

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ВИЗНАЧЕННЯ ЗАТРЕБУВАНOSTІ ПРОКАТУ ВЖИВАНИХ РЕЧЕЙ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
DETERMINING THE DEMAND FOR SECOND HAND ITEMS

Виконав: студент(ка) 2 курсу, групи КНТ-219м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

Сіденко А.Д.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти магістр

Спеціальність 122 Комп'ютерні науки

(код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

С.О. Субботін

“ ” 2020 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Сіденко Анастасії Денисівни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів визначення затребуваності прокату вживаних речей

Research and Software Implementation of Methods for Determining the Demand for Second Hand Items

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “3” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) грудень 2020 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми прокату вживаних речей. 2. Матеріали і методи. 3. Проектування програмного забезпечення. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація та тестування програми. 6. Економіко-організаційна частина, 7. Охорона праці та безпека в надзвичайних ситуаціях.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Льовкін В.М. доцент		
Економіко-організаційна частина	Пожуєва Т.О., професор		
Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “ 7 ” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз та дослідження предметної області	1-2 тиждень	
2	Розробка та затвердження технічного завдання	3 тиждень	Технічне завдання
3	Дослідження методів розв’язання проблеми	4 тиждень	
4	Проектування структури програмного забезпечення	5 тиждень	Структурна схема
5	Розроблення схеми функціонування програмного забезпечення та модулів	6-7 тиждень	Функціональна схема. Модулі
6	Розроблення програмного коду	8 тиждень	Текст програми
7	Тестування та відлагодження програмного забезпечення	9 тиждень	Програма
8	Розробка розділів пояснювальної записки	10-11 тиждень	Пояснювальна записка
9	Розробка графічного матеріалу	12 тиждень	Графічний матеріал
10	Захист магістерської роботи	13 тиждень	

Студент(ка)

(підпис) Сіденко А.Д.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Льовкін В.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
116 с., 28 рис., 7 табл., 3 дод., 27 джерел.

ПРОКАТ ВЖИВАНИХ РЕЧЕЙ, ЗАТРЕБУВАНІСТЬ ПРОКАТУ, ПРОГНОЗУВАННЯ ЧАСОВОГО РЯДКУ, НЕЙРОННІ МЕРЕЖІ, GRU, КОМПОНЕНТНО-ОРІЄТОВАНЕ ПРОГРАМУВАННЯ.

Об'єкт дослідження – процес прокату вживаних речей. Предмет дослідження – методи визначення затребуваності прокату вживаних речей.

Мета роботи – розроблення програмного забезпечення підтримки прокату вживаних речей для забезпечення обґрунтованості керування речами в прокаті.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, компонентно-орієнтоване програмування, мови програмування C# і Python.

Результати. Виконано аналіз проблеми прокату вживаних речей, розглянуто задачу визначення затребуваності прокату вживаних речей як задачу прогнозування часового ряду, запропоновано власний метод розв'язання задачі.

Наукова новизна роботи визначається запропонованим методом визначення затребуваності прокату вживаних речей, що ґрунтується на моделі прогнозування відповідних значень для різних речей заданої категорії на майбутній період за послідовністю попередніх значень на основі використання моделі архітектури GRU, що дозволяє приймати рішення щодо необхідної кількості речей в прокаті сервісу на майбутній період.

Висновки. Програмно реалізовано систему підтримки прокату вживаних речей, описано прийняті щодо цього рішення.

Галузь використання – сервіси і платформи прокату вживаних речей.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 116 pages, 28 figures, 7 tables, 3 appendixes, 27 sources.

RENTAL OF USED ITEMS, RENTAL DEMAND, TIMETABLE FORECASTING, NEURAL NETWORKS, GRU, COMPONENT-ORIENTED PROGRAMMING.

The object of research is the process of renting used things. The subject of research is methods for determining the demand for rental of used items.

The purpose of the work is to develop software to support the rental of used items to ensure the validity of the decisions concerning management of items in the rental.

Materials, methods and technical means: object-oriented programming, component-oriented programming, C# and Python programming languages.

Results. The analysis of the problem of rental of used things is performed, the problem of determining the demand for rental of used things as a problem of forecasting the time series is considered, the own method of solving the problem is offered.

The scientific novelty of the work is determined by the proposed method of determining the demand for used items, based on the model of predicting the corresponding values for different items for a future period in the sequence of previous values based on the GRU architecture model, which allows deciding on the required number of items in the future period.

Conclusions. The system of support of rent of the used things is programmatically implemented, the decisions accepted concerning it are described.

Field of usage is services and platforms for renting used items.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми прокату вживаних речей	11
1.1 Особливості і основні поняття прокату вживаних речей	11
1.2 Програмне забезпечення прокату вживаних речей	14
1.3 Висновки за розділом 1	18
2 Матеріали і методи.....	19
2.1 Задача визначення затребуваності прокату вживаних речей і методи її розв'язання	19
2.2 Рекурентні нейронні мережі	20
2.3 Архітектура рекурентних нейронних мереж Gated Recurrent Units	21
2.4 Запропонований метод визначення затребуваності прокату вживаних речей.....	24
2.5 Висновки за розділом 2	28
3 Проектування програмного забезпечення	29
3.1 Вибір засобів реалізації програмного забезпечення.....	29
3.2 Функціональні вимоги до програмного забезпечення	30
3.3 Проектування бази даних	32
3.4 Структура програмного забезпечення	37
3.5 Висновки за розділом 3	39
4 Основні рішення щодо реалізації компонентів системи.....	40
4.1 Алгоритм функціонування системи.....	40
4.2 Опис модулів реалізованої системи	41
4.3 Висновки за розділом 4	48
5 Експлуатація та тестування програми	49
5.1 Призначення програми	49
5.2 Умови виконання програми	49
5.3 Виконання програми.....	50

6 Економіко-організаційна частина.....	57
6.1 Планування розробки програмного продукту.....	57
6.1.1 Визначення термінів розробки	57
6.1.2 Розрахунок основної заробітної плати	59
6.1.3 Розрахунок додаткової заробітної плати	60
6.1.4 Відрахування на єдиний соціальний внесок	60
6.1.5 Визначення витрат на матеріали	61
6.1.6 Витрати на спеціальне обладнання	62
6.1.7 Розрахунок накладних витрат.....	65
6.2 Визначення економічної ефективності програмного продукту	66
7 Охорона праці та безпека в надзвичайних ситуаціях.....	68
7.1 Аналіз потенційних небезпек.....	68
7.2 Заходи забезпечення безпеки.....	68
7.3 Заходи з забезпечення виробничої санітарії та гігієни праці	70
7.4 Заходи з пожежної безпеки	75
Висновки	78
Перелік джерел посилання	80
Додаток А Технічне завдання	83
Додаток Б Текст програми	88
Додаток В Слайди презентації.....	109

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

GRU – gated recurrent unit;

ЕОМ – електронно-обчислювальна машина;

ПК – персональний комп'ютер;

ПУЕ – правила улаштування електроустановок.

ВСТУП

Прокат вживаних речей є однією з можливостей продовжити життя речам, які вже були у вжитку. Одночасно це і один з напрямків бізнесу, який потребує менших витрат порівняно з придбанням нових речей. Відповідно за рахунок меншого порогу входження такий бізнес привертає достатньо велику увагу. Як і в будь-якій сфері, в даній сфері важливо якісно приймати рішення. Саме від прийняття якісних рішень залежить успіх бізнесу як такого. Як спочатку, так і через певний час основні рішення пов'язані з визначенням затребуваності речей і плануванням роботи сервісу на основі цього.

Затребуваність речей як показник попиту визначає, яка кількість екземплярів речей потрібна для повноцінної роботи сервісу. Відповідно за допомогою даного показника можна зрозуміти, куди саме потрібно витрачати вільні кошти компанії для вдалого розвитку.

Тому актуальною проблемою є не тільки проблема створення сервісу прокату речей як такого, але і проблема визначення затребуваності прокату вживаних речей.

Вживані речі можуть належати як до однієї конкретної категорії, тоді створюється моносервіс, так і одразу до різних категорій. Достатньо популярними у використанні є загалом сервіси прокату одягу загалом та весільних суконь і костюмів зокрема, інструментів, електроніки, речей для заняття спортом або туристичних походів. У кожній з цих галузей людина може вирішити, що через тимчасову потребу в речі, їй не потрібно купляти одразу нову, а потрібно лише оформити прокат. Це дозволить отримати досвід і вирішити з часом, яку саме власну річ потребує людина, або розв'язати проблему, що виникла раз або виникає достатньо рідко. Відповідно в роботі буде створено програмне забезпечення для підтримки прокату вживаних речей та забезпечено методологічну підтримку роботи даного сервісу в розрізі прийняття рішень.

Об'єкт дослідження – процес прокату вживаних речей. Предмет дослідження – методи визначення затребуваності прокату вживаних речей.

Мета роботи – розроблення програмного забезпечення підтримки прокату вживаних речей для забезпечення обґрунтованості керування речами в прокаті.

1 АНАЛІЗ ПРОБЛЕМИ ПРОКАТУ ВЖИВАНИХ РЕЧЕЙ

1.1 Особливості і основні поняття прокату вживаних речей

Прокат вживаних речей вкладається в концепцію циркулярного використання ресурсів.

В умовах нестачі природних ресурсів і наявності глобальних і економічних проблем людство змушене шукати нові моделі розвитку, відмінні від тих, які існують сьогодні: циркуляційна економіка є одним з можливих шляхів виходу з ситуації, що склалася, вона заснована на використанні замкнутих вартісних ланцюгів (замість загальноприйнятих лінійних), що дозволяє розглядати відходи одного виробничого процесу як ресурс для іншого, у глобальному сенсі даний підхід спрямований на створення нової системи взаємин між суспільством, економікою і природою [1].

Головний принцип циркуляційної економіки – оптимізація виробництва і споживання, при якому не утворюються відходи:

- поняття «відходи» взагалі зникає, вони перетворюються в сировину для подальшого виробництва і так до нескінченності;

- на побутовому рівні відповідно формується нова культура споживання: споживач обізнано відмовляється від надмірної кількості продуктів, які використовуються одноразово або постійно: яскравим прикладом може бути буккросинг, оцифрування друкованої продукції та обладнання – адже навіщо купувати новий інструмент для одноразового ремонту, якщо більше ним не потрібно буде користуватися, таким же прикладом є оренда автомобіля в іншому місті – навіщо витратити ресурси для перегону авто в інше місто;

- в Швеції повторно використовується 99% всіх ресурсів: за допомогою вторсировини опалюють будинки і забезпечують їх електроенергією, виробляють пластик, метал, скло, стічні води очищаються до такої міри, що їх можна пити, спеціальні вантажівки їздять по місту і

забирають електроніку і небезпечні відходи, хімічні речовини, фармацевти приймають залишки ліків, об'ємні відходи, як старий телевізор або зламані меблі, шведи відвозять на переробні центри на околицях міст;

– у той же час в Україні повторно використовується тільки 3 % ресурсів, з них менше 2 % компостується або спалюється для одержання енергії [2].

Прокат вживаних речей є частиною циркуляційної економіки, адже речі при цьому не викидаються, не витрачаються ресурси на виробництво та придбання нових речей.

В межах циркуляційної економіки існує модель спільного використання – це модель, націлена на отримання прибутку без залучення в оборот нових ресурсів:

– найголовніший позитивний ефект – збереження природних ресурсів для наступних поколінь, такий тип економіки також дозволяє створити нові робочі місця і підвищити конкурентоспроможність;

– сервіси спільного використання вважаються інструментами для трансформації традиційної лінійної економіки (добування ресурсів – виробництво – споживання – відходи) в модель замкнутого циклу, коли до кожного елементу цього ланцюжка застосовується новий підхід, роблячи всю модель більш усвідомленою.

– зв'язки між виробництвом і споживанням сильно трансформуються: економіка спільного використання дозволяє безпосередньо пов'язувати ключових економічних агентів і розподіляти продукти і послуги між ними без участі посередників;

– з точки зору технологій стало набагато простіше пов'язувати економічних агентів за допомогою платформних технологій: розвинуте програмне забезпечення здатне безпосередньо пов'язувати обидві сторони взаємодії без посередника;

– спільне споживання передбачає, що зручніше платити за тимчасовий доступ до благ, ніж володіти ними [3]-[4].

За прогнозом до 2025 року світовий обсяг економіки спільного використання збільшиться майже в 20 разів і досягне 335 млрд. доларів, свідчать дані зі звіту PwC [5]

Так концепція «позиченого» гардероба була оцінена великими гравцями масового ринку, прибуток яких у зв'язку із загальною стурбованістю екологією і курсом на свідоме споживання різко почав падати: за останній рік подібну послугу ввели в American Eagle, Scotch & Soda, Urban Outfitters – прокат декількох предметів гардероба здійснюється за певну абонентську платню на місяць, при цьому після закінчення тестового використання можна придбати вподобані речі зі знижкою, для деяких компаній така можливість розцінюється як джерело даних для аналізу найпопулярніших моделей, загалом для великих брендів фактично з'являється ще один, хоча і специфічний, канал продажів [6].

Якщо говорити тільки про ринок вживаного одягу і лише в Сполучених Штатах Америки, то за останні роки можна виділити наступні цифри:

- 11 млрд. доларів у 2012 році;
- 12 млрд. доларів у 2013 році;
- 14 млрд. доларів у 2014 році;
- 15 млрд. доларів у 2015 році;
- 18 млрд. доларів у 2016 році;
- 20 млрд. доларів у 2017 році;
- 24 млрд. доларів у 2018 році;
- 28 млрд. доларів у 2019 році;
- 32 млрд. доларів у 2020 році [7].

До найбільш відомих сервісів спільного споживання в світі на даний момент можна віднести:

– Airbnb – шерінговий сервіс, який працює в 191 країні світу, це платформа, яка дозволяє здавати в оренду своє житло, повністю або частково, сайт є посередником між господарем і гостем, Airbnb отримує

відсоток від транзакцій, для роботи потрібно зареєструватися та прив'язати до акаунту банківську карту;

– blablacar – ще один всесвітній і дуже популярний в Україні сервіс, який дозволяє розділити поїздку, засновники називають його сервісом по карпулінгу, тобто відбувається пошук пасажирів, якому по дорозі з водієм, і витрати розділяються (на відміну від райдшерінга, коли спеціально везуть пасажирів в потрібному йому напрямку), для роботи потрібно зареєструватися на сайті або в застосунку, можна запропонувати поїздку – шукати попутників у своєму напрямку – або знайти поїздку, тобто водія;

– getmancar – один з сервісів каршерінгу, це похвилинна або погодинна оренда автомобіля, в основному використовується для коротких поїздок в межах міста, машини належать компанії, а не іншим користувачам сервісу, для роботи потрібно зареєструватися в застосунку, відзначити вік і водійський стаж, дочекатись підтвердження реєстрації, для того, щоб бронювати автомобілі, необхідно прив'язати до акаунту банківську карту;

– сервіс «Rent a Brand» – сервіс оренди нарядів: вечірня, коктейльна або весільна сукня може коштувати дуже дорого, а надягають її зазвичай пару раз на рік, в асортименті у «Rent a Brand» дизайнерські сукні, в тому числі, від провідних дизайнерів галузі з США, Австралії та Великобританії, можна вибрати і забронювати до 10 суконь на сайті і приміряти їх, обрана сукня забирається за день до події і повертається на наступний день, сукню можна застрахувати на випадок пошкодження [8].

1.2 Програмне забезпечення прокату вживаних речей

Сервіс gir.ua (рис. 1.1) дозволяє орендувати будь-які речі, тобто він не є спеціалізованим, а надає можливість орендувати речі, що належать зокрема наступним групам товарів:

- транспорт;
- електроніка;

- одяг і аксесуари;
- інструмент та спеціальну техніку;
- нерухомість;
- товари для дітей та різноманітні інші групи.

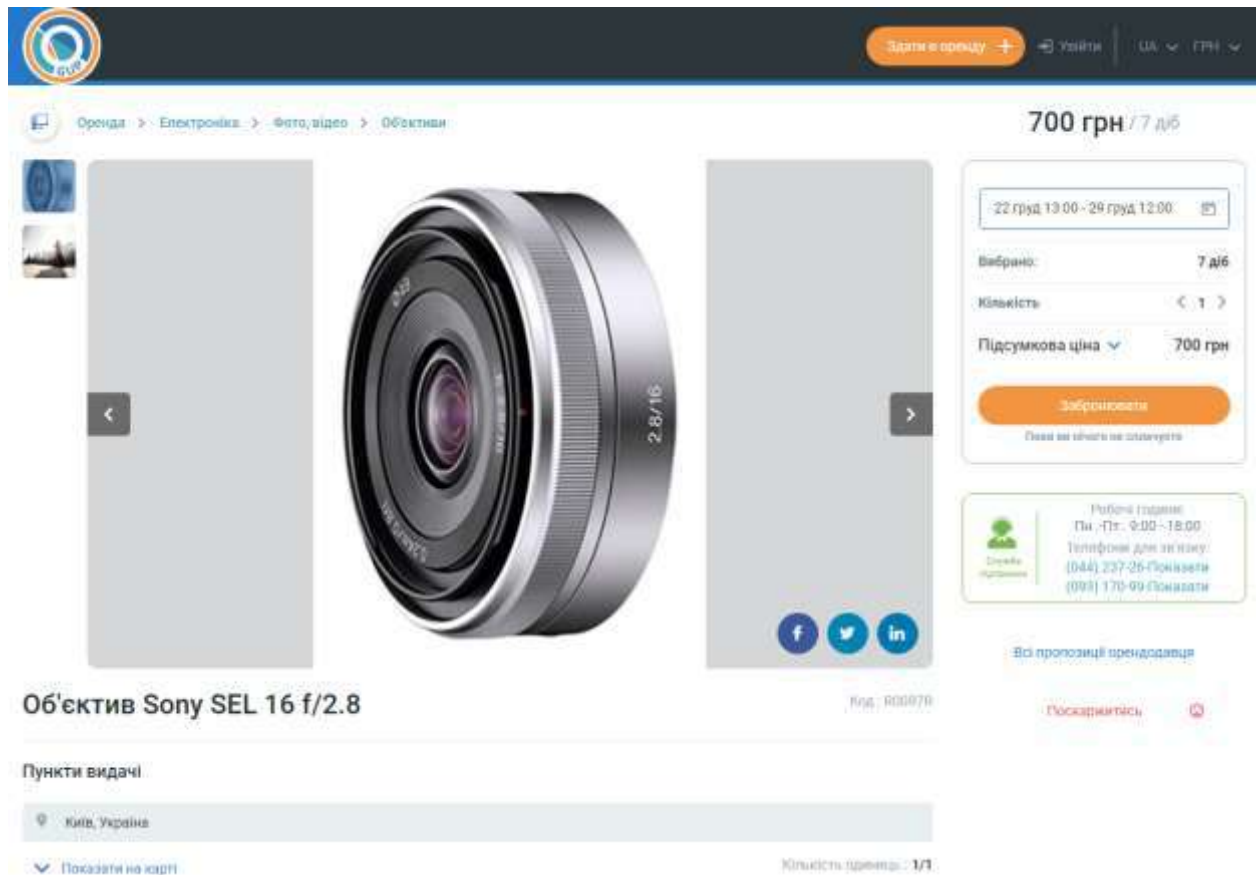


Рисунок 1.1 – Замовлення прокату речі на сервісі gup.ua

Тобто даний сервіс частково можна віднести до сервісів прокату вживаних речей, адже він дозволяє орендувати не тільки вживані речі, але і нерухоме майно, наприклад.

З вебзастосунку можна переглянути інформацію про кожну річ, визначити доступність речі на заданий період, виконати пошук, виставити власну річ в оренду. Надаються варіанти подової, погодинної та похвилинної оренди.

Існує багато сервісів, направлених на виконання прокату речей окремих категорій. Наприклад, baby service (рис. 1.2) – сервіс прокату вживаних речей.



Рисунок 1.2 – Замовлення прокату речі на сервісі baby service

Такі стандартні сервіси дозволяють переглядати доступні товари, а основою такого перегляду є засоби перегляду за категоріями, при цьому засоби пошуку в даному випадку відсутні, при відкритті потрібної речі можна переглянути інформацію про неї, вартість, взяти її на прокат.

Прикладом всесвітнього сервісу прокату є сервіс HSS (рис. 1.3).

The screenshot displays the HSS Hire website interface. At the top, there's a navigation bar with the HSS Hire logo, a search bar, and links for account management. Below this is a secondary navigation bar with categories like 'Search by Category', 'Search by A to Z', 'Specialist equipment', 'Find a branch', 'Tips and advice', and a 'CALL US' button. A third bar highlights services: 'Exclusive Online Discounts', 'Same Day Click & Collect', 'Nationwide Delivery*', and a Trustpilot rating.

The main content area is titled 'Professional Wallpaper Stripper'. It features a large image of a person using the machine. To the right of the image, the product name is repeated, followed by the product code (61132), a 5-star rating from 55 reviews, and a brief description of the machine's function. A 'VIEW FULL DETAILS' link is provided. Below this, there's a 'Safety information & instructions' section with a PDF icon and safety icons (goggles, gloves, RCD, and a vehicle).

On the right side, a pricing table shows rates for different rental durations:

1ST DAY	EXTRA DAY	WEEKEND	WEEK
£22.80	£9.50	£26.60	£38.00

Below the table, there are links for 'Click and collect from our branches nationwide' and 'Same Day delivery available if ordered before 10am'. A 'Click & Collect' button is selected. A form for entering a town or postcode is present, along with 'START DATE' and 'END DATE' dropdowns. A 'CHECK AVAILABILITY' button is at the bottom of this section.

At the bottom left, there's a 'Summary' tab with the following specifications:

- Voltage: 240V
- Power: 2750W
- Tank Capacity: 7.5L
- Base Plate: 253 x 305mm
- Hose Length: 5m run cool anti-kink
- Weight (dry): 8.7kg

On the bottom right, there's a 'Have You Considered?' section featuring an 'Air-Line Extension' product with a price of £15.76/day and a 'Check Availability' button.

Рисунок 1.3 – Замовлення прокату речі на сервісі HSS

Даний сервіс дозволяє переглядати дані про речі, розподілені за характеристиками, отримувати інформацію про виставлені оцінки, відгуки користувачів, надає можливість під час оренди одразу вказувати поштову адресу. При цьому надаються інтервали вартості, тобто в залежності від тривалості прокату визначається вартість її.

1.3 Висновки за розділом 1

Розглянуто концепцію циркуляційної економіки, що є загальним принципом, який реалізується і під час прокату вживаних речей, та економіки спільного використання.

Виконано огляд програмного забезпечення, яке можна використати для забезпечення можливостей прокату вживаних речей. Розглянуто функціональні можливості програмних аналогів, що включають як універсальні сервіси, так і сервіси, що функціонують в межах обмеженої категорії вживаних речей.

Щоб досягти мету, яку було поставлено в роботі, визначено наступні завдання роботи:

- дослідження методів визначення затребуваності прокату вживаних речей;
- розробка технічного завдання;
- вибір засобів розробки;
- проєктування структури програмного забезпечення;
- розробка програмного забезпечення підтримки прокату вживаних речей;
- тестування та відлагодження програми;
- економічне обґрунтування доцільності розробки.

2 МАТЕРІАЛИ І МЕТОДИ

Під час визначення затребуваності прокату речей потрібно визначити, яка кількість замовлень прокату буде оформлена протягом наступного періоду. Визначити це потрібно за минулими періодами, значення для яких відомі. Даний розділ присвячено аналізу існуючих методів, які можуть використовуватися для вирішення даної проблеми, та побудові на основі виділених особливостей власного методу.

2.1 Задача визначення затребуваності прокату вживаних речей і методи її розв'язання

Проблема визначення затребуваності прокату вживаних речей є проблемою прогнозування часового ряду. В різних постановках вона є достатньо широко відомою. Але при цьому саме для прогнозування затребуваності прокату вживаних речей на даний момент запропоновано небагато засобів і найчастіше це на практиці виконується на основі експертного оцінювання.

Можна виділити декілька моделей та методів, що використовуються для прогнозування часових рядів:

а) методи:

1) у одновимірному методі прогнозування часових рядів проблеми прогнозування містять лише дві змінні, в яких одна – час, а інша – поле, яке потрібно прогнозувати (наприклад, якщо потрібно передбачити середню температуру в місті на найближчий тиждень, то одним параметром є час (тиждень), а іншим – місто);

2) у багатовимірному методі прогнозування часових рядів проблеми прогнозування містять множину змінних, зберігаючи одну змінну як фіксований час, а інші будуть кратними за параметрами (у даному випадку при передбаченні температури міста на найближчий

тиждень температура буде враховувати такі фактори, як вплив опадів і тривалість дощу, вологість, швидкість вітру, опади, атмосферний тиск тощо, і тоді температура в місті буде прогнозуватися за цими параметрами, всі ці фактори пов'язані з температурою та впливають на неї);

б) моделі:

1) інтегрована модель авторегресії ковзаючого середнього – це поєднання трьох різних моделей: авторегресія, ковзаюче середнє та інтеграційної, де авторегресія відображає еволюційну змінну, що вивчається, яка реалізує регресію за власними попередніми значеннями, помилка регресії – це лінійна комбінація значень помилок, що мали місце на різних етапах часу раніше, а інтеграція показує, що значення даних замінюються різницею між їх значеннями та попередніми значеннями;

2) авторегресивна умовна гетероскедастичність є найбільш мінливою моделлю для прогнозування часових рядів і добре навчена вловлювати динамічні зміни волатильності від часових рядів;

3) векторна авторегресивна модель встановлює незалежність між різними даними часових рядів, які є узагальненням одновимірної моделі авторегресії;

4) рекурентні нейронні мережі різних архітектур – це моделі глибокого навчання, для зчитування залежностей послідовності, це дозволяє обробляти довгі структури під час навчання на наборі даних та створює прогнози відповідно до попередніх даних [9].

2.2 Рекурентні нейронні мережі

Рекурентна нейронна мережа – це тип штучної нейронної мережі, що використовує послідовні дані або дані часових рядів: ці алгоритми глибокого навчання зазвичай використовуються для порядкових або часових проблем,

таких як переклад, обробка природної мови, розпізнавання мови та субтитрів до зображень, на практиці використовуються в таких програмах, як Siri, голосовий пошук Google та Google Translate, подібно до нейронних мереж прямого поширення сигналу та згорткових нейронних мереж, рекурентні нейронні мережі використовують навчальні дані для навчання, але вони відрізняються своєю “пам’яттю”, оскільки беруть інформацію з попередніх входів, щоб впливати на поточний вхід і вихід, у той час як традиційні глибинні нейронні мережі припускають, що входи та виходи не залежать один від одного, вихід рекурентних нейронних мереж залежить від попередніх елементів у послідовності, хоча майбутні події також могли б допомогти у визначенні результату даної послідовності, однонаправлені рекурентні нейронні мережі не можуть враховувати ці події у своїх прогнозах [10].

Рекурентні нейронні мережі характеризуються наступним:

а) переваги:

- 1) можливість обробки вхідних даних будь-якої довжини;
- 2) розмір моделі не збільшується із збільшенням розміру вхідних даних;
- 3) обчислення враховують історичну інформацію;
- 4) ваги розподіляються за часом;

б) недоліки:

- 1) обчислення повільні;
- 2) труднощі доступу до інформації давнього часу;
- 3) неможливо розглянути будь-яке майбутнє введення для поточного стану [11].

2.3 Рекурентні нейронні мережі Gated Recurrent Units

Довгі добутки матриць можуть призвести до зникнення градієнтів, що на практиці може означати зокрема наступне:

– в ситуаціях, коли раннє спостереження є надзвичайно важливим для прогнозування всіх майбутніх спостережень (наприклад, перше спостереження містить контрольну суму, а мета – визначити, чи правильна контрольна сума в кінці послідовності, тоді вплив першого маркера є життєво важливим для отримання результату), потрібно мати деякі механізми для зберігання важливої ранньої інформації в комірці пам'яті, без такого механізму доведеться визначити дуже великий градієнт цьому спостереженню, оскільки воно впливає на всі подальші спостереження;

– в ситуаціях, коли деякі лексеми не мають відповідного спостереження: наприклад, частина вхідних даних має бути пропущена, адже є допоміжною (наприклад, коментарем) або даними, які не несуть ніякого впливу на остаточний результат, відповідно потрібно мати якийсь механізм пропуску таких даних у прихованому представленні стану;

– в ситуаціями, коли між частинами послідовності відбувається логічний розрив: наприклад, може відбутися перехід між розділами книги або перехід між типами ринків цінних паперів, у цьому випадку потрібно мати засіб скидання внутрішнього представлення [12].

Саме для розв'язання ситуацій, які були перелічені вище, актуальними стають спеціалізовані архітектури рекурентних нейронних мереж: довга короткочасна пам'ять, *gated recurrent unit (GRU)*. Друга архітектура є простішою, відповідно швидшою в реалізації.

Ключовою відмінністю між класичними (або так званими ванільними) рекурентними нейронними мережами та GRU є те, що GRU підтримує регулювання прихованого стану, – це означає, що є спеціальні механізми, коли прихований стан слід оновлювати, а також коли його слід скидати, ці механізми як раз найголовніше і дозволяють вирішувати проблеми, які було перераховано вище (якщо перший маркер має велике значення, мережа буде навчена не оновлювати прихований стан після першого спостереження, аналогічно мережа навчиться пропускати неактуальні спостереження, тобто загалом мережа навчиться скидати прихований стан за потреби):

– першою складовою є ворота скидання та ворота оновлення – це вектори зі значеннями в інтервалі $(0,1)$, такі, що можна виконувати опуклі комбінації: ворота скидання дозволять контролювати, яку частину попереднього стану потрібно пам'ятати, а яку забути, відповідно ворота оновлення дозволить контролювати, яка частина нового стану – це лише копія старого стану, для їх роботи враховуються вхід поточного часового кроку та прихованого стану попереднього часового кроку, виходи двох воріт подаються двома повністю з'єднаними шарами з функцією активації сигмовидної форми;

– ще однією складовою є прихований стан-кандидат – даний елемент називається кандидатом, оскільки потрібно ще включити дію воріт оновлення для обчислення кінцевого результату: на даному кроці вплив попередніх станів можна зменшити за допомогою поелементного множення значення воріт скидання на попередній прихований стан, при цьому кожного разу, коли записи у воротах скидання близькі до 1, відновлюється класичний (ванільний) варіант рекурентної нейронної мережі, для всіх записів воріт скидання, близьких до 0, прихований стан-кандидат є результатом багат шарового персептрона з вхідним сигналом, таким чином, будь-який існуючий прихований стан повертається до значень за замовчуванням.

– останньою складовою є прихований стан, в якому відбувається врахування ефекту воріт оновлення, де визначається, на скільки новий прихований стан є лише старим станом і на скільки використовується новий стан-кандидат, для цього можуть використовуватися ворота оновлення, беручи опуклі комбінації між попереднім прихованим станом та прихованим станом-кандидатом: щоразу, коли ворота оновлення близькі до 1, зберігається старий стан, тоді вхідна інформація по суті ігнорується, фактично пропускаючи даний часовий крок в ланцюжку залежностей, на відміну від цього, коли значення воріт оновлення близьке до 0, новий прихований стан наближається до прихованого стану-кандидата, відповідно можна говорити про те, що ці конструкції допомагають впоратися з

проблемою зникаючого градієнта в рекурентних нейронних мережах та покращити залежності для послідовностей з великими часовими відстанями;

– ворота скидання допомагають фіксувати короточасні залежності в послідовностях відповідно ворота оновлення допомагають фіксувати довгострокові залежності в послідовностях [12]-[13].

Схематично представлена описана логіка роботи комірки GRU на рис. 2.1.

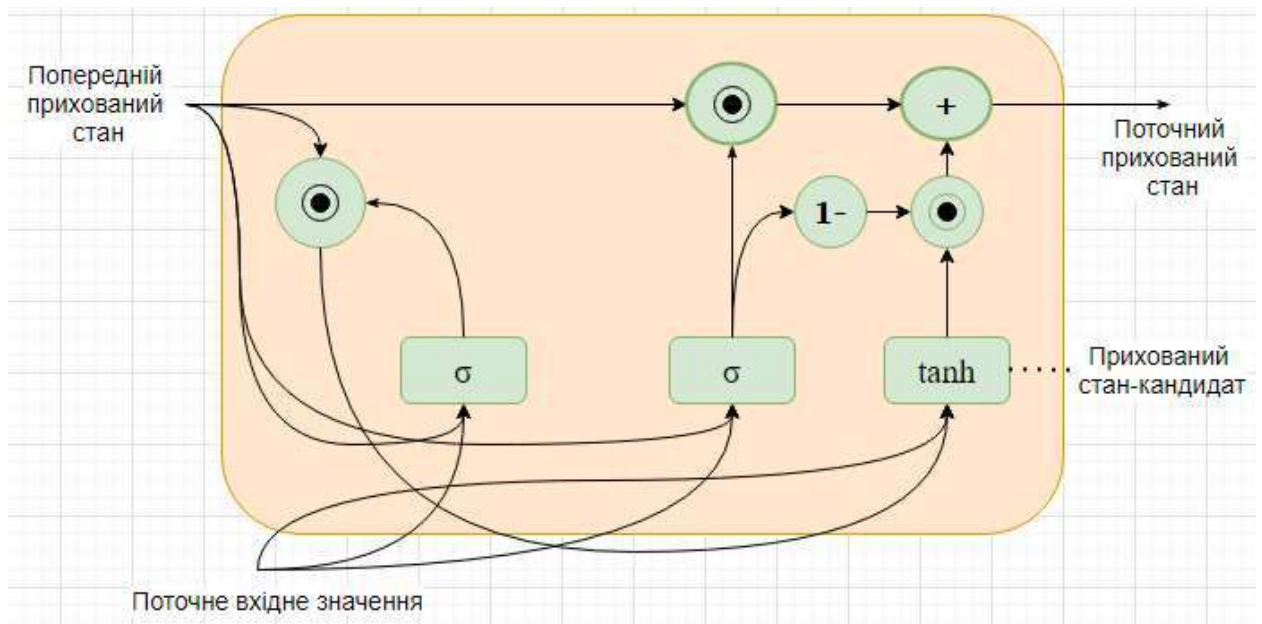


Рисунок 2.1 – Структура комірки GRU

2.4 Запропонований метод визначення затребуваності прокату вживаних речей

Затребуваність прокату вживаних речей – це кількість виконаних прокатів речей у базових одиницях. Під базовою одиницею мається на увазі кількість одиниць часу, на яку відбувається оренда вживаної речі цієї групи. Тобто якщо вартість прокату визначається погодинною оплатою в основі, то сумарна кількість годин, протягом якої відбувався прокат цієї групи речей протягом заданого періоду (місяця, кварталу або півріччя, року) визначає затребуваність прокату групи речей. Якщо вартість прокату оцінюється на

основі щоденного базису, тобто не можна взяти річ в короткостроковий прокат, то сумарна кількість діб прокату буде визначати затребуваність прокату речей.

На початковому етапі виконання визначення затребуваності прокату речей потрібно накопичити дані. Для цього спочатку має бути визначено період T , на який має бути виконано прогнозування в подальшому (місяць, квартал, рік).

Окрім того має бути визначено категорію вживаних речей K , для якої має бути реалізовано прогнозування. Це може бути будь-яка з наявних категорій, для якої накопичено достатню кількість даних. Тобто прогнозування буде виконуватися для заданої категорії вживаних речей. Якщо сервіс прокату займається речима тільки однієї категорії, то відповідно аналізуються всі речі одразу.

Далі всі історичні дані за заданою категорією K , тобто дані з бази даних, мають бути розподілені за всіма заданими періодами T , тобто для всього часового періоду M , протягом якого накопичувались дані в базі даних про прокат вживаних речей, для кожної окремої речі має бути виконано дискретизацію на періоди розміром T . Для кожного періоду має бути визначено кількість прокатів в термінах базової часової одиниці (година або доба в залежності від категорії) для кожної речі, що належить категорії K . При цьому під кожною річчю мається на увазі не екземпляр вживаної речі, а вживана річ загалом, що в базі даних може мати декілька екземплярів, які не відрізняються або майже не відрізняються за характеристиками.

Після цього отриманий набір значень формує часовий ряд для кожної вживаної речі. З отриманого часового ряду має бути сформовано вибірку даних D . Для формування вибірки даних потрібно встановити проміжок базових часових кроків Q , кожен з яких має довжину T .

Відповідно прогнозування буде реалізовуватися на $Q+1$ часовий крок. З кожного екземпляру часового ряду, тобто часового ряду для конкретної вживаної речі, виділяється екземпляр навчальної вибірки $\langle 1, 2, \dots, Q \rangle$,

$Q+1>$, що представляє собою пару з набору вхідних ознак і значення вихідної ознаки, далі виділяється екземпляр $<<2, 3, \dots, Q+1>, Q+2>$, $<<3, 4, \dots, Q+2>, Q+3>$ тощо.

На наступному етапі необхідно реалізувати безпосередньо моделі прогнозування. Модель будується на основі комірок GRU. Після створення моделі має бути забезпечено навчання моделі на основі даних навчальної вибірки. Після цього отримані результати навчання, тобто визначені значення параметрів моделі мають бути збережені.

Наступний етап відбувається тоді, коли минає черговий період T . Тоді реалізується формування для кожної вживаної речі вхідного набору значень. Для цього знову ж, аналогічно першому етапу, виконується обчислення кількості замовлень прокату кожної речі за попередні Q періодів розміром T кожен.

Послідовно на вхід моделі подаються екземпляри даних, і модель прогнозує на виході кількість замовлень прокату на наступний період.

Отримане в результаті роботи методу значення може використовуватися окремо, для кожної речі, або сумарно, для всіх речей даної категорії одразу.

Тоді особа, що приймає рішення, може вирішити, яким чином формувати політику щодо таких речей на майбутнє. Наприклад, можна зрозуміти, чи достатньо кількості екземплярів конкретної речі на майбутній період. Наприклад, якщо якась річ знаходиться в ремонті або була втрачена, то чи потрібно розширювати набір речей в прокаті. Окрім того можна тримати інформацію про те, що прокат якоїсь категорії речей стає неактуальним і змінити роботу сервісу на іншу область (категорію речей).

Схематично представлено метод визначення затребуваності прокату речей, описаний в цьому підрозділі, на рис. 2.2.



Рисунок 2.2 – Схема методу визначення затребуваності прокату вживаних речей

2.5 Висновки за розділом 2

Проаналізовано задачу визначення затребуваності прокату вживаних речей та методів, які можуть до неї застосовуватися як для проблеми прогнозування часового ряду. Розглянуто рекурентні нейронні мережі та детально принципи і особливості роботи архітектури GRU. Запропоновано власний метод для розв'язання задачі.

Наукова новизна роботи визначається запропонованим методом визначення затребуваності прокату вживаних речей, що ґрунтується на моделі прогнозування відповідних значень для різних речей заданої категорії на майбутній період за послідовністю попередніх значень на основі використання моделі архітектури GRU, що дозволяє приймати рішення щодо необхідної кількості речей в прокаті сервісу на майбутній період.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір засобів реалізації програмного забезпечення

У роботі має бути розроблене програмне забезпечення підтримки прокату вживаних речей разом з засобами визначення затребуваності прокату за категоріями. Порівнювались мови Python [14] і C# [15] в якості можливих рішень для реалізації.

Визначення затребуваності буде виконуватися на основі розробленого методу, тобто потребує засобів реалізації машинного навчання, зокрема реалізації нейронних мереж архітектури GRU, що є одним із критеріїв прийняття рішення щодо вибору мови для реалізації. Мова C# не надає бібліотечних засобів роботи саме з цією архітектурою, хоча має засоби створення рекурентних нейронних мереж. Відповідно для зручності розробки потрібно використовувати саме мову Python.

Ще одним важливим критерієм є наявність засобів компонентно-орієнтованого програмування, адже такий підхід дозволяє створювати бібліотеки, які можуть зручно використовуватися в інших проєктах, що належать до однієї предметної області. Оскільки виконується розробка загалом базису предметної області, то важливою відповідно є наявність таких засобів. Мова C# надає такі засоби, на відміну від Python.

Розробка виконується у вигляді застосунку для операційної системи Windows з графічним інтерфейсом, відповідно важливою є зручність реалізації даного процесу. При цьому в подальшому система має розширюватися до рішень на інших платформах, тому засоби кросплатформної розробки також є важливим критерієм прийняття рішень. Python надає засоби розробки застосунків з графічним інтерфейсом, але вони набагато менш зручніші порівняно з C#. Незручність фактично під час розробки в межах дипломної роботи виражається в тому, що за короткі строки зробити це не вдасться.

Обидві мови є сучасними, динамічними, з засобами об'єктно-орієнтованого програмування, при чому мовою C# можна писати більш чисті об'єктно-орієнтовані програми, адже код може складатися тільки з класів.

У результаті вирішено виконувати розробку графічної частини мовою C#, а засобів машинного навчання на основі мови Python. Це дозволить в одній системі об'єднати переваги кожного варіанту і уникнути відповідно недоліків.

Мови порівняно візуально за допомогою табл. 3.1.

Таблиця 3.1 – Порівняння мов програмування

Критерій	C#	Python
Чистота об'єктно-орієнтованого підходу	більша	менша
Час виконання	менший	більший
Компонентно-орієнтоване програмування	+	—
Кросплатформна розробка	+	+
Створення Windows-застосунків з графічним інтерфейсом	+	можливе
Бібліотечні засоби роботи з нейронними мережами архітектури GRU	—	+

3.2 Функціональні вимоги до програмного забезпечення

Програмне забезпечення, що розробляється в роботі, для виконання мети роботи повинно надавати користувачам наступні функціональні можливості:

– пошук речей для прокату;

- перегляд речей за категоріями;
- перегляд найбільш затребуваних речей;
- замовлення прокату речі;
- оформлення прокату речі;
- повернення речі з прокату;
- внесення даних про ремонт речі;
- внесення даних орендаря;
- авторизація користувачів;
- реєстрація користувачів;
- зняття речі з прокату;
- фіксація дефектів речі;
- внесення даних про вартість прокату речі;
- обчислення фактичної вартості прокату речі;
- внесення даних про прокат речей;
- перегляд власних прокатів речей;
- оформлення статистики прокату речей;
- прогнозування затребуваності прокату речей за категоріями.

З програмою працюють менеджери компанії та звичайні користувачі. Звичайні користувачі отримують загальну інформацію і є орендарями. Менеджери відповідно задають дані для них та керують всім процесом. І ті, і ті виступають загалом для системи користувачами. Основний набір сценаріїв виділений на цьому рівні. Відповідно менеджери та орендарі наслідують користувача. До специфічних засобів орендаря належить самостійна реєстрація власних облікових записів та можливість переглядати власні прокати речей, тобто для користувачів підтримується історія.

Загальний сценарій роботи користувачів з системою представлено на рис. 3.1 схематично у вигляді діаграми прецедентів.

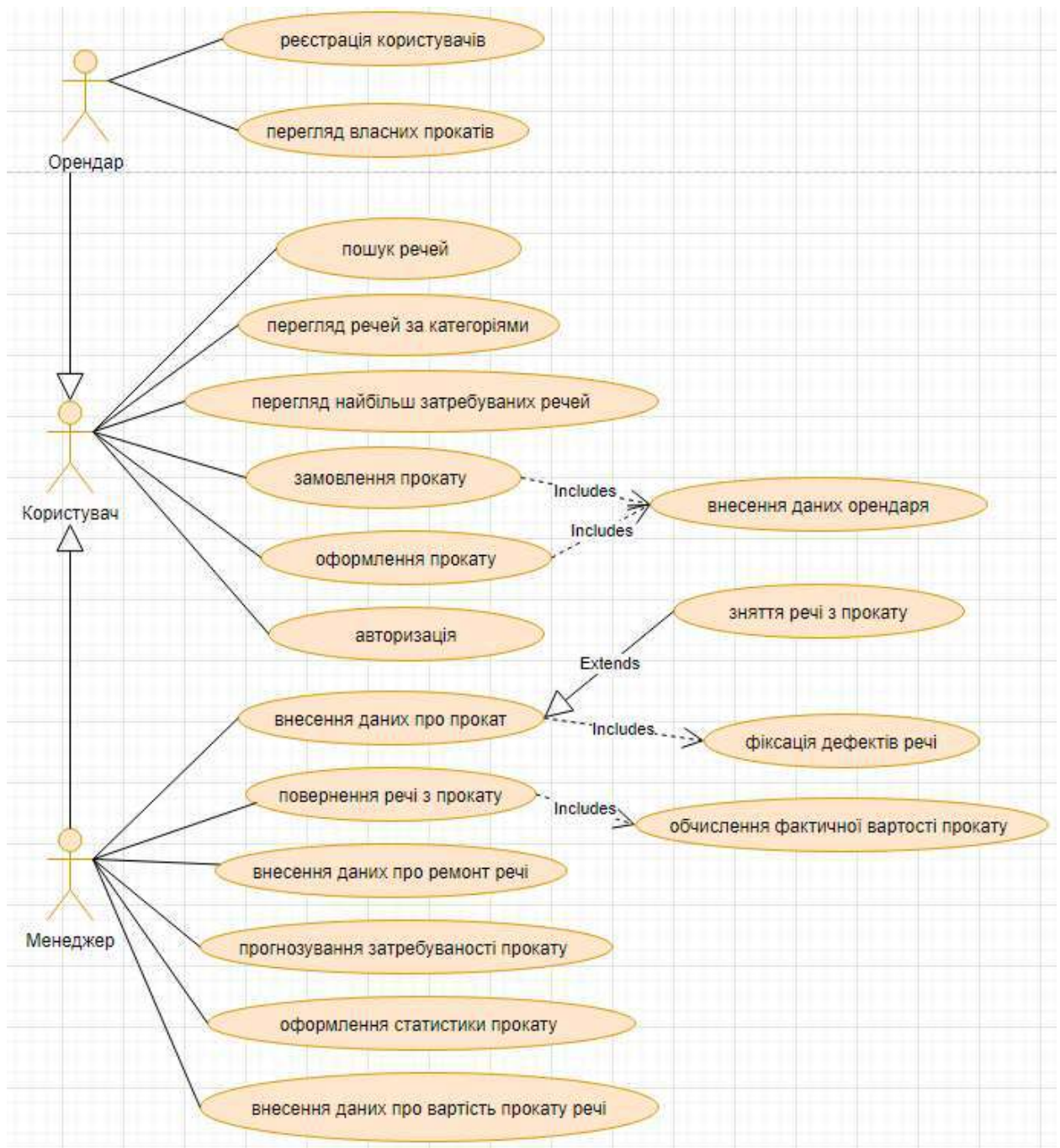


Рисунок 3.1 – Діаграма прецедентів роботи з системою

3.3 Проєктування бази даних

Сутності предметної області прокату вживаних речей:

- User – обліковий запис користувача системи;
- Leaser – інформація про орендаря;
- Item – інформація про вживану річ;

- Lease – інформація про факт прокату речі;
- Repair – інформація про ремонт вживаної речі;
- Defect – інформація про виявлений дефект вживаної речі;
- Category – категорія речей;
- CategoryDemand – інформація про прокат речі за інтервалами часу (місяцями): містить накопичені показники, а також прогноз затребуваності;
- LeasePrice – вартість прокату;
- ItemCopy – екземпляр речі, прокат якої реалізується.

Менеджери компанії, які реалізує прокат речей, мають бути зафіксовані в таблиці User, оскільки кожен з них має взаємодіяти з програмою.

Орендарі можуть бути не зафіксовані в системі, це передбачає, що менеджер виконує через програму фіксацію факту прокату. Тоді в таблиці Leaser фіксується інформація про орендаря.

Якщо орендар зареєстрований в системі, то він відповідно буде мати і обліковий запис користувача, тобто мати запис в таблиці User, і виконувати прокат, тобто мати запис в таблиці Leaser.

Таблиця User включає стовпці:

- id – номер облікового запису;
- login – логін;
- password – пароль;
- type – тип облікового запису (орендар або менеджер).

Таблиця Leaser включає стовпці:

- id – номер орендаря;
- surname – прізвище орендаря;
- name – ім'я орендаря;
- fname – по батькові орендаря;
- passport – номер паспорту орендаря;
- phone – номер телефону орендаря;
- address – адреса орендаря;

– `userid` – обліковий запис користувача, зареєстрованого в системі.

Значення `userid` може бути порожнім, якщо прокат оформлюється менеджером і запис про орендаря створюється в системі.

Таблиця `Category` включає стовпці:

- `id` – номер категорії;
- `name` – назва категорії;
- `description` – опис категорії.

Таблиця `CategoryDemand` включає стовпці:

- `id` – номер запису;
- `periodstart` – початок періоду;
- `periodend` – завершення періоду;
- `categoryid` – номер категорії речей;
- `dnumber` – кількість виконаних прокатів;
- `type` – накопичена статистика (фактична) або прогноз.

Таблиця `LeasePrice` включає стовпці:

- `id` – номер запису;
- `minduration` – мінімальна тривалість прокату, що відповідає даній пропозиції;

- `period` – одиниця часу;
- `price` – вартість одиниці часу;
- `discounted` – визначає, чи застосовується для цієї пропозиції дисконт;
- `itemid` – номер речі, вартість прокату якої встановлюється.

Таблиця `Item` включає стовпці:

- `id` – номер речі;
- `name` – назва;
- `categoryid` – номер категорії, до якої належить річ;
- `description` – опис речі;

Таблиця `ItemCopy` включає стовпці:

- `id` – номер екземпляру речі;
- `itemid` – номер речі;

– state – стан екземпляру.

За допомогою стану екземпляру можна керувати можливістю оренди екземпляру речі.

Таблиця Lease включає стовпці:

- id – номер прокату;
- itemcopyid – номер екземпляру речі;
- userid – користувач, який оформлює прокат (обліковий запис менеджера або користувача);
- leaserid – орендар, для якого виконується прокат;
- starttime – початок прокату;
- endtime – завершення прокату;
- description – опис;
- price – фактична вартість прокату;
- getsum – отримана сума за прокат;
- state – стан прокату.

За допомогою стану прокату можна визначити, на якому етапі знаходиться прокат. Початкове це може бути представлення замовлення, яке може бути скасовано.

Таблиця Repair включає стовпці:

- id – номер запису про ремонт;
- itemcopyid – номер екземпляру речі, ремонт якого відбувається;
- starttime – початок ремонту;
- endtime – завершення ремонту;
- description – опис ремонту;
- interval – час, через який потрібно буде провести планове обслуговування;
- level – складність ремонту.

Загальну структуру бази даних схематично представлено на рис. 3.2.

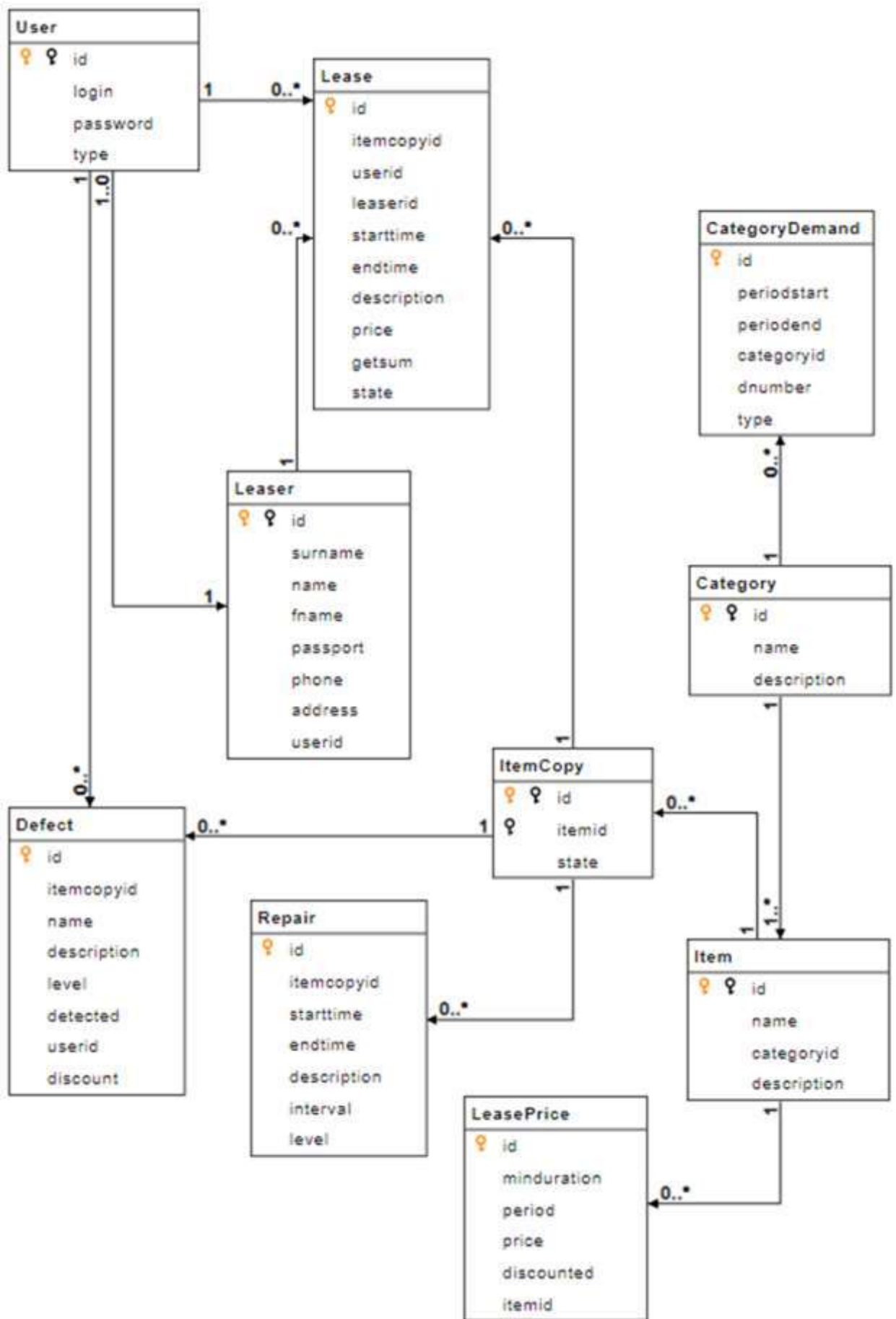


Рисунок 3.2 – Схема бази даних

Таблиця Defect включає стовпці:

- id – номер дефекту;
- itemcopyid – номер екземпляру речі;
- name – назва дефекту;
- description – детальний опис дефекту;
- level – складність дефекту;
- detected – дата виявлення дефекту;
- userid – користувач, який виконав запис про дефект;
- discount – знижка, яка може бути застосована для прокату цієї речі.

3.4 Структура програмного забезпечення

Структуру програмного забезпечення було обрано у відповідності з рішеннями, які було прийнято в цьому розділі раніше, а також враховуючи рішення з розділу 2.

Програмна система складається з трьох підсистем: підсистеми графічного інтерфейсу прокату речей, підсистеми визначення затребуваності прокату речей та підсистеми сутностей предметної області.

Підсистеми графічного інтерфейсу прокату речей і сутностей предметної області є рішеннями, організованими на основі мови C#. Підсистема визначення затребуваності прокату речей організована мовою Python.

Комунікація між підсистемами, реалізованими мовою C#, і підсистемою, реалізованою мовою Python, відбувається тільки за допомогою бази даних, в якій друга запитує і зберігає результати прогнозування, а перша зчитує ці результати і керує всіма іншими даними.

Підсистема сутностей предметної області має бути створена у вигляді DLL-файлу.

На рис. 3.3 схематично представлено структурну схему програмної системи.

До складу підсистеми визначення затребуваності прокату речей входять файли DemandPrediction.py, який повинен виконувати саме прогнозування затребуваності прокату речей, та GRUModel, який повинен реалізовувати модель GRU.

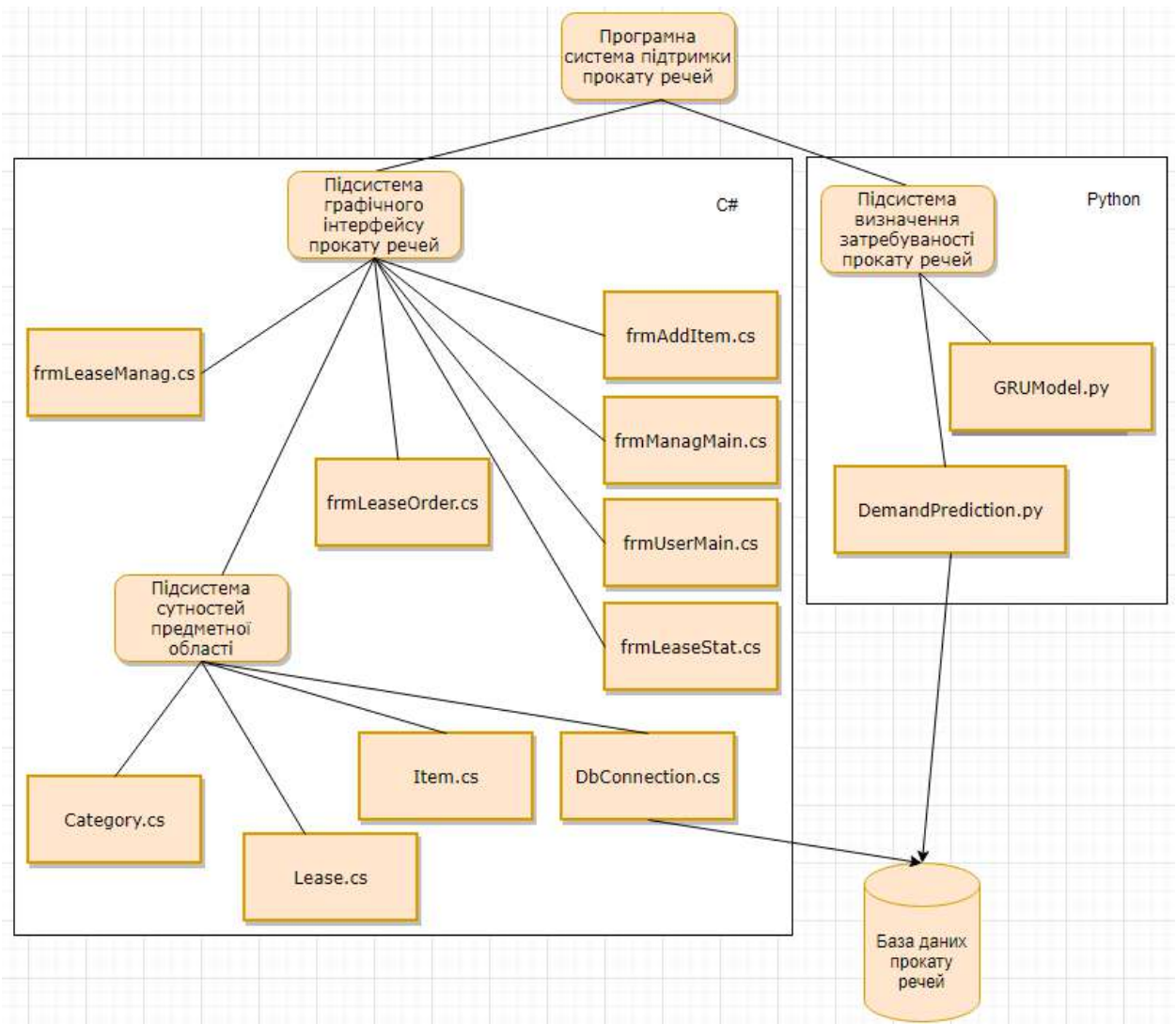


Рисунок 3.3 – Структурна схема програмної системи

До складу підсистеми графічного інтерфейсу повинні входити такі файли (вказано для прикладу основні):

- frmUserMain.cs – головна форма для роботи орендаря з програмою;
- frmLeaseStat.cs – головна форма для перегляду статистики прокату та прогнозування затребуваності прокату речей;

- frmManagMain.cs – головна форма для роботи менеджера з програмою;
- frmAddItem.cs – форма додавання вживаної речі;
- frmLeaseOrder.cs – форма замовлення прокату речі;
- frmLeaseManag.cs – форма керування прокатом речей.

3.5 Висновки за розділом 3

Вирішено виконувати розробку на основі комбінації мов програмування C# та Python. Для цього визначено структуру програмної системи, що функціонально та за рахунок з'єднання з базою даних дозволяє створювати міжмовну програмну систему.

Описано функціональні вимоги до програмного забезпечення, що розробляється, виділено основні сценарії використання програмної системи різними групами користувачів. До користувачів належать менеджери та орендарі.

Забезпечено створення бази даних, для чого було виконано проєктування бази даних, результат якого – побудована схема бази даних та визначені SQL-запити на створення таблиць бази даних.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Алгоритм функціонування системи

Функціонування системи побудовано на основі взаємодії окремих підсистем через базу даних.

На початку відбувається запуск скрипта, після якого ініціалізується навчання моделі. Далі відбувається повторення дій в циклі, кожна ітерація якого виконується щомісяця.

Протягом цієї ітерації спочатку відбувається за допомогою моделі, навчання якої реалізовано на попередньому етапі, визначення затребуваності прокату речей, далі реалізується внесення результатів у бази даних. Саме з бази даних далі будуть братися дані про затребуваність застосунком з графічним інтерфейсом, щоб потім їх відобразити користувачу.

На наступному етапі відбувається цикл роботи з програмою.

Під час даного циклу реалізується завантаження даних з бази даних, де вони зберігаються в табличній формі. Дані перетворюються в форму об'єктів і відображаються з об'єктів у елементах графічного інтерфейсу.

Після цього виконується цикл роботи користувача. Користувач визначає дію, після чого відбувається оновлення даних і за потреби оновлення бази даних, цей цикл повторюється, доки не завершиться роботи програми, тобто доки користувач не закриє програму. Цикл роботи з програмою з графічним інтерфейсом повторюється, допоки не настане черговий місяць. Тоді реалізується наступне прогнозування.

Повторення всіх цих дій реалізується до того моменту, поки адміністратор не схоче змінити розклад запуску скрипта. Якщо розклад не змінюється, то кожного місяця відбувається прогнозування заново.

В результаті створено функціональну схему програми (рис. 4.1).

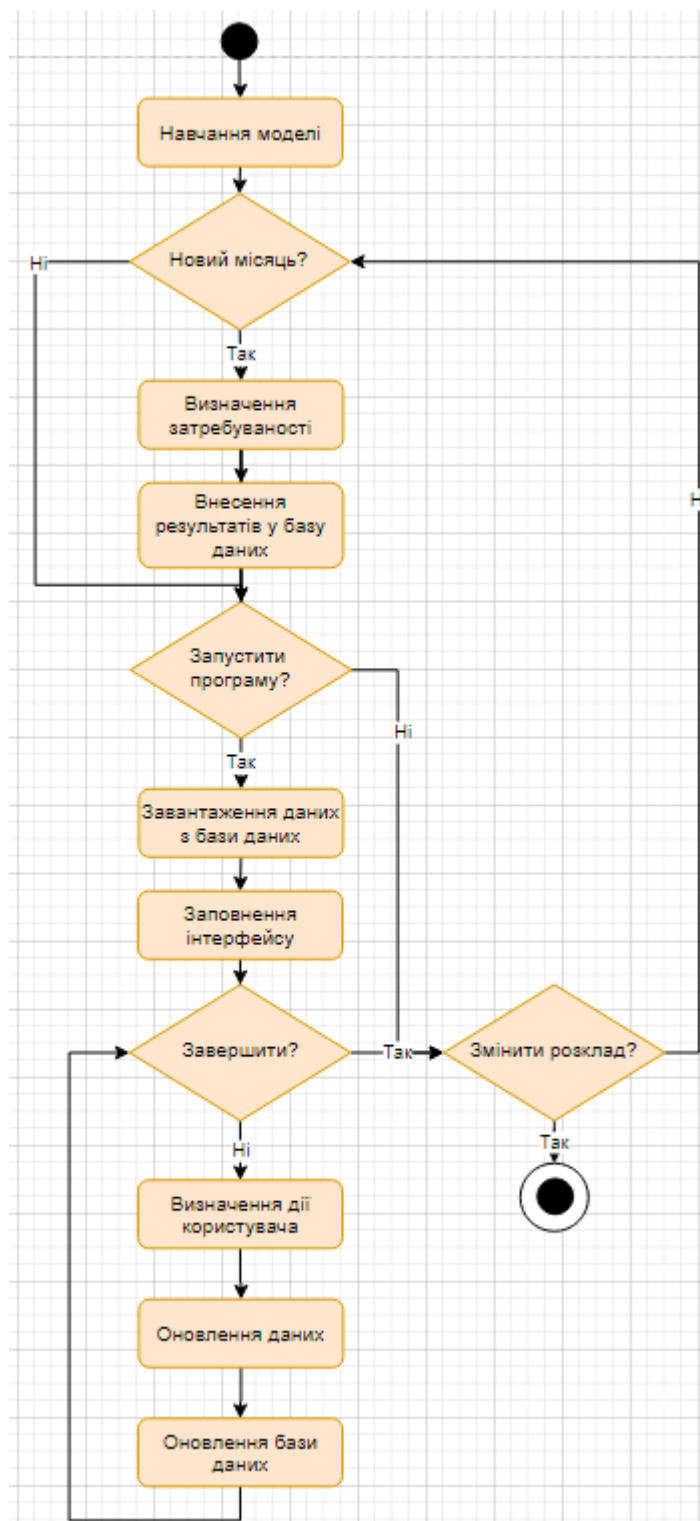


Рисунок 4.1 – Функціональна схема програми

4.2 Опис модулів реалізованої системи

Реалізацію підсистеми сутностей предметної області виконано з забезпеченням програмного інтерфейсу доступу IUsedItems.

В межах інтерфейсу IUsedItems реалізовано наступні класи:

- CategoryDemand – дані про затребуваність прокату речей з однієї категорії;
- Category – категорія речей;
- Lease – прокат одного екземпляру речі;
- Item – вживана річ;
- ItemList – перелік речей, що дозволяє виконувати методи оброблення одразу над всім переліком речей;
- LeaseList – перелік прокатів, що дозволяє виконувати методи оброблення одразу над всім переліком прокатів;
- User – користувач;
- Leaser – орендар;
- Itemcopy – екземпляр речі;
- Repair – виконаний ремонт екземпляру вживаної речі;
- Defect – дефект екземпляру вживаної речі;
- LeasePrice – вартість прокату.

Всі ці класи направлені на те, щоби зберігати дані в програмі, тобто табличне представлення, в якому дані зберігаються в базі даних за допомогою класу UsedItemsDb перетворюються на представлення даних в об'єктній формі. У формі об'єктів перелічених вище класів дані зберігаються в програмі і відповідно використовуються для роботи підсистеми графічного інтерфейсу прокату вживаних речей, тобто класів відповідних форм.

Клас Item окрім методів для встановлення значень полів та отримання цих значень має також наступні методи:

- List <ItemCopy> getActiveItemcopies () – метод, який повертає перелік екземплярів вживаної речі, які в даний момент можуть бути віддані в прокат;
- List <ItemCopy> getRepairItemcopies () – метод, який повертає перелік екземплярів речей, які в даний момент знаходяться в ремонті і відповідно не можуть бути віддані в прокат;

- `List <ItemCopy> getLostItemcopies ()` – метод, який повертає перелік повністю втрачених екземплярів вживаної речі;
- `int getFreeItemcopy ()` – метод, який повертає ідентифікаційний номер першого екземпляру вживаної речі, який може бути відданий в прокат;
- `int getUnactiveItemcopiesCount ()` – метод, який повертає наявну кількість екземплярів речей, які в даний момент знаходяться в ремонті або більше не використовуються в прокаті і відповідно не можуть бути віддані в прокат;
- `int getActiveItemcopiesCount ()` – метод, який повертає кількість екземплярів речей, які в даний момент можуть бути віддані в прокат;
- `float getActiveItemcopiesPercent ()` – метод, який повертає відсоток активних екземплярів речі, відносно всіх наявних екземплярів, тобто відсоток екземплярів, які використовуються в прокаті;
- `float getPrice (DateTime starttime, DateTime endtime)` – метод, який обчислює і повертає вартість прокату цієї вживаної речі, якщо він був реалізований протягом періоду часу з моменту `starttime` до моменту `endtime`;
- `void Save()` – метод, який виконує звернення до класу `UsedItemsDb` і реалізує повне збереження всіх даних, які відповідають вживаній речі, в базу даних;
- `void Read()` – метод, який виконує звернення до класу `UsedItemsDb` і реалізує зчитування всіх даних про вживану річ з заданим ідентифікаційним номером з бази даних.

Метод `getPrice` для обчислення вартості прокату звертається до переліку позицій про вартість прокату речі `Leaseprices`, забезпечує сортування цього переліку, знаходить інтервал між двома датами:

`TimeSpan interval = endtime - starttime;`

`List<LeasePrice> slp = Leaseprices.OrderBy(o=>o.Minduration).ToList();`

Після цього метод знаходить найбільший мінімальний інтервал в переліку Leaseprices, який відповідає даному часовому проміжку, обчислюючи на основі відповідної вартості вартість прокату загалом:

```
foreach (LeasePrice p in slp)
{
    int t;
    if (p.period == 1)
        t = interval.TotalDays;
    else t = interval.TotalHours;
    if (t < p.Minduration)
        return res * t;
    else res = t.Price;
}
```

Клас ItemList містить в якості поля класу перелік вживаних речей Items. Клас включає такий набір методів:

- void AddItem (Item it) – додавання вживаної речі it до переліку Items;
- List <Item> Search (string name) – пошук вживаних речей, що в назві містять name, тобто не обов’язково співпадають, а містять такий підрядок у своєму складі, результат повертається у вигляді переліку вживаних речей;
- List <Item> getCategoryItems (int cat) – метод, який повертає вживані речі, що належать категорії з ідентифікаційним номером cat.

Клас LeaseList містить в якості поля класу перелік прокатів Leases. Клас включає такий набір методів:

- void Order (int itemnum, List <string> desc, float sum, Datetime st, Datetime et, Leaser ls, User us) – метод, який формує замовлення речі з номером itemnum, формуючи при цьому опис замовлення desc, встановлюючи надану клієнтом суму коштів у sum, строки прокату з

моменту *st* до моменту *et*, задаючи в якості орендаря *ls*, а в якості користувача, що формує замовлення, – користувача *us*;

– `int Lease (Item it, List <string> desc, float sum, Datetime st, Datetime et, Leaser ls, User us)` – метод, який формує прокат речі *it*, формуючи при цьому опис прокату *desc*, встановлюючи надану клієнтом суму коштів у *sum*, строки прокату з моменту *st* до моменту *et*, задаючи в якості орендаря *ls*, а в якості користувача, що формує замовлення, – користувача *us*;

– `void LeaseCopy (int ic, List <string> desc, float sum, Datetime st, Datetime et, Leaser ls, User us)` – метод, який формує прокат екземпляру речі з номером *ic*, формуючи при цьому опис прокату *desc*, встановлюючи надану клієнтом суму коштів у *sum*, строки прокату з моменту *st* до моменту *et*, задаючи в якості орендаря *ls*, а в якості користувача, що формує замовлення, – користувача *us*;

– `void LeaseNow (int ic, List <string> desc, float sum, Datetime et, Leaser ls, User us)` – метод, який формує прокат екземпляру речі з номером *ic*, формуючи при цьому опис прокату *desc*, встановлюючи надану клієнтом суму коштів у *sum*, в якості строку завершення прокату – момент *et*, задаючи в якості орендаря *ls*, а в якості користувача, що формує замовлення, – користувача *us*;

– `void ReturnNow (int lnum, List <string> desc, float sum)` – метод, який реалізує повернення з прокату речі з номером *lnum*, змінюючи при цьому відповідний прокат з номером *lnum*, додаючи при цьому опис *desc* та дані суми, сплаченої фактично клієнтом, – *sum*;

– `void Return (int lnum, List <string> desc, float sum, DateTime endper)` – розширений метод, який реалізує повернення з прокату речі з номером *lnum*, змінюючи при цьому відповідний прокат з номером *lnum*, додаючи при цьому опис *desc*, дані суми, сплаченої фактично клієнтом, – *sum*, та дату завершення прокату *endper*, використовується в тих випадках, коли прокат завершено не в момент фіксації в системі, а раніше;

– List <Lease> getUserHistory (int userid) – метод, який повертає перелік прокатів речей, які були замовлені користувачем з номером userid або створені за його участі (тобто менеджери також можуть сформувати такий перелік);

List <Lease> getLeaserHistory (int lid) – метод, який повертає перелік прокатів речей, які були надані користувачу з номером lid;

– List <Item> getLeasesForPeriod (DateTime begdt, DateTime enddt) – повертає перелік речей, які були замовлені за період часу з моменту begdt до моменту enddt;

– int getLeasesForPeriodCount (DateTime begdt, DateTime enddt) – метод, який повертає кількість прокатів речей, які було виконано протягом періоду часу з моменту begdt до моменту enddt.

Для кожного класу забезпечено засоби зчитування та запису даних в базу даних. Для цього створено методи void Save () та void Read ().

Метод Save на прикладі класу CategoryDemand працює наступним чином. Спочатку запитується в базі даних, чи існує запис з вказаним номером. Якщо такий запис існує, то реалізується оновлення даних:

```
if (uidb.existsCategoryDemand(this.Id))
{
    uidb.updateCategoryDemand(this.Id, this.Dnumber, this.Type);
}
```

Якщо такого методу не існує, то реалізується додавання відповідного запису:

```
int cid = this.Categoryid;
int cdn = uidb.addCategoryDemand(this.Periodstart, this.Periodend,
this.Dnumber, this.Type, cid);
this.id = cdn;
```

Клас `UsedItemsDb` містить засоби роботи з базою даних. Для цього окрім параметрів з'єднання з базою даних він надає доступ до набору методів, деякі з них перелічені нижче:

- `open()` – метод, який відкриває з'єднання з базою даних;
- `close()` – метод, який закриває з'єднання з базою даних;
- `existsCategoryDemand(int id)` – метод, який визначає, чи існує запис про затребуваність прокату речей з номером `id`;
- `updateCategoryDemand(int id, int num, int type)` – метод, який оновлює дані про затребуваність прокату речі з номером `id`, кількістю замовлень `num` та типом `type`;
- `addCategoryDemand(Datetime periodstart, Datetime periodend, int num, int type, int cid)` – метод, який додає запис про затребуваність прокату речей про період з ;
- `getCatDemPeriodstart (int id)` – метод, який повертає дату початку періоду прокату для запису про затребуваність прокату речей з номером `id`;
- `getCatDemPeriodend (int id)` – метод, який повертає дату завершення періоду прокату для запису про затребуваність прокату речей з номером `id`;
- `getCatDemDnumber (int id)` – метод, який повертає кількість одиниць прокату запису про затребуваність прокату речей з номером `id`;
- `getCatDemType (int id)` – метод, який повертає тип запису про затребуваність прокату речей з номером `id`;
- `existsCategory(int id)` – метод, який перевіряє, чи існує в базі даних запис про категорію з номером `id`;
- `addCategory(string name, List<String> description)` – метод, який додає Категорію вживаних речей з назвою `name` та описом `description`;
- `getCategoryName (int id)` – метод, який повертає назву категорії вживаних речей з номером `id`;
- `getCategoryDescription (int id)` – метод, який повертає опис категорії вживаних речей з номером `id`.

Побудова моделі виконана мовою Python на основі використання бібліотеки Keras:

```
Demandsmod = Sequential()  
Demandsmod.add (GRU (50, return_sequences = True, input_shape = (10,  
1)))  
Demandsmod.add (GRU (50, return_sequences = False))  
Demandsmod.add (Dense(1))  
adopt = optimizers.Adam (lr = 0.1)  
Demandsmod.compile (optimizer = adopt, loss = 'mean_squared_error')
```

4.3 Висновки за розділом 4

Визначено алгоритм функціонування системи. Описано основні реалізовані класи програмного забезпечення, приведено їх методи. Описано логіку їх реалізації. Описано процес роботи над створенням моделі на основі нейронних мереж.

5 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

5.1 Призначення програми

Програмне забезпечення направлене на реалізацію підтримки роботи менеджерів сервісу щодо внесення можливих варіантів прокату речей, фіксації прокатів речей, отримання статистики за виконаними прокатами та підтримку роботи орендарів щодо замовлення прокатів речей та перегляду і пошуку для цього можливих варіантів.

5.2 Умови виконання програми

Для забезпечення коректної роботи програми потрібно мати наступний набір технічних засобів:

- 32-розрядний або 64-розрядний процесор з частотою не менше 1 ГГц;
- 8 Гб (x32) або 12 Гб (x64) оперативної пам'яті;
- 100 Гб вільного місця на жорсткому диску з обсягом жорсткого диску не менше 500 Гб.

Необхідно враховувати, що вказані параметри мають забезпечувати повноцінну роботу невеликого сервера, тобто за розширення кількості одночасних запитів більше 100 вимоги відповідно мають збільшуватися.

Для експлуатації програми повинен бути виділений сервер або персональний комп'ютер з наявним мережним з'єднанням, який буде відігравати роль сервера. Доступ до сервера може виконуватися зі звичайних комп'ютерів. На всіх має бути інстальовано операційну систему Windows. Окрім того вимагається наявність встановленого фреймворку .NET Framework версії 4.7 або пізнішої.

5.3 Виконання програми

Робота програми розпочинається з запуску на виконання скрипту DemandPredictionSchedule.py. Це дозволяє запускати потім кожен місяць за розкладом скрипт DemandPrediction.py, який реалізує визначення затребуваності прокату вживаних речей на основі методу, який було запропоновано в роботі.

Після цього користувачі зі своїх клієнтських комп'ютерів зможуть запускати програму на виконання. Для цього використовується застосунок з графічним інтерфейсом. Після його запуску потрібно виконати авторизацію, а далі в залежності від типу облікового запису буде виведено один з варіантів графічного інтерфейсу.

Варіант роботи користувача з програмою (рис. 5.1) дозволяє одразу отримати доступ до набору кнопок, що представляють собою меню роботи з програмою:

- кнопка «Історія» дозволяє переглянути історію власних оформлених прокатів речей;
- кнопка «Категорії» дозволяє переглянути доступні для прокату речі за обраною категорією;
- кнопка «Пошук» дозволяє виконувати пошук потрібної речі;
- кнопка «Популярні» дозволяє переглянути найбільш затребувані речі;
- кнопка «Замовлення» дозволяє створити нове власне замовлення (тільки менеджер може оформити прокат, користувач може тільки оформити замовлення, з яким далі буде працювати менеджер);
- кнопка «Профіль» дозволяє змінити власні дані;
- кнопка «Вийти» дозволяє вийти і повернутися до форми авторизації застосунку.

На рис. 5.1 представлена робота в режимі «Історія», відповідно тому дана кнопка має інший колір.

Історія

Категорії

Пошук

Популярні

Замовлення

Профіль

Вийти

Анастасія Сіденко

Прокати

З	До	Вартість	Сплачено	Стан
5.12.2020	8.12.2020	300	100	Виконується
4.12.2020	5.12.2020	300	300	Завершено
1.12.2020	3.12.2020	200	200	Завершено

Сукня Vulkano



Категорія: Одяг

Біла сукня в японському стилі виробництва Vulkano.

Повністю

Рисунок 5.1 – Перегляд власної історії прокату речей

При цьому відобразяться в таблиці «Прокати» всі оформлені прокати речей користувача. Для кожного показується дата початку, дата завершення, вартість, сплачена сума і стан. Якщо річ оцінюється в одиницях менших за добу, то відображається ще й час початку і завершення окрім дат.

Якщо обрати будь-який запис з таблиці, то правіше буде відображено зображення обраної речі (зображення не зберігається в базі даних, а зберігається окремо на сервері, кожна фотографія пронумерована індексом речі), її назву, категорію і опис. Якщо натиснути на кнопку «Повністю», то можна перейти на форму перегляду всіх даних речі. Там буде відображено і всі варіанти вартості прокату, а також буде ширше поле опису для виведення всієї інформації.

В режимі «Категорії» відображається випадючий перелік зі всіма доступними категоріями речей для вибору (рис. 5.2). Потрібно обрати категорію з переліку, після чого в таблиці нижче буде відображено всі речі

цієї категорії. Якщо обрати одну з них, то правіше буде показано зображення речі, опис та варіанти сум сплати за даний товар в залежності від часу прокату.

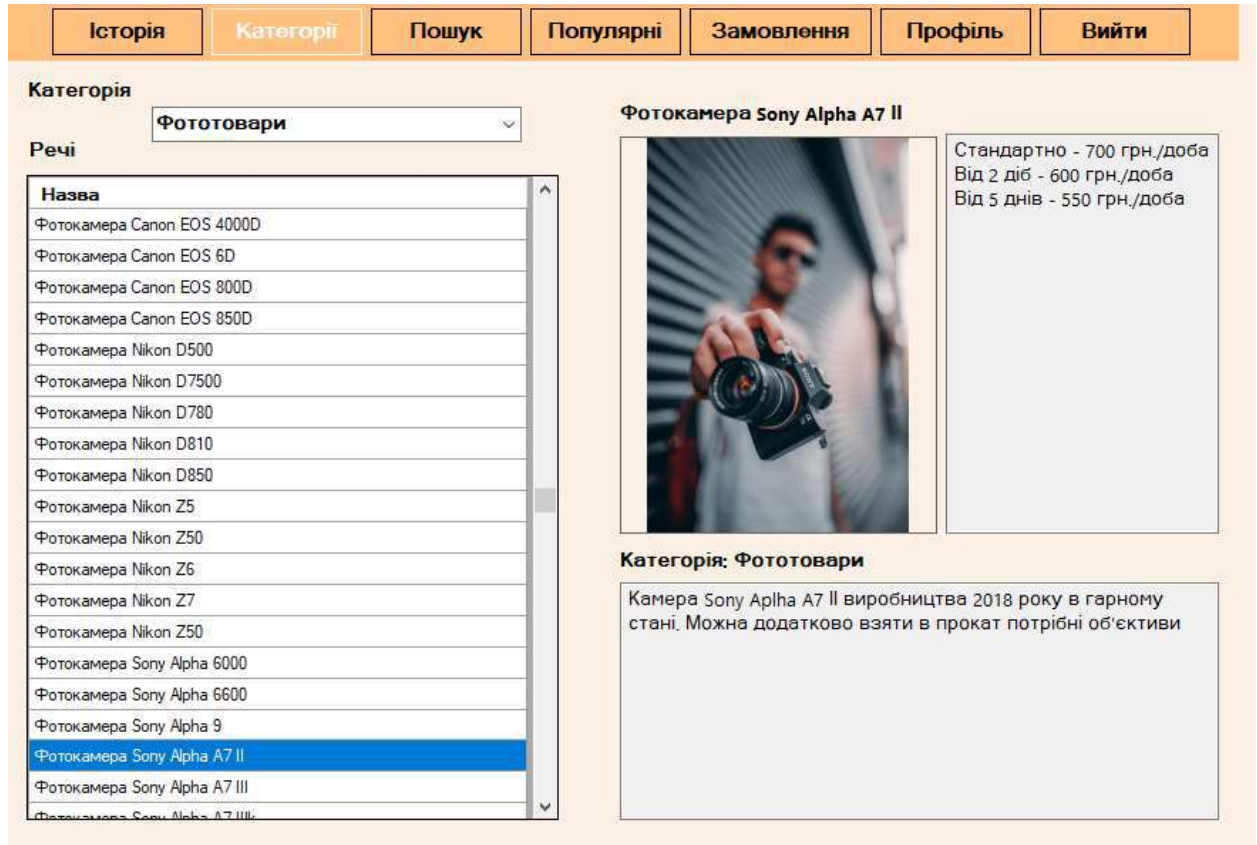


Рисунок 5.2 – Перегляд речей за заданою категорією

У режимі «Пошук» надається можливість ввести назву потрібної речі з можливістю одночасно задати потрібну категорію.

У режимі «Популярні» можна переглянути найбільш затребувані речі в прокаті.

У режимі «Замовлення» можна сформувати замовлення на прокат речі. При цьому орендар може зі свого інтерфейсу тільки створити замовлення. Можливостей повернути з прокату річ або взяти її на прокат він не має.

У режимі «Профіль» орендар може змінити дані про себе, які фактично фіксуються в замовленні на прокат.

Якщо в системі авторизувався менеджер, то відкривається вікно, що має інший набір пунктів меню.

Пункти меню менеджера включають:

- кнопка «Речі» дозволяє переглянути менеджеру доступні для прокату речі, а також виставити необхідну річ на прокат та зафіксувати виявлені дефекти;

- кнопка «Замовити» дозволяє сформувати замовлення на прокат: у даному випадку річ фіксується на потрібний період, але при цьому для видачі речі потрібно використовувати пункт «Прокат»;

- кнопка «Прокат» дозволяє оформити прокат речі в момент, коли орендар забирає річ та вносить суму сплати або передоплати;

- кнопка «Повернути» дозволяє зафіксувати факт завершення прокату речі і за наявності зафіксувати виявлені дефекти при цьому;

- кнопка «Профілі» дозволяє перейти до керування менеджером даними профілів користувачів;

- кнопка «Планувати» дозволяє переглянути статистичні дані про прокат речей, а також переглянути прогнозовані дані;

- кнопка «Ремонт» дозволяє внести дані про ремонт речей і стан виконання даного процесу;

- кнопка «Вийти» дозволяє залишити профіль і вийти з програми до авторизації.

У випадку оформлення прокату речі (рис. 5.3) менеджер повинен спочатку обрати потрібну річ та задати її екземпляр. Для цього він може задати з випадającego переліку категорію та ввести назву речі або її частину. Категорію можна не задавати при цьому. Далі необхідно натиснути на кнопку «Шукати», в переліку «Речі» буде відображено речі, що відповідають запиту. Необхідно обрати одну з позицій з переліку та натиснути на кнопку «Обрати». Далі необхідно обрати конкретний екземпляр речі з випадającego переліку. Тут зафіксовані номери екземплярів речей. Якщо не обрати жодний, то буде обрано перший з доступних. Дане значення може бути

заблоковано, якщо у введені дати прокату виявиться, що екземпляр речі зайнятий.

Речі	Замовити	Прокат	Повернути	Профілі	Планувати	Ремонт	Вийти
------	----------	--------	-----------	---------	-----------	--------	-------

Категорія

Одяг

Назва

весільна біла сукня

Шукати

Речі

Назва
Весільна біла сукня Ema Bride 20-05 38
Весільна біла сукня Elisabetta Franchi

Обрати

Номер екземпляру

93

Номер телефону орендаря

Прізвище орендаря

Пучков

Шукати

Орендар

Пучков Андрій Іванович

Додати

Дата початку

7.12.2020

Дата завершення

8.12.2020

Підтвердити

Отримано

1000

Вартість

1000

Опис

Оформити

Рисунок 5.3 – Оформлення прокату

Після цього потрібно задати дані орендаря, щоб визначити, хто саме буде брати участь в прокаті. Для цього потрібно або ввести номер телефону орендаря, або його прізвище і натиснути на кнопку «Шукати». У випадковому переліку «Орендар» буде відображено всіх знайдених орендарів.

Якщо потрібного орендаря немає в системі, то необхідно натиснути на кнопку «Додати» і внести його дані на окремій формі.

Далі потрібно вказати, з якої дати і до якої буде відбуватися прокат (можна вказати і час), а далі натиснути на кнопку «Підтвердити». Якщо дата початку не буде введена, то вона буде зафіксована поточним часом.

На основі цих даних буде розраховано вартість оренди і виведено в поле «Вартість». Менеджер може внести суму, яку сплатив вже орендар, та зафіксувати необхідні замітки в полі «Опис». В кінці потрібно натиснути на кнопку «Оформити».

З форми «Речі» можна перейти до окремої форми додавання речі на прокат (рис. 5.4).

Мін тривалість	Вартість
1	400
2	350

Рисунок 5.4 – Додавання речі в прокат

В режимі «Планувати» (рис. 5.5) необхідно обрати категорію речей, ввести в поле пошуку назву або частину назви, після чого натиснути на

кнопку «Зберегти» і обрати з переліку «Річ» ту річ, що відповідає потребі, а далі задати рік, для якого потрібно відобразити статистику затребуваності. Після цього необхідно натиснути на кнопку «Показати», тоді відобразиться в таблиці інформація за рік (або меншу кількість, якщо поточний місяць не грудень). В останній позиції відображається прогноз, у всіх інших позиціях – статистично накоплені показники.

Речі	Замовити	Прокат	Повернути	Профілі	Планувати	Ремонт	Вийти
------	----------	--------	-----------	---------	-----------	--------	-------

Категорія

Фототовари

Пошук

Nikon

Зберегти

Річ

Фотокамера Nikon Z7

Рік

2020

Показати

Місяць	Кількість
1	45
2	42
3	45
4	47
5	53
6	60
7	57
8	63
9	59
10	60
11	57
12 прогноз	60

Рисунок 5.5 – Затребуваність прокату

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

6.1 Планування розробки програмного продукту

Розроблений проєкт на тему: “Дослідження та програмна реалізація методів визначення затребуваності прокату вживаних речей”. Програмне забезпечення призначено для виконання прокату вживаних речей. При цьому для формування політики компанії важливо розуміти те, на скільки затребуваними є певні речі в прокаті. Окрім того важливо визначати відповідні інструменти для привертання уваги до деяких речей, розуміючи їхню очікувану затребуваність. Дана функціональність виконується на даний момент експерт, який використовуючи власний досвід знаходження та аналізу джерел, а також інші інструменти, забезпечує відповідне прогнозування. Впровадження програмного забезпечення дозволить скоротити час роботи експерта з 900 до 300 годин.

У розробці беруть участь програміст протягом 3,5 місяців та консультант протягом 1 місяця.

6.1.1 Визначення термінів розробки

Весь комплекс розробки програмного продукту можна розділити на етапи: для кожного етапу вказується трудомісткість, виконавці і тривалість робіт [16]. Розробка програмного забезпечення починається 31 серпня і повинна бути виконана до 15 грудня 2020 року. Зведений стрічковий графік планування розробки представлений у табл. 6.1.

Тривалість робіт визначають за формулою (6.1):

$$T_{\text{ц}} = \frac{Q}{R}, \quad (6.1)$$

де $T_{\text{ц}}$ – тривалість циклу, днів;

Q – трудомісткість, людино-днів;

R – кількість виконавців, чол. [16].

Таблиця 6.1 – Зведений стрічковий графік планування розробки програмного продукту

Найменування роботи	Календарний період, дні														
	31.08	7.09	14.09	21.09	28.09	5.10	12.10	19.10	26.10	2.11	9.11	16.11	23.11	30.11	7.12
1 Аналіз предметної області															
2 Складання технічного завдання															
3 Проєктування структури системи															
4 Розробка схеми функціонування системи															
5 Програмування системи															
6 Тестування та налагодження системи															
7 Складання програмної документації															

Характеристика робіт по розробці програми наведена в табл. 6.2.

Таблиця 6.2 – Характеристика робіт по розробці програми

Задача	Трудомісткість		Виконавці	Тривалість, днів
	люд./днів	%		
Аналіз предметної області	22	22,22	Програміст	11
			Аналітик	
Складання технічного завдання	22	22,22	Програміст	11
			Аналітик	
Проектування структури системи	10	10,1	Програміст	10
Розробка схеми функціонування системи	10	10,1	Програміст	10
Програмування системи	17	17,17	Програміст	17
Тестування та налагодження системи	10	10,1	Програміст	10
Складання програмної документації	8	8	Програміст	8
Всього	99	100		77

6.1.2 Розрахунок основної заробітної плати

Витрати за цією статтею складаються з планового фонду зарплати всіх категорій працівників, зайнятих у розробці програми, розрахунок зарплати ведеться на підставі даних про трудомісткість, представлених в табл. 6.3 [16].

Таблиця 6.3 – Розрахунок основної заробітної плати

Посада виконавця	Чисельність, чол.	Місячний оклад, грн	Кількість місяців	Сума заробітної плати, грн
Аналітик	1	7500	1	7500
Програміст	1	5000	3,5	17500
Підсумок	2			25000

6.1.3 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10% від основної заробітної плати працівника. Розраховують за формулою (6.2):

$$ЗП_{\text{дод}} = ЗП_{\text{осн}} \cdot 0,1, \quad (6.2)$$

де $ЗП_{\text{осн}}$ – сума основної заробітної плати, грн.;

$ЗП_{\text{дод}}$ – сума додаткової заробітної плати, грн. [16].

Підставимо величину основної заробітної плати в формулу (6.2):

$$ЗП_{\text{дод}} = 25000 \cdot 0,1 = 2500 \text{ грн.}$$

6.1.4 Відрахування на єдиний соціальний внесок

Відрахування на єдиний соціальний внесок становлять 22% і беруться від основної та додаткової заробітної плати, вони розраховуються за формулою (6.3):

$$ОТ = (ЗП_{\text{осн}} + ЗП_{\text{дод}}) \cdot 0,22, \quad (6.3)$$

де $ОТ$ – сума єдиного соціального внеску, грн.;

$ЗП_{\text{осн}}$ – сума основної заробітної плати, грн.;

$ЗП_{\text{дод}}$ – сума додаткової заробітної плати, грн. [16].

Розрахуємо:

$$OT = (25000 + 2500) \cdot 0,22 = 6050 \text{ грн.}$$

6.1.5 Визначення витрат на матеріали

Використовується 2 найменування матеріалів: папір – 95 грн. (1 упаковка), картридж – 90 грн.

Витрати на матеріали розраховуються за формулою (6.4):

$$M = \sum_{i=1}^n (C_i \cdot N_i \cdot (1 + K_{\text{т.з.}}) - C_{io} \cdot N_{io}) \quad (6.4)$$

де M – витрати на матеріали, покупні напівфабрикати і комплектуючі вироби, грн.;

$K_{\text{т.з.}}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

C_i – ціна i -го найменування матеріалу, грн.;

N_i – потреба в i -му матеріалі;

C_{io} – ціна зворотних відходів i -го найменування матеріалу, грн.;

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів [16].

Розрахуємо:

$$C_{io}=0, N_{io}=0, K_{\text{т.з.}}=0,05,$$

$$M = (1 + 0,05) \cdot (95 + 90) = 194,25 \text{ грн.}$$

Разом витрати на матеріали складають 194,25 грн.

6.1.6 Витрати на спеціальне обладнання

В цій статті враховуються витрати на оплату машинного часу електронно-обчислювальної машини (ЕОМ) для написання і налагодження даної програми [16].

Амортизаційні відрахування визначаються за формулою (6.5):

$$A = \Phi_6 \cdot H_a, \quad (6.5)$$

де Φ_6 – балансова вартість обчислювальної техніки, грн.;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, % [16].

Балансова вартість обчислювальної техніки становить 12000 грн. одержуємо:

$$A = 12000 \cdot 0,25 = 3000 \text{ грн.}$$

Статтю “Експлуатація обладнання” розраховують підсумовуванням витрат на електроенергію і допоміжні матеріали на основі формули (6.6):

$$C_e = N_n \cdot \Phi_{\text{еф}} \cdot K_{\text{зв}} \cdot K_{\text{зм}} \cdot C_e, \quad (6.6)$$

де N_n – номінальна потужність ЕОМ, кВт;

$\Phi_{\text{еф}}$ – річний ефективний фонд часу роботи ЕОМ, машино-годин;

$K_{\text{зв}}$ – середній коефіцієнт завантаження за часом;

$K_{\text{зм}}$ – коефіцієнт завантаження по потужності;

C_e – ціна одного кВт·год електроенергії, грн./(кВт·год) [16].

Номінальна потужність ЕОМ – 0,5 кВт. Річний ефективний фонд часу роботи ЕОМ становить 1800 годин. Середні коефіцієнти завантаження за

часом і за потужністю рівні відповідно 0,9 і 0,6. Ціна однієї кіловат-години електроенергії становить 2,11 грн.

Отримуємо:

$$C_e = 0,5 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,11 = 1025,46 \text{ грн.}$$

Зарплатня обслуговуючого персоналу розраховується за формулою (6.7):

$$ЗП_{\text{обсл}} = \Phi ЗП_p \cdot (1 + K_{\text{отч}}) \cdot \frac{t_{\text{обсл}}}{\Phi_{\text{еф.обсл}}}, \quad (6.7)$$

де $\Phi ЗП_p$ – річний фонд заробітної плати (основної і додаткової) обслуговуючих робітників, грн.;

$K_{\text{отч}}$ – коефіцієнт, що враховує відрахування на єдиний соціальний внесок;

$t_{\text{обсл}}$ – час протягом року, необхідний на технічне обслуговування ЕОМ, год/рік;

$\Phi_{\text{еф.обсл}}$ – річний ефективний фонд часу обслуговуючого персоналу, год/рік [16].

Місячна заробітна плата обслуговуючого персоналу становить 3800 грн., а річний фонд заробітної плати відповідно дорівнює 45600 грн. Річний ефективний фонд робочого часу обслуговуючого персональний комп'ютер (ПК) працівника дорівнює 1750 год./рік. На обслуговування одного ПК витрачається по 1 годині на місяць, що на рік становить 12 годин.

Згідно формули (6.7):

$$ЗП_{\text{обсл}} = 45600 \cdot (1 + 0,22) \cdot \frac{12}{1750} = 381,47 \text{ грн.}$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від

балансової вартості обладнання і складає 360 грн.

Стаття «Інші витрати» приймається рівною п'яти відсоткам від суми всіх попередніх статей витрат на утримання та експлуатацію устаткування. Сума всіх попередніх статей дорівнює 4766,93 грн., 5% від суми становлять 238,34 грн.

Розраховані статті витрат на утримання та експлуатацію устаткування внесені в табл. 6.4.

Таблиця 6.4 – Кошторис витрат на утримання та експлуатацію обладнання

Найменування статей витрат	Сума, грн
Амортизація обладнання	3000
Експлуатація обладнання (крім витрат на поточний ремонт)	1025,46
Заробітна плата основна і додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	381, 47
Поточний ремонт обладнання	360
Інші витрати	238,34
Підсумок	5005,27

Витрати на оплату машинного часу ЕОМ для написання і налагодження даної програми визначаються за формулою (6.8):

$$C_{\text{мо}} = P_{\text{екс}} \cdot t_{\text{мо}}, \quad (6.8)$$

де $C_{\text{мо}}$ – витрати на оплату машинного часу, грн.;

$P_{\text{екс}}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн. / машино-год.;

$t_{\text{мо}}$ – машинний час цифрової ЕОМ для написання і налагодження даного програмного продукту, машино-год. [16].

Річний ефективний фонд часу роботи ЕОМ дорівнює 1800 годин. У результаті експлуатаційні витрати на одну годину машинного часу дорівнюють:

$$P_{\text{екс}} = \frac{5005,27}{1800} = 2,78 \frac{\text{грн}}{\text{маш.год}}$$

ЕОМ експлуатується 99 днів в одну зміну, що становить в сумі 792 години. Таким чином, витрати на оплату машинного часу складуть:

$$C_{\text{мо}} = 2,78 \cdot 792 = 2201,76 \text{ грн.}$$

6.1.7 Розрахунок накладних витрат

Результати визначення витрат на розробку програми у вигляді калькуляції кошторисної вартості робіт наведено в табл. 6.5.

Таблиця 6.5 – Калькуляція кошторисної вартості робіт з розробки програми

Найменування статей витрат	Сума, грн	Питома вага до підсумку, %
Основна заробітна плата	25000	54,4%
Додаткова заробітна плата	2500	5,44%
Відрахування на єдиний соціальний внесок та інше	6050	13,16%
Матеріали та комплектуючі	194,25	0,42%
Витрати на спец. обладнання	2201,76	4,82
Накладні витрати	10000	21,76%
Підсумок	45946,01	100%

Накладні витрати становлять 40 % до основної заробітної плати розробників, це складає 10000 грн.

Собівартість програмного забезпечення складає 45946,01 грн.

6.2 Визначення економічної ефективності програмного продукту

Річну економію від впровадження програми, що дозволяє знизити витрати машинного часу ЕОМ для рішення певного завдання в порівнянні із програмою, що застосовувалася раніше, визначають за формулою (6.9):

$$E = (T_{M_1} - T_{M_2}) \cdot B_{\text{екс.КТЗ}} + E_n^a \cdot \frac{\Phi_{\text{б.КТЗ}} \cdot (T_{M_1} - T_{M_2})}{\Phi_{\text{еф.КТЗ}}} - \frac{S_{\text{рп}}}{T_c} + E_{\text{зп}}, \quad (6.9)$$

де T_{M_1}, T_{M_2} – машинний час, необхідний для рішення поставлених завдань, відповідно в старому та у новому варіантах, машино-год./рік;

$B_{\text{екс}}$ – експлуатаційні витрати, що доводяться на 1 год машинного часу ЕОМ, грн/ машино-год.;

E_n^a – нормативний коефіцієнт ефективності капітальних вкладень у засоби автоматизації;

$\Phi_{\text{б.КТЗ}}$ – балансова вартість ЕОМ, грн.;

$\Phi_{\text{еф.КТЗ}}$ – річний ефективний фонд часу роботи ЕОМ, машино-год.;

$S_{\text{рп}}$ – сумарні витрати на розробку програми, грн.;

T_c – термін служби впроваджуваної програми до її морального зношування, років;

$E_{\text{зп}}$ – економія на заробітній платі (до впровадження даного програмного забезпечення дані функції виконувались експертом 900 годин на рік (оплата експерта за годину 50 гривень), а зараз 300 годин) [16].

Розрахуємо:

$$E = (900 - 300) \cdot 2,78 + 0,15 \cdot (12000 \cdot (900 - 300)) / 1800 - 45946,01/5 + \\ + 600 \cdot (1 + 0,22) \cdot 1,1 \cdot 50 = 33338,8 \text{ грн.}$$

Коефіцієнт ефективності капітальних вкладень E і строк їхньої окупності $T_{\text{ок}}$ розраховуються за формулами (6.10)-(6.11):

$$Ef = \frac{E}{S_{\text{пп}}}, \quad (6.10)$$

$$T_{\text{ок}} = \frac{S_{\text{пп}}}{E}, \quad (6.11)$$

$$Ef = 33338,8 / 45946,01 = 0,73,$$

$$T_{\text{ок}} = 45946,01 / 33338,8 = 1,38 \text{ року [16].}$$

Зведемо отримані техніко-економічні показники в табл. 6.6.

Таблиця 6.6 – Техніко-економічні показники

Показники	Сума	Одиниця виміру
Витрати на розробку програми	45946,01	грн.
Річна економія	33338,8	грн.
Строк окупності програми	1,38	рік

Строк окупності програми 1,38 року, що значно менше часу нормативного строку окупності для соціальної сфери (6,67 роки). Річна економія при використанні програми становить 33338,8 гривні. Виходячи з виконаних розрахунків, можна зробити висновок, що розробка даного програмного продукту є економічно доцільною.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

В магістерській роботі проводиться дослідження та програмна реалізація методів визначення затребуваності прокату вживаних речей.

За природою виникнення шкідливі та небезпечні виробничі фактори поділяються на такі групи:

- фізичні (підвищена і знижена температура повітря, надмірна загазованість та запиленість повітря та ін.);
- біологічні (підвищений вміст у повітрі мікроорганізмів та ін.), хімічні (вміст у робочій зоні шкідливих речовин та ін.);
- психофізіологічні (нервово-емоційні перевантаження, розумове напруження та ін.) [17].

7.2 Заходи забезпечення безпеки

Обладнання, що використовується в приміщенні обчислювального центру є споживачем електроенергії, що живиться від змінного струму 220 В, частотою струму 50 Гц від мережі з заземленою нейтраллю, та відноситься до електроустановок до 1000 В закритого виконання [17].

Щоб зменшити ймовірність ураження людини електричним струмом живлення всіх електричних пристроїв здійснюється спеціальним кабелем, корпуси всіх електричних пристроїв виготовляються з неструмопровідних матеріалів: у приміщенні для забезпечення безпечності всі кабелі сховані в спеціальні коробки, штепсельні розетки розташовані на висоті одного та 0,2 метри від підлоги, вимикачі на стінах знаходяться на висоті 1,2 метра від підлоги, використовується електропроводка трифазна, чотирьохжильна, з подвійною ізоляцією, електрична мережа вмикається і вимикається за допомогою пускової апаратури (рубильником), відповідно до міжнародного

стандарту СЕ зазначені міри електробезпеки відповідають вимогам, пропонованим до побутових приладів [17].

Відповідно до правил улаштування електроустановок (ПУЕ) в приміщенні розроблено заходи щодо забезпечення електробезпеки [18].

Також для запобігання ураження людини струмом в приміщенні використовується:

- захисне заземлення;
- окремими розетками по 220 В для кожного робочого місця з обчислювальним пристроєм;
- мережевий фільтр до якого підключений кожен блок живлення обчислювального пристрою або периферійного пристрою [19].

Опір захисного заземлення в обчислювальному центрі становить 3 Ом (за стандартом – не більше 4 Ом) згідно з ПУЕ [18]. Також приміщенні встановлено автоматичні вимикачі, пристрій заземлення контуру згідно з ГОСТ 12.2.007.0–75 [20].

Для профілактики втоми та запобігання статистичного навантаження робітників встановлені такі правила організації робочого місця:

- оптимальна висота клавіатури від підлоги – 65-75 см;
- можливість регулювання висоти і нахилу клавіатури (відстань від поверхні стола до середини клавіатури – не більше 30 мм, кут підйому клавіатури – від 2° до 15°);
- наявність у клавіатури підставки для рук та килимка для миші з захистом від тунельного синдрому;
- рекомендовано використовувати перерви в роботі 10 хв. через кожні дві години згідно Державного нормативно-правового акту з охорони праці 0.00-1.31-99 «Правила охорони праці під час експлуатації електронно-обчислювальних машин» [21].

7.3 Заходи з забезпечення виробничої санітарії та гігієни праці

Гігієнічна класифікація праці спрямована на гігієнічну оцінку умов та характеру праці на робочих місцях працівників та застосовуються на підприємствах, в установах, організаціях усіх форм власності у випадках, передбачених законодавством [20].

Так як роботи програмістів в обчислювальному центрі виконуються сидячи і не потребують фізичного напруження, то вони відносять до типу операторської роботи, даний тип роботи відносить до категорії 1а. Робоче місце програміста є постійним, робота проводиться у холодний та теплий періоди року тому рекомендується підтримувати оптимальні показники мікроклімату [20].

Для забезпечення оптимального рівня параметрів повітряного виробничого середовища використовуємо Державні санітарні норми 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень» [22].

Встановлені наступні оптимальні значення параметрів мікроклімату:

– у холодний період року: температура 22-24°C; відносна вологість: 40 – 60%; швидкість переміщення повітря: 0,1 м/с;

– у теплий період року: температура 23-25°C; відносна вологість: 40-60%; швидкість переміщення повітря: 0,1 – 0,2 м/с [22].

Оптимальні рівні позитивних (n+) і негативних (n-) іонів у повітрі приміщення з ПК нормовані згідно ГН 2152-80 «Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень» і становлять: $n^+ = 1500-30000$ (шт. на 1см^3); $n^- = 3000-5000$ (шт. на 1см^3), підтримка оптимального рівня легких позитивних і негативних аероіонів у повітрі на робочих місцях забезпечено за допомогою біполярних коронних аероіонізаторів, в обчислювальному центрі передбачено: устрій системи водяного опалення приміщення – для забезпечення необхідної температури повітря в холодний період року [17].

Дотримання вимог цих документів досягається оснащенням приміщень і транспортних засобів пристроями кондиціонування і вентиляції, дезодорації повітря, опалювання згідно вимог Державних будівельних норм В.2.5-67:2013 «Опалення, вентиляція та кондиціонування» [23].

Зимою значення відносної вологості повітря часто можуть нижчими за встановлені норми і становити в середньому 30–40%, що приводить до надмірного висихання слизових оболонок очей, носа, горла та до нагромадження зарядів статичної електрики, що утворюються в процесі роботи комп'ютера, в приміщенні передбачена наявність 4 зволожувачів повітря, які допомагають підняти вологість до 60% для усунення таких проблем, кількість зволожувачів зумовлена тим, що один такий апарат розраховано на приміщення площею до 15 м² [23].

Літом для підтримки нормальної температури повітря в приміщенні використовуються три кондиціонери типу «спліт-система»: два кондиціонери розташовані у протилежних частинах та один по центру обчислювального центру дозволяють підтримувати температуру повітря не вище 24°C. Кількість кондиціонерів зумовлена тим, що один такий апарат розраховано на кондиціонування приміщення площею до 18 м² [20].

Під впливом шуму знижується концентрація уваги, порушуються фізіологічні функції, з'являється стомленість у зв'язку з підвищеними енергетичними витратами і нервово-психічною напругою, погіршується мовна комутація: рівні звуку, рівні звукового тиску в октавних смугах частот, та еквівалентні рівні звуку на робочих місцях у приміщення нормуються згідно Державних санітарних правил і норм 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [21] та Державних санітарних норм 3.3.6-039-99 [22]. Оптимальний рівень шуму в приміщення забезпечується за допомогою:

- використання блоків живлення персональних комп'ютерів з вентиляторами на гумових підвісках;
- використання більш сучасного обладнання;

– розташування принтерів та різноманітного устаткування колективного користування на значній відстані від більшості робочих місць працівників;

– переведення жорсткого диска в режим сну (Standby), якщо комп'ютер не працює протягом визначеного часу [22].

Інтенсивне електромагнітне випромінювання постійно впливає на програміста в період роботи за дисплеєм, яке може стати причиною професійних захворювань, комп'ютери потрапляють за визначенням в 5-й діапазон (низькі частоти), для зниження потужності експозиційної дози рентгенівського випромінювання для роботи використовуються монітори з рідкокристалічними дисплеями [17].

Освітлення у приміщеннях з ЕОМ суміщене, при якому недостатнє за нормами природне освітлення доповнене штучним. Природне освітлення бічне, одностороннє. Вимоги до освітлення регламентовані Державними будівельними нормами В.2.5-28-2018 «Природне та штучне освітлення» [24].

Згідно з вимогами регламентованими в Державних будівельних нормах В.2.5-28-2018 в обчислювальному центрі коефіцієнт природної освітленості має становити в середньому 1,1% (1,2% – у сонячний день та 1% – у хмарну погоду), вікна в приміщенні обладнанні сонцезахисними металевими жалюзі з системою регульована для забезпечення відносної постійності природного освітлення. коефіцієнт відбиття жалюзі становить 0,7, робочі місця робітників розташовано так, щоб у поле зору робітника не потрапляли світлі поверхні світильників та вікон [24].

Проведемо розрахунок штучного освітлення в офісному приміщенні розміри якого складають: довжина 26 м, ширина 14 м, висота 5,4 м. Висота робочої поверхні $h_p = 0,7$ м, висота звисання світильника від стелі $h_z = 1,7$ м. Рівень нормованого значення освітлення $E_H = 300$ лк, тип світильника – ЛВП, колір стелі, стін, підлоги $\rho_{ст} = 70\%$, $\rho_{с} = 50\%$, $\rho_{п} = 30\%$.

Визначимо висоту підвісу світильника h над робочою поверхнею за формулою (7.1):

$$h = H - h_p - h_3, \quad (7.1)$$

де H – висота приміщення, м;

h_p – висота робочої поверхні, м;

h_3 – висота звисання світильника від стелі, м.

Розрахуємо:

$$h = 5,4 - 0,7 - 1,7 = 3 \text{ м.}$$

Визначимо значення індексу приміщення i , що характеризує співвідношення розмірів освітлювального приміщення і висоти розміщення світильників, за формулою (7.2):

$$i = \frac{A \cdot B}{h \cdot (A + B)}, \quad (7.2)$$

де A – довжина приміщення, м;

B – ширина приміщення, м;

h – висота підвісу світильника над робочою поверхнею, м.

Розрахуємо:

$$i = \frac{26 \cdot 14}{3 \cdot (26 + 14)} = 3,03.$$

Визначимо значення коефіцієнта використання світлового потоку η , створюваного світильниками вибраного типу. Вибирається в залежності від

виду джерела світла, типу обраного світильника, коефіцієнтів відбиття поверхонь приміщення та індексу приміщення:

$$\eta = 59 \%$$

Визначимо сумарний світловий потік освітлювальної установки у даному приміщенні Φ_{Σ} , за формулою (7.3):

$$\Phi_{\Sigma} = \frac{E_H \cdot A \cdot B \cdot k_z \cdot z}{\eta}, \quad (7.3)$$

де E_H – рівень нормованого загального освітлення, лк;

A – довжина приміщення, м;

B – ширина приміщення, м;

k_z – коефіцієнт запасу (для люмінесцентних ламп $k_z = 1,4 \dots 1,7$);

z – коефіцієнт нерівномірності (мінімальної) освітленості (відношення середньої освітленості до мінімальної освітленості), як правило дорівнює (для люмінесцентних ламп $z = 1,1$);

η – коефіцієнт використання світлового потоку.

Розрахуємо:

$$\Phi_z = \frac{300 \cdot 26 \cdot 14 \cdot 1,5 \cdot 1,1}{0,59} = 305389,83 \text{ лм.}$$

Таким чином, сумарний світловий потік освітлювальної установки у даному приміщенні Φ_{Σ} дорівнює 305389,83 лм.

В приміщенні обчислювального центру одночасно працюють 6 чоловік, кожний робітник має свій:

- обчислювальний пристрій, пристрою вводу (клавіатура, миш та інше), виводу (дисплей монітору, принтер, звукова колонка та інше) та набір периферійних пристроїв (принтер, сканер та інше);

- стіл;

- два ортопедичні стільці;

- тумбу [24].

Також кожний робітник може скористатися шафою для одягу, шафою з приладами, двома тумбами призначеними для використання всіма робітниками [24].

Робоче місце працівника має площу 6 м^2 , а об'єм – по 20 м^3 .

Приміщення має природне та штучне освітлення, робочі місця розташовано так, щоб природне світло падало збоку, робочі місця та взаємне розташування всіх його елементів відповідати антропометричним, фізичним і психологічним вимогам [21].

Також всі робочі місця організовані за такими правилами організації робочого місця:

- всім робітникам надані ергономічні та зручних миші і клавіатури;

- висота столу забезпечує розташування клавіатури від підлоги – 65-75 см, що позитивно впливає на працездатність робітника;

- екрани монітору знаходяться на оптимальній відстані 700 мм від працівника;

- кожний стілець має підлокітник [21].

7.4 Заходи з пожежної безпеки

Великою небезпекою для обчислювального центру є пожежа: обчислювальний центр наповнений різноманітними електричними приладами, а пожежа основний фактор для їх знищення, що може призвести до великих матеріальних втрат [25].

Для забезпечення пожежної безпеки проводився пожежна профілактика, яка включає в себе комплекс організаційних і технічних заходів, спрямованих на забезпечення безпеки людей, на запобігання пожежі, обмеження її поширення, а також на створення умов для успішного гасіння пожежі [26].

Обчислювальний центр відноситься до категорії «П-Па» згідно ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою», згідно Державних будівельних норм В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги», а клас можливої пожежі визначається, як «А» [25]-[26].

Для ліквідації пожежі у початковій стадії її розвитку силами персоналу об'єктів застосовуються первинні засоби пожежогасіння, до них відносяться: вогнегасники, пожежний інвентар (покривала з негорючого теплоізоляційного полотна, ящики з піском, пожежні відра, совкові лопати, ломи, сокири тощо), системи автоматичного пожежогасіння [25].

Під час прийняття на роботу і в процесі праці усі працівники обчислювального центру повинні проходити протипожежний інструктаж, перевірку знань з питань пожежної безпеки, евакуаційні виходи і проходи утримуються постійно вільними та легкодоступними [26].

7.5 Заходи забезпечення безпеки у надзвичайних ситуаціях

Закон «Про цивільну оборону України» визначає надзвичайну ситуацію як порушення нормальних умов життя та діяльності людей на об'єкті чи території, спричинених аварією, катастрофою, стихійним лихом, епідемією, епізоотією, великою пожежею, використання засобів ураження, що призвели чи можуть призвести до людських чи матеріальних втрат, зокрема якщо вас захопили злочинці як заручника і ваше життя та здоров'я перебувають у небезпеці, варто пам'ятати таке:

– намагайтесь бути розважливими, спокійними, якомога миролюбнішими, не піддавайте себе зайвому ризику;

– не посилюйте агресивність злочинців непокрою, лайкою, зайвим опором, оскільки перевага сил у таких випадках складається не на вашу користь;

– намагайтесь віднайти найбезпечніше місце у приміщенні [27].

ВИСНОВКИ

Виконано аналіз проблеми прокату вживаних речей. Розглянуто задачу визначення затребуваності прокату вживаних речей як задачу прогнозування, проаналізовано методи, які можуть потенційно на цій основі до неї застосовуватися. Розглянуто рекурентні нейронні мережі та детально принципи і особливості роботи архітектури GRU.

Обґрунтовано виконання розробки на основі комбінації мов програмування C# та Python. Для цього визначено структуру програмної системи, що функціонально та за рахунок з'єднання з базою даних дозволяє створювати міжмовну програмну систему.

Описано функціональні вимоги до програмного забезпечення, виділено основні сценарії використання програмної системи різними групами користувачів. До користувачів належать менеджери та орендарі. Забезпечено створення бази даних, для чого було виконано проектування бази даних, результат якого – побудована схема бази даних та визначені SQL-запити на створення таблиць бази даних.

Реалізовано програмне забезпечення у вигляді програмної системи, що складається з Windows-застосунку з графічним інтерфейсом та скрипту, який запускається для виконання прогнозування затребуваності прокату вживаних речей з заданим періодом у місяць.

Розроблена програма дозволяє виконувати пошук речей для прокату, перегляд речей за категоріями, найбільш затребуваних речей, замовлення і оформлення прокату речі, повернення і зняття речі з прокату, внесення даних про прокат і ремонт речі, внесення даних орендаря, фіксацію дефектів речі, обчислення фактичної вартості прокату речі, перегляд власних прокатів речей.

Наукова новизна роботи визначається запропонованим методом визначення затребуваності прокату вживаних речей, що ґрунтується на моделі прогнозування відповідних значень для різних речей заданої категорії

на майбутній період за послідовністю попередніх значень на основі використання моделі архітектури GRU, що дозволяє приймати рішення щодо необхідної кількості речей в прокаті сервісу на майбутній період.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гунин, М.А. Циркуляционная экономика [Электрон. ресурс] : новая парадигма ведения бизнеса / М.А. Гунин. – Режим доступа : <https://www.hse.ru/edu/vkr/124956649>
2. Андроник, В. Циркулярная экономика или как можно использовать повторно 90 % ресурсов [Электрон. ресурс] / В. Андроник. – Режим доступа : <https://economics.segodnya.ua/economics/business/cirkulyarnaya-ekonomika-ili-kak-mozhno-ispolzovat-povtorno-90-resursov-1251564.html>
3. Чулок, А. Что такое шеринг-экономика? [Электрон. ресурс] / А. Чулок. – Режим доступа : <https://postnauka.ru/faq/82383>
4. Захваткина, А. Есть, делиться и дарить [Электрон. ресурс] : что такое шеринг-экономика и к чему она приведет / А. Захваткина. – Режим доступа : <https://www.asi.org.ru/news/2019/04/24/est-delitsya-i-darit-cto-takoe-shering-ekonomika-i-k-chemu-ona-privedet/>
5. Осман, Я. Надо делиться [Электрон. ресурс] : как шеринг-экономика превращает потребителя в предпринимателя / Я. Осман. – Режим доступа : <https://www.forbes.ru/karera-i-svoy-biznes/360815-nado-delitsya-kak-shering-ekonomika-prevrashchaet-potrebitelya-v>
6. Распопова, О. Дай поносить [Электрон. ресурс] : как меняются сервисы проката одежды / О. Распопова. – Режим доступа : <https://thecity.m24.ru/articles/1408>
7. Ковтун, Н. Аренда вещей теснит торговлю [Электрон. ресурс] / Н. Ковтун. – Режим доступа : <https://marketmedia.ru/media-content/arenda-veshchey-tesnit-torgovlyu/>
8. Шеринг [Электрон. ресурс] : сервисы совместного пользования в Украине. – Режим доступа : <https://senior.ua/articles/shering-servisy-sovmestnogo-polzovaniya-v-ukraine>
9. Tyagi, N. Introduction to Time Series Analysis [Electronic resource] :

Time-Series Forecasting Machine learning Methods & Models / Neelam Tyagi. – Access mode : <https://medium.com/analytics-steps/introduction-to-time-series-analysis-time-series-forecasting-machine-learning-methods-models-ecaa76a7b0e3>

10. Recurrent Neural Networks [Electronic resource]. – Access mode : <https://www.ibm.com/cloud/learn/recurrent-neural-networks>

11. Amidi, A. Recurrent Neural Networks cheatsheet [Electronic resource] / Afshine Amidi, Shervine Amidi. – Access mode : <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>

12. Gated Recurrent Units (GRU) – Dive into Deep Learning [Electronic resource]. – Access mode : https://d2l.ai/chapter_recurrent-modern/gru.html

13. Николенко, С. Глубокое обучение [Текст] / С. Николенко, А. Кадури, Е. Архангельская. – СПб. : Питер, 2018. – 480 с. : ил.

14. Коэльё, Л. П. Построение систем машинного обучения на языке Python [Текст] / Л. П. Коэльё, В. Ричерт. – М. : ДМК Пресс, 2015. – 302 с.

15. Скит, Дж. С# для профессионалов [Текст] : тонкости программирования/ Дж. Скит. – М. : Вильямс, 2014. – 608 с.

16. Методичні вказівки по економічному обґрунтуванню дипломних проектів для студентів спеціальностей 8.05010301 «Програмне забезпечення автоматизованих систем» та 8.05010201 «Інформаційні технології проектування» для усіх форм навчання [Текст] / В.В. Остапенко, Є.М. Сердюк. – Запоріжжя : ЗНТУ, 2015. – 22 с.

17. Жидецький, В.Ц. Основи охорони праці. [Текст] : Підручник / В. Ц. Жидецький. – 5-те вид., доп. – К. : Знання, 2014. – 373 с.

18. Правила улаштування електроустановок [Текст] : Нормативне виробничо-практичне видання : вид. 5-те, перероблене й доповнене : затв. М-вом енергетики та вугільної промисловості України 20.06.14 : введення в дію з 20.11.14. – Х. : Міненерговугілля України, 2014. – 793 с.

19. Правила улаштування електроустановок [Текст]. – К. : ОЕП "ГРІФЕ", 2006. – 77 с.

20. Охорона праці [Текст] : Навч. посібник / З. М. Яремко, С. В. Тимошук, О. І. Третяк, Р. М. Ковтун ; за ред. проф. З. М. Яремка. – Львів : Видавничий центр ЛНУ імені Івана Франка, 2010. – 374 с.
21. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. НПАОП 0.00-7.15-18. Державні санітарні правила і норми [Текст]. – Харків : Форт, 2018. – 48 с.
22. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень [Текст]. – Харків : Форт, 2000 – 24 с.
23. ДБН В.2.5-67:2013 Опалення, вентиляція та кондиціонування [Текст]. – Київ : Мінрегіон України, 2013. – 139 с.
24. ДБН В.2.5-28:2018. Природне та штучне освітлення. Посібник [Текст] : проблеми природного і штучного освітлення. – К. : Етін, 2019. – 180 с.
25. Правила пожежної безпеки в Україні [Текст]. – Харків : Форт, 2015. – 124 с.
26. ДБН В.1.1-7:2016 Пожежна безпека об'єктів будівництва. Загальні вимоги [Текст]. – Київ : Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України, 2017. – 39 с.
27. Стеблюк, М.І. Цивільна оборона та цивільний захист [Текст] : Підручник / М.І. Стеблюк. – К. : Знання-Прес, 2007. – 487 с.

ДОДАТОК А
Технічне завдання

Вступ

Розроблюване програмне забезпечення має підтримувати роботу сервісу прокату вживаних речей. Для цього з одного боку планується забезпечити облік даного процесу, а з іншого – забезпечити інструмент, який може використовуватися в якості підґрунтя прийняття рішень, – інструмент визначення затребуваності прокату вживаних речей на майбутній період.

A.1 Підстави для розробки

Розробка виконується на основі завдання на дипломну кваліфікаційну роботу магістра, яке було визначене за темою “Дослідження та програмна реалізація методів визначення затребуваності прокату вживаних речей” і затверджене наказом № 301 від 3 листопада 2020 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програмне забезпечення, що розробляється, має забезпечувати підтримку роботи менеджерів сервісу щодо внесення можливих варіантів прокату речей, фіксації прокатів речей, отримання статистики за виконаними прокатами, а для орендарів має забезпечувати підтримку замовлення прокатів речей та перегляду і пошуку для цього можливих варіантів.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

У відповідності з описаною логікою за результатами аналізу програмних аналогів та проблеми загалом, в роботі має бути забезпечено

реалізацію функціональних характеристик програми, які загалом включають наступний набір:

- пошук речей для прокату;
- перегляд речей за категоріями;
- перегляд найбільш затребуваних речей;
- замовлення прокату речі;
- оформлення прокату речі;
- повернення речі з прокату;
- внесення даних про ремонт речі;
- внесення даних орендаря;
- авторизація користувачів;
- реєстрація користувачів;
- зняття речі з прокату;
- фіксація дефектів речі;
- внесення даних про вартість прокату речі;
- обчислення фактичної вартості прокату речі;
- внесення даних про прокат речей;
- перегляд власних прокатів речей;
- оформлення статистики прокату речей;
- прогнозування затребуваності прокату речей за категоріями.

А.3.2 Вимоги до надійності

Дані, які вводяться користувачем, повинні в формах програми перевірятися перед виконанням запитів до бази даних, тобто перед збереженням даних. Мають оброблятися дані, які мають відповідний параметру тип. Програма повинна контролювати даний процес і виводити повідомлення у протилежних випадках.

А.3.3 Умови експлуатації

Для експлуатації програми повинен бути виділений сервер або персональний комп'ютер з наявним мережним з'єднанням, який буде відігравати роль сервера. Доступ до сервера може виконуватися зі звичайних комп'ютерів. На всіх має бути встановлено операційну систему Windows. Окрім того вимагається наявність встановленого фреймворку .NET Framework версії 4.7 або пізнішої.

А.3.4 Вимоги до складу та параметрів технічних засобів

Для забезпечення коректної роботи програми потрібно мати наступний набір технічних засобів:

- 32-розрядний або 64-розрядний процесор з частотою не менше 1 ГГц;
- 8 Гб (x32) або 12 Гб (x64) оперативної пам'яті;
- 100 Гб вільного місця на жорсткому диску з обсягом жорсткого диску не менше 500 Гб.

Необхідно враховувати, що вказані параметри мають забезпечувати повноцінну роботу невеликого сервера, тобто за розширення кількості одночасних запитів більше 100 вимоги відповідно мають збільшуватися.

А.3.5 Вимоги до транспортування та збереження

Транспортування програми повинно виконуватися шляхом мережного підключення. Тільки за певної необхідності програма може зберігатися на носії, за допомогою якого виконується транспортування програми.

A.3.6 Вимоги до програмної документації

Не передбачається розробка повного набору програмної документації. У тексті програми наводиться фрагмент програми для її ідентифікації. У технічному завданні визначаються основні початкові параметри розробки.

A.3.7 Вимоги до маркування та упакування

Позначка «UsedItemsService» має бути вказана як на екранних формах програмного забезпечення, так і в назві бінарних файлів. Окрім того за можливості позначка має наноситися на фізичний носій.

A.4 Порядок контролю та приймання

Контроль виконання дипломної кваліфікаційної роботи магістра реалізується на основі календарного плану, визначеного в завданні на роботу. Для кожного етапу роботи зазначено строки її виконання. За результатом тринадцятого тижня має бути виконано, підготовлено до захисту та захищено роботу, тоді виконання розробки можна вважати успішним.

ДОДАТОК Б
Текст програми

Б.1 Текст файла CategoryDemand.cs

```

using System;
using System.Collections.Generic;
using System.Text;

namespace UsedItemsDomain
{
    public class CategoryDemand : IUsedItems
    {
        private int _id;
        private DateTime _periodstart;
        private DateTime _periodend;
        private int _dnumber;
        private int _type;
        private int _categoryid;

        public CategoryDemand ()
        {
            this.Id = 0;
            this.Periodstart = "";
            this.Periodend = "";
            this.Dnumber = 0;
            this.Type = 0;
            this.Catfull = new Category();
        }

        public int Id
        {
            get { return _id; }
            set { _id = value; }
        }

        public DateTime Periodstart
        {
            get { return _periodstart; }
            set { _periodstart = value; }
        }

        public DateTime Periodend

```

```

    {
        get { return _periodend; }
        set { _periodend = value; }
    }

    public int Dnumber
    {
        get { return _dnumber; }
        set { _dnumber = value; }
    }
    public int Type
    {
        get { return _type; }
        set { _type = value; }
    }

    public int Categoryid
    {
        get { return _categoryid; }
        set { _categoryid = value; }
    }

    public void Save()
    {
        uidb = new UsedItemsDb();
        uidb.open();
        if (uidb.existsCategoryDemand(this.Id))
        {
            uidb.updateCategoryDemand(this.Id,
this.Dnumber, this.Type);
        }
        else
        {
            int cid = this.Categoryid;
            int cdn =
uidb.addCategoryDemand(this.Periodstart, this.Periodend, this.Dnumber,
this.Type, cid);
            this.id = cdn;
        }
        uidb.close();
    }

```

```

        public void Read()
        {
            uidb = new UsedItemsDb();
            uidb.open();
            this.Periodstart = uidb.getCatDemPeriodstart
(this.Id);

            this.Periodend = uidb.getCatDemPeriodend (this.Id);
            this.Dnumber = uidb.getCatDemDnumber (this.Id);
            this.Type = uidb.getCatDemType (this.Id);
            this.Categoryid = this.Id;
            uidb.close();
        }
    }
}

```

Б.2 Текст файла Category.cs

```

using System;
using System.Collections.Generic;
using System.Text;

namespace UsedItemsDomain
{
    public class Category : IUsedItems
    {
        private int _id;
        private string _name;
        private List <string> _description;
        private List <CategoryDemand> _demands;

        public Category ()
        {
            this.Id = 0;
            this.Name = "";
            this.Description = new List <string> ();
        }

        public int Id

```

```

{
    get { return _id; }
    set { _id = value; }
}

public string Name
{
    get { return _name; }
    set { _name = value; }
}

public List <string> Description
{
    get { return _description; }
    set { _description = value; }
}

public List <CategoryDemand> Demands
{
    get { return _demands; }
    set { _demands = value; }
}

public void Save()
{
    uidb = new UsedItemsDb();
    uidb.open();
    if (uidb.existsCategory(this.Id))
    {
        uidb.updateCategory(this.Id, this.Name,
this.Description);

        foreach (int cd in Demands)
        {
            cd.Save();
        }
    }
    else
    {
        int id = uidb.addCategory(this.Name,
this.Description);

        this.id = id;
    }
    uidb.close();
}

```

```

    }

    public void Read()
    {
        uidb = new UsedItemsDb();
        uidb.open();
        this.Name = uidb.getCategoryName (this.Id);
        this.Description = uidb.getCategoryDescription
(this.Id);

        List<int> CDIds = uidb.getCDemandsIds(this.Id);
        for (int cdn in CDIds)
        {
            CategoryDemand cd = new CategoryDemand();
            cd.Id = cdn;
            cd.Read();
            this.Demands.add(cd);
        }
        uidb.close();
    }
}
}

```

Б.3 Текст файла Lease.cs

```

using System;
using System.Collections.Generic;
using System.Text;

namespace UsedItemsDomain
{
    public class Lease : IUsedItems
    {
        private int _id;
        private ItemCopy _itemcopyfull;
        private User _userfull;
        private Leaser _leaserfull;
        private DateTime _starttime;
        private DateTime _endtime;
        private List <string> _description;
    }
}

```

```

private float _price;
private float _getsum;
private int _state;

public Lease ()
{
    this.Id = 0;
    this.Itemcopyfull = new Itemcopy();
    this.Userfull = new User ();
    this.Leaserfull = new Leaser();
    this.Starttime = "";
    this.Endtime = "";
    this.Description = new List<string> ();
    this.Price = 0.0;
    this.Getsum = 0.0;
    this.State = 0;
}

public int Id
{
    get { return _id; }
    set { _id = value; }
}

public ItemCopy Itemcopyfull
{
    get { return _itemcopyfull; }
    set { _itemcopyfull = value; }
}

public int Itemcopyid
{
    get { return Itemcopyfull.Id; }
}

public User Userfull
{
    get { return _userfull; }
    set { _userfull = value; }
}

public int Userid
{
    get { return Userfull.Id; }
}

```

```

    }

    public Leaser Leaserfull
    {
        get { return _leaserfull; }
        set { _leaserfull = value; }
    }

    public int Leaserid
    {
        get { return Leaserfull.Id; }
    }

    public DateTime Starttime
    {
        get { return _starttime; }
        set { _starttime = value; }
    }

    public DateTime Endtime
    {
        get { return _endtime; }
        set { _endtime = value; }
    }

    public List <string> Description
    {
        get { return _description; }
        set { _description = value; }
    }

    public float Price
    {
        get { return _price; }
        set { _price = value; }
    }

    public float Getsum
    {
        get { return _getsum; }
        set { _getsum = value; }
    }

```

```

public int State
{
    get { return _state; }
    set { _state = value; }
}

public void Calc ()
{
    int iid = this.Itemcopyfull.Id;
    Item it = new Item();
    uidb = new UsedItemsDb();
    uidb.open();
    it.Id = uidb.getItemcopyItemId(iid);
    it.Read();
    float sum = it.getPrice(this.Starttime,
this.Endtime);

    this.Price = sum;
}

public void Save()
{
    uidb = new UsedItemsDb();
    uidb.open();
    if (uidb.existsLease(this.Id))
        uidb.updateLease(this.Id, this.Itemcopyid,
this.Userid, this.Leaserid, this.Starttime, this.Endtime,
this.Description, this.Price, this.Getsum, this.State);
    else
    {
        int id = uidb.addLease(this.Itemcopyid,
this.Userid, this.Leaserid, this.Starttime, this.Endtime,
this.Description, this.Price, this.Getsum, this.State);
        this.id = id;
    }
    uidb.close();
}

public void Read()
{
    uidb = new UsedItemsDb();
    uidb.open();

```

```

        int iid = this.State =
uidb.getLeaseItemcopy(this.Id);
        this.Itemcopyfull = new Itemcopy();
        this.Itemcopyfull.id = iid;
        this.Itemcopyfull.Read();
        int uid = uidb.getLeaseUser(this.Id);
        this.Userfull = new User();
        this.Userfull.id = uid;
        this.Userfull.Read();
        int lid = uidb.getLeaseLeaser(this.Id);
        this.Leaserfull = new Leaser();
        this.Leaserfull.id = lid;
        this.Leaserfull.Read();
        this.Starttime = uidb.getLeaseStarttime (this.Id);
        this.Endtime = uidb.getLeaseEndtime (this.Id);
        this.Description= uidb.getLeaseDescription
(this.Id);

        this.Price= uidb.getLeasePrice (this.Id);
        this.Getsum= uidb.getLeaseGetsum (this.Id);
        this.State = uidb.getLeaseState (this.Id);
        uidb.close();
    }

}
}

```

Б.4 Текст файлу Item.cs

```

using System;
using System.Collections.Generic;
using System.Text;

namespace UsedItemsDomain
{
    public class Item : IUsedItems
    {
        private int _id;
        private string _name;
        private int _categoryid;
    }
}

```

```

private List <string> _description;
private List <ItemCopy> _itemcopies;
private List <LeasePrice> _leaseprices;

public Item ()
{
    this.Id = 0;
    this.Categoryid = 0;
    this.Name = "";
    this.Description = new List<string> ();
    this.Itemcopies = new List<ItemCopy> ();
    this.Leaseprices = new List<LeasePrice> ();
}

public int Id
{
    get { return _id; }
    set { _id = value; }
}

public int Categoryid
{
    get { return _categoryid; }
    set { _categoryid = value; }
}

public string Name
{
    get { return _name; }
    set { _name = value; }
}

public List <string> Description
{
    get { return _description; }
    set { _description = value; }
}

public List <ItemCopy> Itemcopies
{
    get { return _itemcopies; }
    set { _itemcopies = value; }
}

```

```

public List <LeasePrice> Leaseprices
{
    get { return _leaseprices; }
    set { _leaseprices = value; }
}

public List <ItemCopy> getActiveItemcopies ()
{
    List <ItemCopy> lip = new List <ItemCopy>();
    foreach (ItemCopy cp in Itemcopies)
    {
        if (cp.State == 1)
            lip.add(cp);
    }
    return lip;
}

public List <ItemCopy> getRepairItemcopies ()
{
    List <ItemCopy> rip = new List <ItemCopy>();
    foreach (ItemCopy ic in Itemcopies)
    {
        if (ic.State == 2)
            rip.add(ic);
    }
    return rip;
}

public List <ItemCopy> getLostItemcopies ()
{
    List <ItemCopy> loip = new List <ItemCopy>();
    foreach (ItemCopy ic in Itemcopies)
    {
        if (ic.State == 3)
            loip.add(ic);
    }
    return loip;
}

public int getFreeItemcopy ()
{
    foreach (ItemCopy ic in Itemcopies)
    {

```

```

        if (ic.State == 1)
            return ic.Id;
    }
    return -1;
}

public int getUnactiveItemcopiesCount ()
{
    int c = 0;
    foreach (ItemCopy ic in Itemcopies)
    {
        if (ic.State >= 2 && ic.State <=4)
            c = c + 1;
    }
    return c;
}

public int getActiveItemcopiesCount ()
{
    int c = 0;
    foreach (ItemCopy ic in Itemcopies)
    {
        if (ic.State == 1)
            c = c + 1;
    }
    return c;
}

public float getActiveItemcopiesPercent ()
{
    int a = this.getActiveItemcopiesCount(), b = this.
getUnactiveItemcopiesCount();
    return a / (a + b) * 100;
}

public float getPrice (DateTime starttime, DateTime
endtime)
{
    TimeSpan interval = endtime - starttime;
    List<LeasePrice> slp =
Leaseprices.OrderBy(o=>o.Minduration).ToList();
    bool f = false;
    float res = 0;

```

```

        foreach (LeasePrice p in slp)
        {
            int t;
            if (p.period == 1)
                t = interval.TotalDays;
            else t = interval.TotalHours;
            if (t < p.Minduration)
                return res * t;
            else res = t.Price;
        }
    }

    public void Save()
    {
        uidb = new UsedItemsDb();
        uidb.open();
        if (uidb.existsItem(this.Id))
        {
            uidb.updateItem(this.Id, this.Name,
this.Description, this.Categoryid);
            foreach (LeasePrice lp in Leaseprices)
            {
                lp.Save();
            }
            foreach (ItemCopy ic in Itemcopies)
            {
                ic.Save();
            }
        }
        else
        {
            int aid = uidb.addItem (this.Name,
this.Description, this.Categoryid);
            this.id = aid;
        }
        uidb.close();
    }

    public void Read()
    {
        uidb = new UsedItemsDb();
        uidb.open();
        this.Name = uidb.getItemName (this.Id);
    }

```

```

        this.Categoryid = uidb.getItemCatId (this.Id);
        this.Description = uidb.getItemDescription
(this.Id);

        List<int> ICL = uidb.getItemCopiesForItem(this.Id);
        for (int l in ICL)
        {
            ItemCopy ic = new ItemCopy ();
            ic.Id = l;
            ic.Read();
            if (ic.State != 5)
                this.Itemcopies.add(ic);
        }
        List<int> LPI =
uidb.getLeasePricesForItem(this.Id);
        for (int j in LPI)
        {
            LeasePrice lp = new LeasePrice ();
            lp.Id = j;
            lp.Read();
            this.Leaseprices.add(lp);
        }
        uidb.close();
    }

}

```

Б.5 Текст файла ItemList.cs

```

using System;
using System.Collections.Generic;
using System.Text;

namespace UsedItemsDomain
{
    public class ItemList : IUsedItems
    {
        private List<Item> _items;
    }
}

```

```

public ItemList ()
{
    this.Items = new List<Item>();
}

public List<Item> Items
{
    get { return _items; }
    set { _items = value; }
}

public void AddItem (Item it)
{
    Items.add(it);
}

public List <Item> Search (string name)
{
    List <Item> searchitems = new List <Item>();
    foreach (Item i in Items)
    {
        if (i.Name.Contains (name))
            searchitems.add(i);
    }
    return searchitems;
}

public List <Item> getCategoryItems (int cat)
{
    List <Item> catitems = new List <Item>();
    foreach (Item i in Items)
    {
        if (i.Categoryid == cat)
            catitems.add(i);
    }
    return catitems;
}

}
}

```

Б.6 Текст файла LeaseList.cs

```

using System;
using System.Globalization;
using System.Collections.Generic;
using System.Text;

namespace UsedItemsDomain
{
    public class LeaseList : IUsedItems
    {
        private List<Lease> _leases;

        public LeaseList ()
        {
            this.Leases = new List<Lease>();
        }

        public List<Lease> Leases
        {
            get { return _leases; }
            set { _leases = value; }
        }

        public void Order (int itemnum, List <string> desc,
float sum, Datetime st, Datetime et, Leaser ls, User us)
        {
            Lease ls = new Lease();
            ls.Description = desc;
            ls.Getsum = sum;
            ls.Starttime = st;
            ls.Endtime = et;
            ls.Itemcopyfull = new Itemcopy();
            ls.Userfull = us;
            ls.Leaserfull = ls;
            ls.State = 1;
            ls.Calc();
            ls.Save();
            this.Leases.add(ls);
        }
    }
}

```

```

        public int Lease (Item it, List <string> desc, float
sum, Datetime st, Datetime et, Leaser ls, User us)
        {
            Lease ls = new Lease();
            ls.Description = desc;
            ls.Getsum = sum;
            ls.Starttime = st;
            ls.Endtime = et;
            int id = it.getFreeItemcopy();
            if (id == -1)
            {
                return -1;
            }
            else
            {
                ls.Itemcopyfull.Id = id;
                ls.Itemcopyfull.Read();
                ls.Userfull = us;
                ls.Leaserfull = ls;
                ls.State = 2;
                ls.Calc();
                ls.Save();
                this.Leases.add(ls);
                return 0;
            }
        }
    }

```

```

        public void LeaseCopy (int ic, List <string> desc,
float sum, Datetime st, Datetime et, Leaser ls, User us)
        {
            Lease ls = new Lease();
            ls.Description = desc;
            ls.Getsum = sum;
            ls.Starttime = st;
            ls.Endtime = et;
            ls.Itemcopyfull.id = ic;
            ls.Itemcopyfull.Read();
            ls.Userfull = us;
            ls.Leaserfull = ls;
            ls.State = 2;
            ls.Calc();
            ls.Save();
            this.Leases.add(ls);
        }
    }

```

```

    }

    public void LeaseNow (int ic, List <string> desc, float
sum, Datetime et, Leaser ls, User us)
    {
        Lease ls = new Lease();
        ls.Description = desc;
        ls.Getsum = sum;
        ls.Starttime = DateTime.Now;
        ls.Endtime = et;
        ls.Itemcopyfull.id = ic;
        ls.Itemcopyfull.Read();
        ls.Userfull = us;
        ls.Leaserfull = ls;
        ls.State = 2;
        ls.Calc();
        int li = ls.Save();
        ls.Id = li;
        this.Leases.add(ls);
    }

    public void ReturnNow (int lnum, List <string> desc,
float sum)
    {
        foreach (Lease ls in leases)
        {
            if (ls.Id == lnum)
            {
                this.Leases.remove(ls);
                ls.Description = desc;
                ls.Getsum = sum;
                ls.Endtime = DateTime.Now;
                ls.State = 3;
                ls.Calc();
                ls.Save();
                this.Leases.add(ls);
            }
        }

        public void Return (int lnum, List <string> desc, float
sum, DateTime endper)
        {
            foreach (Lease ls in leases)

```

```

        {
            if (ls.Id == lnum)
            {
                this.Leases.remove(ls);
                ls.Description = desc;
                ls.Getsum = sum;
                ls.Endtime = endper;
                ls.State = 3;
                ls.Calc();
                ls.Save();
                this.Leases.add(ls);
            }
        }

    public List <Lease> getUserHistory (int userid)
    {
        List <Lease> histleases = new List <Lease>();
        foreach (Lease l in Leases)
        {
            if (l.Userid == userid)
                histleases.add(l);
        }
        return histleases;
    }

    public List <Lease> getLeaserHistory (int lid)
    {
        List <Lease> histleases = new List <Lease>();
        foreach (Lease l in Leases)
        {
            if (l.Leaserid == lid)
                histleases.add(l);
        }
        return histleases;
    }

    public List <Item> getLeasesForPeriod (DateTime begdt,
    DateTime enddt)
    {
        List <Lease> periodleases = new List <Lease>();
        foreach (Lease pl in Leases)
        {

```

```

        if (pl.Starttime>=begdt && pl.Endtime<=
enddt)
            periodleases.add(pl);
        }
        return periodleases;
    }

    public int getLeasesForPeriodCount (DateTime begdt,
DateTime enddt)
    {
        int c = 0;
        foreach (Lease pl in Leases)
        {
            if (pl.Starttime>=begdt && pl.Endtime<=
enddt)
                c = c + 1;
        }
        return c;
    }
}

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота магістра

**ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ВИЗНАЧЕННЯ ЗАТРЕБУВАНOSTІ ПРОКАТУ
ВЖИВАНИХ РЕЧЕЙ**

Виконала
студентка групи КНТ-219м

А.Д. Сіденко

Керівник
к.т.н., доцент

В.М. Льовкін

2020

Рисунок В.1 – Слайд 1

2

Об'єкт дослідження – процес прокату вживаних речей.

Предмет дослідження – методи визначення затребуваності прокату вживаних речей.

Мета роботи – розроблення програмного забезпечення підтримки прокату вживаних речей для забезпечення обґрунтованості керування речами в прокаті.

2

Рисунок В.2 – Слайд 2

Завдання

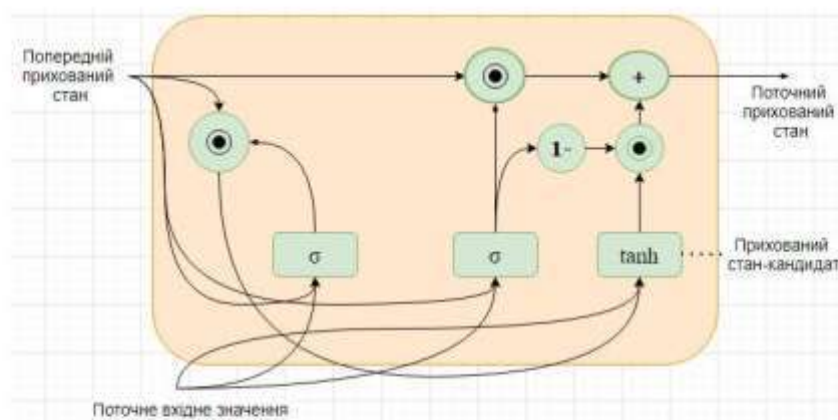
- ✓ дослідження методів визначення затребуваності прокату вживаних речей;
- ✓ розробка технічного завдання;
- ✓ вибір засобів розробки;
- ✓ проєктування структури програмного забезпечення;
- ✓ розробка програмного забезпечення підтримки прокату вживаних речей;
- ✓ тестування та відлагодження програми;
- ✓ економічне обґрунтування доцільності розробки.

3

Рисунок В.3 – Слайд 3

4

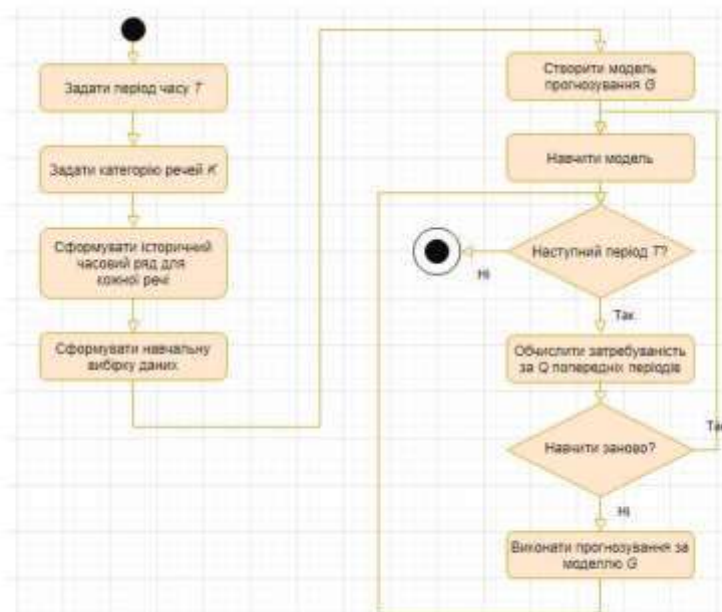
Структура комірки GRU



4

Рисунок В.4 – Слайд 4

Метод визначення затребуваності прокату вживаних речей



5

Рисунок В.5 – Слайд 5

Вибір мови програмування

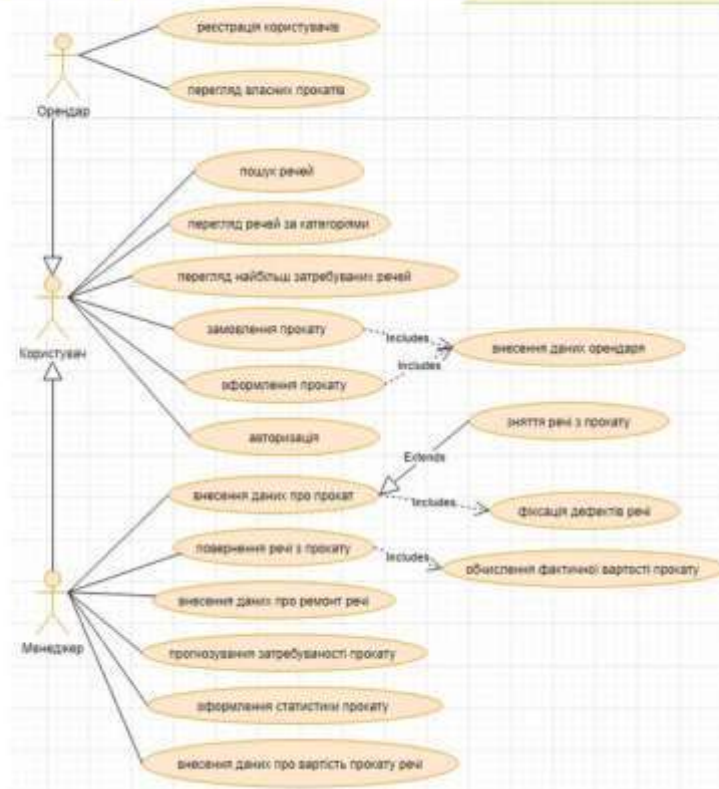
Критерій	C#	Python
Чистота об'єктно-орієнтованого підходу	більша	менша
Час виконання	менший	більший
Компонентно-орієнтоване програмування	+	-
Кросплатформна розробка	+	+
Створення Windows-застосунків з графічним інтерфейсом	+	можливе
Бібліотечні засоби роботи з нейронними мережами архітектури GRU	-	+

6

Рисунок В.6 – Слайд 6

Діаграма прецедентів роботи з системою

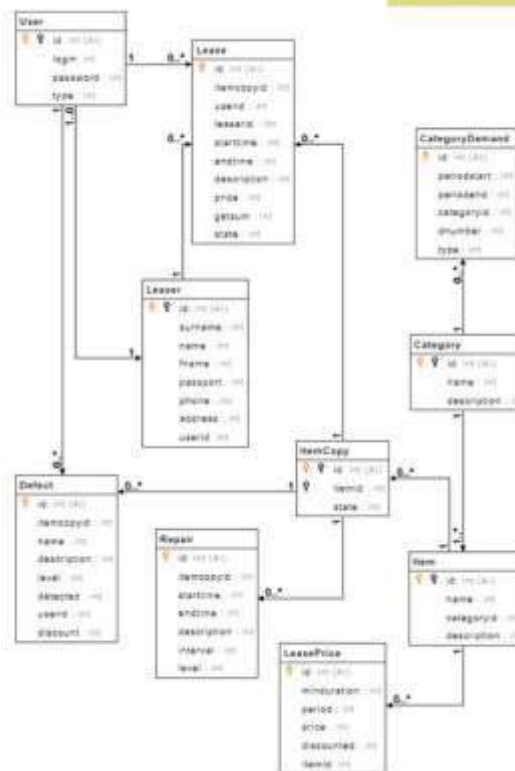
7



7

Рисунок В.7 – Слайд 7

Схема бази даних



8

Рисунок В.8 – Слайд 8

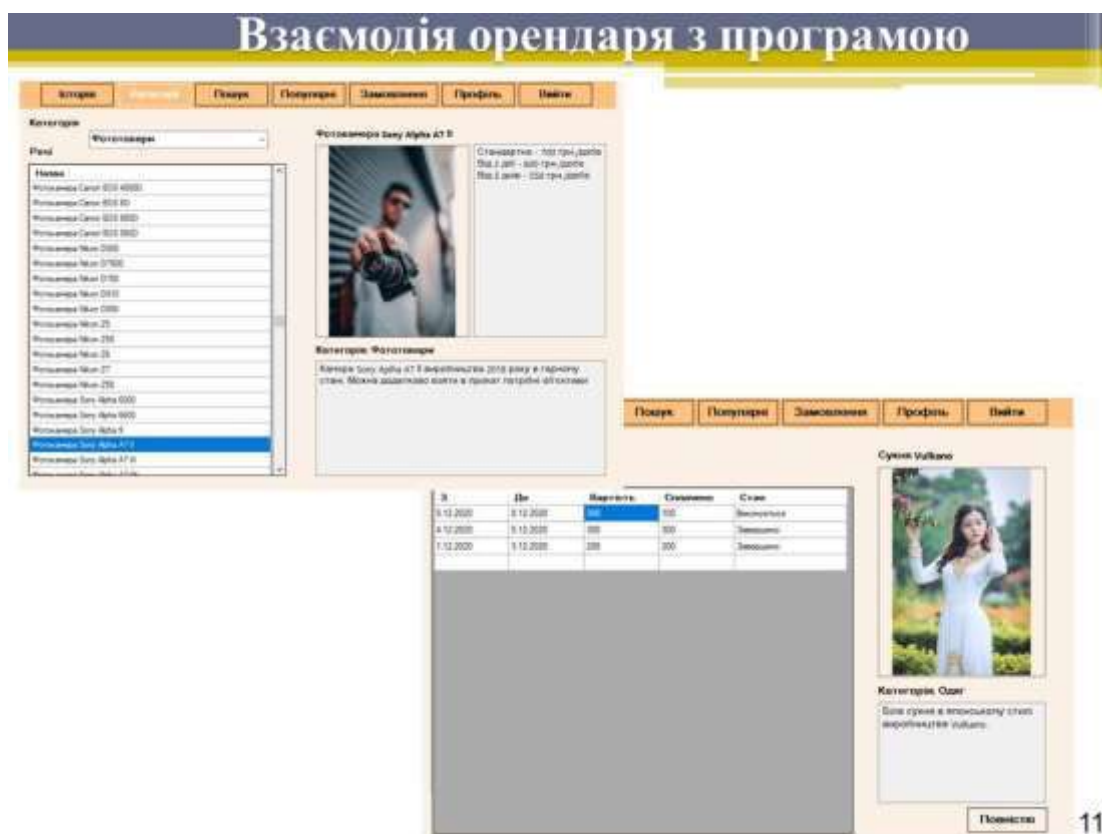


Рисунок В.11 – Слайд 11

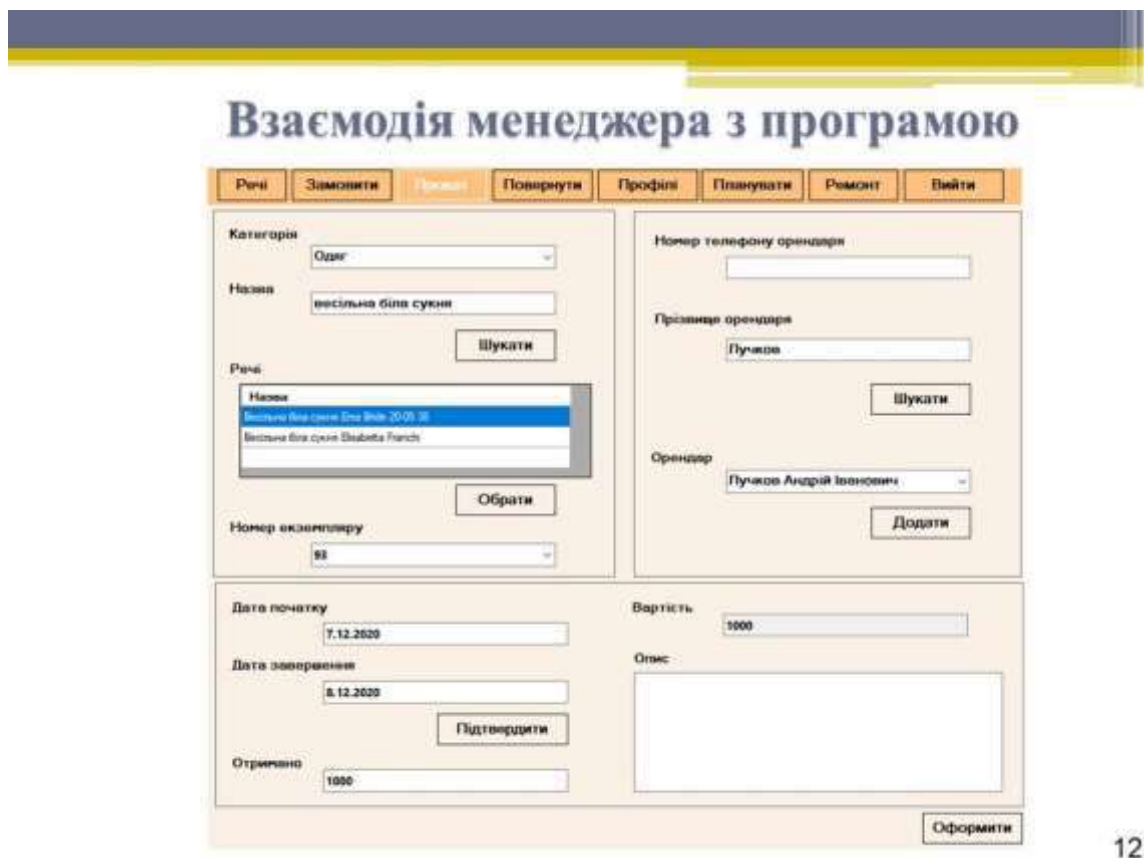


Рисунок В.12 – Слайд 12

Прогнозування затребуваності прокату вживаних речей

Місяць	Кількість
1	45
2	42
3	45
4	47
5	53
6	60
7	57
8	63
9	58
10	60
11	57
12 прогноз	60

13

Рисунок В.13 – Слайд 13

14

Висновки

Виконано аналіз проблеми прокату вживаних речей. Розглянуто задачу визначення затребуваності прокату вживаних речей як задачу прогнозування, проаналізовано методи, які можуть потенційно на цій основі до неї застосовуватися.

Реалізовано програмне забезпечення у вигляді програмної системи, що складається з Windows-застосунку з графічним інтерфейсом та скрипту, який запускається для виконання прогнозування затребуваності прокату вживаних речей з заданим періодом у місяць.

Наукова новизна роботи визначається запропонованим методом визначення затребуваності прокату вживаних речей, що ґрунтується на моделі прогнозування відповідних значень для різних речей заданої категорії на майбутній період за послідовністю попередніх значень на основі використання моделі архітектури GRU, що дозволяє приймати рішення щодо необхідної кількості речей в прокаті сервісу на майбутній період.

14

Рисунок В.14 – Слайд 14