

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ

(назва теми)

ВИЗНАЧЕННЯ ПОПУЛЯРНОСТІ ПІД ЧАС ПРОДАЖУ ТОВАРІВ
РУЧНОЇ РОБОТИ. RESEARCH AND SOFTWARE IMPLEMENTATION
OF METHODS FOR EVALUATING THE POPULARITY OF HAND MADE
ITEMS DURING SALE

Виконав(ла): студент(ка) 2 курсу, групи КНТ-132м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ПАВЛІЧЕНКО О.А.

(ПРІЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(ПРІЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРІЗВИЩЕ та ініціали)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ ФКНТ _____
Кафедра _____ програмних засобів _____
Ступінь вищої освіти _____ магістр _____
Спеціальність _____ 121 Інженерія програмного забезпечення _____
(код і найменування)
Освітня програма (спеціалізація) _____ Інженерія програмного забезпечення _____
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри _____ ПЗ, д.т.н, проф. _____
Сергій СУББОТІН
« _____ » _____ 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПАВЛІЧЕНКО Олени Андріївни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів визначення популярності під час продажу товарів ручної роботи. Research and Software Implementation of Methods for Evaluating the Popularity of Hand Made Items during Sale

керівник проєкту (роботи) к.т.н., доцент СКРУПСЬКИЙ Степан Юрійович _____,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 14 » листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 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. Розробка архітектури програми. 4. Реалізація програми.

5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	СКРУПСЬКИЙ С.Ю., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання « 04 » вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Дослідження проблеми	3-4 тижні	Розділ 2
4	Розробка архітектури програми	5 тиждень	Розділ 3
5	Розробка програми	6-7 тижні	Розділ 4
6	Тестування та експериментальне дослідження програми	8-9 тижні	Розділ 5
7	Оформлення пояснювальної записки та документів до неї	10 тиждень	Додатки
8	Нормоконтроль та рецензування	11 тиждень	
9	Захист роботи	12 тиждень	

Студент(ка)

_____ Олена ПАВЛІЧЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Степан СКРУПСЬКИЙ
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
103 с., 25 рис., 5 табл., 3 дод., 11 джерел.

ПРОДАЖ ТОВАРІВ, РУЧНА РОБОТА, ДЕРЕВА РІШЕНЬ,
ВЕБЗАСТОСУНОК, ПРОГНОЗУВАННЯ.

Об'єкт роботи – процес продажу товарів ручної роботи. Предмет роботи – методи визначення популярності під час продажу товарів ручної роботи. Мета роботи – розробка програмного забезпечення підтримки продажу товарів ручної роботи з можливістю визначення популярності товарів на майбутнє.

Матеріали, методи та технічні засоби: мова Python, фреймворк Django, система MySQL.

Результати. Розроблена як вебзастосунок програма призначена для продажу товарів, які є результатом ручної роботи, крамницями, які представляють майстрів і яких представляють відповідно продавці, та придбання товарів споживачами. Для цього підтримуються засоби керування замовленнями, визначення доступних товарів, збереження товарів і крамниць для подальшого швидкого повернення до них, створення рейтингів товарів ручної роботи та крамниць. Проведено експериментальне дослідження визначення популярності під час продажу товарів ручної роботи.

Висновки. Наукова новизна роботи полягає в запропонованому методі визначення популярності товарів ручної роботи, в якому на основі числових характеристик товару ручної роботи та текстових даних, перетворених до числового представлення, виконується прогнозування кількості рейтингів за допомогою створення дерева рішень CART.

Галузь використання. Продаж товарів ручної роботи, майданчик для підтримки крамниць, що продають товари ручної роботи.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 103 pages, 25 figures, 5 tables, 3 appendixes, 11 sources.

SALE OF GOODS, CRAFT, DECISION TREES, WEB APPLICATION, FORECASTING.

The object of work is the process of selling hand made items. The subject of the work is methods of evaluating the popularity of hand made items. The purpose of the work is to develop software to support the sale of hand made items with the possibility of evaluating the popularity of hand made items for the future.

Materials, methods and technical means: Python language, Django framework, MySQL system.

The results. Developed as a web application, the program is intended for the sale of goods that are the result of handwork, by shops representing craftsmen by sellers, respectively, and for the purchase of items by consumers. To do this, it supports order management, identifying available products, saving products and stores for quick return to them later, creating ratings of hand made items and stores. An experimental study was conducted to determine the popularity during the sale of hand made items.

Conclusions. The scientific novelty of the work lies in the proposed method of evaluating the popularity of hand made items. It is based on the numerical characteristics of hand made items and text data converted to a numerical representation. The number of ratings is predicted by creating CART decision trees.

Field of use. Selling handmade goods, a platform to support shops selling handmade goods.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Програмні інструменти підтримки продажу товарів ручної роботи	10
1.2 Порівняння розроблюваної програми з існуючими	13
1.3 Висновки за розділом 1	14
2 Матеріали і методи.....	16
2.1 Використання методів прогнозування для розв’язання задачі визначення популярності товарів ручної роботи	16
2.2 Метод визначення популярності товарів ручної роботи	18
2.3 Висновки за розділом 2	20
3 Розробка архітектури програми.....	21
3.1 Визначення сценаріїв використання програми користувачами	21
3.2 Рішення щодо вибору засобів розробки	25
3.3 Визначення структури даних у базі даних	26
3.4 Структура програми.....	41
3.5 Висновки за розділом 3	42
4 Реалізація програми	43
4.1 Визначення послідовності роботи з програмою	43
4.2 Особливості реалізації програми.....	44
4.3 Висновки за розділом 4	58
5 Експлуатація, тестування та експериментальне дослідження програми	59
5.1 Призначення програми	59
5.2 Виконання програми.....	59
5.3 Експериментальне дослідження	64
5.4 Висновки за розділом 5	67
Висновки	68
Перелік джерел посилання	69

Додаток А Технічне завдання	70
Додаток Б Текст програми	75
Додаток В Слайди презентації.....	96

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CART – Classification And Regression Tree;

БД – база даних.

ВСТУП

Ручна робота призводить до отримання ексклюзивних варіантів товарів. Вони можуть мати особливості, пов'язані зі споживачем, який їх замовив, але з іншого боку вони зазвичай мають певні особливості виконання. Це широковідомий факт і він як раз і привертає увагу до товарів ручної роботи. Але зворотнім боком є те, що кожен майстер створює невелику кількість товарів, а товар, який створюється іншим майстром, навіть якщо в нього та сама концепція, навіть дуже близьке виконання, все одно є іншим товаром. Тож коли виконується пошук нових товарів ручної роботи, то знайти їх стає достатньо складно, зрозуміти, який саме товар підходить ще складніше, адже на це впливає не тільки концепція, але і конкретне виконання. Отримати інформацію про конкретне виконання дозволяють рейтингові оцінювання або відгуки, які створюють споживачі, які вже такий товар купили. Кількість же таких рейтингових оцінювань, кількість замовлень товарів говорить про його популярність. Популярність разом з середнім рівнем рейтингів впливають на сприйняття інших споживачів. Тому реалізація процесів роботи з товарами ручної роботи вимагає створення програмного забезпечення, яке охоплює цілий ряд базових концептуальних функцій:

- надання інформації про сам товар ручної роботи, замовлення такого товару;
- пошук товарів, що створені в результаті ручної роботи;
- визначення популярності товарів, що створені в результаті ручної роботи, на початку продажів.

Відповідно метою роботи є розробка програмного забезпечення підтримки продажу товарів ручної роботи з можливістю визначення популярності товарів на майбутнє.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Програмні інструменти підтримки продажу товарів ручної роботи

Головним гравцем на ринку програмного забезпечення, що забезпечує продаж товарів ручної роботи, є Etsy (рис. 1.1) [1].

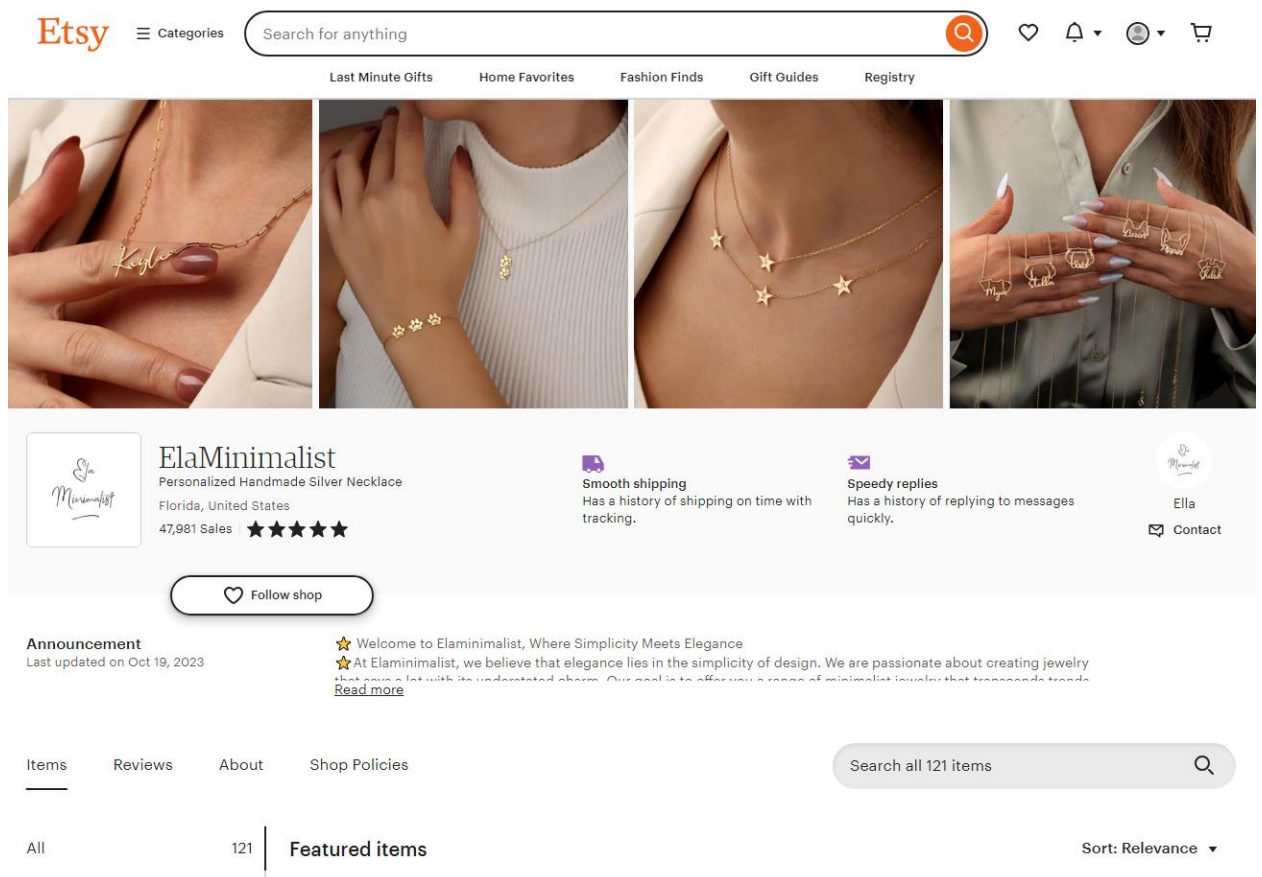
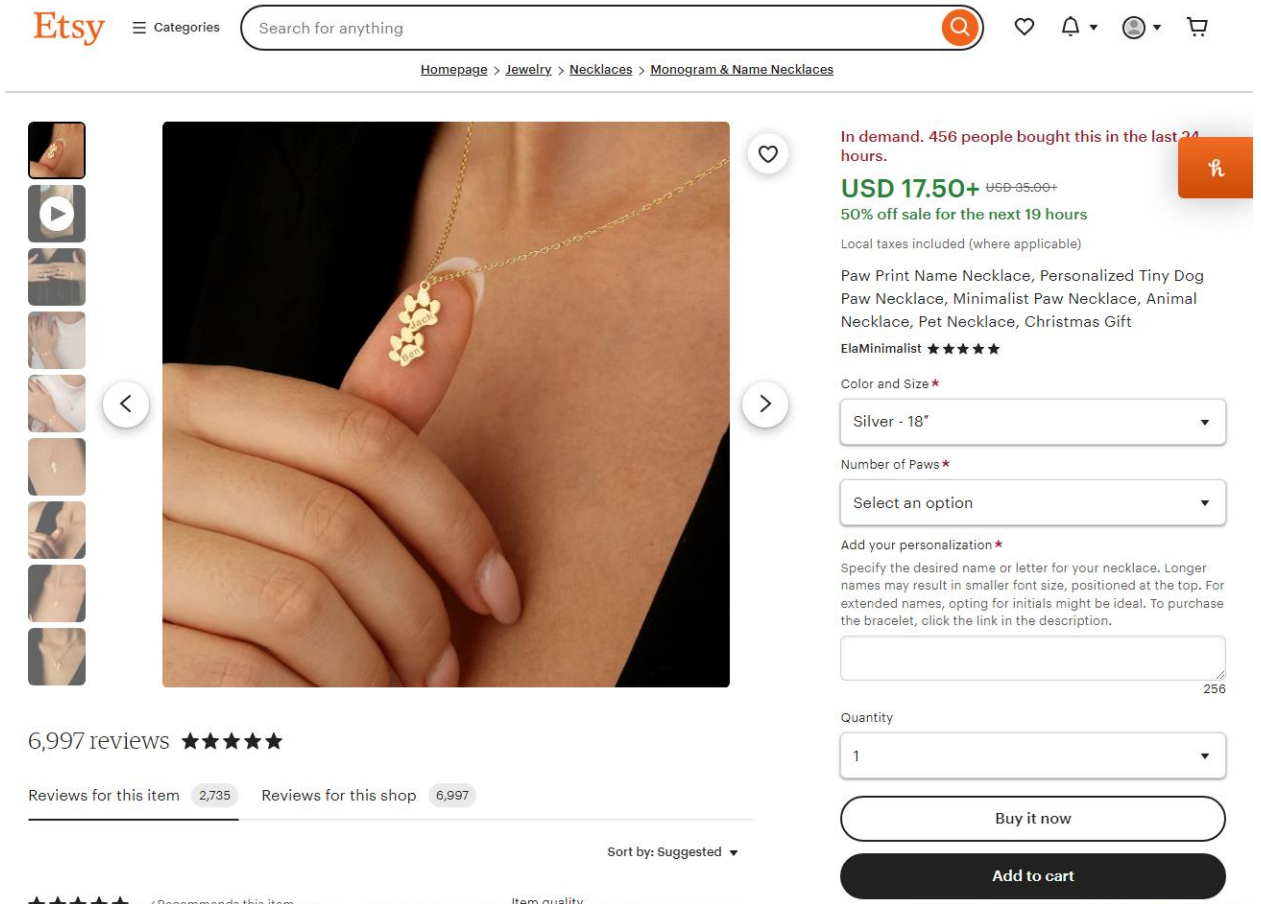


Рисунок 1.1 – Крамниця в програмі Etsy

Etsy побудований на основі створення оголошень про продажу товарів ручної роботи, при чому ці оголошення концентруються навколо крамниць. Тобто саме крамниці є точкою продажу товарів. Товари, що важливо, можуть мати засоби персоналізації, а також визначення властивостей, значення яких може обрати сам споживач (рис. 1.2). Відповідно замовлення формується вже на конкретний варіант товару, а далі вже з замовленням працює продавець, який представляє крамницю.



Etsy Categories Search for anything

Homepage > Jewelry > Necklaces > Monogram & Name Necklaces

In demand. 456 people bought this in the last 24 hours.

USD 17.50+ ~~USD 35.00+~~
50% off sale for the next 19 hours

Local taxes included (where applicable)

Paw Print Name Necklace, Personalized Tiny Dog Paw Necklace, Minimalist Paw Necklace, Animal Necklace, Pet Necklace, Christmas Gift

ElaMinimalist ★★★★★

Color and Size ★
Silver - 18"

Number of Paws ★
Select an option

Add your personalization ★
Specify the desired name or letter for your necklace. Longer names may result in smaller font size, positioned at the top. For extended names, opting for initials might be ideal. To purchase the bracelet, click the link in the description.

Quantity
1

Buy it now

Add to cart

6,997 reviews ★★★★★

Reviews for this item 2,735 Reviews for this shop 6,997

Sort by: Suggested

★★★★★ / Recommend this item Item quality

Рисунок 1.2 – Товар ручної роботи в програмі Etsy

Товари можна переглядати за категоріями, можна шукати їх, встановлюючи назву, а після цього додаючи додаткові фільтри, якщо це необхідно, які включають розташування крамниці, строк, за який товар має бути готовий для відправки, вартість, колір, тип товару (всі, ручна робота або вінтаж). Тут важливо відзначити, що щодо характеристик товару можна задати тільки колір при пошуку, всі інші параметри можуть визначатися на сторінці, але в пошуку не задаються. Проте вибір з властивостей конкретного товару можна виконувати, задаючи одну властивість стандартно та одну персональну. Персональну можна змінювати і відповідно впливати на вартість товару, якщо вона взагалі змінюється. Стандартну властивість можна обирати, але на вартість вона не впливає. Тому якщо можна було в конкретному товарі, наприклад, задати колір і розмір, які на вартість не впливають, а тоді ще задати якусь аплікацію, вибір якої впливає на вартість, то це стає проблемою.

Звичайно, що серед програмних інструментів, які дозволяють працювати в цій предметній області є чимало таких, які є вебсайтами окремих крамниць. Але зважаючи на обмежену функціональність таких рішень, вони не виносились у порівняння в результаті.

Українським майданчиком, на якому представлено ряд крамниць з продажу товарів ручної роботи, є Crafta (рис. 1.3) [2].

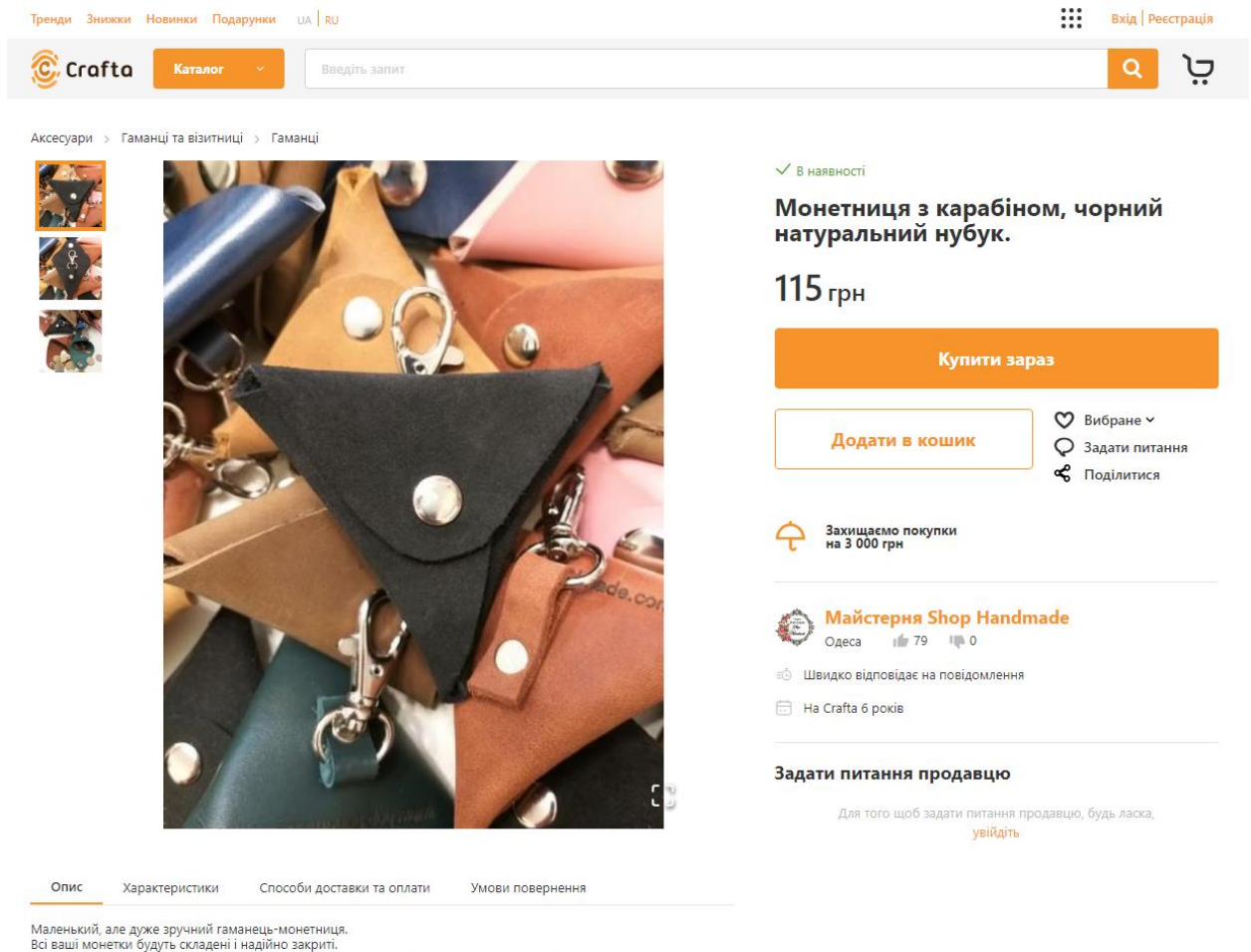


Рисунок 1.3 – Сторінка товару ручної роботи в програмі Crafta

Crafta використовується як майданчик, на якому з'являються крамниці, що продають товари ручної роботи. Програма має дуже обмежений пошук. У пошуку можна задати назву товару, визначити необхідну вартість, тип виробу (в наявності, для прикладу або на замовлення) та місто, звідки товар буде відправлятися. Також можна обрати напрямок сортування. Але інших критеріїв пошуку немає. Проте користувачі можуть зберегти товари до

вибраного, можуть замовляти товари, можуть ставити запитання представникам крамниць. При цьому відгуки можна створювати, але тільки стосовно крамниць.

1.2 Порівняння розроблюваної програми з існуючими

Завершивши вивчення предметної області, виконавши аналіз аналогів, було виконано порівняння програм Etsy і Crafta відносно розробки, яку було заплановано. Результати порівняння є результатами аналізу (табл. 1.1).

Таблиця 1.1 – Аналіз розроблюваної та існуючих програм

Показник	Etsy	Crafta	Власна розробка
Крамниці продажу товарів ручної роботи	Так	Так	Так
Пошук крамниць	Ні	Ні	Так
Пошук товарів ручної роботи за властивостями	Колір	Немає	Різні
Рейтинги крамниць	Так	Так	Так
Рейтинги товарів ручної роботи	Так	Ні	Так
Вибір значень властивостей для товару	Не більше 2	Немає	Можливо більше 2
Визначення популярності товарів	Немає	Немає	Так

Розробка, яка виконується, повинна дозволяти виконувати пошук не тільки за товарами ручної роботи, але і за крамницями, які їх продають. Це

має дозволити виконувати такий пошук цілеспрямовано, якщо споживач одразу знає, товари якої крамниці його цікавлять. Це наприклад, стається тоді, коли споживач має пораду від друга або просто іншого споживача і потребує знайти конкретну крамницю. Разом з наявністю можливості зберегти крамницю для повернення до неї це може бути корисним для споживача, підвищити зручність його роботи.

Crafta не дозволяє шукати товари за властивостями, тобто знайти товар, виготовлений з потрібного матеріалу не вдасться. В той же час Etsy дозволяє встановлювати тільки кольори. Виконувана розробка має дозволити включати різні властивості та їх значення до пошуку.

Etsy дозволяє при замовленні товару вибрати потрібні властивості як опції, однак, лише 1 може бути задана без зміни вартості і одна така, яка призводить або може призводити до зміни вартості. Проте іноді потрібно використовувати 3 опції (розмір, колір, аплікація). У розробці це потрібно було дозволити.

Окрім того програма, що розробляється, повинна надати інструмент для визначення популярності товарів, який має базуватися на прогнозуванні, і дозволити продавцю зрозуміти, на який рівень популярності він може розраховувати з таким товаром ручної роботи та відповідним його фактично описом у програмі.

1.3 Висновки за розділом 1

Набір завдань для виконання в роботі:

- аналіз існуючих програм, які виконують підтримку продажу товарів ручної роботи;
- дослідження методів і моделей визначення популярності товарів ручної роботи;
- проєктування програми підтримки продажу товарів ручної роботи;
- реалізація програми підтримки продажу товарів ручної роботи.

У роботі при виконанні першого розділу було виконано завдання з аналізу існуючих програм, які виконують підтримку продажу товарів ручної роботи. Було порівняно програми Etsy, Crafta, визначено відмінності цих програм порівняно з розробкою даної роботи.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Використання методів прогнозування для розв'язання задачі визначення популярності товарів ручної роботи

Першим питання, яке виникло під час формулювання задачі визначення популярності товарів ручної роботи, було те, що слід вважати популярністю товару ручної роботи, яким чином можна популярність оцінити. Під популярністю можна розглядати кількість проданого товару, створеного в результаті ручної роботи, але з одного боку так враховуються і споживачі, які не оцінили товар, хоча це і можна вважати результатом популярності, проте так не враховуються ті, хто привертає увагу до товару вже після придбання, виконуючи ці дії повторно, тобто товар людина могла придбати лише один раз, але при цьому багато про нього повідомляє і таким чином збільшує його популярність, сприяючи тим процесам, які відбуваються. Тому вирішено було використовувати кількість рейтингів (відгуків) товару, створеного в результаті ручної роботи, щоб таким чином виразити популярність.

Виходячи з цього, задача визначення популярності товарів ручної роботи полягає в тому, щоб виконати прогнозування кількості рейтингів, які отримає товар ручної роботи, якщо про товар відомі значення певних показників, які його характеризують. До таких характеристик може належати назва, опис товару тощо. Тож для кожного екземпляра товару потрібно за значеннями різних властивостей, які називаються вхідними ознаками, виконати прогнозування кількості створених рейтингів.

Значеннями властивостей, якими характеризуються товари ручної роботи, для деяких з товарів відомі наперед на основі зібраних даних і для кожного такого екземпляра відомою є і кількість рейтингів, які такий товар отримав. Тому в підсумку задача визначення популярності товарів ручної роботи полягає в тому, щоби знаючи значення визначеного набору властивостей для певних екземплярів та кількість рейтингів, які зібрали ці

екземпляри, виконати прогнозування кількості рейтингів, які отримають інші товари ручної роботи, для яких така кількість рейтингів наперед невідома.

Задача визначення популярності товарів ручної роботи за класифікацією задач машинного навчання є задачею регресії. Існують різні способи розв'язання таких задач. Найбільш традиційною і відомою моделлю прогнозування є лінійна регресія. Лінійна регресія визначає зв'язок між середнім значенням однієї змінної та відповідними значеннями однієї чи декількох інших змінних, створюючи таким чином певну модель, яка як раз в подальшому, як це було визначено в задачі визначення популярності товарів ручної роботи на основі значень властивостей товару прогнозувати вихідну змінну, якою є кількість рейтингів [3].

Розрізняють наступні варіанти моделі лінійної регресії:

- проста лінійна регресія, яка знаходить функцію, що відображає точки даних на пряму лінію при представленні графіка двох змінних;
- множинна лінійна регресія, яка знаходить функцію, що відображає точки даних на пряму лінію між однією залежною змінною (вихідним значенням моделі), якою і є кількість рейтингів, і функцією двох або більше незалежних змінних (вхідних ознак моделі);
- нелінійна регресія знаходить функцію, що вписує дві або більше змінних на криву, а не на пряму лінію [3].

Ще одним варіантом розв'язання задачі прогнозування на даний момент є моделі, які набрали достатню популярність і активно використовуються на практиці, – дерева рішень Classification And Regression Tree (CART). Це моделі, які можуть одночасно використовуватися і для класифікації даних, і для розв'язання задачі регресії, тобто виконання прогнозування.

CART – це непараметрична модель, яка може обробляти як категоріальні, так і безперервні змінні, працює шляхом рекурсивного поділу даних на підмножини на основі значення змінної, що прогнозується, розбиття

виконується таким чином, щоб отримані підмножини були максимально однорідними щодо цільової змінної [4]-[5].

CART використовує дані з набору для прогнозування цільового значення, Розпочинаючи роботу з кореневого вузла, який розділяється далі на менші вузли, поки не досягне максимальної глибини, а на найнижчому рівні, що відповідає глибині дерева, листові вузли представляють остаточні прогнозовані мітки для кожного зразка в наборі даних [4]-[5].

2.2 Метод визначення популярності товарів ручної роботи

Метод визначення популярності товарів ручної роботи має ґрунтуватися на використанні зокрема текстових даних про товар, адже зазвичай саме в текстовому описі міститься основна інформація. Зважаючи на ручну роботу, тобто індивідуальні особливості товарів, наявність товарів ручної роботи з різних категорій, товарів, які між собою значно відрізняються (тобто мають дуже різні характеристики), саме загальні текстові описи містять детальну інформацію про товари. Тому для роботи даного методу важливим було визначення способу перетворення текстових даних у числові, щоб в подальшому можна було використати дерева рішень для роботи з такими даними, а також даними, які одразу представлені в числовій формі. Це перетворення називається вкладанням слів і воно призводить до отриманні вектору з закодованими словами, при чому важливою особливістю є те, що в такому векторному представленні слова, які мають більш близьке значення, у числовому представленні також будуть мати більш близьке числове представлення. Для виконання такого перетворення було використано модель Word2vec [6].

Робота методу визначення популярності товарів ручної роботи розпочинається з формування навчальної вибірки даних. Навчальна вибірка має бути сформована з екземплярів, які представлені товарами ручної роботи, для кожного з яких відомі всі необхідні вхідні ознаки та кількість

рейтингів, яку зібрав цей товар ручної роботи як вихідна ознака. До вхідних ознак належать:

- встановлена в оголошенні вартість одиниці товару, створеного в результаті ручної роботи;
- категорія, до якої належить товар ручної роботи;
- середня оцінка рейтингу товару ручної роботи;
- опис товару, створеного в результаті ручної роботи, що приводиться в оголошенні;
- назва оголошення, що вказана за цим товаром ручної роботи;
- опис матеріалів, з яких виготовлений товар ручної роботи, що далі називається деталями.

Під час обробки сформованої вибірки даних одразу має бути перетворена категорія, до якої належить товар ручної роботи, до категоріального, тобто числового значення.

Після того, як зібрано дані про товари ручної роботи відносно кожної вхідної ознаки та вихідної ознаки, необхідно опрацювати текстові дані, до яких належать назва оголошення, деталі та опис товару, необхідно виконати перетворення цих даних до числового представлення за допомогою відповідних векторів. Для цього запропоновано побудувати три окремі моделі, кожна з яких буде працювати відповідно з назвою оголошення, деталями та описом товару, виконаними відповідною мовою. Таким чином буде досягнуто спеціалізації, і кожна модель зможе обробляти не всі тексти, слова відповідною мовою, а тільки за вибраною тематикою: перша – назвами оголошень, друга – описом товару, третя – описом матеріалів.

Для створення моделей на основі Word2vec виділяються тексти, які належать до відповідної тематики, обробляються, відкидаючи з них ті слова, які у відповідній мові не є інформативними, та формується набір текстів, на основі якого і відбувається навчання моделі. В результаті створюються 3 окремі моделі. Модель з назвами оголошень має бути сформована з

представленням слів у вигляді вектору з 20 елементів, деталей – 10, а описів – 50.

На наступному етапі має бути оброблено тексти відповідних вхідних ознак для кожного екземпляра, подаючи на вхід відповідної моделі кожне слово тексту за виключенням знаків пунктуації, неінформативних слів, а далі знаходячи суму за кожним елементом вектору для кожного слова тексту і ділячи цю суму на кількість слів, обчислюються значення всіх елементів вектору для кожного текстового опису.

Далі відбувається створення моделі на основі дерева рішень CART і реалізується її навчання за допомогою створеної навчальної вибірки.

Наступний етап роботи методу визначення популярності товарів ручної роботи виконується вже за появи даних стосовно нових товарів ручної роботи. Тоді кожен товар описується середньою оцінкою рейтингів, вартістю товару та текстовими представленнями опису, назви та деталей, оброблених описаним вище чином для перетворення до числової форми. Створена модель прогнозує кількість створених рейтингів щодо даного товару на виході.

2.3 Висновки за розділом 2

Розглянуто проблему визначення популярності товарів ручної роботи і виконано формулювання відповідної задачі. Розглянуто шляхи розв’язання цієї задачі та запропоновано метод для її розв’язання.

Наукова новизна роботи полягає в запропонованому методі визначення популярності товарів ручної роботи, в якому на основі числових характеристик товару ручної роботи та текстових даних, перетворених до числового представлення, виконується прогнозування кількості рейтингів за допомогою створення дерева рішень CART.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

3.1 Визначення сценаріїв використання програми користувачами

Сценарії програми, необхідні для забезпечення для споживачів:

- відображення найкращих товарів ручної роботи за категорією;
- відображення найкращих товарів ручної роботи за крамницею;
- відображення найкращих товарів ручної роботи за тегом;
- відображення товарів ручної роботи, що продаються в крамниці;
- відображення товарів ручної роботи, що найактивніше продаються;
- відображення найпопулярніших товарів ручної роботи за категорією;
- відображення найпопулярніших товарів ручної роботи за крамницею;
- пошук крамниць, які продають товари ручної роботи;
- пошук товарів ручної роботи;
- створення шпильки на крамниці;
- прибирання шпильки з крамниці;
- відображення крамниць, на яких залишено шпильки;
- збереження товару ручної роботи;
- відображення збережених товарів ручної роботи;
- вилучення збереження товару ручної роботи;
- замовлення товарів ручної роботи;
- відображення замовлень товарів ручної роботи;
- авторизація;
- скасування замовлення товарів ручної роботи;
- відображення інформації про товар ручної роботи;
- відображення інформації про крамницю, що продає товари ручної роботи;
- рейтингове оцінювання товару ручної роботи;
- рейтингове оцінювання крамниці (рис. 3.1).

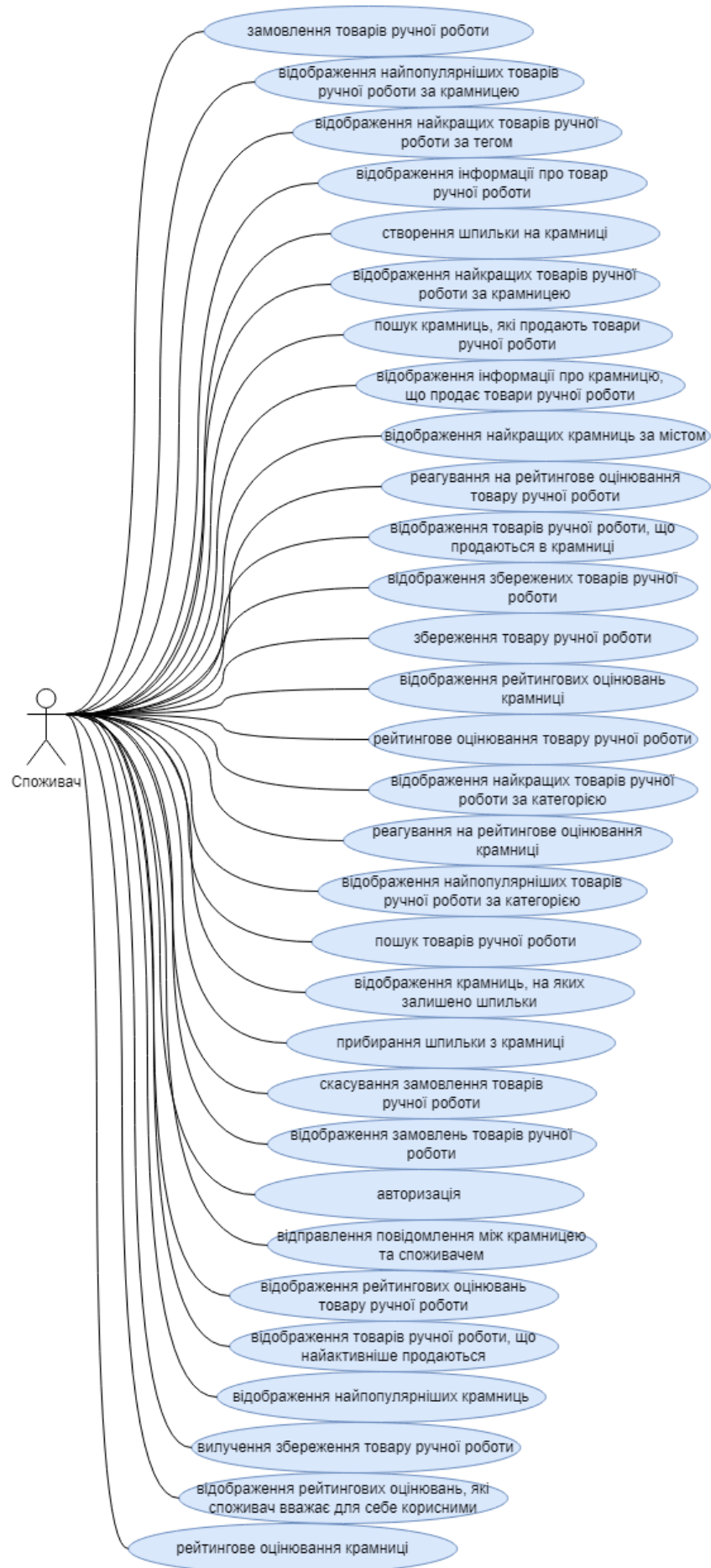


Рисунок 3.1 – Діаграма прецедентів споживача

Також сценаріями споживача є:

- відображення рейтингових оцінювань крамниці;
- відображення рейтингових оцінювань товару ручної роботи;
- реагування на рейтингове оцінювання товару ручної роботи;
- реагування на рейтингове оцінювання крамниці;
- відображення найкращих крамниць за містом;
- відображення найпопулярніших крамниць;
- відправлення повідомлення між крамницею та споживачем;
- відображення рейтингових оцінювань, які споживач вважає для себе

корисними.

Сценарії програми, необхідні для забезпечення для продавців, які представляють крамниці:

- відображення найкращих товарів ручної роботи за категорією;
- відображення найкращих товарів ручної роботи за крамницею;
- відображення найкращих товарів ручної роботи за тегом;
- відображення товарів ручної роботи, що продаються в крамниці;
- відображення товарів ручної роботи, що найактивніше продаються;
- відображення найпопулярніших товарів ручної роботи за категорією;
- відображення найпопулярніших товарів ручної роботи за крамницею;
- пошук крамниць, які продають товари ручної роботи;
- пошук товарів ручної роботи;
- додавання товару ручної роботи для продажу в крамниці;
- редагування даних товару ручної роботи;
- відображення результатів роботи крамниці;
- скасування замовлення товарів ручної роботи;
- відображення замовлень товарів ручної роботи за крамницею;
- відображення замовлень заданого товару ручної роботи;
- авторизація (рис. 3.2).

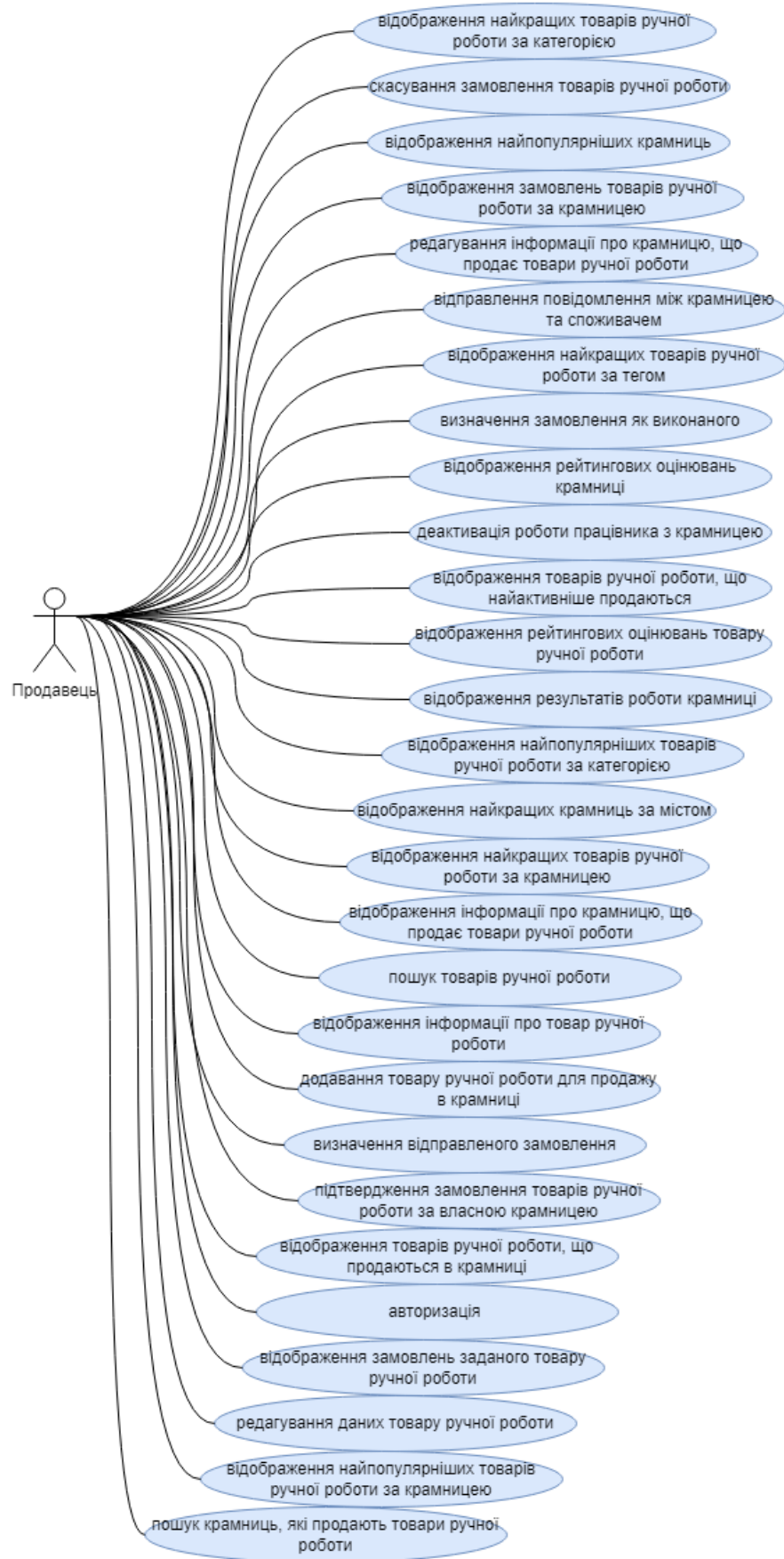


Рисунок 3.2 – Діаграма прецедентів продавця

Також сценаріями продавця є:

- підтвердження замовлення товарів ручної роботи за власною крамницею;
- деактивація роботи працівника з крамницею;
- визначення замовлення як виконаного;
- визначення відправленого замовлення;
- відображення інформації про товар ручної роботи;
- відображення інформації про крамницю, що продає товари ручної роботи;
- редагування інформації про крамницю, що продає товари ручної роботи;
- відображення найкращих крамниць за містом;
- відображення найпопулярніших крамниць;
- відображення рейтингових оцінювань крамниці;
- відображення рейтингових оцінювань товару ручної роботи;
- відправлення повідомлення між крамницею та споживачем.

3.2 Рішення щодо вибору засобів розробки

Враховуючи необхідність роботи віддалено, різні типи пристроїв, обмежені строки, створення вебзастосунку є найкращим рішенням для такої ситуації. Реалізувати вебзастосунок можна більш простим чином, використовуючи систему менеджменту контентом, а можна використати вебфреймворк і реалізувати її повністю власноруч. В якості системи менеджменту контентом розглянуто Drupal [7], в якості фреймворку – Django [8] (табл. 3.1). Обраним рішенням є Django, адже програма має багато функцій, які потребують певної особливої реалізації (наприклад, рейтинги можуть створювати тільки споживачі, які придбали відповідний товар), особливої структури даних для роботи, підтримки машинного навчання, до

того ж Django дозволяє менше витратити часу на забезпечення базового рівня безпеки.

Таблиця 3.1 – Порівняння засобів розробки програми підтримки продажу товарів ручної роботи

Показник	Drupal	Django
Розробка нетипових функцій	Потребує наявності відповідних плагінів і модулів або написання і інтеграції	Реалізація предметної області, бізнес-логіки лежить в основі, додаткова реалізація не потрібна
Безпека роботи	Потребує додаткових дій, налаштувань, обов'язкових стратегій	Має багато вбудованих засобів, що вирішують велику кількість проблем одразу
Підтримка спільнотою розробників	Активна	Активна
Мова програмування	PHP	Python
Підтримка реалізації машинного навчання	Дуже обмежена	На найвищому рівні

3.3 Визначення структури даних у базі даних

База даних (БД) є сховищем, яке вміщує в себе всі дані, на основі яких організована підтримка роботи всіх сценаріїв роботи програми, відбувається формування вибірки даних для підтримки роботи методу визначення популярності товарів ручної роботи.

Звичайно, що сам товар, який є результатом ручної роботи, є однією з головних, а можливо центральною сутністю програми, адже від наданої

інформації про товар ручної роботи залежить те, чи зацікавиться ним споживач, чи вирішить він придбати цей товар, чи звернеться за додатковою консультацією до крамниці.

Таблиця БД Handware для забезпечення роботи з товарами, які є результатом ручної роботи, включає такі характеристики:

- id – ідентифікатор товару, який був створений в результаті ручної роботи;
- adv_title – назва оголошення, в якому міститься інформація про вручну виготовлений товар;
- adv_description – рекламний текст опису з деталями щодо даного товару ручної роботи з представленням всіх особливостей, які має даний товар, що надав його продавець;
- handwarestore – крамниця, в якій продається цей товар, створений в результаті ручної роботи, визначається записом таблиці Handwarestore, тобто є зовнішнім ключем;
- shipdays – кількість днів на поточний момент, через яку продавець очікує відправити цей товар, що важливо зокрема не тільки як стандартний показник, але і для визначення тих товарів, які довго виготовляються вручну, для яких відсутній запас, а також під час святкового періоду, коли час відвантаження є тривалішим ніж зазвичай;
- notice_rate – середній рейтинг цього товару ручної роботи на основі загальної оцінки від споживачів;
- service_rate – середній рейтинг цього товару ручної роботи на основі обслуговування, наданого щодо цього товару, від споживачів;
- shipping_rate – середній рейтинг цього товару ручної роботи на основі доставки товару від споживачів;
- quality_rate – середній рейтинг цього товару ручної роботи на основі оцінки якості виготовлення товару від споживачів;

- `fit_rate` – середній рейтинг цього товару ручної роботи на основі оцінки того, на скільки фактично цей товар відповідає його опису та інформації на вебсайті, від споживачів;

- `availability` – кількість екземплярів товару ручної роботи, що може замовити споживач на даний момент;

- `total_sales` – сумарна кількість одиниць даного товару ручної роботи, яку було продано за весь час роботи з ним даної крамниці;

- `week_sales` – сумарна кількість одиниць даного товару ручної роботи, яку було продано протягом останнього тижня;

- `baseprice` – базова, тобто основна вартість даного товару, яка використовується тоді, коли не можна встановлювати персональні характеристики, що впливають на екземпляр товару, який буде доставлено споживачу, або коли серед варіантів вибору можливий варіант вибору без визначення персональних характеристик;

- `basevariant` – можливість продавати цей товар ручної роботи без встановлення персональних характеристик (наприклад, написів) або неможливість це робити;

- `personaloption` – назва персонального параметра, значення для якого можна вибрати;

- `optionhint` – підказка щодо суті даного персонального параметра, тобто що він позначає.

Фотографії, за якими можна візуально отримати інформацію про товар, який виготовляється вручну, винесені в таблицю `Hardwareimage`, адже фотографій може бути багато для кожного товару. Характеристики фотографії товару ручної роботи:

- `id` – ідентифікатор товару, який був створений в результаті ручної роботи;

- `hardware` – товар ручної роботи, якого стосується ця фотографія, визначений відповідним записом таблиці `Hardware`, тобто є зовнішнім ключем;

- imagefile – фотографія;
- hint – підказка щодо тих особливостей товару, який був створений в результаті ручної роботи і якого стосується ця підказка.

Зважаючи на те, що товари створюються як результат ручної роботи, в програмі може бути дуже багато різних товарів, тому важливо їх структуровано охарактеризувати. Товари можуть мати характеристики, які є стандартними, тобто не можуть вибиратися споживачем, а можуть бути такими, які споживач може обрати в своєму екземплярі. Наприклад, футболка може бути виготовлена тільки з одного матеріалу, але споживач може обрати розмір. Відповідно в таблиці `Hardwareproperty` містяться значення тих властивостей товару ручної роботи, які є незмінними, а в таблицях `Hardwaregeneraloption` і `Hardwarepersonaloptionvalue` встановлюються ті властивості, значення яких можна обирати. При чому в таблиці `Hardwaregeneraloption` знаходяться властивості, які є стандартними, а в таблиці `Hardwarepersonaloptionvalue` – варіанти персоналізації, тобто це особливості самого виробу ручної роботи. Наприклад, розмір є стандартним варіантом вибору, а напис – персональним. Для персональної властивості може встановлюватися ціна. Тобто за різного значення персональної властивості ціна товару ручної роботи може змінюватися. Тоді використовується не базова ціна, яка була задана в таблиці з даними самого товару. У свою чергу стандартні властивості передбачають вибір одного зі значень, але ці значення не впливають на підсумкову вартість товару.

Тож у таблиці `Hardwareproperty` містяться значення властивостей товару ручної роботи, які є незмінними, включаючи до таблиці такі поля в підсумку:

- id – ідентифікатор значення властивості товару ручної роботи;
- hardware – товар, який був створений в результаті ручної роботи та якому визначено цю властивість, визначається записом таблиці `Hardware`, тобто є зовнішнім ключем;

- option_value – значення властивості, яке встановлено для цього товару, який був створений в результаті ручної роботи, що відповідає запису таблиці Optionvalue;

- own_value – власне значення, яке може бути встановлене, якщо для цього товару не є адекватним стандартний вибір за попереднім полем;

- hint – підказка щодо значення цієї властивості (наприклад, вказано матеріал, але це може бути матеріал верху взуття або матеріал устілки).

Властивості товару, створеного в результаті ручної роботи, які можна обирати, але зі стандартних варіантів, знаходяться в таблиці БД Handwaregeneraloption з полями:

- id – ідентифікатор запису про стандартну властивість товару ручної роботи, значення якої можна обирати;

- hardware – товар ручної роботи, якого стосується дана стандартна властивість, визначений відповідним записом таблиці Hardware, тобто є зовнішнім ключем;

- option – стандартна властивість, значення якої можна обирати для даного товару ручної роботи, визначена відповідним записом таблиці Option, тобто є зовнішнім ключем;

- hint – підказка щодо вибору даної стандартної властивості.

Можливі значення відповідних стандартних властивостей з таблиці Handwaregeneraloption представлені в таблиці Handwaregeneraloptionvalue з полями:

- id – ідентифікатор можливого значення загальної властивості товару ручної роботи;

- hardwaregeneraloption – запис про стандартну властивість товару ручної роботи, значення якої можна обирати, що є зв'язком з таблицею Handwaregeneraloption;

- optionvalue – значення цієї стандартної властивості, яке може бути обрано для неї стосовно конкретно цього товару ручної роботи.

Можливі значення, за якими можна виконати персоналізацію товару, представлені в таблиці `Hardwarepersonaloptionvalue` з полями:

- `id` – ідентифікатор можливого значення, за якими можна виконати персоналізацію товару;
- `hardware` – товар, який був створений в результаті ручної роботи та якому визначено це можливе значення, за яким можна виконати персоналізацію, визначається записом таблиці `Hardware`, тобто є зовнішнім ключем;
- `value` – відповідне значення, яке можна обрати стосовно персоналізації товару;
- `price` – вартість товару за даного вибору;
- `availability` – кількість товару, наявна в даному варіанті.

Стандартні властивості, які можна обирати для товарів, знаходяться в таблиці `Option` з полями:

- `id` – ідентифікатор стандартної властивості, що є довідковим варіантом;
- `name` – назва такої стандартної властивості, тобто це наприклад, розмір, колір тощо.

Також довідкові, але вже значення, які можуть приймати ці стандартні властивості, тобто наприклад, для кольору зелений, червоний (або більш розлогі варіанти) представляються в таблиці `Optionvalue` з полями:

- `id` – ідентифікатор можливого значення властивості;
- `option` – властивість, для якої можна обрати таке значення, тобто значення виділяються не просто як загальний каталог, а стосовно конкретної властивості (зеленим може бути колір, а S – розмір);
- `value` – відповідно саме значення, яке може приймати ця властивість товару.

Також можна зазначити, що товар ручної роботи не може приймати якесь власне значення, задане продавцем. Для таких значень є варіанти персоналізації, або вони можуть бути задані в описі товару.

Товари ручної роботи є важливою складовою БД, але вони не існують самі по собі, вони продаються в крамницях, які можуть представляти одного або декількох працівників. Через крамниці і продаються товари ручної роботи. Крамниці представлені в БД таблицею Handwarestore з полями:

- id – ідентифікатор крамниці, через яку продаються товари ручної роботи і яка є точкою доступу до виробництва майстрами;
- store_title – назва крамниці;
- store_description – опис крамниці;
- main_city – місто, де знаходиться крамниця фізично (навіть якщо і не має такої фізичної крамниці, але є приміщення, де зберігаються самі товари) і звідки товар має відправлятися;
- overall_rate – середній рейтинг цієї крамниці на основі загальної оцінки від споживачів;
- service_rate – середній рейтинг цієї крамниці на основі обслуговування, наданого крамницею щодо одного з товарів, від споживачів;
- shipping_rate – середній рейтинг цієї крамниці на основі доставки товару від споживачів;
- quality_rate – середній рейтинг цієї крамниці на основі оцінки якості виготовлення товару від споживачів;
- total_sales – сумарна кількість одиниць всіх товарів ручної роботи даної крамниці, яку було продано за весь час роботи даною крамницею;
- week_sales – сумарна кількість одиниць всіх товарів ручної роботи даної крамниці, яку було продано протягом останнього тижня даною крамницею;
- owner – власник крамниці (або продавець, який може вносити нові дані за крамницею), визначається записом таблиці Selleraccount, тобто є зовнішнім ключем;
- image – фотографія крамниці.

Крамниць у системі може бути багато, але деякі з них можуть бути більш цікавими для споживача. Для того щоб кожен раз не повертатися до

пошуку крамниць, щоб знайти потрібну, можна поставити на магазині шпильку, щоб зберегти його і швидко повернутися потім з переліку таких крамниць до потрібної, вже не виконуючи заново пошук. Збережені шпильки на крамницях, де продаються товари ручної роботи, знаходяться в таблиці БД Handwarestorepin з полями:

- id – ідентифікатор шпильки збереження крамниці споживачем;
- handwarestore – крамниця, через яку продаються товари ручної роботи, визначається записом таблиці Handwarestore, тобто є зовнішнім ключем;
- customeraccount – обліковий запис споживача, який створив для себе цю шпильку, визначається записом таблиці Customeraccount, тобто є зовнішнім ключем;
- position – порядковий номер, за яким ця крамниця відображається у переліку крамниць, на яких було встановлено шпильки.

Рейтинги можуть визначатися споживачами щодо крамниць і щодо товарів ручної роботи. Під рейтингом мається на увазі визначення оцінок крамниці або товару ручної роботи, яке охоплює ряд окремих критеріїв, за якими таці оцінки встановлюються, та загальну оцінку.

Рейтинги крамниць визначаються таблицею Handwarestorenotice з полями:

- id – ідентифікатор рейтингу крамниці;
- customeraccount – обліковий запис споживача, що визначив рейтинг, визначається записом таблиці Customeraccount, тобто є зовнішнім ключем;
- handwarestore – крамниця, за якою створено рейтинг, визначається записом таблиці Handwarestore, тобто є зовнішнім ключем;
- overall_rate – загальна оцінка рейтингу даної крамниці від споживачів;
- service_rate – оцінка рейтингу даної крамниці на основі обслуговування від споживачів;

- shipping_rate – оцінка рейтингу даної крамниці на основі доставки товару від споживачів;
- quality_rate – оцінка рейтингу даної крамниці на основі оцінки якості виготовлення товарів від споживачів;
- text_rate – текстовий коментар до рейтингу даної крамниці;
- date_rate – час, коли було створено даний рейтинг крамниці.

Окрім створення самого рейтингу крамниці споживачем, інші споживачі можуть реагувати на ці рейтинги. Для цього вони визначають позитивну або негативну реакцію на рейтинг, інформуючи про своє ставлення до висловленої інформації. Реакції на рейтинг крамниці заходяться в таблиці Handwarestorenoticeback з полями:

- id – ідентифікатор реакції на рейтинг крамниці, що створений споживачем, іншим споживачем;
- handwarestorenotice – рейтинг крамниці, на який було відреаговано таким чином, визначається записом таблиці Handwarestorenotice, тобто є зовнішнім ключем;
- backtype – тип реакції (позитивна, негативна);
- customeraccount – обліковий запис споживача, що створив цю реакцію на рейтинг, визначається записом таблиці Customeraccount, тобто є зовнішнім ключем.

Рейтинги товарів ручної роботи визначаються таблицею Handwarenotice з полями:

- id – ідентифікатор рейтингу товару ручної роботи;
- customeraccount – обліковий запис споживача, що визначив рейтинг, визначається записом таблиці Customeraccount, тобто є зовнішнім ключем;
- hardware – товар ручної роботи, за яким створено рейтинг, визначається записом таблиці Hardware, тобто є зовнішнім ключем;
- overall_rate – загальна оцінка рейтингу цього товару ручної роботи від споживачів;

- service_rate – оцінка рейтингу даного товару ручної роботи на основі обслуговування від споживачів;
- shipping_rate – оцінка рейтингу даного товару ручної роботи на основі доставки товару від споживачів;
- quality_rate – оцінка рейтингу даного товару ручної роботи на основі оцінки якості виготовлення товару від споживачів;
- fit_rate – оцінка рейтингу даного товару ручної роботи на основі того, на скільки фактично цей товар відповідає його опису та інформації на вебсайті на думку споживача;
- text_rate – текстовий коментар до рейтингу товару ручної роботи;
- date_rate – час, коли було створено даний рейтинг товару ручної роботи.

Реакції на рейтинг товару ручної роботи заходяться в таблиці Handwarenoticeback з полями:

- id – ідентифікатор реакції на рейтинг товару ручної роботи, що створений споживачем, іншим споживачем;
- handwarenotice – рейтинг товару ручної роботи, на який було відреаговано таким чином, визначається записом таблиці Handwarenotice, тобто є зовнішнім ключем;
- backtype – тип реакції (позитивна, негативна);
- customeraccount – обліковий запис споживача, що створив цю реакцію на рейтинг, визначається записом таблиці Customeraccount, тобто є зовнішнім ключем.

Споживачі можуть зберігати товари ручної роботи, які їх зацікавили, для того щоб потім до них повернутися. Таке збереження вноситься в таблицю Handwaresave з полями:

- id – ідентифікатор збереження товару ручної роботи;
- hardware – товар ручної роботи, який було збережено споживачем, визначається записом таблиці Hardware, тобто є зовнішнім ключем;

– `customeraccount` – обліковий запис споживача, що зберіг для себе цей товар ручної роботи, визначається записом таблиці `Customeraccount`, тобто є зовнішнім ключем.

Кожен споживач повинен мати власний обліковий запис у програмі. Споживачі та продавці мають різні облікові записи. Зважаючи на сценарії використання програми, продавець і споживач мають власні підсистеми для роботи з програмою. Тому облікові записи споживачів знаходяться в таблиці `Customeraccount` з полями:

- `id` – ідентифікатор облікового запису споживача;
- `login` – логін споживача, що визначає його обліковий запис;
- `password` – пароль споживача, що визначає його обліковий запис;
- `activity` – активність або неактивність облікового запису споживача на даний момент (може він за ним працювати чи ні);
- `userfrom` – дата, з якої цей споживач має цей обліковий запис у програмі;
- `savedphone` – збережений номер телефону споживача для використання в замовленнях;
- `customer_image` – фотографія споживача.

Облікові записи продавців, які працюють в конкретних крамницях, тобто один продавець за тим же самим обліковим записом не може працювати з різними крамницями, знаходяться в таблиці БД `Selleraccount` з полями:

- `id` – ідентифікатор облікового запису продавця;
- `login` – логін продавця, що визначає його обліковий запис;
- `password` – пароль продавця, що визначає його обліковий запис;
- `activity` – активність або неактивність облікового запису продавця на даний момент (може він за ним працювати чи ні);
- `hardwarestore` – крамниця, в якій працює цей продавець, визначається записом таблиці `Hardwarestore`, тобто є зовнішнім ключем;

– savedphone – номер телефону продавця для його ідентифікації адміністратором, якщо це необхідно.

Замовлення товарів ручної роботи з однієї і тієї самої крамниці визначається в БД таблицею Cart з полями:

- id – ідентифікатор замовлення товарів ручної роботи;
- customeraccount – обліковий запис споживача, який замовив ці товари ручної роботи з крамниці, визначається записом таблиці Customeraccount, тобто є зовнішнім ключем;
- cartdate – час замовлення товарів ручної роботи;
- state – стан замовлення товарів ручної роботи;
- price – сума, яку потрібно сплатити за це замовлення товарів ручної роботи.

Склад замовлення товарів ручної роботи знаходиться в таблиці Carthardware з полями:

- id – ідентифікатор товару ручної роботи з замовлення;
- cart – замовлення товарів ручної роботи, в яке входить як позиція цей товар ручної роботи, визначається записом таблиці Cart, тобто є зовнішнім ключем;
- hardware – товар ручної роботи, який визначає цю позицію замовлення, визначається записом таблиці Hardware, тобто є зовнішнім ключем;
- quantity – кількість екземплярів цього товару ручної роботи, яка потрібна споживачу;
- personalchoice – персональна інформація для цього товару ручної роботи (тоді вона буде однаковою для всіх екземплярів цього товару, кількість яких була зазначена у відповідному полі);
- price – вартість одиниці товару ручної роботи, якою вона була обчислена в замовленні.

Оскільки для товарів ручної роботи можуть визначатися значення певних властивостей, тобто обиратися деякі опції, то для кожного

замовлення стосовно кожного товару має бути зафіксовано, які значення в підсумку обрав споживач. Вказати конкретне значення персонального вибору (напис з іменем, наприклад) можна, але це значення фіксується за конкретним товаром, але далі потрібно обрати варіанти з тих властивостей, які запропоновано споживачу. Визначення обраних споживачем значень стандартних властивостей щодо товару ручної роботи з замовлення відбувається в таблиці `Carthardwaregeneraloption` з полями:

- `id` – ідентифікатор обраного значення стандартної властивості щодо товару ручної роботи з замовлення;
- `carthardware` – товар ручної роботи з замовлення, якому відповідає це значення стандартної властивості, визначається записом таблиці `Carthardware`, тобто є зовнішнім ключем;
- `hardwaregeneraloptionvalue` – значення стандартної властивості щодо товару ручної роботи з замовлення, яке було обрано споживачем, визначається записом таблиці `Hardwaregeneraloptionvalue`, тобто є зовнішнім ключем.

Визначення обраних споживачем значень властивості, якою можна виконати персоналізацію, щодо товару ручної роботи з замовлення відбувається в таблиці `Carthardwarepersonaloption` з полями:

- `id` – ідентифікатор обраного значення властивості, якою можна виконати персоналізацію, щодо товару ручної роботи з замовлення;
- `carthardware` – товар ручної роботи з замовлення, якому відповідає це значення властивості, якою можна виконати персоналізацію, визначається записом таблиці `Carthardware`, тобто є зовнішнім ключем;
- `hardwaregeneraloptionvalue` – значення властивості щодо товару ручної роботи з замовлення, якою можна виконати персоналізацію і яке було обрано споживачем, визначається записом таблиці `Hardwaregeneraloptionvalue`, тобто є зовнішнім ключем.

Для більш зручного пошуку товарів ручної роботи створені теги, які знаходяться в таблиці `Tag` з полями:

- id – ідентифікатор тегу, який може використовуватися для того, щоб охарактеризувати товар ручної роботи;

- name – назва тегу, який може використовуватися для того, щоб охарактеризувати товар ручної роботи.

Зв'язування тегів з конкретними товарами ручної роботи відбувається через таблицю БД Handwaretag з полями:

- id – ідентифікатор товару ручної роботи, якому визначено певний тег;

- hardware – товар ручної роботи, якому встановлено у відповідність тег, визначається записом таблиці Hardware, тобто є зовнішнім ключем;

- tag – тег, який виділено для товару ручної роботи, визначається записом таблиці Tag, тобто є зовнішнім ключем.

Звичайно, що споживачі можуть потребувати уточнити певні питання в крамниці щодо тих чи інших товарів, щодо замовлення, які вони створили, або самі продавці мають звернутися до споживачів щодо деякого замовлення, тому таку роботу було впорядковано, для чого спочатку виділено самі запити, на основі яких таку взаємодію між крамницями та споживачами організовано. Це таблиця Storequery з полями:

- id – ідентифікатор запиту щодо взаємодії між споживачем та крамницею;

- hardwarestore – крамниця, працівники якої можуть брати участь у спілкуванні, визначається записом таблиці Hardwarestore, тобто є зовнішнім ключем;

- customeraccount – споживач, який бере участь у спілкуванні, визначається записом таблиці Customeraccount, тобто є зовнішнім ключем.

Повідомлення зі взаємодії між крамницею та споживачем за самим запитом зберігаються в таблиці Storequerymessage з полями:

- id – ідентифікатор повідомлення, відісланого за запитом щодо взаємодії між споживачем та крамницею;

- storequery – запит щодо взаємодії між споживачем та крамницею, за яким відіслано повідомлення, визначається записом таблиці Storequery, тобто є зовнішнім ключем;
- message – текст повідомлення, що було направлено;
- source – джерело повідомлення, тобто хто його створив (продавець крамниці чи споживач);
- saved_date – час, коли було створено повідомлення;
- seen – бачив це повідомлення споживач або відповідно один з продавців крамниці, яким було відправлено це повідомлення (рис. 3.3).

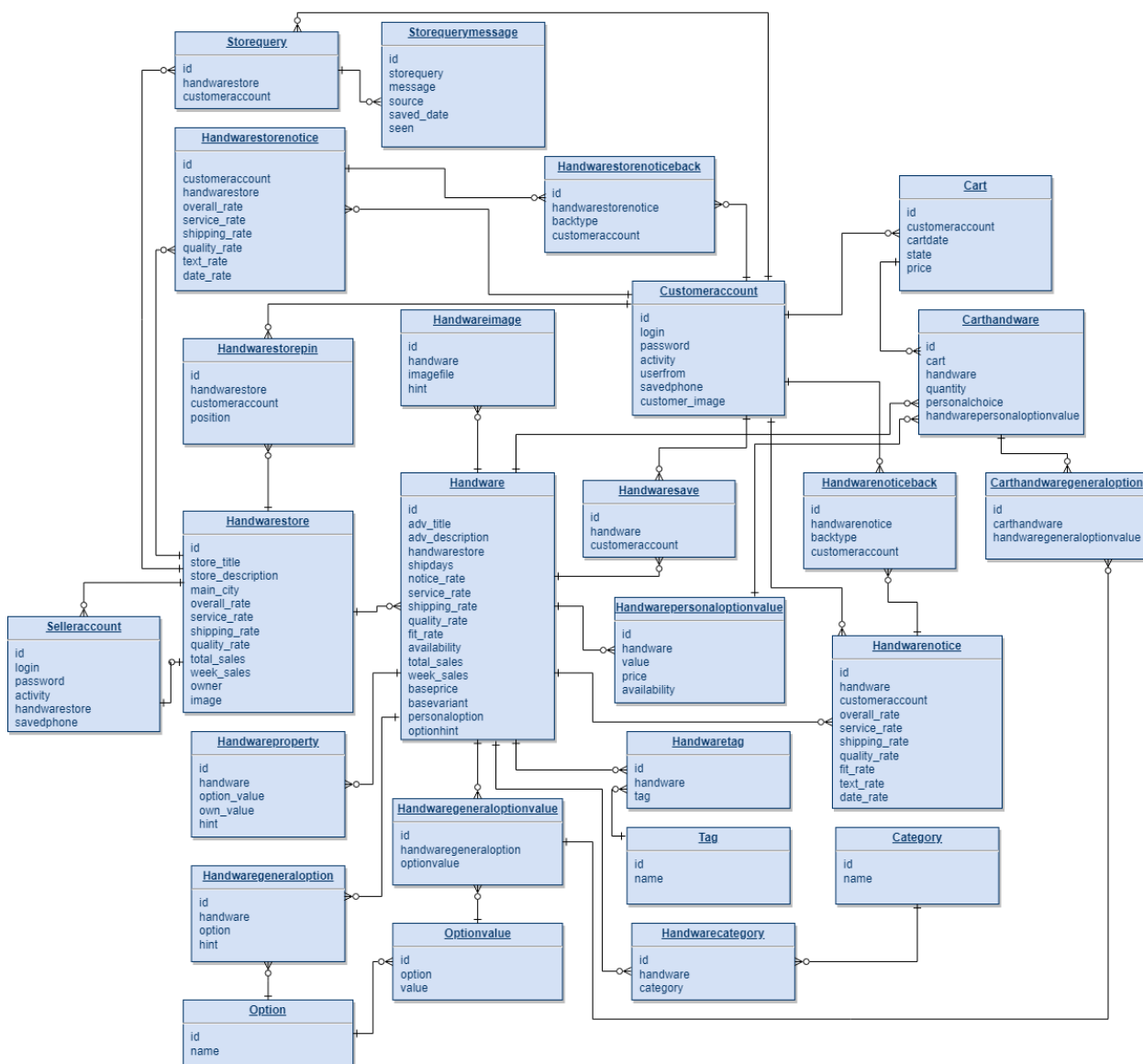


Рисунок 3.3 – Схема реляційної БД

Категорії товарів ручної роботи знаходяться в таблиці Category з полями:

- id – ідентифікатор категорії товарів ручної роботи;
- name – назва категорії товару ручної роботи.

Зв'язування категорії з самими товарами ручної роботи, яким вони відповідають, реалізовано через таблицю Handwarecategory з полями:

- id – ідентифікатор товару ручної роботи, якому визначено категорію, до якої він належить;
- hardware – товар ручної роботи, який охарактеризований цією категорією, визначається записом таблиці Hardware, тобто є зовнішнім ключем;
- category – категорія, яку виділено для товару ручної роботи, визначається записом таблиці Category, тобто є зовнішнім ключем.

Реалізацію роботи з БД виконано через використання MySQL.

3.4 Структура програми

При визначенні структури (рис. 3.4) програми було враховано розділення сценаріїв використання програми між споживачами (підсистема споживача) і продавцями (підсистема керування крамницями).

Окремо було виділено у відповідну підсистему реалізацію визначення популярності товарів ручної роботи на основі запропонованого методу, що фактично виконується на основі окремо створеної моделі та реалізованої послідовності етапів.

Підсистема визначення популярності товарів ручної роботи реагує на звернення підсистеми керування крамницями, тобто вона не є ініціатором, і за запитом виконує необхідне визначення на основі навченої моделі і повертає результат.

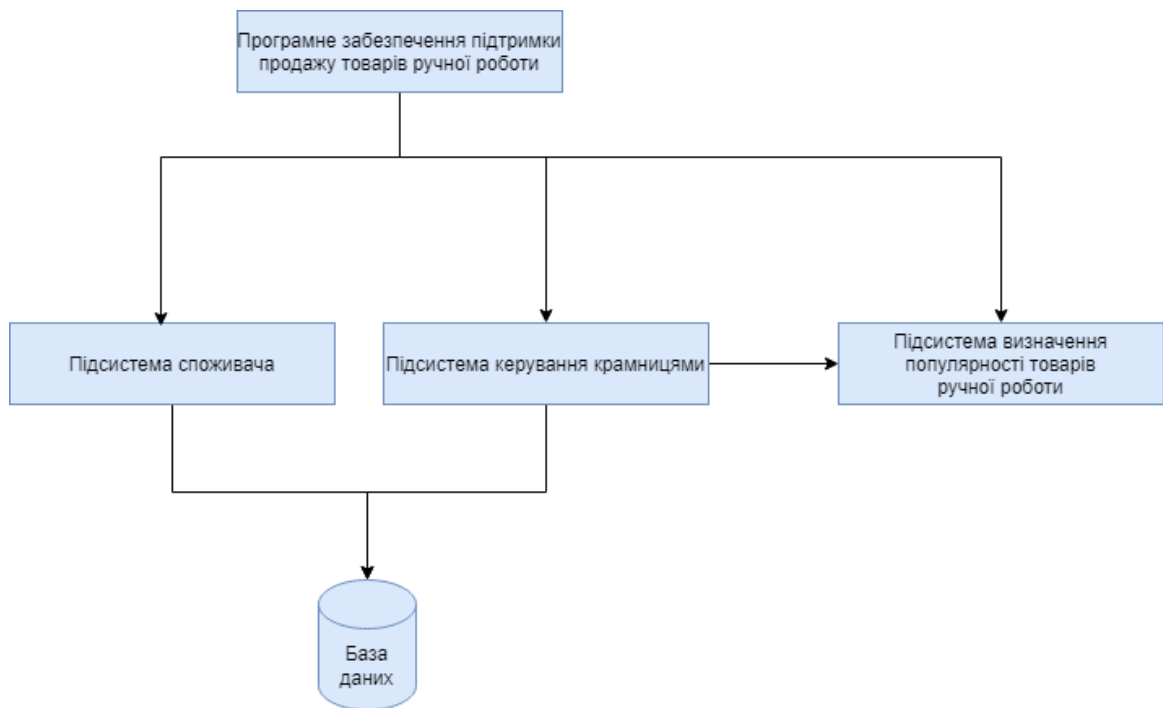


Рисунок 3.4 – Схема структури програмного забезпечення підтримки продажу товарів ручної роботи

3.5 Висновки за розділом 3

У роботі при виконанні третього розділу було виявлено функціональні вимоги і сценарії роботи споживачів та продавців з програмою, виконано вибір фреймворку для розробки, визначено структуру БД.

4 РЕАЛІЗАЦІЯ ПРОГРАМИ

4.1 Визначення послідовності роботи з програмою

Оскільки основні функції програми передбачають відображення певної інформації користувачу, наприклад, щодо крамниць, товарів ручної роботи, а відмінними діями фактично є дії щодо замовлень товарів ручної роботи, то важливо визначити порядок такої роботи (рис. 4.1) та можливі стани, які розпочинаються зі стану нового, а далі за успішної роботи можуть ставати підтвердженим, відправленим, виконаним. У випадку необхідності визначення некоректних дій, відсутності товарів, відмови від замовлення воно переходить до стану скасованого. Також до цього стану замовлення може перевести і сам споживач, але тільки тоді, коли воно ще не було підтверджено продавцем.

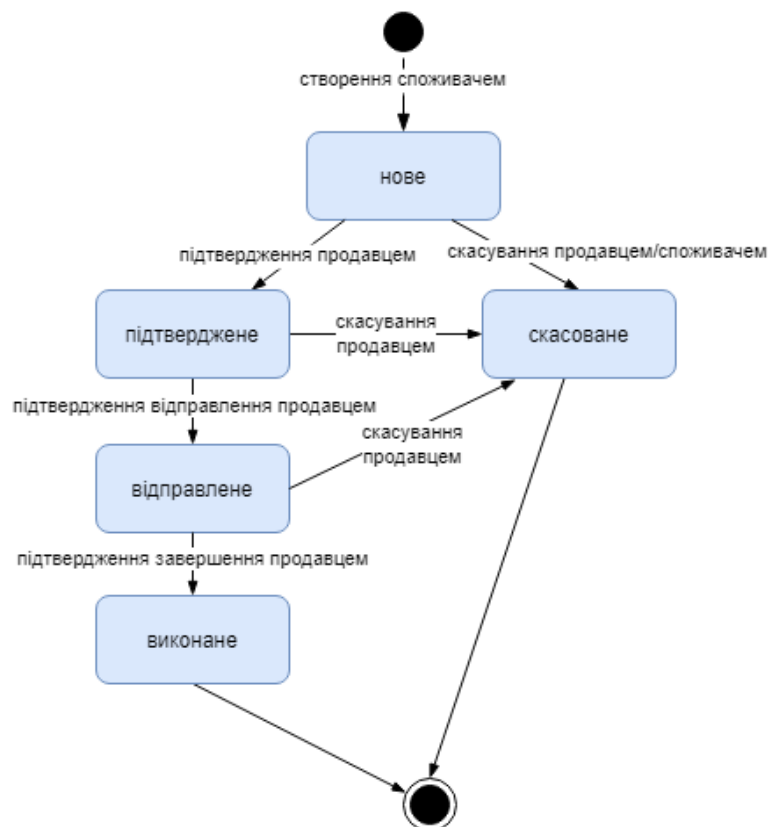


Рисунок 4.1 – Діаграма станів замовлення товарів ручної роботи

4.2 Особливості реалізації програми

Нижче описані основні створені одиниці програми, обумовлені структурою програми як Django-застосунку.

Функція `best_category_hardware_view` реалізує відображення найкращих товарів ручної роботи за категорією `categ`, яка є другим параметром функції, або використовуючи дані всіх товарів, якщо значення третього параметра `old` вказане в `True`, або тільки тих, які є в наявності, якщо значення другого параметра `old` вказане в `False`. Найкращі товари визначаються за загальною оцінкою, тобто у відповідності з полем `notice_rate`. Першим параметром функції є `request`, який визначає дані самого запиту до сторінки. Такий параметр є першим у всіх наступних функціях. Шаблон відповідної вебсторінки та наступних вебсторінок реалізований через використання безкоштовного шаблону [9].

Функція `hardwarestore_hardware_view` реалізує відображення товарів ручної роботи, що продаються в крамниці `hardwarestore` (параметр задає номер крамниці), виконуючи сортування товарів у напрямку `sortway` (0 – за зменшенням вартості товару, 1 – за збільшенням вартості товарів, 2 – за назвою товару, 3 – за кількістю екземплярів товару, що доступна для продажу), що є другим параметром функції.

Функція `sold_hardware_view` реалізує відображення товарів ручної роботи, що найактивніше продаються, визначаючи через другий параметр `way` напрямок впорядкування: 0 – кількість продажів за весь час, 1 – кількість продажів за останній тиждень.

Клас `HardwaresaveView` реалізує збереження товару ручної роботи, основна підтримка виконується методом `post` з параметром `pk`, що визначає номер товару ручної роботи, який зберігається.

Клас `HardwaresaveDeleteView` реалізує вилучення збереження товару ручної роботи, основна підтримка виконується методом `post` з параметром `pk`,

що визначає номер товару ручної роботи, який раніше був збережений, а тепер має бути вилючений зі збережених.

Функція `saved_handwares_view` реалізує відображення збережених товарів ручної роботи.

Функція `popular_category_handware_view` реалізує відображення найпопулярніших товарів ручної роботи за категорією `categ`, яка є другим параметром функції, або використовуючи дані всіх товарів, якщо значення третього параметра `old` вказане в `True`, або тільки тих, які є в наявності, якщо значення другого параметра `old` вказане в `False`. Найкращі товари визначаються за загальною оцінкою, тобто у відповідності з полем `notice_rate`. Популярність визначається за кількістю рейтингів, які було визначено споживачами для такої крамниці. За кожним товаром для авторизованого споживача вноситься додаткова інформація про те, чи зберіг цей споживач цей товар ручної роботи, яку оцінку поставив у рейтингу споживач цьому товару під час останнього разу, коли цей споживач створював рейтинг цьому товару, створення шпильки за крамницею, яка продає цей товар, кількість рейтингів. Аналогічні дії підтримуються і у всіх інших функціях. Тобто якщо відображаються товари ручної роботи або крамниці, то для кожної з них контролюються параметри, пов'язані з авторизованим споживачем. Якщо він зберіг до себе ці дані так чи інакше, якщо він оцінив цю сутність через створення рейтингу, то відповідні результати відображаються одразу або при виборі відповідного контекстного меню, а в цій або подібній функції ці дані заповнюються.

Функція `popular_handwarestore_handware_view` реалізує відображення найпопулярніших товарів ручної роботи за крамницею, номер якої має бути переданий через другий параметр `handwarestore` до функції, або використовуючи дані всіх товарів, якщо значення третього параметра `old` вказане в `True`, або тільки тих, які є в наявності, якщо значення другого параметра `old` вказане в `False`. Популярність визначається за кількістю рейтингів, які було визначено споживачами для такої крамниці.

Функція `search_handwarestore_view` реалізує пошук крамниць, які продають товари ручної роботи, а функція `search_handware_view` реалізує пошук товарів ручної роботи з встановленням відповідних параметрів пошуку.

Клас `HardwarestorepinView` реалізує створення шпильки на крамниці, основна підтримка виконується методом `post` з параметром `pk`, що визначає номер крамниці, за якою створюється шпилька.

Клас `HardwarestorepinDeleteView` реалізує прибирання шпильки з крамниці, основна підтримка виконується методом `post` з параметром `pk`, що визначає номер крамниці, за якою прибирається шпилька.

Функція `pinned_handwarestores_view` реалізує відображення крамниць, на яких залишено шпильки авторизованим в цей момент споживачем.

Функція `cart_view` реалізує замовлення товарів ручної роботи, функція `customer_carts_view` реалізує відображення замовлень товарів ручної роботи авторизованим споживачем, а функція `cancel_cart` реалізує скасування замовлення товарів ручної роботи з номером `cart`, що задається параметром.

Функція `handware_view` реалізує відображення інформації про товар ручної роботи з визначенням через параметр `handware` номеру товару ручної роботи. Стосовно товару ручної роботи надаються дані з таблиці товару з БД, всі фотографії, категорії, кількість рейтингів, останні 10 рейтингів, властивості, які можуть встановлюватися або які задані одразу, теги.

Функція `handwarestore_view` реалізує відображення інформації про крамницю, що продає товари ручної роботи, яка задається своїм номером через параметр `store`. До інформації про крамницю входять самі дані з таблиці БД, 10 найкращих товарів ручної роботи за загальною оцінкою рейтингу, що продаються в магазині, 10 останніх рейтингів споживачів стосовно цієї крамниці, а також персональна інформація для авторизованого споживача стосовно товарів ручної роботи, які було виділено, та рейтингів.

Функція `add_handware_notice_view` реалізує рейтингове оцінювання товару ручної роботи з перевіркою того, що споживач замовляв раніше такий

товар у програмі, після чого тільки надається доступ до форми `Hardwarenoticeform`. Функція `add_handwarestore_notice_view` реалізує рейтингове оцінювання крамниці з перевіркою того, що споживач раніше замовляв один з товарів у цій крамниці.

Функція `handwarestore_notices_view` реалізує відображення рейтингових оцінювань крамниці з номером `store`.

Функція `hardware_notices_view` реалізує відображення рейтингових оцінювань товару ручної роботи з номером `hardware`. Рейтинги впорядковуються за часом створення.

Клас `HardwarenoticePositiveView` реалізує позитивне реагування на рейтингове оцінювання товару ручної роботи, а `HardwarenoticeNegativeView` – негативне.

Клас `HardwarestorenoticePositiveView` реалізує позитивне реагування на рейтингове оцінювання крамниці, а `HardwarestorenoticeNegativeView` – негативне.

Функція `best_handwarestores_view` реалізує відображення найкращих крамниць за містом `storecity`. Сортування відбувається за полем `overall_rate` для визначення найкращих крамниць, тобто за загальною оцінкою з рейтингів.

Функція `best_handwarestore_handware_view` реалізує відображення найкращих товарів ручної роботи за крамницею `handwarestore`, що є другим параметром функції і визначається номером. Для відображення використовуються або дані всіх товарів, якщо значення третього параметра `old` вказане в `True`, або тільки тих, які є в наявності, якщо значення другого параметра `old` вказане в `False`. Найкращі товари визначаються за загальною оцінкою, тобто у відповідності з полем `notice_rate`.

Функція `popular_handwarestores_view` реалізує відображення найпопулярніших крамниць. Популярність визначається за кількістю поданих рейтингів.

Функція `message_view` реалізує відправлення повідомлення між крамницею `hardwarestore` та споживачем `customer`, при чому значення параметра `customer` може бути пропущене, тоді використовуються дані авторизації. Окрім форми відправлення повідомлення надаються останні повідомлення відправлені за цим запитом.

Функція `customer_hardware_notices_view` реалізує відображення рейтингових оцінювань, які споживач вважає для себе корисними. Витягаються ті дані, які стосуються рейтингів, на які позитивно відреагував авторизований у програмі споживач.

Функція `best_tag_hardware_view` реалізує відображення найкращих товарів ручної роботи за тегом `tag`, що є другим параметром функції. Для відображення використовуються або дані всіх товарів, якщо значення третього параметра `old` вказане в `True`, або тільки тих, які є в наявності, якщо значення другого параметра `old` вказане в `False`. Найкращі товари визначаються за загальною оцінкою, тобто у відповідності з полем `notice_rate`.

Клас `Hardware` реалізує модель для забезпечення роботи з товарами, які є результатом ручної роботи, включає такі поля:

- поле `adv_title` – назва оголошення, в якому міститься інформація про вручну виготовлений товар, має тип поля `CharField`;
- поле `adv_description` – рекламний текст опису з деталями щодо даного товару ручної роботи з представленням всіх особливостей, які має даний товар, що надав його продавець, має тип поля `CharField`;
- поле `hardwarestore` – крамниця, в якій продається цей товар, створений в результаті ручної роботи, визначається посиланням за допомогою `ForeignKey` на модель `Hardwarestore`;
- поле `shipdays` – кількість днів на поточний момент, через яку продавець очікує відправити цей товар, має тип поля `SmallIntegerField`;
- поле `notice_rate` – середній рейтинг цього товару ручної роботи на основі загальної оцінки від споживачів, має тип поля `FloatField`;

- поле `service_rate` – середній рейтинг цього товару ручної роботи на основі обслуговування, наданого щодо цього товару, від споживачів, має тип поля `FloatField`;
- поле `shipping_rate` – середній рейтинг цього товару ручної роботи на основі доставки товару від споживачів, має тип поля `FloatField`;
- поле `quality_rate` – середній рейтинг цього товару ручної роботи на основі оцінки якості виготовлення товару від споживачів, має тип поля `FloatField`;
- поле `fit_rate` – середній рейтинг цього товару ручної роботи на основі оцінки того, на скільки фактично цей товар відповідає його опису та інформації на вебсайті, від споживачів, має тип поля `FloatField`;
- поле `availability` – кількість екземплярів товару ручної роботи, що може замовити споживач на даний момент, має тип поля `SmallIntegerField`;
- поле `total_sales` – сумарна кількість одиниць даного товару ручної роботи, яку було продано за весь час роботи з ним даної крамниці, має тип поля `IntegerField`;
- поле `week_sales` – сумарна кількість одиниць даного товару ручної роботи, яку було продано протягом останнього тижня, має тип поля `IntegerField`;
- поле `baseprice` – базова, тобто основна вартість даного товару, яка використовується тоді, коли не можна встановлювати персональні характеристики, має тип поля `FloatField`;
- поле `basevariant` – можливість продавати цей товар ручної роботи без встановлення персональних характеристик (наприклад, написів) або неможливість це робити, має тип поля `BooleanField`;
- поле `personaloption` – назва персонального параметра, значення для якого можна вибрати, має тип поля `CharField`;
- поле `optionhint` – підказка щодо суті даного персонального параметра, тобто що він позначає, має тип поля `CharField`;

- tags – перелік тегів, які відповідають даному товару ручної роботи, представляє реалізацію зв'язку багато до багатьох у БД, а тому реалізована полем ManyToManyField до моделі Tag;

- categories – перелік категорій, до яких належить даний товар ручної роботи, представляє реалізацію зв'язку багато до багатьох у БД, а тому реалізована полем ManyToManyField до моделі Category.

Фотографії, за якими можна візуально отримати інформацію про товар, який виготовляється вручну, реалізовані моделлю Hardwareimage, що включає такі атрибути:

- поле hardware – товар ручної роботи, якого стосується ця фотографія, визначається посиланням за допомогою ForeignKey на модель Hardware;

- поле imagefile – фотографія, має тип поля ImageField;

- поле hint – підказка щодо тих особливостей товару, який був створений в результаті ручної роботи і якого стосується ця підказка, має тип поля CharField.

Клас Hardwareproperty реалізує модель значень властивостей товару ручної роботи, які є незмінними, включаючи такі атрибути:

- поле hardware – товар, який був створений в результаті ручної роботи та якому визначено цю властивість, визначається посиланням за допомогою ForeignKey на модель Hardware;

- поле option_value – значення властивості, яке встановлено для цього товару, який був створений в результаті ручної роботи, визначається посиланням за допомогою ForeignKey на модель Optionvalue;

- поле own_value – власне значення, яке може бути встановлене, якщо для цього товару не є адекватним стандартний вибір за попереднім полем, має тип поля CharField;

- поле hint – підказка щодо значення цієї властивості, має тип поля CharField.

Властивість товару, створеного в результаті ручної роботи, яку можна обирати, але зі стандартних варіантів, визначена класом моделі `Hardwaregeneraloption` з атрибутами:

- поле `hardware` – товар ручної роботи, якого стосується дана стандартна властивість, визначається посиланням за допомогою `ForeignKey` на модель `Hardware`;

- поле `option` – стандартна властивість, значення якої можна обирати для даного товару ручної роботи, визначається посиланням за допомогою `ForeignKey` на модель `Option`;

- поле `hint` – підказка щодо вибору даної стандартної властивості, має тип поля `CharField`;

- `optionvalues` – перелік значень, які може приймати ця стандартна властивість для даного товару ручної роботи, представляє реалізацію зв'язку багато до багатьох у БД, а тому реалізована полем `ManyToManyField` до моделі `Optionvalue`.

Можливі значення, за якими можна виконати персоналізацію товару, реалізовані моделлю `Hardwarepersonaloptionvalue` з атрибутами:

- поле `hardware` – товар, який був створений в результаті ручної роботи та якому визначено це можливе значення, за яким можна виконати персоналізацію, визначається посиланням за допомогою `ForeignKey` на модель `Hardware`;

- поле `value` – відповідне значення, яке можна обрати стосовно персоналізації товару, має тип поля `CharField`;

- поле `price` – вартість товару за даного вибору, має тип поля `FloatField`;

- `availability` – кількість товару, наявна в даному варіанті, має тип поля `PositiveIntegerField`.

Стандартні властивості, які можна обирати для товарів, визначені класом моделі `Option` з атрибутом `name` – назва такої стандартної властивості, має тип поля `CharField`.

Значення стандартних властивостей знаходяться в класі моделі Optionvalue з атрибутами:

- поле option – властивість, для якої можна обрати таке значення, визначається посиланням за допомогою ForeignKey на модель Option;
- поле value – відповідно саме значення, яке може приймати ця властивість товару, має тип поля CharField.

Модель крамниці Handwarestore представлена атрибутами:

- поле store_title – назва крамниці, має тип поля CharField;
- поле store_description – опис крамниці; , має тип поля CharField
- поле main_city – місто, де знаходиться крамниця фізично і звідки товар має відправлятися, має тип поля CharField;
- поле overall_rate – середній рейтинг цієї крамниці на основі загальної оцінки від споживачів, має тип поля FloatField;
- поле service_rate – середній рейтинг цієї крамниці на основі обслуговування, наданого крамницею щодо одного з товарів, від споживачів, має тип поля FloatField;
- поле shipping_rate – середній рейтинг цієї крамниці на основі доставки товару від споживачів, має тип поля FloatField;
- поле quality_rate – середній рейтинг цієї крамниці на основі оцінки якості виготовлення товару від споживачів, має тип поля FloatField;
- поле total_sales – сумарна кількість одиниць всіх товарів ручної роботи даної крамниці, яку було продано за весь час роботи даною крамницею, має тип поля IntegerField;
- поле week_sales – сумарна кількість одиниць всіх товарів ручної роботи даної крамниці, яку було продано протягом останнього тижня даною крамницею, має тип поля IntegerField;
- owner – власник крамниці (або продавець, який може вносити нові дані за крамницею), визначається посиланням за допомогою ForeignKey на модель Selleraccount;
- image – фотографія крамниці, має тип поля ImageField.

Встановлені шпильки на крамницях реалізовані класом моделі `Hardwarestorepin` з атрибутами:

- поле `hardwarestore` – крамниця, через яку продаються товари ручної роботи, визначається посиланням за допомогою `ForeignKey` на модель `Hardwarestore`;
- поле `customeraccount` – обліковий запис споживача, який створив для себе цю шпильку, визначається посиланням за допомогою `ForeignKey` на модель `Customeraccount`;
- поле `position` – порядковий номер, за яким ця крамниця відображається у переліку крамниць, на яких було встановлено шпильки, має тип поля `SmallIntegerField`.

Рейтинги крамниць визначаються класом моделі `Hardwarestorenotice` з атрибутами:

- поле `customeraccount` – обліковий запис споживача, що визначив рейтинг, визначається посиланням за допомогою `ForeignKey` на модель `Customeraccount`;
- поле `hardwarestore` – крамниця, за якою створено рейтинг, визначається посиланням за допомогою `ForeignKey` на модель `Hardwarestore`;
- поле `overall_rate` – загальна оцінка рейтингу даної крамниці від споживачів, має тип поля `SmallIntegerField`;
- поле `service_rate` – оцінка рейтингу даної крамниці на основі обслуговування від споживачів, має тип поля `SmallIntegerField`;
- поле `shipping_rate` – оцінка рейтингу даної крамниці на основі доставки товару від споживачів, має тип поля `SmallIntegerField`;
- поле `quality_rate` – оцінка рейтингу даної крамниці на основі оцінки якості виготовлення товарів від споживачів, має тип поля `SmallIntegerField`;
- поле `text_rate` – текстовий коментар до рейтингу даної крамниці, має тип поля `CharField`;
- поле `date_rate` – час, коли було створено даний рейтинг крамниці, має тип поля `DateTimeField`.

Модель реакцій на рейтинг крамниці реалізована класом `Hardwarestorenoticeback` з атрибутами:

- поле `hardwarestorenotice` – рейтинг крамниці, на який було відреаговано таким чином, визначається посиланням за допомогою `ForeignKey` на модель `Hardwarestorenotice`;
- поле `backtype` – тип реакції, має тип поля `BooleanField`;
- поле `customeraccount` – обліковий запис споживача, що створив цю реакцію на рейтинг, визначається посиланням за допомогою `ForeignKey` на модель `Customeraccount`.

Рейтинги товарів ручної роботи реалізовано класом моделі `Hardwarenotice` з атрибутами:

- поле `customeraccount` – обліковий запис споживача, що визначив рейтинг, визначається посиланням за допомогою `ForeignKey` на модель `Customeraccount`;
- поле `hardware` – товар ручної роботи, за яким створено рейтинг, визначається посиланням за допомогою `ForeignKey` на модель `Hardware`;
- поле `overall_rate` – загальна оцінка рейтингу цього товару ручної роботи від споживачів, має тип поля `SmallIntegerField`;
- поле `service_rate` – оцінка рейтингу даного товару ручної роботи на основі обслуговування від споживачів, має тип поля `SmallIntegerField`;
- поле `shipping_rate` – оцінка рейтингу даного товару ручної роботи на основі доставки товару від споживачі, має тип поля `SmallIntegerField`;
- поле `quality_rate` – оцінка рейтингу даного товару ручної роботи на основі оцінки якості виготовлення товару від споживачів, має тип поля `SmallIntegerField`;
- поле `fit_rate` – оцінка рейтингу даного товару ручної роботи на основі того, на скільки фактично цей товар відповідає його опису та інформації на вебсайті на думку споживача, має тип поля `SmallIntegerField`;
- поле `text_rate` – текстовий коментар до рейтингу товару ручної роботи, має тип поля `CharField`;

- поле `date_rate` – час, коли було створено даний рейтинг товару ручної роботи, має тип поля `DateTimeField`.

Реакції на рейтинг товару ручної роботи реалізовані класом моделі `Handwarenoticeback` з атрибутами:

- поле `handwarenotice` – рейтинг товару ручної роботи, на який було відреаговано таким чином, визначається посиланням за допомогою `ForeignKey` на модель `Handwarenotice`;

- поле `backtype` – тип реакції, має тип поля `BooleanField`;

- поле `customeraccount` – обліковий запис споживача, що створив цю реакцію на рейтинг, визначається посиланням за допомогою `ForeignKey` на модель `Customeraccount`.

Облікові записи споживачів реалізовані класом моделі `Customeraccount` з атрибутами:

- поле `login` – логін споживача, що визначає його обліковий запис, має тип поля `CharField`;

- поле `password` – пароль споживача, що визначає його обліковий запис, має тип поля `CharField`;

- поле `activity` – активність або неактивність облікового запису споживача на даний момент, має тип поля `BooleanField`;

- поле `userfrom` – дата, з якої цей споживач має цей обліковий запис у програмі, має тип поля `DateField`;

- поле `savedphone` – збережений номер телефону споживача для використання в замовленнях, має тип поля `CharField`;

- поле `customer_image` – фотографія споживача, має тип поля `ImageField`;

- поле `handwaresaves` – перелік збережених товарів ручної роботи споживачем, реалізований полем `ManyToManyField` до моделі `Optionvalue`.

Облікові записи продавців реалізовані моделлю `Selleraccount` з атрибутами:

- поле login – логін продавця, що визначає його обліковий запис, має тип CharField;
- поле password – пароль продавця, що визначає його обліковий запис, має тип CharField;
- поле activity – активність або неактивність облікового запису продавця на даний момент, має тип BooleanField;
- поле hardwarestore – крамниця, в якій працює цей продавець, визначається посиланням за допомогою ForeignKey на модель Hardwarestore;
- поле savedphone – номер телефону продавця для його ідентифікації адміністратором, якщо це необхідно, має тип BooleanField.

Модель замовлення товарів ручної роботи з однієї і тієї самої крамниці реалізована класом Cart з атрибутами:

- поле customeraccount – обліковий запис споживача, який замовив ці товари ручної роботи з крамниці, визначається посиланням за допомогою ForeignKey на модель Customeraccount;
- поле cartdate – час замовлення товарів ручної роботи, має тип поля DateTimeField;
- поле state – стан замовлення товарів ручної роботи, має тип поля SmallIntegerField;
- поле price – сума, яку потрібно сплатити за це замовлення товарів ручної роботи, має тип поля SmallIntegerField.

Модель складу замовлення товарів ручної роботи реалізована класом Carthardware з атрибутами:

- поле cart – замовлення товарів ручної роботи, в яке входить як позиція цей товар ручної роботи, визначається посиланням за допомогою ForeignKey на модель Cart;
- поле hardware – товар ручної роботи, який визначає цю позицію замовлення, визначається визначається посиланням за допомогою ForeignKey на модель Hardware;

- поле `quantity` – кількість екземплярів цього товару ручної роботи, яка потрібна споживачу, має тип поля `SmallIntegerField`;
- поле `personalchoice` – персональна інформація для цього товару ручної роботи, має тип поля `CharField`;
- поле `price` – вартість одиниці товару ручної роботи, якою вона була обчислена в замовленні, має тип поля `FloatField`;
- поле `hardwaregeneraloptionvalues` – перелік значень, які обрано для даного товару ручної роботи споживачем стосовно стандартних властивостей, представляє реалізацію зв'язку багато до багатьох у БД, а тому реалізована полем `ManyToManyField` до моделі `Hardwaregeneraloptionvalue`;
- поле `hardwaregeneraloptionvalue` – перелік значень, які обрано для даного товару ручної роботи споживачем стосовно стандартних властивостей, представляє реалізацію зв'язку багато до багатьох у БД, а тому реалізована полем `ManyToManyField` до моделі `Hardwaregeneraloptionvalue`;
- поле `hardwarepersonaloptionvalue` – значення властивості, за якою виконується персоналізація, яке було обрано споживачем для даного товару з замовлення, визначається визначається посиланням за допомогою `ForeignKey` на модель `Hardwarepersonaloptionvalue`.

Модель тегів товарів ручної роботи визначена класом `Tag` з єдиним атрибутом `name`, що задає назву тегу, який може використовуватися для того, щоб охарактеризувати товар ручної роботи, має тип поля `CharField`.

Модель категорії товарів ручної роботи визначена класом `Category` з єдиним атрибутом `name`, що задає назву категорії товару ручної роботи.

Запити, на основі яких взаємодію між крамницями та споживачами організовано, реалізовано моделлю `Storequery` з атрибутами:

- поле `hardwarestore` – крамниця, працівники якої можуть брати участь у спілкуванні, визначається посиланням за допомогою `ForeignKey` на модель `Hardwarestore`;

- поле `customeraccount` – споживач, який бере участь у спілкуванні, визначається посиланням за допомогою `ForeignKey` на модель `Customeraccount`.

Повідомлення зі взаємодії між крамницею та споживачем за самим запитом реалізовані класом моделі `Storequerymessage` з атрибутами:

- поле `storequery` – запит щодо взаємодії між споживачем та крамницею, за яким відіслано повідомлення, визначається посиланням за допомогою `ForeignKey` на модель `Storequery`;

- поле `message` – текст повідомлення, що було направлено, має тип поля `CharField`;

- поле `source` – джерело повідомлення, тобто хто його створив, має тип поля `BooleanField`;

- поле `saved_date` – час, коли було створено повідомлення, має тип поля `DateTimeField`;

- поле `seen` – бачив це повідомлення споживач або відповідно один з продавців крамниці, яким було відправлено це повідомлення, має тип поля `BooleanField`.

4.3 Висновки за розділом 4

У роботі при виконанні четвертого розділу було визначено стани замовлення товарів ручної роботи, порядок зміни цих станів, описано ті класи і функції, які реалізовано для забезпечення роботи підсистем програмного забезпечення.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Програма призначена для продажу товарів, які є результатом ручної роботи, крамницями, які представляють майстрів і яких представляють відповідно продавці, та придбання товарів споживачами.

5.2 Виконання програми

Виконання програми забезпечено окремо для споживача і окремо для продавця. Інтерфейси представлені окремими адресами, звернення відбуваються окремо. Однак меню сторінок складається з тих самих розділів, хоча і пункти в розділах меню не повністю повторюються, а загалом відповідають тим сценаріям, які були розглянуті в розділі 3.

Усі функції розділені на розділи меню Товари, Крамниці, Замовлення, Повідомлення.

Товари містять пункти для пошуку товарів, для відображення переліків найпопулярніших та найкращих (далі на сторінці можна обрати один з варіантів визначення найкращих товарів: за категоріями, за тегами), переліку збережених товарів. Під час пошуку товарів задається назва товару, місто, де товар має знаходитись, проміжок вартості одиниці, проміжок рейтингу, кількість днів, через яку максимально має бути відвантажено замовлення, категорії товарів, теги, властивості та їх значення (властивості, значення яких необхідно контролювати, обираються з переліку і додаються до фільтрів пошуку). Результати пошуку (рис. 5.1) показуються на окремій сторінці, за кожним товаром демонструється його фотографія, місто, де знаходиться крамниця, кількість днів до відправлення, вартість одиниці, оцінка рейтингу, яку виставив сам цей споживач (якщо не виставляв, то біля іконки людини не міститься значення), назва оголошення, крамниця, в якій

цей товар продається (біля назви крамниці розташовується шпилька, що позначає збереження крамниці для повторної роботи, а якщо відображається плюс, то це означає, що крамниця не збережена і можна шпильку створити).

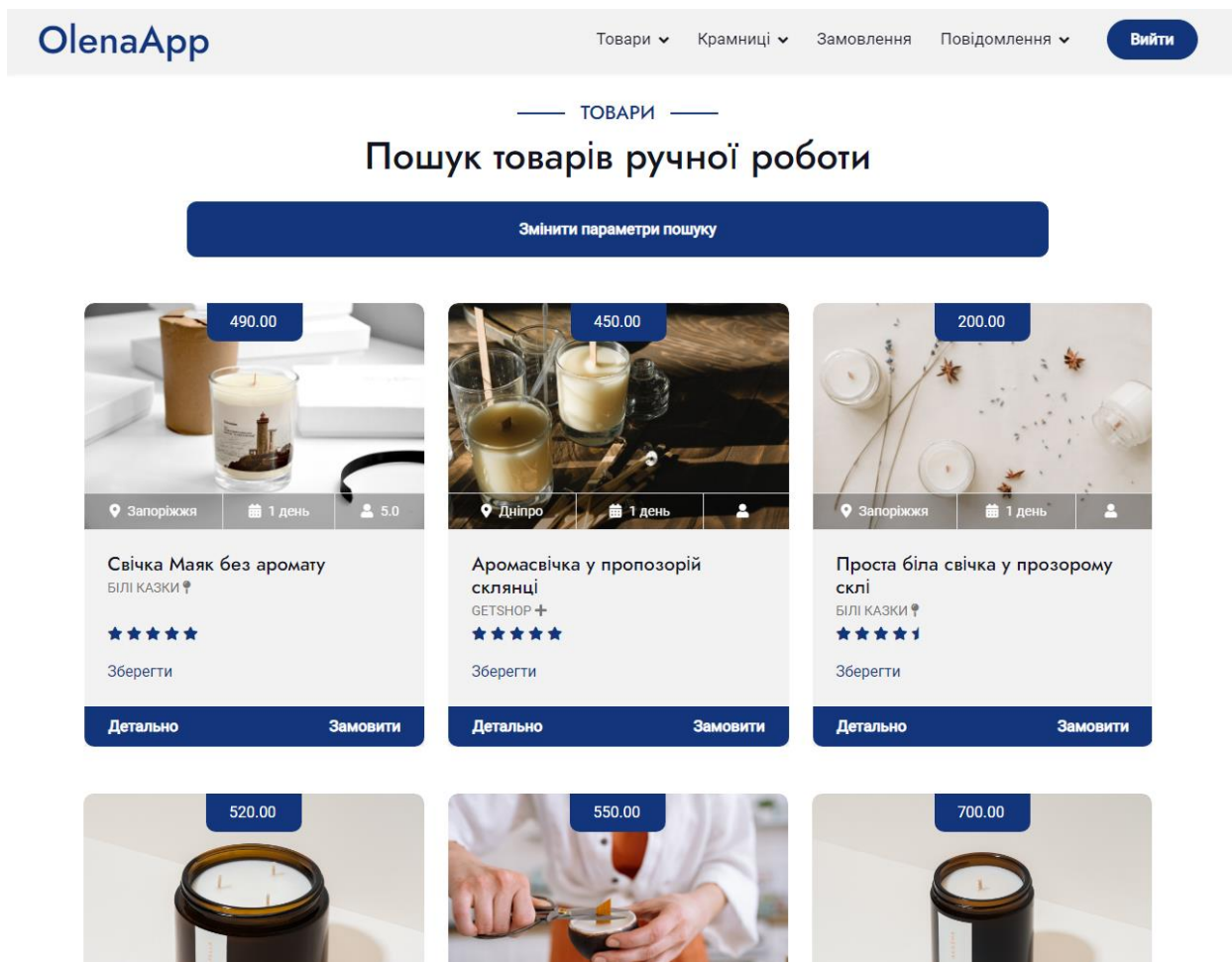


Рисунок 5.1 – Пошук товарів ручної роботи

Також в результатах показується середня оцінка за рейтингами всіх споживачів, посилання для збереження товару, а також посилання для переходу до сторінки товару ручної роботи, де приведена вся інформація про нього, та посилання для переходу до замовлення, що одразу розташовує ту частину сторінки товару, де саме вносяться дані замовлення, натискаючи на кнопку замовити при цьому можна сформувавши замовлення остаточно (рис. 5.2).

На сторінці з інформацією про товар наведено фотографії (щоб перемикатися між фотографіями, потрібно навести курсор на фотографію,

тоді з'являться стрілочки для перемикавання між попереднім і наступним зображенням), назву оголошення, середню загальну оцінку з рейтингів у вигляді зірочок, посилання для збереження товару, опис товару, назви крамниці, звідки можна перейти до сторінки крамниці, а нижче загальна оцінка з рейтингів, але вже крамниць, інформація про значення властивостей товару, а також поля самого замовлення. Нижче на сторінці розташовуються також дані рейтингів. Створити ж новий рейтинг споживач може зі сторінки власних замовлень.



ЦЕ ТОВАР РУЧНОЇ РОБОТИ —

Тарілка з червоними квітами

★★★★★

Зберегти

Тарілка керамічна, доволі стійка. Можна обрати потрібний розмір, доступна зазвичай в різних варіантах. Відтінок кольору може дещо відрізнятись, а саме зображення є результатом ручної роботи, тому містить те ж саме відображення, але з невеликим відхиленням в силу ручної роботи.

Крамниця: Ceramoshop ?

★★★★★

→ Категорія: посуд

→ Відправлення не пізніше: 3 дні

→ Обслуговування: 4.8

→ Якість: 5.0

→ Матеріал: кераміка

→ Загальна оцінка: 4.8

→ Доставка: 4.5

→ Відповідність опису: 5.0

Оберіть значення

40x40

Кількість

1

3000.00

Замовити

Рисунок 5.2 – Інформація про товар ручної роботи

Крамниці містять пункти також для пошуку, для відображення переліків найкращих і найпопулярніших, переліку крамниць, на яких було

поставлено шпильки.

Замовлення для споживача дозволяють працювати тільки з переліком власних оформлених замовлень товарів ручної роботи. Споживач може скасувати замовлення, але тільки до виконання відповідних дій продавцем.

Повідомлення дозволяють перейти до переліку повідомлень, які були написані різним крамницям, одразу написання звітти нових повідомлень у вже створені запити. Щоб створити новий запит, потрібно зайти на сторінку крамниці.

Продавці мають ті самі розділи меню, але розширену кількість пунктів. Зокрема з розділу Крамниці продавцю доступні дані власної крамниці, керування іншими продавцями, які можуть працювати з крамницею. Проте у кожній крамниці є певний основний користувач – власник. Інші продавці не можуть його вилучити.

Особливістю роботи продавця з програмою є зокрема те, що він може працювати над створенням нових оголошень про продаж товарів, керувати наявним переліком товарів. Внесення нового товару ручної роботи (рис. 5.3) відбувається за допомогою відповідної форми. Форма дозволяє додати фотографію для товару. Фотографію можна одразу вилучити кнопка під кожною фотографією). А кнопка Додати дозволяє додати нову фотографію. За товаром можна визначити назву оголошення (для того щоб виконалось прогнозування кількості рейтингів, які отримає товарів від споживачів, потрібно задати назву, опис англійською), обрати категорію (можна додати декілька), зазначити доступну кількість товару (при визначенні персональних опцій можна встановити кількість під кожний варіант, а тут визначається сума), вартість одиниці, кількість днів до відвантаження, опис товару, тег (можна додавати потрібну кількість, кнопка вилучити з'являється тоді, коли тег додано), характеристики товару, які є постійними (наприклад, матеріал, з якого виготовлено товар, при цьому переклад цих значень відбувається в програмі на англійську, а сформоване речення як деталі подається на вхід моделі).

OlenaApp: крамниця

Товари ▾Крамниці ▾Замовлення ▾Повідомлення ▾Вийти

GETSHOP

Додайте новий товар



Вилучити

Додати

Визначення популярності відбувається автоматично при визначенні значень у формі. Популярність визначається кількістю рейтингів.

Назва

Soft pillow with black cats

Категорія

Оберіть значення

Додати

Додана категорія: Декор

Вилучити

Доступна кількість

10

Вартість одиниці

1010

Кількість днів до відвантаження

1

Опис

For all cat lovers. Black cats on soft pillow. It can be used as decor and for pets as well. It has durable materials inside.

Тег

Оберіть значення

Додати

Характеристика товару

Оберіть значення

Додати

Додані характеристики: Матеріал Значення: Поролон

Вилучити

Опція для вибору

Оберіть значення

Додати

Опція для вибору з впливом на вартість

Оберіть значення

Додати

Кількість рейтингів очікується: 307

Опублікувати

Рисунок 5.3 – Створення оголошення про продаж товару ручної роботи

Також на сторінці можна задати ті опції, які за товаром можна задавати при замовленні (обирається назва, а тоді з'являється можливість обрати значення і кількість, вартість для останнього варіанту).

В кінці потрібно опублікувати оголошення, а вже при визначенні параметрів одразу автоматично обчислюється на основі використання запропонованого методу очікуваної кількості рейтингів цього товару, що будуть визначені споживачами.

Окрім того з розділу меню Замовлення доступний вже ряд пунктів, які дозволяють відображати замовлення товарів ручної роботи за крамницею, визначаючи стан замовлень.

5.3 Експериментальне дослідження

Експериментальне дослідження виконувалось на основі вибірки даних [10], забезпеченої [11]. У цій вибірці зібрані дані з програмного аналогу, розглянутого в першому розділі, – Etsy. Вибірка містить більше 10 000 записів про товари ручної роботи, розміщені на вебсайті.

При проведенні дослідження було спочатку виконано обробку даних, в результаті чого відкинуто всі дублікати, відкинуто всі екземпляри даних, які не мають вказаних описів, назв, встановленої вартості, категорії, вказаних деталей. Після цього у вибірці залишилось 8328 екземплярів.

Під час проведення дослідження було використано дві основні моделі: лінійну регресію та дерева рішень CART, на основі яких і був побудований метод, представлений у розділі 2. Ці дві моделі порівнювались між собою за використання різних вхідних ознак, при чому було враховано, що частина вхідних даних є текстовими, а перетворення таких даних може бути виконано по-різному. Зокрема розміри створених векторів можуть бути різними. Тому було перевірено декілька варіантів з різною довжиною векторів, щоб дослідити, як обране значення розміру вектору, яким кодуються текстові дані, впливають на результат прогнозування.

Спочатку було створено декілька моделей за використання базового набору ознак, що включав категорію і вартість, і розширених варіантів вхідних даних, включаючи закодоване представлення одного або декількох варіантів з назви товару, наявного опису товару та деталей товару (табл. 5.1). У першому випадку для представлення і опису, і назви, і деталей використовувався вектор, що складався зі 100 елементів кожен. Найкращий результат отримано без використання опису в якості вхідних ознак. При цьому помітно, що CART демонструє за всіх варіантів кращий результат за лінійну регресію. В найкращому випадку покращення становило 47 %.

Таблиця 5.1 – Результати прогнозування вартості товару ручної роботи на основі даних у 100 елементів

Модель	Категорія, вартість	Опис	Назва	Деталі	Середня абсолютна похибка
Лінійна регресія	+	+	+	+	0,011762
CART	+	+	+	+	0,006469
CART	+	—	+	+	0,006217
CART	+	+	+	—	0,006517
CART	+	+	—	+	0,006409
CART	+	—	—	+	0,00652
CART	+	—	+	—	0,006621
CART	+	+	—	—	0,006526
CART	+	—	—	—	0,006829

Далі було виконано подібне дослідження, використовуючи ті самі дані, але вже створюючи вектори довжиною по 50 елементів кожен (табл. 5.2).

Таблиця 5.2 – Результати прогнозування вартості товару ручної роботи на основі даних у 50 елементів

Модель	Категорія, вартість	Опис	Назва	Деталі	Середня абсолютна похибка
Лінійна регресія	+	+	+	+	0,011349
CART	+	+	+	+	0,006407
CART	+	–	+	+	0,006646
CART	+	+	+	–	0,006242
CART	+	+	–	+	0,006782
CART	+	–	–	+	0,006465
CART	+	–	+	–	0,006247
CART	+	+	–	–	0,006801
CART	+	–	–	–	0,006608

Найкращий результат у даному випадку отримано при використанні категорії, вартості, опису, назви, але відсутності деталей. Цей результат майже на 45 % кращий за результат лінійної регресії.

Також було створено представлення зі змінною довжиною векторів. Для опису було використано довжину в 50 елементів, для назви – 20, для деталей – 10 (табл. 5.3). У такому випадку було отримано найкращий результат зі всіх спостережень. Його дозволила отримати модель CART на основі використання всіх вхідних ознак, які пропонувались. Похибка в цьому випадку склала 0,005889, що на 47,5 % краще за відповідне значення, отримане для лінійної регресії. Тож все це говорить про те, що використання моделей на основі CART, побудованих за запропонованим методом, дійсно мало певний ефект, який виражався зменшенням похибки за результатами прогнозування порівняно з лінійною регресією.

Таблиця 5.3 – Результати прогнозування вартості товару ручної роботи на основі опису в 50 елементів, назви в 20 і деталей в 10 елементів

Модель	Категорія, вартість	Опис	Назва	Деталі	Середня абсолютна похибка
Лінійна регресія	+	+	+	+	0,011211
CART	+	+	+	+	0,005889
CART	+	—	+	+	0,006264
CART	+	+	+	—	0,006204
CART	+	+	—	+	0,006421
CART	+	—	—	+	0,006131
CART	+	—	+	—	0,006466
CART	+	+	—	—	0,006492
CART	+	—	—	—	0,006241

5.4 Висновки за розділом 5

У роботі при виконанні п'ятого розділу було розглянуто процес роботи користувача з програмою та виконано експериментальне дослідження визначення вартості товарів ручної роботи, для чого використано вибірку даних. Використання моделей на основі дерев CART, побудованих за запропонованим у роботі методом, дійсно мало вплив, який виражався зменшенням похибки за результатами прогнозування порівняно з лінійною регресією.

ВИСНОВКИ

У роботі виконана розробка програмного забезпечення підтримки продажу товарів ручної роботи з можливістю визначення популярності товарів на майбутнє. Створена програма має відмінності від проаналізованих програм, які використовуються для роботи над продажем товарів ручної роботи на даний момент.

Розроблена як вебзастосунок програма призначена для продажу товарів, які є результатом ручної роботи, крамницями, які представляють майстрів і яких представляють відповідно продавці, та придбання товарів споживачами. Для цього підтримуються засоби керування замовленнями, визначення доступних товарів, збереження товарів і крамниць для подальшого швидкого повернення до них, створення рейтингів товарів ручної роботи та крамниць. Програма складається з підсистем споживача, керування крамницями, визначення популярності товарів ручної роботи у своєму складі.

Наукова новизна роботи полягає в запропонованому методі визначення популярності товарів ручної роботи, в якому на основі числових характеристик товару ручної роботи та текстових даних, перетворених до числового представлення, виконується прогнозування кількості рейтингів за допомогою створення дерева рішень CART.

Використання моделей на основі дерев рішень CART, побудованих за запропонованим у роботі методом, мало вплив, який виражався зменшенням похибки за результатами прогнозування порівняно з лінійною регресією. Експериментальне дослідження було виконано на основі вибірки даних про товари ручної роботи, використовуючи різні варіанти представлення текстових даних для використання в моделі та самі моделі на основі дерев рішень CART та лінійної регресії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Etsy Ukraine – Shop for handmade, vintage, custom, and unique gifts for everyone [Electronic resource]. – Access mode : <https://www.etsy.com/>.
2. Crafta – авторські вироби ручної роботи, хенд мейд [Електрон. ресурс]. – Режим доступу : <https://crafta.ua/uk/>.
3. Lawton, G. What is Linear Regression? [Electronic resource] / G. Lawton. – Access mode : <https://www.techtarget.com/searchenterpriseai/definition/linear-regression>.
4. Suryakumar, K. Classification and Regression Trees in machine learning [Electronic resource] / K. Suryakumar. – Access mode : <https://www.linkedin.com/pulse/classification-regression-trees-machine-learning-suryakumar-k>.
5. Classification and Regression Trees [Electronic resource]. – Access mode : <https://www.complexica.com/narrow-ai-glossary/classification-regression-trees>.
6. Khare, P. Deep Learning for NLP [Electronic resource] : Word2Vec, Doc2Vec, and Top2Vec Demystified / P. Khare. – Access mode : <https://medium.com/mllearning-ai/deep-learning-for-nlp-word2vec-doc2vec-and-top2vec-demystified-3842b4fad5c9>.
7. Drupal – Open Source CMS. Drupal.org [Electronic resource]. – Access mode : <https://www.drupal.org/>.
8. Django Tutorial [Electronic resource]. – Access mode : <https://www.w3schools.com/django/>.
9. Travela – Free Tourism Website Template [Electronic resource]. – Access mode : <https://htmlcodex.com/tourism-website-template/>.
10. Etsy retail products dataset [Electronic resource]. – Access mode : <https://www.kaggle.com/datasets/yamqwe/etsy-retail-products-dataset>.
11. Crawl Feeds. Web scraping and data mining [Electronic resource]. – Access mode : <https://crawlfeeds.com/>.

ДОДАТОК А
Технічне завдання

Вступ

Ручна робота призводить до отримання ексклюзивних варіантів товарів. Вони можуть мати особливості, пов'язані зі споживачем, який їх замовив, але з іншого боку вони зазвичай мають певні особливості виконання. Це широковідомий факт і він як раз і привертає увагу до товарів ручної роботи. Але зворотнім боком є те, що кожен майстер створює невелику кількість товарів, а товар, який створюється іншим майстром, навіть якщо в нього та сама концепція, навіть дуже близьке виконання, все одно є іншим товаром. Тож коли виконується пошук нових товарів ручної роботи, то знайти їх стає достатньо складно, зрозуміти, який саме товар підходить ще складніше, адже на це впливає не тільки концепція, але і конкретне виконання. Отримати інформацію про конкретне виконання дозволяють рейтингові оцінювання або відгуки, які створюють споживачі, які вже такий товар купили. Кількість же таких рейтингових оцінювань, кількість замовлень товарів говорить про його популярність. Популярність разом з середнім рівнем рейтингів впливають на сприйняття інших споживачів. Тому реалізація процесів роботи з товарами ручної роботи вимагає створення програмного забезпечення, яке охоплює цілий ряд базових концептуальних функцій:

- надання інформації про сам товар ручної роботи, замовлення такого товару;
- пошук товарів, що створені в результаті ручної роботи;
- визначення популярності товарів, що створені в результаті ручної роботи, на початку продажів.

A.1 Підстави для розробки

Завдання на роботу визначено самим виданим завданням від 4 вересня 2023 року, що має входити до складу пояснювальної записки, та наказом

№ 437 від 14 листопада 2023 р. за Національним університетом «Запорізька політехніка», де затверджено тему роботи: «Дослідження та програмна реалізація методів визначення популярності під час продажу товарів ручної роботи».

А.2 Призначення розробки

Програма призначена для продажу товарів, які є результатом ручної роботи, крамницями, які представляють майстрів і яких представляють відповідно продавці, та придбання товарів споживачами.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Для того щоб досягти сформульовано призначення розробки в підсумку, потрібно виконати такі функціональні вимоги:

- відображення найкращих товарів ручної роботи за категорією;
- відображення найкращих товарів ручної роботи за крамницею;
- відображення найкращих товарів ручної роботи за тегом;
- відображення товарів ручної роботи, що продаються в крамниці;
- відображення товарів ручної роботи, що найактивніше продаються;
- відображення найпопулярніших товарів ручної роботи за категорією;
- відображення найпопулярніших товарів ручної роботи за крамницею;
- пошук крамниць, які продають товари ручної роботи;
- пошук товарів ручної роботи;
- додавання товару ручної роботи для продажу в крамниці;
- редагування даних товару ручної роботи;
- відображення результатів роботи крамниці;

- відображення замовлень товарів ручної роботи;
- створення шпильки на крамниці;
- прибирання шпильки з крамниці;
- відображення крамниць, на яких залишено шпильки;
- збереження товару ручної роботи;
- відображення збережених товарів ручної роботи;
- вилучення збереження товару ручної роботи;
- замовлення товарів ручної роботи;
- скасування замовлення товарів ручної роботи;
- відображення замовлень товарів ручної роботи за крамницею;
- відображення замовлень заданого товару ручної роботи;
- авторизація;
- підтвердження замовлення товарів ручної роботи за власною крамницею;
- деактивація роботи працівника з крамницею;
- визначення замовлення як виконаного;
- визначення відправленого замовлення;
- відображення інформації про товар ручної роботи;
- відображення інформації про крамницю, що продає товари ручної роботи;
- відображення найкращих крамниць за містом;
- відображення найпопулярніших крамниць;
- рейтингове оцінювання товару ручної роботи;
- рейтингове оцінювання крамниці;
- відображення рейтингових оцінювань крамниці;
- відображення рейтингових оцінювань товару ручної роботи;
- реагування на рейтингове оцінювання товару ручної роботи;
- реагування на рейтингове оцінювання крамниці;
- відправлення повідомлення між крамницею та споживачем;

- редагування інформації про крамницю, що продає товари ручної роботи;
- відображення рейтингових оцінювань, які споживач вважає для себе корисними.

A.3.2 Умови експлуатації та вимоги

Умови експлуатації програми визначаються тими засобами, які мають бути обрані під час проєктування програми, включаючи зокрема підтримку обраної мови програмування, організацію роботи серверу визначеним чином, забезпечення на ньому роботи системи керування базами даних, яку має бути обрано, а також будь-яких додаткових засобів, які обрані та використані під час розробки програми та потребують інсталяції.

A.4 Порядок контролю та приймання

Контроль виконання заявлених складових роботи має відбуватися згідно з завданням, виданим на роботу, що є однією з підстав для розробки. Виконання роботи має бути завершено протягом 12 тижнів з проведенням захисту перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

```

from .models import Category
from django.shortcuts import render
from .models import Hardwareimage
from .models import Hardware
from .models import Hardwareproperty
from .models import Customeraccount
from .models import Hardwaresave
from .models import Hardwarestorepin
from .models import Hardwarenotice
from .models import Hardwarestore
from .models import Hardwarestorenotice
from .models import Hardwareproperty
from .models import Hardwaregeneraloption
from .models import Hardwarepersonaloptionvalue
from .models import Hardwarenoticeback
from .models import Hardwarestorenoticeback
from django.shortcuts import redirect

def best_category_hardware_view(request, categ, old =
True):
    try:
        categor = Category.objects.get(name = categ)
    except Category.DoesNotExist:
        return render(request,
"olenaprogram/best_category.html", {})
    categories = Category.objects.all()
    categories = categories.order_by("name")
    if old:
        hardware = Hardware.objects.filter(categories =
categor)
    else:
        hardware = Hardware.objects.filter(categories =
categor, availability_gte = 1)
        hardware = hardware.select_related("hardwarestore")
        hardware = hardware.prefetch_related("categories")
        hardware = hardware.order_by("-notice_rate")
        for hardware in hardware:
            images = Hardwareimage.objects.filter(hardware =
hardware)
            images = images.values()[0]
            hardware.image = images
            if request.user.is_authenticated:
                customer = Customeraccount.objects.filter(login =
request.user.username)
                for hardware in hardware:
                    hardwaresave =
Hardwaresave.objects.filter(hardware = hardware, customeraccount
= customer)
                    if hardwaresave and hardwaresave.count() > 0:
                        hardware.saved = True
                    else:

```

```

        hardware.saved = False
        notices =
Hardwarenotice.objects.filter(hardware = hardware,
customeraccount = customer)
        notices = notices.order_by("-date_rate")
        if notices and notices.count() > 0:
            hardware.notice = notices.values()[0]

    else:
        hardware.notice = False
        pins =
Hardwarestorepin.objects.filter(hardwarestore =
hardware.hardwarestore, customeraccount = customer)
        if pins and pins.count() > 0:
            hardware.pin = True
        else:
            hardware.pin = False
    return render(request, "olenaprogram/best_category.html",
{"hardwares": hardwares, "categories": categories })

def popular_hardwarestore_hardware_view(request,
hardwarestore, old = True):
    try:
        hardwarestore = Hardwarestore.objects.get(id =
hardwarestore)
    except Hardwarestore.DoesNotExist:
        return render(request,
"olenaprogram/popular_hardwarestore.html", {})
        categories = Category.objects.all()
        categories = categories.order_by("name")
        if old:
            hardwares = Hardware.objects.filter(hardwarestore =
hardwarestore)
        else:
            hardwares = Hardware.objects.filter(hardwarestore =
hardwarestore, availability_gte = 1)
            hardwares = hardwares.prefetch_related("categories")
            for hardware in hardwares:
                images = Hardwareimage.objects.filter(hardware =
hardware)

                images = images.values()[0]
                hardware.image = images
            pin = False
            if request.user.is_authenticated:
                customer = Customeraccount.objects.filter(login =
request.user.username)
                for hardware in hardwares:
                    hardwaresave =
Hardwaresave.objects.filter(hardware = hardware, customeraccount
= customer)

                    if hardwaresave and hardwaresave.count() > 0:

```

```

        hardware.saved = True
    else:
        hardware.saved = False
    notices =
Hardwarenotice.objects.filter(hardware = hardware,
customeraccount = customer)
    notices = notices.order_by("-date_rate")
    if notices and notices.count() > 0:
        hardware.notice = notices.values()[0]
    else:
        hardware.notice = False
    notices =
Hardwarenotice.objects.filter(hardware = hardware)
    notices = notices.count()
    hardware.notices_count = notices
    pins =
Hardwarestorepin.objects.filter(hardwarestore = hardwarestore,
customeraccount = customer)
    if pins and pins.count() > 0:
        pin = True
    else:
        pin = False
    hardware = hardware.order_by("-notices_count")
    return render(request,
"olenaprogram/popular_hardwarestore.html", {"hardware": hardware,
"categories": categories, "pin": pin, "hardwarestore":
hardwarestore })

def best_tag_hardware_view(request, tag, old = True):
    try:
        tag = Tag.objects.get(name = tag)
    except Tag.DoesNotExist:
        return render(request, "olenaprogram/best_tag.html",
{})

    categories = Category.objects.all()
    categories = categories.order_by("name")
    tags = Tag.objects.all()
    tags = tags.order_by("name")
    if old:
        hardware = Hardware.objects.filter(tags = tag)
    else:
        hardware = Hardware.objects.filter(tags = tag,
availability_gte = 1)
    hardware = hardware.select_related("hardwarestore")
    hardware = hardware.prefetch_related("categories")
    hardware = hardware.order_by("-notice_rate")
    for hardware in hardware:
        images = Hardwareimage.objects.filter(hardware =
hardware)

        images = images.values()[0]
        hardware.image = images

```

```

        if request.user.is_authenticated:
            customer = Customeraccount.objects.filter(login =
request.user.username)
            for hardware in hardware:
                hardware.save =
Hardwaresave.objects.filter(hardware = hardware, customeraccount
= customer)
                if hardware.save and hardware.save.count() > 0:
                    hardware.saved = True
                else:
                    hardware.saved = False
                notices =
Hardwarenotice.objects.filter(hardware = hardware,
customeraccount = customer)
                notices = notices.order_by("-date_rate")
                if notices and notices.count() > 0:
                    hardware.notice = notices.values()[0]
                else:
                    hardware.notice = False
                pins =
Hardwarestorepin.objects.filter(hardwarestore =
hardware.hardwarestore, customeraccount = customer)
                if pins and pins.count() > 0:
                    hardware.pin = True
                else:
                    hardware.pin = False
            return render(request, "olenaprogram/best_tag.html",
{"hardware": hardware, "categories": categories })

def popular_category_hardware_view(request, categ, old =
True):
    try:
        categor = Category.objects.get(name = categ)
    except Category.DoesNotExist:
        return render(request,
"olenaprogram/popular_category.html", {})
    categories = Category.objects.all()
    categories = categories.order_by("name")
    if old:
        hardware = Hardware.objects.filter(categories =
categor)
    else:
        hardware = Hardware.objects.filter(categories =
categor, availability_gte = 1)
        hardware = hardware.select_related("hardwarestore")
        hardware = hardware.prefetch_related("categories")
        for hardware in hardware:
            images = Hardwareimage.objects.filter(hardware =
hardware)
            images = images.values()[0]
            hardware.image = images

```

```

        if request.user.is_authenticated:
            customer = Customeraccount.objects.filter(login =
request.user.username)
            for hardware in hardware:
                hardware.save =
Hardwaresave.objects.filter(hardware = hardware, customeraccount
= customer)
                if hardware.save and hardware.save.count() > 0:
                    hardware.saved = True
                else:
                    hardware.saved = False
                notices =
Hardwarenotice.objects.filter(hardware = hardware,
customeraccount = customer)
                notices = notices.order_by("-date_rate")
                if notices and notices.count() > 0:
                    hardware.notice = notices.values()[0]
                else:
                    hardware.notice = False
                pins =
Hardwarestorepin.objects.filter(hardwarestore =
hardware.hardwarestore, customeraccount = customer)
                if pins and pins.count() > 0:
                    hardware.pin = True
                else:
                    hardware.pin = False
                notices =
Hardwarenotice.objects.filter(hardware = hardware)
                notices = notices.count()
                hardware.notices_count = notices
                hardware = hardware.order_by("-notices_count")
                return render(request,
"olenaprogram/popular_category.html", {"hardware": hardware,
"categories": categories })

def hardwarestore_hardware_view(request, hardwarestore,
sortway = 0):
    try:
        hardwarestore = Hardwarestore.objects.get(id =
hardwarestore)
    except Hardwarestore.DoesNotExist:
        return render(request,
"olenaprogram/hardwarestore_hardware.html", {})
    categories = Category.objects.all()
    categories = categories.order_by("name")
    hardware = Hardware.objects.filter(hardwarestore =
hardwarestore, availability_gte = 1)
    hardware = hardware.prefetch_related("categories")
    for hardware in hardware:
        images = Hardwareimage.objects.filter(hardware =
hardware)

```

```

        images = images.values()[0]
        hardware.image = images
    pin = False
    if request.user.is_authenticated:
        customer = Customeraccount.objects.filter(login =
request.user.username)
        for hardware in hardwarestore:
            hardwaresave =
Hardwaresave.objects.filter(hardware = hardware, customeraccount
= customer)
            if hardwaresave and hardwaresave.count() > 0:
                hardware.saved = True
            else:
                hardware.saved = False
            notices =
Hardwarenotice.objects.filter(hardware = hardware,
customeraccount = customer)
            notices = notices.order_by("-date_rate")
            if notices and notices.count() > 0:
                hardware.notice = notices.values()[0]
            else:
                hardware.notice = False
            notices =
Hardwarenotice.objects.filter(hardware = hardware)
            notices = notices.count()
            hardware.notices_count = notices
        pins =
Hardwarestorepin.objects.filter(hardwarestore = hardwarestore,
customeraccount = customer)
        if pins and pins.count() > 0:
            pin = True
        else:
            pin = False
    if sortway == 0:
        hardwarestore = hardwarestore.order_by("-baseprice")
    elif sortway == 1:
        hardwarestore = hardwarestore.order_by("baseprice")
    elif sortway == 2:
        hardwarestore = hardwarestore.order_by("adv_title")
    elif sortway == 3:
        hardwarestore = hardwarestore.order_by("-availability")
    return render(request,
"olenaprogram/hardwarestore_hardware.html", {"hardwarestore":
hardwarestore, "categories": categories, "pin": pin,
"hardwarestore": hardwarestore })

def best_hardwarestore_hardware_view(request,
hardwarestore, old = True):
    try:
        hardwarestore = Hardwarestore.objects.get(id =
hardwarestore)

```

```

        except Hardwarestore.DoesNotExist:
            return render(request,
                "olenaprogram/best_hardwarestore.html", {})
            categories = Category.objects.all()
            categories = categories.order_by("name")
            if old:
                hardware = Hardware.objects.filter(hardwarestore =
hardwarestore)
            else:
                hardware = Hardware.objects.filter(hardwarestore =
hardwarestore, availability_gte = 1)
                hardware = hardware.prefetch_related("categories")
                hardware = hardware.order_by("-notice_rate")
                for hardware in hardware:
                    images = Hardwareimage.objects.filter(hardware =
hardware)
                    images = images.values()[0]
                    hardware.image = images
                pin = False
                if request.user.is_authenticated:
                    customer = Customeraccount.objects.filter(login =
request.user.username)
                    for hardware in hardware:
                        hardwaresave =
Hardwaresave.objects.filter(hardware = hardware, customeraccount
= customer)
                        if hardwaresave and hardwaresave.count() > 0:
                            hardware.saved = True
                        else:
                            hardware.saved = False
                        notices =
Hardwarenotice.objects.filter(hardware = hardware,
customeraccount = customer)
                        notices = notices.order_by("-date_rate")
                        if notices and notices.count() > 0:
                            hardware.notice = notices.values()[0]
                        else:
                            hardware.notice = False
                pins =
Hardwarestorepin.objects.filter(hardwarestore = hardwarestore,
customeraccount = customer)
                if pins and pins.count() > 0:
                    pin = True
                else:
                    pin = False
            return render(request,
                "olenaprogram/best_hardwarestore.html", {"hardware": hardware,
                "categories": categories, "pin": pin, "hardwarestore":
hardwarestore })

def best_hardwarestores_view(request, storecity = ""):

```

```

        categories = Category.objects.all()
        categories = categories.order_by("name")
        if storecity:
            hardwarestores =
Hardwarestore.objects.filter(main_city = storecity)
        else:
            hardwarestores = Hardwarestore.objects.all()
            hardwarestores = hardwarestores.order_by("-
overall_rate")
        if request.user.is_authenticated:
            customer = Customeraccount.objects.filter(login =
request.user.username)
            for hardwarestore in hardwarestores:
                notices =
Hardwarestorenotice.objects.filter(hardwarestore =
hardwarestore, customeraccount = customer)
                notices = notices.order_by("-date_rate")
                if notices and notices.count() > 0:
                    hardwarestore.notice = notices.values()[0]
                else:
                    hardwarestore.notice = False
                pins =
Hardwarestorepin.objects.filter(hardwarestore = hardwarestore,
customeraccount = customer)
                if pins and pins.count() > 0:
                    hardwarestore.pin = True
                else:
                    hardwarestore.pin = False
            return render(request,
"olenaprogram/best_hardwarestores.html", {"hardwarestores":
hardwarestores, "categories": categories })

def popular_hardwarestores_view(request):
    categories = Category.objects.all()
    categories = categories.order_by("name")
    hardwarestores = Hardwarestore.objects.all()
    if request.user.is_authenticated:
        customer = Customeraccount.objects.filter(login =
request.user.username)
        for hardwarestore in hardwarestores:
            notices =
Hardwarestorenotice.objects.filter(hardwarestore =
hardwarestore, customeraccount = customer)
            notices = notices.order_by("-date_rate")
            if notices and notices.count() > 0:
                hardwarestore.notice = notices.values()[0]
            else:
                hardwarestore.notice = False
            pins =
Hardwarestorepin.objects.filter(hardwarestore = hardwarestore,
customeraccount = customer)

```

```

        if pins and pins.count() > 0:
            hardwarestore.pin = True
        else:
            hardwarestore.pin = False
        notices =
Hardwarestorenotice.objects.filter(hardwarestore =
hardwarestore)
        if notices and notices.count() > 0:
            hardwarestore.notices_count =
notices.count()
        else:
            hardwarestore.notices_count = False
        hardwarestore = hardwarestore.order_by("-
notices_count")
        return render(request,
"olenaprogram/popular_hardwarestores.html", {"hardwarestores":
hardwarestores, "categories": categories })

def sold_hardware_view(request, way):
    categories = Category.objects.all()
    categories = categories.order_by("name")
    hardware = Hardware.objects.filter(hardwarestore =
hardwarestore)
    hardware = hardware.prefetch_related("categories")
    for hardware in hardware:
        images = Hardwareimage.objects.filter(hardware =
hardware)
        images = images.values()[0]
        hardware.image = images
        pin = False
        if request.user.is_authenticated:
            customer = Customeraccount.objects.filter(login =
request.user.username)
            for hardware in hardware:
                hardwaresave =
Hardwaresave.objects.filter(hardware = hardware, customeraccount
= customer)
                if hardwaresave and hardwaresave.count() > 0:
                    hardware.saved = True
                else:
                    hardware.saved = False
                notices =
Hardwarenotice.objects.filter(hardware = hardware,
customeraccount = customer)
                notices = notices.order_by("-date_rate")
                if notices and notices.count() > 0:
                    hardware.notice = notices.values()[0]
                else:
                    hardware.notice = False

```

```

        notices =
Hardwarenotice.objects.filter(hardware = hardware)
        notices = notices.count()
        hardware.notices_count = notices
        pins =
Hardwarestorepin.objects.filter(hardwarestore =
hardware.hardwarestore, customeraccount = customer)
        if pins and pins.count() > 0:
            hardware.pin = True
        else:
            hardware.pin = False
    if way == 0:
        hardware = hardware.order_by("-total_sales")
    else:
        hardware = hardware.order_by("-week_sales")
    return render(request, "olenaprogram/sold_hardware.html",
{"hardware": hardware, "categories": categories})

def hardwarestore_view(request, store):
    categories = Category.objects.all()
    categories = categories.order_by("name")
    try:
        hardwarestore = Hardwarestore.objects.get(id =
store)
    except Hardwarestore.DoesNotExist:
        return render(request,
"olenaprogram/hardwarestore.html", {})
    hardware = Hardware.objects.filter(hardwarestore =
hardwarestore, availability_gte = 1)
    hardware = hardware.prefetch_related("categories")
    for hardware in hardware:
        images = Hardwareimage.objects.filter(hardware =
hardware)
        images = images.values()[0]
        hardware.image = images
    hardware = hardware.order_by("-notice_rate")
    hardware = hardware.values[:10]
    pin = False
    if request.user.is_authenticated:
        customer = Customeraccount.objects.filter(login =
request.user.username)
        for hardware in hardware:
            hardware.save =
Hardwaresave.objects.filter(hardware = hardware, customeraccount
= customer)
            if hardware.save and hardware.save.count() > 0:
                hardware.saved = True
            else:
                hardware.saved = False

```

```

        notices =
Hardwarenotice.objects.filter(hardware = hardware,
customeraccount = customer)
        notices = notices.order_by("-date_rate")
        if notices and notices.count() > 0:
            hardware.notice = notices.values()[0]
        else:
            hardware.notice = False
        notices =
Hardwarenotice.objects.filter(hardware = hardware)
        notices = notices.count()
        hardware.notices_count = notices
        pins =
Hardwarestorepin.objects.filter(hardwarestore = hardwarestore,
customeraccount = customer)
        if pins and pins.count() > 0:
            pin = True
        else:
            pin = False
        notices =
Hardwarestorenotice.objects.filter(hardwarestore =
hardwarestore)
        notices = notices.order_by("-overall_rate")
        notices = notices.values()[:10]
        for handnotice in notices:
            handnotice.backs =
Hardwarestorenoticeback.objects.filter(hardwarestorenotice =
handnotice)
        return render(request, "olenaprogram/hardwarestore.html",
{"hardwares": hardwares, "categories": categories, "pin": pin,
"hardwarestore": hardwarestore, "notices": notices })

def hardware_view(request, hardware):
    categories = Category.objects.all()
    categories = categories.order_by("name")
    try:
        hardware = Hardware.objects.get(id = hardware)
        hardware =
hardware.select_related("hardwarestore")
        hardware =
hardware.prefetch_related("hardwaregeneraloptionvalues")
    except Hardware.DoesNotExist:
        return render(request, "olenaprogram/hardware.html",
{})

    for hardware in hardwares:
        images = Hardwareimage.objects.filter(hardware =
hardware)
        images = images.values()
        hardware.image = images
        if request.user.is_authenticated:

```

```

        customer = Customeraccount.objects.filter(login =
request.user.username)
        hardwaresave = Hardwaresave.objects.filter(hardware
= hardware, customeraccount = customer)
        if hardwaresave and hardwaresave.count() > 0:
            hardware.saved = True
        else:
            hardware.saved = False
        notices = Hardwarenotice.objects.filter(hardware =
hardware, customeraccount = customer)
        notices = notices.order_by("-date_rate")
        if notices and notices.count() > 0:
            hardware.notice = notices.values()[0]
        else:
            hardware.notice = False
        pins =
Hardwarestorepin.objects.filter(hardwarestore = hardwarestore,
customeraccount = customer)
        if pins and pins.count() > 0:
            hardware.pin = True
        else:
            hardware.pin = False
        notices = Hardwarenotice.objects.filter(hardware =
hardware)
        notices = notices.order_by("-date_rate")
        notices_count = notices.count()
        hardware.notices_count = notices_count
        for notice in notices:
            notice.backs =
Hardwarenoticeback.objects.filter(hardwarenotice = notice)
            hardware.notices = notices
            properties = Hardwareproperty.objects.filter(hardware =
hardware)
            hardware.properties = properties
            general_options =
Hardwaregeneraloption.objects.filter(hardware = hardware)
            general_options =
general_options.select_related("option")
            hardware.general_options = general_options
            personaloptions =
Hardwarepersonaloptionvalue.objects.filter(hardware = hardware)
            hardware.options = personaloptions
            return render(request, "olenaprogram/sold_hardware.html",
{"hardwares": hardwares, "categories": categories, "notices":
notices })

    @method_decorator(csrf_exempt, name='dispatch')
    class HardwarestorepinView(LoginRequiredMixin, View):
        def post(self, request, pk):
            hardware_store = get_object_or_404(Hardwarestore,
id = pk)

```

```

        customer = Customeraccount.objects.filter(login =
request.user.username)
        pin = Hardwarestorepin(customeraccount = customer,
hardwarestore = hardware_store)
        try:
            pin.save()
        except IntegrityError:
            pass
        return HttpResponse()

    @method_decorator(csrf_exempt, name='dispatch')
    class HardwarestorepinDeleteView(LoginRequiredMixin, View):
        def post(self, request, pk):
            hardware_store = get_object_or_404(Hardwarestore,
id = pk)
            try:
                customer = Customeraccount.objects.filter(login
= request.user.username)
                pin =
Hardwarestorepin.objects.get(customeraccount = customer,
hardwarestore = hardware_store).delete()
            except Hardwarestorepin.DoesNotExist:
                pass
            return HttpResponse()

    def pinned_hardwarestores_view(request):
        categories = Category.objects.all()
        categories = categories.order_by("name")
        hardwarestores = []
        if request.user.is_authenticated:
            customer = Customeraccount.objects.filter(login =
request.user.username)
            pins =
Hardwarestorepin.objects.filter(customeraccount = customer)
            pins = pins.order_by("position")
            pins = pins.select_related("hardwarestore")
            hardwarestores = pins.values_list("hardwarestore")
            for hardwarestore in hardwarestores:
                notices =
Hardwarestorenotice.objects.filter(hardwarestore =
hardwarestore, customeraccount = customer)
                notices = notices.order_by("-date_rate")
                if notices and notices.count() > 0:
                    hardwarestore.notice = notices.values()[0]
                else:
                    hardwarestore.notice = False
            return render(request,
"olenaprogram/pinned_hardwarestores.html", {"hardwarestores":
hardwarestores, "categories": categories })

    @method_decorator(csrf_exempt, name='dispatch')

```

```

class HardwaresaveView(LoginRequiredMixin, View):
    def post(self, request, pk):
        hardware = get_object_or_404(Hardware, id = pk)
        customer = Customeraccount.objects.filter(login =
request.user.username)
        hardware_save = Hardwaresave(customeraccount =
customer, hardware = hardware)
        try:
            hardware_save.save()
        except IntegrityError:
            pass
        return HttpResponse()

@method_decorator(csrf_exempt, name='dispatch')
class HardwaresaveDeleteView(LoginRequiredMixin, View):
    def post(self, request, pk):
        hardware = get_object_or_404(Hardware, id = pk)
        try:
            customer = Customeraccount.objects.filter(login
= request.user.username)
            hardware_save =
Hardwaresave.objects.get(customeraccount = customer, hardware =
hardware).delete()
        except Hardwaresave.DoesNotExist:
            pass
        return HttpResponse()

def saved_hardware_view(request):
    categories = Category.objects.all()
    categories = categories.order_by("name")
    hardware = []
    if request.user.is_authenticated:
        customer = Customeraccount.objects.filter(login =
request.user.username)
        hardware_saves =
Hardwaresave.objects.filter(customeraccount = customer)
        hardware_saves = hardware_saves.order_by("-id")
        hardware_saves =
hardware_saves.select_related("hardware")
        hardware = hardware_saves.values_list("hardware")
        for hardware in hardware:
            notices =
Hardwarenotice.objects.filter(hardware = hardware,
customeraccount = customer)
            notices = notices.order_by("-date_rate")
            if notices and notices.count() > 0:
                hardware.notice = notices.values()[0]

        else:
            hardware.notice = False

```

```

        notices =
Hardwarenotice.objects.filter(hardware = hardware)
        notices = notices.count()
        hardware.notices_count = notices
        pins =
Hardwarestorepin.objects.filter(hardwarestore =
hardware.hardwarestore, customeraccount = customer)
        if pins and pins.count() > 0:
            hardware.pin = True
        else:
            hardware.pin = False
        return render(request, "olenaprogram/saved.html",
{"hardwares": hardwares, "categories": categories })

def hardwarestore_notices_view(request, store):
    categories = Category.objects.all()
    categories = categories.order_by("name")
    try:
        hardwarestore = Hardwarestore.objects.get(id =
store)
    except Hardwarestore.DoesNotExist:
        return render(request,
"olenaprogram/hardwarestore_notices.html", {})
    notices =
Hardwarestorenotice.objects.filter(hardwarestore =
hardwarestore)
    notices = notices.select_related("customeraccount")
    notices = notices.order_by("-date_rate")
    for notice in notices:
        backs =
Hardwarestorenoticeback.objects.filter(hardwarestorenotice =
notice)
        backs = backs.select_related("customeraccount")
        notice.backs = backs
    if request.user.is_authenticated:
        customer = Customeraccount.objects.filter(login =
request.user.username)
        for hardware in hardwares:
            hardware.save =
Hardwaresave.objects.filter(hardware = hardware, customeraccount
= customer)
            if hardware.save and hardware.save.count() > 0:
                hardware.saved = True
            else:
                hardware.saved = False
            notices =
Hardwarenotice.objects.filter(hardware = hardware,
customeraccount = customer)
            notices = notices.order_by("-date_rate")
            if notices and notices.count() > 0:
                hardware.notice = notices.values()[0]

```

```

        else:
            hardware.notice = False
            notices =
Hardwarenotice.objects.filter(hardware = hardware)
            notices = notices.count()
            hardware.notices_count = notices
        pins =
Hardwarestorepin.objects.filter(hardwarestore = hardwarestore,
customeraccount = customer)
        if pins and pins.count() > 0:
            hardware.pin = True
        else:
            hardware.pin = False
    return render(request,
"olenaprogram/hardwarestore_notices.html", {"notices": notices,
"categories": categories, "hardwarestore": hardwarestore })

def hardware_notices_view(request, hardware):
    categories = Category.objects.all()
    categories = categories.order_by("name")
    try:
        hardware = Hardware.objects.get(id = store)
    except Hardware.DoesNotExist:
        return render(request,
"olenaprogram/hardware_notices.html", {})
    notices = Hardwarenotice.objects.filter(hardware =
hardware)
    notices = notices.select_related("customeraccount")
    notices = notices.order_by("-date_rate")
    for notice in notices:
        backs =
Hardwarenoticeback.objects.filter(hardwarenotice = notice)
        backs = backs.select_related("customeraccount")
        notice.backs = backs
    if request.user.is_authenticated:
        customer = Customeraccount.objects.filter(login =
request.user.username)
        hardwaresave = Hardwaresave.objects.filter(hardware
= hardware, customeraccount = customer)
        if hardwaresave and hardwaresave.count() > 0:
            hardware.saved = True
        else:
            hardware.saved = False
        notices = Hardwarenotice.objects.filter(hardware =
hardware, customeraccount = customer)
        notices = notices.order_by("-date_rate")
        if notices and notices.count() > 0:
            hardware.notice = notices.values()[0]
        else:
            hardware.notice = False

```

```

        pins =
Hardwarestorepin.objects.filter(hardwarestore = hardwarestore,
customeraccount = customer)
        if pins and pins.count() > 0:
            hardware.pin = True
        else:
            hardware.pin = False
    return render(request,
"olenaprogram/hardware_notices.html", {"notices": notices,
"categories": categories, "hardware": hardware })

def customer_hardware_notices_view(request):
    categories = Category.objects.all()
    categories = categories.order_by("name")
    backs = []
    if request.user.is_authenticated:
        customer = Customeraccount.objects.filter(login =
request.user.username)
        backs =
Hardwarenoticeback.objects.filter(customeraccount = customer)
        backs = notices.select_related("hardwarenotice")
        backs =
notices.select_related("hardwarenotice__hardware")
        backs = backs.order_by("-id")
        for back in backs:
            hardwarenotice = back.hardwarenotice
            back.backs =
Hardwarenoticeback.objects.filter(hardwarenotice =
hardwarenotice)
        return render(request,
"olenaprogram/cust_hardware_notices.html", {"backs": backs,
"categories": categories })

from .forms import Hardwarenoticeform

def add_hardware_notice_view(request, hardware):
    try:
        hardware = Hardware.objects.get(id = hardware)
        hardware =
hardware.select_related("hardwarestore")
    except Hardware.DoesNotExist:
        return render(request, "olenaprogram/add_notice.html",
{})

    if request.user.is_authenticated:
        customer = Customeraccount.objects.filter(login =
request.user.username)
        carts = Carthardware.objects.filter(hardware =
hardware, cart__customeraccount = customer, cart__state_gte = 1)
        if request.method == "POST" and carts and
carts.count > 0:

```

```

        hardwarenotice_form =
Hardwarenoticeform(request.POST)
        if hardwarenotice_form.is_valid():
            hardwarenotice =
hardwarenotice_form.save(commit = False)
            hardwarenotice.hardware = hardware
            hardwarenotice.customeraccount = customer
            hardwarenotice.save()
            return redirect(hardware)
        else:
            hardwarenotice_form = None
            return render(request, "olenaprogram/add_notice.html",
{"form": hardwarenotice_form })

from .forms import Hardwarestorenoticeform

def add_hardwarestore_notice_view(request, hardwarestore):
    try:
        hardware_store = Hardwarestore.objects.get(id =
hardwarestore)
    except Hardwarestore.DoesNotExist:
        return render(request,
"olenaprogram/add_storenotice.html", {})
    if request.user.is_authenticated:
        customer = Customeraccount.objects.filter(login =
request.user.username)
        carts =
Carthardware.objects.filter(hardware__hardwarestore =
hardware_store, cart__customeraccount = customer,
cart__state_gte = 1)
        if request.method == "POST" and carts and
carts.count > 0:
            hardwarestorenotice_form =
Hardwarestorenoticeform(request.POST)
            if hardwarestorenotice_form.is_valid():
                hardwarestorenotice =
hardwarestorenotice_form.save(commit = False)
                hardwarestorenotice.hardwarestore =
hardware_store
                hardwarestorenotice.customeraccount =
customer
                hardwarestorenotice.save()
                notices =
Hardwarestorenotice.objects.filter(hardwarestore =
hardware_store)
                sum = 0
                servsum = 0
                shipsum = 0
                qualitysum = 0
                cnt = notices.count()
                for notice in notices:

```

```

        sum = sum + notice.overall_rate
        servsum = servsum + notice.service_rate
        shipsum = shipsum +
notice.shipping_rate
        qualitysum = qualitysum +
notice.quality_rate
        sum = sum / cnt
        hardware_store.overall_rate = sum
        hardware_store.service_rate = servsum
        hardware_store.shipping_rate = shipsum
        hardware_store.quality_rate = qualitysum
        hardware_store.save()
        return redirect(hardware_store)
    else:
        hardwarestorenotice_form = None
        return render(request,
"olenaprogram/add_storenotice.html", {"form":
hardwarestorenotice_form })

class HardwarenoticePositiveView(LoginRequiredMixin, View):
    def post(self, request, pk):
        hardwarenotice = get_object_or_404(Hardwarenotice,
id = pk)
        customer = Customeraccount.objects.filter(login =
request.user.username)
        notice_back = Hardwarenoticeback(customeraccount =
customer, hardwarenotice = hardwarenotice, backtype = True)
        try:
            notice_back.save()
        except IntegrityError:
            pass
        return HttpResponseRedirect()

class HardwarenoticeNegativeView(LoginRequiredMixin, View):
    def post(self, request, pk):
        hardwarenotice = get_object_or_404(Hardwarenotice,
id = pk)
        customer = Customeraccount.objects.filter(login =
request.user.username)
        notice_back = Hardwarenoticeback(customeraccount =
customer, hardwarenotice = hardwarenotice, backtype = False)
        try:
            notice_back.save()
        except IntegrityError:
            pass
        return HttpResponseRedirect()

class HardwarestorenoticePositiveView(LoginRequiredMixin,
View):
    def post(self, request, pk):

```

```

        hardwarestorenotice =
get_object_or_404(Hardwarestorenotice, id = pk)
        customer = Customeraccount.objects.filter(login =
request.user.username)
        notice_back =
Hardwarestorenoticeback(customeraccount = customer,
hardwarestorenotice = hardwarestorenotice, backtype = True)
        try:
            notice_back.save()
        except IntegrityError:
            pass
        return HttpResponse()

class HardwarestorenoticeNegativeView(LoginRequiredMixin,
View):
    def post(self, request, pk):
        hardwarestorenotice =
get_object_or_404(Hardwarestorenotice, id = pk)
        customer = Customeraccount.objects.filter(login =
request.user.username)
        notice_back =
Hardwarestorenoticeback(customeraccount = customer,
hardwarestorenotice = hardwarestorenotice, backtype = False)
        try:
            notice_back.save()
        except IntegrityError:
            pass
        return HttpResponse()

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота магістра на тему:

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ВИЗНАЧЕННЯ ПОПУЛЯРНOSTІ ПІД ЧАС ПРОДАЖУ ТОВАРІВ
РУЧНОЇ РОБОТИ. RESEARCH AND SOFTWARE IMPLEMENTATION
OF METHODS FOR EVALUATING THE POPULARITY OF HAND MADE
ITEMS DURING SALE

Виконала студентка групи КНТ-132м
Керівник доцент, к.т.н.

Олена ПАВЛІЧЕНКО
Степан СКРУПСЬКИЙ

2023

Рисунок В.1 – Титульний слайд

ОБ'ЄКТ, ПРЕДМЕТ, МЕТА РОБОТИ

- Об'єкт роботи – процес продажу товарів ручної роботи.
- Предмет роботи – методи визначення популярності під час продажу товарів ручної роботи.
- Мета роботи – розробка програмного забезпечення підтримки продажу товарів ручної роботи з можливістю визначення популярності товарів на майбутнє.

2

Рисунок В.2 – Об'єкт, предмет, мета роботи

ЗАВДАННЯ РОБОТИ

- аналіз існуючих програм, які виконують підтримку продажу товарів ручної роботи;
- дослідження методів і моделей визначення популярності товарів ручної роботи;
- проектування програми підтримки продажу товарів ручної роботи;
- реалізація програми підтримки продажу товарів ручної роботи.

3

Рисунок В.3 – Завдання роботи

АНАЛІЗ РОЗРОБЛЕНОЇ ТА ІСНУЮЧИХ ПРОГРАМ

Показник	Etsy	Crafta	Власна розробка
Крамниці продажу товарів ручної роботи	Так	Так	Так
Пошук крамниць	Ні	Ні	Так
Пошук товарів ручної роботи за властивостями	Колір	Немає	Різні
Рейтинги крамниць	Так	Так	Так
Рейтинги товарів ручної роботи	Так	Ні	Так
Вибір значень властивостей для товару	Не більше 2	Немає	Можливо більше 2
Визначення популярності товарів	Немає	Немає	Так

4

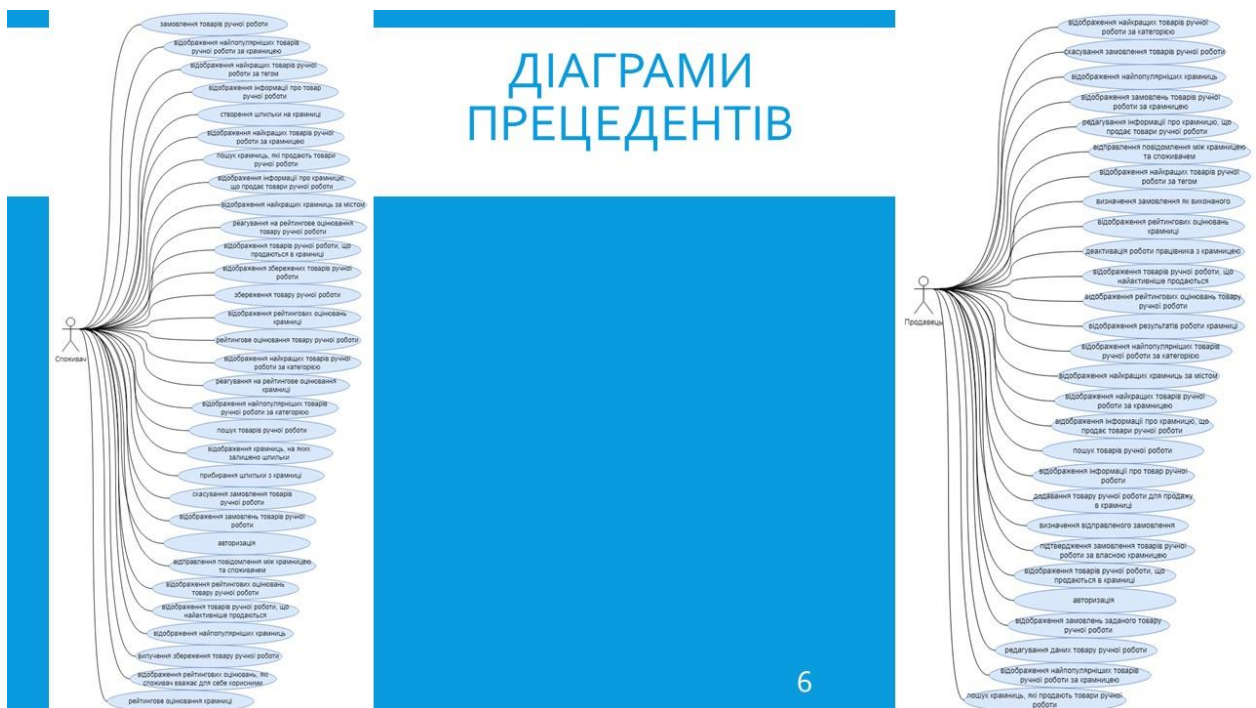
Рисунок В.4 – Аналіз розробленої та існуючих програм

МЕТОД ВИЗНАЧЕННЯ ПОПУЛЯРНОСТІ ТОВАРІВ РУЧНОЇ РОБОТИ

- формування навчальної вибірки даних: навчальна вибірка має бути сформована з екземплярів, які представлені товарами ручної роботи, для кожного з яких відомі всі необхідні вхідні ознаки та кількість рейтингів, яку зібрав цей товар ручної роботи як вихідна ознака;
- опрацювання текстових даних з побудовою моделей для перетворення текстових даних у вектори числових значень;
- оброблення текстів відповідних вхідних ознак для кожного екземпляра на основі використання створених моделей;
- створення моделі на основі дерева рішень CART і її навчання за допомогою створеної навчальної вибірки;
- виконання прогнозування для нових товарів ручної роботи за створеною моделлю.

5

Рисунок В.5 – Метод визначення популярності товарів ручної роботи



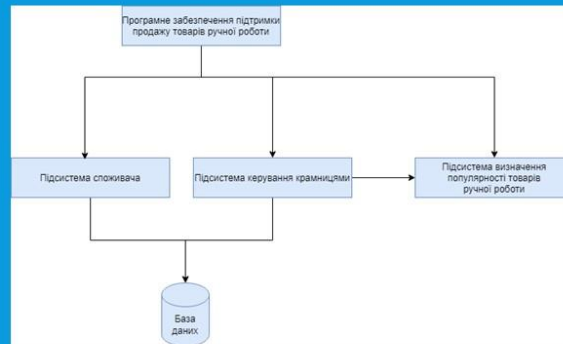
6

Рисунок В.6 – Діаграми прецедентів

Показник	Drupal	Django
Розробка нетипових функцій	Потребує наявності відповідних плагінів і модулів або написання і інтеграції	Реалізація предметної області, бізнес-логіки лежить в основі, додаткова реалізація не потрібна
Безпека роботи	Потребує додаткових дій, налаштувань, обов'язкових стратегій	Має багато вбудованих засобів, що вирішують велику кількість проблем одразу
Підтримка спільнотою розробників	Активна	Активна
Мова програмування	PHP	Python
Підтримка реалізації машинного навчання	Дуже обмежена	На найвищому рівні

[illegible]

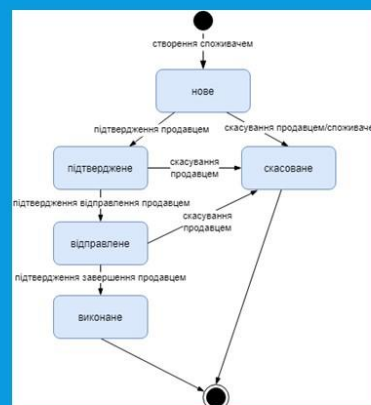
СХЕМА СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПІДТРИМКИ ПРОДАЖУ ТОВАРІВ РУЧНОЇ РОБОТИ



9

Рисунок В.9 – Схема структури програмного забезпечення підтримки продажу товарів ручної роботи

ДІАГРАМА СТАНІВ ЗАМОВЛЕННЯ ТОВАРІВ РУЧНОЇ РОБОТИ



10

Рисунок В.10 – Діаграма станів замовлення товарів ручної роботи

ІНТЕРФЕЙС ПРОГРАМИ ПІДТРИМКИ ПРОДАЖУ ТОВАРІВ РУЧНОЇ РОБОТИ

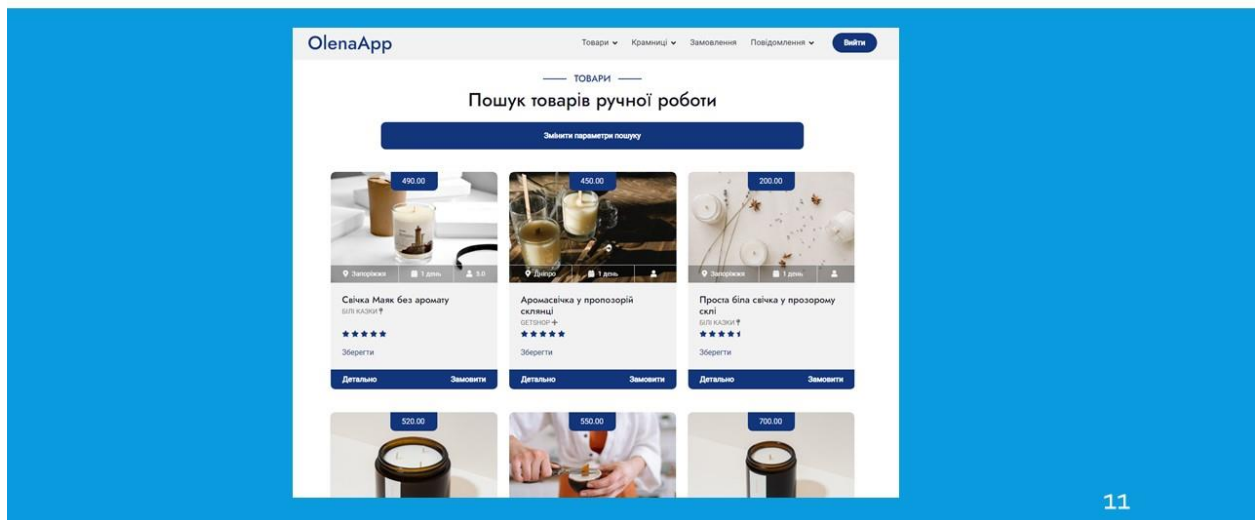


Рисунок В.11 – Інтерфейс програми підтримки продажу товарів ручної роботи

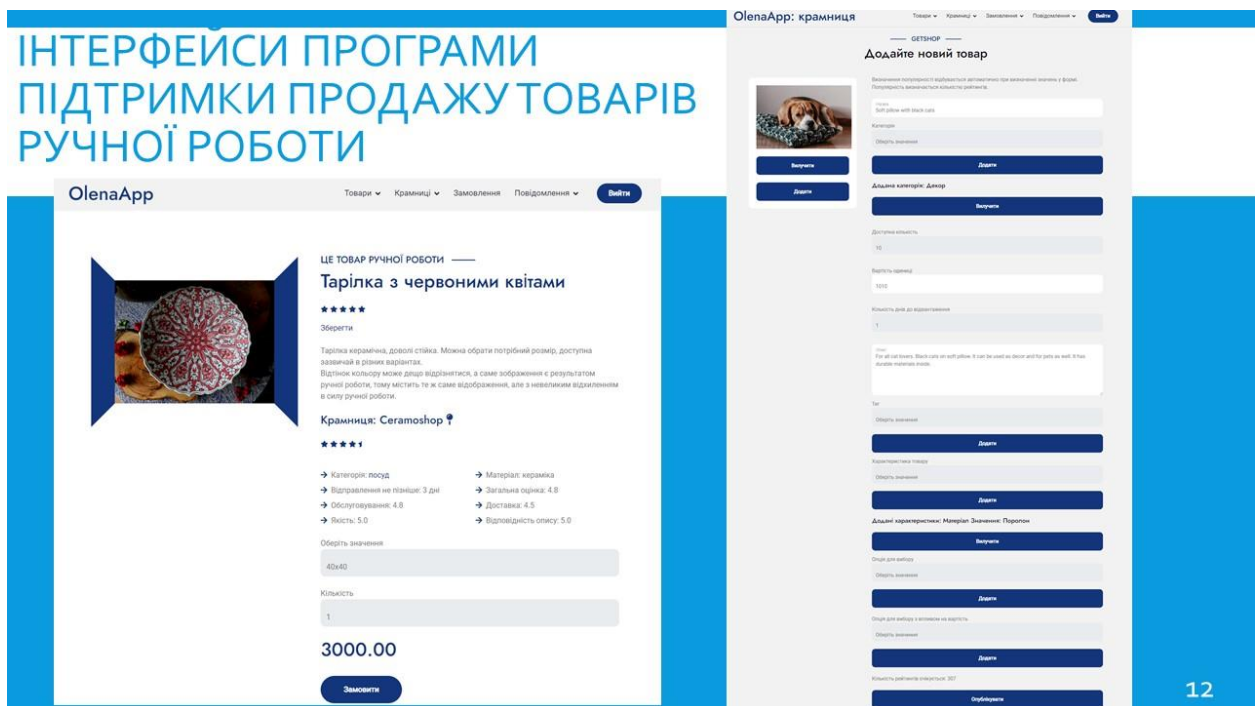


Рисунок В.12 – Інтерфейси програми підтримки продажу товарів ручної роботи

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

Модель	Категорія, вартість	Опис	Назва	Деталі	Середня абсолютна похибка
Лінійна регресія	+	+	+	+	0,011211
CART	+	+	+	+	0,005889
CART	+	–	+	+	0,006264
CART	+	+	+	–	0,006204
CART	+	+	–	+	0,006421
CART	+	–	–	+	0,006131
CART	+	–	+	–	0,006466
CART	+	+	–	–	0,006492
CART	+	–	–	–	0,006241

13

Рисунок В.13 – Експериментальне дослідження

ВИСНОВКИ

- Розроблена як вебзастосунок програма призначена для продажу товарів, які є результатом ручної роботи, крамницями, які представляють майстрів і яких представляють відповідно продавці, та придбання товарів споживачами. Для цього підтримуються засоби керування замовленнями, визначення доступних товарів, збереження товарів і крамниць для подальшого швидкого повернення до них, створення рейтингів товарів ручної роботи та крамниць. Програма складається з підсистем споживача, керування крамницями, визначення популярності товарів ручної роботи у своєму складі.
- Наукова новизна роботи полягає в запропонованому методі визначення популярності товарів ручної роботи, в якому на основі числових характеристик товару ручної роботи та текстових даних, перетворених до числового представлення, виконується прогнозування кількості рейтингів за допомогою створення дерева рішень CART.

14

Рисунок В.14 – Висновки