

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ

(назва теми)

КЛАСИФІКАЦІЇ ДЛЯ СЕРВІСУ РЕМОНТУ ПОБУТОВОЇ ТЕХНІКИ

RESEARCH AND SOFTWARE IMPLEMENTATION OF

CLASSIFICATION METHODS FOR HOME APPLIANCE REPAIR

SERVICE

Виконав(ла): студент(ка) 2 курсу, групи КНТ-212м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

ЦИБУЛЬСЬКИЙ О.Г.

(ПРИЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

2023_____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ 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.
Сергій СУББОТІН
« _____ » _____ 2023 року

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

ЦИБУЛЬСЬКОГО Олега Геннадійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів класифікації для сервісу ремонту побутової техніки. Research and Software Implementation of Classification Methods for Home Appliance Repair Service

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

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

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року
3. Вихідні дані до проєкту (роботи) _____ рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Розробка архітектури програми. 4. Реалізація програми. 5. Експлуатація, тестування та експериментальне дослідження програми.

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

6. Консультанти розділів проєкту (роботи)

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

_____ Олег ЦИБУЛЬСЬКИЙ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

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

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
112 с., 24 рис., 4 табл., 3 дод., 12 джерел.

ПОБУТОВА ТЕХНІКА, РЕМОНТ, МАШИННЕ НАВЧАННЯ,
ЗАДАЧА КЛАСИФІКАЦІЇ, RANDOM FOREST.

Об'єкт роботи – процес підтримки роботи сервісу ремонту побутової техніки. Предмет роботи – методи класифікації заявок на ремонт побутової техніки. Мета роботи – розробити програмне забезпечення підтримки роботи сервісу ремонту побутової техніки для підвищення зручності роботи всіх задіяних сторін.

Матеріали, методи та технічні засоби: мови Python, HTML, CSS, фреймворк Django, система керування PostgreSQL щодо роботи з базою даних, методи Random Forest, Word2vec.

Результати. Створена програма може використовуватися і волонтерськими організаціями, які підтримують виконання ремонту побутової техніки, і комерційними сервісами ремонту побутової техніки, і власниками пристроїв побутової техніки для створення заявок на виконання ремонту, для відслідковування виявлених раніше проблем з пристроями, виконаних обслуговувань, класифікації заявок щодо результатів ремонту.

Висновки. Наукова новизна роботи полягає в створеному методі класифікації заявок на ремонт побутової техніки, який виконує класифікацію заявок на успішно виконані та такі, які не завершилися успішним ремонтом техніки, будує для цього модель класифікації на основі методу Random Forest та перетворює текстовий опис проблеми на основі методу Word2vec для використання в методі.

Галузь використання. Комерційні та некомерційні сервіси ремонту побутової техніки.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 112 pages, 24 figures, 4 tables, 3 appendixes, 12 sources.

HOME APPLIANCES, REPAIR, MACHINE LEARNING, CLASSIFICATION PROBLEM, RANDOM FOREST.

The object of the work is the process of supporting the work of the home appliance repair service. The subject of the work is methods of classifying applications for the repair of household appliances. The purpose of the work is to develop software to support the work of the home appliance repair service to increase the convenience of work for all involved participants.

Materials, methods and technical means: Python, HTML, CSS languages, Django framework, PostgreSQL management system for database support, Random Forest, Word2vec methods.

The results. The created program can be used by volunteer organizations that support the repair of household appliances, commercial services for the repair of household appliances, and owners of household appliances to create requests for repairs, to track previously identified problems with devices, services performed, and classification of requests regarding the results of repairs.

Conclusions. The scientific novelty of the work is in the created method of classifying applications for the repair of household appliances, which performs the classification of applications into successfully completed and those that did not end with successful repair of appliances, builds a classification model for this purpose based on the Random Forest method and transforms the textual description of the problem based on the Word2vec method for usage in the method.

Field of use. Commercial and non-commercial home appliance repair services.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Програмні інструменти для забезпечення роботи з сервісом ремонту побутової техніки.....	11
1.2 Порівняння розроблюваної програми з існуючими	14
1.3 Висновки за розділом 1	15
2 Матеріали і методи.....	16
2.1 Використання методів класифікації для розв’язання задачі класифікації заявок на ремонт побутової техніки	16
2.2 Метод класифікації заявок на ремонт побутової техніки	18
2.3 Висновки за розділом 2	21
3 Розробка архітектури програми.....	22
3.1 Визначення сценаріїв використання програми користувачами	22
3.2 Рішення щодо вибору засобів розробки	26
3.3 Визначення структури даних у базі даних	27
3.4 Висновки за розділом 3	46
4 Реалізація програми	47
4.1 Порядок роботи над заявками у програмі	47
4.2 Програмна реалізація вебзастосунку	48
4.3 Висновки за розділом 4	65
5 Експлуатація, тестування та експериментальне дослідження програми	66
5.1 Призначення програми	66
5.2 Виконання програми.....	66
5.3 Експериментальне дослідження	70
5.4 Висновки за розділом 5	73
Висновки	75
Перелік джерел посилання	76

Додаток А Технічне завдання	78
Додаток Б Текст програми	83
Додаток В Слайди презентації.....	105

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

НБК – наївний баєсів класифікатор.

ВСТУП

Ремонт побутової техніки – це важливе практичне завдання, з яким періодично змушені стикатися люди. Поки діє гарантійний термін, обслуговування може відбуватися у спеціалізованих сервісних центрах, які визначені виробником. Але в подальшому вже власник побутової техніки може вирішувати самостійно, послугами якого сервісу скористатися. Сервіси можуть бути представлені деякими мережами, можуть бути представлені окремими пунктами. Але це стосується традиційних підходів, коли такі сервіси є комерційними.

Проте існує і інший варіант, коли сервіси представлені волонтерськими організаціями. Тоді або волонтер сам виконує ремонт, але безкоштовно, або допомагає користувачу виконати цей ремонт. Для волонтерських організацій характерним є проведення подій, на яких як раз такі процеси і відбуваються. Фінансування такі організації можуть отримувати за рахунок урядових і неурядових грантів. Підґрунтям для такої діяльності, хоч вона і є безкоштовною, є найчастіше екологічні ініціативи. Тобто замість того, щоб викидати пристрій, який можна відремонтувати, і або перевантажувати сміттєзвалища, або збільшувати обсяг роботи для заводів з переробки сміття, де такі можливості є, а відповідно впливати на екологію, фінансуються роботи з ремонту для того, щоб продовжити життя існуючим пристроям.

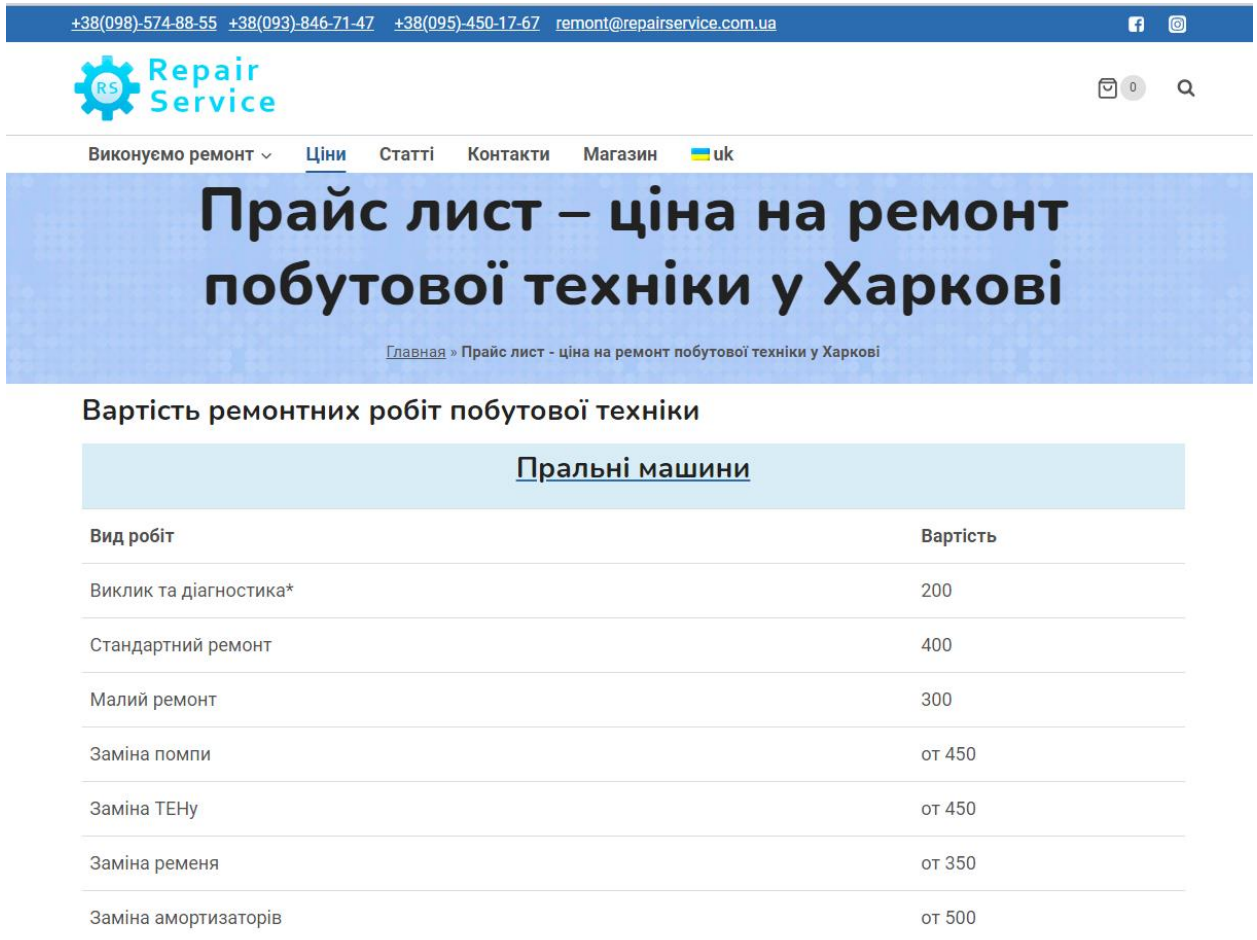
Об'єднання і компаній, які можуть надати комерційні послуги, і волонтерських організацій в одній програмі, має дозволити створити платформу, на якій люди за рахунок більшої кількості можливостей можуть вибирати потрібний їм варіант, розуміти можливості вибору, а таким чином частіше використовувати можливість відремонтувати техніку замість придбання нової, де це можливо. Тож саме таке програмне забезпечення і має бути створено в роботі в підсумку.

Для того щоб користувач одразу отримував додаткову інформацію про те, чи вирогідно, що ремонт вдасться, важливою є інтеграція у програму інструменту класифікації поданих заявок на ремонт побутової техніки у функцію створення таких замовлень.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Програмні інструменти для забезпечення роботи з сервісом ремонту побутової техніки

Окремих сервісів для ремонту побутової техніки в Україні і світі дуже багато, що тільки вказує на те, що створення платформи, яка могла б об'єднати декілька пропозицій від різних сервісів було би корисним для самих користувачів. Як приклад вебсайту одного з таких сервісів виділено Repair Service (рис. 1.1) [1].



The screenshot shows the website of Repair Service. The header includes contact information: +38(098)-574-88-55, +38(093)-846-71-47, +38(095)-450-17-67, and email remont@repairservice.com.ua. The main navigation bar has links for 'Виконуємо ремонт', 'Ціни', 'Статті', 'Контакти', 'Магазин', and a language selector 'uk'. The main heading is 'Прайс лист – ціна на ремонт побутової техніки у Харкові'. Below it is a breadcrumb trail: 'Главная » Прайс лист - ціна на ремонт побутової техніки у Харкові'. The section title is 'Вартість ремонтних робіт побутової техніки'. A table titled 'Пральні машини' lists various repair services and their costs.

Вид робіт	Вартість
Виклик та діагностика*	200
Стандартний ремонт	400
Малий ремонт	300
Заміна помпи	от 450
Заміна ТЕНу	от 450
Заміна ременя	от 350
Заміна амортизаторів	от 500

Рисунок 1.1 – Вигляд сторінки з вартістю ремонтних робіт побутової техніки сервісу Repair Service

Repair Service містить сторінки, які надають інформацію про те, які саме ремонтні роботи виконуються для різних категорій побутової техніки.

Також за категоріями техніки розділена інформація про вартість послуг, яка може бути або встановлена точно, або як мінімальне значення такої вартості.

За деякими категоріями техніки надається не тільки ремонт, але і можливість встановлення. На вебсайті можна подати заявку на ремонт або встановлення техніки. Заявки на ремонт міст ім'я клієнта, номер телефону для контактування, опис проблеми, яка виникла. Ніякої системи авторизації на вебсайті немає.

Особливі можливості щодо ремонту побутової техніки надають волонтерські організації. Прикладом такого варіанта роботи над виправленням побутової техніки є вебсайт Repair Cafe Wales (рис. 1.2) [2].

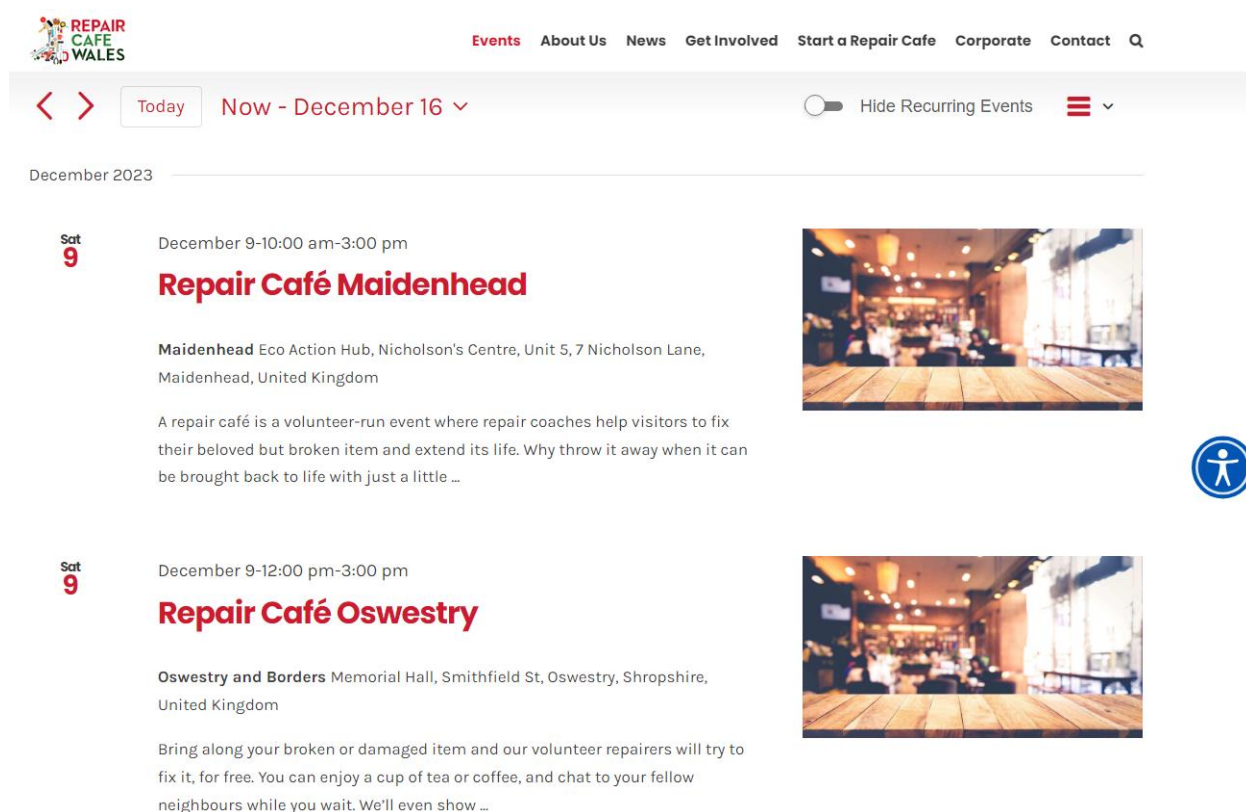


Рисунок 1.2 – Вигляд сторінки пошуку подій Repair Cafe Wales

Вебсайт Repair Cafe Wales має набагато ширші можливості стосовно функціональності. Ремонти реалізуються під час проведення подій, тому програма дозволяє шукати такі події, шукати місця, де такі події проводяться, а за ними переглядати події. Також самі майстри можуть реєструватися для

проведення подій. Можна подати заявку на створення нового місця проведення події, заповнюючи відповідну форму.

Іншим варіантом такої волонтерської організації є Fixit Clinic (рис. 1.3) [3].



Рисунок 1.3 – Вигляд основної сторінки Fixit Clinic

Вебсайт Fixit Clinic має дуже мінімалістичний інтерфейс. Побудований за принципом блогу. Тим не менш функціональність подібна на попередній сервіс, а також наявна можливість авторизації, що дозволяє далі контролювати виконані дії, хоча і можливість дуже обмежені в такому

випадку. Обмеженість пов'язана зокрема з тим, що подача заявок можлива тільки через Google-форму. Надаються форми для подачі заявки на ремонт побутової техніки під час зустрічі, віддалено через організацію Zoom-конференції та для подачі заявки щодо участі в зустрічах як волонтера.

1.2 Порівняння розроблюваної програми з існуючими

Врахувавши особливості тих програмних інструментів, які розглянуті в першому розділі, важливо було визначити, в чому полягають відмінності розроблюваної програми (табл. 1.1).

Таблиця 1.1 – Характеристика розроблюваної та існуючих програм

Критерій	Repair Service	Repair Cafe Wales	Fixit Clinic	Розробка
Тип сервісу	Комерційний	Волонтерський	Волонтерський	Волонтерський, комерційний
Створення заявки на ремонт	Ремонт, підключення	Ремонт, консультації, волонтерство	Ремонт, консультації, волонтерство	Ремонт, консультації, волонтерство
Керування заявками на ремонт	Немає	Немає	Немає	Як користувачем, так і сервісом
Класифікація заявок на ремонт	Немає	Немає	Немає	Є

Розроблювана програма (або розробка) об'єднує на одній платформі одночасно і сервіси, які є комерційними (працюють за прибуток, тобто

вимагають сплати), і сервіси, які є волонтерськими і направлені на допомогу фактично власниками руками, але під керівництвом волонтера. Це дозволяє кожному користувачу вирішувати, в якій саме ситуації, який саме сервіс використовувати. Проте використовуючи один з сервісів, користувач одразу бачить інші можливості, якщо інші сервіси зареєстровані в програмі.

Так само, як і вебсайти, які підтримують роботу волонтерських організацій, розробка дозволяє створювати заявки на ремонт, консультації, реєстрацію на події як волонтерів. Проте на відміну від розглянутих програм, розробка дозволяє після авторизації керувати власними заявками на ремонт, керувати працівникам сервісу з ремонту побутової техніки або волонтерської організації заявками, визначати фактично графіки робіт, керуючи такими заявками. Тож це дозволяє отримувати інформацію про стан заявок, змінювати цей стан, вести історію виконаних дій та проблем, які виявляються під час роботи. Ще однією відмінністю є можливість класифікації заявок на ремонт, що дозволяє одразу розуміти певний прогноз щодо того, чи вдасться відремонтувати такий пристрій побутової техніки, чи ні.

1.3 Висновки за розділом 1

Результати виконання першого розділу зведено до порівняння можливостей вебсайтів сервісів Repair Service, Repair Cafe Wales, Fixit Clinic та розробка, що має бути виконана в роботі, що є першим завданням роботи. Іншими завданнями є:

- дослідження методів і моделей класифікації заявок на ремонт побутової техніки;
- проектування програмного забезпечення для сервісу ремонту побутової техніки;
- програмна реалізація сервісу ремонту побутової техніки.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Використання методів класифікації для розв'язання задачі класифікації заявок на ремонт побутової техніки

Задача класифікації заявок на ремонт побутової техніки полягає у визначенні класу, до якого будуть належати результати ремонту даного пристрою, на момент, коли цей пристрій подається в ремонт.

Клас визначається успішністю проведення ремонту в сенсі приведення пристрою в стан, в якому він може використовуватися:

- пристрій відремонтовано;
- пристрій не вдалося відремонтувати: до такого стану слід віднести і випадки, коли пристрій в принципі більше не можна відремонтувати, і ті, за яких теоретично успішний ремонт можливий, але при цьому поточний ремонт закінчується невдачею (сервіс, наприклад, немає необхідних засобів для успішного ремонту цієї побутової техніки).

Класифікація виконується в момент, коли пристрій подається в сервіс, і визначається за заявкою стосовно ремонту. Заявка повинна мати встановлені значення характеристик пристрою. Ці значення визначають значення вхідних ознак. Вихідною ознакою є описаний вище клас успішності.

Таким чином, задача класифікації заявок на ремонт побутової техніки визначається тим, що необхідно визначити клас успішності ремонту побутової техніки за поданою заявкою на ремонт, якою виконано характеристику пристрою.

Для розв'язання задачі машинного навчання, яка є задачею класифікації можуть використовувати наступні методи.

Наївний баєсів класифікатор (НБК) базується на теоремі Баєса, використовуючи термін «наївний», адже цей класифікатор передбачає, що всі ознаки незалежні одна від одної, що може не відповідати дійсності в реальних ситуаціях на практиці, а виконання базового алгоритму

покладається на обчислення певних ймовірностей та порівняння цих різних апостеріорних ймовірностей відповідно до кількості класів, присутніх у даних, і клас з вищою ймовірністю призначається у відповідність наявному набору ознак як результат класифікації відповідного екземпляру [4].

Дерева рішень є моделями, які можна навчити, розділяючи вхідний набір на підмножини на основі міри вибору атрибутів – критерію, який використовується для оцінки корисності різних атрибутів для поділу набору даних, відповідно для навчання моделі важливим є визначення атрибута, який створить найбільш однорідні підмножини даних після поділу, тим самим максимізуючи приріст інформації, виконуючи це розбиття рекурсивно для кожної підмножини, при цьому важливою характеристикою дерева рішень є те, що дерева можуть обробляти дані великого розміру [5].

Класифікатор Random Forest – це ансамблевий алгоритм навчання моделей класифікації, який об'єднує декілька дерев рішень у єдину групу, якою і є ансамбль, для створення більш точних і надійних прогнозів щодо очікуваного класу на виході, він призначений для пом'якшення недоліків моделей на основі окремих дерев рішень, таких як надмірне навчання та висока дисперсія отриманих результатів [6]-[7].

Щоб виконати класифікацію нового екземпляру, який відсутній у навчальній вибірці даних, класифікатор Random Forest передає екземпляр через кожне окреме дерево рішень в ансамблі, відповідно кожне окреме дерево виконує класифікацію самостійно, а в підсумку голосує за один з класів, які визначені як можливий результат класифікації, клас з найбільшою кількістю голосів від всіх дерев, які беруть участь у роботі ансамблю, стає остаточним прогнозом [6]-[7].

Оскільки кожне дерево навчається на різних підмножинах даних і ознак і голосує для остаточної класифікації, у підсумку досягається менша ймовірність того, що ансамбль дерев рішень буде давати результати, викривлені через шум або викиди [6].

2.2 Метод класифікації заявок на ремонт побутової техніки

Для класифікації заявок на ремонт побутової техніки у роботі було запропоновано наступний метод.

Спочатку необхідно визначити дані, на яких буде виконуватися навчання моделі в підсумку. Потрібно окремо зазначити, що окрім безпосередньо самої моделі класифікації метод передбачає ще й створення моделі на основі методу Word2vec для роботи з текстом, яким є опис проблеми стосовно побутової техніки, що здається в ремонт. Стандартно передбачається створення останньої моделі на основі описів, наявних у вибірці даних. Проте якщо є додаткові тексти, в яких описано проблеми з приладами побутової техніки, тобто можна створити або вже створено відповідний корпус текстів стосовно проблем побутової техніки, то такий корпус може бути створений і використаний окремо. Тобто тоді модель для роботи з текстами створюється на основі такого корпусу текстів, а потім вже таким чином створена і навчена модель використовується в процесі оброблення текстів, які наявні в описах проблем побутової техніки у вибірці даних, яка використовується для навчання моделі класифікації заявок на ремонт побутової техніки.

Тож спочатку має бути створено вибірку даних, яка може або бути єдиною вибіркою даних або однією з двох. Якщо вибірок під час роботи методу має бути дві, то спочатку має бути виконано створення вибірки для формування корпусу текстів. Для виконання таких дій має реалізовуватись послідовність дій, яка описана нижче для роботи з першою вибіркою при обробленні даних для побудови моделі оброблення текстів. Якщо корпус текстів вже готовий, то відповідно передоброблення не виконується.

Тож спершу формується вибірка даних, яка повинна включати наступні характеристики всіх пристроїв побутової техніки, які віддавалися в ремонт і для яких відомі значення за цими характеристиками, а також визначено обов'язково результат ремонту:

- виробник побутової техніки, який створив цей пристрій (бренд компанії);
- кількість повних років, яка пройшла з моменту виготовлення конкретно цього пристрою (тобто різниця між датою подання заявки на ремонт та датою або роком виготовлення пристрою, якщо таке значення потрібно розрахувати);
- категорія побутової техніки, до якої відноситься цей пристрій;
- дата подання заявки на ремонт побутової техніки, яка одразу має бути перетворене на ціле число, що визначає кількість днів, яка минула з дати першої реєстрації заявки на ремонт у цій компанії або у цій вибірці загалом, якщо дані представляють різні компанії з ремонту;
- текстовий опис проблеми, через яку цей пристрій побутової техніки потребує ремонту;
- клас успішності ремонту пристрою побутової техніки.

При обробленні екземплярів вибірки окрім перетворення дати на число також і всі інші ознаки мають бути перетворені до числових значень окрім опису проблеми, який окремо оброблюється на наступних етапах роботи методу. Тобто якщо задано словесне значення категорії, то мають бути або з вибірки виділені наявні категорії, або окремо визначені наявні категорії побутової техніки, а далі кожне значення закодовано певним числовим значенням. Те саме стосується і виробника.

На наступному етапі має бути створено модель оброблення текстів проблем побутової техніки. Для цього мають бути виділені всі описані проблеми з вибірки даних або використано окремий набір текстів. Усі такі тексти мають бути проаналізовані. Якщо текст не є текстом певної мови (наприклад, англійської), то такий екземпляр вибірки має бути виключений з неї. Далі з наявних текстів виділяються слова, мають бути виключені будь-які неалфавітні значення, тобто виключаються спеціальні символи та всі ті символи, які не можуть бути словами, а також мають бути виключені стоп-слова, тобто слова, які є у відповідній мові найрозповсюдженішими і не

несуть важливого змістового відтінку. Таким чином отримані дані мають бути передані для створення моделі Word2vec на основі цих текстів. Тоді модель на виході буде мати набір числових оцінок (коефіцієнтів), які складають вектор, що представляє кожне слово.

На наступному етапі, використовуючи створену модель оброблення текстів, має бути виконано оброблення всіх описів проблем побутової техніки, наявних у екземплярах вибірки. Кожен текст, що визначає проблему конкретного пристрою, має бути перетворений аналогічно перетворенню, виконаному на попередньому етапі для формування даних для створення моделі. Тоді кожне слово має бути пропущено через навчену на попередньому етапі модель, а в підсумку розраховане для кожного коефіцієнта середню оцінку на основі значень, отриманих для кожного слова. Це дозволяє, усереднюючи значення, отримати вектор значень коефіцієнтів для кожного тексту з описом проблеми.

Наступний етап передбачає виконання створення навчальної вибірки даних, що включає як вхідні ознаки:

- виробника побутової техніки;
- кількість повних років, яка пройшла з моменту виготовлення конкретно цього пристрою;
- категорію побутової техніки, до якої відноситься цей пристрій;
- дату подання заявки на ремонт побутової техніки;
- відповідна кількість ознак згідно з кількістю елементів результуючого вектору, що було визначено при створенні моделі оброблення тексту, що представляють сумарно опис проблеми, через яку цей пристрій побутової техніки потребує ремонту.

Вихідною ознакою є клас успішності ремонту побутової техніки.

Далі створюється модель на основі методу Random Forest, що забезпечує створення ансамблю дерев рішень. Тоді виконується навчання моделі на основі вибірки, яку було підготовлено на попередньому етапі.

Останній етап передбачає використання навченої моделі на основі нових даних, які визначені за структурою вхідних ознак, описаної вище для навчальної вибірки. Для такого нового екземпляру даних виконується його класифікації та на виході визначається відповідний клас успішності ремонту побутової техніки.

2.3 Висновки за розділом 2

Наукова новизна роботи полягає в створеному методі класифікації заявок на ремонт побутової техніки, який виконує класифікацію заявок на успішно виконані та такі, які не завершилися успішним ремонтом техніки, будує для цього модель класифікації на основі методу Random Forest та перетворює текстовий опис проблеми на основі методу Word2vec для використання в методі.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

3.1 Визначення сценаріїв використання програми користувачами

Враховуючи проєктування програмного забезпечення і необхідність визначення зокрема засобів розробки, потрібно спочатку чітко визначити функціональні вимоги та зрозуміти принципи їх використання, що має дозволити зрозуміти в підсумку, які саме засоби потрібно обрати для розробки. Функціональні вимоги до програми були встановлені технічним завданням, однак сценарії використання залежать від ролі користувача, а до них в цій програмі належить звичайний користувач, яким є клієнт, працівник сервісу з ремонту побутової техніки, волонтер. Окрім того волонтер може бути або самостійним волонтером, або представляти волонтерську організацію, відповідно маючи певні менеджерські права.

Клієнт має сценарії:

- пошук волонтерських організацій;
- пошук волонтерських подій для ремонту побутової техніки;
- пошук волонтерів;
- пошук відділень сервісу з ремонту побутової техніки;
- перегляд інформації про волонтерську організацію;
- перегляд волонтерських подій за волонтерською організацією;
- перегляд інформації про волонтерську подію;
- перегляд інформації про майстра;
- перегляд сервісів ремонту побутової техніки;
- перегляд інформації про сервіс з ремонту побутової техніки.

За наявності авторизації клієнт має сценарії:

- перегляд власних пристроїв побутової техніки;
- керування даними власного пристрою побутової техніки;
- подання заявки на віртуальне консультування щодо ремонту побутової техніки;
- оцінювання послуги з ремонту;

- подання заявки на ремонт у відділенні сервісу;
- відміна заявки на віртуальне консультування;
- керування даними облікового запису;
- перегляд ремонтів пристрою;
- перегляд заявок на ремонт побутової техніки;
- подання заявки на участь у волонтерській події;
- відміна заявки на участь у волонтерській події.
- відміна заявки на ремонт у відділенні сервісу.

Сценарії працівника сервісу з ремонту побутової техніки:

- керування заявкою на ремонт побутової техніки у відділенні сервісу;
- керування даними відділення сервісу ремонту побутової техніки;
- визначення прав користувача як працівника відділення сервісу ремонту побутової техніки;
- визначення послуг з ремонту, що надаються відділенням сервісу ремонту побутової техніки;
- визначення наявних деталей для ремонту у відділенні;
- визначення розкладу роботи працівника;
- визначення власних вмінь;
- подання заявки на ремонт у відділенні сервісу;
- перегляд доступних деталей для моделі пристрою побутової техніки;
- визначення виконаної ремонтної роботи над пристроєм побутової техніки;
- перегляд виконаних ремонтних робіт;
- пошук волонтерських організацій;
- пошук волонтерських подій для ремонту побутової техніки;
- пошук волонтерів;
- пошук відділень сервісу з ремонту побутової техніки;
- перегляд інформації про волонтерську організацію;
- перегляд волонтерських подій за волонтерською організацією;
- перегляд інформації про волонтерську подію;

- перегляд інформації про майстра;
- перегляд сервісів ремонту побутової техніки;
- перегляд інформації про сервіс з ремонту побутової техніки;
- керування даними облікового запису;
- перегляд ремонтів пристрою;
- перегляд заявок на ремонт побутової техніки.

Сценарії волонтера (рис. 3.1):

- пошук волонтерських організацій;
- пошук волонтерських подій для ремонту побутової техніки;
- пошук волонтерів;
- пошук відділень сервісу з ремонту побутової техніки;
- перегляд інформації про волонтерську організацію;
- перегляд волонтерських подій за волонтерською організацією;
- перегляд інформації про волонтерську подію;
- перегляд інформації про майстра;
- перегляд сервісів ремонту побутової техніки;
- перегляд інформації про сервіс з ремонту побутової техніки;
- відміна заявки на віртуальне консультування;
- керування даними облікового запису;
- перегляд ремонтів пристрою;
- перегляд заявок на ремонт побутової техніки;
- подання заявки на участь у волонтерській події;
- відміна заявки на участь у волонтерській події;
- керування заявкою на ремонт побутової техніки під час волонтерської події;
- перегляд власних запланованих волонтерських подій;
- підтвердження заявки на віртуальне консультування.

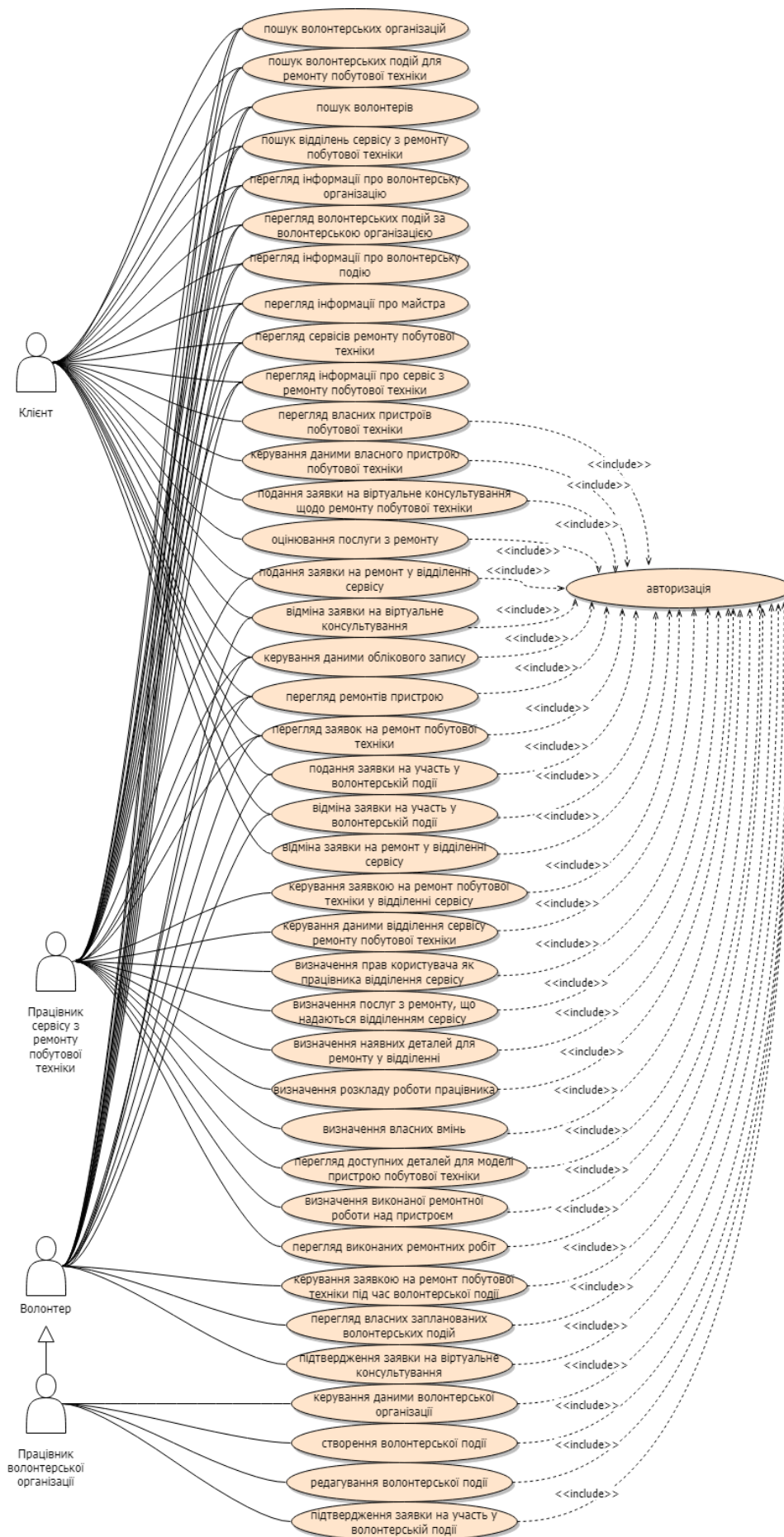


Рисунок 3.1 – Діаграма варіантів використання програми

Сценарії працівника волонтерської організації включають всі сценарії волонтера та додатково (рис. 3.1):

- керування даними волонтерської організації;
- створення волонтерської події;
- редагування волонтерської події;
- підтвердження заявки на участь у волонтерській події.

3.2 Рішення щодо вибору засобів розробки

При прийнятті рішень щодо засобів розробки, по-перше, було враховано, що різні сервіси, окремі відділення, користувачі працюють з програмою, а тому має надаватися віддалений доступ універсальним способом, відповідно програма має бути вебзастосунком. Розробка вебзастосунку вже передбачає використання мов CSS і HTML. Проте основана мова розробки може бути різною. Враховуючи функціональні вимоги, необхідність реалізації методу класифікації, порівняння включало мову Python [8] та мову PHP (табл. 3.1).

Таблиця 3.1 – Порівняння мов для розробки програмного забезпечення для сервісу ремонту побутової техніки

Критерій	Python	PHP
Загальний рівень безпеки	Вищий	Нижчий
Читабельність коду	Вища	Дещо нижча
Наявність і керування додатковими бібліотеками	Високий	Посередній
Бібліотеки, достатні для реалізації методу класифікації заявок на ремонт побутової техніки	Високий	Низький
Вебфреймворки	Високий	Високий

За сукупністю переваг, враховуючи зокрема те, що розробка відноситься не тільки до веброзробки, але і реалізує метод класифікації заявок на ремонт побутової техніки, а з надійними бібліотеками машинного навчання, роботи з даними, Python значно переважає PHP, його і було обрано в результаті.

3.3 Визначення структури даних у базі даних

База даних (БД) має чітко визначену структуру, яка враховує всі вимоги, описані в технічному завданні, сценарії роботи, а також результати виконання першого розділу.

Кожен користувач має обліковий запис, а збереження даних цього облікового запису виконується в таблиці `homeapprepairer_appuser` з полями:

- `id` – номер, який унікально характеризує користувача за його обліковим записом;
- `username` – логін, який використовується користувачем для авторизації у програмі;
- `town` – місто, в якому знаходиться зараз або проживає загалом користувач (перелік міст має бути заданий програмно, адже такий сервіс може бути щонайменше спочатку обмежений країною або взагалі містом);
- `first_name` і `last_name` – відповідно ім'я та прізвище користувача;
- `image` – зображення, де показана людина, що працює за цим обліковим записом;
- `email` – поштова скринька користувача;
- `password` – пароль користувача;
- `active` – цей користувач може авторизуватися в програмі і працювати під цим обліковим записом або він заблокований;
- `superuser` – звичайний користувач або працює як адміністратор цієї програми;
- `lastlog` – час останньої авторизації у програмі;

– phone – номер телефону користувача.

Таблиця БД `homeappprepaier_employee` містить ті поля, які характеризують окремо працівника сервісу ремонту побутової техніки окремо від загальних даних користувача:

– id – номер, який унікально характеризує працівника сервісу ремонту побутової техніки;

– appuser – номер облікового запису користувача, за яким працює цей працівник (зовнішній ключ до таблиці `homeappprepaier_appuser`);

– repairshop – пункт або магазин (загалом відділення сервісу), де цей працівник працює (зовнішній ключ до таблиці `homeappprepaier_repairshop`);

– sinedate – дата, з якої працює цей працівник у цьому сервісі ремонту побутової техніки.

Сервіси ремонту побутової техніки, які надають свої послуги на прибутковій основі, тобто за гроші, знаходяться в таблиці `homeappprepaier_repairstervice` з полями:

– id – номер, який унікально характеризує сервіс ремонту побутової техніки;

– caption – назва сервісу ремонту побутової техніки;

– image – зображення, де показано приміщення або загалом цей сервіс ремонту побутової техніки;

– website – посилання на вебсайт конкретно цього сервісу ремонту побутової техніки;

– textabout – текстова інформація стосовно даного сервісу ремонту побутової техніки;

– manageremployee – працівник, що виконує роль менеджера.

Відділення сервісу ремонту побутової техніки зібрані в таблиці `homeappprepaier_repairshop` з полями:

– id – номер, який унікально характеризує відділення сервісу ремонту побутової техніки;

– caption – назва відділення сервісу ремонту побутової техніки;

- image – зображення, де показано приміщення або умови конкретно цього відділення сервісу ремонту побутової техніки;
- repairservice – сервіс ремонту побутової техніки, до якого належить це відділення (зовнішній ключ до таблиці homeappprepaierer_repairservice);
- town – місто, де відділення сервісу ремонту побутової техніки знаходиться;
- longaddress – повна адреса, за якою відділення сервісу ремонту побутової техніки знаходиться;
- website – посилання на вебсайт конкретно цього відділення сервісу ремонту побутової техніки;
- textabout – текстова інформація стосовно цього відділення сервісу ремонту побутової техніки;
- reviewscore – оцінка з відгуків клієнтів стосовно цього відділення сервісу ремонту побутової техніки;
- manageremployee – працівник, який є основним менеджером цього відділення сервісу ремонту побутової техніки, тобто він відповідає за розподіл замовлень для виконання (зовнішній ключ до таблиці homeappprepaierer_employee).

Окрім прибуткових сервісів ремонту побутової техніки ремонт може виконуватися у волонтерських організаціях, тому дані таких організацій зібрані в таблиці homeappprepaierer_volunteerrepair з полями:

- id – номер, який унікально характеризує волонтерську організацію з ремонту побутової техніки;
- caption – назва волонтерської організації з ремонту побутової техніки;
- town – місто, де волонтерська організація з ремонту побутової техніки знаходиться;
- longaddress – повна адреса, за якою організація з ремонту побутової техніки знаходиться;

- `textabout` – текстова інформація стосовно цієї організації з ремонту побутової техніки;

- `website` – посилання на вебсайт конкретно цієї організації з ремонту побутової техніки;

- `managervolunteer` – волонтер, який є основним менеджером цієї організації з ремонту побутової техніки, тобто він відповідає за оброблення заявок щодо ремонту (зовнішній ключ до таблиці `homeappprepaireer_volunteer`).

Волонтерські організації не є прибутковими, не отримують гроші за ремонт техніки, а підтримуються волонтерами, які і допомагають виконати або виконують такий ремонт. Окрім того волонтери не обов'язково є представниками таких організацій, вони можуть і самостійно консультувати інших користувачів, наприклад, дистанційно щодо виконання ремонту побутової техніки, якщо це можливо зробити. Відповідно волонтери є користувачами програми, а їх відмінні поля з таблиці `homeappprepaireer_volunteer` включають:

- `id` – номер, який унікально характеризує волонтера;

- `appuser` – обліковий запис користувача, використовуваний волонтером (зовнішній ключ до таблиці `homeappprepaireer_appuser`);

- `reviewscore` – оцінка з відгуків клієнтів стосовно цього волонтера;

- `ready` – готовність надавати віртуальна консультації.

Волонтери і працівники сервісу ремонту побутової техніки мають інформацію про себе, що складається з даних облікового запису з таблиці `homeappprepaireer_appuser` і відповідних даних таблиць `homeappprepaireer_volunteer` і `homeappprepaireer_employee`. Кожен волонтер або працівник сервісу ремонту побутової техніки повинні спочатку мати обліковий запис, тож відношення між таблицями `homeappprepaireer_appuser` і `homeappprepaireer_volunteer`, як і таблицями БД `homeappprepaireer_employee` і `homeappprepaireer_appuser` один до одного.

Типові процедури з ремонту побутової техніки, які здатні виконувати волонтери, відрізняються від волонтера до волонтера, а необхідні, щоб

вказати людям, які шукають допомогу в ремонті, з чим такий волонтер в принципі може допомогти, тобто з чим до нього можна звертатися. Типові процедури з ремонту побутової техніки, які здатні виконувати волонтери, містяться в таблиці БД `homeapprepairer_volunteer_typicalrepairjob` з полями:

- `id` – номер, який унікально характеризує типову процедури з ремонту побутової техніки, яку виконує волонтер;
- `volunteer` – номер волонтера, який здатність виконувати цю типову процедуру має (зовнішній ключ до таблиці `homeapprepairer_volunteer`);
- `typicalrepairjob` – типова процедура з ремонту побутової техніки, яку може виконувати волонтер (зовнішній ключ до таблиці `homeapprepairer_typicalrepairjob`).

Віртуальна зустріч, з проханням щодо якої може звертатися користувач (клієнт) до волонтера, покликана реалізувати підтримку клієнта у виконанні ремонту побутової техніки віддалено. Спочатку клієнт звертається з проханням, а потім сам волонтер це прохання задовольняє або ні. Дані віртуальних зустрічей містяться в таблиці БД `homeapprepairer_volunteer_virtualmeeting`:

- `id` – номер, який унікально характеризує віртуальну зустріч волонтера з клієнтом;
- `volunteer` – волонтер, до якого з проханням віртуальної зустрічі звернувся клієнт (зовнішній ключ до таблиці `homeapprepairer_volunteer`);
- `appuser` – користувач (клієнт), який звернувся з проханням про зустріч (зовнішній ключ до таблиці `homeapprepairer_appuser`);
- `meetingdate` – призначений час зустрічі для підтримки ремонту побутової техніки;
- `zoommeeting` – посилання на Zoom-конференцію для проведення зустрічі для підтримки ремонту побутової техніки;
- `createdate` – час створення прохання про зустріч для підтримки ремонту побутової техніки клієнтом;

- problem – основна проблема, з якою звертається клієнт, або деталі щодо зустрічі, які б хотів врахувати клієнт;
- answer – текстова відповідь від волонтера клієнту щодо його прохання і зустрічі;
- state – стан зустрічі для підтримки ремонту побутової техніки (подано прохання, прохання підтверджено, прохання скасовано);
- ownercontact – дозвіл від клієнта (власника побутової техніки) щодо надання його контактних даних для зв'язку (номеру телефону);
- volunteercontact – дозвіл від волонтера щодо надання його контактних даних для зв'язку (номеру телефону).

При створенні прохання про зустріч для підтримки ремонту побутової техніки клієнтом він має вказати, яка саме побутова техніка має ремонтуватися. Ці дані містяться в окремій таблиці БД `homeappprepaierer_volunteer_virtualmeeting_item`, адже таких пристроїв може бути декілька (хоча найчастіше все ж побутова техніка буде представлена одним пристроєм, однак можливість представити декілька речей в одному зверненні має бути), з полями:

- id – номер, який унікально характеризує пристрій побутової техніки, який під час віртуальної зустрічі волонтера з клієнтом клієнт прагне відремонтувати;
- volunteer_virtualmeeting – зустріч для підтримки ремонту побутової техніки, за якою цей пристрій прагне відремонтувати клієнт (зовнішній ключ до таблиці `homeappprepaierer_volunteer_virtualmeeting`);
- applianceitem – номер пристрою побутової техніки клієнта, який окремо визначений, що має бути відремонтовано (зовнішній ключ до таблиці `homeappprepaierer_applianceitem`);
- problem – проблема, з якою в роботі або стані цього пристрою побутової техніки зіштовхнувся власник речі;
- ownerjobdone – процедури, які вже намагався виконати власник цього пристрою;

- `jobdone` – процедури з ремонту, які вже виконав волонтер щодо цієї речі (якщо волонтер бажає це зафіксувати в програмі);

- `jobtodo` – процедури з ремонту, які потрібно ще виконати волонтеру щодо цієї речі (якщо волонтер бажає це зафіксувати в програмі);

- `state` – стан роботи над ремонтом цієї речі.

Волонтерські організації працюють над ремонтами через організацію окремих подій, на яких волонтери допомагають відремонтувати власникам свої пристрої побутової техніки. Дані про події для підтримки ремонтів побутової техніки волонтерами зібрані в таблиці БД `homeapprepaire_repairevent` з полями:

- `id` – номер, який унікально характеризує подію, що проводиться волонтерською організацією для ремонту побутової техніки;

- `volunteerrepair` – волонтерська організація, яка проводить цю подію для ремонту побутової техніки (зовнішній ключ до таблиці `homeapprepaire_volunteerrepair`);

- `eventcaption` – назва події;

- `eventstart` – час початку події, що проводиться волонтерською організацією для ремонту побутової техніки;

- `eventfinish` – час закінчення події, що проводиться волонтерською організацією для ремонту побутової техніки;

- `image` – зображення, де проілюстровано подію, що проводиться волонтерською організацією для ремонту побутової техніки;

- `textabout` – текстовий опис події.

Волонтери, які беруть участь в конкретній події, що проводиться волонтерською організацією для ремонту побутової техніки, знаходяться в таблиці `homeapprepaire_repairevent_volunteer` з полями:

- `id` – номер участі волонтера у події, що проводиться волонтерською організацією для ремонту побутової техніки;

– `repairvent` – подія, що проводиться волонтерською організацією для ремонту побутової техніки, в якій бере участь волонтер (зовнішній ключ до таблиці `homeapprepairer_repairvent`);

– `volunteer` – волонтер, який готовий взяти участь у події, що проводиться волонтерською організацією для ремонту побутової техніки (зовнішній ключ до таблиці `homeapprepairer_volunteer`);

– `approve` – підтвердження участі в події.

Заявка на ремонт побутової техніки під час волонтерської події знаходиться в таблиці `homeapprepairer_repairvent_application` з полями:

– `id` – номер заявки на ремонт побутової техніки під час волонтерської події;

– `repairvent` – волонтерська подія, під час якої має відбутися ремонт побутової техніки (зовнішній ключ до таблиці `homeapprepairer_repairvent`);

– `managervolunteer` – волонтер, який виконує роль менеджера стосовно даної заявки (зовнішній ключ до таблиці `homeapprepairer_volunteer`);

– `appuser` – клієнт, який створив цю заявку на ремонт побутової техніки під час волонтерської події (зовнішній ключ до таблиці `homeapprepairer_appuser`);

– `createdate` – час створення заявки на ремонт побутової техніки під час волонтерської події;

– `problem` – основна проблема, з якою звертається клієнт, або деталізація заявки;

– `answer` – текстова відповідь від волонтера клієнту щодо його заявки;

– `state` – стан заявки на ремонт побутової техніки під час волонтерської події (подано, підтверджено, скасовано);

– `ownercontact` – дозвіл від клієнта (власника побутової техніки) щодо надання його контактних даних для зв'язку (номера телефону);

– `volunteercontact` – дозвіл від волонтера-менеджера щодо надання його контактних даних для зв'язку (номера телефону).

Заявка на ремонт побутової техніки під час волонтерської події може деталізуватися рядом конкретних пристроїв, які користувач прагне відремонтувати. Ці дані містяться в окремій таблиці БД `homeappraier_repairevent_application_item` з полями:

- `id` – номер, який унікально характеризує пристрій побутової техніки, який під час волонтерської події клієнт прагне відремонтувати;
- `repairevent_application` – заявка на ремонт побутової техніки під час волонтерської події, до якої цей пристрій належить (зовнішній ключ до таблиці `homeappraier_repairevent_application`);
- `applianceitem` – номер пристрою побутової техніки клієнта, який належить до заявки на ремонт побутової техніки під час волонтерської події (зовнішній ключ до таблиці `homeappraier_applianceitem`);
- `problem` – проблема, з якою в роботі або стані цього пристрою побутової техніки зіштовхнувся власник речі;
- `ownerjobdone` – процедури, які вже намагався виконати власник цього пристрою;
- `state` – стан роботи над ремонтом цього пристрою.

Уся побутова техніка розділена на категорії, які знаходяться в таблиці `homeappraier_appliancecategory` з полями:

- `id` – номер, який унікально характеризує категорію побутової техніки;
- `category` – назва категорії побутової техніки.

Пристрої, які як раз і є побутовою технікою, яку має конкретний користувач і яку він може ремонтувати на основі сервісів, які пропонуються програмою, містяться в таблиці БД `homeappraier_applianceitem` з полями:

- `id` – номер, який унікально характеризує категорію побутової техніки;
- `appliancemodel` – модель пристрою побутової техніки, якщо вона відома (зовнішній ключ до таблиці `homeappraier_appliancemodel`);

– appuser – клієнт, який володіє цим пристроєм (зовнішній ключ до таблиці homeappraier_appuser);

– productionyear – рік, коли було виготовлено цей пристрій;

– textabout – текстовий опис пристрою побутової техніки;

– serialnumber – серійний номер пристрою побутової техніки;

– image – зображення, де показано цей пристрій побутової техніки.

Виробники пристроїв побутової техніки зведені до таблиці БД homeappraier_appliancemanufacturer з полями:

– id – номер, який унікально характеризує виробника побутової техніки;

– manufacturer – назва виробника побутової техніки.

Моделі пристроїв побутової техніки зведені до таблиці БД homeappraier_appliancemodel з полями:

– id – номер, який унікально характеризує модель пристрою побутової техніки;

– manufacturer – номер виробника, який виготовляє цю модель пристрою побутової техніки (зовнішній ключ до таблиці homeappraier_appliancemanufacturer);

– model – назва моделі пристрою побутової техніки;

– appliancecategory – номер категорії побутової техніки (зовнішній ключ до таблиці homeappraier_appliancecategory).

В ремонті побутової техніки можуть задіюватися (тобто фактично ремонтуватися) або замінюватися різні деталі. Для збереження даних типів цих деталей виділено таблицю homeappraier_typicaldetail з полями:

– id – номер, який унікально характеризує тип деталі;

– detailname – назва типу деталі побутової техніки.

Типи деталей деталізуються конкретними деталями для конкретних моделей пристроїв побутової техніки, що містяться в таблиці БД homeappraier_detail з полями:

- id – номер, який унікально характеризує деталь до моделі пристрою побутової техніки;
- typicaldetail – тип деталі, який має ця деталь до моделі пристрою побутової техніки (зовнішній ключ до таблиці homeappprepaierer_typicaldetail);
- appliancemodel – модель пристрою побутової техніки, до якої належить ця деталь (зовнішній ключ до таблиці homeappprepaierer_appliancemodel);
- caption – повна назва деталі до моделі пристрою побутової техніки;
- abouttext – текстовий опис деталі до моделі пристрою побутової техніки;
- image – зображення, де показано цю модель пристрою побутової техніки.

Деталі до пристрою побутової техніки використовуються в ремонті сервісами. Тобто волонтерські організації не мають своїх деталей, вони направляють роботу на усунення інших варіантів поламки або власник має придбати необхідну деталь. У той же час у сервісу з ремонту побутової техніки такі деталі можуть бути наявні у певній кількості. Дані про наявність деталей до пристроїв побутової техніки містяться в таблиці homeappprepaierer_detailshop з полями:

- id – номер, який унікально характеризує наявність деталі до пристрою побутової техніки у відділенні сервісу;
- detail – деталь до пристрою побутової техніки, наявність якої визначається (зовнішній ключ до таблиці homeappprepaierer_detail);
- repairshop – відділення сервісу з ремонту побутової техніки (зовнішній ключ до таблиці homeappprepaierer_repairshop), де ця деталь наявна в такій кількості;
- number – кількість деталей побутової техніки, що наявна у цьому відділенні сервісу з ремонту побутової техніки;
- price – вартість однієї такої деталі до пристрою побутової техніки на поточний момент.

Окрім заявок на ремонт під час події, що організовується волонтерською організацією, проведення віртуальних зустрічей ще одним варіантом ремонту є ремонт у сервісі, який здійснюється за плату. Заявки на такий ремонт містяться в таблиці `homeapprepairer_repairorder` з полями:

- `id` – номер, який унікально характеризує заявку на ремонт побутової техніки в сервісі;
- `repairshop` – відділення сервісу з ремонту побутової техніки (зовнішній ключ до таблиці `homeapprepairer_repairshop`), де цей ремонт має бути здійснено;
- `appuser` – користувач, який подав цю заявку на ремонт побутової техніки у відділенні сервісу (зовнішній ключ до таблиці `homeapprepairer_appuser`);
- `createdate` – час створення заявки на ремонт побутової техніки у відділенні сервісу;
- `orderdate` – час подання заявки на ремонт побутової техніки у відділенні сервісу;
- `receivedate` – час отримання пристроїв з заявки на ремонт побутової техніки у відділенні сервісу;
- `waitdate` – час очікуваного завершення ремонту побутової техніки у відділенні сервісу за заявкою;
- `waitresult` – очікуваний результат ремонту (тобто фактично, чи очікує сервіс, що таке замовлення вдасться загалом завершити успішно);
- `problem` – основна проблема, з якою звертається клієнт, або деталі щодо заявки, які б хотів врахувати клієнт;
- `answer` – загальна текстова відповідь від сервісу клієнту щодо його заявки на ремонт побутової техніки у відділенні сервісу;
- `state` – стан роботи над заявкою на ремонт побутової техніки у відділенні сервісу (створено, в обробці, підтверджено, скасовано, отримано, в ремонті, відремонтовано, часткові проблеми, не піддається ремонту, віддано);

- ownercontact – дозвіл від клієнта (власника побутової техніки) щодо надання його контактних даних для зв'язку (номера телефону);
- manageremployee – працівник сервісу, який виконує роль менеджера за цим замовленням (зовнішній ключ до таблиці homeappprepairemployee);
- price – вартість виконання ремонту побутової техніки у відділенні сервісу;
- check – позначка про сплату всієї суми за виконання ремонту побутової техніки у відділенні сервісу.

Заявки на ремонт побутової техніки у відділенні сервісу також можуть включати декілька позицій, які представлені конкретними пристроями, які користувач (клієнт) віддає для ремонту. Ці дані містяться в таблиці homeappprepairerepairorder_position з полями:

- id – номер, який унікально характеризує позицію заявки на ремонт побутової техніки в сервісі;
- repairorder – заявка на ремонт побутової техніки у відділенні сервісу, до якої належить ця позиція (зовнішній ключ до таблиці homeappprepairerepairorder);
- applianceitem – пристрій побутової техніки у власництві цього клієнта, який має бути відданий у ремонт за заявкою (зовнішній ключ до таблиці homeappprepaireapplianceitem);
- problem – проблема, з якою в роботі або стані цього пристрою побутової техніки зіштовхнувся власник речі;
- answer – відповідь від сервісу (тобто працівника, який виконував ремонт цього пристрою) щодо цього пристрою;
- ownerjobdone – процедури, які вже намагався виконати власник цього пристрою;
- jobdone – процедури з ремонту, які вже виконав працівник сервісу з ремонту побутової техніки щодо цієї речі (якщо працівник бажає це зафіксувати в програмі);

– `jobtodo` – процедури з ремонту, які потрібно ще виконати працівнику сервісу з ремонту побутової техніки щодо цієї речі (якщо працівник бажає це зафіксувати в програмі);

– `state` – стан роботи над ремонтом цієї речі;

– `plannedstart` – запланований (або фактичний, коли ремонт розпочато) час початку роботи;

– `plannedendv` – запланований (або фактичний, коли ремонт закінчено) час завершення роботи.

Типи процедур з ремонту побутової техніки необхідна для того, щоб кожен волонтер, відділення сервісу могли визначити, які вони роботи готові виконувати, для чого в них є відповідні здібності та можливості. Типи процедур з ремонту побутової техніки містяться в таблиці БД `homeappprepaire_typicalrepairjob` з полями:

– `id` – номер, який унікально характеризує тип процедур з ремонту побутової техніки;

– `appliancecategory` – категорія побутової техніки, для якої така робота виконується;

– `jobname` – назва процедури з ремонту побутової техніки.

Типи процедур з ремонту побутової техніки, які виконуються в конкретному відділенні сервісу з ремонту побутової техніки, містяться в таблиці БД `homeappprepaire_repairshop_typicalrepairjob` з полями:

– `id` – номер, який унікально характеризує тип процедури з ремонту побутової техніки, що виконується в конкретному відділенні сервісу з ремонту побутової техніки;

– `repairshop` – відділення сервісу з ремонту побутової техніки (зовнішній ключ до таблиці `homeappprepaire_repairshop`), де цей тип процедури з ремонту побутової техніки виконується;

– `typicalrepairjob` – тип процедури з ремонту побутової техніки, що виконується в цьому відділенні сервісу (зовнішній ключ до таблиці `homeappprepaire_typicalrepairjob`);

- price – стандартна вартість виконання такої процедури з ремонту побутової техніки;

- duration – стандартна тривалість виконання такої процедури з ремонту побутової техніки.

Типи процедур з ремонту побутової техніки, які виконуються конкретним працівником сервісу з ремонту побутової техніки, містяться в таблиці БД `homeappprepairemployee_typicalrepairjob` з полями:

- id – номер, який унікально характеризує тип процедури з ремонту побутової техніки, що виконується конкретним працівником сервісу з ремонту побутової техніки;

- typicalrepairjob – тип процедури з ремонту побутової техніки, що виконується цим працівником сервісу (зовнішній ключ до таблиці `homeappprepairemployee_typicalrepairjob`);

- employee – працівник сервісу з ремонту побутової техніки, який здатен виконувати цей тип процедури з ремонту побутової техніки (зовнішній ключ до таблиці `homeappprepairemployee`);

- specification – особливості виконання цим працівником сервісу з ремонту побутової техніки цього типу процедури з ремонту побутової техніки.

Працівники сервісу працюють не весь час, а в конкретні проміжки часу (наприклад, повний традиційний робочий день у будні дні). Проміжки часу роботи працівника сервісу з ремонту побутової техніки знаходяться в таблиці `homeappprepairemployee_schedule` з полями:

- id – номер, який унікально характеризує проміжок часу роботи працівника сервісу з ремонту побутової техніки за день;

- employee – працівник сервісу з ремонту побутової техніки, для якого визначений цей проміжок часу роботи на день (зовнішній ключ до таблиці `homeappprepairemployee`);

- typical – розглядається як типовий варіант (наприклад, час роботи в понеділок має повторюватися кожен понеділок) або ні;

- starttime – час початку роботи в цей день працівником сервісу з ремонту побутової техніки (дата і час);

- finishtime – час закінчення роботи в цей день працівником сервісу з ремонту побутової техніки (дата і час).

Конкретні позиції заявки на ремонт побутової техніки в сервісі визначаються для виконання конкретним працівником сервісу з ремонту побутової техніки. Оскільки процедури з ремонту можуть бути різними, відповідно і задіяні працівники можуть бути різними, утворюючи певний ланцюжок. Тоді всі задіяні працівники сервісу з ремонту побутової техніки вносяться в таблицю `homeapppreairer_repairorder_position_employee` з полями:

- id – номер, який унікально характеризує працівника сервісу з ремонту побутової техніки, що виконує роботи над цією позицією заявки на ремонт побутової техніки в сервісі;

- repairorder_position – номер позиції заявки на ремонт побутової техніки в сервісі, роботи з якою має виконувати цей працівник (зовнішній ключ до таблиці `homeapppreairer_repairorder_position`);

- employee – працівник сервісу з ремонту побутової техніки, який має виконати ці роботи (зовнішній ключ до таблиці `homeapppreairer_employee`);

- function – функція або конкретно опис того, що має виконати цей працівник сервісу з ремонту побутової техніки;

- plannedstart – запланований початок робіт працівником сервісу з ремонту побутової техніки над цією позицією заявки на ремонт побутової техніки в сервісі;

- plannedend – заплановане закінчення робіт працівником сервісу з ремонту побутової техніки над цією позицією заявки на ремонт побутової техніки в сервісі;

- state – стан виконання цих робіт;

- result – результат проведення цих ремонтних робіт;

- answer – відповідь працівника сервісу з ремонту побутової техніки щодо виконання цих робіт.

Конкретні ремонтні роботи, які виконуються в сервісі конкретним працівником над конкретною позицією заявки на ремонт побутової техніки в сервісі, знаходяться в таблиці `homeappprepaierer_repairjob` з полями:

- `id` – номер, який унікально характеризує конкретну ремонтну роботу, яка виконуються в сервісі конкретним працівником над конкретною позицією заявки на ремонт побутової техніки в сервісі;

- `repairshop_typicalrepairjob` – тип процедури з ремонту побутової техніки, що виконується в конкретному відділенні сервісу з ремонту побутової техніки і за яким цим працівником сервісу має бути виконано роботу (зовнішній ключ до таблиці `homeappprepaierer_repairshop_typicalrepairjob`);

- `repairorder_position_employee` – номер працівника сервісу з ремонту побутової техніки, що виконує роботи над позицією заявки на ремонт побутової техніки в сервісі, який і має виконати цю конкретну роботу (зовнішній ключ до таблиці `repairorder_position_employee`);

- `plannedstart` – запланований початок виконання цієї роботи працівником сервісу з ремонту побутової техніки над цією позицією заявки на ремонт побутової техніки в сервісі;

- `plannedend` – заплановане закінчення роботи працівником сервісу з ремонту побутової техніки над цією позицією заявки на ремонт побутової техніки в сервісі;

- `textabout` – текстовий опис відповіді працівника сервісу з ремонту побутової техніки щодо виконання цієї роботи;

- `state` – стан виконання цієї роботи;

- `result` – результат проведення цієї ремонтної роботи;

- `price` – вартість виконання цієї конкретної роботи в цьому конкретно випадку.

Деталі, які ремонтуються або застосовуються для виконання ремонту, містяться в таблиці `homeappprepaierer_repairjob_detail` з полями:

- id – номер, який унікально характеризує запис про деталі, які ремонтуються або застосовуються для виконання ремонту;
- repairjob – номер конкретної ремонтної роботи, яка виконується в сервісі конкретним працівником над конкретною позицією заявки на ремонт побутової техніки в сервісі і для якої використано цю деталь (зовнішній ключ до таблиці homeapprepairer_repairjob);
- detail – деталь, яка були використана або відремонтована під час виконання ремонту (зовнішній ключ до таблиці homeapprepairer_detail);
- number – кількість таких деталей;
- operation – тип операції (ремонт або заміна);
- price – вартість, якою було оцінено саму цю деталь (роботи фіксуються окремо).

Оцінки користувачами волонтерів містяться в таблиці homeapprepairer_volunteerpoint з полями:

- id – номер, який унікально характеризує оцінку користувачем волонтера;
- volunteer – волонтер, про якого створено оцінку (зовнішній ключ до таблиці homeapprepairer_volunteer);
- appuser – користувач, який виконав оцінку (зовнішній ключ до таблиці homeapprepairer_appuser);
- pointdate – час збереження оцінки користувачем волонтера;
- point – значення оцінки.

Оцінки користувачами сервісів з ремонту побутової техніки містяться в таблиці БД (рис. 3.2) homeapprepairer_shoppoint з полями:

- id – номер, який унікально характеризує оцінку користувачем сервісу з ремонту побутової техніки;
- repairshop – сервіс з ремонту побутової техніки, про який створено оцінку (зовнішній ключ до таблиці homeapprepairer_repairshop);
- appuser – користувач, який виконав оцінку (зовнішній ключ до таблиці homeapprepairer_appuser);

– pointdate – час збереження оцінки користувачем сервісу ремонту побутової техніки;

– point – значення оцінки.

Системою керування БД обрано PostgreSQL.

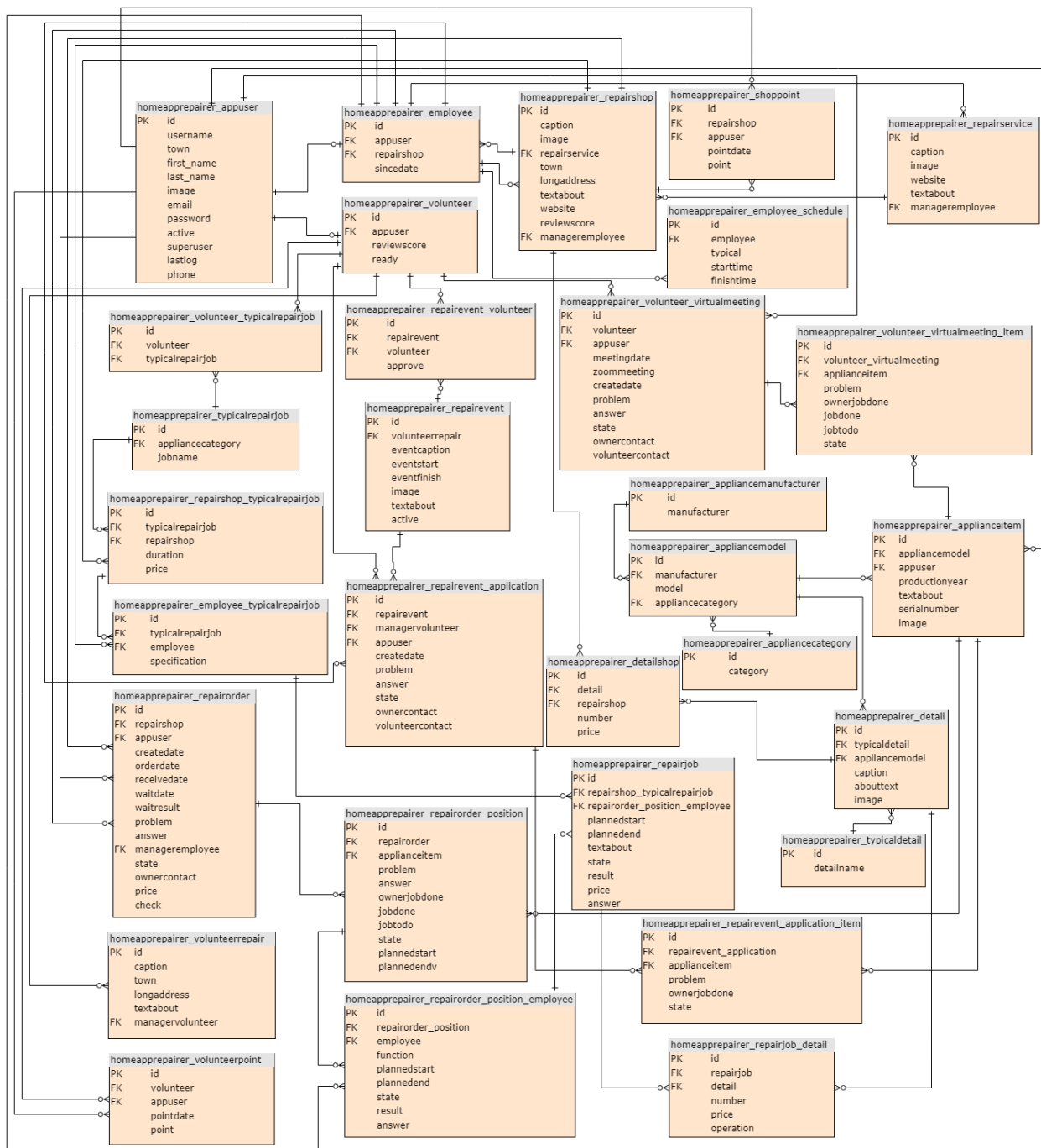


Рисунок 3.2 – Схема БД

3.4 Висновки за розділом 3

Усі функціональні вимоги до програми визначені у вигляді сценаріїв, за якими користувачі програми мають з нею працювати. Виходячи з вимог до програми, мова Python обрана для розробки. Описано всі таблиці БД, визначені за результатами проєктування, встановлено зв'язки між таблицями.

4 РЕАЛІЗАЦІЯ ПРОГРАМИ

4.1 Порядок роботи над заявками у програмі

Заявки бувають в програмі різних типів, але заявка на ремонт побутової техніки у відділенні сервісу є такою, що включає максимальну кількість можливих станів, відповідно від цього залежність послідовність виконання іншої функцій у програмі, тому розглянуто детально її (рис. 4.1).

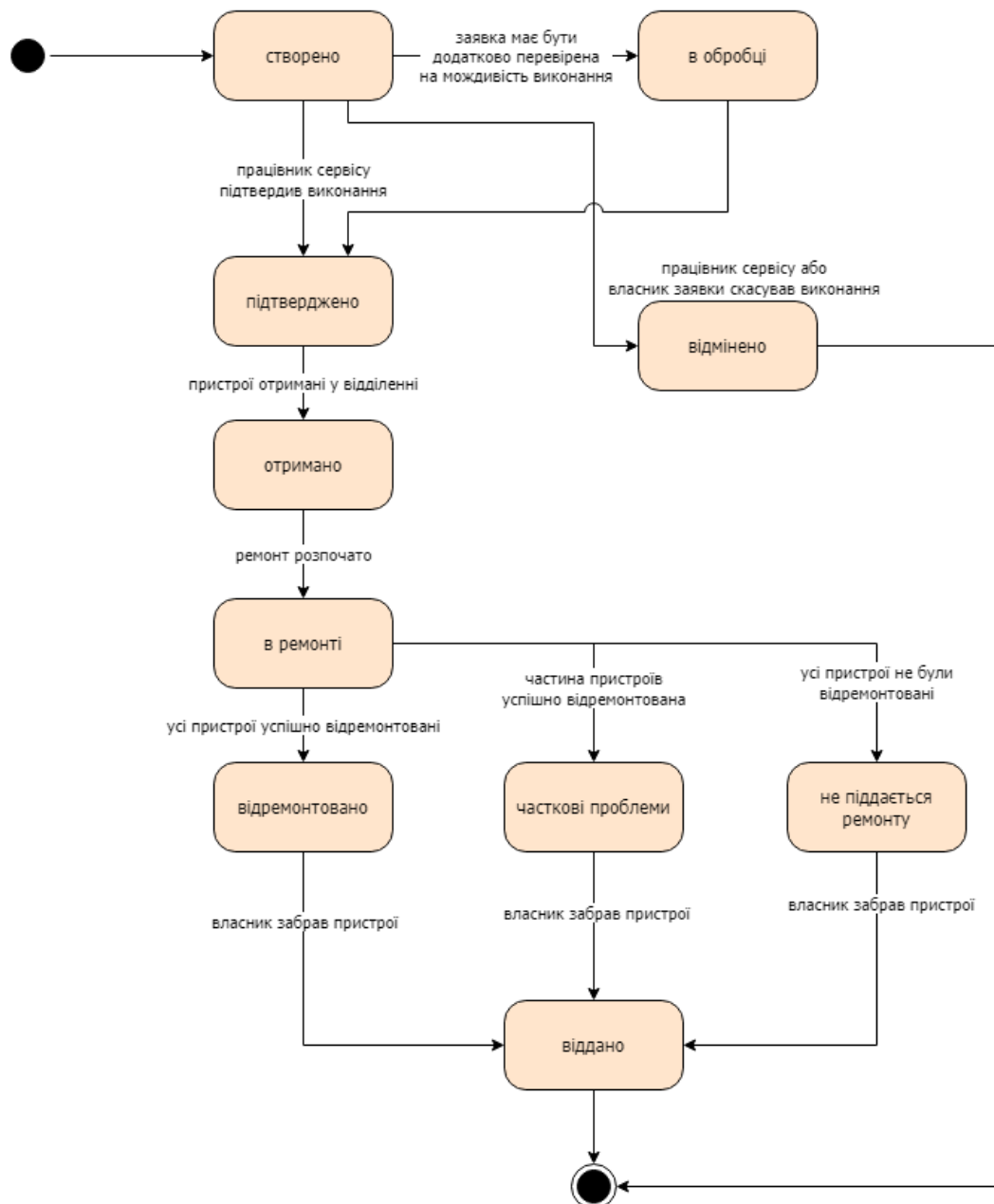


Рисунок 4.1 – Діаграма станів заявки на ремонт побутової техніки у відділенні сервісу

4.2 Програмна реалізація вебзастосунку

Як і було визначено у третьому розділі, програмна реалізація вебзастосунку виконана через використання фреймворку Django і забезпечення відповідних складових проєкту за цим фреймворком за визначеними сценаріями використання програми. Для цього було забезпечено ряд класів, що підтримують роботу відповідних моделей згідно зі структурою БД, таких, що використовуються при роботі представлень, які вже підтримують через створення функцій роботу кожного сценарію на основі визначення шаблону сторінки з допомогою шаблону [9] та формування даних, необхідних для роботи цього шаблону.

Модель користувача Appuser створена класом, який наслідує стандартний клас користувача з Django, розширюючи набір полів.

Модель працівника сервісу ремонту побутової техніки Employee створена класом, у складі якого визначено поля:

- appuser з типом ForeignKey – обліковий запис користувача, за яким працює цей працівник;
- repairshop з типом ForeignKey – відділення сервісу, де цей працівник працює;
- sinedate з типом DateField – дата, з якої працює цей працівник у цьому сервісі ремонту побутової техніки.

Модель сервісу ремонту побутової техніки Repairservice створена класом, у складі якого визначено поля:

- caption з типом CharField – назва сервісу ремонту побутової техніки;
- image з типом ImageField – зображення, де показано приміщення або загалом цей сервіс ремонту побутової техніки;
- website з типом URLField – посилання на вебсайт конкретно цього сервісу ремонту побутової техніки;
- textabout з типом CharField – текстова інформація стосовно даного сервісу ремонту побутової техніки;

– `manageremployee` з типом `ForeignKey` – працівник, який є основним менеджером цього сервісу ремонту побутової техніки.

Модель відділення сервісу ремонту побутової техніки `Repairshop` створена класом, у складі якого визначено поля:

– `caption` з типом `CharField` – назва відділення сервісу ремонту побутової техніки;

– `image` з типом `ImageField` – зображення, де показано приміщення або умови конкретно цього відділення сервісу ремонту побутової техніки;

– `repairservice` з типом `ForeignKey` – сервіс ремонту побутової техніки, до якого належить це відділення;

– `town` з типом `CharField` – місто, де відділення сервісу ремонту побутової техніки знаходиться;

– `longaddress` з типом `CharField` – повна адреса, за якою відділення сервісу ремонту побутової техніки знаходиться;

– `website` з типом `URLField` – посилання на вебсайт конкретно цього відділення сервісу ремонту побутової техніки;

– `textabout` з типом `CharField` – текстова інформація стосовно цього відділення сервісу ремонту побутової техніки;

– `reviewscore` з типом `FloatField` – оцінка з відгуків клієнтів стосовно цього відділення сервісу ремонту побутової техніки;

– `manageremployee` з типом `ForeignKey` – працівник, який є основним менеджером цього відділення сервісу ремонту побутової техніки.

Модель волонтерської організації `Volunteerrepair` створена класом, у складі якого визначено поля:

– `caption` з типом `CharField` – назва волонтерської організації з ремонту побутової техніки;

– `town` з типом `CharField` – місто, де волонтерська організація з ремонту побутової техніки знаходиться;

– `longaddress` з типом `CharField` – повна адреса, за якою організація з ремонту побутової техніки знаходиться;

- textabout з типом CharField – текстова інформація стосовно цієї організації з ремонту побутової техніки;
- website з типом URLField – посилання на вебсайт конкретно цієї організації з ремонту побутової техніки;
- managervolunteer з типом ForeignKey – волонтер, який є основним менеджером цієї організації з ремонту побутової техніки.

Модель волонтера Volunteer створена класом, у складі якого визначено поля:

- appuser з типом ForeignKey – обліковий запис користувача, використовуваний волонтером;
- reviewscore з типом FloatField – оцінка з відгуків клієнтів стосовно цього волонтера;
- typicalrepairjob з типом ManyToManyField – типові процедури з ремонту побутової техніки, які може виконувати волонтер;
- ready з типом BooleanField – готовність надавати віртуальна консультації щодо ремонту побутової техніки.

Модель віртуальної зустрічі, з проханням щодо якої може звертатися користувач (клієнт) до волонтера, Volunteer_virtualmeeting створена класом, у складі якого визначено поля:

- volunteer з типом ForeignKey – волонтер, до якого з проханням віртуальної зустрічі звернувся клієнт;
- appuser з типом ForeignKey – користувач (клієнт), який звернувся з проханням про зустріч;
- meetingdate з типом DateTimeField – призначений час зустрічі для підтримки ремонту побутової техніки;
- zoommeeting з типом CharField – посилання на Zoom-конференцію для проведення зустрічі для підтримки ремонту побутової техніки;
- createdate з типом DateTimeField – час створення прохання про зустріч для підтримки ремонту побутової техніки клієнтом;

- problem з типом CharField – основна проблема, з якою звертається клієнт, або деталі щодо зустрічі, які б хотів врахувати клієнт;
- answer з типом CharField – текстова відповідь від волонтера клієнту щодо його прохання і зустрічі;
- state з типом CharField – стан зустрічі для підтримки ремонту побутової техніки (даний тип в Django використовується для підтримки роботи з кортежами, які визначають можливі, тобто обмежені, варіанти вибору);
- ownercontact з типом BooleanField – дозвіл від клієнта (власника побутової техніки) щодо надання його контактних даних для зв'язку (номера телефону);
- volunteercontact з типом BooleanField – дозвіл від волонтера щодо надання його контактних даних для зв'язку (номера телефону).

Модель пристрою, що відноситься до прохання про волонтерську зустріч для підтримки ремонту побутової техніки клієнтом, Volunteer_virtualmeeting_item створена класом, у складі якого визначено поля:

- volunteer_virtualmeeting з типом ForeignKey – зустріч для підтримки ремонту побутової техніки, за якою цей пристрій прагне відремонтувати клієнт;
- applianceitem з типом ForeignKey – пристрій побутової техніки клієнта, який окремо визначений, що має бути відремонтовано;
- problem з типом CharField – проблема, з якою в роботі або стані цього пристрою побутової техніки зіштовхнувся власник речі;
- ownerjobdone з типом CharField – процедури, які вже намагався виконати власник цього пристрою;
- jobdone з типом CharField – процедури з ремонту, які вже виконав волонтер щодо цієї речі (якщо волонтер бажає це зафіксувати в програмі);

- `jobtodo` з типом `CharField` – процедури з ремонту, які потрібно ще виконати волонтеру щодо цієї речі (якщо волонтер бажає це зафіксувати в програмі);

- `state` з типом `CharField` – стан роботи над ремонтом цієї речі.

Модель події для підтримки ремонтів побутової техніки волонтерами `Rrepairevent` створена класом, у складі якого визначено поля:

- `volunteerrepair` з типом `ForeignKey` – волонтерська організація, яка проводить цю подію для ремонту побутової техніки;

- `eventcaption` з типом `CharField` – назва події;

- `eventstart` з типом `DateTimeField` – час початку події, що проводиться волонтерською організацією для ремонту побутової техніки;

- `eventfinish` з типом `DateTimeField` – час закінчення події, що проводиться волонтерською організацією для ремонту побутової техніки;

- `image` з типом `ImageField` – зображення, де проілюстровано подію, що проводиться волонтерською організацією для ремонту побутової техніки;

- `textabout` з типом `CharField` – текстовий опис події.

Модель заявки на ремонт побутової техніки під час волонтерської події `Repairevent_application` створена класом, у складі якого визначено поля:

- `repairevent` з типом `ForeignKey` – волонтерська подія, під час якої має відбутися ремонт побутової техніки;

- `managervolunteer` з типом `ForeignKey` – волонтер, який виконує роль менеджера стосовно даної заявки;

- `appuser` з типом `ForeignKey` – клієнт, який створив цю заявку на ремонт побутової техніки під час волонтерської події;

- `createdate` з типом `DateTimeField` – час створення заявки на ремонт побутової техніки під час волонтерської події;

- `problem` з типом `CharField` – основна проблема, з якою звертається клієнт, або деталізація заявки;

- `answer` з типом `CharField` – текстова відповідь від волонтера клієнту щодо його заявки;

- state з типом CharField – стан заявки на ремонт побутової техніки під час волонтерської події;
- ownercontact з типом BooleanField – дозвіл від клієнта (власника побутової техніки) щодо надання його контактних даних для зв'язку (номера телефону);
- volunteercontact з типом BooleanField – дозвіл від волонтера-менеджера щодо надання його контактних даних для зв'язку (номера телефону).

Модель пристрою з заявки на ремонт побутової техніки під час волонтерської події `Repairevent_application_item` створена класом, у складі якого визначено поля:

- `repairevent_application` з типом ForeignKey – заявка на ремонт побутової техніки під час волонтерської події, до якої цей пристрій належить;
- `applianceitem` з типом ForeignKey – пристрій побутової техніки клієнта, який належить до заявки на ремонт побутової техніки під час волонтерської події;
- `problem` з типом CharField – проблема, з якою в роботі або стані цього пристрою побутової техніки зіштовхнувся власник речі;
- `ownerjobdone` з типом CharField – процедури, які вже намагався виконати власник цього пристрою;
- state з типом CharField – стан роботи над ремонтом цього пристрою.

Модель категорії побутової техніки `Appliancecategory` створена класом, у складі якого визначено поле `category` з типом CharField – назва категорії побутової техніки.

Модель екземпляру пристрою побутової техніки користувача `Applianceitem` створена класом, у складі якого визначено поля:

- `appliancemodel` з типом ForeignKey – модель пристрою побутової техніки, якщо вона відома;
- `appuser` з типом ForeignKey – клієнт, який володіє цим пристроєм;

- productionyear з типом PositiveIntegerField – рік, коли було виготовлено цей пристрій;
- textabout з типом CharField – текстовий опис пристрою побутової техніки;
- serialnumber з типом CharField – серійний номер пристрою побутової техніки;
- image з типом ImageField – зображення, де показано цей пристрій побутової техніки.

Модель виробника пристроїв побутової техніки Appliancemanufacturer створена класом, у складі якого визначено поле manufacturer з типом CharField – назва виробника побутової техніки.

Модель, яка реалізує модель пристрою побутової техніки, Appliancemodel створена класом, у складі якого визначено поля:

- manufacturer з типом ForeignKey – виробник, який виготовляє цю модель пристрою побутової техніки;
- model з типом CharField – назва моделі пристрою побутової техніки;
- appliancecategory з типом ForeignKey – категорія побутової техніки.

Модель типу деталі побутової техніки Typicaldetail створена класом, у складі якого визначено поле detailname з типом CharField – назва типу деталі побутової техніки.

Модель деталі до моделі пристрою побутової техніки Detail створена класом, у складі якого визначено поля:

- typicaldetail з типом ForeignKey – тип деталі, який має ця деталь до моделі пристрою побутової техніки;
- appliancemodel з типом ForeignKey – модель пристрою побутової техніки, до якої належить ця деталь;
- caption з типом CharField – повна назва деталі до пристрою побутової техніки;
- abouttext з типом CharField – текстовий опис деталі до пристрою побутової техніки;

– image з типом ImageField – зображення, де показано цю модель пристрою побутової техніки.

Модель наявності деталей у сервісі побутової техніки Detailshop створена класом, у складі якого визначено поля:

– detail з типом ForeignKey – деталь до пристрою побутової техніки, наявність якої визначається;

– repairshop з типом ForeignKey – відділення сервісу з ремонту побутової техніки, де ця деталь наявна в такій кількості;

– number з типом SmallIntegerField – кількість деталей побутової техніки, що наявна у цьому відділенні сервісу з ремонту побутової техніки;

– price з типом DecimalField – вартість однієї такої деталі до пристрою побутової техніки на поточний момент.

Модель заявки на ремонт побутової техніки у відділенні сервісу Repairorder створена класом, у складі якого визначено поля:

– repairshop з типом ForeignKey – відділення сервісу з ремонту побутової техніки, де цей ремонт має бути здійснено;

– appuser з типом ForeignKey – користувач, який подав цю заявку на ремонт побутової техніки у відділенні сервісу;

– createdate з типом DateTimeField – час створення заявки на ремонт побутової техніки у відділенні сервісу;

– orderdate з типом DateTimeField – час подання заявки на ремонт побутової техніки у відділенні сервісу;

– receivedate з типом DateTimeField – час отримання пристроїв з заявки на ремонт побутової техніки у відділенні сервісу;

– waitdate з типом DateTimeField – час очікуваного завершення ремонту побутової техніки у відділенні сервісу за заявкою;

– waitresult з типом DateTimeField – очікуваний результат ремонту (тобто фактично, чи очікує сервіс, що таке замовлення вдасться загалом завершити успішно);

- problem з типом CharField – основна проблема, з якою звертається клієнт, або деталі щодо заявки, які б хотів врахувати клієнт;
- answer з типом CharField – загальна текстова відповідь від сервісу клієнту щодо його заявки на ремонт побутової техніки у відділенні сервісу;
- state з типом CharField – стан роботи над заявкою заявки на ремонт побутової техніки у відділенні сервісу;
- ownercontact з типом BooleanField – дозвіл від клієнта (власника побутової техніки) щодо надання його контактних даних для зв'язку (номера телефону);
- manageremployee з типом ForeignKey – працівник сервісу, який виконує роль менеджера за цим замовленням;
- price з типом DecimalField – вартість виконання ремонту побутової техніки у відділенні сервісу;
- check з типом BooleanField – позначка про сплату всієї суми за виконання ремонту побутової техніки у відділенні сервісу.

Модель позиції заявки на ремонт побутової техніки у відділенні сервісу Repairorder_position створена класом, у складі якого визначено поля:

- repairorder з типом ForeignKey – заявка на ремонт побутової техніки у відділенні сервісу, до якої належить ця позиція;
- applianceitem з типом ForeignKey – пристрій побутової техніки у власництві цього клієнта, який має бути відданий у ремонт за заявкою;
- problem з типом CharField – проблема, з якою в роботі або стані цього пристрою побутової техніки зіштовхнувся власник речі;
- answer з типом CharField – відповідь від сервісу (тобто працівника, який виконував ремонт цього пристрою) щодо цього пристрою;
- ownerjobdone з типом CharField – процедури, які вже намагався виконати власник цього пристрою;
- jobdone з типом CharField – процедури з ремонту, які вже виконав працівник сервісу з ремонту побутової техніки щодо цієї речі (якщо працівник бажає це зафіксувати в програмі);

- `jobtodo` з типом `CharField` – процедури з ремонту, які потрібно ще виконати працівнику сервісу з ремонту побутової техніки щодо цієї речі (якщо працівник бажає це зафіксувати в програмі);

- `state` з типом `CharField` – стан роботи над ремонтом цієї речі;

- `plannedstart` з типом `DateTimeField` – запланований (або фактичний, коли ремонт розпочато) час початку роботи;

- `plannedendv` з типом `DateTimeField` – запланований (або фактичний, коли ремонт закінчено) час завершення роботи.

Модель типу процедур з ремонту побутової техніки `Typicalrepairjob` створена класом, у складі якого визначено поля:

- `appliancecategory` з типом `ForeignKey` – категорія побутової техніки, для якої така робота виконується;

- `jobname` з типом `CharField` – назва процедури з ремонту побутової техніки.

Модель типу процедур з ремонту побутової техніки, які виконуються в конкретному відділенні сервісу з ремонту побутової техніки, `Repairshop_typicalrepairjob` створена класом, у складі якого визначено поля:

- `repairshop` з типом `ForeignKey` – відділення сервісу з ремонту побутової техніки, де цей тип процедури з ремонту побутової техніки виконується;

- `typicalrepairjob` з типом `ForeignKey` – тип процедури з ремонту побутової техніки, що виконується в цьому відділенні сервісу;

- `price` з типом `ForeignKey` – стандартна вартість виконання такої процедури з ремонту побутової техніки;

- `duration` з типом `TimeField` – стандартна тривалість виконання такої процедури з ремонту побутової техніки.

Модель типу процедури з ремонту побутової техніки, що виконується конкретним працівником сервісу з ремонту побутової техніки, `Employee_typicalrepairjob` створена класом, у складі якого визначено поля:

- `typicalrepairjob` з типом `ForeignKey` – тип процедури з ремонту побутової техніки, що виконується цим працівником сервісу;
- `employee` з типом `ForeignKey` – працівник сервісу з ремонту побутової техніки, який здатен виконувати цей тип процедури з ремонту побутової техніки;
- `specification` з типом `CharField` – особливості виконання цим працівником сервісу з ремонту побутової техніки цього типу процедури з ремонту побутової техніки.

Модель проміжків часу роботи працівника сервісу з ремонту побутової техніки `Employee_schedule` створена класом, у складі якого визначено поля:

- `employee` з типом `ForeignKey` – працівник сервісу з ремонту побутової техніки, для якого визначений цей проміжок часу роботи на день;
- `typical` з типом `BooleanField` – розглядається як типовий варіант (наприклад, час роботи в понеділок має повторюватися кожен понеділок) або ні;
- `starttime` з типом `DateTimeField` – час початку роботи в цей день працівником сервісу з ремонту побутової техніки (дата і час);
- `finishtime` з типом `DateTimeField` – час закінчення роботи в цей день працівником сервісу з ремонту побутової техніки (дата і час).

Модель позиції заявки на ремонт побутової техніки в сервісі, визначеної для виконання конкретним працівником сервісу з ремонту побутової техніки, `Repairorder_position_employee` створена класом, у складі якого визначено поля:

- `repairorder_position` з типом `ForeignKey` – позиція заявки на ремонт побутової техніки в сервісі, роботи з якою має виконувати цей працівник;
- `employee` з типом `ForeignKey` – працівник сервісу з ремонту побутової техніки, який має виконати ці роботи;
- `function` з типом `CharField` – функція або конкретно опис того, що має виконати цей працівник сервісу з ремонту побутової техніки;

- `plannedstart` з типом `DateTimeField` – запланований початок робіт працівником сервісу з ремонту побутової техніки над цією позицією заявки на ремонт побутової техніки в сервісі;
- `plannedend` з типом `DateTimeField` – заплановане закінчення робіт працівником сервісу з ремонту побутової техніки над цією позицією заявки на ремонт побутової техніки в сервісі;
- `state` з типом `CharField` – стан виконання цих робіт;
- `result` з типом `CharField` – результат проведення цих ремонтних робіт;
- `answer` з типом `CharField` – відповідь працівника сервісу з ремонту побутової техніки щодо виконання цих робіт.

Модель ремонтної роботи, яка виконується в сервісі конкретним працівником над конкретною позицією заявки на ремонт побутової техніки, `Repairjob` створена класом, у складі якого визначено поля:

- `repairshop_typicalrepairjob` з типом `ForeignKey` – тип процедури з ремонту побутової техніки, що виконується в конкретному відділенні сервісу з ремонту побутової техніки і за яким цим працівником сервісу має бути виконано роботу;
- `repairorder_position_employee` з типом `ForeignKey` – працівник сервісу з ремонту побутової техніки, що виконує роботи над позицією заявки на ремонт побутової техніки в сервісі, який і має виконати цю конкретну роботу;
- `plannedstart` з типом `DateTimeField` – запланований початок виконання цієї роботи працівником сервісу з ремонту побутової техніки над цією позицією заявки на ремонт побутової техніки в сервісі;
- `plannedend` з типом `DateTimeField` – заплановане закінчення роботи працівником сервісу з ремонту побутової техніки над цією позицією заявки на ремонт побутової техніки в сервісі;
- `textabout` з типом `CharField` – текстовий опис відповіді працівника сервісу з ремонту побутової техніки щодо виконання цієї роботи;

- state з типом CharField – стан виконання цієї роботи;
- result з типом CharField – результат проведення цієї ремонтної роботи;
- price з типом DecimalField – вартість виконання цієї конкретної роботи в цьому конкретно випадку.

Модель деталі, яка ремонтується або застосовується для виконання ремонту, Repairjob_detail створена класом, у складі якого визначено поля:

- repairjob з типом ForeignKey – конкретна ремонтна робота, яка виконується в сервісі конкретним працівником над конкретною позицією заявки на ремонт побутової техніки в сервісі і для якої використано цю деталь;
- detail з типом ForeignKey – деталь, яка були використана або відремонтована під час виконання ремонту;
- number з типом PositiveIntegerField – кількість таких деталей;
- operation з типом BooleanField – тип операції (ремонт або заміна);
- price з типом DecimalField – вартість, якою було оцінено саму цю деталь (роботи фіксуються окремо).

Модель оцінки користувачами волонтерів Volunteerpoint створена класом, у складі якого визначено поля:

- volunteer з типом ForeignKey – волонтер, про якого створено оцінку;
- appuser з типом ForeignKey – користувач, який виконав оцінку;
- pointdate з типом DateTimeField – час збереження оцінки користувачем волонтера;
- point з типом SmallIntegerField – значення оцінки.

Модель оцінки користувачами сервісів з ремонту побутової техніки Shoppoint створена класом, у складі якого визначено поля:

- repairshop з типом ForeignKey – сервіс з ремонту побутової техніки, про який створено оцінку;
- appuser з типом ForeignKey – користувач, який виконав оцінку;

- pointdate з типом DateTimeField – час збереження оцінки користувачем сервісу ремонту побутової техніки;
- point з типом SmallIntegerField – значення оцінки.

Перегляд інформації про волонтерську організацію забезпечено через функцію `volunteerrepair_info_view`, яка має параметр `volunteerrepairid`, що задає номер волонтерської організації, і яка реалізує представлення, яке виділяє інформацію про саму організацію (ключ `repairer` результатів), наступні 5 подій, які проводять цією волонтерською організацією (ключ `events`), при цьому всі результати передаються в шаблон сторінки `volunteerrepair.html`. Подібно влаштована робота кожного представлення, але з поверненням результатів через окремий шаблон і заповнення ключів у відповідності з даними, які необхідні тим чи іншим сценарієм. Під кожен сценарій було створено функцію представлення, з них деякі перелічені нижче.

Перегляд волонтерських подій за волонтерською організацією забезпечено через функцію `volunteerrepair_events_view`, яка має параметри `volunteerrepairid`, що визначає номер волонтерської організації, `new_events`, що визначає необхідність відображення тільки нових подій, які ще не відбулися (або відсутність такої вимоги).

Перегляд інформації про волонтерську подію забезпечено через функцію `volunteerevent_info_view`, яка має параметр `volunteereventid` з номером волонтерської події, повна інформація про яку знаходиться за ключем `event` результатів у сторінці `volunteerevent.html`.

Перегляд інформації про майстра забезпечено через функцію `master_info_view`, яка має параметр `masterid`, що визначає номер запису майстра, який може бути або волонтером, або працівником сервісу з ремонту побутової техніки, при чому параметром `volunteer` можна вказати, чи роль волонтера є основною (`True`), чи працівника сервісу з ремонту побутової техніки. Однак, якщо виявляється, що при перевазі волонтеру (або навпаки) такого запису не знаходиться, то відбувається пошук працівника сервісу з

ремонту побутової техніки, і тоді його інформація повертається в результаті. Окрім інформації облікового запису також повертаються дані про типові ремонтні роботи, які здатен виконувати цей майстер, оцінювання якості його роботи. Всюди, де це можливо і необхідно, забезпечений саме такий підхід, коли користувачу видаються результати в залежності від параметрів, тобто фактично узагальнюючого поняття майстер у програмі немає (є окремо волонтер, окремо працівник сервісу з ремонту побутової техніки), але користувач може відкрити таку сторінку про певну особу, а йому в залежності від наявного запису надається необхідна інформація. Так само, наприклад, влаштований перегляд заявок на ремонт побутової техніки, який доступний всім типам користувачів, однак по-різному працює в залежності від того, який саме користувач звертається. Це функція `repair_orders_view`, яка має параметр `asuser`, що вказує, що з нею працює звичайний користувач, тобто клієнт (власник пристроїв), якщо параметр має значення `True`, у протилежному випадку витягаються дані стосовно волонтера, який може бути або індивідуальним волонтером, тобто для нього будуть знайдені записи про віртуальні консультації, або представником волонтерської організації, тоді йому будуть витягнуті всі заявки, які стосуються подій цієї організації, де користувач є менеджером. Усі заявки об'єднуються в єдиний перелік та сортуються за датою виконання. Функція також має параметр `neworders`, що визначає те, що повертаються тільки нові (або невиконані заявки, а для замовлень у сервісі ремонту враховується значення 0 як нові заявки, 1 – архівні, 2 – в роботі, 3 – очікувані для прийняття). Параметр `ser` може дорівнювати 0, тоді повертаються і заявки на ремонт у сервісі з ремонту побутової техніки, і їх позиції в повній мірі. Якщо значення параметра 1, то повертається тільки перше, а якщо 2, то тільки друге в основі, тобто фактично тільки роботи за конкретними позиціями заявки.

Перегляд сервісів ремонту побутової техніки забезпечено через функцію `repairserices_view`, яка повертає всі сервіси, тому не має параметрів.

Перегляд інформації про сервіс з ремонту побутової техніки забезпечено через функцію `repairservice_info_view`, параметром якої `repairserviceid` задається номер сервісу з ремонту побутової техніки. За сервісом також повертається перелік всіх відділень за ключем `shops`.

Пошук волонтерських організацій забезпечено через функцію `volunteerrepair_search_view`, яка працює з підтримкою GET-запитів, тому витягує значення всіх параметрів пошуку з такого запиту, включаючи: назва волонтерської організації (`volcaption`), місто, де знаходиться волонтерська організація (`voltown`), адреса розташування волонтерської організації (`voladdress`), заплановані події для ремонту, не пізніше заданої дати (день `volday`, місяць `volmonth`, рік `volyear`), необхідність повної рівності введеним текстовим значенням чи достатність пошуку входження фрази (`eq`).

Пошук волонтерських подій для ремонту побутової техніки забезпечено через функцію `volunteerevents_search_view` і влаштований він аналогічно попередній функції, так само як і пошук волонтерів на основі функції `volunteer_search_view`, пошук відділень сервісу з ремонту побутової техніки на основі функції `repairshop_search_view`.

Перегляд власних пристроїв побутової техніки забезпечено через функцію `applianceitems_view`, яка відбирає всі пристрої побутової техніки, якими володіє користувач за даними авторизації. Дане представлення відповідно перевіряє авторизацію користувача, як і всі інші представлення. Якщо користувач неавторизований, то повертається сторінка з передачею об'єкта, що відповідає порожнім даним.

Перегляд ремонтів пристрою забезпечено через функцію `applianceitem_repairs_view`, яка має параметр `applianceitemid`, що визначає номер пристрою побутової техніки, та параметр `ext`, що визначає варіант розширеного представлення, коли повертаються фактично не тільки виконані роботи з ремонту, але і фактично всі заявки (тобто за волонтерської події є інформація про пристрій, який був внесений у заявку, але немає інформації про виконані роботи, тому розширений варіант дозволяє повернути і ту

інформацію, і ту). У представленні також перевіряється, що користувач або є власником цієї речі, або він є працівником сервісу з ремонту техніки, за яким встановлено, що було подано заявку на виконання ремонту, в яку входив цей пристрій, або він є відповідно волонтером, а цей пристрій входить до заявки на віртуальну консультацію, або він є менеджером волонтерської організації, на подію якої було подано заявку на ремонт цього пристрою. Якщо таких прав немає, або якщо користувач не авторизувався, то ніяких даних не повертається.

Перегляд доступних деталей для моделі пристрою побутової техніки забезпечено через функцію `appliancemodel_details_view`, де параметр `appliancemodelid` визначає номер моделі.

Перегляд виконаних ремонтних робіт забезпечено через функцію `employee_repairs_view`, де параметр `history` визначає, чи видавати тільки минулі (виконані) ремонтні роботи, чи всі, параметр `date_start` визначає початок періоду, в який має увійти початок запланованого ремонту, параметр `date_end` визначає закінчення періоду, в який має увійти початок запланованого ремонту, а параметр `state` визначає стан виконання потрібного ремонту, якщо він уточнений.

Перегляд власних запланованих волонтерських подій забезпечено через функцію `volunteer_events_view`, де параметр `history` визначає тільки події, які відбулись (значення 0), тільки події, які ще не відбулись (значення 1), або всі (значення 2).

Керуючі представлення, тобто ті, які виконують зміни даних, реалізовано також через функції, але зі створенням у відповідності з необхідною в основі моделлю класу форми, за яким і відбувається робота з даними щодо зміни або створення цих даних.

Реалізацію роботи методу виконано через пакет `gensim`, у підпакеті `models` якого знаходиться клас `Word2Vec`, що реалізує процес створення і навчання моделі `Word2Vec`, та через пакет `scikit-learn`, у підпакеті `ensemble` якого знаходиться клас `RandomForestClassifier`.

4.3 Висновки за розділом 4

Реалізація всіх запланованих функцій була забезпечена створенням Django-застосунку і подальшим визначенням класів, функцій, макетів вебсторінок, які представляють моделі, представлення, менеджер, шаблони.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Програма призначена для використання трьома сторонами:

- волонтерськими організаціями, які підтримують виконання ремонту побутової техніки, через надання можливостей інформування про проведення подій, оформлення заявок на ці події та окремими волонтерами для надання віддалених консультацій щодо ремонту;
- комерційними сервісами ремонту побутової техніки, які мають керувати заявками, визначати порядок роботи над ремонтом, формувати вартість ремонту за виконуваними роботами, визначати використані деталі для роботи;
- власниками пристроїв побутової техніки для створення заявок на виконання ремонту, для відслідковування виявлених раніше проблем з власними пристроїв, виконаних обслуговувань.

5.2 Виконання програми

Виконання програми повністю забезпечується стандартним механізмом звернення з клієнта через браузер до сервера, у свою чергу на сервері має бути запущений Django-застосунок. Комунікація з програмою кожного користувача відбувається через одну й ту саму базову адресу, проте авторизація визначає, який саме режим роботи буде використовуватися. На сторінці авторизації можна перейти до авторизації іншого типу користувача, якщо це необхідно, тобто конкретизувати роль. Це може бути необхідно, якщо користувач є і працівником комерційного сервісу, і волонтером (наприклад, у вільний від роботи час). Але стандартна сторінка авторизації дозволяє авторизуватися будь-якому користувачу, а подальше відображення залежить від його ролі.

Клієнт має всі необхідні для роботи розділи, включаючи «Волонтерство», звідки можна виконати пошук і волонтерських організацій, і волонтерських подій (рис. 5.1), і волонтерів (цей вибір доступний через підрозділи меню), «Сервіси», звідки можна виконати пошук відділень сервісу з ремонту побутової техніки, переглянути всі сервіси, що є в програмі, «Заявки», звідки доступна панель керування заявками, до яких має відношення користувач, «Пристрої», звідки доступні всі свої пристрої клієнта, «Інформація» та посилання для виходу з облікового запису або навпаки авторизації (звідки реєстрація доступна).

HomeAppRepairer

ВОЛОНТЕРСТВО ▾
СЕРВІСИ ▾
ЗАЯВКИ
ПРИСТРОЇ
ІНФОРМАЦІЯ ▾

Вийти →

__ Пошук волонтерських подій __

Назва події

Львів ▾

Адреса проведення

Готовий чекати до

20.12.2023

📅

☐ чітка відповідність

Мінімальна кількість волонтерів

1 ▾

Максимальна кількість волонтерів

12 ▾

Підібрати



Ремонт пристроїв з дерев'яними елементами

Львів 11.12.2023

[Детальніше →](#)



Ремонт елементарної електроніки

Львів 15.12.2023

[Детальніше →](#)



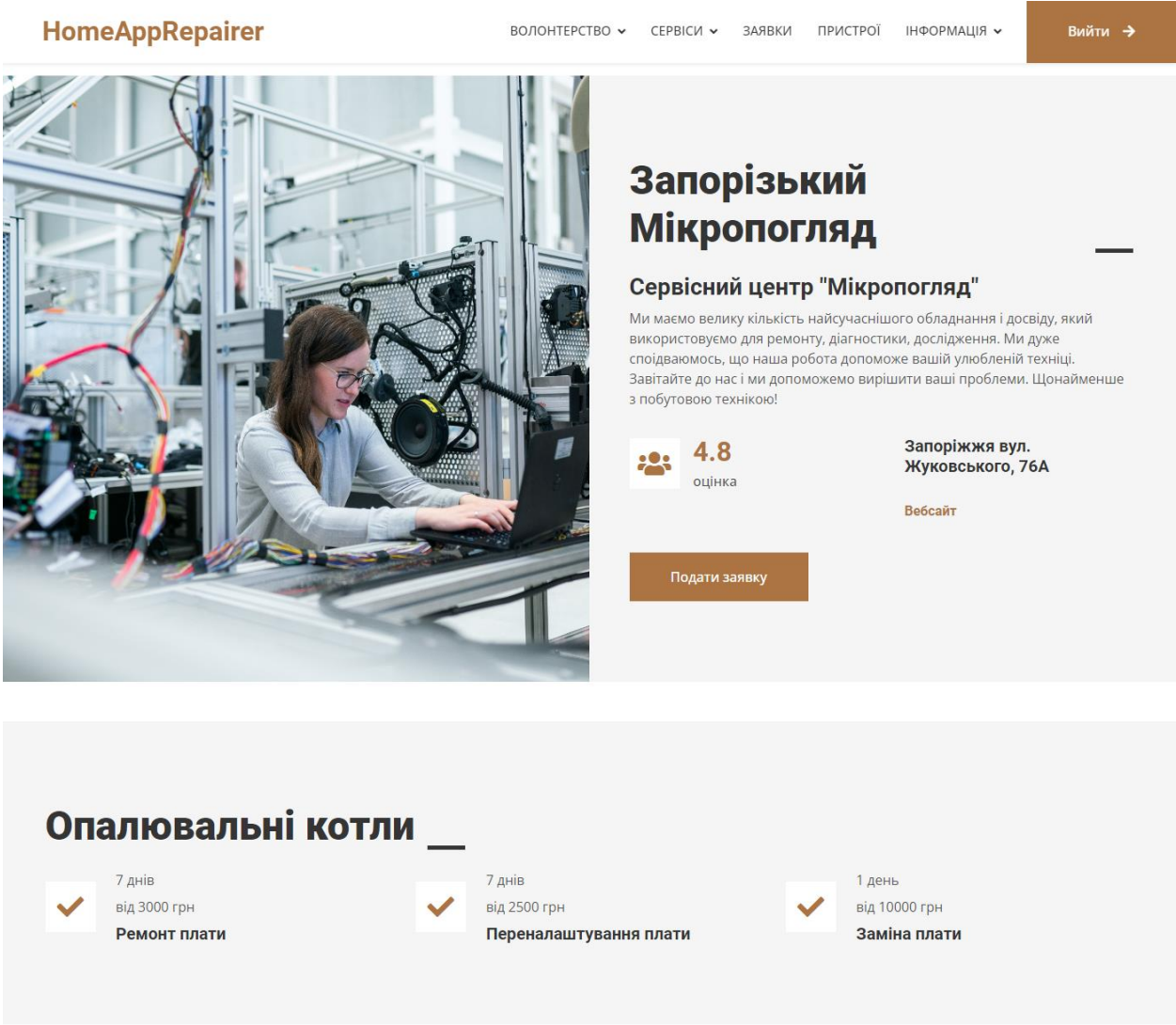
Ремонт і виготовлення найпростіших деталей

Львів 17.12.2023

[Детальніше →](#)

Рисунок 5.1 – Пошук волонтерських подій

Клієнт може шукати необхідну йому інформацію з цих розділів меню, відкривати відповідні сторінки, переглядати інформацію. Наприклад, на сторінці відділення сервісу наводиться інформація про це відділення (наприклад, за назвою сервісу можна перейти на сторінку вже не відділення, а сервісу загалом). На сторінці доступна стандартна вартість робіт (рис. 5.2). Також наявна кнопка для створення заявки на ремонт.



HomeAppRepairer ВОЛОНТЕРСТВО ▾ СЕРВІСИ ▾ ЗАЯВКИ ПРИСТРОЇ ІНФОРМАЦІЯ ▾ [Вийти](#) →

Запорізький Мікропогляд

Сервісний центр "Мікропогляд"

Ми маємо велику кількість найсучаснішого обладнання і досвіду, який використовуємо для ремонту, діагностики, дослідження. Ми дуже сподіваємось, що наша робота допоможе вашій улюбленій техніці. Завітайте до нас і ми допоможемо вирішити ваші проблеми. Щонайменше з побутовою технікою!

4.8
оцінка

Запоріжжя вул. Жуковського, 76А

[Вебсайт](#)

[Подати заявку](#)

Опалювальні котли

✓ 7 днів від 3000 грн Ремонт плати	✓ 7 днів від 2500 грн Переналаштування плати	✓ 1 день від 10000 грн Заміна плати
---	---	--

Рисунок 5.2 – Сторінка відділення сервісу з ремонту побутової техніки

Створення заявки на ремонт вже відбувається за конкретним відділенням, але в подальшому потрібно обрати з наявних у клієнта пристроїв той, який потрібно здати в ремонт, задати опис проблеми, а тоді можна натиснути на посилання «Надати прогноз», після чого зліва від

кнопки буде показано, чи очікується, що цей пристрій буде відремонтовано, чи така ймовірність низька (рис. 5.3).

The screenshot shows the HomeAppRepairer website interface. At the top, there is a navigation bar with the logo 'HomeAppRepairer' and several menu items: 'ВОЛОНТЕРСТВО', 'СЕРВІСИ', 'ЗАЯВКИ', 'ПРИСТРОЇ', and 'ІНФОРМАЦІЯ'. A 'Вийти' button is located on the right. The main content area is titled 'Заявка на ремонт побутової техніки'. It features a 'CheckPoint' section on the left with a photo of technicians and the text 'Відділення для ремонту'. The central form includes a dropdown menu for selecting the appliance type (currently 'Кухонний комбайн'), a text area for describing the problem in English (example: 'Start button does not work.'), a 'Надати прогноз' button, and a checkbox for agreeing to share personal data. A 'Подати заявку' button is at the bottom. On the right, there is a product image of an 'Oster FPSTFP4010' blender with the text 'Обраний пристрій'.

Рисунок 5.3 – Заявка на ремонт побутової техніки

Меню «Інформація» містить підрозділи для визначення власних вмінь майстра, керування обліковим записом. Для працівників сервісів доступна також можливість роботи з даними відділень, включаючи керування працівниками у відділенні, визначення інформації про відділення, наявних деталей (завдання цих значень), послуг за відділенням. При цьому працівник повинен мати відповідні права на роботу з даними відповідного відділення.

З меню «Пристрої» для працівника сервісу відкривається ще додатковий підпункт щодо наявних деталей для ремонту за відділенням. Працівник сервісу може працювати з заявками (рис. 5.4), які надійшли на ремонт, керуючи ними, але спочатку визначаючи, яка група заявок потрібна (тільки нові, тобто необроблені, ті, що в ремонті вже знаходяться, очікувані до ремонту, та архівні).

Для волонтерів, працівників волонтерських організацій доступне також меню «Події» для керування подіями.

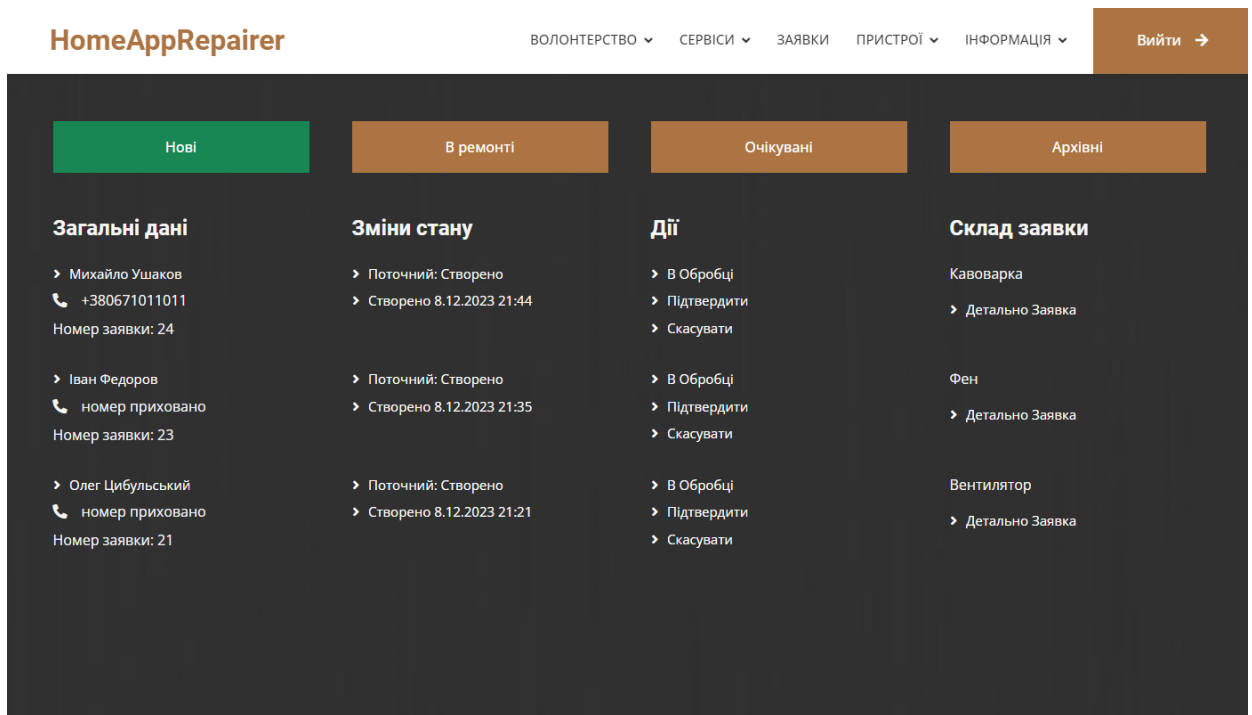


Рисунок 5.4 – Перегляд і керування заявками на ремонт у сервісі

5.3 Експериментальне дослідження

Для того щоб перевірити роботу методу класифікації заявок на ремонт побутової техніки та визначити, чи є доцільним виконувати практичне впровадження методу в програму, було окремо проведено експериментальне дослідження, що ґрунтувалось на вибірці даних [10]. Дана вибірка складена з окремих вибірок, що стосуються сервісів у різних країнах. Враховуючи особливості запропонованого методу, а зокрема необхідність використання опису проблеми, було використано тільки ті вибірки, які містять дані з описами зі США, Великобританії. Усі екземпляри були перевірені стосовно:

- наявності описів, а також того, що ці описи виконані саме англійською мовою;
- наявності даних про рік випуску пристрою;
- наявності результуючого класу, тобто результату ремонту (якщо вписано невідомий стан, то такий екземпляр було вилучено).

Підсумкова вибірка містила тільки екземпляри, які пройшли всі ці перевірки. Саме дослідження включало спочатку перевірку роботи різних моделей, до яких включено:

- НБК;
- дерево рішення для класифікації;
- метод Random Forest для створення ансамблю дерев рішень.

Для цих моделей отримано результати і проаналізовано варіанти стосовно використання або невикористання опису проблеми в якості вхідних ознак (за невикористання на вхід подавались тільки категорія, бренл, вік пристрою, дата звернення, у другому випадку – додатково елементи опису проблеми) та кількості елементів у векторів опису проблеми у випадку, якщо такий опис використовувався на вхід (табл. 5.1). У даному випадку метод Random Forest весь час використовувався у варіанті з 1000 дерев та максимальною глибиною, рівною 10.

З першої частини дослідження одразу помітно, що за всіх варіантів найкращий результат дозволила отримати модель на основі методу Random Forest. Це стосувалося і значення точності (accuracy), і влучності (precision) [11]-[12]. Найкращий результат для Random Forest було отримано у випадку вхідного вектору на 50 елементів. Хоча окремо за показником точності варіанти на 20 та 100 елементів виявились кращими, але гіршими за влучністю.

При цьому НБК показував кращі результати за меншої кількості елементів, тобто модель починає працювати гірше за більшої кількості вхідних ознак на вході.

Також перша частина дослідження вказала на те, що використання опису проблеми при побудові моделі є доцільним, адже:

- для НБК використання опису підвищило точність на величину від 4,65 до 29 %, а величину влучності від 5,89 до 18,33 % (окрім зниження для 100 елементів);

- для дерева рішень використання опису підвищило точність на величину від 12,65 до 23,47 %, а величину влучності від 9,63 до 15,31 %;
- для Random Forest використання опису підвищило точність на величину від 25,46 до 27,88 %, а величину влучності від 15,99 до 21,92 %.

Таблиця 5.1 – Результати проведеного експериментального дослідження

Модель	Використання опису проблеми	Точність	Влучність
НБК	Ні	0,494062	0,518207
Дерево рішень	Ні	0,519398	0,536116
Random Forest	Ні	0,556611	0,58042
НБК	Так, 100 елементів	0,517022	0,488596
Дерево рішень	Так, 100 елементів	0,64133	0,607553
Random Forest	Так, 100 елементів	0,711797	0,678571
НБК	Так, 20 елементів	0,632621	0,570713
Дерево рішень	Так, 20 елементів	0,626287	0,588832
Random Forest	Так, 20 елементів	0,711006	0,673203
НБК	Так, 10 елементів	0,637371	0,613195
Дерево рішень	Так, 10 елементів	0,585115	0,587762
Random Forest	Так, 10 елементів	0,698337	0,689189
НБК	Так, 50 елементів	0,582739	0,548733
Дерево рішень	Так, 50 елементів	0,620744	0,618211
Random Forest	Так, 50 елементів	0,701504	0,707641

Використовуючи сформований опис проблеми на 50 елементів, метод Random Forest був окремо досліджений стосовно його параметрів. Було

створено ряд моделей, що включали 100, 500 або 1000 дерев, обмежуючи або не обмежуючи глибину дерева (табл. 5.2).

Таблиця 5.2 – Результати експериментального дослідження параметрів методу Random Forest

Модель	Дерева	Глибина	Точність	Влучність
Random Forest	1000	10	0,701504	0,707641
Random Forest	500	без обмежень	0,725257	0,702454
Random Forest	100	без обмежень	0,707047	0,713115
Random Forest	1000	без обмежень	0,725257	0,744373

Найкращий результат отримано на основі моделі з необмеженою глибиною та 1000 дерев. Точність у такому випадку дорівнює 0,725, а влучність – 0,744. Це дозволяє вважати даний метод прийнятним для практичного впровадження.

Результуюча модель, яка була створена на основі методу класифікації заявок на ремонт побутової техніки, має точність на 24,46 % кращу і влучність на 35,65 % кращу за НБК у тому ж самому випадку та точність на 13,79 % кращу і влучність на 21,39 % кращу у найкращому варіанті НБК. Окрім того результуюча модель має точність на 16,84 % кращу і влучність на 20,41 % кращу за дерево рішень у тому ж самому випадку та точність на 13,09 % кращу і влучність на 22,52 % кращу у найкращому варіанті дерева.

5.4 Висновки за розділом 5

Реалізоване програмне забезпечення дозволяє виконувати всі необхідні сервісні функції, які було визначено технічним завданням, та

використовує у своєму складі навчену модель для класифікації заявок на ремонт побутової техніки. Модель було створено на основі використання запропонованого методу, ефективність якого підтверджено результатами експериментального дослідження, за яким даний метод дозволив отримати точність класифікації на від 13,09 до 24,26 % кращу за аналоги та влучність на від 20,41 до 35,65 % кращу за аналоги.

ВИСНОВКИ

У роботі розроблено програмне забезпечення підтримки роботи сервісу ремонту побутової техніки для підвищення зручності роботи всіх задіяних сторін.

Створена програма може використовуватися волонтерськими організаціями, які підтримують виконання ремонту побутової техніки, через надання можливостей інформування про проведення подій, оформлення заявок на ці події та окремими волонтерами для надання віддалених консультацій щодо ремонту, комерційними сервісами ремонту побутової техніки, які мають керувати заявками, визначати порядок роботи над ремонтом, формувати вартість ремонту за виконуваними роботами, визначати використані деталі для роботи, власниками пристроїв побутової техніки для створення заявок на виконання ремонту, для відслідковування виявлених раніше проблем з власними пристроїв, виконаних обслуговувань.

Наукова новизна роботи полягає в створеному методі класифікації заявок на ремонт побутової техніки, який виконує класифікацію заявок на успішно виконані та такі, які не завершилися успішним ремонтом техніки, будує для цього модель класифікації на основі методу Random Forest та перетворює текстовий опис проблеми на основі методу Word2vec для використання в методі.

Реалізоване програмне забезпечення дозволяє виконувати всі необхідні сервісні функції, які було визначено технічним завданням, та використовує у своєму складі навчену модель для класифікації заявок на ремонт побутової техніки. Модель було створено на основі використання запропонованого методу, ефективність якого підтверджено результатами експериментального дослідження, за яким даний метод дозволив отримати точність класифікації на від 13,09 до 24,26 % кращу за аналоги та влучність на від 20,41 до 35,65 % кращу за аналоги.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ремонт побутової техніки. Сервісний центр у Харкові "Repair Service" [Електрон. ресурс]. – Режим доступу : <https://repairservice.com.ua/>.
2. Repair Cafe Wales – Local Community Repair Cafes [Electronic resource]. – Access mode : <https://repaircafewales.org/>.
3. Fixit Clinic : Get Repair [Electronic resource]. – Access mode : <https://fixitclinic.blogspot.com/>.
4. Gaussian Naive Bayes – GeeksforGeeks [Electronic resource]. – Access mode : <https://www.geeksforgeeks.org/gaussian-naive-bayes/>.
5. Decision Tree – GeeksforGeeks [Electronic resource]. – Access mode : <https://www.geeksforgeeks.org/decision-tree/>.
6. Otten, N. V. Random Forest Classifier – How To Python Tutorial & Practical Guide [Electronic resource] / N. V. Otten. – Access mode : <https://spotintelligence.com/2023/08/31/random-forest-classifier/>.
7. Ramesh, S. Demystifying the Random Forest [Electronic resource] / S. Ramesh. – Access mode : <https://towardsdatascience.com/demystifying-the-random-forest-8a46f4fd416f>.
8. Welcome to django-paypal's documentation! [Electronic resource]. – Access mode : <https://django-paypal.readthedocs.io/en/latest/>.
9. Woody – Responsive Free Bootstrap 5 HTML5 Business Template [Electronic resource]. – Access mode : <https://themewagon.com/themes/responsive-free-bootstrap-5-html5-business-template-woody/>.
10. Data downloads – Open Repair Alliance [Electronic resource]. – Access mode : <https://openrepair.org/open-data/downloads/>.
11. Accuracy vs. precision vs. recall in machine learning : what's the difference? [Electronic resource]. – Access mode : [https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall#:~:text=Accuracy%20shows%20how%20often%20a,objects%20of%20the%](https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall#:~:text=Accuracy%20shows%20how%20often%20a,objects%20of%20the%20)

20target%20class.

12. Evaluation Metrics for Classification [Electronic resource]. – Access mode : <https://medium.com/@impythonprogrammer/evaluation-metrics-for-classification-fc770511052d>.

ДОДАТОК А
Технічне завдання

Вступ

Ремонт побутової техніки – це важливе практичне завдання, з яким періодично змушені стикатися люди. Поки діє гарантійний термін, обслуговування може відбуватися у спеціалізованих сервісних центрах, які визначені виробником. Але в подальшому вже власник побутової техніки може вирішувати самостійно, послугами якого сервісу скористатися. Сервіси можуть бути представлені деякими мережами, можуть бути представлені окремими пунктами. Але це стосується традиційних підходів, коли такі сервіси є комерційними. Проте існує і інший варіант, коли сервіси представлені волонтерськими організаціями. Тоді або волонтер сам виконує ремонт, але безкоштовно, або допомагає користувачу виконати цей ремонт. Для волонтерських організацій характерним є проведення подій, на яких як раз такі процеси і відбуваються. Фінансування такі організації можуть отримувати за рахунок урядових і неурядових грантів. Підґрунтям для такої діяльності, хоч вона і є безкоштовною, є найчастіше екологічні ініціативи. Тобто замість того, щоб викидати пристрій, який можна відремонтувати, і або перевантажувати сміттєзвалища, або збільшувати обсяг роботи для заводів з переробки сміття, де такі можливості є, а відповідно впливати на екологію, фінансуються роботи з ремонту для того, щоб продовжити життя існуючим пристроям. Об'єднання і компаній, які можуть надати комерційні послуги, і волонтерських організацій в одній програмі, має дозволити створити платформу, на якій люди за рахунок більшої кількості можливостей можуть вибирати потрібний їм варіант, розуміти можливості вибору, а таким чином частіше використовувати можливість відремонтувати техніку замість придбання нової, де це можливо. Тож саме таке програмне забезпечення і має бути створено в роботі в підсумку. Для того щоб користувач одразу отримував додаткову інформацію про те, чи вирогідно, що ремонт вдасться, важливою є інтеграція у програму інструменту класифікації поданих заявок на ремонт побутової техніки.

A.1 Підстави для розробки

Завдання на роботу визначено самим виданим завданням від 4 вересня 2023 року, що має входити до складу пояснювальної записки, та наказом № 437 від 14 листопада 2023 р. за Національним університетом «Запорізька політехніка», де затверджено тему роботи: «Дослідження та програмна реалізація методів класифікації для сервісу ремонту побутової техніки».

A.2 Призначення розробки

Концептуально програма має бути призначена для використання трьома сторонами:

- волонтерськими організаціями, які підтримують виконання ремонту побутової техніки, через надання можливостей інформування про проведення подій, оформлення заявок на ці події та окремими волонтерами для надання віддалених консультацій щодо ремонту;
- комерційними сервісами ремонту побутової техніки, які мають керувати заявками, визначати порядок роботи над ремонтом, формувати вартість ремонту за виконуваними роботами, визначати використані деталі для роботи;
- власниками пристроїв побутової техніки для створення заявок на виконання ремонту, для відслідковування виявлених раніше проблем з власними пристроїв, виконаних обслуговувань.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

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

- керування заявкою на ремонт побутової техніки під час

волонтерської події;

- пошук волонтерських організацій;
- перегляд інформації про волонтерську організацію;
- керування даними волонтерської організації;
- перегляд волонтерських подій за волонтерською організацією;
- пошук волонтерських подій для ремонту побутової техніки;
- створення волонтерської події;
- редагування волонтерської події;
- перегляд інформації про волонтерську подію;
- подання заявки на участь у волонтерській події;
- підтвердження заявки на участь у волонтерській події;
- відміна заявки на участь у волонтерській події;
- перегляд власних запланованих волонтерських подій;
- пошук волонтерів;
- перегляд інформації про майстра;
- подання заявки на віртуальне консультування щодо ремонту

побутової техніки;

- відміна заявки на віртуальне консультування;
- підтвердження заявки на віртуальне консультування;
- оцінювання послуги з ремонту;
- керування даними облікового запису;
- авторизація;
- перегляд сервісів ремонту побутової техніки;
- перегляд інформації про сервіс ремонту побутової техніки;
- керування заявкою на ремонт побутової техніки у відділенні сервісу;
- пошук відділень сервісу ремонту побутової техніки;
- подання заявки на ремонт у відділенні сервісу;
- керування даними відділення сервісу ремонту побутової техніки;
- визначення прав користувача як працівника відділення сервісу

ремонту побутової техніки;

- визначення послуг з ремонту, що надаються відділенням сервісу ремонту побутової техніки;
- визначення наявних деталей для ремонту у відділенні;
- перегляд заявок на ремонт побутової техніки;
- визначення розкладу роботи працівника;
- визначення власних вмінь;
- перегляд власних пристроїв побутової техніки;
- керування даними власного пристрою побутової техніки;
- перегляд ремонтів пристрою;
- перегляд доступних деталей для моделі пристрою побутової техніки;
- визначення виконаної роботи над пристроєм побутової техніки;
- перегляд виконаних ремонтних робіт.

A.3.2 Умови експлуатації та вимоги

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

A.4 Порядок контролю та приймання

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

ДОДАТОК Б
Текст програми

```

from .models import Volunteerrepair, Repairevent, Volunteer,
Employee, Employee_typicalrepairjob, Volunteer_typicalrepairjob,
Repairservice, Repairshop, Repairevent_volunteer, Applianceitem,
Repairorder, Virtualmeeting, Repairevent_application,
Repairorder_position, Volunteer_virtualmeeting_item,
Repairevent_application_item, Repairorder_position_employee,
Repairjob, Appliancemodel, Detailshop
    from django.shortcuts import render
    import datetime
    from django.db.models import Q

def volunteerrepair_info_view(request, volunteerrepairid):
    try:
        vol_repair = Volunteerrepair.objects.get(id =
volunteerrepairid)
        vol_repair =
vol_repair.select_related("managervolunteer")
    except Volunteerrepair.DoesNotExist:
        return render(request,
"homeapprepaier/volunteerrepair.html", {})
    import datetime
    later_than = datetime.datetime.now()
    repairevents = Repairevent.objects.filter(volunteerrepair =
vol_repair, eventstart__gte = later_than)
    repairevents = repairevents.order_by("eventstart")[:5]
    return render(request,
"homeapprepaier/volunteerrepair.html", {"repairer": vol_repair,
"events": repairevents })

def volunteerrepair_events_view(request, volunteerrepairid,
new_events = True):
    try:
        vol_repair = Volunteerrepair.objects.get(id =
volunteerrepairid)
    except Volunteerrepair.DoesNotExist:
        return render(request,
"homeapprepaier/volteerevents.html", {})
    later_than = datetime.datetime.now()
    if new_events:
        repairevents =
Repairevent.objects.filter(volunteerrepair = vol_repair,
eventstart__gte = later_than)
        repairevents = repairevents.order_by("eventstart")
    else:
        repairevents =
Repairevent.objects.filter(volunteerrepair = vol_repair,
eventstart__lt = later_than)
        repairevents = repairevents.order_by("eventstart")

```

```

        return render(request,
"homeapprepairer/volunteerevents.html", {"repairer": vol_repair,
"events": repairevents})

    def volunteerevent_info_view(request, volunteereventid):
        try:
            vol_event = Repairevent.objects.get(id =
volunteereventid)
            vol_event = vol_event.select_related("volunteerrepair")
            vol_event =
vol_event.select_related("volunteerrepair__managervolunteer")
        except Repairevent.DoesNotExist:
            return render(request,
"homeapprepairer/volunteerevent.html", {})
        return render(request,
"homeapprepairer/volunteerevent.html", {"event": vol_event})

    def master_info_view(request, masterid, volunteer = True):
        if volunteer:
            try:
                vol_user = Volunteer.objects.get(id = masterid)
                vol_user = vol_user.select_related("appuser")
            except Volunteer.DoesNotExist:
                try:
                    empl_user = Employee.objects.get(id = masterid)
                    empl_user = empl_user.select_related("appuser")
                    empl_user =
empl_user.select_related("repairshop")
                    empl_user =
empl_user.select_related("repairshop__repairservice")
                    empl_user = empl_user.select_related
("repairshop__manageremployee")
                except Employee.DoesNotExist:
                    return render(request,
"homeapprepairer/masterinfo.html", {})
                jobs =
Employee_typicalrepairjob.objects.filter(employee = empl_user)
                jobs = jobs.select_related("typicalrepairjob")
                jobs =
jobs.select_related("typicalrepairjob__appliancecategory")
                jobs =
jobs.order_by("typicalrepairjob__appliancecategory")
                return render(request,
"homeapprepairer/volunteerevent.html", {"profile": empl_user, "user":
"employee", "jobs": jobs, "points": [] })
                jobs = Volunteer_typicalrepairjob.filter(volunteer=
vol_user)
                jobs = jobs.select_related("typicalrepairjob")

```

```

        jobs =
jobs.select_related("typicalrepairjob__appliancecategory")
        jobs =
jobs.order_by("typicalrepairjob__appliancecategory")
        points = Volunteerpoint.objects.filter(volunteer =
vol_user)
        points = points.select_related("volunteer")
        points = points.select_related("appuser")
        points = points.select_related("volunteer__appuser")
        points = points.order_by("-pointdate")
        return render(request,
"homeapprepaier/volunteerevent.html", {"profile": vol_user, "user":
"volunteer", "jobs": jobs, "points": points })
    else:
        try:
            empl_user = Employee.objects.get(id = masterid)
            empl_user = empl_user.select_related("appuser")
            empl_user = empl_user.select_related("repairshop")
            empl_user =
empl_user.select_related("repairshop__repairservice")
            empl_user = empl_user.select_related
("repairshop__manageremployee")
        except Employee.DoesNotExist:
            try:
                vol_user = Volunteer.objects.get(id = masterid)
                vol_user = vol_user.select_related("appuser")
            except Volunteer.DoesNotExist:
                return render(request,
"homeapprepaier/masterinfo.html", {})
            jobs = Volunteer_typicalrepairjob.filter(volunteer=
vol_user)
            jobs = jobs.select_related("typicalrepairjob")
            jobs =
jobs.select_related("typicalrepairjob__appliancecategory")
            jobs =
jobs.order_by("typicalrepairjob__appliancecategory")
            points = Volunteerpoint.objects.filter(volunteer =
vol_user)
            points = points.select_related("volunteer")
            points = points.select_related("appuser")
            points = points.select_related("volunteer__appuser")
            points = points.order_by("-pointdate")
            return render(request,
"homeapprepaier/volunteerevent.html", {"profile": vol_user, "user":
"volunteer", "jobs": jobs, "points": points })
            jobs = Employee_typicalrepairjob.objects.filter(employee
= empl_user)
            jobs = jobs.select_related("typicalrepairjob")

```

```

        jobs =
jobs.select_related("typicalrepairjob__appliancecategory")
        jobs =
jobs.order_by("typicalrepairjob__appliancecategory")
        return render(request,
"homeapprepaier/volunteerevent.html", {"profile": empl_user, "user":
"employee", "jobs": jobs, "points": [] })

    def repairserices_view(request):
        repair_services = Repairservice.objects.all()
        repair_services =
repair_services.select_related("manageremployee")
        repair_services = repair_services.order_by("caption")
        return render(request, "homeapprepaier/services.html",
{"services": repair_services})

    def repairservice_info_view(request, repairserviceid):
        try:
            repair_service = Repairservice.objects.get(id =
repairserviceid)
            repair_service =
repair_service.select_related("manageremployee")
        except Repairservice.DoesNotExist:
            return render(request, "homeapprepaier/service.html",
{})
        repairshops = Repairshop.objects.filter(repairservice =
repair_service)
        repairshops = repairshops.order_by("caption")
        return render(request, "homeapprepaier/service.html",
{"service": repair_service, "shops": repairshops })

    def volunteerrepair_search_view(request):
        vol_repairs = Volunteerrepair.objects.filter()
        vol_repairs =
vol_repairs.select_related('managervolunteer__appuser')
        eq = False
        if request.method == 'GET' and 'eq' in request.GET:
            eq = True
        if request.method == 'GET' and 'volcaption' in request.GET
and not eq:
            vol_repair_caption = self.request.GET.get('volcaption')
            vol_repairs = vol_repairs.filter(caption__icontains =
vol_repair_caption)
        elif request.method == 'GET' and 'volcaption' in request.GET
and eq:
            vol_repair_caption = self.request.GET.get('volcaption')
            vol_repairs = vol_repairs.filter(caption =
vol_repair_caption)

```

```

        if request.method == 'GET' and 'voltown' in request.GET and
not eq:
            vol_repair_town = self.request.GET.get('voltown')
            vol_repairs = vol_repairs.filter(town__icontains =
vol_repair_town)
        elif request.method == 'GET' and 'voltown' in request.GET
and eq:
            vol_repair_town = self.request.GET.get('voltown')
            vol_repairs = vol_repairs.filter(town = vol_repair_town)
        if request.method == 'GET' and 'voladdress' in request.GET
and not eq:
            vol_repair_address = self.request.GET.get('voladdress')
            vol_repairs = vol_repairs.filter(longaddress__icontains
= vol_repair_address)
        from_than = datetime.datetime.now()
        if request.method == 'GET' and 'volday' in request.GET and
'volmonth' and request.GET and 'volyear' in request.GET:
            volday = self.request.GET.get('volday')
            volmonth = self.request.GET.get('volmonth')
            volyear = self.request.GET.get('volyear')
            volday = int(volday)
            volmonth = int(volmonth)
            volyear = int(volyear)
            voldate = datetime.datetime(volyear, volmonth, volday)
            repair_excl_id = []
            for volunteer_repair in vol_repairs:
                planned_events =
Repairevent.objects.filter(volunteerrepair = volunteer_repair,
eventstart__gte = from_than, eventstart__lte = voldate)
                if not planned_events or planned_events.count() ==
0:
                    repair_excl_id.append(volunteer_repair.id)
            vol_repairs = vol_repairs.exclude(id__in =
repair_excl_id)
            vol_repairs = vol_repairs.order_by("caption")
            return render(request, "homeappreparer/repairsearch.html",
{"repairs": vol_repairs})

def volunteerevents_search_view(request):
    eq = False
    if request.method == 'GET' and 'eq' in request.GET:
        eq = True
    vol_events = Repairevent.objects.filter(active = (True))
    vol_events = vol_events.select_related(volunteerrepair)
    if request.method == 'GET' and 'volcaption' in request.GET
and not eq:
        vol_repair_caption = self.request.GET.get('volcaption')

```

```

        vol_events =
vol_events.filter(volunteerrepair__caption__icontains =
vol_repair_caption)
        elif request.method == 'GET' and 'volcaption' in request.GET
and eq:
            vol_repair_caption = self.request.GET.get('volcaption')
            vol_events = vol_events.filter(volunteerrepair__caption
= vol_repair_caption)
        if request.method == 'GET' and 'voltown' in request.GET and
not eq:
            vol_repair_town = self.request.GET.get('voltown')
            vol_events =
vol_events.filter(volunteerrepair__town__icontains = vol_repair_town)
        elif request.method == 'GET' and 'voltown' in request.GET
and eq:
            vol_repair_town = self.request.GET.get('voltown')
            vol_events = vol_events.filter(volunteerrepair__town =
vol_repair_town)
        if request.method == 'GET' and 'voladdress' in request.GET
and not eq:
            vol_repair_address = self.request.GET.get('voladdress')
            vol_events =
vol_events.filter(volunteerrepair__longaddress__icontains =
vol_repair_address)
        from_than = datetime.datetime.now()
        if request.method == 'GET' and 'volday' in request.GET and
'volmonth' and request.GET and 'volyear' in request.GET:
            volday = self.request.GET.get('volday')
            volmonth = self.request.GET.get('volmonth')
            volyear = self.request.GET.get('volyear')
            volday = int(volday)
            volmonth = int(volmonth)
            volyear = int(volyear)
            voldate = datetime.datetime(volyear, volmonth, volday)
            vol_events = vol_events.filter(eventstart__gte =
from_than, eventstart__lte = voldate)
        vol_set = False
        if request.method == 'GET' and 'volnumber1' in request.GET:
            vol_set = True
            vol_min = self.request.GET.get('volnumber1')
            vol_min = int(vol_min)
        if request.method == 'GET' and 'volnumber2' in request.GET:
            vol_set = True
            vol_max = self.request.GET.get('volnumber2')
            vol_max = int(vol_min)
        vol_event_excl_id = []
        if vol_set:
            for vol event in vol events:

```

```

        event_volunteers =
Repairevent_volunteer.objects.filter(repairevent = vol_event)
        if request.method == 'GET' and 'volnumber1' in
request.GET and ((event_volunteers and event_volunteers.count() <
vol_min) or not event_volunteers):
            vol_event_excl_id.append(vol_event.id)
        if request.method == 'GET' and 'volnumber2' in
request.GET and ((event_volunteers and event_volunteers.count() >
vol_max) or not event_volunteers):
            vol_event_excl_id.append(vol_event.id)
        vol_repairs = vol_repairs.exclude(id__in =
repair_excl_id)
        vol_events = vol_events.order_by("eventstart")
        return render(request,
"homeapprepaire/voleventsearch.html", {"events": vol_events })

def volunteer_search_view(request):
    eq = False
    if request.method == 'GET' and 'eq' in request.GET:
        eq = True
    volunteers = Volunteer.objects.filter()
    volunteers = volunteers.select_related('appuser')
    if request.method == 'GET' and 'volscore1' in request.GET:
        vol_score1 = self.request.GET.get('volscore1')
        vol_score1 = float(vol_score1)
        volunteers = volunteers.filter(reviewscore__gte =
vol_score1)
    if request.method == 'GET' and 'volscore2' in request.GET:
        vol_score2 = self.request.GET.get('volscore2')
        vol_score2 = float(vol_score2)
        volunteers = volunteers.filter(reviewscore__lte =
vol_score2)
    if request.method == 'GET' and 'volusername' in request.GET
and eq:
        vol_username = self.request.GET.get('volusername')
        volunteers =
volunteers.filter(appuser__username__icontains = vol_username)
    elif request.method == 'GET' and 'volusername' in
request.GET and not eq:
        vol_username = self.request.GET.get('volusername')
        volunteers = volunteers.filter(appuser__username =
vol_username)
    if request.method == 'GET' and 'volready' in request.GET:
        volunteers = volunteers.filter(ready = True)
    if request.method == 'GET' and 'voltown' in request.GET and
not eq:
        vol_town = self.request.GET.get('voltown')
        volunteers = volunteers.filter(appuser__town__icontains
= vol_town)

```

```

        elif request.method == 'GET' and 'voltown' in request.GET
and eq:
            vol_town = self.request.GET.get('voltown')
            volunteers = volunteers.filter(appuser__town = vol_town)
        if request.method == 'GET' and 'volname' in request.GET and
not eq:
            vol_name = self.request.GET.get('volname')
            vol_name = vol_name.split(' ')
            name = vol_name[0]
            surname = ' '.join(vol_name[1:])
            volunteers =
volunteers.filter(Q(appuser__first_name__icontains = vol_name) |
Q(appuser__last_name__icontains = vol_name) |
Q(appuser__first_name__icontains = name, appuser__last_name__icontains
= surname) | Q(appuser__first_name__icontains = surname,
appuser__last_name__icontains = name))
        elif request.method == 'GET' and 'volname' in request.GET
and eq:
            vol_name = self.request.GET.get('volname')
            vol_name = vol_name.split(' ')
            name = vol_name[0]
            surname = ' '.join(vol_name[1:])
            volunteers = volunteers.filter(Q(appuser__first_name =
vol_name) | Q(appuser__last_name = vol_name) | Q(appuser__first_name =
name, appuser__last_name = surname) | Q(appuser__first_name = surname,
appuser__last_name = name))
        if request.method == 'GET' and 'voicemail' in request.GET and
not eq:
            vol_email = self.request.GET.get('voicemail')
            volunteers = volunteers.filter(appuser__email__icontains
= vol_email)
        elif request.method == 'GET' and 'voicemail' in request.GET
and eq:
            vol_email = self.request.GET.get('voicemail')
            volunteers = volunteers.filter(appuser__email =
vol_email)
            volunteers = volunteers.order_by("-reviewscore")
            return render(request, "homeapprepairer/volsearch.html",
{"volunteers": volunteers })

def repairshop_search_view(request):
    eq = False
    if request.method == 'GET' and 'eq' in request.GET:
        eq = True
    repair_shops = Repairshop.objects.filter()
    repair_shops = repair_shops.select_related('repairservice')
    if request.method == 'GET' and 'volcaption' in request.GET
and not eq:
        vol_shop_caption = self.request.GET.get('volcaption')

```

```

        repair_shops = repair_shops.filter(caption__icontains =
vol_shop_caption)
        elif request.method == 'GET' and 'volcaption' in request.GET
and eq:
            vol_shop_caption = self.request.GET.get('volcaption')
            repair_shops = repair_shops.filter(caption =
vol_shop_caption)
        if request.method == 'GET' and 'voltown' in request.GET and
not eq:
            vol_shop_town = self.request.GET.get('voltown')
            repair_shops = repair_shops.filter(town__icontains =
vol_shop_town)
        elif request.method == 'GET' and 'voltown' in request.GET
and eq:
            vol_shop_town = self.request.GET.get('voltown')
            repair_shops = repair_shops.filter(town = vol_shop_town)
        if request.method == 'GET' and 'voladdress' in request.GET
and not eq:
            vol_shop_address = self.request.GET.get('voladdress')
            repair_shops =
repair_shops.filter(longaddress__icontains = vol_shop_address)
        if request.method == 'GET' and 'vol_score1' in request.GET:
            vol_score1 = self.request.GET.get('vol_score1')
            vol_score1 = float(vol_score1)
            repair_shops = repair_shops.filter(reviewscore__gte =
vol_score1)
        if request.method == 'GET' and 'vol_score2' in request.GET:
            vol_score2 = self.request.GET.get('vol_score2')
            vol_score2 = float(vol_score2)
            repair_shops = repair_shops.filter(reviewscore__lte =
vol_score2)
        repair_shops = repair_shops.order_by("-reviewscore")
        return render(request, "homeapprepairer/repshopsearch.html",
{"shops": repair_shops })

def applianceitems_view(request):
    if not request.user:
        return render(request,
"homeapprepairer/appliances.html", {"items": [] })
    else:
        appuser = request.user
        appliances = Applianceitem.objects.filter(appuser =
appuser)
        appliances = appliances.select_related("appuser")
        appliances = appliances.order_by("-id")
        return render(request,
"homeapprepairer/applianceitems.html", {"items": appliances })

```

```

def repair_orders_view(request, asuser = True, neworders = 0,
sep = 0):
    if not request.user:
        return render(request, "homeapprepairer/orders.html",
{"orders": [] })
    else:
        appuser = request.user
        if asuser:
            appuser_orders = []
            repairorders = Repairorder.objects.filter(appuser =
appuser)
            repairorders =
repairorders.select_related("repairshop")
            repairorders =
repairorders.select_related("manageremployee")
            repairorders =
repairorders.select_related("appuser")
            repairorders =
repairorders.select_related("repairshop__repairservice")
            from_than = datetime.datetime.now()
            if neworders == 0:
                repairorders = repairorders.filter(status=0)
            elif neworders == 1:
                repairorders =
repairorders.filter(status__gte=6)
            elif neworders == 2:
                repairorders =
repairorders.filter(status__gte=4, status__lte=5)
            elif neworders == 3:
                repairorders =
repairorders.filter(status__gte=1)
                repairorders = repairorders.order_by("orderdate")
                for repairorder in repairorders:
                    positions =
Repairorder_position.objects.filter(repairorder = repairorder)
                    positions =
positions.select_related("repairorder")
                    positions =
positions.select_related("applianceitem")
                    positions =
positions.select_related("applianceitem__appliancemodel")
                    appuser_orders.append([repairorder.orderdate,
"repair", repairorder, positions])
                vmeetings = Virtualmeeting.objects.filter(appuser =
appuser)
                vmeetings = vmeetings.select_related("volunteer")
                vmeetings = vmeetings.select_related("appuser")
                from_than = datetime.datetime.now()
                if neworders == 0:

```

```

        vmeetings = vmeetings.filter(meetingdate__gte =
from_than)
    else:
        vmeetings = vmeetings.filter(meetingdate__lt =
from_than)
    vmeetings = vmeetings.order_by("meetingdate")
    for vmeeting in vmeetings:
        items =
Volunteer_virtualmeeting_item.objects.filter(volunteer_virtualmeeting
= vmeeting)
        items =
items.select_related("volunteer_virtualmeeting")
        items = items.select_related("applianceitem")
        items =
items.select_related("applianceitem__appliancemodel")
        appuser_orders.append([vmeeting.meetingdate,
"meeting", vmeeting, items])
        repevents =
Repair_event_application.objects.filter(appuser = appuser)
        repevents = repevents.select_related("repair_event")
        repevents =
repevents.select_related("manager_volunteer")
        repevents =
repevents.select_related("repair_event__volunteer_repair")
        from_than = datetime.datetime.now()
        if neworders == 0:
            repevents =
repevents.filter(repair_event__eventstart__gte = from_than)
        else:
            repevents =
repevents.filter(repair_event__eventstart__lt = from_than)
        repevents =
repevents.order_by("repair_event__eventstart")
        for revent in repevents:
            items =
Repair_event_application_item.objects.filter(repair_event_application =
revent)
            items =
items.select_related("repair_event_application")
            items = items.select_related("applianceitem")
            items =
items.select_related("applianceitem__appliancemodel")

appuser_orders.append([revent.repair_event.eventstart, "event", revent,
items])
    appuser_orders.sort(key = lambda repdate:
repdate[0])
    return render(request,
"home_app_repairer/applianceitems.html", {"orders": appuser_orders })

```

```

        else:
            try:
                empl_user =
Employee.objects.select_related("appuser").select_related("repairshop"
).get(appuser = appuser)
            except Employee.DoesNotExist:
                try:
                    vol_user =
Volunteer.objects.select_related("appuser").get(appuser = appuser)
                except Volunteer.DoesNotExist:
                    return render(request,
"homeapprepairer/applianceitems.html", {"orders": [] })
                    vmeetings =
Virtualmeeting.objects.filter(volunteer = vol_user)
                    vmeetings =
vmeetings.select_related("volunteer")
                    vmeetings = vmeetings.select_related("appuser")
                    from_than = datetime.datetime.now()
                    if neworders == 0:
                        vmeetings =
vmeetings.filter(meetingdate__gte = from_than)
                    else:
                        vmeetings = vmeetings.filter(meetingdate__lt
= from_than)
                        vmeetings = vmeetings.order_by("meetingdate")
                        volunteer_orders = []
                        for vmeeting in vmeetings:
                            items =
Volunteer_virtualmeeting_item.objects.filter(volunteer_virtualmeeting
= vmeeting)
                            items =
items.select_related("volunteer_virtualmeeting")
                            items =
items.select_related("applianceitem")
                            items =
items.select_related("applianceitem__appliancemodel")
                            volunteer_orders.append([vmeeting.meetingdate, "meeting", vmeeting,
items])
                        repevents =
Repairevent_application.objects.filter(managervolunteer = vol_user)
                        repevents =
repevents.select_related("repairevent")
                        repevents =
repevents.select_related("managervolunteer")
                        repevents =
repevents.select_related("repairevent__volunteerrepair")
                        from_than = datetime.datetime.now()
                        if neworders == 0:

```

```

        repevents =
repevents.filter(repairevent__eventstart__gte = from_than)
        else:
            repevents =
repevents.filter(repairevent__eventstart__lt = from_than)
            repevents =
repevents.order_by("repairevent__eventstart")
            for revent in repevents:
                items =
Repairevent_application_item.objects.filter(repairevent_application =
revent)

                items =
items.select_related("repairevent_application")
                items =
items.select_related("applianceitem")
                items =
items.select_related("applianceitem__appliancemodel")

volunteer_orders.append([revent.repairevent.eventstart, "event",
revent, items])
            volunteer_orders.sort(key = lambda repdate:
repdate[0])

            return render(request,
"homeapprepairer/applianceitems.html", {"orders": appuser_orders })
            service_orders = []
            if sep == 0 or sep == 1:
                repairorders =
Repairorder.objects.filter(manageremployee = empl_user)
                repairorders =
repairorders.select_related("repairshop")
                repairorders =
repairorders.select_related("manageremployee")
                repairorders =
repairorders.select_related("appuser")
                repairorders =
repairorders.select_related("repairshop__repairservice")
                if neworders == 0:
                    repairorders = repairorders.filter(status=0)
                elif neworders == 1:
                    repairorders =
repairorders.filter(status__gte=6)
                elif neworders == 2:
                    repairorders =
repairorders.filter(status__gte=4, status__lte=5)
                elif neworders == 3:
                    repairorders =
repairorders.filter(status__gte=1)
                    repairorders =
repairorders.order_by("orderdate")
                    for repairorder in repairorders:

```

```

        positions =
Repairorder_position.objects.filter(repairorder = repairorder)
        positions =
positions.select_related("repairorder")
        positions =
positions.select_related("applianceitem")
        positions =
positions.select_related("applianceitem__appliancemodel")

service_orders.append([repairorder.orderdate, "repair", repairorder,
positions])
        if sep == 0 or sep == 2:
            positions =
Repairorder_position_employee.objects.select_related("employee")
            positions = positions.filter(employee =
empl_user)
            positions =
positions.select_related("repairorder_position")
            positions =
positions.select_related("repairorder_position__repairorder")
            positions =
positions.select_related("repairorder_position__repairorder__appliance
item")
            positions =
positions.select_related("repairorder_position__repairorder__repairsho
p")
            positions =
positions.select_related("repairorder_position__repairorder__appuser")
            positions = positions.order_by("plannedstart")
            for serv_position in positions:

service_orders.append([serv_position.plannedstart, "repair",
serv_position])
        service_orders.sort(key = lambda repdate: -
repdate[0])
        return render(request,
"homeapprepairer/applianceitems.html", {"orders": service_orders })

def applianceitem_repairs_view(request, applianceitemid, ext =
False):
    if not request.user:
        return render(request,
"homeapprepairer/appliancerepairs.html", {"repairs": [] })
    else:
        author = False
        appuser = request.user
        appliance_orders = []
        try:

```

```

        appliance = Applianceitem.objects.get(id =
applianceitemid)
        appliance = appliance.select_related("appuser")
    except Applianceitem.DoesNotExist:
        return render(request,
"homeapprepairer/appliancerepairs.html", {"repairs": [] })
        if appliance.appuser == appuser:
            author = True
            empl = False
            positions =
Repairorder_position.objects.filter(applianceitem = appliance)
            positions = positions.select_related("repairorder")
            positions = positions.values_list("repairorder__id")
            repairorders = Repairorder.objects.filter(id__in =
positions)
            repairorders = repairorders.select_related("repairshop")
            repairorders =
repairorders.select_related("manageremployee")
            repairorders = repairorders.select_related("appuser")
            repairorders =
repairorders.select_related("repairshop__repairservice").values().dist
inct()
            repairshops = repairorders.values("repairshop")
            try:
                empl_user = Employee.objects.get(appuser = appuser)
                repairshops2 =
Repairshop.objects.filter(manageremployee = empl_user).values()
                repairshops3 =
Repairorder_position_employee.objects.filter(employee = empl_user)
                repairshops3 =
repairshops3.select_related("repairorder_position")
                repairshops3 =
repairshops3.select_related("repairorder_position__repairorder")
                repairshops3 =
repairshops3.select_related("repairorder_position__repairorder__repair
shop")
                repairshops3 =
repairshops3.values_list("repairorder_position__repairorder__repairsho
p")
                repairshops3 = repairshops3.distinct()
                found = False
                for repairshop in repairshops:
                    if repairshop in repairshops2 or repairshop in
repairshops3:
                        empl = True
                        break
            except Employee.DoesNotExist:
                empl = False
            if empl or author:

```

```

        for repairorder in repairorders:
            positions =
Repairorder_position.objects.filter(repairorder = repairorder)
            positions =
positions.select_related("repairorder")
            positions =
positions.select_related("applianceitem")
            positions =
positions.select_related("applianceitem__appliancemodel")
            appliance_orders.append([repairorder.orderdate,
"repair", repairorder, positions])
            vol = False
            try:
                vol_user = Volunteer.objects.get(appuser = appuser)
                vol = True
            except Volunteer.DoesNotExist:
                pass
            if ext and (author or vol):
                virtualmeeting_items =
Volunteer_virtualmeeting_item.objects.filter(applianceitem =
appliance)
                virtualmeeting_items =
virtualmeeting_items.select_related("applianceitem")
                virtualmeeting_items =
virtualmeeting_items.select_related("volunteer_virtualmeeting")
                positions =
positions.values_list("volunteer_virtualmeeting__id")
                vmeetings = Virtualmeeting.objects.filter(id__in =
positions)
                vmeetings = vmeetings.select_related("volunteer")
                vmeetings = vmeetings.select_related("appuser")
                vmeetings = vmeetings.order_by("meetingdate")
                if vol:
                    vol_vmeetings =
vmeetings.values_list("volunteer")
                    vol_vmeetings = vol_vmeetings.distinct()
                    if vol_user in vol_vmeetings:
                        vol = True
                    else:
                        vol = False
                if author or vol:
                    for vmeeting in vmeetings:
                        items =
Volunteer_virtualmeeting_item.objects.filter(volunteer_virtualmeeting
= vmeeting)
                        items =
items.select_related("volunteer_virtualmeeting")
                        items =
items.select_related("applianceitem")

```

```

        items =
items.select_related("applianceitem__appliancemodel")

appliance_orders.append([vmeeting.meetingdate, "meeting", vmeeting,
items])

        positions =
Repairevent_application_item.objects.filter(applianceitem = appliance)
        positions =
positions.select_related("repairevent_application")
        positions =
positions.select_related("repairevent_application__managervolunteer")
        volunteers =
positions.values_list("managervolunteer")
        positions =
positions.values_list("repairevent__id")
        repevents =
Repairevent_application.objects.filter(id__in = positions)
        repevents =
repevents.select_related("repairevent")
        repevents =
repevents.select_related("managervolunteer")
        repevents =
repevents.select_related("repairevent__volunteerrepair")
        repevents =
repevents.order_by("repairevent__eventstart")
        volunteers2 =
Repairevent_volunteer.objects.filter(repairevent__in = repevents,
approve = True)
        volunteers2 =
volunteers2.values_list("volunteer")
        if vol_user in volunteers2 or vol_user in
volunteers:
            vol = True
        else:
            vol = False
        if author or vol:
            for revent in repevents:
                items =
Repairevent_application_item.objects.filter(repairevent_application =
revent)
                items =
items.select_related("repairevent_application")
                items =
items.select_related("applianceitem")
                items =
items.select_related("applianceitem__appliancemodel")

appliance_orders.append([revent.repairevent.eventstart, "event",
revent, items])

```

```

        appliance_orders.sort(key = lambda repdate: repdate[0])
        return render(request,
"homeapprepaierer/appliancerepairs.html", {"repairs":
appliance_orders})

    def appliancemodel_details_view(request, appliancemodelid):
        if not request.user:
            return render(request, "homeapprepaierer/details.html",
{"details": [] })
        else:
            appuser = request.user
            try:
                empl_user =
Employee.objects.select_related("repairshop")
                empl_user = empl_user.get(appuser = appuser)
            except Employee.DoesNotExist:
                return render(request,
"homeapprepaierer/details.html", {"details": [] })
            try:
                applmodel = Appliancemodel.objects.get(id =
appliancemodelid)
            except Appliancemodel.DoesNotExist:
                return render(request,
"homeapprepaierer/details.html", {"details": [] })
            details = Detailshop.objects.select_related("detail")
            details = details.select_related("repairshop")
            details =
details.select_related("detail__appliancemodel")
            details =
details.select_related("detail__typicaldetail")
            details = details.filter(repairshop =
empl_user.repairshop)
            details = details.filter(detail_appliancemodel =
applmodel)
            return render(request, "homeapprepaierer/details.html",
{"details": details })

    def employee_repairs_view(request, history = True, date_start =
None, date_end = None, state = None):
        if not request.user:
            return render(request,
"homeapprepaierer/emplrepairs.html", {"repairs": [] })
        else:
            appuser = request.user
            try:
                empl_user =
Employee.objects.select_related("repairshop")
                empl_user =
Employee.objects.select_related("repairshop__manageremployee")

```

```

        empl_user = empl_user.get(appuser = appuser)
    except Employee.DoesNotExist:
        return render(request,
"homeapprepairer/emplrepairs.html", {"repairs": [] })
        if date_start:
            date_start = datetime.datetime.strptime(date_start,
'%d-%M-%Y').date()
        if date_end:
            date_end = datetime.datetime.strptime(date_start,
'%d-%M-%Y').date()
        repairshop = empl_user.repairshop
        if repairshop.manageremployee == empl_user:
            repairjobs =
Repairorder_position_employee.objects.select_related("repairorder_posi
tion")
            repairjobs = repairjobs.select_related("employee")
            repairjobs =
repairjobs.select_related("repairorder_position__repairorder")
            repairjobs =
repairjobs.select_related("repairorder_position__applianceitem")
            repairjobs =
repairjobs.select_related("repairorder_position__repairorder__repairsh
op")
            repairjobs =
repairjobs.select_related("repairorder_position__repairorder__
appuser")
            repairjobs =
repairjobs.select_related("repairorder_position__repairorder__managere
mployee")
            repairjobs =
repairjobs.select_related("repairorder_position__repairorder__managere
mployee__appuser")
            repairjobs =
repairjobs.filter(repairorder_position__repairorder__repairshop =
repairshop)
            if date_start:
                repairjobs = repairjobs.filter(plannedstart__gte
= date_start)
            if date_end:
                repairjobs = repairjobs.filter(plannedstart__lte
= date_end)
            if history:
                from_than = datetime.datetime.now().date()
                repairjobs = repairjobs.filter(plannedend__lte =
from_than)
            if state:
                repairjobs = repairjobs.filter(state = state)
            repairjobs = repairjobs.order_by("plannedstart")
            repairs = []

```

```

        for repairjob in repairjobs:
            jobs =
Repairjob.objects.filter(repairorder_position_employee = repairjob)
            repairs.append([repairjob, jobs])
            return render(request,
"homeapprepairer/emplrepairs.html", {"repairs": repairs })

def volunteer_events_view(request, history = 0):
    if not request.user:
        return render(request,
"homeapprepairer/volunteerevents.html", {"events": [] })
    else:
        appuser = request.user
        try:
            vol_user =
Volunteer.objects.select_related("appuser")
            vol_user = vol_user.get(appuser = appuser)
        except Volunteer.DoesNotExist:
            return render(request,
"homeapprepairer/volunteerevents.html", {"events": [] })
        events =
Repairevent.objects.select_related("volunteerrepair")
        events =
events.select_related("volunteerrepair__managervolunteer")
        events = events.filter(volunteerrepair__managervolunteer
= vol_user)
        events = events.values_list("id")
        events2 =
Repairevent_volunteer.objects.select_related("repairevent")
        events2 =
events2.objects.select_related("repairevent__volunteerrepair")
        events2 =
events2.objects.select_related("repairevent__volunteerrepair__managervolunteer")
        events2 = events2.filter(volunteer = vol_user)
        events2 = events2.values_list("id")
        events_final =
Repairevent.objects.select_related("volunteerrepair")
        events_final =
events_final.select_related("volunteerrepair__managervolunteer")
        events_final = events_final.filter(id__in = [events,
events2])
        from_than = datetime.datetime.now().date()
        if history == 0:
            events_final = events_final.filter(eventstart__lte =
from_than)
            events_final = events_final.order_by("-eventstart")
        elif history == 1:

```

```
        events_final = events_final.filter(eventstart__gte =
from_than)
        events_final = events_final.order_by("eventstart")
    elif history == 2:
        events_final = events_final.order_by("-eventstart")
    return render(request,
"homeapprepaierer/volunteerevents.html", {"events": events_final})
```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
кафедра програмних засобів

Дипломна кваліфікаційна робота магістра на тему:
«Дослідження та програмна реалізація методів
класифікації для сервісу ремонту побутової техніки. Research and
Software Implementation of Classification Methods for Home
Appliance Repair Service»

Виконав: студент групи КНТ-212м

Олег ЦИБУЛЬСЬКИЙ

Керівник: к.т.н., доцент

Степан СКРУПСЬКИЙ

2023



Рисунок В.1 – Титульний слайд

Об'єкт, предмет, мета роботи

Об'єкт роботи – процес підтримки роботи сервісу ремонту побутової техніки.

Предмет роботи – методи класифікації заявок на ремонт побутової техніки.

Мета роботи – розробити програмне забезпечення підтримки роботи сервісу ремонту побутової техніки для підвищення зручності роботи всіх задіяних сторін.

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

Завдання роботи

- аналіз існуючого стану проблеми;
- дослідження методів і моделей класифікації заявок на ремонт побутової техніки;
- проектування програмного забезпечення для сервісу ремонту побутової техніки;
- програмна реалізація сервісу ремонту побутової техніки.

3

Рисунок В.3 – Завдання роботи

Характеристика розроблюваної та існуючих програм

Критерій	Repair Service	Repair Cafe Wales	Fixit Clinic	Розробка
Тип сервісу	Комерційний	Волонтерський	Волонтерський	Волонтерський, комерційний
Створення заявки на ремонт	Ремонт, підключення	Ремонт, консультації, волонтерство	Ремонт, консультації, волонтерство	Ремонт, консультації, волонтерство
Керування заявками на ремонт	Немає	Немає	Немає	Як користувачем, так і сервісом
Класифікація заявок на ремонт	Немає	Немає	Немає	Є

4

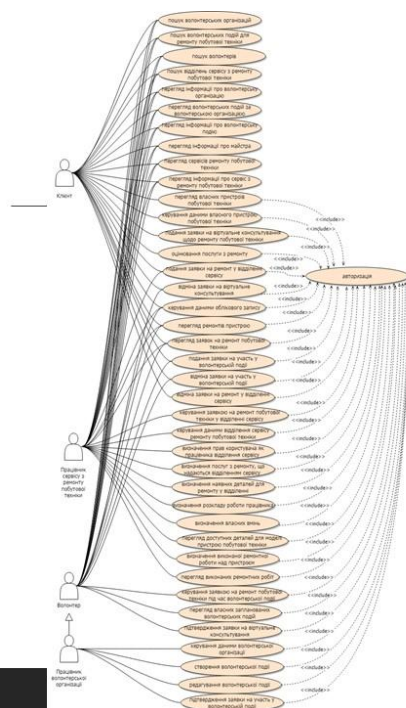
Рисунок В.4 – Характеристика розроблюваної та існуючих програм

Метод класифікації заявок на ремонт побутової техніки

- виділення даних для формування вибірки даних;
- оброблення вибірки даних з переведенням всіх ознак окрім опису проблеми в числові;
- підготовка корпусу текстів та створення моделі оброблення текстів проблем побутової техніки;
- навчання моделі оброблення текстів проблем побутової техніки
- перетворення опису проблеми у вибірці даних у числову форму на основі моделі;
- створення навчальної вибірки даних;
- створення і навчання моделі класифікації на основі методу Random Forest;
- використання навченої моделі класифікації на основі нових даних, які визначені за структурою вхідних ознак.

5

Рисунок В.5 – Метод класифікації заявок на ремонт побутової техніки

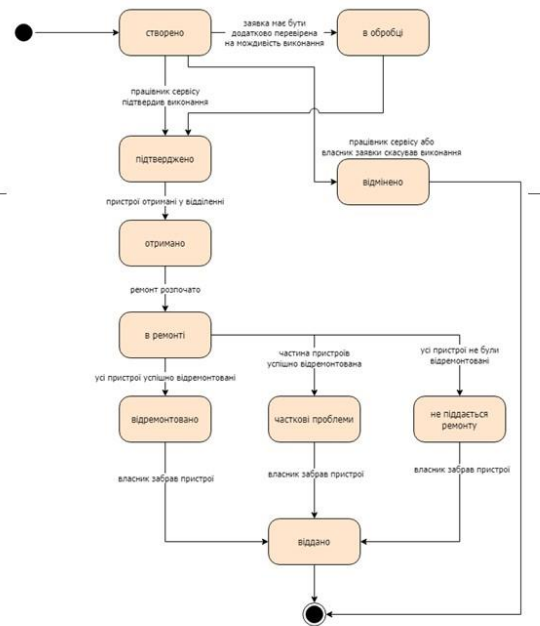


Визначення сценаріїв використання програми користувачами

6

Рисунок В.6 – Визначення сценаріїв використання програми користувачами

Діаграма станів заявки на ремонт побутової техніки у відділенні сервісу



9

Рисунок В.9 – Діаграма станів заявки на ремонт побутової техніки у відділенні сервісу

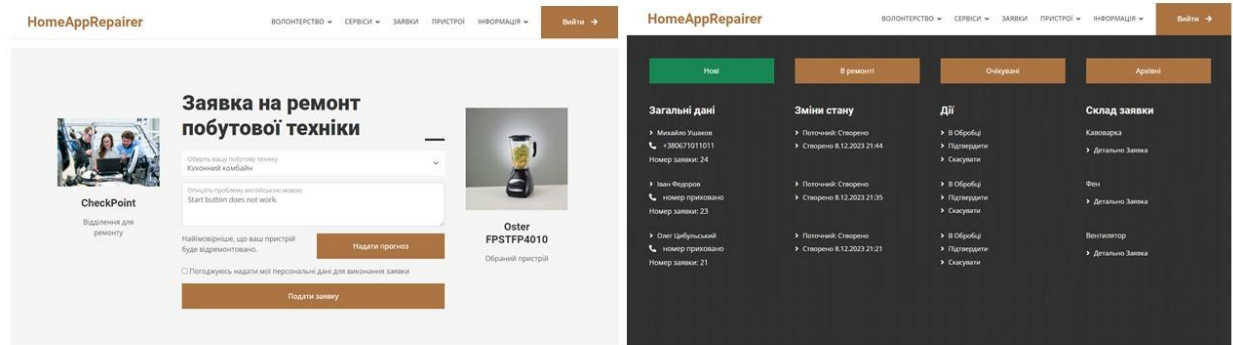
Робота з програмою для сервісу ремонту побутової техніки

The screenshot displays the HomeAppRepairer website interface. The top navigation bar includes links for 'Волонтерство', 'Сервіси', 'Заявки', 'Пристрої', and 'Інформація'. The main content area is divided into two sections. The left section, titled 'Пошук волонтерських подій', features a search form with filters for location (Lviv), date (20.12.2023), and availability. The right section, titled 'Запорізький Мікропогляд', provides details about the 'Сервісний центр "Мікропогляд"', including a 4.8 rating and a 'Попрати заявку' button. Below these sections, there are three featured services: 'Ремонт пристроїв з дерев'яними елементами', 'Ремонт елементарної електроніки', and 'Ремонт і виготовлення найпростіших деталей'. The bottom right corner shows a section for 'Опалювальні котли' with pricing and a 'Замовити' button.

10

Рисунок В.10 – Робота з програмою для сервісу ремонту побутової техніки

Робота з програмою для сервісу ремонту побутової техніки



11

Рисунок В.11 – Робота з програмою для сервісу ремонту побутової техніки

Результати проведеного експериментального дослідження

Модель	Використання опису проблеми	Точність	Влучність
НБК	Ні	0,494062	0,518207
Дерево рішень	Ні	0,519398	0,536116
Random Forest	Ні	0,556611	0,58042
НБК	Так, 100 елементів	0,517022	0,488596
Дерево рішень	Так, 100 елементів	0,64133	0,607553
Random Forest	Так, 100 елементів	0,711797	0,678571
НБК	Так, 20 елементів	0,632621	0,570713
Дерево рішень	Так, 20 елементів	0,626287	0,588832
Random Forest	Так, 20 елементів	0,711006	0,673203
НБК	Так, 10 елементів	0,637371	0,613195
Дерево рішень	Так, 10 елементів	0,585115	0,587762
Random Forest	Так, 10 елементів	0,698337	0,689189
НБК	Так, 50 елементів	0,582739	0,548733
Дерево рішень	Так, 50 елементів	0,620744	0,618211
Random Forest	Так, 50 елементів	0,701504	0,707641

12

Рисунок В.12 – Результати проведеного експериментального дослідження

Результати проведеного експериментального дослідження

Модель	Дерева	Глибина	Точність	Влучність
Random Forest	1000	10	0,701504	0,707641
Random Forest	500	без обмежень	0,725257	0,702454
Random Forest	100	без обмежень	0,707047	0,713115
Random Forest	1000	без обмежень	0,725257	0,744373

13

Рисунок В.13 – Результати проведеного експериментального дослідження

Висновки

Створена програма може використовуватися волонтерськими організаціями, які підтримують виконання ремонту побутової техніки, через надання можливостей інформування про проведення подій, оформлення заявок на ці події та окремими волонтерами для надання віддалених консультацій щодо ремонту, комерційними сервісами ремонту побутової техніки, які мають керувати заявками, визначати порядок роботи над ремонтом, формувати вартість ремонту за виконуваними роботами, визначати використані деталі для роботи, власниками пристроїв побутової техніки для створення заявок на виконання ремонту, для відслідковування виявлених раніше проблем з власними пристроїв, виконаних обслуговувань.

Наукова новизна роботи полягає в створеному методі класифікації заявок на ремонт побутової техніки, який виконує класифікацію заявок на успішно виконані та такі, які не завершилися успішним ремонтом техніки, будує для цього модель класифікації на основі методу Random Forest та перетворює текстовий опис проблеми на основі методу Word2vec для використання в методі.

14

Рисунок В.14 – Висновки