

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультет)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ВИЗНАЧЕННЯ ВАРТОСТІ ВИРОБІВ РУЧНОЇ РОБОТИ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
ESTIMATING THE COST OF HANDMADE PRODUCTS

Виконав: студент(ка) 2 курсу, групи КНТ-210м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

Грабчук А.А.

(прізвище та ініціали)

Керівник Степаненко О.О.

(прізвище та ініціали)

Рецензент Поляков М.О.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Грабчука Андрія Андрійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів визначення вартості виробів ручної роботи

Research and Software Implementation of Methods for Estimating the Cost of Handmade Products

керівник проєкту (роботи) Степаненко Олександр Олексійович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “5” листопада 2021 року №435

2. Строк подання студентом проєкту (роботи) 13 грудня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Степаненко О.О., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 7 ” вересня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-6 тижні	Розділи 1, 2
3	Розробка архітектури програми	7 тиждень	Розділ 3
4	Розробка програми	8-12 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	13-14 тижні	Розділ 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	15 тиждень	Додатки
7	Захист роботи	16 тиждень	

Студент(ка)

(підпис)

Грабчук А.А.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Степаненко О.О.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 91 с., 28 рис., 3 табл., 3 дод., 17 джерел.

ВИРОБИ РУЧНОЇ РОБОТИ, МІШКИ СЛІВ, ПРОГНОЗУВАННЯ,
ДЕРЕВА РІШЕНЬ, ВЕБЗАСТОСУНОК.

Об'єкт роботи – процес продажу виробів ручної роботи.

Предмет роботи – методи визначення вартості виробів ручної роботи.

Мета роботи – розробити програмне забезпечення продажу виробів ручної роботи для підвищення обґрунтованості рішень, які приймаються продавцями щодо встановлення вартості виробів ручної роботи на конкретній платформі або загалом.

Матеріали, методи та технічні засоби: мова програмування Python, вебфреймворк Django, система керування базами даних MySQL, бібліотека LightGBM.

Результати. Наукова новизна роботи полягає в запропонованому методі визначення вартості виробів ручної роботи, який ґрунтується на обробленні тексту за допомогою моделі мішків слів з подальшим застосуванням дерев рішень, що дозволяє приймати рішення, враховуючи не тільки витрати на створення виробу, але і очікувану вартість в умовах відповідної платформи.

Висновки. Розроблена програма дозволяє продавцям створювати власні магазини, об'єднуючи таким чином вироби ручної роботи, які вони створюють і бажають продавати, керувати замовленнями покупців, відповідно покупцям дозволяє організовувати пошук виробів ручної роботи, замовляти їх, надає інструментар відгуків для створення зворотного зв'язку про вироби.

Галузь використання – продаж виробів ручної роботи.

ABSTRACT

Explanatory note to the qualifying work of the master: 91 pages, 28 figures, 3 tables, 3 appendixes, 17 sources.

HANDMADE PRODUCTS, WORD BAGS, FORECASTING, DECISION TREES, WEB APPLICATION.

The object of work is the process of selling handmade products.

Subject of work is methods of determining the price of handmade products.

The purpose of the work is to develop software for the sale of handmade products to increase the validity of decisions made by sellers to determine the price of handmade products on a particular platform or in general.

Materials, methods and technical means: Python programming language, Django web framework, MySQL database management system, LightGBM library.

Results. The scientific novelty of the work lies in the proposed method of determining the value of handmade products, which is based on text processing using a model of word bags with the subsequent use of decision trees, which allows to make a decision, taking into account not only product costs but also expected cost.

Conclusions. The developed program enables sellers to create their own stores, thus combining handmade products which they create and want to sell, manage customer orders, respectively, enables customers to organize the search for handmade products, order them, provides feedback tools to create feedback about ware.

Field of use is sale of handmade products.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Аналіз існуючих програмних засобів продажу виробів ручної роботи	11
1.2 Основні сутності предметної області.....	15
1.3 Висновки за розділом 1	17
2 Матеріали і методи.....	19
2.1 Методи роботи з текстами.....	19
2.2 Дерева рішень	21
2.3 Метод визначення вартості виробів ручної роботи.....	23
2.4 Висновки за розділом 2	24
3 Проектування програмного забезпечення	26
3.1 Вибір мови програмування	26
3.2 Функціональні вимоги до програми.....	26
3.3 Проектування бази даних	31
3.4 Структура програми.....	38
3.5 Висновки за розділом 3	39
4 Основні рішення щодо реалізації компонентів системи.....	41
4.1 Алгоритм функціонування програми.....	41
4.2 Опис основних розроблених класів	43
4.3 Висновки за розділом 4	50
5 Експлуатація, тестування та експериментальне дослідження програми	51
5.1 Призначення програми	51
5.2 Умови виконання програми	51
5.3 Робота з програмою	51
5.4 Експериментальне дослідження	55
5.5 Висновки за розділом 5	55
Висновки	56

Перелік джерел посилання	57
Додаток А Технічне завдання	59
Додаток Б Текст програми	64
Додаток В Слайди презентації.....	83

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

Визначення вартості виробів ручної роботи може бути складним: занадто низька ціна означає, що не буде отримано достатнього прибутку, щоб залишатися в справі, зависокі ціни означають, що клієнти не куплять продукти, тому очевидно, що перше, що потрібно визначити, – це скільки було витрачено на матеріали для виготовлення виробу, тоді потрібно вирішити, скільки коштує власний час, усе це в сумі дозволить обчислити вартість виробу, яку необхідно далі помножити на надбавку, що зазвичай встановлюється у вигляді значення від 2 до 2,5, після цього вартість множить на надбавку ще раз і буде отримано роздрібну ціну [1].

Особливістю предметної області є наявність великої кількості виробів при активній участі продавців у роботі програмного забезпечення (ПЗ). Кожен продавець може виставляти достатньо велику кількість оголошень, адже кожний дрібний елемент ручної роботи може ставати окремим виробом. Тому важливим завданням ПЗ стає надати ефективні інструменти роботи з цими оголошеннями.

Окрім інструментів, які надаються покупцям, звичайно потрібно організувати ефективний процес роботи і для продавця. Для підтримки такого процесу потрібно виконувати автоматичне визначення вартості виробів ручної роботи, таким чином створюючи пораду для прийняття рішення продавцем. Тоді окрім власного досвіду, який може бути і достатньо обмеженим на конкретній платформі, користувач може отримати інформацію про автоматично визначену вартість, таким чином узгоджуючи позиції і формуючи універсальне рішення. Вартість продажу на різних платформах може відрізнятися, тому такий інструмент дозволить продавцю орієнтуватися на ті можливості, які виникають у нього при роботі в ринкових умовах конкретно на цій платформі. Відповідно наявність такого інструменту є кроком з підтримки збільшення кількості користувачів на платформі щодо

відкриття нових магазинів, для яких буде зручно працювати з власними оголошеннями таким чином.

У роботі потрібно досягти мети, що полягає в розробленні ПЗ продажу виробів ручної роботи для підвищення обґрунтованості рішень, які приймаються продавцями щодо встановлення вартості виробів ручної роботи на конкретній платформі або загалом.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз існуючих програмних засобів продажу виробів ручної роботи

Одним з найвідоміших вебзастосунків у сфері продажу виробів ручної роботи є Etsy (рис. 1.1) [2].

Etsy забезпечує надійну платформу, за допомогою якої онлайн продавці виробів ручної роботи можуть пропонувати свої товари та зв'язуватися з покупцями:

- на платформі знаходиться мільйонна спільнота виробників-однодумців та інді-майстрів: часто це перша зупинка для покупців, які шукають щось саморобне;

- коло високомотивованих покупців: згідно з останнім річним звітом компанії, минулого року 46,4 мільйона людей зробили покупки на Etsy, і це не просто покупці (на відміну від покупців на більш загальних сайтах роздрібної торгівлі), користувачі Etsy, як правило, дуже мотивовані зробити покупку;

- декілька рівнів підтримки: Etsy пропонує багато інструментів, які допомагають продавцям оптимізувати свої вітрини в Інтернеті і навіть керувати запасами та бухгалтерією, вебсайт також забезпечує посередництво між постачальниками та клієнтами у разі виникнення суперечки, крім того, форуми та блоги Etsy є чудовим місцем для спілкування продавців, обміну порадами, запитаннями та порадами;

- жорстка конкуренція: приблизно 2,7 мільйона продавців були активними на Etsy минулого року, оскільки так багато постачальників (багато з яких пропонують подібні та споріднені продукти), які борються за увагу користувачів Інтернету, суперництво може стати інтенсивним, що робить 5-зіркові огляди, стратегічне ціноутворення та ідеальне представлення в Інтернеті є першорядним для успіху;

– Etsy – це лише вебсайт, де постачальники можуть спілкуватися з покупцями, на відміну від Amazon, Etsy не має власних складів чи служб доставки.

– програма не має місячної чи річної підписки чи плати за реєстрацію для продажу, замість цього компанія стягує такі збори за перелік і продаж: Etsy стягує з онлайн продавців ремісничих виробів 20 центів за оголошення, кожен з яких зберігається протягом 4 місяців, Etsy стягує комісію за транзакцію в розмірі 5% і комісію за обробку платежу в розмірі 3% та 25 центів за кожен продаж [3].

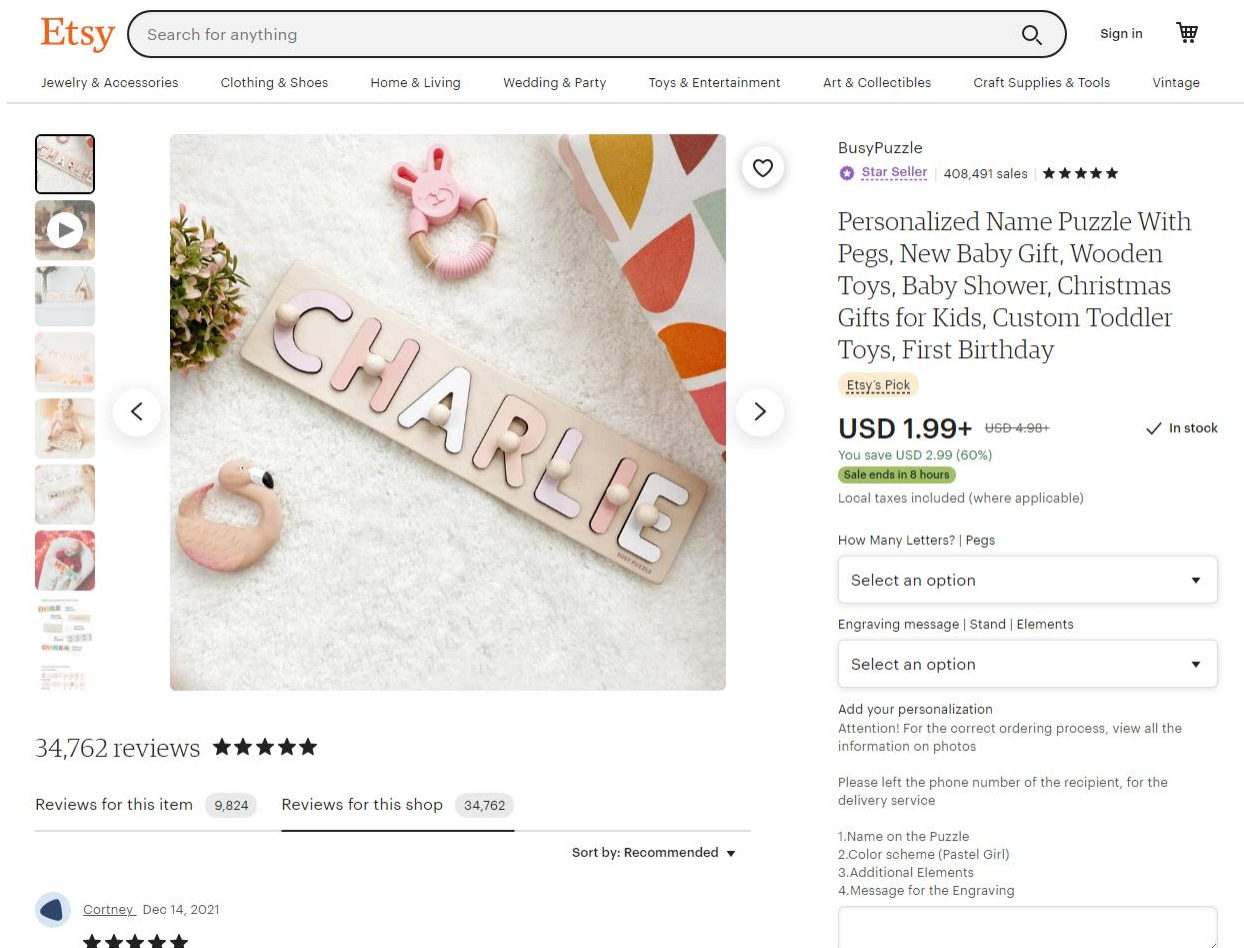


Рисунок 1.1 – Інтерфейс вебзастосунку Etsy

Etsy – це програмний засіб, який дозволяє продавцям і покупцям знаходити спосіб відповідно продажу і придбання виробів ручної роботи, створюючи таким чином ринок у цій сфері.

Для підтримки даних процесів програма має достатньо зручно організовані засоби роботи окрім стандартного оформлення замовлення на придбання певних виробів. Так користувач може виконувати уподобання виробів, вносити їх до власних переліків, виконувати пошук виробів, переглядати вироби за категоріями, сортуючи їх за релевантністю запиту, за кількістю відгуків або обираючи одну з цінових категорій. Користувачі можуть залишати відгуки про вироби, можуть обирати додаткові параметри виробів при їх замовленні.

Вебсайт ArtFire (рис. 1.2) є однією з найпопулярніших альтернатив Etsy [4].

Ринок ArtFire функціонує в набагато меншому масштабі, ніж Etsy, хоча деякі онлайн продавці можуть насправді віддати перевагу цьому рішенню, враховуючи при цьому наступні переваги і недоліки даного застосунку:

- менший ринок означає м'яку конкуренцію між продавцями;
- ArtFire не показує рекламу на сторінках товарів або на сторінці магазину;
- залежно від плану підписки можна мати до 250 активних списків лише за 23 центи за один продукт, при цьому чим більше користувач платить ArtFire на місяць, тим більше оголошень буде отримано;
- програма пропонує блоги та форуми для підтримки, а також пропонує множину рекомендацій щодо маркетингу та підтримки інтернет-магазину;
- менший ринок і менша видимість означають менше клієнтів, що є недоліком;
- продавати на ArtFire дорожче, ніж на Etsy, а оголошення зберігаються менший період часу;
- програма ArtFire пропонує три різні плани підписки [3].

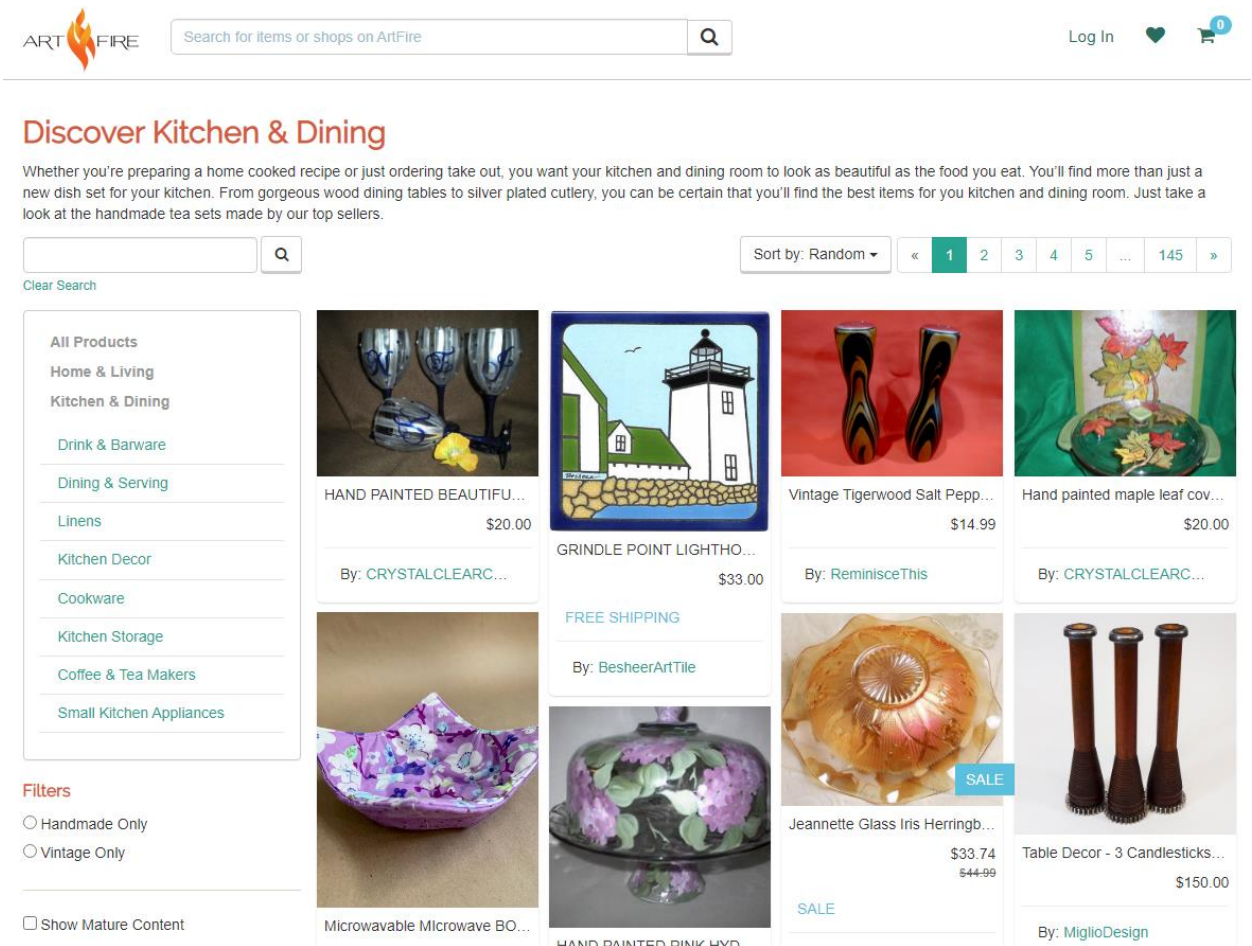


Рисунок 1.2 – Інтерфейс вебзастосунку ArtFire

ArtFire дозволяє переглядати вироби за категоріями, виконувати їх пошук, у тому числі пошук серед виробів магазину, сортуючи результати за часом створення, за вартістю, за кількістю переглядів.

Користувач може отримати інформацію про різні доступні значення певних характеристик виробу, але не може обрати потрібний йому варіант, він може тільки задати загалом кількість відповідних найменувань.

Користувачам також доступна можливість уподобання виробів, створення переліків виробів.

Основним аналогом було обрано програму Etsy, яка відрізняється від ArtFire можливостями персоналізації виробів, вибору потрібних характеристик виробу. Програма, яка розробляється в роботі, відрізняється відсутністю платних підписок та автоматичним визначенням вартості виробів.

За результатами аналізу оглянутих програмних рішень було створено табл. 1.1.

Таблиця 1.1 – Результати аналізу програмних аналогів

Характеристика	Etsy	ArtFire	Програма-розробка
Платні підписки	Так	Так	Ні
Врахування можливості вибору певної характеристики виробу	Так	Ні	Так
Створення власних переліків виробів	Так	Так	Так
Персоналізація виробів	Є	Немає	Є
Автоматичне визначення вартості виробів	Ні	Ні	Так

1.2 Основні сутності предметної області

Виконаний огляд програм, які виконують подібні функції, дозволив сформулювати основний набір таких функцій. Для підтримки їх роботи потрібно визначити основні сутності предметної області, які будуть перелічені нижче.

Сутність Category визначає категорії, в які об'єднуються вироби ручної роботи, адже для них дуже важливе упорядкування, інакше знайти потрібний буде практично неможливо або достатньо складно. У такому випадку витрати часу будуть такими великими, що людина просто відмовиться від пошуку.

Щоб підвищити зручність систематизації виробів у межах всієї системи, усі категорії були розбиті на підкатегорії. Сутність Subcategory визначає підкатегорію, якій належить виріб ручної роботи.

Для того щоб додати можливості об'єднання подібних виробів, а також внесення можливості персоналізації виробів, коли може бути зазначено ім'я покупця на виробі або визначено якусь фразу, яку бажає бачити покупець, необхідно визначити опції персоналізації та категоризації за кольорами одних і тих же виробів. Персоналізація може бути вирішення на рівні замовлення, а ось колір потребує окремої сутності, адже ці опції мають бути визначені окремо для кожного виробу, щоб розуміти, які саме кольори доступні в цьому виробі.

Сутність `ColorOption` визначає опцію вибору кольору виробу.

Сутність `List` визначає перелік виробів, який створює користувач (покупець) для того, щоб відслідковувати зміни їх вартості, а також повертатися до окремих виробів і не виконувати їх пошук заново, що має сприяти зменшенню витрат часу.

Сутність `Handmade` визначає безпосередньо сам виріб ручної роботи, роботи яким є основою роботи всієї програми.

Сутність `HandmadeLike` визначає уподобання покупцем виробу ручної роботи.

Роботи з продажу виробів ручної роботи реалізована через так звані магазини, в яких продаються певні найменування виробів. Відповідно кожен магазин може бути представлений набором акаунтів, які працюють з його замовленнями і виробами.

Сутність `Shop` визначає магазин з продажу виробів ручної роботи.

Сутність `Buyer` визначає покупця. Це звичайний користувач, який має обмежену функціональність у програмі, яка стосується впорядкування певних виробів потрібним йому чином, придбання виробів.

У свою чергу окремо виділено акаунти продавців. Вони прив'язані до конкретного магазину виробів ручної роботи. Тому визначаються сутністю `ShopAccount`.

Сутність Order визначає замовлення на купівлю виробів ручної роботи. Передбачається, що до замовлення можуть входити в тому числі вироби з різних магазинів.

Для того щоб покупці могли орієнтуватися в тому, на скільки реальні вироби відрізняються від описів, на скільки якісно працює магазин, потрібно надати можливість створення відгуків про вироби самі і про магазин загалом. Це дозволить врахувати специфіку предметної області, а також надати потенційно вільний доступ до відкриття магазинів на платформі, що розробляється.

Сутність ShopReview визначає відгук покупця про магазин, в якому він придбав вироби ручної роботи.

Сутність HandmadeReview визначає відгук покупця про виріб ручної роботи.

Сутність PriceHistory визначає вартість ціни виробу ручної роботи. Дану сутність було виокремлено, оскільки вона дозволяє переглядати, як змінилась вартість виробу.

1.3 Висновки за розділом 1

Завданнями магістерської роботи є:

- виконати аналіз предметної області;
- виконати дослідження методів визначення вартості виробів ручної роботи;
- виконати проектування ПЗ продажу виробів ручної роботи;
- виконати реалізацію ПЗ продажу виробів ручної роботи.

У першому розділі розв'язано першу задачу – виконано аналіз предметної області. Розглянуто і проаналізовано можливості основних програмних засобів у цій області, найвідомішим з яких є Etsy, а також ArtFire. Основним прототипом програми, що розробляється, було обрано Etsy, порівняно з нею розробка повинна головним чином надавати

можливості автоматичного визначення вартості виробів. Також було виконано виділення основних сутностей предметної області, необхідних для забезпечення роботи з функціями, що надають основний масив засобів серед програмних аналогів.

2 МАТЕРІАЛИ І МЕТОДИ

Складність встановлення вартості виробів ручної роботи автоматизованим чином полягає в тому, що інформація про такі вироби зазвичай недостатньо структурована. Багато в чому не пояснюється, що такі вироби зазвичай належать до дуже різних товарів і фактично їх об'єднувати і систематизувати в межах однієї системи достатньо складно. Тому зазвичай єдиними характеристиками таких виробів є їх назва, текстовий опис та фотографія або фотографії. Все це ускладнює подальшу роботу.

У даній роботі вирішено було зробити акцент на оброблення виробів ручної роботи за допомогою їх текстових описів.

2.1 Методи роботи з текстами

Обробка природної мови – це методологія, призначена для виділення понять і значень із створеного людиною неструктурованого тексту, вона призначена для реалізації за допомогою комп'ютерних алгоритмів, щоб її можна було швидко та надійно запустити на корпусі документів [5].

Модель мішків слів – це представлення тексту, яке можна використовувати алгоритмами машинного навчання: більшість алгоритмів машинного навчання не можуть працювати безпосередньо з нечисловими даними, тому потрібно використовувати різні методи кодування, як-от One-Hot-Encoding, щоб перетворити ці текстові дані в числові матриці, які будуть використовуватися алгоритмом, у свою чергу модель мішків слів має на меті витягти елементи з тексту, які можуть бути використані в подальшому моделюванні [6].

Мішок слів не враховує положення слова в документі: ідея полягає в тому, щоб підрахувати, скільки разів кожне слово з'являється в кожному з документів [7].

При цьому на виході роботи методу утворюється вектор або ціла матриця, якщо аналізується декілька речень текстів: розмір матриці становить кількість документів помножену на розмір словника, в якому мають міститись всі слова, які можуть зустрітись в даному корпусі текстів[7].

Після декількох етапів попередньої обробки, включаючи токенізацію (виділення слів з речення), видалення стоп-слів, нормалізацію лексем і створення основного словника, техніка мішка слів може бути використана для представлення кожного слова, що залишилося, як ознаки документа: етапи попередньої обробки спрощують документи, але також потенційно погіршують значення, значення ознак у моделі мішків слів можна змінити за допомогою таких методів, як підрахунок термінів, частота термінів і частота термінів – зворотна частота документа (TF-IDF) [5].

TF-IDF є підходом до представлення слів у вигляді векторів для отримання інформації з текстових даних, при цьому числове представлення слів є абстрактним, значення, присвоєні кожному слову для кожного документа, не мають конкретного значення, абстрактний характер TF-IDF впливає з того факту, що він не є фактичною статистикою, це продукт 2 окремих статистичних даних:

- частота термінів (TF): термін частота відноситься до кількості разів, яку зустрічаються слова у виділених частинах тексту, відповідно дана оцінка обчислюється як відношення кількості таких випадків до кількості слів у документі загалом;

- зворотна частота документів: цей компонент використовується для зміщення слів, які зустрічаються з високою частотою в багатьох документах, обернену частоту документа можна отримати за формулою, в якій обчислюється логарифм від відношення кількості документів до кількості документів з заданим словом, при цьому компонент IDF розглядається як противага, яка гарантує, що компонент TF не переоцінює релевантність слова в документі [8].

2.2 Дерева рішень

Дерева рішень є звичайним способом представлення процесу прийняття рішень через розгалужену деревоподібну структуру, він часто використовується для планування та планування ділових і операційних рішень у вигляді візуальної блок-схеми, цей підхід передбачає розгалуження рішень, які закінчуються результатами, що призводить до деревоподібної структури або візуалізації, відносно якої можна відзначити:

- дерева рішень використовуються як підхід у машинному навчанні для структурування алгоритму;
- алгоритм дерева рішень буде використовуватися для поділу функцій набору даних за допомогою функції вартості;
- дерево рішень вирощується перед оптимізацією для видалення гілок, які можуть використовувати невідповідні функції, створюючи процес, який називається обрізанням;
- можна встановити такі параметри, як глибина дерева рішень, щоб знизити ризик перенавчання або занадто складного дерева.;
- більшість дерев рішень у машинному навчанні використовуються для задач класифікації, щоб категоризувати об'єкти за вивченими ознаками, цей підхід також можна використовувати для задач регресії або як метод прогнозування безперервних результатів на основі невидимих даних;
- основними перевагами використання дерева рішень у машинному навчанні є його простота, оскільки процес прийняття рішень легко уявити та зрозуміти, однак дерева рішень у машинному навчанні можуть стати занадто складними, створюючи дуже дрібні гілки, тому обрізання структури дерева часто є необхідністю;
- дерева рішень – це спосіб моделювання рішень і результатів, відображення рішень у структурі, що розгалужується, дерева рішень використовуються для обчислення потенційного успіху різних серій рішень, прийнятих для досягнення конкретної мети;

– дерева рішень складаються з різних вузлів: кореневий вузол є початком дерева рішень, яке зазвичай являє собою весь набір даних у рамках машинного навчання, листкові вузли є кінцевою точкою гілки або кінцевим результатом серії рішень, дерево рішень не буде розгалужуватися далі від листкового вузла, у деревах рішень у машинному навчанні ознаками даних є внутрішні вузли, а результатом є листовий вузол [9].

Існують різні алгоритми навчання дерев рішень.

LightGBM вирощує дерева по листам (найкращі спершу): метод вибирає лист з максимальною дельтою втрати для зростання, а утримуючи такий лист фіксованим, алгоритми, що працюють по листу, мають тенденцію досягати менших втрат, ніж рівневі алгоритми, при цьому другі можуть спричинити перенавчання, коли обсяг даних малий, тому LightGBM включає параметр `max_depth` для обмеження глибини дерева [10].

LightGBM – це високопродуктивний алгоритм підвищення градієнта для багатьох завдань машинного навчання, що використовує підхід вирощування дерева по листам, використовує дискретні значення, які були перетворені з безперервних значень, забезпечують високу точність і швидшу процедуру навчання [11].

Основні поняття, які стосуються LightGBM:

– ансамблеве навчання – це модель, яка робить прогнози на основі низки різних моделей: завдяки поєднанню декількох різних моделей навчання ансамблю, як правило, більш гнучке (менше упередженості) і менш чутливе до даних (менша дисперсія);

– бустинг – послідовне навчання ансамблю моделей: кожна модель вчиться на помилках попередньої моделі;

– бустинг працює за принципом покращення помилок попереднього учня через наступного: у бустингу використовуються слабкі учні, які працюють лише трохи краще, ніж випадковий шанс, підвищення зосереджено на послідовному додаванні цих слабких учнів і фільтрації спостережень, які учень отримує на кожному кроці, в основному наголос

робиться на розвитку нових слабких учнів, щоб на кожному кроці справлятися з рештою важких спостережень;

- у деревах рішень, що підвищують градієнт, об'єднується багато слабких учнів, щоб створити одного сильного, слабкими учнями тут є окремі дерева рішень;

- усі дерева з'єднані послідовно, і кожне дерево намагається мінімізувати помилку попереднього дерева: завдяки такому послідовному з'єднанню алгоритми підвищення зазвичай повільно навчаються, але також дуже точні;

- слабкі учні пристосовані таким чином, що кожен новий учень вписувався в залишки попереднього кроку, щоб модель удосконалювалася, остаточна модель агрегує результат кожного кроку, і таким чином досягається сильний учень [12].

2.3 Метод визначення вартості виробів ручної роботи

Метод визначення вартості виробів ручної роботи дозволяє вирішити задачу, в якій потрібно на основі текстового опису виробу ручної роботи визначити вартість такого виробу, за яку його очікувано можна продати.

Для цього на першому етапі необхідно сформувати вибірку даних, що складається з пари: опис виробу – вартість виробу.

Одразу після цього потрібно виконати передоброблення описів виробів ручної роботи. Для цього мають бути вилучені стоп-слова, слова, які не несуть змістовного сенсу, спеціальні символи, знаки пунктуації та мають бути відділені безпосередньо слова.

На наступному етапі потрібно застосувати модель мішка слів для кодування слів у описах і визначення цифрового вектору, який характеризує частоту входження слів у текст.

Після цього отримані частоти на попередньому етапі мають бути перетворені на оцінки TF-IDF.

На наступному кроці відбувається навчання моделі на основі дерев рішень з використанням алгоритму LightGBM.

Оцифровані на попередньому етапі ознаки текстових описів виробів ручної роботи мають бути подані разом з вартістю виробів для навчання дерева рішень. У результаті має бути організована модель, здатна прогнозувати вартість виробу за попередньо обробленим текстовим описом.

Результуюча процедура роботи метода представлена на рис. 2.1.

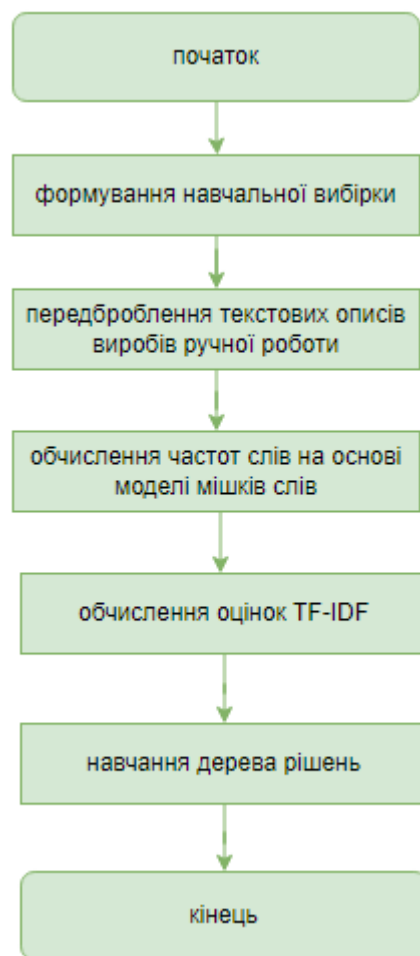


Рисунок 2.1 – Метод визначення вартості виробів ручної роботи

2.4 Висновки за розділом 2

Проведено аналіз технік, які використовуються для роботи з текстом з метою переведення його до форми вектора. Розглянуто принципи створення

дерев рішень. Запропоновано метод визначення вартості виробів ручної роботи, який ґрунтується на обробленні тексту за допомогою моделі мішків слів з подальшим застосуванням дерев рішень, що дозволяє приймати рішення, враховуючи не тільки витрати на створення виробу, але і очікувану вартість в умовах відповідної платформи.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір мови програмування

PHP є найбільш широко використовуваною мовою вебпрограмування на даний момент, проте достатньо високою є підтримка в цій галузі і мови Python, яка має свої особливості, що можуть бути важливими в розробці, яка виконується [13]-[15]. Тому порівняння виконувалось саме цих мов для вибору мови розробки (табл. 3.1).

Таблиця 3.1 – Порівняння мов програмування

Критерій	PHP	Python
Вебпрограмування	+	+
Вебфреймворки	+	+
Засоби розширення можливостей за допомогою бібліотек	+	++
Оброблення даних	+—	++
Швидкість написання коду	+—	++

У результаті для розробки обрано Python, враховуючи, що він має величезну кількість додаткових бібліотек, що дозволяє швидше писати код, окрім того сама мова на стільки гнучка в своїх засобах, в тому числі через синтаксис мови, що створювані проєкти на її основі вимагають менше зусиль, до того ж вона є дуже відомою щодо ефективності своїх рішень відносно оброблення даних.

3.2 Функціональні вимоги до програми

Функціональними вимогами до програми, що мають бути реалізовані для роботи продавців виробів ручної роботи та покупців, належать:

- виставлення власних виробів ручної роботи на продаж;
- керування оголошеннями про продаж власних виробів ручної роботи;
- створення замовлень про придбання виробів ручної роботи;
- керування створеними замовленнями про придбання виробів ручної роботи в магазині;
- пошук виробів ручної роботи;
- публікація відгуків про виріб ручної роботи;
- визначення середньої оцінки виробу ручної роботи;
- публікація відгуків про магазин;
- визначення середньої оцінки магазину;
- уподобання виробу ручної роботи;
- створення власних переліків з виробами ручної роботи;
- керування і перегляд переліків ручної роботи;
- перегляд уподобаних користувачем виробів ручної роботи;
- збереження історії зміни вартості виробу ручної роботи;
- прогнозування вартості виробу ручної роботи;
- перегляд виробів ручної роботи за заданою підкатегорією;
- формування рейтингу виробів ручної роботи за середньою оцінкою, кількістю відгуків, кількістю уподобань, кількістю замовлень;
- перегляд і керування власними замовленнями виробів ручної роботи;
- створення магазину для продажу виробів ручної роботи;
- визначення облікових записів користувачів, які можуть працювати з замовленнями даного магазину;
- перегляд переліку виробів ручної роботи, що продаються магазином;
- авторизація облікових записів звичайного користувача (покупця) та представників магазину;
- реєстрація облікових записів покупців.

Тепер потрібно на основі сформульованого набору функціональних вимог визначити сценарії використання програми різними типами користувачів.

Користувач без авторизації (рис. 3.1) є основою роботи будь-якого користувача з програмою, адже спочатку будь-який користувач ідентифікується саме так. Він отримує засоби пошуку виробів ручної роботи та перегляду їх різних переліків. Користувачу також доступний сценарій авторизації, для якого альтернативним варіантом у випадку відсутності акаунта є сценарій реєстрації.

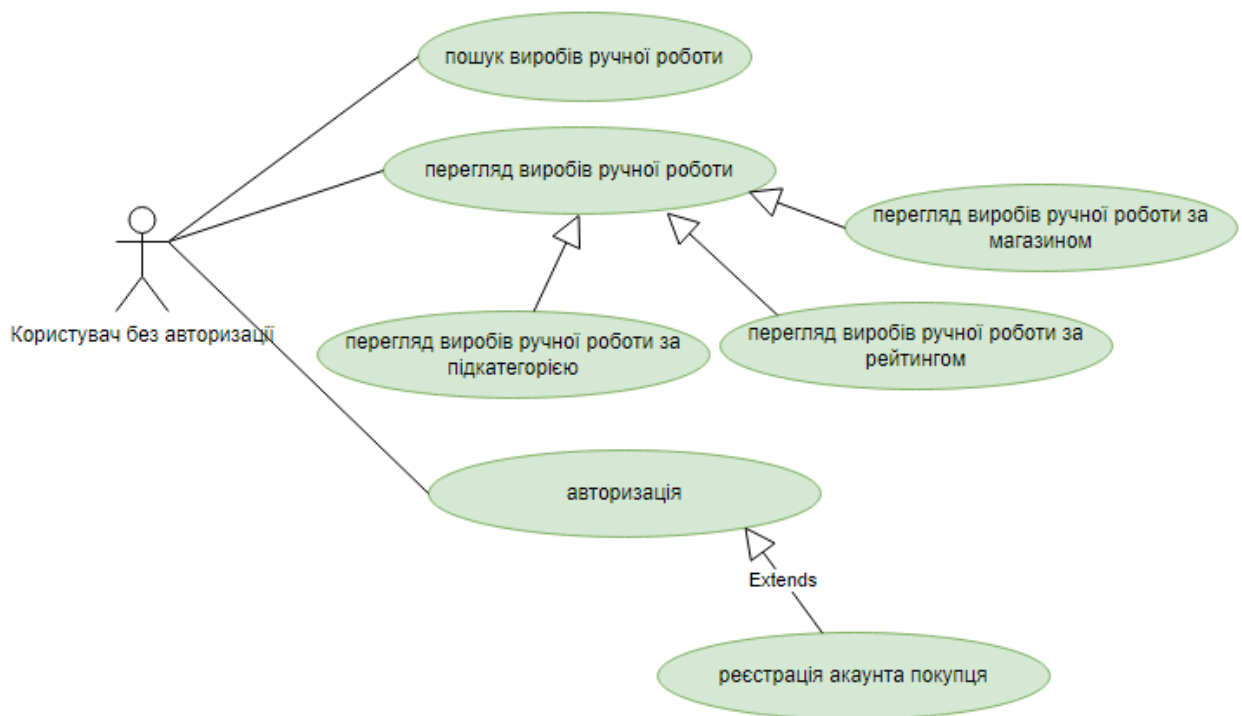


Рисунок 3.1 – Діаграма прецедентів користувача без авторизації

Повний перелік основних сценаріїв користувача без авторизації:

- пошук виробів ручної роботи;
- перегляд виробів ручної роботи за заданою підкатегорією;
- перегляд переліку виробів ручної роботи, що продаються магазином;
- авторизація;

– реєстрація.

Покупець (рис. 3.2) має такий же набір засобів пошуку оголошень про продаж виробів ручної роботи, але при цьому він також має розширений набір засобів підтримки роботи з цими виробами, що включають засоби створення уподобаних виробів, переліків виробів, а також безпосередньо засоби створення замовлень і керування створеними власними замовленнями.

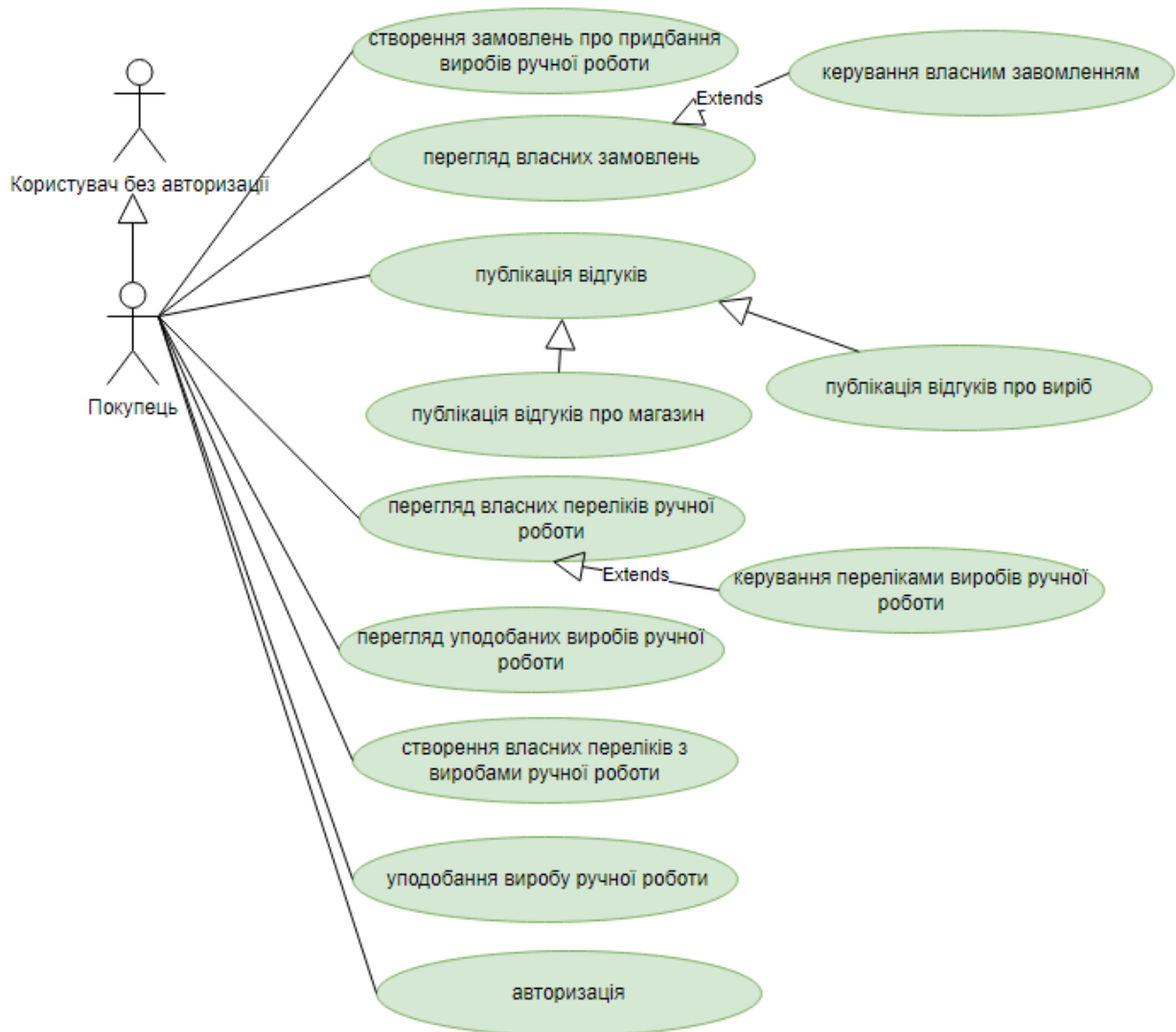


Рисунок 3.2 – Діаграма прецедентів покупця

Основні сценарії покупця:

- створення замовлень про придбання виробів ручної роботи;
- перегляд власних замовлень;
- публікація відгуків про магазин;

- публікація відгуків про виріб ручної роботи;
- перегляд переліків ручної роботи;
- уподобання виробу ручної роботи;
- створення власних переліків з виробами ручної роботи;
- перегляд уподобаних користувачем виробів ручної роботи.

До альтернативних сценаріїв належать:

- керування власними замовленнями;
- керування переліками виробів ручної роботи.

Продавець (рис. 3.3) може керувати виробами ручної роботи магазину, може керувати замовленнями щодо придбання виробів з магазину, за який він відповідає.

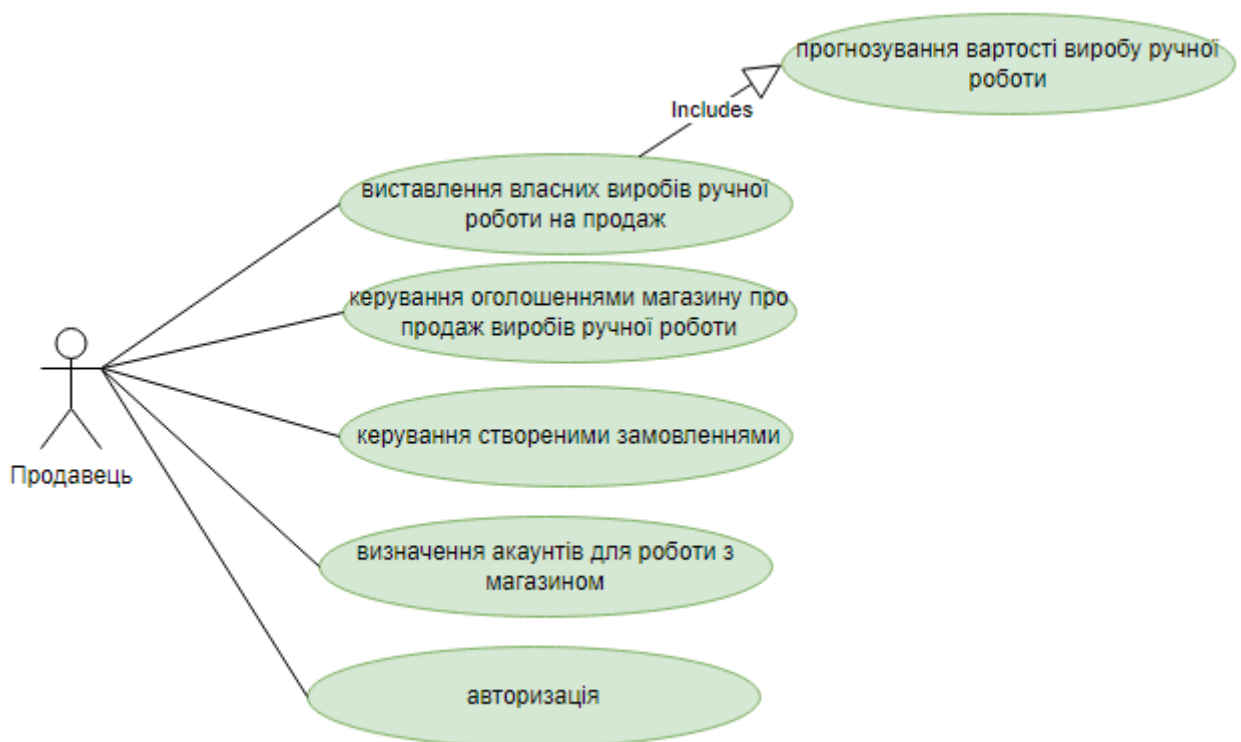


Рисунок 3.3 – Діаграма прецедентів продавця

Основний потік:

- виставлення власних виробів ручної роботи на продаж;
- керування оголошеннями про продаж власних виробів ручної роботи;

- керування створеними замовленнями про придбання виробів ручної роботи в магазині;

- визначення акаунтів для роботи з магазином;

- авторизація.

Адміністратор (рис. 3.4) повинен керувати вищим рівнем роботи, тому йому надається перелік наступних сценаріїв:

- створення магазинів;

- керування магазинами;

- створення акаунтів для роботи з магазином;

- авторизація.

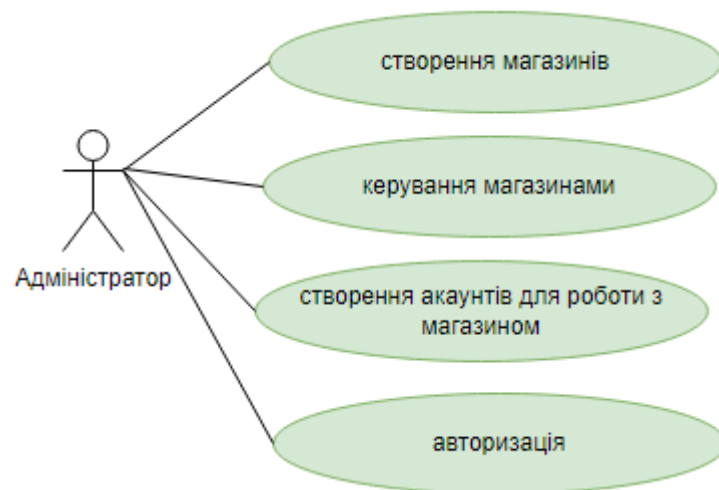


Рисунок 3.4 – Діаграма прецедентів адміністратора

3.3 Проєктування бази даних

Під час проєктування БД було використано сутності предметної області, визначені в розділі 1.

Таблиця Category містить наступні поля:

- id – номер категорії;

- name – назва категорії.

Таблиця Subcategory містить наступні поля:

- id – номер підкатегорії;

- name – назва підкатегорії;
- category – категорія, до якої належить дана підкатегорія.

Таблиці Category і Subcategory зв'язані зв'язком один-до-багатьох: кожна категорія повинна містити набір підкатегорій, а кожна підкатегорія може належати тільки одній категорії.

Таблиці Handmade і Subcategory зв'язані зв'язком багато-до-багатьох: кожна підкатегорія може мати багато виробів ручної роботи (задля цього вони і створюються), при цьому і деякі вироби ручної роботи можуть належати до різних підкатегорій. Реалізовано даний зв'язок створенням зв'язної таблиці HandmadeSubcategory.

Таблиця HandmadeSubcategory містить наступні поля:

- id – номер запису про підкатегорію, до якої віднесено виріб ручної роботи;
- subcategory – підкатегорія, до якої належить виріб;
- handmade – виріб ручної роботи.

Таблиця ColorOption містить наступні поля:

- id – номер кольору;
- name – назва кольору.

Таблиця List містить наступні поля:

- id – номер переліку;
- name – назва переліку;
- buyer – покупець, який створив цей перелік.

Таблиці Buyer і List зв'язані зв'язком один-до-багатьох: кожен покупець може мати багато переліків для роботи з виробами ручної роботи, проте створений кожен перелік може бути тільки одним користувачем.

Таблиці Handmade і List зв'язані зв'язком багато-до-багатьох: кожен виріб ручної роботи може входити в різні переліки різних користувачів, до того ж кожен перелік може містити різні вироби ручної роботи. Реалізовано даний зв'язок створенням зв'язної таблиці HandmadeList.

Таблиця HandmadeList містить наступні поля:

- id – номер запису про входження виробу ручної роботи в перелік;
- list – перелік, в який внесено виріб ручної роботи;
- handmade – виріб ручної роботи.

Таблиця Handmade містить наступні поля:

- id – номер виробу ручної роботи;
- title – назва виробу;
- description – опис виробу;
- coloroption – опція, яка визначає, чи можна стосовно даного виробу обрати колір;
- personoption – опція, яка визначає, чи можна стосовно даного виробу задати варіант спеціалізації (зробити відповідний напис);
- reviews – кількість відгуків;
- reviewscore – середня оцінка за відгуками покупців;
- likes – кількість уподобань виробу покупцями;
- orders – кількість замовлень, в яких було придбано виріб;
- state – стан виробу ручної роботи: необхідний для зняття виробів з продажу;
- forecast – прогнозована вартість виробу ручної роботи.

Таблиця HandmadeLike містить наступні поля:

- id – номер уподобання покупцем виробу;
- handmade – виріб ручної роботи;
- buyer – покупець.

Таблиці Buyer і HandmadeLike зв'язані зв'язком один-до-багатьох: кожен покупець може уподобати різні вироби ручної роботи, але звичайно, що кожне уподобання може відповідати тільки одному користувачу.

Таблиці Handmade і HandmadeLike зв'язані зв'язком один-до-багатьох: кожен виріб ручної роботи може мати багато уподобань, при цьому кожне уподобання відноситься до одного конкретного виробу.

Таблиця Buyer містить наступні поля:

- id – номер покупця;

- username – логін покупця;
- password – пароль покупця;
- fullname – повне ім'я покупця;
- phone – номер покупця;
- state – стан облікового запису (необхідний для заборони доступу за необхідності).

Таблиця Order містить наступні поля:

- id – номер замовлення;
- buyer – покупець, який оформив замовлення;
- totalprice – загальна вартість замовлення;
- orderdate – дата оформлення замовлення;
- note – нотатка від покупця щодо замовлення;
- state – стан замовлення.

Таблиці Buyer і Order зв'язані зв'язком один-до-багатьох: кожен покупець може мати багато замовлень, але кожне замовлення може бути сформовано тільки одним покупцем.

Таблиця ShopReview містить наступні поля:

- id – номер відгуку на магазин з продажу виробів ручної роботи;
- buyer – покупець, який створив відгук;
- shop – магазин виробів ручної роботи, про який створено цей відгук;
- score – оцінка відгуку;
- reviewtext – текст відгуку;
- reviewdate – час створення відгуку.

Таблиці Buyer і ShopReview зв'язані зв'язком один-до-багатьох: кожен покупець може мати багато відгуків про магазини, проте звичайно, що автором кожного відгуку є тільки один покупець.

Таблиці Shop і ShopReview зв'язані зв'язком один-до-багатьох: кожен магазин може мати багато відгуків про свою роботу, проте звичайно, що кожен відгук створений тільки одним покупцем.

Таблиця Shop містить наступні поля:

- id – номер магазину;
- name – назва магазину;
- description – опис магазину;
- reviewscore – загальна оцінка всіх відгуків про цей магазин;
- reviews – кількість написаних відгуків;
- orders – кількість замовлень, які були оформлені на вироби цього магазину;
- state – стан магазину (необхідний для заборони роботи за необхідності).

Таблиця ShopAccount містить наступні поля:

- id – номер акаунту, який представляє магазин виробів ручної роботи;
- shop – магазин виробів ручної роботи, до якого належить цей акаунт;
- username – логін користувача;
- password – пароль користувача;
- state – стан облікового запису (необхідний для заборони доступу за необхідності).

Таблиці Shop і ShopAccount зв'язані зв'язком один-до-багатьох: кожен магазин може мати багато користувачів, які підтримують роботу з його замовленнями, проте кожен такий користувач може бути відповідальним тільки за один магазин і не має можливості працювати з різними магазинами під одним акаунтом.

Таблиця OrderHandmade визначає склад замовлення і містить наступні поля:

- id – номер запису про включення виробу ручної роботи до замовлення;
- handmade – виріб ручної роботи;
- order – замовлення, в межах якого оформлено даний виріб;
- number – кількість виробів даного найменування;

- `handmadecoloroption` – обраний колір за наявності такої можливості;
- `personoption` – персональна інформація, яка має бути внесена в виріб за наявності такої можливості;
- `unitprice` – вартість одиниці;
- `totalprice` – вартість всієї позиції за цим найменуванням виробів у замовленні.

Таблиця `OrderHandmade` по суті реалізує зв'язок між таблицями `Order` і `Handmade`, оскільки кожне замовлення може мати багато виробів ручної роботи, але і кожне найменування виробу ручної роботи може міститися в різних замовленнях.

Таблиці `HandmadeColorOption` і `OrderHandmade` зв'язані зв'язком один-до-багатьох: кожен колір конкретного виробу ручної роботи може бути представлений і різних замовленнях.

Таблиця `HandmadeReview` містить наступні поля:

- `id` – номер відгуку про виріб ручної роботи;
- `buyer` – покупець, який написав відгук;
- `handmade` – виріб ручної роботи;
- `score` – оцінка у відгуку виробу ручної роботи;
- `reviewtext` – текст відгуку;
- `reviewdate` – час створення відгуку.

Таблиці `Buyer` і `HandmadeReview` зв'язані зв'язком один-до-багатьох: кожен покупець може мати багато відгуків про вироби ручної роботи, але кожен відгук створений тільки одним користувачем.

Таблиці `Handmade` і `HandmadeReview` зв'язані зв'язком один-до-багатьох: кожен виріб ручної роботи може мати багато відгуків про себе, проте кожен відгук може бути оформлений тільки на один конкретний виріб ручної роботи і не може охоплювати одразу декілька виробів.

Схема БД подана на рис. 3.5.

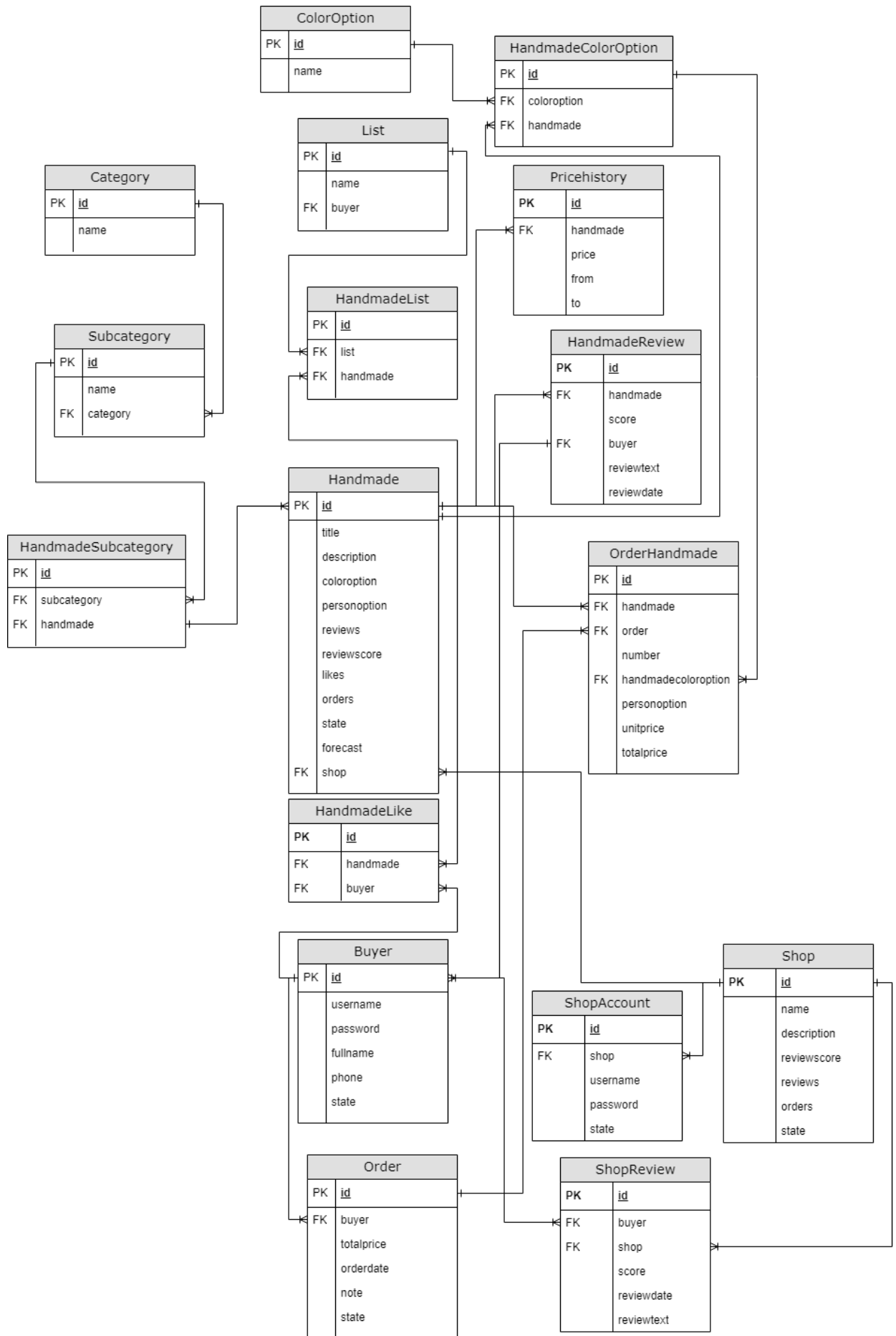


Рисунок 3.5 – Схема БД

Таблиця PriceHistory містить наступні поля:

- id – номер запису про вартість виробу ручної роботи;
- handmade – виріб ручної роботи;
- price – вартість;
- from – дата, з якої встановлено цю вартість;
- to – дата, до якої встановлено цю вартість.

Таблиці Handmade і PriceHistory зв'язані зв'язком один-до-багатьох: кожен виріб ручної роботи може мати багато записів про вартість його у певний момент часу, проте кожен запис характеризує тільки один виріб.

Таблиці Handmade і ColorOption зв'язані зв'язком багато-до-багатьох: один виріб ручної роботи може мати багато різних опцій вибору кольору (хоча може і не мати жодної), при цьому і кожна опція може бути представлена в різних виробих ручної роботи. Реалізовано даний зв'язок створенням зв'язної таблиці HandmadeColorOption.

Таблиця HandmadeColorOption містить наступні поля:

- id – номер запису про доступність кольору для конкретного виробу ручної роботи;
- coloroption – обраний колір виробу;
- handmade – виріб ручної роботи.

3.4 Структура програми

Програмний проєкт створюється у вигляді Django-проєкту, тобто проєкту, створеного на основі використання вебфреймворку Django (рис. 3.6).

До складу Django-проєкту входить сам вебзастосунок, який реалізує основну функціональність, визначену предметною областю, та підсистема визначення вартості виробів ручної роботи, яка реалізує і дії на основі методу, описаного в розділі 2.

Вебзастосунок у момент створення оголошення про виріб ручної роботи звертається до підсистеми визначення вартості виробів ручної роботи, щоб таку вартість встановити, а отримане значення передає через інтерфейс БД до самої БД, щоб зберегти для витягання під час подальших звернень.

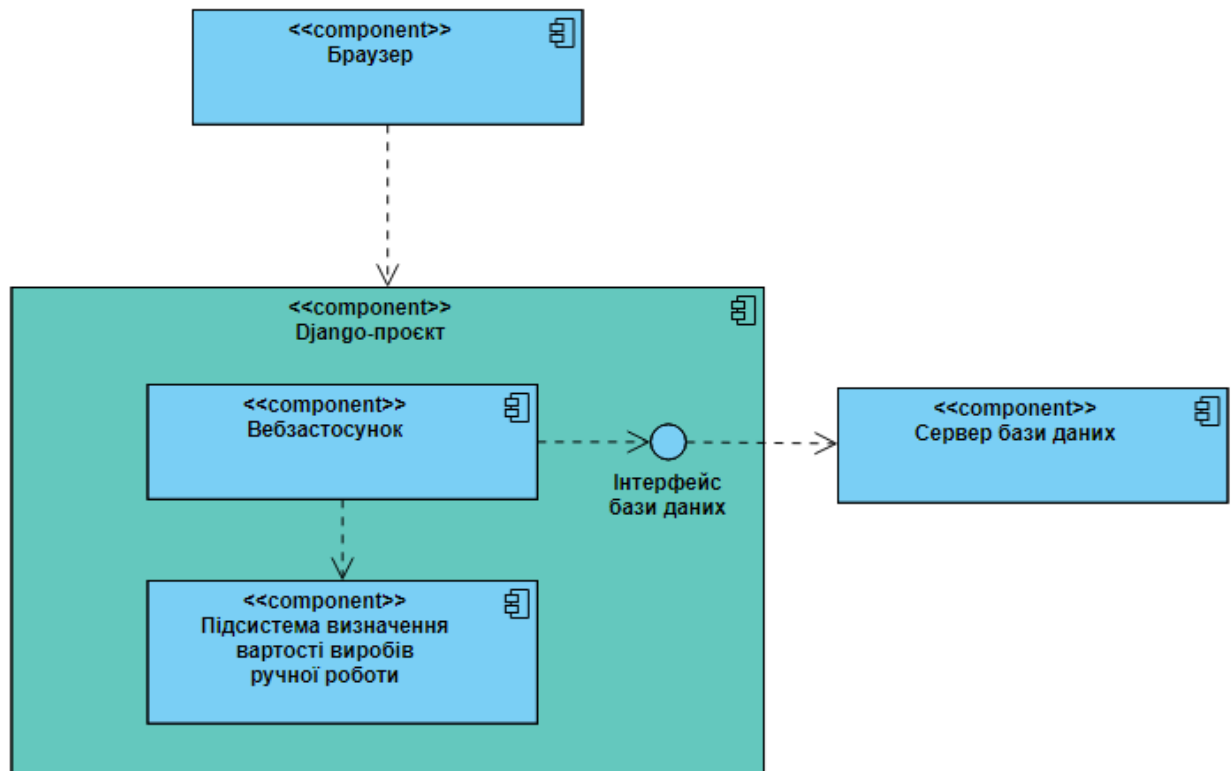


Рисунок 3.6 – Схема структури програми

У свою чергу користувач взаємодіє з програмою через браузер, який запускається на його клієнтському обладнанні. В залежності від способу звернення обирається або адміністративна панель для підтримки роботи адміністратора на основі вбудованої адміністративної панелі в Django, або звичайний інтерфейс вебзастосунку.

3.5 Висновки за розділом 3

У даному розділі виконано порівняння мов програмування і визначено мову Python для реалізації програми. Визначено основні сценарії роботи

користувачів з програмою зі створенням діаграм прецедентів у результаті. Визначено структуру БД. Визначено структурну схему програмної системи, у якій окремо виділено функціональність стосовно розробленого методу в окрему підсистему.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Алгоритм функціонування програми

На початку роботи користувачу відображається перелік виробів ручної роботи. Користувач може обрати конкретний варіант формування переліку виробів ручної роботи, включаючи можливість вибору підкатегорії виробів, рейтинг виробів. Тоді користувач може або продовжувати працювати з цією функцією, або перейти до пошуку виробів ручної роботи, або виконати авторизацію.

Неавторизованому користувачу надається два варіанти авторизації: авторизація продавця та авторизація покупця.

Якщо користувач авторизувався як продавець, то він може обрати один з варіантів роботи з меню, що включають керування каталогом власних виробів ручної роботи, керування замовленнями покупців та керування засобами магазину. У будь-який момент продавець може вийти з акаунту та повернутися до початкового меню. Також користувач може завершити роботу з програмою.

Якщо користувач авторизувався як покупець, то він може також, як і на початку, виконати перегляд виробів ручної роботи, може виконати пошук виробів ручної роботи, може керувати власними замовленнями та керувати корзиною поточного замовлення, переглядати власні переліки виробів ручної роботи, переглядати власні відгуки, включаючи відгуки про продавців або виробу ручної роботи, що можна обрати з підменю. У будь-який момент покупець може вийти з акаунту та повернутися до початкового меню. Також користувач може завершити роботу з програмою.

Відповідна описаному алгоритму схема функціонування програми представлена на рис. 4.1.

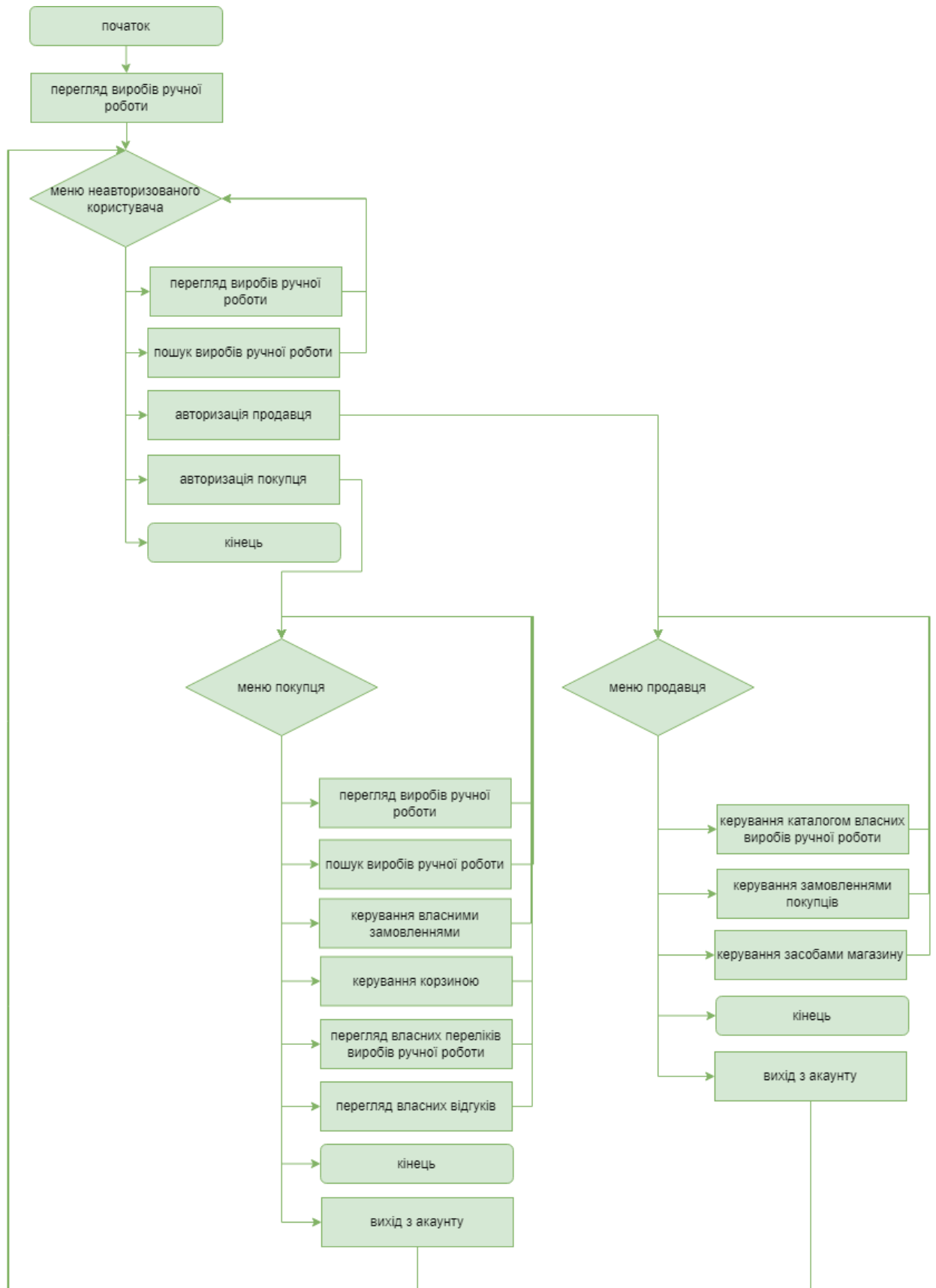


Рисунок 4.1 – Схема функціонування програми

4.2 Опис основних розроблених класів

Клас моделі `Category` містить окрім автоматично створюваного атрибута номера категорії також атрибут `name`, що є текстовим полем `CharField` з максимальною довжиною у 100 символів.

Клас моделі `Subcategory` містить також наступні атрибути:

- `name` – назва підкатегорії, що є текстовим полем `CharField` зі 100 символів;
- `category` – категорія, до якої належить дана підкатегорія, що є зовнішнім ключем з моделі `Category` типу `models.ForeignKey (Category, on_delete=models.CASCADE)`.

Клас моделі `ColorOption` містить окрім автоматично створюваного атрибута номера кольору також атрибут `name`, що є текстовим полем `CharField` з максимальною довжиною у 100 символів.

Клас моделі `List` містить окрім автоматично створюваного атрибута номера переліку також наступні атрибути:

- `name` – назва переліку, що є текстовим полем `CharField` з 200 символів;
- `buyer` – покупець, який створив цей перелік, що є зовнішнім ключем з моделі `Buyer` типу `models.ForeignKey (Buyer, on_delete=models.CASCADE)`;
- `handmades` – виробу ручної роботи з даного переліку, визначені шляхом встановлення зв'язку багато-до-багатьох на основі поля `models.ManyToManyField(Handmade)`.

Також дана модель надає наступні методи:

- `__str__(self)`, що повертає текстове представлення переліку, яке є його назвою;
- `get_absolute_url(self)`, що повертає повний шлях до сторінки переліку виробів, збережених користувачем.

Таблиця `HandmadeList` містить наступні поля:

- `list` – перелік, в який внесено виріб ручної роботи;

– handmade – виріб ручної роботи.

Клас моделі Handmade містить окрім автоматично створюваного атрибута номера виробу ручної роботи також наступні атрибути:

- title – назва виробу, що є текстовим полем CharField зі 100 символів;
- description – опис виробу, що є текстовим полем TextField;
- coloroption – опція, яка визначає, чи можна стосовно даного виробу обрати колір, що є логічним полем BooleanField;
- personoption – опція, яка визначає, чи можна стосовно даного виробу задати варіант спеціалізації, що є логічним полем BooleanField;
- reviews – кількість відгуків, що є цілочисельним полем IntegerField;
- reviewscore – середня оцінка за відгуками покупців, що є полем дійсного типу FloatField;
- likes – кількість уподобань виробу покупцями, що є цілочисельним полем IntegerField;
- orders – кількість замовлень, в яких було придбано виріб, що є цілочисельним полем IntegerField;
- state – стан виробу ручної роботи, що є цілочисельним полем SmallIntegerField.

Також дана модель надає наступні методи:

- __str__(self), що повертає текстове представлення виробу, яке є його назвою;
- get_absolute_url(self), що повертає повний шлях до сторінки виробу ручної роботи;
- get_hmstate(self), що повертає текстовий опис стану виробу.

Клас моделі HandmadeLike містить окрім автоматично створюваного атрибута номера уподобання виробу ручної роботи також наступні атрибути:

- handmade – виріб ручної роботи, що є зовнішнім ключем з моделі Handmade типу models.ForeignKey (Handmade, on_delete=models.CASCADE);
- buyer – покупець, що є зовнішнім ключем з моделі Buyer типу models.ForeignKey (Buyer, on_delete=models.CASCADE).

Клас моделі Buyer організований наслідуванням класу AbstractUser і містить окрім його атрибутів ще наступні:

- fullname – повне ім'я покупця, що є текстовим полем CharField з 200 символів;
- phone – номер покупця, що є текстовим полем CharField з 13 символів;
- state – стан облікового запису, що є цілочисельним полем SmallIntegerField.

Клас моделі Order містить окрім автоматично створюваного атрибута номера виробу ручної роботи також наступні атрибути:

- buyer – покупець, що є зовнішнім ключем з моделі Buyer типу models.ForeignKey (Buyer, on_delete=models.CASCADE);
- totalprice – загальна вартість замовлення, що є полем дійсного типу FloatField;
- orderdate – дата оформлення замовлення, що є полем дати і часу DateTimeField;
- note – нотатка від покупця щодо замовлення, що є текстовим полем CharField з 255 символами;
- state – стан замовлення, що є цілочисельним полем SmallIntegerField.

Також дана модель надає наступні методи:

- __str__(self), що повертає текстове представлення замовлення, яке складається з його номеру та сумарної вартості замовлення;
- get_absolute_url(self), що повертає повний шлях до сторінки замовлення;
- get_orderstate (self), що повертає текстовий опис стану замовлення.

Клас моделі ShopReview містить окрім автоматично створюваного атрибута номера відгуку про магазин також наступні атрибути:

- buyer – покупець, який створив відгук, що є зовнішнім ключем з моделі Buyer типу models.ForeignKey (Buyer, on_delete=models.CASCADE);

- shop – магазин виробів ручної роботи, про який створено цей відгук, що є зовнішнім ключем з моделі Shop типу `models.ForeignKey (Shop, on_delete=models.CASCADE)`;

- score – оцінка відгуку, що є цілочисельним полем типу `IntegerField`;

- reviewtext – текст відгуку, що є текстовим полем `TextField`;

- reviewdate – час створення відгуку, що є полем дати і часу `DateTimeField`.

Також дана модель надає метод `get_absolute_url(self)`, що повертає повний шлях до сторінки відгуку про магазин.

Клас моделі Shop містить окрім автоматично створюваного атрибута номера магазину також наступні атрибути:

- name – назва магазину, що є текстовим полем `CharField`, що складається з 255 символів;

- description – опис магазину, що є текстовим полем `TextField`;

- reviewscore – загальна оцінка всіх відгуків про цей магазин, що є полем дійсного типу `FloatField`;

- reviews – кількість написаних відгуків, що є цілочисельним полем `IntegerField`;

- orders – кількість замовлень, які були оформлені на вироби цього магазину, що є цілочисельним полем `IntegerField`;

- state – стан магазину, що є цілочисельним полем `SmallIntegerField`.

Також дана модель надає метод `get_absolute_url(self)`, що повертає повний шлях до сторінки магазину.

Клас моделі ShopAccount містить окрім автоматично створюваного атрибута номера акаунта представника магазину також наступні атрибути:

- shop – магазин виробів ручної роботи, до якого належить цей акаунт, що є зовнішнім ключем з моделі Shop типу `models.ForeignKey (Shop, on_delete=models.CASCADE)`.

Клас моделі `OrderHandmade` містить окрім автоматично створюваного атрибута номера виробу ручної роботи в замовленні також наступні атрибути:

- `handmade` – виріб ручної роботи, що є зовнішнім ключем з моделі `Handmade` типу `models.ForeignKey (Handmade, on_delete=models.CASCADE)`;
- `order` – замовлення, в межах якого оформлено даний виріб, що є зовнішнім ключем з моделі `Order` типу `models.ForeignKey (Order, on_delete=models.CASCADE)`;
- `number` – кількість виробів даного найменування, що є цілочисельним полем `IntegerField`;
- `handmadecoloroption` – обраний колір за наявності такої можливості, що є зовнішнім ключем з моделі `HandmadeColorOption` типу `models.ForeignKey (HandmadeColorOption, on_delete=models.CASCADE)`;
- `personoption` – персональна інформація, яка має бути внесена в виріб за наявності такої можливості, що є текстовим полем `CharField`, що складається з 255 символів;
- `unitprice` – вартість одиниці, що є полем дійсного типу `FloatField`;
- `totalprice` – вартість всієї позиції за цим найменуванням виробів у замовленні, що є полем дійсного типу `FloatField`.

Клас моделі `HandmadeReview` містить окрім автоматично створюваного атрибута номера відгуку про виріб ручної роботи також наступні атрибути:

- `buyer` – покупець, який написав відгук, що є зовнішнім ключем з моделі `Buyer` типу `models.ForeignKey (Buyer, on_delete=models.CASCADE)`;
- `handmade` – виріб ручної роботи, що є зовнішнім ключем з моделі `Handmade` типу `models.ForeignKey (Handmade, on_delete=models.CASCADE)`;
- `score` – оцінка у відгуку виробу ручної роботи, що є цілочисельним полем `IntegerField`;
- `reviewtext` – текст відгуку, що є текстовим полем `TextField`;

– reviewdate – час створення відгуку, що є полем дати і часу DateTimeField.

Клас моделі PriceHistory містить окрім автоматично створюваного атрибута номера вартості також наступні атрибути:

– handmade – виріб ручної роботи, що є зовнішнім ключем з моделі Handmade типу models.ForeignKey (Handmade, on_delete=models.CASCADE);

– price – вартість, що є полем дійсного типу FloatField;

– from – дата, з якої встановлено цю вартість, що є полем дати і часу DateTimeField;

– to – дата, до якої встановлено цю вартість, що є полем дати і часу DateTimeField.

Шаблони програмного застосунку створено на основі використання вільного шаблону [16].

Представлення реалізовано у вигляді відповідних класів, які під основні існуючі сутності визначають класи для деталізації цих даних, створення нових об'єктів, редагування існуючих та вибірки даних у переліки. Розглянемо на прикладі сутності Handmade, що працює з однойменною моделлю, даний процес.

Клас HandmadeView реалізує виведення повної інформації за виробом ручної роботи з номером, переданим до методу get_context_data(self, **kwargs) через словник параметрів.

Клас HandMadeNewView реалізує підтримку роботи з формою створення нового виробу ручної роботи. Метод form_valid (self, form) забезпечує внесення даних за базовими значеннями кількості уподобань, стану, кількості відгуків і замовлень, роблячи їх нульовими, а в якості магазину, відповідального за даний виріб встановлює магазин, якому належить акаунт, авторизований в даний момент.

Клас HandmadeUpdateView реалізує оновлення виробу ручної роботи з перенесенням даних в модель з відповідної форми.

Клас `ShopHandmadesView` реалізує заповнення переліку виробів ручної роботи, які належать магазину з номером, переданим через GET-запит, забезпечуючи при цьому сортування шляхом, визначеним другим параметром усередині методу `get_queryset(self, *args, **kwargs)`.

Клас `ShopAccountHandmadesView` реалізує заповнення переліку виробів ручної роботи, що внесені і продаються магазином, до якого належить авторизований у поточний момент користувач, тобто за акаунтом магазину.

Клас `SubcategoryHandmadesView` реалізує заповнення переліку виробів ручної роботи за заданою через GET-запит підкатегорією виробів ручної роботи.

Клас `HandmadeLikesView` реалізує заповнення переліку уподобань виробу ручної роботи за заданим через GET-запит номером.

Клас `SearchHandmadeView` реалізує заповнення переліку виробів ручної роботи за заданою через GET-запит частиною або повною назвою виробу.

Клас `BestLikeHandmadesView` реалізує заповнення переліку виробів ручної роботи першими 100 за рейтингом виробами відносно показника кількості уподобань.

Клас `BestOrderHandmadesView` реалізує заповнення переліку виробів ручної роботи першими 100 за рейтингом виробами відносно показника кількості замовлень.

Клас `BestReviewsHandmadesView` реалізує заповнення переліку виробів ручної роботи першими 100 за рейтингом виробами відносно показника кількості відгуків.

Клас `BestReviewScoreHandmadesView` реалізує заповнення переліку виробів ручної роботи першими 100 за рейтингом виробами відносно показника середньої оцінки виробу.

4.3 Висновки за розділом 4

Представлено основні розроблені класи моделей, представлень і шаблонів програми, приведено алгоритм функціонування програми з наданням відповідної схеми функціонування.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Програма призначена для авторів виробів ручної роботи, що виступають у ролі продавців, для реалізації їх діяльності щодо внесення даних про свої вироби на продаж, для керування цими даними і створеними на основі цих виробів замовленнями.

Окрім того програма призначена для покупців щодо налагодження для них пошуку виробів ручної роботи, оформлення замовлень щодо купівлі цих виробів.

5.2 Умови виконання програми

При запуску ПЗ продажу виробів ручної роботи потрібно враховувати, що використовувані засоби напряду залежать від того, з якою кількістю користувачів буде працювати програма. Відповідно в залежності від цього встановлені вимоги можуть збільшуватися.

Базовими вимогами визначено:

- чотириядерний процесор з частотою від 2 ГГц;
- обсяг оперативної пам'яті від 8 Гб;
- 2 Гб вільного простору на жорсткому диску.

Програма також потребує встановлення на сервері системи керування БД MySQL.

5.3 Робота з програмою

З програмою можна працювати загалом в 4 режимах.

Перший режим призначений для неавторизованого користувача, він

дозволяє виконати авторизацію, при чому можна перемкнутися між авторизацією продавця і покупця.

В другому режимі покупець може виконувати пошук виробів і працювати над замовленнями на основі меню, що реалізує логіку, описану в алгоритмів роботи програми.

Зокрема на рис. 5.1 представлено перегляд підкатегорій категорії «Декор». Якщо натиснути на одну з фотографій, то відкриється сторінка перегляду виробів за цією підкатегорією. При наведенні на кожну фотографію виводиться назва підкатегорії. Аналогічно виконується вибір категорій.

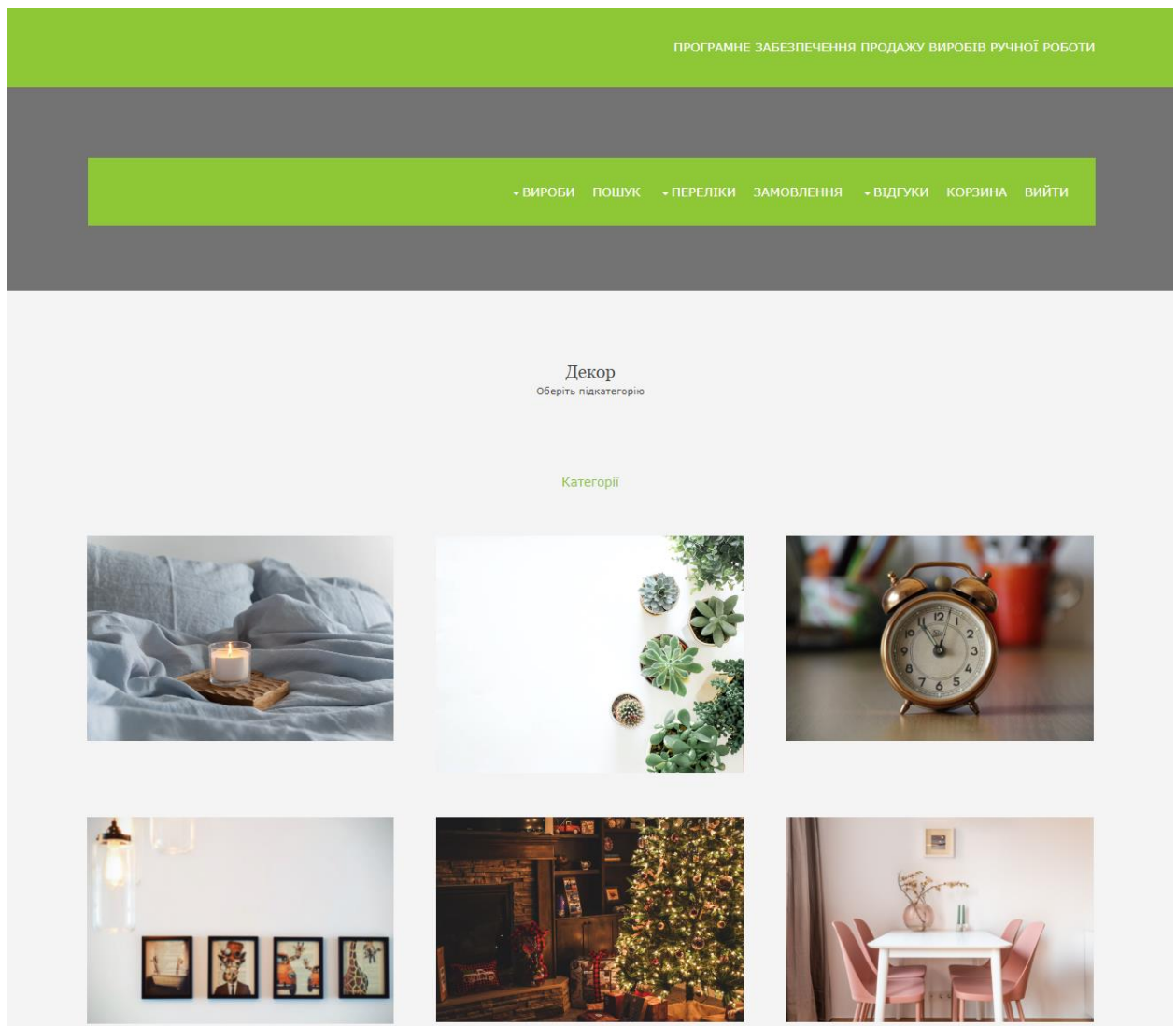


Рисунок 5.1 – Вибір підкатегорій в категорії виробів ручної роботи

В третьому режимі працює продавець. Його логіка роботи також обумовлена алгоритмом функціонування програми. Під час керування замовленнями продавець може обрати необхідний йому стан замовлення, після чого нижче відображаються окремі замовлення (рис. 5.2). За кожним замовленням продавець бачить автора замовлення, інформацію про час, номер та вартість замовлення. Щоб переглянути склад замовлення, потрібно перейти на сторінку замовлення, натискаючи на посилання «Переглянути». Якщо замовлення прийняте до виконання, то потрібно натиснути на кнопку «Підтвердити» (рис. 5.2).

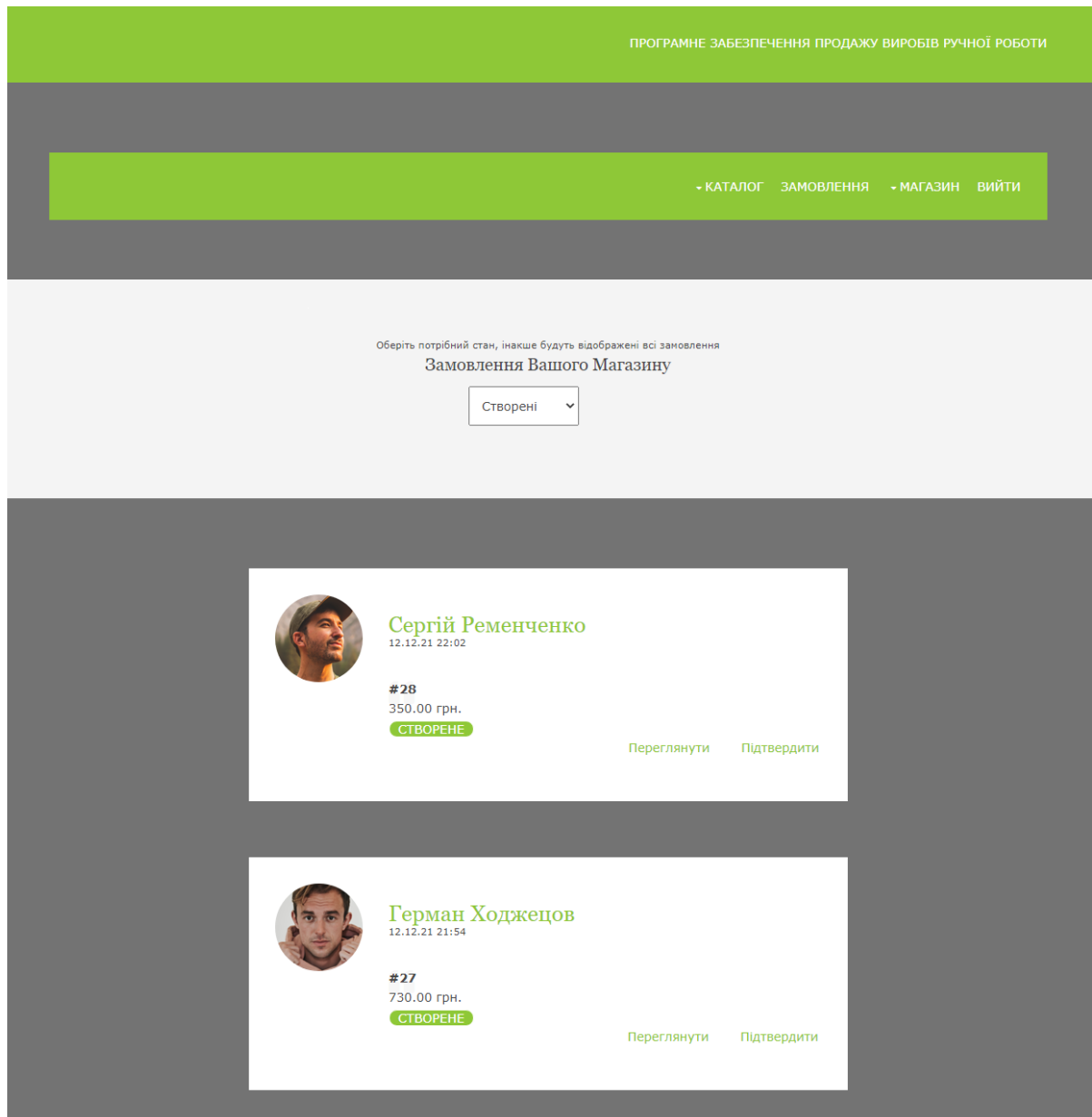


Рисунок 5.2 – Керування замовленнями магазину

Описані засоби прогнозування в інтерфейсі користувача доступні через перегляд власного каталогу (рис. 5.3). При виведенні каталогу власних виробів вказується назва виробу, категорія та підкатегорія, до якої він належить, вартість, за якою на поточний момент продається виріб, прогнозована вартість, оцінка, виставлена покупцями.

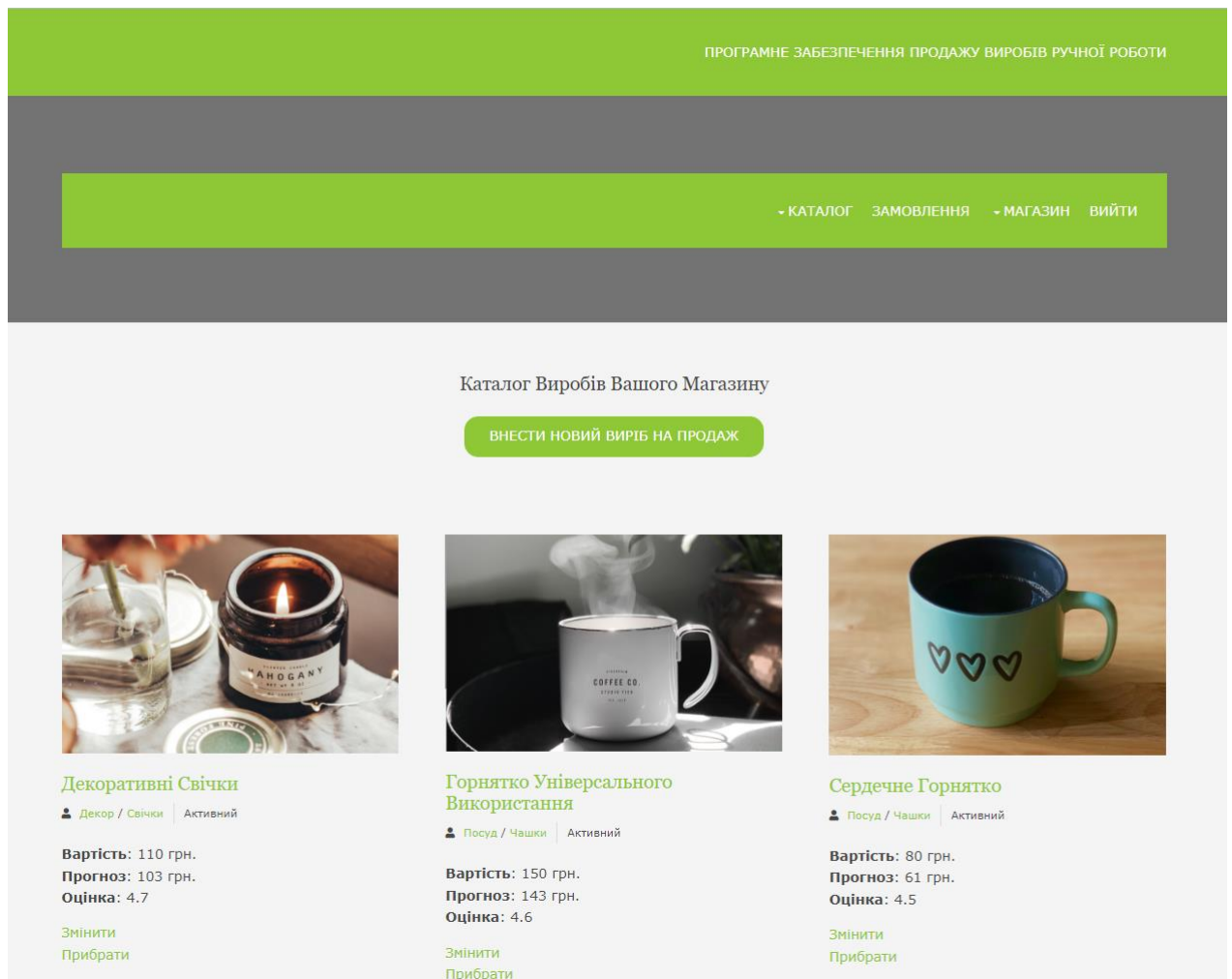


Рисунок 5.3 – Керування каталогом магазину

Якщо натиснути на посилання «Змінити», то відкриється сторінка, де можна змінити дані виробу, а також змінити його стан. Звідти можна переглянути і історію зміни вартості виробу, і відгуки користувачів. Відгуки користувачів про магазин доступні через підпункти пункту меню «Магазин».

Четвертий режим призначений для адміністратора і забезпечує створення магазинів. Він реалізується через адміністративну панель Django, а

для запуску потрібно додати «/admin» до адреси вебсервера.

5.4 Експериментальне дослідження

Експериментальне дослідження було проведено на основі вибірки даних [17], що характеризує вироби ручної роботи, включаючи їх опис, фактичну вартість, середню оцінку, виставлену користувачами. Дослідження проводилось шляхом порівняння стандартних дерев рішень з деревами рішень на основі удосконаленого алгоритма навчання на основі бібліотеки LightGBM, що лежить в основі представленого метода (табл. 5.1).

Таблиця 5.1 – Результати експериментального дослідження

Метод	Похибка
Дерево рішень	0,295
Метод, представлений у роботі	0,238

Результати дослідження продемонстрували зменшення похибки прогнозування на 0,057. Прогнозування в даному випадку виконувалось на основі нормалізованої вартості, щоб уникнути коливань вартостей, адже різні вироби можуть значно відрізнятися в ціні.

5.5 Висновки за розділом 5

Описано основні режими роботи користувача з програмою з приведенням відповідних інтерфейсів. Проведено експериментальне дослідження проблеми, за результатами якого представлений метод продемонстрував свою практичну придатність.

ВИСНОВКИ

Виконано аналіз предметної області. Розглянуто і проаналізовано можливості основних програмних засобів у цій області, найвідомішим з яких є Etsy, а також ArtFire. Основним прототипом програми, що розроблялась, було обрано Etsy, порівняно з нею розробка має функціональну особливість – вона надає можливість автоматичного визначення вартості виробів. Також було виконано виділення основних сутностей предметної області, необхідних для забезпечення роботи з функціями, що надають основний масив засобів серед програмних аналогів.

Проведено аналіз технік, які використовуються для роботи з текстом з метою переведення його до форми вектора. Розглянуто принципи створення дерев рішень. Запропоновано метод визначення вартості виробів ручної роботи, який ґрунтується на обробленні тексту за допомогою моделі мішків слів з подальшим застосуванням дерев рішень, що дозволяє приймати рішення, враховуючи не тільки витрати на створення виробу, але і очікувану вартість в умовах відповідної платформи.

Виконано порівняння мов програмування і визначено мову Python для реалізації програми. Визначено основні сценарії роботи користувачів з програмою зі створенням діаграм прецедентів у результаті. Визначено структуру БД. Визначено структурну схему програмної системи, у якій окремо виділено функціональність стосовно розробленого методу в окрему підсистему.

Реалізовано програму. Розроблена програма дозволяє продавцям створювати власні магазини, об'єднуючи таким чином вироби ручної роботи, які вони створюють і бажають продавати, керувати замовленнями покупців, відповідно покупцям дозволяє організовувати пошук виробів ручної роботи, замовляти їх, надає інструментар відгуків для створення зворотного зв'язку про вироби.

Усі поставлені завдання на роботу виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Deloe, J. Best Places to Sell Handmade Products [Electronic resource] / J. Deloe. – Access mode : <https://www.ecomdash.com/10-places-to-sell-handmade-items-online/>.
2. Etsy Ukraine – Shop for handmade, vintage, custom, and unique gifts for everyone [Electronic resource]. – Access mode : <https://www.etsy.com/>.
3. Why, Where, and How to Sell Crafts and Handmade Items Online [Electronic resource]. – Access mode : <https://www.skuvault.com/blog/blog-why-where-and-how-to-sell-crafts-and-handmade-items-online/>.
4. ArtFire.com – Premier handmade marketplace to buy & sell handmade crafts, supplies, vintage and art [Electronic resource]. – Access mode : <https://www.artfire.com/>.
5. Bag-of-Words Technique in Natural Language Processing [Electronic resource] : A Primer for Radiologists / Krishna Juluru, Hao-Hsin Shih, Krishna Nand Keshava Murthy, Pierre Elnajjar // RadioGraphics. – 2021. – Vol. 41. – No. 5.
6. Kedia, P. Bag-of-Words models in NLP [Electronic resource] / P. Kedia. – Access mode : <https://medium.com/mllearning-ai/bag-of-words-models-in-nlp-565e631dbd7f>.
7. Introduction to Natural Language Processing [Electronic resource]. – Access mode : <https://openclassrooms.com/en/courses/6532301-introduction-to-natural-language-processing/6980811-apply-a-simple-bag-of-words-approach>.
8. Nair, A. TF-IDF Demystified [Electronic resource] / A. Nair. – Access mode : <https://towardsdatascience.com/tf-idf-demystified-7dd3ef071b24>.
9. Trotter, M. Decision Trees in Machine Learning Explained [Electronic resource] / M. Trotter. – Access mode : https://www.seldon.io/decision-trees-in-machine-learning/?utm_source=rss&utm_medium=rss&utm_campaign=decision-trees-in-machine-learning.
10. Features – LightGBM 3.3.1.99 documentation Processing [Electronic

resource]. – Access mode : <https://lightgbm.readthedocs.io/en/latest/Features.html>.

11. Ucar, K. T. An End-to-End Web Service Implementation for Text Classification using Word2Vec and LGBM [Electronic resource] / K. T. Ucar. – Access mode : <https://towardsdatascience.com/an-end-to-end-web-service-implementation-for-text-classification-using-word2vec-and-lgbm-3d605c77fd6e>.

12. An Introduction to Gradient Boosting Decision Trees [Electronic resource]. – Access mode : <https://www.machinelearningplus.com/machine-learning/an-introduction-to-gradient-boosting-decision-trees/>.

13. McKinney, W. Python for Data Analysis [Text] : Data Wrangling with Pandas, NumPy, and IPython / W. McKinney. – Sebastopol : O'Reilly Media, 2017. – 550 p.

14. Web Development with Django [Text] : Learn to build modern web applications with a Python-based framework / B. Shaw, S. Badfwar, A. Bird et al. – Birmingham : Packt Publishing, 2021. – 826 p.

15. Barry, P. Head First Python [Text] : A Brain-Friendly Guide / P. Barry. – Sebastopol, CA : O'Reilly Media, 2016. – 624 p.

16. Spourmo Free Website Template [Electronic resource]. – Access mode : <https://www.os-templates.com/free-website-templates/spourmo>.

17. Etsy items price [Electronic resource]. – Access mode : <https://www.kaggle.com/dimakyn/etsy-items-price>.

ДОДАТОК А
Технічне завдання

Вступ

ПЗ продажу виробів ручної роботи має підтримувати взаємодію продавців, які створюють вироби ручної роботи, та покупців. Для цього кожній зі сторін мають надаватися засоби для організації зручної роботи щодо виконання своїх функцій. Однією з таких функцій для продавця є автоматичне визначення вартості виробів ручної роботи. Для створення цієї функції має бути розроблено відповідний метод, який буде підтримувати її роботу.

A.1 Підстави для розробки

Підставою для розробки є завдання на кваліфікаційну роботу магістра, визначене за темою «Дослідження та програмна реалізація методів визначення вартості виробів ручної роботи», яку затверджено наказом № 435 від 5 листопада 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програма призначена для авторів виробів ручної роботи, що виступають у ролі продавців, для реалізації їх діяльності щодо внесення даних про свої вироби на продаж, для керування цими даними і створеними на основі цих виробів замовленнями.

Окрім того програма призначена для покупців щодо налагодження для них пошуку виробів ручної роботи, оформлення замовлень щодо купівлі цих виробів.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати користувачам виконання таких функцій:

- виставлення власних виробів ручної роботи на продаж;
- керування оголошеннями про продаж власних виробів ручної роботи;
- створення замовлень про придбання виробів ручної роботи;
- керування створеними замовленнями про придбання виробів ручної роботи в магазині;
- пошук виробів ручної роботи;
- публікація відгуків про виріб ручної роботи;
- визначення середньої оцінки виробу ручної роботи;
- публікація відгуків про магазин;
- визначення середньої оцінки магазину;
- уподобання виробу ручної роботи;
- створення власних переліків з виробами ручної роботи;
- керування і перегляд переліків ручної роботи;
- перегляд уподобаних користувачем виробів ручної роботи;
- збереження історії зміни вартості виробу ручної роботи;
- прогнозування вартості виробу ручної роботи;
- перегляд виробів ручної роботи за заданою підкатегорією;
- формування рейтингу виробів ручної роботи за середньою оцінкою, кількістю відгуків, кількістю уподобань, кількістю замовлень;
- перегляд і керування власними замовленнями виробів ручної роботи;
- створення магазину для продажу виробів ручної роботи;
- визначення облікових записів користувачів, які можуть працювати з замовленнями даного магазину;

- перегляд переліку виробів ручної роботи, що продаються магазином;
- авторизація облікових записів звичайного користувача (покупця) та представників магазину;
- реєстрація облікових записів покупців.

A.3.2 Вимоги до складу та параметрів технічних засобів

Технічні засоби для підтримки роботи вебсервера повинні включати наступний мінімальний набір:

- чотириядерний процесор з частотою від 2 ГГц;
- обсяг оперативної пам'яті від 8 Гб;
- 2 Гб вільного простору на жорсткому диску.

A.3.3 Умови експлуатації

Для експлуатації програмного забезпечення окрім безпосередньо наявності самої програмної системи потрібно також встановити на сервер систему керування базами даних MySQL.

A.3.4 Вимоги до транспортування та збереження

Збереження програми має реалізовуватися шляхом розташування її на виділеному для цього сервері.

A.3.5 Вимоги до програмної документації

До складу програмної документації входять технічне завдання, керівництво оператора, що вноситься до складу розділу 5 пояснювальної записки, текст програми.

А.4 Порядок контролю та приймання

Календарний план є основою для контролю виконання роботи, приймання роботи здійснюється під час її захисту перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Б.1 Текст файла `models.py`

```
from django.contrib.auth.models import AbstractUser, UserManager
from django.db import models
from django.urls import reverse

class Category (models.Model):
    name = models.CharField(max_length=100)

class Subcategory (models.Model):
    name = models.CharField(max_length=100)
    category = models.ForeignKey (Category,
on_delete=models.CASCADE)

class ColorOption (models.Model):
    name = models.CharField(max_length=100)

class List (models.Model):
    name = models.CharField(max_length=200)
    buyer = models.ForeignKey (Buyer, on_delete=models.CASCADE)
    handmades = models.ManyToManyField(Handmade)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('list', args=[self.id])

class Handmade (models.Model):
```

```

title = models.CharField(max_length=100)
description = models.TextField()
coloroption = models.BooleanField()
personoption = models.BooleanField()
reviews = models.IntegerField()
orders = models.IntegerField()
likes = models.IntegerField()
reviewscore = models.FloatField()
forecast = models.FloatField()
state = models.SmallIntegerField()
coloroptions = models.ManyToManyField(ColorOption)
shop = models.ForeignKey(Shop, on_delete=models.CASCADE)
subcategories = models.ManyToManyField(Sucategory)

def __str__(self):
    return self.title

def get_absolute_url(self):
    return reverse('handmade', args=[self.id])

class Meta:
    ordering = ["-reviewscore"]

def get_hmstate(self):
    if self.state == 0:
        return 'активний'
    elif self.state == 1:
        return 'недоступний'
    elif self.state == 2:
        return 'тимчасово відсутній'

```

```

elif self.state == 3:
    return 'не виробляється'

```

```

class HandmadeLike (models.Model):
    buyer = models.ForeignKey (Buyer, on_delete=models.CASCADE)
    handmade = models.ForeignKey (Handmade,
on_delete=models.CASCADE)

```

```

class Buyer (AbstractUser):
    fullname = models.CharField(max_length=200)
    phone = models. CharField(max_length=13)
    state = models.SmallIntegerField()

```

```

class Order (models.Model):
    buyer = models.ForeignKey (Buyer, on_delete=models.CASCADE)
    totalprice = models.FloatField()
    orderdate = models.DateTimeField ()
    note = models. CharField(max_length=255)
    state = models.SmallIntegerField()

```

```

def __str__(self):
    return self.pk + " + self.totalprice

```

```

def get_absolute_url(self):
    return reverse('order', args=[self.id])

```

```

def get_orderstate(self):
    if self.state == 0:
        return 'створене'
    elif self.state == 1:

```

```

        return 'підтверджене'
    elif self.state == 2:
        return 'доставка'
    elif self.state == 3:
        return 'успішно завершено'
    elif self.state == 4:
        return 'завершено неуспішно'
    elif self.state == 5:
        return 'скасовано'

```

```

class ShopReview (models.Model):
    buyer = models.ForeignKey (Buyer, on_delete=models.CASCADE)
    shop = models.ForeignKey (Shop, on_delete=models.CASCADE)
    score = models.IntegerField()
    reviewtext = models.TextField()
    reviewdate = models.DateTimeField ()

    def get_absolute_url(self):
        return reverse('shopreview', args=[self.id])

    class Meta:
        ordering = ["-reviewdate "]

```

```

class Shop (models.Model):
    name = models.CharField(max_length=255)
    reviews = models.IntegerField()
    orders = models.IntegerField()
    reviewscore = models.FloatField()
    description = models.TextField()
    state = models.SmallIntegerField()

```

```
def get_absolute_url(self):
    return reverse('shop', args=[self.id])
```

```
class Meta:
    ordering = ["-reviewscore "]
```

```
class ShopAccount (AbstractUser):
    shop = models.ForeignKey (Shop, on_delete=models.CASCADE)
```

```
class OrderHandmade (models.Model):
    handmade = models.ForeignKey (Handmade,
on_delete=models.CASCADE)
    order = models.ForeignKey (Order, on_delete=models.CASCADE)
    handmadecoloroption = models.ForeignKey (HandmadeColorOption,
on_delete=models.CASCADE)
    number = models.IntegerField()
    personoption = models.CharField(max_length=255)
    unitprice = models.FloatField()
    totalprice = models.FloatField()
```

```
class HandmadeReview (models.Model):
    buyer = models.ForeignKey (Buyer, on_delete=models.CASCADE)
    handmade = models.ForeignKey (Handmade,
on_delete=models.CASCADE)
    score = models.IntegerField()
    reviewtext = models.TextField()
    reviewdate = models.DateTimeField ()
```

```
class Meta:
    ordering = ["-reviewdate "]
```

```

class PriceHistory (models.Model):
    handmade = models.ForeignKey (Handmade,
on_delete=models.CASCADE)
    price = models.FloatField()
    from = models. DateTimeField ()
    to = models.DateTimeField()

class Meta:
    ordering = ["-to "]

```

Б.2 Текст файла **handmade.py**

```

from django.views.generic.detail import DetailView
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import CreateView, UpdateView
import handmadeapp.models
import handmadeapp.funs

class HandmadeView (DetailView):

    model = Handmade
    template_name = 'handmade.html'

    def get_context_data(self, **kwargs):
        hms = super().get_context_data(**kwargs)
        hmobject = Handmade.objects.get(id=self.kwargs['pk'])
        hms ['details'] = hmobject
        hmreviews = HandmadeReview.objects.filter(handmade =
self.kwargs['pk'])

```

```

hms ['revlist'] = hmreviews
hms ['photos'] = handmadeapp.photogetter(1, self.kwargs['pk'])
return hms

```

```

class HandMadeNewView (LoginRequiredMixin, CreateView):

```

```

    template_name = 'handmadeadd.html'
    model = Handmade

```

```

    fields = [
        'title',
        'description',
        'coloroption',
        'personoption',
        'coloroptions'
    ]

```

```

    login_url = 'loginview'
    success_url = 'handmadeview'

```

```

    def form_valid(self, form):
        currentuser = self.request.user
        acc = ShopAccount.objects.get(id = currentuser.id)
        form.shop = acc.shop
        form.state = 0
        form.likes = 0
        form.orders = 0
        form.reviews = 0
        form.reviewscore = 0
        return super().form_valid(form)

```

```
class HandmadeUpdateView (LoginRequiredMixin, UpdateView):
```

```
    template_name = 'handmadeupdate.html'
```

```
    model = Handmade
```

```
    fields = [
```

```
        'title',
```

```
        'description',
```

```
        'coloroption',
```

```
        'personoption',
```

```
        'coloroptions'
```

```
        'state'
```

```
    ]
```

```
    login_url = 'loginview'
```

```
    success_url = 'handmadeview'
```

```
class ShopHandmadeView (ListView):
```

```
    model = Handmade
```

```
    template_name = 'shophm.html'
```

```
    def get_queryset(self, *args, **kwargs):
```

```
        hmcont = super(ShopHandmadeView, self).get_queryset(*args,
**kwargs)
```

```
        shopid = self.request.GET.get('shop')
```

```
        sortn = self.request.GET.get('sort')
```

```
        handmades = Handmade.objects.filter(shop = shopid)
```

```
        handmades = handmades.order_by(sortn)
```

```
        hmcont ['list'] = handmades
```

```
return hmcont
```

```
class ShopAccountHandmadesView (ListView):
```

```
    model = Handmade
```

```
    template_name = 'ashophm.html'
```

```
    def get_queryset(self, *args, **kwargs):
```

```
        hmcont = super(ShopAccountHandmadesView,
self).get_queryset(*args, **kwargs)
```

```
        currentuser = self.request.user
```

```
        acc = ShopAccount.objects.get(id = currentuser.id)
```

```
        cshop = Shop.objects.get(pk = acc.shop)
```

```
        sortn = self.request.GET.get('sort')
```

```
        handmades = Handmade.objects.filter(shop = cshop.id)
```

```
        handmades = handmades.order_by(sortn)
```

```
        hmcont ['list'] = handmades
```

```
        return hmcont
```

```
class SubcategoryHandmadesView (ListView):
```

```
    model = Handmade
```

```
    template_name = 'categhm.html'
```

```
    def get_queryset(self, *args, **kwargs):
```

```
        hmcont = super(SubcategoryHandmadesView,
self).get_queryset(*args, **kwargs)
```

```
        categid = self.request.GET.get('subcategoty')
```

```
        sortn = self.request.GET.get('sort')
```

```
        handmades = Handmade.objects.filter(subcategories __id= <categid>)
```

```

handmades = handmades.order_by(sortn)
hmcont ['list'] = handmades
return hmcont

```

```
class HandmadeLikesView (ListView):
```

```

    model = HandmadeLike
    template_name = 'hmlikes.html'

```

```

    def get_queryset(self, *args, **kwargs):
        hmcont = super(HandmadeLikesView, self).get_queryset(*args,
**kwargs)
        hmid = self.request.GET.get('handmade')
        handmadelikes = HandmadeLike.objects.filter(handmade= hmid)
        hmcont ['list'] = handmadelikes
        return hmcont

```

```
class SearchHandmadeView (ListView):
```

```

    model = Handmade
    template_name = 'searchhandmade.html'

```

```

    def get_queryset(self, *args, **kwargs):
        hmcont = super(SearchHandmadeView, self).get_queryset(*args,
**kwargs)
        hmtitle = self.request.GET.get('title')
        handmades = Handmade.objects.filter(handmade__contains= hmtitle)
        hmcont ['list'] = handmades
        return hmcont

```

```
class BestLikeHandmadeView (ListView):
```

```
    model = Handmade
```

```
    template_name = 'bestlikehm.html'
```

```
    def get_queryset(self, *args, **kwargs):
```

```
        hmcont = super(BestLikedHandmadeView, self).get_queryset(*args,
**kwargs)
```

```
        handmades = Handmade.objects.filter(state = 0)
```

```
        handmades = handmades.order_by('-likes')
```

```
        hmcont ['list'] = handmades [0:99]
```

```
        hmcont carcont
```

```
class BestOrderHandmadeView (ListView):
```

```
    model = Handmade
```

```
    template_name = 'bestorderhm.html'
```

```
    def get_queryset(self, *args, **kwargs):
```

```
        hmcont = super(BestOrderHandmadeView, self).get_queryset(*args,
**kwargs)
```

```
        handmades = Handmade.objects.filter(state = 0)
```

```
        handmades = handmades.order_by('-orders')
```

```
        hmcont ['list'] = handmades [0:99]
```

```
        hmcont carcont
```

```
class BestReviewsHandmadeView (ListView):
```

```
    model = Handmade
```

```
    template_name = 'bestreviewshm.html'
```

```

def get_queryset(self, *args, **kwargs):
    hmcont = super(BestReviewsHandmadeView,
self).get_queryset(*args, **kwargs)
    handmades = Handmade.objects.filter(state = 0)
    handmades = handmades.order_by('-reviews')
    hmcont ['list'] = handmades [0:99]
    hmcont carcont

class BestReviewScoreHandmadeView (ListView):

    model = Handmade
    template_name = 'bestrevscorehm.html'

    def get_queryset(self, *args, **kwargs):
        hmcont = super(BestReviewScoreHandmadeView,
self).get_queryset(*args, **kwargs)
        handmades = Handmade.objects.filter(state = 0)
        handmades = handmades.order_by('-reviewscore')
        hmcont ['list'] = handmades [0:99]
        hmcont carcont

```

Б.3 Текст файла shopreview.py

```

from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import CreateView
import handmadeapp.models
import datetime

```

```

class NewShopReviewView (LoginRequiredMixin, CreateView):

```

```
template_name = 'shopreviewadd.html'
model = ShopReview
```

```
fields = [
    'shop',
    'score',
    'reviewtext'
]
```

```
login_url = 'loginview'
success_url = 'shopview'
```

```
def form_valid(self, form):
    currentuser = self.request.user
    acc = Buyer.objects.get(id = currentuser.id)
    form.buyer = acc.id
    form.reviewdate = datetime.now()
    return super().form_valid(form)
```

```
class BuyerShopReviewsView (ListView):
```

```
    model = ShopReview
    template_name = 'buyershopreview.html'
```

```
    def get_queryset(self, *args, **kwargs):
        srcont = super(BuyerShopReviewsView, self).get_queryset(*args,
**kwargs)
        currentuser = self.request.user
        acc = Buyer.objects.get(id = currentuser.id)
```

```

shopreviews = ShopReview.objects.filter(buyer = acc)
shopreviews = shopreviews.order_by('-reviewdate')
srcont ['list'] = shopreviews
return srcont

```

```
class ShopReviewsView (ListView):
```

```

    model = ShopReview
    template_name = 'shopreviews.html'

```

```

    def get_queryset(self, *args, **kwargs):
        srcont = super(ShopReviewsView, self).get_queryset(*args,
**kwargs)
        sortn = self.request.GET.get('sort')
        shopreviews = ShopReview.objects.filter(shop = cshop.id)
        shopreviews = shopreviews.order_by('-reviewdate')
        srcont ['list'] = shopreviews
        return srcont

```

Б.4 Текст файла `handmadereview.py`

```

from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import CreateView
import handmadeapp.models
import datetime

```

```
class NewHandmadeReviewView (LoginRequiredMixin, CreateView):
```

```

    template_name = 'hmreviewadd.html'

```

```
model = HandmadeReview
```

```
fields = [
    'handmade',
    'score',
    'reviewtext'
]
```

```
login_url = 'loginview'
```

```
success_url = 'handmadereview'
```

```
def form_valid(self, form):
    currentuser = self.request.user
    acc = Buyer.objects.get(id = currentuser.id)
    form.buyer = acc.id
    form.reviewdate = datetime.now()
    return super().form_valid(form)
```

```
class BuyerHandmadeReviewsView (ListView):
```

```
    model = HandmadeReview
```

```
    template_name = 'handmadereview.html'
```

```
    def get_queryset(self, *args, **kwargs):
        srcont = super(BuyerHandmadeReviewsView,
self).get_queryset(*args, **kwargs)
        currentuser = self.request.user
        acc = Buyer.objects.get(id = currentuser.id)
        hmreviews = HandmadeReview.objects.filter(buyer = acc)
        hmreviews = hmreviews.order_by('-reviewdate')
```

```

srcont ['list'] = hmreviews
return srcont

```

Б.5 Текст файла `historyprice.py`

```

from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import CreateView
import handmadeapp.models
import datetime

class NewPriceHistoryView (LoginRequiredMixin, CreateView):

    template_name = 'priceadd.html'
    model = PriceHistory

    fields = [
        'handmade',
        'price',
        'from',
        'to'
    ]

    login_url = 'loginview'
    success_url = 'handmadeview'

class PriceHistoryView (ListView):

    model = PriceHistory
    template_name = 'handmadereview.html'

```

```

def get_queryset(self, *args, **kwargs):
    pccont = super(PriceHistoryView, self).get_queryset(*args, **kwargs)
    hm = self.request.GET.get('handmade')
    phs = PriceHistory.objects.filter(handmade = hm)
    phs = phs.order_by('-to')
    pccont ['list'] = hmreviews
    return pccont

```

Б.6 Текст файла like.py

```

from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import CreateView, DeleteView
import handmadeapp.models
import datetime

```

```

class LikeDeleteView(DeleteView):

```

```

    model = HandmadeLike
    success_url = "/"

```

```

class NewLikeView (LoginRequiredMixin, CreateView):

```

```

    template_name = 'likeadd.html'
    model = HandmadeLike

```

```

    fields = [
        'handmade'
    ]

```

```
login_url = 'loginview'
```

```
success_url = 'handmadeview'
```

```
def form_valid(self, form):
```

```
    currentuser = self.request.user
```

```
    acc = Buyer.objects.get(id = currentuser.id)
```

```
    form.buyer = acc.id
```

```
    return super().form_valid(form)
```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»

кафедра програмних засобів

Кваліфікаційна робота магістра

**Дослідження та програмна реалізація методів
визначення вартості виробів ручної роботи**
**Research and Software Implementation of Methods for
Estimating the Cost of Handmade Products**

Виконав
ст. гр. КНТ-210м

А. А. Грабчук

Керівник
к.т.н., доц.

О. О. Степаненко

• 2021 •

Рисунок В.1 – Слайд 1

Об’єкт роботи – процес продажу виробів ручної роботи.

Предмет роботи – методи визначення вартості виробів ручної роботи.

Мета роботи – розробити програмне забезпечення продажу виробів ручної роботи для підвищення обґрунтованості рішень, які приймаються продавцями щодо встановлення вартості виробів ручної роботи на конкретній платформі або загалом.

• 2

Рисунок В.2 – Слайд 2

Завдання роботи

- виконати аналіз предметної області;
- виконати дослідження методів визначення вартості виробів ручної роботи;
- виконати проєктування програмного забезпечення продажу виробів ручної роботи;
- виконати реалізацію програмного забезпечення продажу виробів ручної роботи.

Рисунок В.3 – Слайд 3

Результати аналізу програмних аналогів

Характеристика	Etsy	ArtFire	Програма-розробка
Платні підписки	Так	Так	Ні
Врахування можливості вибору певної характеристики виробу	Так	Ні	Так
Створення власних переліків виробів	Так	Так	Так
Персоналізація виробів	Є	Немає	Є
Автоматичне визначення вартості виробів	Ні	Ні	Так

Рисунок В.4 – Слайд 4

Метод визначення вартості виробів ручної роботи



Рисунок В.5 – Слайд 5

Порівняння мов програмування

Критерій	PHP	Python
Вебпрограмування	+	+
Вебфреймворки	+	+
Засоби розширення можливостей за допомогою бібліотек	+	++
Оброблення даних	+-	++
Швидкість написання коду	+-	++

Рисунок В.6 – Слайд 6



Рисунок В.7 – Слайд 7



Рисунок В.8 – Слайд 8



Рисунок В.9 – Слайд 9



Рисунок В.10 – Слайд 10



Рисунок В.11 – Слайд 11

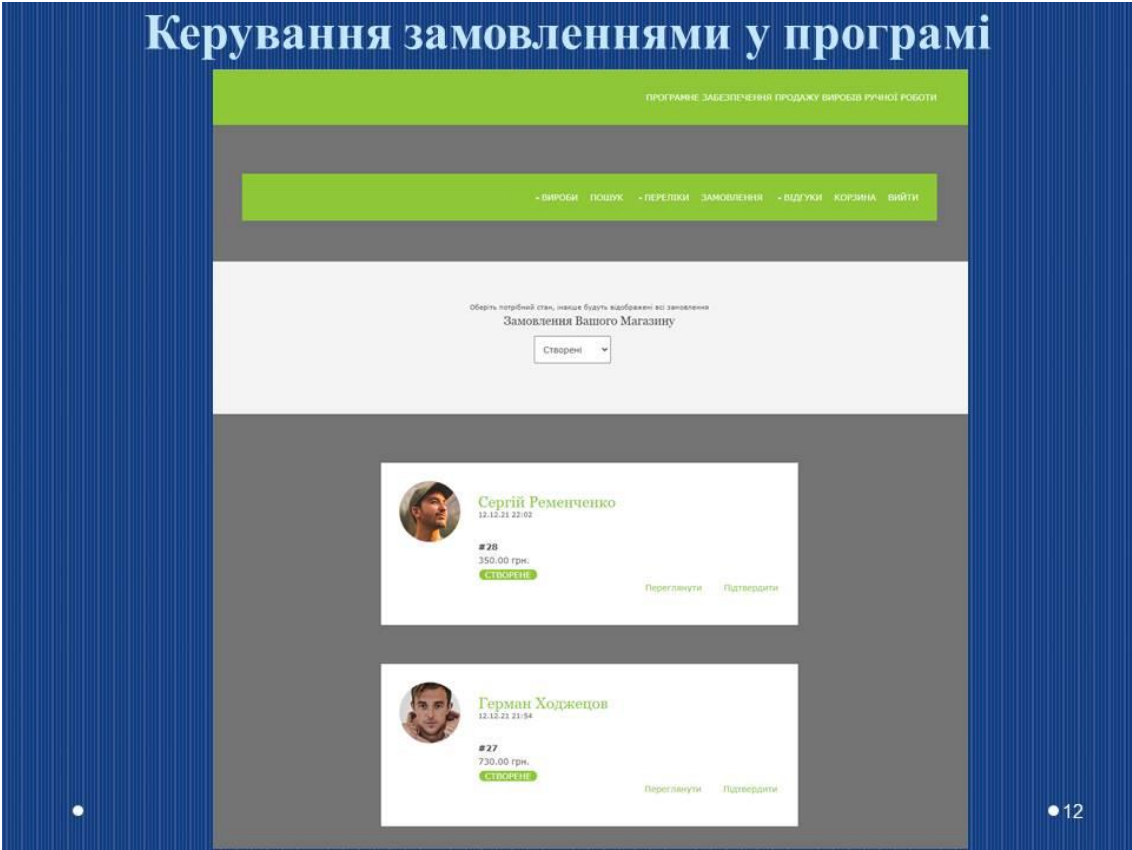


Рисунок В.12 – Слайд 12

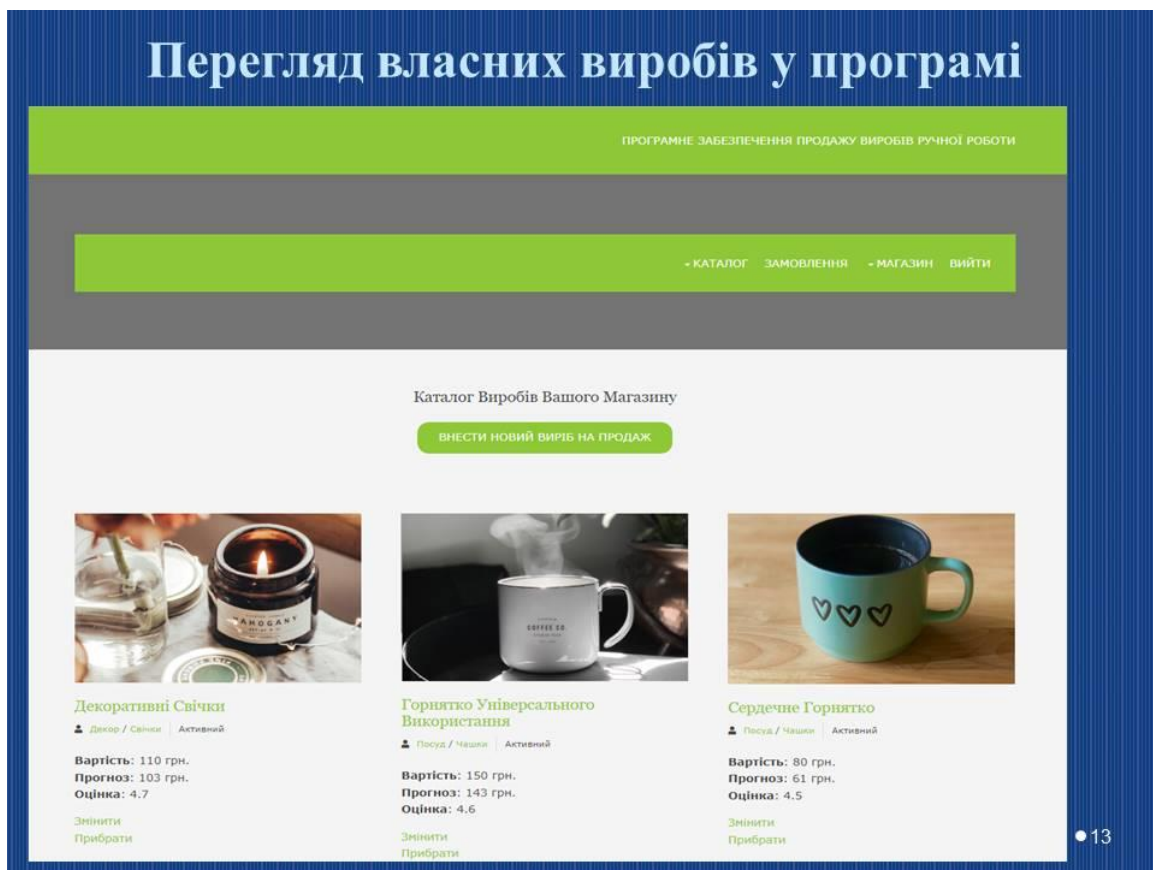


Рисунок В.13 – Слайд 13

Результати експериментального дослідження

Метод	Похибка
Дерево рішень	0,295
Метод, представлений у роботі	0,238

14

Рисунок В.14 – Слайд 14

Висновки

Виконано аналіз предметної області. Проведено аналіз технік, які використовуються для роботи з текстом з метою переведення його до форми вектора. Розглянуто принципи створення дерев рішень. Запропоновано метод визначення вартості виробів ручної роботи, який ґрунтується на обробленні тексту за допомогою моделі мішків слів з подальшим застосуванням дерев рішень, що дозволяє приймати рішення, враховуючи не тільки витрати на створення виробу, але і очікувану вартість в умовах відповідної платформи.

Реалізовано програму. Розроблена програма дозволяє продавцям створювати власні магазини, об'єднуючи таким чином вироби ручної роботи, які вони створюють і бажають продавати, керувати замовленнями покупців, відповідно покупцям дозволяє організовувати пошук виробів ручної роботи, замовляти їх, надає інструментар відгуків для створення зворотного зв'язку про вироби.

Усі поставлені завдання на роботу виконано.

•

•15

Рисунок В.15 – Слайд 15