Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-магазину та бота для оформлення замовлень на основі інтегрованого каталогу

Виконав(ла): студент(ка) 4 курсу, групи СП-43
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Пушкар В.С. (прізвище та ініціали)

Керівник (підпис) Михалик Д.М. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю. М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М. Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль 2025

АНОТАЦІЯ

Розробка веб-магазину та бота для оформлення замовлень на основі інтегрованого каталогу// Кваліфікаційна робота на здобуття освітнього рівня «Бакалавр»// Пушкар Віталій // Тернопільський національний технічний університет імені Івана Пулюя , група СП-43 // Тернопіль,2025 // Робота містить с.- 68 , рис. - 17 ,додат. - 3 .

Мета роботи - Розробка веб-магазину та бота для оформлення замовлень на основі інтегрованого каталогу, для забезпечення зручного доступу до інформації про товари через месенджер.

У першому розділі виконано аналіз предметної області, досліджено тенденції розвитку електронної комерції, значення автоматизації онлайн-продажів і можливості використання чат-ботів. Надано порівняння сучасних технологій для створення веб-магазинів і Telegram-ботів, обґрунтовано вибір інструментів реалізації проєкту (HTML, CSS, C#, Telegram.Bot, HtmlAgilityPack).

Другий розділ присвячений проєктуванню та реалізації системи. Розглянуто архітектуру інтегрованої інформаційної системи, описано структуру сайту з категоріями та картками товарів, а також функціональність Telegram-бота, який зчитує HTML-каталог і формує замовлення. Висвітлено логіку обробки команд, парсингу та зв’язку між компонентами системи без використання бази даних.

У третьому розділі представлено результати тестування проєкту. Проведено функціональне та нефункціональне тестування веб-магазину й Telegram-бота, оцінено їхню ефективність, зручність, стабільність і швидкодію. Здійснено порівняльний аналіз роботи сайту і бота, виявлено помилки та шляхи їх усунення. Сформовано підсумкову оцінку системи та окреслено перспективи її вдосконалення.

ABSTRACT

Development of a web store and a bot for placing orders based on an integrated catalog// Qualification work for obtaining the educational level "Bachelor"// Pushkar Vitaliy // Ivan Pulyuy Ternopil National Technical University, group SP-43 // Ternopil, 2025 // The work contains p.- 68 , fig. - 17 ,append. - 3 .

Purpose of the work - Development of a web store and a bot for placing orders based on an integrated catalog, to ensure convenient access to information about products via the messenger.

The first section analyzes the subject area, examines the trends in the development of e-commerce, the importance of online sales automation and the possibilities of using chat bots. A comparison of modern technologies for creating web stores and Telegram bots is provided, and the choice of project implementation tools is justified (HTML, CSS, C#, Telegram.Bot, HtmlAgilityPack).

The second section is devoted to the design and implementation of the system. The architecture of the integrated information system is considered, the structure of the site with categories and product cards is described, as well as the functionality of the Telegram bot, which reads the HTML catalog and forms an order. The logic of command processing, parsing and communication between system components without using a database is highlighted.

The third section presents the results of project testing. Functional and non-functional testing of the web store and Telegram bot was carried out, their efficiency, convenience, stability and speed were assessed. A comparative analysis of the site and bot operation was carried out, errors were identified and ways to eliminate them were identified. A final assessment of the system was formed and prospects for its improvement were outlined.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ,ОДИНИЦЬ,СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – інтерфейс прикладного програмування, набір функцій для взаємодії між програмами.

HTML (HyperText Markup Language) – мова розмітки гіпертексту, що використовується для створення структури веб-сторінок.

CSS (Cascading Style Sheets) – каскадні таблиці стилів, що використовуються для оформлення вигляду веб-сторінок.

C# – мова програмування, яка використовується для розробки серверної логіки, у тому числі Telegram-ботів.

Telegram – месенджер з відкритим API, що дозволяє розробникам створювати ботів та інтеграції.

Чат-бот – програма, яка імітує спілкування з користувачами через текстові або голосові повідомлення, автоматизуючи обслуговування.

HTML-каталог – набір веб-сторінок у форматі HTML, які містять структуровану інформацію про товари.

Telegram.Bot – .NET-бібліотека для розробки Telegram-ботів, що забезпечує зручний інтерфейс до Telegram Bot API.

HtmlAgilityPack – бібліотека .NET для обробки та парсингу HTML-документів.

Live Server – розширення для Visual Studio Code, яке дозволяє запускати сайт локально без розгортання серверу.

Bootstrap – фронтенд-фреймворк для розробки адаптивного дизайну веб-інтерфейсів.

UX (User Experience) – користувацький досвід, загальні враження користувача від взаємодії з інтерфейсом.

UI (User Interface) – інтерфейс користувача, елементи управління та відображення інформації на веб-сторінці.

DOM (Document Object Model) – об’єктна модель документа, структура HTML або XML-документа, що дозволяє доступ і модифікацію його вмісту через програмний код.

Адаптивний дизайн – спосіб створення веб-інтерфейсу, який коректно відображається на пристроях з різними розмірами екранів.

UX-тестування – процес оцінювання зручності, доступності та задоволеності користувача від продукту.

Фреймворк – набір бібліотек і інструментів, який полегшує розробку програмного забезпечення, надаючи готову структуру.

ЗМІСТ

ВСТУП

У сучасному цифровому середовищі електронна комерція відіграє ключову роль у формуванні споживчого досвіду. З поширенням високошвидкісного Інтернету, мобільних пристроїв та зростанням цифрової грамотності все більше компаній прагнуть забезпечити свою присутність в онлайн-просторі, використовуючи веб-магазини для реалізації товарів і послуг. Сучасний користувач очікує від інтернет-ресурсу не лише наявності актуальної інформації, але й високого рівня зручності, швидкості, інтерактивності та персоналізованого підходу до обслуговування. Саме ці фактори обумовлюють необхідність постійного вдосконалення цифрових сервісів.

Паралельно з цим, у бізнес-середовищі стрімко зростає попит на автоматизацію процесів обслуговування клієнтів. Основною метою такого підходу є зниження витрат часу та ресурсів на обробку запитів користувачів, підвищення швидкості реагування та оптимізація взаємодії з клієнтами. Особливої актуальності це набуває в умовах високої конкуренції на ринку електронної комерції, де навіть незначні затримки в обслуговуванні чи складність у навігації можуть призвести до втрати потенційного покупця.

Одним із найефективніших інструментів автоматизації обслуговування є чат-боти — програми, що імітують людське спілкування і здатні взаємодіяти з користувачами через текстові або голосові повідомлення. Вони інтегруються у популярні платформи (зокрема, Telegram, Viber, Facebook Messenger), що дозволяє забезпечити постійну доступність сервісу незалежно від часу доби та місця перебування користувача. Завдяки використанню чат-ботів можна не лише зменшити навантаження на операторів підтримки, але й значно розширити функціональність системи, надавши змогу користувачеві швидко знайти товар, отримати інформацію, здійснити замовлення чи дізнатися про акції.

Інтеграція чат-ботів із веб-магазинами відкриває нові можливості в організації ефективної взаємодії з користувачами. У процесі спілкування бот може надавати актуальну інформацію про товари, демонструвати візуальні елементи (зображення, ціни, описи), а також приймати та обробляти замовлення. Все це робить онлайн-шопінг більш динамічним, гнучким та орієнтованим на користувача.

Особливістю даного проєкту є реалізація єдиного HTML-каталогу товарів, з якого одночасно зчитують дані як веб-магазин, так і Telegram-бот. Такий підхід дозволяє уникнути необхідності дублювання інформації, зберігає її актуальність в обох компонентах системи та значно спрощує обслуговування структури сайту. Завдяки використанню бібліотеки HtmlAgilityPack в Telegram-боті, реалізовано зчитування структурованого HTML-коду з подальшою трансформацією його в повідомлення з товарними позиціями.

Для реалізації системи використано сучасні програмні інструменти та мови: HTML і CSS — для створення візуального інтерфейсу сайту, C# — для логіки Telegram-бота, а також Visual Studio Code і Visual Studio 2022 як основні середовища розробки. Локальне тестування і запуск сайту здійснюється через Live Server, що забезпечує оперативне внесення змін без потреби розгортання повноцінного серверного середовища.

Розроблена система охоплює одразу кілька напрямів сучасного програмного забезпечення: веб-розробку, інтеграцію з месенджерами, парсинг HTML-контенту, забезпечення зручності користувацького досвіду. У результаті користувач отримує можливість обирати зручний для себе спосіб взаємодії — або переглядати каталог і робити замовлення через сайт, або ж виконати ці ж дії у Telegram за допомогою бота.

Таким чином, дана кваліфікаційна робота має не лише прикладну, а й методичну цінність, адже демонструє, як можна об'єднати різні підходи у побудові доступної та функціональної інформаційної системи навіть без використання складних серверних рішень чи баз даних. Вона також показує, як мінімалістичні

технологічні рішення можуть ефективно задовольняти базові потреби бізнесу і користувачів.

Метою роботи є створення повноцінної інтегрованої інформаційної системи, що поєднує зручність веб-інтерфейсу з функціональністю чат-бота для оформлення замовлень, перегляду товарів та покращення клієнтського сервісу.

Об’єктом дослідження виступає інформаційна система, яка надає користувачеві можливість перегляду каталогу комп’ютерних товарів та оформлення замовлення як через сайт, так і через Telegram-бота.

У межах цієї роботи було поставлено та вирішено ряд завдань, серед яких:

- проведення аналізу предметної області та актуальності обраної теми;
- обґрунтування вибору інструментів для реалізації веб-магазину та Telegram-бота;
- проєктування та створення структури сайту з категоріями товарів;
- розробка Telegram-бота з підтримкою інтерактивного парсингу HTML-каталогу;
- забезпечення зв'язку між сайтом і ботом через спільний каталог;
- тестування системи на коректність роботи, зручність і стабільність;
- визначення напрямів удосконалення розробленої інформаційної системи.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Стан і тенденції розвитку електронної комерції

Упродовж останніх десяти років електронна комерція стала невід'ємною складовою світової економіки. Рівень поширеності інтернету, зростання цифрової грамотності населення, розвиток логістичних сервісів та безконтактних платіжних систем сприяли стрімкому розвитку ринку онлайн-продажів. На рисунку 1.1 можна побачити графіки які відповідаю за кількістю людей по всьому світу які використовують інтернет. За даними аналітичної платформи Statista, у 2025 році обсяг роздрібних продажів електронної комерції у всьому світі перевищить 4,3 трильйона доларів США, і очікується, що ця цифра досягне нових висот у найближчі роки. Протягом прогнозованого періоду доходу для всіх сегментів очікуються значні коливання. Загалом, показник демонструє позитивну тенденцію, причому до 2029 року більше сегментів демонструють зростаючі значення, а не зменшувальні. Серед них сегмент «Продукти харчування» досягає найвищого значення протягом усього періоду, сягнувши 5,89 трильйонів доларів США [1].

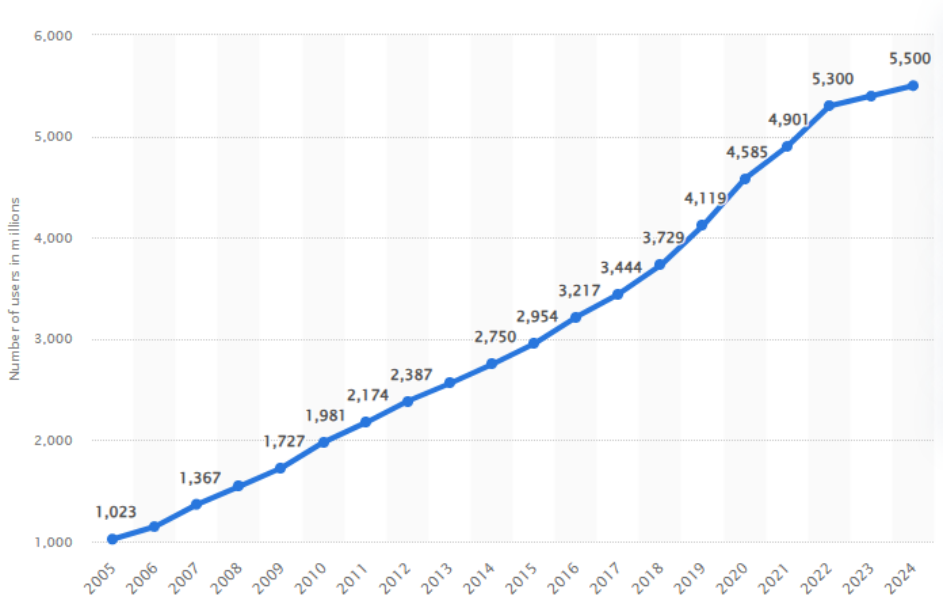

Рисунок 1.1 – Графік кількісті користувачів Інтернету у світі

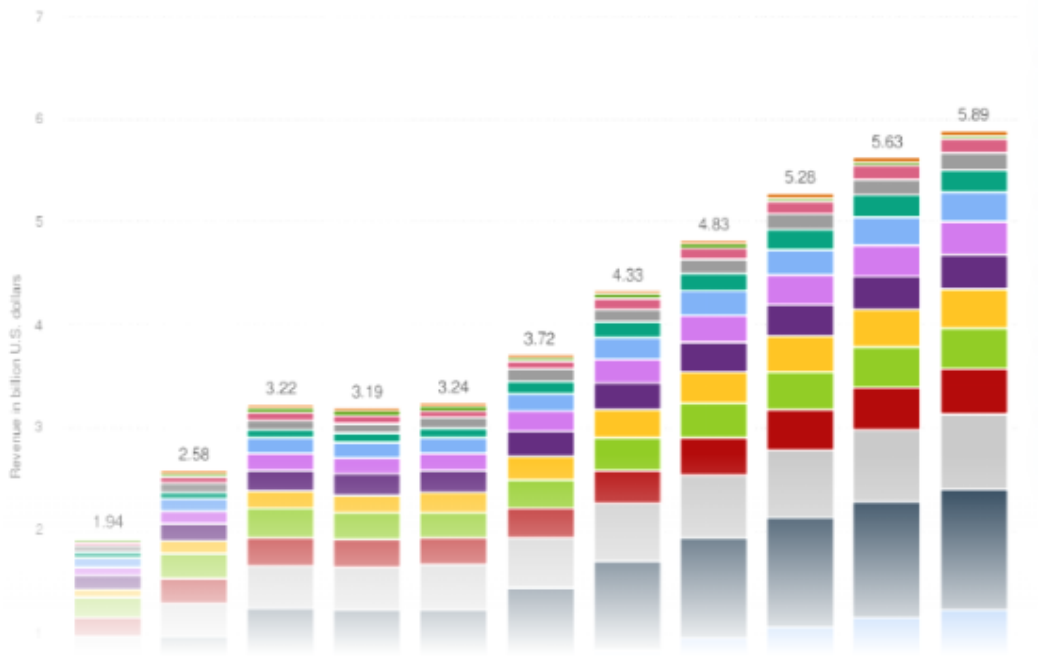

Рисунок 1.2 – Графік прогнозованого зросту продажів електронної комерції

Окремо варто відзначити розвиток мобільної торгівлі. Все більше користувачів здійснюють покупки через смартфони та планшети, що вимагає від розробників створення адаптивних інтерфейсів і мобільних рішень. У відповідь на ці виклики зростає попит на ефективні засоби автоматизації продажів і взаємодії з клієнтами.

В умовах високої конкуренції серед онлайн-магазинів ключову роль починає відігравати не лише асортимент і ціна, а й зручність сервісу, швидкість оформлення замовлень, наявність інтерактивної підтримки. Ці чинники формують підґрунтя для активного впровадження інтелектуальних систем, таких як чат-боти.

1.2 Значення автоматизації у сфері онлайн-продажів

Автоматизація є визначальним фактором ефективності сучасних електронних сервісів. У сфері електронної комерції вона охоплює низку процесів:

від обробки замовлень і виставлення рахунків до управління складом, комунікації з клієнтом та постпродажного обслуговування.

Одним із перспективних напрямів автоматизації є використання чат-ботів. Вони забезпечують швидке реагування на запити користувачів, можуть надавати консультації, підказувати категорії товарів, здійснювати попереднє оформлення замовлень, відправляти повідомлення про акції чи статус доставки. Згідно з аналітикою Verloop.io, понад 80 % бізнесів уже впровадили або планують впровадити чат-боти до 2025 року [2].

Особливо ефективними є чат-боти в екосистемі месенджерів. Telegram, як платформа з відкритим API, пропонує зручні інструменти для створення ботів [3]. Серед основних переваг Telegram-ботів: доступність, безкоштовність, мультиплатформеність, підтримка кнопок, меню та інтерактивних сценаріїв.

Завдяки таким можливостям, чат-боти можуть виступати повноцінним інтерфейсом між клієнтом та інформаційною системою магазину, зменшуючи навантаження на операторів та покращуючи взаємодію з користувачем.

1.3 Аналіз технологічних рішень для веб-магазинів та Telegram-ботів

Серед платформ для створення веб-магазинів умовно можна виділити три основні підходи:

1. CMS-платформи з модулями e-commerce (WordPress + WooCommerce, Joomla) — забезпечують більшу кастомізацію, проте вимагають навичок адміністрування та знання PHP [4].
2. SaaS-рішення (Shopify, Wix, BigCommerce) — швидкі в розгортанні, мають набір готових шаблонів, однак є менш гнучкими й потребують постійної оплати [5].

3. Власноручна розробка (HTML, CSS, JS, C#, інше) — дає максимальну гнучкість, дозволяє повністю контролювати логіку і зовнішній вигляд сайту, але потребує більше часу і знань [6].

Telegram-боти розробляються за допомогою Telegram Bot API [3]. Для .NET- розробників доступна популярна бібліотека Telegram.Bot, яка дозволяє працювати з повідомленнями, клавіатурами, inline-командами тощо. Щоб реалізувати зчитування інформації з HTML-сторінок, часто застосовується бібліотека HtmlAgilityPack [7]. Вона дає змогу парсити DOM структуру сайту, знаходити потрібні елементи за класами, тегами або ID.

Для реалізації веб-магазину та чат-бота на початковому рівні найкращим вибором є використання HTML/CSS для інтерфейсу, C# для серверної логіки Telegram-бота [8], а також зв’язок через HtmlAgilityPack [7]. Такий підхід забезпечує зрозумілу архітектуру та легкість у підтримці.

1.4 Формування цілей, завдань і вибір інструментів реалізації

За мету поставлено розробити простий, але функціональний веб-магазин комп’ютерних товарів з інтегрованим Telegram-ботом для перегляду товарів і оформлення замовлень.

Цілі проєкту:

- спростити доступ до товарів як через веб-інтерфейс, так і через мобільний месенджер;
- забезпечити синхронізацію між сайт-каталогом і Telegram-ботом;
- забезпечити просту структуру, зрозумілу для користувачів і розробників-початківців.

Основні інструменти реалізації:

- HTML/CSS — створення структури та стилю сайту [6];
- C# (Telegram.Bot) — логіка взаємодії бота з користувачем [7];

- HtmlAgilityPack — зчитування товарів із HTML-сторінок [8];
- Visual Studio Code — середовище розробки для сайту [9];
- Visual Studio 2022 — середовище для Telegram-бота на C# [10];
- Live Server — локальний запуск сайту без розгортання сервера.

Таким чином, вибрані інструменти дозволяють реалізувати повноцінний інтегрований проєкт, що поєднує інтерфейсну частину та автоматизованого помічника.

2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Архітектура системи

Інформаційна система, що розробляється в межах цієї кваліфікаційної роботи, складається з двох основних модулів — веб-магазину та Telegram-бот [3]. Їх взаємодія відбувається через спільну HTML-структуру каталогу товарів, яку бот може зчитувати за допомогою бібліотеки HtmlAgilityPack [8].

Кожен із модулів виконує свою функціональну роль:

- Веб-магазин - забезпечує візуальне представлення товарів, категоризацію та базову навігацію для користувача;
- Telegram-бот - реалізує можливість перегляду товарів у месенджері та швидкого оформлення замовлення.

Принцип взаємодії:

- Веб-сайт зберігає каталог товарів у форматі HTML-сторінок з єдиною структурою.
- Telegram-бот зчитує ці HTML-сторінки за допомогою HtmlAgilityPack, парсить необхідні елементи (зображення, назву, ціну) та формує з них повідомлення для користувача.
- Оскільки структура HTML є сталою, обидва компоненти (сайт і бот) використовують один і той самий набір файлів, що дозволяє уникнути дублювання даних.

Перевага такої архітектури — у відсутності потреби в окремій базі даних. Веб-сайт та бот працюють із єдиним джерелом — HTML-документами каталогу, що спрощує реалізацію та синхронізацію.

2.2 Структура веб-магазину

Веб-магазин створено за допомогою мов HTML та CSS із використанням фреймворку Bootstrap [11] для адаптивної верстки. Уся інформація про товари структурована у вигляді карток, яку можна побачити на рисунку 2.2, що містять:

- зображення товару (одного розміру для збереження візуальної єдності);
- назву товару;
- ціну;
- кнопку "Buy", що веде на сторінку оформлення або Telegram-бота.

```
<div class="card" style="width: 20rem; margin-left: 25%;">
    <img src="../images/ps1.jpg" class="card-img-top imagess" alt="...">
    <div class="card-body">
      <p class="card-text">be quiet! Pure Power 12 M 750Вт</p>
      <p class="card-text">5500 грн</p>
        <a href="login.html" class="btn btn-dark" style="margin-left: 50%; transform: translate(-50%);">Buy</a>
    </div>
    </div>
```

Рисунок 2.2 – Код картки товару

Каталог поділений на категорії, кожна з яких представлена окремим HTML- файлом (рис 2.3). Всі ці файли розміщено в директорії /pages, а зображення — у /images.

```
<li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle text-white" href="#" role="button"
        data-bs-toggle="dropdown" aria-expanded="false">
        Catalog
    </a>
    <ul class="dropdown-menu">
        <li><a class="dropdown-item" href="builds.html">Pre-built pc</a></li>
        <li><a class="dropdown-item" href="cpus.html">CPUs</a></li>
        <li><a class="dropdown-item" href="gpus.html">GPUs</a></li>
        <li><a class="dropdown-item" href="motherboards.html">Motherboards</a></li>
        <li><a class="dropdown-item" href="memory.html">Memory</a></li>
        <li><a class="dropdown-item" href="storage.html">Storage</a></li>
        <li><a class="dropdown-item" href="powersupplies.html">Power Supplies</a></li>
        <li><a class="dropdown-item" href="cases.html">Cases</a></li>
    </ul>
</li>
```

Рисунок 2.3 – Код каталогу

Сторінки пов’язані між собою навігаційним меню. У футері сайту також вказано контактну інформацію. Код сторінок запускається локально через Live Server у середовищі Visual Studio Code [12]. Це дозволяє швидко переглядати зміни без розгортання на хостингу.

Візуальний стиль сайту виконано в світлих тонах, шрифт — poppins. Основна увага зосереджена на доступності товарів, простоті інтерфейсу.

2.3 Telegram-бот: функції та логіка

Telegram-бот розроблений мовою C# у середовищі Visual Studio 2022 [10] з використанням бібліотеки Telegram.Bot [13]. Основні функції реалізовано у вигляді обробників команд і повідомлень. Логіка побудована наступним чином:

Основні функції бота:

- /start — запуск вітального повідомлення з поясненням можливостей.
- /catalog або "Каталог" — відображення списку категорій.
- Обробка вибору категорії — зчитування HTML-файлу й виведення карток товарів.
- Обробка кнопки "Купити" — формування повідомлення-підтвердження замовлення.
- Обробка замовлення — послідовне збирання логіну, пароля, номера телефону користувача.

Алгоритм роботи бота:

1. Користувач надсилає команду /start
    - Бот відправляє привітальне повідомлення з клавіатурою.
2. Користувач натискає кнопку "Каталог" або вводить команду /catalog
    - Бот формує меню категорій (згідно з локальними HTML-файлами).

3. Користувач обирає категорію
    - Бот зчитує відповідний HTML-файл.
    - Визначає всі блоки .card, витягує назву, зображення, ціну.
    - Формує повідомлення з товаром і кнопкою "Купити".
4. Користувач натискає кнопку "Купити"
    - Бот переходить до етапу оформлення замовлення.
5. Послідовна обробка введених даних
    - Запит логіну → зберігається.
    - Запит паролю → зберігається.
    - Запит телефону → зберігається.
    - Бот надсилає підтвердження замовлення із зведеною інформацією.

Такий підхід дозволяє забезпечити стабільність, масштабованість і зручність Telegram-бота навіть без використання бази даних або зовнішнього API.

2.4 Інтеграція HTML-каталогу та Telegram-бота

Ключовим елементом у роботі є зв’язок між HTML-файлами сайту та Telegram-ботом. Цей зв’язок реалізується за допомогою HtmlAgilityPack — бібліотеки для аналізу HTML-документів у C#.

Бот завантажує HTML-файл, розташований у локальній директорії, і знаходить усі блоки з класом card. У межах цих блоків він витягує дані про:

- заголовок товару (<p class="card-text">),
- ціну (другий елемент card-text),
- зображення (<img src=...>).

Ці елементи перетворюються на повідомлення, які бот надсилає в Telegram. Для підтримки зручності розмітка HTML має бути сталою, із фіксованими класами. У випадку оновлення каталогу достатньо змінити HTML-файли — бот автоматично працюватиме з оновленими даними без зміни коду.

Такий підхід дозволяє забезпечити гнучку інтеграцію без складної серверної частини або бази даних. Система залишається простою, прозорою та ефективною для локального використання або як демонстраційний проєкт.

3 ТЕСТУВАННЯ, РЕЗУЛЬТАТИ ТА ОЦІНКА ЕФЕКТИВНОСТІ

3.1 Методологія тестування системи

Для перевірки працездатності інформаційної системи, що складається з веб-магазину та Telegram-бота, було розроблено набір тестів, які охоплюють усі функціональні модулі. Основна мета тестування — виявити технічні помилки, оцінити стабільність парсингу HTML, перевірити взаємодію між компонентами та оцінити зручність для кінцевого користувача.

У процесі тестування були визначені наступні цілі:

- Перевірити функціональність усіх ключових модулів системи.
- Виявити помилки, що виникають під час взаємодії з користувачем.
- Визначити стабільність роботи системи при різному навантаженні.
- Виміряти швидкодію бот-системи та веб-інтерфейсу.
- Перевірити коректність обміну даними між ботом і HTML-каталогом.

Тестування проводилось у локальному середовищі розробки. Сценарії поділялись на функціональні (перевірка відповідей бота, логіки переходів, запитів) та нефункціональні (продуктивність, відгук, помилки при парсингу або втраті зв'язку). Використовувались такі інструменти, як Chrome DevTools для візуального контролю верстки та Telegram Desktop для перевірки UX взаємодії з ботом.

Кожен тестовий сценарій документувався: вводились вхідні дані, реєструвався очікуваний результат і порівнювався з фактичним. У результаті створено набір кейсів, які демонструють надійність роботи системи в типових і граничних умовах.

3.2 Тестування веб-магазину

Основне тестування веб-магазину полягало у перевірці коректності завантаження сторінок, візуального представлення товарів та адаптивності інтерфейсу для різних пристроїв. Було перевірено відповідність верстки стандартам HTML/CSS та її поведінку на різних розширеннях екрана.

3.2.1 Завантаження головної сторінки

Мета: перевірити швидкість завантаження та відображення контенту на головній сторінці сайту.

Умови: локальний запуск сайту через Live Server у VS Code. Очікується коректне завантаження логотипу, навігаційного меню, категорій товарів.

Результат: сайт завантажується за <1 секунду; всі основні елементи присутні, візуальні помилок не виявлено їх можна побачити на рисунку 3.1.

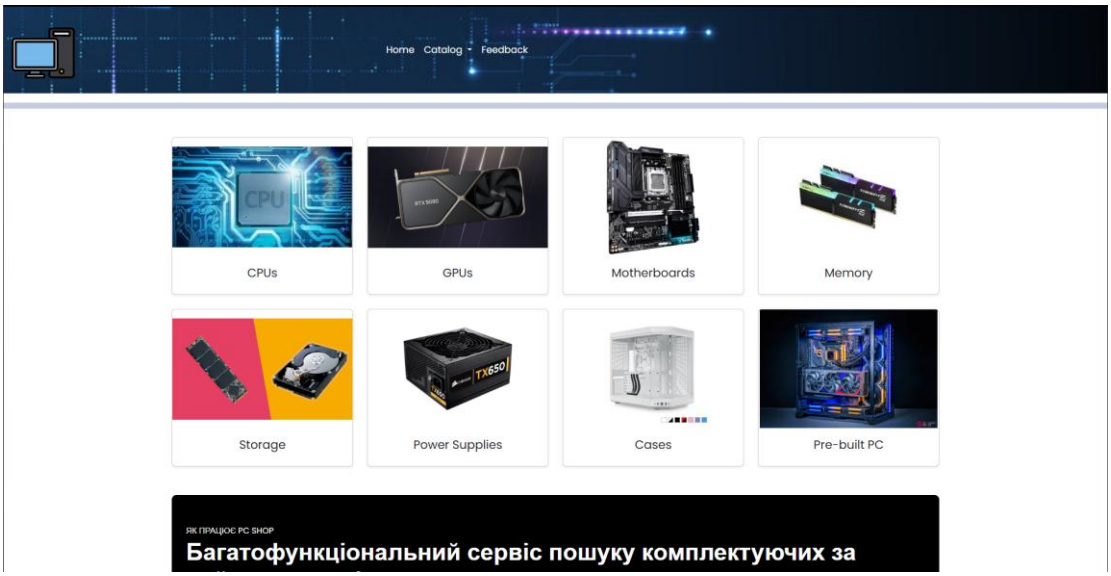

Рисунок 3.1 – Головна сторінка сайту.

3.2.2 Навігація між категоріями

Мета: оцінити функціональність посилань між сторінками категорій товарів.

Умови: при натисканні на пункт меню в каталозі користувач переходить до відповідного HTML-файлу з картками товарів (рис 3.2).

Результат: усі переходи здійснюються миттєво, відсутні помилки 404 або порожні сторінки. Весь вміст підвантажується без артефактів (рис 3.3).

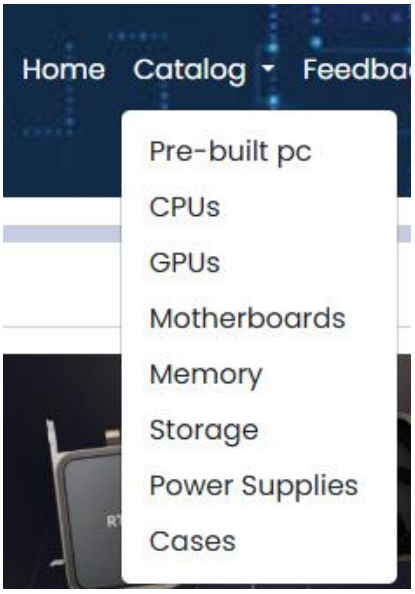

Рисунок 3.2 – Каталог

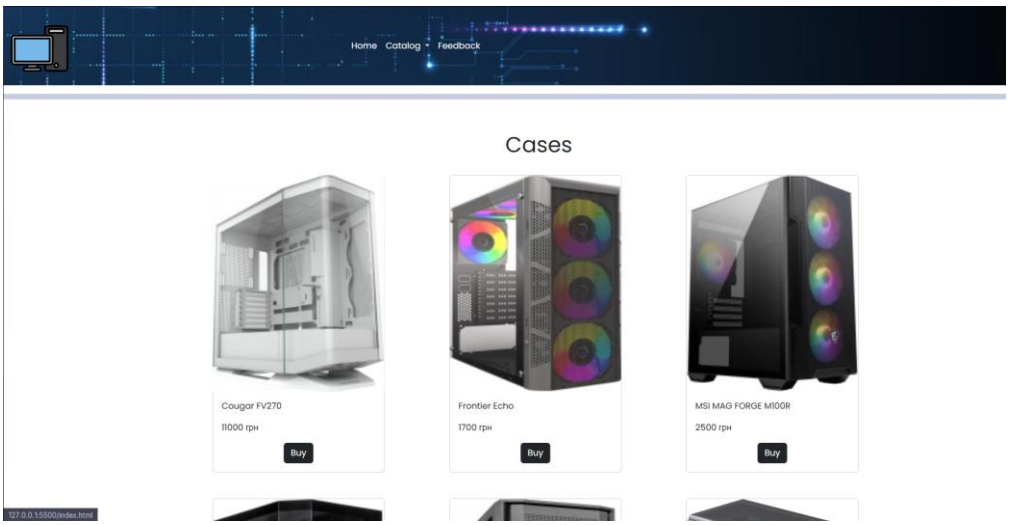

Рисунок 3.3 – Сторінка корпусів.

3.2.3 Відображення карток товарів

Мета: перевірити правильність відображення зображення, назви та ціни товару на картці (рис 3.4).

```
<div class="container">
    <div class="row">
        <div class="col">
            <div class="card" style="width: 20rem; margin-left: 25%;">
                <img src="../images/pc1.jpg" class="card-img-top imagess" alt="...">
                <div class="card-body">
                  <p class="card-text">Artline Gaming Tank</p>
                  <p class="card-text">Проц:Core i5, 14400F</p>
                  <p class="card-text">ОЗП:32 ГБ</p>
                  <p class="card-text">Відео:RTX 4070</p>
                  <p class="card-text">Накопичувач:1 ТБ</p>
                  <p class="card-text">67000 грн</p>
                    <a href="login.html" class="btn btn-dark" style="margin-left: 50%; transform: translate(-50%);">Buy</a>
                </div>
                </div>
        </div>
        <div class="col">
            <div class="card" style="width: 20rem; margin-left: 25%;">
                <img src="../images/pc2.jpg" class="card-img-top imagess" alt="...">
                <div class="card-body">
                  <p class="card-text">Artline Gaming X47</p>
                  <p class="card-text">Проц:Ryzen 5, 3600</p>
                  <p class="card-text">ОЗП:32 ГБ</p>
                  <p class="card-text">Відео:RTX 4060</p>
                  <p class="card-text">Накопичувач:1 ТБ</p>
                  <p class="card-text">39000 грн</p>
                    <a href="login.html" class="btn btn-dark" style="margin-left: 50%; transform: translate(-50%);">Buy</a>
                </div>
              </div>
        </div>
        <div class="col">
            <div class="card" style="width: 20rem; margin-left: 25%;">
                <img src="../images/pc3.jpg" class="card-img-top imagess" alt="...">
                <div class="card-body">
                  <p class="card-text">Apple Mac mini</p>
                  <p class="card-text">Проц:Apple M4</p>
                  <p class="card-text">ОЗП:24 ГБ</p>
                  <p class="card-text">Відео:M4 10-Core</p>
                  <p class="card-text">Накопичувач:512 ГБ</p>
                  <p class="card-text">45000 грн</p>
                    <a href="login.html" class="btn btn-dark" style="margin-left: 50%; transform: translate(-50%);">Buy</a>
                </div>
              </div>
        </div>
```

Рисунок 3.4 – Код картки товару

Результат: усі картки мають однакову висоту та ширину, стилі не порушуються. Зображення не обрізані, текст поміщається в межах контейнера (рис 3.5).

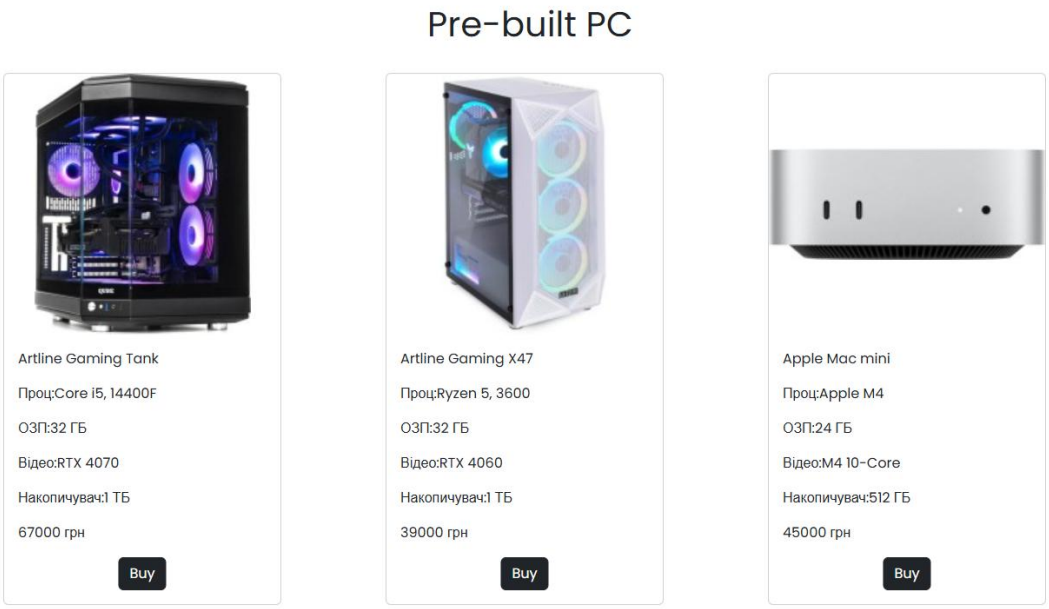

Рисунок 3.5 – Картки товару

3.2.4 Реакція на відсутність даних

Мета: перевірити поведінку сайту при відсутності даних — наприклад, коли немає ціни або зображення товару.

Умови: вручну видалено <img> або <p> з тестової картки.

Результат: при відсутності зображення — зберігається місце під нього (рис 3.6). При відсутності ціни — картка виглядає неактивною (рис 3.7). Це відповідає очікуваному поведінковому дизайну.

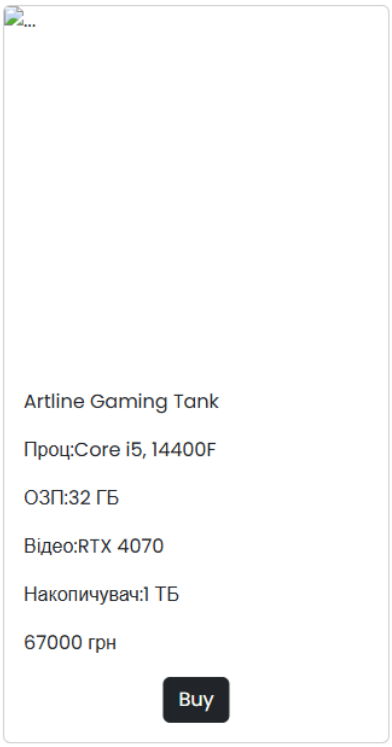

Рисунок 3.6 – Видалено зображення з картки товару

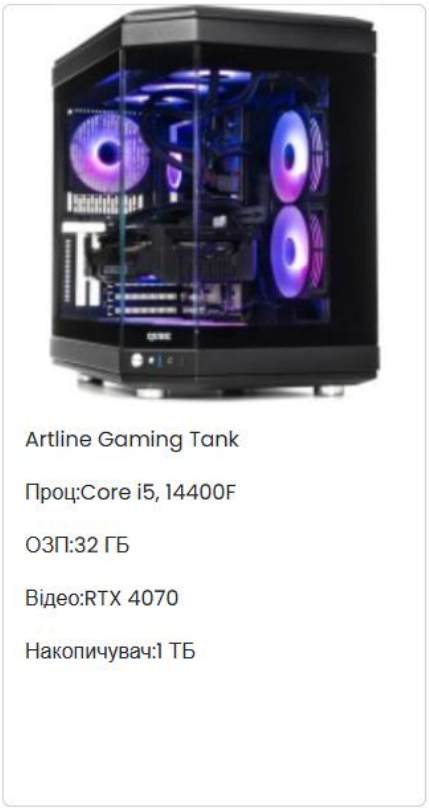

Рисунок 3.7 – Видалено ціну з картки товару.

3.3 Тестування Telegram-бота

Telegram-бот був протестований на обробку основних сценаріїв використання, включаючи наступні етапи.

На першому етапі після запуску команди /start бот ініціалізує сесію взаємодії з користувачем. У відповідь він надсилає повідомлення з клавіатурою, де з’являється кнопка "Каталог". Це дозволяє користувачу одразу перейти до перегляду товарів, не вводячи додаткові команди (рис.3.8).

Рисунок 3.8 – Інтерфейс бота

Після натискання на кнопку "Каталог" бот формує меню з категоріями товарів, які відповідають різним HTML-сторінкам у локальному каталозі. Кожна кнопка веде до завантаження HTML-файлу, що містить відповідну підбірку карток товарів, їх можна побачити на рисунку 3.9.

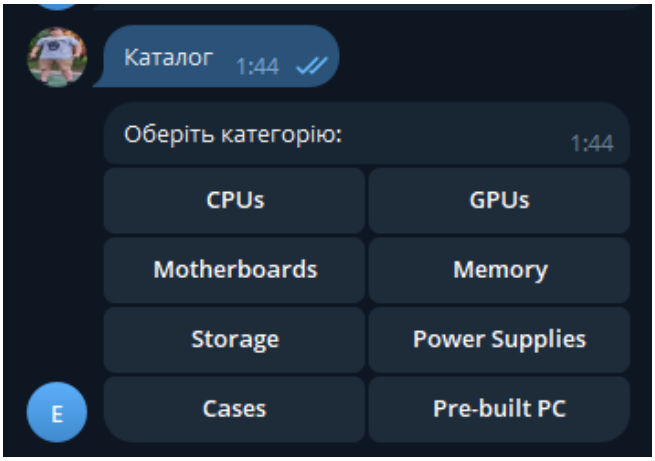

Рисунок 3.9 – Функціонал кнопки каталог

Коли користувач обирає одну з категорій (рис.3.9), бот використовує бібліотеку HtmlAgilityPack для зчитування HTML-структури. Він шукає блоки з товарами, витягує назви, ціни та зображення. На рисунку 3.10 можна ознайомитися за яким принципом відбувається зчитування HTML-структури.

```
1 reference
private static List<Product> ParseProducts(string file)
{
    var list = new List<Product>();
    var doc = new HtmlDocument();
    doc.Load(file);
    foreach (var card in doc.DocumentNode.SelectNodes("//div[contains(@class,'card')]") ?? Enumerable.Empty<HtmlNode>())
    {
        var img = card.SelectSingleNode(".//img")?.GetAttributeValue("src", "") ?? "";
        var pTags = card.SelectNodes(".//p[contains(@class,'card-text')]")?.ToList() ?? new();
        if (pTags.Count < 2) continue;
        var title = pTags.First().InnerText.Trim();
        var price = pTags.Last().InnerText.Trim();
        var desc = pTags.Count > 2 ? string.Join("\n", pTags.Skip(1).Take(pTags.Count - 2).Select(t => t.InnerText.Trim())) : string.Empty;
        list.Add(new Product { Title = title, Price = price, Description = desc, ImageRelative = img });
    }
    return list;
}
```

Рисунок 3.10 – Код для зчитування HTML-структури

Якщо хоча б один з елементів відсутній, бот застосовує додаткову перевірку на наявність необхідних тегів.

Отримавши валідні дані, бот формує повідомлення, в якому зазначається назва товару, його ціна та додається кнопка "Купити". З виглядом повідомлення можна ознайомитися на рисунку 3.11.

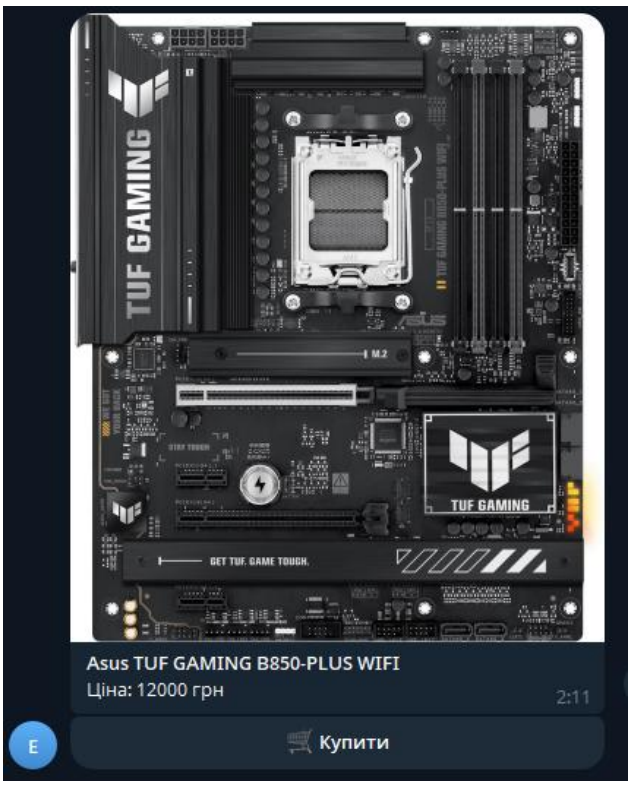

Рисунок 3.11 – Вигляд повідомлення бота

Ця кнопка дозволяє користувачеві швидко сформувати запит на придбання. Формат повідомлення є уніфікованим, що підвищує зручність користування ботом. Зразок сформованого запиту на придбання можна побачити на рисунку 3.12.

Рисунок 3.12 – Сформоване замовлення

У випадках, коли структура HTML є некоректною або не містить жодного товару, бот інформує про відсутність позицій у вибраній категорії. Також реалізована перевірка на наявність помилок, пов’язаних із неправильними або пустими тегами, приклад коду показаний на рисунку 3.13. Таким чином, користувач завжди отримує зрозумілий результат.

```
var path = System.IO.Path.Combine(SHOP_ROOT, htmlFile);
if (!System.IO.File.Exists(path))
{
    await _bot.SendTextMessageAsync(chat, "Файл категорії не знайдено.", cancellationToken: ct);
    return;
}

var prods = ParseProducts(path);
if (prods.Count == 0)
{
    await _bot.SendTextMessageAsync(chat, "У цій категорії немає товарів.", cancellationToken: ct);
    return;
}
```

Рисунок 3.13 – Код перевірки на наявність помилок

3.4 Порівняльний аналіз ефективності

Для об’єктивного порівняння функціоналу було проведено аналіз ключових аспектів зручності використання, швидкодії та стабільності сайту і Telegram-бота. Основна мета полягає у виявленні сильних та слабких сторін кожного з компонентів системи, а також у формуванні рекомендацій для покращення функціональності в майбутньому.

Було виділено три ключові критерії порівняння: час відгуку, зручність інтерфейсу та обсяг необхідних дій для формування замовлення.

Час відгуку:

- Сайт: середній час переходу між сторінками — 0,7 с. Це дозволяє забезпечити відносно швидкий доступ до товарів, проте швидкість може знижуватись при великій кількості одночасних запитів.

- Бот: середній час відповіді після натискання кнопки — 1,1 с. Незважаючи на дещо більший час, Telegram-бот працює стабільно і з фіксованою затримкою, що не перевищує 1.5 с навіть при серії запитів.

Зручність інтерфейсу:

- Сайт дозволяє швидко переглядати товари, проте потребує більше кліків для навігації, а також уваги до візуального контенту (прокручування сторінки, навігаційне меню, кнопки «купити»).
- Бот є оптимальним для мобільних пристроїв, навігація виконується за допомогою інлайн-клавіатури, що значно спрощує інтерфейс. Весь функціонал доступний у діалоговому вікні без потреби завантаження сторонніх ресурсів.

Обсяг дій:

- Сайт: для оформлення замовлення користувач повинен відкрити товар, перейти на сторінку входу, авторизуватись, повернутись до товару та натиснути кнопку покупки. Це може забрати кілька хвилин.
- Telegram-бот: достатньо натиснути «Каталог», обрати товар і натиснути кнопку «Купити». Формується запит на замовлення, який може бути миттєво підтверджений. Весь процес триває до 15–20 секунд.

Додатковими перевагами бота можна вважати, що бот як інструмент взаємодії з користувачем демонструє ефективність завдяки своїй простоті, доступності та здатності до інтеграції з іншими сервісами. Його функціонал забезпечує легке використання незалежно від технічних навичок користувача.

- Підвищена мобільність.
- Інтеграція із хмарними сервісами Telegram.
- Можливість легкої масштабованості через API.

Додатковими перевагами веб-сайту є більш традиційним каналом взаємодії з клієнтами та дозволяє створювати багатофункціональні рішення з широкими

можливостями кастомізації. Він надає розширені можливості представлення товарів та брендування.

- Більша гнучкість у дизайні.
- Можливість демонстрації візуального брендингу.
- Зручність SEO-просування для пошукових систем.

Telegram-бот найкраще підходить для здійснення швидких покупок і забезпечення максимально простої взаємодії з користувачем, завдяки своїй компактності та інтуїтивному інтерфейсу. Водночас веб-сайт є зручнішим для детального ознайомлення з широким асортиментом товарів і має кращі можливості для представлення інформації у візуально привабливому вигляді. Комбіноване використання обох інструментів дозволяє ефективно охопити потреби різних типів користувачів, забезпечуючи як глибоку навігацію по товарах, так і швидкий доступ до покупки [14].

3.5 Помилки та усунення

У процесі тестування були виявлені такі типові помилки:

- Проблеми з адаптивністю сайту: при зменшенні ширини екрану окремі блоки карток товарів накладалися один на одного.
- Відсутність fallback-логіки при помилках структури HTML: у разі видалення або пошкодження елементів <img>, <p> парсер HTML повертав порожні об'єкти, що призводило до збоїв відображення карток у Telegram-боті.
- Telegram-бот дублював повідомлення при швидкому натисканні кнопок: через відсутність захисту від багаторазових запитів користувача бот надсилав ідентичні повідомлення кілька разів.

- Неправильне кодування повідомлень: при відображенні символів кирилиці в деяких системах виникали помилки кодування (UTF-8 vs Windows-1251) [15].

Прийняті рішення для усунення помилок:

- Додано медіа-запити CSS (@media) для адаптації карток під екрани меншої ширини [16].
- Впроваджено логіку перевірки валідності HTML-елементів через HtmlAgilityPack, додано перевірку: if (element != null && element.InnerText != null) [17,18].
- Встановлено затримку між запитами користувача до Telegram-бота (Debounce ~700 мс).
- Всі вихідні повідомлення Telegram були явно закодовані у UTF-8 перед відправкою.

Усі перелічені дії дозволили усунути основні проблеми взаємодії, підвищити стабільність системи та забезпечити однакову якість роботи на різних пристроях і в різних умовах використання [19].

3.6 Підсумкова оцінка системи

Після проходження усіх етапів тестування було сформульовано загальну оцінку реалізованої системи:

- Веб-магазин виконує базові функції електронної торгівлі: демонстрацію товарів, адаптивне відображення, підтримку категорій. Його структура є простою, але придатною до подальшого розширення (наприклад, підключення бази даних або фреймворку на кшталт React).
- Telegram-бот реалізований як асинхронна система взаємодії з користувачем, що забезпечує швидке отримання інформації, просте

замовлення та можливість гнучкого масштабування. Бот показав стабільну роботу з HTML-файлами будь-якої складності (за умови наявності валідної структури).

- Інтеграція компонентів виконана у форматі “парсинг HTML → формування повідомлення → реакція користувача”. Замість бази даних використовується локальний каталог сторінок, що значно спрощує розгортання системи на будь-якому середовищі без потреби в сервері чи хостингу.

Загалом, система показала себе як ефективна модель яка успішно виконує поставлені завдання, має просту архітектуру, зрозумілу логіку, а також високу адаптивність до нових вимог [20]. У майбутньому її можна вдосконалити за рахунок додавання:

- системи авторизації користувачів,
- збереження замовлень до бази даних,
- інтеграції платіжних сервісів (наприклад, LiqPay, Stripe),
- розміщення на хостингу з використанням серверної частини (наприклад, ASP.NET Core або Node.js).

Система є повністю працездатною, зручною для тестового використання і придатною для подальшого масштабування.

4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Психологічні чинники небезпеки

Визначення терміна «безпека» міститься в усіх тлумачних словниках, де його розглядають його передусім як складне багатостороннє явище. В енциклопедичному словнику безпека (особиста або майнова) розглядається як головний аспект людського розвитку, причина виникнення та сенс існування держави. «Безпека створюється для попередження небезпек, які можуть загрожувати як окремим громадянам, так і суспільству та державі в цілому. Забезпечується безпека шляхом ужиття цілої низки заходів» [21].

Психологічні чинники небезпеки – це сукупність внутрішніх психоемоційних станів людини, її реакцій, особистісних особливостей, а також умов праці, що впливають на зниження уваги, зосередженості, здатності адекватно сприймати інформацію та приймати рішення. Такі чинники можуть істотно підвищувати ризик виникнення небезпечних ситуацій на виробництві, в транспорті, військовій справі, охороні праці, а також у повсякденному житті.

Основні психологічні чинники небезпеки: емоційне перенапруження та стрес, фізичне та психічне перевтомлення, монотонність та рутина, психоемоційна нестабільність, недостатній рівень психологічної підготовки, відволікання уваги, психологічна несумісність у колективі

Емоційне перенапруження, стрес та сильні емоційні навантаження можуть знижувати здатність людини до логічного мислення, уповільнювати реакції та спричиняти помилки. Хронічний стрес може призвести до виснаження нервової системи та професійного вигорання.

Фізичне та психічне перевтомлення - втомлена людина має знижену концентрацію уваги, повільнішу реакцію та підвищену ймовірність припуститися помилки. Особливо небезпечна втома при керуванні транспортними засобами, у роботі з механізмами чи на висоті.

Монотонність та рутина - виконання одноманітної роботи протягом тривалого часу спричиняє зниження інтересу до діяльності, сонливість, втрату зосередженості. Це підвищує ймовірність небезпечних дій або бездіяльності.

Психоемоційна нестабільність - індивідуальні особливості такі як дратівливість, імпульсивність, підвищена тривожність – можуть стати передумовами до неправильних рішень у критичних ситуаціях.

Недостатній рівень психологічної підготовки, якщо працівник не підготовлений до роботи в умовах ризику або підвищеної відповідальності, це може проявитися в неадекватних реакціях на стрес або небезпеку.

Відволікання уваги - зовнішні подразники (шум, сторонні розмови, погані умови праці) можуть призвести до того, що людина не помічає небезпеки або реагує занадто пізно.

Психологічна несумісність у колективі - конфлікти, недовіра між колегами, поганий психологічний клімат на робочому місці знижують загальний рівень безпеки праці, впливають на мотивацію та підвищують ризик порушень техніки безпеки.

Наявність перелічених чинників не лише створює дискомфорт для працівника, але й безпосередньо впливає на ймовірність виникнення аварій, нещасних випадків та помилок. В екстремальних ситуаціях – наприклад, у надзвичайних умовах або при управлінні небезпечними об'єктами – навіть незначні психологічні відхилення можуть мати фатальні наслідки.

Працівнику не може пропонуватися робота, яка за медичним висновком протипоказана йому за станом здоров'я. До виконання робіт підвищеної небезпеки та тих, що потребують професійного добору, допускаються особи за наявності висновку психофізіологічної експертизи [22].

Психологічні чинники небезпеки є критично важливою складовою загальної системи безпеки праці та життєдіяльності. Їх ігнорування може мати серйозні наслідки, тоді як своєчасна профілактика та контроль дозволяють значно підвищити ефективність роботи, знизити рівень аварійності та покращити загальне психічне здоров’я працівників.

4.2 Заходи пожежної безпеки

Пожежна безпека — це стан захищеності людей, матеріальних цінностей та довкілля від пожеж. Основна мета заходів пожежної безпеки — запобігти виникненню пожежі, обмежити її розповсюдження, а також забезпечити ефективне реагування на випадок займання. Ці заходи охоплюють як технічні, так і організаційні дії на всіх етапах діяльності підприємства, установи чи об’єкта.

Пожежна безпека повинна забезпечуватися шляхом проведення організаційних заходів та технічних засобів, спрямованих на запобігання пожежам, забезпечення безпеки людей, зниження можливих майнових втрат і зменшення негативних екологічних наслідків у разі їх виникнення, створення умов для успішного гасіння пожеж.

Профілактичні заходи:

1. Дотримання правил пожежної безпеки
    - Заборона використання відкритого вогню у заборонених місцях.
    - Забезпечення правильної експлуатації електроприладів, обігрівачів, газового обладнання.
    - Встановлення заборон на паління у виробничих і вибухонебезпечних зонах.
2. Вогнестійкість будівель і конструкцій
    - Використання негорючих або важкогорючих матеріалів.
    - Обробка дерев’яних конструкцій вогнезахисними розчинами.
    - Монтаж протипожежних дверей, перегородок, клапанів.
3. Профілактичні інструктажі та навчання
    - Первинний, повторний, позаплановий інструктажі для персоналу.
    - Навчання діям при пожежі та евакуації.

Технічні засоби пожежогасіння та виявлення пожежі:

1. Системи виявлення та оповіщення
    - Автоматичні пожежні сигналізації (АСПС)..
    - Системи димовидалення, вентиляція.
    - Оповіщення людей про пожежу звуковими/світловими сигналами.
2. Засоби пожежогасіння
    - Вогнегасники (вуглекислотні, порошкові, пінні).
    - Внутрішні пожежні крани з рукавами та стволами.
    - Системи автоматичного пожежогасіння (спринклерні, дренчерні).
3. Стан протипожежного водопостачання
    - Резервуари з водою, гідранти, насоси.
    - Розміщення схем водопостачання для пожежників.

Порушення вимог пожежної безпеки тягне за собою адміністративну або кримінальну відповідальність згідно з чинним законодавством України, Кодекс України про адміністративні правопорушення, ст. 175, або Кримінальний кодекс, ст. 270 [23,24].

Заходи пожежної безпеки — це багаторівнева система, яка вимагає постійного контролю, навчання персоналу та дотримання нормативних вимог. Комплексне впровадження профілактичних, технічних та організаційних заходів дозволяє суттєво зменшити ризик виникнення пожеж, зберегти життя людей та матеріальні ресурси.

ВИСНОВКИ

У межах цієї кваліфікаційної роботи було здійснено повний цикл розробки інтегрованої інформаційної системи, що поєднує функціональність веб-магазину та Telegram-бота для оформлення замовлень на основі спільного HTML-каталогу. У ході дослідження було розглянуто актуальні аспекти електронної комерції, роль автоматизації у взаємодії з клієнтами, а також технологічні засоби реалізації таких систем.

Протягом розробки цієї кваліфікаційної роботи були досягненні основні цілі:

1. Створено повнофункціональний веб-магазин, який дозволяє користувачам переглядати товари за категоріями, ознайомлюватися з цінами, зображеннями та оформлювати замовлення. Сайт має адаптивний інтерфейс, виконаний з використанням HTML, CSS та Bootstrap, що забезпечує його доступність для різних пристроїв.
2. Розроблено Telegram-бота, який зчитує HTML-каталог і надає можливість взаємодії з товарами у зручному форматі через мобільний месенджер. Використання бібліотеки HtmlAgilityPack дозволило ефективно парсити HTML-структуру сайту, зберігаючи єдиний формат даних між сайтом і ботом.
3. Реалізовано взаємодію між двома компонентами через спільну структуру HTML-файлів, що дозволяє уникнути використання бази даних і забезпечує гнучкість та простоту підтримки проєкту.
4. Проведено повноцінне тестування системи, включаючи функціональні, нефункціональні та UX-тести. Система продемонструвала стабільну роботу, швидкий відгук, зручний інтерфейс та коректну обробку помилок і виняткових ситуацій.
5. Здійснено порівняльний аналіз ефективності компонентів, що дозволив виділити переваги кожного з них і сформулювати рекомендації для подальшого розвитку системи.

Отримані результати мають важливе практичне значення для початкового етапу автоматизації електронної комерції в малих та середніх підприємствах. Робота демонструє, що навіть за умов обмежених ресурсів можна реалізувати інтуїтивно зрозумілу систему обслуговування користувачів, здатну працювати без складних інфраструктурних рішень.

Запропонована архітектура дозволяє забезпечити:

- доступність і легкість у реалізації;
- мінімальні вимоги до середовища запуску (працює локально без серверів і БД);
- можливість подальшого масштабування або міграції на складніші рішення (наприклад, підключення бази даних, авторизації, платіжних систем).

Основними перевагами реалізованої системи є :

- Telegram-бот є зручним мобільним інтерфейсом, швидким у реагуванні та інтуїтивним у використанні, що підходить для користувачів без технічних навичок.
- Веб-магазин забезпечує розширену візуалізацію товарів, адаптивний дизайн, підтримку категорій і можливість подальшого розширення.
- Система загалом є простою, легкою в налаштуванні, що робить її ефективною моделлю для демонстраційних і навчальних цілей.

У процесі тестування було виявлено низку типових помилок (адаптивність, дублювання повідомлень, кодування), які були успішно усунені за допомогою засобів CSS, дебаунсінгу запитів та обробки помилок у HTML-парсингу.

Ці дії підвищили стабільність системи, зменшили ймовірність помилок при оновленнях структури каталогу та забезпечили надійність роботи навіть у несприятливих умовах (відсутність зображень, пошкодження коду тощо).

Хоча розроблена система успішно виконує поставлені завдання, вона має значний потенціал для вдосконалення:

- Додавання авторизації користувачів через сайт і бот.

- Реалізація системи збереження замовлень у базі даних.
- Інтеграція платіжних систем для прийому онлайн-оплат.
- Створення адаптивної адміністративної панелі для редагування каталогу без ручного оновлення HTML.
- Перехід на хмарне або серверне розгортання з використанням ASP.NET Core, Node.js або інших платформ.

Проєкт досягнув усіх поставлених цілей та продемонстрував свою ефективність у сфері автоматизації онлайн-продажів.Система є завершеною, працездатною, адаптивною і може служити як базова платформа для розгортання більш складних комерційних рішень у майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Statista. E-commerce Worldwide – statistics & facts [Електронний ресурс]. – Режим доступу: відкритий https://www.statista.com/topics/871/online-shopping-worldwide/

2. Gartner. How Chatbots Are Transforming Retail [Електронний ресурс]. – Режим доступу: відкритий https://www.gartner.com/en/articles/how-chatbots-are-transforming-retail

3. Telegram Bot API Reference [Електронний ресурс]. – Режим доступу: відкритий https://core.telegram.org/bots/api

4. WooCommerce – Build exactly the eCommerce website you want [Електронний ресурс]. – Режим доступу: відкритий https://woocommerce.com/

5. Shopify – Start an Online Store [Електронний ресурс]. – Режим доступу: відкритий https://www.shopify.com/

6. MDN Web Docs – HTML [Електронний ресурс]. – Режим доступу: відкритий https://developer.mozilla.org/en-US/docs/Web/HTML

7. HtmlAgilityPack Documentation [Електронний ресурс]. – Режим доступу: відкритий https://html-agility-pack.net/documentation

8. Microsoft Learn – C# Guide [Електронний ресурс]. – Режим доступу: відкритий https://learn.microsoft.com/en-us/dotnet/csharp/

9. Visual Studio Code [Електронний ресурс]. – Режим доступу: відкритий https://code.visualstudio.com/

10. Visual Studio 2022 [Електронний ресурс]. – Режим доступу: відкритий https://visualstudio.microsoft.com/vs/

11. Bootstrap v5.3 Documentation [Електронний ресурс]. – Режим доступу: відкритий https://getbootstrap.com/docs/5.3/getting-started/introduction/

12. Live Server Extension for VS Code [Електронний ресурс]. – Режим доступу: відкритий https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer

13. Telegram.Bot Library – GitHub [Електронний ресурс]. – Режим доступу: відкритий https://github.com/TelegramBots/Telegram.Bot

14. Telegram Bot vs Website – Chatimize [Електронний ресурс]. – Режим доступу: відкритий https://chatimize.com/telegram-bot-vs-website/

15. Stack Overflow – Telegram Bot UTF-8 Encoding Issue [Електронний ресурс]. – Режим доступу: відкритий https://stackoverflow.com/questions/46052075/utf-8-telegram-bot

16. Microsoft Docs – CSS Media Queries [Електронний ресурс]. – Режим доступу: відкритий https://learn.microsoft.com/en-us/windows/apps/design/layout/media-queries

17. Stack Overflow – HtmlAgilityPack and Telegram Bot [Електронний ресурс]. – Режим доступу: відкритий https://stackoverflow.com/questions/64043390/html-agility-pack-and-telegram-bot

18. HtmlAgilityPack Issues – GitHub Tracker [Електронний ресурс]. – Режим доступу: відкритий https://github.com/zzzprojects/html-agility-pack/issues

19. Microsoft Learn – Hosting and deployment in ASP.NET Core [Електронний ресурс]. – Режим доступу: відкритий https://learn.microsoft.com/en-us/aspnet/core/host-and-deploy/?view=aspnetcore-7.0

20. Parker, D. E-Commerce MVPs Without Database – Dev.to [Електронний ресурс]. – Режим доступу: відкритий https://dev.to/dparker/building-lightweight-ecommerce-mvp-without-database-4ei1

21. Закон України «Про освіту» (№ 2694-VI від 2 липня 2010 р.) [Електронний ресурс] – Режим доступу: відкритий https://zakon.rada.gov.ua/laws/show/2694-12#Text

22. Сакуренко Б. Ж. Основи безпеки життєдіяльності : навч. посіб. [Електронний ресурс] / Б. Ж. Сакуренко – К. : ТКФК, 2024. – 112 с. – Режим доступу: відкритий www.tkfk.te.ua/wp-content/uploads/2024/02/Сакуренко-БЖД.pdf

23. Кодекс України про адміністративні правопорушення. Ст. 175 [Електронний ресурс]. – Режим доступу: відкритий https://zakon.rada.gov.ua/laws/show/80731-10#Text

24. Кримінальний кодекс України. Ст. 270 [Електронний ресурс]. – Режим доступу: відкритий https://zakon.rada.gov.ua/laws/show/2341-14#Text

25. Методичні вказівки до виконання дипломної роботи освітнього рівня - бакалавр студентами усіх форм навчання для напряму підготовки 121 – Інженерія програмного забезпечення/ Укладачі : Петрик М.Р.,Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 28 с.

26. Дистанційний курс «Кваліфікаційні роботи бакалаврів» сайту дистанційного навчання ТНТУ [Електронний ресурс]. – Режим доступу: URL: https://dl.tntu.edu.ua/bounce.php?course=5329

ДОДАТКИ

ДОДАТОК А – Лістинг файлу “Program.cs”

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using HtmlAgilityPack;
using Telegram.Bot;
using Telegram.Bot.Polling;
using Telegram.Bot.Types;
using Telegram.Bot.Types.Enums;
using Telegram.Bot.Types.ReplyMarkups;

namespace EleShopBot
{

    internal static class Program
    {
        static async Task Main() => await Bot.RunAsync();
    }


    internal static class Bot
    {
        private const string BOT_TOKEN =
"7835525157:AAGOcn0jdYobM_EcJVJfQAFBQJ63ndli5hQ";
        private const string SHOP_ROOT =
@"C:\Users\User\Desktop\ELEShop";

        private static readonly Dictionary<string, string>
CategoryPages = new()
        {
```

```
            ["CPUs"] = "cpus.html",
            ["GPUs"] = "gpus.html",
            ["Motherboards"] = "motherboards.html",
            ["Memory"] = "memory.html",
            ["Storage"] = "storage.html",
            ["Power Supplies"] = "powersupplies.html",
            ["Cases"] = "cases.html",
            ["Pre-built PC"] = "builds.html"
        };

        private static readonly Dictionary<long, List<Product>>
LastProducts = new();
        private static readonly Dictionary<long, OrderState>
OrderStates = new();
        private static ITelegramBotClient _bot = null!;


        internal static async Task RunAsync()
        {
            _bot = new TelegramBotClient(BOT_TOKEN);
            using var cts = new CancellationTokenSource();

            _bot.StartReceiving(HandleUpdateAsync, HandleErrorAsync,
                new ReceiverOptions { AllowedUpdates =
Array.Empty<UpdateType>() }, cts.Token);

            Console.WriteLine("Bot started. Press ENTER to exit…");
            Console.ReadLine();
            cts.Cancel();
        }


        private static async Task
HandleUpdateAsync(ITelegramBotClient _, Update u, CancellationToken
ct)
```

```
        {
            if (u.CallbackQuery is { } cb)
            {
                await HandleCallbackAsync(cb, ct);
                return;
            }

            if (u.Message is { Type: MessageType.Text } msg)
            {
                var chat = msg.Chat.Id;
                var txt = msg.Text!.Trim();

                if (OrderStates.TryGetValue(chat, out var ord) &&
ord.Step != OrderStep.None)
                {
                    await ContinueOrderAsync(chat, txt, ord, ct);
                    return;
                }

                switch (txt.ToLowerInvariant())
                {
                    case "/start":
                    case "/help":
                        await SendWelcomeAsync(chat, ct);
                        break;
                    case "каталог":
                    case "/catalog":
                        await SendCatalogAsync(chat, ct);
                        break;
                    default:
                        await _bot.SendTextMessageAsync(chat,
"Невідома команда. Спробуйте /help", cancellationToken: ct);
                        break;
                }
            }
```

```
        }
        private static Task HandleErrorAsync(ITelegramBotClient _,
Exception ex, CancellationToken __)
        {
            Console.WriteLine("Telegram error: " + ex.Message);
            return Task.CompletedTask;
        }


        private static async Task HandleCallbackAsync(CallbackQuery
cb, CancellationToken ct)
        {
            var chat = cb.Message!.Chat.Id;
            var data = cb.Data ?? string.Empty;
            await _bot.AnswerCallbackQueryAsync(cb.Id,
cancellationToken: ct);

            if (data.StartsWith("cat:"))
            {
                var key = data[4..];
                if (CategoryPages.TryGetValue(key, out var file))
                    await SendProductsAsync(chat, file, ct);
                return;
            }

            if (data.StartsWith("buy:"))
            {
                if (!int.TryParse(data[4..], out int idx)) return;
                if (!LastProducts.TryGetValue(chat, out var list) ||
idx < 0 || idx >= list.Count) return;

                OrderStates[chat] = new OrderState { Step =
OrderStep.AwaitLogin, Product = list[idx] };
                await _bot.SendTextMessageAsync(chat, "Введіть
логін:", cancellationToken: ct);
```

```
            }
        }


        private static async Task ContinueOrderAsync(long chat,
string input, OrderState st, CancellationToken ct)
        {
            switch (st.Step)
            {
                case OrderStep.AwaitLogin:
                    st.Login = input; st.Step =
OrderStep.AwaitPassword;
                    await _bot.SendTextMessageAsync(chat, "Введіть
пароль:", cancellationToken: ct);
                    break;
                case OrderStep.AwaitPassword:
                    st.Password = input; st.Step =
OrderStep.AwaitPhone;
                    await _bot.SendTextMessageAsync(chat, "Введіть
номер телефону:", cancellationToken: ct);
                    break;
                case OrderStep.AwaitPhone:
                    st.Phone = input; st.Step = OrderStep.None;
OrderStates.Remove(chat);
                    var conf = $"✅ Замовлення прийнято!\n\nТовар:
{Escape(st.Product.Title)}\nЦіна:
{Escape(st.Product.Price)}\n\nОператор зв'яжеться з вами протягом 2-
5 хв.";
                    await _bot.SendTextMessageAsync(chat, conf,
cancellationToken: ct);
                    break;
            }
        }
```

```
        private static async Task SendWelcomeAsync(long chat,
CancellationToken ct)
        {
            var kb = new ReplyKeyboardMarkup(new[] { new[] { new
KeyboardButton("Каталог") } }) { ResizeKeyboard = true };
            await _bot.SendTextMessageAsync(chat, "Вітаємо в
EleShop! Натисніть «Каталог».", replyMarkup: kb, cancellationToken:
ct);
        }
        private static async Task SendCatalogAsync(long chat,
CancellationToken ct)
        {
            var rows = CategoryPages.Keys.Select(k =>
InlineKeyboardButton.WithCallbackData(k, $"cat:{k}"))
                                        .ChunkSafe(2).Select(r =>
r.ToArray()).ToArray();
            await _bot.SendTextMessageAsync(chat, "Оберіть
категорію:", replyMarkup: new InlineKeyboardMarkup(rows),
cancellationToken: ct);
        }


        private static async Task SendProductsAsync(long chat,
string htmlFile, CancellationToken ct)
        {
            var path = System.IO.Path.Combine(SHOP_ROOT, htmlFile);
            if (!System.IO.File.Exists(path))
            {
                await _bot.SendTextMessageAsync(chat, "Файл
категорії не знайдено.", cancellationToken: ct);
                return;
            }

            var prods = ParseProducts(path);
            if (prods.Count == 0)
```

```
            {
                await _bot.SendTextMessageAsync(chat, "У цій
категорії немає товарів.", cancellationToken: ct);
                return;
            }
            LastProducts[chat] = prods;

            for (int i = 0; i < prods.Count; i++)
            {
                var p = prods[i];
                var caption = $"*{Escape(p.Title)}*\nЦіна:
{Escape(p.Price)}" +

(string.IsNullOrWhiteSpace(p.Description) ? string.Empty :
$"\n{Escape(p.Description)}");
                var kb = new
InlineKeyboardMarkup(InlineKeyboardButton.WithCallbackData("🛒
Купити", $"buy:{i}"));

                var imgPath = System.IO.Path.Combine(SHOP_ROOT,
p.ImageRelative.Replace("../", string.Empty).Replace('/',
System.IO.Path.DirectorySeparatorChar));

                if (System.IO.File.Exists(imgPath))
                {
                    await using var fs =
System.IO.File.OpenRead(imgPath);
                    await _bot.SendPhotoAsync(chat,
InputFile.FromStream(fs, System.IO.Path.GetFileName(imgPath)),
                        caption: caption, parseMode:
ParseMode.MarkdownV2, replyMarkup: kb, cancellationToken: ct);
                }
                else
                {
```

```
                    await _bot.SendTextMessageAsync(chat, caption,
parseMode: ParseMode.MarkdownV2, replyMarkup: kb, cancellationToken:
ct);
                }
            }
        }


        private static List<Product> ParseProducts(string file)
        {
            var list = new List<Product>();
            var doc = new HtmlDocument();
            doc.Load(file);
            foreach (var card in
doc.DocumentNode.SelectNodes("//div[contains(@class,'card')]") ??
Enumerable.Empty<HtmlNode>())
            {
                var img =
card.SelectSingleNode(".//img")?.GetAttributeValue("src", "") ?? "";
                var pTags =
card.SelectNodes(".//p[contains(@class,'card-text')]")?.ToList() ??
new();
                if (pTags.Count < 2) continue;
                var title = pTags.First().InnerText.Trim();
                var price = pTags.Last().InnerText.Trim();
                var desc = pTags.Count > 2 ? string.Join("\n",
pTags.Skip(1).Take(pTags.Count - 2).Select(t => t.InnerText.Trim()))
: string.Empty;
                list.Add(new Product { Title = title, Price = price,
Description = desc, ImageRelative = img });
            }
            return list;
        }

        private static string Escape(string t)
```

```
        {
            foreach (var s in new[] { "_", "*", "[", "]", "(", ")",
"~", "`", ">", "#", "+", "-", "=", "|", "{", "}", ".", "!" })
                t = t.Replace(s, "\\" + s);
            return t;
        }
    }


    internal record Product(string Title = "", string Price = "",
string Description = "", string ImageRelative = "");

    internal enum OrderStep { None, AwaitLogin, AwaitPassword,
AwaitPhone }
    internal class OrderState
    {
        public OrderStep Step { get; set; } = OrderStep.None;
        public Product Product { get; init; } = new();
        public string Login { get; set; } = string.Empty;
        public string Password { get; set; } = string.Empty;
        public string Phone { get; set; } = string.Empty;
    }


    internal static class EnumerableExt
    {
        public static IEnumerable<IEnumerable<T>> ChunkSafe<T>(this
IEnumerable<T> src, int size)
        {
            var bucket = new List<T>(size);
            foreach (var item in src)
            {
                bucket.Add(item);
                if (bucket.Count == size) { yield return bucket;
bucket = new List<T>(size); }
```

```
            }
            if (bucket.Count > 0) yield return bucket;
        }
    }
}
```

ДОДАТОК Б – Лістинг файлу “index.html”

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>EleShop</title>
    <link rel="shortcut icon" href="../images/Wlogo.png"
type="image/png">
    <link rel="stylesheet" href="css/style.css">
    <style>
        .card-img-top {
          width: 100%;
          height: 200px;
          object-fit: contain;
        }
      </style>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-
KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ"
crossorigin="anonymous">
</head>

<body>
    <header>
        <nav class="navbar navbar-expand-lg bg-body-tertiary">
            <div class="container-fluid">
                <a class="navbar-brand" href="index.html"><img
src="../images/logo.png" class="image1" width="110px"
```

```
                        style="margin-top: 10px;"></a>
                <button class="navbar-toggler" type="button" data-
bs-toggle="collapse"
                    data-bs-target="#navbarNavDropdown" aria-
controls="navbarNavDropdown" aria-expanded="false"
                    aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="collapse navbar-collapse"
id="navbarNavDropdown" style="margin-left: 500px;">
                    <ul class="navbar-nav">
                        <li class="nav-item">
                            <a class="nav-link active text-white"
aria-current="page" href="#">Home</a>
                        </li>
                        <li class="nav-item dropdown">
                            <a class="nav-link dropdown-toggle text-
white" href="#" role="button"
                                data-bs-toggle="dropdown" aria-
expanded="false">
                                Catalog
                            </a>
                            <ul class="dropdown-menu">
                              <li><a class="dropdown-item"
href="builds.html">Pre-built pc</a></li>
                              <li><a class="dropdown-item"
href="cpus.html">CPUs</a></li>
                              <li><a class="dropdown-item"
href="gpus.html">GPUs</a></li>
                              <li><a class="dropdown-item"
href="motherboards.html">Motherboards</a></li>
                              <li><a class="dropdown-item"
href="memory.html">Memory</a></li>
                              <li><a class="dropdown-item"
href="storage.html">Storage</a></li>
```

```
                                <li><a class="dropdown-item"
href="powersupplies.html">Power Supplies</a></li>
                                <li><a class="dropdown-item"
href="cases.html">Cases</a></li>
                            </ul>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-white"
href="#">Feedback</a>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <hr>
    <main class="container my-5">

        <div class="row row-cols-1 row-cols-md-4 g-4">
          <div class="col">
            <a href="cpus.html" class="text-decoration-none text-
dark">
              <div class="card h-100 shadow-sm">
                <img src="images/cpu.jpg" class="card-img-top"
alt="CPUs">
                <div class="card-body text-center">
                  <h5 class="card-title">CPUs</h5>
                </div>
              </div>
            </a>
          </div>
          <div class="col">
            <a href="gpus.html" class="text-decoration-none text-
dark">
              <div class="card h-100 shadow-sm">
```

```
                <img src="images/gpu.jpg" class="card-img-top"
width="100px" alt="GPUs">
                <div class="card-body text-center">
                  <h5 class="card-title">GPUs</h5>
                </div>
              </div>
            </a>
          </div>
          <div class="col">
            <a href="motherboards.html" class="text-decoration-none
text-dark">
              <div class="card h-100 shadow-sm">
                <img src="images/mboard.jpg" class="card-img-top"
width="10px" alt="Motherboards">
                <div class="card-body text-center">
                  <h5 class="card-title">Motherboards</h5>
                </div>
              </div>
            </a>
          </div>
          <div class="col">
            <a href="memory.html" class="text-decoration-none text-
dark">
              <div class="card h-100 shadow-sm">
                <img src="images/ram.jpg" class="card-img-top"
alt="Memory">
                <div class="card-body text-center">
                  <h5 class="card-title">Memory</h5>
                </div>
              </div>
            </a>
          </div>
          <div class="col">
            <a href="storage.html" class="text-decoration-none text-
dark">
```

```
              <div class="card h-100 shadow-sm">
                <img src="images/memory.jpg" class="card-img-top"
alt="Storage">
                <div class="card-body text-center">
                  <h5 class="card-title">Storage</h5>
                </div>
              </div>
            </a>
          </div>
          <div class="col">
            <a href="powersupplies.html" class="text-decoration-none
text-dark">
              <div class="card h-100 shadow-sm">
                <img src="images/powerS.png" class="card-img-top"
alt="Power Supplies">
                <div class="card-body text-center">
                  <h5 class="card-title">Power Supplies</h5>
                </div>
              </div>
            </a>
          </div>
          <div class="col">
            <a href="cases.html" class="text-decoration-none text-
dark">
              <div class="card h-100 shadow-sm">
                <img src="images/case.jpg" class="card-img-top"
alt="Cases">
                <div class="card-body text-center">
                  <h5 class="card-title">Cases</h5>
                </div>
              </div>
            </a>
          </div>
          <div class="col">
```

```
            <a href="builds.html" class="text-decoration-none text-
dark">
              <div class="card h-100 shadow-sm">
                <img src="images/Prebuild.jpg" class="card-img-top"
alt="Готові збірки">
                <div class="card-body text-center">
                  <h5 class="card-title">Pre-built PC</h5>
                </div>
              </div>
            </a>
          </div>
        </div>

        <br>
        <br>
        <section class="py-5 bg-black text-white rounded-3">
            <div class="px-4">
              <small class="text-uppercase text-white">Як працює PC
Shop</small>
              <h2 class="display-6 fw-bold mb-
3">Багатофункціональний сервіс пошуку комплектуючих за найкращими
цінами</h2>
              <div class="row g-4">
                <div class="col-md-4 text-center">
                  <img src="images/search.png" alt="Пошук"
class="mb-3" width="48" height="48">
                  <h5>Швидкий пошук</h5>
                  <p class="small">Зручний фільтр за параметрами
дозволяє знайти потрібні компоненти за лічені секунди.</p>
                </div>
                <div class="col-md-4 text-center">
                  <img src="images/compare.png" alt="Порівняння"
class="mb-3" width="48" height="48">
                  <h5>Порівняння товарів</h5>
```

```
                <p class="small">Порівнюйте характеристики і ціни різних моделей в одному вікні.</p>
              </div>
              <div class="col-md-4 text-center">
                <img src="images/store.png" alt="Магазини" class="mb-3" width="48" height="48">
                <h5>База магазинів</h5>
                <p class="small">Понад 1000 магазинів з тисячами товарів, дані оновлюються щохвилини.</p>
              </div>
            </div>
          </div>
        </section>
      </main>

    <footer class="text-center text-lg-start bg-light text-muted">
        <section class="">
            <div class="container text-center text-md-start mt-5">
                <div class="row mt-3">
                    <div class="col-md-3 col-lg-4 col-xl-3 mx-auto mb-4">
                        <h6 class="text-uppercase fw-bold mb-4">
                            <i class="fas fa-gem me-3"><img src="../images/logo.png" width="70px"></i>EleShop
                        </h6>
                        <p>
                            We hope that you like our service.
                        </p>
                    </div>
                    <div class="col-md-2 col-lg-2 col-xl-2 mx-auto mb-4">

                        <h6 class="text-uppercase fw-bold mb-4">
                            Order
                        </h6>
```

```
                        <p>
                            <a href="#!" class="text-reset">Track
your order</a>
                        </p>
                        <p>
                            <a href="#!" class="text-
reset">Delivery</a>
                        </p>
                        <p>
                            <a href="#!" class="text-
reset">Return</a>
                        </p>
                        <p>
                            <a href="#!" class="text-
reset">Payment</a>
                        </p>
                    </div>
                    <div class="col-md-3 col-lg-2 col-xl-2 mx-auto
mb-4">
                        <h6 class="text-uppercase fw-bold mb-4">
                            Help
                        </h6>
                        <p>
                            <a href="#!" class="text-
reset">Support</a>
                        </p>
                        <p>
                            <a href="#!" class="text-
reset">Contact</a>
                        </p>
                        <p>
                            <a href="#!" class="text-reset">Sizing
chart</a>
                        </p>
                    </div>
```

```
                    <div class="col-md-4 col-lg-3 col-xl-3 mx-auto
mb-md-0 mb-4">
                        <h6 class="text-uppercase fw-bold mb-
4">Contact</h6>
                        <p><i class="fas fa-home me-3"></i>
Ternopil, Ukraine</p>
                        <p>
                            <i class="fas fa-envelope me-3"></i>
                            support@gmail.com
                        </p>
                        <div class="href-image">
                            <a href="..."><img
src="../images/instagram.png" width="20px"></a>
                            <a href="..."><img
src="../images/twitter.png" width="20px"></a>
                            <a href="..."><img
src="../images/face.png" width="20px"></a>
                        </div>
                    </div>
                </div>
            </div>
        </section>
    </footer>

    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.7/dist/umd/pop
per.min.js"
        integrity="sha384-
zYPOMqeu1DAVkHiLqWBUTcbYfZ8osu1Nd6Z89ify25QV9guujx43ITvfi12/QExE"
        crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/js/bootstrap.min.js"
        integrity="sha384-
Y4oOpwW3duJdCWv5ly8SCFYWqFDsfob/3GkgExXKV4idmbt98QcxXYs9UoXAB7BZ"
        crossorigin="anonymous"></script>
```

```
</body>

</html>
```

ДОДАТОК В – Диск із кваліфікаційною роботою бакалавра