

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)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ЗАБЕЗПЕЧЕННЯ ЦІНОУТВОРЕННЯ ПІД ЧАС ПРОКАТУ СУКОНЬ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS
FOR DRESS RENTAL PRICING

Виконав(ла): студент(ка) 2 курсу, групи КНТ-223м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

ПІДГОРНА Ю.Ю.

(ПРИЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(КОД І НАЙМЕНУВАННЯ)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

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

ПІДГОРНОЇ Юлії Юріївни

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів забезпечення ціноутворення під час прокату суконь. Research and Software Implementation of Methods for Dress Rental Pricing

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

затверджені наказом закладу вищої освіти від “ 7 ” жовтня 2024 року № 398

2. Строк подання студентом проєкту (роботи) 01 грудня 2024 року

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

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

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	СКРУПСЬКИЙ С.Ю., доцент		
Нормоконтролер	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 08 ” вересня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

(підпис)

Юлія ПІДГОРНА

(Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Степан СКРУПСЬКИЙ

(Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
109 с., 4 табл., 26 рис., 3 дод., 10 джерел.

ПРОКАТ СУКОНЬ, ОГОЛОШЕННЯ, ДЕРЕВО УХВАЛЕННЯ
РІШЕНЬ, КАТЕГОРІЯ, ПРОГНОЗУВАННЯ ВАРТОСТІ, ВЕБЗАСТОСУНОК.

Об'єкт роботи – процес прокату суконь. Предмет роботи – методи визначення ціноутворення під час прокату суконь. Мета роботи – розробити програмне забезпечення прокату суконь для підвищення зручності прийняття рішень щодо визначення вартості прокату.

Матеріали, методи та технічні засоби: засоби машинного навчання, включаючи зокрема моделі прогнозування на основі дерев ухвалення рішень, методу опорних векторів та лінійної регресії, мова програмування Python, фреймворк Django, система MySQL.

Результати. Програма, яка була розроблена для підтримки процесу прокату суконь, призначена для виставлення на прокат власних суконь користувачами та надання варіантів прокату самим сервісом прокату суконь, керування взаємодією з користувачами стосовно прокату суконь, пошуку тих варіантів прокату суконь, які задовольняють користувачів. Під час створення нового оголошення про прокат сукні автоматично відбувається формування пропозиції щодо встановлення вартості прокату на основі навченої моделі.

Висновки. Робота має наукову новизну, представлена методом визначення ціноутворення під час прокату суконь, який визначає процес формування вхідних ознак, виділення найбільш інформативних з них, створення моделі на основі дерева ухвалення рішень, що дозволяє прогнозувати вартість прокату сукні.

Галузь використання. Прокат суконь від окремих власників та сервісу прокату загалом.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 109 pages, 4 tables, 26 figures, 3 appendixes, 10 sources.

DRESS RENTAL, ADS, DECISION TREE, CATEGORY, COST PREDICTION, WEB APPLICATION.

The object of the work is the process of renting dresses. The subject of the work is methods of determining pricing during dress rental. The purpose of the work is to develop software for renting dresses to increase the convenience of decision-making process regarding the rental price.

Materials, methods and technical means: machine learning tools, including in particular forecasting models based on decision trees, the method of support vectors and linear regression, the Python programming language, the Django framework, the MySQL system.

The results. The software, which was developed to support the process of renting dresses, is intended for renting out own dresses by users and providing rental options by the dress rental service itself, managing interaction with users regarding the rental of dresses, finding those rental options of dresses that satisfy users. When creating a new dress rental ad, an offer to set the rental price is automatically generated based on the trained model.

Conclusions. The work has a scientific novelty, represented by the method of determining pricing during dress rental, which determines the process of forming input features, selecting the most informative of them, creating a model based on a decision-making tree, which allows owners to predict the cost of renting a dress.

Field of use. Dress rental from individual owners and rental service in general.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Функціональні можливості та концепція існуючих вебзастосунків прокату суконь	10
1.2 Визначення рівня створюваного вебзастосунку прокату суконь.....	14
1.3 Висновки за розділом 1	15
2 Матеріали і методи.....	16
2.1 Задача визначення ціноутворення під час прокату суконь.....	16
2.2 Метод визначення ціноутворення під час прокату суконь.....	18
2.3 Висновки за розділом 2	21
3 Проектування програмного забезпечення	22
3.1 Визначення сценаріїв роботи користувачів з вебзастосунком прокату суконь.....	22
3.2 Вибір засобів розробки вебзастосунку прокату суконь	30
3.3 Проектування структури бази даних.....	31
3.4 Висновки за розділом 3	41
4 Основні рішення щодо реалізації програмного забезпечення	42
4.1 Загальний порядок роботи програми прокату суконь.....	42
4.2 Реалізовані програмні одиниці у складі вебзастосунку прокату суконь.....	43
4.3 Висновки за розділом 4	60
5 Експлуатація, тестування та експериментальне дослідження програми	61
5.1 Основне призначення програми	61
5.2 Виконання програми.....	61
5.3 Експериментальне дослідження прогнозування вартості прокату суконь.....	66
5.4 Висновки за розділом 5	69

Висновки	71
Перелік джерел посилань	72
Додаток А Технічне завдання	73
Додаток Б Текст програми	79
Додаток В Слайди презентації.....	102

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

САП – середня абсолютна похибка;

СКП – середня квадратична похибка.

ВСТУП

Останні роки відбувається відновлення сфери продажу одягу, а разом з нею і прокату. Зниження попиту в минулому було пов'язано з пандемією, через що потреба знизилась, адже не було необхідності відвідування офісів, подій [1]. На 2023 рік 6,2 мільярда американських доларів складає розмір світового ринку оренди одягу, що більш ніж удвічі перевищує цей показник (2,6 мільярда американських доларів) у 2016 році [1].

Використання прокату суконь може дозволити вирішити проблеми, які виникають перед людьми в різних ситуаціях, при цьому знижуючи витрати на вирішення таких проблем. Визначення вартості прокату при цьому є наступним ключовим питанням після визначення готовності виставити сукню для прокату. Від рішення, яке приймається на цьому етапі, залежить те, чи вдасться привернути увагу клієнтів, а фактично в підсумку, чи вдасться заробити кошти на організації прокату. Тому для реалізації даного процесу важливою є наявність програмного механізму, який би створив основу для прийняття відповідного рішення. Тому при реалізації програмного забезпечення прокату суконь важливо передбачити такий механізм. Це відповідно має з одного боку підтримувати користувачів, які виставляють сукні на прокат, у прийнятті рішень, а з іншого визначити конкурентну перевагу створеного програмного забезпечення. Зважаючи на наявність ряду вебзстосунків для різних прокатів одягу, в тому числі і суконь, наявність конкурентної переваги є важливою.

Відповідно у даній роботі необхідно реалізувати програмне забезпечення прокату суконь, що має забезпечуватися як від сервісу прокату суконь, так і від окремих власників, при цьому надаючи засіб прогнозування вартості прокату суконь на основі параметрів цих суконь. Метою роботи є розробити програмне забезпечення прокату суконь для підвищення зручності прийняття рішень щодо визначення вартості прокату.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Функціональні можливості та концепція існуючих вебзастосунків прокату суконь

Hirestreet (рис. 1.1) виконує прокат суконь за двома моделями: підписки, надаючи за певну плату на кожен місяць 5 суконь для користування, або за вимогою, коли сукня надається в прокат на 21 день за окрему плату [2]. Така модель дещо обмежує прокат, однак може викликати інтерес у певних користувачів. Усі сукні надаються для прокату самим сервісом.

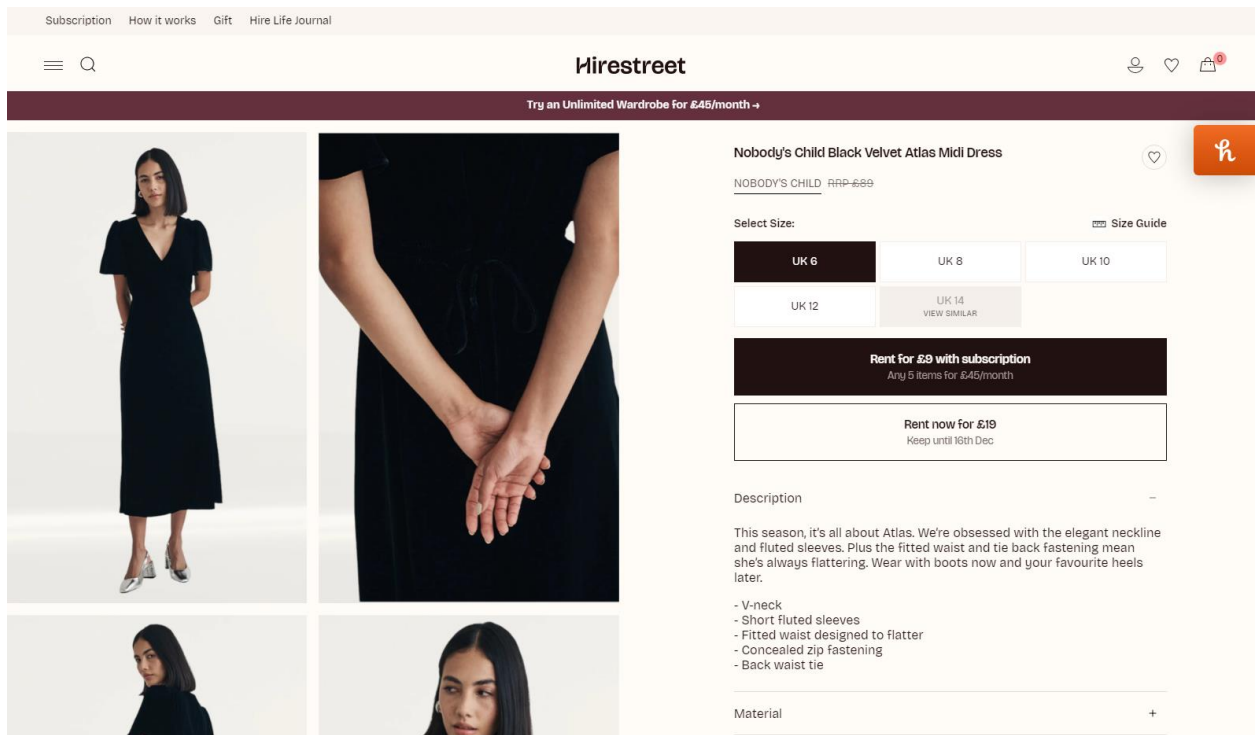


Рисунок 1.1 – Інтерфейс прокату сукні у сервісі Hirestreet [2]

Під час пошуку можна здавати необхідність наявності сукні на поточний момент, розмір, бренд, колір, довжину (міні, максимальна та середня), наявність принтів або їх відсутність, тип принта за наявності, посадку, довжину рукава.

Сукні можна зберігати для того, щоб потім повернутись до відібраних. Через вебзастосунок можна оформити замовлення прокату, а також здійснити підписку.

За кожною сукнею доступний її опис, склад, доступні розміри, з яких можна обрати потрібний для прокату. Загалом представлені типові сукні, з яких можна вибрати потрібний розмір.

Сервіс прокату By Rotation (рис. 1.2) розділяє сукні на категорії (весільні, вечірні, випускні тощо) [3].

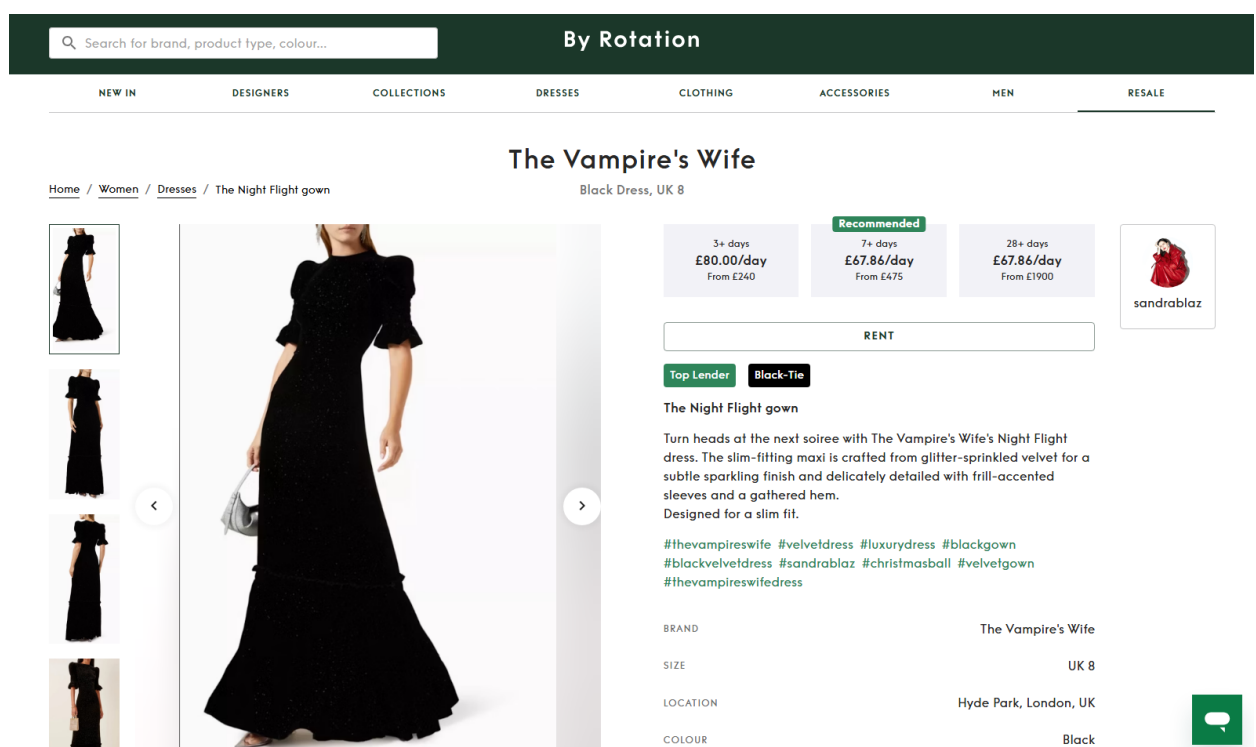


Рисунок 1.2 – Інтерфейс прокату сукні у сервісі By Rotation [3]

Існують різні варіанти прокату суконь щодо тривалості. Мінімальна тривалість складає 3 дні, але далі пакетно можуть визначатися інші пропозиції (від певної кількості днів зі встановленням вартості).

Під час організації пошуку введені текстові дані переглядаються за різними параметрами, але далі результати можна відфільтрувати, встановлюючи підкатегорію (включає різні варіанти і фасонів, і категорій), бренд, розмір, колір, колекцію, обмеження вартості на день.

Усі сукні надаються сервісом прокату. Для кожної сукні визначено розмір, тобто варіанти не є типовими, а кожен окремо характеризується своїм розміром. За сукнями визначені теги, однак за тегами пошук не виконується, сукні за тегами не відображаються.

У сервісі Hurr (рис. 1.3) серед всіх варіантів одягу сукні виділені в єдиний розділ, не розділяючи за категоріями, відповідно звернення до такого розділу призводить до відображення каталогу з можливістю фільтрації за розміром, кольором, дизайнерами, поточною доступністю, вартістю прокату [4]. Сукні надаються не тільки на прокат, але можуть бути і придбані.

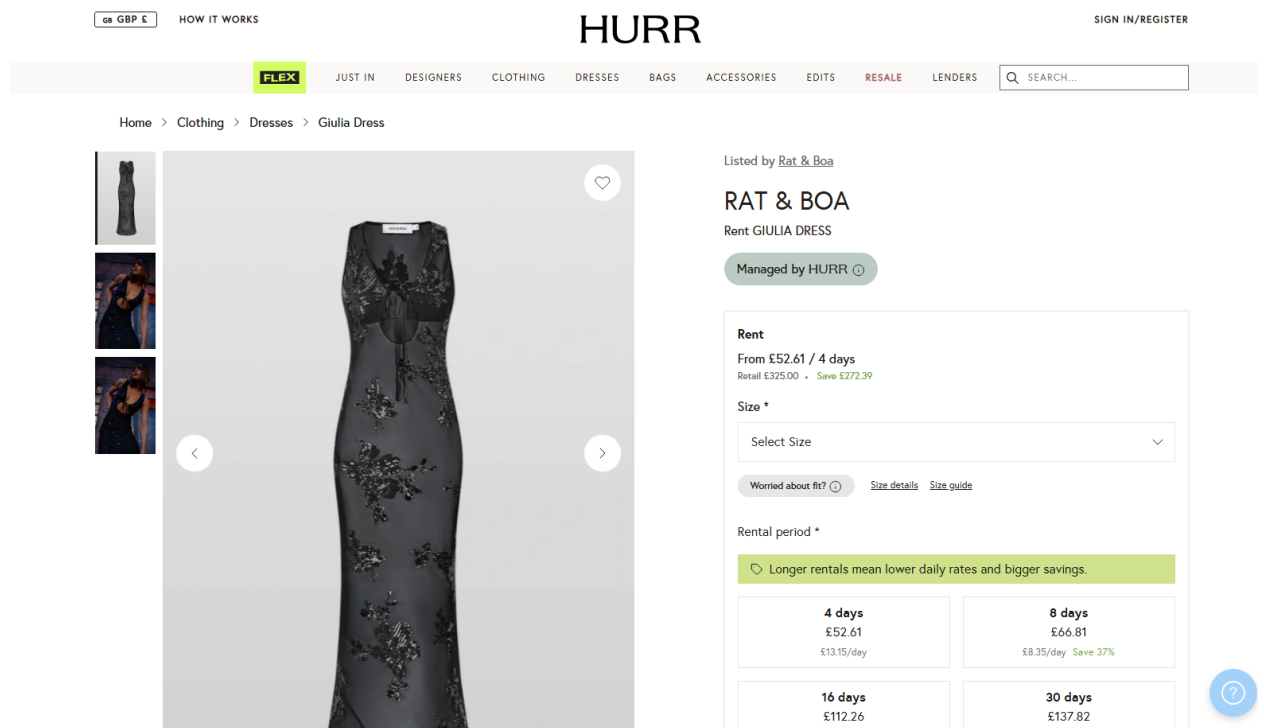


Рисунок 1.3 – Інтерфейс прокату сукні у сервісі Hurr [4]

Окрім того сукні надаються не тільки сервісом Hurr, але і іншими користувачами. Відгуки можна залишати за користувачами. Власникам можна написати повідомлення щодо суконь.

Сукні можна зберігати для повернення до них у подальшому у єдиний перелік. Сукні можуть бути представлені як типові з набором доступних стандартних розмірів, так і в одному екземплярі, коли розмір не вибирається.

Drexcode (рис. 1.4) є ще одним подібним сервісом, але прокат здійснюється тільки з суконь, які надаються самим сервісом [5]. Прокат здійснюється на чітко визначений період, а сукня має бути повернута на четвертий день.

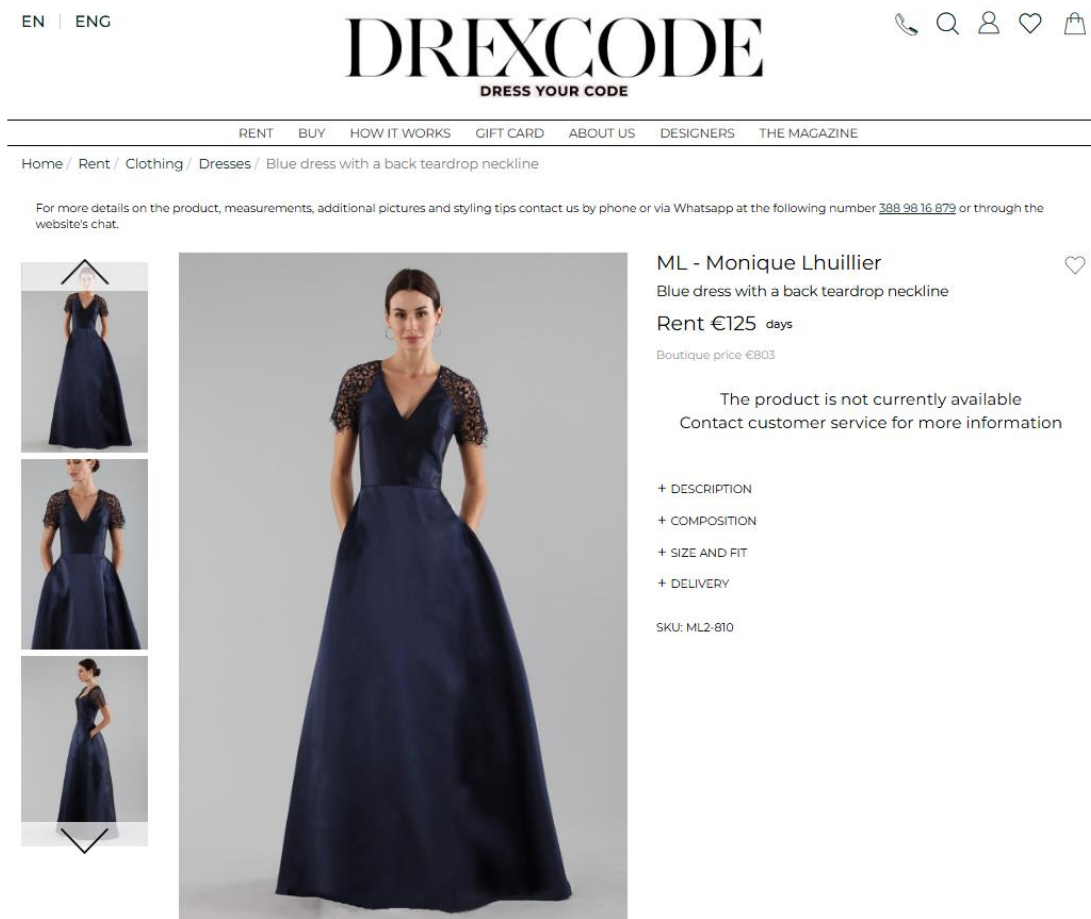


Рисунок 1.4 – Інтерфейс прокату сукні у сервісі Drexcode [5]

Пошук суконь реалізується з фільтрацією за вартістю на 4 дні, розміром, довжиною, кольором, дизайнером, подією, для якої сукня призначена, довжиною рукава. Для конкретної сукні можна обрати розмір, але зазвичай це один доступний варіант.

Сукню можна взяти на прокат, а можна придбати. Можна обрати дату, на яку має бути доставлена сукня. Також можна зберегти сукню для повернення потім до перегляду обраних варіантів.

1.2 Визначення рівня створюваного вебзастосунку прокату суконь

У підсумкове порівняння було внесено вебзастосунки Hirestreet, By Rotation та Hurr (табл. 1.1). Порівняння виконувалось з розробкою, що забезпечувалась у цій роботі та названа RenDress.

Таблиця 1.1 – Порівняння можливостей аналогів розробки вебзастосунку прокату суконь

Показник	Hirestreet	By Rotation	Hurr	RenDress
Прокат суконь	+	+	+	+
Джерело прокату	Сервіс	Сервіс	Сервіс, користувачі	Сервіс, користувачі
Тривалість прокату	21 день	Від 3 днів стандартні варіанти	Від 4 днів стандартні варіанти	Гнучке налаштування
Пошук суконь	З використанням фільтрів	3 використанням фільтрів	3 використанням фільтрів	Більша кількість фільтрів
Створення переліків перегляду	Тільки збереження у стандартний	Тільки збереження у стандартний	Тільки збереження у стандартний	Користувачькі переліки
Прогнозування вартості прокату	—	—	—	+

У вебзастосунку RenDress вирішено реалізувати логіку різних джерел прокату, таким чином формуючи альтернативи до пропозицій, наявних у сервіса. Тривалість прокату вирішено дозволити визначати у кількості днів,

але не обмежувати конкретним набором варіантів. Таким чином, індивідуальні власники, що формують пропозиції у вебзастосунку матимуть можливість визначати власні пропозиції гнучко. Для пошуку вирішено запропонувати більшу кількість фільтрів. Зокрема це фільтри, які стосуються складу (матеріалів, з яких виготовлено) сукні. Також вирішено надати можливість створення різних власних переліків, у які користувачі можуть групувати різні сукні, таким чином формуючи варіанти під різні події, на різні випадки. Окремою можливістю є прогнозування вартості прокату, що може використовуватися користувачами для встановлення розуміння вартості, на яку за загальними власними критеріями слід орієнтуватися, а також власниками суконь для того, щоб встановити адекватну ціну на прокат.

1.3 Висновки за розділом 1

Виконання роботи має забезпечувати досягнення таких завдань:

- виконати аналіз програмних застосунків прокату суконь;
- розробити метод визначення ціноутворення під час прокату суконь;
- виконати проектування вебзастосунку прокату суконь;
- реалізувати вебзастосунок прокату суконь;
- виконати експериментальне дослідження прогнозування вартості прокату суконь.

Розділ 1 містить огляд вебзастосунків прокату суконь з визначенням основних особливостей і можливостей та порівняння цих застосунків між собою і з вебзастосунком, що створюється. Визначено його особливості, що включають можливість прогнозування вартості прокату в програмі, створення користувацьких переліків суконь, надання розширених фільтрів під час організації пошуку.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача визначення ціноутворення під час прокату суконь

Прокат суконь за своєю суттю є доволі врівноваженим процесом, який базується на попиті та пропозиції. Вони визначають вартість прокату, за яку фактично орендодавець може віддати сукню в прокат та отримати фінансову вигоду, а орендар відповідно згоден взяти сукню у прокат. Якщо ціна прокату є завищеною, то потенційні орендарі відмовляться від цієї послуги в такому варіанті, зокрема за рахунок іншої пропозиції, яку вони вважають кращою. Відповідно процес формування такої пропозиції є доволі складним та потребує певного досвіду. Для цього потрібно уважно вивчити ринок, зрозуміти позицію власних суконь на ринку і тоді винести зважене рішення. Таким чином, не маючи такого досвіду, винести рішення достатньо складно.

На підтримку такої діяльності і направлена задача визначення ціноутворення під час прокату суконь. Ця задача за своєю суттю є задачею прогнозування або задачею регресії. Це означає, що вона передбачає прогнозування вартості прокату сукні на основі значень різних параметрів, якою ця сукня характеризується.

При постановці задачі визначення ціноутворення під час прокату суконь було використано структуру, закладену у вибірку даних [6]. У цій вибірці кожна сукня представлена наступними характеристиками:

- порядковий номер запису про сукню;
- унікальний ідентифікатор сукні;
- назва сукні (фактично назва оголошення про прокат сукні);
- бренд, який представляє ця сукня;
- колір, який має ця сукня;
- перелік категорій, до яких належить сукня;
- вартість прокату сукні.

Відповідна остання ознака, вартість доби прокату сукні, є цільовою ознакою, тобто величиною, прогнозування якої має бути здійснено.

У свою чергу інші ознаки представлено переважно текстовими значеннями. Відповідно під час розв'язання задачі визначення ціноутворення під час прокату суконь необхідно визначити варіант перетворення цих значень на ті, які можуть бути передані у відповідну математичну модель. Також необхідно вирішити питання стосовно тих ознак, які мають бути подані на вхід цієї моделі. Це пов'язано з тим, що частина тих ознак, які є у вибірці даних, можуть бути неінформативними. З іншого боку у вибірці представлену ознаку, яка визначає категорії суконь. Для частини суконь це значення представлено записом, який відповідає сукням загалом. Але для іншої частини міститься перелічений набір категорій. Відповідно додатковим питанням є те, як така ознака має бути опрацьована та змінена для роботи моделі.

Таким чином, задача визначення ціноутворення під час прокату суконь полягає в тому, що визначити функціональну залежність, яка дозволяє перетворити вхідні дані екземплярів суконь, виставлених на прокат, на значення вартості доби прокату.

Дана функціональна залежність має встановлювати також і спосіб перетворення вхідних даних на вхідні ознаки моделі, що реалізує безпосередньо прогнозування.

При цьому є набір відомих екземплярів суконь, для яких вже визначено параметри прокату. Кожен такий екземпляр даних відповідає оголошенню прокату. За кожним екземпляром даних відомі значення вхідних даних та значення цільової ознаки. Відповідно задача визначення ціноутворення під час прокату суконь полягає у визначенні функціональної залежності, яка за відомими екземплярами вибірки даних суконь у прокаті здатна за результатами навчання прогнозувати вартість прокату, а для нових екземплярів суконь у прокаті встановлювати значення вартості прокату за навченої таким чином функціональною залежністю.

Враховуючи, що дана задача визначення ціноутворення під час прокату суконь є задачею прогнозування, існує цілий ряд моделей, які можуть застосовуватися для виконання такого прогнозування, тож розв'язання цієї

задачі передбачає також і визначення моделі, яка буде застосована. У подальшому під час проведення експериментального дослідження прогнозування вартості прокату суконь на основі наявної вибірки даних необхідно перевірити, що такий метод цілісно та обрана модель зокрема здатні призводити до отримання придатних для практичного використання результатів. У такому випадку навчена на основі створеного у результаті використання методу модель може бути використана для орендодавців при створенні оголошення про прокат сукні, а також орендарями для розуміння обґрунтованої вартості прокату сукні за необхідними їм параметрами.

2.2 Метод визначення ціноутворення під час прокату суконь

Метод визначення ціноутворення під час прокату суконь є рішенням описаної у попередньому підрозділі задачі. Його використання потребує застосування послідовності етапів.

На першому етапі має бути отримано або зібрано набір даних екземплярів суконь у прокаті, для яких відомі значення щонайменше наступних характеристик:

- бренд, який представляє ця сукня;
- колір, який має ця сукня;
- перелік категорій, до яких належить сукня;
- вартість прокату сукні.

Додатково можуть бути надані значення також і інших характеристик сукні.

На другому етапі має бути реалізовано оброблення отриманих на першому етапі даних з відповідним їх опрацюванням.

Для цього спершу наступні характеристики мають бути перетворені на категоріальні:

- бренд, який представляє ця сукня;
- колір, який має ця сукня.

Для такого перетворення визначається набір всіх наявних брендів, а також додаються ті бренди, які потенційно можуть з'явитися, якщо така ситуація можлива. Далі кожному бренду присвоюється порядковий номер. Цей порядковий номер і є категоріальним значенням, яке далі у відповідності з брендом сукні встановлюється кожному екземпляру даних.

Далі так само визначається набір наявних кольорів суконь у прокаті, для кожного з можливих значень встановлюється порядковий номер, а тоді цей порядковий номер присвоюється у якості значення вхідної ознаки кожного екземпляра даних, представленого сукнею.

У результаті кожен екземпляр даних представлений брендом сукні, кольором та вартістю. Останнє значення є цільовою ознакою і не перетворюється ніяким чином. Окрім ситуації, коли вартість визначена в різній валюті, тоді такі значення мають бути зведені до представлення у одній і тій самій валюті.

Далі виконується обробка категорій суконь, представлена відповідним рядком. З рядка має бути виділено всі можливі значення категорій. Тоді має бути створено перелік можливих категорій. Після цього кожна сукня як екземпляр даних має бути проаналізована щодо наявного переліку категорій, після чого для кожної виділеної у переліку можливих категорій категорії має бути встановлено значення, що дорівнює 0 або 1.

Значення 0 встановлюється у випадку, якщо у переліку категорій для цієї сукні відсутня відповідна категорія суконь.

Значення 1 встановлюється у випадку, якщо у переліку категорій для цієї сукні присутня відповідна категорія суконь.

Тоді кожен екземпляр суконь має додатково до представлених вище 3 ознак, ода з яких є цільовою, додатково ще набір ознак. Усі ці ознаки використовуються як вхідні ознаки та відповідно одна цільова ознака. Значення кожної з таких ознак має бути задано для кожного екземпляра вибірки даних. Таким чином формується у підсумку навчальна вибірка даних, яка представлена відомими оголошеннями прокату суконь.

На наступному етапі здійснюється відбір ознак, які є найбільш інформативними, для формування набору вхідних ознак моделі прогнозування. Для цього кожна вхідна ознака має бути оцінена стосовно впливу на вартість прокату сукні та обрано ті, які мають найбільший вплив. В якості основи для такого порівняння можуть бути використані значення індексу Джині. Таким чином, визначаючи, які ознаки призводять до кращого розподілу в дереві ухвалення рішень для більш точного прогнозування вартості прокату суконь на виході.

На наступному етапі створюється модель прогнозування вартості прокату суконь. Дану модель запропоновано створювати на основі алгоритму Classification and Regression Tree [7], організовуючи відповідне дерево ухвалення рішень.

За результатами експериментального дослідження фактично ця модель створювалась на основі наступної множини вхідних ознак:

- категоріальне представлення бренду сукні;
- категоріальне представлення кольору сукні;
- позначка про те, чи є ця сукня міні-сукнею.

Для створеного таким чином дерева ухвалення рішень необхідно реалізувати навчання на основі використання середньої абсолютної похибки (САП) як критерія навчання.

Маючи навчену модель для прогнозування вартості прокату суконь, необхідно її використовувати у подальшому при отриманні нових оголошень про прокат суконь. При цьому для кожного оголошення має бути встановлено значення для кожної ознаки, яка виділена як вхідна для моделі. У даному випадку це категоріальне представлення бренду сукні (на основі переліку з можливими брендами суконь, для яких встановлено у відповідність порядкові номери), категоріальне представлення кольору сукні та позначка про те, чи є ця сукня міні-сукнею (1 якщо так, 0 – якщо ні). Тоді організовані таким чином дані сукні у вигляді екземпляру даних мають бути подані на вхід моделі, а на

виході має бути отримано як результат прогнозоване значення вартості прокату сукні.

2.3 Висновки за розділом 2

Наукова новизна роботи полягає в методі визначення ціноутворення під час прокату суконь, який визначає процес формування вхідних ознак, виділення найбільш інформативних з них, створення моделі на основі дерева ухвалення рішень, що дозволяє прогнозувати вартість прокату сукні.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Визначення сценаріїв роботи користувачів з вебзастосунком прокату суконь

Усі користувачі категоризовані на наступні ролі:

- користувач з правами орендаря суконь;
- користувач з правами виставляти сукні на прокат;
- користувач з правами представництва сервісу прокату суконь.

Користувач з правами орендаря суконь має такі основні сценарії:

- надання інформації про сукню загалом;
- надання інформації про сукню у варіанті, доступному для прокату;
- пошук за дизайнером суконь;
- надання інформації про дизайнера суконь;
- відтворення пропонованих для прокату суконь за категорією;
- відтворення пропонованих для прокату суконь за колекцією;
- відтворення пропонованих для прокату суконь за брендом;
- відтворення пропонованих для прокату суконь за дизайнером;
- відтворення пропонованих для прокату суконь за власником;
- відтворення пропонованих для прокату суконь за стилем;
- надання інформації про бренд суконь;
- надання інформації про колекцію суконь;
- пошук серед колекцій суконь;
- пошук сукні для прокату;
- пошук за брендом суконь;
- відтворення пропонованих для прокату варіантів сукні;
- надання інформації про власника суконь;
- відтворення запитів на прокат суконь, які створив користувач;
- відтворення архіву прокату суконь користувачем;
- відтворення запланованих прокатів суконь користувачем;
- створення запиту на прокат сукні;

- створення відгуку про сукню в прокаті;
- надання інформації про запит на прокат або прокат сукні (рис. 3.1).

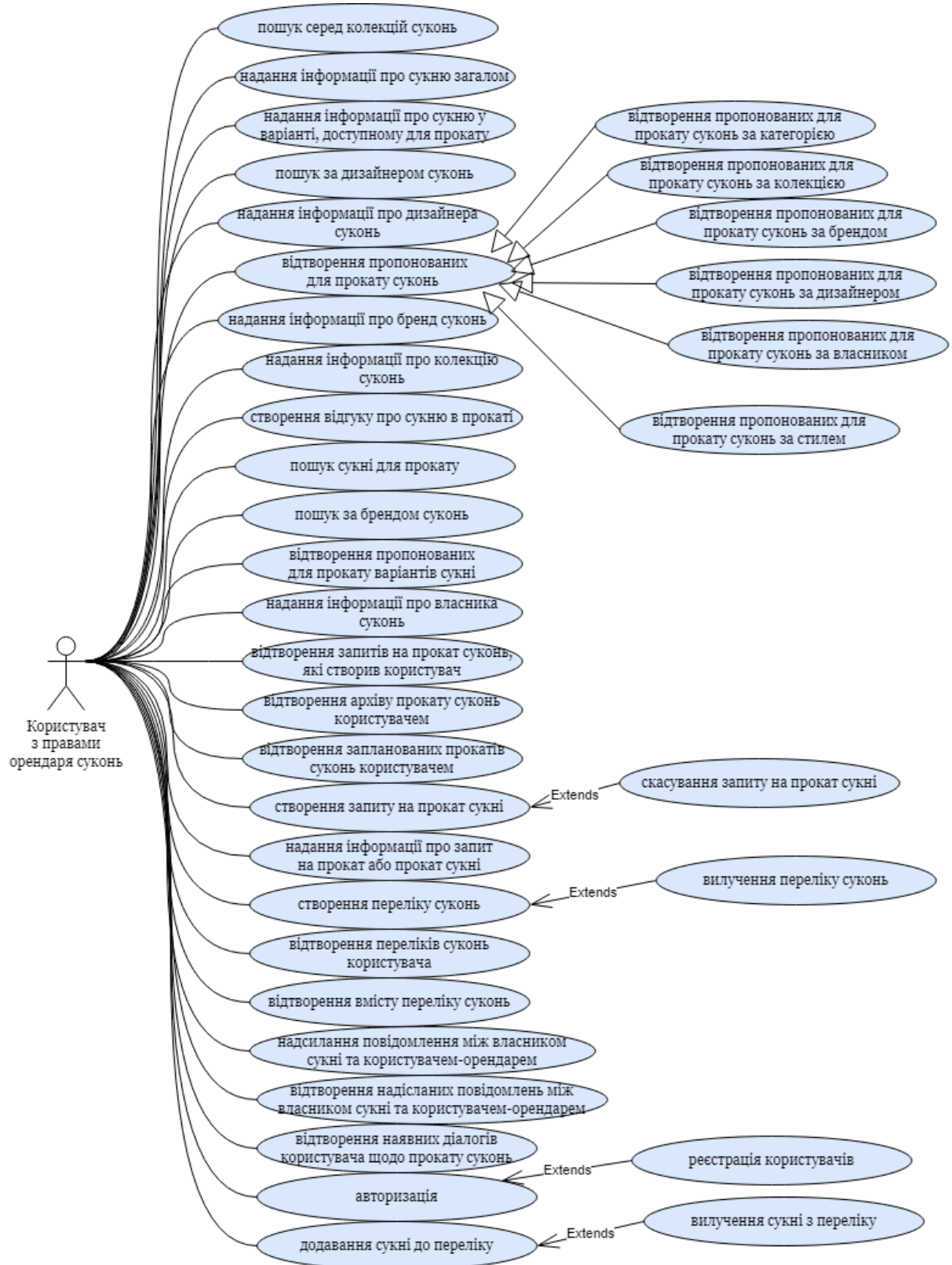


Рисунок 3.1 – Діаграма прецедентів для користувача з правами орендаря суконь

Інші сценарії основного потоку:

- створення переліку суконь;
- відтворення переліків суконь користувача;
- відтворення вмісту переліку суконь;
- надсилання повідомлення між власником сукні та користувачем-орендарем;
- відтворення надісланих повідомлень між власником сукні та користувачем-орендарем;
- відтворення наявних діалогів користувача щодо прокату суконь;
- авторизація;
- додавання сукні до переліку.

Альтернативні сценарії для користувача з правами орендаря суконь:

- реєстрація користувачів, якщо користувач не має зареєстрованого облікового запису, відповідно є альтернативним сценарієм для авторизації;
- вилучення сукні з переліку, якщо така сукня у цей же перелік цього ж користувача вже була додана, відповідно є альтернативним для додавання сукні до переліку;
- вилучення переліку суконь, якщо такий перелік більше не потрібний, відповідно є альтернативним сценарієм для створення переліку суконь;
- скасування запиту на прокат сукні, якщо такий запит створений, проте не був підтверджений власником сукні і вже не відповідає потребам, відповідно є альтернативним сценарієм для створення запиту на прокат сукні.

Користувач з правами виставляти сукні на прокат має такі основні сценарії:

- надання інформації про сукню загалом;
- надання інформації про сукню у варіанті, доступному для прокату;
- пошук за дизайнером суконь;
- надання інформації про дизайнера суконь;
- відтворення пропонованих для прокату суконь за категорією;
- відтворення пропонованих для прокату суконь за колекцією;

- відтворення пропонованих для прокату суконь за брендом;
- відтворення пропонованих для прокату суконь за дизайнером;
- відтворення пропонованих для прокату суконь за власником;
- відтворення пропонованих для прокату суконь за стилем;
- надання інформації про бренд суконь;
- надання інформації про колекцію суконь;
- пошук серед колекцій суконь;
- пошук сукні для прокату;
- пошук за брендом суконь;
- відтворення пропонованих для прокату варіантів сукні;
- надання інформації про власника суконь;
- надання інформації про запит на прокат або прокат сукні;
- надсилання повідомлення між власником сукні та користувачем-орендарем;
- відтворення надісланих повідомлень між власником сукні та користувачем-орендарем;
- відтворення наявних діалогів користувача щодо прокату суконь;
- авторизація;
- виставлення сукні на прокат;
- внесення даних про вартість прокату сукні;
- відтворення запитів на прокат суконь користувача;
- відтворення архіву прокату за сукнею у варіанті, доступному для прокату;
- відтворення запланованих прокатів за сукнею у варіанті, доступному для прокату;
- створення відгуку про орендаря;
- відтворення запланованих прокатів за сукнями користувача;
- відтворення переліку робіт за сукнею у прокаті;
- внесення даних про роботи над сукнею у прокаті;
- підтвердження запиту на прокат сукні (рис. 3.2).

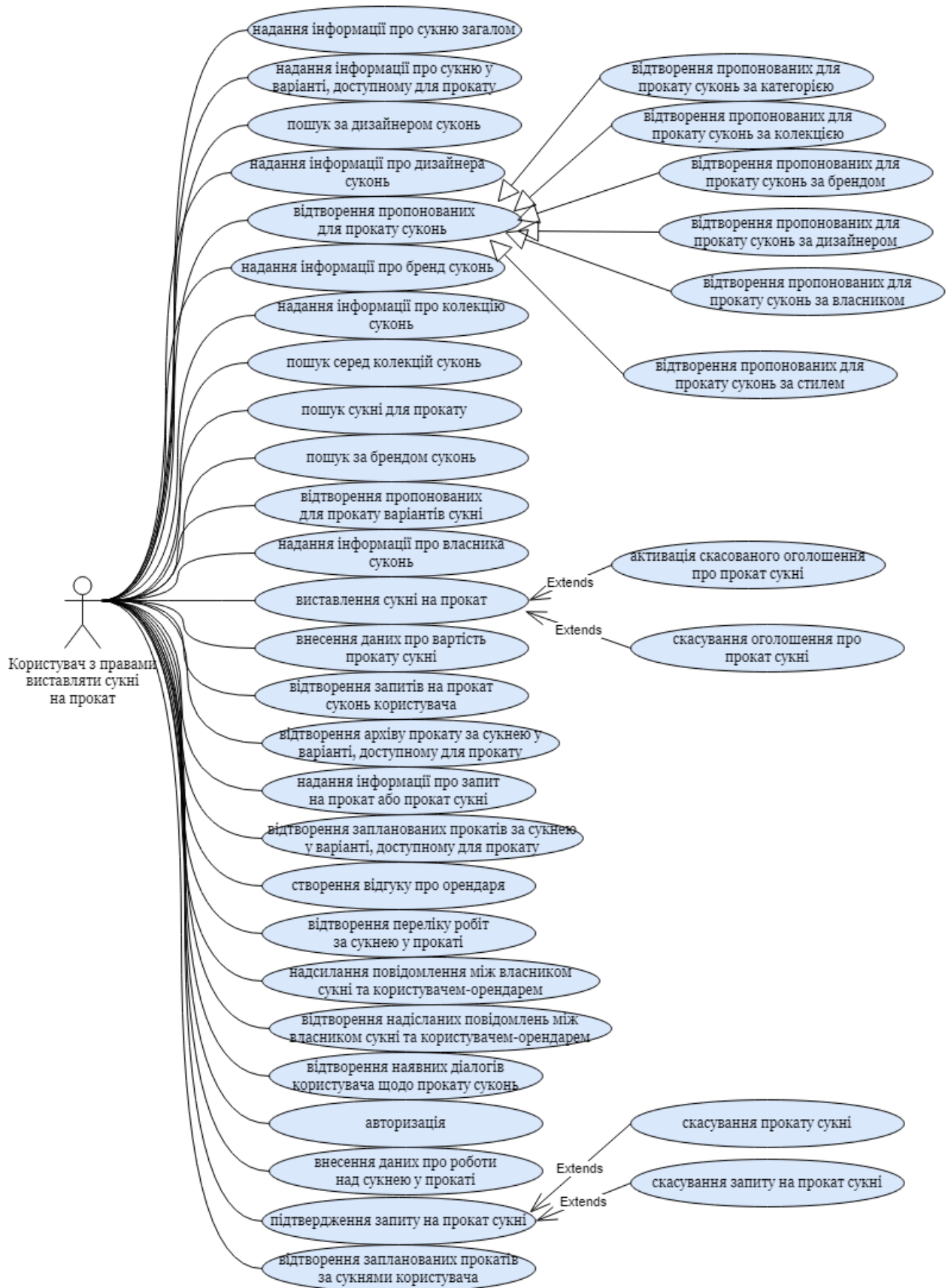


Рисунок 3.2 – Діаграма прецедентів для користувача з правами виставляти сукні на прокат

Альтернативні сценарії для користувача з правами виставляти сукні на прокат:

- скасування запиту на прокат сукні, якщо такий запит створений, проте власник не бажає орендувати сукню такому користувачу, відповідно є альтернативним сценарієм для підтвердження запиту на прокат сукні;

- скасування прокату сукні, якщо прокат не може бути виконаний, хоча був підтверджений раніше, відповідно є альтернативним сценарієм для підтвердження запиту на прокат сукні;

- скасування оголошення про прокат сукні, якщо власник більше не хоче виставляти сукню на прокат, відповідно є альтернативним сценарієм для виставлення сукні на прокат;

- активація скасованого оголошення про прокат сукні, якщо власник хоче повернути сукню в прокат після попереднього скасування, відповідно є альтернативним сценарієм для виставлення сукні на прокат.

Користувач з правами представництва сервісу прокату суконь має такі основні сценарії:

- надання інформації про сукню загалом;
- надання інформації про сукню у варіанті, доступному для прокату;
- пошук за дизайнером суконь;
- надання інформації про дизайнера суконь;
- відтворення пропонованих для прокату суконь за категорією;
- відтворення пропонованих для прокату суконь за колекцією;
- відтворення пропонованих для прокату суконь за брендом;
- відтворення пропонованих для прокату суконь за дизайнером;
- відтворення пропонованих для прокату суконь за власником;
- відтворення пропонованих для прокату суконь за стилем;
- надання інформації про бренд суконь;
- надання інформації про колекцію суконь;
- пошук серед колекцій суконь;
- пошук сукні для прокату;

- пошук за брендом суконь;
- відтворення пропонованих для прокату варіантів сукні;
- надання інформації про власника суконь;
- надання інформації про запит на прокат або прокат сукні;
- надсилання повідомлення;
- відтворення надісланих повідомлень до користувачів-орендарів;
- відтворення наявних діалогів користувача щодо прокату суконь;
- авторизація;
- внесення даних дизайнера суконь;
- внесення даних про колекцію суконь;
- внесення даних про бренд суконь;
- внесення даних про сукню;
- виставлення сукні на прокат;
- внесення даних про вартість прокату сукні;
- відтворення запитів на прокат суконь сервісу;
- відтворення архіву прокату за сукнею у варіанті, доступному для прокату;
- відтворення запланованих прокатів за сукнею у варіанті, доступному для прокату;
- відтворення запланованих прокатів за сукнями сервісу;
- пошук за користувачами, які є власниками суконь у прокаті;
- підтвердження запиту на прокат сукні;
- відтворення переліку робіт за сукнею у прокаті;
- внесення даних про роботи над сукнею у прокаті;
- створення відгуку про орендаря;
- надання користувачу права виставляти сукні в прокат (рис. 3.3).

Сценарії внесення даних фактично передбачають реалізацію у вигляді додавання нових даних та редагування цих даних тоді, коли вони вже є у базі даних, а їх потрібно далі змінювати.

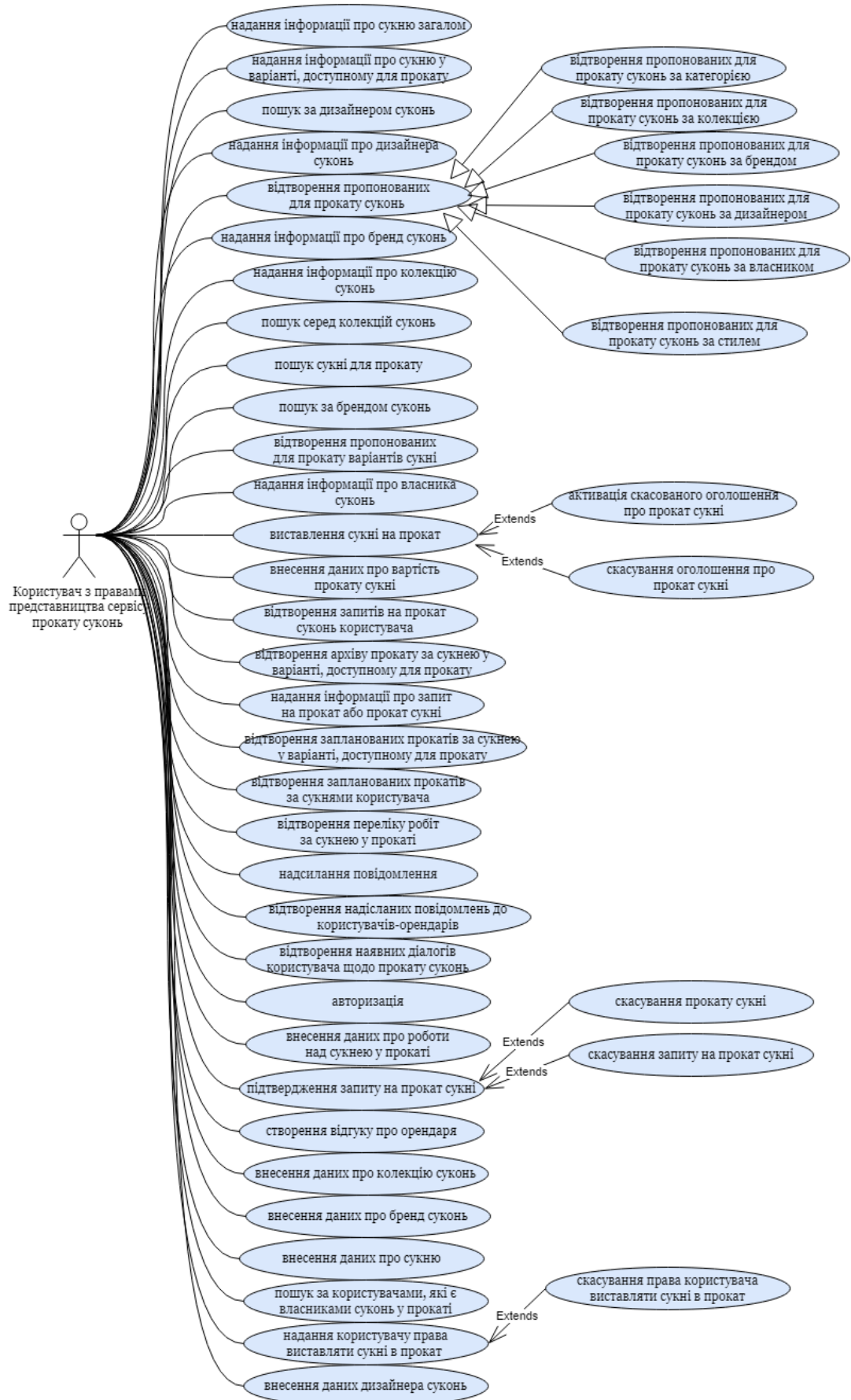


Рисунок 3.3 – Діаграма прецедентів для користувача з правами представництва сервісу з прокату суконь

Альтернативні сценарії для користувача з правами представництва сервісу прокату суконь:

- скасування запиту на прокат сукні, якщо такий запит створений, проте сервіс не бажає орендувати сукню такому користувачу, відповідно є альтернативним сценарієм для підтвердження запиту на прокат сукні;
- скасування прокату сукні, якщо прокат не може бути виконаний, хоча був підтверджений раніше, відповідно є альтернативним сценарієм для підтвердження запиту на прокат сукні;
- скасування оголошення про прокат сукні, якщо сервіс більше не хоче виставляти сукню на прокат, відповідно є альтернативним сценарієм для виставлення сукні на прокат;
- активація скасованого оголошення про прокат сукні, якщо сервіс хоче повернути сукню в прокат після попереднього скасування, відповідно є альтернативним сценарієм для виставлення сукні на прокат;
- скасування права користувача виставляти сукні в прокат, якщо таке право раніше було надано, але більше не відповідає потребам, відповідно є альтернативним сценарієм для надання користувачу права виставляти сукні в прокат.

3.2 Вибір засобів розробки вебзастосунку прокату суконь

При формуванні альтернатив для реалізації вебзастосунку прокату суконь одразу було враховано, що у цьому вебзастосунку має бути реалізовано роботу з визначення ціноутворення під час прокату суконь, відповідно створено, навчено і інтегровано навчену модель. Це потребує широких можливостей Python-пакетів для машинного навчання, оскільки ще й заплановане експериментальне дослідження з додатковими методами прогнозування. Тому вибір засобів реалізації вебзастосунку прокату суконь виконувався з фреймворків для веброзробки на Python, включаючи Django [8]

як найбільш потужний фреймворк, що і стало в підсумку основою його вибору для реалізації, і Falcon [9] як один з легковісних фреймворків (табл. 3.1).

Таблиця 3.1 – Порівняння можливих рішень для реалізації вебзастосунку прокату суконь

Ознака порівняння	Django	Falcon
Можливості адміністративної роботи з каталогом даних	Присутні, використовуються після реєстрації створених моделей	Відсутні
Програмна робота з базою даних	Присутня для різних систем управління	Відсутня, необхідно вибирати, встановлювати
Надання безпеки створеному вебзастосунку	Широке охоплення за замовчуванням	Необхідно передбачати і додавати
Можливість реалізації методу визначення ціноутворення під час прокату суконь	Мовою Python з використанням відповідних пакетів	Мовою Python з використанням відповідних пакетів

3.3 Проєктування структури бази даних

Проєктування, а в подальшому реалізацію роботи з базою даних виконано на основі використання системи управління MySQL, яка зарекомендувала себе на практиці протягом тривалого періоду часу, має широкі засоби та можливості, здатна підтримувати потужні вебзастосунки, є достатньо розповсюдженою, що сприяє різним варіантам хостингу вебзастосунку, при цьому є безкоштовною. Далі, виходячи з використання MySQL, визначена структура створених таблиць бази даних.

Таблиця Designer необхідна була для збереження даних про дизайнерів, які створюють сукні, включаючи наступні поля для представлення даних про них:

- id – унікальний числовий номер дизайнера суконь;
- slug – унікальний текстовий ідентифікатор, утворений за ім'ям дизайнера суконь або його псевдонімом для використання в адресі при зверненнях до вебсторінок;
- pseudo – повне ім'я дизайнера суконь або його псевдонім;
- presentation – опис діяльності, біографії дизайнера суконь;
- photo – фотографія, де представлений дизайнер суконь.

Таблиця Dresscategory необхідна була для збереження категорій суконь, включаючи наступні поля для представлення даних про них:

- id – унікальний числовий номер категорії суконь;
- slug – унікальний текстовий ідентифікатор, утворений за назвою категорії суконь для використання в адресі при зверненнях до вебсторінок;
- categoryname – повна назва категорії суконь.

Таблиця Collection необхідна була для збереження даних про колекції суконь, включаючи наступні поля для представлення даних про них:

- id – унікальний числовий номер колекції суконь;
- appscaption – назва колекції суконь, якою вона відображається в короткій формі у вебзастосунку, тобто в тих позиціях у інтерфейсі, де необхідна коротка форма;
- fulltitle – повна назва колекції суконь;
- slug – унікальний текстовий ідентифікатор, утворений за назвою колекції суконь для використання в адресі при зверненнях до вебсторінок;
- brand – бренд, який виготовляв цю колекцію суконь, забезпечений через зв'язок з таблицею Brand у даному полі як зовнішньому ключі за типом один-до-багатьох;
- presentation – текстовий опис особливостей цієї колекції суконь;
- photo – фотографія для ілюстрації колекції суконь;

- startseason – рік, з якого почалось виготовлення цієї колекції суконь;
- endseason – рік, коли закінчилось виготовлення цієї колекції суконь;
- seasontype – тип сезону, якому відповідає колекція (якщо це саме колекція під один конкретний сезон року).

Таблиця Brand необхідна була для збереження даних про бренди суконь, включаючи наступні поля для представлення даних про них:

- id – унікальний числовий номер бренду суконь;
- title – назва бренду суконь;
- website – вебсайт бренду суконь.

Таблиця Dress необхідна була для збереження даних про сукні, включаючи наступні поля для представлення даних про них:

- id – унікальний числовий номер сукні;
- dresscategory – категорія, до якої належить ця сукня, забезпечена через зв'язок з таблицею Dresscategory у даному полі як зовнішньому ключі за типом один-до-багатьох;
- collection – колекція, до якої належить ця сукня, забезпечена через зв'язок з таблицею Collection у даному полі як зовнішньому ключі за типом один-до-багатьох;
- designer – дизайнер, який створив цю сукню, забезпечений через зв'язок з таблицею Designer у даному полі як зовнішньому ключі за типом один-до-багатьох;
- slug – унікальний текстовий ідентифікатор, утворений за назвою сукні для використання в адресі при зверненнях до вебсторінок;
- appcaption – назва, якою ілюструється ця сукня для відображення у вебзастосунку;
- colour – колір сукні;
- lengthtype – тип довжини сукні;
- sleevetype – тип довжини рукава сукні;
- description – текстовий опис, що ілюструє сукню.

Таблиця Dressphoto необхідна була для збереження фотографій, які ілюструють сукню загалом (не конкретний екземпляр, який є у власника, а саме сукню загалом), включаючи наступні поля для представлення даних про них:

- id – унікальний числовий номер фотографії сукні;
- dress – сукня, яка показана на фотографії, забезпечена через зв'язок з таблицею Dress у даному полі як зовнішньому ключі за типом один-до-багатьох;
- photo – фотографія, що ілюструє цю сукню, представлена збереженими у полі даними.

Таблиця Dresscomposition необхідна була для збереження даних про склад сукні (матеріали, з яких сукня виготовлена), включаючи наступні поля для представлення даних:

- id – унікальний числовий номер позиції у складі сукні;
- dress – сукня, до складу якої входить цей матеріал, забезпечена через зв'язок з таблицею Dress у даному полі як зовнішньому ключі за типом один-до-багатьох;
- material – матеріал, з якого виготовлена ця сукня, забезпечений через зв'язок з таблицею Material у даному полі як зовнішньому ключі за типом один-до-багатьох;
- percent – відсоток вмісту матеріалу в складі цієї сукні;
- dresspart – частина, до якої входить цей матеріал (наприклад, верх сукні та низ може бути виготовлений з різних матеріалів).

Таблиця Material необхідна була для збереження даних про матеріали, з яких можуть виготовлятися сукні, представлені у вебзастосунку, включаючи наступні поля для представлення даних:

- id – унікальний числовий номер матеріалу;
- name – назва матеріалу.

Таблиця Dresssize необхідна була для збереження даних про конкретні сукні (з вказаними розмірами), представлені у вебзастосунку, включаючи наступні поля для представлення даних:

- id – унікальний числовий номер конкретної сукні з розміром;
- dress – загальна сукня (без деталізації розміру та права власності), яку представляє цей екземпляр з розміром та власництвом, забезпечена через зв'язок з таблицею Dress у даному полі як зовнішньому ключі за типом один-до-багатьох;
- dressowner – власник цього екземпляру сукні, забезпечений через зв'язок з таблицею Rentuser у даному полі як зовнішньому ключі за типом один-до-багатьох;
- size – конкретний розмір, який має ця сукня;
- fit – варіант посадки, якому відповідає ця сукня (мініатюрна, висока тощо);
- rank – середній бал, у який цю конкретно сукню у такому розмірі та у цього власника оцінено орендарями;
- quantity – кількість екземплярів сукні в цьому розмірі у цього власника;
- active – активне оголошення або вже архівне;
- address – місто або додаткова адреса, де ця сукня доступна.

Таблиця Dresssizephoto необхідна була для збереження даних про фотографії конкретних екземплярів суконь (з вказанням розміру та власництва), представлені у вебзастосунку, включаючи наступні поля для представлення даних:

- id – унікальний числовий номер фотографії конкретної сукні з розміром;
- dresssize – сукня в конкретному розмірі та у конкретного власника, яка показана на фотографії, забезпечена через зв'язок з таблицею Dresssize у даному полі як зовнішньому ключі за типом один-до-багатьох;

– photo – фотографія, що ілюструє цей розмір сукні у даного власника, представлена збереженими у полі даними.

Таблиця Dressprice необхідна була для збереження даних про вартість прокату конкретної сукні (з вказаними розмірами), визначену її власником, включаючи наступні поля для представлення даних:

– id – унікальний числовий номер визначеного варіанта вартості прокату сукні в даному розмірі у заданого власника;

– dresssize – сукня в конкретному розмірі та у конкретного власника, для якої встановлено цю пропозицію прокату за даною вартістю, забезпечена через зв'язок з таблицею Dresssize у даному полі як зовнішньому ключі за типом один-до-багатьох;

– typical – дана пропозиція є типовою, тобто використовується для всіх розмірів цієї сукні у того ж власника, чи власник визначає окремо вартість прокату для кожного варіанта;

– mindays – мінімальна кількість днів прокату сукні, за якої починає діяти цей варіант вартості прокату;

– price – вартість дня прокату цієї сукні в заданому розмірі та у заданого власника, яка в такому випадку діє;

– startperiod – дата, з якої цей варіант вартості прокату сукні в даному розмірі у заданого власника починає діяти;

– endperiod – дата, до якої цей варіант вартості прокату сукні в даному розмірі у заданого власника діє.

Оскільки сукня може бути не доступна для прокату, то необхідно визначити в такому випадку роботи, які виконуються з сукнею. Таблиця Dresswork необхідна була для збереження даних про виконувані роботи з екземпляром сукні в конкретному розмірі та у конкретного власника, включаючи наступні поля для представлення даних:

– id – унікальний числовий номер робіт, які виконуються з сукнею в конкретному розмірі та у конкретного власника;

– dresssize – сукня в конкретному розмірі та у конкретного власника, з якою виконуються ці роботи, забезпечена через зв'язок з таблицею Dresssize у даному полі як зовнішньому ключі за типом один-до-багатьох;

– startwork – дата початку робіт, які виконуються з сукнею в конкретному розмірі та у конкретного власника та через які ця сукня не буде доступна;

– endwork – дата закінчення робіт, які виконуються з сукнею в конкретному розмірі та у конкретного власника та через які ця сукня не буде доступна;

– worktype – тип робіт, які виконуються з цією сукнею в конкретному розмірі та у конкретного власника.

Таблиця Dressrental необхідна була для збереження даних про замовлення прокату суконь, включаючи наступні поля для представлення даних:

– id – унікальний числовий номер замовленого прокату суконь;

– rentuser – користувач, який замовив цей прокату суконь, забезпечений через зв'язок з таблицею Rentuser у даному полі як зовнішньому ключі за типом один-до-багатьох;

– createdon – дата і час створення замовленого прокату суконь;

– state – поточний стан цього замовленого прокату суконь;

– deliveryon – дата і час доставки (запланованої або виконаної) замовленого прокату суконь;

– deliveryto – дата і час, до яких має тривати замовлений прокат суконь;

– price – вартість замовленого прокату суконь;

– rentuserreview – відгук власника суконь у замовленні про користувача, який замовив прокат;

– rentuserreviewon – дата і час створення відгуку власника суконь у замовленні про користувача, який замовив прокат;

– rentuserreviewrank – бал, виставлений у відгуку власника суконь у замовленні про користувача, який замовив прокат.

Таблиця Dressrentalcontent необхідна була для збереження даних про конкретний вміст замовлення прокату суконь, включаючи наступні поля для представлення даних:

- id – унікальний числовий номер позиції в складі замовленого прокату суконь;
- dressrental – замовлений прокат суконь, вміст якого розкривається цією позицією, забезпечений через зв'язок з таблицею Dressrental у даному полі як зовнішньому ключі за типом один-до-багатьох;
- dresssize – сукня в конкретному розмірі та у конкретного власника, що входить до цього замовленого прокату суконь, забезпечена через зв'язок з таблицею Dresssize у даному полі як зовнішньому ключі за типом один-до-багатьох;
- dressreview – відгук користувача, який замовив прокат, про сукню;
- dressreviewon – дата і час створення відгуку користувача, який замовив прокат, про сукню;
- dressreviewrank – бал, виставлений у відгуку користувача, який замовив прокат, про сукню.

Таблиця Rentuser необхідна була для збереження даних про користувачів вебзастосунку прокату суконь, включаючи наступні поля для представлення даних:

- id – унікальний числовий номер запису про користувача вебзастосунку прокату суконь;
- userinapp – обліковий запис, який використовує цей користувач вебзастосунку прокату суконь;
- fullname – повне ім'я користувача вебзастосунку прокату суконь;
- photo – фотографія, що ілюструє цього користувача вебзастосунку прокату суконь, представлена збереженими у полі даними;
- description – опис користувача вебзастосунку прокату суконь для самопрезентації;

– canbeowner – позначка, що визначає дозвіл користувачу вебзастосунку прокату суконь виставляти для прокату власні сукні;

– rank – середній бал, у який роботу з цим конкретно користувачем вебзастосунку прокату суконь оцінили власники суконь.

Таблиця Dresslist необхідна була для збереження даних про переліки суконь, створені користувачами, включаючи наступні поля для представлення даних:

– id – унікальний числовий номер переліку суконь, створеного користувачем;

– rentuser – користувач вебзастосунку прокату суконь, який створив цей перелік суконь, забезпечений через зв'язок з таблицею Rentuser у даному полі як зовнішньому ключі за типом один-до-багатьох;

– listcaption – назва переліку суконь;

– slug – унікальний текстовий ідентифікатор, утворений за назвою переліку суконь для використання в адресі при зверненнях до вебсторінок.

Таблиця Dresslistcontent необхідна була для збереження даних про вміст переліку суконь, включаючи наступні поля для представлення даних:

– id – унікальний числовий номер позиції у переліку суконь, створеного користувачем;

– dresslist – перелік суконь, створений користувачем і який містить цю позицію, забезпечений через зв'язок з таблицею Dresslist у даному полі як зовнішньому ключі за типом один-до-багатьох;

– dresssize – сукня в конкретному розмірі та у конкретного власника, що входить до переліку користувача, забезпечена через зв'язок з таблицею Dresssize у даному полі як зовнішньому ключі за типом один-до-багатьох.

Таблиця Style необхідна була для збереження даних про можливі стилі суконь, включаючи наступні поля для представлення даних:

– id – унікальний числовий номер стиля сукні;

– name – назва стиля сукні (рис. 3.4).

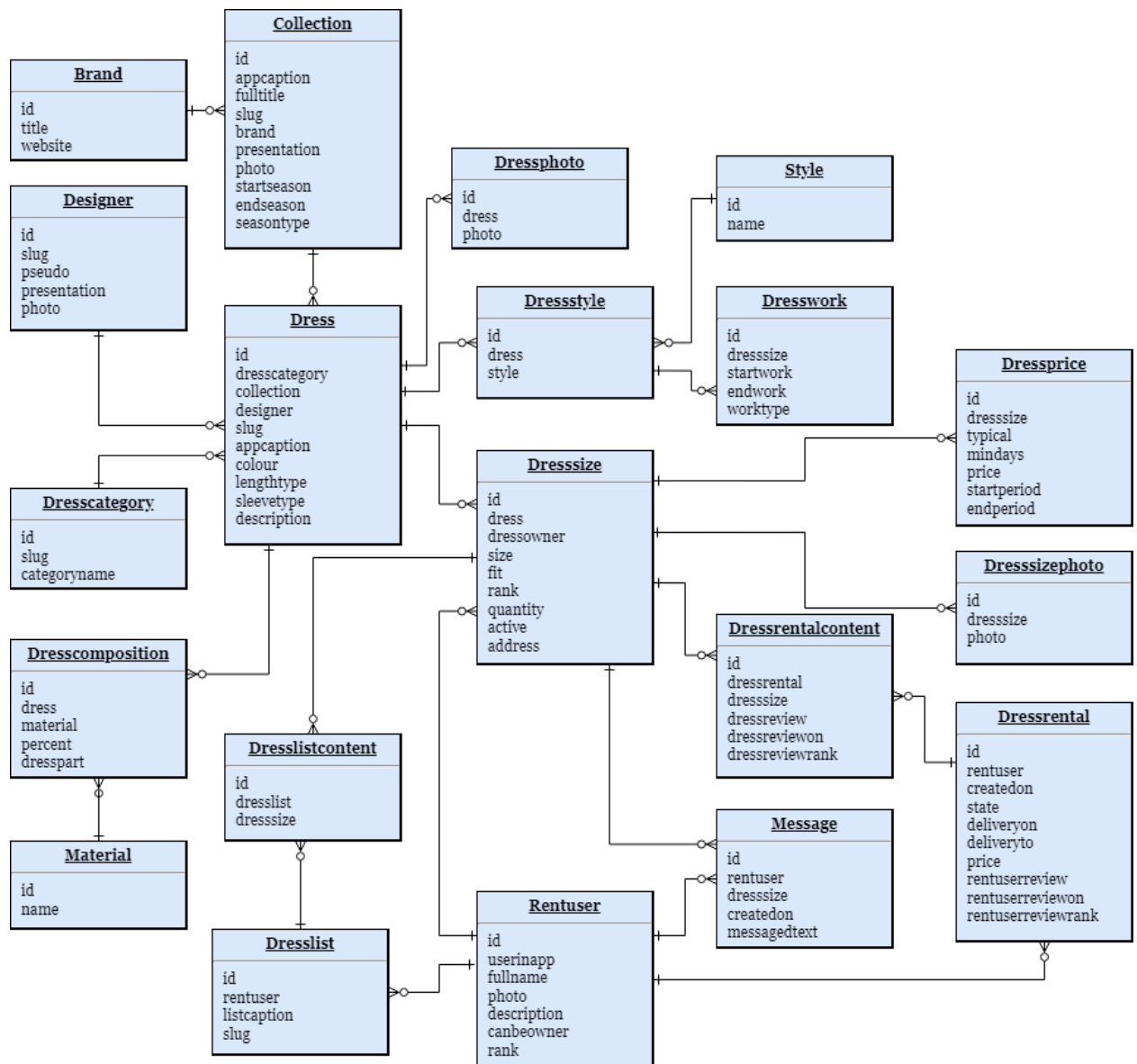


Рисунок 3.4 – Схема бази даних

Таблиця Message необхідна була для збереження даних про повідомлення до власників суконь від орендарів, включаючи наступні поля для представлення даних:

- id – унікальний числовий номер повідомлення до власника суконь від орендаря;
- rentuser – користувач вебзастосунку прокату суконь, який створив це повідомлення, забезпечений через зв'язок з таблицею Rentuser у даному полі як зовнішньому ключі за типом один-до-багатьох;

– `dresssize` – сукня в конкретному розмірі та у конкретного власника, стосовно якої адресовано повідомлення, забезпечена через зв'язок з таблицею `Dresssize` у даному полі як зовнішньому ключі за типом один-до-багатьох;

– `createdon` – дата і час, коли це повідомлення до власника суконь від орендаря було створено;

– `messagedtext` – текст повідомлення до власника суконь від орендаря.

Таблиця `Dressstyle` необхідна була для збереження даних про конкретні стилі конкретних суконь, включаючи наступні поля для представлення даних:

– `id` – унікальний числовий номер відповідності сукні стилю;

– `dress` – сукня, яка відповідає цьому стилю, забезпечена через зв'язок з таблицею `Dress` у даному полі як зовнішньому ключі за типом один-до-багатьох;

– `style` – стиль, якому відповідає ця сукня, забезпечений через зв'язок з таблицею `Dress` у даному полі як зовнішньому ключі за типом один-до-багатьох.

3.4 Висновки за розділом 3

Розділ 3 включає визначення сценаріїв роботи користувачів з вебзастосунком прокату суконь, до яких віднесено користувача з правами орендаря суконь, користувача з правами виставляти сукні на прокат, користувача з правами представництва сервісу прокату суконь, порівняння фреймворків `Falcon` і `Django` та вибір `Django` для реалізації вебзастосунку прокату суконь, визначення структури, зв'язків таблиць у складі бази даних на основі системи `MySQL`.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Загальний порядок роботи програми прокату суконь

Загальний порядок роботи програми прокату суконь визначається доступним на кожній сторінці меню, з якого в залежності від поточної ролі користувача можна вибрати ті чи інші варіанти продовження роботи, кожен раз далі маючи відповідним чином наповнене меню (рис. 4.1).

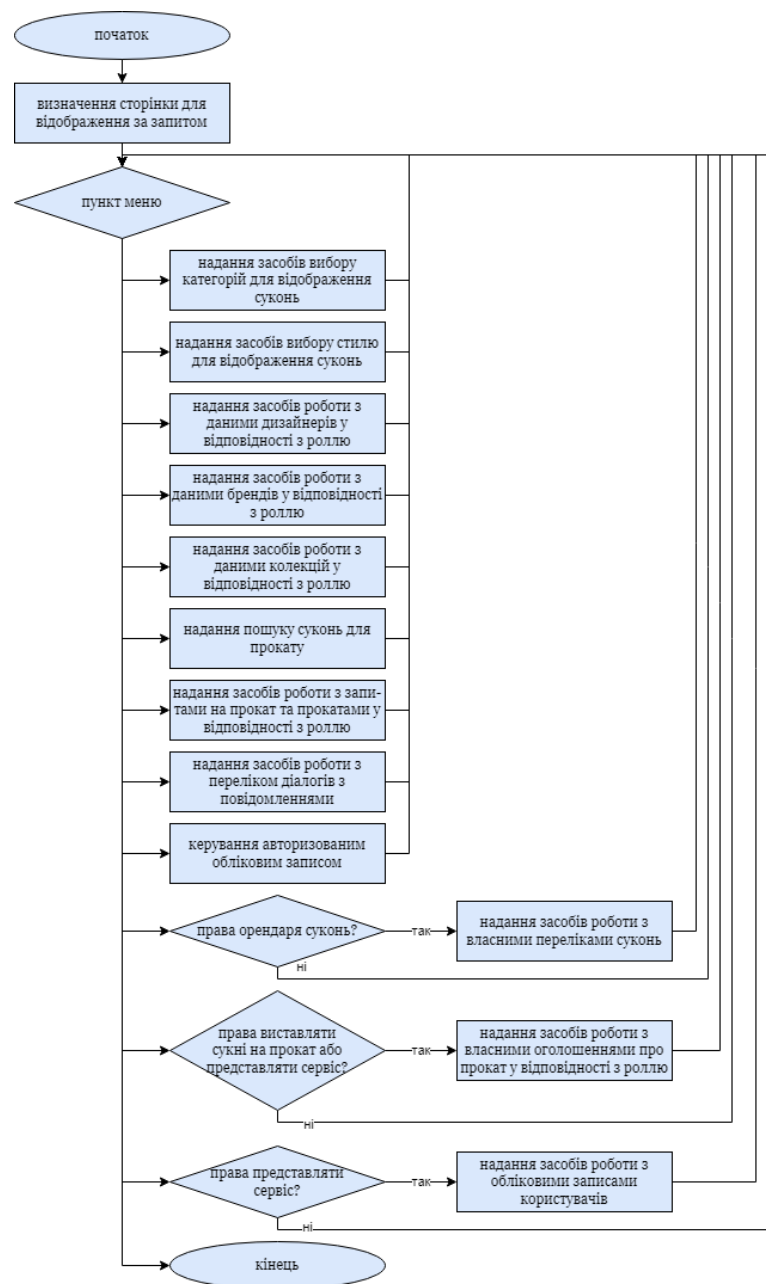


Рисунок 4.1 – Схема організації меню програми прокату суконь

4.2 Реалізовані програмні одиниці у складі вебзастосунку прокату суконь

Реалізацію моделі прогнозування у складі методу визначення ціноутворення під час прокату суконь виконано через клас `DecisionTreeRegressor` з підпакета `tree` пакета `scikit-learn`. Для навчання передано підготовлену вибірку в метод `fit`, а прогнозування реалізовано через виклик методу `predict`. В якості критерію навчання для моделі в параметрі `criterion` задано `absolute_error`, тобто САП. Відбір ознак реалізовано також на основі класу `DecisionTreeRegressor` на основі виклику атрибуту `feature_importances_` за результатами навчання.

Представлення відтворення архіву прокату суконь користувачем реалізовано функцією `view_rentuser_dress_rentals_archive`, при цьому користувач визначається за виконаною авторизацією, а якщо вона не виконана, то перелік прокатів не буде повернуто. Архівними є ті прокати, які були підтверджені, але дата завершення прокату суконь за ними менше поточної. Дані наповнюють шаблон вебсторінки `userarchrentals.html`. Шаблон створений за шаблоном [10] як і наступні.

Представлення відтворення пропонованих для прокату суконь за категорією реалізовано функцією `view_category_dresses`, яка має параметр `dresscategory_slug`, що задає текстовий ідентифікатор потрібної категорії суконь, `orderapp`, що задає напрямок сортування (0 – за назвою за алфавітом, 1 – протилежний напрямок, 2 – за оцінкою, 3 – за оцінкою за зменшенням), `typical`, що за значення `True` призводить до повернення даних загальних суконь (без розміру та власництва), а за значення `False` призводить до повернення даних суконь у варіантах прокату, що пропонуються конкретними власниками. Дані наповнюють шаблон вебсторінки `categ_dresses.html`.

Представлення надання інформації про дизайнера суконь реалізовано функцією `view_designer`, яка має параметр `designer_slug`, що задає текстовий ідентифікатор потрібного дизайнера. Дані включають 20 суконь цього

дизайнера, 20 найкращих пропозицій їх прокату, перелік брендів, для яких робив сукні цей дизайнер. Дані наповнюють шаблон вебсторінки `designer.html`.

Представлення відтворення пропонованих для прокату суконь за колекцією реалізовано функцією `view_collection_dresses`, яка має параметр `collection_slug`, що задає текстовий ідентифікатор потрібної колекції суконь, `orderby`, що задає напрямок сортування (0 – за назвою за алфавітом, 1 – протилежний напрямок, 2 – за оцінкою, 3 – за оцінкою за зменшенням), `typical`, що за значення `True` призводить до повернення даних загальних суконь (без розміру та власництва), а за значення `False` призводить до повернення даних суконь у варіантах прокату, що пропонуються конкретними власниками. Дані наповнюють шаблон вебсторінки `collection_dresses.html`.

Представлення надання інформації про власника суконь реалізовано функцією `view_rentuser`, яка має параметр `rentuser_username`, що задає логін користувача. Якщо логін не встановлено, то використовуються дані авторизації. Дані результатів містять дані профілю, перелік з 20 найкращих суконь у прокаті цього власника. Дані наповнюють шаблон вебсторінки `rentuser_dresses.html`.

Представлення надання інформації про запит на прокат або прокат сукні реалізовано функцією `view_dress_rental`, яка має параметр `dressrental_id`, що задає номер прокату сукні. Дані надаються тільки, якщо користувач, що авторизувався, є власником суконь, які замовлені у прокат, або якщо він сам створив цей запит на прокат, або якщо він є представником сервісу, після чого вони наповнюють шаблон вебсторінки `dressrental.html`.

Представлення відтворення пропонованих для прокату суконь за брендом реалізовано функцією `view_brand_dresses`, яка має параметр `brand_id`, що задає числовий номер потрібного бренду суконь, `orderby`, що задає напрямок сортування (0 – за назвою за алфавітом, 1 – протилежний напрямок, 2 – за оцінкою, 3 – за оцінкою за зменшенням), `typical`, що за значення `True` призводить до повернення даних загальних суконь (без розміру та власництва), а за значення `False` призводить до повернення даних суконь у

варіантах прокату, що пропонуються конкретними власниками. Дані наповнюють шаблон вебсторінки `brand_dresses.html`.

Представлення відтворення запитів на прокат суконь, які створив користувач, реалізовано функцією `view_rentuser_dress_rentals`, при цьому користувач визначається за виконаною авторизацією, а якщо вона не виконана, то перелік запитів не буде повернуто. Дані наповнюють шаблон вебсторінки `userrentals.html`.

Представлення відтворення пропонованих для прокату суконь за дизайнером реалізовано функцією `view_designer_dresses`, яка має параметр `designer_slug`, що задає текстовий ідентифікатор потрібного дизайнера, `orderapp`, що задає напрямок сортування (0 – за назвою за алфавітом, 1 – протилежний напрямок, 2 – за оцінкою, 3 – за оцінкою за зменшенням), `typical`, що за значення `True` призводить до повернення даних загальних суконь (без розміру та власництва), а за значення `False` призводить до повернення даних суконь у варіантах прокату, що пропонуються конкретними власниками. Дані наповнюють шаблон вебсторінки `designer_dresses.html`.

Представлення надання інформації про бренд суконь реалізовано функцією `view_dressbrand`, яка має параметр `brand_id`, що задає номер бренду сукні. Дані включають 20 суконь цього бренду, 20 найкращих пропозицій їх прокату, перелік брендів, для яких робив сукні цей дизайнер. Дані наповнюють шаблон вебсторінки `brand.html`.

Представлення відтворення пропонованих для прокату суконь за власником реалізовано функцією `view_rent_user_dresses`, яка має параметр `rentuser_username`, що задає нік користувача, що є власником суконь (якщо значення не вказано, то використовуються дані авторизації), `orderapp`, що задає напрямок сортування (0 – за назвою за алфавітом, 1 – протилежний напрямок, 2 – за оцінкою, 3 – за оцінкою за зменшенням). Дані наповнюють шаблон вебсторінки `rentuser_dresses.html`.

Представлення відтворення запланованих прокатів суконь користувачем реалізовано функцією `view_rentuser_dress_rentals_planned`, при

цьому користувач визначається за виконаною авторизацією, а якщо вона не виконана, то перелік прокатів не буде повернуто. Дані наповнюють шаблон вебсторінки `userplanrentals.html`.

Представлення відтворення пропонованих для прокату суконь за стилем реалізовано функцією `view_style_dresses`, яка має параметр `style_id`, що задає числовий номер потрібного стилю, `orderapp`, що задає напрямок сортування (0 – за назвою за алфавітом, 1 – протилежний напрямок, 2 – за оцінкою, 3 – за оцінкою за зменшенням), `typical`, що за значення `True` призводить до повернення даних загальних суконь (без розміру та власництва), а за значення `False` призводить до повернення даних суконь у варіантах прокату, що пропонуються конкретними власниками. Дані наповнюють шаблон вебсторінки `style_dresses.html`.

Представлення надання інформації про колекцію суконь реалізовано функцією `view_dresscollection`, яка має параметр `collection_slug`, що задає текстовий ідентифікатор колекції суконь. Дані включають 20 суконь цього дизайнера, 20 найкращих пропозицій їх прокату, перелік дизайнерів, які робили сукні для цієї колекції. Дані наповнюють шаблон вебсторінки `collection.html`.

Представлення відтворення переліків суконь користувача реалізовано функцією `view_rentuser_dress_lists`, беручи дані користувача з авторизації. Дані наповнюють шаблон вебсторінки `dresslists.html`.

Представлення відтворення вмісту переліку суконь реалізовано функцією `view_rentuser_dress_list`, яка має параметр `dresslist_id`, що задає номер переліку. Дані повертаються тільки тоді, коли власником цього переліку суконь є користувач, що авторизувався. Дані наповнюють шаблон вебсторінки `dresslist.html`.

Представлення створення запиту на прокат сукні реалізовано функцією `view_dress_rental_do`, що в основі має форму, створену на основі класу `DressrentalCreateForm`. Перед виконання збереження даних виконується перевірка того, що сукні в запиті на прокат належать одному й тому ж самому власнику або належать сервісу прокату. Якщо це не так, то збереження не

відбувається. Окрім того відбувається перевірка, що запитані дати не є занятими для вказаних екземплярів суконь. Зайнятими сукні можуть бути або для виконання робіт з ними, або в інших підтверджених на цей момент запитах на прокат.

Представлення надання інформації про сукню загалом реалізовано функцією `view_dress`, яка має параметр `dress_slug`, що задає текстовий ідентифікатор сукні. Дані в результаті включають не тільки дані самої сукні загалом, але і перелік максимум 20 екземплярів цієї сукні, які можуть бути взяті на прокат, перелік останніх відгуків про екземпляри цієї сукні, фотографії сукні, дані дизайнера, колекції, категорії сукні, склад, з якого виготовлено сукню.

Представлення надання інформації про сукню у варіанті, доступному для прокату, реалізовано функцією `view_dresssize`, яка має параметр `dresssize_id`, що задає номер екземпляра сукні, який відповідно має свій розмір та власника, `rentstart`, що задає необхідну дату початку прокату, `rentend`, що задає необхідну дату завершення прокату. Дані в результаті включають дані самого цього екземпляра, вартість прокату, перелік відгуків, дані сукні загалом, фотографії сукні, дані дизайнера, колекції, категорії сукні, склад, з якого виготовлено сукню.

Представлення відтворення надісланих повідомлень між власником сукні та користувачем-орендарем реалізовано функцією `view_messages_from_rentuser`, яка має параметр `rentuser_username`, який задає нік користувача, який є власником суконь, з яким мають бути відображені повідомлення для переписки з авторизованим користувачем. Тут же міститься форма класу `MessageForm`, але збереження даних з неї відбувається через обробку функцією `view_send_dress_message`, яка реалізує представлення надсилання повідомлення.

Представлення відтворення наявних діалогів користувача щодо прокату суконь реалізовано функцією `view_dialogs`, сам користувач при цьому

визначається даними авторизації. Дані наповнюють шаблон вебсторінки `dialogs.html`.

Представлення відтворення пропонованих для прокату варіантів сукні реалізовано функцією `view_dress_sizes`, яка має параметр `dress_slug`, що задає текстовий ідентифікатор сукні. Дані в результаті включають перелік всіх екземплярів цієї сукні, які можуть бути взяті на прокат.

Представлення пошуку за дизайнером суконь реалізовано функцією `view_designer_search`, отримуючи всі необхідні параметри через GET-запит, включаючи сторінку відтворення результатів (`page`), ім'я дизайнера (`desname`), текстовий ідентифікатор колекції суконь, для якої він створював сукні (`collection_slug`), номер бренду, для якого дизайнер створював сукні (`brand_id`). Дані наповнюють шаблон вебсторінки `designersearch.html`.

Представлення пошуку серед колекцій суконь реалізовано функцією `view_collection_search`, отримуючи всі необхідні параметри через GET-запит, включаючи сторінку відтворення результатів (`page`), назву колекції (`collectionname`), номер бренду, який представляє ця колекція (`brand_id`), тип сезону, для якого призначена колекція (`season_type`), інтервал сезонів за роками, в яких ця колекція була створена (`start_year` і `end_year`). Дані наповнюють шаблон вебсторінки `collectionsearch.html`.

Представлення пошуку за брендом суконь реалізовано функцією `view_brand_search`, отримуючи всі необхідні параметри через GET-запит, включаючи сторінку відтворення результатів (`page`), ім'я бренду (`brandname`), частина назви колекції, яку створив бренд (`collection_name`). Дані наповнюють шаблон вебсторінки `brandsearch.html`.

Представлення створення переліку суконь реалізовано функцією `view_dress_list_do`, що працює на основі класу `DresslistForm`. Представлення додавання або вилучення сукні до або з переліку `view_add_dress_to_list` має параметр `dresslist_id`, що задає номер переліку, та параметр `dresssize_id`, що задає номер екземпляру сукні. При цьому перелік має належати тому користувачу, який авторизувався, для збереження у базі даних. Додавання

відбувається, якщо цього екземпляру сукні немає у переліку досі, а вилучення відбувається, якщо цей екземпляр сукні вже є у переліку. Представлення вилучення переліку суконь реалізовано функцією `view_delete_dress_list`, що має параметр `dresslist_id`, що задає номер переліку, при цьому авторизований користувач повинен бути власником переліку суконь.

Представлення відтворення архіву прокату за сукнею у варіанті, доступному для прокату, реалізовано функцією `view_dressowner_rentals_archive`, яка має параметр `dresssize_id`, що задає номер екземпляру сукні в прокату. Дані надаються тільки користувачу, який є власником цього екземпляру сукні. Архівними є ті прокати, які були підтверджені, але дата завершення прокату суконь за ними менше поточної. Дані наповнюють шаблон вебсторінки `archrentals.html`.

Представлення відтворення запитів на прокат суконь користувача реалізовано функцією `view_dressowner_rentals_requests`. Дані надаються тільки користувачу, який має права виставляти сукні в прокат, і стосуються всіх його суконь. Дані наповнюють шаблон вебсторінки `reqrentals.html`.

Представлення відтворення запланованих прокатів за сукнею у варіанті, доступному для прокату, реалізовано функцією `view_dressowner_rentals_planned`, яка має параметр `dresssize_id`, що задає номер екземпляру сукні в прокату. Дані надаються тільки користувачу, який є власником цього екземпляру сукні. Запланованими є ті прокати, які були підтверджені, а дата завершення прокату суконь є пізнішою за поточну. Дані наповнюють шаблон вебсторінки `planrentals.html`.

Представлення відтворення запланованих прокатів за сукнями користувача реалізовано функцією `view_dressowner_rentals_all_planned`. Дані надаються за всіма екземплярами суконь, які є у авторизованого користувача. Запланованими є ті прокати, які були підтверджені, а дата завершення прокату суконь є пізнішою за поточну. Дані наповнюють шаблон вебсторінки `plrentals.html`.

Представлення відтворення переліку робіт за сукнею у прокаті реалізовано функцією `view_dresssize_works`, яка має параметр `dresssize_id`, що задає номер екземпляру сукні в прокату. Дані надаються тільки користувачу, який є власником цього екземпляру сукні. Дані наповнюють шаблон вебсторінки `dressworks.html`.

Представлення виставлення сукні на прокат реалізовано функцією `view_dresssize_add` на основі класу форми `DresssizeForm` та на основі функції `view_dresssize_edit`, що має параметр `dress_id` з номером екземпляру сукні, включаючи ту саму форму. Перший варіант функції необхідний для додавання, другий – для редагування.

Представлення внесення даних про колекцію суконь реалізовано функцією `view_collection_add` на основі класу форми `CollectionForm` та на основі функції `view_collection_edit`, що має параметр `collection_slug` з текстовим ідентифікатором колекції суконь, включаючи ту саму форму, при цьому доступ надається тільки користувачам з правами представництва сервісу прокату. Перший варіант функції необхідний для додавання, другий – для редагування.

Представлення внесення даних дизайнера суконь реалізовано функцією `view_designer_add` на основі класу форми `DesignerForm` та на основі функції `view_designer_edit`, що має параметр `designer_slug` з текстовим ідентифікатором дизайнера, включаючи ту саму форму, при цьому доступ надається тільки користувачам з правами представництва сервісу прокату. Перший варіант функції необхідний для додавання, другий – для редагування.

Представлення внесення даних про бренд суконь реалізовано функцією `view_brand_add` на основі класу форми `BrandForm` та на основі функції `view_brand_edit`, що має параметр `brand_id` з номером бренду, включаючи ту саму форму, при цьому доступ надається тільки користувачам з правами представництва сервісу прокату. Перший варіант функції необхідний для додавання, другий – для редагування.

Представлення внесення даних про сукню реалізовано функцією `view_dress_add` на основі класу форми `DressForm` та на основі функції `view_dress_edit`, що має параметр `dress_slug` з текстовим ідентифікатором сукні, включаючи ту саму форму, при цьому доступ надається тільки користувачам з правами представництва сервісу прокату. Перший варіант функції необхідний для додавання, другий – для редагування.

Представлення пошуку за користувачами, які є власниками суконь у прокаті, отримуючи всі необхідні параметри через GET-запит, включаючи сторінку відтворення результатів (`page`), нік користувача (`usnick`), ім'я користувача (`usname`). Дані наповнюють шаблон вебсторінки `gusersearch.html`.

Також реалізовано інші представлення у вигляді функцій, таким чином виконавши всі завдання щодо надання необхідних сценаріїв роботи користувачів, використовуючи аналогічні принципи до описаних вище.

`Designer` є реалізацією класу моделі дизайнерів.

Поле `slug` є реалізацією унікального текстового ідентифікатору, утвореного за іменем дизайнера суконь або його псевдонімом для використання в адресі при зверненнях до вебсторінок, на основі класу `SlugField`.

Поле `pseudo` є реалізацією повного імені дизайнера суконь або його псевдоніму на основі класу `CharField`.

Поле `presentation` є реалізацією опису діяльності, біографії дизайнера суконь на основі класу `TextField`.

Поле `photo` є реалізацією фотографії, де представлений дизайнер суконь, на основі класу `ImageField`.

`Dresscategory` є реалізацією класу моделі категорій суконь.

Поле `slug` є реалізацією унікального текстового ідентифікатора, утвореного за назвою категорії суконь для використання в адресі при зверненнях до вебсторінок, на основі класу `SlugField`.

Поле `categoryname` є реалізацією повної назви категорії суконь на основі класу `CharField`.

Collection є реалізацією класу моделі колекції суконь.

Поле appcaption є реалізацією назви колекції суконь, якою вона відображається в короткій формі у вебзастосунку, тобто в тих позиціях у інтерфейсі, де необхідна коротка форма на основі класу, на основі класу CharField.

Поле fulltitle є реалізацією повної назви колекції суконь на основі класу CharField.

Поле slug є реалізацією унікального текстового ідентифікатора, утвореного за назвою колекції суконь для використання в адресі при зверненнях до вебсторінок, на основі класу SlugField.

Поле brand є реалізацією бренда, який виготовляв цю колекцію суконь, забезпеченою через зв'язок з моделлю Brand у даному полі як зовнішньому ключі на основі класу ForeignKey.

Поле presentation є реалізацією текстового опису особливостей цієї колекції суконь на основі класу TextField.

Поле photo є реалізацією фотографії для ілюстрації колекції суконь на основі класу ImageField.

Поле startseason є реалізацією року, з якого почалось виготовлення цієї колекції суконь, на основі класу SmallIntegerField.

Поле endseason є реалізацією року, коли закінчилось виготовлення цієї колекції суконь, на основі класу SmallIntegerField.

Поле seasontype є реалізацією типу сезону, якому відповідає колекція (якщо це саме колекція під один конкретний сезон року), на основі класу SmallIntegerField.

Brand є реалізацією класу моделі брендів суконь.

Поле title є реалізацією назви бренду суконь на основі класу CharField.

Поле website є реалізацією вебсайту бренду суконь на основі класу URLField.

Dress є реалізацією класу моделі суконь.

Поле `dresscategory` є реалізацією категорії, до якої належить ця сукня, забезпеченою через зв'язок з моделлю `Dresscategory` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `collection` є реалізацією колекції, до якої належить ця сукня, забезпеченою через зв'язок з моделлю `Collection` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `designer` є реалізацією дизайнера, який створив цю сукню, забезпеченою через зв'язок з моделлю `Designer` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `slug` є реалізацією унікального текстового ідентифікатора, утвореного за назвою сукні для використання в адресі при зверненнях до вебсторінок, на основі класу `SlugField`.

Поле `appcaption` є реалізацією назви, якою ілюструється ця сукня для відображення у вебзастосунку, на основі класу `CharField`.

Поле `colour` є реалізацією кольору сукні на основі класу `CharField`.

Поле `lengthtype` є реалізацією типу довжини сукні на основі класу `CharField`, забезпечуючи підтримку переліку кортежів для вибору одного з фіксованих значень.

Поле `sleevetype` є реалізацією типу довжини рукава сукні на основі класу `CharField`, забезпечуючи підтримку переліку кортежів для вибору одного з фіксованих значень.

Поле `description` є реалізацією текстового опису, що ілюструє сукню, на основі класу `TextField`.

Поле `styles` є реалізацією стилів, яким відповідає ця сукня, забезпеченою через зв'язок з моделлю `Style` у даному полі як зовнішньому ключі на основі класу `ManyToManyField`.

`Dressphoto` є реалізацією класу моделі фотографій, які ілюструють сукню загалом.

Поле `dress` є реалізацією сукні, яка показана на фотографії, забезпеченою через зв'язок з моделлю `Dress` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `photo` є реалізацією фотографії, що ілюструє цю сукню, на основі класу `ImageField`.

`Dresscomposition` є реалізацією класу моделі складу сукні (матеріалів, з яких сукня виготовлена).

Поле `dress` є реалізацією сукні, до складу якої входить цей матеріал, забезпеченою через зв'язок з моделлю `Dress` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `material` є реалізацією матеріалу, з якого виготовлена ця сукня, забезпеченою через зв'язок з моделлю `Material` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `percent` є реалізацією відсотку вмісту матеріалу в складі цієї сукні на основі класу `DecimalField`.

Поле `dresspart` є реалізацією частини, до якої входить цей матеріал, на основі класу `CharField`, забезпечуючи підтримку переліку кортежів для вибору одного з фіксованих значень.

`Material` є реалізацією класу моделі матеріалів, з яких можуть виготовлятися сукні, представлені у вебзастосунку, додаючи до стандартної моделі тільки поле `name`, є реалізацією назви матеріалу на основі класу `CharField`.

`Dresssize` є реалізацією класу моделі конкретної сукні (з вказаними розмірами та власництвом).

Поле `dress` є реалізацією загальної сукні (без деталізації розміру та права власності), яку представляє цей екземпляр з розміром та власництвом, забезпеченою через зв'язок з моделлю `Dress` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `dressowner` є реалізацією власника цього екземпляра сукні, забезпеченою через зв'язок з моделлю `Rentuser` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `size` є реалізацією конкретного розміру, який має ця сукня, на основі класу `CharField`, забезпечуючи підтримку переліку кортежів для вибору одного з фіксованих значень.

Поле `fit` є реалізацією варіанта посадки, якому відповідає ця сукня, на основі класу `CharField`, забезпечуючи підтримку переліку кортежів для вибору одного з фіксованих значень.

Поле `rank` є реалізацією середнього балу, у який цю конкретно сукню у такому розмірі та у цього власника оцінено орендарями, на основі класу `DecimalField`.

Поле `quantity` є реалізацією кількості екземплярів сукні в цьому розмірі у цього власника на основі класу `PositiveSmallIntegerField`.

Поле `active` є реалізацією визначення активного оголошення або вже архівного на основі класу `BooleanField`.

Поле `address` є реалізацією міста або додаткової адреси, де ця сукня доступна, на основі класу `CharField`.

`Dresssizephoto` є реалізацією класу моделі фотографій конкретних екземплярів суконь.

Поле `dresssize` є реалізацією сукні в конкретному розмірі та у конкретного власника, яка показана на фотографії, забезпеченою через зв'язок з моделлю `Dresssize` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `photo` є реалізацією фотографії, що ілюструє цей розмір сукні у даного власника, на основі класу `ImageField`.

`Dressprice` є реалізацією класу моделі вартості прокату конкретної сукні (з вказаними розмірами).

Поле `dresssize` є реалізацією сукні в конкретному розмірі та у конкретного власника, для якої встановлено цю пропозицію прокату за даною

вартістю, забезпеченою через зв'язок з моделлю Dresssize у даному полі як зовнішньому ключі на основі класу ForeignKey.

Поле typical є реалізацією визначення того, що дана пропозиція є типовою, тобто використовується для всіх розмірів цієї сукні у того ж власника, чи власник визначає окремо вартість прокату для кожного варіанта, при цьому реалізована на основі класу BooleanField.

Поле mindays є реалізацією мінімальної кількості днів прокату сукні, за якої починає діяти цей варіант вартості прокату, на основі класу PositiveSmallIntegerField.

Поле price є реалізацією вартості дня прокату цієї сукні в заданому розмірі та у заданого власника, яка в такому випадку діє, на основі класу DecimalField.

Поле startperiod є реалізацією дати, з якої цей варіант вартості прокату сукні в даному розмірі у заданого власника починає діяти, на основі класу DateField.

Поле endperiod є реалізацією дати, до якої цей варіант вартості прокату сукні в даному розмірі у заданого власника діє, на основі класу DateField.

Dresswork є реалізацією класу моделі виконуваних робіт з екземпляром сукні в конкретному розмірі та у конкретного власника.

Поле dresssize є реалізацією сукні в конкретному розмірі та у конкретного власника, з якою виконуються ці роботи, забезпеченою через зв'язок з моделлю Dresssize у даному полі як зовнішньому ключі на основі класу ForeignKey.

Поле startwork є реалізацією дати початку робіт, які виконуються з сукнею в конкретному розмірі та у конкретного власника та через які ця сукня не буде доступна, на основі класу DateField.

Поле endwork є реалізацією дати закінчення робіт, які виконуються з сукнею в конкретному розмірі та у конкретного власника та через які ця сукня не буде доступна, на основі класу DateField.

Поле `worktype` є реалізацією типу робіт, які виконуються з цією сукнею в конкретному розмірі та у конкретного власника, на основі класу `CharField`, забезпечуючи підтримку переліку кортежів для вибору одного з фіксованих значень.

`Dressrental` є реалізацією класу моделі замовлення прокату суконь.

Поле `rentuser` є реалізацією користувача, який замовив цей прокат суконь, забезпеченою через зв'язок з моделлю `Rentuser` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `createdon` є реалізацією дати і часу створення замовленого прокату суконь на основі класу `DateTimeField`.

Поле `state` є реалізацією поточного стану цього замовленого прокату суконь на основі класу `CharField`, забезпечуючи підтримку переліку кортежів для вибору одного з фіксованих значень.

Поле `deliveryon` є реалізацією дати і часу доставки (запланованої або виконаної) замовленого прокату суконь на основі класу `DateTimeField`.

Поле `deliveryto` є реалізацією дати і часу, до яких має тривати замовлений прокат суконь на основі класу `DateTimeField`.

Поле `price` є реалізацією вартості замовленого прокату суконь на основі класу `DecimalField`.

Поле `rentuserreview` є реалізацією відгуку власника суконь у замовленні про користувача, який замовив прокат на основі класу `TextField`.

Поле `rentuserreviewon` є реалізацією дати і часу створення відгуку власника суконь у замовленні про користувача, який замовив прокат, на основі класу `DateTimeField`.

Поле `rentuserreviewrank` є реалізацією балу, виставленого у відгуку власника суконь у замовленні про користувача, який замовив прокат, на основі класу `PositiveSmallIntegerField`.

`Dressrentalcontent` є реалізацією класу моделі конкретного вмісту замовлення прокату суконь.

Поле `dressrental` є реалізацією замовленого прокату суконь, вміст якого розкривається цією позицією, забезпеченою через зв'язок з моделлю `Dressrental` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `dresssize` є реалізацією сукні в конкретному розмірі та у конкретного власника, що входить до цього замовленого прокату суконь, забезпеченою через зв'язок з моделлю `Dresssize` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `dressreview` є реалізацією відгуку користувача, який замовив прокат, про сукню на основі класу `TextField`.

Поле `dressreviewon` є реалізацією дати і часу створення відгуку користувача, який замовив прокат, про сукню на основі класу `DateTimeField`.

Поле `dressreviewrank` є реалізацією балу, виставленого у відгуку користувача, який замовив прокат, про сукню на основі класу `PositiveSmallIntegerField`.

`Rentuser` є реалізацією класу моделі користувача вебзастосунку прокату суконь.

Поле `userinapp` є реалізацією облікового запису, який використовує цей користувач вебзастосунку прокату суконь, на основі класу `OneToOneField` зі стандартною Django-моделлю `User`.

Поле `fullname` є реалізацією повного імені користувача вебзастосунку прокату суконь на основі класу `CharField`.

Поле `photo` є реалізацією фотографії, що ілюструє цього користувача вебзастосунку прокату суконь, на основі класу `ImageField`.

Поле `description` є реалізацією опису користувача вебзастосунку прокату суконь для самопрезентації на основі класу `TextField`.

Поле `canbeowner` є реалізацією позначки, що визначає дозвіл користувачу вебзастосунку прокату суконь виставляти для прокату власні сукні, на основі класу `BooleanField`.

Поле `rank` є реалізацією середнього балу, у який роботу з цим конкретно користувачем вебзастосунку прокату суконь оцінили власники суконь, на основі класу `DecimalField`.

`Dresslist` є реалізацією класу моделі переліків суконь, створених користувачами вебзастосунку прокату суконь.

Поле `rentuser` є реалізацією користувача вебзастосунку прокату суконь, який створив цей перелік суконь, забезпеченою через зв'язок з моделлю `Rentuser` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `listcaption` є реалізацією назви переліку суконь на основі класу `CharField`.

Поле `slug` є реалізацією унікального текстового ідентифікатору, утвореного за назвою переліку суконь для використання в адресі при зверненнях до вебсторінок, на основі класу `SlugField`.

Поле `dresssizes` є реалізацією вмісту переліку суконь, що включає набір конкретних розмірів суконь у власності орендодавців, забезпеченою через зв'язок з моделлю `Dresssize` у даному полі як зовнішньому ключі на основі класу `ManyToManyField`.

`Message` є реалізацією класу моделі повідомлень до власників суконь від орендарів.

Поле `rentuser` є реалізацією користувача вебзастосунку прокату суконь, який створив це повідомлення, забезпеченою через зв'язок з моделлю `Rentuser` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `dresssize` є реалізацією сукні в конкретному розмірі та у конкретного власника, стосовно якої адресовано повідомлення, забезпеченою через зв'язок з моделлю `Dresssize` у даному полі як зовнішньому ключі на основі класу `ForeignKey`.

Поле `createdon` є реалізацією дати і часу, коли це повідомлення до власника суконь від орендаря було створено, на основі класу `DateTimeField`.

Поле `messagedtext` є реалізацією тексту повідомлення до власника суконь від орендаря на основі класу `TextField`.

Style є реалізацією класу моделі можливих стилів суконь, додаючи до стандартної реалізації моделі поле name, що є реалізацією назви стиля сукні на основі класу CharField.

4.3 Висновки за розділом 4

Розділ 4 містить деталізований опис розроблених програмних одиниць у складі вебзастосунку Django, що включають різні сторони реалізації функціональних можливостей на основі забезпечених у підсумку моделей, представлень, пов'язаних з ними шаблонів вебсторінок.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Основне призначення програми

Програма прокату суконь призначена для виставлення на прокат власних суконь користувачами та надання варіантів прокату самим сервісом прокату суконь, керування взаємодією з користувачами стосовно прокату суконь, пошуку тих варіантів прокату суконь, які задовольняють користувачів.

5.2 Виконання програми

Для роботи достатньо звернутися за адресою розміщення вебзастосунку на сервері та почати працювати, використовуючи логіку користувача з правами орендаря суконь.

Використовуючи меню, користувач з правами орендаря суконь може в різних варіантах отримати доступ до переліків суконь, вибираючи одразу з меню певну категорію або стиль, виконуючи пошук за дизайнерами, за брендами, за колекціями, а тоді вже переходячи до сторінки потрібного дизайнера, бренда, колекції, а з неї на сторінку з відображенням відповідних суконь. Також можна виконати пошук сукні, визначаючи зокрема і склад матеріалів або те, які матеріали не повинні міститись, місто, де сукня має бути. З загальної сторінки сукні можна перейти до потрібного конкретного оголошення про прокат сукні з розміром та правом власництва. Відповідна сторінка оголошення (рис. 5.1) містить інформацію про сукню загалом, а також дані, які стосуються конкретної власності на відповідний екземпляр сукні. Ці дані включають власника, оцінку, виставлену цьому екземпляру користувачами, які взяли її в оренду (при натисканні на кнопку з оцінкою відбувається переміщення по сторінці вниз до переліку відгуків), фотографії екземпляра сукні, загальні дані сукні (опис, назва, категорія, бренд тощо, а також нижче розташований склад матеріалів, з яких виготовлено сукню). На

сторінці також міститься форма, де можна задати потрібні дати прокату, після чого визначити, чи ці дати вільні, а також отримати інформацію про вартість прокату. На сторінці також вказується розмір сукні.

Ren
DRESS

Категорії

Стилі

Дизайнери

Бренди

Колекції

Пошук

Прокат

Перепіки

Повідомлення

Профіль

Сукня в квітах

Власник



Віктор Микитченко

Оцінка: 4.8



Вільна сукня, вкрита величезними квітами, які віддалено створюють оптичну ілюзію, яка надзвичайно гарно виглядає та привертає увагу в танцях.

Хоча звісно сукня пасує не тільки для танців. Може бути прекрасним варіантом для урочистостей. Має достатньо універсальну довжину.



Характеристика

Категорія	Сукня А-силуету
Бренд	Indigo Dresses
Колекція	Summer Madness
Дизайнер	Августо Долоетто
Довжина сукні	Середня
Довжина рукава	Безрукавна
Колір	Оливковий
Посадка	Висока талія
Розмір	М

Параметри прокату

Початок прокату

26.11.2024

Завершення прокату

27.11.2024

Вартість: 800 грн.

До замовлення

Рисунок 5.1 – Сторінка оголошення про прокат сукні з екземпляром, виставленим користувачем

Тільки для орендаря доступний розділ меню з переліками, звідки можна переглянути створені власні переліки суконь, перейти до потрібного, вилучити непотрібний. Розділ прокату доступний для всіх типів користувачів, при цьому можна обрати потрібний варіант, але варіанти відрізняються для різних типів користувачів. Тобто для орендаря вони стосуються власних запитів на прокат суконь, склад поточного замовлення прокату суконь, який звідти можна підтвердити та створити запит, для орендаря вже включають адресовані йому запити.

Розділ меню з повідомленнями дозволяє отримати доступ до переліку діалогів. Це діалоги між власниками суконь та користувачами, які хочуть їх орендувати. Для створення діалогу потрібно зі сторінки оголошення про прокат сукні перейти на сторінку користувача-власника сукні, а там міститься окрім інформації про нього кнопка, за допомогою якої можна перейти до діалогу з ним, отримуючи перелік повідомлень та надсилаючи нове, якщо потрібно.

Уся структурна інформація про сукню (каталог, дизайнер, бренд тощо) дозволяє натиснути на відповідне значення та відобразити або його сторінку, або сукні за ним в залежності від того, щ саме функціонально передбачено базове.

Для користувача з правами виставляти сукні на прокат доступний пункт меню «Сукні», який відображає власні оголошення (рис. 5.2), дозволяє додати нове, керувати існуючими, включаючи перехід до самого оголошення, перехід до редагування оголошення, скасування оголошення, перехід до архіву замовлень за сукнею, до запланованих замовлень за сукнею.

Користувач з правами виставляти сукні на прокат може додавати оголошення за сукнею, яка відсутня у базі даних, додаючи конкретний екземпляр одразу. Тобто цей процес включає два кроки: визначення даних самої сукні, якщо вона в загальному вигляді відсутня у програмі, та визначення параметрів конкретного екземпляру. А вже зі сторінки оголошення сам

власник може додати дані про виконувані роботи, про варіанти вартості прокату.

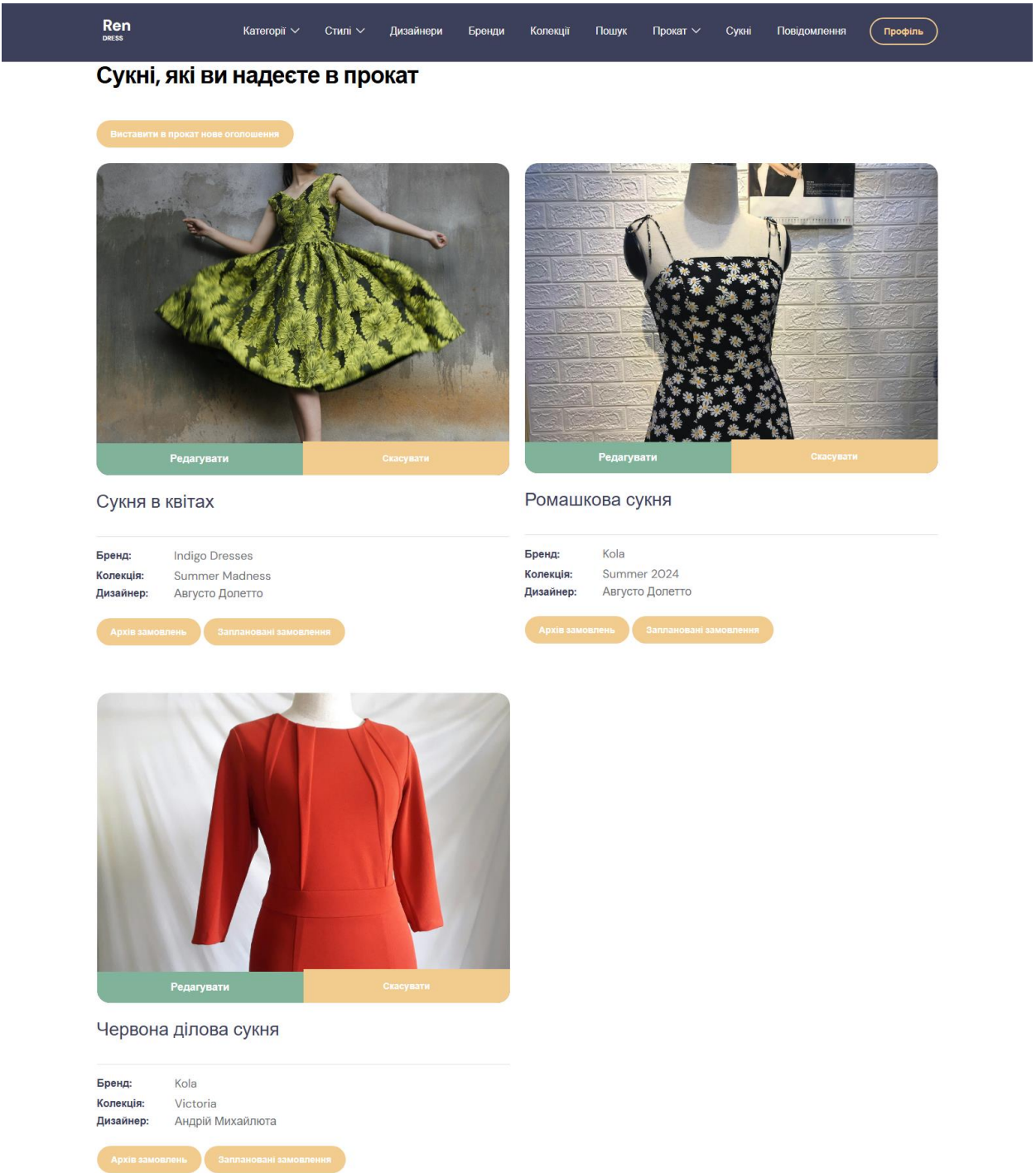


Рисунок 5.2 – Сторінка власних оголошень користувача про прокат суконь

У свою чергу користувачі з правами представництва сервісу з прокату суконь спочатку створюють запис про сукню загалом (рис. 5.3), а вже далі можуть вносити конкретні екземпляри, якщо це потрібно. З цієї ж сторінки

можна спочатку, ввівши точно колекцію (можна вводити і таким чином виконувати пошук), що дозволяє задати зокрема і бренд, колір та довжину сукні, натиснути на кнопку «Оцінити», що дозволить спрогнозувати вартість прокату такої сукні на основі навченої моделі за запропонованим методом. Ці результати також доступні під час пошуку суконь звичайному користувачу зі сторінки пошуку суконь.

Ren DRESS Категорії ▾ Стилі ▾ Дизайнери ▾ Бренди ▾ Колекції ▾ Пошук Користувачі Прокат ▾ Сукні ▾ Повідомлення Профіль

Додавання нової сукні

Назва

Категорія

Весільна сукня

Колекція

Adello

Дизайнер

введіть значення

Колір

Червоний

Довжина

Середня

Довжина рукава

введіть значення

Склад

Додати матеріал

Бавовна

Відсоток

оберіть частину сукні

Додати матеріал

Опис

Оцінити

Вартість: 1650 грн.

Додати фотографію

Зберегти

Рисунок 5.3 – Сторінка внесення даних про сукню

Розділи меню користувача з правами представництва сервісу з прокату суконь додатково містять у випадających переліках варіанти для додавання, наприклад, колекцій, брендів, дизайнерів. Також доступний розділ з користувачами, звідки можна виконувати пошук користувачів та надавати їм право виставляти свої сукні на прокат.

5.3 Експериментальне дослідження прогнозування вартості прокату суконь

Експериментальне дослідження прогнозування вартості прокату сукні виконано на основі вибірки даних [6], яка є вільно розповсюджуваною та представляє результати роботи сервісу з прокату суконь від власників до орендарів, який функціонував у Великобританії. Вибірка містить 30 тисяч екземплярів даних.

На основі цієї вибірки даних було виконано створення трьох типів моделей прогнозування:

- на основі дерев ухвалення рішень, що є реалізацією запропонованого методу;
- на основі методу опорних векторів, що також може використовуватися для розв’язання задачі прогнозування;
- на основі лінійної регресії.

Для кожної з цих моделей було проаналізовано наступні варіанти вхідних ознак:

- категоріальне значення бренду сукні, категоріальне значення кольору сукні та всі категорії, які було виділено з переліку категорій;
- 3 найінформативніші ознаки, які були виділені з множини, що включала категоріальне значення бренду сукні, категоріальне значення кольору сукні та всі категорії, які було виділено з переліку категорій;

– 5 найінформативніших ознак, які були виділені з множини, що включала категоріальне значення бренду сукні, категоріальне значення кольору сукні та всі категорії, які було виділено з переліку категорій.

Оцінювання результатів прогнозування вартості прокату суконь (табл. 5.1) виконувалось на основі двох показників:

- показник САП;
- показник середньої квадратичної похибки (СКП).

Таблиця 5.1 – Результати прогнозування вартості прокату суконь

Метод	САП	СКП
Дерево ухвалення рішень (бренд, колір, всі категорії)	0,0014377564	7,443249334E-6
Метод опорних векторів (бренд, колір, всі категорії)	0,0236879684	0,0005693981
Лінійна регресія (бренд, колір, всі категорії)	0,0019476795	9,6877848478E-6
Дерево ухвалення рішень (3 найінформативніші ознаки)	0,0013618961	6,6147218613E-6
Метод опорних векторів (3 найінформативніші ознаки)	0,0236879686	0,0005693981
Лінійна регресія (3 найінформативніші ознаки)	0,0019502269	9,7534916269E-6
Дерево ухвалення рішень (5 найінформативніших ознак)	0,001376565	6,7204571135E-6
Метод опорних векторів (5 найінформативніших ознак)	0,0236879683	0,0005693981
Лінійна регресія (5 найінформативніших ознак)	0,0019498398	9,7492971896E-6

Вхідні ознаки результуючої моделі, які було запропоновано використовувати за методом, представленим у роботі, включають:

- категоріальне представлення бренду сукні;
- категоріальне представлення кольору сукні;
- позначка про те, чи є ця сукня міні-сукнею.

Цей результат відповідає дереву ухвалення рішень з 3 найінформативнішими ознаками, що має найменше значення САП та СКП. Порівняно з лінійної регресією рівень САП виявився за такого ж набору ознак на 30,17 % меншим, а значення СКП – на 32,18 %. У свою чергу значення САП для методу опорних векторів перевищило цей результат у 16,4 разів, а СКП – у 85 разів.

Результати методу опорних векторів та лінійної регресії за різного набору вхідних ознак були приблизно близькими. Для дерева ухвалення рішень відкидання зайвих вхідних ознак дозволило покращити результати прогнозування.

Окремо було також виконано спробу представити назву сукні у числовому вигляді як набір з 20 ознак на основі моделі word2vec. Ці значення було додано як додаткові вхідні ознаки (табл. 5.2).

Однак, отримані результати характеризувались збільшенням показників і САП, і СКП. Таким чином, не варто рекомендувати застосовувати таку додаткову ознаку.

Тож застосування запропонованого методу визначення ціноутворення під час прокату суконь на основі 3 вхідних ознак та дерева ухвалення рішень дозволило досягти найкращих результатів, які слід вважати придатними для впровадження у програмі. Впровадження цієї моделі і було реалізовано при програмній реалізації функції додавання нової сукні, редагування існуючої та при виконанні пошуку суконь для прокату в якості інформаційної підказки користувачам.

Таблиця 5.2 – Результати прогнозування вартості прокату суконь з врахуванням числового представлення назви

Метод	САП	СКП
Дерево ухвалення рішень (бренд, колір, всі категорії, назва)	0,0018121931	1,1339784543E-5
Метод опорних векторів (бренд, колір, всі категорії, назва)	0,0236879684	0,0005693981
Лінійна регресія (бренд, колір, всі категорії, назва)	0,0019263405	9,5777029703E-6
Дерево ухвалення рішень (3 найінформативніші ознаки, назва)	0,001766945	1,0694461521E-5
Метод опорних векторів (3 найінформативніші ознаки, назва)	0,0236879684	0,0005693981
Лінійна регресія (3 найінформативніші ознаки, назва)	0,00192824	9,6289118996E-6

5.4 Висновки за розділом 5

Розділ 5 містить опис роботи з вебзастосунком за логікою організації такої роботи та основними сценаріями, експериментальне дослідження прогнозування вартості прокату суконь, включаючи опис такого дослідження та отримані результати.

Отримані в підсумку результати вказують, що застосування запропонованого методу визначення ціноутворення під час прокату суконь

дозволило відібрати найбільш інформативні вхідні ознаки моделі та застосувати дерева ухвалення рішень, що призвело до досягнення найкращих результатів, які слід вважати придатними для впровадження у вебзастосунок прокату суконь.

ВИСНОВКИ

Мету роботи, що полягала в розробці програмного забезпечення прокату суконь для підвищення зручності прийняття рішень щодо визначення вартості прокату, виконано. Усі завдання, необхідні для цього, включаючи аналіз програмних застосунків прокату суконь, розроблення методу визначення ціноутворення під час прокату суконь, проєктування вебзастосунку прокату суконь, реалізацію вебзастосунку прокату суконь, виконання експериментального дослідження прогнозування вартості прокату суконь, виконано.

Програма, яка була розроблена для підтримки процесу прокату суконь, призначена для виставлення на прокат власних суконь користувачами та надання варіантів прокату самим сервісом прокату суконь, керування взаємодією з користувачами стосовно прокату суконь, пошуку тих варіантів прокату суконь, які задовольняють користувачів. Під час створення нового оголошення про прокат сукні автоматично відбувається формування пропозиції щодо встановлення вартості прокату на основі навченої моделі. Реалізацію програми забезпечено мовою Python з фреймворком Django.

Робота має наукову новизну, представлену методом визначення ціноутворення під час прокату суконь, який визначає процес формування вхідних ознак, виділення найбільш інформативних з них, створення моделі на основі дерева ухвалення рішень, що дозволяє прогнозувати вартість прокату сукні. Результати експериментального дослідження вказують, що застосування запропонованого методу визначення ціноутворення під час прокату суконь дозволило відібрати найбільш інформативні вхідні ознаки моделі та застосувати дерева ухвалення рішень, що призвело до досягнення найкращих результатів порівняно з лінійною регресією та методом опорних векторів, тому їх слід вважати придатними для впровадження у вебзастосунку прокату суконь, що і було виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Rim, H. Fashion Rental Market Makes A Comeback – As Gen Z And Millennials Revive Brands Like Nuuly, Rent The Runway [Electronic resource] / H. Rim. – Access mode : <https://www.forbes.com/sites/hyunsoorim/2024/05/28/fashion-rental-market-makes-a-comeback-as-gen-z-and-millennials-revive-brands-like-nuuly-rent-the-runway/>.
2. Hirestreet. Rent Dresses For All Occasions [Electronic resource]. – Access mode : <https://www.hirestreetuk.com/>.
3. Rent, Resell designer fashion dresses bags. By Rotation UK [Electronic resource]. – Access mode : <https://byrotation.com/>.
4. Rent Designer Dresses, Bags and Clothes UK. HURR [Electronic resource]. – Access mode : <https://www.hurrcollective.com/>.
5. Drexcode – Top designer dress rental online – Rent Fashion [Electronic resource]. – Access mode : <https://drexcode.com/>.
6. Peer to peer dress rental prices [Electronic resource]. – Access mode : <https://www.kaggle.com/datasets/kristianbarnsley/peer-to-peer-dress-rental-prices>.
7. Classification And Regression Trees for Machine Learning [Electronic resource]. – Access mode : <https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/>.
8. Python Web Development With Django – GeeksForGeeks Learning [Electronic resource]. – Access mode : <https://www.geeksforgeeks.org/python-web-development-django/>.
9. Python Falcon Tutorial [Electronic resource]. – Access mode : https://www.tutorialspoint.com/python_falcon/index.htm.
10. Free Template 587 Tiya Golf Club [Electronic resource]. – Access mode : <https://templatemo.com/tm-587-tiya-golf-club>,

ДОДАТОК А
Технічне завдання

Вступ

Використання прокату суконь може дозволити вирішити проблеми, які виникають перед людьми в різних ситуаціях, при цьому знижуючи витрати на вирішення таких проблем. Визначення вартості прокату при цьому є наступним ключовим питанням після визначення готовності виставити сукню для прокату. Від рішення, яке приймається на цьому етапі, залежить те, чи вдасться привернути увагу клієнтів, а фактично в підсумку, чи вдасться заробити кошти на організації прокату. Тому для реалізації даного процесу важливою є наявність програмного механізму, який би створив основу для прийняття відповідного рішення. Тому при реалізації програмного забезпечення прокату суконь важливо передбачити такий механізм. Це відповідно має з одного боку підтримувати користувачів, які виставляють сукні на прокат, у прийнятті рішень, а з іншого визначити конкурентну перевагу створеного програмного забезпечення. Зважаючи на наявність ряду вебзстосунків для різних прокатів одягу, в тому числі і суконь, наявність конкурентної переваги є важливою.

Відповідно у даній роботі необхідно реалізувати програмне забезпечення прокату суконь, що має забезпечуватися як від сервісу прокату суконь, так і від окремих власників, при цьому надаючи засіб прогнозування вартості прокату суконь на основі параметрів цих суконь.

A.1 Підстави для розробки

Наказом № 398 від 7 жовтня 2024 року за Національним університетом «Запорізька політехніка» визначена тема дипломної кваліфікаційної роботи магістра «Дослідження та програмна реалізація методів забезпечення ціноутворення під час прокату суконь», що обумовило отримання завдання на роботу.

A.2 Призначення розробки

Програма прокату суконь призначена для виставлення на прокат власних суконь користувачами та надання варіантів прокату самим сервісом прокату суконь, керування взаємодією з користувачами стосовно прокату суконь, пошуку тих варіантів прокату суконь, які задовольняють користувачів.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

У роботі шляхом реалізації програмного забезпечення необхідно надати наступні функціональні можливості:

- надання інформації про сукню загалом;
- надання інформації про сукню у варіанті, доступному для прокату;
- пошук за дизайнером суконь;
- надання інформації про дизайнера суконь;
- відтворення пропонованих для прокату суконь за категорією;
- надання інформації про бренд суконь;
- внесення даних дизайнера суконь;
- надання інформації про колекцію суконь;
- пошук серед колекцій суконь;
- пошук сукні для прокату;
- відтворення пропонованих для прокату суконь за колекцією;
- відтворення пропонованих для прокату суконь за брендом;
- відтворення пропонованих для прокату суконь за дизайнером;
- внесення даних про колекцію суконь;
- внесення даних про бренд суконь;
- пошук за брендом суконь;
- відтворення пропонованих для прокату варіантів сукні;
- відтворення пропонованих для прокату суконь за власником;

- відтворення пропонованих для прокату суконь за стилем;
- надання інформації про власника суконь;
- внесення даних про сукню;
- виставлення сукні на прокат;
- внесення даних про вартість прокату сукні;
- відтворення запитів на прокат суконь користувача;
- відтворення запитів на прокат суконь, які створив користувач;
- відтворення архіву прокату за сукнею у варіанті, доступному для прокату;
- відтворення запланованих прокатів за сукнею у варіанті, доступному для прокату;
- відтворення запланованих прокатів за сукнями користувача;
- відтворення архіву прокату суконь користувачем;
- відтворення запланованих прокатів суконь користувачем;
- створення запиту на прокат сукні;
- надання інформації про запит на прокат або прокат сукні;
- авторизація з можливістю реєстрації користувачів;
- пошук за користувачами, які є власниками суконь у прокаті;
- створення переліку суконь;
- відтворення переліків суконь користувача;
- додавання або вилучення сукні з переліку;
- вилучення переліку суконь;
- відтворення вмісту переліку суконь;
- підтвердження або скасування запиту на прокат сукні;
- скасування прокату сукні власником;
- надсилання повідомлення між власником сукні та користувачем-орендарем;
- відтворення надісланих повідомлень між власником сукні та користувачем-орендарем;
- відтворення переліку робіт за сукнею у прокаті;

- внесення даних про роботи над сукнею у прокаті;
- скасування або активація оголошення про прокат сукні;
- відтворення наявних діалогів користувача щодо прокату суконь;
- надання користувачу права виставляти сукні в прокат або скасування такого права.

A.3.2 Умови експлуатації

Забезпечення експлуатації програмного забезпечення прокату суконь вимагає наявності адміністратора системи або делегування відповідних функцій іншому працівнику. Адміністратор має налагодити роботу сервера та розгорнути на ньому вебзастосунок, підтримувати роботу з налаштуваннями у подальшому, а також вносити за необхідності базові дані каталогу в базу даних через адміністративну панель, зокрема включаючи категорії суконь, стилі, матеріали.

A.3.3 Вимоги до складу та параметрів технічних засобів

Враховуючи, що програмне забезпечення прокату суконь має бути створено у вигляді вебзастосунку, то вимоги до сервера є основними вимогами до технічних засобів, а відповідно мінімально включають двоядерний процесор, 4 Гб оперативної пам'яті, 100 Гб вільного обсягу жорсткого диску, щоби зберігати дані, які всі окрім файлів кодів вебзастосунку мають бути представлені базою даних, де велику частину обсягу потенційно мають займати фотографії, зокрема фотографії суконь.

A.4 Порядок контролю та приймання

Завдання на роботу визначає календарний план виконання роботи, включаючи періоди виконання, очікувані результати та консультантів, які

забезпечують контроль виконання відповідних етапів. У підсумку відбувається захист дипломної кваліфікаційної роботи у відведений для цього випусковою кафедрою період, де представляються остаточні результати з підтвердженням проходження всіх попередніх етапів.

ДОДАТОК Б
Текст програми

```

from .models import Designer
from .models import Dresscategory
from .models import Collection
from .models import Brand
from .models import Dress
from .models import Dressphoto
from .models import Dresscomposition
from .models import Material
from .models import Dresssize
from .models import Dresssizephoto
from .models import Dressprice
from .models import Dresswork
from .models import Dressrental
from .models import Dressrentalcontent
from .models import Rentuser
from .models import Dresslist
from .models import Message
from .models import Style
from django.shortcuts import render
from django.core.paginator import Paginator
from datetime import datetime

def view_rentuser_dress_rentals_archive(request):
    dressdata = {}
    if request.user.is_authenticated:
        rent_user = Rentuser.objects.filter(userinapp =
request.user)
        dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
        dressdata["lists"] = dress_lists
        dress_categories =
Dresscategory.objects.all().order_by("categoryname")
        dressdata["categories"] = dress_categories
        dress_styles = Style.objects.all().order_by("name")
        dressdata["styles"] = dress_styles
        if not request.user.is_authenticated:
            return render(request,
"rendress/userarchrentals.html", dressdata)
        rent_user = Rentuser.objects.filter(userinapp =
request.user)
        time_for_archive = datetime.now()
        rent_user_dressrentals =
Dressrental.objects.filter(rentuser = rent_user, state = 1,
deliveryto__lt = time_for_archive).order_by("-deliveryto")
        dresspaginator = Paginator(rent_user_dressrentals, 20)
        dresspage = request.GET.get("page")
        rent_user_dressrentals = paginator.get_page(dresspage)
        dressdata["rentals"] = rent_user_dressrentals
        return render(request, "rendress/userarchrentals.html",
dressdata)

```

```

def view_category_dresses(request, dresscategory_slug,
orderapp, typical = True):
    dressdata = {}
    dress_categories =
Dresscategory.objects.all().order_by("categoryname")
    dressdata["categories"] = dress_categories
    dress_styles = Style.objects.all().order_by("name")
    dressdata["styles"] = dress_styles
    if request.user.is_authenticated:
        rent_user = Rentuser.objects.filter(userinapp =
request.user)
        dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
        dressdata["lists"] = dress_lists
    if dresscategory_id:
        try:
            dress_category = Dresscategory.objects.get(slug
= dresscategory_slug)
        except Dresscategory.DoesNotExist:
            dress_category = None
        if not dress_category:
            dressdata["error"] = "Такої категорії суконь не
існує"
        return render(request,
"rendress/categ_dresses.html", dressdata)
        dressdata["category"] = dress_category
        if typical and orderapp == 0:
            category_dresses =
Dress.objects.select_related("collection", "designer",
"collection__brand").filter(dresscategory =
dress_category).order_by("appcaption")
            for category_dress in category_dresses:
                category_dress.sizes =
Dresssize.objects.filter(dress =
category_dress).values("size").distinct()
            elif typical and orderapp == 1:
                category_dresses =
Dress.objects.select_related("collection", "designer",
"collection__brand").filter(dresscategory =
dress_category).order_by("-appcaption")
                for category_dress in category_dresses:
                    category_dress.sizes =
Dresssize.objects.filter(dress =
category_dress).values("size").distinct()
            elif not typical and orderapp == 0:
                category_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dress__dresscategory =
dress_category).order_by("dress__appcaption")
            elif not typical and orderapp == 1:

```

```

        category_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dress__dresscategory =
dress_category).order_by("-dress__appcaption")
        elif not typical and orderapp == 2:
            category_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dress__dresscategory =
dress_category).order_by("rank")
        elif not typical and orderapp == 3:
            category_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dress__dresscategory =
dress_category).order_by("-rank")
            dresspaginator = Paginator(category_dresses, 20)
            dresspage = request.GET.get("page")
            category_dresses = paginator.get_page(dresspage)
            if request.user.is_authenticated:
                if typical:
                    rent_user =
Rentuser.objects.filter(userinapp = request.user)
                    dress_lists =
Dresslist.objects.prefetch_related("dressssizes",
"dressssizes__dress").filter(rentuser = rent_user)
                    for category_dress in category_dresses:
                        category_dress.dress_in_lists = []
                        for dress_list in dress_lists:
                            dress_found = False
                            for dresssize in
dress_list.dressssizes:
                                if category_dress ==
dresssize.dress:
                                    dress_found = True
                                    break
                        if dress_found:
category_dress.dress_in_lists.append(dress_list)
                    else:
                        rent_user =
Rentuser.objects.filter(userinapp = request.user)
                        dress_lists =
Dresslist.objects.prefetch_related("dressssizes").filter(rentuser
= rent_user)
                        for category_dress in category_dresses:
                            category_dress.dress_in_lists = []
                            for dress_list in dress_lists:
                                if category_dress in
dress_list.dressssizes:
category_dress.dress_in_lists.append(dress_list)

```

```

        dressdata["dresses"] = category_dresses
        return render(request,
"rendress/categ_dresses.html", dressdata)

    def view_designer(request, designer_slug):
        dressdata = {}
        if request.user.is_authenticated:
            rent_user = Rentuser.objects.filter(userinapp =
request.user)
            dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
            dressdata["lists"] = dress_lists
            dress_categories =
Dresscategory.objects.all().order_by("categoryname")
            dressdata["categories"] = dress_categories
            dress_styles = Style.objects.all().order_by("name")
            dressdata["styles"] = dress_styles
            if designer_slug:
                try:
                    dress_designer = Designer.objects.get(slug =
designer_slug)
                except Designer.DoesNotExist:
                    dress_designer = None
                if not dress_designer:
                    dressdata["error"] = "Такого дизайнера суконь
не ичыє"
                return render(request,
"rendress/designer.html", dressdata)
                dressdata["designer"] = dress_designer
                designer_dresses =
Dress.objects.select_related("designer", "collection",
"collection__brand").filter(designer =
dress_designer).order_by("appcaption")[:20]
                for designer_dress in designer_dresses:
                    designer_dress.sizes =
Dresssize.objects.filter(dress =
designer_dress).values("size").distinct()
                    designer_dress.sizes =
Dresssize.objects.select_related("dress__designer",
"dress__collection", "dress__collection__brand", "dress",
"dressowner").filter(designer = dress_designer).order_by("-
rank")[:20]
                    designer_brands =
Dress.objects.select_related("designer", "collection",
"collection__brand").filter(designer =
dress_designer).values("collection__brand").distinct()
                    if request.user.is_authenticated:
                        rent_user = Rentuser.objects.filter(userinapp =
request.user)
                        dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)

```

```

        for designer_dress in designer_dresses:
            designer_dress.dress_in_lists = []
            for dress_list in dress_lists:
                dress_found = False
                for dresssize in dress_list.dresssizes:
                    if designer_dress ==
dresssize.dress:
                        dress_found = True
                        break
                if dress_found:
                    designer_dress.dress_in_lists.append(dress_list)
                    for designer_dress in designer_dresssizes:
                        designer_dress.dress_in_lists = []
                        for dress_list in dress_lists:
                            if designer_dress in
dress_list.dresssizes:
                                designer_dress.dress_in_lists.append(dress_list)
                                dressdata["dresses"] = designer_dresses
                                dressdata["dresssizes"] = designer_dresssizes
                                dressdata["brands"] = designer_brands
                                return render(request, "rendress/designer.html",
dressdata)

    def view_collection_dresses(request, collection_slug,
orderapp, typical = True):
        dressdata = {}
        if request.user.is_authenticated:
            rent_user = Rentuser.objects.filter(userinapp =
request.user)
            dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
            dressdata["lists"] = dress_lists
            dress_categories =
Dresscategory.objects.all().order_by("categoryname")
            dressdata["categories"] = dress_categories
            dress_styles = Style.objects.all().order_by("name")
            dressdata["styles"] = dress_styles
            if collection_slug:
                try:
                    dress_collection = Collection.objects.get(slug
= collection_slug)
                except Collection.DoesNotExist:
                    dress_collection = None
                if not dress_collection:
                    dressdata["error"] = "Такої колекції суконь не
існує"
                return render(request,
"rendress/collection_dresses.html", dressdata)
            dressdata["collection"] = dress_collection
            if typical and orderapp == 0:

```



```

        for dress_list in dress_lists:
            dress_found = False
            for dresssize in
dress_list.dresssizes:
                if collection_dress ==
dresssize.dress:
                    dress_found = True
                    break
            if dress_found:

collection_dress.dress_in_lists.append(dress_list)
        else:
            rent_user =
Rentuser.objects.filter(userinapp = request.user)
            dress_lists =
Dresslist.objects.prefetch_related("dresssizes").filter(rentuser
= rent_user)
            for collection_dress in collection_dresses:
                collection_dress.dress_in_lists = []
                for dress_list in dress_lists:
                    if collection_dress in
dress_list.dresssizes:

collection_dress.dress_in_lists.append(dress_list)
                dressdata["dresses"] = collection_dresses
                return render(request,
"rendress/collection_dresses.html", dressdata)

def view_rentuser(request, rentuser_username = None):
    dressdata = {}
    if request.user.is_authenticated:
        rent_user = Rentuser.objects.filter(userinapp =
request.user)
        dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
        dressdata["lists"] = dress_lists
        dress_categories =
Dresscategory.objects.all().order_by("categoryname")
        dressdata["categories"] = dress_categories
        dress_styles = Style.objects.all().order_by("name")
        dressdata["styles"] = dress_styles
        if not rentuser_username:
            rentuser_for_search =
Rentuser.objects.filter(userinapp = request.user)
        else:
            try:
                rentuser_for_search =
Rentuser.objects.get(userinapp__username = rentuser_username)
            except Rentuser.DoesNotExist:
                rentuser_for_search = None
            if not rentuser_for_search:

```

```

        dressdata["error"] = "Такого користувача не  

        ichye"
        return render(request,
            "rendress/rentuser.html", dressdata)
        dressdata["profile"] = rentuser_for_search
        rentuser_dresses =
        Dresssize.objects.select_related("dress", "dress__collection",
            "dress__designer", "dressowner",
            "dress__collection__brand").filter(dressowner =
            rentuser_for_search).order_by("-rank")[:20]
        if request.user.is_authenticated:
            rent_user = Rentuser.objects.filter(userinapp =
            request.user)
            dress_lists =
            Dresslist.objects.prefetch_related("dresssizes").filter(rentuser
            = rent_user)
            for rentuser_dress in rentuser_dresses:
                rentuser_dress.dress_in_lists = []
                for dress_list in dress_lists:
                    if rentuser_dress in
            dress_list.dresssizes:
            rentuser_dress.dress_in_lists.append(dress_list)
            dressdata["dresses"] = rentuser_dresses
            return render(request,
            "rendress/rentuser_dresses.html", dressdata)

        def view_dress_rental(request, dressrental_id):
            dressdata = {}
            if request.user.is_authenticated:
                rent_user = Rentuser.objects.filter(userinapp =
            request.user)
                dress_lists =
            Dresslist.objects.prefetch_related("dresssizes",
            "dresssizes__dress").filter(rentuser = rent_user)
                dressdata["lists"] = dress_lists
                dress_categories =
            Dresscategory.objects.all().order_by("categoryname")
                dressdata["categories"] = dress_categories
                dress_styles = Style.objects.all().order_by("name")
                dressdata["styles"] = dress_styles
                if not request.user.is_authenticated:
                    return render(request, "rendress/dressrental.html",
            dressdata)
                rent_user = Rentuser.objects.filter(userinapp =
            request.user)
                try:
                    dress_rental =
            Dressrental.objects.select_related("rentuser").get(id =
            dressrental_id)
                except Dressrental.DoesNotExist:
                    dress_rental = None
                if not dress_rental:

```

```

        dressdata["error"] = "Такого прокаты не існує"
        return render(request, "rendress/dressrental.html",
dressdata)
        dresses =
Dressrentalcontent.objects.select_related("dresssize",
"dresssize__dressowner", "dresssize__dress",
"dresssize__dress__collection",
"dresssize__dress__collection_brand",
"dresssize__dress__designer")
        dress_owner = dresses.first()
        if rent_user == dress_rental.rentuser or
dress_owner.dresssize.dressowner == rent_user or
rent_user.is_staff:
            dressdata["rental"] = dress_rental
            dressdata["content"] = dresses
            return render(request, "rendress/dressrental.html",
dressdata)
        else:
            dressdata["error"] = "Доступ заборонений"
            return render(request, "rendress/dressrental.html",
dressdata)

    def view_brand_dresses(request, brand_id, orderapp, typical
= True):
        dressdata = {}
        if request.user.is_authenticated:
            rent_user = Rentuser.objects.filter(userinapp =
request.user)
            dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
            dressdata["lists"] = dress_lists
            dress_categories =
Dresscategory.objects.all().order_by("categoryname")
            dressdata["categories"] = dress_categories
            dress_styles = Style.objects.all().order_by("name")
            dressdata["styles"] = dress_styles
            if brand_id:
                try:
                    dress_brand = Brand.objects.get(id = brand_id)
                except Brand.DoesNotExist:
                    dress_brand = None
                if not dress_brand:
                    dressdata["error"] = "Такого бренду суконь не
існує"
                return render(request,
"rendress/brand_dresses.html", dressdata)
                dressdata["brand"] = dress_brand
                if typical and orderapp == 0:
                    brand_dresses =
Dress.objects.select_related("designer", "collection",
"collection__brand").filter(collection__brand =
dress_brand).order_by("appcaption")

```



```

        if dress_found:
brand_dress.dress_in_lists.append(dress_list)
        else:
            rent_user =
Rentuser.objects.filter(userinapp = request.user)
            dress_lists =
Dresslist.objects.prefetch_related("dresssizes").filter(rentuser
= rent_user)
            for brand_dress in brand_dresses:
                brand_dress.dress_in_lists = []
                for dress_list in dress_lists:
                    if brand_dress in
dress_list.dresssizes:
brand_dress.dress_in_lists.append(dress_list)
                dressdata["dresses"] = brand_dresses
                return render(request,
"rendress/brand_dresses.html", dressdata)

    def view_rentuser_dress_rentals(request):
        dressdata = {}
        if request.user.is_authenticated:
            rent_user = Rentuser.objects.filter(userinapp =
request.user)
            dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
            dressdata["lists"] = dress_lists
            dress_categories =
Dresscategory.objects.all().order_by("categoryname")
            dressdata["categories"] = dress_categories
            dress_styles = Style.objects.all().order_by("name")
            dressdata["styles"] = dress_styles
            if not request.user.is_authenticated:
                return render(request, "rendress/userrentals.html",
dressdata)
            rent_user = Rentuser.objects.filter(userinapp =
request.user)
            rent_user_dressrentals =
Dressrental.objects.filter(rentuser = rent_user, state =
0).order_by("-createdon")
            dressdata["rentals"] = rent_user_dressrentals
            return render(request, "rendress/userrentals.html",
dressdata)

    def view_designer_dresses(request, designer_slug, orderapp,
typical = True):
        dressdata = {}
        if request.user.is_authenticated:
            rent_user = Rentuser.objects.filter(userinapp =
request.user)

```

```

        dress_lists =
Dresslist.objects.prefetch_related("dressssizes",
"dressssizes__dress").filter(rentuser = rent_user)
        dressdata["lists"] = dress_lists
        dress_categories =
Dresscategory.objects.all().order_by("categoryname")
        dressdata["categories"] = dress_categories
        dress_styles = Style.objects.all().order_by("name")
        dressdata["styles"] = dress_styles
        if designer_slug:
            try:
                dress_designer = Designer.objects.get(slug =
designer_slug)
            except Designer.DoesNotExist:
                dress_designer = None
            if not dress_designer:
                dressdata["error"] = "Такого дизайнера суконь
не існує"
            return render(request,
"rendress/designer_dresses.html", dressdata)
            dressdata["designer"] = dress_designer
            if typical and orderapp == 0:
                designer_dresses =
Dress.objects.select_related("designer", "collection",
"collection__brand").filter(designer =
dress_designer).order_by("appcaption")
                for designer_dress in designer_dresses:
                    designer_dress.sizes =
Dresssize.objects.filter(dress =
designer_dress).values("size").distinct()
                elif typical and orderapp == 1:
                    designer_dresses =
Dress.objects.select_related("designer", "collection",
"collection__brand").filter(designer =
dress_designer).order_by("-appcaption")
                    for designer_dress in designer_dresses:
                        designer_dress.sizes =
Dresssize.objects.filter(dress =
designer_dress).values("size").distinct()
                elif not typical and orderapp == 0:
                    designer_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dress__designer =
dress_designer).order_by("dress__appcaption")
                elif not typical and orderapp == 1:
                    designer_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dress__designer =
dress_designer).order_by("-dress__appcaption")
                elif not typical and orderapp == 2:

```

```

        designer_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dress__designer =
dress_designer).order_by("rank")
        elif not typical and orderapp == 3:
            designer_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dress__designer =
dress_designer).order_by("-rank")
            dresspaginator = Paginator(designer_dresses, 20)
            dresspage = request.GET.get("page")
            designer_dresses = paginator.get_page(dresspage)
            if request.user.is_authenticated:
                if typical:
                    rent_user =
Rentuser.objects.filter(userinapp = request.user)
                    dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
                    for designer_dress in designer_dresses:
                        designer_dress.dress_in_lists = []
                        for dress_list in dress_lists:
                            dress_found = False
                            for dresssize in
dress_list.dresssizes:
                                if designer_dress ==
dresssize.dress:
                                    dress_found = True
                                    break
                            if dress_found:
                                designer_dress.dress_in_lists.append(dress_list)
                        else:
                            rent_user =
Rentuser.objects.filter(userinapp = request.user)
                            dress_lists =
Dresslist.objects.prefetch_related("dresssizes").filter(rentuser
= rent_user)
                            for designer_dress in designer_dresses:
                                designer_dress.dress_in_lists = []
                                for dress_list in dress_lists:
                                    if designer_dress in
dress_list.dresssizes:
                                        designer_dress.dress_in_lists.append(dress_list)
                                dressdata["dresses"] = designer_dresses
                                return render(request,
"rendress/designer_dresses.html", dressdata)

def view_dressbrand(request, brand_id):
    dressdata = {}

```



```

        for dress_list in dress_lists:
            if brand_dress in
dress_list.dresssizes:

brand_dress.dress_in_lists.append(dress_list)
    dressdata["dresses"] = brand_dresses
    dressdata["dresssizes"] = brand_dresssizes
    dressdata["collections"] = brand_collections
    return render(request, "rendress/brand.html",
dressdata)

    def view_rent_user_dresses(request, orderapp,
rentuser_username = None):
        dressdata = {}
        if request.user.is_authenticated:
            rent_user = Rentuser.objects.filter(userinapp =
request.user)
            dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
            dressdata["lists"] = dress_lists
            dress_categories =
Dresscategory.objects.all().order_by("categoryname")
            dressdata["categories"] = dress_categories
            dress_styles = Style.objects.all().order_by("name")
            dressdata["styles"] = dress_styles
            if not rentuser_username:
                rentuser_for_search =
Rentuser.objects.filter(userinapp = request.user)
            else:
                try:
                    rentuser_for_search =
Rentuser.objects.get(userinapp__username = rentuser_username)
                except Rentuser.DoesNotExist:
                    rentuser_for_search = None
                if not rentuser_for_search:
                    dressdata["error"] = "Такого користувача не
ичує"

            return render(request,
"rendress/rentuser_dresses.html", dressdata)
            dressdata["profile"] = rentuser_for_search
            if orderapp == 0:
                rentuser_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dressowner =
rentuser_for_search).order_by("dress__appcaption")
            elif orderapp == 1:
                rentuser_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dressowner =
rentuser_for_search).order_by("-dress__appcaption")

```

```

        elif orderapp == 2:
            rentuser_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dressowner =
rentuser_for_search).order_by("-rank")
        elif orderapp == 3:
            rentuser_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dressowner =
rentuser_for_search).order_by("rank")
            dresspaginator = Paginator(rentuser_dresses, 20)
            dresspage = request.GET.get("page")
            rentuser_dresses = paginator.get_page(dresspage)
            if request.user.is_authenticated:
                rent_user = Rentuser.objects.filter(userinapp =
request.user)

                dress_lists =
Dresslist.objects.prefetch_related("dresssizes").filter(rentuser
= rent_user)

                for rentuser_dress in rentuser_dresses:
                    rentuser_dress.dress_in_lists = []
                    for dress_list in dress_lists:
                        if rentuser_dress in
dress_list.dresssizes:
                            rentuser_dress.dress_in_lists.append(dress_list)
                            dressdata["dresses"] = rentuser_dresses
                            return render(request,
"rendress/rentuser_dresses.html", dressdata)

    def view_rentuser_dress_rentals_planned(request):
        dressdata = {}
        if request.user.is_authenticated:
            rent_user = Rentuser.objects.filter(userinapp =
request.user)
            dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
            dressdata["lists"] = dress_lists
            dress_categories =
Dresscategory.objects.all().order_by("categoryname")
            dressdata["categories"] = dress_categories
            dress_styles = Style.objects.all().order_by("name")
            dressdata["styles"] = dress_styles
            if not request.user.is_authenticated:
                return render(request,
"rendress/userplanrentals.html", dressdata)
            rent_user = Rentuser.objects.filter(userinapp =
request.user)
            time_for_plan = datetime.now()

```

```

        rent_user_dressrentals =
Dressrental.objects.filter(rentuser = rent_user, state = 1,
deliveryto__gte = time_for_plan).order_by("deliveryto")
        dresspaginator = Paginator(rent_user_dressrentals, 20)
        dresspage = request.GET.get("page")
        rent_user_dressrentals = paginator.get_page(dresspage)
        dressdata["rentals"] = rent_user_dressrentals
        return render(request, "rendress/userplanrentals.html",
dressdata)

def view_style_dresses(request, style_id, orderapp, typical
= True):
    dressdata = {}
    if request.user.is_authenticated:
        rent_user = Rentuser.objects.filter(userinapp =
request.user)
        dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
        dressdata["lists"] = dress_lists
        dress_categories =
Dresscategory.objects.all().order_by("categoryname")
        dressdata["categories"] = dress_categories
        dress_styles = Style.objects.all().order_by("name")
        dressdata["styles"] = dress_styles
        if style_id:
            try:
                dress_style = Style.objects.get(id = style_id)
            except Style.DoesNotExist:
                dress_style = None
            if not dress_style:
                dressdata["error"] = "Такого стилю суконь не
ичные"

            return render(request,
"rendress/style_dresses.html", dressdata)
            if typical and orderapp == 0:
                style_dresses =
Dress.objects.prefetch_related("styles").select_related("designer", "collection", "collection__brand").filter(styles__id_exact =
style.id).order_by("appcaption")
                for style_dress in style_dresses:
                    style_dress.sizes =
Dresssize.objects.filter(dress =
designer_dress).values("size").distinct()
                elif typical and orderapp == 1:
                    style_dresses =
Dress.objects.prefetch_related("styles").select_related("designer", "collection", "collection__brand").filter(styles__id_exact =
style.id).order_by("-appcaption")
                    for style_dress in style_dresses:
                        style_dress.sizes =
Dresssize.objects.filter(dress =
designer_dress).values("size").distinct()

```

```

        elif not typical and orderapp == 0:
            style_dresses =
Dresssize.objects.prefetch_related("dress__styles").select_relat
ed("dress", "dress__collection", "dress__designer",
"dressowner",
"dress__collection__brand").filter(dress__styles__id_exact =
style.id).order_by("dress__appcaption")
        elif not typical and orderapp == 1:
            style_dresses =
Dresssize.objects.prefetch_related("dress__styles").select_relat
ed("dress", "dress__collection", "dress__designer",
"dressowner",
"dress__collection__brand").filter(dress__styles__id_exact =
style.id).order_by("-dress__appcaption")
        elif not typical and orderapp == 2:
            style_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dress__styles__id_exact =
style.id).order_by("rank")
        elif not typical and orderapp == 3:
            style_dresses =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dress__styles__id_exact =
style.id).order_by("-rank")
            dresspaginator = Paginator(style_dresses, 20)
            dresspage = request.GET.get("page")
            style_dresses = paginator.get_page(dresspage)
            if request.user.is_authenticated:
                if typical:
                    rent_user =
Rentuser.objects.filter(userinapp = request.user)
                    dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
                    for style_dress in style_dresses:
                        style_dress.dress_in_lists = []
                        for dress_list in dress_lists:
                            dress_found = False
                            for dresssize in
dress_list.dresssizes:
                                if style_dress ==
dresssize.dress:
                                    dress_found = True
                                    break
                        if dress_found:
style_dress.dress_in_lists.append(dress_list)
                    else:
                        rent_user =
Rentuser.objects.filter(userinapp = request.user)

```

```

        dress_lists =
Dresslist.objects.prefetch_related("dresssizes").filter(rentuser
= rent_user)
        for style_dress in style_dresses:
            style_dress.dress_in_lists = []
            for dress_list in dress_lists:
                if style_dress in
dress_list.dresssizes:

style_dress.dress_in_lists.append(dress_list)
        dressdata["dresses"] = style_dresses
        return render(request,
"rendress/style_dresses.html", dressdata)

def view_dresscollection(request, collection_slug):
    dressdata = {}
    if request.user.is_authenticated:
        rent_user = Rentuser.objects.filter(userinapp =
request.user)
        dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
        dressdata["lists"] = dress_lists
        dress_categories =
Dresscategory.objects.all().order_by("categoryname")
        dressdata["categories"] = dress_categories
        dress_styles = Style.objects.all().order_by("name")
        dressdata["styles"] = dress_styles
        if collection_id:
            try:
                dress_collection = Collection.objects.get(slug
= collection_slug)
            except Collection.DoesNotExist:
                dress_collection = None
            if not dress_collection:
                dressdata["error"] = "Такої колекції суконь не
існує"
            return render(request,
"rendress/collection.html", dressdata)
            dressdata["collection"] = dress_collection
            collection_dresses =
Dress.objects.select_related("designer",
"collection__brand").filter(collection =
dress_collection).order_by("appcaption")[:20]
            collection_dresssizes =
Dresssize.objects.select_related("dress", "dress__collection",
"dress__designer", "dressowner",
"dress__collection__brand").filter(dress__collection =
dress_collection).order_by("-rank")[:20]
            collection_designers =
Dress.objects.select_related("designer",
"collection__brand").filter(collection =
dress_collection).values("designer").distinct()

```

```

        if request.user.is_authenticated:
            rent_user = Rentuser.objects.filter(userinapp =
request.user)
            dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
            for collection_dress in collection_dresses:
                collection_dress.dress_in_lists = []
                for dress_list in dress_lists:
                    dress_found = False
                    for dresssize in dress_list.dresssizes:
                        if collection_dress ==
dresssize.dress:
                            dress_found = True
                            break
                    if dress_found:
collection_dress.dress_in_lists.append(dress_list)
                for collection_dress in collection_dresssizes:
                    collection_dress.dress_in_lists = []
                    for dress_list in dress_lists:
                        if collection_dress in
dress_list.dresssizes:
collection_dress.dress_in_lists.append(dress_list)
                    dressdata["dresses"] = collection_dresses
                    dressdata["dresssizes"] = collection_dresssizes
                    dressdata["designers"] = collection_designers
                    return render(request, "rendress/collection.html",
dressdata)

    def view_rentuser_dress_lists(request):
        dressdata = {}
        if request.user.is_authenticated:
            rent_user = Rentuser.objects.filter(userinapp =
request.user)
            dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
            dressdata["lists"] = dress_lists
            dress_categories =
Dresscategory.objects.all().order_by("categoryname")
            dressdata["categories"] = dress_categories
            dress_styles = Style.objects.all().order_by("name")
            dressdata["styles"] = dress_styles
            if not request.user.is_authenticated:
                dressdata["error"] = "Користувач не авторизований"
                return render(request, "rendress/dresslists.html",
dressdata)
            rent_user = Rentuser.objects.filter(userinapp =
request.user)
            dress_lists = Dresslist.objects.filter(rentuser =
rent_user)

```

```

        dresspaginator = Paginator(dress_lists, 20)
        dresspage = request.GET.get("page")
        dress_lists = paginator.get_page(dresspage)
        dressdata["lists"] = dress_lists
        return render(request, "rendress/dresslists.html",
dressdata)

    def view_dressowner_rentals_archive(request, dresssize_id):
        dressdata = {}
        if request.user.is_authenticated:
            rent_user = Rentuser.objects.filter(userinapp =
request.user)
            dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
            dressdata["lists"] = dress_lists
            dress_categories =
Dresscategory.objects.all().order_by("categoryname")
            dressdata["categories"] = dress_categories
            dress_styles = Style.objects.all().order_by("name")
            dressdata["styles"] = dress_styles
            if not request.user.is_authenticated:
                return render(request, "rendress/archrentals.html",
dressdata)
            rent_user = Rentuser.objects.filter(userinapp =
request.user)
            time_for_archive = datetime.now()
            if dresssize_id:
                try:
                    dress =
Dresssize.objects.select_related("dressowner").get(id =
dresssize_id)
                except Dresssize.DoesNotExist:
                    dress = None
                if not dress:
                    dressdata["error"] = "Такої сукні не існує"
                    return render(request,
"rendress/archrentals.html", dressdata)
                else:
                    dressdata["error"] = "Немає сукні"
                    return render(request, "rendress/archrentals.html",
dressdata)
                if dress.dressowner != rent_user
                    dressdata["error"] = "Немає прав доступу"
                    return render(request, "rendress/archrentals.html",
dressdata)
                dressdata["dress"] = dress
                dress_rental_contents =
Dressrentalcontent.object.filter(dresssize =
dress).values("dressrental__id").distinct()
                if dress_rental_contents:

```

```

        dressrentals = Dressrental.objects.filter(state =
1, deliveryto__lt = time_for_archive, id__in =
dress_rental_contents).order_by("-deliveryto")
        dresspaginator = Paginator(dressrentals, 20)
        dresspage = request.GET.get("page")
        dressrentals = paginator.get_page(dresspage)
        dressdata["rentals"] = dressrentals
        return render(request, "rendress/archrentals.html",
dressdata)

def view_rentuser_dress_list(request, dresslist_id):
    dressdata = {}
    if request.user.is_authenticated:
        rent_user = Rentuser.objects.filter(userinapp =
request.user)
        dress_lists =
Dresslist.objects.prefetch_related("dresssizes",
"dresssizes__dress").filter(rentuser = rent_user)
        dressdata["lists"] = dress_lists
        dress_categories =
Dresscategory.objects.all().order_by("categoryname")
        dressdata["categories"] = dress_categories
        dress_styles = Style.objects.all().order_by("name")
        dressdata["styles"] = dress_styles
        if not request.user.is_authenticated:
            dressdata["error"] = "Користувач не авторизований"
            return render(request, "rendress/dresslist.html",
dressdata)
        rent_user = Rentuser.objects.filter(userinapp =
request.user)
        dress_list = Dresslist.objects.filter(id =
dresslist_id)
        if dress_list.rentuser != rent_user:
            dressdata["error"] = "Немає прав доступу"
            return render(request, "rendress/dresslist.html",
dressdata)
        dressdata["list"] = dress_list
        return render(request, "rendress/dresslist.html",
dressdata)

```

ДОДАТОК В
Слайди презентації

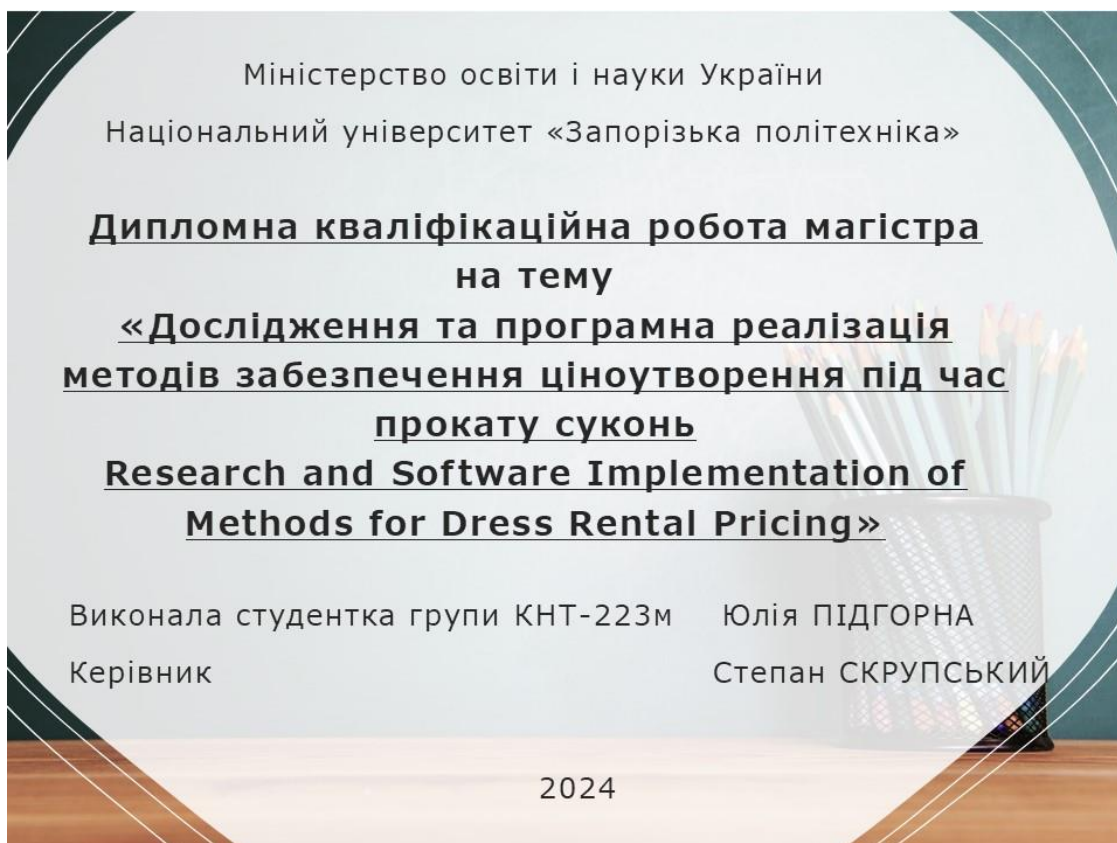


Рисунок В.1 – Початковий слайд

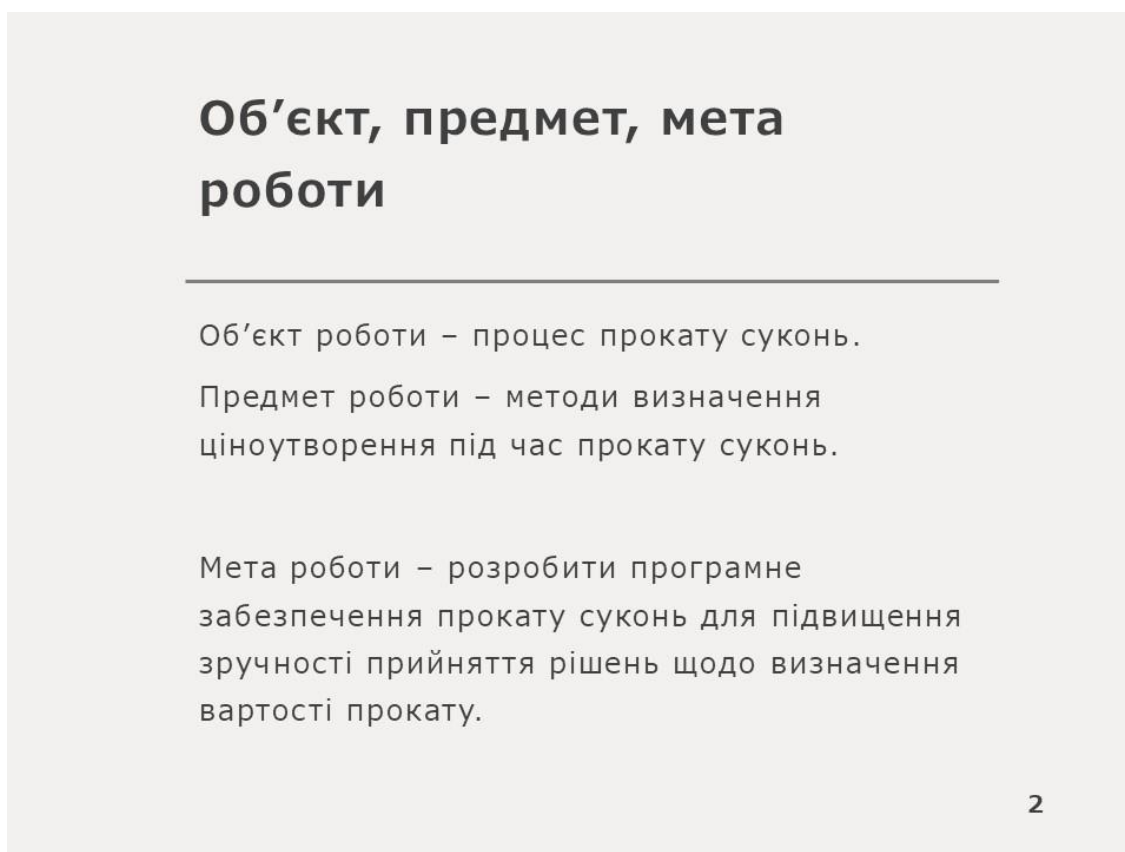


Рисунок В.2 – Об'єкт, предмет, мета роботи

Завдання роботи

- виконати аналіз програмних застосунків прокату суконь;
- розробити метод визначення ціноутворення під час прокату суконь;
- виконати проектування вебзастосунку прокату суконь;
- реалізувати вебзастосунок прокату суконь;
- виконати експериментальне дослідження прогнозування вартості прокату суконь.

3

Рисунок В.3 – Завдання роботи

Порівняння можливостей аналогів розробки вебзастосунку прокату суконь

Показник	Hirestreet	By Rotation	Hurr	RenDress
Прокат суконь	+	+	+	+
Джерело прокату	Сервіс	Сервіс	Сервіс, користувачі	Сервіс, користувачі
Тривалість прокату	21 день	Від 3 днів стандартні варіанти	Від 4 днів стандартні варіанти	Гнучке налаштування
Пошук суконь	3 використанням фільтрів	3 використанням фільтрів	3 використанням фільтрів	Більша кількість фільтрів
Створення переліків перегляду	Тільки збереження у стандартний	Тільки збереження у стандартний	Тільки збереження у стандартний	Користувачькі переліки
Прогнозування вартості прокату	–	–	–	+

4

Рисунок В.4 – Порівняння можливостей аналогів розробки вебзастосунку прокату суконь

- 5

Діаграми прецедентів



Рисунок В.6 – Діаграми прецедентів

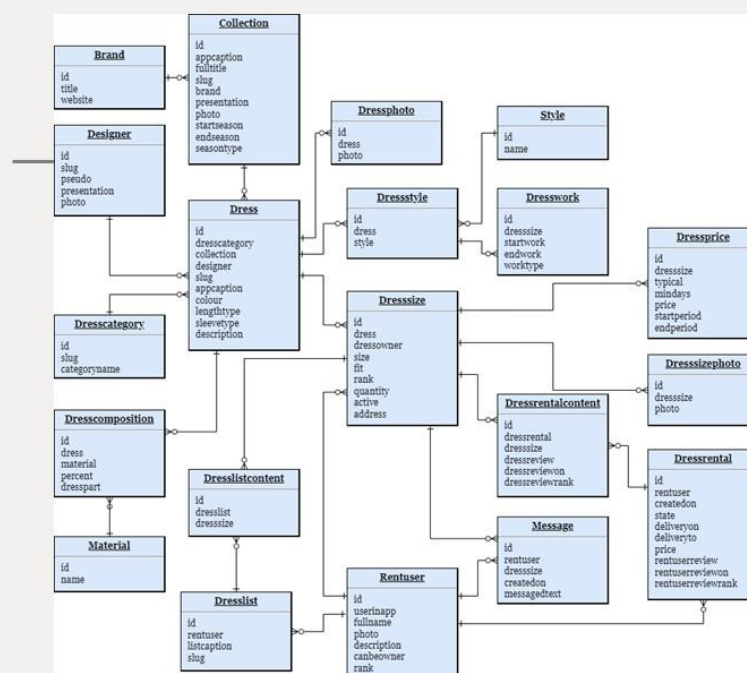
Порівняння можливих рішень для реалізації вебзастосунку прокату суконь

Ознака порівняння	Django	Falcon
Можливості адміністративної роботи з каталогом даних	Присутні, використовуються після реєстрації створених моделей	Відсутні
Програмна робота з базою даних	Присутня для різних систем управління	Відсутня, необхідно вибирати, встановлювати
Надання безпеки створеному вебзастосунку	Широке охоплення за замовчуванням	Необхідно передбачати і додавати
Можливість реалізації методу визначення ціноутворення під час прокату суконь	Мовою Python з використанням відповідних пакетів	Мовою Python з використанням відповідних пакетів

7

Рисунок В.7 – Порівняння можливих рішень для реалізації вебзастосунку прокату суконь

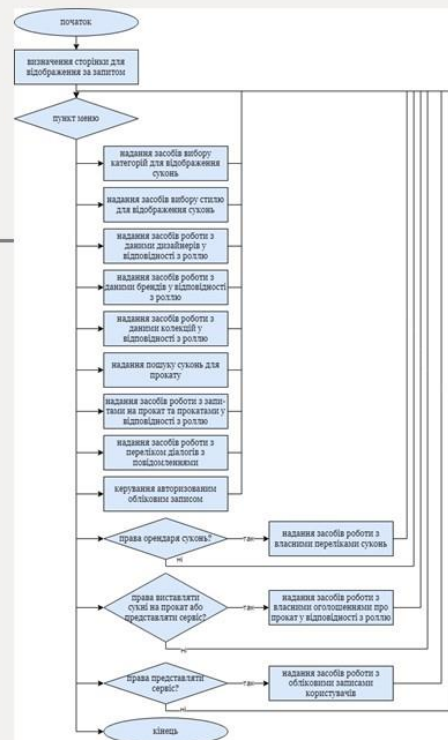
Схема бази даних



8

Рисунок В.8 – Схема бази даних

Схема організації меню програми прокату суконь



9

Рисунок В.9 – Схема організації меню програми прокату суконь

Характеристики	
Категорія	Сукня А-силуету
Бренд	Indigo Dreams
Колекція	Дизайнер: Madfina
Дизайнер	Андреа Деліто
Довжина сукні	Середня
Довжина рукава	Видовжена
Колір	Оливковий
Посада	Висока талия
Розмір	М

Параметри прокату	
Дата прокату	24.03.2024
Відпрацювання	07.03.2024
Вартість	800 грн.
Детальніше	

Сторінка оголошення про прокат сукні з екземпляром, виставленим користувачем

10

Рисунок В.10 – Сторінка оголошення про прокат сукні з екземпляром, виставленим користувачем

Ren
моб

Категорії ▾

Стиль ▾

Дизайнери ▾

Бренди ▾

Коліри ▾

Підсти ▾

Користувачі

Проекти ▾

Серед ▾

Повідомлення

Профіль

Додавання нової сукні

Назва

Категорія

Дизайнер сукні

Коліри

Матеріал

Дизайнер

Вартість виготовлення

Колір

Підстилка

Довжина

Середина

Довжина рукава

Вартість виготовлення

Склад

Додати матеріал

Бавовна

Висота

Вартість виготовлення

Вартість виготовлення

Опис

Вартість виготовлення

Вартість виготовлення

Вартість виготовлення

Сторінка
внесення
даних про
сукню

12

Рисунок В.12 – Сторінка внесення даних про сукню

Результати прогнозування вартості прокату суконь

Метод	Середня абсолютна похибка	Середня квадратична похибка
Дерево ухвалення рішень (бренд, колір, всі категорії)	0,0014377564	7,443249334E-6
Метод опорних векторів (бренд, колір, всі категорії)	0,0236879684	0,0005693981
Лінійна регресія (бренд, колір, всі категорії)	0,0019476795	9,6877848478E-6
Дерево ухвалення рішень (3 найінформативніші ознаки)	0,0013618961	6,6147218613E-6
Метод опорних векторів (3 найінформативніші ознаки)	0,0236879686	0,0005693981
Лінійна регресія (3 найінформативніші ознаки)	0,0019502269	9,7534916269E-6
Дерево ухвалення рішень (5 найінформативніших ознак)	0,001376565	6,7204571135E-6
Метод опорних векторів (5 найінформативніших ознак)	0,0236879683	0,0005693981
Лінійна регресія (5 найінформативніших ознак)	0,0019498398	9,7492971896E-6

13

Рисунок В.13 – Результати прогнозування вартості прокату суконь

Висновки

Мету роботи, що полягала в розробці програмного забезпечення прокату суконь для підвищення зручності прийняття рішень щодо визначення вартості прокату, виконано. Усі завдання, необхідні для цього, включаючи аналіз програмних застосунків прокату суконь, розроблення методу визначення ціноутворення під час прокату суконь, проєктування вебзастосунку прокату суконь, реалізацію вебзастосунку прокату суконь, виконання експериментального дослідження прогнозування вартості прокату суконь, виконано.

Робота має наукову новизну, представлену методом визначення ціноутворення під час прокату суконь, який визначає процес формування вхідних ознак, виділення найбільш інформативних з них, створення моделі на основі дерева ухвалення рішень, що дозволяє прогнозувати вартість прокату сукні.

14

Рисунок В.14 – Висновки