Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістра
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ КЕРУВАННЯ ЗАМОВЛЕННЯМИ ЗАСОБІВ ДЛЯ ДОГЛЯДУ ЗА ЗДОРОВ'ЯМ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR MANAGING THE ORDERS FOR HEALTHCARE PRODUCTS

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

Сердюк А.В.
(прізвище та ініціали)
Керівник Субботін С.О.
(прізвище та ініціали)
Рецензент Гофман Є.О.
(прізвище та ініціали)

2021

Форма № 25

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

Інститут, факультет ІІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____С.О. Субботін
"____" _____2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Сердюка Антона Валерійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів керування замовленнями засобів для догляду за здоров'ям
Research and Software Implementation of Methods for Managing the Orders for Healthcare Products

керівник проєкту (роботи) Субботін Сергій Олександрович, д.т.н., професор ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від " 5 " листопада 2021 року № 435
2. Строк подання студентом проєкту (роботи) грудень 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

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

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Субботін С.О., зав. кафедри		
Нормоконтролер	Дейнега Л.Ю., ст. викладач		

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програми	7-9 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	10-11 тижні	Розділ 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
7	Захист роботи	13 тиждень	

Студент(ка) _____ Сердюк А.В.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Субботін С.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 87 с., 22 рис., 4 табл., 3 дод., 16 джерел.

КЕРУВАННЯ ЗАМОВЛЕННЯМИ, ДОГЛЯД ЗА ЗДОРОВ'ЯМ, ЗАДАЧА КЛАСИФІКАЦІЇ, ДЕРЕВА РІШЕНЬ, ВЕБЗАСТОСУНОК.

Об'єкт дослідження – процес замовлення засобів для догляду за здоров'ям. Предмет дослідження – методи керування замовленнями засобів для догляду за здоров'ям.

Мета роботи – розробити програмне забезпечення замовлення засобів для догляду за здоров'ям для підвищення ефективності роботи менеджерів щодо обслуговування замовлень.

Матеріали, методи та технічні засоби: мова програмування Python, вебфреймворк Django, дерева рішень.

Результати. Результати роботи охоплюють створену діаграму прецедентів, створений метод класифікації замовлень засобів для догляду за здоров'ям, результати порівняння мов програмування, схему бази даних, діаграму станів, розроблене програмне забезпечення і виконане експериментальне дослідження запропонованого методу.

Висновки. Наукова новизна роботи полягає в запропонованому методі класифікації замовлень засобів для догляду за здоров'ям, який виконує класифікацію на основі параметрів замовлення та часу останнього повернення засобу з використанням дерева рішень на основі алгоритму CART, що дозволяє керувати замовленнями щодо подальшої їх підтримки. Проведене експериментальне дослідження застосування даного методу підтвердило можливість його практичного використання у даній сфері.

Галузь використання – торгівля засобами для догляду за здоров'ям.

ABSTRACT

Explanatory note to the qualifying work of the master: 87 pages, 22 figures, 4 tables, 3 appendixes, 16 sources.

ORDER MANAGEMENT, HEALTH CARE, CLASSIFICATION PROBLEM, DECISION TREES, WEB APPLICATION.

The object of research is the process of ordering health care products. The subject of research is methods of managing orders for health care products.

The purpose of the work is to develop software for ordering health care products to increase the convenience of order managers.

Materials, methods and technical means: Python programming language, Django web framework, decision trees.

Results. The results of the work include a set of precedents, a method for classifying health care orders, results of comparing programming languages, database schema, state diagram, developed software and experimental study of the proposed method.

Conclusions. The scientific novelty of the work lies in the proposed method of classification of health care orders, which performs classification based on the description of the tool and the time of the last return of the tool using the decision tree based on the CART algorithm, which allows to manage orders for further support. An experimental study of the application of this method confirmed the possibility of its practical use in this field.

Area of use ши trade in health care products.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CART – classification and regression tree;

IDF – inverse document frequency;

TF – term frequency;

UPC – universal product code;

ПЗ – програмне забезпечення.

ВСТУП

Здоровий спосіб життя поступово стає достатньо актуальним у суспільстві, що свою чергу призводить до популяризації засобів для догляду за здоров’ям.

У свою чергу згідно з даними Центрів контролю та профілактики захворювань, більше однієї третини або понад 78 мільйонів дорослих у США страждають на ожиріння, тому слід вважати, що ожиріння – це захворювання, яке не можна ігнорувати у суспільстів, адже це пов’язано з багатьма основними причинами смерті, такими як хвороби серця, серцево-судинні захворювання, інсульти та деякі форми раку, однак всього цього можна уникнути за допомогою здорового способу життя. Таким чином, актуальність даної проблеми в суспільстві є ще одним з факторів, які визначають цілий простір для продажу засобів для догляду за здоров’ям [1].

Замовлення засобів для догляду здоров’ям потребує програмного забезпечення (ПЗ), яке буде підтримувати даний процес. Однак варто враховувати, що дана сфера є достатньо специфічною і важливою її особливістю є те, що частіше за все клієнт сам обирає засоби для догляду за здоров’ям, а тому може виявлятися, що його вибір виявився не найкращим. Тоді клієнт може не забрати товар або прийти за ним і вирішити повернути. Будь-яка компанія з продажів не зацікавлена у поверненнях, але має забезпечити за певних умов, в залежності від країни, де це відбувається, повернення замовлень, від яких відмовився покупець. Звичайно, що витрати в такому випадку для компанії зростають, адже проводиться робота, яка не отримує ефективний результат. Все це є проблемою для функціонування таких сервісів. До того ж з іншого боку, великі компанії в сфері продажу в сучасному світі намагаються зменшити необхідність обслуговування клієнтів людьми, а тому часто навіть не перетелефоновують покупцю при відправленні замовлення. Усе це говорить про необхідність диференціації – визначення ситуацій, коли над замовленням потрібно попрацювати

додатково. Такою додатковою роботою може бути взаємодія з клієнтом менеджера, який намагається уточнити, чи не потребує клієнт додаткової консультації, чи точно він впевнений в своєму виборі, які саме вимоги має клієнт до таких засобів. Для цього потрібно мати відповідний інструмент всередині ПЗ. На даний момент існує цілий набір інтелектуальних методів, які можуть використовуватися для розв’язання цієї проблеми, зокрема [2]-[4]. Саме на розроблення ПЗ з таким інструментом у своєму складі і направлено виконання цієї роботи.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд існуючих програмних застосунків замовлення засобів для догляду за здоров'ям

Засоби для догляду за здоров'ям можуть замовлятися і в звичайних магазинах, але є спеціалізовані, які не зважаючи на популярність великих магазинів, які охоплюють зовсім різні групи товарів, є також дуже відомими і популярними у світі.

iHerb (рис. 1.1) є одним з найбільших у США магазинів харчових добавок, продуктів здорового харчування та засобів особистої гігієни з оформленням замовлень онлайн, пропонує величезний вибір найпопулярніших брендів у галузі, надає практично всі можливі в світі добавки [5].

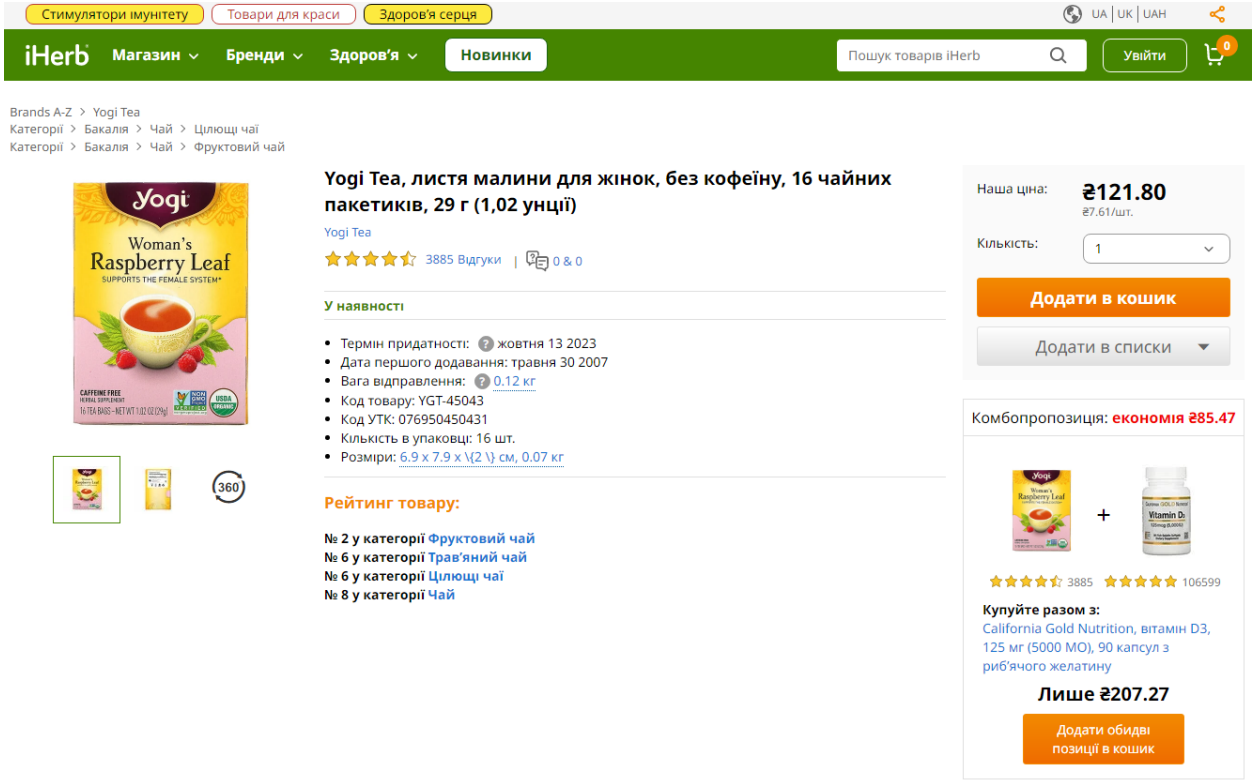

Рисунок 1.1 – Вигляд інтерфейсу вебсайту iHerb

Вебсайт надає користувачу можливість оформити замовлення,

пропонуючи для цього можливість шукати засоби за їх назвою, переглядаючи певні категорії засобів. Окрім самого формування замовлень доступне також створення списків засобів для подальшого перегляду їх та достатньо зручна система висловлення власних вражень від використання засобів. Ця система складається зі звітів користувачів з виставленням оцінок, але при цьому в подальшому такі звіти оцінюють інші користувачі. Вони вказують, що цей звіт був важливим для них (позитивним) або ні (негативним).

Вебсайт PureFormulas (рис. 1.2) є альтернативою iHerb, надає можливість замовити харчові добавки, вітаміни, мінерали тощо [6].

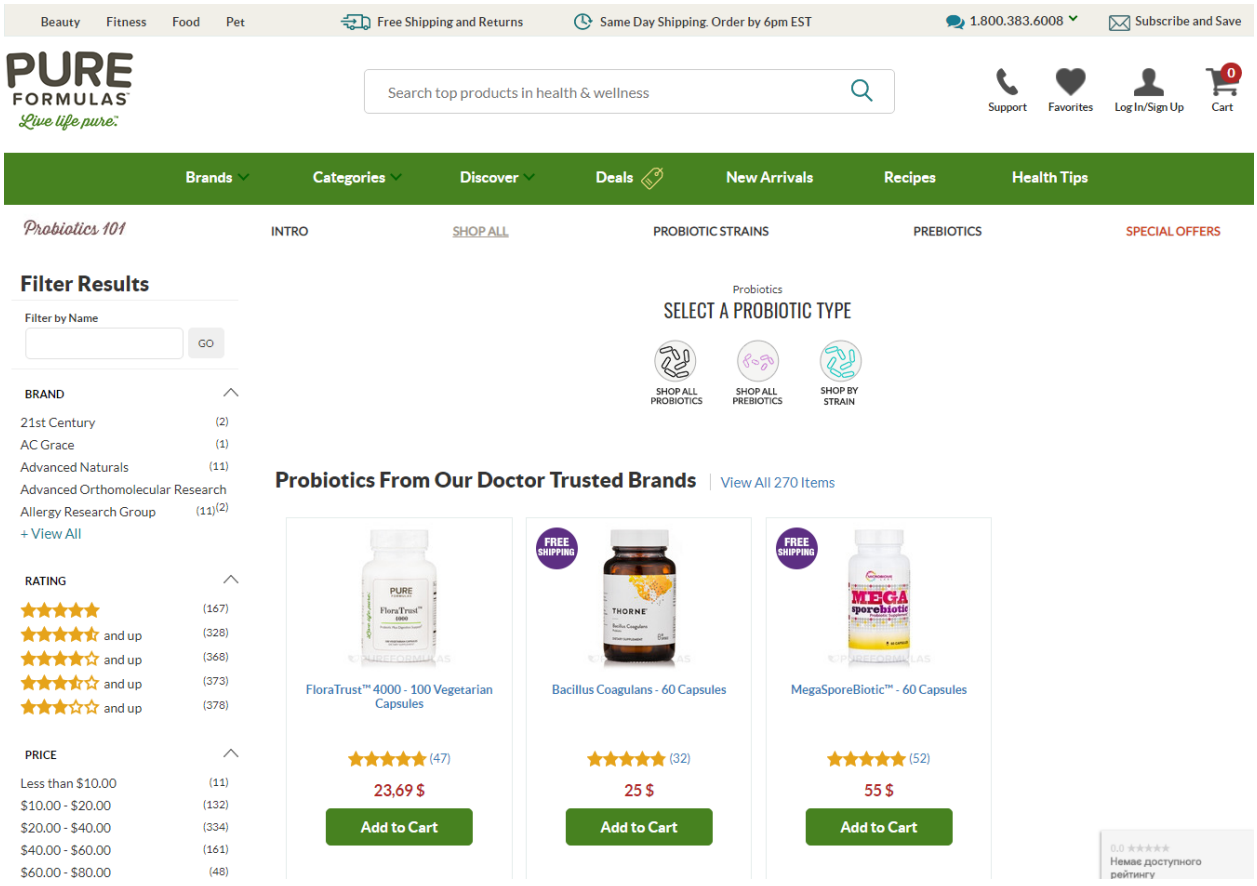

Рисунок 1.2 – Вигляд інтерфейсу вебсайту PureFormulas

Вебсайт також підтримує можливість замовлення засобів. Система оцінювання звітів інших користувачів тут дещо спрощена, оскільки щодо кожного такого звіту можна тільки вказати, чи був він корисним, а тоді за

кожним звітом відображається сумарна кількість висловлювань та кількість таких, що вважають відповідний звіт корисним. Тобто у випадку iHerb використовується більш широка модель, яка передбачає в тому числі активне негативне висловлення, у випадку PureFormulas звіти підтримуються тільки в позитивний бік. Можна або підтримати звіт, або ні. Надається можливість створювати список улюблених засобів, але інших списків створювати не можна. Окрім того надається можливість пошуку з врахуванням параметрів, а не тільки за назвою, зокрема можна задати кількісну категорію виставленої оцінки за звітами користувачів.

Загалом слід відзначити, що iHerb надає ширшу кількість засобів роботи порівняно з PureFormulas, тому в якості прототипа для виконуваної розробки було обрано саме його.

1.2 Визначення функціональних вимог до програмного забезпечення

Враховуючи проаналізовані аналоги і можливості прототипа, для забезпечення мети роботи програма, що розробляється, має надавати можливість виконувати:

– замовлення засобів для догляду за здоров’ям;

– пошук засобів для догляду за здоров’ям;

– публікація звіту про досвід використання засобу для догляду за здоров’ям;

– визначення реакції на звіт іншого користувача;

– відображення звітів за засобом за заданою категорією середньої оцінки;

– внесення і керування даними про засоби для догляду за здоров’ям;

– визначення фасувань засобів для догляду за здоров’ям;

– організація списків з засобів для догляду за здоров’ям, включаючи створення таких списків, додавання засобів до списків, вилучення їх зі списків;

– формулювання запитання про засіб для догляду за здоров’ям;

– формулювання відповіді на запитання іншого клієнта;

– визначення реакції на відповідь на запитання іншим клієнтом;

– керування замовленнями засобів;

– перегляд власних замовлень з можливістю скасувати замовлення;

– перегляд засобів за виробником;

– перегляд засобів за категорією;

– перегляд категорій за групою;

– пошук засобів за ідентифікатором або за universal product code (UPC)-номером для менеджера;

– внесення даних про кількісну наявність засобів.

Сформульовані вимоги до програми було використано для визначення сценаріїв роботи користувачів з програмою (рис. 1.3).

Серед основних ролей користувачів потрібно виділити неавторизованого користувача, менеджера та клієнта.

Під клієнтом розуміється авторизований користувач, основною діяльністю якого є замовлення засобів для догляду за здоров’ям. Йому надаються сценарії роботи з власними замовленнями, замовлення засобів, роботи зі звітами шляхом створення власних та реагування на звіти інших клієнтів.

Під менеджером розуміється представник сервісу, який забезпечує підтримку замовлень клієнтів, керування ними та надання всіх потрібних даних про засоби, які продаються сервісом. При цьому врато звернути увагу, що менеджер не бере участь в роботі над створенням звітів, формуванням відповідей на запитання, все це належить саме до клієнтської взаємодії.

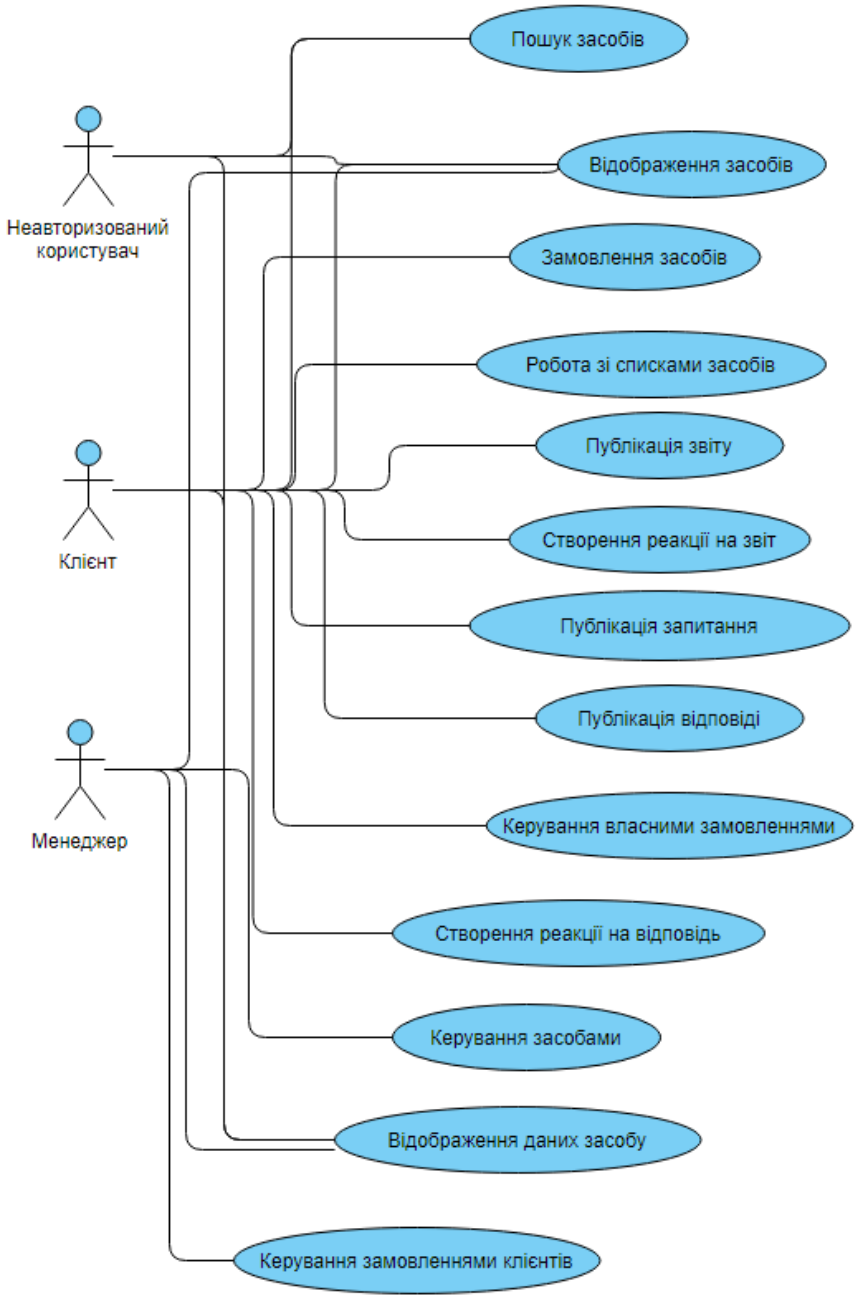

Рисунок 1.3 – Діаграма прецедентів

1.3 Висновки за розділом 1

Аналіз предметної області було виконано за програмними засобами iHerb і PureFormulas, за якими було виділено основні функціональні особливості, які в результаті було покладено в основу ПЗ, що розробляється. Промодельовано основні сценарії використання програми неавторизованим

користувачем, клієнтом та менеджером з представленням відповідної діаграми прецедентів.

Завдання роботи:

– виконати аналіз програмних засобів керування замовленнями засобів для догляду за здоров'ям;

– виконати проєктування ПЗ;

– реалізувати ПЗ;

– створити новий метод класифікації замовлень засобів для догляду за здоров’ям;

– виконати експериментальне дослідження використання створеного методу.

2 МАТЕРІАЛИ І МЕТОДИ

У даному розділі розглядається задача класифікації замовлень засобів для догляду за здоров’ям щодо можливості повернення їх після доставки замовлення клієнту, якщо він за ним не прибув або відмовився від отримання.

Однією з основних характеристик засобу для догляду за здоров’ям є його назва, яка зазвичай включає певний короткий опис засобу. Для того щоб обробити його, перетворивши на форму, яку здатні сприймати методи класифікації даних, потрібно визначити відповідний метод оброблення природної мови.

2.1 Методи оброблення природної мови

Оброблення природної мови є одним із компонентів інтелектуальної автоматизації – набору пов’язаних технологій, які дозволяють комп’ютерам автоматизувати роботу зі знаннями та підвищити продуктивність людей, які працюють зі своїм розумом, іншими компонентами інтелектуальної автоматизації є комп’ютерний зір (інтерпретація зображень і відео, наприклад, у безпілотних автомобілях або медичній діагностиці), мислення та навчання (наприклад, розвиток стратегій і прийняття рішень на основі даних) і виконання (взаємодія з фізичним світом або з наявним програмним забезпеченням, а також об’єднання інших можливостей разом в автоматизовані конвеєри) [7].

У машинному навчанні потрібно перетворити текст у вектор, щоб можна було виконувати деяке моделювання та аналіз машинного навчання, одним з підходів до реалізації даного процесу є оцінка term frequency (TF) – inverse document frequency (IDF):

– оцінка TF означає частоту слова, що представляє собою обчислення кількості входжень заданого слова в даний документ, поділену на кількість слів у документі;

– оцінка IDF визначає обернену частоту документа і обчислюється як відношення кількості документів до кількості документів, у яких зустрічається задане слово, після чого отримане значення передається в логарифмічну функцію;

– оцінка TF-IDF обчислюється шляхом множення оцінок TF і IDF;

– оцінка TF-IDF надає важливіше значення рідкісним словам;

– оцінка TF-IDF надає менш важливе значення словам, що дуже часто зустрічаються;

– оцінка TF-IDF не розглядає семантичне значення [8].

2.2 Дерева рішень

Дерево рішень визначається потоком, який починається з однієї основної ідеї, а потім розгалужується на основі наслідків рішень, при цьому така структура називається деревом рішень, оскільки модель зазвичай виглядає як дерево з гілками, має застосування для:

– аналізу, який передбачає візуальне окреслення потенційних результатів, витрат і наслідків складного рішення;

– обчислення очікуваної цінності кожного результату на основі рішень і наслідків, які до нього призвели, потім, порівнявши результати один з одним, можна швидко оцінити найкращий спосіб дій;

– вирішення проблем, управління витратами та виявлення можливостей [9].

Дерево рішень містить такі символи:

– альтернативні гілки – це дві лінії, які відходять від одного рішення в дереві рішень: ці гілки показують два результати або рішення, які випливають із початкового рішення щодо дерева;

– вузли рішення, які позначаються квадратами і представляють рішення, яке приймається в дереві: кожне дерево рішень починається з вузла прийняття рішень;

– вузли шансів – це кола, які показують декілька можливих результатів;

– кінцеві вузли – це трикутники, які показують кінцевий результат [9].

Перевагами дерев рішень є:

– дерева рішень прості для розуміння, реалізації та візуалізації;

– дерева рішень стійкі до відсутніх значень і викидів, єдиним винятком можуть бути відхилення в результаті в задачах регресії;

– дерева рішень є успішним способом віддзеркалення людського прийняття рішень [10].

Мінусами дерев рішень є:

– схильність до перенавчання, особливо коли дерево є особливо глибоким, що змушує модель вивчати набір даних з високою точністю замість того, щоб вивчати різні шаблони, які лежать за набором даних: висока точність в навчальному наборі даних вводять в оману, оскільки точність тестового набору даних буде нижчою;

– менше дерево з меншою кількістю розділень може призвести до меншої дисперсії та кращої інтерпретації ціною невеликого упередження;

– невеликі зміни в наборі даних можуть призвести до іншого дерева рішень [10].

Одним з алгоритмів, які використовуються для побудови дерев рішень, є алгоритм classification and regression tree (CART).

У дереві рішень вузли розбиваються на підвузли на основі порогового значення атрибута, а алгоритм CART робить це шляхом пошуку найкращої однорідності для підвузлів, використовуючи критерій Gini Index для задачі класифікації, коли кореневий вузол береться як навчальний набір і розбивається на два, враховуючи найкращий атрибут і порогове значення, крім того підмножини також розбиваються з використанням тієї ж логіки, це

триває до тих пір, поки на дереві не буде знайдено останню чисту підмножину або максимальну кількість листків у цьому зростаючому дереві, це також відомо як обрізка дерев [3], [11].

Переваги алгоритму CART:

– алгоритм непараметричний, тому він не залежить від інформації з певного виду розподілу;

– поєднує обидва тести з набором тестових даних і перехресну перевірку, щоб точніше виміряти відповідність;

– дозволяє використовувати одні й ті ж змінні багато разів у різних областях дерева – цей навик здатний виявити складні взаємозалежності між групами змінних.

– викиди у вхідних змінних не мають суттєвого впливу на CART: можна послабити зупиняючі обмеження, щоб дозволити деревам рішень зростати, а потім обрізати дерево до його ідеального розміру, цей метод зменшує ймовірність втрати важливої структури в наборі даних через занадто швидке завершення;

– щоб вибрати вхідний набір змінних, CART можна використовувати в поєднанні з іншими алгоритмами прогнозування [12].

Кроки для виконання алгоритму CART:

–вхідні змінні та точки розщеплення вибираються за допомогою жадібного алгоритму: побудова бінарного дерева рішень – це техніка розподілу вхідного простору, заздалегідь визначена кінцева умова, така як мінімальна кількість навчальних прикладів, наданих кожному листковому вузлу дерева, використовується для зупинки будівництва дерева, при цьому вхідний простір розділяється за допомогою жадібного підходу Greedy (відомий як рекурсивне двійкове розщеплення, це числовий метод, у якому всі значення вирівнюються, а декілька точок розщеплення випробовуються й оцінюються за допомогою функції вартості, при цьому вибирається розподіл з найменшою вартістю)

– застосування кроку перевірки необхідності зупинки, яким найчастіше є використання мінімальної кількості навчальних даних, виділених кожному листовому вузлу (якщо кількість менша за певний поріг, розділення відхиляється, і вузол вважається останнім листовим вузлом, кількість учасників навчання коригується відповідно до набору даних, визначаючи, наскільки точно дерево буде відповідати навчальним даним);

– обрізка дерев: складність дерева рішень визначається як кількість розділень у дереві, рекомендуються дерева з меншою кількістю гілок, вони прості для сприйняття і менш схильні до групування даних, найшвидший і найпростіший підхід до обрізання є найшвидшим і найпростішим підходом до обробки кожного листового вузла в дереві та оцінки ефекту від його видалення за допомогою тестового набору затримки, лише листові вузли усуваються, якщо функція загальної вартості для повного тестового набору зменшується, якщо ніяких додаткових покращень не можна досягти, то більше не слід видаляти вузли [12].

Порівняння алгоритмів побудови дерев рішень представлено в табл. 2.1.

Таблиця 2.1 – Порівняння алгоритмів навчання дерев рішень

Характеристика	ID3	CART
Задачі, які розв’язуються	Класифікація	Класифікація, регресія
Типи даних, з якими може працювати	Категорійні	Категорійні, числові
Час виконання	Повільніший	Швидший

За результатами порівняння було обрано алгоритм CART, адже він, по-перше, швидший в роботі, забезпечує роботу в різних формах, що дозволить підтримувати ознаки різних типів, дозволяє розв’язувати ще й

задачі прогнозування, що може бути важливим, якщо перевести задачу, що розглядається, з часом до задачі прогнозування часу повернення.

2.3 Метод класифікації замовлень засобів для догляду за здоров'ям

Метод класифікації замовлень засобів для догляду за здоров'ям направлений на визначення замовлень, які будуть повернуті клієнтами. До таких належать замовлення, які не прийшов забрати замовник, замовлення, від яких замовник відмовився у пункту отримання служби доставки або замовлення, за якими клієнт скористався своїм правом повернути замовлення.

На першому етапі роботи методу має бути сформовано вибірку даних, яка повинна включати екземпляри, для яких вихідною ознакою є клас щодо відмови клієнта від отримання замовлення, а вхідними ознаками є:

– статус сплати замовлення (передплачене чи ні);

– наявність відмітки для кур'єра;

– вартість замовлення;

– кількість замовлених засобів;

– назва засобу;

– кількість діб, яка пройшла з моменту останньої відмови від отримання такого засобу для догляду за здоров'ям.

Кожне замовлення, що складається з декількох засобів має бути перетворене на окремі замовлення.

На другому етапі виконується оброблення текстових даних з вилученням зайвих слів, зайвих символів, обчисленням оцінок TF-IDF.

На третьому етапі виконується навчання дерева рішень на основі алгоритму CART.

На четвертому етапі виконується класифікація нових замовлень засобів для догляду за здоров'ям.

2.4 Висновки за розділом 2

Розглянуто методи оброблення природної мови на основі формування оцінок TF-IDF, дерева рішень, що можуть використовуватися для класифікації.

Запропоновано метод класифікації замовлень засобів для догляду за здоров’ям, який виконує класифікацію на основі опису засобу та часу останнього повернення засобу з використанням дерева рішень на основі алгоритму CART, що дозволяє керувати замовленнями щодо подальшої їх підтримки.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір мови програмування

Вибір мови для розробки ПЗ було зведено до порівняння мов Python і C# (табл. 3.1), враховуючи актуальний стан засобів веброзробки [13]-[14]. Важливість даного етапу полягає в тому, що це концептуальне рішення впливає безпосередньо на наступний вибір потрібних засобів, а відповідно і на ефективність роботи над проєктом, яка у даному випадку критично оцінюється часом виконання при забезпеченні потрібних результатів роботи.

Таблиця 3.1 – Порівняння мов програмування для реалізації ПЗ замовлення засобів для догляду за здоров’ям

Характеристика	C#	Python
Підтримка вебфреймворків	Так	Так
Підтримка об’єктно-реляційного відображення	Так	Так
Підтримка динамічної типізації	Ні	Так
Легкість серверного розгортання	Відносно	Так
Зручність роботи з масивами даних	Відносно	Так

Основними характеристиками мов програмування, на основі яких виконувалось порівняння, визначено:

– підтримка вебфреймворків: сучасна розробка без вебфреймворків в умовах недостатнього часу на створення проєкту неможлива, адже у зворотному випадку час, витрачений на розробку, значно збільшується, а якість рішень швидше за все зменшиться;

– підтримка об’єктно-реляційного відображення: підтримка об’єктно-реляційного відображення надає можливість створювати менше коду для роботи з даними, уникати прописування кожного разу запитів до бази даних;

– підтримка динамічної типізації: динамічна типізація в даному випадку – це фактор прискорення розробки;

– легкість серверного розгортання: час на супутні роботи має бути за можливості зменшений;

– зручність роботи з масивами даних: прочитані з бази даних зазвичай представляють в тій чи інший формі масив даних, який потрібно певним чином обробляти (за певними параметрами), тому якщо такі засоби зручно використовувати, то це спростить знову ж таки сам процес розробки.

За даними характеристиками при порівнянні мов програмування можна виділити наступне:

– мови C# і Python надають доступ до сучасних вебфреймворків, відповідно ASP.NET Core і Django, кожний з яких є актуальним інструментом на практиці;

– мови C# і Python надають через ці вебфреймворки можливості організації об’єктно-реляційного відображення зручним чином;

– саме Python в повній мірі дозволяє використовувати динамічну типізацію і не оголошувати змінні, визначати типи потрібно тільки під час створення моделей програмного застосунку;

– вебсервер на платформі .NET є дуже потужним рішенням, проте його налаштування може бути початково достатньо складним процесом, який потребує додаткового часу на налаштування, у той же час проєкти Python і Django навпаки направлені на те, щоб якомога швидше відбувся перший запуск;

– мова Python відома своїми засобами роботи з даними, оброблення цих даних: отримані в результаті масиви дуже легко обробити і витягти з них будь-яку частину, величезний набір відповідних функцій значно спрощує цей процес.

3.2 Проєктування бази даних

Проєктування бази даних виконувалось за допомогою системи керування базами даних MySQL. Для проєктування було проаналізовано організацію інформації в застосунках, проаналізованих у першому розділі, було також проаналізовано визначені сценарії використання програми, що дозволило виділити основні сутності та перевести їх до форми реляційного представлення.

Таблиця бази даних Manufacturer представляє виробника засобів для догляду за здоров’ям, включає такі стовпці:

– id – ідентифікатор виробника засобів для догляду за здоров’ям;

– name – назва виробника.

Таблиця HealthCareProduct представляє засіб для догляду за здоров’ям, включає такі стовпці:

– id – ідентифікатор засобу для догляду за здоров’ям;

– name – назва засобу для догляду за здоров’ям;

– description – опис засобу для догляду за здоров’ям;

– score – оцінка засобу для догляду за здоров’ям;

– state – стан засобу (деякі засоби з часом можуть зникати з роботи сервісу);

– manufacturer – виробник засобу для догляду за здоров’ям, є зовнішнім ключем з таблиці Manufacturer за полем id;

– creation – дата і час внесення засобу для догляду за здоров’ям до бази даних;

– photos – адреса папки з файлами зображень, які представляють цей засіб для догляду за здоров’ям;

– category – категорія, до якої належить засіб для догляду за здоров’ям, є зовнішнім ключем з таблиці Category за ключем id;

– returned – дата і час останнього повернення цього засобу для догляду за здоров’ям після його придбання.

Зв’язок між таблицями Manufacturer і HealthCareProduct має тип один-до-багатьох, враховуючи, що один виробник може мати багато засобів, проте кожен засіб виробляється тільки одним виробником.

Зв’язок між таблицями Category і HealthCareProduct має тип один-до-багатьох, враховуючи, що кожна категорія може мати багато засобів, проте кожен засіб належить тільки одній категорії.

Таблиця HealthCareProductVariant представляє варіанти фасування засобу для догляду за здоров’ям, включає такі стовпці:

– id – ідентифікатор фасування засобу для догляду за здоров’ям;

– variant – назва даного фасування;

– quantity – кількість мінімальних одиниць, які вміщує фасування (наприклад, кількість пігулок в банці);

– left – кількість одиниць даного засобу в наявності у варіанті цього фасування;

– expirationdate – дата, до якої придатна дата партія засобів для догляду за здоров’ям;

– upc – UPC, що відповідає цьому фасуванню засобу для догляду за здоров’ям;

– price – поточна вартість засобу для догляду за здоров’ям у цьому фасуванні;

– stableprice – стандартна вартість засобу для догляду за здоров’ям у цьому фасуванні;

– weight – вага;

– dimensions – розміри;

– hcproduct – засіб для догляду за здоров’ям, фасування якого представлено, є зовнішнім ключем таблиці HealthCareProduct за її ключем id.

Зв’язок між таблицями HealthCareProduct і HealthCareProductVariant має тип один-до-багатьох, враховуючи, що кожен засіб може мати багато фасувань, проте кожне конкретне фасування є не фасуванням загалом, а представляє конкретний засіб.

Таблиця GroupCategory представляє групи категорій засобів для догляду за здоров’ям, включає такі стовпці:

– id – ідентифікатор групи категорій;

– name – назва групи категорій.

Таблиця Category представляє категорії засобів для догляду за здоров’ям, включає такі стовпці:

– id – ідентифікатор категорії;

– name – назва категорії;

– groupcategory – група, до якої належить ця категорія, є зовнішнім ключем таблиці GroupCategory за її ключем id.

Зв’язок між таблицями GroupCategory і Category має тип один-до-багатьох, враховуючи, що кожна група категорія може мати багато категорій, проте кожна категорія належить тільки одній групі.

Таблиця Question представляє запитання щодо засобу для догляду за здоров’ям, включає такі стовпці:

– id – ідентифікатор запитання;

– questiontext – текст запитання;

– creation – дата і час створення запитання;

– hcproduct – засіб для догляду за здоров’ям, щодо якого поставлено це запитання, є зовнішнім ключем з таблиці HealthCareProduct за ключем id;

– customer – клієнт, який поставив це запитання, є зовнішнім ключем з таблиці Customer.

Зв’язок між таблицями HealthCareProduct і Question має тип один-до-багатьох, враховуючи, що кожен засіб може мати багато запитань стосовно нього, проте кожне запитання стосується конкретного засобу, адже запитання щодо різних засобів не можуть бути створені в системі.

Зв’язок між таблицями Customer і Question має тип один-до-багатьох, враховуючи, що кожен клієнт може мати багато запитань стосовно різних засобів, проте кожне запитання створюється тільки одним клієнтом.

Таблиця Answer представляє відповіді на запитання клієнтів клієнтами, включає такі стовпці:

– id – ідентифікатор відповіді;

– customer – клієнт, який написав відповідь, є зовнішнім ключем таблиці Customer;

– question – запитання, до якого написано відповідь, є зовнішнім ключем таблиці Question;

– answertext – текст відповіді;

– creation – дата створення відповіді на запитання;

– positive – кількість позитивних реакцій на відповідь від інших клієнтів;

– negative – кількість негативних реакцій на відповідь від інших клієнтів.

Зв’язок між таблицями Customer і Answer має тип один-до-багатьох, враховуючи, що кожен клієнт може мати багато відповідей на різні запитання інших клієнтів, проте кожна відповідь створюється тільки одним клієнтом.

Зв’язок між таблицями Question і Answer має тип один-до-багатьох, враховуючи, що кожне запитання може мати багато відповідей на нього, проте кожна відповідь належить тільки одному запитанню.

Таблиця AnswerReaction представляє реакцію на відповідь на запитання, включає такі стовпці:

– int – ідентифікатор відповіді на запитання, якій відповідає ця реакція;

– creation – дата і час виникнення реакції на відповідь на запитання;

– answer – відповідь на запитання, до якої належить ця реакція, є зовнішнім ключем таблиці Answer;

– customer – клієнт, який створив цю реакцію, є зовнішнім ключем таблиці Customer;

– positive – логічне значення, що визначає, якою була ця реакція: позитивною або негативною.

Зв’язок між таблицями Customer і AnswerReaction має тип один-до-багатьох, враховуючи, що кожен клієнт може мати багато реакцій на відповіді різних інших клієнтів, проте кожна реакція відповідає тільки одному клієнту.

Зв’язок між таблицями Answer і AnswerReaction має тип один-до-багатьох, враховуючи, що кожна відповідь може мати багато реакцій від інших клієнтів, проте кожна реакція відповідає тільки одній конкретній відповіді.

Таблиця ReportReaction представляє реакцію клієнта на звіт про засіб для догляду за здоров’ям від іншого клієнта, включає такі стовпці:

– int – ідентифікатор реакції на звіт;

– creation – дата і час створення звіту;

– report – звіт, якого стосується ця реакція, є зовнішнім ключем таблиці Report;

– customer – клієнт, який створив звіт, є зовнішнім ключем таблиці Customer;

– positive – логічне значення, що визначає, якою була ця реакція: позитивною або негативною.

Зв’язок між таблицями Customer і ReportReaction має тип один-до-багатьох, враховуючи, що кожен клієнт може мати багато реакцій на звіти інших клієнтів, проте кожна реакція створюється тільки одним клієнтом.

Зв’язок між таблицями Report і ReportReaction має тип один-до-багатьох, враховуючи, що кожен звіт може мати багато реакцій від інших клієнтів, проте кожна реакція відповідає тільки одному звіту.

Таблиця Customer представляє клієнта, включає такі стовпці:

– id – ідентифікатор клієнта;

– username – ім’я користувача, що має клієнт;

– password – пароль клієнта;

– photo – шлях до файлу фотографії клієнта;

– registration – дата і час реєстрації клієнта;

– last – дата і час останнього відвідування вебсайту клієнтом;

– reports – кількість звітів, створена клієнтом;

– answers – кількість відповідей на запитання, створена клієнтом;

– questions – кількість запитань, створених клієнтом.

Таблиця Report представляє безпосередньо звіт клієнта щодо досвіду використання засобу для догляду за здоров'ям, включає такі стовпці:

– id – ідентифікатор звіта клієнта щодо досвіду використання засобу для догляду за здоров'ям;

– hcproduct – засіб для догляду за здоров'ям, щодо якого написаний звіт, є зовнішнім ключем таблиці HealthCareProduct;

– bought – логічне значення, яке визначає, чи придбав до написання звіту клієнт цей засіб через програму, чи ні;

– review – текст звіту;

– creation – дата і час створення звіту;

– customer – клієнт, який створив звіт, є зовнішнім ключем таблиці Customer;

– score – оцінка, виставлена засобу для догляду за здоров'ям клієнтом;

– positive – кількість позитивних реакцій на звіт;

– negative – кількість негативних реакцій на звіт.

Зв'язок між таблицями HealthCareProduct і Report має тип один-до-багатьох, враховуючи, що кожен засіб може мати багато звітів від інших клієнтів, проте кожен звіт відповідає тільки одному засобу для догляду за здоров'ям.

Зв'язок між таблицями Customer і Report має тип один-до-багатьох, враховуючи, що кожен клієнт може мати багато звітів про різні засобу для догляду за здоров'ям, проте кожен звіт створений тільки одним клієнтом.

Таблиця HealthCareListProducts представляє реалізацію зв'язку багато до багатьох між таблицями HealthCareList і HealthCareProduct, включає такі стовпці:

– id – ідентифікатор;

– healthcarelist – перелік, до якого внесено засіб для догляду за здоров’ям, є зовнішнім ключем з таблиці HealthCareList;

– healthcareptoduct – засіб для догляду за здоров’ям, є зовнішнім ключем з таблиці HealthCareProduct.

Таблиця Order (рис. 3.1) представляє замовлення на придбання клієнтом засобів для догляду за здоров’ям, включає такі стовпці:

– id – ідентифікатор замовлення;

– customer – клієнт, який створив замовлення, є зовнішнім ключем таблиці Customer:

– creation – дата і час створення замовлення;

– price – сумарна вартість замовлення;

– discountprice – знижена вартість замовлення, яку мав сплатити клієнт;

– address –адреса доставки замовлення;

– payed – логічне значення, що визначає, чи було сплачено замовлення;

– status – стан замовлення;

– comment – коментар від клієнта до оформлення замовлення;

– delivered – дата і час доставки замовлення клієнту;

– returned – дата і час повернення замовлення;

– possiblereturn – логічна змінна, яка визначає результат обчислення представленим методом щодо того, чи очікується повернення даного замовлення клієнтом.

Таблиця OrderProduct представляє склад замовлення щодо конкретного засобу в заданому фасуванні, включає такі стовпці:

– id – ідентифікатор;

– order – замовлення, до якого належить ця позиція, є зовнішнім ключем з таблиці Order;

– variant – фасування засобу для догляду за здоров’ям, яке бажає придбати клієнт, є зовнішнім ключем з таблиці HealthCareProductVariant;

– price – вартість цієї позиції;

– quantity – кількість одиниць засобу для догляду за здоров’ям у цьому фасуванні.

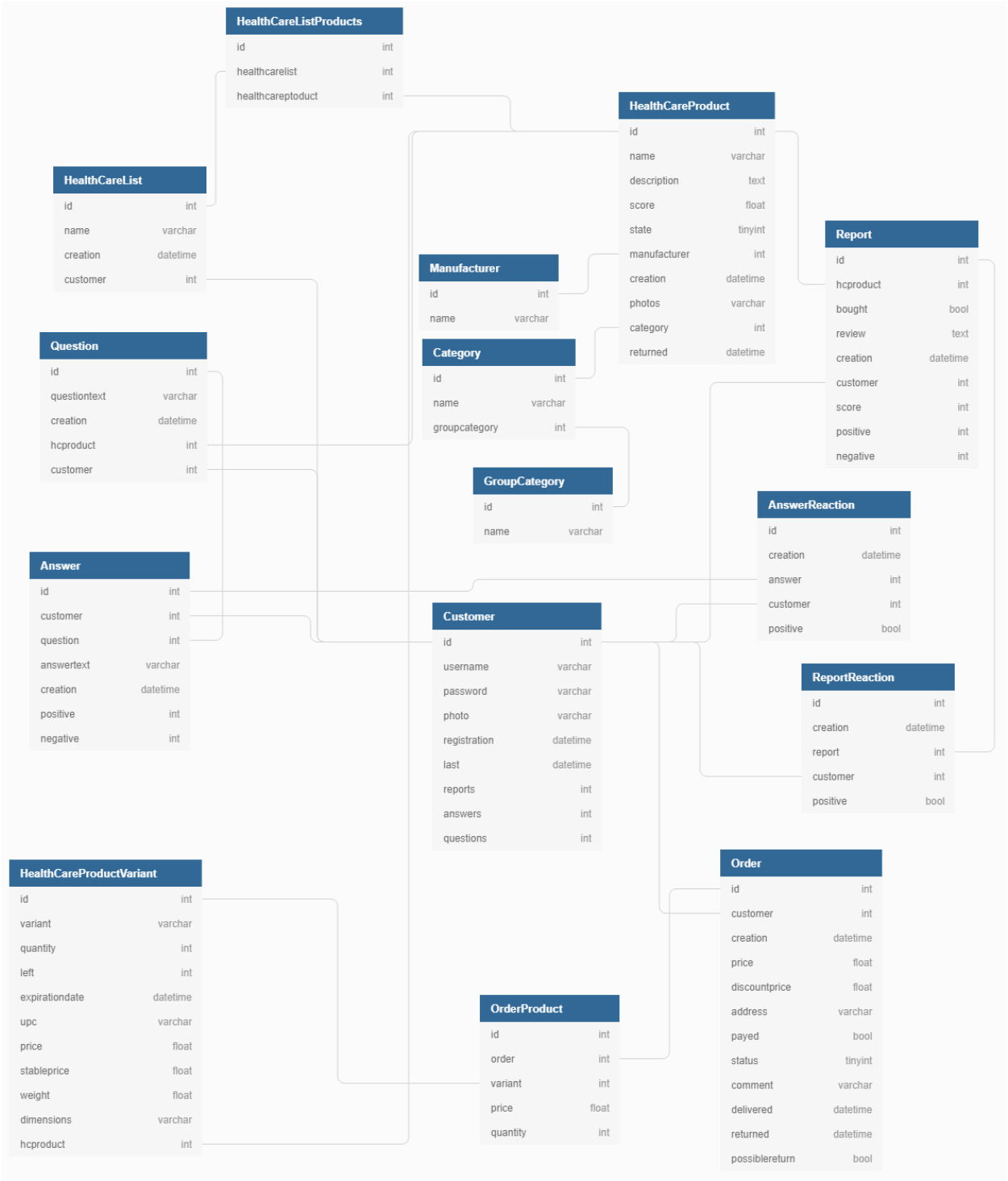

Рисунок 3.1 – Схема бази даних

Зв’язок між таблицями HealthCareProductVariant і OrderProduct має тип один-до-багатьох, враховуючи, що кожна позиція замовлення може відповідати тільки одному конкретному фасуванню засобу для догляду за здоров’ям, проте кожне фасування засобу для догляду за здоров’ям може бути представлене в різних замовленнях.

Зв’язок між таблицями Order і OrderProduct має тип один-до-багатьох, враховуючи, що кожне замовлення може мати багато позицій з різними фасуваннями різних або тих самих засобів, проте кожне фасування може зустрічатися в замовленні тільки один раз, у протилежному випадку позиції об’єднуються в одну і значення кількості просто додаються.

Таблиця HealthCareList представляє список засобів для догляду за здоров’ям, створений клієнтом для власного перегляду, включає стовпці:

– id – ідентифікатор списку;

– name – назва списку;

– creation – дата і час створення замовлення;

– customer – клієнт, який створив даний список, є зовнішнім ключем з таблиці Customer.

Зв’язок між таблицями Customer і HealthCareList має тип один-до-багатьох, враховуючи, що кожне замовлення може мати багато позицій з різними фасуваннями різних або тих самих засобів, проте кожне фасування може зустрічатися в замовленні тільки один раз, у протилежному випадку позиції об’єднуються в одну і значення кількості просто додаються.

Це повний набір таблиць та зв’язків між ними.

3.3 Структура програмного забезпечення

Структуру ПЗ (рис. 3.2) було розроблено на основі використання вебфреймворку Django. Окремо у складі програмної системи виділено підсистему класифікації замовлень, яка має визначити, чи дане замовлення буде повернуто, чи ні.

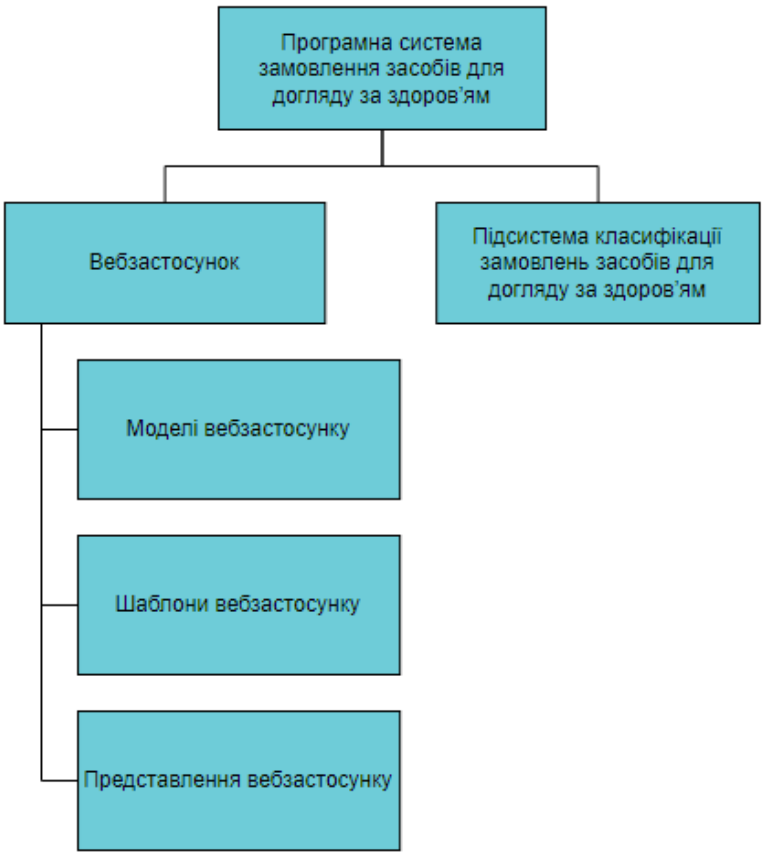

Рисунок 3.2 – Структурна схема програмного проєкту

3.4 Висновки за розділом 3

Порівняння мов програмування C# і Python для реалізації ПЗ призвело до вибору мови Python. Мова Python разом з вебфреймворком стали засобами розробки. Детально виконано опис результатів проєктування бази даних. Створено структуру програми з приведенням структурної схеми.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Логіка функціонування програмної системи

Зважаючи на те, що вся робота програмної системи направлена на підтримку замовлень, а запропонований метод визначає можливість повернення замовлення, то саме на прикладі стану замовлення було проілюстровано логіку роботи з програмою менеджерів та клієнтів (рис. 4.1).

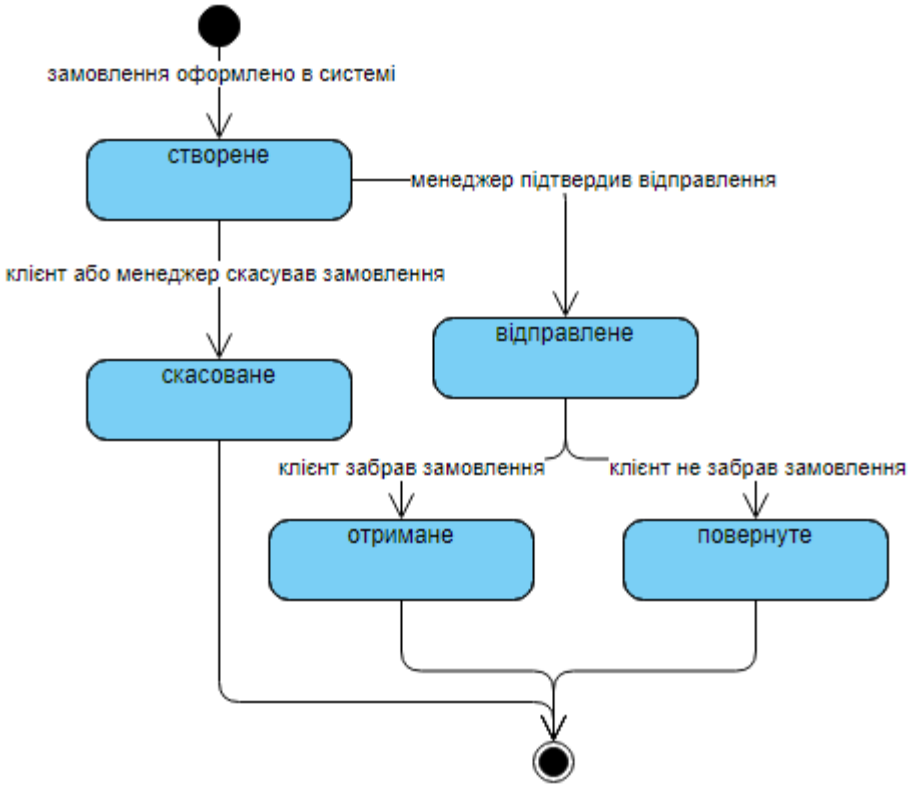

Рисунок 4.1 – Діаграма станів замовлення засобів для догляду за здоров’ям

4.2 Опис розроблених класів програмної системи

Клас моделі Manufacturer забезпечує роботу з об’єктами виробника засобів для догляду за здоров’ям, містить окрім стандартно створюваного ідентифікатора за атрибутом id також атрибут name, що є назвою виробника і представляє собою поле типу CharField. Даний клас також включає метод

__str__(self), який повертає текстове представлення виробника за його назвою, та метод get_absolute_url(self), який повертає абсолютний шлях до сторінки з цим виробником.

Клас моделі HealthCareProduct забезпечує роботу з об’єктами засобів для догляду за здоров’ям на основі наступних атрибутів класу:

– id – стандартний атрибут ідентифікатора засобу для догляду за здоров’ям, формується автоматично;

– name, визначений як текстове поле TextField, містить назву засобу для догляду за здоров’ям;

– description, визначений як дійсне поле FloatField, містить опис засобу для догляду за здоров’ям;

– score, визначений як цілочисельне поле малого розміру SmallIntegerField, містить оцінку засобу для догляду за здоров’ям;

– state, визначений як поле зовнішнього ключа ForeignKey, містить стан засобу;

– manufacturer, визначений як поле дати і часу DateTimeField, містить виробника засобу для догляду за здоров’ям;

– creation, визначений як символьне поле CharField, містить дату і час внесення засобу для догляду за здоров’ям до бази даних;

– photos, визначений як символьне поле CharField, містить адресу папки з файлами зображень, які представляють цей засіб для догляду за здоров’ям;

– category, визначений як поле зовнішнього ключа ForeignKey, містить категорію, до якої належить засіб для догляду за здоров’ям;

– returned, визначений як поле дати і часу DateTimeField, містить дату і час останнього повернення цього засобу для догляду за здоров’ям після його придбання.

Даний клас також включає метод __str__(self), який повертає текстове представлення засобу для догляду за здоров’ям за його назвою, та метод

get_absolute_url(self), який повертає абсолютний шлях до сторінки з цим засобом для догляду за здоров’ям.

Клас моделі HealthCareProductVariant забезпечує роботу з об’єктами варіантів фасування засобу для догляду за здоров’ям на основі наступних атрибутів класу:

– id – стандартний атрибут ідентифікатора фасування засобу для догляду за здоров’ям, формується автоматично;

– variant, визначений як символьне поле CharField, містить назву даного фасування;

– quantity, визначений як цілочисельне поле IntegerField, містить кількість мінімальних одиниць, які вміщує фасування;

– left, визначений як п цілочисельне поле IntegerField, містить кількість одиниць даного засобу в наявності у варіанті цього фасування;

– expirationdate, визначений як поле дати і часу DateTimeField, містить дату, до якої придатна дата партія засобів для догляду за здоров’ям;

– upc, визначений як символьне поле CharField, містить UPC-код, що відповідає цьому фасуванню засобу для догляду за здоров’ям;

– price, визначений як дійсне поле FloatField, містить поточну вартість засобу для догляду за здоров’ям у цьому фасуванні;

– stableprice, визначений як дійсне поле FloatField, містить стандартну вартість засобу для догляду за здоров’ям у цьому фасуванні;

– weight, визначений як дійсне поле FloatField, містить вагу;

– dimensions, визначений як символьне поле CharField, містить розміри;

– hcproduct, визначений як поле зовнішнього ключа ForeignKey, містить засіб для догляду за здоров’ям.

Клас моделі GroupCategory забезпечує роботу з об’єктами груп категорій засобів для догляду за здоров’ям на основі наступних атрибутів класу:

– id – стандартний атрибут ідентифікатора групи категорій, формується автоматично;

– name, визначений як символьне поле CharField, містить назву групи категорій.

Даний клас також включає метод __str__(self), який повертає текстове представлення групи категорії за її назвою, та метод get_absolute_url(self), який повертає абсолютний шлях до сторінки з цією групою категорій.

Клас моделі Category забезпечує роботу з об’єктами категорії засобів для догляду за здоров’ям на основі наступних атрибутів класу:

– id – стандартний атрибут ідентифікатора категорії;

– name, визначений як символьне поле CharField, містить назву категорії;

– groupcategory, визначений як поле зовнішнього ключа ForeignKey, містить групу, до якої належить ця категорія.

Даний клас також включає метод __str__(self), який повертає текстове представлення категорії за її назвою, та метод get_absolute_url(self), який повертає абсолютний шлях до сторінки з цією категорією.

Клас моделі Question забезпечує роботу з об’єктами запитання щодо засобу для догляду за здоров’ям на основі наступних атрибутів класу:

– id – стандартний атрибут ідентифікатора запитання, формується автоматично;

– questiontext, визначений як символьне поле CharField, містить текст запитання;

– creation, визначений як поле дати і часу DateTimeField, містить дату і час створення запитання;

– hcproduct, визначений як поле зовнішнього ключа ForeignKey, містить засіб для догляду за здоров’ям, щодо якого поставлено це запитання;

– customer, визначений як поле зовнішнього ключа ForeignKey, містить об’єкт клієнта, який поставив це запитання.

Даний клас також включає метод __str__(self), який повертає текстове представлення запитання.

Клас моделі Answer забезпечує роботу з об’єктами відповіді на запитання клієнтів клієнтами на основі наступних атрибутів класу:

– id – стандартний атрибут ідентифікатора відповіді, формується автоматично;

– customer, визначений як поле зовнішнього ключа ForeignKey, містить об’єкт клієнта, який написав відповідь;

– question, визначений як поле зовнішнього ключа ForeignKey, містить об’єкт запитання;

– answertext, визначений як символьне поле CharField, містить текст відповіді;

– creation, визначений як поле дати і часу DateTimeField, містить дату створення відповіді на запитання;

– positive, визначений як цілочисельне поле IntegerField, містить кількість позитивних реакцій на відповідь від інших клієнтів;

– negative, визначений як цілочисельне поле IntegerField, містить кількість негативних реакцій на відповідь від інших клієнтів.

Даний клас також включає метод __str__(self), який повертає текстове представлення запитання.

Клас моделі AnswerReaction забезпечує роботу з об’єктами реакцію на відповідь на запитання, на основі наступних атрибутів класу:

– id – стандартний атрибут ідентифікатора відповіді на запитання, якій відповідає ця реакція, формується автоматично;

– creation, визначений як поле дати і часу DateTimeField, містить дату і час виникнення реакції на відповідь на запитання;

– answer, визначений як поле зовнішнього ключа ForeignKey, містить об’єкт відповіді на запитання, до якої належить ця реакція;

– customer, визначений як поле зовнішнього ключа ForeignKey, містить об’єкт клієнта, який створив цю реакцію;

– positive, визначений як логічне поле BooleanField, містить значення, що визначає, якою була ця реакція: позитивною або негативною.

Клас моделі ReportReaction забезпечує роботу з об’єктами реакції клієнта на звіт про засіб для догляду за здоров’ям від іншого клієнта на основі наступних атрибутів класу:

– id – стандартний атрибут ідентифікатора реакції на звіт, формується автоматично;

– creation, визначений як поле дати і часу DateTimeField, містить дату і час створення звіту;

– report, визначений як поле зовнішнього ключа ForeignKey, містить об’єкт звіту, якого стосується ця реакція;

– customer, визначений як поле зовнішнього ключа ForeignKey, містить об’єкт клієнта, який створив звіт;

– positive, визначений як логічне поле BooleanField, містить значення, що визначає, якою була ця реакція: позитивною або негативною.

Клас моделі Customer забезпечує роботу з об’єктами клієнта на основі наступних атрибутів класу:

– id – стандартний атрибут ідентифікатора клієнта, формується автоматично;

– username, визначений як символьне поле CharField, містить ім’я користувача, що має клієнт;

– password, визначений як символьне поле CharField, містить пароль клієнта;

– photo, визначений як символьне поле CharField, містить шлях до файлу фотографії клієнта;

– registration, визначений як поле дати і часу DateTimeField, містить дату і час реєстрації клієнта;

– last, визначений як поле дати і часу DateTimeField, містить дату і час останнього відвідування вебсайту клієнтом;

– reports, визначений як цілочисельне поле IntegerField, містить кількість звітів, створена клієнтом;

– answers, визначений як цілочисельне поле IntegerField, містить кількість відповідей на запитання, створена клієнтом;

– questions, визначений як цілочисельне поле IntegerField, містить кількість запитань, створених клієнтом.

Даний клас також включає метод __str__(self), який повертає текстове представлення об’єкта клієнта за його користувацьким іменем, та метод get_absolute_url(self), який повертає абсолютний шлях до сторінки профілю клієнта.

Клас моделі Report забезпечує роботу з об’єктами безпосередньо звіту клієнта щодо досвіду використання засобу для догляду за здоров’ям на основі наступних атрибутів класу:

– id стандартний атрибут ідентифікатора звіту, формується автоматично;

– hcproduct, визначений як поле зовнішнього ключа ForeignKey, містить об’єкт засобу для догляду за здоров’ям;

– bought, визначений як логічне поле BooleanField, містить значення, яке визначає, чи придбав до написання звіту клієнт цей засіб через програму, чи ні;

– review, визначений як текстове поле TextField, містить текст звіту;

– creation, визначений як поле дати і часу DateTimeField, містить дату і час створення звіту;

– customer, визначений як поле зовнішнього ключа ForeignKey, містить об’єкт клієнта, який створив звіт;

– score, визначений як цілочисельне поле IntegerField, містить оцінку, виставлену засобу для догляду за здоров’ям клієнтом;

– positive, визначений як цілочисельне поле IntegerField, містить кількість позитивних реакцій на звіт;

– negative, визначений як цілочисельне поле IntegerField, містить кількість негативних реакцій на звіт.

Даний клас також включає метод __str__(self), який повертає текстове представлення звіту.

Клас моделі Order забезпечує роботу з об’єктами замовлення на придбання клієнтом засобів для догляду за здоров’ям, на основі наступних атрибутів класу:

– id – стандартний атрибут ідентифікатора замовлення, формується автоматично;

– customer, визначений як поле зовнішнього ключа ForeignKey, містить клієнт, який створив замовлення, є зовнішнім ключем таблиці Customer:

– creation, визначений як поле дати і часу DateTimeField, містить дату і час створення замовлення;

– price, визначений як текстове поле FloatField, містить сумарну вартість замовлення;

– discountprice, визначений як дійсне поле FloatField, містить знижену вартість замовлення, яку мав сплатити клієнт;

– address, визначений як символьне поле CharField, містить адресу доставки замовлення;

– payed, визначений як логічне поле BooleanField, містить значення, що визначає, чи було сплачено замовлення;

– status, визначений як невелике цілочисельне поле SmallIntegerField, містить стан замовлення;

– comment, визначений як символьне поле CharField, містить коментар від клієнта до оформлення замовлення;

– delivered, визначений як поле дати і часу DateTimeField, містить дату і час доставки замовлення клієнту;

– returned, визначений як поле дати і часу DateTimeField, містить дату і час повернення замовлення;

– possiblereturn, визначений як логічне поле BooleanField, ймовірність повернення замовлення.

Даний клас також включає метод get_absolute_url(self), який повертає абсолютний шлях до сторінки з цим замовленням.

Клас моделі OrderProduct забезпечує роботу з об’єктами складу замовлення щодо конкретного засобу в заданому фасуванні на основі наступних атрибутів класу:

– id – стандартний атрибут ідентифікатора, формується автоматично;

– order, визначений як поле зовнішнього ключа ForeignKey, містить об’єкт замовлення, до якого належить ця позиція;

– variant, визначений як поле зовнішнього ключа ForeignKey, містить об’єкт фасування засобу для догляду за здоров’ям, яке бажає придбати клієнт;

– price, визначений як дійсне поле FloatField, містить вартість цієї позиції;

– quantity, визначений як цілочисельне поле IntegerField, містить кількість одиниць засобу для догляду за здоров’ям у цьому фасуванні.

Клас моделі HealthCareList забезпечує роботу з об’єктами списку засобів для догляду за здоров’ям, який був створений користувачем для власного перегляду, на основі наступних атрибутів класу:

– id – стандартний атрибут ідентифікатора, формується автоматично;

– name, визначений як символьне поле CharField, містить назву списку;

– creation, визначений як поле дати і часу DateTimeField, містить дату і час створення замовлення;

– healthcarelistproducts, який є полем зв’язку багато-до-багатьох ManyToManyField з моделлю HealthCareProduct і містить перелік об’єктів засобів для догляду за здоров’ям, що входять до переліку.

Даний клас також включає метод get_absolute_url(self), який повертає абсолютний шлях до сторінки цим списком.

Класи представлень структуровано за файлами, що відповідають основним моделям. За кожною моделлю підтримуються засоби відображення, редагування та оновлення даних. Самі інтерфейси вебзастосунку створені на основі шаблону [15].

Клас HealthCareProductView забезпечує представлення перегляду повних даних засобу для догляду за здоров’ям. Заповнення даних представлення реалізується як виділенням відповідного об’єкту з його повною інформацією, так і додаванням до нього всіх звітів інших користувачів, запитань за цим засобом, а за кожним запитанням – відповідей на них. Також до складу даних включаються всі варіанти фасувань цього засобу.

Клас AddHealthCareProductView забезпечує представлення створення нового засобу для догляду за здоров’ям, що реалізується через підтримку відповідної форми, на якій задаються його назва, опис, стан, виробник, шлях до фотографій та категорія.

Клас UpdateHealthCareProductView забезпечує представлення оновлення даних засобу для догляду за здоров’ям, що реалізується через підтримку відповідної форми, на якій задаються його назва, опис, стан, виробник, шлях до фотографій, категорія та дата останнього повернення засобу в межах будь-якого замовлення.

Клас CategoryHealthCareProductView забезпечує представлення перегляду списку всіх засобів для догляду за здоров’ям за категорією, номер якої передається через GET-запит.

Клас ManufacturerHealthCareProductView забезпечує представлення перегляду списку всіх засобів для догляду за здоров’ям за виробником, номер якого передається через GET-запит.

Клас SearchHealthCareProductView забезпечує представлення перегляду списку результатів пошуку всіх засобів для догляду за здоров’ям, які в назві містять частину, значення якої передається через GET-запит.

Клас IdHealthCareProductView забезпечує представлення перегляду засобу для догляду за здоров’ям за його номером, переданим через GET-запит, необхідний для організації роботи менеджера сервісу.

4.3 Висновки за розділом 4

Розглянуто стани, в яких може перебувати замовлення засобів для догляду за здоров’ям у процесі роботи ПЗ, побудовано діаграму станів. Описано основні створені класи з приведенням їх структури та поясненням кожного елементу.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Експлуатація програми

Експлуатація програми може відбуватися через 4 режими роботи: режим неавторизованого користувача, режим клієнта, режим менеджера та режим адміністратора. Останній реалізований через стандартну адміністративну панель Django і доступний з додаванням /admin до адреси вебсайту. Останній використовується для специфічних цілей, що охоплюють редагування і додавання груп категорій, категорій тощо.

Приклад роботи в режимі клієнта приведено на рис. 5.1.

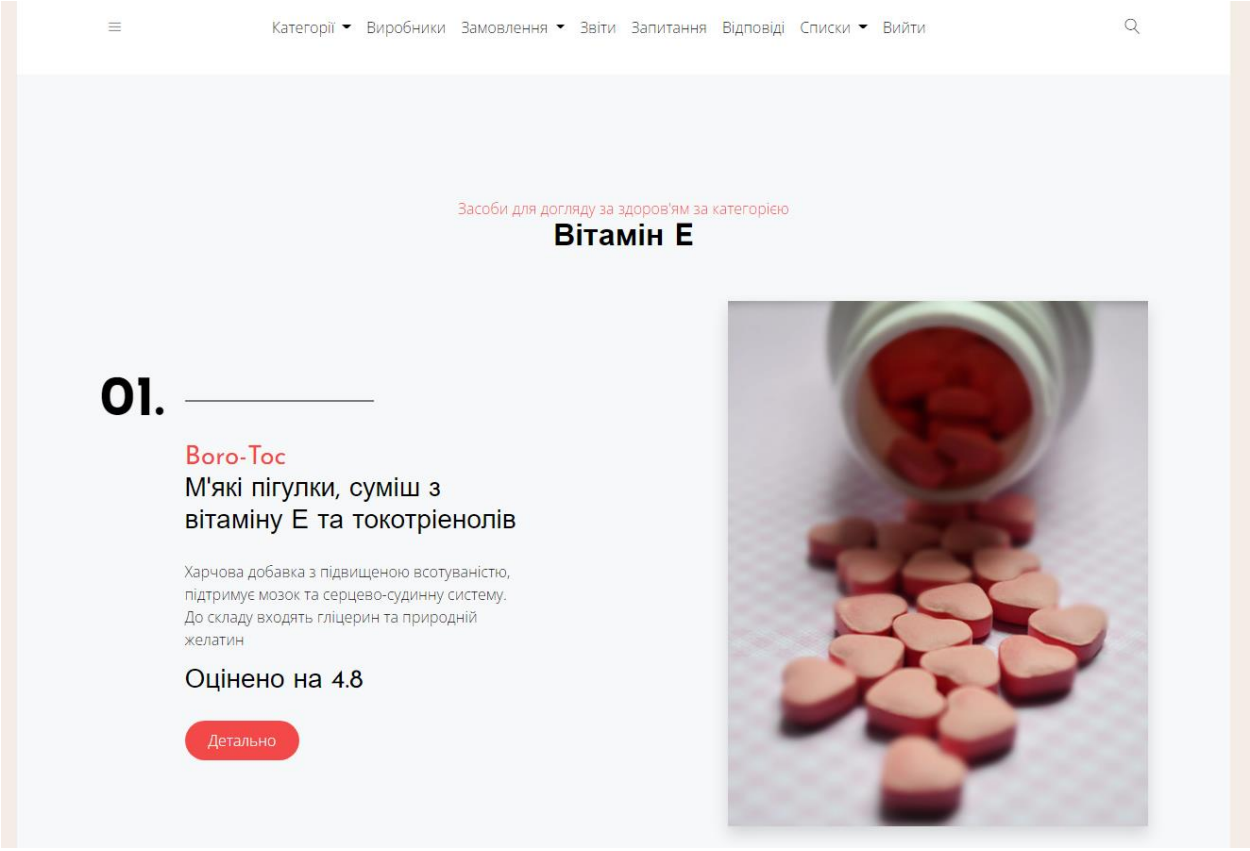

Рисунок 5.1 – Відображення списку засобів за обраною категорією

У даному випадку виводиться список засобів за заданою категорією. По кожному товару доступні його назва, виробник (червоним кольором), середня оцінка та короткий опис, що представляє перший абзац повного

опису. Для відображення детальних даних засобу, включаючи його фасування, звіти за ним, запитання, потрібно натиснути на кнопку «Детально» і відкриється відповідна сторінка.

Клієнту доступне меню, що включає «Категорії» для вибору групи з випадаючого переліку груп категорій і подальшого вибору на відповідній сторінці необхідної категорії, «Виробники» для аналогічного вибору на сторінці, що відкриється, виробника і переходу до його засобів, «Замовлення» для відображення замовлень клієнта, звідки доступний перехід і до поточного замовлення, до якого додано засоби, «Звіти» для перегляду всіх звітів, створених клієнтом і створення нових, «Запитання» для перегляду створених клієнтом запитань, «Відповіді» для перегляду відповідей, створених клієнтом, «Списки» для перегляду засобів за своїм переліком або редагування існуючого переліку.

Приклад роботи менеджера приведено на рис. 5.2.

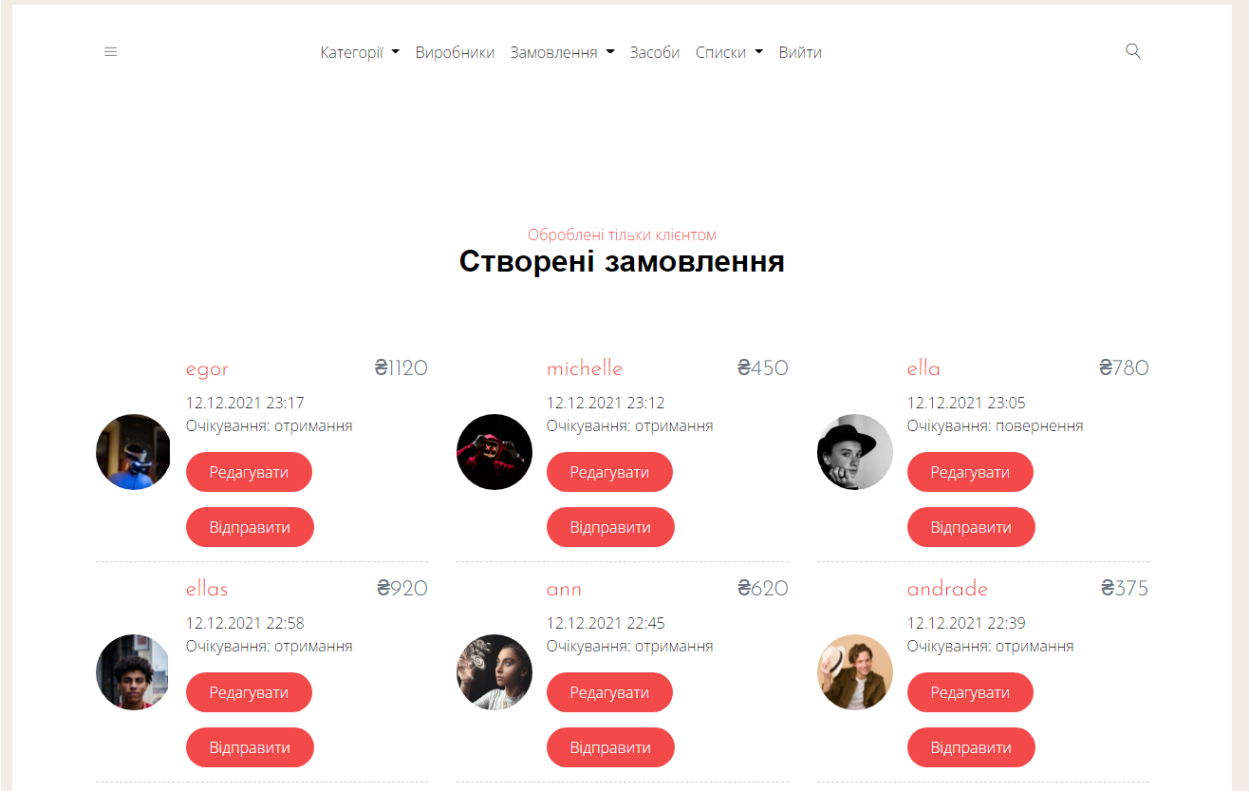

Рисунок 5.2 – Робота менеджера над замовленнями клієнтів

У даному випадку менеджер працює з переліком створених замовлень, тобто таких, які ще не підтверджені, а тільки створені клієнтами. Для кожного замовлення виводиться логін та фотографія користувача (можна перейти до його сторінки, натиснувши логін), дата і час створення, прогнозований клас (повернення або ні), сумарна вартість, а також надаються кнопки для керування замовленнями, що включають «Редагувати» і «Відправити» (в інших вибраних станах замовлень має іншу назву – визначає перехід до наступного стану замовлення).

Менеджеру доступне меню, що включає «Категорії» для вибору групи з випадаючого переліку груп категорій і подальшого вибору на відповідній сторінці необхідної категорії, «Виробники» для аналогічного вибору на сторінці, що відкриється, виробника і переходу до його засобів, «Замовлення» для відображення замовлень у сервісі за заданим станом (рис. 5.2), «Засоби» для перегляду і редагування засобів, наявних у сервісі на продаж, «Списки» для перегляду засобів за своїм переліком або редагування існуючого переліку.

За допомогою пункту «Вийти» користувачі можуть залишити свій обліковий запис.

5.2 Тестування програми

Тестування програми виконано шляхом перевірки виконання функцій програми, які стосуються забезпечення функціональних вимог до програми, сформульованих у першому розділі. Результати тестування для кожного типу користувачів представлено в табл. 5.1.

Усі функції були коректно опрацьовані. У таблиці пропущені значення у клітинках означають, що можливість відсутня для даного користувача.

Таблиця 5.1 – Результати тестування програми

Функція	Неавторизований користувач	Клієнт	Менеджер
Замовлення засобів		+	
Пошук	+	+	+
Публікація звіту		+	
Реакція на звіт		+	
Публікація запитання		+	
Публікація відповіді		+	
Реакція на відповідь		+	
Засоби за категорією	+	+	+
Категорії за групою	+	+	+
Звіти за середньою оцінкою	+	+	+
Внесення засобу			+
Редагування даних засобу			+
Внесення фасувань засобу			+
Пошук засобу за UPC			+
Створення списку		+	+
Змінювання засобів у списку		+	+
Відправлення замовлення			+
Скасування замовлення		+	+
Повернення замовлення			+
Перегляд створених замовлень			+
Перегляд повернених замовлень			+
Перегляд власних замовлень		+	
Засоби за виробником	+	+	+
Перегляд даних засобу	+	+	+

5.3 Експериментальне дослідження класифікації замовлень засобів для догляду за здоров’ям

Експериментальне дослідження класифікації замовлень засобів для догляду за здоров’ям проводилось на основі використання вибірки [16]. Дана вибірка складалась на основі результатів роботи компанії з Індії, що займається продажем засобів для догляду за здоров’ям на основі електронної комерції. Вибірка мала наступні ознаки:

– унікальний номер замовлення;

– ім’я покупця;

– місто, де замовлення було отримано;

– штат, де було отримано замовлення;

– адреса отримання замовлення;

– дата, коли було розміщено замовлення;

– визначення того, чи було повернуто замовлення;

– статус сплати замовлення (передоплачене чи ні);

– відмітки для кур’єра;

– вартість замовлення;

– дата доставки або дата повернення замовлення;

– номер засобу для догляду за здоров’ям;

– категорія засобу;

– кількість замовлених засобів;

– назва засобу.

Дану вибірку в результаті було зведено до простору ознак, описаному в запропонованому методі в результаті передоброблення даних.

Запропонований метод був порівняний за точністю класифікації з методом k-найближчих сусідів. Результати представлені в табл. 5.2. Запропонований метод призвів до отримання точності на 21,96 % кращої за метод k-найближчих сусідів.

Таблиця 5.2 – Результати експериментального дослідження

Метод	Точність
Метод k-найближчих сусідів	58,31
Метод класифікації замовлень засобів для догляду за здоров’ям	80,27

5.4 Висновки за розділом 5

Описано процес експлуатації програмного продукту. Виконано тестування ПЗ, яке підтвердило працездатність розробленого ПЗ з підтвердженням виконання всіх заявлених функцій програми.

Результати проведеного експериментального дослідження дозволили отримати точність класифікації у 80,27 % відносно визначення замовлень, які будуть повернені після відправлення службою доставки покупцю.

ВИСНОВКИ

У роботі розроблено ПЗ замовлення засобів для догляду за здоров’ям для підвищення ефективності роботи менеджерів щодо обслуговування замовлень.

Аналіз предметної області було виконано за програмними засобами iHerb і PureFormulas, за якими було виділено основні функціональні особливості, які в результаті було покладено в основу ПЗ, що розроблялось. Промодельовано основні сценарії використання програми неавторизованим користувачем, клієнтом та менеджером з представленням відповідної діаграми прецедентів.

Порівняння мов програмування C# і Python для реалізації ПЗ призвело до вибору мови Python. Мова Python разом з вебфреймворком Django стали засобами розробки. Детально виконано опис результатів проєктування бази даних. Створено структуру програми з приведенням структурної схеми.

Розглянуто стани, в яких може перебувати замовлення засобів для догляду за здоров’ям у процесі роботи ПЗ, побудовано діаграму станів. Реалізовано ПЗ. Описано основні створені класи з приведенням їх структури та поясненням кожного елементу.

Програма призначена для організації підтримки замовлень користувачами засобів для догляду за здоров’ям відносно створення цих замовлень клієнтами та керування цими замовленнями менеджерами з можливістю класифікації замовлень.

Описано процес експлуатації програмного продукту. Виконано тестування ПЗ, яке підтвердило працездатність розробленого ПЗ з підтвердженням виконання всіх заявлених функцій програми.

Результати проведеного експериментального дослідження дозволили отримати точність класифікації у 80,27 % відносно визначення замовлень, які

будуть повернені після відправлення службою доставки покупцю, на основі запропонованого методу.

Наукова новизна роботи полягає в запропонованому методі класифікації замовлень засобів для догляду за здоров’ям, який виконує класифікацію на основі параметрів замовлення та часу останнього повернення засобу з використанням дерева рішень на основі алгоритму CART, що дозволяє керувати замовленнями щодо подальшої їх підтримки. Проведене експериментальне дослідження застосування даного методу підтвердило можливість його практичного використання у даній сфері.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 5 Key Elements to a Healthy Lifestyle And What You Can Do Today! [Electronic resource]. – Access mode : https://connect.advocare.com/5-key-elements-healthy-lifestyle/.

2. Subbotin, S. Research advance based on big data processing and analysis [Text] / S. Subbotin, A. Oliinyk, S. Leoshchenko / ed. by S. Subbotin. – Zilina : EDIS-Editing Centre of University of Zilina, 2017. – 129 p.

3. Субботін, С. О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей [Текст] : монографія / С. О. Субботін, А. О. Олійник, О. О. Олійник ; під заг. ред. С.О. Субботіна. – Запоріжжя : ЗНТУ, 2009. – 375 с.

4. Oliinyk, A. The decision tree construction based on a stochastic search for the neuro-fuzzy network synthesis [Text] / A. Oliinyk, S. A. Subbotin // Optical Memory and Neural Networks (Information Optics). – 2015. – Vol. 24, № 1. – P. 18–27.

5. iHerb.com [Електрон. ресурс] : вітаміни, добавки й натуральні продукти для здоров'я. – Режим доступу : https://ua.iherb.com/.

6. Health Supplements & Vitamins for Sale [Electronic resource]. – Access mode : https://www.pureformulas.com/.

7. Natural language processing is shaping intelligent automation [Electronic resource]. – Access mode : https://venturebeat.com/2021/12/08/natural-language-processing-is-shaping-intelligent-automation/.

8. Dash, J. TF-IDF (Term Frequency – Inverse Document Frequency) [Electronic resource] / J Dash. – Access mode : https://medium.com/analytics-vidhya/tf-idf-term-frequency-inverse-document-frequency-985e39c03c.

9. What is decision tree analysis? 5 steps to make better decisions [Electronic resource]. – Access mode : https://asana.com/ru/resources/decision-tree-analysis.

10. Dogan, P. Classification and Regression Tree – CART [Electronic

resource] / P. Dogan. – Access mode : https://pinard.medium.com/classification-and-regression-tree-cart-b2d61412b60b.

11. Maimon, O. Z. Data Mining with Decision Trees [Text] : Theory and Applications / O. Z. Maimon, L. Rokach. –

12. Dutta, B. A Classification and Regression Tree (CART) Algorithm [Electronic resource] / B. Dutta. – Access mode : https://www.analyticssteps.com/blogs/classification-and-regression-tree-cart-algorithm.

13. Мартелли, А. Python. Справочник. Полное описание языка [Текст] / А. Мартелли, А. Рейвенскрофт, С. Холден. – М. : Диалектика, 2019. – 896 с.

14. Саммерфилд, М. Программирование на Python 3. Подробное руководство [Текст] / М. Саммерфилд. – М. : Символ-Плюс, 2009. – 608 с.

15. Resto Free Website Template [Electronic resource]. – Access mode : https://www.free-css.com/free-css-templates/page274/resto.

16. E-Commerce Healthcare Orders Dataset [Electronic resource]. – Access mode : https://www.kaggle.com/adishgolechha/ecommerce-healthcare-orders-dataset.

ДОДАТОК А

Технічне завдання

Вступ

Будь-яка система, в якій відбуваються замовлення товарів, потребує інструменту для класифікації замовлень. Для сфери продажу засобів для догляду за здоров’ям важливим є класифікація замовлень за фактом відмови від їх отримання (повернення). Щоб реалізувати такий інструмент, потрібно спочатку створити відповідний метод, а потім його впровадити на практиці у вигляді відповідної програми.

А.1 Підстави для розробки

Підставою для розробки є завдання на кваліфікаційну роботу магістра, визначене за темою «Дослідження та програмна реалізація методів керування замовленнями засобів для догляду за здоров'ям», яку затверджено наказом № 435 від 5 листопада 2021 р. за Національним університетом «Запорізька політехніка».

А.2 Призначення розробки

Програма призначена для організації підтримки замовлень користувачами засобів для догляду за здоров’ям відносно створення цих замовлень клієнтами та керування цими замовленнями менеджерами з можливістю класифікації замовлень за схильністю їх замовників до відмови від доставлених замовлень.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик програми повинні включати:

– замовлення засобів для догляду за здоров’ям;

– пошук засобів для догляду за здоров’ям;

– публікація звіту про досвід використання засобу для догляду за здоров’ям;

– визначення реакції на звіт іншого користувача;

– відображення звітів за засобом за заданою категорією середньої оцінки;

– внесення і керування даними про засоби для догляду за здоров’ям;

– визначення фасувань засобів для догляду за здоров’ям;

– організація списків з засобів для догляду за здоров’ям, включаючи створення таких списків, додавання засобів до списків, вилучення їх зі списків;

– формулювання запитання про засіб для догляду за здоров’ям;

– формулювання відповіді на запитання іншого клієнта;

– визначення реакції на відповідь на запитання іншим клієнтом;

– керування замовленнями засобів;

– перегляд власних замовлень з можливістю скасувати замовлення;

– перегляд засобів за виробником;

– перегляд засобів за категорією;

– перегляд категорій за групою;

– пошук засобів за ідентифікатором або за UPC-номером для менеджера;

– внесення даних про кількісну наявність засобів.

А.3.2 Вимоги до складу та параметрів технічних засобів

Звичайно, що вимоги до складу та параметрів технічних засобів мають визначатися планованою кількістю покупців, що будуть працювати з сервісом. Від цього залежать і встановлені вимоги.

В якості базових вимог у результаті для роботи сервісу замовлення

засобів для догляду за здоров’ям середнього розміру було винесено:

– процесор, що має характеристики: чотири ядра, тактова частота – 2,9 ГГц;

– оперативна пам’ять: 10 Гб;

– жорсткий диск: 100 Гб доступного простору.

А.4 Порядок контролю та приймання

Контроль виконання роботи здійснюється за календарним планом з прийманням її за відповідними результатами, зазначеними в плані у відведений термін виконання роботи загалом.

ДОДАТОК Б

Текст програми

Б.1 Текст файла models.py

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


class Manufacturer (models.Model):
    name = models.CharField(max_length=255)

    def __str__(self):
        return self.name

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

class HealthCareProduct (models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField()
    score = models.FloatField()
    state = models.SmallIntegerField()
    manufacturer = models.ForeignKey (Manufacturer,
on_delete=models.CASCADE)
    creation = models.DateTimeField()
    photos = models.CharField(max_length=255)
    category = models.ForeignKey (Category,
on_delete=models.CASCADE)
    returned = models.DateTimeField()
```

```
        def __str__(self):
            return self.name

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

    class HealthCareProductVariant (models.Model):
        variant = models.CharField(max_length=255)
        quantity = models.IntegerField()
        left = models.IntegerField()
        expirationdate = models.DateTimeField()
        upc = models.CharField(max_length=12)
        price = models.FloatField()
        stableprice = models.FloatField()
        weight = models.FloatField()
        dimensions = models.CharField(max_length=100)
        hcproduct ind = models.ForeignKey (HealthCareProduct,
on_delete=models.CASCADE)

    class Category (models.Model):
        name = models.CharField(max_length=255)
        groupcategory= models.ForeignKey (GroupCategory,
on_delete=models.CASCADE)

        def __str__(self):
            return self.name

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

```
        class GroupCategory (models.Model):
            name = models.CharField(max_length=255)

            def __str__(self):
                return self.name

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

        class Question (models.Model):
            question= models.CharField(max_length=255)
            creation = models.DateTimeField()
            hcproduct = models.ForeignKey (HealthCareProduct,
on_delete=models.CASCADE)
            customer = models.ForeignKey (Customer,
on_delete=models.CASCADE)

            def __str__(self):
                return self.question

        class Answer (models.Model):
            customer = models.ForeignKey (Customer,
on_delete=models.CASCADE)
            question = models.ForeignKey (Question,
on_delete=models.CASCADE)
            answer= models.CharField(max_length=255)
            creation = models.DateTimeField()
            positive = models.IntegerField()
            negative = models.IntegerField()
```

```
        def __str__(self):
            return self.answer

    class AnswerReaction (models.Model):
        creation = models.DateTimeField()
        answer = models.ForeignKey (Answer,
on_delete=models.CASCADE)
        customer = models.ForeignKey (Customer,
on_delete=models.CASCADE)
        positive = models.BooleanField()

    class ReportReaction (models.Model):
        creation = models.DateTimeField()
        report = models.ForeignKey (Report, on_delete=models.CASCADE)
        customer = models.ForeignKey (Customer,
on_delete=models.CASCADE)
        positive = models.BooleanField()

    class Customer (AbstractUser):
        username = models.CharField(max_length=100)
        password = models.CharField(max_length=100)
        photo = models.CharField(max_length=255)
        registration = models.DateTimeField()
        last = models.DateTimeField()
        reports = models.IntegerField()
        answers = models.IntegerField()
        questions = models.IntegerField()

        def __str__(self):
            return self.username
```

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

        class Report (models.Model):
            hcproduct = models.ForeignKey (HealthCareProduct,
on_delete=models.CASCADE)
            bought = models.BooleanField()
            review = models.TextField()
            creation = models.DateTimeField()
            customer = models.ForeignKey (Customer,
on_delete=models.CASCADE)
            score = models.IntegerField()
            positive = models.IntegerField()
            negative = models.IntegerField()

            def __str__(self):
                return self.review

        class Order (models.Model):
            customer = models.ForeignKey (Customer,
on_delete=models.CASCADE)
            creation = models.DateTimeField()
            price = models.FloatField()
            discountprice = models.FloatField()
            address = models.CharField(max_length=255)
            payed = models.BooleanField()
            status = models.SmallIntegerField()
            comment = models.CharField(max_length=255)
            delivered = models.DateTimeField()
            returned = models.DateTimeField()
```

```
        possiblereturn = models.BooleanField()

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

    class OrderProduct (models.Model):
        order = models.ForeignKey (Order, on_delete=models.CASCADE)
        variant = models.ForeignKey (HealthCareProductVariant,
on_delete=models.CASCADE)
        price = models.FloatField()
        quantity = models.IntegerField()

    class HealthCareList (models.Model):
        name = models.CharField(max_length=255)
        creation = models.DateTimeField()
        customer = models.ForeignKey (Customer,
on_delete=models.CASCADE)
        healthcarelistproducts = models.ManyToManyField(HealthCareProduct)

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

Б.2 Текст файла healthcareproduct.py

```
    from django.views.generic.detail import DetailView
    from django.contrib.auth.mixins import LoginRequiredMixin
    from django.views.generic.edit import CreateView, UpdateView
    from datetime import datetime
    import healthproseller.models
```

```
class HealthCareProductView (DetailView):

    model = HealthCareProduct
    template_name = 'hcp.html'

    def get_context_data(self, **kwargs):
        cont = super().get_context_data(**kwargs)
        hcp = HealthCareProduct.objects.get(id=self.kwargs['pk'])
        cont ['product'] = hcp
        reports = Report.objects.filter(hcproduct = self.kwargs['pk'])
        cont ['reports'] = reports
        ques = Question.objects.filter(hcproduct = self.kwargs['pk'])
        for q in ques:
            answers = Answer.obkects.filter(question = q.id)
            q['answers'] = answers
        cont ['questions'] = ques
        variants = HealthCareProductVariant.objects.filter(hcproduct =
self.kwargs['pk'])
        cont ['variants'] = variants
        return cont

class AddHealthCareProductView (LoginRequiredMixin, CreateView):

    template_name = 'ahcp.html'
    model = HealthCareProduct

    fields = [
        'name',
        'description',
        'state',
```

```
            'manufacturer',
            'photos',
            'category'
        ]

        login_url = 'managerloginview'
        success_url = 'healthcareproductview'

        def form_valid(self, form):
            form.creation = datetime.now()
            return super().form_valid(form)

    class UpdateHealthCareProductView (LoginRequiredMixin, UpdateView):

        template_name = 'uhcp.html'
        model = HealthCareProduct

        fields = [
            'name',
            'description',
            'state',
            'manufacturer',
            'photos',
            'category',
            'returned'
        ]

        login_url = 'managerloginview'
        success_url = 'healthcareproductview'
```

```
class CategoryHealthCareProductView (ListView):

    model = HealthCareProduct
    template_name = 'cathcp.html'

    def get_queryset(self, *args, **kwargs):
        cont = super().get_context_data(**kwargs)
        hcpcat = self.request.GET.get('category')
        products = HealthCareProduct.objects.filter(category = hcpcat)
        products = products.order_by('-score')
        cont ['products'] = products
        return cont

class ManufacturerHealthCareProductView (ListView):

    model = HealthCareProduct
    template_name = 'manhcp.html'

    def get_queryset(self, *args, **kwargs):
        cont = super().get_context_data(**kwargs)
        hcpmanuf = self.request.GET.get('manuf')
        products = HealthCareProduct.objects.filter(manufacturer =
hcpmanuf)
        products = products.filter(state = 0)
        products = products.order_by('-id')
        cont ['products'] = products
        return cont

class SearchHealthCareProductView (ListView):
```

```
        model = HealthCareProduct
        template_name = 'searchhcp.html'

        def get_queryset(self, *args, **kwargs):
            cont = super().get_context_data(**kwargs)
            searchtext = self.request.GET.get('search')
            products = HealthCareProduct.objects.filter(name__contains =
searchtext)
            products = products.filter(state = 0)
            products = products.order_by('-score')
            cont ['products'] = products
            return cont

    class IdHealthCareProductView (LoginRequiredMixin, ListView):

        model = HealthCareProduct
        template_name = 'idhcp.html'

        def get_queryset(self, *args, **kwargs):
            cont = super().get_context_data(**kwargs)
            hcpid = self.request.GET.get('hcpid')
            product = HealthCareProduct.objects.get(id = hcpid)
            cont ['product'] = product
            return cont
```

Б.3 Текст файла healthcareproductvariant.py

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

```
import healthproseller.models


class HealthCareProductVariantView (DetailView):

    model = HealthCareProductVariant
    template_name = 'hcpv.html'

    def get_context_data(self, **kwargs):
        cont = super().get_context_data(**kwargs)
        hcp = HealthCareProductVariant.objects.get(id=self.kwargs['pk'])
        cont ['prodvar'] = hcp
        return cont

class AddHealthCareProductVariantView (LoginRequiredMixin,
CreateView):

    template_name = 'ahcpv.html'
    model = HealthCareProductVariant

    fields = [
        'variant',
        'quantity',
        'left',
        'expirationdate',
        'upc',
        'price',
        'stableprice',
        'weight',
        'dimensions',
```

```
            'hcproduct'
        ]

        login_url = 'managerloginview'
        success_url = 'healthcareproductview'

    class UpdateHealthCareProductVariantView (LoginRequiredMixin,
UpdateView):

        template_name = 'uhcpv.html'
        model = HealthCareProductVariant

        fields = [
            'variant',
            'quantity',
            'left',
            'expirationdate',
            'upc',
            'price',
            'stableprice',
            'weight',
            'dimensions',
            'hcproduct'
        ]

        login_url = 'managerloginview'
        success_url = 'healthcareproductview'

    class HCPHealthCareProductVariantView (ListView):
```

```
    model = HealthCareProductVariant
    template_name = 'hcphcpv.html'

    def get_queryset(self, *args, **kwargs):
        cont = super().get_context_data(**kwargs)
        product = self.request.GET.get('prod')
        variants = HealthCareProduct.objects.filter(hcproduct = product)
        variants = products.filter(left__gte = 0)
        variants = products.order_by('-id')
        cont ['variants'] = variants
        return cont

class UpcHealthCareProductVariantsView (ListView):

    model = HealthCareProductVariant
    template_name = 'upchcpv.html'

    def get_queryset(self, *args, **kwargs):
        cont = super().get_context_data(**kwargs)
        upcvalue = self.request.GET.get('upc')
        variants = HealthCareProduct.objects.filter(upc = upcvalue)
        variants = products.order_by('-id')
        cont ['variants'] = variants
        return cont

class HCPHealthCareProductVariantsView (ListView):

    model = HealthCareProductVariant
    template_name = 'hcphcpvs.html'
```

```
    def get_queryset(self, *args, **kwargs):
        cont = super().get_context_data(**kwargs)
        product = self.request.GET.get('prod')
        variants = HealthCareProduct.objects.filter(hcproduct = product)
        variants = products.order_by('-id')
        cont ['variants'] = variants
        return cont

class ExpDateHCPVariantsView (ListView):

    model = HealthCareProductVariant
    template_name = 'exphcpv.html'

    def get_queryset(self, *args, **kwargs):
        cont = super().get_context_data(**kwargs)
        product = self.request.GET.get('prod')
        variants = HealthCareProduct.objects.filter(hcproduct = product)
        variants = products.filter(left__gte = 0)
        variants = products.order_by('-expirationdate')
        cont ['variants'] = variants
        return cont
```

Б.4 Текст файла report.py

```
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import CreateView, UpdateView
import healthproseller.models
from datetime import datetime
```

```
class AddReportView (LoginRequiredMixin, CreateView):

    template_name = 'areport.html'
    model = Report

    fields = [
        'hcproduct',
        'review',
        'score'
    ]

    login_url = 'customerloginview'
    success_url = 'reportsview'

    def form_valid(self, form):
        ruser = self.request.user
        cust = Customer.objects.get(id = ruser.id)
        form.customer = cust
        form.creation = datetime.now()
        prod = form.hcproduct
        variants = OrderProduct.objects.filter(variant__hcproduct__id = prod
& order__customer = cust)
        if variants.count() > 0:
            form.bought = True
        else:
            form.bought = False
        form.positive = 0
        form.negative = 0
        return super().form_valid(form)
```

```
class UpdateReportView (LoginRequiredMixin, UpdateView):

    template_name = 'ureport.html'
    model = Report

    fields = [
        'hcproduct',
        'review',
        'score'
    ]

    login_url = 'customerloginview'
    success_url = 'reportsview'

class MyReports (LoginRequiredMixin, ListView):

    model = Report
    template_name = 'myreps.html'

    def get_queryset(self, *args, **kwargs):
        cont = super().get_context_data(**kwargs)
        ruser = self.request.user
        cust = Customer.objects.get(id = ruser.id)
        reports = Report.objects.filter(customer = cust)
        reports = reports.order_by('-creation')
        cont ['reporrts'] = reports
        return cont

class CustomerReports (ListView):
```

```
        model = Report
        template_name = 'custreps.html'

        def get_queryset(self, *args, **kwargs):
            cont = super().get_context_data(**kwargs)
            custom = self.request.GET.get('custom')
            reports = Report.objects.filter(customer__id = custom)
            reports = reports.order_by('-creation')
            cont ['reports'] = reports
            return cont

    class HealthCareProductReports (ListView):

        model = Report
        template_name = 'hcpreps.html'

        def get_queryset(self, *args, **kwargs):
            cont = super().get_context_data(**kwargs)
            prod = self.request.GET.get('prod')
            b = self.request.GET.get('bought')
            if b:
                reports = Report.objects.filter(hcproduct__id = prod & bought =
True)
            else:
                reports = Report.objects.filter(hcproduct__id = prod & bought =
False)
            reports = reports.order_by('-creation')
            cont ['reports'] = reports
            return cont
```

```
class HealthCareDividedReports (ListView):

    model = Report
    template_name = 'dichcpreps.html'

    def get_queryset(self, *args, **kwargs):
        cont = super().get_context_data(**kwargs)
        star = self.request.GET.get('star')
        reports = Report.objects.filter(score = star)
        reports = reports.order_by('-positive')
        cont ['reports'] = reports
        return cont
```

ДОДАТОК В

Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Кваліфікаційна робота бакалавра

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ КЕРУВАННЯ ЗАМОВЛЕННЯМИ ЗАСОБІВ ДЛЯ ДОГЛЯДУ ЗА ЗДОРОВ'ЯМ

RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR MANAGING THE ORDERS FOR HEALTHCARE PRODUCTS

Виконав
студент групи КНТ-210м А.В. Сердюк
Керівник
д.т.н., професор С.О. Субботін

2021

Рисунок В.1 – Слайд 1

2

Об’єкт дослідження – процес замовлення засобів для догляду за здоров’ям.

Предмет дослідження – методи керування замовленнями засобів для догляду за здоров’ям.

Мета роботи – розробити програмне забезпечення замовлення засобів для догляду за здоров’ям для підвищення ефективності роботи менеджерів щодо обслуговування замовлень.

Рисунок В.2 – Слайд 2

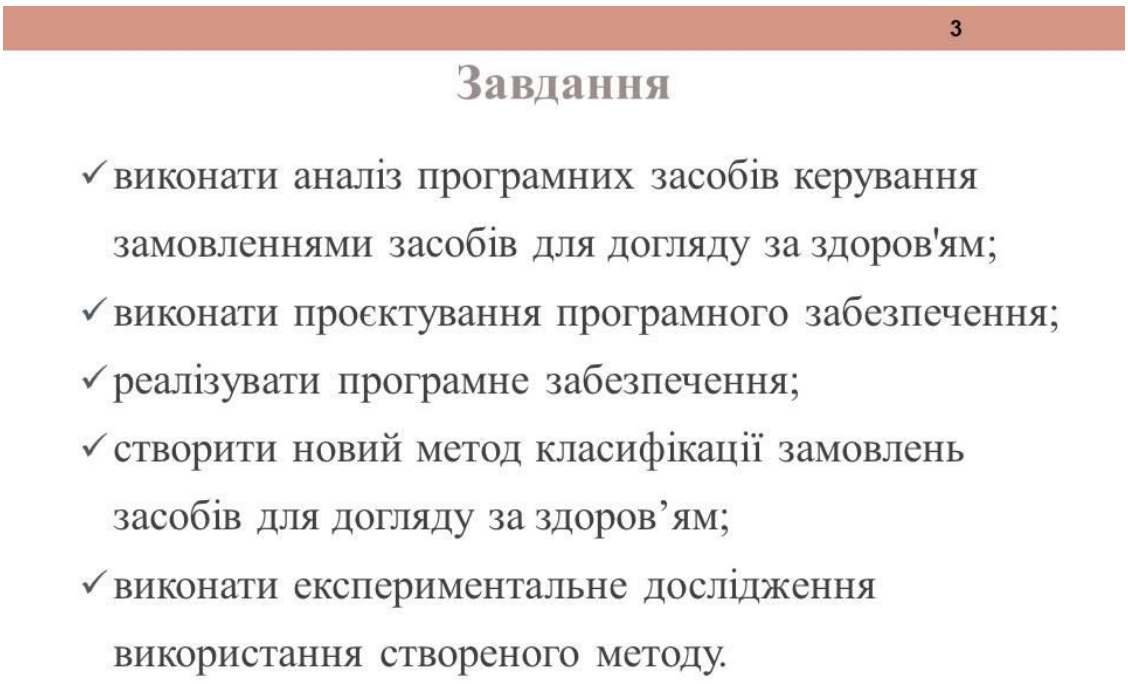

Рисунок В.3 – Слайд 3

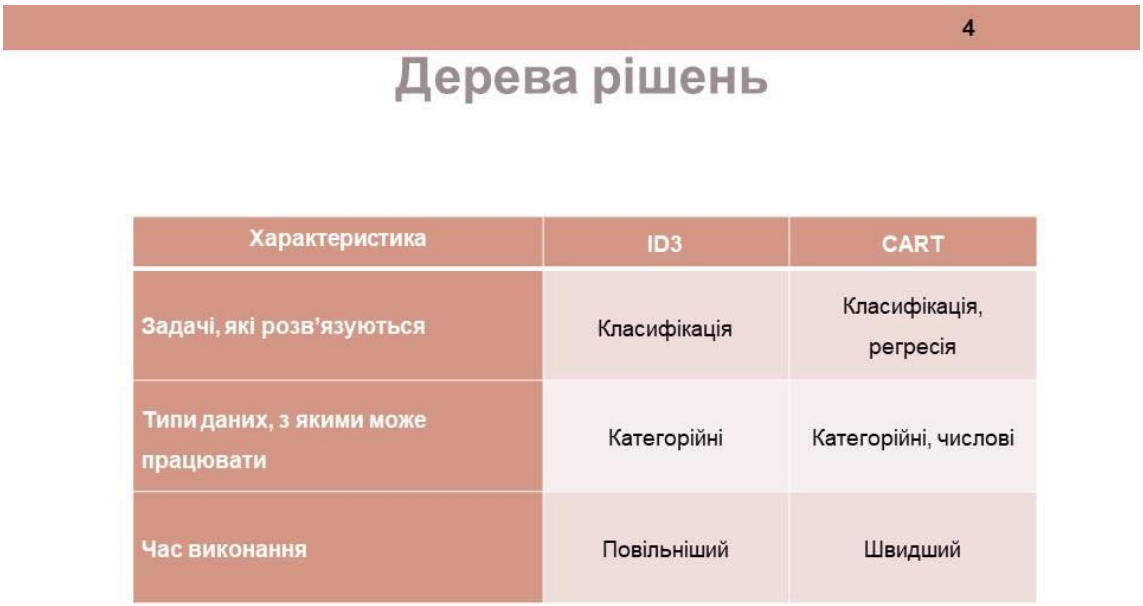

Рисунок В.4 – Слайд 4

5

Метод класифікації замовлень засобів для догляду за здоров'ям

На першому етапі роботи методу має бути сформовано вибірку даних, яка повинна включати екземпляри, для яких вихідною ознакою є клас щодо відмови клієнта від отримання замовлення, а вхідними ознаками є:
– статус сплати замовлення;
– наявність відмітки для кур'єра;
– вартість замовлення;
– кількість замовлених засобів;
– назва засобу;
– кількість діб, яка пройшла з моменту останньої відмови від отримання такого засобу для догляду за здоров'ям.
На другому етапі виконується оброблення текстових даних з вилученням зайвих слів, зайвих символів, обчисленням оцінок TF-IDF.
На третьому етапі виконується навчання дерева рішень на основі алгоритму CART.
На четвертому етапі виконується класифікація нових замовлень засобів для догляду за здоров'ям.

Рисунок В.5 – Слайд 5

6

Діаграма прецедентів

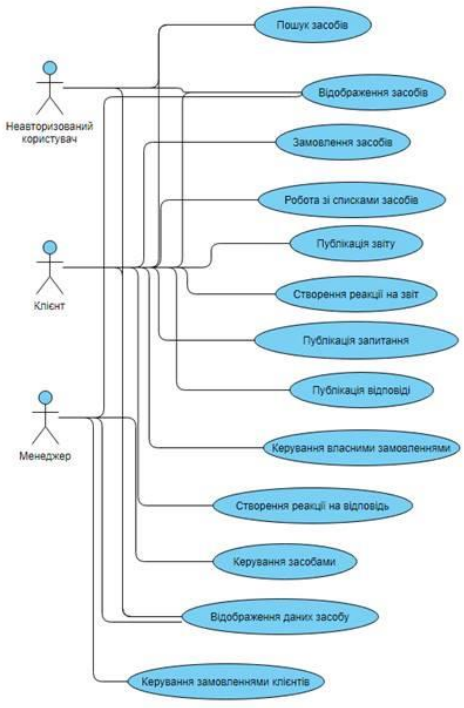

Рисунок В.6 – Слайд 6

7

Порівняння мов програмування для реалізації програмного забезпечення замовлення засобів для догляду за здоров'ям

Характеристика	C#	Python
Підтримка вебфреймворків	Так	Так
Підтримка об'єктно-реляційного відображення	Так	Так
Підтримка динамічної типізації	Ні	Так
Легкість серверного розгортання	Відносно	Так
Зручність роботи з масивами даних	Відносно	Так

Рисунок В.7 – Слайд 7

8

Схема бази даних

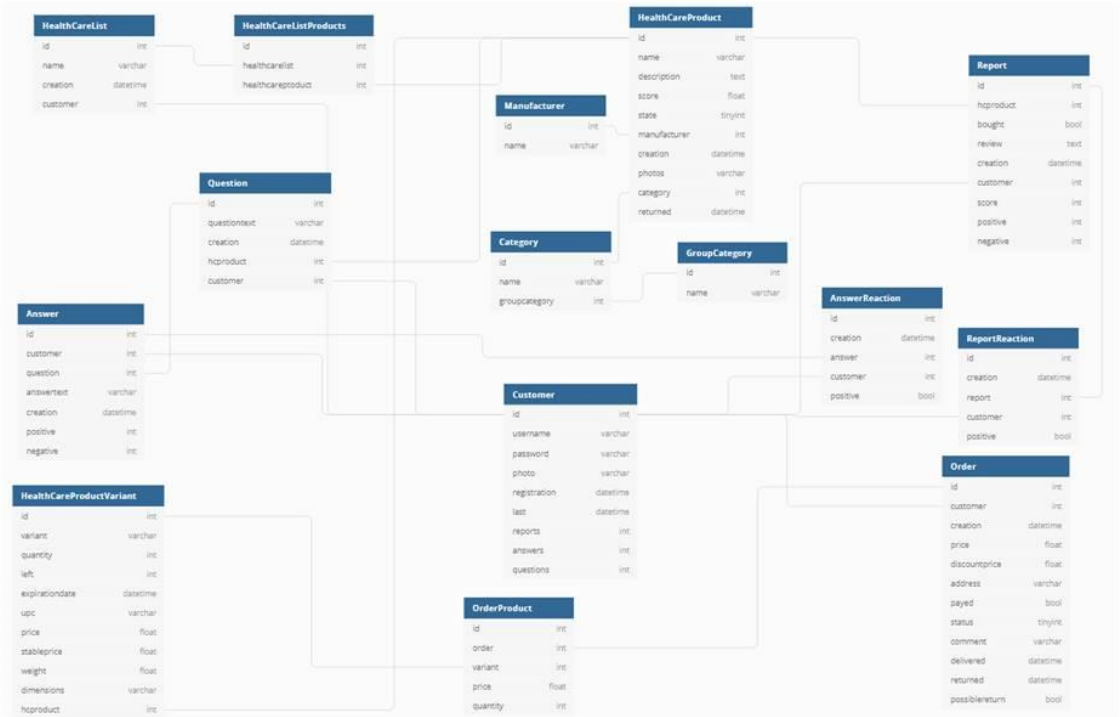

Рисунок В.8 – Слайд 8

9

Структурна схема програмного проєкту

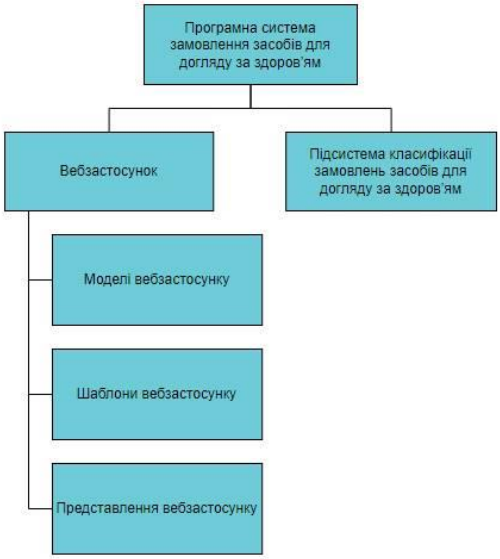

Рисунок В.9 – Слайд 9

10

Діаграма станів замовлення засобів для догляду за здоров’ям

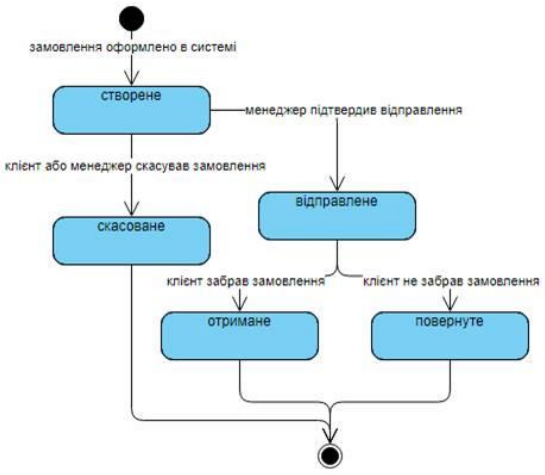

Рисунок В.10 – Слайд 10

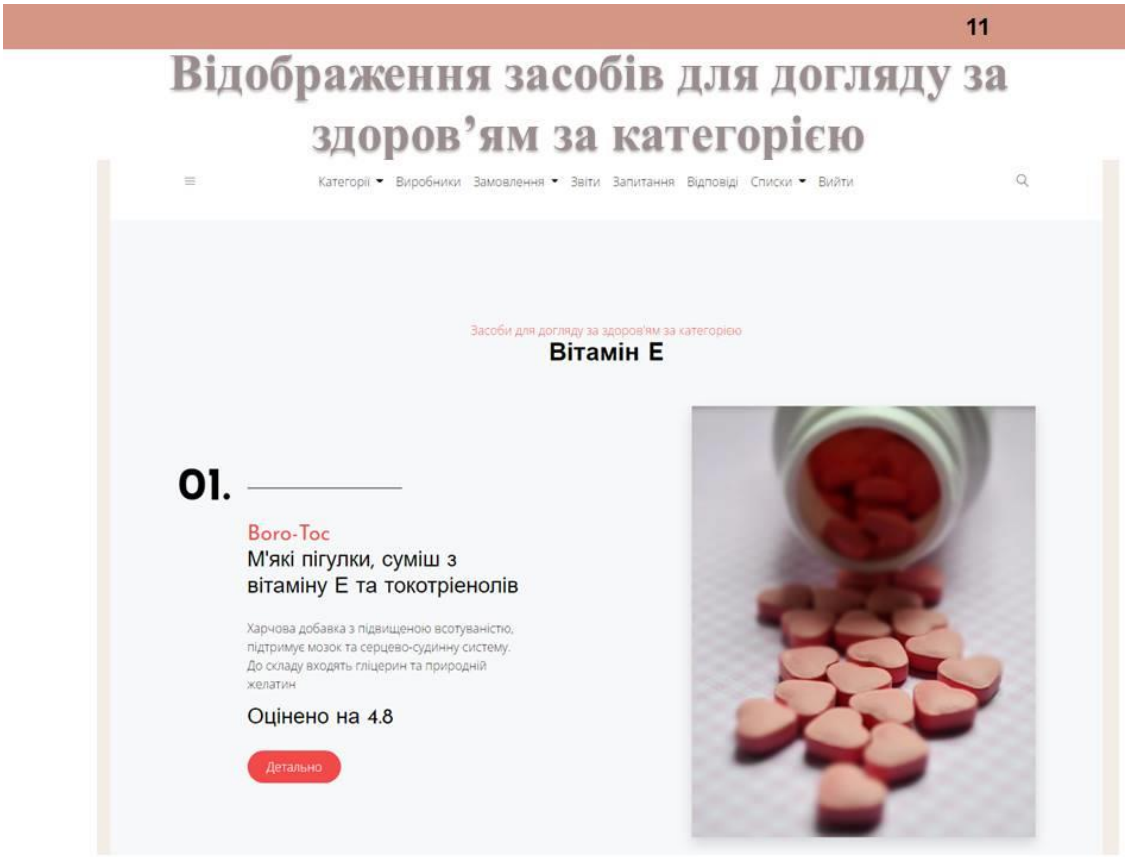

Рисунок В.11 – Слайд 11

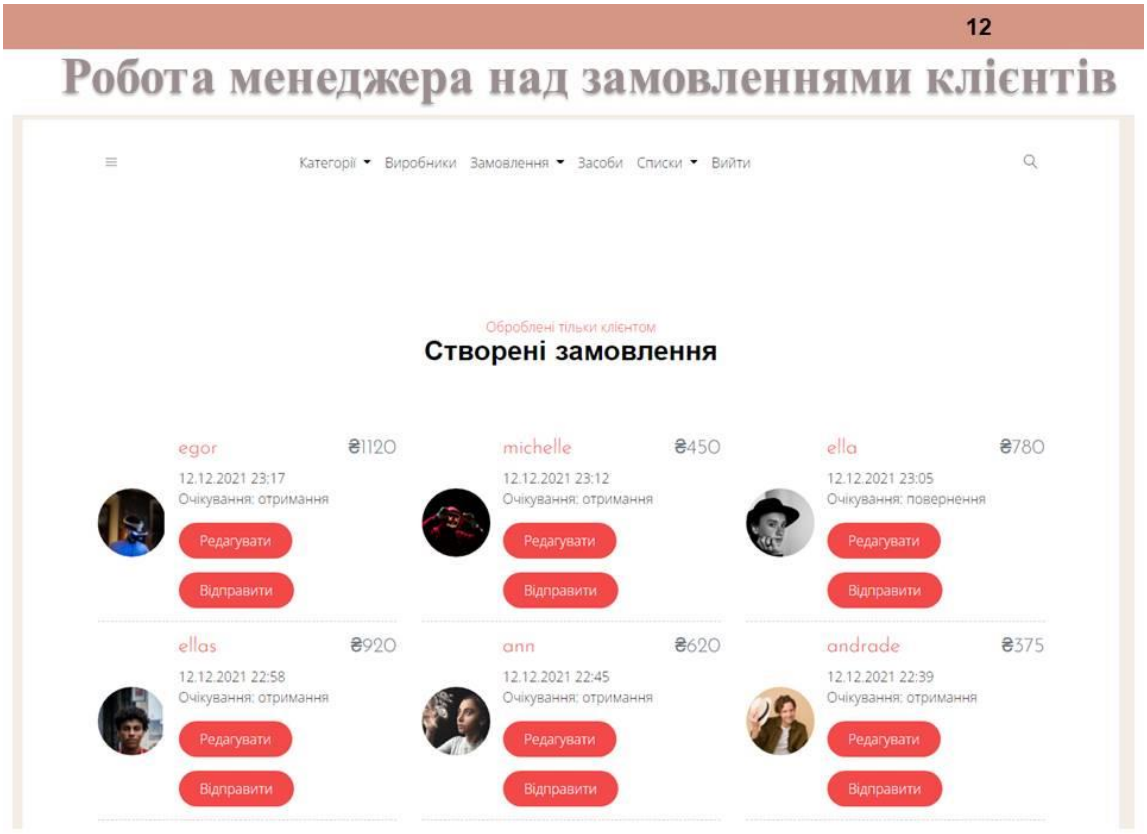

Рисунок В.12 – Слайд 12

13

Результати експериментального дослідження

Метод	Точність
Метод k-найближчих сусідів	58,31
Метод класифікації замовлень засобів для догляду за здоров'ям	80,27

Рисунок В.13 – Слайд 13

Висновки

Результати роботи охоплюють створену діаграму прецедентів, створений метод класифікації замовлень засобів для догляду за здоров'ям, результати порівняння мов програмування, схему бази даних, діаграму станів, розроблене програмне забезпечення і виконане експериментальне дослідження запропонованого методу.

Наукова новизна роботи полягає в запропонованому методі класифікації замовлень засобів для догляду за здоров'ям, який виконує класифікацію на основі параметрів замовлення та часу останнього повернення засобу з використанням дерева рішень на основі алгоритму CART, що дозволяє керувати замовленнями щодо подальшої їх підтримки. Проведене експериментальне дослідження застосування даного методу підтвердило можливість його практичного використання у даній сфері.

14

Рисунок В.14 – Слайд 14