НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмний застосунок моніторингу стану лікарських засобів

Виконав студент IV курсу, групи ІТ-01
(шифр групи)

Гончаренко Дмитро Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст.викл. Вітковська І.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент к.т.н., доцент, Демчинський В.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2024

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Гончаренко Дмитру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Програмний застосунок моніторингу стану лікарських засобів

керівник проєкту Вітковська Ірина Іванівна, ст. викл.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «27» травня 2024 р. №2112-с

2. Термін подання студентом проєкту « 17 » червня 2024 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Передпроєктне обстеження предметної області: аналіз предметної області, існуючих рішень, аналіз відомих програмних продуктів, аналіз відомих алгоритмічних та технічних рішень, опис бізнес процесів, постановка задачі.
2) Розроблення вимог до програмного забезпечення: варіанти використання програмного забезпечення, аналіз системних вимог, розроблення функціональних вимог, розроблення нефункціональних вимог.
3) Конструювання та розроблення програмного забезпечення: архітектура програмного забезпечення, обґрунтування вибору засобів розробки, конструювання програмного забезпечення, аналіз безпеки даних.
4) Аналіз якості та тестування програмного забезпечення: аналіз якості ПЗ, опис процесів тестування, опис контрольного прикладу.

5) Розгортання та супровід програмного забезпечення: розгортання програмного забезпечення, супровід програмного забезпечення _____

5. Перелік графічного матеріалу

1) Схема структурна діяльності_____

2) Схема структурна варіантів використань_____

3) Схема бази даних_____

_____

_____

_____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «11» березня 2024 року_____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	21.03.2024	
2	Аналіз існуючих методів розв’язання задачі	26.03.2024	
3	Постановка та формалізація задачі	29.03.2024	
4	Розробка інформаційного забезпечення	04.04.2024	
5	Алгоритмізація задачі	08.04.2024	
6	Обґрунтування вибору використаних технічних засобів	16.04.2024	
7	Розробка програмного забезпечення	29.04.2024	
8	Налагодження програми	12.05.2024	
9	Виконання графічних документів	19.05.2024	
10	Оформлення пояснювальної записки	29.05.2024	
11	Подання ДП на попередній захист	02.06.2024	
12	Подання ДП рецензенту	10.06.2024	
13	Подання ДП на основний захист	14.06.2024	

Студент _____ Дмитро ГОНЧАРЕНКО

(підпис) (ініціали, прізвище)

Керівник _____ Ірина ВІТКОВСЬКА

(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п’яти розділів, містить 27 таблиць, 17 рисунків та 9 джерел – загалом 48 сторінок.

Дипломний проєкт присвячений розробці програмного застосунку для моніторингу стану лікарських засобів.

Мета: покращення процесу управління лікарськими засобами шляхом впровадження прогнозування попиту на медикаменти.

Об'єкт дослідження: процес управління запасами лікарських засобів.

Предмет дослідження: методи для моніторингу та управління станом лікарських засобів, включаючи операції над ліками, генерацію звітів та прогнозування попиту.

У першому розділі проведено аналіз предметної області та існуючих рішень, також розглянуто відомі аналоги. Крім цього описані бізнес-процеси й поставлені задачі, які треба вирішити під час розробки.

У другому розділі наведені варіанти використання програмного забезпечення, а також розроблено функціональні та нефункціональні вимоги.

Третій розділ присвячений безпосередньо розробці програмного забезпечення. Розглянуто архітектуру застосунку, обґрунтовано вибір засобів розробки, описано процес створення програмного забезпечення, включаючи алгоритми та методи, які були використані.

У четвертому розділі проведено аналіз якості та тестування програмного забезпечення.

У п’ятому розділі описано процес впровадження та супровід програмного забезпечення.

Результатом роботи є програмний застосунок, який надає можливість користувачу взаємодіяти з запасами лікарських засобів.

КЛЮЧОВІ СЛОВА: ПРОГРАМНИЙ ЗАСТОСУНОК, ЛІКАРСЬКИЙ ЗАСІБ, МОНІТОРИНГ, УПРАВЛІННЯ СТАНОМ, ПРОГНОЗУВАННЯ ПОПИТУ.

ABSTRACT

The explanatory note of the diploma project consists of four chapters, 27 tables, 17 figures, and 9 sources - 48 pages in total.

The diploma project is dedicated to the development of a software application for monitoring the condition of medicines.

The goal is to create a software application that provides a user-friendly and intuitive interface for managing medicines inventory, the ability to manage the process of adding, updating, and monitoring the availability of medicines, as well as generate reports and forecast demand for goods in the warehouse.

Object of research: the process of managing medicines inventory.

Subject of the study: methods for monitoring and managing the status of medicines, including drug operations, report generation, and demand forecasting.

The first section analyzes the subject area and existing solutions, and reviews well-known analogs. It also describes the business processes and tasks to be solved during development.

In the second section, we describe the options for using the software and develop functional and non-functional requirements.

The third section is devoted to software development. The architecture of the application is considered, the choice of development tools is justified, and the process of creating software is described, including the algorithms and methods that were used.

The fourth section analyzes software quality and testing.

The fifth section describes the process of implementing and maintaining the software.

The result is a software application that allows the user to interact with the inventory of medicines.

KEYWORDS: SOFTWARE APPLICATION, MEDICINAL PRODUCT, MONITORING, CONDITION MANAGEMENT, DEMAND FORECASTING.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-0105.045430.01.91	Технічне завдання	13	
3	А4	КПІ.ІТ-0105.045430.02.81	Пояснювальна записка	48	
4	А4	КПІ.ІТ-0105.045430.03.12	Текст програми	24	
5	А4	КПІ.ІТ-0105.045430.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-0105.045430.05.34	Керівництво користувача	7	
7	А3	КПІ.ІТ-0105.045430.06.99	Графічний матеріал	3	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ-0105.045430.00.90		
Розроб.		Гончаренко Д.С.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Вітковська І.І.					
Н.контр.		Вітковська І.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-01	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2024 р.

ПРОГРАМНИЙ ЗАСТОСУНОК МОНІТОРИНГУ СТАНУ ЛІКАРСЬКИХ ЗАСОБІВ

Технічне завдання

КПІ.ІТ-0105.045430.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Ірина ВІТКОВСЬКА

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Дмитро ГОНЧАРЕНКО

Київ – 2024

Зміст

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмний застосунок моніторингу стану лікарських засобів.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення “Програмний застосунок моніторингу стану лікарських засобів” КПІ.ІТ-0105.045430.01.91, котра використовується для моніторингу стану лікарських засобів, генерації звітів, прогнозування попиту на товари та призначена для керування процесом управління запасами в медичних закладах та аптеках.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки “Програмний застосунок моніторингу стану лікарських засобів” є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизації та полегшенню роботи з моніторингом стану лікарських засобів.

Метою розробки є покращення процесу управління лікарськими засобами шляхом впровадження прогнозування попиту на медикаменти.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу

Виведення головного меню. Прототип меню зображено на рисунку 4.1.

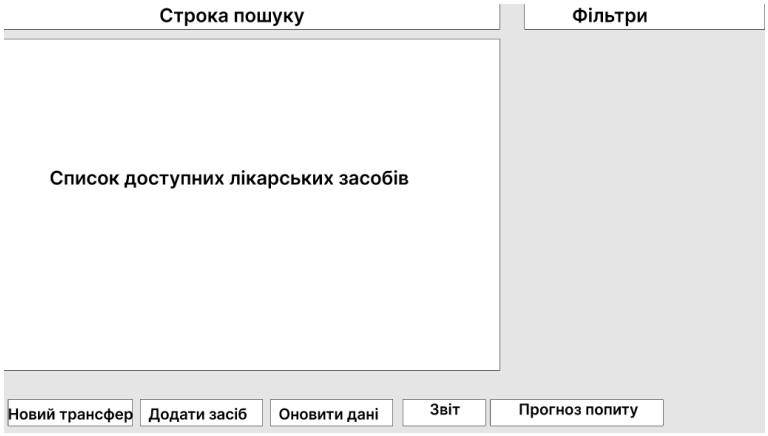


Рисунок 4.1 - Головне меню

Виведення форм для додавання та оновлення інформації про лікарський засіб(рис. 4.2).

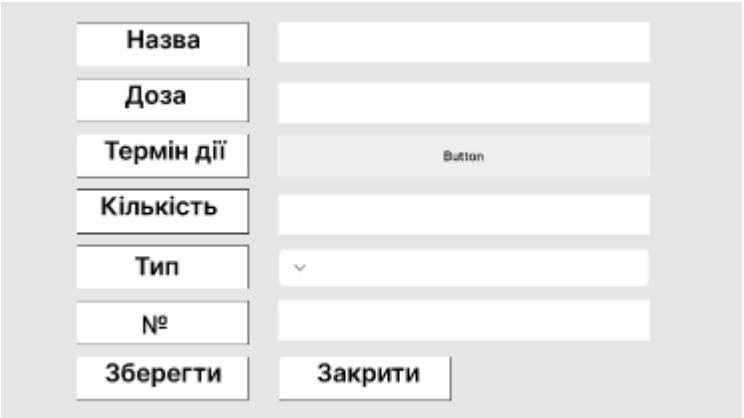


Рисунок 4.2 - Форма для додавання та оновлення інформації

Виведення форми для заповнення інформації про новий трансфер(рис. 4.3).

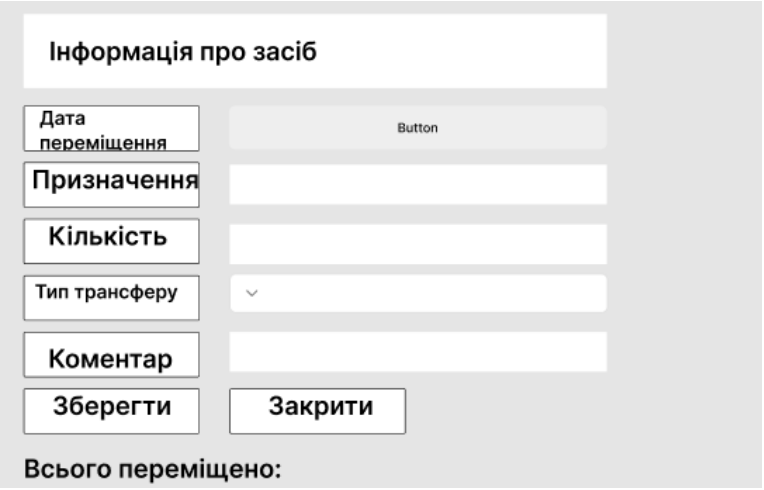


Рисунок 4.3 - Форма для нового трансферу

Виведення форми для генерації звіту(рис. 4.4).

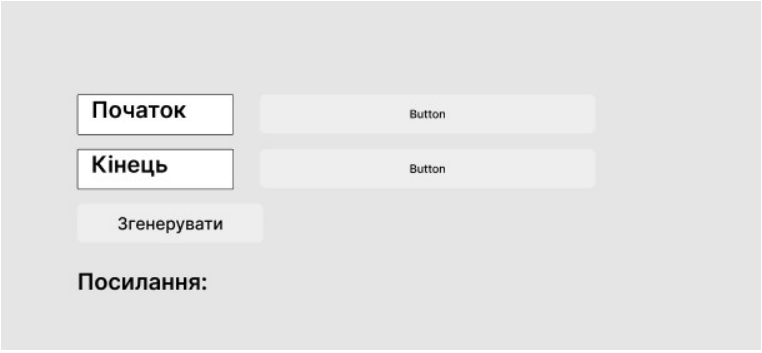


Рисунок 4.4 - Форма для генерації звіту

Виведення форми для прогнозування попиту(рис. 4.5).

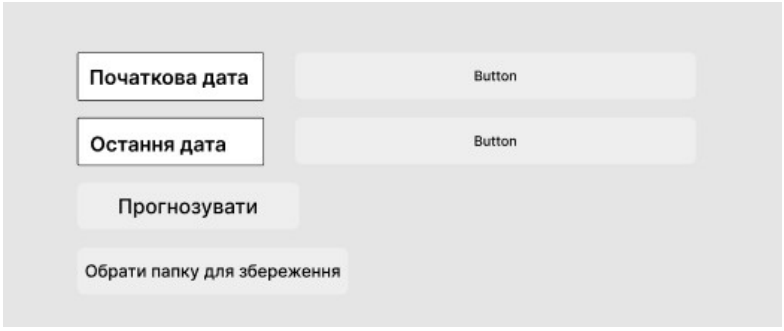


Рисунок 4.5 - Форма для прогнозування попиту

4.1.2 Для користувача:

– перегляд доступних ліків;

– пошук лікарського засобу за назвою;

– перегляд лікарських засобів за допомогою фільтрів;

– додавання нового лікарського засобу;

– оновлення інформації про лікарський засіб;

– додавання нового трансферу;

– можливість згенерувати звіт;

– можливість спрогнозувати попит на товари.

4.1.3 Для адміністратора системи:

Вимоги не висуваються.

4.1.4 Додаткові вимоги:

– безпечне збереження даних;

– мова інтерфейсу - українська.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

– тип процесору: AMD Ryzen 3;

– об‘єм ОЗП: 2 Гб;

– кількість ядер процесору: 2.

Рекомендована конфігурація технічних засобів:

– тип процесору: AMD Ryzen 5;
– об‘єм ОЗП: 16 Гб;
– кількість ядер процесору: 4.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем Windows 10 та Windows 11.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: заповнення відповідних полів та форм текстом або цифрами.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: таблиці з даними про лікарські засоби, графіки, що побудовані на основі цих даних.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Python.

4.5.4 Вимоги до середовища розробки

Розробку виконати за допомогою середовища PyCharm.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді репозиторію на сайті Gitlab.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Локальне розгортання проєкту.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– схема структура діяльності;
– схема бази даних.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	28.02	
2.	Розробка технічного завдання	10.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	26.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	1.05	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	8.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	12.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	19.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	24.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Програмний застосунок моніторингу стану лікарських засобів

КПІ.ІТ-0105.045430.02.81

Київ – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
IT	–	Інформаційні технології
ОС	–	Операційна система.
LOC	–	Lines of code
GUI	–	Graphical User Interface
ПЗ	–	Програмне забезпечення.
ER	–	Entity-Relation diagram
БД	–	База даних.

ВСТУП

Охорона здоров'я є однією з найважливіших галузей, яка потребує постійного вдосконалення та інновацій. Ефективне управління лікарськими засобами є критично важливим аспектом діяльності медичних закладів, оскільки воно безпосередньо впливає на якість медичної допомоги та безпеку пацієнтів. Важливість розробки програмного додатку для моніторингу стану лікарських засобів зумовлена необхідністю забезпечення належного контролю за інвентаризацією, умовами зберігання, термінами придатності та своєчасним оновленням інформації.

Програмні застосунки для моніторингу стану лікарських засобів можуть значно оптимізувати роботу медичного персоналу, зменшуючи адміністративні навантаження та ризики, пов'язані з людським фактором. Автоматизація процесів контролю за лікарськими засобами сприяє зниженню витрат на їх утримання, підвищенню прозорості та точності обліку, а також покращенню загальної роботи системи охорони здоров'я.

Світові тенденції свідчать про все більше використання інформаційних технологій у сфері охорони здоров'я для оптимізації процесів управління та підвищення рівня надання медичних послуг. Розробка програмного забезпечення, що забезпечує моніторинг лікарських засобів, стає пріоритетним завданням для багатьох закладів охорони здоров'я по всьому світу.

Існує багато різних підходів та систем для моніторингу стану лікарських засобів, але більшість з них мають певні обмеження. Ці системи часто не дозволяють швидко реагувати на зміни умов зберігання або потреби в поповненні запасів.

Метою даної роботи є покращення процесу управління лікарськими засобами шляхом впровадження прогнозування попиту на медикаменти. В даній дипломній роботі будуть розглянуті сучасні методи розробки

програмного забезпечення, включаючи опис архітектури, складових частин, технічних і функціональних вимог.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Моніторинг лікарських засобів має вагоме значення для забезпечення високого рівня та безпеки медичних послуг. Він включає в себе управління розподілом, зберіганням, використанням та утилізацією лікарських засобів. Програмне забезпечення, яке забезпечує моніторинг за лікарськими засобами необхідно в сучасному світі, де інформаційні технології є життєво важливими для багатьох сфер.

Існує широкий вибір програмного забезпечення для моніторингу лікарських засобів, включаючи спеціалізовані системи управління аптеками та загальні системи електронних медичних записів. Деякі з цих систем надають функції для відстеження запасів, управління термінами придатності, реєстрації переміщення ліків та обробки інформації про постачальників.

У рамках дипломного проєкту обрано розробку програмного застосунку для моніторингу стану лікарських засобів, який буде базуватися на сучасних IT-технологіях та враховувати потреби медичних закладів для управління лікарськими засобами. Для реалізації програмного застосунку для моніторингу стану лікарських засобів планується використання мови програмування Python та бази даних SQLite[1].

Важливою особливістю програмного застосунку буде модуль прогнозування, який на основі історичних даних та алгоритмів зможе генерувати графік майбутніх потреб в лікарських засобах. Це сприятиме більш раціональному використанню ресурсів та оптимізації витрат медичних закладів.

1.2 Аналіз існуючих рішень

Буде розглянуто поточне програмне забезпечення в цій галузі, а також технічні рішення, які допоможуть у впровадженні програмного застосунку для

моніторингу стану лікарських засобів. Після цього буде зроблено аналіз програмних рішень, а також інструментів розробки для реалізації цього застосунку.

1.2.1 Аналіз відомих програмних продуктів

За аналоги було взято такі застосунки: АНР-Аптека[2] та IBS Аптека. АНР-Аптека - це застосунок для моніторингу стану лікарських засобів та управління аптечними запасами. Користувачі можуть легко та швидко переглядати інформацію про доступні ліки, робити пошук, а також керувати наявними запасами лікарських препаратів. Вони можуть використовувати зручний інтерфейс додатка для доступу до всіх функцій системи. IBS Аптека - це потужний інструмент для аптекарів і менеджерів аптек, який значно полегшує управління запасами лікарських засобів і забезпечує ефективний моніторинг їх стану. Порівняння дипломного проекту з аналогом наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння з аналогом

Функціонал	PharmaStock	АНР-Аптека	IBS Аптека	Пояснення
Керування асортиментом	Так	Так	Так	Можливість керувати асортиментом доступних ліків
Управління товарними запасами	Так	Так	Так	Можливість редагувати залишки товарів

Продовження таблиці 1.1

Функціонал	PharmaStock	AHP-Аптека	IBS Аптека	Пояснення
Інтерфейс користувача	Так	Так	Так	Інтуїтивно зрозумілий інтерфейс
Фільтрація	Так	Так	Так	Можливість фільтрувати доступні ліки
Генерування звіту	Так	Так	Ні	Можливість згенерувати звіти
Прогноз попиту	Так	Ні	Ні	Можливість прогнозування попиту на товари

1.2.2 Аналіз відомих алгоритмічних та технічних рішень

Для розробки програмного застосунку для моніторингу стану лікарських засобів можуть бути використані різноманітні алгоритмічні та технічні рішення. Нижче представлено деякі з них:

Алгоритми валідації даних[3].

Використання регулярних виразів для перевірки формату назв лікарських препаратів та дат їх придбання. Перевірка діапазону значень для підтвердження коректності введених доз та кількості одиниць препаратів.

Переваги: забезпечують точність та коректність введених даних, що допомагає у підтримці якості інформації у системі. Допомагають у запобіганні помилкових операцій та уникненні недоречних дій користувачів.

Недоліки: можуть стати додатковим джерелом складнощів для користувачів, якщо вимоги до формату даних є занадто обмеженими.

Управління даними.

Застосування баз даних для зберігання та організації інформації про лікарські препарати, їх кількість, терміни придбання тощо. Використання структур даних, таких як стеки або черги, для управління рухом лікарських засобів на складі.

Переваги: допомагають ефективно організувати та зберігати великий обсяг інформації про лікарські засоби. Забезпечують можливість швидкого доступу до даних та їх оновлення в реальному часі.

Недоліки: можуть виникнути проблеми з безпекою даних при неправильній конфігурації або несанкціонованому доступі. Вимагають ретельного контролю за якістю та цілісністю даних, щоб уникнути втрати інформації або помилкової обробки.

Використання методу випадкового лісу[4].

Переваги: висока точність, яка досягається завдяки здатності обробляти складні нелінійні залежності, обробка великого обсягу даних та гнучкість.

Недоліки: вибір оптимальних параметрів для моделі може бути складним і потребувати багато часу, це в свою чергу залежить від розміру набору даних, тому що ця модель займає велику частину обчислювальних ресурсів.

Крім того, використання алгоритмів обчислень для автоматизованого обчислення різних параметрів дозволяє ефективно використовувати дані в подальшому для прогнозування попиту.

1.3 Опис бізнес-процесів

Бізнес процеси наведені в BPMN моделі, яка представлена в графічному матеріалі, креслення 1.

Опис послідовності бізнес-процесів моніторингу ліків:

– користувач увійшов в систему(відкрив застосунок);

– користувач переглядає доступні ліки;

– користувач приймає рішення в залежності від потреби;

– пошук лікарських засобів за назвою або через фільтри;

– якщо товар в наявності, то може оновити інформацію;

– користувач може спрогнозувати попит на товари за певний період;

– користувач може додавати нові лікарські засоби;

– якщо товар закінчився, то оновлює запас;

– генерує звіт.

1.4 Постановка задачі

Метою розробки є покращення процесу управління лікарськими засобами шляхом впровадження прогнозування попиту на медикаменти.

Цілі:

– розробка інтерфейсу для користувачів, що дозволяє зручно відстежувати стан запасів лікарських засобів та виконувати різноманітні операції з ними;

– реалізація процесу моніторингу запасів, включаючи оновлення даних про наявність препаратів;

– генерування звітів використання лікарських засобів для оптимізації управління запасами та планування закупівель;

– можливість прогнозувати попит товарів на основі історичних даних.

Задачі:

– розробка структури бази даних для збереження інформації про лікарські засоби, їх кількість та параметри;

– реалізація функціоналу інтерфейсу для відстеження стану запасів, включаючи можливість перегляду, додавання, редагування та видалення записів;

– реалізація функціоналу генерування звітів та попиту на лікарські засобі.

Висновки до розділу

В даному розділі було проведено передпроєктне обстеження предметної області розробки програмного застосунку для моніторингу стану лікарських засобів.

Проаналізовано існуючи програмні продукти, аналоги, які виконують подібний функціонал, з вказанням відмінностей між ними та розроблюваним програмним застосунком для моніторингу лікарських засобів.

Висвітлено технології та алгоритми, що будуть використовуватися під час розробки даного застосунку.

Далі було проаналізовано бізнес-процеси, які будуть відбуватися при використанні застосунку користувачами.

Завершивши аналіз, були визначені мета, цілі розробки та задачі, які мають бути вирішені у процесі розробки програмного забезпечення для моніторингу стану лікарських засобів.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Програмне забезпечення для моніторингу лікарських засобів має певний спектр функціональних можливостей. Основні сценарії використання включають контроль запасів, оптимізацію процесу постачання, забезпечення відповідності термінам придатності, а також генерація звітів та прогнозування попиту.

Головний функціонал програмного забезпечення для моніторингу лікарських засобів буде виглядати наступним чином:

- перегляд доступних ліків - користувач може переглядати доступні ліки;
- додавання нового товару - користувач може додати новий лікарський засіб;
- оновлення товару - користувач може оновити дані про лікарський засіб;
- генерація звіту - користувач може згенерувати звіт за певний час;
- пошук ліків - користувач може зробити пошук ліків за назвою або за допомогою фільтрів;
- прогнозування попиту - користувач може згенерувати графік попиту на товари за певний проміжок часу.

Діаграма варіантів використання представлена в креслені 2, яке наведено в графічному матеріалі.

В таблицях 2.1 - 2.7 наведені варіанти використання програмного забезпечення.

Таблиця 2.1 - Варіант використання UC-1

Use case name	Перегляд доступних ліків
Use case ID	UC-01
Goals	Можливість переглянути доступні ліки
Actors	Користувач
Trigger	Користувач хоче переглянути доступні ліки
Pre-conditions	Програмний застосунок запущено
Flow of Events	Користувач потрапляє на головне вікно зі всіма доступними ліками. Далі він може переглянути всі доступні ліки або передивитись детальну інформацію
Extension	-
Post-Condition	Користувач передивився доступні ліки

Таблиця 2.2 - Варіант використання UC-2

Use case name	Пошук доступних ліків
Use case ID	UC-02
Goals	Можливість знайти відповідні ліки за назвою
Actors	Користувач
Trigger	Користувач хоче знайти відповідний лікарський засіб
Pre-conditions	-
Flow of Events	Користувач у відповідній комірці вводить назву для пошуку лікарського засобу. Якщо товар є на складі, користувач побачить відповідний лікарський засіб
Extension	В випадку введення некоректних даних користувач не побачить результат пошуку

Продовження таблиці 2.2

Post-Condition	Користувач зробив пошук ліків за назвою

Таблиця 2.3 - Варіант використання UC-3

Use case name	Оновлення інформації лікарських засобів
Use case ID	UC-03
Goals	Можливість оновити дані доступних ліків
Actors	Користувач
Trigger	Користувач бажає оновити дані про доступні ліки
Pre-conditions	-
Flow of Events	Користувач обирає потрібний йому лікарський засіб. Далі натискає на відповідну кнопку й оновлює інформацію про засіб у новому вікні. Далі зберігає інформацію натиснувши на відповідну кнопку.
Extension	-
Post-Condition	Користувач оновив інформацію про лікарський засіб

Таблиця 2.4 - Варіант використання UC-4

Use case name	Додавання нового товару
Use case ID	UC-04
Goals	Можливість додати новий лікарський засіб
Actors	Користувач
Trigger	Користувач бажає додати новий лікарський засіб

Продовження таблиці 2.4

Pre-conditions	-
Flow of Events	Користувач натискає на відповідну кнопку щоб додати новий засіб. У новому вікні вводить дані й зберігає інформацію.
Extension	-
Post-Condition	Користувач додав новий лікарський засіб

Таблиця 2.5 - Варіант використання UC-5

Use case name	Генерація звіту
Use case ID	UC-05
Goals	Можливість згенерувати звіт
Actors	Користувач
Trigger	Користувач бажає згенерувати звіт
Pre-conditions	-
Flow of Events	Користувач натискає на відповідну кнопку для генерації звіту. В новому вікні обирає за який період потрібен звіт. Натискає на відповідну кнопку для генерації звіту. Далі натискає на посилання для перегляду звіту.
Extension	-
Post-Condition	Користувач згенерував звіт

Таблиця 2.6 - Варіант використання UC-6

Use case name	Пошук доступних ліків за фільтрами
Use case ID	UC-02
Goals	Можливість знайти відповідні ліки за допомогою фільтрів

Продовження таблиці 2.6

Actors	Користувач
Trigger	Користувач хоче знайти відповідні лікарські засоби
Pre-conditions	-
Flow of Events	Користувач у відповідній комірці вводить назву для пошуку лікарського засобу. Якщо товар є на складі, користувач побачить відповідний лікарський засіб
Extension	В випадку якщо дані відсутні користувач не побачить результат пошуку.
Post-Condition	Користувач зробив пошук ліків за допомогою фільтрів

Таблиця 2.7 - Варіант використання UC-7

Use case name	Прогнозування попиту
Use case ID	UC-07
Goals	Можливість згенерувати графік попиту
Actors	Користувач
Trigger	Користувач бажає побачити попит на товари за певний проміжок часу
Pre-conditions	-
Flow of Events	Користувач натискає на відповідну кнопку для генерації попиту на товари. В новому вікні обирає за який період потрібен графік. Натискає на відповідну кнопку для прогнозування попиту. Далі може зберегти графік за потреби.
Extension	-
Post-Condition	Користувач спрогнозував попит на товари

2.2 Аналіз системних вимог

Програмними вимогами є наявні операційна Windows 10 або Windows 11 та сервер бази даних SQLite з налаштованими правами доступу для зберігання даних про ліки.

Інші програмні вимоги не висуваються.

2.3 Розроблення функціональних вимог

В рамках дипломного проекту визначені такі функціональні вимоги:

– перегляд доступних ліків: користувач може переглядати доступні ліки для подальшої взаємодії з ними;

– додавання нового лікарського засобу: користувач може додати новий лікарський засіб, вписавши необхідну інформацію й зберегти його;

– оновлення інформації про лікарський засіб: користувач може оновити інформацію про певний лікарський засіб;

– пошук лікарських засобів: користувач може знайти певний лікарський засіб, який є на складі або ж знайти необхідні ліки в залежності від потреб за допомогою фільтрів;

– генерація звіту: користувач може згенерувати звіт за певний період часу;

– прогнозування попиту: користувач може спрогнозувати графік попиту на товари за обраний період.

ПЗ розділене на модулі, де кожний модуль виконує свої певні функції. Таблиця 2.8 містить опис загальної моделі вимог, а таблиці 2.9 — 2.15 містять опис функціональних вимог до ПЗ. Матриця трасування вимог наведена на рисунку 2.1.

Таблиця 2.8 – Загальна модель вимог

№	Назва	ID вимоги	Пріоритети	Ризики
1	Перегляд доступних ліків	FR-1	Високий	Низький
2	Додавання нового лікарського засобу	FR-2	Високий	Середній
3	Оновлення інформації про лікарський засіб	FR-3	Середній	Середній
4	Пошук лікарських засобів	FR-4	Високий	Високий
5	Генерація звіту	FR-5	Середній	Низький
6	Інтерфейс користувача	FR-6	Високий	Низький
7	Прогнозування попиту	FR-7	Високий	Середній

Таблиця 2.9 – Функціональна вимога - 1

ID вимоги	Перегляд доступних ліків
FR-01	Користувач може переглянути доступні ліки.

Таблиця 2.10 – Функціональна вимога - 2

ID вимоги	Додавання нового лікарського засобу
FR-02	Користувач може додати новий лікарський засіб. Потім зберегти його й перевірити результат.

Таблиця 2.11 – Функціональна вимога - 3

ID вимоги	Оновлення інформації про лікарський засіб
FR-03	Користувач може переглянути деталі про лікарський засіб. За потреби оновити дані й зберегти.

Таблиця 2.12 – Функціональна вимога - 4

ID вимоги	Пошук лікарських засобів
FR-04	Користувач може знайти за назвою певні ліки або ж застосувати необхідні фільтри для пошуку ліків.

Таблиця 2.13 – Функціональна вимога - 5

ID вимоги	Генерація звіту
FR-05	Користувач може згенерувати звіт за певний період часу.

Таблиця 2.14 – Функціональна вимога - 6

ID вимоги	Інтерфейс користувача
FR-06	Простий й зрозумілий користувацький інтерфейс

Таблиця 2.15 – Функціональна вимога - 7

ID вимоги	Прогнозування попиту
FR-07	Користувач може згенерувати графік попиту на товари за певний період часу.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7
UC-01	+					+	
UC-02				+		+	
UC-03			+			+	
UC-04		+				+	
UC-05					+	+	
UC-06				+		+	
UC-07						+	+

Рисунок 2.1 — Матриця трасування вимог

2.4 Розроблення нефункціональних вимог

Швидкодія: застосунок для моніторингу стану лікарських засобів повинен забезпечувати достатню швидкодію, щоб користувачі могли оперативно обробляти інформацію про запаси. Обробка великих обсягів даних, наприклад, під час генерації звітів, має виконуватись не більше ніж за 60 секунд.

Надійність: застосунок повинен функціонувати стабільно і без збоїв, забезпечуючи безперервність роботи користувачів. Це особливо важливо для аптек, де інформація про запаси лікарських засобів має бути доступною в будь-який час.

Висновки до розділу

У цьому розділі було розглянуто варіанти використання програмного застосунку для моніторингу стану лікарських засобів. Надано детальний опис сценаріїв використання, що охоплюють дії користувача під час роботи із застосунком.

Також проведено аналіз системних вимог, які необхідно враховувати під час розробки. Розглянуто функціональні та нефункціональні вимоги, зокрема їх детальний опис у вигляді таблиць. Сформовано матрицю трасування вимог.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

В якості типу архітектури для розробки програмного забезпечення було обрано шарову архітектуру(layered architecture)[5]. Це обумовлено кількома ключовими факторами.

По-перше, розподіл на шари забезпечує чітке розмежування функціональностей, що сприяє кращій організації коду та спрощує його розуміння та підтримку. Кожен шар виконує конкретну роль, що дозволяє зосередитися на окремих аспектах системи.

По-друге, шарова архітектура підвищує гнучкість та адаптивність системи. Завдяки такій структурі можна легко вносити зміни або додавати нові функціональні можливості, не порушуючи роботу інших частин програми. Це особливо важливо в контексті управління запасами лікарських засобів, де можуть змінюватися бізнес-процеси та вимоги до системи.

По-третє, така архітектура спрощує тестування та налагодження. Оскільки кожен шар може бути протестований незалежно, це зменшує ймовірність помилок та дозволяє швидко виявляти та виправляти проблеми.

Нижче наведено опис архітектури програмного забезпечення:

– для розробки програмного засобу для управління запасами лікарських засобів використовується шарова архітектура, яка розділяє програмне забезпечення на рівні з визначеними функціональностями;

– шар доступу до даних відповідає за взаємодію з базою даних SQLite та виконання операцій з даними, такими як додавання, оновлення та вибірка;

– бізнес-логіка додатку визначає способи взаємодії з даними та виконання бізнес-процесів, пов'язаних з управлінням запасами лікарських

засобів. Його компоненти включають функції для роботи з даними про ліки, рух ліків та інвентаризації;

– шар представлення даних містить інтерфейс користувача для взаємодії з програмою;

– компонентами є таблиця ліків, яка містить дані про лікарські засоби, їх дозування, термін придатності та іншу інформацію. Також таблиця інвентар, яка зберігає дані про переміщення ліків, такі як дату, тип переміщення, кількість тощо.

На рисунку 3.1 наведено модель архітектури програмного забезпечення.

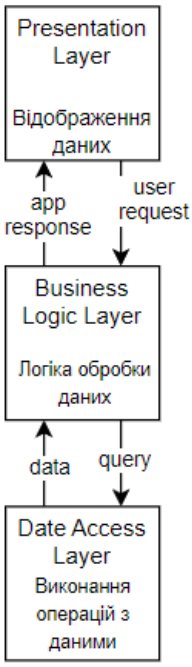


Рисунок 3.1 — Модель архітектури

Виконання нефункціональних вимог:

– швидкодія та надійність: використання SQLite дозволяє швидко здійснювати операції з базою даних;

– масштабованість: база даних SQLite добре підходить для проектів з обмеженим обсягом даних та легко масштабується зі зростанням обсягу інформації.

Тип баз даних: SQlite вибрано через її гнучкість у роботі з динамічними схемами даних та можливість легко масштабувати систему під час збільшення об'єму даних.

Технологічний стек: мова програмування Python та база даних SQlite.

Обґрунтування вибору технологічного стека:

– використання Python для розробки дозволяє швидко та ефективно реалізувати функціональність десктопного застосунку. Python має чистий синтаксис, велику кількість бібліотек та фреймворків для роботи з базами даних, що полегшує розробку;

– база даних SQLite є легкою, локально вбудованою базою даних, що ідеально підходить для десктопних застосунків. Вона не потребує окремого сервера баз даних, тому вона проста у використанні та підтримці.

3.2 Обґрунтування вибору засобів розробки

Для розробки програмного забезпечення для моніторингу стану лікарських засобів будуть використані наступні сторонні бібліотеки:

а) графічний інтерфейс користувача PySimpleGUI[6]:

1) переваги: простий синтаксис, швидка розробка, підтримка різних типів графічних елементів;

2) недоліки: обмежена функціональність порівняно з більш розширеними інструментами розробки GUI;

3) аналоги: Tkinter, wxPython;

б) Scikit-learn[7]:

4) переваги: великий набір алгоритмів для класифікації, регресії, кластеризації, та зниження вимірності, підтримка перехресної валідації для оцінки моделей, підтримка обробки попередніх даних через конвеєрні інструменти;

5) недоліки: обмежена підтримка глибокого навчання у порівнянні з такими бібліотеками, як TensorFlow чи PyTorch, а також може бути неефективним для роботи з дуже великими наборами даних, оскільки не підтримує обробку даних на GPU;

6) аналоги: PyTorch, TensorFlow, XGBoost;

в) Pandas[8]:

7) переваги: зручні та потужні інструменти для роботи з табличними даними, включаючи обробку, маніпулювання та аналіз даних, а також підтримка різних форматів даних, включаючи CSV, Excel, SQL, та інші;

8) недоліки: деякі операції можуть бути повільними для дуже великих наборів даних й може споживати багато пам'яті при роботі з великими наборами даних;

9) аналоги: PySpark, Vaex, Dask.

Вибір технологічного стека: для розробки програмного засобу для управління та моніторингу ліками було обрано Python через його простоту та широкий вибір бібліотек для розробки застосунків. SQLite була обрана як база даних через її легкість використання та інтеграцію з Python. Використання PyCharm як основного редактора коду забезпечує зручну роботу з проектом та розширення для підвищення продуктивності розробки.

3.3 Конструювання програмного забезпечення

Алгоритми валідації даних: використання регулярних виразів для перевірки правильності формату назв лікарських препаратів та дат їх придбання перед їх збереженням у базі даних. Перевірка діапазону значень для підтвердження коректності введених доз та кількості одиниць препаратів перед їх збереженням. Перевірка введених даних користувачем для правильного відображення пошуку товарів за назвою або за допомогою вбудованих фільтрів. Приклад коду зображено на рисунку 3.2.

```
pattern = r"\b(.+)\s(\d{1,4})([a-zA-Z]{1,4})(?=\s|$)\b"

pattern_doseunits = r"\b(\d{1,4})([a-zA-Z]{1,2})\b"

match = re.match(pattern, s)
if match and len(re.findall(pattern_doseunits, s)) == 1:
    dose = match.group(2)
    units = match.group(3)
    name = s.replace(dose + units, "")
    name = re.sub( pattern: r"\s+", repl: " ", name)
    name = name.strip()

pattern = r"\b(.+)\s(\d{1,4},\d{1,4})([a-zA-Z]{1,4})(?=\s|$)\b"
match = re.match(pattern, s)
if match and len(re.findall(pattern_doseunits, s)) == 1:
    dose = match.group(2)
    units = match.group(3)
    name = s.replace(dose + units, "")
    name = re.sub( pattern: r"\s+", repl: " ", name)
    name = name.strip()

return name, dose, units
```

Рисунок 3.2 — Приклад коду алгоритму валідації даних

Алгоритм випадкового лісу: використання методу випадкового лісу для прогнозування попиту на лікарські засоби. Спочатку йде процес збирання даних про використання лікарських засобів за попередні періоди. Далі попередня обробка даних, яка включає в себе обробку значень, нормалізацію даних та перетворення їх у формат придатний для моделі. Потім навчання моделі на основі підготовлених даних. Модель навчається виявляти патерни та залежності в даних. Використання навченого алгоритму для прогнозування попиту на основі вхідних значень. Приклад коду зображено на рисунку 3.3.

```
def main(database_path, start_date, end_date):  1 usage
    df = fetch_historical_data(database_path)
    df = preprocess_data(df)
    model = train_model(df)

    forecast_results = []
    date_range = pd.date_range(start=start_date, end=end_date, freq='M')
    for current_date in date_range:
        forecast = forecast_demand(model, current_date.month, current_date.year)
        forecast_results.append(forecast[0])

    print(f"Прогнозування попиту від {start_date} до {end_date}: {forecast_results}")
    return date_range, forecast_results
```

Рисунок 3.3 - Приклад коду алгоритму випадкового лісу

Управління даними: застосування баз даних для зберігання та організації інформації про лікарські препарати: використовується база даних SQLite для зберігання інформації про лікарські препарати. Вона дозволяє організувати дані у вигляді таблиць та забезпечує ефективний доступ до них. Використання структур даних, таких як стеки або черги, для управління рухом лікарських засобів на складі.

На рисунку 3.4 наведено ER-діаграму сутностей та їх зв’язків

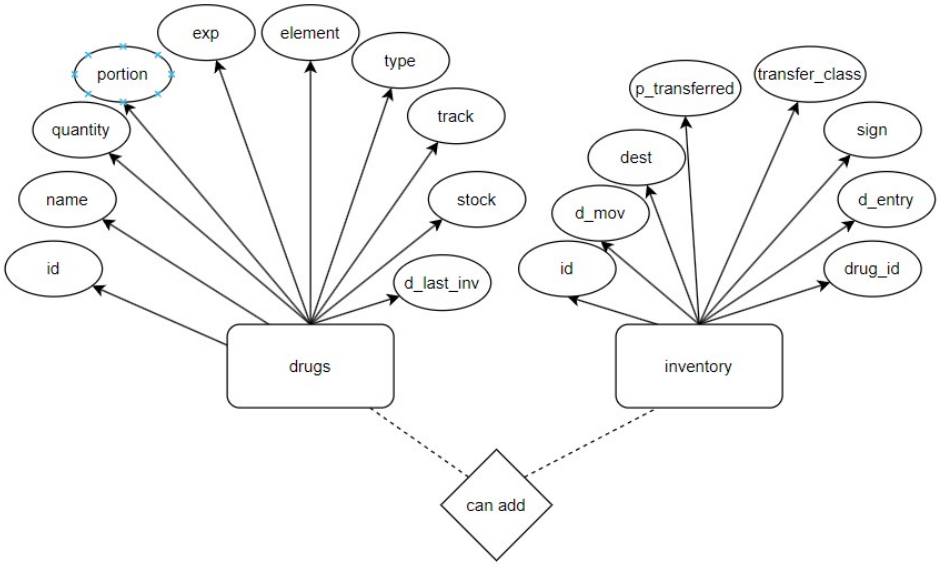


Рисунок 3.4 – ER діаграма сутностей drugs, inventory

Були розроблені наступні структури даних:

– drugs є об’єктом, що представляє ліки, містить поля id, name, quantity, portion, exp, element, type, track, stock, d_last_inv;

– inventory є об’єктом, що представляє собою інвентар, містить поля id, d_mov, dest, p_transferred, transfer_class, sign, d_entry, drug_id(FK).

Опис концептуальної моделі бази даних:

– сутностями є drugs, inventory;

– зв’язками є кожний рух лікарського засобу (запис у таблиці "inventory") пов'язаний з певним лікарським засобом (запис у таблиці "drugs"). Цей зв'язок є багато-до-одного (Many-to-One), оскільки багато рухів може бути пов'язано з одним і тим же лікарським засобом.

Таблиці 3.1-3.2 містять опис таблиць бази даних. Модель бази даних наведена в графічному матеріалі, креслення 3.

Таблиця 3.1 – Опис таблиці drugs

Назва поля	Тип даних	Опис
id	integer	унікальний ідентифікатор для кожного лікарського засобу.
name	text	назва лікарського засобу
quantity	integer	кількість
portion	text	одиниці виміру для дозування
exp	date	дата закінчення терміну придатності
element	integer	кількість одиниць препарату в упаковці
type	text	тип лікарського засобу
track	text	партія лікарського засобу
stock	integer	поточний залишок на складі
d_last_inv	date	остання дата інвентаризації

Таблиця 3.2 — Опис таблиці inventory

Назва поля	Тип даних	Опис
id	integer	унікальний ідентифікатор для кожного пересування
d_mov	date	дата руху лікарських засобів
dest	text	призначення або походження руху
p_transferred	integer	кількість одиниць, які були переміщені

Продовження таблиці 3.2

Назва поля	Тип даних	Опис
transfer_class	text	тип руху (вхід, вихід, інвентаризація)
sign	text	підпис чи інша інформація про автора руху
d_entry	timestamp	дата та час створення запису
drug_id	integer	зовнішній ключ, який посилається на id лікарського засобу у таблиці "drugs"

Таблиця 3.3 містить опис утиліт, бібліотек й іншого стороннього ПЗ, що використовується у розробці.

Таблиця 3.3 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	PyCharm	PyCharm було обрано як основне середовище розробки ПЗ дипломної роботи.
2	SQLite	База даних для зберігання даних. SQLite - це вбудоване СУБД, яке пропонує простий і ефективний спосіб управління БД за допомогою запитів SQL.
3	PySimpleGUI	PySimpleGUI - це проста бібліотека для розробки графічного інтерфейсу користувача (GUI) в програмах Python. Вона надає можливість легко створювати інтерактивні вікна, кнопки, поля для введення даних, таблиці та інші елементи GUI, що робить її відмінним вибором для швидкої розробки додатків.

Продовження таблиці 3.3

№ п/п	Назва утиліти	Опис застосування
4	Scikit-learn	Scikit-learn є відомою бібліотекою для машинного навчання в Python. Ця бібліотека пропонує велику кількість інструментів для створення та оцінювання моделей машинного навчання, включаючи алгоритми для задач класифікації, регресії, кластеризації та зниження розмірності. Додатково, бібліотека включає різні методи передобробки даних, такі як масштабування, нормалізація та кодування.
5	Pandas	Pandas є необхідною бібліотекою в сфері обробки та аналізу даних. Вона пропонує структури даних і інструменти для роботи з табличними даними, що робить завантаження, очищення, обробку та візуалізацію даних простими. Ця бібліотека є незамінним інструментом для аналізу даних у Python, оскільки вона дозволяє виконувати різноманітні завдання з даними, такі як фільтрація, сортування, групування та агрегація.

Тексти програмного коду наведені в окремому документі «Текст програми».

3.4 Аналіз безпеки даних

Захист від введення неправильних даних: використовується механізм валідації введених даних, щоб уникнути помилки при занесенні інформації про лікарські засоби.

Обмеження прав доступу: лише авторизовані користувачі мають доступ до чутливих даних та функціоналу до додатка.

Висновки до розділу

Після проведеного аналізу можна зробити висновок, що обраний стек технологій є оптимальним для реалізації програмного засобу для моніторингу стану лікарських засобів. Цей стек дозволяє забезпечити ефективність, легкість у використанні та надійність, що відповідає вимогам проекту.

Вибір технологій: було розглянуто різноманітні технології та фреймворки для розробки програмного забезпечення. На основі цього аналізу були обрані найкращі технології для даного проекту. У моєму випадку, я обрав Python як основну мову програмування, а також бібліотеки такі як SQLite для роботи з базою даних та PySimpleGUI для створення графічного інтерфейсу користувача.

Середовище розробки: в якості середовища розробки було обрано PyCharm, яке забезпечує зручний редактор коду та інструменти для відлагодження і тестування програм.

База даних: для зберігання даних про лікарські засоби було обрано SQLite, яка є легко використовуваною та має невеликий обсяг пам'яті.

Моделювання даних: були розроблені моделі ER та база даних для більш чіткого розуміння взаємодії даних всередині системи. Це дозволяє ефективно організувати та зберігати інформацію про лікарські засоби.

В результаті, обраний стек технологій і підходи до розробки дозволяють створити безпечний та ефективний програмний застосунок для моніторингу стану лікарських засобів.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Оцінка якості ПЗ та аналіз його продуктивності є ключовими елементами у розробці програмних проектів. Необхідно використовувати інструменти для оцінювання ПЗ, щоб гарантувати якість продукту і виявити проблемні зони. Тому використання різних метрик є очевидним рішенням. Вони дозволяють оцінювати різні характеристики ПЗ, такі як, надійність, розмір, складність і подібні. Дані, які вони надають, можна проаналізувати та використати для подальших рішень щодо покращення застосунку.

Ось деякі популярні метрики якості:

– Lines of Code (LOC) – це метрика, яка визначає загальну кількість рядків у програмному коді. LOC використовується для оцінки розміру проекту. Вона дає загальне уявлення про масштаб проекту.

– Code churn – вимірює обсяги доданого, видаленого та зміненого коду за певний період. Використовується для оцінки активності та стабільності проекту.

– Cyclomatic Complexity – вимірює складність структури коду.

– Code Coverage – показує на скільки код був покритий тестами. Вимірює ефективність тестування і допомагає виявити нетестовані частини коду. Однак, високий відсоток покриття не завжди гарантує відсутність дефектів.

– Change Frequency – відображає, як часто змінюється певний модуль або файл у програмному проекті. Висока частота змін може вказувати на нестабільність або на те, що цей код є критичним і потребує постійного вдосконалення. Вона також може допомогти ідентифікувати потенційні точки ризику в проекті.

– Mean Time to Failure – середній час до відмови системи або компонента. Використовується для оцінки надійності програмного забезпечення.

– Code Reusability – оцінює, наскільки частини коду можуть бути повторно використані в інших проектах або модулях. Високий показник свідчить про хорошу модульність та загальну якість дизайну коду, що дозволяє зменшити дублювання і прискорити процес розробки нових функцій.

– Defect Count - показує кількість виявлених помилок або дефектів у ПЗ. За допомогою цієї метрики можна оцінити якість коду та ефективність процесу тестування. Якість коду та процесів розробки підвищується, коли кількість дефектів зменшується з часом.

На рисунку 4.1 наведено результат оцінки за допомогою сервісу CodeScene.

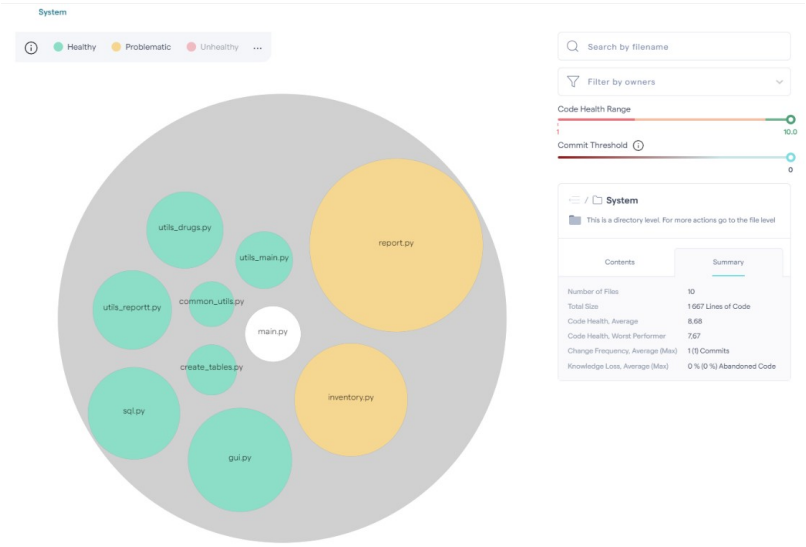


Рисунок 4.1 - Результат оцінки платформою CodeScene

4.2 Опис процесів тестування

Далі було проведене мануальне тестування ПЗ, де докладний опис відповідних тестів наведено у таблицях 4.3 – 4.10.

Таблиця 4.3 – Тест 1.1 Запуск застосунку

Початковий стан системи	Користувач запустив застосунок
Вхідні дані	-
Опис проведення тесту	Користувач натискає на .exe файл для запуску застосунку. Після цього відкривається застосунок й користувач потрапляє на головне меню з переліком доступних лікарських засобів.
Очікуваний результат	Застосунок запускається успішно, користувач бачить головне меню інтерфейсу.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.4 – Тест 1.2 Перегляд доступних ліків

Початковий стан системи	Користувач знаходиться в головному меню
Вхідні дані	-
Опис проведення тесту	Користувач бачить наявні лікарські засоби. За потребою може натиснути на будь-який для перегляду детальної інформації
Очікуваний результат	В меню відображаються всі зареєстровані ліки. Якщо натиснути на лікарський засіб, то відкривається вікно з детальним описом.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.5 – Тест 1.3 Пошук доступних ліків

Початковий стан системи	Користувач знаходиться в головному меню

Продовження таблиці 4.5

Вхідні дані	Назва лікарського засобу
Опис проведення тесту	Користувач в відповідній комірці пошуку вписує потрібну назву лікарського засобу.
Очікуваний результат	Якщо такий лікарський засіб є, то користувач побачить результат. На екран виводиться відповідний лікарський засіб.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.6 – Тест 1.4 Оновлення інформації лікарських засобів

Початковий стан системи	Користувач знаходиться в головному меню
Вхідні дані	-
Опис проведення тесту	Користувач обирає потрібний йому лікарський засіб. Далі відкривається вікно з детальною інформацію про засіб. Користувач змінює відповідні поля. Натискає на кнопку зберегти. Якщо ж ніякі зміни не були внесені натискає кнопку закрити
Очікуваний результат	На екран виводиться нове вікно з детальним описом засобу. Оновлення інформації відбувається успішно. Система попереджає про типові помилки, якщо такі були допущені. Інформація зберігається в базі даних й інформація про засіб оновлюється в списку.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.7 – Тест 1.5 Додавання нового товару

Початковий стан системи	Користувач знаходиться в головному меню
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку “Додати новий засіб”. Після цього в новому вікні заповнює всі необхідні поля. Натискає кнопку “Зберегти”.
Очікуваний результат	На екран виводиться нове вікно з полями, які користувач повинен заповнити. Інформація зберігається. Новий товар з’являється в списку товарів.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.8 – Тест 1.6 Генерація звіту

Початковий стан системи	Користувач знаходиться в головному меню
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку “Звіт”. Після цього в новому вікні обирає за який період хоче отримати дані, натискає на відповідні кнопки “Початок” та “Кінець”. Натискає кнопку “Згенерувати”. Натискає на посилання куди зберігся звіт. Переглядає звіти.
Очікуваний результат	Вікно відкривається. Звіти генеруються. Інформація зберігається в відповідну . Відповідні файли заповнені та відкриваються.
Фактичний результат	Користувач може отримати помилку, якщо був обраний неправильний період або ж якщо дані відсутні.

Таблиця 4.9 – Тест 1.7 Прогнозування попиту

Початковий стан системи	Користувач знаходиться в головному меню
Вхідні дані	-
Опис проведення тесту	Користувач натискає на кнопку “Прогноз попиту”. Після цього в новому вікні обирає за який період хоче отримати дані, натискає на відповідні кнопки “Вибрати початкову дату” та “Вибрати кінцеву дату”. Натискає кнопку “Прогнозувати”. В окремому вікні виводиться графік з даними. За потреби може зберегти цей графік.
Очікуваний результат	Вікно відкривається. Графік відображається. Файли зберігаються у вибрану папку.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.10 – Тест 1.8 Додавання нового трансферу

Початковий стан системи	Користувач знаходиться в головному меню
Вхідні дані	-
Опис проведення тесту	Користувач обирає потрібний йому лікарський засіб та натискає на кнопку “Новий трансфер”. Після цього в новому вікні заповнює відповідні поля. Натискає кнопку “Зберегти”. Бачить результат в списку товарів.
Очікуваний результат	Вікно відкривається. Дані зберігаються. Дані оновлюються в списку лікарських засобів.
Фактичний результат	Збігається з очікуваним.

4.3 Опис контрольного прикладу

Для початку роботи, користувач відкриває .exe файл. Потрапляє на головне меню з списком доступних ліків, яке зображене на рисунку 4.1.

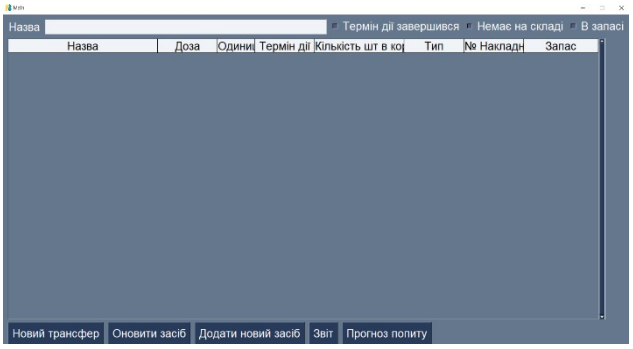


Рисунок 4.1 - Головне меню

Потім в залежності від потреби, користувач може:

– знайти лікарський засіб за назвою як це показано на рисунку 4.2. В відповідній комірці вписується назва потрібного засобу, якщо він є в наявності, то користувач побачить результат в списку;

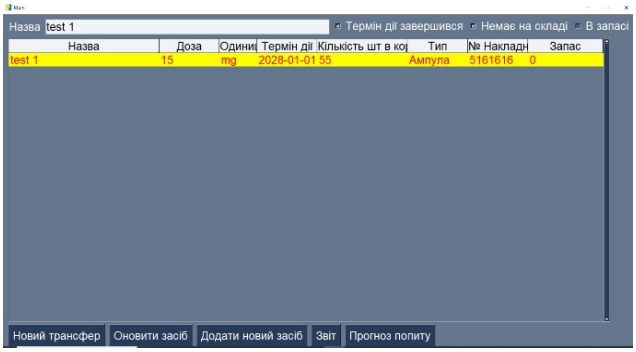


Рисунок 4.2 - Пошук лікарського засобу за назвою

– знайти лікарські засоби за допомогою фільтрів(рис. 4.3). Обираються потрібні фільтри “Термін дії завершився”, “Немає на складі” або “В запасі”. В залежності від обраних фільтрів користувач побачить доступні ліки;

Рисунок 4.3 - Пошук лікарських засобів за допомогою фільтрів

– додати новий лікарський засіб(рис. 4.6). Користувач натискає на кнопку “Додати новий засіб” й вводить відповідні поля. Після цього зберігає інформацію;

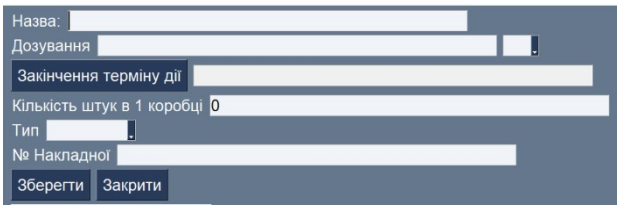


Рисунок 4.6 - Додавання нового лікарського засобу

– оновити інформацію про лікарський засіб(рис. 4.7). Далі користувач може натиснути на кнопку “Оновити засіб”, у новому вікні побачить всю доступну інформацію й за потреби змінює відповідні поля та натискає кнопку “Зберегти”;

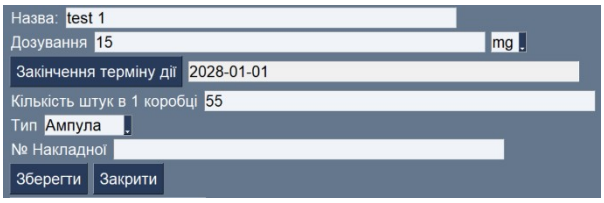


Рисунок 4.7 - Оновлення інформації

– зробити новий трансфер лікарського засобу(рис. 4.8). Користувач вирішив додати новий трансфер й натискає на кнопку “Новий трансфер”. У новому вікні заповнює відповідні поля й зберігає інформацію. Після цього може переглянути зміни в каталогу засобів;

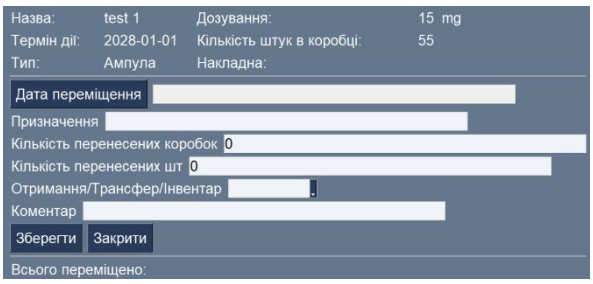


Рисунок 4.8 - Додавання нового трансферу

– також користувач може згенерувати звіт. Натискає на кнопку “Звіт”. Далі обирає за який період йому потрібен звіт, після цього натискає на посилання локальної папки, куди збереглися відповідні файли. Після цього може переглянути вміст цих файлів. На рисунку 4.9 зображено вікно генерації звітів;

Рисунок 4.9 - Генерація звітів

– щоб спрогнозувати попит на товари, користувач натискає на відповідну кнопку “Прогноз попиту”. В новому вікні обирає початкову та кінцеву дату. За потреби може зберегти графік(рис. 4.10).

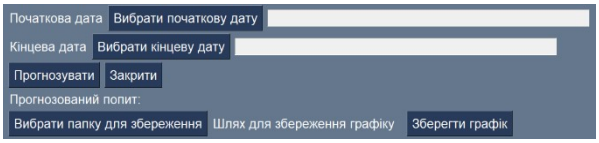


Рисунок 4.10 - Прогнозування попиту

Висновки до розділу

В цьому розділі були зазначені різні метрики якості, які можна обрати для оцінки коду дипломного програмного забезпечення. Використовуючи платформу CodeScene було проаналізовано код на деякі з цих метрик. Аналіз показав, що є певні проблеми з LCOM4(Lack of Cohesion Measure) та незначні зауваження про Bumby Road. Це вказує на те що, деякі функції містять кілька

частин вкладеної умовної логіки всередині однієї функції. Чим глибша вкладеність і більше бампів, тим нижчий рівень “здоров'я” коду. По всім іншим метрикам, які були зазначені, проблем не виявлено.

Також були проведені мануальні тести для перевірки основного функціоналу програмного забезпечення й було проведено контрольний приклад як працює застосунок.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Для розгортання десктопного застосунку використовуються стандартні засоби операційної системи Windows. Основним форматом дистрибутиву є .exe файл, який буде викладено на GitLab у відповідному репозиторії(рис. 5.1).

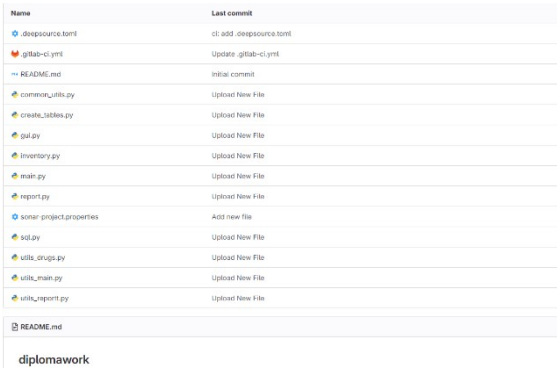


Рисунок 5.1 - Репозиторій GitLab

Використання .exe файлів дозволяє зручно розповсюджувати програмне забезпечення та забезпечує легкість його інсталяції кінцевими користувачами. Даний формат обрано через його популярність та зручність у використанні, що дозволяє користувачам швидко і без зайвих складнощів встановити програму на свій комп'ютер.

Створення .exe файлу та підготовка до його розгортання включає декілька кроків, які виконуються за допомогою bash команд. Нижче наведені основні етапи цього процесу:

– інсталяція необхідних пакетів та бібліотек: перед початком роботи необхідно переконатися, що всі необхідні пакети та бібліотеки встановлені;

– підготовка скриптів: створення директорій для зберігання файлів проекту;

– створення основного Python скрипту: написання коду програми та збереження його у відповідному файлі;

– компіліювання Python коду в .exe файл: використання допоміжних засобів для створення .exe файлу з Python скрипту;

– створення GitLab репозиторію: ініціалізація GitLab репозиторію та додавання файлів проекту;

– додавання згенерованого .exe файлу до репозиторію: переміщення згенерованого .exe файлу до директорії з репозиторієм та коміт цих змін.

Даний формат обрано через його популярність та зручність у використанні, що дозволяє користувачам швидко і без зайвих складнощів встановити програму на свій комп'ютер.

Далі розглянуто локальне розгортання, яке можна побачити на рисунках 5.2 — 5.3.

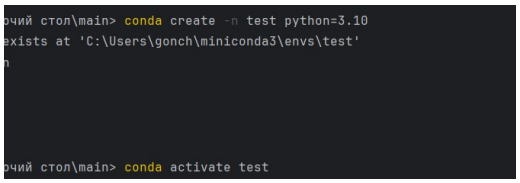


Рисунок 5.2 - Розгортання застосунку - інсталювання віртуального середовища

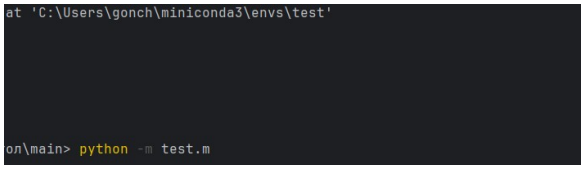


Рисунок 5.3 - Розгортання застосунку - запуск віртуального середовища

Для розгортання проекту локально було вибране середовище розробки Pycharm.

Одна з ключових функцій PyCharm - це можливість робити роботу з репозиторіями Git. Він надає можливість клонувати репозиторії, здійснювати коміти та змінювати гілки прямо з інтерфейсу PyCharm. Крім того, PyCharm надає інтеграцію з GitLab та іншими сервісами для спрощення процесу роботи з репозиторіями.

Після клонування репозиторію на локальний комп'ютер за допомогою PyCharm можна використовувати всі його можливості для розробки проекту. Це включає в себе написання коду, відлагодження програми та інше.

5.2 Супровід програмного забезпечення

Підтримка програмного забезпечення: користувачі повинні мати можливість отримувати нову версію десктопного застосунку з кожним випуском. Для забезпечення безперервної підтримки та оновлення програмного забезпечення використовується GitLab CI/CD[9].

GitLab - це веб-сервіс для керування проектами та система керування версіями, яка базується на відкритому програмному забезпеченні Git. Він надає широкий спектр функцій для спільної роботи над програмними проектами, починаючи від зберігання вихідного коду та завершуючи автоматизованими процесами розгортання та CI/CD.

Однією з ключових особливостей GitLab є можливість створювати репозиторії для зберігання коду та інших файлів проекту. Кожен репозиторій має власний URL та доступ до набору інструментів для спільної роботи, включаючи відстеження змін, управління версіями, можливість робити коментарі до коду, веб-інтерфейс для виправлення помилок та завдань і багато іншого.

Крім того, GitLab надає розширені можливості для автоматизації процесів розробки та розгортання. Зокрема, він має інтегровану систему CI/CD, яка дозволяє автоматизувати тестування коду, збірку програми та розгортання нових версій на тестові та продукційні сервери. Також, GitLab пропонує широкий набір інтеграцій з іншими інструментами та сервісами, що дозволяє розробникам використовувати їхні улюблені інструменти безпосередньо в GitLab. Це може включати інтеграцію з різними системами керування завданнями, сервісами для тестування коду, веб-хостингом статичних сайтів та багато іншого.

Висновки до розділу

У розділі було наведено опис розгортання ПЗ на платформі GitLab, а також локальне розгортання проєкту.

GitLab є повноцінною платформою для розробки програмного забезпечення, що надає розробникам широкий спектр інструментів для спільної роботи над проектами. Наявність усіх інструментів в одному місці спрощує розгортання, відстеження змін та керування проектами, роблячи GitLab зручним вибором для команд будь-якого розміру.

Однак, успіх розгортання програмного забезпечення не обмежується лише створенням .exe файлу. Важливо також враховувати підтримку та подальший розвиток програмного продукту. PyCharm, з іншого боку, допомагає розробникам підтримувати і покращувати код, що входить у склад програмного забезпечення. Інтегрованість PyCharm з Git дозволяє розробникам ефективно керувати кодом, перевіряти його на відповідність стандартам та робити зміни з використанням усіх переваг системи контролю версій.

Разом з тим, для ефективного управління програмним забезпеченням необхідно мати зручний інструмент для спільної роботи над проектом та зберігання коду. GitLab і PyCharm, використовуючи разом, створюють потужну інфраструктуру для розробки програмного забезпечення, яка забезпечує надійність, доступність та ефективність управління проектами. Це дозволяє розробникам сконцентруватися на створенні високоякісного програмного забезпечення і забезпечується швидкий і стабільний цикл розробки.

Розглянувши можливості GitLab та PyCharm, можна зрозуміти, як ці інструменти сприяють ефективному розгортанню програмного забезпечення та подальшому управлінню ним. GitLab, зокрема, надає зручний інтерфейс для зберігання коду, відстеження змін та спільної роботи над проектом. За допомогою GitLab можна ефективно керувати різними версіями коду та легко

співпрацювати з іншими розробниками, які можуть вносити свій внесок у проект.

Отже, використання сучасних інструментів, таких як GitLab та PyCharm, дозволяє ефективно керувати розробкою програмного забезпечення, забезпечує його надійність та доступність, а також полегшує спільну роботу над проектом.

ВИСНОВКИ

В результаті виконання дипломного проекту було спроектовано та реалізовано програмний застосунок для моніторингу стану лікарських засобів. Основною метою проекту було покращення процесу управління лікарськими засобами шляхом впровадження прогнозування попиту на медикаменти.

В якості середовища розробки було обрано PyCharm, що забезпечило зручну та ефективну розробку коду. Для зберігання даних про лікарські засоби та їхні рухи було використано базу даних SQLite, яка забезпечує легкість в налаштуванні та використанні. Прогнозування попиту лікарських засобів здійснювалося за допомогою моделі Random Forest Regressor.

Після реалізації застосунку він був протестований на різних пристроях з операційною системою Windows, щоб переконатися в його коректній роботі на різних конфігураціях. Це включало тестування інтерфейсу, функціональності та продуктивності додатку.

Актуальність роботи полягає в наступному: реалізовано можливість прогнозування потреби лікарських засобів на основі історичних даних за допомогою моделі Random Forest Regressor. Використання цього методу для прогнозування попиту дозволило підвищити точність та ефективність управління запасами лікарських засобів.

Впровадження даного застосунку можливе у лікарнях, аптеках та інших медичних установах, де необхідно забезпечити точний облік та управління лікарськими засобами. Це дозволить зменшити витрати на утримання запасів, уникнути перевищення термінів придатності та забезпечити безперебійне постачання необхідних медикаментів.

Науково-технічна та соціально-економічна значущість роботи полягає в оптимізації управління лікарськими засобами, що в свою чергу сприяє покращенню якості медичних послуг та зменшенню витрат на утримання запасів.

Всі поставлені в дипломному проектуванні задачі були успішно вирішені. Вибір алгоритмів показав доцільність використання сучасних методів розробки для моніторингу та прогнозування попиту на основі історичних даних. Перспективи подальших оновлювань полягають у вдосконаленні алгоритмів прогнозування, розширенні функціональності застосунку та його адаптації для використання в різних медичних установах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SQLite Documentation. SQLite Home Page. URL: https://www.sqlite.org/docs.html (дата звернення: 28.04.2024).
2. АНР-Аптека - програмне забезпечення для автоматизації аптек. АНР-Аптека - програмне забезпечення для автоматизації аптек. URL: https://anr.ua/ (дата звернення: 30.04.2024).
3. Data Validation: Types, Benefits, and Accuracy Process. Atlan | Third-Gen Data Catalog. URL: https://atlan.com/what-is-data-validation/#:~:text=Data%20validation%20is%20a%20crucial%20process%20in%20data,format%20specifications,%20to%20verify%20its%20integrity%20and%20correctness. (дата звернення: 08.05.2024).
4. Machine Learning Random Forest Algorithm - Javatpoint. www.javatpoint.com. URL: https://www.javatpoint.com/machine-learning-random-forest-algorithm (дата звернення: 09.05.2024).
5. Layered Architecture | Baeldung on Computer Science. Baeldung on Computer Science. URL: https://www.baeldung.com/cs/layered-architecture#:~:text=Layered%20architectures%20are%20said%20to%20be%20the%20most,function%20together%20as%20a%20single%20unit%20of%20software. (дата звернення: 11.05.2024).
6. The Project - PySimpleGUI Documentation. The Project - PySimpleGUI Documentation. URL: https://docs.pysimplegui.com/en/latest/ (дата звернення: 12.05.2024).
7. User Guide. scikit-learn. URL: https://scikit-learn.org/dev/user_guide.html (дата звернення: 12.05.2024).
8. pandas documentation – pandas 2.2.2 documentation. pandas - Python Data Analysis Library. URL: https://pandas.pydata.org/docs/ (дата звернення: 12.05.2024).

9. Get started with GitLab CI/CD | GitLab. GitLab Documentation. URL: https://docs.gitlab.com/ee/ci/ (дата звернення: 14.05.2024).

ДОДАТОК ЗВІТ ПОДІБНОСТІ

UNICHECK
by Turnitin

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1016333549

Дата перевірки:
08.06.2024 03:24:47 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
08.06.2024 07:15:39 EEST

ID користувача:
76913

Назва документа: **ІТ-01_Гончаренко_ПЗ**

Кількість сторінок: **46** Кількість слів: **6210** Кількість символів: **51149** Розмір файлу: **816.51 KB** ID файлу: **1016133745**

16.4%
Схожість

Найбільша схожість: **5.56%** з джерелом з Бібліотеки (ID файлу: **1016110951**)

2.95% Джерела з Інтернету	192	Сторінка 48
16.3% Джерела з Бібліотеки	266	Сторінка 49

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	1

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ПРОГРАМНИЙ ЗАСТОСУНОК МОНІТОРИНГУ СТАНУ ЛІКАРСЬКИХ ЗАСОБІВ

Текст програми

КПІ.ІТ-0105.045430.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Ірина ВІТКОВСЬКА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Дмитро ГОНЧАРЕНКО

Київ – 2024

Посилання на репозиторій з повним текстом програмного коду
https://gitlab.com/Gon4apenko/diplomawork/-/tree/main

Файл main_layout.py
Реалізація функціональної вимоги перегляду доступних ліків

```
import PySimpleGUI

tnr = ("Times New Roman", 14)


def main_layout():
    gui = [
        [
            PySimpleGUI.Text("Назва", font=tnr),
            PySimpleGUI.Input("", key="name", enable_events=True, font=tnr),
            PySimpleGUI.Checkbox(
                "Термін придатності",
                default=True,
                key="expired",
                enable_events=True,
                font=tnr,
            ),
            PySimpleGUI.Checkbox(
                "Немає в запасі",
                default=False,
                key="o_stock",
                enable_events=True,
                font=tnr,
            ),
            PySimpleGUI.Checkbox(
                "Наявні в запасі",
                default=True,
                key="i_stock",
                enable_events=True,
                font=tnr,
            ),
        ],
        [
            PySimpleGUI.Table(
                values=[],
                headings=[
                    "Назва",
                    "Дозування",
                    "Одиниця виміру",
                    "Термін придатності",
                    "Кількість",
                    "Тип",
                    "№ Накладної",
                    "Запас",
                ],
                key="catalogue",
                font=tnr,
                col_widths=[15, 10, 7, 9, 11, 10, 10, 12],
                justification="left",
                auto_size_columns=True,
                num_rows=1000,
                alternating_row_color=PySimpleGUI.theme_button_color()[1],
                selected_row_colors="black on white",
            )
        ],
        [
```

```
            PySimpleGUI.Button("Новий трансфер", key="n_tr", font=tnr),
            PySimpleGUI.Button("Оновити засіб", key="ch_dr", font=tnr),
            PySimpleGUI.Button("Додати засіб", key="n_dr", font=tnr),
            PySimpleGUI.Button("Згенерувати звіт", key="rep", font=tnr),
        ],
    ]
    return gui
```

Файл search_utils.py

Реалізація функціональної вимоги пошук лікарських засобів

```
import diploma.sql_utils as sql_utils
from diploma.parse_regex import separate_digits


def search(connection, form, event, values):
    a = connection.cursor()

    t_find = values["name"]

    col = []
    if t_find:
        col += [query_name_str(t_find)]
        pass

    if not values["exp"]:
        col += [d_n_exp()]

    if not values["o_stock"] and values["i_stock"]:
        col += [i_stock()]
    elif values["o_stock"] and not values["i_stock"]:
        col += [o_stock()]
    elif not values["o_stock"] and not values["i_stock"]:
        col += [ch_err()]

    if col:
        col_str = " AND ".join(col)
        string = " ".join(
            [get_all(), "WHERE", col_str, get_name()]
        )
    else:
        string = " ".join([get_all(), get_name()])

    print(string)

    a.execute(string)
    rows = a.fetchall()
    a.close()
    return rows


def query_name_str(t_find):
    return f"name LIKE '{t_find}%'"


def get_name():
    return f"ORDER BY name ASC"


def d_exp():
    return f"expiration < date('now')"
```

```
def d_n_exp():
    return f"expiration >= date('now')"


def o_stock():
    return f"current_stock = 0"


def i_stock():
    return f"current_stock > 0"


def get_all():
    return f"SELECT * FROM drugs"


def get_and():
    return f"AND"


def ch_err():
    return f"1 = 0"


def get_all_drugs(connection, form=None, event=None, values=None):
    a = connection.cursor()
    a.execute(f"SELECT * FROM drugs ORDER BY name ASC")
    rows = a.fetchall()
    a.close()
    return rows


def d_tbl(form, rows=[]):
    tbl = [row[0:-1] + (separate_digits(str(row[-1])),) for row in rows]

    form["catalogue"].update(values=tbl)


def get_drug(connection, form):
    drug_id = sql_utils.get_last_row_id(connection, "drugs")
    drug = sql_utils.get_row(connection, "drugs", drug_id)
    drug_dict = sql_utils.parse_drug(connection, "drugs", drug)
    form["name"].update(value=drug_dict["name"])
    form["o_stock"].update(value=True)
    form.value("name", drug_dict["name"])
```

Файл sql_utils.py

Реалізація функціональних вимог додавання/оновлення лікарських засобів

```
from datetime import datetime, date
import pandas as pd


def dr_add(
    conn, name, quantity, portion, exp, element, type, track, stock=0
):
    c = conn.cursor()
    c.execute(
        "INSERT INTO drugs (name, dose, units, expiration, pieces_per_box,
type, lote, current_stock) VALUES (?, ?, ?, ?, ?, ?, ?, ?)",
        (name, name, quantity, portion, exp, element, type, track, stock),
    )
    c.close()
```

```
    conn.commit()


def dr_upd(
    conn,
    drug_id,
    name,
    quantity,
    portion,
    exp,
    element,
    d_class,
    track,
    stock=None,
    d_last_inv=None,
):
    a = conn.cursor()
    a.execute(
        "UPDATE drugs SET  name=?, quantity=?, portion=?, exp=?, element=?,
type=?, track=? WHERE id=?",
        (name, quantity, portion, exp, element, type, track, drug_id),
    )
    if stock is not None:
        a.execute("UPDATE drugs SET  stock=? WHERE id=?", (stock, drug_id))
    if d_last_inv is not None:
        a.execute(
            "UPDATE drugs SET  d_last_inv=? WHERE id=?",
            (d_last_inv, drug_id),
        )
    a.close()
    conn.commit()


def tr_add(
    conn,
    d_mov,
    dest,
    p_transferred,
    transfer_class,
    sign,
    drug_id,
):
    a = conn.cursor()
    a.execute(
        """INSERT INTO inventory (d_mov, dest, p_transferred, transfer_class,
sign, drug_id)
                  VALUES (?, ?, ?, ?, ?, ?)""",
        (
            d_mov,
            dest,
            p_transferred,
            transfer_class,
            sign,
            drug_id,
        ),
    )
    a.close()
    conn.commit()


def tr_upd(
    conn,
    d_mov,
    dest,
```

```
    p_transferred,
    transfer_class,
    sign,
    drug_id,
    transfer_id,
):
    a = conn.cursor()
    a.execute(
        "UPDATE inventory SET d_mov=?, dest=?, p_transferred=?,
transfer_class=?, sign=? WHERE id=?",
        (
            d_mov,
            dest,
            p_transferred,
            transfer_class,
            sign,
            transfer_id,
        ),
    )
    a.close()
    conn.commit()


def fr_id(connection, tbl_name):
    a = connection.cursor()
    a.execute("SELECT id FROM {} LIMIT 1".format(tbl_name))
    row = a.fetchone()
    a.close()
    if row is not None:
        return row[0]
    else:
        return None


def lr_id(connection, tbl_name):
    a = connection.cursor()
    a.execute(f"SELECT MAX(id) FROM {tbl_name}")
    row = a.fetchone()[0]
    a.close()
    return row


def tbl_col(connection, tbl_name):
    a = connection.cursor()
    a.execute("tbl_info()".format(tbl_name))
    columns = [row[1] for row in a.fetchall()]
    a.close()
    return columns


def f_row(connection, tbl_name, id):
    c = connection.cursor()
    c.execute(f"SELECT * FROM {tbl_name} WHERE id = ?", (id,))
    row = c.fetchone()
    c.close()
    return row


def tr_parse(connection, tbl_name, row):
    col = tbl_col(connection, tbl_name)
    return tr_parse(row, col)


def dr_row(row, columns):
```

```
    row_dict = dict(zip(columns, row))
    row_dict["exp"] = datetime.strptime(
        row_dict["exp"], "%Y-%m-%d"
    ).date()
    row_dict["d_last_inv"] = datetime.strptime(
        row_dict["d_last_inv"], "%Y-%m-%d"
    ).date()
    return row_dict


def d_parse(conn, tbl_name, row):
    columns = tbl_col(conn, tbl_name)
    return dr_row(row, columns)


def a_rows(connection, tbl_name):
    a = connection.cursor()
    a.execute("SELECT * FROM {}".format(tbl_name))
    rows = a.fetchall()
    a.close()
    return rows


def a_transfers(conn):
    inventory = a_rows(conn, "inventory")
    columns = tbl_col(conn, "inventory")
    dataframe = pd.DataFrame(inventory, columns=columns)
    dataframe["d_mov"] = dataframe["d_mov"].apply(
        lambda x: datetime.strptime(x, "%Y-%m-%d").date()
    )
    dataframe["entry_datetime"] = dataframe["entry_datetime"].apply(
        lambda x: datetime.strptime(x, "%Y-%m-%d %H:%M:%S")
    )
    return dataframe
```

Файл utils.py

Реалізація функціональних вимог додавання/оновлення лікарських засобів

```
import main_layout as layouts
import PySimpleGUI as sg
import drug_utils as drugs_win_utils
import time
from parse_regex import (
    non_negative_int
)


def s_med(form_data, event, input_values, conn,drug_id=None):
    if not validate_entries(form_data, input_values):
        return False

    name = san_str(input_values["name"])
    quantity = input_values.get("quantity") or extract_quantity(name)
    portion = input_values.get("portion") or extract_portion(name)

    if drug_id:
        dr_upd(conn, drug_id, name, quantity, portion, input_values)
    else:
        add_drug(conn, name, quantity, portion, input_values)

    return True
```

```
def fill_medication_form(form, medication_data):
    field_mapping = {
        "name": medication_data.get("name", ""),
        "quantity": medication_data.get("quantity", ""),
        "portion": medication_data.get("portion", ""),
        "exp": medication_data.get("exp", 0),
        "element": medication_data.get("element", 1),
        "type": medication_data.get("type", ""),
        "track": medication_data.get("track", ""),
    }

    for key, value in field_mapping.items():
        form[key].update(value=value)


def check_entries(form, values):
    if (
            not values["name"]
            or not values["exp_date"]
            or not non_negative_int(values["element"])
    ):
        sg.popup("Заповніть всі поля правильно")
        return False
    return True


def drug_f(db_conn, drug=None, timeout=None):
    layout = layouts.gn_dr_layout()
    form = sg.Window("Drug", layout)
    form.finalize()

    drug_id = drug["id"] if drug else None

    start_time = time.time()
    while True:
        event, values = form.read(timeout=100)

        if event == sg.WIN_CLOSED or event == "close":
            break
        elif event == "save":
            if drugs_win_utils.s_med(
                    form, event, values, db_conn, id=drug_id
            ):
                break

        if timeout and time.time() - start_time > timeout:
            break

    form.close()
```

Файл transfer_utils.py

Реалізація функціональної вимоги додавання нового трансферу

```
import time
from datetime import datetime

import PySimpleGUI as sg

import diploma.sql_utils as sql_utils
import new_transfer_layout
import transfer_utils
```

```
from diploma.parse_regex import non_negative_int


def new_transfer(
    prev_stock,
    p_transferred,
    transfer_class,
    d_mov,
    d_last_inv,
    ch_inv=0,
):
    pres_stock = prev_stock
    if d_mov > d_last_inv:
        if transfer_class == "inventory":
            d_last_inv = d_mov
            pres_stock = p_transferred
            ch_inv = pres_stock
        elif transfer_class == "entry":
            pres_stock = prev_stock + p_transferred
        elif transfer_class == "exit":
            pres_stock = prev_stock - p_transferred
    else:
        pres_stock = prev_stock
    return max(0.0, pres_stock), d_last_inv, ch_inv


def st_upd(
    conn,
    p_transferred,
    d_mov,
    transfer_class,
    drug_id,
):

    drug = sql_utils.f_row(conn, "drugs", drug_id)
    d_drug = sql_utils.d_parse(conn, "drugs", drug)

    d_transf = datetime.strptime(d_mov, "%Y-%m-%d").date()

    new_stock, d_last_inv, inv_last = new_transfer(
        d_drug["st_present"],
        p_transferred,
        transfer_class,
        d_transf,
        d_drug["d_last_inv"],
    )
    d_drug["st_present"] = new_stock
    d_drug["d_last_inv"] = d_last_inv

    sql_utils.update_drug(
        conn=conn,
        drug_id=drug_id,
        name=d_drug["name"],
        quantity=d_drug["quantity"],
        portion=d_drug["portion"],
        exp=d_drug["exp"],
        element=d_drug["element"],
        type=d_drug["type"],
        track=d_drug["track"],
        stock=d_drug["stock"],
        d_last_inv=d_drug["d_last_inv"],
    )
```

```
def ch_entr(form, values):
    err_message = ""
    if values["d_trans"] == "":
        err_message += "\nОберіть дату"
    if values["transfer_class"] == "":
        err_message += "\nОберіть тип трансферу"
    if not non_negative_int(values["b_transferred"]):
        err_message += f"\nКількість упаковок повинна бути більше 0"
    if not non_negative_int(values["p_transferred"]):
        err_message += f"\nКількість повинна бути більше 0"

    if err_message != "":
        sg.popup(err_message)
        return False
    return True


def save_move(form, event, values, connection, drug, movement_id):
    global transfer_class
    if not ch_entr(form, values):
        return False

    if values["mov_type"] == "Отримання":
        transfer_class = "entry"
    elif values["mov_type"] == "Трансфер":
        transfer_class = "exit"
    elif values["mov_type"] == "Інвентар":
        transfer_class = "inventory"

    p_transferred = p_all_transferred(form, values, drug)

    if transfer_id:
        sql_utils.update_movement(
            conn=connection,
            d_mov=values["d_mov"],
            dest=values["dest"],
            p_transferred=p_transferred,
            transfer_class=transfer_class,
            sign=values["sign"],
            transfer_id=transfer_id,
        )
    else:
        sql_utils.add_movement(
            conn=connection,
            d_mov=values["d_mov"],
            dest=values["dest"],
            p_transferred=p_transferred,
            transfer_class=transfer_class,
            sign=values["sign"],
            drug_id=drug["id"],
        )

    st_upd(
        connection,
        p_transferred,
        values["d_mov"],
        transfer_class,
        drug["id"],
    )

    return True


def f_transfer(
```

```
    form,
    d_mov="",
    dest="",
    n_box=0,
    n_elements=0,
    mov_type="",
    sign="",
):
    form["d_mov"].update(value=d_mov)
    form["dest"].update(value=orig_dest)
    form["b_transferred"].update(value=n_box)
    form["p_transferred"].update(value=n_pieces)
    form["transfer_class"].update(mov_type)
    form["sign"].update(value=sign)


def f_dr(
    form,
    drug_name="",
    quantity="",
    portion="",
    exp="",
    element="",
    type="",
    track="",
):
    form["name"].update(value=drug_name)
    form["quantity"].update(value=quantity)
    form["portion"].update(value=portion)
    form["exp"].update(value=exp)
    form["element"].update(value=element)
    form["type"].update(value=type)
    form["track"].update(value=track)


def f_wind(
    form,
    drug_name="",
    quantity="",
    portion="",
    exp="",
    element="",
    type="",
    track="",
    d_mov="",
    dest="",
    n_box=0,
    n_elements=0,
    transfer_class="",
    sign="",
):
    f_transfer(
        form,
        dmov=d_mov,
        dest=dest,
        n_box=n_box,
        n_elements=n_elements,
        transfer_class=transfer_class,
        sign=sign,
    )

    f_dr(
        form,
        drug_name=drug_name,
```

```
        quantity=quantity,
        portion=portion,
        exp=exp,
        element=element,
        type=type,
        track=track,
    )


def el_transferred(form, values, drug):
    if values["p_transferred"].isdigit() and int(values["p_transferred"]) >
0:
        p_transferred = int(values["p_transferred"])
    else:
        p_transferred = 0
    if values["-boxes_moved-"].isdigit() and int(values["b_transferred"]) >
0:
        b_transferred = int(values["b_transferred"])
    else:
        b_transferred = 0

    tot_pieces_moved = p_transferred + b_transferred * drug["p_transferred"]
    return tot_elements_moved


def transfer_ses(
    db_connection, drug, inventory=None,timeout=None
):

    layout = new_transfer_layout.n_transf_layout()
    window = sg.Window("Drug", layout)
    window.finalize()

    transfer_id = None
    if inventory:
        transfer_utils.fill_win(
            window=window,
            drug_name=drug["name"],
            quantity=drug["dose"],
            portion=drug["units"],
            exp=drug["expiration"],
            element=drug["element"],
            type=drug["type"],
            track=drug["track"],
            d_mov=inventory["d_mov"],
            dest=inventory["dest"],
            n_box=0,
            n_elements=inventory["p_transferred"],
            transfer_class=inventory["transfer_class"],
            sign=inventory["sign"],
        )
        transfer_id = inventory["id"]
    else:
        transfer_utils.f_dr(
            window=window,
            drug_name=drug["name"],
            quantity=drug["quantity"],
            portion=drug["portion"],
            exp=drug["exp"],
            element=drug["element"],
            type=drug["type"],
            track=drug["track"],
        )
```

```
    tstart = time.time()
    while True:
        event, values = window.read(timeout=100)
        if event == sg.WIN_CLOSED:
            break
        elif event == "b_save":
            if transfer_utils.save_move(
                window, event, values, db_connection, drug, transfer_id
            ):
                break
        elif event == "b_close":
            break

        tot_pieces_moved(window, values, drug)

        if timeout:
            if time.time() - tstart > timeout:
                break


    window.close()
```

Файл demand_forecast.py

Реалізація функціональної вимоги прогнозування попиту

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import sqlite3
from datetime import datetime

def fetch_historical_data(database_path):
    conn = sqlite3.connect(database_path)
    query = "SELECT date_movement, pieces_moved FROM movements WHERE
movement_type='exit'"
    df = pd.read_sql_query(query, conn)
    conn.close()
    return df

def preprocess_data(df):
    df['d_mov'] = pd.to_datetime(df['d_mov'])
    df.set_index('d_mov', inplace=True)
    df = df.resample('M').sum()
    df['month'] = df.index.month
    df['year'] = df.index.year
    return df

def train_model(df):
    X = df[['month', 'year']]
    y = df['p_transferred']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    print(f"Model Mean Squared Error: {mse}")

    return model
```

```
def forecast_demand(model, month, year):
    X_pred = pd.DataFrame({'month': [month], 'year': [year]})
    return model.predict(X_pred)

def main(database_path, start_date, end_date):
    df = fetch_historical_data(database_path)
    df = preprocess_data(df)
    model = train_model(df)

    forecast_results = []
    date_range = pd.date_range(start=start_date, end=end_date, freq='M')
    for current_date in date_range:
        forecast = forecast_demand(model, current_date.month,
current_date.year)
        forecast_results.append(forecast[0])

    print(f"Прогнозування попиту від {start_date} до {end_date}:
{forecast_results}")
    return date_range, forecast_results

def save_forecast_plot(date_range, forecast_results, save_path):
    import matplotlib.pyplot as plt
    fig, ax = plt.subplots()
    ax.plot(date_range, forecast_results, marker='o')
    ax.set_title('Forecasted Demand')
    ax.set_xlabel('Date')
    ax.set_ylabel('Portion')
    fig.savefig(save_path)
    plt.close(fig)
```

Файл reports_gen.py

Реалізація функціональної вимоги генерація звіту

```
      import diploma.sql_utils as sql_utils
from datetime import date
import os
from tabulate import tabulate
import pandas as pd
import diploma.reports_utils as reports_utils
import datetime


BASE_DIR = "reports"


if not os.path.exists(BASE_DIR):
    os.makedirs(BASE_DIR)


def add_stock(df):


    df.sort_values(by=["d_mov"], inplace=True)


    df["stk_new"] = 0
    df["d_last_inv"] = date(2024, 1, 1)
```

```
    df["last_inv_stock"] = 0

    prev_row = None
    indexes = df.index
    for index in indexes:
        row = df.loc[index]
        if prev_row is None:
            prev_row = row

        stk_new, d_last_inv, last_inv_stock = n_stk(
            old_stock=prev_row["stk_new"],
            p_transferred=row["p_transferred"],
            transfer_class=row["transfer_class"],
            d_mov=row["d_mov"],
            d_last_inv=prev_row["d_last_inv"],
            last_inv_stock=prev_row["last_inv_stock"],
        )

        df.loc[index, "stk_new"] = stk_new
        df.loc[index, "d_last_inv"] = d_last_inv
        df.loc[index, "last_inv_stock"] = last_inv_stock

        prev_row = df.loc[index]

    return df


def add_us_stock(df, groupby_cols):

    grouped_df = df.groupby(groupby_cols)
    df["stk_new"] = 0
    df["d_last_inv"] = date(2024, 1, 1)

    for group_name, group_df in grouped_df:
        indices = group_df.index
        group_df = add_stock(group_df.copy())

        df.loc[indices, "d_last_inv"] = group_df.loc[
            indices, "d_last_inv"
        ]
        df.loc[indices, "stk_new"] = group_df.loc[
            indices, "stk_new"
        ]
```

```
        df.loc[indices, "last_inv_stock"] = group_df.loc[
            indices, "last_inv_stock"
        ]

    return df


def save_mov(
    df,
    folder_path,
    file_name="mov_per_ID.txt",
    mask_col=None,
    labels=None,
    groupby_cols=[
        "drug_id",
    ],
):
    file_path = os.path.join(folder_path, file_name)
    df.sort_values(by=["name"], inplace=True)

    with open(file_path, "w") as f:
        grouped_df = df.groupby(groupby_cols)

        groups_sorted = grouped_df.groups.keys()
        groups_sorted = sorted(groups_sorted)

        df_reordered = pd.concat(
            [grouped_df.get_group(group) for group in groups_sorted]
        )

        for group_name, group_df in df_reordered.groupby(groupby_cols):
            if mask_col is not None:
                group_df = group_df[mask_col]
            if labels is not None:
                if len(labels) == len(group_df.columns):
                    group_df.columns = labels

            group_df.sort_values(by=["d_mov"], inplace=True)
            table = tabulate(group_df, headers="keys", tablefmt="psql")
            f.write(table)
            f.write("\n\n")
```

```
def save_usage_ID(
    df_drug,
    usage_ID,
    folder_path,
    mask=None,
    file_name="usage_ID.txt",
    col_mask_drug=None,
    col_mask_mov=None,
):
    df_drug.sort_values(by=["name"], inplace=True)

    if mask is not None:
        usage_ID = usage_ID[mask]

    if col_mask_drug is None:
        col_mask_drug = df_drug.columns

    if col_mask_mov is None:
        col_mask_mov = usage_ID.columns

    out_path = os.path.join(folder_path, file_name)

    with open(out_path, "w") as f:
        for index, row in df_drug.iterrows():
            df_consumption_drug_id = df_usage_ID[
                df_usage_ID["drug_id"] == index
            ]
            row_df = pd.DataFrame([row], columns=df_drug.columns)
            table_row = tabulate(
                row_df[col_mask_drug],
                headers="keys",
                tablefmt="simple",
                showindex=False,
            )
            f.write(table_row)
            f.write("\n")
            table_mov = tabulate(
                df_consumption_drug_id[col_mask_mov],
                headers="keys",
                tablefmt="psql",
                showindex=False,
            )
            f.write(table_mov)
```

```
            f.write("\n\n")


def save_usage(
    df,
    folder_path,
    mask=None,
    labels=None,
    file_name="usage_per_ID.xlsx",
):
    if mask is not None:
        df = df[mask]
    if labels is not None:
        if len(labels) == len(df.columns):
            df.columns = labels

    out_path = os.path.join(folder_path, file_name)

    writer = pd.ExcelWriter(out_path, engine="tblxls")

    sheet_name = "Витрати"
    df.to_excel(writer, index=False, sheet_name=sheet_name)

    format(df, writer, sheet_name=sheet_name)

    writer.close()


def save_dataset(
    df_drugs,
    df_movs,
    folder_path,
    file_name,
):
    df_drugs["drug_id"] = df_drugs.index
    df_merged = pd.merge(df_movs, df_drugs, on="drug_id", how="outer")

    out_path = os.path.join(folder_path, file_name)

    writer = pd.ExcelWriter(out_path, engine="tblxls")

    sheet_name = "Повна інформація"
    df_merged.to_excel(writer, index=False, sheet_name=sheet_name)
```

```
    format_xlsx(df_merged, writer, sheet_name=sheet_name)

    writer.close()

    pass


def usage_stock(
    df_cum,
    end_date=date(2040, 1, 1),
    groupby_cols=[
        "drug_id",
    ],
):

    output_cols = groupby_cols + [
        "stock",
        "d_last_inv",
        "last_inventory_stock",
    ]

    df_out = pd.DataFrame([], columns=output_cols)

    df_cum = df_cum[df_cum["d_mov"] <= end_date]

    if df_cum.empty:
        return df_out

    df_cum_grouped = df_cum.groupby(groupby_cols)

    for group_name, group_df in df_cum_grouped:
        latest_mov_date = group_df["d_mov"].max()
        latest_date = group_df["d_last_inv"].max()
        inv_last_date = group_df.loc[
            (group_df["d_mov"] == latest_date)
            & (group_df["transfer_class"] == "inventory"),
            "stk_new",
        ].min()
        idx_latest_mov_date = group_df[
            group_df["d_mov"] == latest_mov_date
        ].index
        max_entry_datetime = group_df.loc[idx_latest_mov_date,
```

```
"d_entry"].max()
        stock = group_df.loc[
            (group_df["d_mov"] == latest_mov_date)
            & (group_df["d_entry"] == max_entry_datetime),
            "stk_new",
        ].min()

        df_tmp = pd.DataFrame(
            {
                "stock": stock,
                "d_last_inv": latest_date,
                "last_inv_stock": inv_last_date,
            },
            index=[0],
        )

        for col in groupby_cols:
            df_tmp[col] = group_name[groupby_cols.index(col)]

        df_out = pd.concat([df_out, df_tmp], ignore_index=True)

    return df_out


def calc_usage_group(
    df_cum,
    start_date=date(2024, 1, 1),
    end_date=date(2040, 1, 1),
    groupby_cols=[
        "drug_id",
    ],
):
    output_cols = groupby_cols + [
        "entry",
        "exit",
        "stock",
        "d_last_inv",
        "last_inv_stock",
    ]

    df_out = pd.DataFrame([], columns=output_cols)

    df_cum = df_cum[
```

```
        (df_cum["d_mov"] >= start_date) & (df_cum["d_mov"] <= end_date)
    ]

    if df_cum.empty:
        return df_out

    df_cum["entry"] = df_cum.apply(
        lambda x: x["p_transferred"] if x["mov_type"] == "entry" else 0,
axis=1
    )
    df_cum["exit"] = df_cum.apply(
        lambda x: x["p_transferred"] if x["mov_type"] == "exit" else 0,
axis=1
    )
    df_cum["inventory"] = df_cum.apply(
        lambda x: x["p_transferred"] if x["mov_type"] == "inventory" else 0,
axis=1
    )

    df_cum_grouped = df_cum.groupby(groupby_cols)

    for group_name, group_df in df_cum_grouped:
        entry = group_df["entry"].sum()
        exit = group_df["exit"].sum()
        latest_mov_date = group_df["d_mov"].max()
        latest_last_inventory_date = group_df["d_last_inv"].max()

        inventory_on_last_inventory_date = group_df.loc[
            (group_df["d_mov"] == latest_last_inventory_date)
            & (group_df["mov_type"] == "inventory"),
            "stk_new",
        ].min()

        idx_latest_mov_date = group_df[
            group_df["d_mov"] == latest_mov_date
        ].index

        max_entry_datetime = group_df.loc[idx_latest_mov_date,
"d_entry"].max()
        stock = group_df.loc[
            (group_df["d_mov"] == latest_mov_date)
            & (group_df["d_entry"] == max_entry_datetime),
            "stk_new",
```

```
        ].min()

        df_tmp = pd.DataFrame(
            {
                "entry": entry,
                "exit": exit,
                "stock": stock,
                "d_last_inv": latest_last_inventory_date,
                "last_inv_stock": inventory_on_last_inventory_date,
            },
            index=[0],
        )

        for col in groupby_cols:
            df_tmp[col] = group_name[groupby_cols.index(col)]

        df_out = pd.concat([df_out, df_tmp], ignore_index=True)

    return df_out


def cr_folders(base_folder_path=BASE_DIR):
    today = datetime.date.today()
    year_folder = os.path.join(base_folder_path, str(today.year))
    month_folder = os.path.join(year_folder, str(today.month))
    day_folder = os.path.join(month_folder, str(today.day))

    if not os.path.exists(year_folder):
        os.makedirs(year_folder)

    if not os.path.exists(month_folder):
        os.makedirs(month_folder)

    if not os.path.exists(day_folder):
        os.makedirs(day_folder)

    report_folders = [f for f in os.listdir(day_folder) if
f.startswith("ID_")]
    if report_folders:
        last_id = max([int(f.split("_")[1]) for f in report_folders])
        new_id = str(last_id + 1).zfill(2)
    else:
        new_id = "1"
```

```
    report_folder_path = os.path.join(day_folder, "ID_" + new_id)
    os.makedirs(report_folder_path)

    ID_folder_path = os.path.join(report_folder_path, "usage_ID")
    os.makedirs(ID_folder_path)

    name_folder_path = os.path.join(report_folder_path, "name_portion")
    os.makedirs(name_folder_path)

    return report_folder_path, ID_folder_path, name_folder_path


def format(df, writer, sheet_name="Sheet1"):
    (max_row, max_col) = df.shape

    workbook = writer.book
    worksheet = writer.sheets[sheet_name]

    for i, column in enumerate(df.columns):
        max_len = max(df[column].astype(str).map(len).max(), len(column))
        worksheet.set_column(i, i, max_len + 5)

    worksheet.autofilter(0, 0, max_row, max_col - 1)


def computed_cum_res(df_drugs, df_movs, groupby_cols):
    df_merged = pd.merge(df_movs, df_drugs, on="drug_id", how="left")
    return reports_utils.add_cum_stock_df(df_merged,
groupby_cols=groupby_cols)


def add_drug_info_from_ID(df_drugs, df):
    cols = [
        "drug_id",
        "name",
        "quantity",
        "portion",
        "exp",
        "element",
        "type",
        "track",
    ]
```

```
    df_merged = pd.merge(df, df_drugs[cols], on="drug_id", how="left")
    return df_merged


def save_INFO_txt(folder_path, file_name, start_date, end_date):
    out_path = os.path.join(folder_path, file_name)
    with open(out_path, "w") as f:
        f.write("Початок: {}\n".format(start_date))
        f.write("Кінець: {}\n".format(end_date))
        f.write(
            "Згенеровано: {}\n".format(
                datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            )
        )
        f.write("Згенерував: {}\n".format(os.getlogin()))


def save_usage_ID(
    db_connection,
    start_date,
    end_date,
    folder_path,
    file_name="usage_per_ID.xlsx",
):
    groupby_cols = [
        "drug_id",
    ]

    df_movs = sql_utils.get_all_movements_df(db_connection)
    df_drugs = sql_utils.get_all_drugs_df(db_connection)

    cumulative_result = reports_utils.computed_cum_res(
        df_drugs=df_drugs, df_movs=df_movs, groupby_cols=groupby_cols
    )

    df_consumption_ID = reports_utils.compute_consumption_group(
        cumulative_result,
        start_date=start_date,
        end_date=end_date,
        groupby_cols=groupby_cols,
    )

    df_out = reports_utils.add_drug_info_from_ID(df_drugs, df_consumption_ID)
```

```
    mask = [
        "name",
        "quantity",
        "portion",
        "exp",
        "element",
        "type",
        "track",
        "entry",
        "exit",
        "stock",
        "d_last_inv",
        "last_inv_stock",
    ]
    labels = [
        "name",
        "quantity",
        "portion",
        "exp",
        "element",
        "type",
        "track",
        "entry",
        "exit",
        "stock",
        "d_last_inv",
        "last_inv_stock",
    ]

    reports_utils.save_xlsx_consumption(
        df_out,
        mask=mask,
        labels=labels,
        folder_path=folder_path,
        file_name=file_name,
    )


def save_usage_type(
    db_connection,
    start_date,
    end_date,
```

```
    folder_path,
    file_name="usage_type.xlsx",
):
    groupby_cols = ["name", "quantity", "type"]

    df_movs = sql_utils.get_all_movements_df(db_connection)
    df_drugs = sql_utils.get_all_drugs_df(db_connection)

    cumulative_result = reports_utils.computed_cum_res(
        df_drugs=df_drugs, df_movs=df_movs, groupby_cols=groupby_cols
    )

    df_consumption = reports_utils.compute_consumption_group(
        cumulative_result,
        start_date=start_date,
        end_date=end_date,
        groupby_cols=groupby_cols,
    )

    mask = [
        "name",
        "quantity",
        "type",
        "entry",
        "exit",
        "stock",
        "d_last_inv",
        "last_inv_stock",
    ]
    labels = [
        "name",
        "quantity",
        "type",
        "entry",
        "exit",
        "stock",
        "d_last_inv",
        "last_inv_stock",
    ]

    reports_utils.save_xlsx_consumption(
        df_consumption,
        mask=mask,
```

```
        labels=labels,
        folder_path=folder_path,
        file_name=file_name,
    )


def gen_rep(
    db_connection,
    folder_path,
    file_name="mov_per_id.txt",
):
    groupby_cols = [
        "drug_id",
    ]
    df_movs = sql_utils.get_all_movements_df(db_connection)
    df_drugs = sql_utils.get_all_drugs_df(db_connection)

    cumulative_result = reports_utils.computed_cum_res(
        df_drugs=df_drugs, df_movs=df_movs, groupby_cols=groupby_cols
    )

    mask_col = [
        "name",
        "quantity",
        "portion",
        "exp",
        "element",
        "type",
        "trakc",
        "d_mov",
        "dest",
        "mov_type",
        "stk_new",
    ]
    labels = None
    reports_utils.save_txt_mov_group(
        cumulative_result,
        folder_path=folder_path,
        file_name=file_name,
        mask_col=mask_col,
        labels=labels,
    )
```

```
def gen_rep_type(
    db_connection,
    folder_path,
    file_name="mov_type.txt",
):
    groupby_cols = ["name", "quantity", "type"]
    df_movs = sql_utils.get_all_movements_df(db_connection)
    df_drugs = sql_utils.get_all_drugs_df(db_connection)

    cumulative_result = reports_utils.computed_cum_res(
        df_drugs=df_drugs, df_movs=df_movs, groupby_cols=groupby_cols
    )

    mask_col = [
        "name",
        "quantity",
        "type",
        "d_mov",
        "dest",
        "mov_type",
        "stk_new",
    ]
    labels = None
    reports_utils.save_txt_mov_group(
        cumulative_result,
        folder_path=folder_path,
        file_name=file_name,
        mask_col=mask_col,
        labels=labels,
        groupby_cols=groupby_cols,
    )


def save_stock(
    db_connection,
    folder_path,
    file_name="stock_per_ID.xlsx",
    end_date=date(2040, 1, 1),
):
    groupby_cols = [
        "drug_id",
    ]
```

```
df_movs = sql_utils.get_all_movements_df(db_connection)
df_drugs = sql_utils.get_all_drugs_df(db_connection)

cumulative_result = reports_utils.computed_cum_res(
    df_drugs=df_drugs, df_movs=df_movs, groupby_cols=groupby_cols
)

df_stock = reports_utils.usage_stock(
    cumulative_result,
    end_date=end_date,
    groupby_cols=groupby_cols,
)

df_out = reports_utils.add_drug_info_from_ID(df_drugs, df_stock)

mask = [
    "name",
    "quantity",
    "portion",
    "exp",
    "element",
    "type",
    "track",
    "stock",
    "d_last_inv",
    "last_inv_stock",
]
labels = [
    "name",
    "quantity",
    "portion",
    "exp",
    "element",
    "type",
    "track",
    "stock",
    "d_last_inv",
    "last_inv_stock",
]

reports_utils.save_usage(
    df_out,
    mask=mask,
```

```
        labels=labels,
        folder_path=folder_path,
        file_name=file_name,
    )


def save_stk_type_xlsx(
    db_connection,
    folder_path,
    file_name="stk_type.xlsx",
    end_date=date(2100, 1, 1),
):
    groupby_cols = ["name", "quantity", "type"]
    df_movs = sql_utils.get_all_movements_df(db_connection)
    df_drugs = sql_utils.get_all_drugs_df(db_connection)

    cumulative_result = reports_utils.computed_cum_res(
        df_drugs=df_drugs, df_movs=df_movs, groupby_cols=groupby_cols
    )

    df_stock = reports_utils.usage_stock(
        cumulative_result,
        end_date=end_date,
        groupby_cols=groupby_cols,
    )

    mask = [
        "name",
        "quantity",
        "type",
        "stock",
        "d_last_inv",
        "last_inv_stock",
    ]
    labels = [
        "name",
        "quantity",
        "type",
        "stock",
        "d_last_inv",
        "last_inv_stock",
    ]
```

```
    reports_utils.save_usage(
        df_stock,
        mask=mask,
        labels=labels,
        folder_path=folder_path,
        file_name=file_name,
    )


def full_ds(db_connection, folder_path, file_name="full_dataset.xlsx"):
    df_movs = sql_utils.get_all_movements_df(db_connection)
    df_drugs = sql_utils.get_all_drugs_df(db_connection)
    reports_utils.save_f_ds(
        df_drugs=df_drugs,
        df_movs=df_movs,
        folder_path=folder_path,
        file_name=file_name,
    )
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ПРОГРАМНИЙ ЗАСТОСУНОК МОНІТОРИНГУ СТАНУ ЛІКАРСЬКИХ ЗАСОБІВ

Програма та методика тестування

КПІ.ІТ-0105.045430.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Ірина ВІТКОВСЬКА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Дмитро ГОНЧАРЕНКО

Київ – 2024

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є програмний застосунок для моніторингу стану лікарських засобів. Цей застосунок призначений для забезпечення керування та управлінням лікарськими засобами. Основна увага в тестуванні приділяється функціональності, продуктивності та користувацькому інтерфейсу.

Тестування включає різні типи сценаріїв: від базового перегляду інформації про препарати до складніших операцій, таких як оновлення даних про запаси та адміністрування системи. Це дозволяє переконатися, що програмний застосунок надійний і працює належним чином, забезпечуючи широке охоплення користувацьких сценаріїв та стабільну роботу у реальних умовах використання.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження та оновлення даних;

– перевірка сумісності застосунку з різними операційними системами;

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально, з метою знаходження помилок та недоліків як у функціональній частині так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

– динамічне тестування на відповідність функціональним вимогам;

– тестування на виведення повідомлень про помилку, коли це необхідно;

– тестування інтерфейсу користувача;

– тестування зручності використання.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ПРОГРАМНИЙ ЗАСТОСУНОК МОНІТОРИНГУ СТАНУ ЛІКАРСЬКИХ ЗАСОБІВ

Керівництво користувача

КПІ.ІТ-0105.045430.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Ірина ВІТКОВСЬКА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Дмитро ГОНЧАРЕНКО

Київ – 2024

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«PharmaStock» - це десктопний застосунок для моніторингу стану лікарських засобів, розроблений за допомогою Python. Основне призначення програмного застосунку для моніторингу стану лікарських засобів полягає у забезпеченні зручного інтерфейсу для медичних працівників, що дозволяє ефективно відстежувати запаси лікарських препаратів, їх терміни придатності та інші параметри. Застосунок надає можливість моніторингу та управління запасами лікарських засобів, генерації звітів, а також прогнозування попиту на товари.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Мінімальна конфігурація технічних засобів:

— тип процесору: AMD Ryzen 3;

— об‘єм ОЗП: 2 Гб;

— кількість ядер процесору: 2.

Рекомендована конфігурація технічних засобів:

— тип процесору: AMD Ryzen 5;

— об‘єм ОЗП: 16 Гб;

— кількість ядер процесору: 4.

2.2 Завантаження застосунку

На даний момент застосунок можна встановити власноруч, з відповідного репозиторію. Для цього спершу необхідно завантажити Python версії 3.12.4, а потім, виконати завантаження архіву файлів з репозиторія GitLab. Або ж виконати завантаження .exe файлу з репозиторію.

2.3 Перевірка коректної роботи

По завершенню встановлення на робочому столі повинен відобразитись .exe файл. Після натискання на виконавчий файл, повинно відобразитись головне меню застосунку.

3 ВИКОНАННЯ ПРОГРАМИ

При запуску програмного застосунку користувачу буде відображено головне меню застосунку (рис. 3.1).

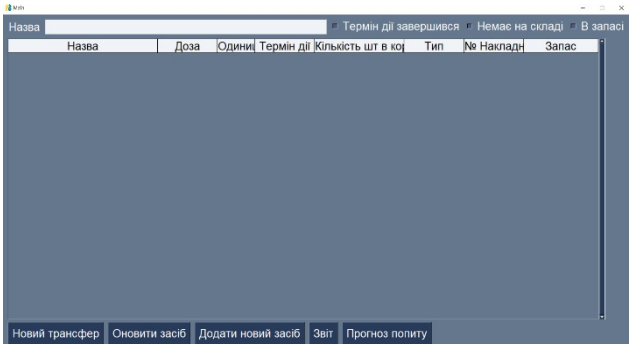


Рисунок 3.1 – Головне меню

Користувач має змогу переглянути доступні ліки. Далі, користувач може знайти лікарські засоби за допомогою поля пошуку: в відповідній комірці “Назва” вписати потрібний лікарський засіб. Якщо він є, то користувач побачить його в списку (рис.3.2).

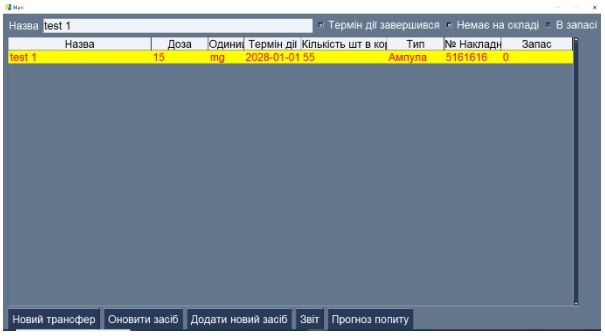


Рисунок 3.2 — Пошук за назвою

Також користувач може переглянути ліки за допомогою фільтрів “Термін дії”, “Немає на складі” та “В запасі”(рис. 3.3).

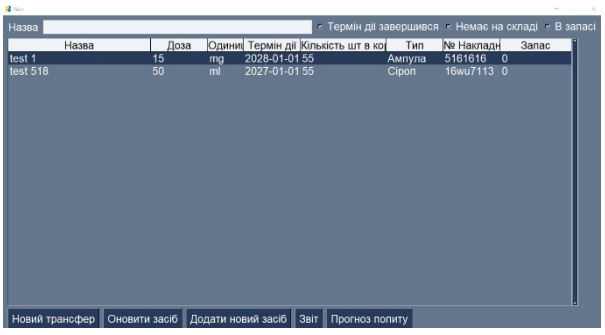


Рисунок 3.3 — Перегляд ліків за допомогою фільтрів

При натисканні на кнопку “Новий трансфер” користувач побачить нове вікно для вводу інформації. Для цього йому потрібно обрати лікарський засіб для якого він бажає додати трансфер та натиснути на відповідну кнопку. Після цього він побачить детальну інформацію про засіб та поля, які треба заповнити для додавання нового трансферу (рис. 3.4).

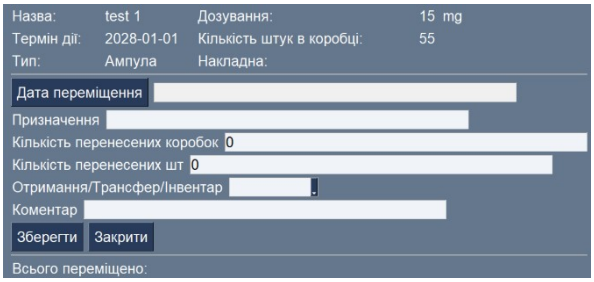


Рисунок 3.4 — Додавання нового трансферу

Для додавання нового засобу, користувачу потрібно натиснути на відповідну кнопку “Додати новий засіб”. Після заповнення всіх полів, користувач побачить результат в списку в головному меню (рис. 3.5).

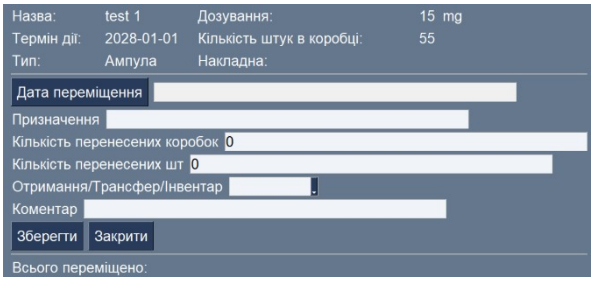


Рисунок 3.5 — Додавання нового засобу

Якщо користувач бажає оновити інформацію про лікарський засіб, йому потрібно обрати відповідний засіб у списку, і далі натиснути на відповідну кнопку “Оновити засіб”. Після цього побачить форму з інформацією про цей засіб (рис. 3.6).

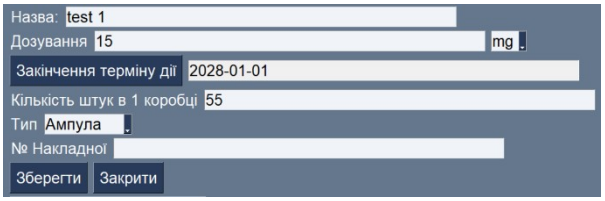


Рисунок 3.6 — Оновлення інформації про засіб

Користувач також може згенерувати звіт. Для цього натискає на відповідну кнопку “Звіт”. Далі обирає потрібні дати початку та кінця для

генерування звіту. Після цього він може перейти за локальним посиланням та передивитись інформацію (рис. 3.7).

Рисунок 3.7 — Генерування звіту

Крім цього, користувач може згенерувати попит на товари. Натиснувши на кнопку “Прогноз попиту”, користувач побачить нову форму, де він може обрати потрібні йому дати для прогнозування попиту на товари (рис.3.8).

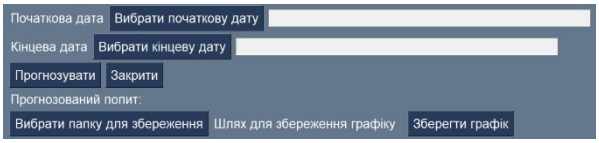


Рисунок 3.8 — Прогнозування попиту

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ПРОГРАМНИЙ ЗАСТОСУНОК МОНІТОРИНГУ СТАНУ ЛІКАРСЬКИХ ЗАСОБІВ

Графічний матеріал

КПІ.ІТ-0105.045430.06.99

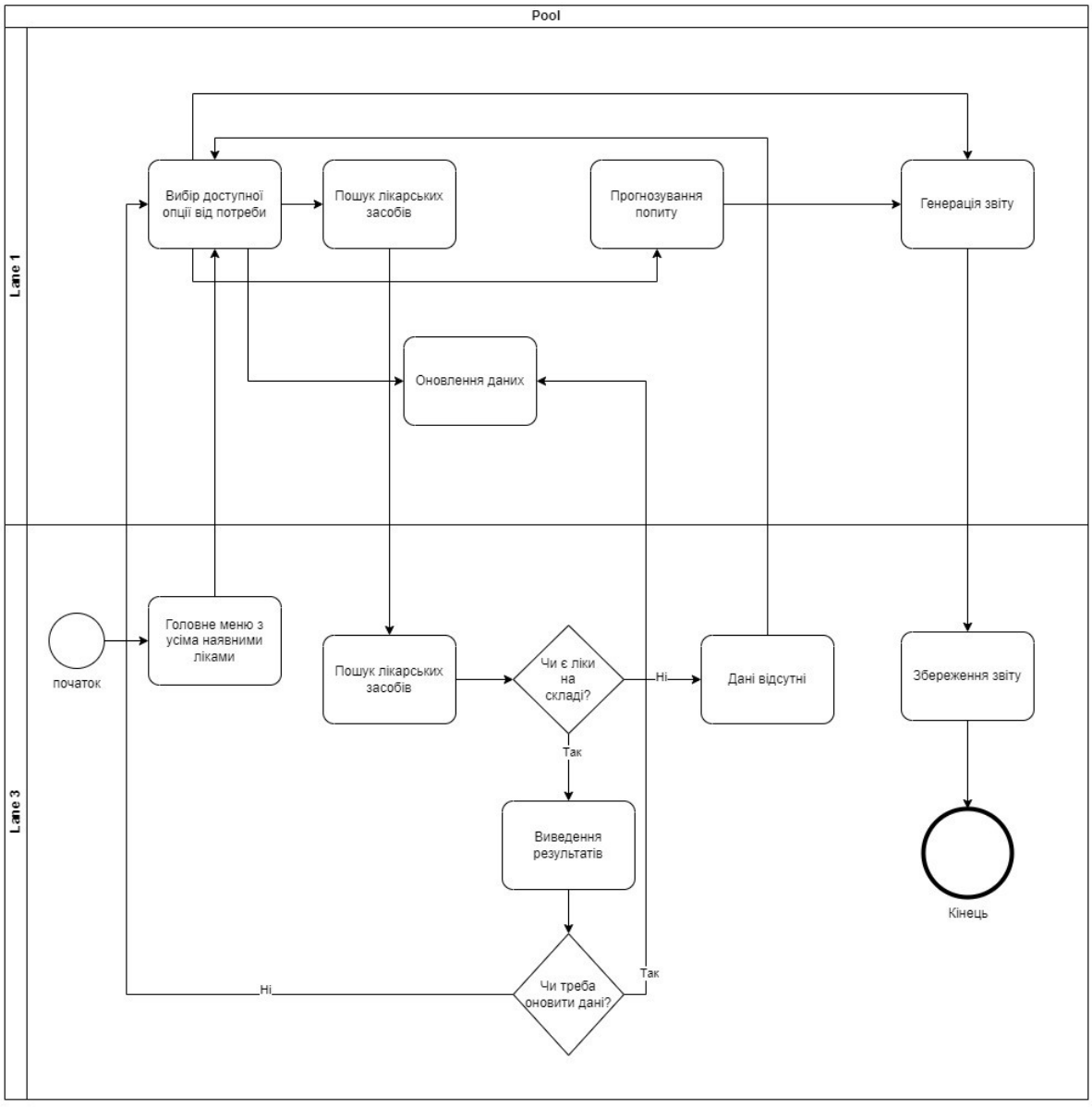
“ПОГОДЖЕНО”
Керівник проєкту:
_____ Ірина ВІТКОВСЬКА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Дмитро ГОНЧАРЕНКО

Київ – 2024

КПІ.ІТ-0105.045430.ССД



					КПІ.ІТ-0105.045430.ССД			
Зм.	Арк.	№ докум.	Підп.	Дата				
Розробив		Гончаренко Д.С.			Схема структурна діяльності	Лит.	Маса	Масштаб
Перевірив		Вітковська І.І.						
Т. контр.						Аркуш	Аркушів	
Н. контр.		Вітковська І.І.			Програмний застосунок моніторингу стану лікарських засобів	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-01		
Затвердив		Жаріков Е.В.						

КПІ.ІТ-0105.045430.ССВ

					КПІ.ІТ-0105.045430.ССВ		
						Лит.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна варіантів використань			
Розробив		Гончаренко Д.С.				Аркуш	Аркушів	
Перевірив		Вітковська І.І.						
Т. контр.					Програмний застосунок моніторингу стану лікарських засобів	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-01	
Н. контр.		Вітковська І.І.					
Затвердив		Жаріков Е.В.					

КПІ.ІТ-0105.045430.СБД

					КПІ.ІТ-0105.045430.СБД
Зм.	Арк.	№ докум.	Підп.	Дата	
Розробив		Гончаренко Д.С.			Схема бази даних
Перевірив		Вітковська І.І.			
Т. контр.					
Н. контр.		Вітковська І.І.			Програмний застосунок моніторингу стану лікарських засобів
Затвердив		Жаріков Е.В.			

Лит.	Маса	Масштаб
Аркуш	Аркушів	
КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-01		