МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

МУДРЕНКО АРТЕМ ЮРІЙОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій
канд. техн. наук, доцент
_____ О. В. Зелінська
«____» _____ 20__ р.

ВЕБДОДАТОК ДЛЯ ВІДСТЕЖЕННЯ ТА АНАЛІЗУ ЦІН НА ОБРАНІ ТОВАРИ

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (бакалаврська) робота

Керівник:
Антонов Ю.С., кан. фіз.–мат. наук, доцент,
доцент кафедри інформаційних
технологій

_____
(Підпис)

Оцінка _____/_____/_____
(бали/за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(Підпис)

Вінниця 2025

АНОТАЦІЯ

Мудренко А.Ю. Веб–додаток для відстеження та аналізу цін на обрані товари. Спеціальність 122 “Комп’ютерні науки”, Освітня програма “Комп’ютерні технології обробки даних (Data Science)”. Донецький національний університет імені Василя Стуса, Вінниця, 2025.

У кваліфікаційній роботі представлено розробку вебдодатку для моніторингу та аналізу змін цін на вибрані товари в онлайн–магазинах. Досліджено сучасні методи збору цінової інформації, підходи до реалізації аналітичних функцій та архітектуру веб застосунків. Реалізовано інтерфейс користувача, систему візуального оформлення та логотип проекту. Застосовані технології дозволяють автоматизувати процес відстеження цін та надавати користувачу корисну аналітику у зручному форматі.

Ключові слова**:** вебдодаток, відстеження цін, аналіз даних, вебінтерфейс, логотип, моніторинг.

60 с., 14 рис., 49 джерела.

Mudrenko A.Y. Web Application for Tracking and Analyzing Prices of Selected Products. Specialty 122 “Computer Science”, Educational Program “Computer Data Processing Technologies (Data Science)”. Vasyl Stus Donetsk National University, Vinnytsia, 2025.

The qualification thesis presents the development of a web application for monitoring and analyzing price changes of selected products in online stores. The paper examines modern methods of price data collection, approaches to implementing analytical functions, and web application architecture. A user interface, visual identity system, and project logo were developed. The applied technologies enable automated price tracking and provide users with valuable analytics in a convenient format.

Keywords**:** web application, price tracking, data analysis, web interface, logo, monitoring.

60 p., 14 fig., 49 sources.

ЗМІСТ

ВСТУП

Активне зростання електронної комерції та зростаючий вплив цифрових технологій на повсякденне життя суттєво змінили поведінку сучасного споживача. В умовах онлайн–торгівлі користувачі отримали широкий доступ до асортименту товарів із численних платформ, що, водночас, створило нові виклики – постійні коливання цін, сезонні акції, вплив валютних курсів та ускладнене порівняння пропозицій із різних джерел.

У такій ситуації зростає актуальність інструментів, які дозволяють автоматизувати моніторинг та аналіз цін у реальному часі. Вебдодатки для відстеження вартості товарів мають на меті вирішити кілька завдань одночасно: зекономити час користувача, надати об'єктивну інформацію про зміну цін, допомогти у прийнятті раціональних рішень щодо купівлі та уникнути зайвих витрат.

Попри наявність ряду вже реалізованих сервісів, більшість із них мають обмеження – зосереджуються на окремих майданчиках, не пропонують розгорнутої аналітики, не підтримують персональні списки або не мають інтерактивної візуалізації. Це підкреслює доцільність створення нового вебдодатку, орієнтованого на зручність, ефективність і наочність.

Метою даної дипломної роботи є розробка вебдодатку для моніторингу та аналізу цін на обрані товари з різних онлайн–джерел, що дозволяє формувати персоналізований список, переглядати графічну динаміку змін і отримувати автоматичні сповіщення про зниження вартості.

Об’єктом дослідження виступає процес моніторингу цін в онлайн–просторі. Предметом дослідження є методи збору, обробки, зберігання та візуалізації цінової інформації у форматі вебдодатку.

Для досягнення поставленої мети у дипломній роботі передбачено вирішення таких завдань:

- дослідити сучасні методи побудови вебдодатків для відстеження цін;

- проаналізувати наявні інструменти збору інформації з зовнішніх джерел (API, парсинг);
- спроєктувати архітектуру системи з урахуванням масштабованості й гнучкості;
- реалізувати користувацький інтерфейс відповідно до принципів UX/UI–дизайну;
- створити логотип і елементи візуальної ідентичності проєкту;
- запрограмувати основні модулі: додавання товарів, збереження історії цін, аналітичні функції, система сповіщень;
- провести тестування, оцінити продуктивність і зручність користування системою.

Дипломна робота складається з трьох розділів. Перший розділ присвячений теоретичним аспектам, аналізу ринку та технологій. У другому подано архітектуру системи, описано логічні моделі, вибір технологій, розроблено логотип та інтерфейсні макети. Третій розділ містить реалізацію вебдодатку, тестування його функцій та аналіз отриманих результатів.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ВІДСТЕЖЕННЯ ТА АНАЛІЗУ ЦІН НА ТОВАРИ

1.1 Сучасні підходи до моніторингу та аналізу цін у вебсередовищі.

В нинішніх умовах хвилі діджиталізації торгівлі, коли більшість купує товари в Інтернеті, настав час стежити за змінами цін на продукцію. На цінову політику інтернет–магазинів впливає кілька факторів, це сезонні знижки, маркетингові стратегії, акційні пропозиції, сильна конкуренція серед платформ електронної комерції, коливання курсів. Усе це разом призведе до того, що ціни на один і той самий продукт значно відрізнятимуться протягом дуже короткого періоду часу; отже, у споживачів, які хочуть приймати обґрунтовані та економічно вигідні рішення щодо своїх покупок, виникнуть труднощі.

На тлі зростаючого попиту на цифрові рішення для контролю витрат, автоматизовані інструменти для моніторингу цін на товари набувають дедалі більшої актуальності.

Розробка вебдодатку, здатного виконувати такі функції, має не лише прикладне значення для повсякденного користувача, а й потенціал для інтеграції у ширші системи – від бізнес–аналітики до маркетингових досліджень та автоматизації закупівель. Такий інструмент відкриває доступ до аналізу цінових змін, а також до глибших аналітичних даних, що можуть бути корисними для прогнозування, оптимізації витрат і прийняття рішень, заснованих на фактах.

У цьому контексті ідея створення вебдодатку для моніторингу цін виглядає цілком доречною з огляду на сучасні технічні виклики, ринкові потреби та очікування користувачів. Вона передбачає об'єднання знань у сфері веб розробки, роботи з даними, дизайну інтерфейсів і створення інтерактивних систем візуалізації.

Веб додатки для цінового моніторингу – це спеціалізоване програмне забезпечення, основною функцією якого є автоматизоване отримання, збереження та аналіз інформації про вартість товарів із різноманітних онлайн–

ресурсів. Користувачі завдяки їм отримують змогу оперативно слідкувати за актуальними пропозиціями, відстежувати зміну цін у динаміці та формувати уявлення про майбутні тенденції. Це дозволяє приймати більш раціональні рішення при здійсненні покупок.

Водночас такі системи мають цінність не лише для кінцевих користувачів. Вони є корисним інструментом для аналітиків, маркетологів, представників малого та середнього бізнесу, а також компаній, що займаються оцінкою конкурентного середовища. Зібрані дані допомагають аналізувати ринок, порівнювати ціни, формувати ефективні стратегії ціноутворення та прогнозувати рівень попиту у різні періоди.

Архітектура таких рішень зазвичай побудована на трьох ключових компонентах.

1. Модуль збору даних, що діє як місток між джерелами цінової інформації та внутрішньою системою. Він може реалізовуватись як через веб скрейпінг (scraping), так і шляхом взаємодії з API, які надають онлайн–платформи чи магазини.
2. База даних, що акумулює як поточну, так і історичну інформацію про ціни. Це створює основу для аналізу змін вартості товарів у часі й побудови прогнозів.
3. Аналітичний модуль, який обробляє зібрані дані та надає їх у зручному для користувача вигляді: графіки, таблиці, сповіщення про зміну цін. Його головна мета – допомогти швидко й точно інтерпретувати інформацію: [1].

Проєктування таких вебдодатків базується на сучасних принципах веброзробки. Доволі часто використовується клієнт–серверна модель, де інтерфейс реалізується за допомогою HTML, CSS, Python і популярних фреймворків, а серверна частина забезпечує логіку обробки, взаємодію з базами даних та зовнішніми джерелами. Зв'язок між компонентами зазвичай налагоджується через REST API, що дає змогу ефективно передавати інформацію у форматах JSON, PY або XML.

Формат реалізації залежить від цільової аудиторії та вимог: це може бути SPA – динамічний односторінковий застосунок, MPA – класичний багатосторінковий варіант, або ж PWA – прогресивна вебаплікація, яка поєднує переваги веб і мобільних додатків, включаючи офлайн–режим, push–сповіщення й адаптивний інтерфейс. Що стосується джерел цінової інформації, застосовуються кілька методів: офіційні або відкриті API торгових платформ, вебскрейпінг сторінок з товарами, або інтеграція з агрегаторами, що вже мають структуровані дані.

Водночас важливо враховувати технічні та етичні аспекти. Необхідно дотримуватись юридичних норм щодо збору інформації, обмежувати частоту запитів, уникати блокувань і впроваджувати механізми обходу захисту від ботів.

Отже, створення вебдодатку для моніторингу цін – це міждисциплінарне завдання, що поєднує технічну експертизу, аналітичне мислення, знання у сфері UX/UI та роботу з великими обсягами даних. Такий продукт може значно спростити життя користувача та одночасно стати цінним бізнес–інструментом у сфері управління інформацією.

1.2 Технології збору та обробки цінової інформації з онлайн–ресурсів.

Щоб веб–додаток для моніторингу цін працював ефективно, критично важливо налагодити процес збору, обробки та аналізу даних. Саме ці етапи забезпечують створення надійної інформаційної основи, яка дозволяє будувати аналітику, виявляти тренди цін і вчасно інформувати користувачів про зміну вартості товарів [2, 3].

1.2.1 Збір даних

Цінову інформацію зазвичай отримують двома основними способами:

1. **API–інтеграція** – це структурований, стабільний та легальний спосіб доступу до даних, за якого онлайн–магазини чи торгові платформи надають відкриті або приватні API. Дані при цьому передаються у форматах JSON або XML. Метод швидкий і надійний, хоча доступність API не завжди гарантована [4].

2. **Вебскрейпінг** – універсальний підхід, що дозволяє автоматично зчитувати інформацію безпосередньо зі сторінок сайтів з допомогою таких бібліотек, як BeautifulSoup, Scrapy або Puppeteer. Однак скрейпінг часто супроводжується технічними [5] викликами (наприклад, зміни в структурі сторінок) і юридичними обмеженнями, оскільки деякі сайти прямо забороняють таку активність у своїх політиках.

У складніших випадках для обходу антибот–захисту застосовуються проксі, контроль частоти запитів, ротація заголовків user–agent тощо. Але при цьому важливо зберігати етичний підхід і не створювати надмірне навантаження на сайти–джерела.

1.2.2 Обробка даних

Зібрана інформація зазвичай має неструктурований або частково структурований вигляд. Вона може містити HTML–теги, зайві символи, дублі або неповні записи. На цьому етапі важливо провести очищення даних, що включає:

- Усунення повторюваних чи зайвих елементів;
- Приведення до єдиного формату (наприклад, уніфікація валют);
- Перевірку значень – щоб переконатися у коректності чисел, дат тощо;
- Фільтрацію порожніх або некоректних полів.

Далі проводиться нормалізація: упорядкування назв товарів, категорій, брендів, щоб забезпечити точну ідентифікацію елементів з різних джерел.

1.2.3 Зберігання даних

Оброблену інформацію зазвичай розміщують у базах даних – як реляційних (SQL), так і нереляційних (NoSQL) [6]. Найчастіше використовують такі технології:

- **MySQL, PostgreSQL** – класичні СУБД для зберігання структурованих даних з можливістю складного запитування;
- **MongoDB** – документо–орієнтована база, де дані зберігаються у форматі BSON, що підходить для гнучких, змінних структур.

Для підвищення продуктивності також можуть залучатися кешуючі рішення (наприклад, Redis) або хмарні сервіси для розподіленого зберігання даних – особливо в масштабних системах із високим навантаженням.

1.2.4 Аналіз даних

На основі впорядкованих даних система виконує цілий спектр аналітичних задач:

- **Побудова графіків** – візуалізація змін ціни у часі;
- **Виявлення акцій і знижок** – фіксація стрибків вартості та генерація відповідних сповіщень;
- **Прогнозування** – застосування моделей регресії, часових рядів або машинного навчання для передбачення майбутніх цін;
- **Порівняльний аналіз** – оцінка різниці цін між постачальниками одного товару.

Для цього дедалі частіше використовуються сучасні бібліотеки аналітики, зокрема Pandas, NumPy, Scikit–learn, TensorFlow, а також інструменти для візуалізації – Plotly, Chart.js, D3.js. Це дає змогу досягти високого рівня наочності та забезпечити зручну взаємодію з користувачем.

1.3 Актуальні тренди у створенні вебдодатків для відстеження цін.

Сфера веб розробки перебуває у постійній трансформації, що зумовлено не лише динамікою користувацьких очікувань, а й жорсткішою конкуренцією серед цифрових сервісів та активною появою інноваційних технологій. Ці процеси безпосередньо впливають і на розвиток вебрішень для моніторингу цін, які тепер мають відповідати не лише базовим функціональним критеріям, а й сучасним стандартам у продуктивності, юзабіліті та візуальному оформленні.

- **Індивідуалізація на основі поведінкової аналітики.** Одна з ключових тенденцій – створення персоналізованого користувацького досвіду. Вебдодатки адаптуються під інтереси та дії конкретного користувача, враховуючи його історію переглядів і вибір товарів, пропонуючи цільові рекомендації, добірки або сповіщення щодо змін у вартості саме

тих товарів, що його цікавлять. Зазвичай це досягається через впровадження алгоритмів машинного навчання, зокрема методів кластеризації, предиктивної аналітики або колаборативної фільтрації [7].

- **Інтеграція хмарних сервісів.** Переважає концепція cloud–first, яка базується на повноцінному використанні хмарної інфраструктури – таких платформ, як AWS, Azure чи Google Cloud. Це забезпечує не лише масштабованість і стабільну роботу, а й моментальний доступ до інформації в будь–якому регіоні. Додатково, хмара дозволяє організувати автоматичне резервування, гнучке управління ресурсами та безперервне оновлення за DevOps–підходами [8].
- **Адаптивність під мобільні пристрої та PWA.** У відповідь на постійне зростання мобільного трафіку, сучасні розробники все активніше реалізують адаптивний інтерфейс і створюють прогресивні вебдодатки (PWA). Такі рішення суміщають зручність вебформату із функціональністю нативних застосунків: можливість працювати без інтернету, надсилання push–сповіщень, розміщення ярлика на головному екрані смартфона. Це суттєво розширює охоплення аудиторії [9].
- **Інтерактивна аналітика та візуалізація даних.** Користувач очікує не просто дані у вигляді таблиць, а наочні, зручні й гнучко налаштовані інструменти – графіки, дашборди, діаграми, які можна фільтрувати та налаштовувати. Завдяки бібліотекам на кшталт Chart.js, Plotly, D3.js або Highcharts, реалізуються сучасні інтерфейси, що дозволяють швидко оцінити динаміку цін без потреби глибокого аналізу числових масивів [10].
- **Соціальні інтеграції та месенджери.** Актуальним напрямком стає поєднання вебдодатків із популярними каналами комунікації – Instagram, Facebook, Telegram чи Viber. Так користувач має змогу отримувати актуальні повідомлення про зміну цін прямо в месенджері,

а також ділитися інформацією з іншими, що сприяє природному поширенню сервісу [11].

- **Автоматизація тестування та DevOps.** Інтеграція DevOps–практик, автоматизованого тестування, CI/CD–процесів і системного моніторингу є вже стандартом у розробці. Такі підходи дозволяють прискорити оновлення, підвищити стабільність системи та мінімізувати людські помилки при розгортанні нових версій [12, 13].
- **Захист персональних даних.** Зі збільшенням обсягів користувацької інформації, питання інформаційної безпеки виходить на перший план. Стандарти безпеки передбачають застосування HTTPS–протоколу, JWT–токенів для автентифікації, шифрування, чітких правил доступу, а також аудитів. Особливо це важливо для додатків, що містять облікові записи, індивідуальні добірки чи платіжні дані. Таким чином, сучасні системи для моніторингу цін – це вже не просто бази з парсерами, а повноцінні технологічні рішення, що поєднують в собі гнучкий інтерфейс, аналітику, високий рівень автоматизації та відповідність актуальним трендам в IT–галузі [14].

1.4 Огляд ринку та аналіз подібних вебдодатків.

У світлі зростаючого інтересу до онлайн–шопінгу та прагнення споживачів економити, вебдодатки для відстеження цін демонструють поступове, але стабільне зростання популярності. На сьогодні існує безліч інструментів, що дозволяють не лише слідкувати за динамікою вартості товарів, а й порівнювати ціни в різних інтернет–магазинах, зберігати обрані позиції до списків та оперативно отримувати повідомлення про акції чи знижки. Головна мета подібних сервісів – надати користувачам точну й актуальну інформацію, яка допоможе приймати більш вигідні та зважені рішення при купівлі. Огляд провідних сервісів в Україні та за кордоном:

- **Price.ua** – відомий український агрегатор цін, що дає змогу порівнювати вартість конкретного товару у десятках інтернет–магазинів. Користувачам доступні дані про динаміку змін, рейтинг

продавця, наявність продукції та інші супутні параметри. Основну увагу сервіс приділяє таким категоріям, як побутова техніка, електроніка та аксесуари до неї (записи, індивідуальні добірки чи платіжні дані). Таким чином, сучасні системи для моніторингу цін – це вже не просто бази з парсерами, а повноцінні технологічні рішення, що поєднують в собі гнучкий інтерфейс, аналітику, високий рівень автоматизації та відповідність актуальним трендам в IT–галузі.

- **Hotline.ua** – ще один вагомий гравець на українському ринку порівняння цін. Відрізняється власною пошуковою системою, широкою класифікацією товарів, зручними фільтрами та механізмами сортування. Доступна історія змін вартості, а також інтеграція з акаунтом користувача для формування списків бажаного.
- **Keepa.com** – міжнародний сервіс, що спеціалізується на відстеженні цін на платформі Amazon. Основні можливості включають графіки цінових змін, оповіщення про знижки та статистичні дані. Платформа підтримує браузерні розширення, що робить її зручною для використання під час безпосереднього перегляду товарів.
- **CamelCamelCamel.com** – ще один сервіс, орієнтований виключно на Amazon. Пропонує повноцінну історію змін цін, можливість встановлення цільових значень для автоматичних сповіщень. Відзначається простим, але функціональним інтерфейсом і стабільною роботою.
- **Idealo.de** – популярний німецький ресурс, який проводить аналіз цін у численних європейських онлайн–магазинах. Має потужну систему фільтрів, мобільний застосунок, оцінювання продавців та ефективні механізми сповіщення. База товарів регулярно оновлюється і включає сотні тисяч позицій.

У межах порівняльного аналізу було розглянуто п’ять популярних сервісів моніторингу цін: Hotline.ua, Price.ua, Keepa, CamelCamelCamel та Idealo. Оцінювання здійснювалося за кількома ключовими критеріями.

- **Цільовий ринок.** Hotline.ua та Price.ua орієнтовані переважно на українського споживача. У той час як Keepa і CamelCamelCamel працюють з платформою Amazon у США, а Idealo охоплює європейський ринок.
- **Типи джерел.** Hotline.ua, Price.ua та Idealo отримують дані безпосередньо з онлайн–магазинів. Keepa та CamelCamelCamel використовують API Amazon як основне джерело.
- **Історія цін.** Усі розглянуті сервіси мають функцію перегляду історії цін, що дозволяє користувачам відстежувати динаміку вартості товарів.
- **Сповіщення про знижки.** Повноцінні механізми сповіщення реалізовані в Keepa, CamelCamelCamel, Idealo та Hotline.ua. У Price.ua ця функція доступна частково.
- **Мобільна версія або додаток.** Мобільні рішення наявні у Hotline.ua, Price.ua, Idealo та частково у CamelCamelCamel. Keepa пропонує розширення для браузера, яке адаптоване до мобільного використання.
- **Можливість інтеграції.** Найбільшу відкритість демонструє Keepa, що пропонує API для розробників. В Idealo інтеграція відсутня, а в інших сервісах вона реалізована частково або обмежено.
- **Дизайн та UX.** Найвищий рівень візуального оформлення та користувацького досвіду представлений у Keepa та Idealo. CamelCamelCamel характеризується мінімалістичним підходом, а Hotline.ua та Price.ua мають середній рівень дизайну.

Проведений огляд ринку дозволяє стверджувати, що більшість доступних вебдодатків орієнтуються переважно на локальні торговельні майданчики (наприклад, Hotline, Price.ua) або вузькоспеціалізовані платформи, такі як Amazon (Keepa, CamelCamelCamel). Водночас спостерігаються кілька типових обмежень, які притаманні майже всім сервісам:

- **Недостатній рівень персоналізації** – сервіси рідко враховують індивідуальні інтереси користувачів і майже не використовують адаптивні механізми рекомендацій.

- **Обмежена інтеграційність** – часто відсутня можливість підключення сторонніх інструментів через API або API реалізоване частково.
- **Низька ефективність візуальної аналітики** – представлені графіки й таблиці здебільшого статичні та не надають повного аналітичного контексту.
- **Вузька товарна або географічна спеціалізація** – що звужує можливості для різностороннього використання.

Ці недоліки створюють сприятливі передумови для розробки нових рішень, які відповідатимуть вимогам часу, а саме: будуть мультиплатформенними, персоналізованими, глибоко аналітичними, орієнтованими на зручність користувача та відкритими до інтеграцій через API або внутрішні сервіси.

1.5 Актуальні тренди у створенні вебдодатків для відстеження цін.

Одним із ключових етапів у створенні вебдодатків для моніторингу цін є отримання первинної інформації з зовнішніх ресурсів. Коли офіційні API або недоступні, або мають обмежену функціональність, на допомогу приходить вебпарсинг (web scraping) – автоматизований процес витягування структурованих даних зі сторінок сайтів.

1.5.1 Як працює парсинг

Парсинг імітує поведінку звичайного користувача у браузері: надсилається запит до сайту, отримується HTML–сторінка у відповідь, далі відбувається її аналіз – із витягом необхідних фрагментів (наприклад, ціни, назви товару чи інформації про наявність). Загалом, процес поділяється на кілька послідовних кроків:

- Надсилання HTTP–запиту (метод GET) до потрібної сторінки;
- Отримання HTML–коду;
- Аналіз структури DOM (Document Object Model);
- Визначення потрібних елементів (наприклад, div з класом product–price);

- Витяг тексту або значень атрибутів;
- Збереження результатів у форматі, зручному для подальшої обробки – JSON, CSV чи база даних.

1.5.2 Мови програмування, що найчастіше застосовуються для реалізації вебдодатків

Хоча реалізувати парсер можна практично будь–якою сучасною мовою, найчастіше для цього використовують:

- **Python** – лідер завдяки простому синтаксису та широкому набору бібліотек:
  – requests – для HTTP–запитів;
  – BeautifulSoup – для обробки HTML;
  – lxml – для швидкого парсингу XML/HTML;
  – Selenium – підходить для роботи з динамічними сторінками.
- **JavaScript / Node.js** – зручно використовувати у середовищах, близьких до браузера:
  – requests – для HTTP–запитів;
  – BeautifulSoup – для обробки HTML;
- **PHP** – раніше був популярний, але нині рідко використовується для парсингу.
- **Java** – застосовується у складніших системах, хоча має менше спеціалізованих рішень.

Python залишається найзручнішим вибором – через свою універсальність, підтримку спільноти та швидкість розробки.

1.5.3 Об’єктно–орієнтований підхід у створенні парсера

Замість створення одноразових скриптів доцільно будувати гнучкі системи, які можна легко адаптувати до змін на сайтах чи підключення нових джерел. У цьому допомагає об’єктно–орієнтоване програмування (ООП). Основні переваги ООП:

- **Інкапсуляція** – логіка роботи з кожним сайтом розміщується в окремому класі (наприклад, з методами get_price() або parse_html()).

- **Наслідування** – спільні дії описуються в базовому класі (BaseParser), тоді як специфіка окремих сайтів – у дочірніх.
- **Поліморфізм** – можна однаково звертатися до різних парсерів через універсальний інтерфейс, викликаючи метод get_data() незалежно від конкретного джерела.

Висновок до першого розділу

У першому розділі дипломного дослідження було здійснено всебічне теоретичне обґрунтування проблематики, пов’язаної з розробкою вебдодатку для моніторингу й аналізу цін на вибрані товари. Спираючись на проаналізовані джерела та практичні приклади, встановлено, що мінливість цінової політики в електронній комерції, високий рівень ринкової конкуренції та зростаючі очікування користувачів зумовлюють потребу у створенні рішень, здатних автоматизувати процес збирання, обробки й візуалізації інформації про вартість товарів.

У роботі розглянуто ключові підходи до архітектури таких вебсистем, включаючи клієнт–серверну модель, використання SPA, MPA та PWA форматів, застосування REST API для взаємодії між компонентами. Також описано сучасні методи отримання даних – через API, засоби вебскрейпінгу чи підключення до агрегаторів. Особливу увагу приділено етапам обробки інформації: очищенню, нормалізації, збереженню у структурованому вигляді та подальшій аналітичній інтерпретації для надання користувачеві цінної інформації. Було проаналізовано основні сучасні напрями у веброзробці, зокрема персоналізований підхід, інтеграцію з мобільними платформами, впровадження хмарних рішень, адаптивний інтерфейс, UX–дизайн, захищеність даних і можливість масштабування. Виявлено, що найуспішніші сервіси поєднують потужний функціонал із привабливим візуальним оформленням та аналітичними інструментами, що дозволяє користувачам не просто стежити за цінами, а й ухвалювати обґрунтовані рішення щодо часу купівлі.

Деталізовано принципи роботи парсингу та технічні засоби його реалізації, серед яких найбільш поширеними є Python–бібліотеки та браузерні інструменти

на базі JavaScript. Окремо наголошено на важливості застосування об’єктно–орієнтованого підходу для створення гнучких та масштабованих систем, що дозволяє адаптувати парсери до змін структури сайтів і нових джерел.

Окремо було розглянуто наявні рішення на ринку – Hotline.ua, Price.ua, Keepa, Idealo тощо. Їхній функціонал забезпечує базовий рівень моніторингу, однак за результатами аналізу виявлено низку недоліків: обмежену персоналізацію, слабку інтеграцію з іншими платформами, недостатньо розвинену аналітику та низьку гнучкість у роботі з різними категоріями товарів. Це свідчить про наявність вільної ринкової ніші для створення інноваційного вебрішення, здатного не лише збирати інформацію, а й якісно інтерпретувати її з урахуванням індивідуальних потреб користувача.

Таким чином, висновки, сформульовані в цьому розділі, закладають міцне підґрунтя для подальшого етапу – проєктування та реалізації власного вебдодатку, який відповідатиме сучасним технологічним стандартам і очікуванням цільової аудиторії.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА АРХІТЕКТУРА ВЕБДОДАТКУ

2.1 Постановка задачі та функціональні вимоги

Основною метою створення вебдодатку є розробка зручного, ефективного й адаптивного інструменту, що дозволяє автоматично відстежувати зміни цін на вибрані товари з різноманітних інтернет–магазинів, а також надавати користувачеві аналітичну інформацію, необхідну для прийняття економічно вигідних рішень щодо купівлі.

2.1.1 Постановка задачі

У межах цього проекту визначено завдання реалізації вебдодатку з такими функціональними можливостями:

- автоматизоване отримання актуальних цінових даних із заданих джерел за допомогою API або методів веб скрейпінгу;
- збереження зібраної інформації з фіксацією часу, дати та джерела отримання;
- формування історії змін вартості для кожної товарної позиції;
- створення облікового запису з функцією додавання товарів до персонального списку спостереження;
- можливість налаштування повідомлень про зниження ціни нижче заданого порогу;
- представлення інформації у вигляді графіків, таблиць чи текстових сповіщень;
- базова аналітика, включно з розрахунками середньої, мінімальної, максимальної ціни та темпу змін;
- адаптивний дизайн інтерфейсу, що коректно функціонує на десктопах і мобільних пристроях.

2.1.2 Основні функціональні вимоги до системи

1. Реєстрація користувача та процедура входу в систему – Високий пріоритет;

2. Можливість додавання товарів до персонального списку – Високий пріоритет;
3. Відображення динаміки змін цін у вигляді графіка для кожного товару – Високий пріоритет;
4. Отримання цінової інформації через зовнішні API або методи скрейпінгу – Середній пріоритет;
5. Автоматичне фонове оновлення даних про ціни – – Середній пріоритет;
6. Сповіщення при зміні ціни або досягненні встановленого порогу – Високий пріоритет;
7. Функції фільтрації та сортування у списку товарів – Середній пріоритет;
8. Адаптивне відображення інтерфейсу для мобільних пристроїв – Високий пріоритет;
9. Збереження цінової історії в БД із зазначенням часу та дати – Високий пріоритет;
10. Базова адмінпанель для тестування та наповнення бази даних– Високий пріоритет.

2.1.3 Нефункціональні вимоги

- **Швидкодія:** сторінки вебдодатку повинні завантажуватись не довше ніж за 2 секунди.
- **Масштабованість:** система має підтримувати стабільну роботу при зростанні кількості користувачів і обсягу товарів.
- **Надійність:** у разі тимчасової недоступності окремого джерела даних система повинна зберігати працездатність.
- **Безпека:** передбачено реалізацію механізмів автентифікації, захисту персональних даних та обмеження доступу до API.
- **Юзабіліті:** інтерфейс користувача має бути інтуїтивним і зрозумілим без потреби додаткового навчання.

2.1.4 Асинхронність і фоновий режим роботи системи збору даних

Окрему увагу варто приділити тому, що парсери, відповідальні за отримання актуальних цін із вебресурсів, мають функціонувати у фоновому

режимі. Це означає, що процес збору даних повинен виконуватись автоматично – за заданим розкладом (наприклад, щогодини або кілька разів на добу) – незалежно від активності користувача, без потреби ручного запуску кожного запиту. Такий підхід дозволяє:

- Гарантувати актуальність цін на момент відкриття сторінки;
- Безперервно накопичувати історію змін, навіть без участі користувача;
- Зменшити навантаження на сервер завдяки кешуванню результатів;
- Створити базу для глибшого аналізу – наприклад, відстеження трендів, виявлення різких коливань чи обчислення середніх значень.

З технічної точки зору, реалізація фонової роботи забезпечується через:

- **Планувальники завдань** – як–от Celery з Redis у середовищі Flask або Django, або cron на рівні операційної системи;
- **Асинхронний запуск парсерів** – з використанням asyncio, ThreadPoolExecutor тощо, щоб уникати блокувань при великій кількості запитів;
- **Черги завдань** – для ефективного масштабування процесу збору з багатьох джерел одночасно.

2.2 Вибір технологій та інструментів розробки

Вибір відповідних технологій є одним із ключових етапів реалізації будь–якого вебдодатку, адже він визначає не тільки функціональність системи, а й зручність її розробки, обслуговування, подальшого масштабування та можливість інтеграції з іншими сервісами. У межах цієї дипломної роботи прийнято рішення застосовувати сучасні, водночас доступні та добре задокументовані інструменти, які забезпечать стабільність, простоту реалізації та належну якість кінцевого продукту.

2.2.1 Користувацький інтерфейс (фронтенд)

Оскільки зручність взаємодії з додатком є важливою складовою, значна увага приділяється візуальному оформленню. Для проєктування інтерфейсу використовується Figma – онлайн–інструмент для створення прототипів, макетів та UX/UI–дизайну. Це дозволяє заздалегідь пропрацювати логіку взаємодії й

забезпечити послідовність оформлення. Для розробки інтерфейсу планується створення HTML/CSS–шаблонів без застосування складних JavaScript–фреймворків. Шаблони будуть інтегруватися у бекенд за допомогою Jinja2 (у Flask) або Django Templates. Для пришвидшення верстки використовуватимуться CSS–фреймворки Bootstrap або Tailwind, що дає змогу легко реалізувати адаптивний та привабливий дизайн.

2.2.2 Серверна частина (бекенд)

Для реалізації серверної логіки вебдодатку для відстеження та аналізу цін на обрані товари було прийнято низку архітектурних і технологічних рішень, що відповідають функціональним потребам системи: регулярне оновлення цін, зберігання історії, підтримка багатокористувацького доступу, простота розширення та масштабування.

Архітектура Серверна частина побудована за принципом клієнт–серверної архітектури з використанням REST API, яке дозволяє фронтенду надсилати запити (наприклад, на отримання історії цін або додавання товару) й отримувати у відповідь структуровані дані у форматі JSON.

Вся логіка взаємодії між компонентами розділена на:

Контролери (routes) – відповідають за обробку вхідних HTTP–запитів;

Сервіси – містять бізнес–логіку (наприклад, перевірка дублювання товару, запуск оновлення цін);

Моделі – взаємодіють з базою даних (через ORM);

Парсери/API–клієнти – окремі модулі для збору цін з різних джерел.

Взаємодія з джерелами даних

Оскільки більшість онлайн–магазинів не надають відкритих API, було розроблено кілька підходів до збору цін: Використання API (якщо доступне) – наприклад, Prom, Rozetka, або інші надають JSON–відповіді за відкритими ключами. Для цього реалізовано окремий API–клієнт, який надсилає запити через requests, обробляє відповіді, витягує необхідні поля (назва, ціна, id, тощо). HTML–парсинг (BeautifulSoup) – для статичних сторінок, де ціна та опис товару доступні без JavaScript. Визначається структура DOM, вказуються селектори для

отримання значень. Динамічні сторінки (Selenium) – для ресурсів, де дані підвантажуються JavaScript (наприклад, через AJAX), використовується headless–браузер, який відкриває сторінку, чекає на повне завантаження і витягує ціни. Ці модулі мають уніфікований інтерфейс: кожен парсер повертає словник із полями name, price, source, timestamp.

Структура та проектування бази даних

Для забезпечення ефективного зберігання та швидкого доступу до цінової інформації було спроєктовано реляційну базу даних, що включає такі основні сутності: users – зберігає облікові записи користувачів; products – інформація про товари, які можна відстежувати (назва, посилання, джерело); price_history – таблиця, в якій фіксується зміна цін по кожному товару з датою та часом; user_products – таблиця зв’язку "багато до багатьох", що дозволяє кожному користувачу мати список своїх товарів; додаткові таблиці: sources, categories, logs. Така структура дозволяє зберігати історію змін у зручному форматі, формувати запити для аналітики, побудови графіків, а також легко розширювати функціонал (наприклад, додати систему сповіщень при зниженні ціни). Вибір технологій та гнучкість

Для реалізації REST API та інтеграції з БД розглядалися два основні фреймворки: Flask – як мікрофреймворк, надає повний контроль над архітектурою, легко інтегрується з SQLAlchemy, дозволяє розділити логіку на модулі. Було обрано як основний інструмент у цьому проєкті завдяки гнучкості, мінімалізму та хорошій сумісності з парсинг–інструментами. Django – має повну екосистему для швидкого запуску, але його структура виявилася надлишковою для даного MVP–проєкту.

Для зберігання даних під час розробки обрано SQLite як легке рішення без необхідності розгортання окремого сервера. У майбутньому структура бази дозволяє безболісно мігрувати на PostgreSQL, якщо система розшириться та потребуватиме підвищеної продуктивності, стабільності в мережі та підтримки складних транзакцій.

2.2.3 Зберігання та обробка даних

З метою збереження даних про товари, ціни, користувачів та їхні дії в системі, було прийнято рішення про використання реляційної бази даних. На етапі розробки використовується SQLite – вбудована легковагова СУБД, яка не потребує налаштування окремого серверного середовища. Вона забезпечує достатню функціональність для локального тестування та швидкого прототипування. Проте, для продакшн–версії доцільно перейти на PostgreSQL, яка забезпечує вищу стабільність, масштабованість та кращу підтримку при роботі в мережевому середовищі. PostgreSQL підтримує складні транзакції, типи даних, перевірки цілісності та розширені можливості оптимізації запитів, що особливо актуально у випадку з великими обсягами історичних цін або активною багатокористувацькою взаємодією. Щодо вибору засобів доступу до бази даних, Python–фреймворки пропонують зручні інтегровані інструменти: Flask працює у парі з бібліотекою SQLAlchemy, що забезпечує зручний об'єктно–реляційний доступ до даних, дозволяє визначати структуру таблиць у вигляді Python–класів, а також полегшує міграції та рефакторинг; Django має вбудовану ORM–систему, яка також дозволяє ефективно працювати з базою даних на рівні об'єктів. Таким чином, обраний підхід дає змогу зберегти гнучкість у розробці, адаптуючи СУБД під задачі конкретного етапу проєкту – від прототипу до готового продукту з високими вимогами до надійності.

2.2.4 Інструменти збору цінової інформації

Оскільки однією з ключових функцій системи є моніторинг актуальних цін, важливою складовою є автоматичне отримання та оновлення даних про товари. Для цього у вебдодатку передбачено використання низки інструментів, що дозволяють реалізувати автоматизований парсинг та інтеграцію з відкритими API. Отримання даних з вебресурсів: requests – стандартна Python–бібліотека для надсилання HTTP–запитів. З її допомогою система звертається до відкритих API різних торговельних майданчиків або сайтів магазинів для отримання структурованої інформації (наприклад, у форматі JSON) [15].

Переваги: простота використання, висока швидкість, підтримка авторизації та заголовків. BeautifulSoup – бібліотека для розбору та аналізу

HTML–коду сторінок. Застосовується у випадках, коли відповідне API відсутнє, і дані необхідно отримувати безпосередньо зі структури вебсторінки.

Дає змогу «витягнути» текстові або числові дані з HTML–елементів (наприклад, назва товару, ціна, наявність). Selenium – фреймворк для автоматизованого керування браузером, що використовується для обробки динамічних сторінок, вміст яких завантажується через JavaScript.

Дає змогу «емуляції» дій користувача в браузері: відкривати сторінки, прокручувати, натискати на елементи – і таким чином отримувати необхідну інформацію навіть із захищених або складних ресурсів. Обробка та підготовка даних: Після збору цінової інформації необхідно виконати її первинну обробку та підготувати до збереження у базі даних або подальшої візуалізації: pandas – потужна бібліотека для роботи з табличними даними. Дає змогу зручно структурувати зібрані значення, виконувати фільтрацію, сортування, агрегацію та інші операції над масивами даних. datetime – використовується для обробки часових міток, необхідних для фіксації моменту зняття ціни (формування історії). json – дозволяє працювати зі структурами у форматі JSON, які часто використовуються у відповіді API, а також для серіалізації даних перед збереженням або передачею. Переваги такого підходу: Автоматизація процесу збору даних виключає необхідність ручного оновлення цін; Гнучкість – можливість адаптувати збір під будь–який тип сайту: з API, зі статичним HTML або динамічним вмістом; Сумісність з іншими модулями вебдодатку, зокрема обробкою у Flask та збереженням у SQLite. Таким чином, використання зазначених інструментів забезпечує системі надійну основу для автоматичного моніторингу цін, що дозволяє реалізувати ключову функціональність проєкту – оперативне отримання, обробку та подальший аналіз змін у вартості товарів.2.3 Архітектура системи та моделі даних [16].

Розробка будь–якого інформаційного вебдодатку неможлива без ретельно продуманої архітектури та логічної моделі зберігання й обробки даних. Це особливо актуально у випадку систем, які працюють із динамічними та регулярно оновлюваними масивами інформації – зокрема, ціновими

показниками на товари в онлайн–магазинах. Саме тому доцільно побудувати чітку структуру взаємодії між компонентами системи, а також визначити основні сутності (моделі), що будуть лежати в основі бази даних і всієї логіки додатку.

2.3.1 Загальна архітектура системи

У якості основи для реалізації проєкту обрано класичну трирівневу архітектуру: фронтенд (інтерфейс користувача), бекенд (серверна логіка) та база даних (перманентне збереження інформації). Такий підхід дозволяє досягти відокремленості компонентів, покращити масштабованість, забезпечити зрозумілу логіку роботи додатку та спростити його подальший супровід.

Користувач взаємодіє з додатком через інтерфейс, реалізований у вигляді вебсторінок. Цей інтерфейс надсилає запити до серверної частини, яка опрацьовує ці запити, взаємодіє з базою даних або зовнішніми API/сайтами, і повертає відповідь у зручному форматі [17].

2.3.2 Опис основних моделей даних

Користувач (User). Ця модель відображає інформацію про осіб, які використовують вебдодаток. Кожен користувач має унікальний обліковий запис, що дозволяє йому створювати персоналізовані списки товарів, встановлювати сповіщення та переглядати історію цін. Поля моделі:

- **ID** – унікальний ідентифікатор користувача;
- **Ім’я користувача** – текстове поле, яке дозволяє персоналізувати інтерфейс;
- **Email** – використовується як логін і для надсилання сповіщень;
- **Пароль** – зберігається в зашифрованому вигляді для безпеки;
- **Дата реєстрації** – фіксує, коли було створено акаунт.

Ця модель є основою для всієї взаємодії користувача із системою, і з нею пов’язано інші сутності, такі як список товарів, сповіщення, історія переглядів тощо.

Товар (Product). Модель «Товар» є однією з центральних у системі, оскільки саме ці об'єкти відстежуються з точки зору їх вартості. Кожен

користувач може додавати власні товари для моніторингу – зазвичай це посилання на конкретну сторінку товару з певного магазину. Поля моделі:

- **ID** – унікальний ідентифікатор товару;
- **Назва товару** – зручна для користувача назва;
- **Категорія** – наприклад, “ноутбуки”, “смартфони”, “побутова техніка” тощо;
- **Посилання (URL)** – адреса сторінки з товаром у магазині;
- **ID користувача** – прив’язка до акаунта, якому належить цей запис.

Ця модель дозволяє організувати індивідуальні списки спостереження, виводити товари на головну сторінку кабінету користувача, сортувати й фільтрувати за категоріями або назвою.

Цінова Інформація (PriceEntry). Модель «Цінова інформація» є фактичним ядром системи моніторингу. Вона зберігає результати збору даних, які відображають зміну вартості конкретного товару в певний момент часу. Ці записи накопичуються й використовуються для аналітики та побудови графіків. Поля моделі:

- **ID** – унікальний ідентифікатор запису;
- **ID товару** – вказує, до якого товару належить цей ціновий показник;
- **Ціна** – числове значення вартості товару;
- **Дата та час збору** – дозволяє побудувати історію цін;
- **Джерело (магазин)** – назва або домен онлайн–магазину, з якого отримано ціну.

Ця модель безпосередньо підживлює графіки й сповіщення, є джерелом статистичної інформації та формує історичний контекст для кожного продукту.

Сповіщення (Notification). Модель «Сповіщення» відповідає за інформування користувачів про зниження ціни до бажаного рівня. Користувач може вказати ціновий поріг, і система, аналізуючи нові значення, автоматично повідомить його у разі досягнення цієї межі. Поля моделі:

- **ID** – унікальний ідентифікатор сповіщення;
- **ID користувача** – кому адресоване сповіщення;

- **ID товару** – товар, для якого діє сповіщення;
- **Ціновий поріг** – бажана ціна, нижче або рівно якій надсилається сповіщення;
- **Статус** – чи було вже надіслано сповіщення (наприклад, "очікує", "відправлено").

Ця модель забезпечує інтерактивність і динамічність системи, підвищує її корисність для кінцевого користувача й економить час, адже немає потреби щодня перевіряти ціни вручну: [18].

2.4 UX/UI дизайн вебінтерфейсу, неймінг та логотип.

У сучасному цифровому просторі користувач щодня має доступ до безлічі вебдодатків, що пропонують подібні функціональні можливості. В умовах інформаційного перенасичення вирішальну роль у виборі сервісу починає відігравати не стільки функціонал, скільки якість взаємодії – користувацький досвід (UX) і візуальне оформлення інтерфейсу (UI). Якщо технічна складова є своєрідним «двигуном» системи, то інтерфейс виступає її «обличчям», із яким користувач контактує першим – воно створює перше враження, викликає довіру або, навпаки, відштовхує [19].

Розроблений у межах цієї роботи вебдодаток працює з динамічно змінною інформацією, яка оновлюється на регулярній основі. Це означає, що користувач постійно перебуває у процесі спостереження, аналізу та прийняття рішень. У такому контексті особливо важливо, аби інтерфейс не перевантажував, не відволікав зайвими деталями, а навпаки – спрощував взаємодію й був інтуїтивно зрозумілим навіть для недосвідченого користувача. Саме тому UX/UI–дизайн у цьому випадку є не допоміжним, а базовим компонентом ефективності створеного рішення.

Процес проектування користувацького досвіду розпочнеться з розроблення сценаріїв взаємодії (user flow), які відображають послідовність дій, що виконує користувач під час роботи з вебдодатком. Основними сценаріями було визначено наступні:

- **Вхід/реєстрація:** користувач створює новий акаунт або входить до існуючого за допомогою спрощеної форми. При цьому система перевіряє унікальність електронної пошти, пропонує безпечні варіанти паролів і підтримує функцію збереження сесії.
- **Додавання товару до списку:** користувач може вставити посилання на товар або скористатися вбудованим пошуком. Після вибору елемент додається до списку спостереження, з якого автоматично розпочинається збір даних.
- **Перегляд історії цін:** при переході до товару відображається графік змін вартості з датами, джерелами та додатковими показниками – середньою, мінімальною та максимальною цінами.
- **Налаштування сповіщень:** користувач задає бажаний ціновий поріг, і при його досягненні система надсилає відповідне повідомлення.
- **Редагування або видалення товару:** через контекстне меню надається можливість змінити параметри позиції або повністю її видалити зі списку.

Кожен із кроків має бути зрозумілим без додаткових пояснень. З цією метою макети інтерфейсу створювались у Figma з урахуванням ключових принципів юзабіліті: мінімізація кількості кліків до цільової дії, наявність візуальних підказок, достатня контрастність елементів управління та інтуїтивне розміщення функціональних блоків.

2.4.1 Неймінг та формування ідентичності

Процес вибору назви для вебдодатку – це не лише складова брендингу, а й стратегічне рішення, яке повинно передавати суть сервісу, бути легко впізнаваним, адаптивним до масштабування та унікальним на фоні конкурентів. Ім’я проєкту має викликати довіру, бути універсальним для різних платформ і чітко асоціюватися з основною функцією системи.

У результаті аналізу кількох потенційних варіантів, серед яких були:

- PriceTrack
- WatchCost
- Priceradar
- Costpulse
- Monitag

Фінальний вибір зупинився на назві MoniTag (поєднання слів money і tag). Це коротка, виразна й смислова назва, яка добре звучить різними мовами, легко запам’ятовується та водночас чітко передає концепцію: моніторинг обраних товарів і їхнє позначення чи маркування.

Назва MoniTag вдало підходить для використання у доменному імені, логотипі, мобільному додатку й у соцмережах. Її універсальність не обмежується лише товарною тематикою – вона залишає простір для масштабування, наприклад, на послуги або підписки, у разі розширення функціональності сервісу в майбутньому.

2.4.2 Роль логотипа та його значення.

Логотип – це перший візуальний контакт користувача з продуктом. Він виконує не лише естетичну функцію, а й уособлює цінності, стиль і функціональну суть вебдодатку. В умовах насиченого цифрового середовища, де увага користувача розсіюється за лічені секунди, вдало створений логотип може стати вирішальним аргументом на користь вибору саме MoniTag серед численних альтернатив – у браузері чи на екрані смартфона. Під час розробки логотипа було поставлено кілька ключових завдань:

- візуалізувати ідею спостереження та відстеження;
- закласти у символіку натяк на аналітику або динаміку змін;
- зберегти мінімалістичний стиль і забезпечити адаптивність – логотип має виглядати доречно як у великих форматах, так і у компактному вигляді (favicon, іконка застосунку);
- відповідати сучасним графічним трендам, щоб уникнути візуального старіння у найближчі роки.

У результаті буде створений абстрактний знак у формі кола, трохи нахиленого вправо – як символ розвитку. Графічна частина супроводжується написом “MoniTag”, виконаним сучасним геометричним гротескним шрифтом без зарубок.

2.4.3 Користувацькі сценарії та логіка взаємодії (User Journey).

Одним із ключових етапів проєктування користувацького досвіду є моделювання логіки взаємодії – послідовності дій, які виконує користувач для досягнення цільової мети. У випадку з MoniTag:

- **Перший контакт із системою**. Користувач потрапляє на головну сторінку, де одразу бачить зрозуміле пояснення сервісу, кнопку «Спробувати» або «Зареєструватися». Інтерфейс у цьому моменті має бути максимально простим: логотип, слоган, коротка презентація функцій MoniTag і декілька іконок з ключовими можливостями.
- **Реєстрація/вхід.** Натиснувши «Зареєструватися», користувач потрапляє на форму, що містить лише три поля: email, пароль і підтвердження пароля. Передбачені підказки щодо заповнення, індикатор складності пароля, а також опція показу/приховування введених символів.
- **Додавання товару.** Одразу після успішної реєстрації система пропонує додати перший товар. Користувач вставляє посилання (наприклад, із сайтів Rozetka чи Prom.ua). Якщо магазин підтримується, MoniTag автоматично зчитує назву, зображення, актуальну ціну та категорію. Якщо ні – користувач отримує відповідне повідомлення й може внести дані вручну.
- **Перегляд інформації про товар.** Після додавання позиції відображається її картка з актуальною ціною та кнопкою «Показати графік змін». Візуалізація цін реалізована у вигляді графіка (лінійного або стовпчикового), що відображає коливання вартості за останні 7, 30 або 90 днів. Доступний вибір періоду для перегляду.

- **Налаштування сповіщення.** Під графіком розміщено блок, у якому користувач задає бажану ціну. Після її досягнення він отримує сповіщення. Можна обрати тип повідомлення – email або браузерне сповіщення.
- **Подальша взаємодія.** Користувач має змогу редагувати або видаляти товари, переглядати статус відстеження, змінювати налаштування, сортувати позиції за ціною, відслідковувати загальну економію та аналізувати зміни цін у динаміці.

Кожен із цих кроків проєктувався з урахуванням зниження когнітивного навантаження на користувача – інтерфейс має «наводити» на дію, а не змушувати її шукати.

2.4.5 Принципи побудови макетів у Figma

Для розробки інтерфейсу було використано інструмент Figma, який надав можливість:

- створити всі макети на основі адаптивної 12–колонкової сітки;
- застосовувати компоненти та стилі, що забезпечують повторюваність елементів і спрощують внесення змін;
- протестувати інтерфейс у режимі прототипування – з переходами між екранами, анімацією та перевіркою поведінки на різних пристроях.

Під час проєктування враховувалися ключові принципи UX/UI:

- **Візуальна ієрархія:** акценти розставлені на головних елементах – назвах, кнопках, цінах, графіках;
- **Простота та послідовність:** використані єдині стилі для шрифтів, кольорів та елементів керування;
- **Контрастність:** важливі кнопки чітко виділяються на фоні, ключові елементи підкреслені кольором;
- **Фокус на цільовій дії:** наприклад, кнопка «Додати товар» має візуальний акцент для залучення уваги;

- **Мобільна адаптивність:** створено окрему версію для смартфонів зі спрощеною структурою – зміненим меню, спрощеним відображенням товарів і прихованими другорядними блоками.

Усі макети умовно поділялися на такі структурні блоки:

- **Хедер:** логотип, навігаційне меню, кнопка виходу або доступ до профілю;
- **Основна частина:** список товарів, графіки цін, налаштування;
- **Футер:** контактна інформація, посилання, короткий опис сервісу.

2.4.6 Вплив дизайну на довіру та залучення користувача

Окрім реалізації функціональних можливостей, веб–додаток повинен формувати відчуття довіри. Це особливо важливо для сервісів, які працюють із персональними обліковими записами, регулярно збирають дані та надсилають сповіщення. У цьому контексті велику роль відіграють не лише технічні, а й візуальні та емоційні аспекти взаємодії. Серед ключових елементів, що сприяють довірі користувача:

- **Простота інтерфейсу та відсутність надлишкових деталей:** усе виглядає чисто, дії чітко окреслені;
- **Передбачувана поведінка системи:** користувач отримує саме ту реакцію, яку очікує після взаємодії з елементами;
- **Зрозумілі повідомлення:** замість технічних помилок або кодів – прості фрази, як–от “Товар додано”, “Ціну оновлено о 14:32”;
- **Брендова цілісність:** наявність логотипа, фірмових кольорів і стилю створює враження стабільності та професійності.

Дизайн також виконує роль емоційного підсилювача. Наприклад, коли ціна на товар знижується, графік підсвічується зеленим кольором – це створює у користувача приємне відчуття вигоди чи перемоги. Анімації під час додавання товарів, зміни параметрів або появи сповіщень не лише інформують, а й роблять досвід використання легким та приємним на емоційному рівні.

Висновок до другого розділу

У другому розділі було здійснено поетапне проєктування майбутнього вебдодатку для моніторингу й аналізу цін на обрані товари, що дозволило сформувати чітке уявлення про функціональні межі, технічну архітектуру та логіку взаємодії ключових складових системи.

В результаті аналізу предметної області сформульовано технічне завдання, яке охоплює основні функціональні вимоги: авторизацію користувача, додавання товарів, збирання даних із зовнішніх джерел, побудову історії змін цін, графічну візуалізацію, налаштування сповіщень та створення персоналізованого кабінету.

Вибір технологічного стеку було обґрунтовано з урахуванням сучасних тенденцій веброзробки та практичних навичок розробника. У якості серверної частини обрано Python із використанням Flask або Django, дизайн реалізовано у Figma, а для зберігання даних – SQLite або PostgreSQL.

Визначено архітектурну модель системи на основі принципу клієнт–серверної взаємодії з розподілом на фронтенд, бекенд і базу даних. Описано основні логічні сутності – користувач, товар, ціна, сповіщення – з акцентом на їхню структуру, функціональність і взаємозв’язки. Такий підхід забезпечує гнучкість, стабільність і простоту обслуговування вебдодатку.

Особливу увагу приділено дизайну інтерфейсу, як одному з ключових елементів користувацької привабливості. Розглянуто базові сценарії взаємодії, побудову логіки інтерфейсу, макетування у Figma, з фокусом на доступність, адаптивність, емоційну привабливість і мінімізацію когнітивного навантаження.

У межах розробки візуальної ідентичності було створено назву проєкту MoniTag, що об’єднує смислові образи моніторингу, динаміки та точності. Детально описано концепцію графічного символу, типографіку, колірну схему, а також роль логотипа у формуванні довіри до продукту.

Отже, другий розділ заклав надійну логічну, технічну й візуальну основу, яка слугуватиме фундаментом для реалізації частини проєкту, що буде висвітлена у наступному розділі.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ДОДАТКУ

3.1 Реалізація front–end частини

Візуальна складова вебдодатку, орієнтованого на роботу з аналітичними даними, повинна не лише забезпечувати привабливий зовнішній вигляд, а й бути функціональною, зручною у використанні та сприяти ефективній взаємодії з інтерфейсом.

У контексті MoniTag особливо важливо сформувати дизайн, який дозволяє швидко орієнтуватися в інформації про зміну цін, зручно сприймати графіки та гнучко налаштовувати параметри без візуального перевантаження або надмірної складності.

3.1.1. Реалізація логотипу

Рисунок 3.1.1 – Логотип “MoniTag”

Логотип MoniTAG – це сучасне, динамічне й детально продумане візуальне рішення для цифрової платформи.

Композиція логотипа поєднує стилізований символ ліворуч і чітку шрифтову частину праворуч. Разом вони формують сильну візуальну ідентичність, яка легко впізнається та добре масштабується для цифрового й друкованого використання. Це не просто назва – це знак, що передає ключові цінності бренду: точність, швидкість, технологічність і фокус на користувачеві.

Графічний знак розташовано зліва – він має форму хрестоподібної фігури, складеної з чотирьох стрілок, спрямованих у центр. Така структура символізує:

- збір і моніторинг даних з різних джерел; концентрацію уваги на найвигідніших пропозиціях;
- технічну точність сервісу;
- централізацію інформації в одному місці.

Червоний колір символу виконує сигнальну функцію – він асоціюється з динамікою, ексклюзивністю та швидкістю реакції. Візуально форма також перегукується з тегом або міткою – пряме посилання на слово “TAG” у назві.

Напис “MoniTAG” виконано в геометричному гротеску, що виглядає сучасно й професійно. Він поділений на дві смислові частини: “Moni” – вказує на аналітику, моніторинг, спостереження; “TAG” – акцентує на товарних тегах, категоризації, фільтрах. Велика літера “T” створює візуальний ритм і логічне відокремлення частин, що підсилює запам’ятовуваність. Чорний колір тексту додає логотипу відчуття глибини, надійності та цифрової строгості.

Рисунок 3.1.2 – Знак з логотипу “MoniTag”

Знак логотипа MoniTAG – це не просто візуальний елемент, а концентрований носій смислів, який поєднує естетику, функціональність і глибину ідеї.

Його основою є проста геометрична сітка, що надає формі логічності, а бренду – чіткої впізнаваності та універсальності.

Композиція знака складається з чотирьох чорних квадратів, розміщених хрестоподібно навколо центрального порожнього квадрата, а також чотирьох діагонально спрямованих стрілоподібних трикутників. Разом вони створюють: симетричну, центровану фігуру – як символ стабільності й балансу; динамічну структуру, що викликає асоціації з рухом, потоками даних і системною логікою.

Це не випадкова форма, а метафора суті бренду – збирання, моніторинг і концентрація інформації в одному місці.

Знак можна трактувати в кількох смислових напрямках:

- **Моніторинг і фокусування**. Стрілки, спрямовані до центру, уособлюють потоки даних з різних джерел, які зводяться в одну точку – так працює MoniTAG.
- **Тег або мітка.** Форма натякає на цифровий “тег” – елемент, що асоціюється з категорією, акцією чи цінником. Це створює прямий зв’язок із контекстом товарів і пропозицій.
- **Цифрова естетика**. Чітка модульна структура й чорний колір перегукуються з піксельними або інтерфейсними іконками – надаючи знаку технологічного характеру.

Геометрія та логіка побудови:

- Використання сітки забезпечує ідеальну модульність і збереження якості при масштабуванні;
- Діагоналі додають динаміки й напрямку;
- Контраст чорного на червоному фокусує увагу та покращує видимість у будь–якому середовищі.

Завдяки продуманому мінімалізму, символ легко адаптується:

- Інвертується (на темному/світлому фоні);
- Анімується (стрілки можуть з’єднуватися до центру);
- Перетворюється на патерн;
- Інтегрується в інші елементи айдентики.

3.1.2 Реалізація кольорової палітри

Кольори відіграють ключову роль у сприйнятті вебінтерфейсу – вони визначають не лише емоційний фон, а й забезпечують ефективну навігацію, розставляють візуальні акценти, позначають повідомлення та індикатори стану, як–от зростання чи падіння цін, помилки або підтвердження дій. Під час формування палітри було враховано сучасні підходи до UI–дизайну (flat, material, neutral–базовані рішення), стандарти доступності WCAG (контрастність і зручність читання), потребу підтримки темної та світлої теми, а також психологічні асоціації з кольорами – довіру, стабільність, аналітичність. Основна палітра включає такі кольори:

- Світло чорний (#1C1A1B) – базовий колір для шапки сайту, основних кнопок та елементів навігації. Передає відчуття надійності й професійності.
- Графітовий сірий (#2A2E35) – допоміжний, особливо ефективний у темному режимі. Добре працює як нейтральний фон.
- Білий (#FFFFFF) – використовується для блоків вмісту, карток товарів, створює легкий і ненав'язливий фон.
- Червоний (#FF4B4B) – застосовується для попереджень, повідомлень про помилки або підвищення ціни.

Переваги обраної палітри:

- Чітка контрастність між фоновими й основними кольорами;
- Візуально збалансоване поєднання нейтральних і акцентних тонів;
- Відсутність візуального перенавантаження, комфорт для очей.

Рисунок 3.1.3 – Кольорова палітра "MoniTag"

3.1.3 Реалізація шрифтових композицій

Типографіка – ще один ключовий елемент, що впливає на зручність користування й загальне враження від інтерфейсу. Для MoniTag обрано шрифт,

який поєднує сучасну естетику, хорошу читабельність та повну підтримку української й англійської мов.

Основний шрифт інтерфейсу: Onest – відкритий геометричний гротеск, оптимізований для цифрового використання. Його переваги:

- збереження високої читабельності при різних розмірах;
- підтримка необхідних мовних символів і валют; збалансоване поєднання технічного стилю з м’якістю подачі;
- універсальність у застосуванні – від заголовків до дрібного тексту.

Рекомендовані розміри шрифтів:

- Заголовки: 20–28 pt (h1–h2), переважно жирні або напівжирні;
- Основний текст: 14–16 pt, стандартна товщина;
- Допоміжна інформація: 12 pt, з використанням сірого кольору або вторинного стилю для зниження візуальної ваги.

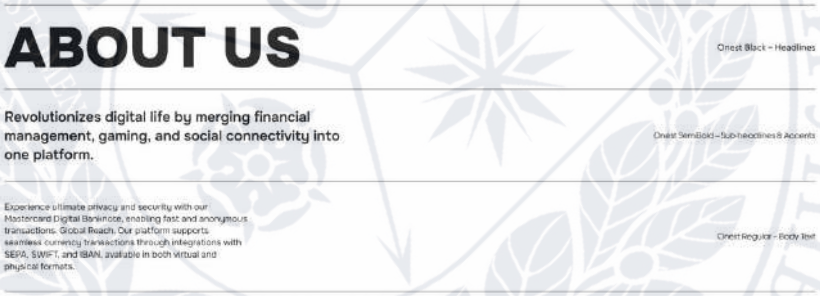

Рисунок 3.1.4 – Основний шрифт інтерфейсу “MoniTag”

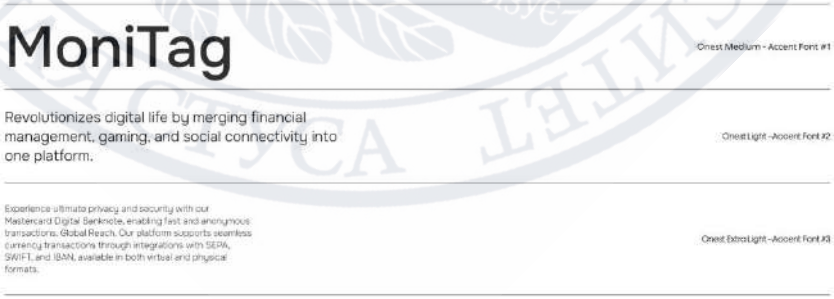

Рисунок 3.1.5 – Акцентний шрифт інтерфейсу “MoniTag”

3.1.4 Реалізація UX – частини вебдодатку

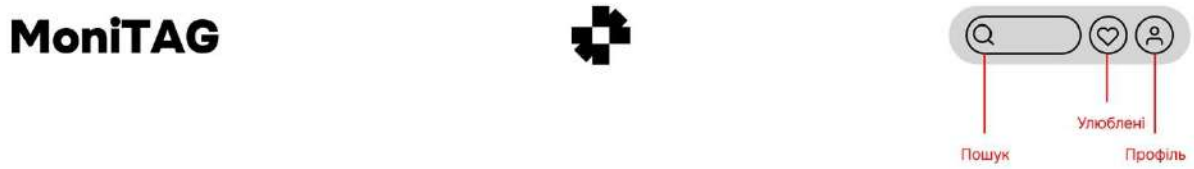

Рисунок 3.1.6 – UX–Макет блоку “Хедер” у вебдодатку “MoniTAG”

Header – це верхня частина інтерфейсу, яка поєднує в собі елементи брендування, навігації та швидкого доступу до ключових функцій. Він завжди залишається на екрані, розташований у верхній зоні сторінки, і має симетричну структуру, що поділяється на три основні частини – ліва, центральна та права частини:

Ліва частина – Логотип–шрифтова частина:

- Назва – MoniTAG. Брендова назва, в якій акцент на “TAG” вказує на процес тегування та відстеження товарів. Типографіка – жирний геометричний гротеск.

Центральна частина – Логотип–символ:

- Фірмовий знак – абстрактний символ. Символізує зосередженість даних, увагу до ціни й інтеграцію товарів в одному сервісі.

Права частина – панель навігації:

- Оформлена як округлена прямокутна панель з іконками. Піктограми роблять інтерфейс зрозумілим та не навантажують його. Складові цієї панелі: 1. Пошук: іконка лупи. Призначений для введення запиту і швидкого пошуку потрібного товару. 2. Улюблені: іконка серця. Тут товари, що користувач позначив як обрані. 3. Профіль: іконка людини. Відповідає за вхід, перегляд персональних даних і налаштувань.

Окрім того, така побудова забезпечує функціональні можливості Sticky Header – блок залишається на місці при скролі, тому користувач має постійний доступ до основних інструментів. Дизайн іконок дозволяє комфортно працювати як на мобільних пристроях, так і на комп’ютерах. Чіткі зображення та текстові пояснення полегшують використання навіть новачкам.

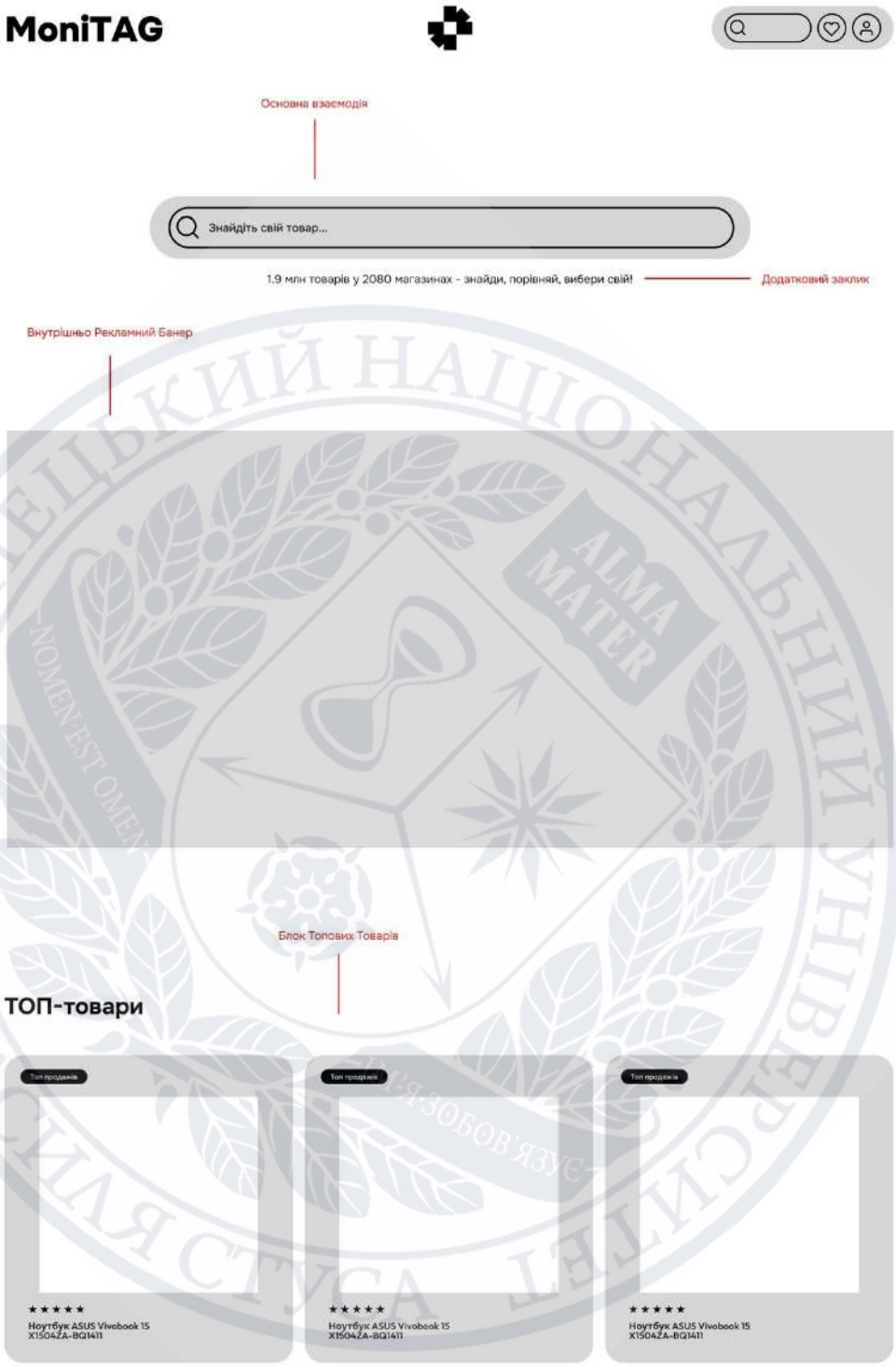

Рисунок 3.1.7 – UX–Макет блоку “Головна сторінка” у вебдодатку “MoniTAG”

Головний екран MoniTAG. Основна взаємодія відбувається через – пошуковий блок (велике поле із заокругленими кутами). Це ключовий інструмент користувача для взаємодії з платформою: дозволяє швидко знайти потрібний товар.

- Оформлення лаконічне: чітке поле для введення, іконка лупи та текст–запрошення «Знайдіть свій товар...», який спонукає до дії.
- Додатковий заклик Фраза «1.9 млн товарів у 2080 магазинах – знайди, порівняй, вибери свій!» слугує яскравим акцентом і водночас маркетинговим меседжем. Вона підкреслює масштаб сервісу та заохочує користувача розпочати пошук. Розміщення – відразу під пошуком, що підтримує логіку візуального потоку.

Внутрішньо рекламний банер – прямокутний блок великого розміру призначений для розміщення графіки або банерів з актуальними пропозиціями, акціями, партнерськими товарами. Цей візуальний елемент покликаний підсилити взаємодію з користувачем і може динамічно змінюватися в залежності від контенту.

Блок топових товарів Заголовок «ТОП–товари» виділено жирним – він привертає увагу до найпопулярніших позицій. Карточки мають прямокутну форму, містять позначку «Топ продажів», зірковий рейтинг, назву й модель товару. Сітка з трьох елементів із достатнім білим простором забезпечує зручний перегляд незалежно від типу пристрою.

Рисунок 3.1.8 – UX–Макет блоку “Список Бажань” у вебдодатку “MoniTAG”

Список Бажань – це персоналізований розділ у MoniTAG, де зберігаються обрані користувачем товари. Структура сторінки:

Заголовок «Список Бажань» розміщується під логотипом і виконує роль орієнтиру на сторінці.

Основна частина побудована у форматі сітки 3×3 товарних карток, яка автоматично розширюється під час скролу.

Картки товарів. Кожна картка містить стандартний набір елементів (позначка «Топ продажів» у верхньому лівому куті – сигналізує про популярність позиції. Зображення товару розміщується по центру, у великому прямокутному блоці. Під фото – рейтинг у вигляді зірок. Повна назва, наприклад: «Ноутбук ASUS Vivobook 15 X1504ZA–BQ1411». Ціна зазначена у гривнях, розташована в нижньому правому куті, в акцентному полі.

Особливості – усі картки виконані в єдиному стилі – це забезпечує чітке, впорядковане представлення інформації.

UX–фокус – інтерфейс логічно побудований: користувач легко орієнтується у вибраних позиціях. Візуальні акценти зосереджені на зображеннях і ціні, що спрощує сприйняття. Уся навігація зберігає загальну стилістику сервісу, підтримуючи відчуття цілісного інтерфейсу.

Рисунок 3.1.9 – UX–Макет блоку “Картка товару” у вебдодатку “MoniTAG”

Заголовок і структура сторінки:

- **Назва товару** – подається виразно, праворуч від зображення. Фото пристрою велике, чітке, розміщене ліворуч – акцент на візуальному сприйнятті.
- **Опис** – стисла технічна інформація з ключовими перевагами;
- **Виділено ціновий діапазон**;
- **Пропозиції магазинів.** Кожен варіант покупки містить: Назву продавця (наприклад, "Ябко", "Yabluka", "Алло") – оформлену як заголовок. Логотип ліворуч для зручної ідентифікації бренду. Повтор коротких характеристик моделі. Умови доставки й гарантії.
- **Ціна** – вказується у чорному полі, помітно і без відволікань. Для магазину з найнижчою ціною застосовується маркування “Найкраща ціна” – зверху відповідного блоку.
- **Елемент “Більше”** – Кнопка внизу сторінки дає змогу завантажити додаткові пропозиції. Це особливо зручно, коли товар представлений у кількох магазинах.

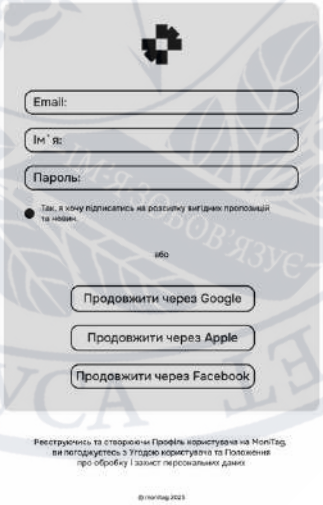

Рисунок 3.1.10 – UX–Макет блоку “Створити аккаунт” у вебдодатку “MoniTAG”

Форма реєстрації MoniTAG – мінімалістичний, інтуїтивно зрозумілий інтерфейс, створений для швидкої регестрації. Основні поля для введення:

- Email – стандартне текстове поле для введення електронної адреси.

- Ім’я – дає змогу персоналізувати обліковий запис.
- Пароль – поле з прихованим введенням для безпеки.

Усі поля розміщено вертикально, вони мають однакову ширину та заокруглені кути, що робить процес заповнення простим і комфортним. Підписка на розсилку Опціональний чекбокс з текстом – «Так, я хочу підписатись на розсилку вигідних пропозицій та новин.». Функція – інформування користувачів про акції, знижки та новини сервісу.

Альтернативна авторизація. Передбачено три кнопки для входу через сторонні сервіси:

- Продовжити через Google
- Продовжити через Apple
- Продовжити через Facebook

Це дозволяє швидко пройти реєстрацію без створення нового паролю та спрощує процес для більшості користувачів.

Юридичний блок. Текст – «Реєструючись та створюючи Профіль користувача на MoniTAG, ви погоджуєтесь з Угодою користувача та Положенням про обробку і захист персональних даних.» Цей блок виконує роль повідомлення про політику конфіденційності й захист персональних даних.

3.1.5 Реалізація UI – частини вебдодатку

Завершальний етап створення користувацького інтерфейсу є не лише кульмінацією візуального проектування, а й ключовим моментом, коли всі попередні концепції, дизайнерські рішення та теоретичні підходи втілюються в реальну, взаємодійну систему. Саме на цьому етапі абстрактні напрацювання, графічні прототипи та стилістичні припущення трансформуються у практичний, інтерактивний продукт, з яким безпосередньо взаємодіятиме кінцевий користувач.

Від ефективності реалізації UI залежить, наскільки інтуїтивно зрозумілою та комфортною буде взаємодія з вебдодатком, як швидко користувач орієнтуватиметься в інтерфейсі, здійснюватиме цільові дії, сприйматиме інформацію та прийматиме рішення. Користувацький інтерфейс виступає

візуальним і функціональним фасадом додатку, який формує перше враження, створює емоційний зв’язок і впливає на готовність повертатися до сервісу знову.

Тому реалізація інтерфейсної частини передбачала не лише технічне впровадження окремих компонентів, а й ретельне дотримання стилістичної єдності, відповідності елементів наперед визначеній візуальній системі, гармонійного використання палітри кольорів, шрифтів та ієрархій. Окрему увагу було приділено адаптивності інтерфейсу.

Рисунок 3.1.11 – UI–вигляду блоку “Хедер” у вебдодатку “MoniTAG”

Рисунок 3.1.13 – UI–вигляду блоку “Створити профіль” у вебдодатку “MoniTAG”

Рисунок 3.1.12 – UI–вигляду блоку “Головна ” у вебдодатку “MoniTAG”

Рисунок 3.1.14 – UI–вигляду блоку “Список Бажань” у вебдодатку “MoniTAG”

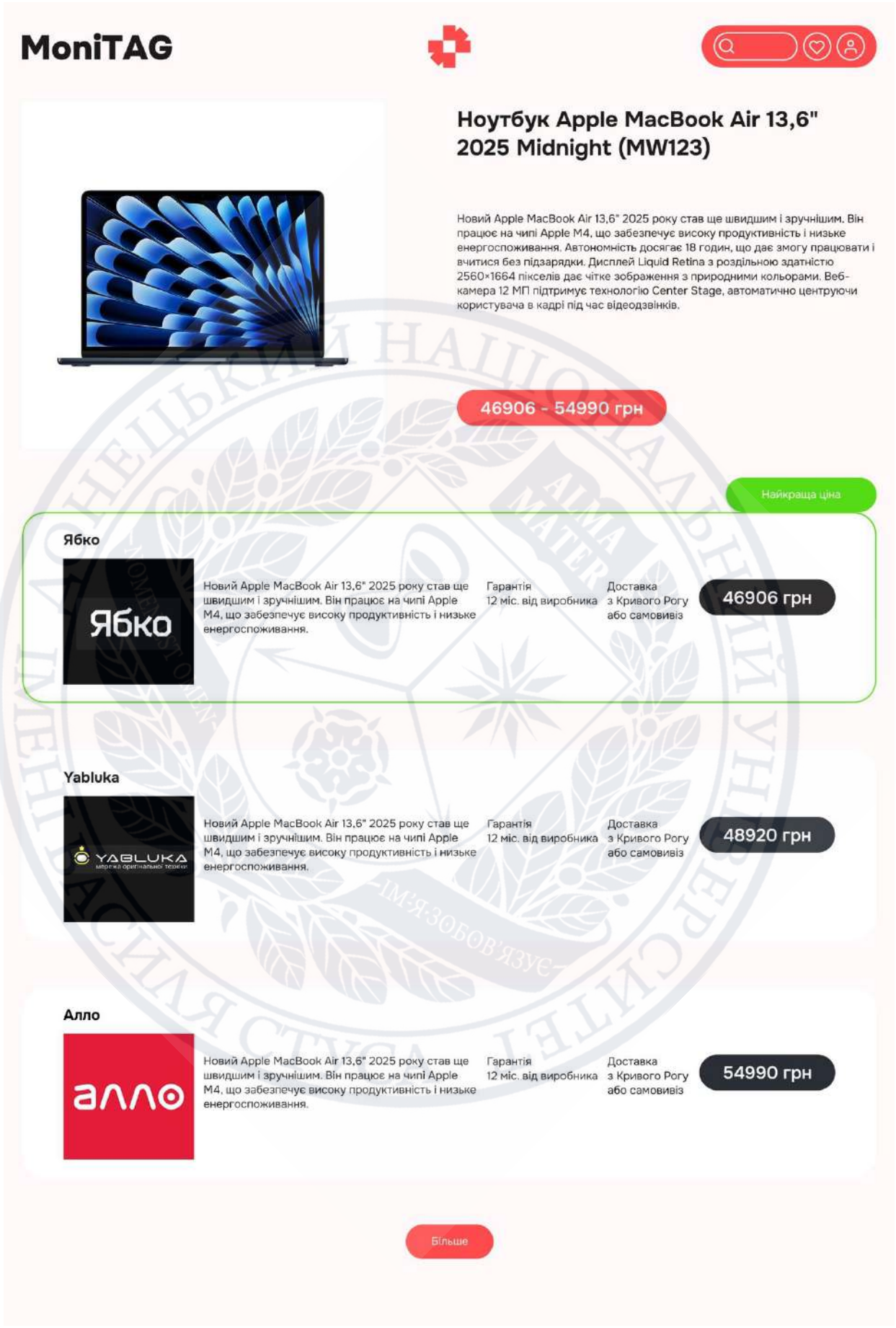

Рисунок 3.1.15 – UI–вигляду блоку “Картки товару” у вебдодатку “MoniTAG”

3.2 Реалізація back–end частини

Back–end частина вебдодатку для відстеження та аналізу цін на обрані товари реалізована за допомогою мови Python, фреймворку Flask та бази даних SQLite. Обраний стек технологій забезпечує просту, надійну й водночас гнучку архітектуру, що дозволяє ефективно реалізувати необхідний функціонал для обробки запитів, зберігання даних та взаємодії з front–end частиною.

Flask

Flask – це легкий мікрофреймворк для Python, який дозволяє швидко створювати вебдодатки та REST API. Його перевагами є: зрозуміла та гнучка система маршрутизації; підтримка розширень та інтеграцій; можливість швидкої реалізації CRUD–операцій. У проєкті Flask використовується для обробки HTTP–запитів, створення маршрутів для додавання, редагування, видалення та перегляду товарів, а також для управління користувачами та їх персональними списками обраного.

SQLite

SQLite – це вбудована реляційна база даних, яка не потребує запуску окремого сервера. Вона ідеально підходить для легких і середніх проєктів, особливо на етапі розробки MVP. У нашому додатку SQLite використовується для зберігання: інформації про користувачів; даних про товари, що відстежуються; історії зміни цін; інших допоміжних таблиць (категорії товарів, магазини тощо). Для взаємодії з базою даних використовується ORM SQLAlchemy, що забезпечує зручну роботу з таблицями через Python–код, з автоматичним перетворенням об’єктів у SQL–запити.

3.3 Результати роботи вебдодатку

У результаті реалізації дипломного проєкту було створено вебдодаток, який дозволяє користувачам ефективно відстежувати зміну цін на обрані товари та здійснювати базовий аналіз цієї динаміки.

Основні функціональні результати:

- Реалізована класична система входу, де дані користувача зберігаються у базі даних.
- Паролі хешуються для забезпечення базового рівня безпеки.
- Користувач має можливість додати товари до персонального списку для подальшого відстеження цін.
- Додаток показує актуальну інформацію щодо вартості кожного товару з доступних джерел або введених вручну даних.
- У системі реалізовано механізм збереження змін вартості товарів у часі. Це дозволяє переглядати, як змінювалася ціна певного товару протягом обраного періоду.

Інтерфейс користувача та зручність використання:

Інтерфейс реалізовано таким чином, щоб користувач мав змогу зручно

- взаємодіяти з функціоналом системи:
- переглядати список обраних товарів;
- додавати нові товари у систему;
- отримувати інформацію у зручному, зрозумілому вигляді.

Сторінки додатку відповідають сучасним вимогам до UX/UI і є адаптивними для різних пристроїв.

Технічна ефективність:

Завдяки використанню легкого стеку технологій (Python + Flask + SQLite), система демонструє високу швидкість обробки запитів. Базу даних оптимізовано для швидкого збереження та вибірки цінових даних.

Висновок до третього розділу

У процесі розробки вебдодатку для відстеження та аналізу цін було реалізовано ключові компоненти, що забезпечують як функціональну, так і візуальну цілісність системи. По–перше, бекенд–частина забезпечує обробку запитів, збереження й аналіз даних про товари та користувачів. Вона реалізована на основі Python і Flask, що дозволило побудувати легкий, гнучкий і масштабований сервер із використанням бази даних SQLite. Уся логіка

розподілена за принципами REST–архітектури, що забезпечує зручну взаємодію з фронтендом. По–друге, було створено логотип, який формує візуальний образ проєкту. Він відображає суть вебдодатку _ моніторинг змін і динаміку, а також підкреслює простоту та доступність сервісу для користувача. Логотип став основою для подальшого оформлення інтерфейсу. По–третє, розроблено інтерфейс вебсайту, що відповідає принципам сучасного UI/UX–дизайну. Він адаптивний, інтуїтивно зрозумілий і дає змогу користувачам легко керувати обраними товарами, переглядати історію змін цін та аналізувати інформацію у зручному форматі. Сукупність цих трьох складових формує завершений продукт, який виконує як технічні, так і естетичні задачі, забезпечуючи ефективну роботу системи та комфорт для користувача.

ВИСНОВОК

У межах дипломної роботи було реалізовано вебдодаток, що дозволяє користувачам ефективно відстежувати та аналізувати зміну цін на обрані товари. Основна мета – створення інструменту для зручного моніторингу цін та прийняття раціональних рішень щодо покупок – була досягнута.

У процесі роботи:

- проаналізовано наявні рішення та виділено функціональні вимоги до системи;
- спроєктовано архітектуру клієнт–серверного вебдодатку;
- реалізовано front–end частину з адаптивним та зручним інтерфейсом для взаємодії з користувачем;
- створено back–end частину з використанням Python, Flask і SQLite для обробки даних, авторизації, збереження історії змін цін та відповіді на запити;
- розроблено логотип і сформовано базову візуальну ідентичність, що підтримує цілісність дизайну; впроваджено можливість перегляду динаміки цін у візуальній формі, що допомагає користувачу оцінити тенденції зміни вартості.

Створений вебдодаток є гнучкою та функціонально завершеною системою, яку можна розвивати далі – автоматизувати збір цін із зовнішніх джерел, додати push–сповіщення, реалізувати аналітику на основі машинного навчання або створити мобільну версію.

Таким чином, розробка вебдодатку не лише вирішує поставлену практичну задачу, але й демонструє застосування сучасних технологій веброзробки для створення корисного прикладного рішення з реальним потенціалом подальшого використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бондаренко О. В. Розробка веб–застосунків: сучасні технології і методології : навч. посіб. Київ : КНЕУ, 2020. 242 с.
2. Бублик О. І., Федорчук І. С. Проєктування та розробка веб–додатків з використанням Python та Django // Вісник Житомирського державного технологічного університету. 2021. № 1(95). С. 112–117.
3. Щербак В. В. Основи UX/UI–дизайну // Науковий вісник Херсонського державного університету. Серія: Технічні науки. 2022. № 2. С. 88–93.
4. Кухаренко В. М. Автоматизований моніторинг цін: сучасні підходи // Східноєвропейський журнал передових технологій. 2020. № 5/2 (107). С. 45–51.
5. Фігурна Н. П., Кисельов І. А. Веб–дизайн : навч. посіб. Київ : КНУКіМ, 2019. 168 с.
6. Nielsen J. Usability Engineering. San Francisco : Morgan Kaufmann, 1994. 362 p.
7. Krug S. Don't Make Me Think : A Common Sense Approach to Web Usability. 3rd ed. Berkeley : New Riders, 2014. 216 p.
8. Freeman A., Sanderson P. Pro ASP.NET Core MVC 2. New York : Apress, 2017. 1017 p.
9. Mertz J. Learning Python Web Penetration Testing. Birmingham : Packt Publishing, 2016. 196 p.
10. W3C. Architecture of the World Wide Web, Volume One. URL: https://www.w3.org/TR/webarch/ (дата звернення: 25.03.2025).
11. Mozilla Developer Network. HTML, CSS, JavaScript Guidelines. URL: https://developer.mozilla.org (дата звернення: 02.04.2025).
12. Django Software Foundation. Django Documentation. URL: https://docs.djangoproject.com (дата звернення: 02.02.2025).
13. Flask Project. Flask Documentation. URL: https://flask.palletsprojects.com (дата звернення: 07.01.2025).

14. Tailwind Labs. Tailwind CSS Documentation. URL: https://tailwindcss.com/docs (дата звернення: 21.01.2025).
15. Scrapy Developers. Scrapy Documentation. URL: https://docs.scrapy.org (дата звернення: 19.01.2025).
16. REST API Tutorial. REST API Design and Best Practices. URL: https://restfulapi.net (дата звернення: 19.04.2025).
17. Figma Help Center. Figma Design Docs & Prototyping. URL: https://help.figma.com (дата звернення: 05.04.2025).
18. Google Design. Material Design Guidelines. URL: https://m3.material.io (дата звернення: 25.04.2025).
19. Smith J. Modern Full–Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker. Sebastopol : O'Reilly Media, 2021. 326 p.
20. Traversy B. Modern HTML & CSS From The Beginning : онлайн–курс. – Udemy, 2022. URL: https://www.udemy.com (дата звернення: 14.05.2025).
21. Oxylabs. Best Programming Languages for Effective Web Scraping. URL: https://oxylabs.io/blog/best–web–scraping–language (дата звернення: 24.04.2025).
22. GeeksforGeeks. Python Web Scraping Tutorial. URL: https://www.geeksforgeeks.org/python–web–scraping–tutorial/ (дата звернення: 25.05.2025).
23. ZenRows. Web Scraping in Java in 2025: The Complete Guide. URL: https://www.zenrows.com/blog/web–scraping–java (дата звернення: 29.03.2025).
24. Scrape.do. Most Popular Web Scraping Techniques & Languages. URL: https://scrape.do/blog/popular–web–scraping–techniques–explore–them/ (дата звернення: 02.05.2025).
25. Medium. Web Scraping with Python and Object–Oriented Programming. URL: https://medium.com/analytics–vidhya/web–scraping–with–python–and–object–oriented–programming–14638a231f14 (дата звернення: 06.03.2025).

26. Stack Overflow. Object Oriented Programming with Web Parsing in Python. URL: https://stackoverflow.com/questions/21970555/object–oriented–programming–with–web–parsing–in–python (дата звернення: 21.01.2025).
27. Real Python. A Practical Introduction to Web Scraping in Python. URL: https://realpython.com/python-web-scraping-practical-introduction/ (дата звернення: 10.03.2025).
28. Real Python. Beautiful Soup: Build a Web Scraper With Python. URL: https://realpython.com/beautiful-soup-web-scraper-python/ (дата звернення: 05.04.2025).
29. Meyer, Bertrand. Object-Oriented Software Construction, 2nd Edition. URL: https://bertrandmeyer.com/wp-content/upLoads/OOSC2.pdf (дата звернення: 23.04.2025).
30. GeeksforGeeks. Encapsulation in Java. URL: https://www.geeksforgeeks.org/encapsulation-in-java/ (дата звернення: 09.03.2025).
31. Arxiv. Web Data Extraction, Applications and Techniques: A Survey. URL: https://arxiv.org/abs/1207.0246 (дата звернення: 11.01.2025).
32. Arxiv. Engineering Semantic Web Applications by Using Object–Oriented Paradigm. URL: https://arxiv.org/abs/1006.4562 (дата звернення: 06.02.2025).
33. Arxiv. Deep Learning and Machine Learning, Advancing Big Data Analytics and Management: Object–Oriented Programming. URL: https://arxiv.org/abs/2409.19916 (дата звернення: 07.05.2025).
34. Arxiv. OWLOOP: A Modular API to Describe OWL Axioms in OOP Objects Hierarchies. URL: https://arxiv.org/abs/2112.15544 (дата звернення: 07.02.2025).
35. GeeksforGeeks. Object–Oriented Programming Concepts. URL: https://www.geeksforgeeks.org/object–oriented–programming–oops–concept–in–java/ (дата звернення: 24.04.2025).

36. Wirfs–Brock, R. A Brief Tour of Responsibility–Driven Design. URL: https://www.wirfs–brock.com/PDFs/A_Brief–Tour–of–RDD.pdf (дата звернення: 05.04.2025).
37. Bright Data. C# vs. Python for Web Scraping Guide. URL: https://brightdata.com/blog/how–tos/c–sharp–vs–python–for–web–scraping (дата звернення: 18.02.2025).
38. Evomi. 6 Key Programming Languages for Web Scraping & Proxies. URL: https://evomi.com/blog/6–key–languages–web–scraping–proxies (дата звернення: 07.01.2025).
39. Oxylabs. Python Web Scraping Tutorial: Step–By–Step (2025). URL: https://oxylabs.io/blog/python–web–scraping (дата звернення: 21.01.2025).
40. YouTube. Web Scraping with Python – Beautiful Soup Crash Course. URL: https://www.youtube.com/watch?v=XVv6mJpFOb0 (дата звернення: 15.04.2025).
41. Reddit. Any tips on making my web scraping scripts more Object Oriented?. URL: https://www.reddit.com/r/learnpython/comments/eybvcw/any_tips_on_making_my_web_scraping_scripts_more/ (дата звернення: 26.03.2025).
42. Lab 4. Object Oriented Programming. Web Scraping. URL: https://ocw.cs.pub.ro/courses/ewis/laboratoare/04 (дата звернення: 11.02.2025).
43. Medium. Build OOP Program with Scraping Feature in Python – Chapter 2. URL: https://medium.com/@senchooo/build–oop–program–with–scraping–feature–in–python–chapter–2–implementation–requests–e1c01ef97612 (дата звернення: 08.03.2025).
44. Антонов Ю. С., Шамарін Ю. В. Об’єктно-орієнтоване програмування: методичні рекомендації до виконання індивідуальних завдань. Вінниця: ДонНУ імені Василя Стуса, 2018. 50 с. URL: https://r2.donnu.edu.ua/server/api/core/bitstreams/43b3ab8b-7376-4eeb-9c77-f094e07d7725/content (дата звернення: 10.03.2025).

45. Refactoring.Guru. Design Patterns. URL: https://refactoring.guru/design-patterns (дата звернення: 12.02.2025).
46. GeeksforGeeks. MVC Design Pattern. URL: https://www.geeksforgeeks.org/mvc–design–pattern/ (дата звернення: 31.01.2025).
47. RESTful API Tutorial. REST API Tutorial: What is REST?. URL: https://restfulapi.net/ (дата звернення: 03.04.2025).
48. GeeksforGeeks. What is an API?. URL: https://www.geeksforgeeks.org/what–is–an–api/ (дата звернення: 20.03.2025).
49. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.

ДЕКЛАРАЦІЯ

про дотримання академічної доброчесності

Я, _____

_____

_____

_____

_____

Повністю вказується ПІБ та статус (освітня (освітньо–наукова) програма – для здобувачів вищої освіти, назва кваліфікаційної роботи)

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21–25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)