

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ

(назва теми)

ПЛАНУВАННЯ ДІЯЛЬНОСТІ ПЕРУКАРНІ

RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS
FOR PLANNING THE ACTIVITY OF THE HAIRDRESSER'S SHOP

Виконав(ла): студент(ка) 2 курсу, групи КНТ-123м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

МИХАЙЛИЧЕНКО В.О.

(ПРІЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(ПРІЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.

(ПРІЗВИЩЕ та ініціали)

2024_____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

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

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф. _____
Сергій СУББОТІН
« _____ » _____ 2024 року

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

МИХАЙЛИЧЕНКА Владислава Олександровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів планування діяльності перукарні. Research and Software Implementation of Methods for Planning the Activity of the Hairdresser's Shop

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

затверджені наказом закладу вищої освіти від « 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, кількості слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

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

7. Дата видачі завдання « 06 » 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	9-10 тижні	Розділ 5
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	11 тиждень	Додатки
8	Захист роботи	12 тиждень	

Студент(ка)

_____ Владислав МИХАЙЛИЧЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

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

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
101 с., 33 рис., 6 табл., 3 дод., 10 джерел.

ПЛАНУВАННЯ, ПЕРУКАРНЯ, ПРОГНОЗУВАННЯ, ДЕРЕВА
УХВАЛЕННЯ РІШЕНЬ, БРОНЮВАННЯ, .

Об'єкт роботи – процеси підтримки та планування діяльності перукарні. Предмет роботи – методи планування діяльності перукарні.

Мета роботи – розробити програмне забезпечення підтримки діяльності перукарні для підвищення інформованості рішень щодо планування цієї діяльності.

Матеріали, методи та технічні засоби: засоби програмування мовою Python та веброзробки на основі фреймворку Django, методи машинного навчання, система керування базами даних MySQL.

Результати. Розроблене програмне забезпечення здатне підтримувати роботу перукарні з клієнтами, зокрема отримання основної необхідної інформації про ті послуги, які надаються перукарнею, бронювання послуг, планування діяльності перукарні. До основних результатів роботи також відносяться результати експериментального дослідження, під час якого не тільки порівняно різні моделі, до яких належить ARIMA, метод опорних векторів, запропонований метод, але також і різна довжина часового інтервалу.

Висновки. Наукова новизна роботи полягає в методі прогнозування кількості відвідувачів перукарні за день, що характеризується використанням часового ряду, ансамблю дерев рішень на основі методу Random Forest, що дозволяє визначати сумарну кількість відвідувачів за день з інтервалом у задану кількість днів.

Галузь використання. Перукарні та мережі перукарень.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 101 pages, 33 figures, 6 tables, 3 appendixes, 10 sources.

PLANNING, HAIRDRESSER, FORECASTING, DECISION TREES, RESERVATIONS.

The object of the work is the processes of support and planning the activity of the hairdresser's shop. The subject of the work is the methods of planning the activity of the hairdresser's shop.

The goal of the work is to develop software to support the activity of the hairdresser's shop to increase awareness of decisions regarding the planning of this activity.

Materials, methods and technical means: Python programming language and web development tools based on the Django framework, machine learning methods, MySQL database management system.

The results. The developed software is able to support the work of the hairdresser's shop with clients, in particular, obtaining the basic necessary information about the services provided by the hairdresser's shop, booking services, planning the activities of the hairdresser's shop. The main results of the work also include the results of an experimental study, during which not only different models are compared, which include ARIMA, the method of support vectors, the proposed method, but also different lengths of the time interval.

Conclusions. The scientific novelty of the work consists in the method of forecasting the number of visitors to the hair salon per day, which is characterized by the use of a time series, an ensemble of decision trees based on the Random Forest method, which allows determining the total number of visitors per day with the given interval.

Field of use. Hairdresser's shops and chains of hairdresser's shops.

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 підтримки діяльності перукарні.....	10
1.2 Визначення рівня створюваного вебзастосунку підтримки діяльності перукарні	15
1.3 Висновки за розділом 1	16
2 Матеріали і методи.....	17
2.1 Задача прогнозування кількості відвідувачів перукарні за день	17
2.2 Метод прогнозування кількості відвідувачів перукарні за день.....	18
2.3 Висновки за розділом 2	21
3 Проєктування програмного забезпечення	22
3.1 Визначення сценаріїв роботи користувачів з вебзастосунком підтримки діяльності перукарні.....	22
3.2 Вибір засобів розробки вебзастосунку підтримки діяльності перукарні	27
3.3 Проєктування структури бази даних.....	28
3.4 Висновки за розділом 3	38
4 Основні рішення щодо реалізації програмного забезпечення	39
4.1 Загальний порядок роботи програми підтримки діяльності перукарні	39
4.2 Реалізовані програмні одиниці у складі вебзастосунку підтримки діяльності перукарні.....	40
4.3 Висновки за розділом 4	54
5 Експлуатація, тестування та експериментальне дослідження програми	55
5.1 Основне призначення програми	55
5.2 Виконання програми.....	55

5.3 Експериментальне дослідження прогнозування кількості відвідувачів перукарні за день	59
5.4 Висновки за розділом 5	64
Висновки	65
Перелік джерел посилання	66
Додаток А Технічне завдання	67
Додаток Б Текст програми	72
Додаток В Слайди презентації.....	93

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ARIMA – AutoRegressive Integrated Moving Average;

MAE – Mean Absolute Error;

MSE – Mean Squared Error;

RMSE – Root Mean Squared Error.

ВСТУП

Перукарські послуги є специфічною галуззю. З одного боку це дуже традиційна галузь, хоча в сучасному світі вона і активно розвивається за рахунок того, що більша кількість людей бажає змінювати свій зовнішній вигляд.

З іншого боку перукарень на ринку зазвичай представлено достатньо багато, але отримати потрібну інформацію, обрати такого перукаря, якому одразу з більшим ступенем можна довіритись, достатньо складно. Кожний має свій власний досвід, при цьому поділитися ним у вебзастосунках перукарень зазвичай ніхто не може, оскільки функція створення відгуків або подібна функція відсутня.

У той же час інформація, яка загалом присутня у вебзастосунку, зазвичай є дуже обмеженою. Є певна статична інформація, а користувачу надається можливість забронювати певні перукарські послуги, що не дуже сприяє зануреності в процес взаємодії з вебзастосунком користувача. Відповідно з іншого боку засоби планування діяльності перукарні обмежені тільки контролем та керуванням тих бронювань, які здійснили клієнти. Застосування ж засобів машинного навчання при накопиченні потрібної інформації у вебзастосунку підтримки діяльності перукарні може бути достатньо корисним для прийняття рішень щодо відпусток працівників перукарні, розширення або зменшення штату, зокрема тимчасово. Усе це потребує планування, а планування потребує розуміння того, яка кількість клієнтів звернеться до перукарні через певний час.

Відповідно метою роботи є розробити програмне забезпечення підтримки діяльності перукарні для підвищення інформованості рішень щодо планування цієї діяльності.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Функціональні можливості та концепція існуючих вебзастосунків підтримки діяльності перукарні

У світі існує величезна кількість перукарень або салонів подібного типу, практично кожен з яких має свій власний вебсайт. Результати проведеного аналізу вказують на те, що функціонально такі програмні застосунки дуже сильно відрізняються.

Спочатку було виділено можливості вебзастосунку перукарні Barberking (рис. 1.1) [1]. Ця перукарня позиціонує себе як така, що надає послуги найвищого класу, тому важливим було проаналізувати можливості вебзастосунку. Однак, проведений аналіз вказав, що сам вебзастосунок має багато недоліків зокрема під час відображення. Елементи нерідко накладаються один на один, перекриваючись. Функціональна ж можливість охоплюють перегляд послуг з вказанням вартості таких послуг та можливість бронювання послуг перукарів. Хоча дослідити функцію бронювання не вдалось, адже в браузері вона не відобразилась.

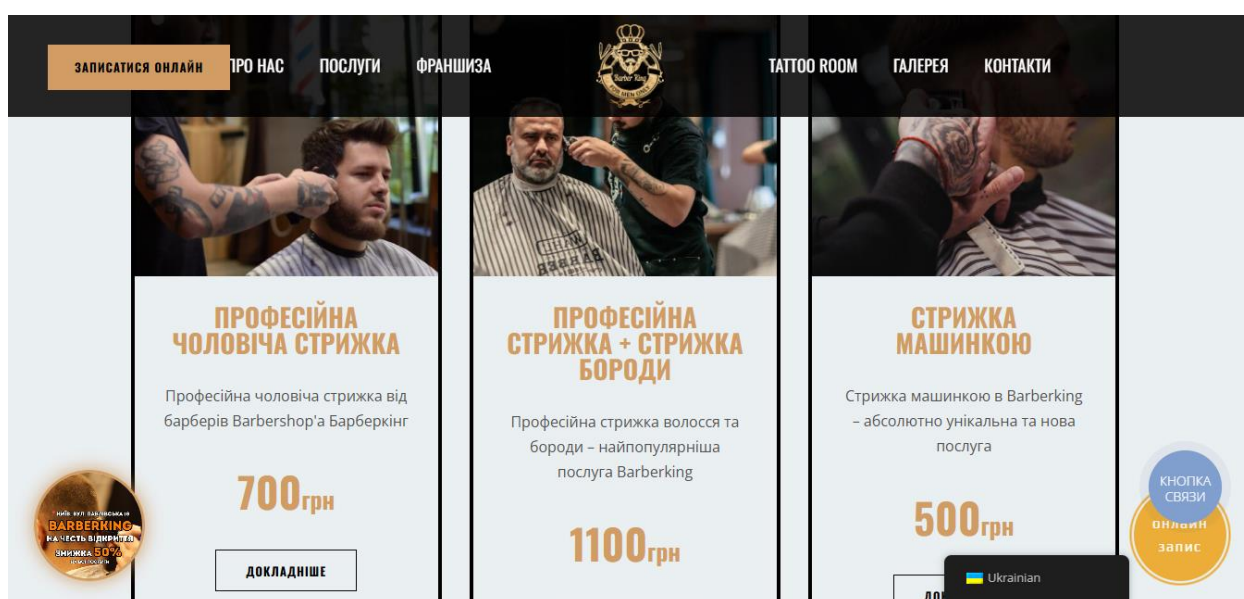


Рисунок 1.1 – Вебсторінка послуг перукарні Barberking

Перукарня Barberking надає послуги чоловікам, а сам ряд послуг, наявний через вебзастосунок, є досить обмеженим. Аналізуючи можливості вебзастосунків таких перукарень, було розглянуто невелику мережу перукарень у Нью-Йорку Barber Shop NYC (рис. 1.2) [2]. Функціонально рішення схоже на попереднє, хоча додатково містить, наприклад, блог, де публікуються матеріали за темою.

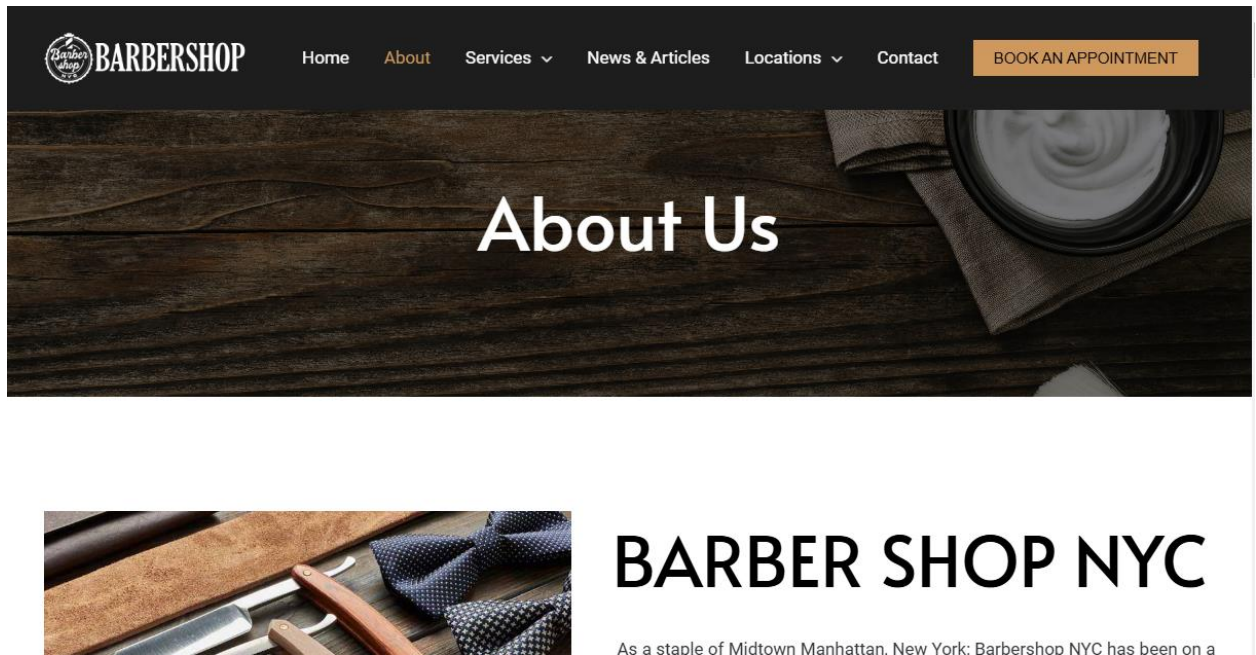


Рисунок 1.2 – Вебзастосунок Barber Shop NYC

Наявне повноцінне бронювання послуг з відображенням вартості таких послуг (рис. 1.3). Можна обрати потрібний час та задати конкретного перукаря, якщо в цьому є необхідність.

Якщо ж розглядати послуги та функціональні можливості відповідних вебзастосунків перукарень, які надають послуги не тільки чоловікам, але і жінкам, а також такі, що надають ширший набір послуг, то кількість можливостей збільшується.

Вебзастосунок мережі Larry King (рис. 1.4) дозволяє проаналізувати такі можливості [3]. Він містить окрім представлення інформації про перукарні також і вбудований магазин, який дозволяє придбати різні необхідні засоби для волосся. Ці засоби можна зберігати для повернення до них.

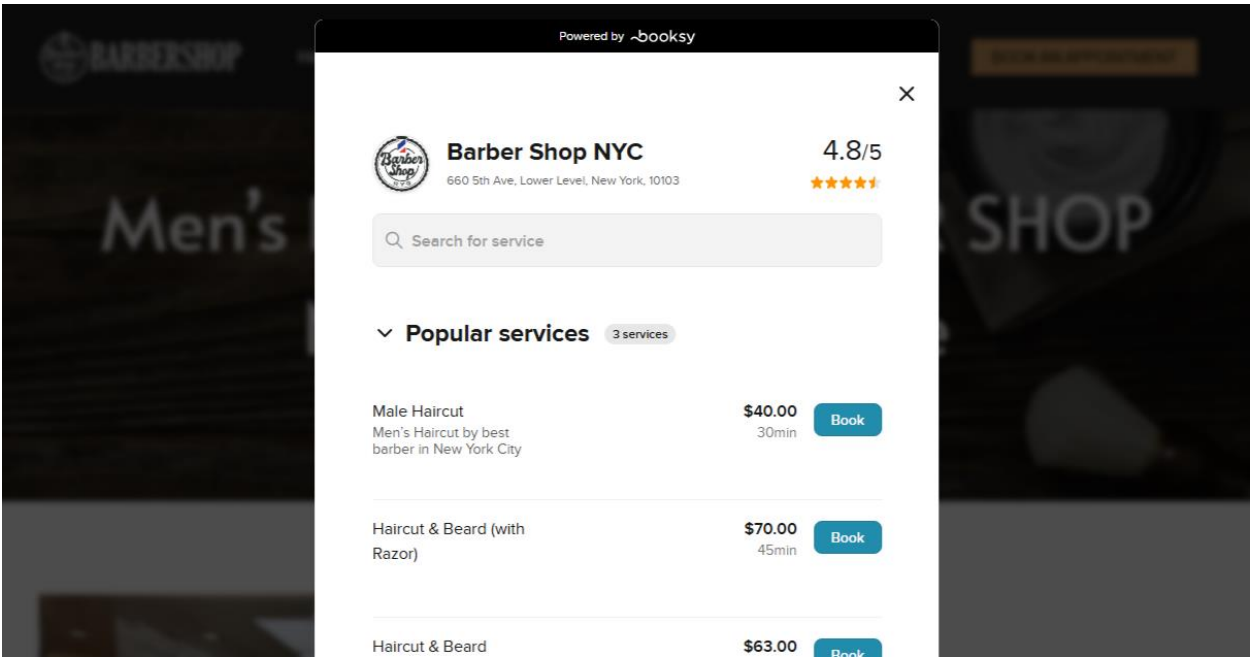


Рисунок 1.3 – Бронювання послуг перукарні Barber Shop NYC

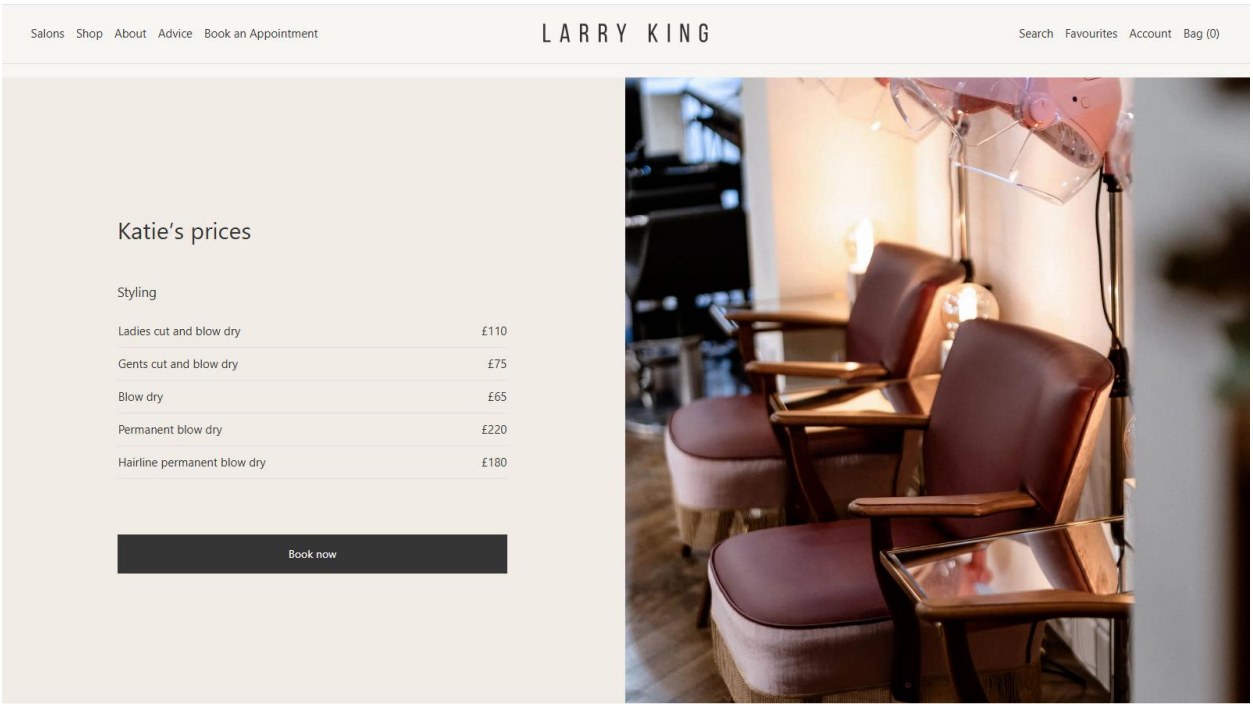


Рисунок 1.4 – Вебзастосунок мережі Larry King

Стосовно самих перукарень, то вебзастосунок дозволяє отримати інформацію про кожну конкретну перукарню, забронювати послуги в перукарні, визначаючи конкретну послугу, конкретного перукаря або консультанта, а тоді вже його відповідну послугу. При цьому в кінці потрібно

вже за конкретним перукарем обрати день для бронювання та один з варіантів доступного часу. Також за кожною перукарнею доступний перелік стилістів та колористів, що там працюють, дозволяючи переглянути перелік послуг кожного з них, отримати про них інформацію, забронювати їхні послуги. За кожним перукарем доступні також фотографії його робіт. Окрім того користувачам доступний також набір статей про волосся, зокрема фарбування, виділений у окремий розділ.

Вебзастосунок Blue Tit (рис. 1.5) також надає інформацію про перукарні в мережі, перелік перукарів та основну інформацію за кожною перукарнею [4].

Our prices are gender neutral as we believe they should be determined by the service you receive rather than the gender you identify as. If you are unsure of what service to book in for, our helpful front of house team is available to assist you with your booking. Please give us a call.

[Blue Tit](#) [SALONS](#) [SERVICES](#) [PRICING](#) [NEWS](#) [ACADEMY](#) [CAREERS](#) [SHOP](#) [CONTACT](#)

[Contact us](#)

Please note, our salons only accept card payments and are unable to accept Amex.

Highlights

These services include a natural toner, require you book an accompanying haircut or blow-dry, and you will need a valid allergy test.

Please note that we use Hcolor or Agricolor for our toning services. Should clients request we use HNectar as a toner, a £20 surcharge will be applied.

	Stylist Level	1	2	3	4	5	
B2B Highlights	Colouring hair using a back-to-back foiling technique for maximum coverage.	220	237	254	285	300	3 hrs
Full Head Highlights	Colouring a full head of hair using foils.	163	176	194	227	239	2.5 hrs
B2B Half Head Highlights	Colouring the top half of hair using a back-to-back foiling technique for maximum coverage.	151	163	180	211	222	2.5 hrs
Half Head Highlights	Colouring the top half of hair using foils.	116	127	145	169	178	2 hrs
T-Section Highlights	Colouring the parting and hairline using foils.	88	100	115	134	141	1hr 45
Face Framing	Colouring the hairline to frame the face using foils.	111	124	138	159	168	1hr 45

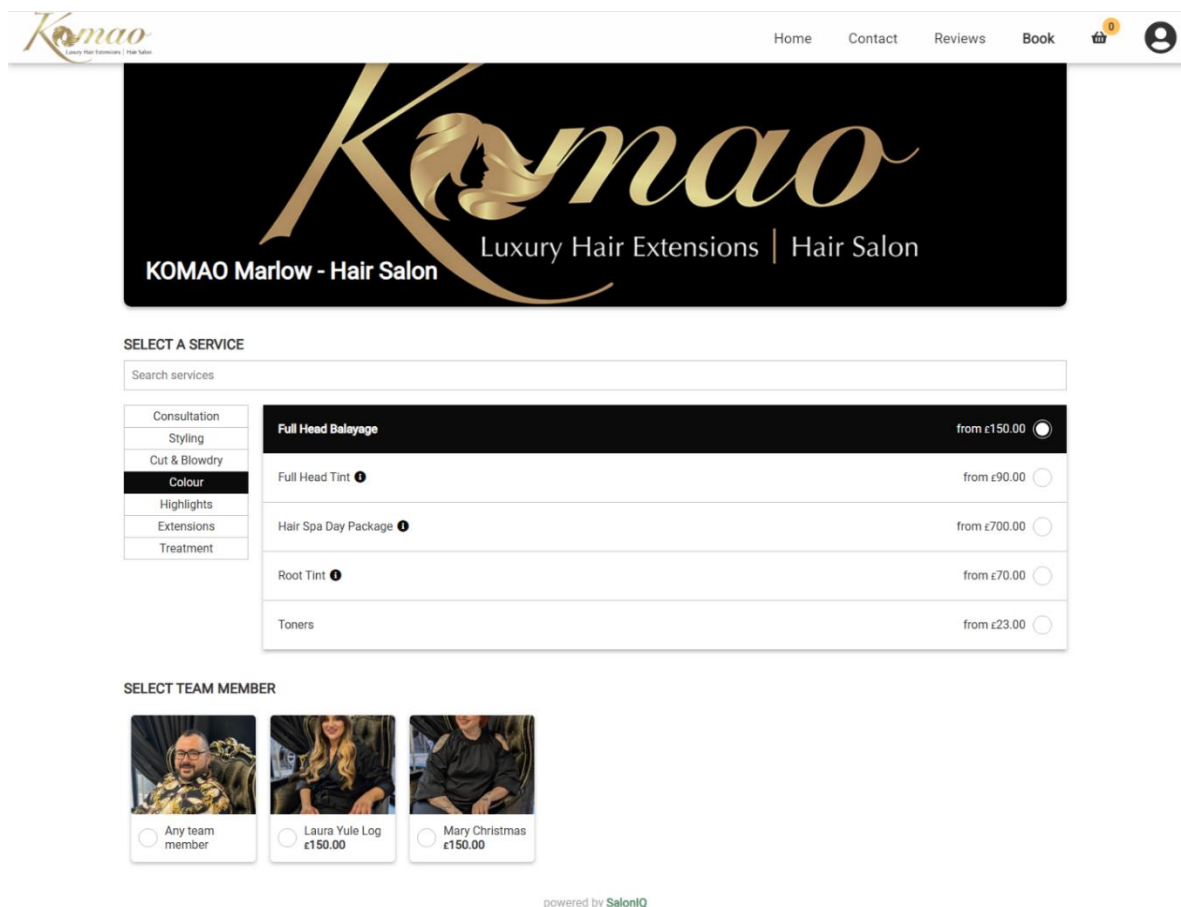
Рисунок 1.5 – Вебзастосунок Blue Tit

При цьому на відміну від попереднього вебзастосунку отримати детальнішу інформацію про перукарів у вебзастосунку Blue Tit неможливо. За кожною перукарнею можна переглянути вартість послуг та їх список, однак всі послуги розділені за категоріями, а за кожною категорією доступний список послуг та у табличній формі вартість кожної послуги за рівнем стиліста. При цьому отримати інформацію про рівень конкретного стиліста

неможливо. Тобто інформація щодо вартості є загальною, а її важко застосувати для практики. Таким чином можливо тільки отримати певну консультативну інформацію.

Вебзастосунок також дозволяє забронювати послуги перукарні. Вже під час бронювання можна обрати послугу та задати конкретного перукаря, а за ним відображається конкретна вартість. Відповідно отримання загальної інформації ускладнено, але процес бронювання послуг є доволі визначеним. У кінці також доступний вибір конкретних дат та часу бронювання послуг перукарні. У вебзастосунку доступні окремі сторінки з описом послуг, які надаються перукарнею, доступні також новини. Окрім того доступний магазин та набір моделей.

Вебзастосунок Komaо (рис. 1.6) містить інформацію про ті сервіси, які надаються перукарнею, про перелік вартості послуг, надає засоби бронювання [5].



The screenshot displays the Komaо website interface. At the top, there is a navigation bar with links for Home, Contact, Reviews, and Book, along with a shopping cart icon and a user profile icon. The main header features the Komaо logo and the text "Luxury Hair Extensions | Hair Salon". Below the header, the "SELECT A SERVICE" section includes a search bar and a list of services with their prices. The "SELECT TEAM MEMBER" section shows three team members with their names and prices.

SELECT A SERVICE	
Consultation	
Styling	
Cut & Blowdry	
Colour	
Highlights	
Extensions	
Treatment	

SELECT TEAM MEMBER	
	Any team member
	Laura Yule Log £150.00
	Mary Christmas £150.00

powered by **SaloniQ**

Рисунок 1.6 – Вебзастосунок Komaо

Під час бронювання послуг у перукарні можна обрати конкретного перукаря. Доступна також загальна сторінка з інформацією про всіх перукарів.

1.2 Визначення рівня створюваного вебзастосунку підтримки діяльності перукарні

До підсумкового порівняння (табл. 1.1) було віднесено вебзастосунки Barberking як такий, що представляє українську перукарню, Larry King та Blue Tit як такі, що надають найширший ряд функціональних можливостей.

Таблиця 1.1 – Порівняння можливостей аналогів розробки вебзастосунку підтримки діяльності перукарні

Показник	Barberking	Larry King	Blue Tit	Створювана програма
Виокремлена інформація про кожну конкретну послугу	Тільки загальна	—	+	+
Інформація про перукарів	Ім'я та фотографія	+	—	+
Відгуки про перукарів	—	—	—	+
Бронювання послуг у перукарні	+	+	+	+
Продаж супутніх товарів	—	+	+	+
Прогнозування кількості відвідувачів перукарні за день	—	—	—	+

Результати порівняння визначають, що основними відмінностями програми підтримки діяльності перукарні є:

- наявність можливості прогнозування кількості відвідувачів перукарні за день;
- наявність можливості створення відгуків про перукарів;
- можливість отримання розширеної інформації про перукарів та послуги, що не підтримується абсолютно всіма аналогами.

1.3 Висновки за розділом 1

У даному розділі розглянуто вебзастосунки для підтримки діяльності перукарні, включаючи Barberking, Barber Shop NYC, Larry King, Blue Tit, Комао. Виділено відмінності створюваної програми підтримки діяльності перукарні. У процесі виконання роботи необхідно забезпечити реалізацію завдань:

- виконати аналіз програмних застосунків підтримки діяльності перукарні;
- розробити метод прогнозування кількості відвідувачів перукарні за день;
- виконати проєктування вебзастосунку підтримки діяльності перукарні;
- реалізувати вебзастосунок підтримки діяльності перукарні;
- виконати експериментальне дослідження прогнозування кількості відвідувачів перукарні за день.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача прогнозування кількості відвідувачів перукарні за день

Кількість відвідувачів перукарні за день – це сумарна кількість клієнтів перукарні, які отримували послуги протягом всього робочого дня перукарні.

Основна суть задачі прогнозування кількості відвідувачів перукарні за день полягає в тому, щоб визначити кількість клієнтів перукарні, які отримають послуги перукарні протягом певного робочого дня у майбутньому. Такий день знаходиться у певному віддаленні від поточного. Відповідно має бути встановлена кількість днів, на яку відбувається прогнозування. Позначимо її як N .

При цьому наявна інформація про кількість відвідувачів перукарні за день стосовно деякої кількості попередніх днів. Позначимо цю кількість як M .

Тоді задача прогнозування кількості відвідувачів перукарні за день полягає в тому, щоб за відомими значеннями кількості відвідувачів перукарні за M попередніх днів, які розташовані поспіль, виконати прогнозування кількості відвідувачів перукарні на $(M + N)$ -ий день.

Відомий ряд спостережень, який дозволяє встановити кількість відвідувачів перукарні протягом певного періоду часу за кожний день цього періоду. Це дозволяє встановити послідовність спостережень, кожне спостереження в якій складається з $M + N$ значень за таку кількість послідовних днів. Відповідно з цієї послідовності виділяється M поспіль днів, а далі з наступних N днів виділяється тільки $(M + N)$ -ий, пропускаючи $N - 1$ попередній з номерами від $M + 1$ до $M + N - 1$.

Враховуючи, що доступна кількість спостережень за певний період часу, це дозволяє сформувати наступні окремі часові ряди.

Перший часовий ряд формує перше спостереження і складається з M перших значень в якості вхідних ознак та $(M + N)$ -ого значення кількості відвідувачів перукарні за день в якості вихідної ознаки.

Другий часовий ряд формує друге спостереження і складається зі значень кількості відвідувачів перукарні за дні з номерами від 2 до $M + 1$ в якості вхідних ознак та $(M + N + 1)$ -ого значення кількості відвідувачів перукарні за день в якості вихідної ознаки.

Третій часовий ряд формує третє спостереження і складається зі значень кількості відвідувачів перукарні за дні з номерами від 3 до $M + 2$ в якості вхідних ознак та $(M + N + 2)$ -ого значення кількості відвідувачів перукарні за день в якості вихідної ознаки.

Відповідно далі продовжується формування цих спостережень на основі відповідних часових рядів, зсуваючи значення на 1 день до того моменту, поки вихідна ознака не буде відповідати останньому доступному дню спостережень.

Таким чином, задача прогнозування кількості відвідувачів перукарні за день полягає в тому, щоб за наявним рядом спостережень, кожне з яких представлено часовим рядом за відомими значеннями кількості відвідувачів перукарні за M попередніх днів, які розташовані поспіль, та кількістю відвідувачів перукарні на $(M + N)$ -ий день, створити модель, яка здатна буде за цим наявним рядом відтворювати залежність для цих спостережень, а також виконувати прогнозування кількості відвідувачів перукарні за день з номером $M + N$ на основі послідовності значень за будь-які M попередніх днів. Ця задача є задачею прогнозування у сфері машинного навчання. Також вона відома як задача регресії.

2.2 Метод прогнозування кількості відвідувачів перукарні за день

Для розв'язання задачі прогнозування кількості відвідувачів перукарні за день у цьому підрозділі запропоновано відповідний метод прогнозування кількості відвідувачів перукарні за день. Цей метод в якості вхідних даних потребує наявності даних про певну кількість днів роботи конкретної перукарні стосовно кількості її відвідувачів. При цьому дані не можуть бути

замінені частково даними іншої перукарні. Усі представлені дані мають визначати кількість відвідувачів саме однієї конкретної перукарні. Усі значення мають представляти результати спостережень за перукарнею протягом послідовних днів. Використовуючи таким чином визначені вхідні дані, необхідно виконати послідовність наступних етапів.

На першому етапі методу прогнозування кількості відвідувачів перукарні за день необхідно опрацювати отриману вибірку даних для її застосування в якості навчальної вибірки далі. Початкова вибірка даних складається з послідовності значень кількості відвідувачів перукарні за день. Якщо одне зі значень у цій вибірці пропущене, то воно має бути відновлено шляхом використання інтерполяції. Це дозволяє, знаючи інші послідовні значення кількості відвідувачів перукарні за день, відновити проміжне пропущене значення. У підсумку має бути вибірка даних без пропущених значень. При цьому слід враховувати, що у вибірці стосовно вихідних днів або таких днів, коли перукарня не працює, має бути встановлено нульові значення. До таких днів мають відноситися всі святкові дні та вихідні, якщо в такі дні перукарня не працює. Значення кількості відвідувачів перукарні в такі дні має в результаті дорівнювати 0, а інтерполяція для них не виконуватися.

При цьому якщо початкові дані представлені записами про відвідування клієнтами перукарні, тобто кожен запис представляє собою певну дату (можливо дату і час) та номер клієнта (або іншим чином ідентифікованого клієнта), а також можливо значення додаткових характеристик, то має бути на основі цих даних сформовано початкову вибірку даних. Для цього спочатку мають бути усунуті будь-які наявні дублювання в даних. Після того, як це зроблено, має бути ідентифіковано найменшу дату в переліку спостережень та відповідно найбільшу. Далі має бути сформовано послідовність значень за кожну дату в проміжку між мінімальною та максимальною. Кожне значення за відповідну дату формується на основі обчислення кількості записів про відвідування перукарні клієнтами, що відбулись у таку дату.

Отримані таким чином значення мають бути розбиті на часові ряди, починаючи з початкових $(M + N)$ значень і продовжуючи далі зі зміщенням на 1 день. Відповідно K -ий часовий ряд формує K -те спостереження і складається зі значень кількості відвідувачів перукарні за дні з номерами від K до $M + K - 1$ в якості вхідних ознак та $(M + N + K - 1)$ -ого значення кількості відвідувачів перукарні за день в якості вихідної ознаки.

Усе це в підсумку дозволяє за результатами першого етапу сформувати навчальну вибірку даних стосовно кількості відвідувачів перукарні за день.

На другому етапі має бути створено модель для прогнозування кількості відвідувачів перукарні за день через забезпечення для цього ансамблю дерев ухвалення рішень, об'єднання яких має реалізовуватись на основі методу Random Forest [6]. При цьому має бути встановлено кількість таких дерев. Рекомендовано дерева створювати в повній мірі, не виконуючи обрізання, а кількість задавати рівною 600, що було визначено під час експериментів з представленою вибіркою даних.

На третьому етапі методу прогнозування кількості відвідувачів перукарні за день має бути виконано навчання створеної моделі за допомогою створеної вибірки даних. У подальшому вибірка даних стосовно кількості відвідувачів перукарні на день має розширюватися шляхом додавання спостережень до неї про наступні дні. У певні моменти може реалізовуватися перенавчання моделі. Такі моменти визначаються особою, що приймає рішення, і можуть встановлюватися періодично, а може бути визначено на основі врахування змін, які відбулись у перукарні протягом наявного періоду часу, поки перенавчання не реалізовувалось. Ці зміни можуть включати, наприклад, розширення колективу, збільшення сумарного графіку роботи або навпаки відповідні зменшення.

На четвертому етапі кожен день має визначатися кількість відвідувачів перукарні за наступний N -ий день відносно поточного дня. Для цього має використовуватися навчена модель у поточному стані. На вхід моделі мають подаватися значення за M попередніх днів відносно поточного. Прогнозування

має відбуватися в кінці дня, після того як відома кількість відвідувачів перукарні за поточний день. При цьому ця кількість відвідувачів перукарні має бути додана до сформованої вибірки даних з проведенням на основі нового отриманого значення першого етапу роботи методу прогнозування кількості відвідувачів перукарні за день для розширення вибірки.

2.3 Висновки за розділом 2

У розділі представлено як постановку задачі прогнозування кількості відвідувачів перукарні на день, так і метод її розв'язання. Наукова новизна роботи полягає в методі прогнозування кількості відвідувачів перукарні за день, що характеризується використанням часового ряду, ансамблю дерев рішень на основі методу Random Forest, що дозволяє визначати сумарну кількість відвідувачів за день з інтервалом у задану кількість днів.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Визначення сценаріїв роботи користувачів з вебзастосунком підтримки діяльності перукарні

Клієнт перукарні повинен мати можливість доступу до таких сценаріїв як основних:

- перегляд наявних перукарень у мережі;
- перегляд основної інформації про перукарню;
- перегляд інформації про перукаря;
- перегляд інформації про перукарську послугу;
- пошук перукаря;
- збереження запису перукаря;
- перегляд збережених записів перукарів;
- бронювання послуг перукарні;
- створення відгуку про перукаря;
- перегляд відгуків про перукаря;
- перегляд відгуків про перукарів, створених користувачем;
- перегляд бронювань послуг у перукарні, самостійно створених;
- авторизація;
- перегляд детальної інформації бронювання послуг у перукарні;
- перегляд користувацької інформації;
- перегляд перукарських засобів у продажі;
- пошук перукарських засобів у продажі;
- перегляд інформації про перукарський засіб;
- бронювання перукарського засобу для придбання;
- додавання фотографії з результатами перукарських послуг;
- перегляд бронювань перукарських засобів, самостійно створених;
- перегляд детальної інформації бронювання перукарських засобів

(рис. 3.1).

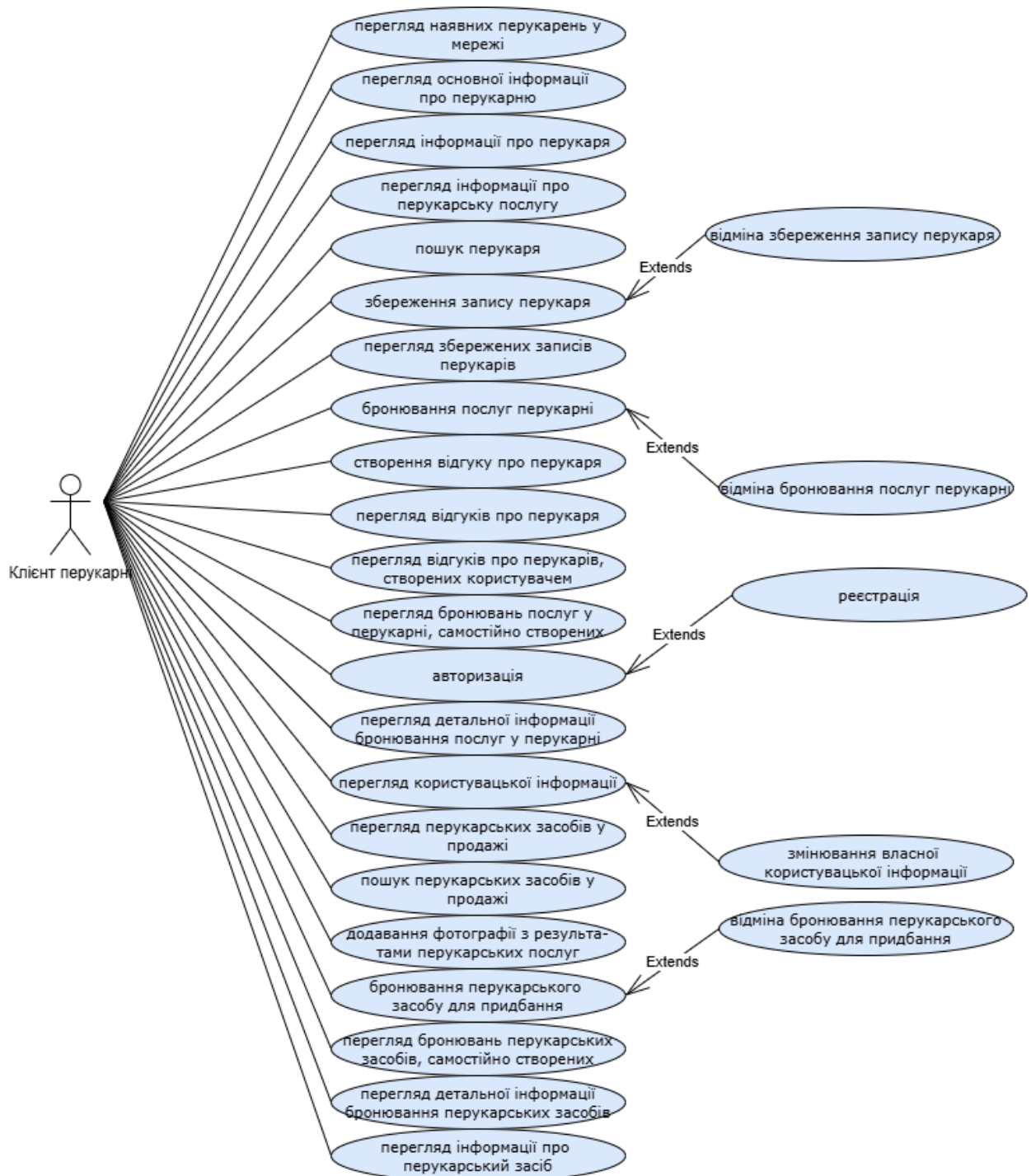


Рисунок 3.1 – Діаграма прецедентів для клієнта перукарні

Альтернативні сценарії для клієнта перукарні:

– змінювання власної користувацької інформації, коли вона має бути оновлена за своїм обліковим записом для відповідності поточним даним, що є альтернативним стандартному перегляду користувацької інформації;

– відміна бронювання послуг перукарні, що є альтернативою для бронювання послуг перукарні, коли бронювання вже є, але воно більше не потрібне клієнту;

– відміна збереження запису перукаря, що є альтернативою для збереження запису перукаря, коли збереження вже є, але воно більше не потрібне клієнту;

– реєстрація, що є альтернативою для авторизації, коли обліковий запис клієнта перукарні відсутній;

– відміна бронювання перукарського засобу для придбання, що є альтернативою для бронювання перукарського засобу для придбання, коли бронювання вже є, але воно більше не потрібне клієнту.

Менеджер перукарні повинен мати можливість доступу до таких сценаріїв як основних:

- перегляд наявних перукарень у мережі;
- перегляд основної інформації про перукарню;
- перегляд інформації про перукаря;
- перегляд інформації про перукарську послугу;
- пошук перукаря;
- бронювання послуг перукарні;
- перегляд відгуків про перукаря;
- перегляд відгуків про перукарів, створених користувачем;
- перегляд запланованих бронювань послуг перукарні за календарем;
- перегляд всіх бронювань послуг перукарні за станом;
- перегляд бронювань послуг у перукарні за користувачем;
- авторизація;
- прогнозування кількості відвідувачів перукарні за день;
- перегляд детальної інформації бронювання послуг у перукарні;
- перегляд користувацької інформації;
- пошук бронювань послуг перукарні;
- додавання перукарні до мережі;

- додавання перукарської послуги;
- додавання перукаря;
- визначення робочого графіку перукаря в перукарні;
- перегляд робочого графіку перукаря;
- перегляд перукарських засобів у продажі;
- пошук перукарських засобів у продажі;
- перегляд інформації про перукарський засіб;
- бронювання перукарського засобу для придбання;
- перегляд всіх бронювань перукарських засобів;
- перегляд бронювань перукарських засобів за користувачем;
- пошук бронювань перукарських засобів;
- перегляд детальної інформації бронювання перукарських засобів;
- додавання перукарського засобу до продажу;
- визначення наявної кількості перукарського засобу в перукарні;
- затвердження бронювання послуг перукарні;
- затвердження бронювання перукарського засобу для придбання.

Альтернативні сценарії для менеджера перукарні:

- змінювання інформації про перукаря, що є альтернативою для додавання перукаря, коли перукар вже доданий, а інформацію необхідно відкоригувати;
- змінювання інформації про перукарську послугу, що є альтернативою для додавання перукарської послуги, коли дані застаріли;
- змінювання інформації про перукарню, що є альтернативою для додавання перукарні до мережі, коли дані застаріли;
- змінювання власної користувацької інформації, що є альтернативою для перегляду користувацької інформації, коли вона має бути оновлена за своїм обліковим записом для відповідності поточним даним;
- додавання або вилучення доступного часового проміжку в роботі перукаря, що є альтернативою для визначення робочого графіку перукаря в перукарні, коли наявні слоти мають бути змінені (рис. 3.2).

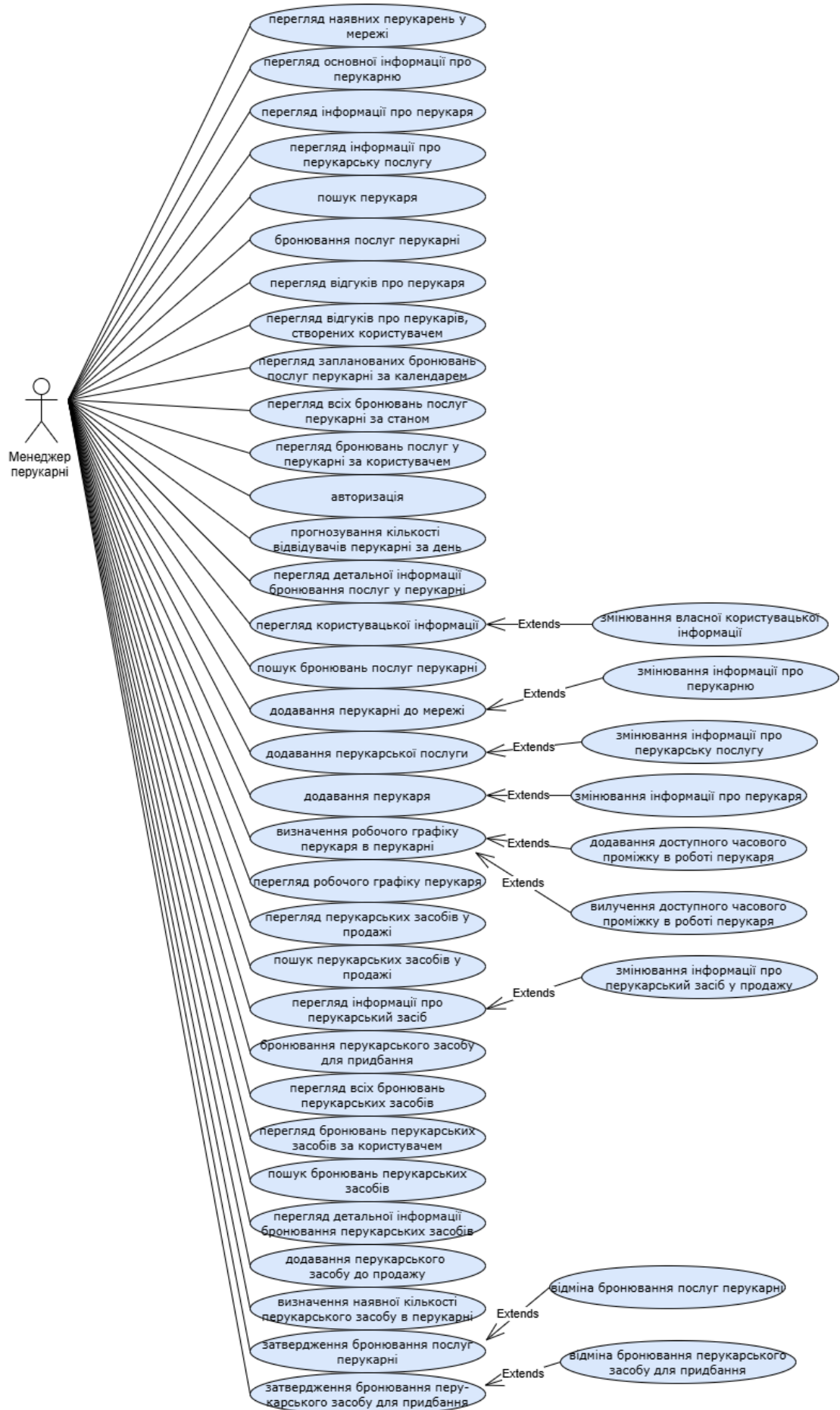


Рисунок 3.2 – Діаграма прецедентів для менеджера перукарні

Ще альтернативні сценарії:

- змінювання інформації про перукарський засіб у продажу, що є альтернативою для перегляду інформації про власний перукарський засіб, коли дані застаріли;
- відміна бронювання послуг перукарні, що є альтернативою для затвердження бронювання послуг перукарні, коли послуги не можуть бути надані цьому клієнту;
- відміна бронювання перукарського засобу для придбання, що є альтернативою для затвердження бронювання перукарського засобу для придбання, коли послуги не можуть бути надані цьому клієнту.

3.2 Вибір засобів розробки вебзастосунку підтримки діяльності перукарні

Вебзастосунок підтримки діяльності перукарні можна розробити на основі всього комплексу можливих засобів для веброзробки, однак зважаючи на популярність мов програмування для веброзробки, зокрема Python і PHP, вебфреймворки Django [7] і CodeIgniter [8] виокремлені для порівняння (табл. 3.1).

Вебфреймворк Django дещо переважає CodeIgniter, враховуючи, що Django надає потужні вбудовані засоби авторизації та аутентифікації, які зокрема включають систему дозволів для користувачів, зокрема це призводить до того, що всі користувачі можуть мати додаткову роль працівника, що передбачає контроль за доступом та надання ширших засобів роботи, при цьому стандартні засоби адміністративної панелі дозволяють керувати таким доступом. Окрім того було враховано, що Django дозволяє як результуючі створювати рішення, які мають складний або середній рівень складності, відповідно такі рішення більше придатні для нарощування функціональності зокрема, що може бути важливим у подальшому для цієї предметної області.

Таблиця 3.1 – Порівняння можливих рішень для реалізації вебзастосунку підтримки діяльності перукарні

	Django	CodeIgniter
Мова програмування в основі	Python	PHP
Тип рішення	Вебфреймворк	Вебфреймворк
Забезпечення авторизації з аутентифікацією	Аутентифікація з підключенням зовнішніх засобів, встановлення дозволів для користувачів	Базова аутентифікація, реєстрація
Підтримуваний рівень складності готового проєкту	Середедній або високий	Легкий або середній

3.3 Проєктування структури бази даних

База даних знаходиться в основі роботи вебзастосунку підтримки діяльності перукарні, адже ті функціональні сценарії, які було представлено, передбачають роботу з даними, одним з найбільш вдалих варіантів роботи з якими у великій кількості є реляційні бази даних, а однією з найбільш ефективних на практиці для роботи таких вебзастосунків є система MySQL, яку було обрано для підтримки.

Таблиця Useracc визначає запис користувача вебзастосунку підтримки діяльності перукарні з полями:

- id – ідентифікатор запису користувача вебзастосунку підтримки діяльності перукарні;
- userid – обліковий запис вебзастосунку Django, що використовується цим користувачем вебзастосунку підтримки діяльності перукарні (зв'язане

поле з таблицею User, що знаходиться в стандартній структурі бази даних в Django, зовнішнім ключем);

- fullname – ім'я користувача вебзастосунку підтримки діяльності перукарні;

- callphone – номер телефону користувача вебзастосунку підтримки діяльності перукарні;

- photo – фотографія користувача вебзастосунку підтримки діяльності перукарні.

Таблиця Hairdresser визначає перукаря з полями:

- id – ідентифікатор перукаря;

- workname – ім'я перукаря, за яким він відомий;

- description – текстова інформація про перукаря;

- avmark – середня оцінка, отримана перукарем;

- hairdresserfrom – дата, з якої ця людина працює перукарем;

- authorizedat – дата внесення даних цього перукаря в програму;

- photo – фотографія перукаря.

Таблиця HairdresserCategory визначає категорії перукарів з полями:

- id – ідентифікатор категорії конкретного перукаря;

- hairdresserid – ідентифікатор перукаря, для якого задано цю категорію (зв'язане з таблицею Hairdresser зовнішнім ключем);

- category – категорія, в якій здатен працювати перукар (виконання стрижок, колорист, стиліст).

Таблиця Diplomature визначає типи дипломів, сертифікатів, які можуть отримувати перукарі, з полями:

- id – ідентифікатор типу диплому, сертифікату, який може отримувати перукар;

- dtypename – назва типу диплому, сертифікату, який може отримувати перукар.

Таблиця HairdresserDiploma визначає дипломи, сертифікати, які отримали перукарі, з полями:

- id – ідентифікатор диплому, сертифікату перукаря;
- hairdresserid – ідентифікатор перукаря (зв'язане з таблицею Hairdresser зовнішнім ключем);
- diplomatypeid – ідентифікатор типу диплому, сертифікату, який було отримано перукарем (зв'язане з таблицею Diplomatype зовнішнім ключем);
- diplomanumber – текстовий номер (серія та номер) диплому, сертифікату, який було отримано перукарем;
- coursestart – дата, коли розпочалось навчання, за яким було перукарем отримано цей сертифікат, диплом;
- authorizedby – організація, якою було видано цей сертифікат, диплом перукарю;
- authorizedat – дата, коли було отримано цей сертифікат, диплом перукарем, тобто фактично завершення навчання;
- diplomatile – назва диплому, сертифікату, отриманого перукарем;
- description – опис диплому, сертифікату перукаря;
- photo – фотографія з зображенням диплому, сертифікату перукаря.

Таблиця Hairdresserwork визначає фотографії з результатами роботи перукаря з полями:

- id – ідентифікатор фотографії результатів роботи перукаря;
- hairdresserid – перукар, який виконав цю роботу (зв'язане з таблицею Hairdresser зовнішнім ключем);
- photo – фотографія, на якій показано результат роботи;
- workday – дата, коли цю роботу перукарем було виконано;
- authorizedat – дата і час занесення фотографії результатів роботи перукаря в програму;
- hairdresserreservationprocedureid – перукарська послуга з бронювання, за яким виконувались ті роботи, результати яких показані на фотографії: при цьому це зовнішній ключ до таблиці HairdresserReservationProcedure, але значення може бути і не заповнене, наприклад, для загальних демонстраційних фотографій, фотографій з минулого, тому в цій таблиці необхідно було

передбачити окрім цього поля ще додатково поля `hairstresserid` та `workday`, щоб встановити відповідну інформацію, якщо не можна цю інформацію встановити безпосередньо з даних бронювання.

Таблиця `HairprocedureCategory` визначає категорії перукарських послуг з полями:

- `id` – ідентифікатор категорії перукарських процедур;
- `haircategoryname` – назва категорії перукарських послуг.

Таблиця `Hairprocedure` визначає загальні перукарські послуги (тобто не пов'язані з конкретним перукарем) з полями:

- `id` – ідентифікатор перукарської послуги;
- `hrcategoryid` – категорія, яку представляє цю перукарську послугу (зв'язане з таблицею `HairprocedureCategory` зовнішнім ключем);
- `hrname` – назва перукарської послуги;
- `description` – опис цієї перукарської послуги;
- `mainphoto` – фотографія, що демонструє цю перукарську послугу;
- `duration` – тривалість виконання перукарської послуги: тривалість вирішено узагальнити і встановити для послуги загалом, а не за окремим перукарем.

Таблиця `HairstresserProcedure` визначає перукарські послуги, які реалізує конкретний перукар, з полями:

- `id` – ідентифікатор перукарської послуги у виконанні конкретного перукаря;
- `hairstresserid` – ідентифікатор перукаря, який може виконувати цю перукарську послугу (зв'язане з таблицею `Hairstresser` зовнішнім ключем);
- `hairprocedureid` – ідентифікатор перукарської послуги, яку здатен виконувати цей перукар (зв'язане з таблицею `Hairprocedure` зовнішнім ключем);
- `price` – вартість виконання цієї перукарської послуги цим перукарем.

Таблиця `Hairshop` визначає перукарні мережі з полями:

- `id` – ідентифікатор перукарні в мережі;

- `hsname` – назва перукарні мережі;
- `description` – текстова інформація про перукарню мережі;
- `city` – місто, де знаходиться перукарня мережі (програмно задано має бути певний перелік варіантів);
- `fulladdress` – адреса, за якою в цьому місті знаходиться ця перукарня мережі;
- `hsemail` – електронна адреса (пошта) перукарні мережі;
- `callphone` – номер телефону для дзвінків до перукарні мережі;
- `photo` – фотографія, що демонструє перукарню мережі.

Таблиця `HairdresserShop` визначає перукарні мережі, в яких працюють перукарі, з полями:

- `id` – ідентифікатор роботи перукаря в певній перукарні мережі;
- `hairdresserid` – ідентифікатор перукаря, який працює в цій перукарні мережі (зв'язане з таблицею `Hairdresser` зовнішнім ключем);
- `hairshopid` – ідентифікатор перукарні мережі, в якій працює цей перукар (зв'язане з таблицею `Hairshop` зовнішнім ключем);
- `active` – визначення активності цієї роботи цього перукаря в цій перукарні мережі, тобто чи надається все ще можливість бронювання послуг цього перукаря в цій перукарні мережі.

Необхідність таблиці `HairdresserShop` виходить з того, щоб дозволити підтримувати одночасну роботу перукарів у різних перукарнях мережі. Тобто в різні дні перукар може працювати в різних перукарнях. Наприклад, як стиліст, надаючи послуги клієнтам у зручніших для них місцях у різні дні.

Таблиця `HairdresserBook` визначає збереження користувачами вебзастосунку підтримки діяльності перукарні перукарів з полями:

- `id` – ідентифікатор збереження користувачем вебзастосунку підтримки діяльності перукарні перукаря;
- `hairdresserid` – ідентифікатор перукаря, якого збережено цим користувачем вебзастосунку підтримки діяльності перукарні (зв'язане з таблицею `Hairdresser` зовнішнім ключем);

– useraccid – ідентифікатор запису користувача вебзастосунку підтримки діяльності перукарні (зв’язане з таблицею Useracc зовнішнім ключем).

Таблиця HairdresserReservation визначає бронювання перукарських послуг користувачами вебзастосунку підтримки діяльності перукарні з полями:

– id – ідентифікатор бронювання перукарських послуг користувачами вебзастосунку підтримки діяльності перукарні;

– hairdressershopid – ідентифікатор перукаря в перукарні, послуги якого заброньовано користувачем вебзастосунку підтримки діяльності перукарні (зв’язане з таблицею HairdresserShop зовнішнім ключем);

– useraccid – ідентифікатор користувача вебзастосунку підтримки діяльності перукарні, який забронював ці перукарські послуги (зв’язане з таблицею Useracc зовнішнім ключем);

– reservedat – дата і час бронювання перукарських послуг користувачем вебзастосунку підтримки діяльності перукарні;

– procedurestart – дата і час, коли має розпочатися виконання перукарських послуг за цим бронюванням;

– procedureend – дата і час, коли має завершитися виконання перукарських послуг за цим бронюванням;

– comment – коментар клієнта стосовно перукарських послуг за бронюванням;

– reservestate – стан бронювання перукарських послуг;

– clientmark – оцінка клієнта перукарю стосовно наданих перукарських послуг після завершення бронювання;

– price – сума коштів, яку необхідно сплатити за виконання цих перукарських послуг.

Таблиця HairdresserReservationProcedure визначає перукарські послуги, які входять до складу бронювання, з полями:

– id – ідентифікатор перукарської послуги, яка входить до складу бронювання перукарських послуг;

– hairdresserreservationid – ідентифікатор бронювання перукарських послуг користувачем вебзастосунку підтримки діяльності перукарні (зв'язане з таблицею HairdresserReservation зовнішнім ключем);

– hairdresserprocedureid – ідентифікатор перукарської послуги у виконанні певного перукаря, яку було заброньовано (зв'язане з таблицею HairdresserProcedure зовнішнім ключем);

– price – сума коштів, яку необхідно сплатити за виконання саме цієї перукарської послуги у складі бронювання.

Таблиця HairdresserReservationProcedureWork визначає фотографії виконаних робіт за перукарськими послугами з бронювання з полями:

– id – ідентифікатор фотографії виконання перукарської послуги з бронювання;

– hairdresserreservationprocedureid – ідентифікатор перукарської послуги зі складу бронювання перукарських послуг, яку продемонстровано на фотографії (зв'язане з таблицею HairdresserReservationProcedure зовнішнім ключем);

– photo – фотографія, на якій продемонстровано результати виконання перукарської послуги зі складу бронювання перукарських послуг.

Таблиця HairdresserShopStat визначає відвідуваність перукарні з полями:

– id – ідентифікатор запису щодо відвідуваності перукарні;

– hairshopid – ідентифікатор перукарні, відвідуваність якої зафіксовано (зв'язане з таблицею Hairshop зовнішнім ключем);

– clients – кількість клієнтів, які скористались послугами цієї перукарні протягом цієї дати;

– prediction – визначення того, чи це було фактична кількість відвідувачів перукарні, чи це було прогнозоване значення;

– workdate – дата роботи перукарні.

Таблиця `HairdresserTimetable` визначає стандартний графік перукаря в перукарні з полями:

- `id` – ідентифікатор запису стандартного графіку роботи перукаря в перукарні;
- `hairdressershopid` – ідентифікатор запису про роботу перукаря в перукарні, стосовно якого визначено цей графік роботи (зв'язане з таблицею `HairdresserShop` зовнішнім ключем);
- `workday` – порядковий номер дня тижня, за яким визначено цей графік щодо роботи цього перукаря в цій перукарні;
- `daystart` – час початку робочого дня цього перукаря в цій перукарні в цей робочий день;
- `dayend` – час завершення робочого дня цього перукаря в цій перукарні в цей робочий день;
- `pausestart` – час початку перерви у цього перукаря в цій перукарні в цей робочий день;
- `pauseend` – час завершення перерви у цього перукаря в цій перукарні в цей робочий день.

Таблиця `HairdresserSlot` визначає доступні для бронювання часові проміжки в роботі перукаря в перукарні з полями:

- `id` – ідентифікатор доступного для бронювання часового проміжку в роботі перукаря в перукарні;
- `hairdressershopid` – ідентифікатор запису про роботу перукаря в перукарні, за яким визначається цей доступний для бронювання часовий проміжок (зв'язане з таблицею `HairdresserShop` зовнішнім ключем);
- `slotdate` – дата часового проміжку, доступного для бронювання, в роботі перукаря в перукарні;
- `timestart` – час початку часового проміжку, доступного для бронювання, в роботі перукаря в перукарні;
- `timeend` – час завершення часового проміжку, доступного для бронювання, в роботі перукаря в перукарні.

Таблиця `HairitemCategory` визначає категорії перукарських засобів з полями:

- `id` – ідентифікатор категорії перукарських засобів;
- `hcategoryname` – назва категорії перукарських засобів.

Таблиця `Hairitem` визначає перукарські засоби з полями:

- `id` – ідентифікатор перукарського засобу;
- `hairitemcategoryid` – ідентифікатор категорії перукарського засобу (зв'язане з таблицею `HairitemCategory` зовнішнім ключем);
- `itemname` – назва перукарського засобу;
- `description` – текстова інформація про перукарський засіб;
- `mainphoto` – фотографія перукарського засобу;
- `price` – вартість цього перукарського засобу;
- `volume` – обсяг цього перукарського засобу.

Таблиця `HairShopItem` визначає доступність перукарських засобів у перукарні з полями:

- `id` – ідентифікатор запису про доступність перукарських засобів у перукарні;
- `hairitemid` – ідентифікатор перукарського засобу, який доступний в цій перукарні для продажу (зв'язане з таблицею `Hairitem` зовнішнім ключем);
- `hairshopid` – ідентифікатор перукарні, в якій доступний цей перукарський засіб (зв'язане з таблицею `Hairshop` зовнішнім ключем);
- `quantity` – доступна кількість цього перукарського засобу в одиницях у цій перукарні для продажу.

Таблиця `HairshopReservation` визначає бронювання перукарських засобів у перукарні з полями:

- `id` – ідентифікатор бронювання перукарських засобів у перукарні користувачем вебзастосунку підтримки діяльності перукарні;
- `hairshopid` – ідентифікатор перукарні, в якій було заброньовано ці перукарські засоби (зв'язане з таблицею `Hairshop` зовнішнім ключем);

- useraccid – ідентифікатор запису користувача вебзастосунку підтримки діяльності перукарні, який виконав це бронювання перукарських засобів (зв'язане з таблицею Useracc зовнішнім ключем);
- reservedat – дата і час бронювання перукарських засобів користувачем вебзастосунку підтримки діяльності перукарні;
- comment – коментар до бронювання перукарських засобів;
- reservestate – стан бронювання перукарських засобів користувачем вебзастосунку підтримки діяльності перукарні;
- price – сумарний обсяг коштів, які необхідно сплатити за придбання перукарських засобів, визначених у бронюванні (рис. 3.3).

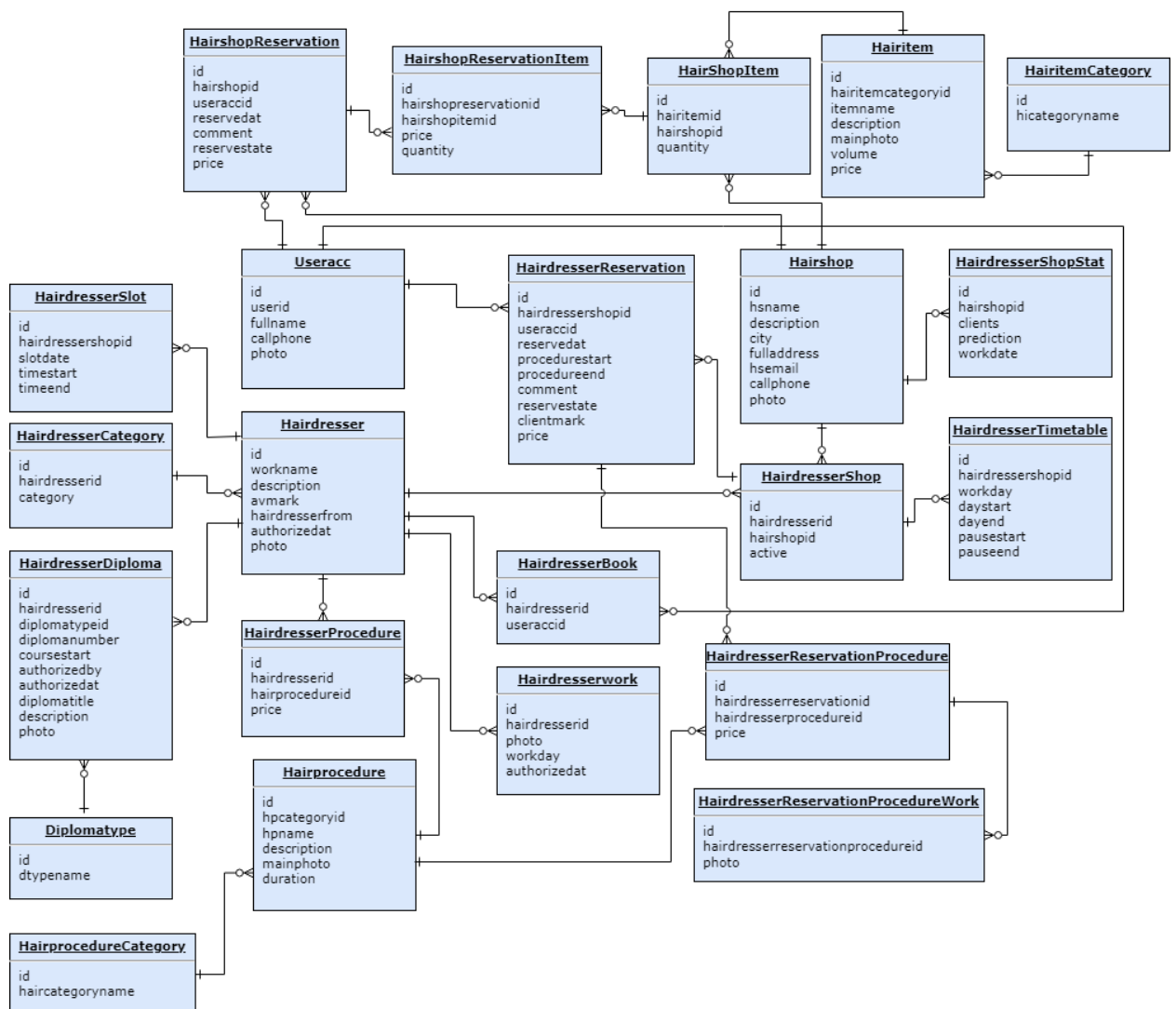


Рисунок 3.3 – Схема бази даних підтримки діяльності перукарні

Таблиця `HairshopReservationItem` визначає перукарські засоби з бронювання перукарських засобів у перукарні з полями:

- `id` – ідентифікатор перукарського засобу з бронювання перукарських засобів у перукарні;
- `hairshopreservationid` – ідентифікатор бронювання перукарських засобів у перукарні користувачем вебзастосунку підтримки діяльності перукарні, до якого відноситься цей перукарський засіб (зв'язане з таблицею `HairshopReservation` зовнішнім ключем);
- `hairshopitem` – ідентифікатор перукарського засобу, який входить до цього бронювання перукарських засобів у перукарні (зв'язане з таблицею `HairShopItem` зовнішнім ключем);
- `price` – вартість одиниці перукарського засобу в бронюванні;
- `quantity` – кількість одиниць перукарського засобу, які було заброньовано.

3.4 Висновки за розділом 3

Розділ 3 містить сценарії, які мають бути реалізовані для клієнта та менеджера перукарні, вибір Python разом з фреймворком Django для розробки за результатами порівняння, визначену структуру бази даних, що включає всі необхідні сутності, пов'язані як з перукарськими послугами, так і з перукарськими засобами, які доступні для продажу, на основі системи керування MySQL.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Загальний порядок роботи програми підтримки діяльності перукарні

Вебзастосунок підтримки діяльності перукарні при базовому зверненні починає для користувача роботу з відображення перукарень мережі, дозволяючи користувачу обрати подальший шлях роботи з меню (рис. 4.1).



Рисунок 4.1 – Схема функціонування програми підтримки діяльності перукарні

4.2 Реалізовані програмні одиниці у складі вебзастосунку підтримки діяльності перукарні

Модель прогнозування кількості відвідувачів перукарні на день створено через клас `RandomForestRegressor` з бібліотеки `scikit-learn`, викликаючи з нього метод `fit` для навчання за підготовленими даними за методом прогнозування кількості відвідувачів перукарні за день та метод `predict` для самого прогнозування за новими даними після навчання.

Перегляд наявних перукарень у мережі підтримується в основі функцією представлення `hairshopsview` з параметром `cityname`, який може бути пропущений, але якщо заданий, то визначає назву міста. Обчислені дані повертаються в шаблон вебсторінки `hairshops.html`. При створенні шаблонів застосовано шаблон [9].

Перегляд основної інформації про перукарню підтримується в основі функцією представлення `hairshopview` з параметром `hairshopid`, який задає ідентифікатор перукарні. Обчислені дані повертаються в шаблон вебсторінки `hairshop.html`, включаючи дані самої перукарні за ключем `hairshop`, перукарів, які працюють в перукарні, за ключем `hairdressers`, перукарські послуги, які надаються перукарнею, за ключем `hairprocedures`.

Перегляд інформації про перукаря підтримується в основі функцією представлення `hairdresserview` з параметром `hairdresserid`, який задає ідентифікатор перукаря. Обчислені дані повертаються в шаблон вебсторінки `hairdresser.html`, включаючи дані перукаря за ключем `hairdresser`, перелік категорій, за якими працює перукар, за ключем `categories`, перелік перукарських послуг, які надаються цим перукарем, за ключем `procedures`, перелік сертифікатів, дипломів, отриманих перукарем, за ключем `diplomas`, перелік фотографій робіт перукаря за ключем `works`, перелік робіт за конкретними перукарськими послугами цього перукаря за ключем `resworks`, перелік перукарень, в яких працює перукар, за ключем `hairshops`, перелік відгуків про перукаря за ключем `hdmarks`, доступні часові проміжки в роботі у

перукаря за ключем `slots`, стандартний розклад роботи перукаря за ключем `timetable`.

Перегляд бронювань послуг у перукарні, самостійно створених, підтримується в основі функцією представлення `ownhairdresserreservations` з параметром `reservestate`, який може бути пропущений, але якщо заданий, то задає стан бронювання послуг у перукарні. Обчислені дані повертаються в шаблон вебсторінки `reservations.html`.

Перегляд відгуків про перукаря підтримується в основі функцією представлення `hairdressermarksview` з параметром `hairdresserid`, який задає ідентифікатор перукаря. Обчислені дані повертаються в шаблон вебсторінки `markhairdresser.html` з даними перукаря за ключем `hairdresser`, з категоріями роботи перукаря за ключем `categories` та з відгуками за ключем `hdmarks`.

Перегляд відгуків про перукарів, створених користувачем, підтримується в основі функцією представлення `ownhairdressermarksview` з параметром `user`, який задає логін користувача. Якщо логін пропущений, то з авторизації беруться ці дані. Обчислені дані повертаються в шаблон вебсторінки `ownmarks.html`.

Перегляд збережених записів перукарів підтримується в основі функцією представлення `bookhairdressersview`. Обчислені дані повертаються в шаблон вебсторінки `book.html`.

Перегляд бронювань перукарських засобів за користувачем підтримується в основі функцією представлення `userhairshopreservationsview` з параметром `user`, який задає логін користувача, `reservestate`, який задає потрібний стан бронювання перукарських засобів. Обчислені дані повертаються в шаблон вебсторінки `hsuserreservations.html`.

Пошук перукаря підтримується в основі функцією представлення `searchhairdressersview`, в якій розбираючи GET-запит, виділяються ім'я перукаря (`name`), мінімальна оцінка перукаря (`markmin`), максимальна оцінка перукаря (`markmax`), категорія (`categ`), необхідний тип диплому або сертифікату (`diploma`), перукарня, де має працювати перукар (`shop`),

мінімальний необхідний досвід перукаря (exper). Обчислені дані повертаються в шаблон вебсторінки search.html.

Перегляд інформації про перукарську послугу підтримується в основі функцією представлення hairprocedureview з параметрами hairprocedureid, який задає ідентифікатор перукарської послуги, hairdresserid, який задає ідентифікатор перукаря. Обчислені дані повертаються в шаблон вебсторінки hairproced.html.

Перегляд детальної інформації бронювання послуг у перукарні підтримується в основі функцією представлення hairdresserreservationview з параметром hairdresserreservationid, який задає номер бронювання послуг у перукарні. Обчислені дані повертаються в шаблон вебсторінки reservation.html.

Перегляд бронювань послуг у перукарні за користувачем підтримується в основі функцією представлення userhairdresserreservationsview з параметром user, який задає логін користувача, reservestate, який задає стан бронювання послуг у перукарні. Обчислені дані повертаються в шаблон вебсторінки ureservations.html.

Перегляд всіх бронювань послуг перукарні за станом підтримується в основі функцією представлення allhairdresserreservationsview з параметром reservestate, який задає стан бронювання послуг у перукарні. Обчислені дані повертаються в шаблон вебсторінки allreservations.html.

Перегляд запланованих бронювань послуг перукарні за календарем підтримується в основі функцією представлення allcalhairdresserreservationsview з параметром resstart, який задає початкову дату, resend, який задає кінцеву дату, reservestate, який задає стан бронювання послуг перукарні. Обчислені дані повертаються в шаблон вебсторінки calreservations.html.

Перегляд бронювань перукарських засобів, самотійно створених, підтримується в основі функцією представлення ownhairshopreservationsview з

параметром `reservestate`, який задає стан бронювань перукарських засобів. Обчислені дані повертаються в шаблон вебсторінки `hsireervations.html`.

Перегляд всіх бронювань перукарських засобів підтримується в основі функцією представлення `allhairshopreservationsview` з параметром `reservestate`, який задає стан бронювань перукарських засобів. Обчислені дані повертаються в шаблон вебсторінки `allitreservations.html`.

Перегляд детальної інформації бронювання перукарських засобів підтримується в основі функцією представлення `hairshopreservationview` з параметром `hairshopreservationid`, який задає ідентифікатор бронювання перукарських засобів. Обчислені дані повертаються в шаблон вебсторінки `hsirereservation.html`.

Пошук перукарських засобів у продажі підтримується в основі функцією представлення `searchhairitemsview`, в якій розбираючи GET-запит, виділяються назва перукарського засобу (`name`), мінімальна вартість перукарського засобу (`minprice`), максимальна вартість перукарського засобу (`maxprice`), категорія (`categ`), необхідний тип диплому або сертифікату (`diploma`), максимальний обсяг перукарського засобу (`maxvol`), мінімальний обсяг перукарського засобу (`minvol`), перукарня, де має бути доступний перукарський засіб (`shop`). Обчислені дані повертаються в шаблон вебсторінки `searchhairitems.html`.

Перегляд інформації про перукарський засіб підтримується в основі функцією представлення `hairitemview` з параметром `hairitemid`, який задає ідентифікатор перукарського засобу. Обчислені дані повертаються в шаблон вебсторінки `hairitem.html`.

Збереження запису перукаря підтримується в основі функцією представлення `dobookhairdresserview` з параметром `hairdresserid`, який задає ідентифікатор перукаря.

Додавання перукарні до мережі підтримується в основі функцією представлення `hairshopaddview` за допомогою класу форми `HairshopForm`. Змінювання інформації про перукарню підтримується в основі функцією

представлення `hairshopeditview` за допомогою того ж класу форми `HairshopForm`.

Додавання перукарської послуги підтримується в основі функцією представлення `hairprocedureaddview` за допомогою класу форми `HairprocedureForm`. Змінювання інформації про перукарську послугу підтримується в основі функцією представлення `hairprocedureeditview` за допомогою того ж класу форми `HairprocedureForm`.

Додавання перукаря підтримується в основі функцією представлення `hairstylistaddview` за допомогою класу форми `HairstylistForm`. Змінювання інформації про перукарську послугу підтримується в основі функцією представлення `hairstylisteditview` за допомогою того ж класу форми `HairstylistForm`.

Бронювання послуг перукарні підтримується в основі функцією представлення `dohairstylistreservationview` за допомогою класів форм `HairstylistReservationForm` та `HairstylistReservationProcedureForm`.

Створення відгуку про перукаря підтримується в основі функцією представлення `markhairstylistreservationview` з параметром `hairstylistreservationid`, який задає ідентифікатор бронювання перукарських послуг, на основі якого створюється відгук.

Перегляд користувацької інформації підтримується в основі функцією представлення `useraccview` з параметром `user`, який задає логін користувача, а якщо пропущений, то беруться дані авторизації.

Відміна бронювання перукарського засобу для придбання підтримується в основі функцією представлення `cancelhairshopreservationview` з параметром `hairshopreservationid`, який задає ідентифікатор бронювання перукарського засобу.

Затвердження бронювання послуг перукарні підтримується в основі функцією представлення `submithairstylistreservationview` з параметром `hairstylistreservationid`, який задає ідентифікатор бронювання перукарських послуг.

Затвердження бронювання перукарського засобу для придбання підтримується в основі функцією представлення `submithairshopreservationview` з параметром `hairshopreservationid`, який задає ідентифікатор бронювання перукарського засобу.

Так само організовані представлення за всіма сценаріями, які було передбачено.

Відміна бронювання послуг перукарні підтримується в основі функцією представлення `cancelhairdresserreservationview` з параметром `hairdresserreservationid`, який задає ідентифікатор бронювання перукарських послуг.

Клас `Useracc` визначає модель запису користувача вебзастосунку підтримки діяльності перукарні.

Поле `user` класу представляє обліковий запис вебзастосунку Django, що використовується цим користувачем вебзастосунку підтримки діяльності перукарні, реалізовано класом `ForeignKey` на основі класу `User`.

Поле `fullname` класу представляє ім'я користувача вебзастосунку підтримки діяльності перукарні, реалізоване класом `CharField`.

Поле `callphone` класу представляє номер телефону користувача вебзастосунку підтримки діяльності перукарні, реалізований класом `CharField`.

Поле `photo` класу представляє фотографію користувача вебзастосунку підтримки діяльності перукарні, реалізовану класом `ImageField`.

Поле `hairdressers` класу представляє перукарів, яких було збережено цим користувачем вебзастосунку підтримки діяльності перукарні, реалізованих класом `ManyToManyField` на основі класу `Hairdresser`.

Клас `Hairdresser` визначає модель перукаря.

Поле `workname` класу представляє ім'я перукаря, за яким він відомий, реалізоване класом `CharField`.

Поле `description` класу представляє текстову інформація про перукаря, реалізовану класом `TextField`.

Поле `avmark` класу представляє середню оцінку, отримана перукарем, реалізовану класом `DecimalField`.

Поле `hairstresserfrom` класу представляє дату, з якої ця людина працює перукарем, реалізовану класом `DateField`.

Поле `authorizedat` класу представляє дату і час внесення даних цього перукаря в програму, реалізовану класом `DateTimeField`.

Поле `photo` класу представляє фотографію перукаря, реалізовану класом `ImageField`.

Клас `HairdresserCategory` визначає категорії перукарів.

Поле `hairstresser` класу представляє перукаря, для якого задано цю категорію, реалізовано класом `ForeignKey` на основі класу `Hairdresser`.

Поле `category` класу представляє категорію, в якій здатен працювати перукар, реалізовану класом `CharField`.

Клас `Diplomatype` визначає модель типів дипломів, сертифікатів, які можуть отримувати перукарі. Поле `dtypename` класу представляє назву типу диплому, сертифікату, який може отримувати перукар, реалізовану класом `CharField`.

Клас `HairdresserDiploma` визначає модель дипломів, сертифікатів, які отримали перукарі.

Поле `hairstresser` класу представляє перукаря, реалізовано класом `ForeignKey` на основі класу `Hairdresser`.

Поле `diplomatype` класу представляє тип диплому, сертифікату, який було отримано перукарем, реалізовано класом `ForeignKey` на основі класу `Diplomatype`.

Поле `diplomanumber` класу представляє текстовий номер (серія та номер) диплому, сертифікату, який було отримано перукарем, реалізоване класом `CharField`.

Поле `coursestart` класу представляє дату, коли розпочалось навчання, за яким було перукарем отримано цей сертифікат, диплом, реалізовану класом `DateField`.

Поле `authorizedby` класу представляє організацію, якою було видано цей сертифікат, диплом перукарю, реалізовану класом `CharField`.

Поле `authorizedat` класу представляє дату, коли було отримано цей сертифікат, диплом перукарем, тобто фактично завершення навчання, реалізовану класом `DateTimeField`.

Поле `diplomatitle` класу представляє назву диплому, сертифікату, отриманого перукарем, реалізовану класом `CharField`.

Поле `description` класу представляє опис диплому, сертифікату перукаря, реалізований класом `TextField`.

Поле `photo` класу представляє фотографію з зображенням диплому, сертифікату перукаря, реалізовану класом `ImageField`.

Клас `Hairdresserwork` визначає модель фотографії з результатами роботи перукаря.

Поле `hairdresser` класу представляє перукаря, який виконав цю роботу, реалізованого класом `ForeignKey` на основі класу `Hairdresser`.

Поле `photo` класу представляє фотографію, на якій показано результат роботи, реалізовану класом `ImageField`.

Поле `workday` класу представляє дату, коли цю роботу перукарем було виконано, реалізовану класом `DateTimeField`.

Поле `authorizedat` класу представляє дату і час занесення фотографії результатів роботи перукаря в програму, реалізовані класом `DateTimeField`.

Поле `hairdresserreservationprocedureid` класу представляє перукарську послугу з бронювання, за яким виконувались ті роботи, результати яких показані на фотографії, реалізовану класом `ForeignKey` на основі класу `HairdresserReservationProcedure`.

Клас `HairprocedureCategory` визначає модель категорії перукарських послуг. Поле `haircategoryname` класу представляє назву категорії перукарських послуг, реалізовану класом `CharField`.

Клас `Hairprocedure` визначає модель загальних перукарських послуг.

Поле `hpcategory` класу представляє категорію, до якої належить ця перукарська послуга, реалізована класом `ForeignKey` на основі класу `HairprocedureCategory`.

Поле `hpname` класу представляє назву перукарської послуги, реалізовану класом `CharField`.

Поле `description` класу представляє опис цієї перукарської послуги, реалізований класом `TextField`.

Поле `mainphoto` класу представляє фотографію, що демонструє цю перукарську послугу, реалізовану класом `ImageField`.

Поле `duration` класу представляє тривалість виконання перукарської послуги, реалізовану класом `SmallIntegerField`.

Клас `HairdresserProcedure` визначає модель перукарських послуг, які реалізує конкретний перукар.

Поле `hairdresser` класу представляє перукаря, який може виконувати цю перукарську послугу, реалізований класом `ForeignKey` на основі класу `Hairdresser`.

Поле `hairprocedure` класу представляє перукарську послугу, яку здатен виконувати цей перукар, реалізовану класом `ForeignKey` на основі класу `Hairprocedure`.

Поле `price` класу представляє вартість виконання цієї перукарської послуги цим перукарем, реалізовану класом `DecimalField`.

Клас `Hairshop` визначає модель перукарні мережі.

Поле `hsname` класу представляє назву перукарні мережі, реалізовану класом `CharField`.

Поле `description` класу представляє текстову інформацію про перукарню мережі, реалізовану класом `TextField`.

Поле `city` класу представляє місто, де знаходиться перукарня мережі, реалізовану класом `CharField`.

Поле `fulladdress` класу представляє адресу, за якою в цьому місті знаходиться ця перукарня мережі, реалізовану класом `CharField`.

Поле `hsemail` класу представляє електронну адресу перукарні мережі, реалізовану класом `EmailField`.

Поле `callphone` класу представляє номер телефону для дзвінків до перукарні мережі, реалізовану класом `CharField`.

Поле `photo` класу представляє фотографію, що демонструє перукарню мережі, реалізовану класом `ImageField`.

Клас `HairdresserShop` визначає модель перукарень мережі, в яких працюють перукарі.

Поле `hairdresser` класу представляє перукаря, який працює в цій перукарні мережі, реалізованого класом `ForeignKey` на основі класу `Hairdresser`.

Поле `hairshop` класу представляє перукарню мережі, в якій працює цей перукар, реалізовану класом `ForeignKey` на основі класу `Hairshop`.

Поле `active` класу представляє визначення активності цієї роботи цього перукаря в цій перукарні мережі, реалізовану класом `BooleanField`.

Клас `HairdresserReservation` визначає модель бронювання перукарських послуг користувачами вебзастосунку підтримки діяльності перукарні.

Поле `hairdressershop` класу представляє перукаря в перукарні, послуги якого заброньовано користувачем вебзастосунку підтримки діяльності перукарні, реалізованого класом `ForeignKey` на основі класу `HairdresserShop`.

Поле `useracc` класу представляє користувача вебзастосунку підтримки діяльності перукарні, який забронював ці перукарські послуги, реалізованого класом `ForeignKey` на основі класу `Useracc`.

Поле `reservedat` класу представляє дату і час бронювання перукарських послуг користувачем вебзастосунку підтримки діяльності перукарні, реалізовану класом `DateTimeField`.

Поле `procedurestart` класу представляє дату і час, коли має розпочатися виконання перукарських послуг за цим бронюванням, реалізовану класом `DateTimeField`.

Поле `procedureend` класу представляє дату і час, коли має завершитися виконання перукарських послуг за цим бронюванням, реалізовану класом `DateTimeField`.

Поле `comment` класу представляє коментар клієнта стосовно перукарських послуг за бронюванням, реалізований класом `CharField`.

Поле `reservestate` класу представляє стан бронювання перукарських послуг, реалізований класом `PositiveSmallIntegerField`.

Поле `clientmark` класу представляє оцінку клієнта перукарю стосовно наданих перукарських послуг після завершення бронювання, реалізовану класом `PositiveSmallIntegerField`.

Поле `price` класу представляє суму коштів, яку необхідно сплатити за виконання цих перукарських послуг, реалізовану класом `DecimalField`.

Клас `HairdresserReservationProcedure` визначає модель перукарських послуг, які входять до складу бронювання.

Поле `hairdresserreservation` класу представляє бронювання перукарських послуг користувачем вебзастосунку підтримки діяльності перукарні, реалізоване класом `ForeignKey` на основі класу `HairdresserReservation`.

Поле `hairdresserprocedure` класу представляє перукарську послугу у виконанні певного перукаря, яку було заброньовано, реалізовану класом `ForeignKey` на основі класу `HairdresserProcedure`.

Поле `price` класу представляє суму коштів, яку необхідно сплатити за виконання саме цієї перукарської послуги у складі бронювання, реалізовану класом `DecimalField`.

Клас `HairdresserReservationProcedureWork` визначає модель фотографії виконаних робіт за перукарськими послугами з бронювання.

Поле `hairdresserreservationprocedure` класу представляє перукарську послугу зі складу бронювання перукарських послуг, яку продемонстровано на фотографії, реалізовану класом `ForeignKey` на основі класу `HairdresserReservationProcedure`.

Поле `photo` класу представляє фотографію, на якій продемонстровано результати виконання перукарської послуги зі складу бронювання перукарських послуг, реалізовану класом `ImageField`.

Клас `HairdresserShopStat` визначає модель відвідуваності перукарні.

Поле `hairshop` класу представляє перукарню, відвідуваність якої зафіксовано, реалізовану класом `ForeignKey` на основі класу `Hairshop`.

Поле `clients` класу представляє кількість клієнтів, які скористались послугами цієї перукарні протягом цієї дати, реалізовану класом `PositiveIntegerField`.

Поле `prediction` класу представляє визначення того, чи це було фактична кількість відвідувачів перукарні, чи це було прогнозоване значення, реалізоване класом `BooleanField`.

Поле `workdate` класу представляє дату роботи перукарні, реалізовану класом `DateField`.

Клас `HairdresserTimetable` визначає модель стандартного графіку перукаря в перукарні.

Поле `hairdressershop` класу представляє запис про роботу перукаря в перукарні, стосовно якого визначено цей графік роботи, реалізований класом `ForeignKey` на основі класу `HairdresserShop`.

Поле `workday` класу представляє порядковий номер дня тижня, за яким визначено цей графік щодо роботи цього перукаря в цій перукарні, реалізований класом `PositiveSmallIntegerField`.

Поле `daystart` класу представляє час початку робочого дня цього перукаря в цій перукарні в цей робочий день, реалізований класом `TimeField`.

Поле `dayend` класу представляє час завершення робочого дня цього перукаря в цій перукарні в цей робочий день, реалізований класом `TimeField`.

Поле `pausestart` класу представляє час початку перерви у цього перукаря в цій перукарні в цей робочий день, реалізований класом `TimeField`.

Поле `pauseend` класу представляє час завершення перерви у цього перукаря в цій перукарні в цей робочий день, реалізований класом `TimeField`.

Клас `HairdresserSlot` визначає модель доступних для бронювань часових проміжків у роботі перукаря в перукарні.

Поле `hairdressershop` класу представляє запис про роботу перукаря в перукарні, за яким визначається цей доступний для бронювання часовий проміжок, реалізований класом `ForeignKey` на основі класу `HairdresserShop`.

Поле `slotdate` класу представляє дату часового проміжку, доступного для бронювання, в роботі перукаря в перукарні, реалізовану класом `DateField`.

Поле `timestart` класу представляє час початку часового проміжку, доступного для бронювання, в роботі перукаря в перукарні, реалізований класом `TimeField`.

Поле `timeend` класу представляє час завершення часового проміжку, доступного для бронювання, в роботі перукаря в перукарні, реалізований класом `TimeField`.

Клас `HairitemCategory` визначає модель категорії перукарських засобів. Поле `hicategoryname` класу представляє назву категорії перукарських засобів, реалізовану класом `CharField`.

Клас `Hairitem` визначає модель перукарських засобів.

Поле `hairitemcategory` класу представляє ідентифікатор категорії перукарського засобу, реалізований класом `ForeignKey` на основі класу `HairitemCategory`.

Поле `itemname` класу представляє назву перукарського засобу, реалізовану класом `CharField`.

Поле `description` класу представляє текстову інформацію про перукарський засіб, реалізовану класом `TextField`.

Поле `mainphoto` класу представляє фотографію перукарського засобу, реалізовану класом `ImageField`.

Поле `price` класу представляє вартість цього перукарського засобу, реалізований класом `DecimalField`.

Поле `volume` класу представляє обсяг цього перукарського засобу, реалізований класом `PositiveSmallIntegerField`.

Клас `HairShopItem` визначає модель доступності перукарських засобів у перукарні.

Поле `hairitemid` класу представляє перукарський засіб, який доступний в цій перукарні для продажу, реалізований класом `ForeignKey` на основі класу `Hairitem`.

Поле `hairshop` класу представляє перукарню, в якій доступний цей перукарський засіб, реалізовано класом `ForeignKey` на основі класу `Hairshop`.

Поле `quantity` класу представляє доступну кількість цього перукарського засобу в одиницях у цій перукарні для продажу, реалізовану класом `PositiveIntegerField`.

Клас `HairshopReservation` визначає модель бронювання перукарських засобів у перукарні.

Поле `hairshop` класу представляє перукарню, в якій було заброньовано ці перукарські засоби, реалізовану класом `ForeignKey` на основі класу `Hairshop`.

Поле `useracc` класу представляє запис користувача вебзастосунку підтримки діяльності перукарні, який виконав це бронювання перукарських засобів, реалізовану класом `ForeignKey` на основі класу `Useracc`.

Поле `reservedat` класу представляє дату і час бронювання перукарських засобів користувачем вебзастосунку підтримки діяльності перукарні, реалізовану класом `DateTimeField`.

Поле `comment` класу представляє коментар до бронювання перукарських засобів, реалізований класом `CharField`.

Поле `reservestate` класу представляє стан бронювання перукарських засобів користувачем вебзастосунку підтримки діяльності перукарні, реалізований класом `PositiveSmallIntegerField`.

Поле `price` класу представляє сумарний обсяг коштів, які необхідно сплатити за придбання перукарських засобів, визначених у бронюванні, реалізований класом `DecimalField`.

Клас `HairshopReservationItem` визначає модель перукарських засобів з бронювання перукарських засобів у перукарні.

Поле `hairshopreservation` класу представляє бронювання перукарських засобів у перукарні користувачем вебзастосунку підтримки діяльності перукарні, до якого відноситься цей перукарський засіб, реалізований класом `ForeignKey` на основі класу `HairshopReservation`.

Поле `hairshopitem` класу представляє перукарський засіб, який входить до цього бронювання перукарських засобів у перукарні, реалізований класом `ForeignKey` на основі класу `HairShopItem`.

Поле `price` класу представляє вартість одиниці перукарського засобу в бронюванні, реалізовану класом `DecimalField`.

Поле `quantity` класу представляє кількість одиниць перукарського засобу, які було заброньовано, реалізовану класом `SmallIntegerField`.

4.3 Висновки за розділом 4

Розділ містить опис реалізованих програмних одиниць у складі вебзастосунку підтримки діяльності перукарні стосовно реалізації методу прогнозування кількості відвідувачів перукарні за день та сценаріїв роботи клієнта та менеджера перукарні.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Основне призначення програми

Програма призначена для отримання основної необхідної інформації про ті послуги, які надаються перукарнею, бронювання послуг, бронювання придбання перукарських засобів, планування діяльності перукарні.

5.2 Виконання програми

Для клієнта перукарні одним з основних елементів роботи з програмою є сторінка перукарні, на яку він зазвичай потрапляє зі сторінки з переліком перукарень, доступної з однойменного розділу меню (рис. 5.1).

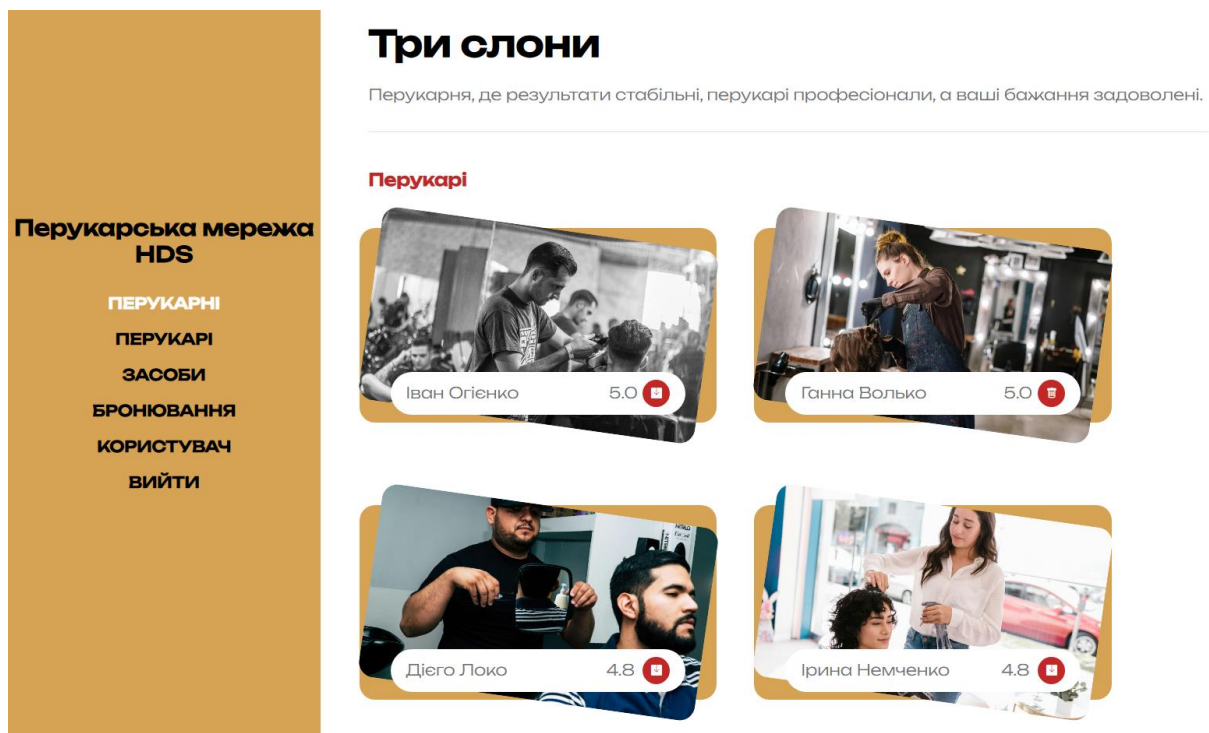


Рисунок 5.1 – Сторінка перукарні

Ця сторінка зверху містить фоторафію перукарні, її адресу, номер телефону, а вже нижче додаткові дані, які включають зокрема опис

(розташований під назвою), перелік перукарів та перелік перукарських послуг, доступних у цій перукарні. Для цього необхідно прогортати сторінку нижче.

Перелік перукарів у будь-якому варіанті містить кнопки для збереження перукаря до власного переліку, щоб потім відобразити збережених зі сторінки, яка відкривається за розділом перукарів та надає можливість перемикатися між пошуком та збереженими перукарями.

Зі сторінки перукарні, наприклад, можна обрати потрібну послугу та одразу перейти до бронювання її (рис. 5.2). Під час бронювання також можна задати перукарню, перукаря, необхідний початок обслуговування як дату, а вже за датою тоді з випадаючого переліку обрати необхідний час початку обслуговування. При цьому час завершення обслуговування в перукарні визначається автоматично на основі обраних перукарських послуг та заданого для них часу виконання.

Рисунок 5.2 – Сторінка бронювання перукарських послуг

Далі потрібно обрати категорію перукарських послуг, конкретну послугу, а за необхідності визначити декілька, потрібно натиснути на кнопку додати, а після цього задати додаткові. Вартість розраховується на основі

обраних послуг перукаря. Можна також додати коментар до бронювання перукарських послуг (рис. 5.3).

Рисунок 5.3 – Нижня частина сторінки бронювання перукарських послуг

Також клієнт перукарні може працювати з перукарськими засобами, виконуючи їх перегляд у наявності за категоріями, пошук, перемикаючись між ними за допомогою кнопок. З розділу меню бронювань жоступна можливість працювати з власними бронюваннями перукарських послуг та перукарських засобів. Це ті бронювання, які були створені клієнтом. Для менеджера перукарні цей розділ містить більше засобів, включаючи засоби пошуку бронювань обох типів, перегляду бронювань за станом, за календарем.

Розділ користувача дозволяє керувати власними даними акаунту, а також переміщуватись до відгуків клієнта, а для менеджера перукарні також переходити до бронювань перукарських послуг та перукарських засобів за обраним користувачем.

Для менеджера перукарні доступні додаткові засоби роботи. Частина з них може бути викликана з тих самих розділів, включаючи перукарні, перукарів, для яких тоді надаються можливості переходу до додавання нових перукарень, перукарів, змінювання даних існуючих.

Але менеджер перукарні також має додатковий розділ, який доступний з меню як планування. Це розділ, який дозволяє переглядати фактичну відвідуваність обраної перукарні з випадającego переліку за обраним проміжком дат, а також дозволяє переглянути прогноз, якщо в цей проміжок дат входять прогнозовані значення (рис. 5.4).

Прогнозування кількості відвідувачів перукарні на день реалізовувалось на основі використання скрипта, який запускався раз на добу. Скрипт програмно запускався опівночі. Хоча можна даний час змінити на завершення робочого дня, наприклад.

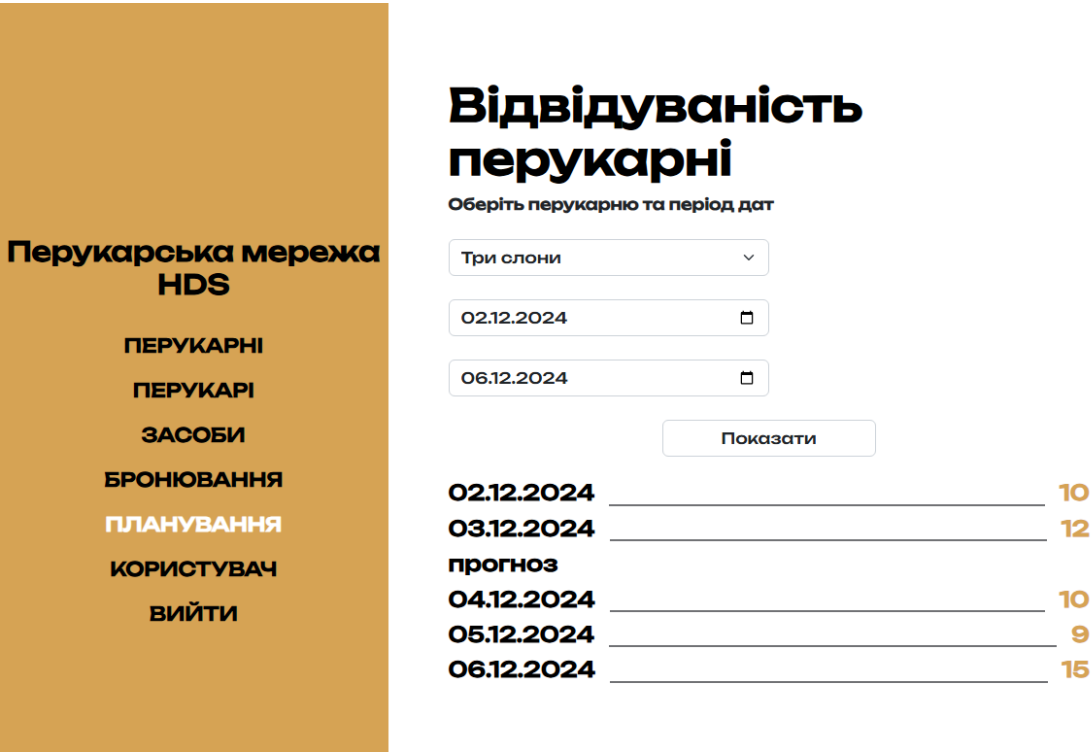


Рисунок 5.4 – Сторінка планування діяльності перукарні

У підсумку кожен користувач має велику кількість засобів роботи, які згруповані у невелику кількість розділів, що майже ідентична і для клієнта перукарні, і для менеджера перукарні.

5.3 Експериментальне дослідження прогнозування кількості відвідувачів перукарні за день

Вибірка [10] складається з даних про відвідування перукарні, включаючи 13427 записів про відвідування перукарні 1007 клієнтами. Дані були опрацьовані та проаналізовані.

Спершу було проаналізовано, на скільки наявні дані відповідають реальним ситуаціям. Для цього було за кожним клієнтом проаналізовано час, який минає до наступного його відвідування перукарні. Тоді було визначено середній час для кожного клієнта, а за цими даними було визначено перелік з 10 найактивніших клієнтів (рис. 5.5). Отримані результати відповідають загалом очікуваним. Зокрема помітно, що тільки 2 клієнти відвідують перукарню в середньому кожен день, всі інші мають більший проміжок.

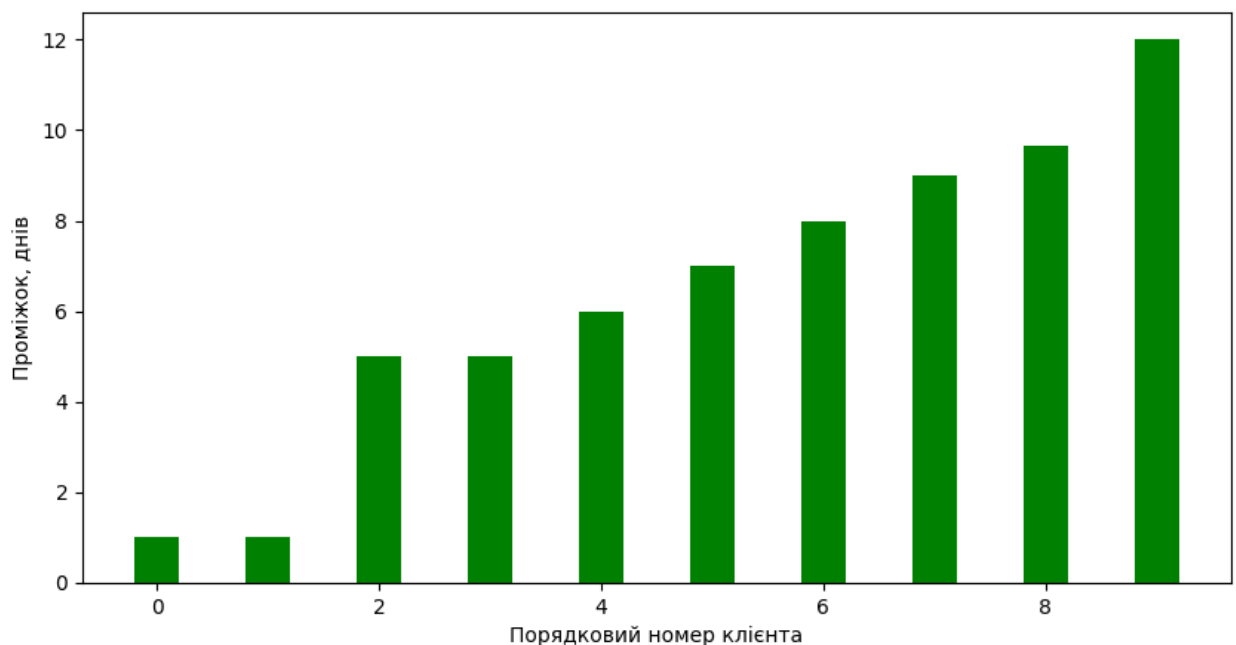


Рисунок 5.5 – Діаграма, що відображає клієнтів, які найчастіше всередньому відвідують перукарню

Далі було виконано розподіл за датами з визначенням кількості записів, що стосуються різних клієнтів перукарні на відповідну дату, таким чином було

сформовано часові ряди. Отримані значення продемонстрували, що у вихідні дні перукарня не працює, що також стосується і святкових днів (рис. 5.6).

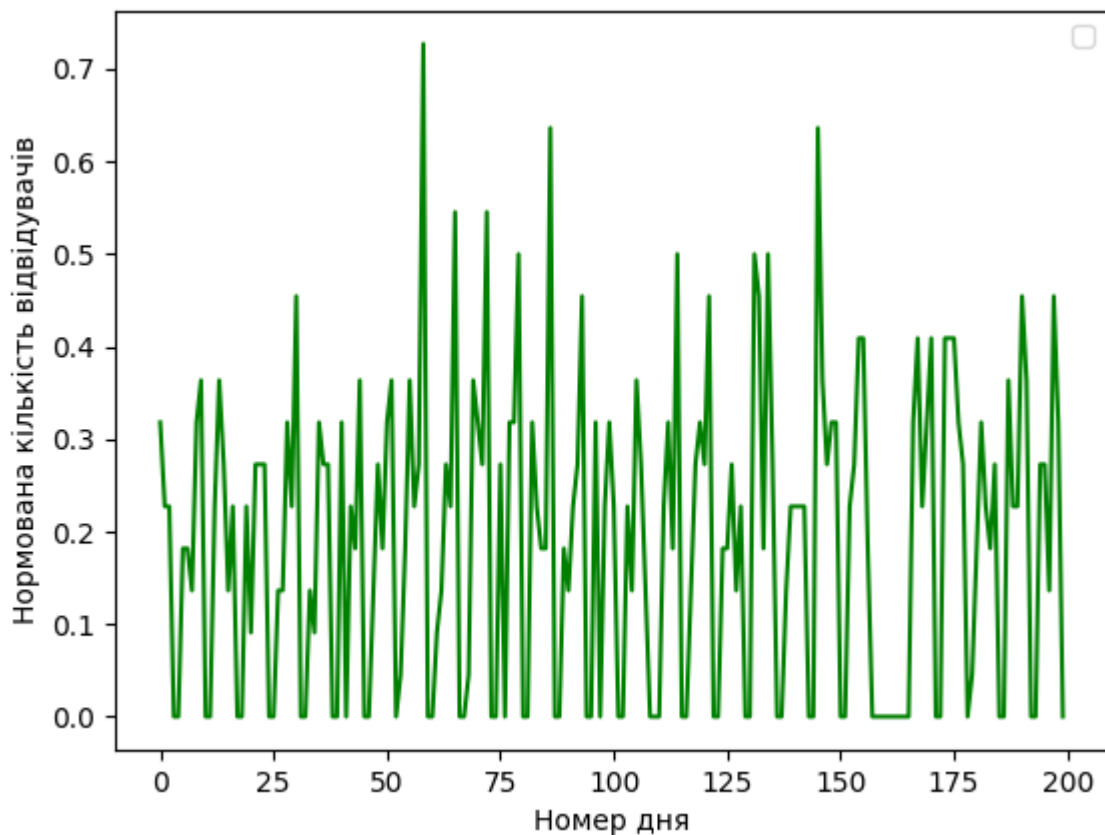


Рисунок 5.6 – Графік з нормованою кількістю відвідувачів перукарні протягом перших днів, що відносяться до навчальної вибірки

Та сама ж тенденція стосується і тестових даних, до яких було віднесено 20 % всіх даних. Відповідно 80 % перших днів було віднесено до навчальної частини вибірки, а 20 % – до тестової (рис. 5.7). Для формування графіків було використано тільки початкову частину навчальної та тестової вибірок (перші дні).

Для проведення самого експериментального дослідження на основі даних, сформованих у відповідності з запропонованим методом, було реалізовано навчання та тестування за допомогою таких методів:

– метод опорних векторів;

- запропонований метод прогнозування кількості відвідувачів перукарні за день на основі методу Random Forest;
- метод AutoRegressive Integrated Moving Average (ARIMA).

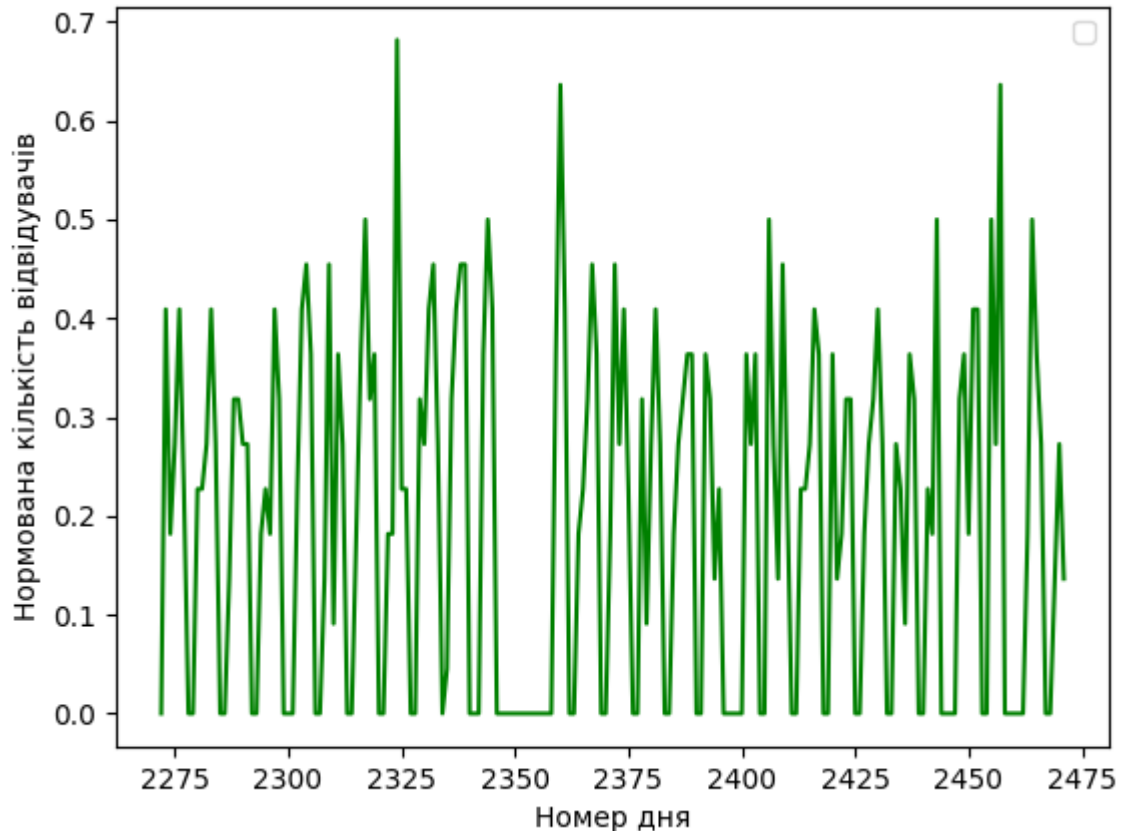


Рисунок 5.7 – Графік з нормованою кількістю відвідувачів перукарні протягом перших днів, що відносяться до тестової вибірки

Для оцінювання отриманих результатів кількості відвідувачів перукарні протягом дня використовувались:

- показник Mean Absolute Error (MAE);
- показник Mean Squared Error (MSE);
- показник Root Mean Squared Error (RMSE).

Далі було спочатку досліджено вплив тривалості вхідного часового ряду на результати прогнозування кількості відвідувачів перукарні на 1 день вперед. Спочатку було порівняно подання 31 значення (табл. 5.1) та 14 значень (табл. 5.2) на вхід.

Таблиця 5.1 – Результати прогнозування кількості відвідувачів перукарні на 1 день вперед за 31 попередніми днями

Метод прогнозування	MAE	RMSE	MSE
Запропонований метод прогнозування кількості відвідувачів перукарні за день	0,09174179	0,11620617	0,01350387
Метод опорних векторів	0,10189119	0,12784052	0,0163432
ARIMA	0,11359356	0,14538335	0,02113632

Таблиця 5.2 – Результати прогнозування кількості відвідувачів перукарні на 1 день вперед за 14 попередніми днями

Метод прогнозування	MAE	RMSE	MSE
Запропонований метод прогнозування кількості відвідувачів перукарні за день	0,09176622	0,1157393	0,01339559
Метод опорних векторів	0,09507379	0,12042564	0,01450234
ARIMA	0,11268256	0,14429026	0,02081968

Далі вхідний ряд був збільшений до 60 значень (табл. 5.3). Помітно, що для запропонованого методу за всіма 3 варіантами похибки (MAE, MSE,

RMSE) було отримано менші значення при використанні вхідного ряду з 60 значень.

Таблиця 5.3 – Результати прогнозування кількості відвідувачів перукарні на 1 день вперед за 60 попередніми днями

Метод прогнозування	MAE	RMSE	MSE
Запропонований метод прогнозування кількості відвідувачів перукарні за день	0,09031194	0,11429624	0,01306363
Метод опорних векторів	0,09968378	0,12702983	0,01613658
ARIMA	0,11271393	0,14418937	0,02079057

Далі було збільшено проміжок прогнозування, виконуючи прогнозування на 30 днів вперед (табл. 5.4).

Порівнюючи результати, помітно, що запропонований метод прогнозування кількості відвідувачів перукарні за день дозволив зменшити на 33,54 % значення MAE порівняно з методом опорних векторів, а RMSE – на 31,55 %, MSE – на 53,14 %.

У свою чергу порівняно з результатами ARIMA запропонований метод прогнозування кількості відвідувачів перукарні за день зменшив значення MAE на 8,1 %, RMSE – на 12,41 %, MSE – на 23,29 %.

Тож результати вказують, що запропонований метод прогнозування кількості відвідувачів перукарні за день дійсно дозволив покращити результати прогнозування.

Таблиця 5.4 – Результати прогнозування кількості відвідувачів перукарні на 30 днів вперед за 60 попередніми днями

Метод прогнозування	MAE	RMSE	MSE
Запропонований метод прогнозування кількості відвідувачів перукарні за день	0,07655473	0,09437362	0,00890638
Метод опорних векторів	0,11519007	0,13786621	0,01900709
ARIMA	0,08330045	0,10774955	0,01160997

5.4 Висновки за розділом 5

Розроблене програмне забезпечення підтримки діяльності перукарні перевірено під час використання з представленням основних з них за інтерфейсами, виконано експериментальне дослідження прогнозування кількості відвідувачів перукарні за день, що визначило вибір тривалості горизонту прогнозування, обсягу вхідних даних, а також порівняння з іншими подібними методами прогнозування, підтвердивши в підсумку зменшення похибки прогнозування кількості відвідувачів перукарні за день на основі показників MAE, RMSE, MSE.

ВИСНОВКИ

За підсумками виконання роботи розроблено програмне забезпечення підтримки діяльності перукарні для підвищення інформованості рішень щодо планування цієї діяльності.

За результатами аналізу програмних застосунків підтримки діяльності перукарні розглянуто вебзастосунки Barberking, Barber Shop NYC, Larry King, Blue Tit, Комао. Виділено відмінності програми підтримки діяльності перукарні, що створена в роботі, включаючи наявність можливості прогнозування кількості відвідувачів перукарні за день, наявність можливості створення відгуків про перукарів, можливість отримання розширеної інформації про перукарів та послуги.

Розроблене програмне забезпечення здатне підтримувати роботу перукарні з клієнтами, зокрема отримання основної необхідної інформації про ті послуги, які надаються перукарнею, бронювання послуг, планування діяльності перукарні. Для програми розроблено сценарії для клієнта та менеджера перукарні, які було реалізовано та описано. Виконано вибір Python разом з фреймворком Django для розробки за результатами порівняння. Визначено структуру бази даних, що включає всі необхідні сутності, пов'язані як з перукарськими послугами, так і з перукарськими засобами, які доступні для продажу, на основі системи керування MySQL.

Наукова новизна роботи полягає в методі прогнозування кількості відвідувачів перукарні за день, що характеризується використанням часового ряду, ансамблю дерев рішень на основі методу Random Forest, що дозволяє визначати сумарну кількість відвідувачів за день з інтервалом у задану кількість днів. До основних результатів роботи також відносяться результати експериментального дослідження, під час якого не тільки порівняно різні моделі, до яких належить ARIMA, метод опорних векторів, запропонований метод, але також і різна довжина часового інтервалу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Топ Барбершоп Київ [Електрон. ресурс] : Рейтинг, Відгуки, Хороші Ціни на Чоловічі Послуги. – Режим доступу : <https://kyiv.bking.com.ua/>.
2. Best Barber Shop NYC. Men's Grooming Midtown Manhattan [Electronic resource]. – Access mode : <https://www.barbershop.nyc/>.
3. Larry King Hair [Electronic resource]. – Access mode : <https://larrykinghair.com/>.
4. Hairdressers in London. Award-Winning Salons. Blue Tit London [Electronic resource]. – Access mode : <https://bluetitlondon.com/>.
5. Komao – Luxury Hair Extensions & Hair Boutique [Electronic resource]. – Access mode : <https://komao-marlow.com/>.
6. Donges, N. Random Forest [Electronic resource] : A Complete Guide for Machine Learning / N. Donges. – Access mode : <https://builtin.com/data-science/random-forest-algorithm>.
7. Django documentation contents – Django 5.1.3 documentation [Electronic resource]. – Access mode : <https://django.readthedocs.io/en/stable/contents.html>.
8. CodeIgniter4 User Guide – CodeIgniter 4.5.5 documentation [Electronic resource]. – Access mode : https://www.codeigniter.com/user_guide/index.html.
9. Free Template 585 Barber Shop [Electronic resource]. – Access mode : <https://templatemo.com/tm-585-barber-shop>.
10. Barber Shop Frequency Data [Electronic resource]. – Access mode : <https://www.kaggle.com/datasets/fredericobaker/barber-shop-frequency-data>.

ДОДАТОК А
Технічне завдання

Вступ

Перукарські послуги є специфічною галуззю, оскільки перукарень на ринку зазвичай представлено достатньо багато, але отримати потрібну інформацію, обрати такого перукаря, якому одразу з більшим ступенем можна довіритись, достатньо складно. Кожний має свій власний досвід, при цьому поділитися ним у вебзастосунках перукарень зазвичай ніхто не може, оскільки функція створення відгуків або подібна функція відсутня. З іншого боку інформація, яка загалом присутня у вебзастосунку, зазвичай є дуже обмеженою. Є певна статична інформація, а користувачу надається можливість забронювати певні перукарські послуги, що не дуже сприяє зануреності в процес взаємодії з вебзастосунком користувача. Відповідно з іншого боку засоби планування діяльності перукарні обмежені тільки контролем та керуванням тих бронювань, які здійснили клієнти. Застосування ж засобів машинного навчання при накопиченні потрібної інформації у вебзастосунку підтримки діяльності перукарні може бути достатньо корисним для прийняття рішень щодо відпусток працівників перукарні, розширення або зменшення штату, зокрема тимчасово. Усе це потребує планування, а планування потребує розуміння того, яка кількість клієнтів звернеться до перукарні через певний час.

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 планування діяльності перукарні», що обумовило отримання завдання на роботу.

А.2 Призначення розробки

Програма призначена для отримання основної необхідної інформації про ті послуги, які надаються перукарнею, бронювання послуг, бронювання придбання перукарських засобів, планування діяльності перукарні.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

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

- перегляд наявних перукарень у мережі;
- перегляд основної інформації про перукарню;
- перегляд інформації про перукаря;
- перегляд інформації про перукарську послугу;
- пошук перукаря;
- збереження запису перукаря;
- перегляд збережених записів перукарів;
- бронювання послуг перукарні;
- створення відгуку про перукаря;
- перегляд відгуків про перукаря;
- відміна бронювання послуг перукарні;
- затвердження бронювання послуг перукарні;
- перегляд відгуків про перукарів, створених користувачем;
- перегляд запланованих бронювань послуг перукарні за календарем;
- перегляд всіх бронювань послуг перукарні за станом;
- перегляд бронювань послуг у перукарні за користувачем;
- авторизація та реєстрація за потреби;
- прогнозування кількості відвідувачів перукарні за день;
- змінювання інформації про перукаря;

- змінювання інформації про перукарську послугу;
- змінювання інформації про перукарню;
- перегляд детальної інформації бронювання послуг у перукарні;
- перегляд користувацької інформації;
- змінювання власної користувацької інформації;
- пошук бронювань послуг перукарні;
- додавання перукарні до мережі;
- додавання перукарської послуги;
- додавання перукаря;
- відміна бронювання перукарського засобу для придбання;
- затвердження бронювання перукарського засобу для придбання;
- визначення робочого графіку перукаря в перукарні;
- додавання або вилучення доступного часового проміжку в роботі перукаря;
- перегляд робочого графіку перукаря;
- перегляд перукарських засобів у продажі;
- пошук перукарських засобів у продажі;
- перегляд інформації про перукарський засіб;
- бронювання перукарського засобу для придбання;
- перегляд всіх бронювань перукарських засобів;
- перегляд бронювань перукарських засобів за користувачем;
- пошук бронювань перукарських засобів;
- перегляд детальної інформації бронювання перукарських засобів;
- додавання перукарського засобу до продажу;
- змінювання інформації про перукарський засіб у продажу;
- визначення наявної кількості перукарського засобу в перукарні.

A.3.2 Умови експлуатації

Забезпечення експлуатації програмного забезпечення підтримки

діяльності перукарні 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.

А.3.3 Вимоги до складу та параметрів технічних засобів

Враховуючи, що програмне забезпечення підтримки діяльності перукарні 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.

А.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 django.shortcuts import render
from django.core.paginator import Paginator
from .models import Useracc
from .models import Hairdresser
from .models import HairdresserCategory
from .models import Diplomatype
from .models import HairdresserDiploma
from .models import Hairdresserwork
from .models import HairprocedureCategory
from .models import Hairprocedure
from .models import HairdresserProcedure
from .models import Hairshop
from .models import HairdresserShop
from .models import HairdresserReservation
from .models import HairdresserReservationProcedure
from .models import HairdresserReservationProcedureWork
from .models import HairdresserShopStat
from .models import HairdresserTimetable
from .models import HairdresserSlot
from .models import HairitemCategory
from .models import Hairitem
from .models import HairShopItem
from .models import HairshopReservation
from .models import HairshopReservationItem

```

```

#перегляд наявних перукарень у мережі
def hairshopsview(request, cityname = None):
    if cityname:
        hairshops = Hairshop.objects.filter(city = cityname)
    else:
        hairshops = Hairshop.objects.all()
        hairshops = hairshops.order_by("hsname")
        hairshopspag = Paginator(hairshops, 10)
        hairshopspart = request.GET.get('page')
        hairshops = hairshopspag.get_page(hairshopspart)
        return render(request, "hds/hairshops.html", {"hairshops":
hairshops })

```

```

#перегляд основної інформації про перукарню
def hairshopview(request, hairshopid):
    try:
        hairshop = Hairshop.objects.get(id = hairshopid)
    except Hairshop.DoesNotExist:
        hairshop = None
    if not hairshop:

```

```

        return render(request, "hds/hairshop.html", {"message":
"Такої перукарні немає"})
        shop_hairdressers = HairdresserShop.objects.filter(hairshop
= hairshop)
        shop_hairdressers =
shop_hairdressers.values_list("hairdresser__id")
        hairdressers = Hairdresser.objects.filter(id__in =
shop_hairdressers)
        hairdresserpag = Paginator(hairdressers, 10)
        hairdresserpart = request.GET.get('pagedressers')
        hairdressers = hairdresserpag.get_page(hairdresserpart)
        for hairdresser in hairdressers:
            hairdresser.categories =
HairdresserCategory.objects.filter(hairdresser = hairdresser)
            hairdressers_procedures =
HairdresserProcedure.objects.filter(hairdresser__in = hairdressers)
            hairdressers_procedures =
hairdressers_procedures.values_list("hairprocedure__id")
            hairprocedures =
Hairprocedure.objects.select_related("hpcategory").all()
            hairprocedures = hairprocedures.filter(id__in =
hairdressers_procedures)
            hairprocpag = Paginator(hairdressers_procedures, 10)
            hairprocpart = request.GET.get('pageproc')
            hairdressers_procedures =
hairdresserpart.get_page(hairprocpart)
            if request.user.is_authenticated:
                hds_user_auth = request.user
                hds_user =
Useracc.objects.prefetch_related("hairdressers").all()
                hds_user = hds_user.get(user = hds_user_auth)
                for hairdresser in hairdressers:
                    if hairdresser in hds_user.hairdressers:
                        hairdresser.save = True
                    else:
                        hairdresser.save = False
        return render(request, "hds/hairshop.html", {"hairshop":
hairshop,
                "hairdressers": hairdressers,
                "hairprocedures": hairprocedures })

#перегляд інформації про перукаря
def hairdresserview(request, hairdresserid):
    try:
        hairdresser= Hairdresser.objects.get(id = hairdresserid)
    except Hairdresser.DoesNotExist:

```

```

        hairdresser = None
    if not hairdresser:
        return render(request, "hds/hairdresser.html",
{"message": "Такого пепукаря немає"})
        hairdresser_categories =
HairdresserCategory.objects.filter(hairdresser = hairdresser)
        hairdresser_procedures =
HairdresserProcedure.objects.filter(hairdresser = hairdresser)
        hairprocpag = Paginator(hairdresser_procedures, 10)
        hairprocpart = request.GET.get('pageproc')
        hairdresser_procedures =
hairdresserpart.get_page(hairprocpart)
        for hairdresser_procedure in hairdresser_procedures:
            hairdresser_reservation_workss =
HairdresserReservationProcedureWork.objects.select_related("hairdresse
reservationprocedure",
"hairdresserreservationprocedure__hairdresserprocedure",
"hairdresserreservationprocedure__hairdresserprocedure__hairprocedure"
).all()
            hairdresser_reservation_workss =
hairdresser_reservation_workss.filter(hairdresserreservationprocedure_
_hairdresserprocedure = hairdresser_procedure)
            hairdresser_reservation_workss =
hairdresser_reservation_workss.order_by("-
hairdresserreservationprocedure__hairdresserreservation__reservedat")
            hairdresser_procedure.works =
hairdresser_reservation_workss[:5]
            hairdresser_diplomas =
HairdresserDiploma.objects.select_related("diplomatype").all()
            hairdresser_diplomas =
hairdresser_diplomas.filter(hairdresser = hairdresser)
            hairdresser_diplomas = hairdresser_diplomas.order_by("-
authorizedat")
            hairdiplpag = Paginator(hairdresser_diplomas, 10)
            hairdiplpart = request.GET.get('pagedip')
            hairdresser_diplomas = hairdiplpag.get_page(hairdiplpart)
            hairdresser_works =
Hairdresserwork.objects.filter(hairdresser = hairdresser)
            hairdresser_works = hairdresser_works.order_by("-
authorizedat")
            hairworkpag = Paginator(hairdresser_works, 10)
            hairworkpart = request.GET.get('pagework')
            hairdresser_works = hairworkpag.get_page(hairworkpart)
            hairdresser_reservation_works =
HairdresserReservationProcedureWork.objects.select_related("hairdresse
reservationprocedure",
"hairdresserreservationprocedure__hairdresserprocedure",

```

```

"hairstresserreservationprocedure__hairstresserprocedure__hairprocedure"
).all()
    hairstresser_reservation_works =
hairstresser_reservation_works.filter(hairstresserreservationprocedure__
hairstresserreservation__hairstresser = hairstresser)
    hairstresser_reservation_works =
hairstresser_reservation_works.order_by("-
hairstresserreservationprocedure__hairstresserreservation__reservedat")
    hairrespag = Paginator(hairstresser_reservation_works, 10)
    hairrespart = request.GET.get('pageres')
    hairstresser_reservation_works =
hairrespag.get_page(hairrespart)
    hairstresser_hairshops =
HairstresserShop.objects.filter(hairstresser =
hairstresser).values_list("hairshop__id")
    hairstresser_hairshops = Hairshop.objects.get(id__in =
hairstresser_hairshops)
    marks =
HairstresserReservation.objects.select_related("useracc").all()
    marks = marks.filter(hairstresser = hairstresser)
    marks = marks.order_by("-reservedat")
    marks = marks[:10]
    hairstresser_timetable =
HairstresserTimetable.objects.select_related("hairstressershop").all()
    hairstresser_timetable =
hairstresser_timetable.filter(hairstressershop__hairstresser =
hairstresser)
    hairstresser_slots =
HairstresserSlot.objects.select_related("hairstressershop").all()
    hairstresser_slots =
hairstresser_slots.filter(hairstressershop__hairstresser = hairstresser)
    from datetime import date
    from_date = date.today()
    hairstresser_slots = hairstresser_slots.filter(slotdate__gte =
from_date)
    hairstresser_slots = hairstresser_slots.order_by("-slotdate")
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user =
Useracc.objects.prefetch_related("hairstressers").all()
        hds_user = hds_user.get(user = hds_user_auth)
        if hairstresser in hds_user.hairstressers:
            hairstresser.save = True
        else:
            hairstresser.save = False
    return render(request, "hds/hairstresser.html",
{"hairstresser": hairstresser,

```

```

        "categories": hairdresser_categories,
        "procedures": hairdresser_procedures,
        "diplomas": hairdresser_diplomas,
        "works": hairdresser_works,
        "resworks": hairdresser_reservation_works,
        "hairshops": hairdresser_hairshops,
        "hdmakes": marks,
        "slots": hairdresser_slots,
        "timetable": hairdresser_timetable })

#перегляд бронювань послуг у перукарні, самотійно створених
def ownhairdresserreservations(request, reservestate = None):
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user =
Useracc.objects.prefetch_related("hairdressers").select_related("user"
).all()
        hds_user = hds_user.get(user = hds_user_auth)
    else:
        return render(request, "hds/reservations.html",
{"message": "Потрібно виконати авторизацію"})
        hairreservations =
HairdresserReservation.objects.select_related("hairdresser").all()
        hairreservations = hairreservations.filter(useracc =
hds_user)
        if reservestate:
            hairreservations = hairreservations.filter(reservestate =
reservestate)
            hairreservations = hairreservations.order_by("-reservedat")
            return render(request, "hds/reservations.html",
{"reservations": hairreservations,
            "user": hds_user })

#перегляд відгуків про перукаря
def hairdressermarksviw(request, hairdresserid):
    try:
        hairdresser= Hairdresser.objects.get(id = hairdresserid)
    except Hairdresser.DoesNotExist:
        hairdresser = None
    if not hairdresser:
        return render(request, "hds/markhairdresser.html",
{"message": "Такого перукаря немає"})
        hairdresser_categories =
HairdresserCategory.objects.filter(hairdresser = hairdresser)

```

```

        marks =
HairdresserReservation.objects.select_related("useracc").all()
        marks = marks.filter(hairdresser = hairdresser)
        marks = marks.order_by("-reservedat")
        markpag = Paginator(marks, 10)
        markpart = request.GET.get('pagemark')
        marks = markpag.get_page(markpart)
        if request.user.is_authenticated:
            hds_user_auth = request.user
            hds_user =
Useracc.objects.prefetch_related("hairdressers").all()
            hds_user = hds_user.get(user = hds_user_auth)
            hds_user = hds_user.filter(user = hds_user)
            if hairdresser in hds_user.hairdressers:
                hairdresser.save = True
            else:
                hairdresser.save = False
        return render(request, "hds/markhairdresser.html",
{"hairdresser": hairdresser,
    "categories": hairdresser_categories,
    "hdmmarks": marks })

#перегляд відгуків про перукарів, створених користувачем
def ownhairdressermarksview(request, user = None):
    if user:
        hds_user_auth = request.user
        hds_user =
Useracc.objects.prefetch_related("hairdressers").all()
        hds_user = hds_user.get(user__username = user)
    elif not user and request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user =
Useracc.objects.prefetch_related("hairdressers").all()
        hds_user = hds_user.get(user = hds_user_auth)
    else:
        return render(request, "hds/ownhdmmarks.html",
{"message": "Потрібно виконати авторизацію"})
        marks =
HairdresserReservation.objects.select_related("hairdresser").all()
        marks = marks.filter(useracc = hds_user)
        marks = marks.order_by("-reservedat")
        markpag = Paginator(marks, 10)
        markpart = request.GET.get('pagemark')
        marks = markpag.get_page(markpart)
        for mark in marks:
            if mark.hairdresser in hds_user.hairdressers:

```

```

        mark.hairdresser.save = True
    else:
        mark.hairdresser.save = False
    return render(request, "hds/ownmarks.html", {"hdm": mark, "hdm_marks": marks })

#перегляд збережених записів перукарів
def bookhairdressersview(request):
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user = Useracc.objects.prefetch_related("hairdressers").all()
        hds_user = hds_user.get(user = hds_user_auth)
        hairdresses_by_user = hds_user.values_list("hairdressers__id")
    else:
        return render(request, "hds/book.html", {"message": "Потрібно виконати авторизацію"})
    hairdressers = Hairdresser.objects.filter(id__in = hairdresses_by_user)
    hairdresserpag = Paginator(hairdressers, 10)
    hairdresserpart = request.GET.get('pagedressers')
    hairdressers = hairdresserpag.get_page(hairdresserpart)
    for hairdresser in hairdressers:
        hairdresser.categories = HairdresserCategory.objects.filter(hairdresser = hairdresser)
        if hairdresser in hds_user.hairdressers:
            hairdresser.save = True
        else:
            hairdresser.save = False
    return render(request, "hds/book.html", {"user": hds_user, "hairdressers": hairdressers })

#перегляд бронювань перукарських засобів за користувачем
def userhairshoppreservationsview(request, user, reservestate = None):
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user = Useracc.objects.select_related("user").all()
        hds_user = hds_user.get(user__username = user)
    else:
        return render(request, "hds/hsiuserreservations.html", {"message": "Потрібно виконати авторизацію"})
    hairreservations = HairshopReservation.objects.select_related("hairshop").all()

```

```

        hairreservations = hairreservations.filter(useracc =
hds_user)
        if reservestate:
            hairreservations = hairreservations.filter(reservestate =
reservestate)
            hairreservations = hairreservations.order_by("-reservedat")
            return render(request, "hds/hsiuserreservations.html",
{"reservations": hairreservations,
    "user": hds_user })

```

```

#пошук перукаря
def searchhairdressersview(request):
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user =
Useracc.objects.prefetch_related("hairdressers").all()
        hds_user = hds_user.get(user = hds_user_auth)
        hairdresses_by_user =
hds_user.values_list("hairdressers__id")
        hairdressers = Hairdresser.objects.all()
        hairdressername = request.GET.get('name')
        hairdressers = hairdressers.filter(workname__icontains =
hairdressername)
        hairdressermark = request.GET.get('markmin')
        if hairdressermark:
            hairdressermark = float(hairdressermark)
            hairdressers = hairdressers.filter(avmark__gte =
hairdressermark)
        hairdressermark = request.GET.get('markmax')
        if hairdressermark:
            hairdressermark = float(hairdressermark)
            hairdressers = hairdressers.filter(avmark__lte =
hairdressermark)
        hairdressercategory = request.GET.get('categ')
        if hairdressercategory:
            hairdressercategory = int(hairdressercategory)
            hairdressercategories =
HairdresserCategory.objects.select_related('hairdresser')
            hairdressercategories =
hairdressercategories.filter(hairdresser__in = hairdressers)
            hairdressercategories =
hairdressercategories.filter(category = hairdressercategory)
            hairdressercategories =
hairdressercategories.values_list('hairdresser__id')
            hairdressers = hairdressers.filter(id__in =
hairdressercategories)

```

```

hairdresserdiploma = request.GET.get('diploma')
if hairdresserdiploma:
    hairdresserdiploma = int(hairdresserdiploma)
    try:
        diplomatype = Diplomatype.objects.get(id =
hairdresserdiploma)
    except Diplomatype.DoesNotExist:
        diplomatype = None
    hairdresserdiplomas =
HairdresserDiploma.objects.select_related('hairdresser')
    hairdresserdiplomas =
hairdresserdiplomas.filter(hairdresser = hairdressers)
    hairdresserdiplomas =
hairdresserdiplomas.filter(diplomatype = diplomatype)
    hairdresserdiplomas =
hairdresserdiplomas.values_list('hairdresser__id')
    hairdressers = hairdressers.filter(id__in =
hairdresserdiplomas)
    hairdressersshop = request.GET.get('shop')
    if hairdressersshop:
        hairdressersshop = int(hairdressersshop)
        try:
            hairdressersshop = Hairshop.objects.get(id =
hairdressersshop)
        except Hairshop.DoesNotExist:
            hairdressersshop = None
        shops =
HairdresserShop.objects.select_related('hairdresser')
        shops = shops.filter(hairdresser__in = hairdressers)
        shops = shops.filter(hairshop = hairdressersshop)
        shops = shops.values_list('hairdresser__id')
        hairdressers = hairdressers.filter(id__in = shops)
    hairdresserexper = request.GET.get('exper')
    if hairdresserexper:
        hairdresserexper = int(hairdresserexper)
        hairdressers_selected = []
        from datetime import date
        to_date = date.today()
        from dateutil.relativedelta import relativedelta
        for hairdresser in hairdressers:
            experience = relativedelta(to_date,
hairdresser.hairdresserfrom).years
            if experience >= hairdresserexper:
                hairdressers_selected.append(hairdresser.id)
        hairdressers = hairdressers.filter(id__in =
hairdressers_selected)
    hairdresserpag = Paginator(hairdressers, 10)

```

```

    hairdresserpart = request.GET.get('pagedressers')
    hairdressers = hairdresserpag.get_page(hairdresserpart)
    for hairdresser in hairdressers:
        hairdresser.categories =
HairdresserCategory.objects.filter(hairdresser = hairdresser)
        if hairdresser in hds_user.hairdressers:
            hairdresser.save = True
        else:
            hairdresser.save = False
    hairshops = Hairshop.objects.all()
    return render(request, "hds/search.html", {"user": hds_user,
        "hairdressers": hairdressers,
        "hairshops": hairshops })

```

```

#перегляд інформації про перукарську послугу
def hairprocedurereview(request, hairprocedureid, hairdresserid =
None):
    try:
        hairprocedure = Hairprocedure.objects.get(id =
hairprocedureid)
    except Hairprocedure.DoesNotExist:
        hairprocedure = None
    if not hairprocedure:
        return render(request, "hds/hairproced.html",
{"message": "Такої перукарської послуги немає"})
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user =
Useracc.objects.prefetch_related("hairdressers").all()
        hds_user = hds_user.get(user = hds_user_auth)
    if hairdresserid:
        try:
            hairdresser = Hairdresser.objects.get(id =
hairdresserid)
        except Hairdresser.DoesNotExist:
            hairdresser = None
        hairprocedures =
HairdresserProcedure.objects.select_related("hairprocedure",
"hairdresser", "hairprocedure__hpcategory").all()
        hairprocedures = hairprocedures.filter(hairdresser =
hairdresser)
        hairprocedure = hairprocedures.filter(hairprocedure =
hairprocedure)
        if hairdresser in hds_user.hairdressers:
            hairprocedure.hairdresser.save = True
        else:

```

```

        hairprocedure.hairdresser.save = False
        hairdresser_reservation_works =
HairdresserReservationProcedureWork.objects.select_related("hairdresse
rreservationprocedure",
"hairdresserreservationprocedure__hairdresserprocedure",
"hairdresserreservationprocedure__hairdresserprocedure__hairprocedure"
).all()
        hairdresser_reservation_works =
hairdresser_reservation_works.filter(hairdresserreservationprocedure__
hairdresserprocedure = hairprocedure)
        hairdresser_reservation_works =
hairdresser_reservation_works.order_by("-
hairdresserreservationprocedure__hairdresserreservation__reservedat")
        hairrespag = Paginator(hairdresser_reservation_works, 10)
        hairrespart = request.GET.get('pageres')
        hairdresser_reservation_works =
hairrespag.get_page(hairrespart)
        return render(request, "hds/hairproced.html", {"procedure":
hairprocedure,
        "works": hairdresser_reservation_works })

```

```

#перегляд детальної інформації бронювання послуг у перукарні
def hairdresserreservationview(request,
hairdresserreservationid):
    try:
        hairreservation =
HairdresserReservation.objects.select_related("hairdresser",
"useracc")
        hairreservation = hairreservation.get(id =
hairdresserreservationid)
    except HairdresserReservation.DoesNotExist:
        hairreservation = None
    if not hairreservation:
        return render(request, "hds/reservation.html",
{"message": "Такого бронювання перукарських послуг немає"})
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user =
Useracc.objects.prefetch_related("hairdressers").select_related("user"
).all()
        hds_user = hds_user.get(user = hds_user_auth)
    else:
        return render(request, "hds/reservation.html",
{"message": "Потрібно виконати авторизацію"})
    if not (hds_user.user.is_staff or hds_user ==
hairreservation.useracc):

```

```

        return render(request, "hds/reservation.html",
{"message": "Недостатньо прав"}
        if hairreservation.hairdresser in hds_user.hairdressers:
            hairreservation.hairdresser.save = True
        else:
            hairreservation.hairdresser.save = False
        hairprocedures =
HairdresserReservationProcedure.objects.filter(hairdresserreservation
= hairreservation)
        for hairprocedure in hairprocedures:
            hairprocedure.works =
HairdresserReservationProcedureWork.objects.filter(hairdresserreservat
ionprocedure = hairprocedure)
        return render(request, "hds/reservation.html",
{"reservation": hairreservation,
        "procedures": hairprocedures }

```

```

#перегляд бронювань послуг у перукарні за користувачем
def userhairdresserreservationsview(request, user, reservestate
= None):
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user =
Useracc.objects.prefetch_related("hairdressers").select_related("user"
).all()
        hds_user = hds_user.get(user__username = user)
    else:
        return render(request, "hds/ureservations.html",
{"message": "Потрібно виконати авторизацію"}
        hairreservations =
HairdresserReservation.objects.select_related("hairdresser").all()
        hairreservations = hairreservations.filter(useracc =
hds_user)
        if reservestate:
            hairreservations = hairreservations.filter(reservestate =
reservestate)
            hairreservations = hairreservations.order_by("-reservedat")
        return render(request, "hds/ureservations.html",
{"reservations": hairreservations,
        "user": hds_user }

```

```

#перегляд всіх бронювань послуг перукарні за станом
def allhairdresserreservationsview(request, reservestate =
None):
    if request.user.is_authenticated:

```

```

        hds_user_auth = request.user
        hds_user = Useracc.objects.select_related("user").all()
        hds_user = hds_user.get(user = hds_user_auth)
    else:
        return render(request, "hds/allreservations.html",
{"message": "Потрібно виконати авторизацію"})
        hairreservations =
HairdresserReservation.objects.select_related("hairdresser").all()
        if not hds_user.user.is_staff:
            return render(request, "hds/allreservations.html",
{"message": "Користувач має бути менеджером перукарні. У вас
недостатньо прав"})
        if reservestate:
            hairreservations = hairreservations.filter(reservestate =
reservestate)
            hairreservations = hairreservations.order_by("-reservedat")
            return render(request, "hds/allreservations.html",
{"reservations": hairreservations })

```

```

#перегляд запланованих бронювань послуг перукарні за календарем
def allcalhairdresserreservationsview(request, resstart = None,
resend = None, reservestate = None):
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user = Useracc.objects.select_related("user").all()
        hds_user = hds_user.get(user = hds_user_auth)
    else:
        return render(request, "hds/calreservations.html",
{"message": "Потрібно виконати авторизацію"})
    from datetime import datetime
    resstart = datetime.strptime(resstart, '%d.%m.%y %H:%M')
    resend = datetime.strptime(resend, '%d.%m.%y %H:%M')
    hairreservations =
HairdresserReservation.objects.select_related("hairdresser").all()
    if resstart:
        hairreservations =
hairreservations.filter(procedurestart__gte = resstart)
    if resend:
        hairreservations =
hairreservations.filter(procedurestart__lte = resend)
    if not hds_user.user.is_staff:
        return render(request, "hds/allreservations.html",
{"message": "Користувач має бути менеджером перукарні. У вас
недостатньо прав"})
    if reservestate:

```

```

        hairreservations = hairreservations.filter(reservestate =
reservestate)
        hairreservations = hairreservations.order_by("-reservedat")
        return render(request, "hds/calreservations.html",
{"reservations": hairreservations })

```

```

#перегляд бронювань перукарських засобів, самостійно створених
def ownhairshoppreservationsview(request, reservestate = None):
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user = Useracc.objects.select_related("user").all()
        hds_user = hds_user.get(user = hds_user_auth)
    else:
        return render(request, "hds/hsireservations.html",
{"message": "Потрібно виконати авторизацію"})
    hairreservations =
HairshopReservation.objects.select_related("hairshop").all()
    hairreservations = hairreservations.filter(useracc =
hds_user)
    if reservestate:
        hairreservations = hairreservations.filter(reservestate =
reservestate)
        hairreservations = hairreservations.order_by("-reservedat")
        return render(request, "hds/hsireservations.html",
{"reservations": hairreservations,
"user": hds_user })

```

```

#перегляд всіх бронювань перукарських засобів
def allhairshoppreservationsview(request, reservestate = None):
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user = Useracc.objects.select_related("user").all()
        hds_user = hds_user.get(user = hds_user_auth)
    else:
        return render(request, "hds/allitreservations.html",
{"message": "Потрібно виконати авторизацію"})
    hairreservations =
HairshopReservation.objects.select_related("hairshop").all()
    if not hds_user.user.is_staff:
        return render(request, "hds/allhsireservations.html",
{"message": "Користувач має бути менеджером перукарні. У вас
недостатньо прав"})
    if reservestate:
        hairreservations = hairreservations.filter(reservestate =
reservestate)

```

```

        hairreservations = hairreservations.order_by("-reservedat")
        return render(request, "hds/allitreservations.html",
{"reservations": hairreservations })

```

```

#перегляд детальної інформації бронювання перукарських засобів
def hairshopreservationview(request, hairshopreservationid):
    try:
        hairshopreservation =
HairshopReservation.objects.select_related("hairshop",
"useracc").all()
        hairshopreservation = hairshopreservation.get(id =
hairshopreservationid)
    except HairshopReservation.DoesNotExist:
        hairshopreservation = None
    if not hairshopreservation:
        return render(request, "hds/hsireservation.html",
{"message": "Такого бронювання перукарських засобів немає"}
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user = Useracc.objects.select_related("user").all()
        hds_user = hds_user.get(user = hds_user_auth)
    else:
        return render(request, "hds/hsireservation.html",
{"message": "Потрібно виконати авторизацію"})
    if not (hds_user.user.is_staff or hds_user ==
hairreservation.useracc):
        return render(request, "hds/hsireservation.html",
{"message": "Недостатньо прав"})
    hairitems =
HairshopReservationItem.objects.select_related("hairshopitem").all()
    hairitems = hairitems.filter(hairshopreservation =
hairshopreservation)
    return render(request, "hds/hsireservation.html",
{"reservation": hairreservation,
    "items": hairitems })

```

```

#пошук перукарських засобів у продажі
def searchhairitemsview(request):
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user = Useracc.objects.get(user = hds_user_auth)
    hairitems =
Hairitem.objects.select_related('hairitemcategory').all()
    hairitemname = request.GET.get('name')

```

```

        hairitems = hairitems.filter(itemname__icontains =
hairitemname)
        hairitemcategory = request.GET.get('categ')
        if hairitemcategory:
            hairitemcategory = int(hairitemcategory)
            try:
                hairitemcategory = HairitemCategory.objects.get(id =
hairitemcategory)
            except HairitemCategory.DoesNotExist:
                hairitemcategory = None
            hairitems = hairitems.filter(hairitemcategory =
hairitemcategory)
        hairitemmaxprice = request.GET.get('maxprice')
        if hairitemmaxprice:
            hairitemmaxprice = float(hairitemmaxprice)
            hairitems = hairitems.filter(price__lte =
hairitemmaxprice)
        hairitemminprice = request.GET.get('minprice')
        if hairitemminprice:
            hairitemminprice = float(hairitemminprice)
            hairitems = hairitems.filter(price__gte =
hairitemminprice)
        hairitemmaxvolume = request.GET.get('maxvol')
        if hairitemmaxvolume:
            hairitemmaxvolume = float(hairitemmaxvolume)
            hairitems = hairitems.filter(volume__lte =
hairitemmaxvolume)
        hairitemminvolume = request.GET.get('minvol')
        if hairitemminvolume:
            hairitemminvolume = float(hairitemminvolume)
            hairitems = hairitems.filter(volume__gte =
hairitemminvolume)
        hairdressersshop = request.GET.get('shop')
        if hairdressersshop:
            hairdressersshop = int(hairdressersshop)
            try:
                hairdressersshop = Hairshop.objects.get(id =
hairdressersshop)
            except Hairshop.DoesNotExist:
                hairdressersshop = None
        shops = HairShopItem.objects.select_related('hairitem')
        shops = shops.filter(hairitem__in = hairitems)
        shops = shops.filter(hairshop__in = hairdressersshop)
        shops = shops.values_list('hairitem__id')
        hairitems = hairitems.filter(id__in = shops)
        hairitemspag = Paginator(hairitems, 10)
        hairitempart = request.GET.get('part')

```

```

        hairitems = hairitemspag.get_page(hairitempart)
        hairitemcategories = HairitemCategory.objects.all()
        return render(request, "hds/searchhairitems.html", {"user":
hds_user,
                        "items": hairitems,
                        "categories": hairitemcategories })

#перегляд інформації про перукарський засіб
def hairitemview(request, hairitemid):
    try:
        haititem =
Hairitem.objects.select_related("hairitemcategory").all()
        haititem = haititem.get(id = hairitemid)
    except Hairitem.DoesNotExist:
        haititem = None
    if not haititem:
        return render(request, "hds/hitem.html", {"message":
"Такого перукарського засобу немає"}
        shops = HairShopItem.objects.select_related("hairshop")
        shops = shops.filter(hairitem = haititem)
        shops = shops.values_list("hairshop__id")
        return render(request, "hds/hairitem.html", {"item":
haititem,
                        "shops": shops })

#збереження запису перукаря
def dobookhairdresserview(request, hairdresserid):
    try:
        hairdresser = Hairdresser.objects.get(id =
hairdresserid)
    except Hairdresser.DoesNotExist:
        hairdresser = None
    if request.user.is_authenticated:
        hds_user_auth = request.user
        hds_user =
Useracc.objects.prefetch_related("hairdressers").all()
        hds_user = hds_user.get(user = hds_user_auth)
    if hairdresser in hds_user.hairdressers:
        hds_user.hairdressers.remove(hairdresser)
    else:
        hds_user.hairdressers.add(hairdresser)

from .forms import HairshopForm

```

```

#додавання перукарні до мережі
def hairshopaddview(request):
    if not (request.user.is_active and request.user.is_staff):
        return render(request, "hds/hairshopadd.html", {"form":
None })
    if request.method == "POST":
        hairshop_form = HairshopForm(request.POST)
        if hairshop_form.is_valid():
            hairshop = hairshop_form.save(commit=True)
            return redirect('hairshopview', hairshopid =
hairshop.id)
        else:
            hairshop_form = HairshopForm()
            return render(request, "hds/hairshopadd.html", {"form":
hairshop_form })

#змінювання інформації про перукарню
def hairshopeditview(request, hairshopid):
    if request.user.is_active and request.user.is_staff:
        try:
            hairshop = Hairshop.objects.get(id = hairshopid)
        except Hairshop.DoesNotExist:
            hairshop = None
        if hairshop:
            hairshop_form = HairshopForm(request.POST or None,
instance = hairshop)
            if request.method == "POST" and
hairshop_form.is_valid():
                hairshop_form.save()
                return redirect('hairshopview', hairshopid =
hairshopid)
            else:
                hairshop_form = None
            return render(request, "hds/hairshopedit.html", {"form":
hairshop_form })

from .forms import HairprocedureForm

#додавання перукарської послуги
def hairprocedureaddview(request):
    if not (request.user.is_active and request.user.is_staff):
        return render(request, "hds/hairprocadd.html", {"form":
None })
    if request.method == "POST":

```

```

        hairprocedure_form = HairprocedureForm(request.POST)
        if hairprocedure_form.is_valid():
            hairprocedure = hairprocedure_form.save(commit=True)
            return redirect('hairprocedureview', hairprocedureid
= hairprocedure.id)
        else:
            hairprocedure_form = HairprocedureForm()
            return render(request, "hds/hairprocadd.html", {"form":
hairprocedure_form })

```

```

#змінювання інформації про перукарську послугу
def hairprocedureeditview (request, hairprocedureid):
    if request.user.is_active and request.user.is_staff:
        try:
            hairproc = Hairprocedure.objects.get(id =
hairprocedureid)
        except Hairprocedure.DoesNotExist:
            hairproc = None
        if hairproc:
            hairproc_form = HairprocedureForm(request.POST or
None, instance = hairproc)
            if request.method == "POST" and
hairproc_form.is_valid():
                hairproc_form.save()
                return redirect('hairprocedureview',
hairprocedureid = hairproc.id)
            else:
                hairproc_form = None
            return render(request, "hds/hairprocedit.html", {"form":
hairproc_form })

```

```

from .forms import HairdresserForm

```

```

#додавання перукаря
def hairdresseraddview(request):
    if not (request.user.is_active and request.user.is_staff):
        return render(request, "hds/hairdresseradd.html",
{"form": None })
    if request.method == "POST":
        hairdresser_form = HairdresserForm(request.POST)
        if hairdresser_form.is_valid():
            hairdresser = hairdresser_form.save()
            return redirect('hairdresserview', hairdresserid =
hairdresser.id)

```

```

        else:
            hairdresser_form = HairdresserForm()
            return render(request, "hds/hairdresseradd.html", {"form":
hairdresser_form })

#змінювання інформації про перукарську послугу
def hairdressereditview(request, hairdresserid):
    if request.user.is_active and request.user.is_staff:
        try:
            hairdresser = Hairdresser.objects.get(id =
hairdresserid)
        except Hairdresser.DoesNotExist:
            hairdresser = None
        if hairdresser:
            hairdresser_form = HairdresserForm(request.POST or
None, instance = hairdresser)
            if request.method == "POST" and
hairdresser_form.is_valid():
                hairdresser_form.save()
                return redirect('hairdressersview', hairdresserid
= hairdresserid)
            else:
                hairdresser_form = None
        return render(request, "hds/hairdresseredit.html", {"form":
hairdresser_form })

```

ДОДАТОК В
Слайди презентації

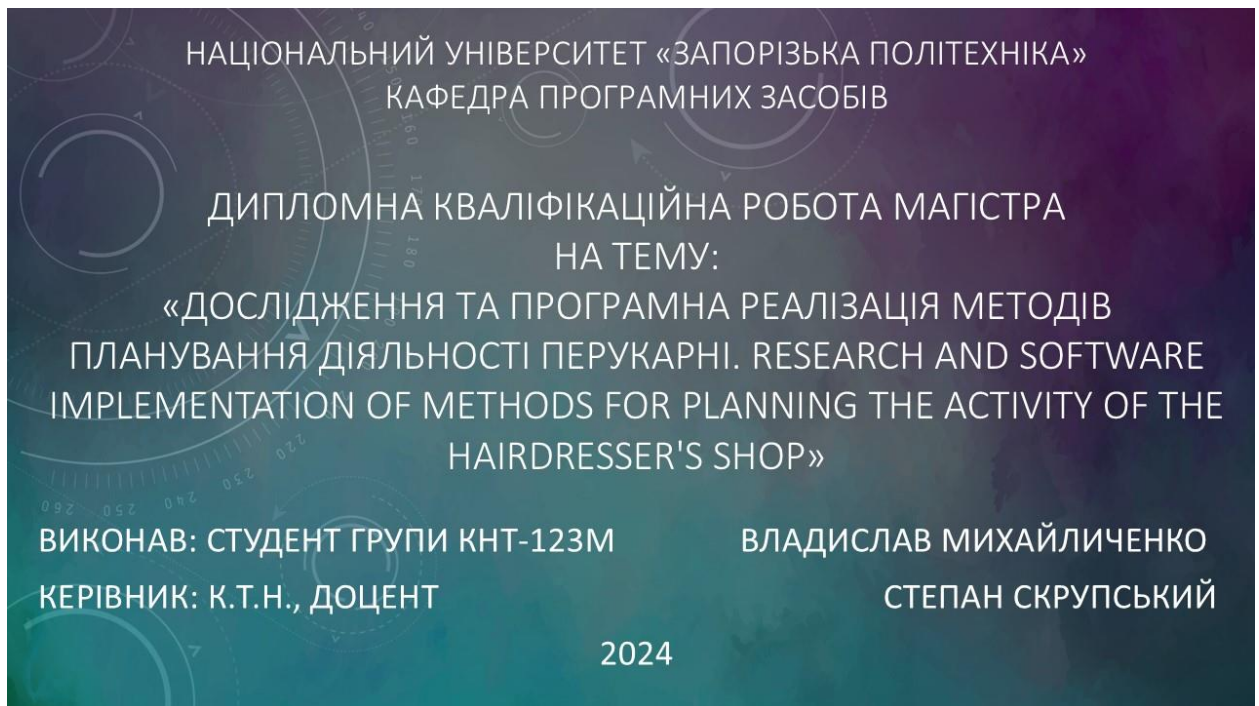


Рисунок В.1 – Початковий слайд

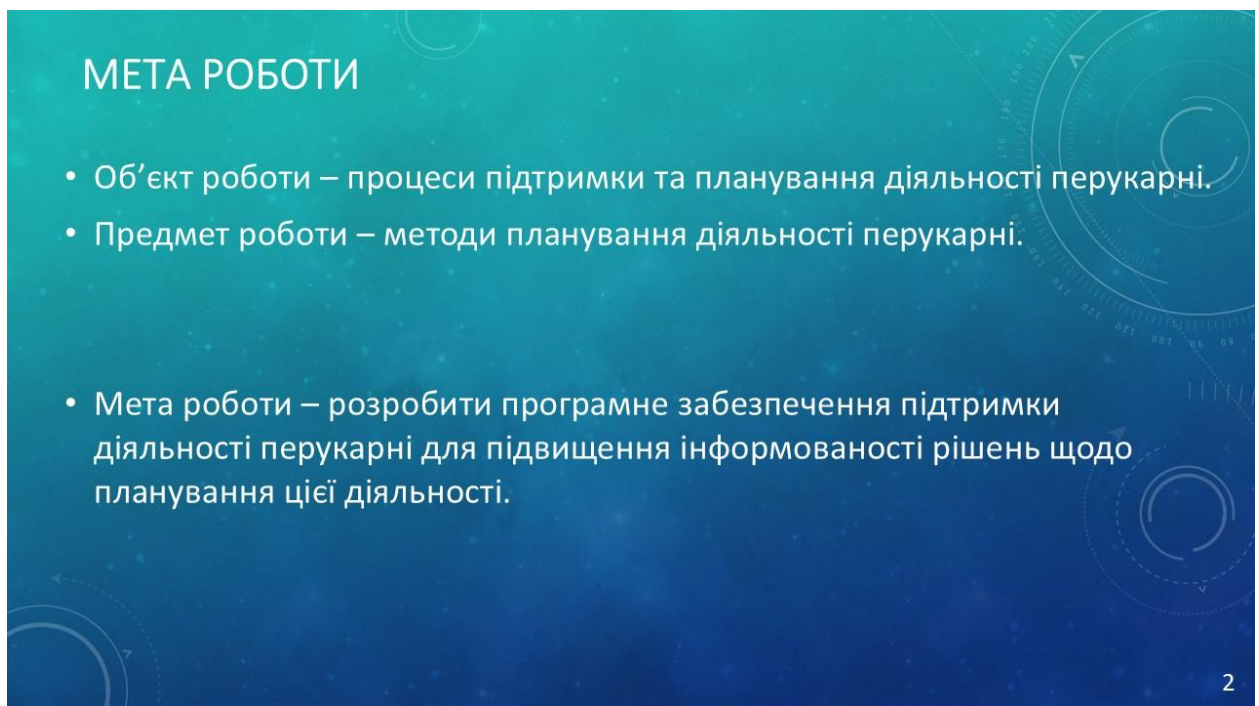


Рисунок В.2 – Мета роботи

ЗАВДАННЯ РОБОТИ

- виконати аналіз програмних застосунків підтримки діяльності перукарні;
- розробити метод прогнозування кількості відвідувачів перукарні за день;
- виконати проектування вебзастосунку підтримки діяльності перукарні;
- реалізувати вебзастосунок підтримки діяльності перукарні;
- виконати експериментальне дослідження прогнозування кількості відвідувачів перукарні за день.

3

Рисунок В.3 – Завдання роботи

ПОРІВНЯННЯ МОЖЛИВОСТЕЙ АНАЛОГІВ РОЗРОБКИ ВЕБЗАСТОСУНКУ ПІДТРИМКИ ДІЯЛЬНОСТІ ПЕРУКАРНІ

Показник	Barberking	Larry King	Blue Tit	Створена програма
Виокремлена інформація про кожну конкретну послугу	Тільки загальна	–	+	+
Інформація про перукарів	Ім'я та фотографія	+	–	+
Відгуки про перукарів	–	–	–	+
Бронювання послуг у перукарні	+	+	+	+
Продаж супутніх товарів	–	+	+	+
Прогнозування кількості відвідувачів перукарні за день	–	–	–	+

4

Рисунок В.4 – Порівняння можливостей аналогів розробки вебзастосунку підтримки діяльності перукарні

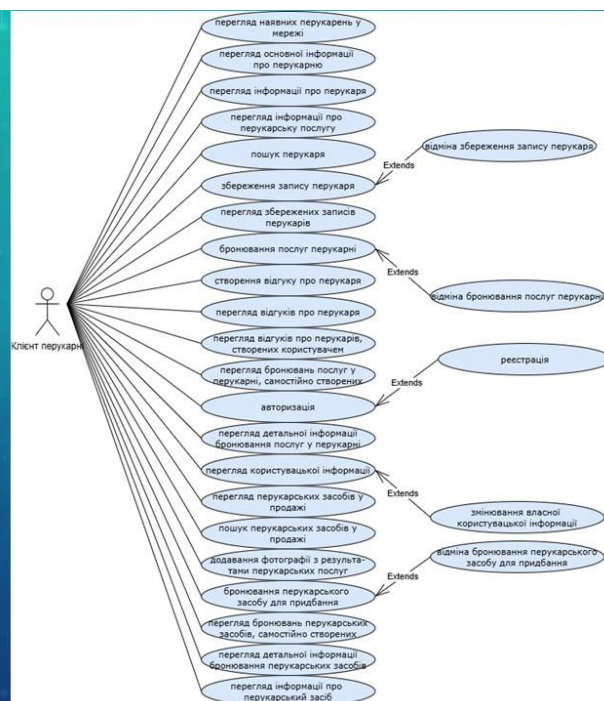
МЕТОД ПРОГНОЗУВАННЯ КІЛЬКОСТІ ВІДВІДУВАЧІВ ПЕРУКАРНІ ЗА ДЕНЬ

- опрацювання отриманої вибірки даних для її застосування в якості навчальної вибірки: визначення неробочих днів, формування часових рядів, нормування;
- створення моделі для прогнозування кількості відвідувачів перукарні за день через забезпечення для цього ансамблю дерев ухвалення рішень, об'єднання яких має реалізовуватись на основі методу Random Forest;
- навчання створеної моделі за допомогою створеної вибірки даних;
- формування нових екземплярів даних та визначення кількості відвідувачів перукарні за майбутній день відносно поточного дня.

5

Рисунок В.5 – Метод прогнозування кількості відвідувачів перукарні за день

ДІАГРАМА ПРЕЦЕДЕНТІВ ДЛЯ КЛІЄНТА ПЕРУКАРНІ



6

Рисунок В.6 – Діаграма прецедентів для клієнта перукарні

	Django	CodeIgniter
Мова програмування в основі	Python	PHP
Тип рішення	Вебфреймворк	Вебфреймворк
Забезпечення авторизації з аутентифікацією	Аутентифікація з підключенням зовнішніх засобів, встановлення дозволів для користувачів	Базова аутентифікація, реєстрація
Підтримуваний рівень складності готового проєкту	Середній або високий	Легкий або середній

Рисунок В.8 – Порівняння можливих рішень для реалізації вебзастосунку підтримки діяльності перукарні



СХЕМА ФУНКЦІОНУВАННЯ ПРОГРАМИ ПІДТРИМКИ ДІЯЛЬНОСТІ ПЕРУКАРНІ



Рисунок В.10 – Схема функціонування програми підтримки діяльності
перукарні

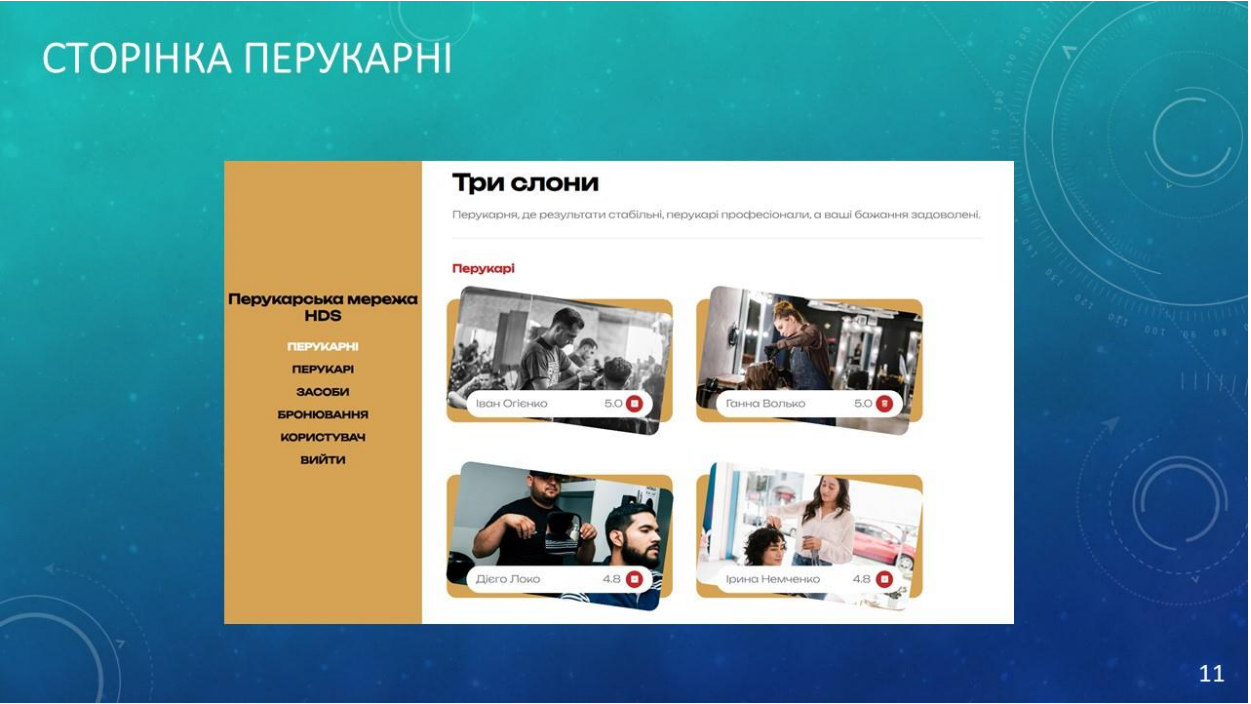


Рисунок В.11 – Сторінка перукарні

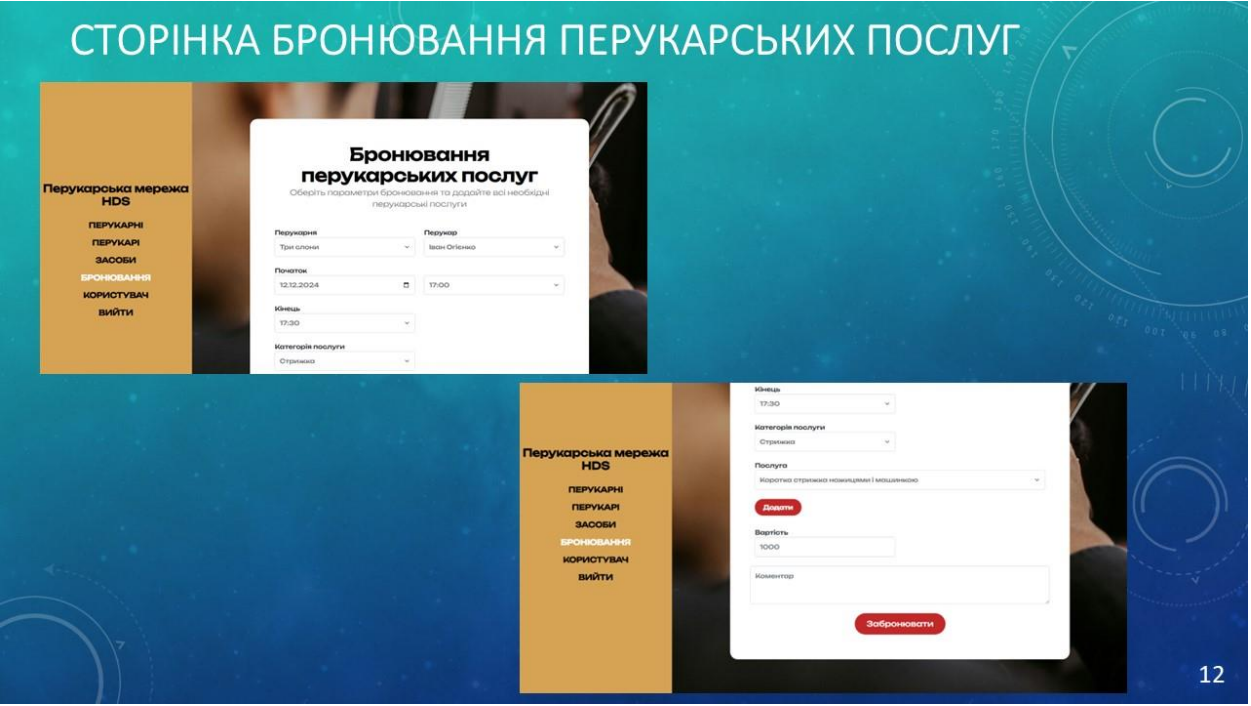
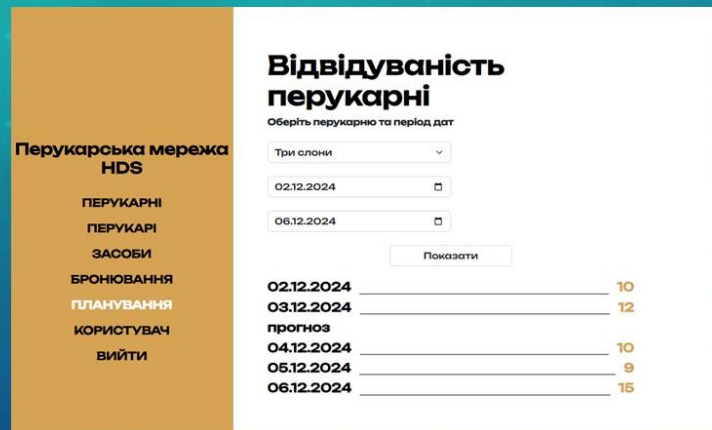


Рисунок В.12 – Сторінка бронювання перукарських послуг

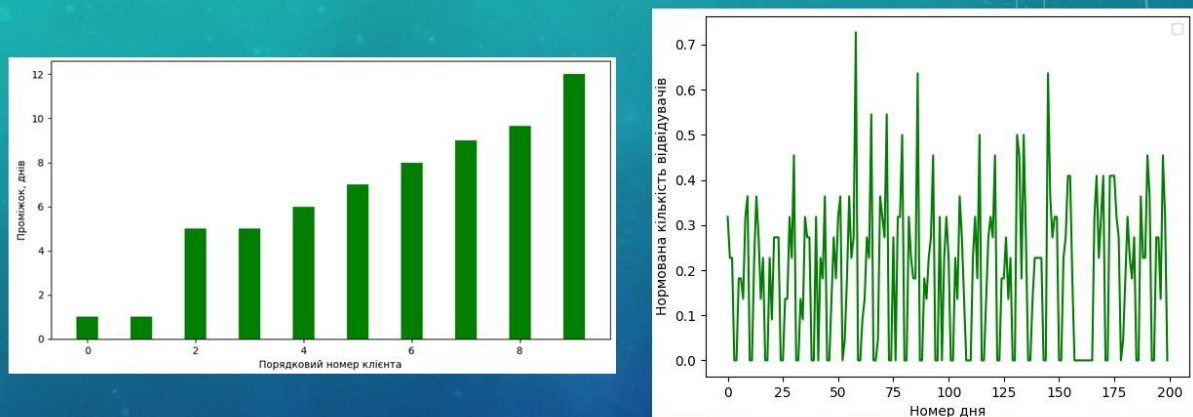
СТОРІНКА ПЛАНУВАННЯ ДІЯЛЬНОСТІ ПЕРУКАРНІ



13

Рисунок В.13 – Сторінка планування діяльності перукарні

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГНОЗУВАННЯ КІЛЬКОСТІ ВІДВІДУВАЧІВ ПЕРУКАРНІ ЗА ДЕНЬ



14

Рисунок В.14 – Експериментальне дослідження прогнозування кількості відвідувачів перукарні за день

РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНОГО ДОСЛІДЖЕННЯ ПРОГНОЗУВАННЯ КІЛЬКОСТІ ВІДВІДУВАЧІВ ПЕРУКАРНІ ЗА ДЕНЬ

Метод прогнозування	MAE	RMSE	MSE
Запропонований метод прогнозування кількості відвідувачів перукарні за день	0,07655473	0,09437362	0,00890638
Метод опорних векторів	0,11519007	0,13786621	0,01900709
ARIMA	0,08330045	0,10774955	0,01160997

15

Рисунок В.15 – Результати експериментального дослідження прогнозування кількості відвідувачів перукарні за день

ВИСНОВКИ

Розроблене програмне забезпечення здатне підтримувати роботу перукарні з клієнтами, зокрема отримання основної необхідної інформації про ті послуги, які надаються перукарнею, бронювання послуг, планування діяльності перукарні. Для програми розроблено сценарії для клієнта та менеджера перукарні, які було реалізовано та описано. Виконано вибір Python разом з фреймворком Django для розробки за результатами порівняння. Визначено структуру бази даних, що включає всі необхідні сутності, пов'язані як з перукарськими послугами, так і з перукарськими засобами, які доступні для продажу, на основі системи керування MySQL.

Наукова новизна роботи полягає в методі прогнозування кількості відвідувачів перукарні за день, що характеризується використанням часового ряду, ансамблю дерев рішень на основі методу Random Forest, що дозволяє визначати сумарну кількість відвідувачів за день з інтервалом у задану кількість днів. До основних результатів роботи також відносяться результати експериментального дослідження, під час якого не тільки порівняно різні моделі, до яких належить ARIMA, метод опорних векторів, запропонований метод, але також і різна довжина часового інтервалу.

16

Рисунок В.16 – Висновки