Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота
за освітнім ступенем “бакалавр”
на тему:

МАРКЕТПЛЕЙС ДЛЯ ОБМІНУ ТА ПРОДАЖУ ПІДРУЧНИКІВ МІЖ СТУДЕНТАМИ

Виконала:
здобувач IV курсу
групи КН-41
спеціальності 122 “Комп'ютерні науки”
Олександр Ігорович Кравчук

Науковий керівник:
кандидат технічних наук,
доцент Степанія Михайлівна Бабич

Рівне – 2026

ЗМІСТ

ВСТУП

Актуальність дослідження. Цифрові платформи для обміну книгами між користувачами стають дедалі актуальнішими, оскільки вони допомагають розширити доступ до книг та заохочують до сталого читання. Традиційні методи обміну книгами часто стикаються з проблемами охоплення та управління запасами, які цифрові платформи можуть вирішити, безпосередньо з'єднуючи людей та створюючи доступніший ринок.

Цифрові платформи трансформували вторинні ринки вживаних книг, зменшивши витрати на пошук та транзакції, а також розширивши доступність книг. Це дозволяє користувачам позичати, ділитися або обмінюватися книгами, допомагає підвищити освіченість та робить книги доступнішими для читачів та молоді, що навчається. Такі онлайн-системи дозволяють користувачам розміщувати списки та шукати книги, запитувати обмін та спілкуватися з іншими. Акцент на безпеці даних користувачів та зручний інтерфейс забезпечують широку доступність, сприяють сталому розвитку та покращують загальний досвід обміну книгами.

У той же час більшість платформ для обміну та продажу вживаних книг пропонують художню, нон-фікшн літературу, шкільні підручники. Але проблемою залишається доступність вузько спеціалізованих підручників, за якими вчаться студенти закладів вищої освіти. З однієї сторони, ці підручники випускаються невеликими тиражами, тому їх не так часто можна зустріти в бібліотеках або магазинах, з іншої сторони ціна на нову професійну літературу може бути занадто високою для студентів. Тому перед студентами завжди стояло актуальне питання пошуку доступної навчальної літератури. І цифрові платформи для продажу вживаних університетських підручників повинні вирішити це питання.

Метою дослідження є розробка та впровадження вебплатформи для автоматизації процесів обміну та продажу навчальних матеріалів між студентами з використанням сучасних фреймворків та методів безпечної автентифікації.

Для досягнення мети необхідно вирішити такі **завдання**:

1. проаналізувати предметну область та потреби студентської спільноти у сфері обміну навчальною літературою;
2. обґрунтувати вибір архітектури MTV та фреймворку Django для реалізації проєкту;
3. розробити проєкт системи продажу/обміну навчальною літературою;
4. реалізувати функціонал пошуку, фільтрації та багатоваріантного сортування оголошень;
5. інтегрувати систему авторизації через Google OAuth 2.0 для підвищення рівня безпеки та довіри користувачів;
6. впровадити систему логування та моніторингу активності користувачів для аудиту доступу до персональних даних.

Об’єкт дослідження – процес інформаційної взаємодії та обміну навчальними матеріалами всередині студентського середовища.

Предмет дослідження: методи, алгоритми та програмні засоби розробки динамічних вебдодатків на базі Python-фреймворку Django.

Методи дослідження:

– аналітичні (аналіз існуючих рішень та вимог користувачів);

– об'єктно-орієнтоване проєктування (розробка моделей даних);

– програмне моделювання (безпосередня розробка backend та frontend частин);

– експериментальні (тестування функціоналу в режимі Debug та аналіз логів).

Практичне значення дослідження полягає у створенні готового до експлуатації вебсервісу, який спрощує пошук необхідних підручників, мінімізує витрати студентів на навчання та забезпечує безпечний канал комунікації між продавцем і покупцем. Розроблена система логування та автентифікації дозволяє використовувати платформу в реальних умовах ЗВО з дотриманням принципів захисту персональної інформації.

Апробація і впровадження результатів дослідження. Основні теоретичні та практичні результати дослідження доповідалися та обговорювалися на звітній науковій конференції викладачів, співробітників і здобувачів вищої освіти Рівненського державного гуманітарного університету за 2025 рік (м. Рівне, 14 травня 2026 року).

Структура роботи. Кваліфікаційна робота складається з вступу, 4 розділів, висновків, списку використаних джерел та додатків. Загальний обсяг – 81 сторінок, з них 58 сторінки основного тексту. Робота містить 37 рисунків та 3 таблиці. Список використаних джерел складає 25 одиниць.

РОЗДІЛ 1
АНАЛІЗ ПІДХОДІВ ДО ОРГАНІЗАЦІЇ ОБМІНУ КНИГАМИ МІЖ СТУДЕНТАМИ

1.1. Використання літератури в навчальному процесі

Підручники для закладів вищої освіти є одним з основних видів навчальних матеріалів, носіями контенту, які дозволяють студентам досягати цілей, викладених у стандарті освітньої програми. Вони є важливими ресурсами та інструментами для викладання та навчання.

Студенти використовують підручники переважно для

– пошуку підтримки для вирішення завдань і проблем;
– закріплення нового матеріалу;
– отримання додаткових знань;
– додаткове джерело в підготовці наукових робіт.

Так, дослідження, проведене в 2021 році [1], показало, що більшість британських та шанхайських учнів використовували свої підручники для попереднього читання, повторення, навчання в класі та збору інформації.

У той же час поступовий перехід до цифрового контенту призводить до зменшення кількості часу, який учні витрачали на читання підручників. За даними роботи [2], час, проведений студентами за підручниками був значно менший, ніж кількість годин, необхідних для курсу. Наприклад, студенти витрачали лише 2,6 години на тиждень на вивчення та читання навчальних матеріалів, таких як підручники з хімії та навчальні посібники для студентів, хоча освітня програма з хімії зазначає, що студенти повинні витрачати 4,1 години на тиждень на читання підручників.

Згідно дослідження [3] підручники є одним із факторів, що впливають на навчання студентів. Використання підручників допомагає учням та студентам навчатися та закріплювати свої знання та навички. Використання підручників студентами можна представити у вигляді діаграми (рис.1.1), запропонованої

Цзенгом [4], яка демонструє вплив підручників на результати навчання студентів.

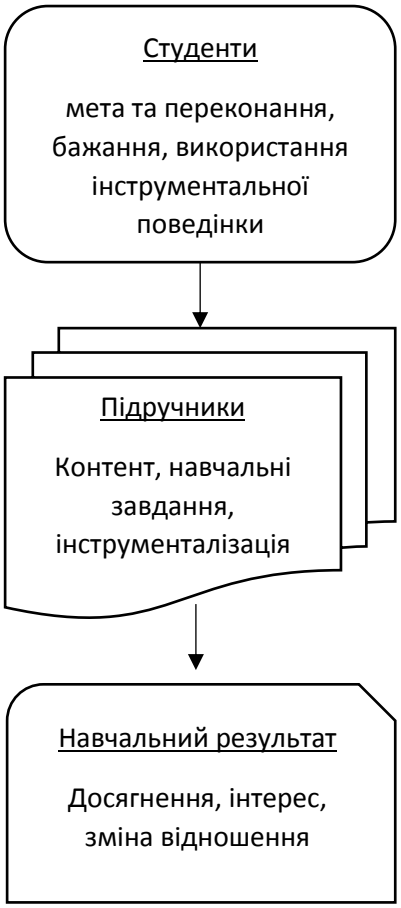

Рисунок 1.1. Модель використання підручника студентами

Підсумовуючи, можна стверджувати, що використання підручників студентами може впливати на інтерес, ставлення та їхні академічні досягнення в певній дисципліні.

У той же час розвиток електронних книг, планшетів та комп'ютерне навчання зараз розповсюджені серед всіх закладів освіти. Одна проведені дослідження показали, що, наприклад, студенти коледжів США все ще залишаються відданими текстовим підручникам. Опитування Student Monitor, проведене у жовтні 2022 року серед близько 1200 студентів у 100 американських коледжах, показало, що майже для кожного виду шкільних завдань студенти надають перевагу книзі, а не комп'ютеру [5].

Якщо об’єднати всі цифрові уподобання (включаючи настільні комп’ютери, смартфони та інші цифрові пристрої, не включені до діаграми), вони переважають друковані. Але в усьому, окрім планування завдань чи досліджень, студенти все одно воліють використовувати паперову версію, з великим відривом, ніж будь-який інший варіант (рис.1.2).

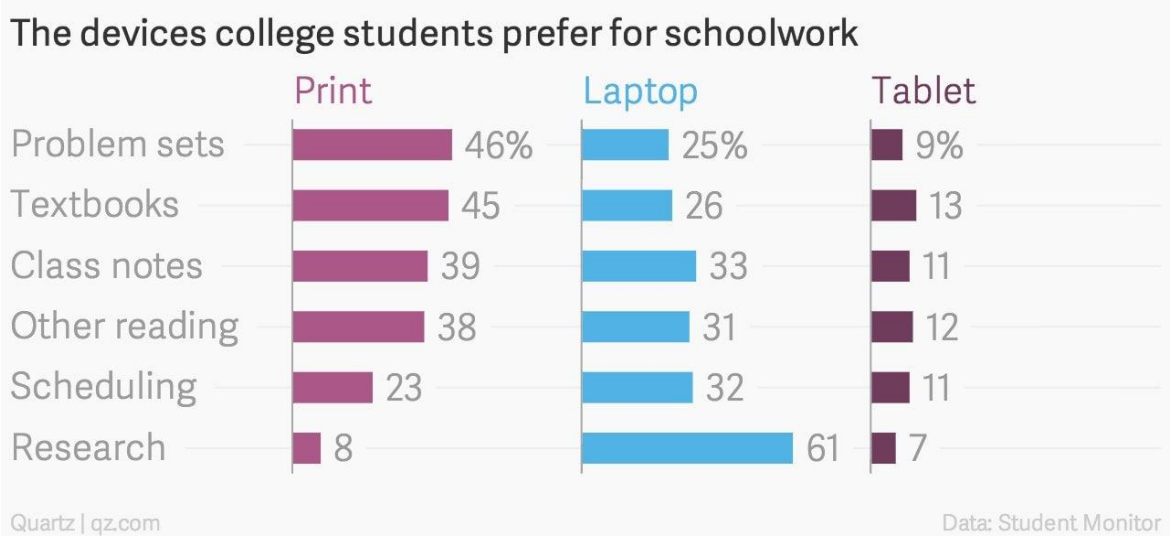

Рисунок 1.2. Вподобання студентів коледжів США щодо вибору джерела навчання [5]

Отже, можна стверджувати, що, не зважаючи на широке розповсюдження різних цифрових інструментів, книга в цілому та підручник зокрема залишаються важливим інструментом для навчання.

1.2. Огляд сучасних цифрових платформ для обміну та продажу підручників

Основною проблемою при використанні текстових підручників є їх ціна та відносна рідкість, оскільки викладач може зазначати використання у своєму курсі підручників, виданих достатньо давно. Також після проходження курсу студенти, які не планують продовжувати глибше навчання даної дисципліни

можуть мати підручники, які їм вже непотрібні. Все це породило в студентській спільноті давню практику дарування та продажу вживаних підручників.

Протягом усієї історії обміну та продажу студентських книг традиційні методи були поширеними, пропонуючи стійкий, але дещо неефективний спосіб отримання та утилізації навчальних матеріалів. Попит на доступні підручники завжди був нагальною проблемою, а купівля підручників часто виявлявся фінансовим тягарем для студентів.

Історично обмін та продаж студентських книг значною мірою спиралися на неформальні мережі, фізичні дошки оголошень та усну комунікацію. Студенти розміщували фізичні оголошення по всьому університету або покладалися на особисті зв'язки, щоб знайти можливості для придбання або продажу книг. Ці методи, хоча й мали допомогти прилаштувати підручники, були за своєю суттю обмеженими за обсягом та ефективністю. Вони залежать від випадкових зустрічей та особистих мереж, що робить їх непередбачуваними та складними для доступу для всіх студентів.

З появою інтернету та розповсюдженням інформаційно-комунікаційних технологій продаж вживаних книг перемістився в Інтернет.

В Україні існує декілька сайтів, які можуть запропонувати таку послугу.

Найбільш відомий сайт – це OLX. Він є найбільшим в Україні сервісом оголошень, де користувачі купують, продають або обмінюють товари та послуги. Платформа охоплює різноманітні категорії: від електроніки та нерухомості до роботи та особистих речей (рис.1.3). Окрім сайту, популярний однойменний мобільний додаток.

Портал забезпечує безпечний сервіс, який дозволяє покупцеві оглянути товар у відділенні (Нова пошта, Укрпошта, Meest) перед оплатою, що зменшує ризик шахрайства. OLX функціонує як майданчик для приватних осіб та бізнесу, об'єднуючи людей для швидких угод.

На сайті є окремий розділ для продажу книг та журналів, де користувач може представити свої книги. Однак на сайті існують обмеження щодо кількості

оголошень для приватних осіб та немає окремого розділу для продажів підручників. Для їх знаходження потрібно скористатися пошуком.

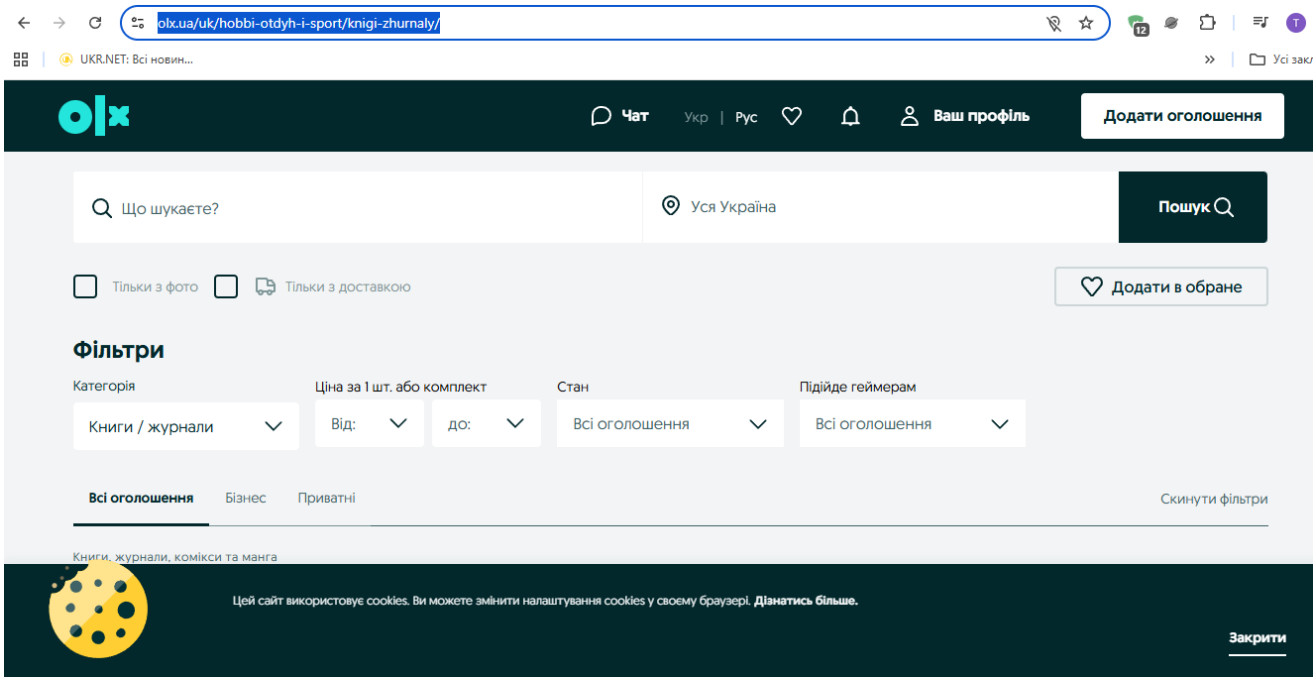

Рисунок 1.3. Розділ "Продаж книг" сайту OLX

Спеціалізованим сайтом з продажу книг є сайт Bookstock (рис.1.4). На сайті представлено безліч книг різних жанрів, серед них є окрема категорія для продажів підручників, причому можна виділити підручники для студентів. Сайт забезпечує прямий контакт з продавцем, прозорий спосіб оплати, можливість гнучко налаштовувати способи оплати та доставки.

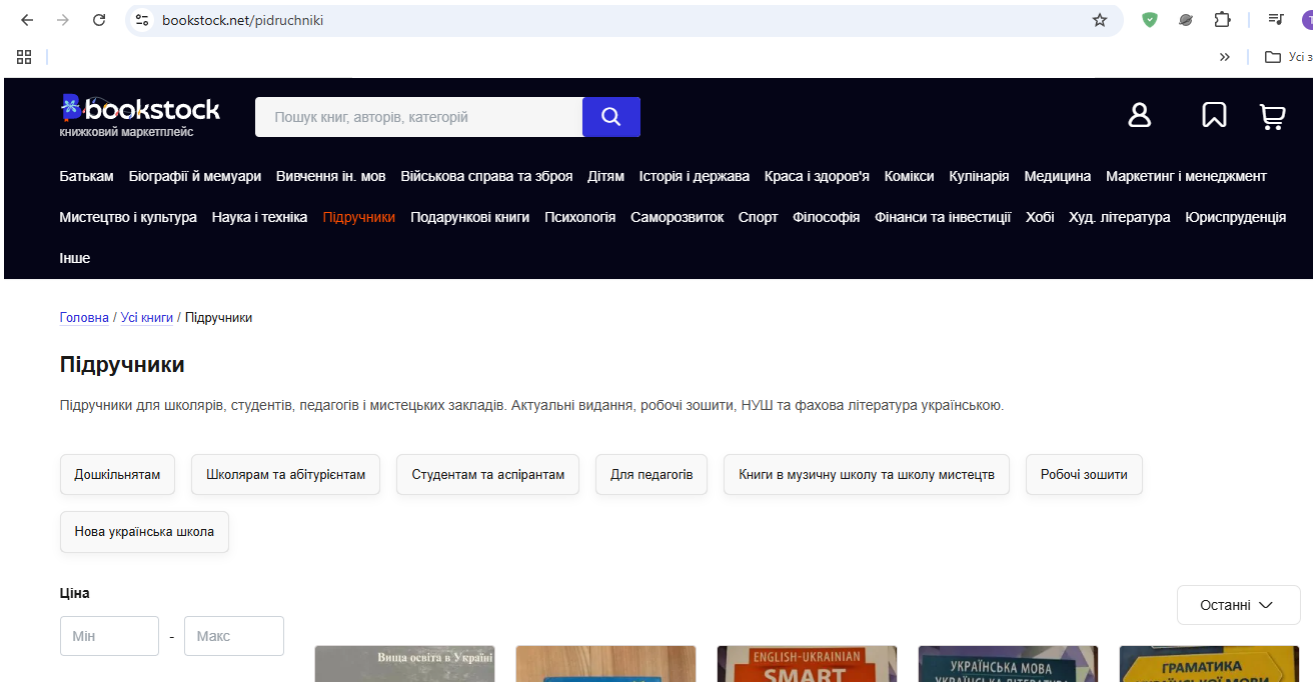

Рисунок 1.4. Категорія підручників сайту Bookstock

Ще одним сайтом з продажу вживаних книг є сайт BookFlea (рис.1.5). На сайті представлена велика кількість книг різних жанрів, є пошук, який дозволяє знайти необхідну книгу. Однак немає окремого підрозділу для продажів підручників.

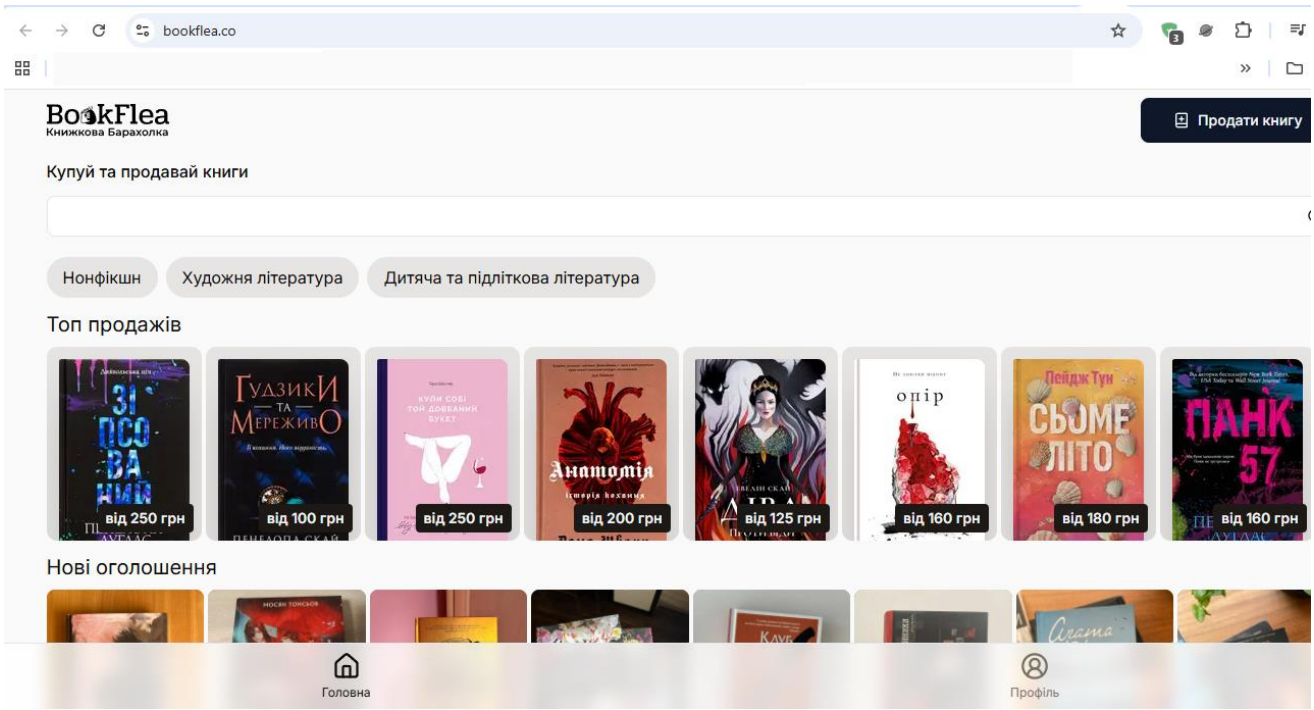

Рисунок 1.5. Головна сторінка сайту BookFlea

Всі сайти мають авторизацію для покупців та продавців, для кожного товару є опис, встановлена ціна, є можливість порівняти ціни в окремих продавців.

Аналіз представлених на розглянутих сайтах підручників показав їх загальну обмеженість щодо цього варіанту товару.

Так, пошук книг за ключовим словом "Програмування" показав, що на сайті BookFlea таких оголошень всього 2, і вони не належать до студентських підручників (рис.1.6).

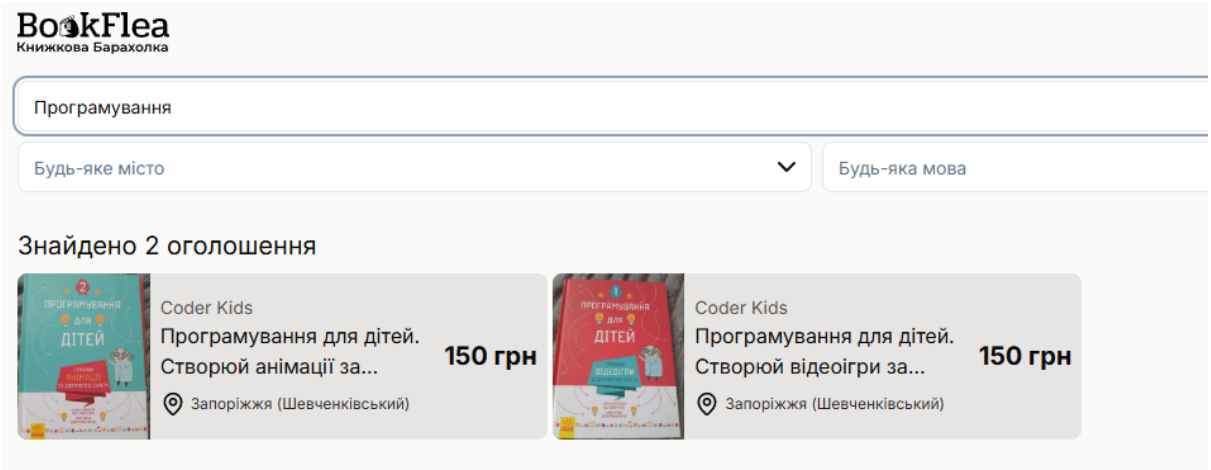

Рисунок 1.6. Пошук книг за словом "Програмування " на BookFlea

На сайті Bookstock за таким запитом була знайдена лише одна книга (рис.1.7).

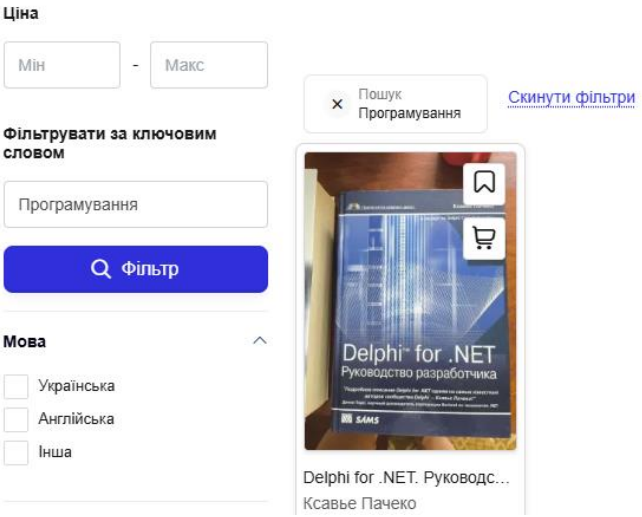

Рисунок 1.7. Результат пошуку книг за запитом "Програмування" на сайті Bookstock

Найкращий результат був отриманий на сайті OLX – 455 оголошень (рис.1.8). Але подальший аналіз показав, що серед цих книг є оголошення з надання послуг з програмування, книги для дітей тощо.

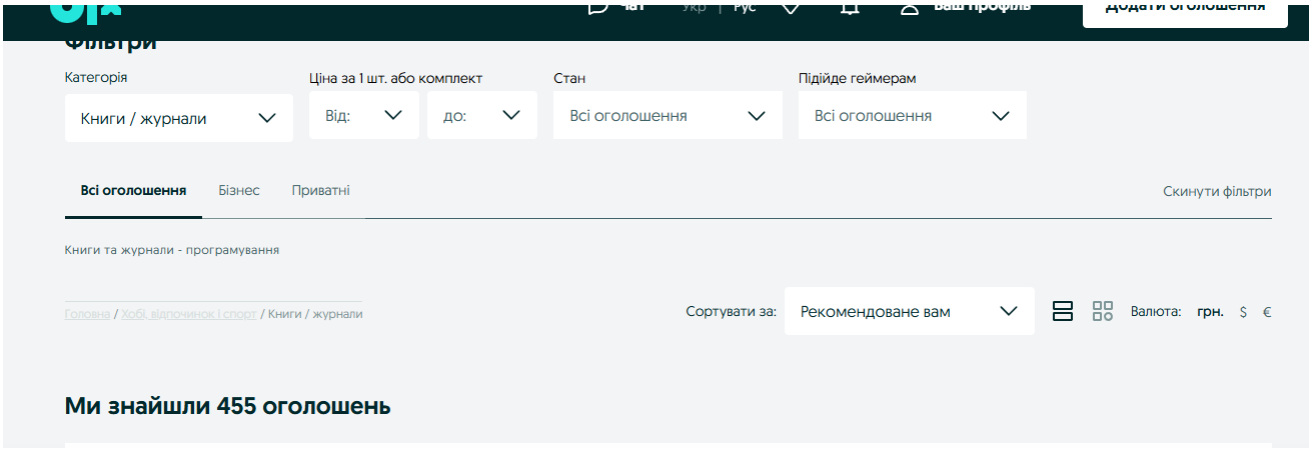

Рисунок 1.8. Результат пошуку за запитом "Програмування" в категорії Книги на OLX

Таким чином, проведений аналіз показав, що Конкурентне середовище фрагментоване, бракує єдиної платформи, яка б пропонувала комплексні рішення саме для студентів.

Більшість студентів хоче отримати книги, які вже використовувались викладачами як навчальний інструмент при викладанні своїх курсів, а не просто підручник. Використання сторонніх сайтів також може передбачати необхідність доставки книги в інше місто, що може бути пов’язано з додатковими витратами.

Отже, загалом, існуючі в українському сегменті інтернету рішення щодо продажу вживаних книг не спеціалізуються на продажі саме підручників. Ці платформи часто страждають від обмеженого використання користувачами та фрагментованого обміну, що ускладнює для студентів зв'язок з відповідними колегами для задоволення їхніх потреб, пов'язаних з книгами. Відсутність можливого пошуку за курсами, що викладаються в університеті, можливості викладання конспектами, ще більше ускладнює процес ефективного пошуку конкретних навчальних матеріалів. Студенти також у деяких випадках можуть стикнутися з високими комісіями або накладними витратами, що підриває доступність, яку ці платформи прагнуть забезпечити.

1.3. Постановка завдання

У відповідь на ці проблеми необхідно розробити платформу для обміну та продажу вживаних навчальних матеріалів, розроблену виключно для студентів. Ця платформа використовуватиме можливості Інтернету та сучасних технологій для оптимізації обміну, продажу та пожертвування вживаних книг серед студентів. Вона повинна забезпечити, щоб студенти могли ефективніше знаходити необхідні їм навчальні матеріали.

Отже, основне завдання – це розробка спеціалізованого маркетплейсу для студентів, який функціонує як дошка оголошень для купівлі, продажу та обміну навчальною літературою (підручниками) та конспектами лекцій.

Основна мета проєкту – створення локальної університетської спільноти для вторинного використання освітніх ресурсів та економії коштів студентів.

Головним результатом проєкту буде створення робочого прототипу маркетплейсу, який забезпечує студентів можливістю продажу та обміну навчальними матеріалами в межах одного університету.

Платформа повинна бути розрахована на зростаючу базу користувачів та буде сприяти розвитку активної спільноти студентів, які беруть участь в обміні освітніми ресурсами.

РОЗДІЛ 2
ПРОЄКТУВАННЯ МАРКЕТПЛЕЙСУ З ПРОДАЖУ ВЖИВАНИХ КНИГ

2.1. Аналіз вимог до маркетплейсу книг

Для розробки системи з продажу та обміну книг (навчальних матеріалів) необхідно проаналізувати вимоги, які визначають її функціональність та архітектуру. Сформульовані вимоги визначають основні технічні рішення та забезпечують ефективну і стабільну роботу системи.

Функціональні вимоги визначають основні функціональні можливості, якими повинна володіти система, щоб відповідати призначеній меті. Вони описують поведінку програмної системи за різних умов.

У таблиці 2.1 наведено перелік функціональних вимог, які визначають маркетплейс книг.

Таблиця 2.1

Функціональні вимоги до маркеплейсу книг

№	Категорія	Опис вимоги
1	Керування контентом	Розміщення оголошень із зазначенням назви, опису, ціни, стану книги та прикріпленням фото.
2	Класифікація	Фільтрація оголошень за типом (підручник/конспект), курсом, автором.
3	Авторизація та безпека	Реалізація входу через Google OAuth 2.0 для ідентифікації користувачів як представників студентської спільноти
4	Захист персональних даних	Доступ до контактної інформації продавця (Email, WhatsApp) надається лише авторизованим користувачам

Продовження табл.. 2.1

5	Комунікація	Інтеграція з месенджерами та електронною поштою для швидкого зв’язку між покупцем і продавцем.
	Аналітика	Відстеження кількості переглядів оголошень

Нефункціональні вимоги доповнюють функціональні вимоги. Вони визначають якості, характеристики та обмеження системи. Нефункціональні вимоги визначають особливості продуктивності, безпеки та зручності використання системи.

Проєкт необхідно реалізовувати з урахуванням сучасних стандартів щодо продуктивності, зручності використання та безпеки.

Сучасний підхід до безпеки передбачає використання спеціальних засобів авторизації без збереження паролів на боці сервера для безпечного керування обліковими записами.

Для забезпечення безпеки угод необхідно передбачити механізми перевірки прав доступу, обмежуючи анонімних користувачів у перегляді конфіденційної інформації.

Оскільки система розробляється для студентів українського університету, то необхідна повна адаптація інтерфейсу під українського користувача з використанням української мови та інших регіональних налаштувань (часовий пояс, валюта).

Оскільки книги можуть бути також представлені своїми фотографіями, то для оптимізації роботи потрібно передбачити автоматичну обробку зображень для зменшення навантаження на сервер.

Система повинна мати зручний користувацький інтерфейс, який відповідає уподобанням студентам щодо сучасного дизайну. Крім того система повинна забезпечувати автоматичну оптимізацію розміру завантажених зображень для швидкої роботи сайту та адаптивний дизайн для мобільних пристроїв.

2.2. Розробка архітектури та проєкту маркетплейсу

Для розробки проєкту було вирішено використовувати архітектурний шаблон MVC, який розділяє інтерфейс користувача (UI), сховище даних та бізнес-логіку на три окремі компоненти. Це дозволяє розробникам працювати над кожним компонентом незалежно, не впливаючи на дві інші частини програми.

Фреймворк MVC складається з трьох основних компонентів:

1. Модель представляє логіку даних та дані, що зберігаються у програмі. Модель зазвичай зберігає інформацію про те, що сталося в одній або кількох подіях, таких як введення користувачем або запити до бази даних.

Модель також надає спосіб доступу до даних з інтерфейсу користувача. Це включає зберігання введених користувачем даних, отримання даних з бази даних та виконання інших завдань, необхідних для функціональності вашої програми.

2. Представлення – це інтерфейс користувача програми, і це те, що бачить користувач. Воно приймає дані, надані моделлю, та відображає їх на екрані таким чином, щоб користувачі могли взаємодіяти з ними. Представлення зазвичай містить розмітку HTML, стилі CSS та код JavaScript, який забезпечує інтерактивні функції, такі як перевірка форм або взаємодія перетягуванням. Важливо зазначити, що представлення не містить жодної логіки. Воно лише відображає дані та обробляє введені користувачем дані.

3. Контролер – це компонент, який розташований між моделлю та представленням. Він відповідає за обробку взаємодії користувача з представленням, а також за виконання будь-якої логіки, необхідної для підготовки даних до відображення. У великих програмах контролер фактично стає посередником між наміром продукту, поведінкою UX та реалізацією. Наприклад, якщо користувач натискає кнопку в інтерфейсі користувача програми, це запускає подію в JavaScript, яка оброблятиметься контролером, який, у свою чергу, може використовувати цю інформацію для отримання даних з бази даних або виконання інших завдань.

Архітектура системи представлена на рис.2.1.

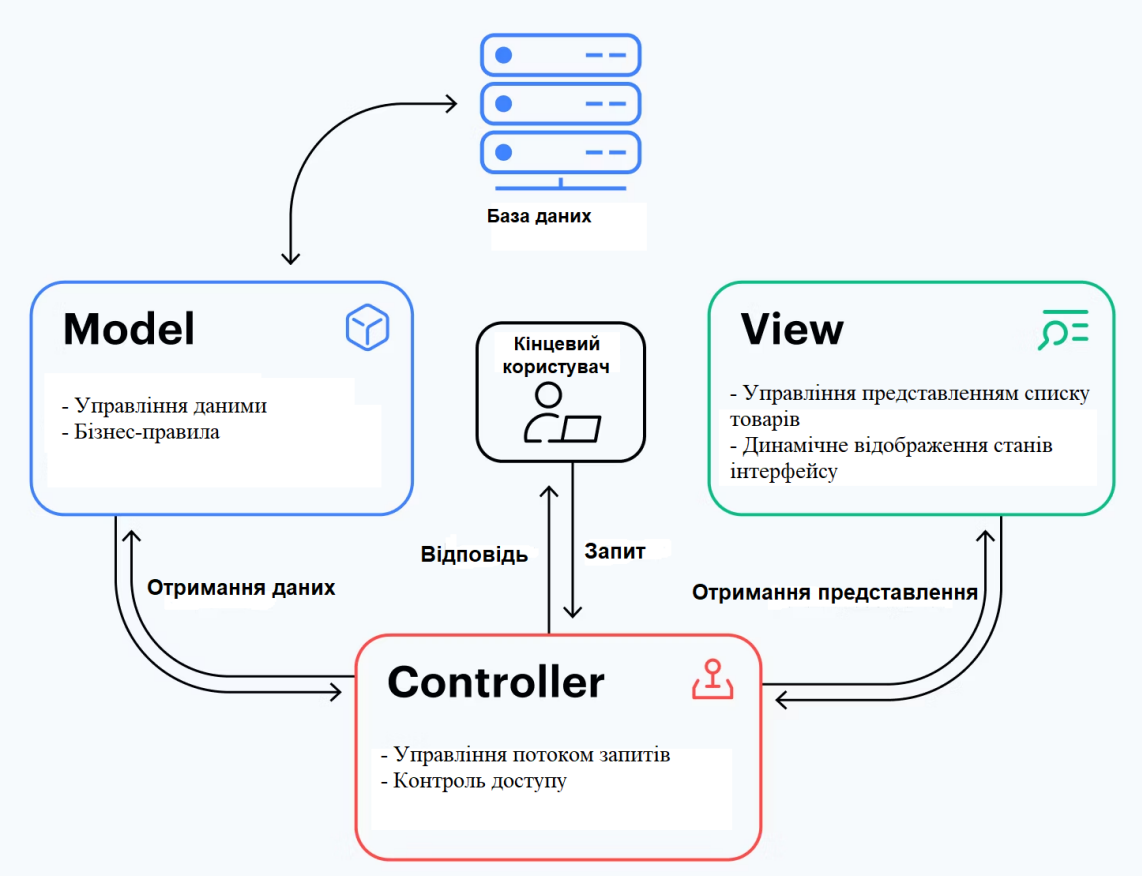

Рисунок 2.1. Архітектура маркетплейсу

1. Model (Шар даних та бізнес-логіки) відповідає за те, що система знає.

Вона управляє даними, а саме, зберігає структуру об'єктів (книг, конспектів), включаючи назви, ціни, описи та фото. У бізнес-правилах моделі реалізована внутрішня логіка, яка не залежить від того, як виглядає сайт. Наприклад, автоматична обробка зображень та індексація переглядів. Модель знає, коли книга була виставлена на продаж.

2. View (Шар представлення / Інтерфейс) – це те, як користувач бачить дані.

У розроблювальні системі цей шар відповідає за відображення списків товарів та сторінок з детальною інформацією. Він забезпечує взаємодію з користувачем, надаючи йому елементи керування (кнопки зв’язку, форми

пошуку). Також цей шар відповідає за динамічне генерування представлення, відображаючи різні стани інтерфейсу залежно від вхідних даних.

3. Controller (Шар управління логікою) – це посередник, який визначає, як система реагує на дії користувача.

Він займається обробкою запитів, приймаючи вхідні сигнали (наприклад, "користувач хоче подивитися книгу №5"). Шар займається координацією: звертається до Моделі, щоб отримати дані про книгу, і вирішує, яке Представлення (View) використовувати для показу результату.

Також саме контролер здійснює контроль доступу, тобто перевіряє права користувача перед тим, як дозволити перегляд конфіденційних даних.

Вибір даного шаблону забезпечить гнучкість системи. Так, наприклад, можна повністю змінити дизайн (View), зробивши мобільний додаток замість вебсайту, не змінюючи логіку збереження книг (Model).

Оскільки "Контролер" стоїть між даними та користувачем, він слугує фільтром, який не пропускає конфіденційну інформацію до неавторизованих користувачів.

Така архітектура забезпечує масштабованість. Можна додавати нові функції, змінюючи модель та розширюючи представлення за допомогою нового екрану, не порушуючи існуючу систему.

Наступним етапом при розробці інформаційних систем є проєктування системи, що визначає поведінку системи та її взаємодію з користувачем.

При реалізації маркетплейсу книг було вирішено використовувати об’єктно-орієнтований підхід [9]. Дана методологія поєднує в собі процес об’єктної декомпозиції та представлення статичної та логічної моделей системи, що проєктується. Основним завданням об’єктно-орієнтованого проєктування є розробка діаграм, які демонструють поведінку системи як взаємодію об’єктів.

При проєктуванні було виділено три сутності: Адміністратор, Користувач (Студент), Система.

Діаграма варіантів використання демонструє, як взаємодіють з системою Клієнт та Адміністратор (рис. 2.2). Вона показує, які дії можуть виконувати ці

користувачі:

Клієнт може переглядати список доступних книг, шукати їх за автором, назвою, курсом, надсилати повідомлення авторам оголошень, додавати оголошення щодо продажу книг.

Адміністратор має розширені права по роботі з системою, видаляти оголошення, додавати способи авторизації через соціальні акаунти.

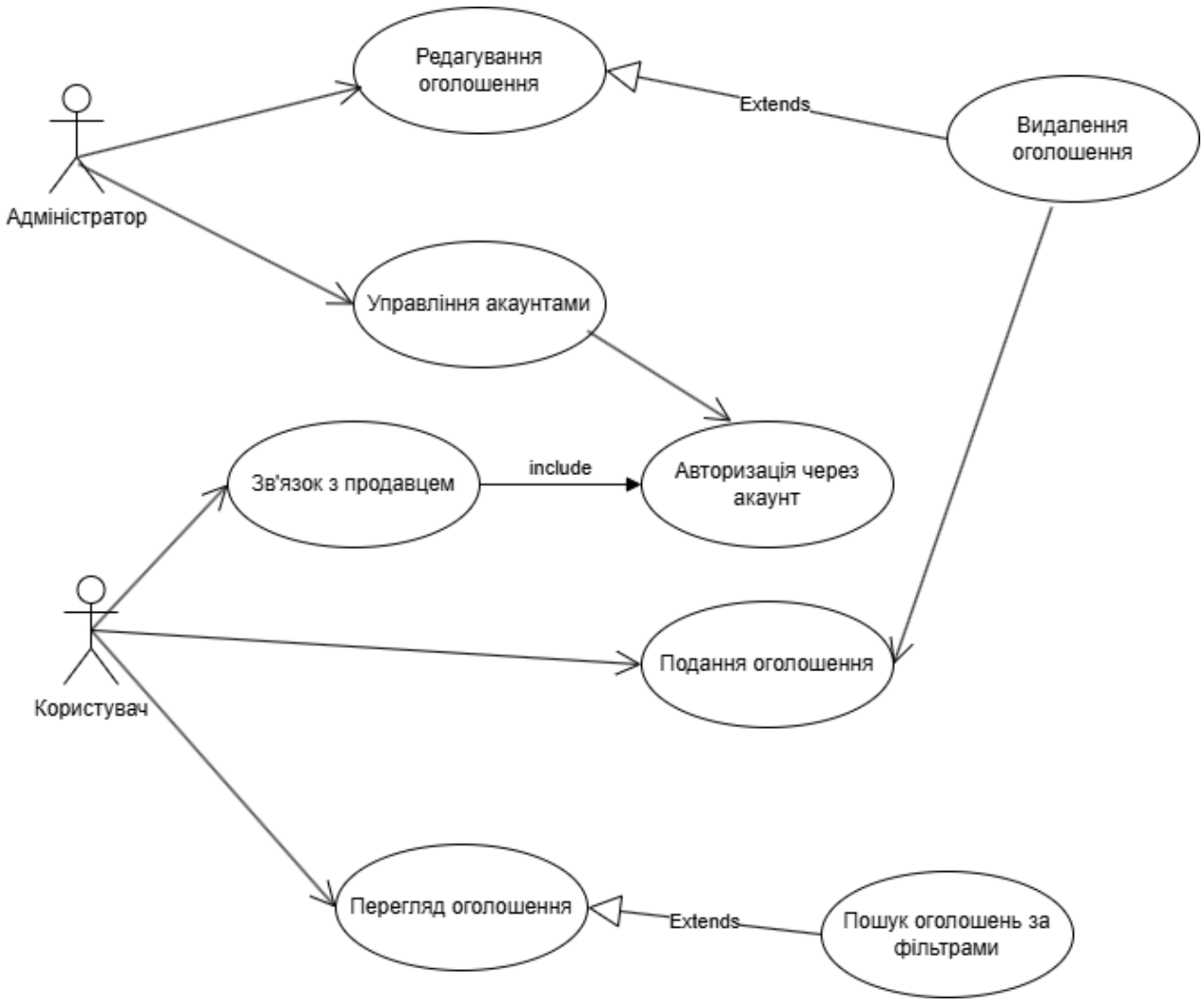

Рисунок 2.2 Діаграма варіантів використання

Таблиця 2.2.

Опис варіантів використання для актора "Адміністратор"

Варіант використання	**Короткий опис**	**Основна мета**	**Зв'язки**
Управління акаунтами	Адміністратор визначає вид соціального акаунта, який використовує користувач.	Забезпечити безпеку даних та надання доступу до конфеденційної інформації авторизованим користувачам	
Управління оголошеннями	Редагує та видаляє оголошення.	Для забезпечення функціонування маркетплейсу та вирішення спірних питань	

У таблиці 2.3 наведено опис варіантів використання для користувача маркетплейсу.

Таблиця 2.3.

Опис варіантів використання для актора "Користувач"

Варіант використання	**Короткий опис**	**Основна мета**	**Зв'язки**
Подання оголошення	Користувач описує навчальні матеріали, які він хоче продати	Організувати продажу навчальних матеріалів	
Перегляд оголошень	Користувач може переглядати оголошення.	Надати користувачу можливість ознайомитись з навчальними матеріалами	

Продовження табл. 2.3

Пошук оголошення	Користувач може шукати навчальні матеріали за автором, назвою, курсом.	Надати користувачу можливість формувати запити до системи згідно своїх вимог	Розширення варіанту Перегляд оголошень
Зв'язок з продавцем	Користувач зв'язується з продавцем через наданий контакт.	Для організації продажу книги	Включення варіанту "Авторизація"
Авторизація	Авторизація користувача через соціальний акаунт.	Забезпечення безпеки конфіденційної інформації	

Для відображення динаміки системи використовується діаграма послідовності, яка демонструє взаємодію користувача зі системою. Продемонструємо це при описі варіанти використання перегляд оголошення (рис. 2.3).

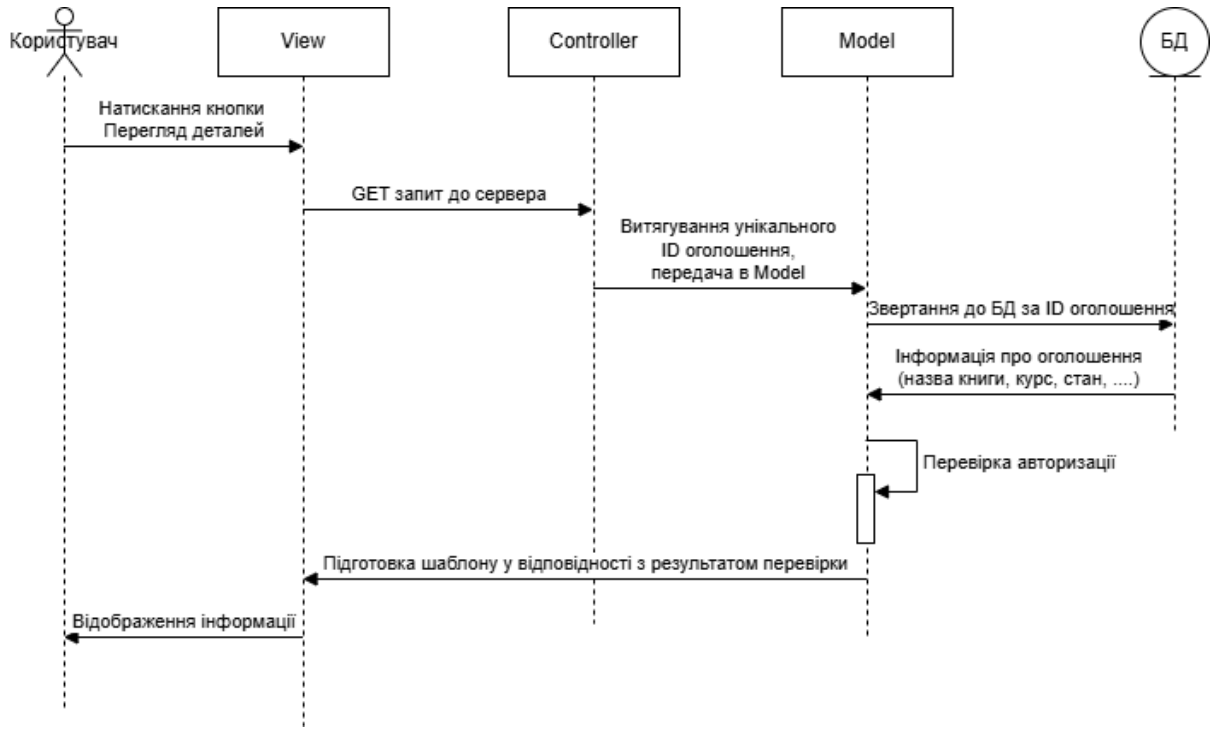

Рисунок 2.3 Діаграма послідовності для варіанту використання "Перегляд оголошення"

Опис етапів взаємодії:

1. Користувач породжує процес запиту деталей, натискаючи на сторінці з книгами кнопку "View Details".

2. Браузер надсилає GET-запит до сервера.

2. Обробка запиту здійснюється контролером, який отримує запит, витягує унікальний ID оголошення та звертається до Моделі.

3. Модель виконує запит до бази даних, щоб отримати всі поля: назву, ціну, опис, фото, ціну та контакти. Також у цей момент викликається метод increment_views(), який оновлює лічильник переглядів у базі.

4. Сервер перевіряє чи авторизований користувач.

Якщо користувач авторизований, то View готує шаблон з відкритою інформацією про контакти.

Якщо користувач є гостем, то View готує шаблон, де замість контактів відображається кнопка авторизації.

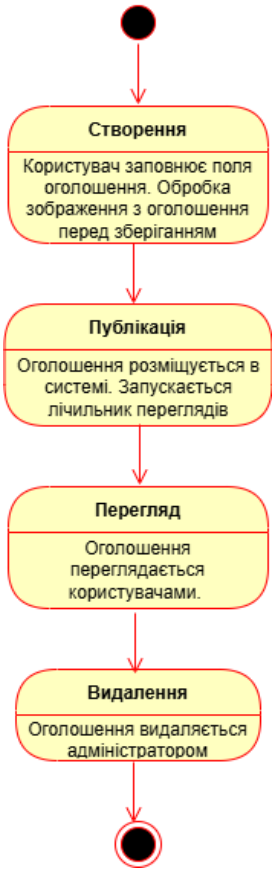

Рисунок 2.4.Діаграма станів для об’єкта Оголошення

5. Відображення (View) направляє користувачу сторінку з детальною інформацією.

Оскільки основним об’єктом, на який робиться акцент про розробці системи, є оголошення, то для демонстрації його життєвого циклу була розроблена діаграма станів (рис.2.4).

Ключовими станами оголошення є:

1. Створення. Користувач заповнює форму. У цей момент спрацьовує бізнес-логіка в моделі (щодо стиснення зображення).

2. Публікація. Оголошення з'являється в загальному списку та на головній сторінці. Система починає відстежувати кількість переглядів

3. Перегляд. Спеціальний стан, коли дані відображаються користувачу. Якщо користувач авторизований, він бачить повну інформацію, якщо ні – обмежену.

4. Видалення. Оголошення видаляється адміністратором за заявою користувача, що є кінцевою точкою життєвого циклу об'єкта.

Діаграма класів – це діаграма, що відображає статичну структуру системи. Для даної системи вона наведена на рис.2.5. і показує основні класи системи, які визначають її функціональність.

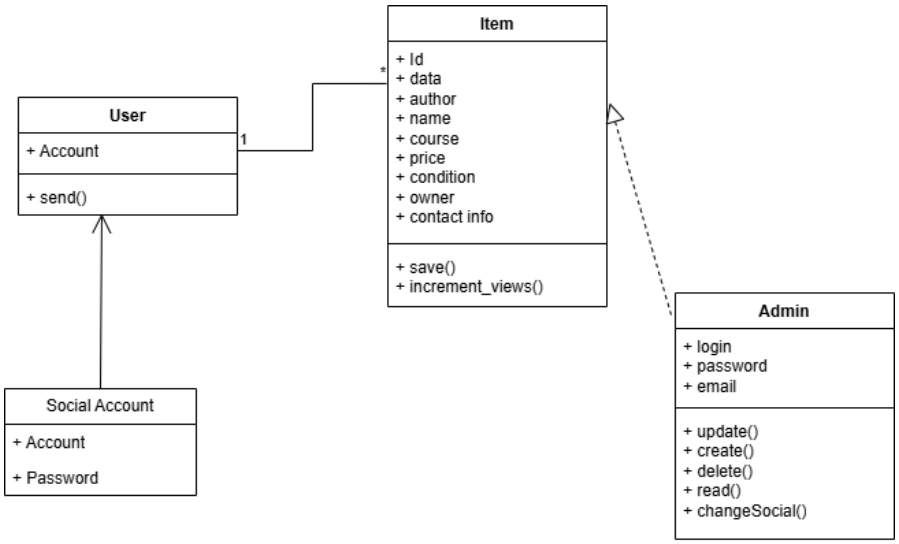

Рисунок 2.5. Діаграма класів

1. Клас Item відображає сутність книгу (навчальний матеріал), яку виставили на продаж. Клас містить основні дані, що характеризують об’єкт для продажу. Особливостями класу є наявність методів save(), який вирішує питання оптимізації збереження інформації та increment_views(), який відраховує Кількість переглядів.

Поля owner та contact_info заповнюються вручну автором при публікації.

2. Клас User визначає користувача, який зайшов у систему переглянути або опублікувати оголошення. Кожний гість може здійснити перегляд оголошень.

3. Клас SocialAccount є ключом доступу до конфіденційної інформації. Це зовнішній об'єкт, який не володіє книгою в базі, але дає право її "бачити". Він виконує роль агрегатора прав. Якщо об'єкт user існує в сесії, то інтерфейс активує приховані поля.

3. Клас Admin (Суперкористувач) має повний доступ до всіх Item через вбудовану адміністративну панель.

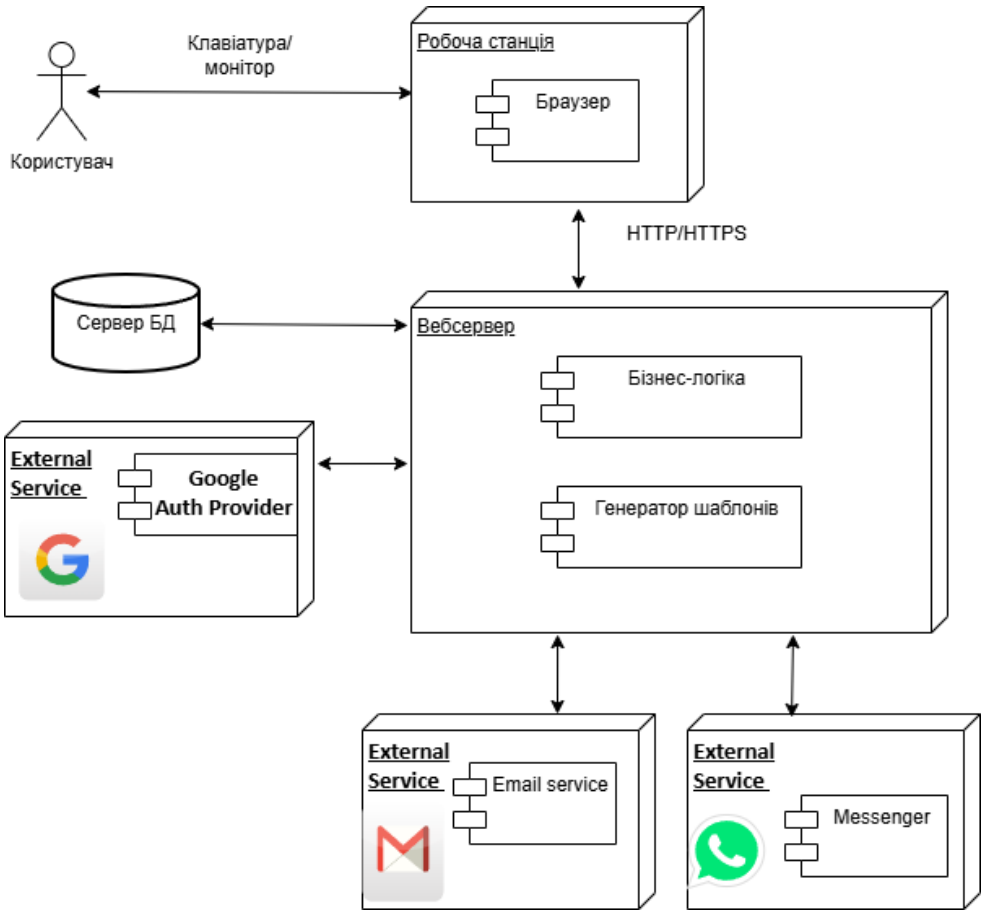

Рисунок 2.6. Діаграма розгортання

Діаграма розгортання відображає фізичну архітектуру системи. Вона демонструє фізичне розташування та взаємодію компонентів.

Для даного маркетплейсу діаграма наведена на рис.2.6

Основними вузлами на діаграмі є:

1. Клієнтський вузол – робоча станція користувача, на якій запущений веббраузер, у якому працює користувач. Він взаємодіє з сервером через протоколи HTTP/HTTPS.

2. Вебсервер, на якому запускається бекенд програмної системи. На вебсервері знаходяться модулі, які містять бізнес-логіку додатку та здійснюють рендеринг HTML-файлів.

3. Сервер БД – місце зберігання даних про книги та локальні записи про соціальні акаунти.

4. Зовнішній сервіс (Google Auth Provider) – це сторонній вузол, до якого звертається вебсервер для перевірки токена користувача, що забезпечує захист даних.

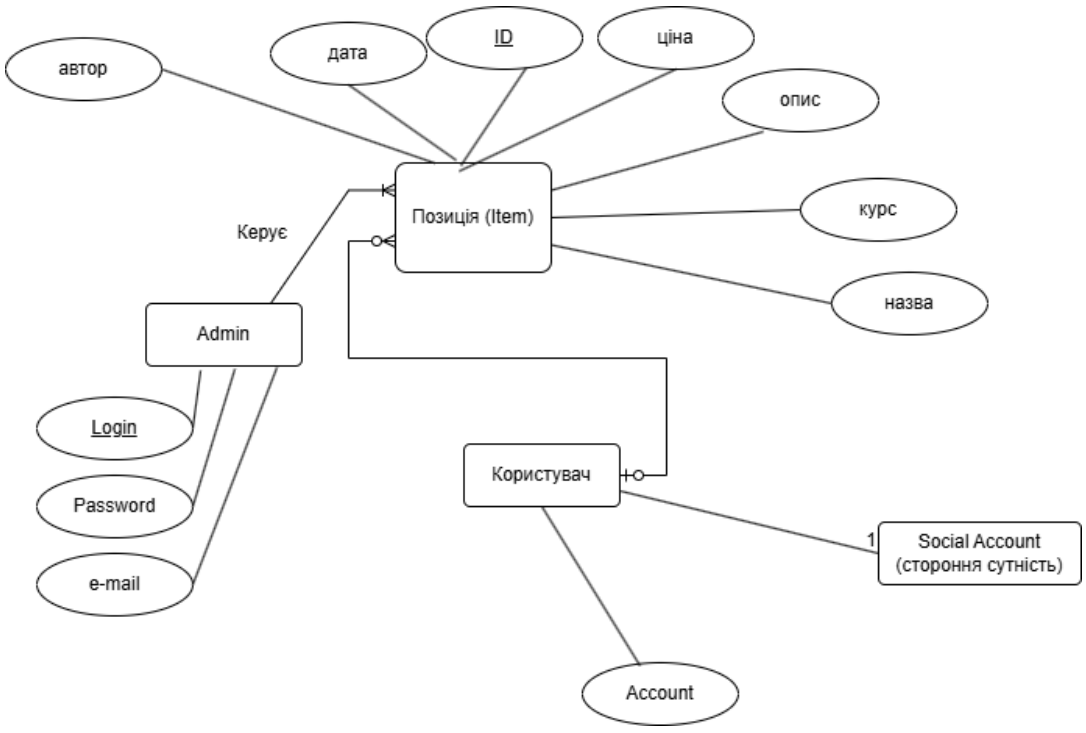

Рисунок 2.7. ERD-діаграма

Також допоміжними компонентами є зовнішні сервіси, до яких звертається вебсервер при звертанні до електронної пошти або до дзвінка через WhatsUp.

Таким чином, діаграма компонентів підтримує запропоновану архітектуру системи.

Перейдемо до проєктування БД системи. Концептуальна схема БД у вигляді ERD-діаграми представлена на рис.2.7.

Основною сутністю системи є сутність Позиція (Item), яка описує основні властивості навчального матеріалу, який виставляється на продаж. До них відноситься ідентифікаційний номер, назва, автор, курс, стан, ціна, власник та його контакти.

Другою важливою сутністю є сутність Admin, яка відповідає за контроль над всіма оголошеннями в системі.

Сутність Користувач може переглядати та виставляти на продаж навчальні матеріали. Вона також зв’язана зі сторонньою сутністю Social Account, яка забезпечує авторизацію користувача як зареєстрованого.

Така структура забезпечує ефективну роботу маркетплейсу саме як дошки оголошення, що доступна кожному студенту.

2.3. Проєктування користувацького інтерфейсу

Інтерфейс користувача – це сукупність інформаційної моделі проблемної галузі, засобів і способів взаємодії користувача з інформаційною моделлю, а також компонентів, що забезпечують формування інформаційної моделі в процесі роботи програмної системи [11, 12].

Проєктування інтерфейсів ґрунтується на кількох ключових принципах, які допомагають створити інтуїтивно зрозумілий та ефективний інтерфейс.

1. Орієнтація користувача. Користувач завжди має бути в центрі проєктування інтерфейсу. Це означає, що всі елементи системи повинні враховувати його потреби, завдання та очікування.

2. Простота та мінімалізм. Чим простіше інтерфейс, тим швидше користувач зможе освоїти його та ефективно використовувати. Мінімалізм допомагає уникнути перевантаження інформацією та зайвими елементами.

3. Послідовність. Послідовність оформлення та поведінки елементів інтерфейсу знижує когнітивне навантаження на користувача. Усі кнопки, шрифти, кольори та іконки повинні відповідати єдиному стилю.

4. Зворотній зв'язок. Інтерфейс повинен чітко повідомляти користувача про виконання його дій або помилки. Це дозволяє підвищити передбачуваність системи та знизити рівень стресу у користувачів.

5. Ефективність взаємодії. Система повинна допомагати користувачеві виконувати завдання максимально швидко та зручно. Це можна досягти за рахунок автоматизації рутинних процесів, використання автозаповнення або скорочення числа кліків.

Для реалізації прототипів інтерфейсів часто використовується схематичне зображення (Wireframe) – прямокутники, що позначають розташування елементів.

При проєктуванні маркетплейсу було вирішено дотримуватися принципів мінімалізму та швидкої доступності до інформації.

У системі було передбачено 4 основні сторінки.

1. Головна сторінка (Dashboard) призначена для швидкого ознайомлення зі станом маркетплейсу (рис.2.8).

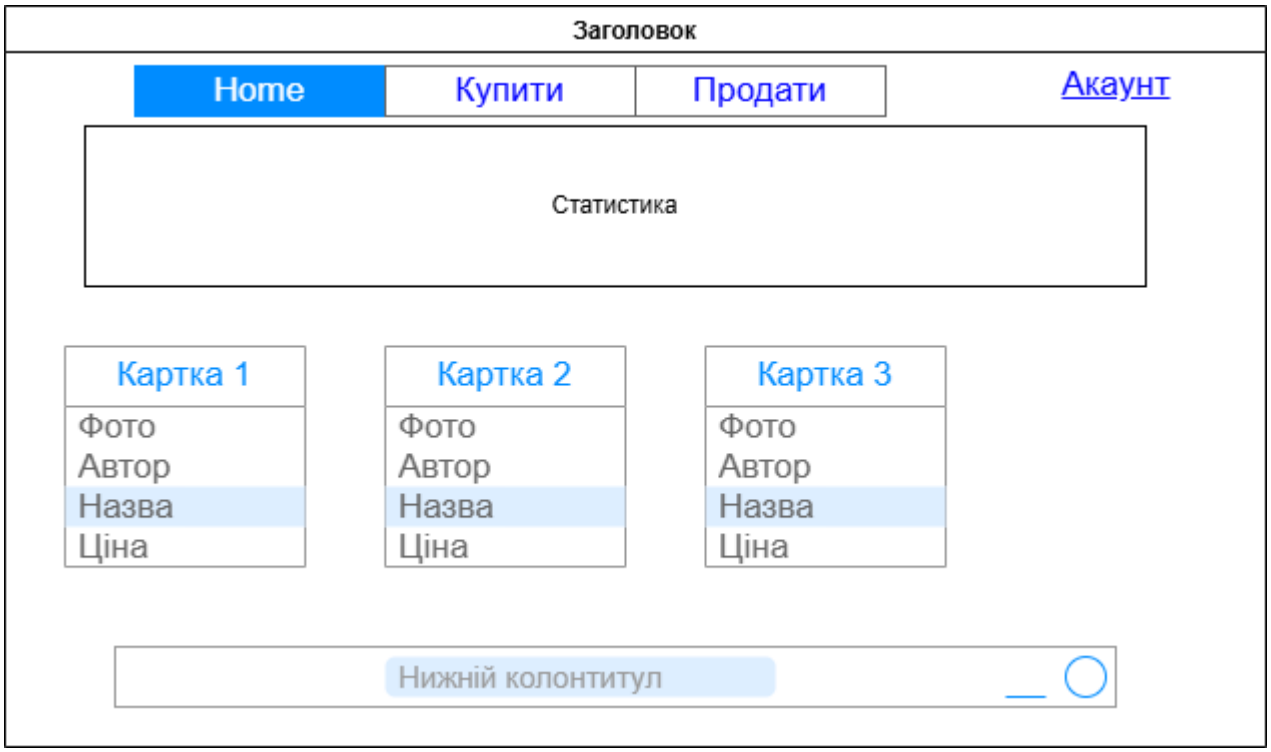

Рисунок 2.8. Вайрфрейм головної сторінки

Пропонується організувати сторінку як сукупність декількох секцій. Hero-секція містить повідомлення про систему з кнопками "Купити" та "Продати". Потім йдуть інформаційні картки (Stats), які динамічно відображають кількість доступних книг та конспектів для стимулювання активності користувачів.

У секції "Останні надходження" знаходяться 3-4 найновіших оголошення без переходу в каталог.

2. Сторінка каталогу призначена для навігації по всіх доступних пропозиціях (рис.2.9). На даній сторінці передбачена пошукова панель за назвою, автором або курсом.

Кожне оголошення представлене карткою з фотографією, назвою та ціною.

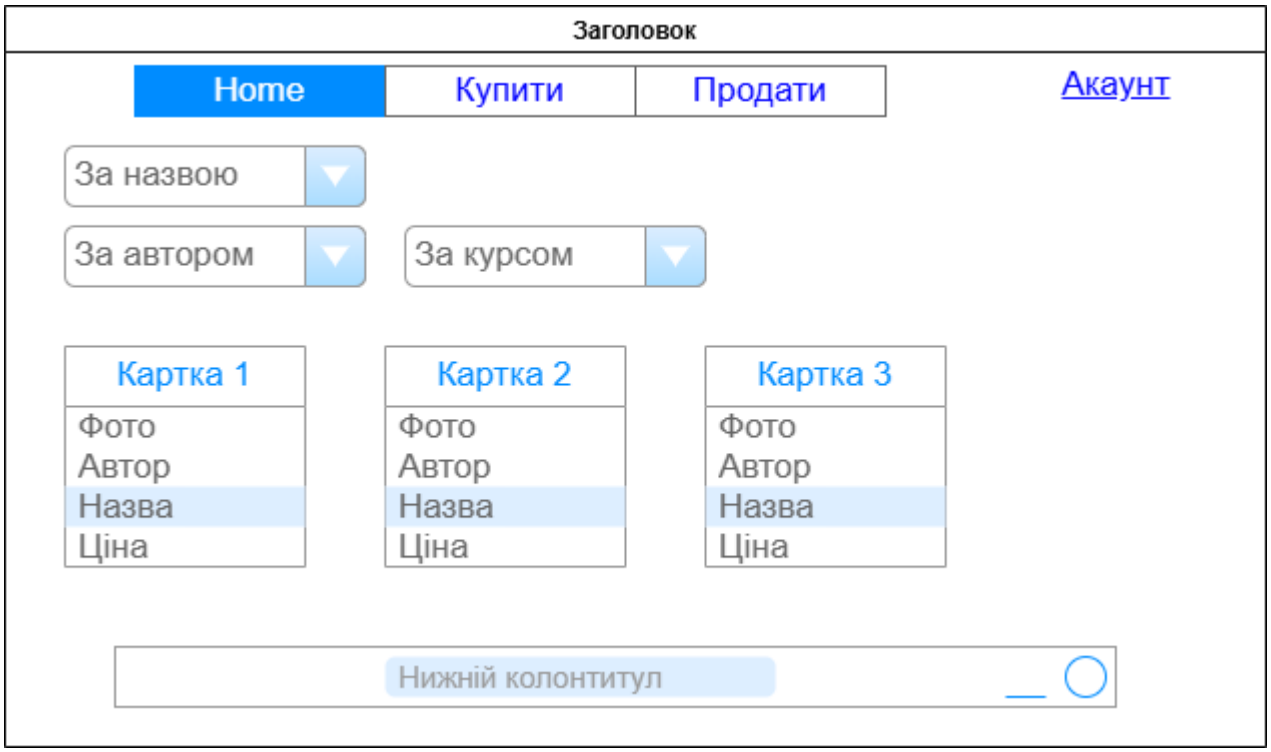

Рисунок 2.9. Вайрфрейм сторінки каталогу

3. Сторінка публікації оголошення призначена для збору даних від користувача для створення нового запису в БД (рис.2.10).

Вона складається з переліку полів (назва, опис, ціна, контактні дані), які необхідно заповнити користувачу, щоб опублікувати оголошення.

Поле для вибору фотографії підручника повинно бути інтуїтивно зрозумілим. Також повинна бути валідація даних перед відправкою на сервер.

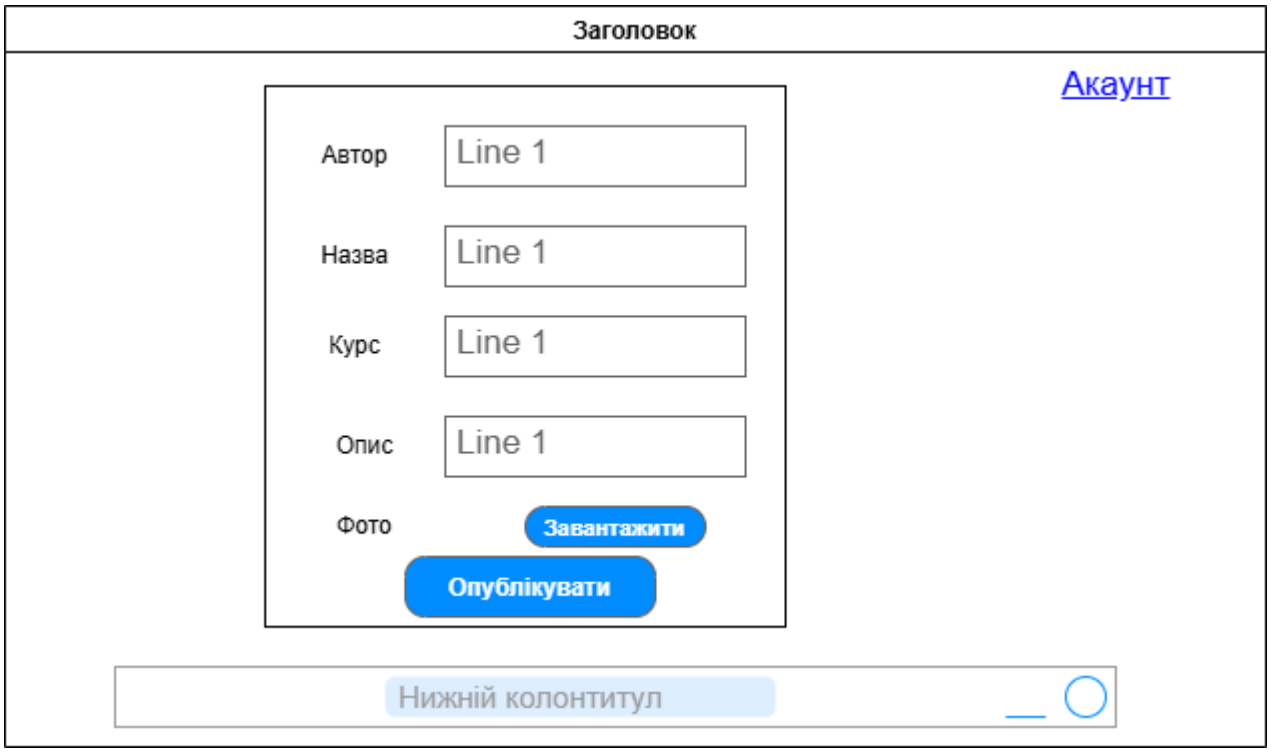

Рисунок 2.10. Вайрфрейм публікації оголошення

4. Сторінка перегляду оголошення призначенв для надання повної інформації про товар та забезпечення зв'язку (рис.2.11).

Сторінка організована у вигляді двоколонної структури: зліва знаходиться велике зображення товару, справа знаходиться ціна, опис та лічильник переглядів.

Внизу сторінки знаходиться блок комунікації. Користувачу повідомляється інформація про автора оголошення, але контакти лишаються недоступними і приховані. Динамічні кнопки зв’язку з продавцем активуються лише після авторизації користувача.

Інформація про навчальний матеріал супроводжується бейджиком, який відображає тип навчального матеріалу (книга/конспект).

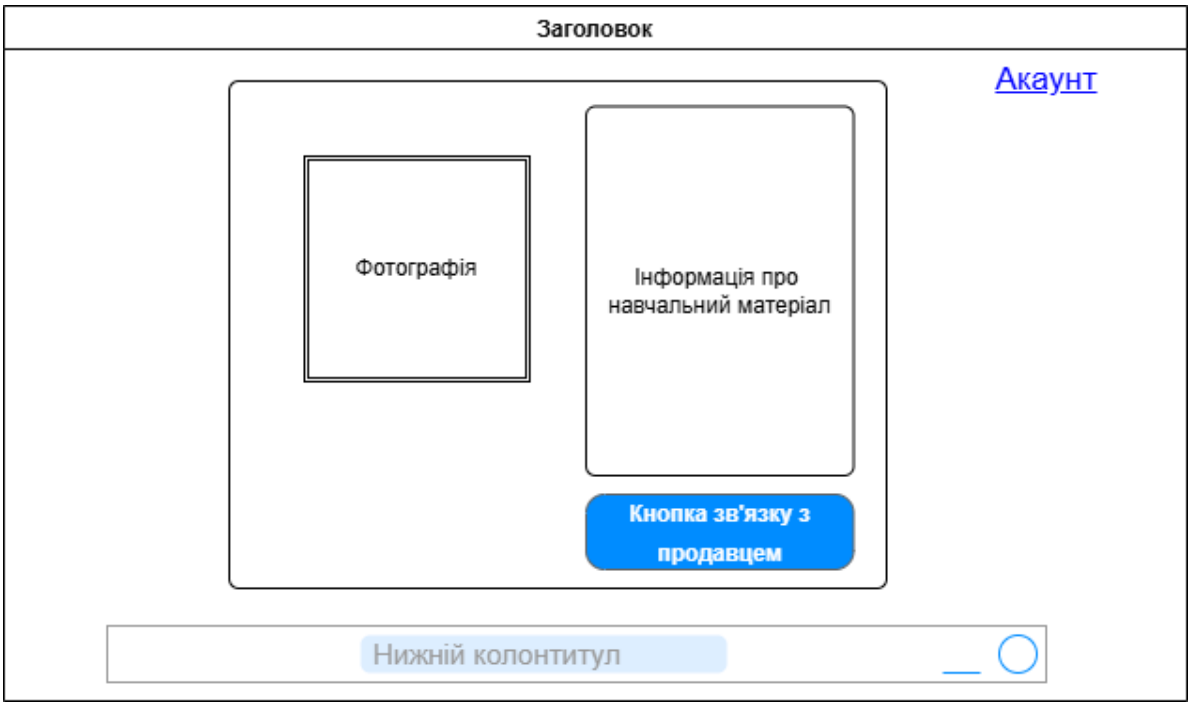

Рисунок 2.11. Сторінка перегляду конкретного навчального матеріалу

Розроблений проєкт користувацького інтерфейсу є досить простим і в той же час зручний. Для кращого залучення студентської аудиторії пропонується використовувати приємну кольорову гаму та значки. Такі значки можуть бути використані в подальшому при розвитку проєкту та переходу на мобільну версію.

РОЗДІЛ 3. РОЗРОБКА ТА РЕАЛІЗАЦІЯ МАРКЕТПЛЕЙСУ КНИГ

3.1. Вибір та обґрунтування стеку технологій розробки

Для розробки маркетплейсу з продажу вживаних студентських книг та навчальних матеріалів було обрано мову програмування Python, вебфреймворк Django та CSS-фреймворк Bootstrap. Такий вибір зумовлений функціональними можливостями технологій, швидкістю розробки, зручністю підтримки системи та відповідністю поставленим вимогам.

Python є однією з найпоширеніших мов програмування для веброзробки завдяки простому синтаксису, великій кількості бібліотек та високій швидкості створення програмного забезпечення. Використання Python дозволяє зменшити складність реалізації серверної логіки, забезпечує хорошу читабельність коду та спрощує подальшу модифікацію системи. Наразі, мова активно використовується у сфері вебтехнологій, що забезпечує значну кількість готових рішень і документації.

Одним з підходів у веброзробці з використанням Python є розробка додатків на основі фреймворку Django. Його було обрано через його архітектурні особливості та вбудовані засоби для створення додатків. Django реалізує концепцію MTV (Model–Template–View), що дозволяє логічно розділити структуру проєкту та спростити підтримку коду. Однією з головних переваг Django є наявність готових механізмів для роботи з користувачами, автентифікацією, адміністративною панеллю, маршрутизацією та базами даних. Це скоротить час розробки маркетплейсу та дозволить зосередитися на реалізації бізнес-логіки системи.

Також Django містить вбудовані засоби захисту від поширених вебзагроз, зокрема CSRF-атак, SQL-ін’єкцій та XSS-уразливостей. Для маркетплейсу, де користувачі взаємодіють із персональними даними та публікують оголошення, питання безпеки є важливим аспектом.

Для реалізації користувацького інтерфейсу було використано Bootstrap. Даний CSS-фреймворк дозволяє швидко створювати адаптивний та зручний дизайн вебсторінок. Bootstrap містить набір готових компонентів, таких як форми, кнопки, навігаційні панелі та картки товарів, що значно прискорює процес розробки інтерфейсу маркетплейсу. Крім того, адаптивність Bootstrap забезпечує коректне відображення системи на різних пристроях, зокрема, персональних комп'ютерах, планшетах та смартфонах, що є важливим для студентської аудиторії.

Для режиму відлагодження (Debug) було обрано систему керування базами даних SQLite. SQLite є легкою вбудованою базою даних, яка не потребує встановлення окремого серверного програмного забезпечення та додаткового налаштування. Це дозволяє швидко розпочати розробку та тестування проєкту, зменшує складність локального середовища та пришвидшує виконання початкових етапів створення системи. SQLite добре підходить для розробки та тестування невеликих проєктів, оскільки база даних зберігається у вигляді одного файлу.

Проте для продуктивного середовища (Production) планується використання PostgreSQL. Дана система керування базами даних є більш продуктивною, масштабованою та надійною у порівнянні з SQLite. PostgreSQL підтримує одночасну роботу великої кількості користувачів, ефективну обробку транзакцій та механізми забезпечення цілісності даних. Для маркетплейсу, де передбачається активна взаємодія користувачів із товарами, повідомленнями та оголошеннями, підтримка конкурентного доступу до бази даних є критично важливою.

Крім того, PostgreSQL забезпечує розширені можливості оптимізації запитів, резервного копіювання, масштабування та роботи зі складними структурами даних. Django має повноцінну інтеграцію з PostgreSQL, що дозволяє ефективно використовувати переваги цієї СКБД у реальному середовищі експлуатації. Таким чином, використання SQLite на етапі розробки

та PostgreSQL у продакшн-середовищі є обґрунтованим рішенням з точки зору продуктивності, зручності розробки та подальшого масштабування системи.

3.2. Реалізація архітектурних компонентів системи (MTV)

Принцип роботи вебфреймворку Django базується на архітектурному шаблоні MVT (Model–View–Template). Даний підхід забезпечує розділення логіки обробки даних, бізнес-логіки та користувацького інтерфейсу.

Модель (Model) відповідає за структуру даних та взаємодію з базою даних. Саме в моделях визначаються таблиці, поля, зв’язки між об’єктами та правила роботи з даними. Django використовує ORM (Object-Relational Mapping), що дозволяє працювати з базою даних через Python-об’єкти без необхідності написання SQL-запитів вручну.

Компонент View відповідає за обробку HTTP-запитів користувача та реалізацію бізнес-логіки системи. Представлення отримує дані з моделей, виконує необхідну обробку та передає результат у шаблони для відображення користувачу.

Шаблони (Template) відповідають за формування інтерфейсу користувача. Вони містять HTML-код із вбудованими засобами шаблонізації Django, що дозволяє динамічно відображати дані, отримані від View. Такий підхід забезпечує відокремлення логіки програми від візуального оформлення.

Принцип роботи Django можна описати наступним чином: користувач надсилає HTTP-запит через браузер, система маршрутизації URL визначає відповідне представлення (View), яке обробляє запит, взаємодіє з моделями (Model) та передає отримані дані до шаблону (Template). Після цього формується HTML-сторінка, яка повертається користувачу.

У розробленому проєкті структура каталогів організована відповідно до рекомендацій Django та забезпечує логічний розподіл компонентів системи (рис.3.1).

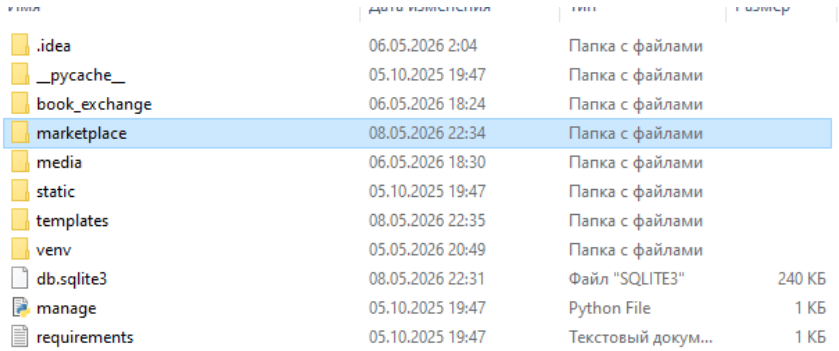

Рисунок 3.1. Структура проєкту маркетплейсу

Опишемо дану структуру.

Папка bookexchange є основною конфігураційною директорією проєкту. У ній розташовані файли:

__init__.py – службовий файл Python, який визначає директорію як пакет;

settings.py містить основні налаштування проєкту: параметри бази даних, підключення застосунків, налаштування статичних файлів, безпеки та конфігурацію системи;

urls.py відповідає за маршрутизацію URL-адрес та визначає, які представлення будуть обробляти конкретні HTTP-запити.

Папка marketplace є основним застосунком системи маркетплейсу. Вона містить логіку роботи вебзастосунку та основні програмні компоненти:

- admin.py використовується для налаштування адміністративної панелі Django та керування моделями через вебінтерфейс;
- apps.py містить конфігурацію застосунку;
- models.py містить моделі даних, які відображають сутностями, з якими буде працювати система;
- forms.py використовується для створення та обробки вебформ
- views.py реалізує логіку обробки HTTP-запитів та взаємодію між моделями і шаблонами.

Папка static призначена для збереження статичних ресурсів вебзастосунку. До таких ресурсів належать CSS-файли, JavaScript-файли, шрифти та

зображення, які використовуються для оформлення та функціонування інтерфейсу.

Папка templates містить HTML-шаблони вебсторінок. Тут реалізується користувацький інтерфейс системи із використанням механізму шаблонів Django (рис.3.2).

```
<!-- Statistics Cards -->
<div class="row mb-5">
    <div class="col-md-4 mb-3">
        <div class="stat-card text-center p-4">
            <h3 class="stat-number">{{ stats.total_items }}</h3>
            <p class="stat-label mb-0">Всього доступно позицій</p>
        </div>
    </div>
    <div class="col-md-4 mb-3">
        <div class="stat-card text-center p-4">
            <h3 class="stat-number">{{ stats.total_books }}</h3>
            <p class="stat-label mb-0">Доступно книг</p>
        </div>
    </div>
    <div class="col-md-4 mb-3">
        <div class="stat-card text-center p-4">
            <h3 class="stat-number">{{ stats.total_notes }}</h3>
            <p class="stat-label mb-0">Доступно конспектів</p>
        </div>
    </div>
</div>
```

Рисунок 3.2. Приклад частини коду файлу home.html з формуванням дашборду

Папка media використовується для збереження файлів, які завантажуються користувачами під час роботи із системою. У межах даного проєкту вона призначена для збереження фотографій книг та навчальних матеріалів, що додаються до оголошень маркетплейсу.

Повний лістинг коду проєкту знаходиться в додатку

3.3. Реалізація організації роботи з маркетплейсом: пошук, сортування, фільтрація

У файлі models.py знаходиться модель модель даних для маркетплейсу студентських книг та навчальних матеріалів. Основним елементом даного файлу є оголошення класу Item, який наслідується від класу models.Model.

Клас Item використовується для збереження інформації про книги, конспекти та інші навчальні матеріали, які користувачі можуть розміщувати на платформі.

Основні поля моделі реалізують збереження ключової інформації про товар:

item_name – назва позиції;

owner – зв’язок із користувачем системи через зовнішній ключ ForeignKey;

description – текстовий опис товару;

price – ціна товару;

seller_name та contact_info – інформація про продавця;

image – поле для завантаження фотографії товару (рис.3.3).

```
  item_name = models.CharField(max_length=200, verbose_name="Назва позиції")
  owner = models.ForeignKey(User, on_delete=models.CASCADE, null=True, blank=True, verbose_name="Власник")
  description = models.TextField(verbose_name="Опис", help_text="Опишіть стан, вкажіть відсутні сторінки.")
  price = models.DecimalField(max_digits=10, decimal_places=2, verbose_name="Ціна (грн)")
  seller_name = models.CharField(max_length=100, verbose_name="Ваше імя")
  contact_info = models.CharField(max_length=200, verbose_name="Контакти", help_text="Номер WhatsApp або email")
  image = models.ImageField(upload_to='item_images/', blank=True, null=True, verbose_name="Фото позиції")

  # Additional useful fields
  item_type = models.CharField(max_length=10, choices=ITEM_TYPES, default='book', verbose_name="Тип")
  author = models.CharField(max_length=100, blank=True, null=True, verbose_name="Автор")
  course = models.CharField(max_length=50, blank=True, null=True, verbose_name="Назва курсу", help_text="напр.,
ца математика для економістів")
  condition = models.CharField(max_length=20, choices=CONDITIONS, default='good', verbose_name="Стан")
  date_posted = models.DateTimeField(default=timezone.now, verbose_name="Дата публікації")
  is_sold = models.BooleanField(default=False, verbose_name="Продано")
  view_count = models.PositiveIntegerField(default=0, verbose_name="Перегляди")
```

Рисунок 3.3. Частина коду з описом класу Item

Поле image реалізовано за допомогою ImageField, що дозволяє користувачам додавати фотографії книг або навчальних матеріалів. Завантажені файли автоматично зберігаються в директорії media/item_images/. Для оптимізації роботи із зображеннями був перевизначений метод save(), який використовується для автоматичної обробки завантажених зображень. Після збереження об’єкта виконується відкриття файлу зображення за допомогою бібліотеки Pillow (PIL.Image). Якщо зображення має формат з прозорістю (RGBA або P), воно автоматично конвертується у формат RGB. Це необхідно

для забезпечення коректного збереження та сумісності з різними форматами файлів.

Додатково реалізовано механізм оптимізації зображень. Якщо розмір фотографії перевищує 800×800 пікселів, виконується автоматичне масштабування методом thumbnail(). Для зміни розміру використовується алгоритм LANCZOS, який забезпечує високу якість зменшеного зображення. Після цього файл повторно зберігається з параметрами optimize=True та quality=85, що дозволяє зменшити розмір файлу без суттєвої втрати якості.

Лістинг коду функції:

```
def save(self, *args, **kwargs):
        super().save(*args, **kwargs)

        # Optimize uploaded images
        if self.image:
            img_path = self.image.path
            if os.path.exists(img_path):
                img = Image.open(img_path)
                # Convert to RGB if necessary
                if img.mode in ("RGBA", "P"):
                    img = img.convert("RGB")
                # Resize if too large
                if img.height > 800 or img.width > 800:
                    img.thumbnail((800, 800), Image.Resampling.LANCZOS)
                    img.save(img_path, optimize=True, quality=85)
```

Такий підхід відповідає логіці вебзастосунку, дозволяючи йому швидше завантажувати сторінки для перегляду.

У класі також реалізовано допоміжні методи:

increment_views(), який автоматично збільшує лічильник переглядів;

get_contact_type(), який визначає тип контактної інформації (електронна пошта або телефон).

У файлі views.py реалізовано основну логіку роботи вебзастосунку маркетплейсу студентських книг та навчальних матеріалів. Даний файл містить

набір функцій-представлень (View), які обробляють HTTP-запити користувачів, взаємодіють із моделями та формують відповіді для відображення у шаблонах.

Функція home(request) реалізує головну сторінку системи. Вона виконує отримання останніх доданих товарів, які ще не продані, та формує статистичні дані для відображення на головній сторінці. Зокрема, обчислюється: загальна кількість активних оголошень; кількість книг; кількість конспектів; кількість продавців.

Також виконується аналіз популярних навчальних курсів за кількістю опублікованих матеріалів. Отримані дані передаються до шаблону home.html через словник context.

Функція post_item(request) відповідає за додавання нового оголошення до системи. Якщо користувач надсилає форму методом POST, створюється об’єкт форми ItemForm, який містить текстові дані та завантажені файли (request.FILES). Після проходження валідації дані зберігаються у базі даних. У випадку успішного додавання система відображає повідомлення про успішне створення оголошення та перенаправляє користувача на сторінку товару. Якщо форма містить помилки, виводиться повідомлення про необхідність виправлення даних.

Функція all_items(request) реалізує сторінку каталогу всіх товарів. На початку формується запит до бази даних для отримання лише непроданих товарів. Далі реалізовано механізм пошуку та фільтрації.

Пошук працює через параметр search, який дозволяє знаходити товари за:

- назвою;
- автором;
- назвою курсу;
- описом;
- ім’ям продавця.

Фільтрація здійснюється:

- за типом товару (книга або конспект);
- за навчальним курсом;

– за станом товару.

Для реалізації пошуку та фільтрації у системі використовується клас Q із Django ORM. Результати функцій Q застосовуються під час формування складних умов фільтрації записів у базі даних. Це дозволяє виконувати пошук одночасно за декількома полями моделі із використанням логічних операторів OR та AND.

Лістинг коду функції:

```
search_query = request.GET.get('search', '').strip()
    if search_query:
        items = items.filter(
            Q(item_name__icontains=search_query) |
            Q(author__icontains=search_query) |
            Q(course__icontains=search_query) |
            Q(description__icontains=search_query) |
            Q(seller_name__icontains=search_query)
        )
```

Використання оператора icontains забезпечує нечутливість до регістру та частковий збіг рядків.

Функція search_ajax(request) реалізує асинхронний пошук із використанням AJAX. Коли користувач вводить текст у поле пошуку, система надсилає запит без перезавантаження сторінки. Якщо довжина пошукового запиту перевищує два символи, виконується пошук у базі даних за назвою товару, курсом та автором.

Результати формуються у вигляді JSON-структури, яка містить:

– ідентифікатор товару;
– назву;
– курс;
– ціну;
– URL-адресу сторінки товару.

Лістинг коду функції:

```
def search_ajax(request):
    """AJAX search for autocomplete"""
    query = request.GET.get('q', '').strip()
    if len(query) >= 2:
        items = Item.objects.filter(
            Q(item_name__icontains=query) |
            Q(course__icontains=query) |
            Q(author__icontains=query),
            is_sold=False
        )[:10]

        results = []
        for item in items:
            results.append({
                'id': item.pk,
                'name': item.item_name,
                'course': item.course or '',
                'price': str(item.price),
                'url': item.get_absolute_url()
            })
        return JsonResponse({'results': results})

    return JsonResponse({'results': []})
```

Використання AJAX дозволяє реалізувати автодоповнення пошуку та покращує взаємодію користувача із системою без необхідності повного перезавантаження вебсторінки.

Для реалізації сортування товарів використовується стандартний механізм Django ORM – метод order_by(). Параметр сортування передається через змінну sort_by, після чого застосовується до вибірки даних.

У системі здійснюється сортування:

– за датою додавання;

– за ціною;

– за популярністю (кількістю переглядів);
– за назвою.

Такий варіант забезпечує релевантну видачу позицій, які є в каталозі.

Лістинг коду:

```
sort_by = request.GET.get('sort', '-date_posted')
    valid_sorts = ['-date_posted', 'price', '-price', 'item_name', '-view_count']
    if sort_by in valid_sorts:
        items = items.order_by(sort_by)
```

На рівні користувацького інтерфейсу (all_items.html) реалізовано інтерактивне меню сортування. Завдяки інтеграції JavaScript-події onchange, зміна критерію впорядкування автоматично ініціює GET-запит до сервера з відповідним параметром, що забезпечує безперервність користувацького досвіду (UX). Лістинг коду для реалізації користувацького інтерфейсу сортування (рис.3.4):

```
<div class="col-md-3">
    <label for="sort" class="form-label">Сортувати за</label>
    <select name="sort" id="sort" class="form-select" onchange="this.form.submit()">
        <option value="-date_posted" {% if current_sort == '-date_posted' %}selected{% endif %}>Нові оголошення</option>
        <option value="price" {% if current_sort == 'price' %}selected{% endif %}>Найдешевші</option>
        <option value="-price" {% if current_sort == '-price' %}selected{% endif %}>Найдорожчі</option>
        <option value="item_name" {% if current_sort == 'item_name' %}selected{% endif %}>За назвою (А-Я)</option>
        <option value="-view_count" {% if current_sort == '-view_count' %}selected{% endif %}>Популярні</option>
    </select>
</div>
```

Рисунок 3.4. Лістинг реалізації сортування в інтерфейсі

Для покращення продуктивності та зручності користування реалізовано пагінацію за допомогою класу Paginator. На одній сторінці відображається 12 товарів, а навігація між сторінками здійснюється через параметр page.

```
paginator = Paginator(items, 12)
page_number = request.GET.get('page')
items_page = paginator.get_page(page_number)
```

Функція item_detail(request, pk) реалізує сторінку окремого товару. Об’єкт отримується за первинним ключем (pk) через функцію get_object_or_404, що дозволяє автоматично повертати помилку 404 у випадку відсутності запису.

Після відкриття сторінки автоматично збільшується лічильник переглядів через метод increment_views(). Також реалізовано можливість позначення товару як проданого. Якщо користувач надсилає POST-запит із параметром mark_sold, поле is_sold встановлюється у значення True.

Таким чином, у системі реалізовані основні можливості щодо перегляду та пошуку навчальних матеріалів

3.4. Технічна реалізація підсистем авторизації

Для забезпечення зручності використання системи всіма студентами університету було вирішено реалізувати можливість перегляду наявних позицій у каталозі та їх придбання будь-якому користувачу. Однак, у будь-якому разі, у системі необхідно передбачити механізм запобігання шахрайству та скаму.

Для реалізації функціоналу реєстрації та входу було обрано використання стороннього сервісу автентифікації як паралельну систему для автеттифікації, яку запропоновує Django. Це рішення дозволило відмовитися від розробки власної системи зберігання паролів, перекладаючи цю функціональність на спеціалізовані сервіси. Такий підхід зараз є досить розповсюдженим і значно підвищує рівень безпеки проєкту.

При натисканні кнопки "Увійти через Google", система перенаправляє користувача на захищений шлюз Google. Після успішного підтвердження особи, сервер отримує унікальний токен та базові дані (email, ім'я), на основі яких автоматично створюється запис у таблиці User.

Такий підхід гарантує, що доступ до контактних даних продавців отримують лише реальні студенти, які мають підтверджений цифровий профіль.

Захист даних реалізовано за принципом «мінімальних привілеїв» безпосередньо в логіці шаблонів та контролерів.

Вся інформація в системі поділена на "загальну" (доступна всім) та "конфіденційну" (контакти власника).

У файлі детального перегляду оголошення (item_detail.html) реалізовано умовну логіку: сервер перевіряє стан об'єкта request.user.is_authenticated. Тільки у випадку істинного значення (користувач залогінений), Django рендерить блоки з кнопками зв'язку (WhatsApp/Email). В іншому випадку ці дані не просто приховуються стилями CSS, а фізично не передаються в HTML-коді, що унеможливлює їх збір автоматизованими скриптами (парсерами).

Для реалізації функції видалення оголошень за скаргами користувачів активовано стандартний інтерфейс Django Admin, який надає суперкористувачу повний набір інструментів для модерації контенту.

Також був реалізований підхід на рівні модулів аудиту системи. Був реалізований моніторинг активності для збереження інформації про доступ до контактної інформації. Оскільки контакти показуються лише авторизованим користувачам, логування дозволяє точно фіксувати, хто отримує доступ до контактних даних продавця. Було реалізовано уніфікований підхід, який дозволяє відстежувати обидва канали (електронну пошту та WhatsUp).

Лістинг функції легування:

```
def contact_email_log(request, pk):
    item = get_object_or_404(Item, pk=pk)
    if request.user.is_authenticated:
        logger.info(f"USER_EMAIL_CLICK: {request.user.email} -> Item ID:
{item.pk}")
    subject = urllib.parse.quote(f"Питання щодо: {item.item_name}")
    mailto_url = fmailto:{item.contact_info}?subject={subject}
    return HttpResponseRedirect(mailto_url)
```

Приклад логування у файлі activity.txt наведений на рис.3.5

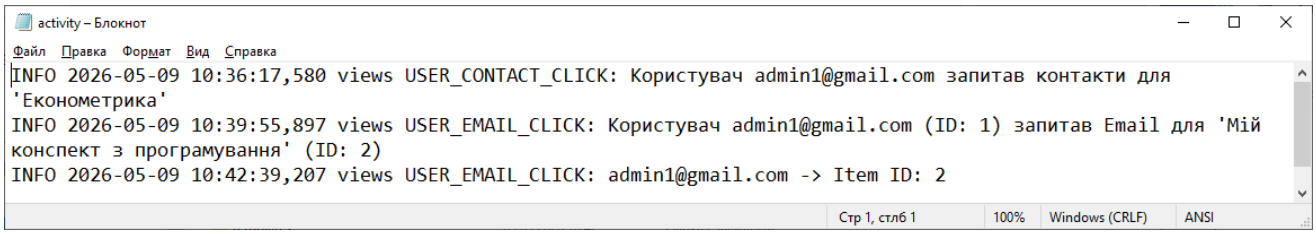

Рисунок 3.5. Приклад логування активності користувачів на сайті Маркетплейсу підручників

Запропонована система перехоплення кліків приховує прямі посилання на месенджери від автоматизованих ботів, оскільки URL формується динамічно після перевірки авторизації. Це дозволяє вирішити проблему парсингу сайтів.

Отже, при такому рішенні Google бере на себе всі питання щодо відновлення паролів, двофакторної автентифікації (2FA) та захисту від брутфорс-атак.

РОЗДІЛ 4. ТЕСТУВАННЯ СИСТЕМИ

4.1. Проведення тестування системи

Тестування розробленого додатка здійснювалося на локальній машині за адресою http://127.0.0.1:8000/. Перевірка працездатності системи проводилася в режимі Debug на персональному комп'ютері з процесором Intel Core i3 та оперативною пам'яттю обсягом 8 Гб.

Запуск системи здійснювався із середовища розробки PyCharm у браузері Google Chrome. Основні конфігураційні параметри системи залишалися незмінними, за винятком налаштувань поштової служби.

Для середовища розробки було налаштовано EmailBackend у режимі консольного виводу. Це дозволило протестувати повний життєвий цикл реєстрації та авторизації користувачів без використання реального SMTP-сервера, підтвердивши коректність формування шаблонів листів.

Тестування інтеграції з месенджером WhatsApp було проведено до етапу перенаправлення користувача та формування автоматичного повідомлення продавцю.

Після запуску системи на екрані з'являється головне вікно з дашбордом про кількість доступних книг (рис.4.1).

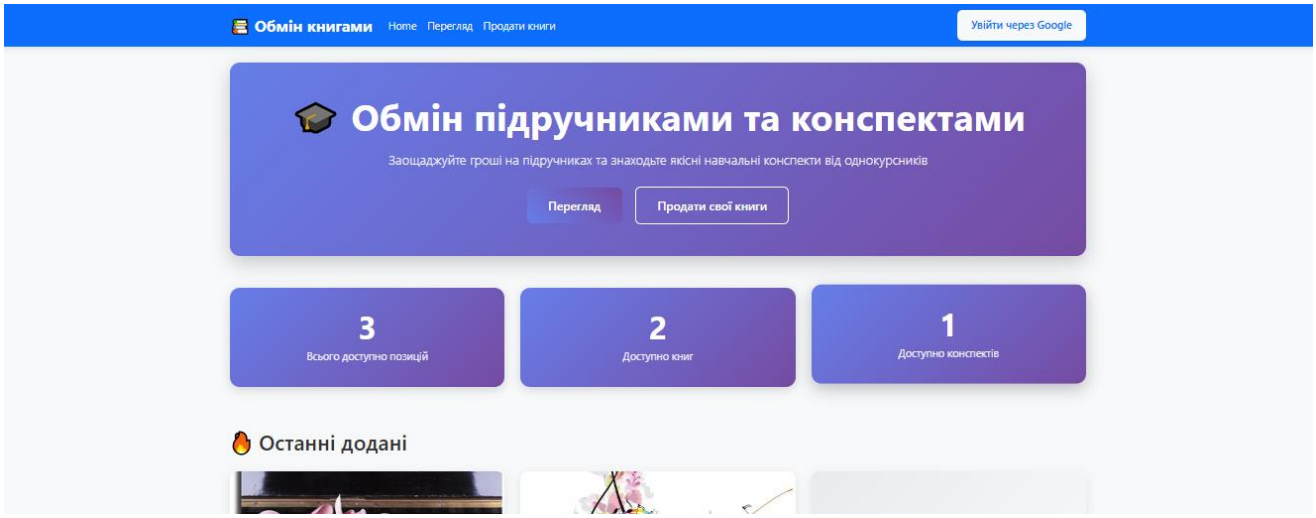

Рисунок 4.1. Домашня сторінка маркетплейсу підручників

Користувач бачить загальну кількість доступних книг, кількість конспектів та внизу сторінки знаходяться останні надходження до системи.

Далі користувач може продивитися існуючі книги, перебуваючи в системі як Гість, а може увійти за допомогою Google, або використовуючи авторизацію Django.

Для цього необхідно натиснути відповідну кнопку (рис.4.2).

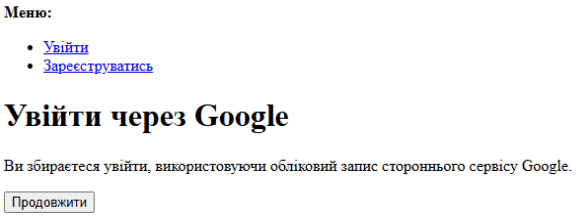

Рисунок 4.2. Вхід у систему

При натисканні кнопки Продовжити перед користувачем з’являється стандартне вікно авторизації Google, де він може ввести свої логін та пароль. Після цього користувач повертається до домашньої сторінки системи.

Внизу домашнього екрану знаходиться кнопка Переглянути всі позиції, за допомогою якої користувач може перейти до каталогу (рис.4.3).

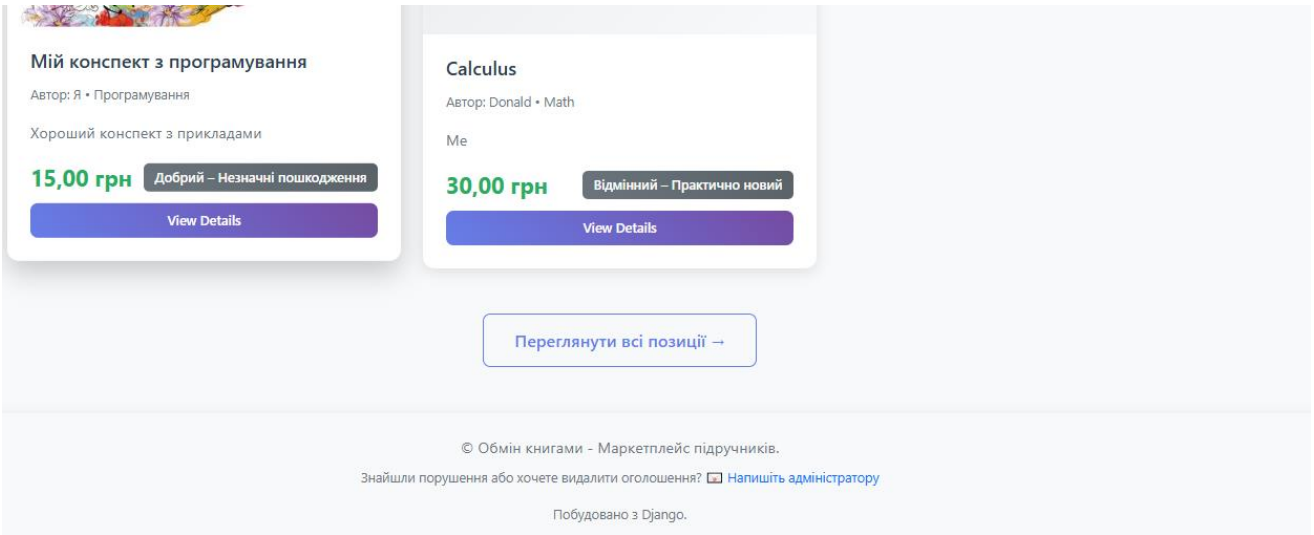

Рисунок 4.3. Нижня частина домашньої сторінки з переходом до каталогу

У каталозі користувачу доступні пошук, фільтрація та сортування наявних книг.

Приклад пошуку за першими 5 літерами слова "Економетрика" наведений на рис.4.4.

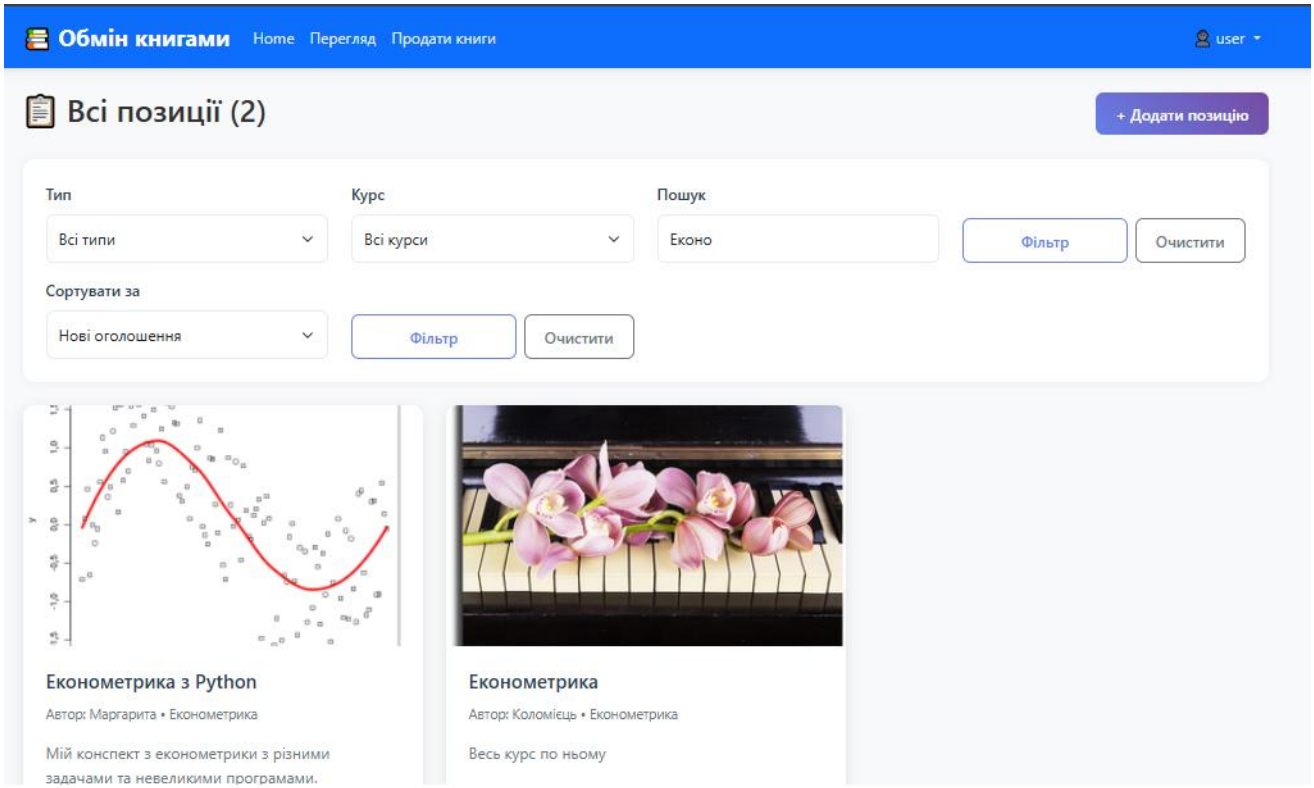

Рисунок 4.4. Приклад пошуку всіх навчальних матеріалів з буквосполученням "Екон"

Користувач може здійснити процес фільтрації книг за типом, курсом. На рис. 4.5. наведений приклад фільтрації навчальних матеріалів з курсу "Алгоритмізація". Список курсів формується динамічно в залежності від наявних у каталогу книг та конспектів.

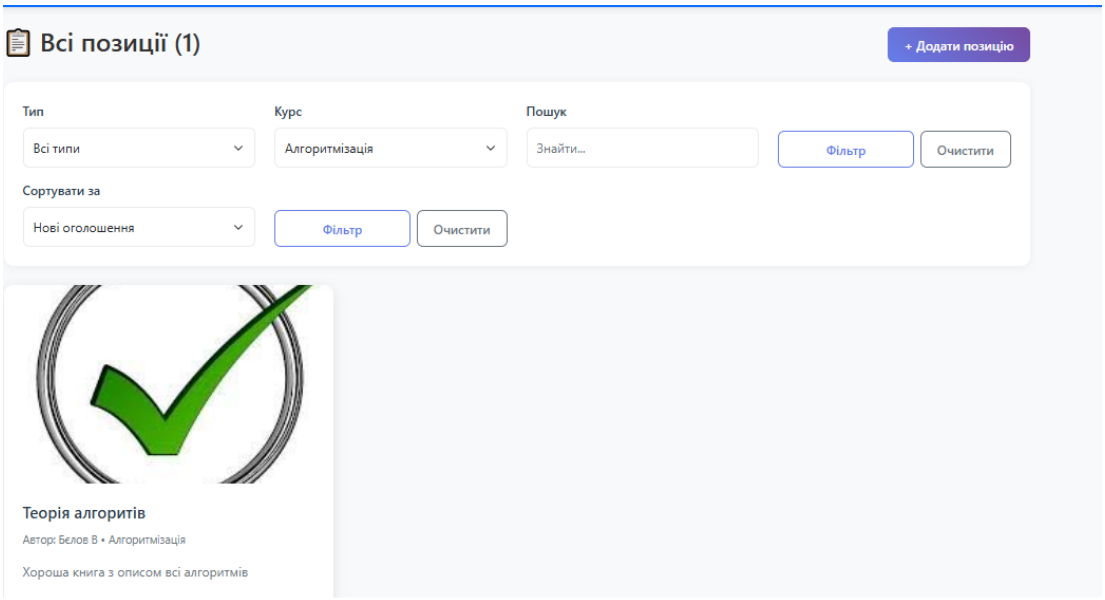

Рисунок 4.5. Приклад фільтрації каталогу

Також користувач може відсортувати список книг за різними критеріями. На рис. 4.6 наведений приклад сортування навчальних матеріалів за ціною в сторону спадання.

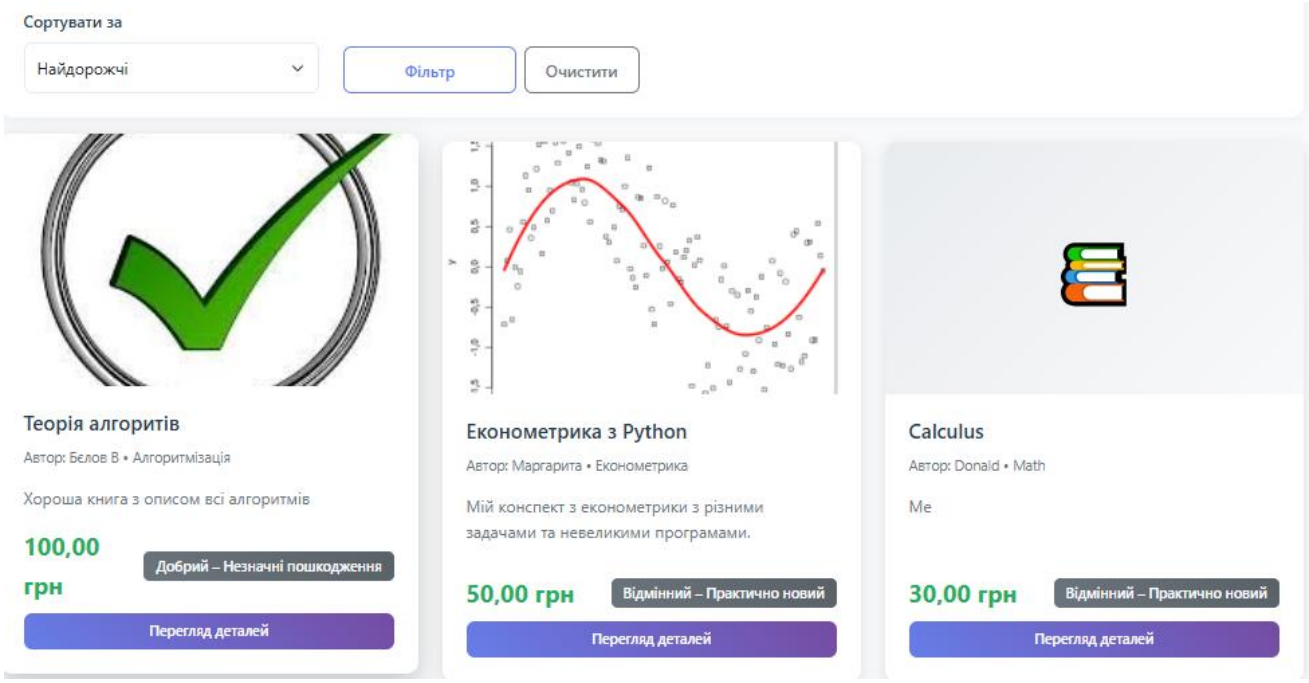

Рисунок 4.6. Приклад сортування каталогу за ціною

Користувач може додати свою книгу для продажу. Для цього в системі організований базовий шаблон base.html, який забезпечує можливість навігації верхнім рядком меню. Для додавання своєї книги в каталог користувач повинен натиснути кнопку Продати книги в верхньому рядку вікна. На екрані з’явиться форма Опублікуйте свою позицію, де червоною зірочкою відмічені обов’язкові поля (рис.4.7).

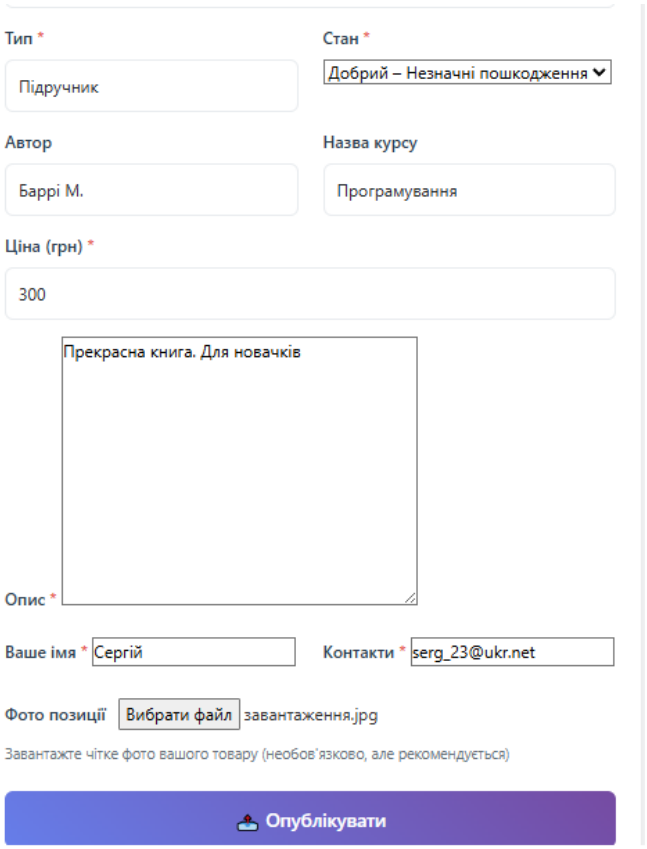

Рисунок 4.7. Форма для публікації інформації про книгу

Після натискання кнопки Опублікувати з’являється повідомлення про успішне (або неуспішне) занесення книги в каталог (рис.4.8)

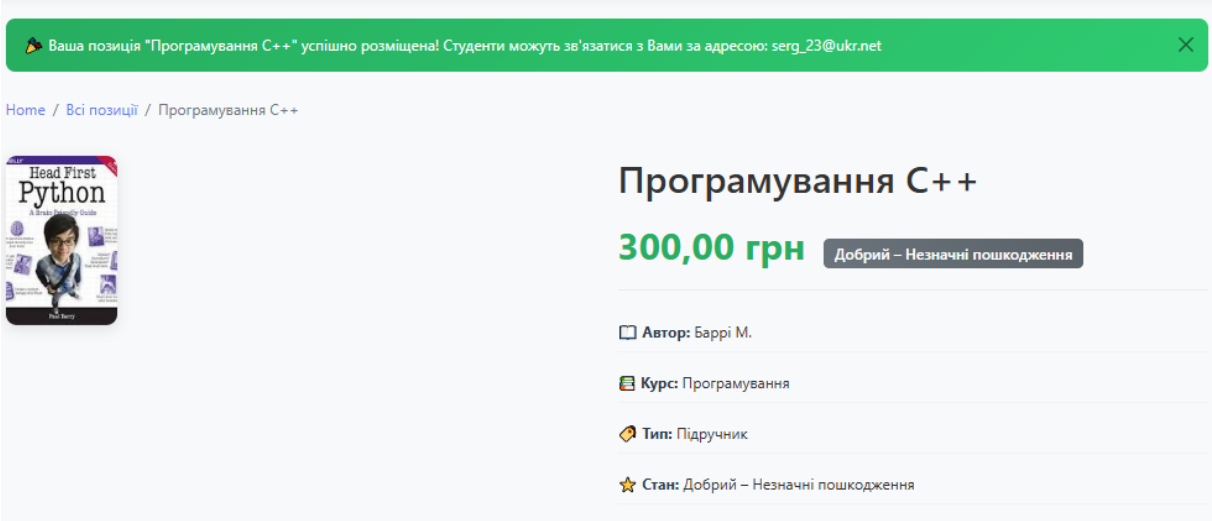

Рисунок 4.8. Інформація про публікація книги

Для покупки книги користувачу потрібно обрати книгу зі каталогу та натиснути кнопку Перегляд деталей. На екрані з’являється детальна інформація про книгу (рис.4.9). Внизу екрану знаходиться кнопка для зв’язку з продавцем. Дана кнопка доступна тільки для авторизованих користувачів. Її вигляд також залежить від типу контакту, який залишив користувач: телефон чи електронну пошту.

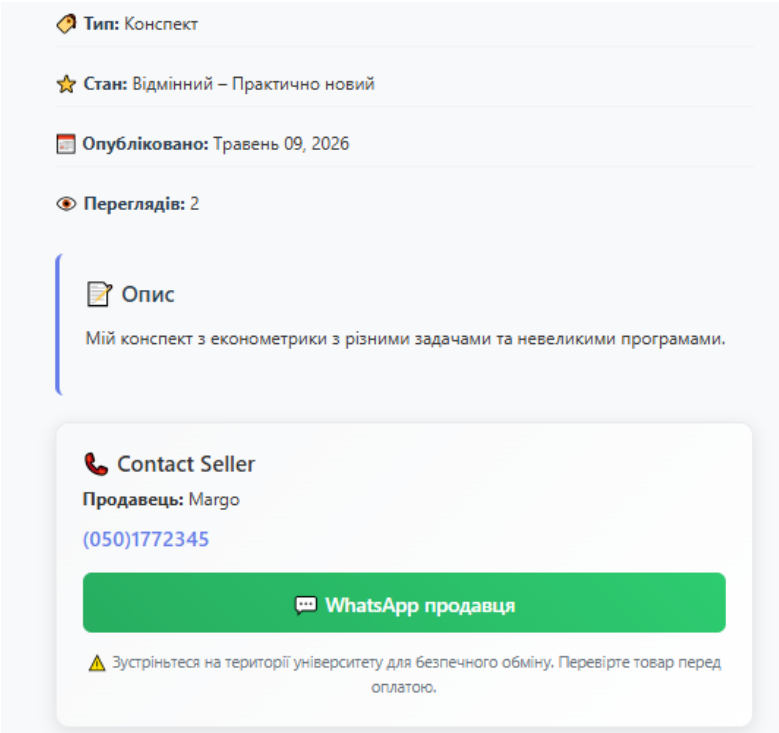

Рисунок 4.9. Можливість перегляду контактів продавця для авторизованих користувачів

Після натискання кнопки користувач переходить на сервер месенджера WhatsUp (рис.4.10). В умовах тестування був введений випадковий номер, тому результатом переходу є неіснуюча сторінка

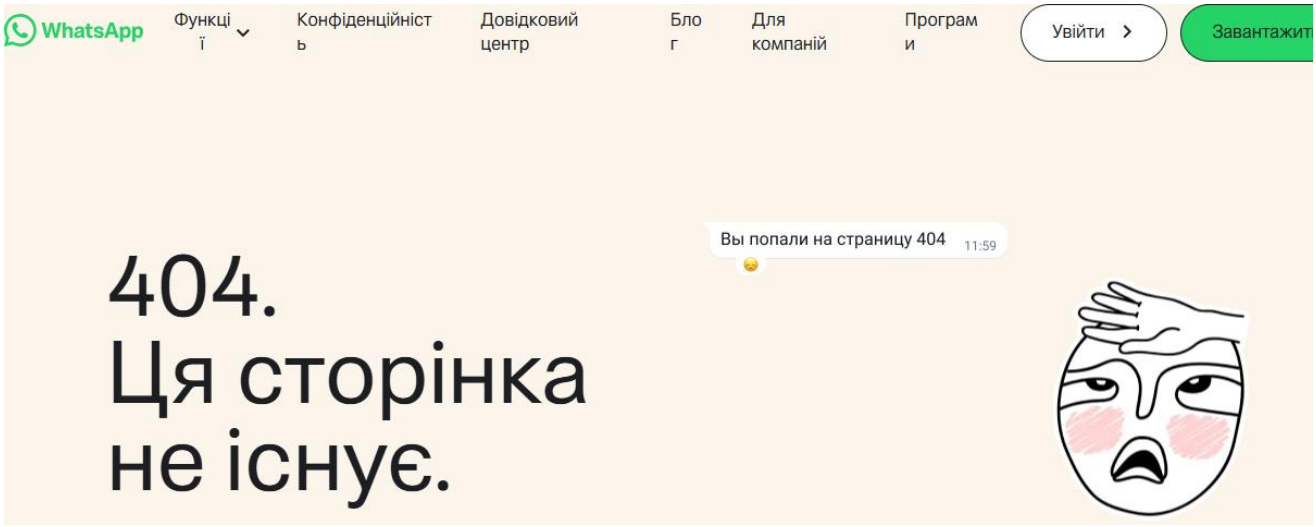

Рисунок 4.10. Перехід у WhatsUp

Для незареєстрованих користувачів контактні дані залишаються недоступними, причому вони не передаються в браузер, що унеможливлює їх захоплення (рис.4.11).

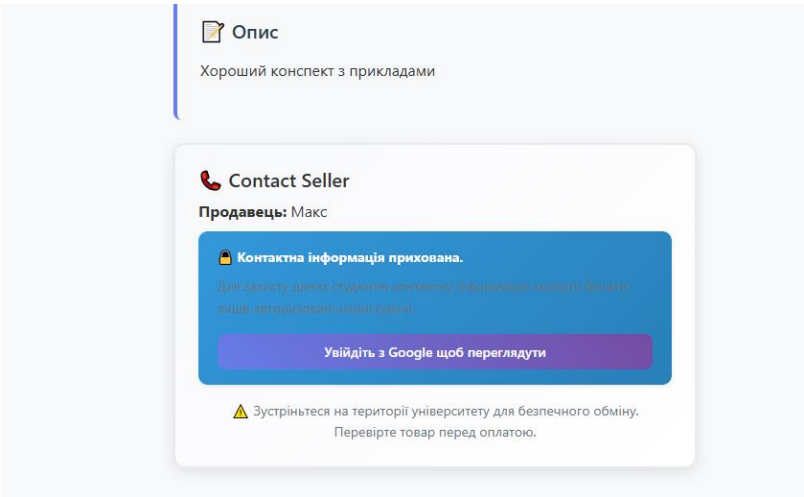

Рисунок 4.11. Дані контактів приховані для незареєстрованих користувачів

Для адміністратора системи доступ до адмін панелі здійснюється переходом за адресою http://127.0.0.1:8000/admin (рис.4.12).

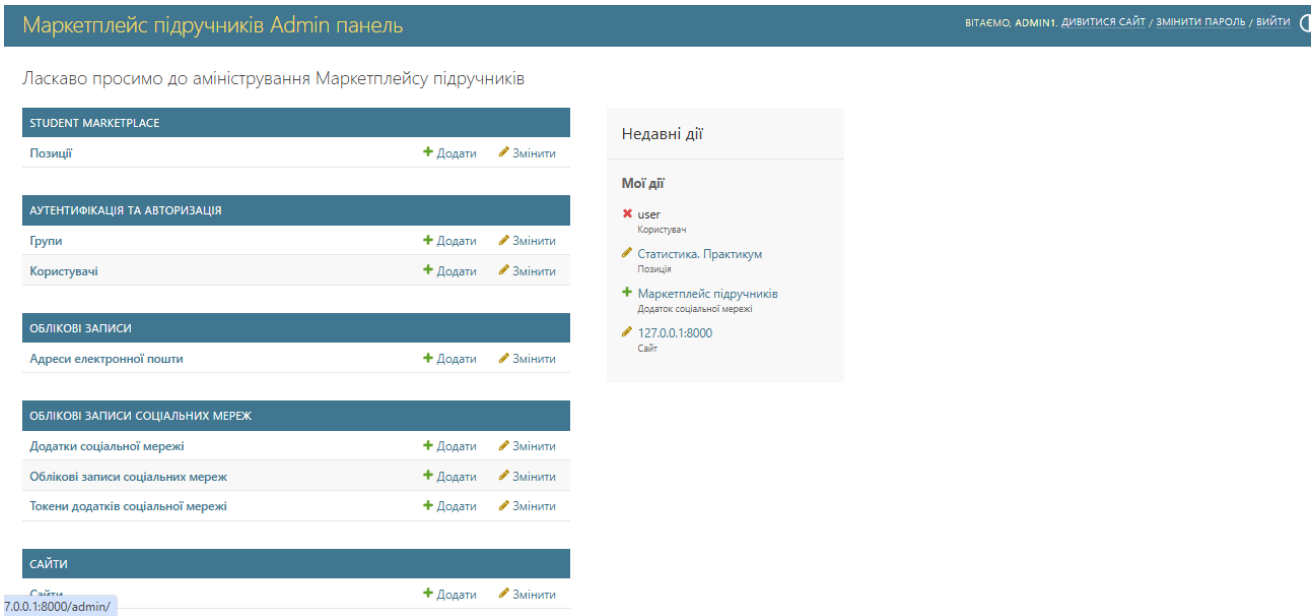

Рисунок 4.12. Адмін панель системи

Використовуючи дані панель адміністратор може управляти користувачами та їхніми оголошеннями (редагувати, видаляти тощо) (рис.4.13).

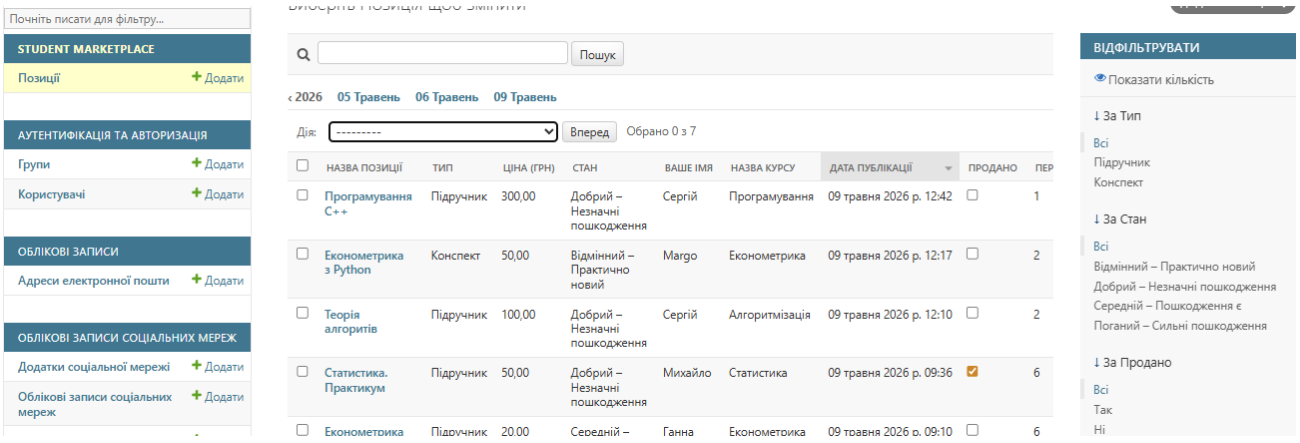

Рисунок 4.13. Доступ до оголошень користувачів через панель адміністратора

Такий підхід дозволяє залишати маркеплейс підручників керованим інструм

4.2. Аналіз результатів тестування

Проведене тестування розробленої системи показало, що система відповідає визначеним функціональним та нефункціональним вимогам.

Вона забезпечує користувачів зручним простим та зрозумілим інтерфейсом, який надає їм можливість продавати свої вживані навчальні матеріали (книги та конспекти).

З одного боку, система реалізована у вигляді дошки оголошень, що дає можливість кожному бажаючому виставити на продаж свої книги. З іншого боку, у системі передбачено використання авторизації Google OAuth для отримання контактів продавця та логування дій користувачів, що забезпечує захист та безпеку інформації.

Така система може бути використана як базова система для університету з мінімальними змінами в рамках міграції на СУБД PostgreSQL, яка забезпечує надійний захист бази даних.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено вебплатформу Маркетплейс підручників, яка вирішує актуальну проблему обміну навчальними матеріалами у студентському середовищі. Система надає можливість користувачам публікувати свої оголошення щодо продажу/ обміну вживаних підручників, а бажаючим студентам зв’язуватися з продавцями для подальшого обговорення угоди.

За результатами роботи можна зробити наступні висновки:

Паперові підручники і досі залишаються популярним джерелом знань у студентів, оскільки вони можуть містити як додаткову інформацію, так виступати як стимул для покращення результатів навчання. Тому перепродаж навчальних матеріалів залишається актуальною проблемою студентської спільноти. Аналіз існуючих аналогів показав, що більшість сайтів, де представлені вживані книжки, орієнтуються скоріше на популярну художню літературу, ніж на університетські підручники.

Під час виконання дослідження було проведено проєктування та розробка маркетплейсу вживаних навчальних матеріалів

На основі аналізу предметної області було обрано та впроваджено архітектурний патерн MTV (Model-Template-View) фреймворку Django. Це забезпечило чіткий поділ бізнес-логіки, структури даних та інтерфейсу користувача, що дозволило створити гнучку та масштабовану систему.

Розроблено зручну та різнопланову систему навігації контентом, яка включає багатоваріантне сортування (за ціною, актуальністю, популярністю) та пошук із використанням складних запитів. Це значно скорочує час пошуку необхідної літератури для студента.

Інтеграція протоколу Google OAuth 2.0 дозволила реалізувати безпечний вхід у систему без необхідності зберігання паролів на сервері. Це спростило процес реєстрації. І в майбутньому має підвищити рівень довіри користувачів до системи.

Впроваджено авторську систему логування активності, яка фіксує кожен факт запиту контактної інформації продавця. Таке рішення забезпечує аудит безпеки персональних даних та дозволяє адміністратору відстежувати ефективність взаємодії між користувачами.

Створений програмний продукт є готовим інструментом для цифровізації студентського побуту. Використання асинхронних технологій (AJAX) та оптимізація медіаконтенту забезпечують високу швидкість роботи додатку навіть на мобільних пристроях.

Напрямами подальшого розвитку проєкту є розробка системи внутрішніх миттєвих повідомлень (чатів) з використанням WebSockets, впровадження алгоритмів рекомендацій на основі уподобань користувача (Machine Learning), а також створення мобільного додатку для платформ iOS та Android на базі існуючого API.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wang Y., Fan L. Investigating students' perceptions concerning textbook use in mathematics: A comparative study of secondary schools between Shanghai and England. Journal of Curriculum Studies. 2021. Vol. 53, № 5. P. 675–691.

2. Sikorski J. F., Rich K., Saville. Student use of introductory texts: Comparative survey findings from two universities. Teaching of Psychology. 2002. Vol. 29, № 4. P. 312–312.

3. Son J. W., Diletti J. What Can We Learn from Textbook Analysis? What Matters? Research Trends in International Comparative Studies in Mathematics Education / edited by J. W. Son, T. Watanabe, J. J. Lo. Cham : Springer International Publishing, 2017. P. 3–32.

4. Zeng J., Cui Y. Student Utilization of Textbooks: A New Orientation in Textbook Research. Curriculum, Teaching Material, and Methodology. 2019. Vol. 39, № 11. P. 67–74.

5. American college students say they would rather study with real books, not laptops. URL: https://qz.com/316001/american-college-students-say-they-would-rather-study-with-real-books-not-laptops (дата звернення: 09.04.2026).

6. Мур Ч. Функціональні та нефункціональні вимоги. URL: https://www.guru99.com/uk/functional-vs-non-functional-requirements.html (дата звернення: 09.04.2026).

7. Mika A. MVC App Architecture. URL: https://www.ramotion.com/blog/what-is-mvvm/ (дата звернення: 09.04.2026).

8. Мартін Р. Чиста архітектура: мистецтво розробки програмного забезпечення. Львів : Фабула, 2019. 416 с.

9. Технології розроблення програмного забезпечення. Лабораторний практикум: навч. посіб. / КПІ ім. Ігоря Сікорського ; уклад.: О. А. Амонс, М. Ю. Мягкий. Київ : КПІ ім. Ігоря Сікорського, 2025. 113 с.

10. Дудзяний І. М. Об'єктно-орієнтоване моделювання програмних систем : навч. посіб. Львів : Видавничий центр ЛНУ ім. І. Франка, 2007. 108 с.

11. Allabarton R. What Is The UX Design Process? A Complete, Actionable Guide. URL: https://careerfoundry.com/en/blog/ux-design/the-ux-design-process-an-actionable-guide-to-your-first-job-in-ux/#1-what-is-ux-design (дата звернення: 09.04.2026).

12. Smith Q. Prototyping User Experience. 2019. URL: https://www.uxmatters.com/mt/archives/2019/01/prototyping-userexperience.php (дата звернення: 09.04.2026).

13. Wireframing. URL: https://xd.adobe.com/ideas/process/wireframing/ (дата звернення: 09.04.2026).

14. Соколова Н., Хара Г., Балалаєва О., Олевський В. Розробка шаблонів користувацького інтерфейсу на основі патернів проєктування. Вісник Приазовського державного технічного університету. Серія: Технічні науки. 2025. № 51. С. 9–18. DOI: 10.31498/2225-6733.51.2025.344593.

15. MVC pattern in Django. URL: https://medium.com/@iamwankang/mvc-pattern-in-django-ef1ca4d1d64e (дата звернення: 24.04.2026).

16. Django Підручник URL: https://w3schoolsua.github.io/django/index.html#gsc.tab=0 (дата звернення: 20.04.2026).

17. Mueller J. P. Security for Web Developers: Using JavaScript, HTML, and CSS. Sebastopol : O'Reilly Media, 2019. 349 p.

18. Django documentation | Django documentation. URL: https://docs.djangoproject.com/en/5.0/ (дата звернення: 07.05.2026).

19. How to manage static files (e.g. images, JavaScript, CSS) | Django documentation URL: https://docs.djangoproject.com/en/4.2/howto/static-files/ (дата звернення: 07.05.2026).

20. Jordon W. Python Django Web Development: The Ultimate Django Web Framework Guide for Beginners. Independently Published, 2019.

21. Чому слід використовувати SQLite URL: https://uk.peterfeatherstone.com/962-why-you-should-use-sqlite (дата звернення: 09.05.2026).

22. Безпека веб-додатків Django URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/web_application_security (дата звернення: 09.05.2026).

23. Братковський О. В., Тушич А. М. Удосконалення захисту даних у вебзастосунках із використанням Django. *Зв'язок*. 2024. № 2. DOI: 10.31673/2412-9070.2024.021417.

24. Using OAuth 2.0 to Access Google APIs URL: https://developers.google.com/identity/protocols/oauth2 (дата звернення: 02.05.2026).

25. Kharovyuk V. OAuth 2.0: що це таке і як правильно використовувати URL: https://webscraft.org/blog/oauth-20-scho-tse-take-i-yak-pravilno-vikoristovuvati?lang=de (дата звернення: 02.05.2026).