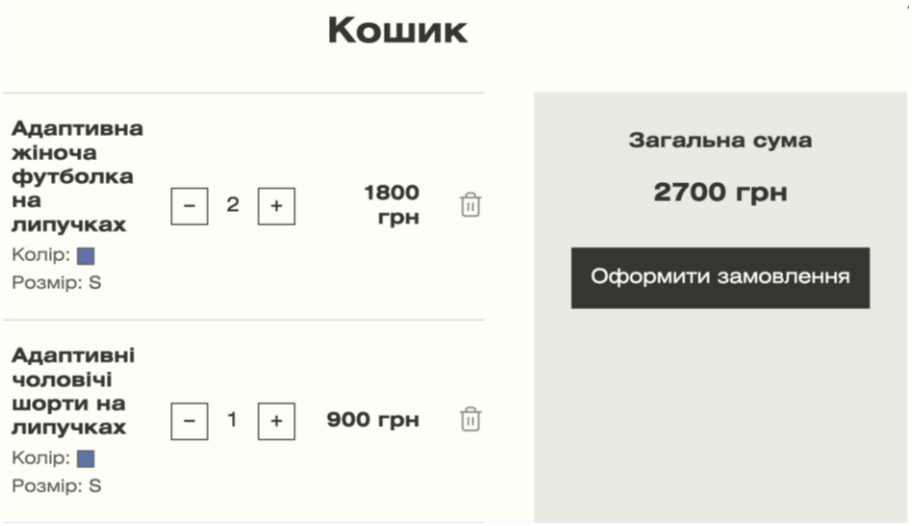

Рис. 3.1. Відображення кошику

(розроблено автором)

Логіка реєстрації (див. додаток Г) та входу (див. Додаток Д) реалізована через обробники подій submit відповідних форм. Форма реєстрації на сторінці register.html надсилає POST-запит до /api/register з п'ятьма полями: ім'ям, прізвищем, датою народження, номером телефону та паролем. Перед відправкою виконується клієнтська перевірка збігу паролів; у разі успішної відповіді сервера SweetAlert2 сповіщає про створення акаунту і виконується перехід на попередню сторінку.

Особливу увагу приділено захисту сторінки профілю. Під час її завантаження JavaScript перевіряє наявність токена в localStorage, і якщо його немає, користувача одразу перенаправляє на попередню сторінку або головну. Якщо токен присутній, паралельно запускаються два fetch-запити: GET /api/profile для отримання персональних даних та GET /api/my-orders для завантаження історії замовлень.

У разі якщо сервер повертає помилку верифікації токена (прострочений або підроблений), відображається сповіщення SweetAlert2 про закінчення сесії і токен видаляється з localStorage. Такий підхід забезпечує двошаровий захист: на рівні клієнта (перевірка наявності токена) та на рівні сервера (верифікація його підпису).

3.3. Програмна реалізація інтерфейсу користувача

Клієнтська частина застосунку являє собою MPA (Multi-Page Application) – набір HTML-сторінок із спільним файлом стилів style.css та єдиним JavaScript-файлом script.js. Кожна сторінка є самодостатньою HTML-одиницею зі своїм контентом, однак усі вони відтворюють однаковий хедер, навігацію та футер, що дає користувачу відчуття єдиного застосунку.

Хедер зафіксовано у верхній частині екрана через position: fixed, що забезпечує постійну доступність навігації незалежно від позиції прокручування (рис. 3.2). Для компенсації перекриття контенту фіксованим хедером до основного блоку кожної сторінки додано padding-top зі значенням, що відповідає висоті хедера.

Рис. 3.2. Хедер веб-застосунку ADAPT з навігацією та блоком управління
(розроблено автором)

Головна сторінка (index.html) складається з декількох тематичних секцій: головний екран з ключовим повідомленням та CTA-кнопкою, секції переваг адаптивного одягу, блоку-баннера із зверненням до потенційних партнерів, секції FAQ та футера. Головний еркна містить великий заголовок, короткий опис та кнопку «ОБРАТИ МОДЕЛЬ», що веде до каталогу (рис. 3.3).

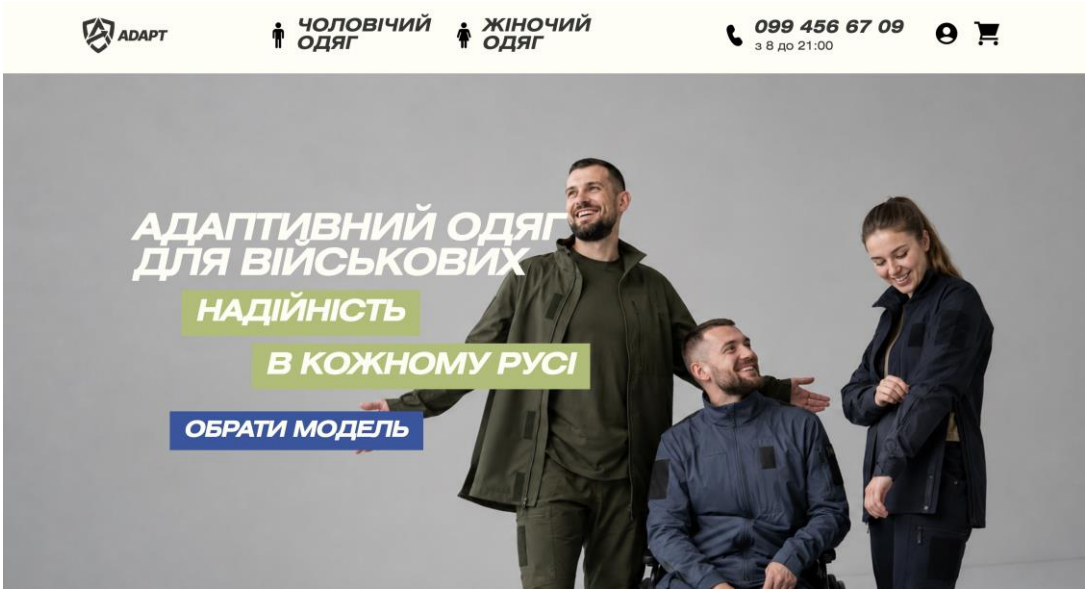

Рис. 3.3. Головна сторінка вебзастосунку ADAPT

(розроблено автором)

Сторінки каталогу чоловічого та жіночого одягу побудовані за єдиним шаблоном: заголовок із назвою розділу, сітка карток товарів і підключений script.js. Кожна картка містить зображення, кольорові свотчі, назву та ціну. Клік на свотч змінює зображення картки без переходу на іншу сторінку, клік на назву або зображення веде на деталізовану сторінку товару.

Сторінки окремих товарів мають дворівневий макет: ліва частина відведена під велике зображення товару, вибір кольору, розміру та таблицю вимірів; права – під назву, опис, ціну, кнопку купівлі, блок матеріалів, іконки переваг та секцію умов доставки. Такий поділ дозволяє виконати ключову дію: обрати розмір та колір і купити товар без необхідності прокручувати сторінку (рис. 3.4).

Таблиця розмірів реалізована стандартним HTML-елементом <table> зі стилізованими заголовками та рядками:

```
const sizeBoxes = document.querySelectorAll('.size-box');
    sizeBoxes.forEach(box => {
        box.addEventListener('click', function() {
            sizeBoxes.forEach(b =>
b.classList.remove('selected'));
```

```
            this.classList.add('selected');
        });
    });
```

Вибір розміру здійснюється кліком на один із блоків size-box; обраний варіант підсвічується через клас selected. Відповідна логіка реалізована через перебір querySelector і перемикання класів.

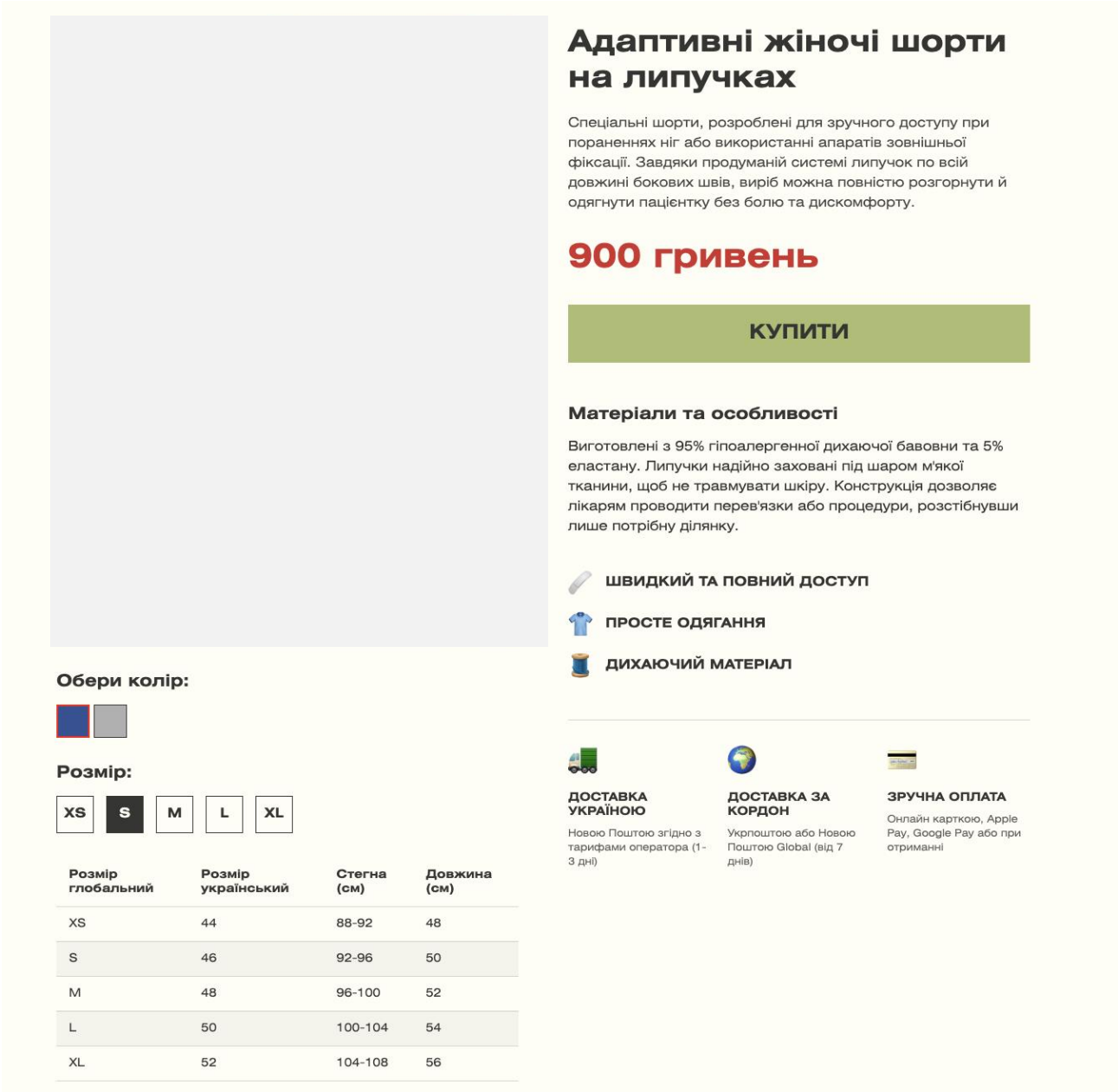

Рис. 3.4. Сторінка товару

(розроблено автором)

Модальне вікно входу (рис. 3.5) відображається через функцію openLoginModal(). При виклику функція спочатку перевіряє наявність токена: якщо авторизований користувач натискає на іконку профілю, відбувається перехід, а не відкриття форми входу. Сам модальний елемент отримує клас active через classList.add(), що змінює його видимість.

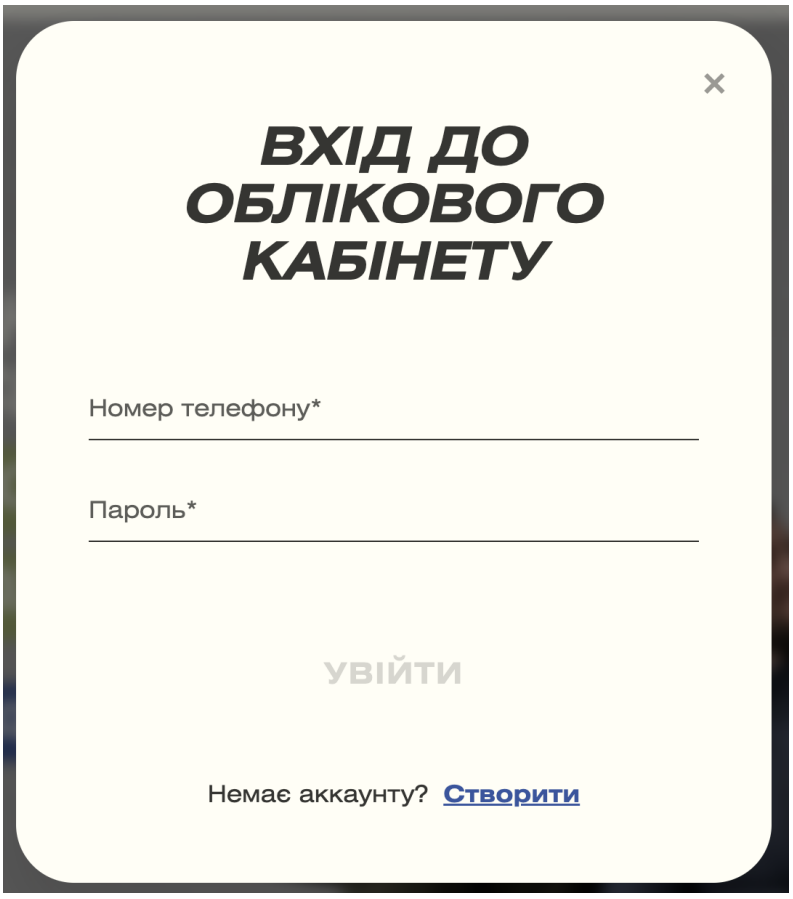

Рис. 3.5. Вікно входу до облікового кабінету

(розроблено автором)

Сторінка реєстрації є окремою HTML-сторінкою зі своїми стилями (рис. 3.6). Форма містить сім полів: ім'я, прізвище, дата народження (з перетворенням типу input між text і date при фокусуванні), номер телефону, чекбокс згоди на обробку персональних даних, пароль та підтвердження пароля. Поле дати народження реалізовано із динамічною зміною type через onfocus та onblur, що забезпечує відображення підказки-placeholder до вибору дати, але при активації відкриває нативний date-picker браузера.

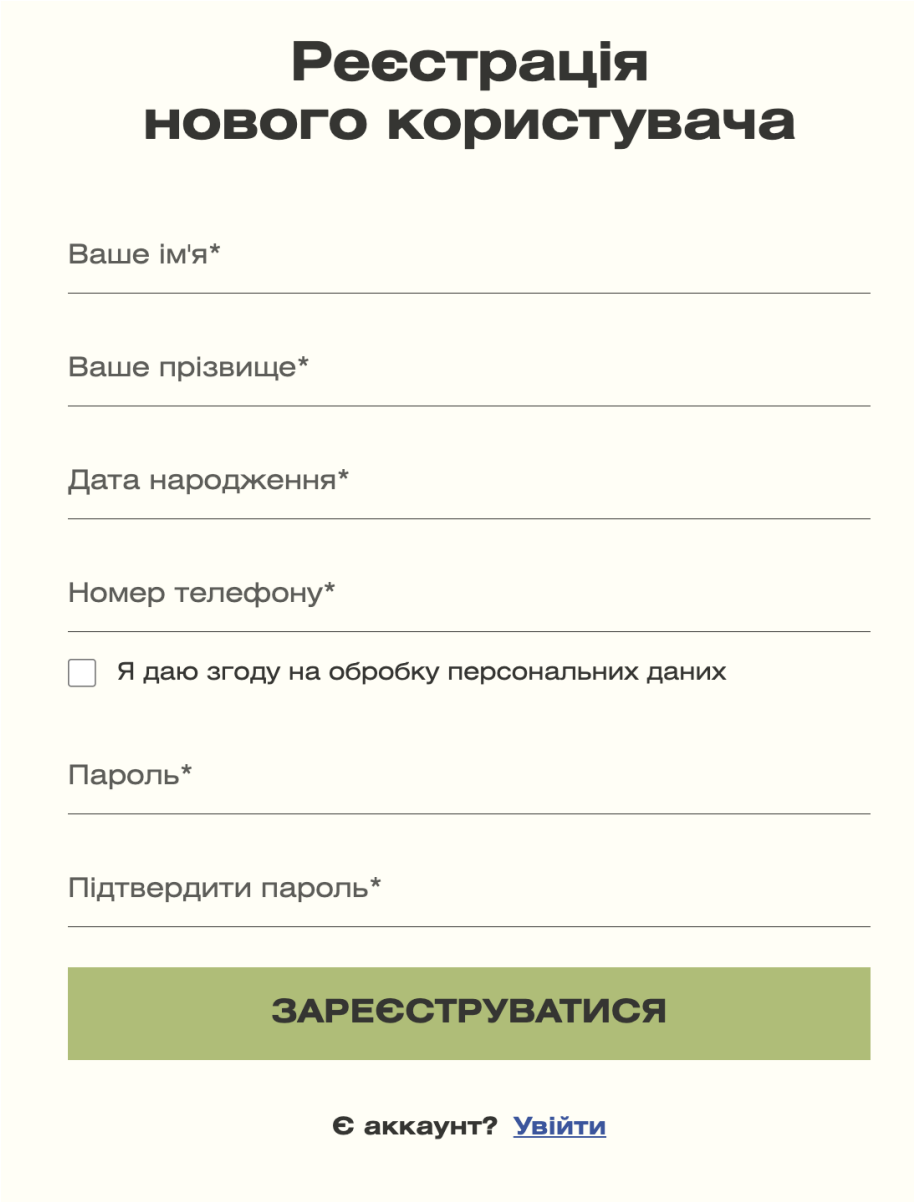

Рис. 3.6. Сторінка реєстрації нового користувача

(розроблено автором)

Для сповіщень та підтверджуючих діалогів у застосунку використовується бібліотека SweetAlert2, підключена через CDN. Вона замінює стандартні браузерні попередження на стилізовані, анімовані спливаючі вікна, які за зовнішнім виглядом узгоджуються з дизайном застосунку. SweetAlert2 підтримує кастомний HTML-контент всередині вікна, що використано в діалозі оформлення замовлення для вбудовування форми з полями введення адреси та вибору способу оплати (рис. 3.7).

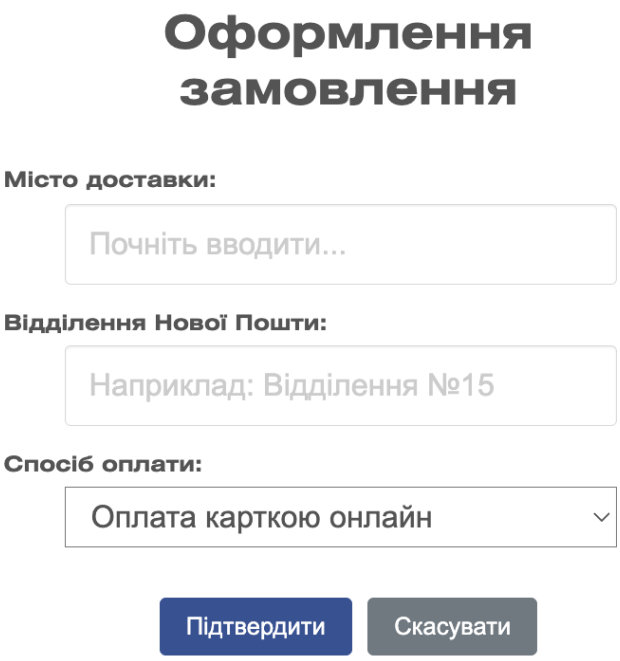

Рис. 3.7. Модальне вікно оформлення замовлення

(розроблено автором)

Наявність параметра preConfirm дозволяє валідувати вміст форми перед закриттям діалогу, не вдаючись до сторонніх бібліотек для клієнтської валідації [8].

3.4. Тестування вебсайту

Тестування є невід'ємним етапом розробки програмного забезпечення, що дозволяє переконатися у відповідності реалізованого продукту поставленим вимогам і виявити відхилення у поведінці системи до її розгортання. У межах даного проєкту проведено функціональне та UI-тестування, спрямоване на перевірку коректності роботи всіх ключових сценаріїв взаємодії користувача з платформою

Реєстрація нового користувача перевірялась двома способами із правильно заповненими полями та з даними користувача, який вже зареєстрований у системі. У першому випадку система відобразила сповіщення про створення акаунту (рис. 3.8) і перенаправила на голову сторінку.

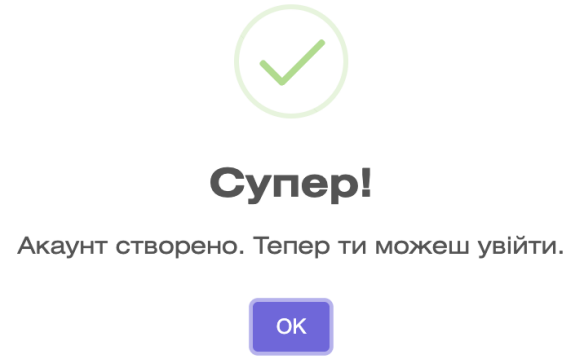

Рис. 3.8. Сповіщення про реєстрацію акаунту

(розроблено автором)

У другому варіанті – з'явилось повідомлення про помилку, а запит до сервера не відправлявся.

Авторизація користувача тестувалась із коректними та некоректними обліковими даними: при успішному вході відбувалося перенаправлення до профілю; при невірних даних сервер повертав статус 400 і відповідне повідомлення про помилку (рис. 3.9).

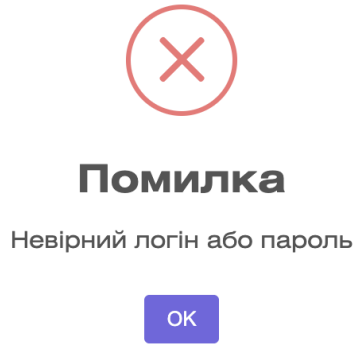

Рис. 3.9. Повідомлення про помилку авторизації

(розроблено автором)

Додавання товару до кошика тестувалось у двох контекстах: для авторизованого та неавторизованого користувача. Авторизований користувач при натисканні «КУПИТИ» отримував відкрите модальне вікно кошика з доданою позицією. Неавторизований – сповіщення з пропозицією увійти; після натискання «Увійти» у діалозі відкривалось модальне вікно входу. Перевірено також коректність злиття позицій: повторне додавання того самого товару в тому

самому кольорі та розмірі збільшувало кількість наявної позиції, а не створювало дублікат.

Оформлення замовлення тестувалось зі спробою підтвердити форму з порожніми полями та з заповненими даними. У першому випадку відобразив повідомлення про помилку; у другому – кошик очищувався і виконувався перехід до профілю. На стороні сервера у MongoDB Atlas було підтверджено появу нового документа у колекції orders зі всіма о чікуваними полями:


```
{
  "_id": {
    "$oid": "6a2f013530dd727b2586d34f"
  },
  "userId": "6a176b0b454340fc792db20c",
  "items": [
    {
      "id": "Адаптивна жіноча футболка на липучках",
      "title": "Адаптивна жіноча футболка на липучках",
      "colorName": "Синій",
      "colorCode": "rgb(46, 82, 150)",
      "size": "S",
      "price": 900,
      "image":          "http://localhost:3000/images/woman_t-
shirt_blue.jpg",
      "quantity": 1
    }
  ],
  "address": "Кривий Ріг, 27",
  "payment": "Оплата карткою",
  "date": {
    "$date": "2026-06-14T19:29:57.559Z"
  },
  "__v": 0
```


Особистий кабінет перевірявся на коректне відображення даних профілю та списку замовлень. Спроба прямого переходу до профілю без авторизації призводить до появи модального вікна входу. Після входу сторінка коректно

завантажувала ім'я, прізвище та телефон користувача, а секція замовлень відображала хронологічно відсортований перелік із деталями кожного замовлення (адреса доставки, спосіб оплати, склад позицій).

Перевірка безпеки JWT здійснювалась через ручне маніпулювання токеном у localStorage: при підміні токена на довільний рядок сервер повертав статус 401, клієнт відображав повідомлення про завершення сесії і видаляв некоректний токен зі сховища. Перевірено також поведінку при простроченні токена (через 1 годину) – результат ідентичний: сервер відхиляв запит, клієнт ініціював повторну авторизацію. Ці тести підтверджують, що JWT-механізм функціонує коректно і не допускає несанкціонованого доступу до захищених ресурсів.

Загальний результат тестування підтвердив, що розроблений веб-застосунок відповідає визначеному функціональному завданню: усі основні сценарії взаємодії: перегляд каталогу, вибір товару, управління кошиком, реєстрація, авторизація, оформлення замовлення та перегляд особистого кабінету – функціонують коректно як на десктопі, так і на мобільних пристроях. Виявлені в процесі тестування дрібні неточності у валідаційних повідомленнях та граничних сценаріях були усунуті в ході ітераційного доопрацювання.

Висновки до розділу 3

Програмна реалізація засвідчила технічну здійсненність усіх проєктних рішень, сформованих у попередньому розділі, та підтвердила їх коректність у ході тестування.

Серверна частина реалізована на Node.js із Express.js та забезпечує чотири REST API-маршрути. Маршрут POST /api/register хешує пароль алгоритмом bcrypt із коефіцієнтом складності 10 перед записом до бази, що унеможливлює відновлення оригінального пароля навіть при компрометації сховища. Маршрут POST /api/login генерує JWT-токен із вбудованим ідентифікатором користувача та терміном дії одна година. Маршрут GET /api/profile виключає поле пароля з відповіді через метод .select('-password'). Маршрут POST /api/orders реалізує

необов'язкову авторизацію: неавторизований користувач може оформити замовлення зі статусом «Гість», авторизований отримує прив'язку замовлення до власного облікового запису.

Клієнтська функціональність зосереджена в єдиному файлі script.js. Кошик реалізовано через localStorage з реактивним оновленням інтерфейсу після кожної операції без перезавантаження сторінки. Функція handleAddToCart() перевіряє наявність JWT-токена до відправки запиту на сервер, забезпечуючи миттєвий зворотний зв'язок для неавторизованого користувача. Оформлення замовлення відбувається через модальне вікно SweetAlert2 із вбудованою формою, де preConfirm-логіка валідує обов'язкові поля до закриття діалогу. Захист сторінки профілю реалізовано двошарово: клієнтська перевірка наявності токена блокує відображення сторінки без авторизації, серверна верифікація підпису відхиляє прострочені або підроблені токени зі статусом 401.

Інтерфейс реалізовано як MPA (Multi-Page Application) із єдиними файлами style.css та script.js, спільними для всіх сторінок. Хедер закріплено через position: fixed, що забезпечує постійний доступ до навігації при прокрученні. Вибір кольорового свотча змінює головне зображення товару через атрибут data-image без переходу на іншу сторінку. Вибір розміру реалізовано через перемикання CSS-класу selected між блоками size-box.

Функціональне тестування охопило всі ключові сценарії взаємодії як у позитивному, так і в негативному варіантах. Реєстрація з існуючим номером телефону коректно повертала повідомлення про помилку без відправки запиту на сервер. Авторизація з некоректними даними отримувала статус 400 від сервера. Повторне додавання ідентичного товару збільшувало кількість наявної позиції замовлення, а не створювало дублікат. Перевірка JWT-механізму через ручну підміну токена підтвердила коректне відхилення сервером підроблених і прострочених токенів. Усі виявлені неточності у валідаційних повідомленнях та граничних сценаріях усунуто в ході ітераційного доопрацювання. Застосунок функціонує коректно на десктопних і мобільних пристроях.

ВИСНОВКИ

Виконана кваліфікаційна робота присвячена розробці веб-платформи для замовлення адаптивного одягу для поранених військових і вирішує практичну проблему, пов’язану з відсутністю в Україні зручного цифрового інструменту, який поєднував би функції інтернет-магазину з урахуванням специфічних потреб людей, що перебувають на лікуванні та реабілітації. Замість окремих рішень, представлених наразі лише чатами волонтерських ініціатив або інтернет-магазинами загального типу з не зовсім зручним інтерфейсом, у роботі запропоновано цілісний застосунок, у якому кожен етап взаємодії, від перегляду асортименту до отримання підтвердження замовлення, підпорядкований одній меті: максимально спростити процес для аудиторії, яка фізично обмежена у можливостях і часі.

Шлях до реалізації цієї ідеї пройшов через кілька логічно пов’язаних стадій, кожна з яких знімала одне з ключових питань проєкту. Спочатку було з’ясовано, чому існуючі ініціативи не вирішують проблему повністю: одні з них не мають власної платформи і працюють виключно через соціальні мережі, інші є повноцінними магазинами, проте перевантажують користувача дрібними елементами інтерфейсу та не пропонують особистого кабінету для контролю за статусом замовлення. Це дало чітке розуміння того, яких саме функцій не вистачає ринку, і визначило технологічний орієнтир: пара HTML/CSS/JavaScript на клієнті та Node.js з Express.js і MongoDB Atlas на сервері, обрана через єдність мови програмування на обох рівнях та гнучкість документоорієнтованої моделі даних.

Наступним кроком стало перетворення цього розуміння на конкретний проєкт системи. П’ять сутностей предметної області (користувач, категорія, товар, замовлення та позиція замовлення) і трирівнева архітектура дали застосунку технічний каркас, а семиетапний сценарій взаємодії, від головної сторінки до підтвердження замовлення, додав до цього каркасу логіку поведінки реального відвідувача. Саме на цьому етапі закладено принцип, який згодом

визначив усю подальшу реалізацію: перевірка авторизації виконується саме перед збереженням товару в кошику, а не раніше, що дозволяє користувачу вільно ознайомлюватися з каталогом і лише в момент прийняття рішення про покупку прив’язує дію до конкретного облікового запису.

Дизайнерська частина роботи перетворила набір вимог на конкретний вигляд інтерфейсу. Вибір кольорової палітри, типографіки на основі шрифту HeliosExt та восьми евристичних принципів юзабіліті не були довільними рішеннями, а слугували одній меті: зробити взаємодію передбачуваною для людини, яка може мати обмежену мобільність або перебувати у стресовому стані. Картка товару з фіксованим набором кольорів і розмірів, таблиця відповідності розмірів безпосередньо на сторінці товару та розділ часто задаваних питань є прикладами того, як абстрактний принцип юзабіліті перетворюється на конкретний елемент сторінки, що знижує кількість дій, потрібних для завершення замовлення.

Проєктування бази даних завершило підготовчий етап і водночас заклало основу для надійної роботи всієї системи. Рішення зберігати позиції замовлення як вкладений масив всередині документа orders, а не як окремі записи, що потребують додаткових запитів, є прямим наслідком специфіки предметної галузі: кошик і замовлення завжди розглядаються як єдина сутність. Фіксація ціни товару на момент покупки усуває потенційну розбіжність між історією замовлень і поточним каталогом, а індекси на полях phone та date забезпечують швидкість роботи системи навіть за зростання обсягу даних.

Програмна реалізація об’єднала всі попередні рішення в працюючий продукт. На сервері це виявилося у чотирьох API-маршрутах, які забезпечують повний життєвий цикл користувача: реєстрацію з хешуванням пароля, вхід із видачею JWT-токена, отримання даних профілю та збереження замовлення з можливістю прив’язки до конкретного облікового запису або без реєстрації. На клієнті це виявилося у динамічному кошику на основі localStorage, перевірці авторизації безпосередньо в момент додавання товару, а також у формі оформлення замовлення з валідацією введених даних, яка реалізована без

перезавантаження сторінки. Захист сторінки особистого кабінету через комбінацію клієнтської перевірки токена та його серверної верифікації є прикладом того, як архітектурне рішення про трирівневу структуру системи реалізувалося на рівні конкретного коду.

Останнім етапом стала перевірка того, наскільки реалізований застосунок відповідає закладеним у нього вимогам. Тестування основних сценаріїв, реєстрації, входу, роботи з кошиком, оформлення замовлення та перегляду особистого кабінету, а також окрема перевірка стійкості JWT-механізму до підміни та прострочення токена показали, що система поводиться передбачувано в межах визначеного функціоналу, а виявлені дрібні недоліки валідації було усунуто без зміни архітектурних рішень. Це підтверджує, що прийняті на попередніх етапах рішення виявилися узгодженими між собою та працездатними на практиці.

У підсумку можна стверджувати, що мету роботи, створення інтерактивної веб-платформи для замовлення адаптивного одягу для поранених військових із повним циклом розробки від архітектури до реалізації й тестування, було досягнуто, а результати кожного етапу логічно переходять у наступний, утворюючи завершений продукт. Практичне значення розробленого застосунку полягає у тому, що він може бути використаний як основа для реального сервісу замовлення адаптивного одягу, що дозволить централізувати обробку запитів, які наразі здійснюються переважно через розрізнені канали комунікації волонтерських ініціатив. Водночас закладені у проєкті архітектурні принципи, поділ на клієнтську, серверну частини та хмарну базу даних, а також система автентифікації на основі JWT, роблять реалізований код придатним для повторного використання як шаблону при створенні інших спеціалізованих платформ електронної торгівлі з аналогічними вимогами до каталогу товарів, кошика та особистого кабінету користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 10 порад, як обрати найкращий стек технологій для розробки веб-додатків у 2025 році. SECL Group. 2025. URL: https://secl.com.ua/tips-to-choose-tech-stack-for-web-app-development/ (дата звернення: 08.04.2026).
2. Адаптивний одяг для поранених військовослужбовців шиють волонтерки «Швейної роти». Varosh.com.ua. URL: https://varosh.com.ua/novyny/ adaptyvnyj-odyag-dlya-poranenyh-vijskovosluzhbovcziv-shyyut-volonterky-shvejnoyi-roty/ (дата звернення: 10.04.2026).
3. Міністерство оборони України. 50 тисяч одиниць адаптивного одягу передано до шпиталів. Листопад 2025. URL: https://www.mil.gov.ua/ (дата звернення: 10.04.2026).
4. Міністерство охорони здоров'я України. Медична реабілітація в умовах воєнного часу. URL: https://moz.gov.ua/ (дата звернення: 10.04.2026).
5. Що таке JavaScript. 22.06.2023. URL: https://cases.media/article/sho-take-javascript (дата звернення: 10.04.2026).
6. Швейна рота. Сторінка в Instagram. URL: https://www.instagram.com/shveina_rota/ (дата звернення: 10.04.2026).
7. Adaptive – офіційний інтернет-магазин адаптивного одягу. URL: https://adaptive.com.ua/ (дата звернення: 10.04.2026).
8. Azzurro A. SweetAlert2: a beautiful, responsive, customizable popup. URL: https://sweetalert2.github.io/ (дата звернення: 10.04.2026).
9. Banker K. et al. MongoDB in Action. 2nd ed. Manning Publications, 2016. 390 p.
10. Chodorow K. MongoDB: The Definitive Guide. 2nd ed. O'Reilly Media, 2013. 432 p.
11. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. Pearson, 2015. 1400 p.
12. Express.js. Fast, unopinionated, minimalist web framework for Node.js. URL: https://expressjs.com (дата звернення: 20.04.2026).
13. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : doctoral dissertation. University of California, Irvine, 2000. 162 p.

14. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002. 533 p.
15. Hubz.ua. Як в Україні створюють адаптивний одяг для поранених військових? URL: https://hubz.ua/ekspertna-dumka/zvernit-uvagu/yak-v-ukrayini-stvoryuyut-adaptyvnyj-odyag-dlya-poranenyh/ (дата звернення: 10.04.2026).
16. Krug S. Don’t Make Me Think, Revisited: A Common Sense Approach to Web Usability. 3rd ed. New Riders, 2014. 216 p.
17. MDN Web Docs. The web standards model. URL: https://developer.mozilla.org/enUS/docs/Learn_web_development/Getting_started/Web_standards/The_web_standards_model (дата звернення: 21.06.2026).
18. Nielsen J., Molich R. Heuristic evaluation of user interfaces. Proceedings of CHI’90. ACM, 1990. P. 249–256.
19. Norman D. The Design of Everyday Things. Rev. and expanded ed. Basic Books, 2013. 368 p.
20. Plus Plus. Адаптивний одяг для поранених військових. URL: https://plusplus.kyiv.ua/adaptive-clothing-wound (дата звернення: 10.04.2026).
21. Schwartz B., Zaitsev P., Tkachenko V. High Performance MySQL. 3rd ed. O’Reilly Media, 2012. 826 p.
22. Tilkov S., Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE Internet Computing. 2010. Vol. 14, № 6. P. 80–83.
23. W3C. Web Standards. URL: https://www.w3.org/standards/ (дата звернення: 10.04.2026).
24. Web Content Accessibility Guidelines (WCAG) 2.1. W3C Recommendation, 5 June 2018. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 10.04.2026).
25. Wheelny. Адаптивний одяг для людей на кріслах колісних. URL: https://wheelny.onepage.me/ (дата звернення: 10.04.2026).
26. Wilson J. Node.js 8 the Right Way. Pragmatic Bookshelf, 2018. 292 p.

ЗГОДА здобувачки вищої освіти

Державного університету економіки і технологій про
перевірку кваліфікаційної роботи на прояви
академічного плагіату
та розміщення в Репозитарії Університету

Я, Горбаль Євгенія Олександрівна,

підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота

Розробка інтерактивної веб-платформи для замовлення адаптивного одягу для поранених військових

виконана самостійно та не містить академічного плагіату. Я не надавала і не одержувала недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлена. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформована, що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлена.

15.06.26 р Горбаль Є. О.