МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

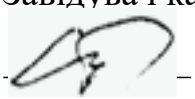
Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

 _ проф. Приходько С.Б.

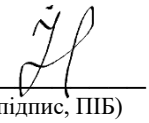
«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

Виконав: студент групи 4157ст3

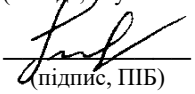
 _____ Нестер І.В.

(підпис, ПІБ)

Керівник роботи:

доцент кафедри ПЗАС, к.т.н., доцент

(посада, науковий ступень, вчене знання)

 _____ Макарова Л.М.

(підпис, ПІБ)

Миколаїв – 2026 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
«_07_» __04__ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Нестеру Івану Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»
Керівник роботи доцент кафедри ПЗАС, к.т.н., доцент Макарова Л.М.
Затверджено наказом ректора №275-уч від 07.04.2026 р.
2. Термін подання роботи: 01.06.2026 р.
3. Вихідні дані по роботі: _____
4. Перелік питань, що належать до розробки (найменування розділів)

Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів _____

Титульний слайд, Мета та завдання кваліфікаційної роботи, Аналіз вимог до ПЗ, Діаграма варіантів використання, Архітектура програмного забезпечення, Концептуальна модель даних, Діаграми взаємодії, Логічна модель бази даних, Технічний проект ПЗ, Результати розробки, Висновки, Заключний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 07.04.2026 р.

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	30.03.2026	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	06.04.2026	ПП*
3.	Аналіз вимог до ПЗ	13.04.2026	ПП*
4.	Розробка архітектури ПЗ	20.04.2026	
5.	Розробка проекту ПЗ	04.05.2026	
6.	Реалізація та тестування ПЗ	11.05.2026	
7.	Підготовка розділу Результати розробки ПЗ	15.05.2026	
8.	Підготовка розділу з охорони праці	18.05.2026	ПП*
9.	Підготовка розділу ВИСНОВКИ	22.05.2026	
10.	Оформлення списку використаних джерел та додатків	25.05.2026	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	01.06.2026	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
12.	Підготовка презентації та доповіді	05.06.2026	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	08.06.2026	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	15.06.2026	

* - за результатами переддипломної практики (ПП), яка була з 02.02.2026 до 22.03.2026 р.

Студент _____ ____ Нестер І.В.____
(підпис) (ПІБ)

Керівник роботи _____ ___Макарова Л.М.___
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові», що належить до практичної задач інженерії програмного забезпечення, яка характеризується комплексністю та невизначеністю умов.

У процесі виконання кваліфікаційної роботи було проведено аналіз практичної задачі інженерії програмного забезпечення, досліджено особливості автоматизації бізнес-процесів аптечних закладів, а також розглянуто існуючі програмні рішення для обліку товарів і продажу. На основі проведеного аналізу сформовано вимоги до програмного забезпечення, спроєктовано архітектуру вебзастосунку та структуру бази даних, реалізовано функціональні модулі керування товарами, обліку продажів, формування звітності та авторизації користувачів. Також проведено тестування програмного забезпечення та розроблено необхідну програмну документацію.

Кваліфікаційна робота викладена на 112 сторінках друкованого тексту, містить 19 рисунків, 6 таблиць, список використаних джерел з 19 найменувань та 5 додатків.

ABSTRACT

This thesis is devoted to the development of a web application for automating inventory and sales management at the private pharmacy «Be healthy», which falls under the category of practical software engineering tasks characterized by complexity and uncertainty.

As part of this thesis, a practical software engineering problem was analyzed, the specifics of automating business processes in pharmacies were investigated, and existing software solutions for inventory and sales management were examined. Based on this analysis, software requirements were defined, the architecture of the web application and the database structure were designed, and functional modules for inventory management, sales accounting, reporting, and user authorization were implemented. Also was conducted software testing, and the necessary software documentation was developed.

The thesis consists of 112 pages of printed text, includes 19 figures, 6 tables, list of references from 19 names, and 5 appendices.

ЗМІСТ

ВСТУП

У сучасних умовах розвитку інформаційних технологій важливим напрямом є автоматизація бізнес-процесів підприємств за допомогою програмного забезпечення. Практична задача інженерії програмного забезпечення, яка характеризується комплексністю та невизначеністю умов, полягає у розробці програмних систем, що забезпечують ефективне зберігання, обробку та аналіз даних. Розробка такого програмного забезпечення належить до складних інженерних задач, оскільки воно характеризується великою кількістю взаємопов’язаних компонентів, а також необхідністю врахування різноманітних функціональних вимог та умов експлуатації.

Зі збільшенням кількості елементів програмної системи зростає і кількість взаємодій між ними, що підвищує складність програмного забезпечення та ризик виникнення помилок під час його модифікації та супроводження. Крім того, у процесі розробки програмних систем часто виникає невизначеність вимог, оскільки потреби користувачів можуть змінюватися протягом життєвого циклу програмного забезпечення [1].

Аналіз існуючих програмних рішень, наприклад SwilERP [2] та Unisolve [3], показав, що вони мають низку недоліків, таких як висока вартість, відсутність української локалізації, застарілий або незручний інтерфейс користувача, а також орієнтація на інші сфери торгівлі. У зв’язку з цим виникає доцільність розробки власного програмного рішення, яке буде враховувати специфіку роботи роздрібної аптеки та забезпечувати необхідний набір функціональних можливостей.

Актуальність розробки програмного забезпечення для автоматизації діяльності аптечних закладів зумовлена необхідністю підвищення ефективності обліку товарів, контролю залишків, збереження інформації про продажі та формування звітності. Використання спеціалізованих програмних систем дозволяє значно зменшити кількість помилок, пов’язаних із ручною обробкою

інформації, прискорити виконання операцій та забезпечити зручний доступ до необхідних даних.

Метою кваліфікаційної роботи є розробка вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові».

Для досягнення поставленої мети необхідно вирішити такі завдання:

– визначити та проаналізувати основні бізнес-процеси аптеки, що підлягають автоматизації;

– провести аналіз існуючих програмних рішень для автоматизації обліку товарів і продажу та визначити їх переваги і недоліки;

– сформувати функціональні та нефункціональні вимоги до програмного забезпечення з урахуванням автоматизації процесів обліку та продажу;

– спроєктувати структуру вебзастосунку та базу даних для підтримки автоматизованого обліку товарів і операцій продажу;

– реалізувати програмні модулі автоматизації керування товарами, обліку продажів, формування звітності та авторизації користувачів;

– забезпечити інтегровану роботу всіх компонентів системи для підтримки бізнес-процесів аптеки;

– провести тестування програмного забезпечення та оцінити ефективність автоматизації бізнес-процесів

– розробити технічну та користувацьку документацію до програмного забезпечення.

Об’єктом кваліфікаційної роботи є процес розробки вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові».

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБЛІКУ ТОВАРІВ ТА ПРОДАЖУ В ПРИВАТНІЙ АПТЕЦІ «БУДЬТЕ ЗДОРОВІ»

1.1 Аналіз практичної задачі інженерії програмного забезпечення з автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

Одним із важливих напрямів інженерії програмного забезпечення є розробка програмних продуктів для автоматизації діяльності підприємств. Такі програмні продукти дозволяють оптимізувати процеси зберігання, обробки та аналізу інформації, підвищити ефективність роботи персоналу та зменшити кількість помилок, пов’язаних із ручним введенням даних. У сучасних умовах значна частина бізнес-процесів організацій пов’язана з обробкою великих обсягів інформації, що потребує використання спеціалізованого програмного забезпечення.

Практична задача інженерії програмного забезпечення, яка характеризується комплексністю та невизначеністю умов, що розглядається у даній кваліфікаційній роботі, полягає у розробці вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові». Діяльність аптечного закладу пов’язана з постійним контролем наявності лікарських засобів та їх терміном придатності, веденням обліку товарів, фіксацією операцій продажу, а також формуванням різноманітних звітів. Виконання цих процесів без використання автоматизованих програмних систем значно ускладнює роботу персоналу та може призводити до виникнення помилок у веденні обліку.

Особливістю задачі розробки програмного забезпечення для аптечного закладу є необхідність обробки великої кількості взаємопов’язаних даних, серед яких інформація про товари, їх залишки та терміни придатності, операції продажу та авторизації користувачів програмної системи. Крім того,

вебзастосунок повинен забезпечувати зручний інтерфейс взаємодії з користувачем, підтримувати операції додавання, редагування та видалення даних, а також надавати можливість формування звітів на основі наявної інформації.

Ще однією важливою вимогою є забезпечення контролю доступу до програмної системи. У зв'язку з цим програмне забезпечення повинно містити механізм авторизації користувачів, що дозволяє обмежити доступ до певних функцій вебзастосунку відповідно до ролі користувача. Це підвищує безпеку даних та дозволяє організувати роботу більш ефективно.

У рамках кваліфікаційної роботи буде розроблено вебзастосунок, який дозволить автоматизувати основні процеси діяльності приватної аптеки «Будьте здорові». Використання вебтехнологій забезпечує можливість доступу до програмної системи з різних пристроїв через веббраузер, що підвищує мобільність роботи та спрощує супроводження програмного забезпечення. Такий підхід також дозволяє централізовано зберігати дані у базі даних та забезпечувати їх актуальність для всіх користувачів.

1.2 Аналіз існуючих програмних рішень

SwilERP Software

SwilERP – це програмне забезпечення, яке має на меті забезпечити інтелектуальні бізнес-рішення для роздрібних аптечних магазинів. Програмне забезпечення для роздрібної аптеки має функції, що підвищують гнучкість бізнес-системи та загальну продуктивність. Програмне забезпечення розроблене таким чином, щоб відповідати унікальним бізнес-вимогам і є досить гнучким, щоб адаптуватися до будь-яких майбутніх проблем роздрібної торгівлі [4].

SwilERP з'єднується з касовими системами, що використовуються в роздрібних закладах, для проведення транзакцій з клієнтами, прийому платежів та оновлення рівня запасів. Крім того, система дозволяє власникам аптек

відстежувати дані про лікарів, продажі за рецептами, кредитну інформацію про пацієнтів та керувати виставленням рахунків з самого початку. SwilERP розроблений як комплексне рішення для роздрібної торгівлі, і постачальник може похвалитися тим, що тисячі роздрібних аптечних підприємств перейшли на його використання.

Меню програмного застосунку SwilERP наведено на рисунку 1.1.

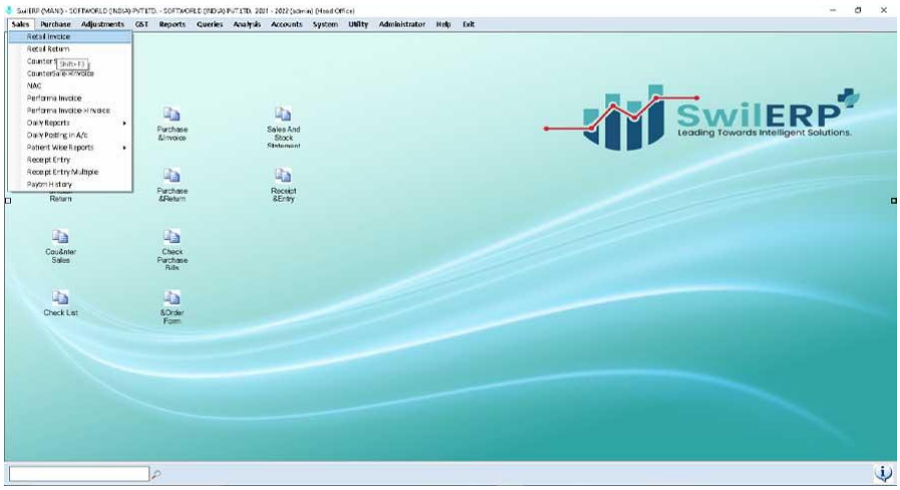


Рисунок 1.1 – Головне меню програмного застосунку SwilERP

Вікно програми, яке відображає список продажів наведено на рисунку 1.2.

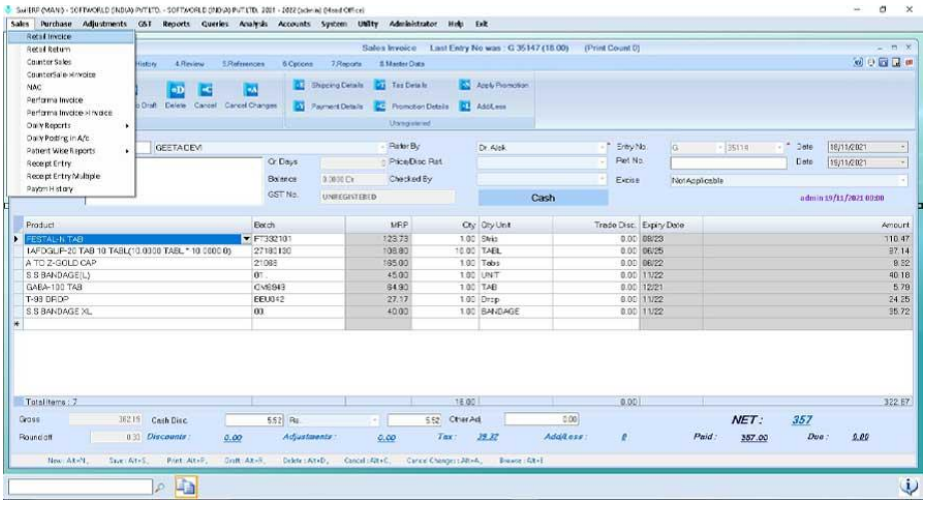


Рисунок 1.2 – Вікно ПЗ SwilERP, яке відображає список продаж

Вікно програми, яке відображає список закупів наведено на рисунку 1.3.

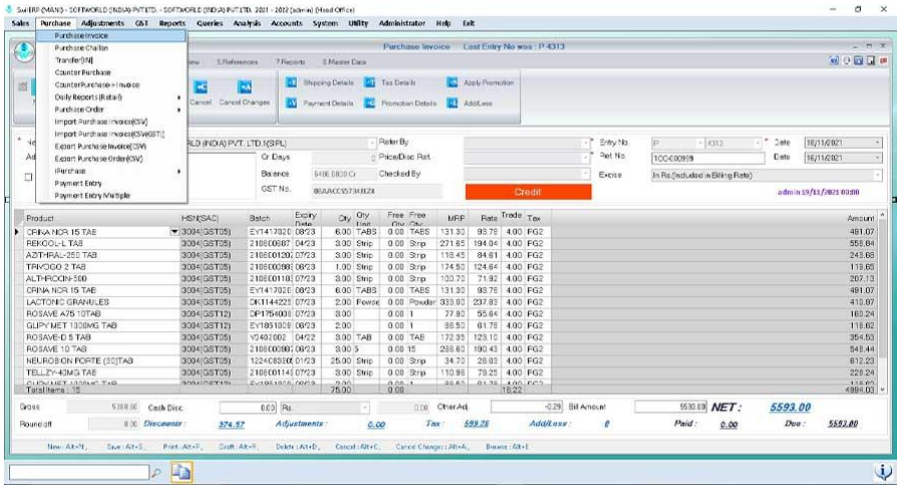


Рисунок 1.3 – Вікно ПЗ SwilERP, яке відображає список закупів

Переваги та недоліки SwilERP Software

Програма має класичний інтерфейс, що робить її доволі практичною та інтуїтивною.

Недоліком же, являється висока ціна продукту та відсутність української локалізації.

Unisolve Software

Програмне забезпечення UNISOLVE – це комплексна система управління, пристосована для задоволення потреб фармацевтичної оптової та дистриб'юторської галузей. Програмне забезпечення має інтуїтивно зрозумілий користувацький інтерфейс, що забезпечує швидкий доступ до його функцій. Його можна використовувати як онлайн, так і офлайн, забезпечуючи власникам бізнесу гнучкість [5].

За допомогою UNISOLVE автоматизовано такі операційні процеси, як управління замовленнями, контроль запасів, відповідність нормативним вимогам та фінансові транзакції для доставки підконтрольних товарів.

Вікно програми, яке відображає список продажів наведено на рисунку 1.4.

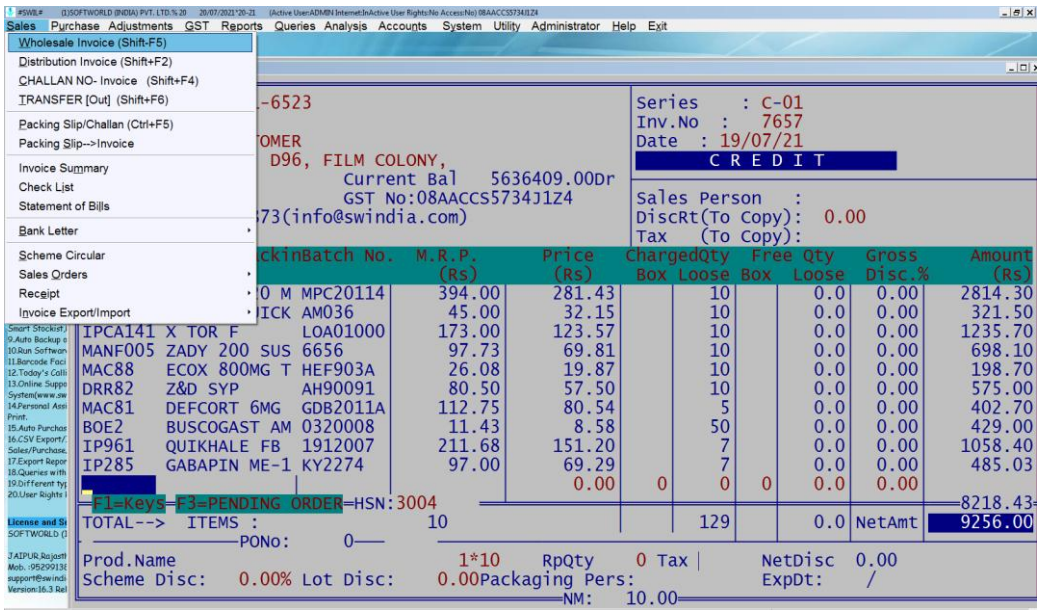


Рисунок 1.4 – Вікно ПЗ UNISOLVE, яке відображає список продаж

Вікно програми, яке відображає список закупів наведено на рисунку 1.5.

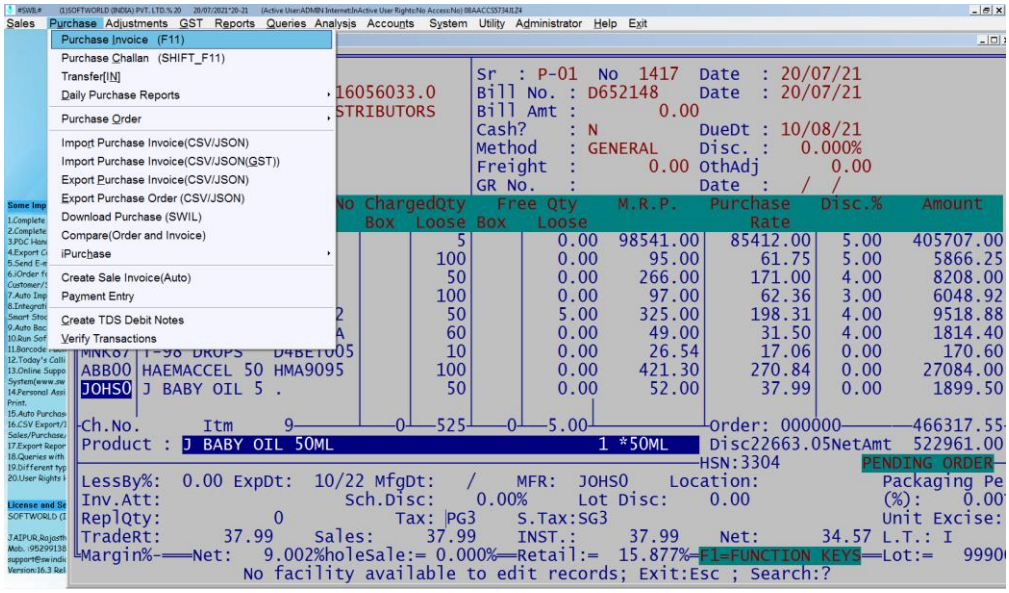


Рисунок 1.5 – Вікно ПЗ UNISOLVE, яке відображає список закупів

Вікно програми, де проходить створення звітів на рисунку 1.6.

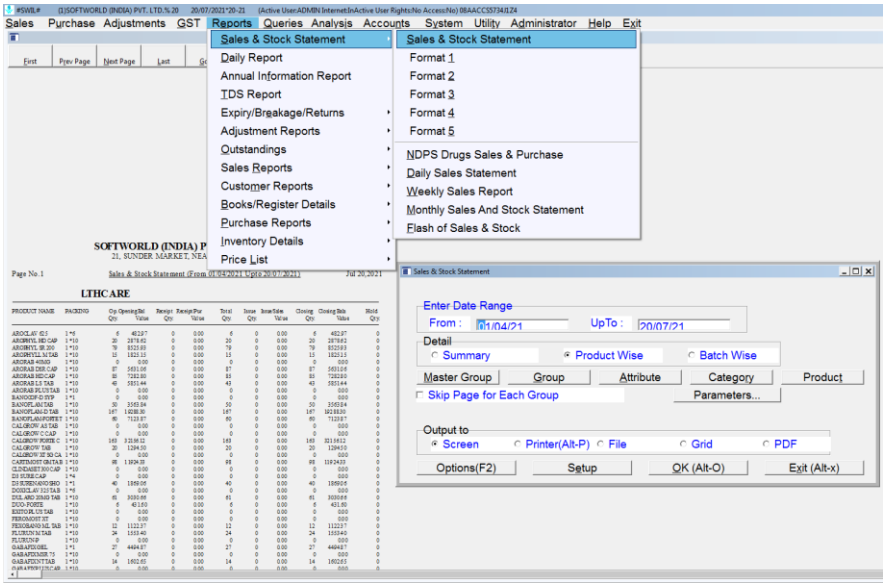


Рисунок 1.6 – Вікно ПЗ UNISOLVE, де проходить створення звітів

Переваги та недоліки Unisolve Software

Головним недоліком продукту є застаріла та неінтуїтивна програмна частина, яка дозволяє користувачу взаємодіяти з базою даних. Окрім цього, у зв'язку з тим що програма була вироблена за кордоном, в ній відсутня українська локалізація. Також, як сказано вище, програма орієнтована для оптової та дрібнооптової торгівлі, використання в сфері роздрібної торгівлі, а саме в кінцевій аптеці, потребувала б певного допрацювання.

Через застарілий інтерфейс, продукт має відносно невелику ціну. Проте через вищеперераховані недоліки, а саме через потребу удосконалення програми, вибір даної програми не буде логічним та ефективним.

1.3 Постановка задачі на розробку вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

На основі проведеного аналізу предметної області та існуючих програмних рішень встановлено, що для ефективної організації роботи приватної аптеки «Будьте здорові» необхідне програмне забезпечення, яке забезпечить автоматизацію процесів обліку товарів, фіксації продажів та формування

звітності. Існуючі програмні продукти не завжди повністю відповідають потребам невеликих аптечних закладів через високу вартість, складність використання або невідповідність функціональних можливостей специфіці роздрібної торгівлі. У зв’язку з цим виникає необхідність розробки власного програмного рішення.

Метою розробки є створення вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові», який забезпечить зручне управління даними про товари, реєстрацію операцій продажу та формування необхідної звітності.

Застосунок має функцію авторизації та два типи користувачів: фармацевт та завідувач.

База даних застосунку включає наступні таблиці: користувачі, препарати, обладнання, продажі, категорії препаратів та обладнання, форми препаратів, системи живлення обладнання, постачальники, виробники.

Для скорочення далі по тексту, використовується абревіатура CRUD (Create, Read, Update, Delete – «створення, читання, оновлення, видалення») – це чотири основні операції для управління даними в базах даних і вебдодатках.

Завідувач аптеки може:

- продавати товар;
- виконувати CRUD-операції з інформацією препаратів;
- виконувати CRUD-операції з інформацією обладнання;
- виконувати CRUD-операції з інформацією категорій товарів;
- виконувати CRUD-операції з інформацією форм препаратів;
- виконувати CRUD-операції з інформацією систем живлення;
- виконувати CRUD-операції з інформацією виробників;
- виконувати CRUD-операції з інформацією постачальників;
- виконувати CRUD-операції з інформацією користувачів;
- переглядати список продажів;
- скасовувати продажі;
- формувати звіти.

Фармацевт аптеки може лише переглядати та продавати товар.

До складу технічних засобів належать серверне обладнання, клієнтські пристрої користувачів, а також мережеве обладнання, що забезпечує доступ до вебзастосунку.

Серверна частина вебзастосунку призначена для обробки запитів користувачів, виконання логіки програмного забезпечення та роботи з базою даних. Для її функціонування необхідний сервер або персональний комп'ютер з такими мінімальними характеристиками:

– процесор із тактовою частотою 2,0 ГГц;
– оперативна пам'ять обсягом 4 ГБ;
– вільні 10 ГБ на жорсткому диску для зберігання програмного забезпечення та бази даних;
– операційна система Windows 7, що підтримує роботу вебсерверного програмного забезпечення (Apache, Nginx).

Клієнтська частина вебзастосунку використовується працівниками аптеки для взаємодії з ним. Доступ до вебзастосунку здійснюється через браузер, тому клієнтські пристрої повинні відповідати мінімальним вимогам, наведеним нижче.

Для персональних комп'ютерів:

– персональний комп'ютер з доступом до мережі Інтернет;
– процесор із тактовою частотою 1,5 ГГц;
– оперативна пам'ять обсягом в 2 ГБ;
– наявність сучасного веббраузера (Google Chrome, Mozilla Firefox, Microsoft Edge).

Для мобільних пристроїв:

– смартфон або планшет з доступом до мережі Інтернет;
– операційна система Android 8.0 або iOS 13;
– оперативна пам'ять обсягом в 2 ГБ;
– наявність сучасного мобільного веббраузера (Google Chrome, Safari або аналогічного).

Для забезпечення взаємодії між клієнтською та серверною частинами системи необхідне підключення до локальної мережі або мережі Інтернет. Швидкість мережевого з’єднання повинна бути достатньою для передачі даних між користувачем та сервером без значних затримок.

2 РОЗРОБКА ПРОЄКТУ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБЛІКУ ТОВАРІВ ТА ПРОДАЖУ В ПРИВАТНІЙ АПТЕЦІ «БУДЬТЕ ЗДОРОВІ»

2.1 Аналіз вимог вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

2.1.1 Побудова моделі варіантів використання

Діаграма варіантів використання (Use Case Diagram) – це тип діаграми в мові UML, який наочно показує, хто взаємодіє із системою та які саме функції вона виконує. Вона демонструє зовнішню поведінку системи з точки зору користувача, не заглиблюючись у її внутрішню технічну реалізацію.

Діаграма складається з трьох головних елементів:

1. Актори – це зовнішні сутності, які взаємодіють із системою. Вони можуть представляти користувачів, інші інформаційні системи або апаратні пристрої.
2. Прецеденти (Use Cases) – описують функціональність системи, тобто дії або сервіси, які система надає акторам. Кожен прецедент відображає окремий сценарій взаємодії між актором і системою.
3. Зв’язки (Relationships) – лінії, що показують взаємодії між акторами та функціями, а також зв’язки між самими функціями.

Діаграми варіантів використання застосовуються переважно на етапі аналізу вимог до системи, коли необхідно визначити її функціональні можливості та взаємодію з користувачами. Вони допомагають зрозуміло представити систему для замовника, відобразити ролі користувачів і їхні дії, а також слугують основою для подальшого проєктування та документування програмного забезпечення.

Вебзастосунок для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові» буде мати наступних акторів:

– завідувач;

– фармацевт.

Опис користувачів наведено в таблиці 2.1.

Таблиця 2.1 – Короткий опис акторів

Актор	Короткий опис
Завідувач	Користувач з найбільшою кількістю привілей. Створюється автоматично в базі даних при першому розгортанні системи. Має можливість працювати (виконання базових операцій перегляду, додавання, редагування та видалення) з даними препаратів, обладнання, користувачів та довідників (категорії товарів, форми і т.д.). Також має можливість виконувати продаж товарів, скасування продажів та формувати звіти.
Фармацевт	Базовий користувач системи якого додає адміністратор. Може переглядати список товарів та виконувати продаж.

Опис виявлених варіантів використання наведено в таблиці 2.2.

Таблиця 2.2 – Виявлені варіанти використання

Актор	Назва прецеденту	Короткий опис
1	2	3
Завідувач, фармацевт	Авторизація	Вхід до застосунку за логіном та паролем
Завідувач, фармацевт	Перегляд препаратів	Дозволяє користувачу переглядати список препаратів
Завідувач	Додавання препарату	Дозволяє завідувачу додати новий препарат до бази даних
Завідувач	Редагування препарату	Дозволяє завідувачу змінювати інформацію препарату в базі даних
Завідувач	Видалення препарату	Дозволяє завідувачу видаляти препарат з бази даних
Завідувач, фармацевт	Перегляд обладнання	Дозволяє користувачу переглядати список обладнання
Завідувач	Додавання обладнання	Дозволяє завідувачу додати нове обладнання до бази даних
Завідувач	Редагування обладнання	Дозволяє завідувачу змінювати інформацію обладнання в базі даних
Завідувач	Видалення обладнання	Дозволяє завідувачу видаляти обладнання з бази даних
Завідувач	Перегляд категорій препаратів	Дозволяє завідувачу переглядати всі категорії препаратів
Завідувач	Додавання категорії препаратів	Дозволяє завідувачу додавати нову категорію препаратів до бази даних
Завідувач	Редагування категорії препаратів	Дозволяє завідувачу редагувати категорію препаратів в базі даних

Продовження таблиці 2.2

1	2	3
Завідувач	Видалення категорії препаратів	Дозволяє завідувачу видаляти категорію препаратів в базі даних
Завідувач	Перегляд категорій обладнання	Дозволяє завідувачу переглядати всі категорії обладнання
Завідувач	Додавання категорії обладнання	Дозволяє завідувачу додавати нову категорію обладнання до бази даних
Завідувач	Редагування категорії обладнання	Дозволяє завідувачу редагувати категорію обладнання в базі даних
Завідувач	Видалення категорії обладнання	Дозволяє завідувачу видаляти категорію обладнання в базі даних
Завідувач	Перегляд форм препаратів	Дозволяє завідувачу переглядати всі форми препаратів
Завідувач	Додавання форми препаратів	Дозволяє завідувачу додавати нову форму препаратів до бази даних
Завідувач	Редагування форми препаратів	Дозволяє завідувачу редагувати форму препаратів в базі даних
Завідувач	Видалення форми препаратів	Дозволяє завідувачу видаляти форму препаратів в базі даних
Завідувач	Перегляд типів живлення	Дозволяє завідувачу переглядати всі типи живлення
Завідувач	Додавання типу живлення	Дозволяє завідувачу додавати новий тип живлення до бази даних
Завідувач	Редагування типу живлення	Дозволяє завідувачу редагувати тип живлення в базі даних
Завідувач	Видалення типу живлення	Дозволяє завідувачу видаляти тип живлення в базі даних
Завідувач	Перегляд постачальників	Дозволяє завідувачу переглядати всіх постачальників
Завідувач	Додавання постачальника	Дозволяє завідувачу додавати нового постачальника до бази даних
Завідувач	Редагування постачальника	Дозволяє завідувачу редагувати постачальника в базі даних
Завідувач	Видалення постачальника	Дозволяє завідувачу видаляти постачальника в базі даних
Завідувач	Перегляд виробників	Дозволяє завідувачу переглядати всіх виробників
Завідувач	Додавання виробника	Дозволяє завідувачу додавати нового виробника до бази даних
Завідувач	Редагування виробника	Дозволяє завідувачу редагувати виробника в базі даних
Завідувач	Видалення виробника	Дозволяє завідувачу видаляти виробника в базі даних
Завідувач, фармацевт	Продаж товарів	Дозволяє користувачу виконувати продаж товарів та створювати чек в базі даних
Завідувач	Перегляд списку продажів	Дозволяє користувачу переглядати інформацію певного продажу
Завідувач	Скасування продажу	Дозволяє завідувачу міняти статус продажу на «скасований»
Завідувач	Формування звітів	Дозволяє завідувачу створювати звіти звіти

Кінець таблиці 2.2

1	2	3
Завідувач	Перегляд списку користувачів	Дозволяє завідувачу переглядати інформацію фармацевтів аптеки
Завідувач	Додавання користувача	Дозволяє завідувачу додавати нового фармацевта до бази даних
Завідувач	Редагування користувача	Дозволяє завідувачу редагувати інформацію фармацевта в базі даних
Завідувач	Видалення користувача	Дозволяє завідувачу видаляти фармацевта з бази даних

Діаграма варіантів використання вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові» наведена на рисунку 2.1.

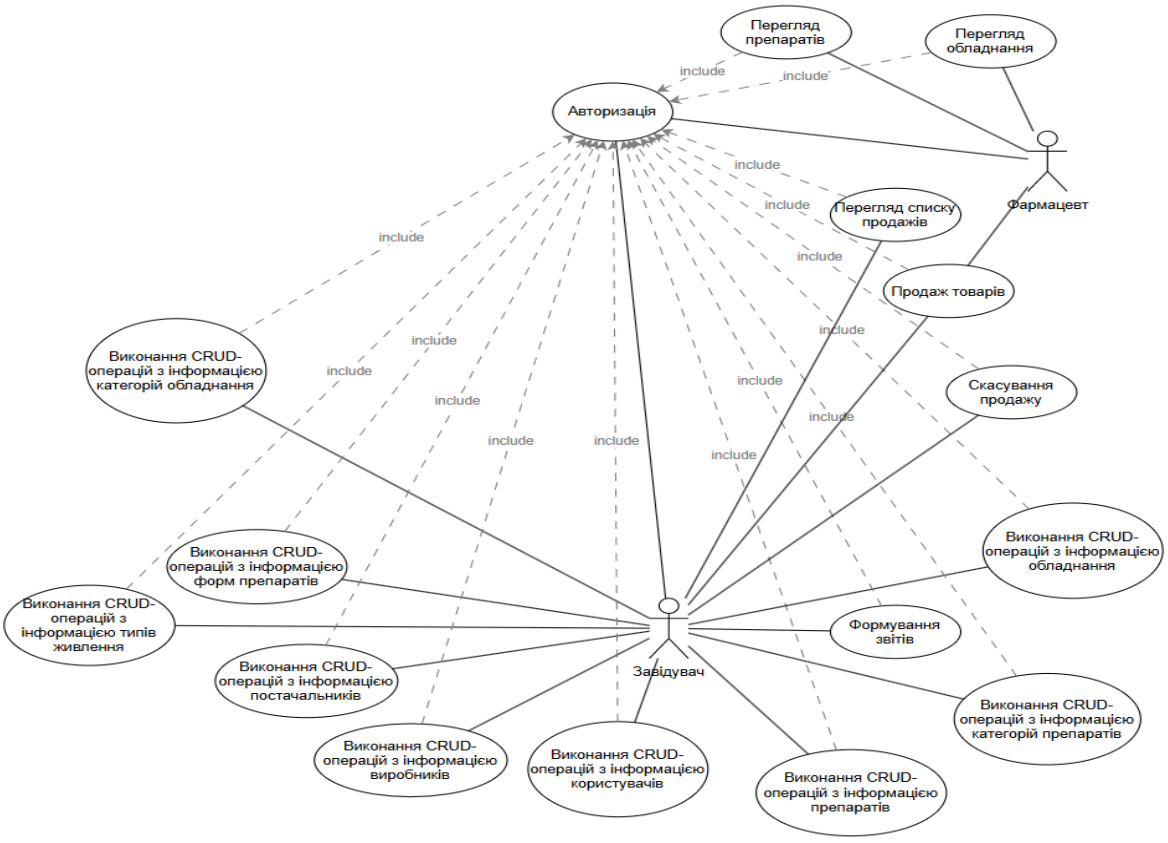


Рисунок 2.1 – Діаграма варіантів використання вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

У результаті, ми описали акторів, виявили варіанти використання та розробили діаграму варіантів використання для вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові».

2.1.2 Розробка специфікації варіантів використання

Розберемо докладніше деякі варіанти використання вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

Специфікація основних варіантів використання

Специфікації додавання, редагування та видалення у всіх сутностях є однаковими (препарати, обладнання, категорії препаратів та обладнання, форми препаратів, типи живлення, виробники, постачальники, облікові записи), тому їх розглянемо разом.

1. Додавання препарату

Призначення: ця специфікація описує процес додавання нового препарату до бази даних сайту.

Учасники: завідувач, система.

Передумови: успішна авторизація та право на додавання препаратів.

Стартова точка: відкриття головної сторінки сайту.

Процедура (сценарій):

1) Користувач відкриває сторінку керування препаратами.

2) Система відображає сторінку керування препаратами.

3) Користувач обирає опцію додавання препарату.

4) Система відкриває форму додавання препарату.

5) Користувач вводить дані нового препарату.

6) Користувач запускає функцію додавання препарату.

7) Система перевіряє валідність даних.

7.1) Якщо дані валідні, система додає препарат до бази даних.

7.2) Якщо ж дані товару невалідні, система повертає користувача до форми та виводить помилки.

7.3) Користувач відміняє процес додавання препарату.

8) Система повертає користувача до сторінки керування товарами.

Альтернативна процедура: немає.

Аварійні умови: немає.

Результати: база даних сайту з новим препаратом (поява нового препарату

у списку препаратів).

Умови після завершення використання: немає.

Особливості: немає.

Коментарі: немає.

2. Формування звітів

Призначення: ця специфікація описує процес формування звітів.

Учасники: завідувач, система.

Передумови: успішна авторизація та право на формування звітів.

Стартова точка: відкриття головної сторінки сайту.

Процедура (сценарій):

1) Користувач відкриває сторінку формування звітів.

2) Система відображає сторінку формування звітів.

3) Користувач обирає необхідний звіт зі списку.

4) Користувач запускає функцію формування звіту.

5) Система формує відповідний звіт.

6) Система повертає користувача до сторінки формування звітів.

Альтернативна процедура: немає.

Аварійні умови: немає.

Результати: база даних сайту з новим звітом (поява нового звіту у списку звітів).

Умови після завершення використання: немає.

Особливості: немає.

Коментарі: немає.

2.2 Архітектура вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

Обґрунтування вибору архітектурного стилю

Для розробки вебзастосунку для автоматизації обліку товарів та продажу в

приватній аптеці «Будьте здорові» доцільним є використання клієнт-серверної архітектури з REST-підходом до організації взаємодії між компонентами системи. Такий вибір обумовлений особливостями предметної області, вимогами до масштабованості системи, зручності доступу до даних та можливості подальшого розвитку програмного забезпечення.

Клієнт-серверна архітектура передбачає розподіл системи на дві основні частини: клієнтську та серверну. Клієнтська частина реалізує інтерфейс користувача та забезпечує взаємодію працівників аптеки із системою через веббраузер. Серверна частина відповідає за обробку бізнес-логіки, доступ до бази даних, перевірку прав доступу користувачів та виконання основних операцій над даними.

Застосування REST (Representational State Transfer) як стилю взаємодії між клієнтом і сервером дозволяє організувати обмін даними у вигляді стандартних HTTP-запитів (GET, POST, PUT, DELETE), що забезпечує простоту інтеграції, гнучкість та незалежність клієнтської частини від реалізації серверної логіки. Дані при цьому, як правило, передаються у форматі JSON, що є легким для обробки та широко підтримується сучасними вебтехнологіями.

Використання такої архітектури є доцільним з огляду на те, що система повинна підтримувати одночасну роботу декількох користувачів (завідувач, фармацевт тощо), забезпечувати централізоване зберігання даних та гарантувати їх актуальність. Крім того, клієнт-серверна модель дозволяє легко масштабувати систему у разі збільшення кількості користувачів або розширення функціональних можливостей.

Додатковою перевагою даного підходу є можливість розділення відповідальності між компонентами системи, що спрощує супровід, тестування та подальшу модернізацію програмного забезпечення. Зокрема, зміни в інтерфейсі користувача не впливають на бізнес-логіку, і навпаки, що підвищує гнучкість розробки.

За обраним стилем було розроблено схему архітектури вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові».

Загальна діаграма архітектури представлена на рисунку 2.2.

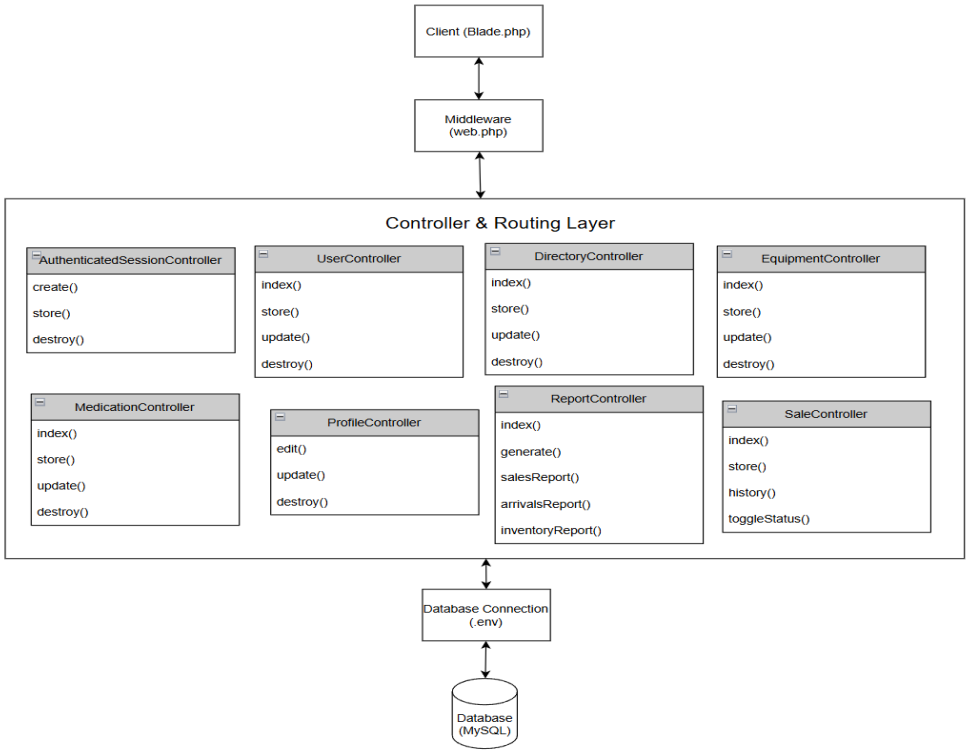


Рисунок 2.2 – Загальна діаграма архітектури вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

Структура:

1. Рівень представлення (Presentation Layer) – даний рівень відповідає за взаємодію користувача з вебзастосунком. До нього входять сторінки інтерфейсу, форми введення даних, таблиці, модальні вікна та інші елементи користувацького інтерфейсу.

У системі цей рівень реалізовано за допомогою Blade-шаблонів Laravel, HTML, Tailwind CSS та Alpine.js.

2. Рівень проміжної логіки та безпеки (Middleware Layer) – даний рівень виконує функції проміжного рівня та забезпечує перевірку прав доступу користувачів. Він використовується для перевірки автентифікації, контролю прав доступу, захисту від несанкціонованих дій та обробки службових перевірок перед виконанням основної логіки застосунку. У системі middleware забезпечує обмеження доступу до адміністративних функцій, перевірку авторизації користувачів та захист HTTP-запитів.

3. Рівень контролерів та маршрутизації (Controller & Routing Layer) – даний рівень відповідає за приймання та обробку HTTP-запитів від клієнта. Маршрути визначають, який контролер і метод повинні обробити конкретний запит, а контролери реалізують основну логіку взаємодії між інтерфейсом користувача, моделями та базою даних. Також цей рівень забезпечує передачу даних до представлень, перенаправлення між сторінками та формування відповідей користувачу.

4. Рівень доступу до даних (Data Access Layer) – даний рівень відповідає за встановлення з’єднання та роботу з базою даних MySQL. Він забезпечує створення, отримання, оновлення та видалення даних, а також керування зв’язками між таблицями. У системі цей рівень реалізовано за допомогою ORM Eloquent, моделей Laravel та міграцій, що дозволяє спростити роботу з базою даних і забезпечити структурований доступ до інформації.

5. Рівень бази даних (Database Layer) – відповідає за фізичне зберігання та керування даними вебзастосунку. У системі використовується СУБД MySQL, яка забезпечує збереження інформації про користувачів, товари, продажі, постачальників та інші сутності. Також цей рівень підтримує цілісність даних, зв’язки між таблицями та виконання SQL-запитів.

Діаграма розгортання (Deployment Diagram) – це один із видів UML-діаграм, який використовується для відображення фізичної структури програмної системи. Вона показує апаратні та програмні компоненти системи, вузли їх розміщення, а також зв’язки між ними. Діаграма розгортання дозволяє наочно представити середовище функціонування програмного забезпечення та спосіб розміщення його складових на серверному й клієнтському обладнанні [6].

Діаграма розгортання модельованої системи вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові» представлена на рисунку 2.3.

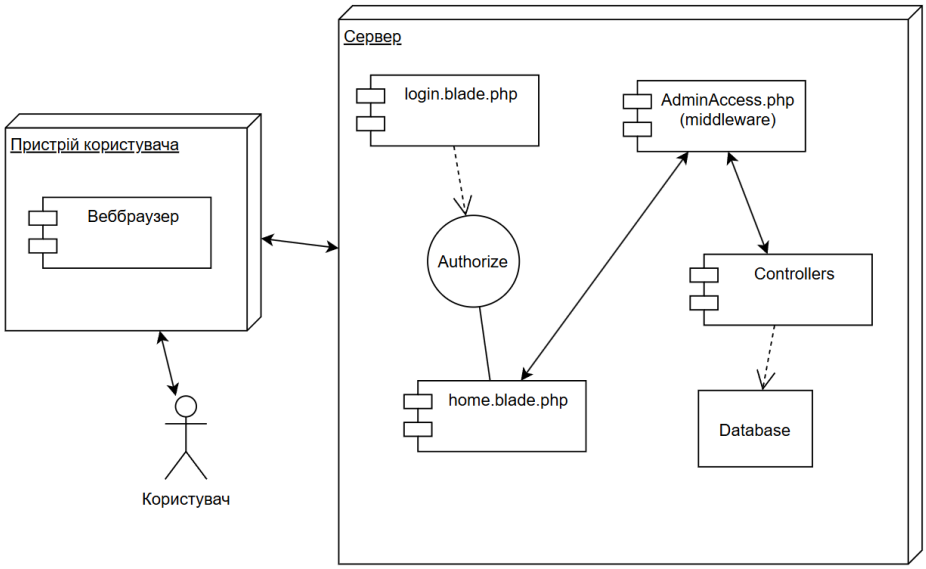


Рисунок 2.3 – Діаграма розгортання модельованої системи вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

На діаграмі розгортання представлено два основних вузли:

1. Пристрій користувача де розміщено веббраузер, за допомогою якого користувач взаємодіє з вебзастосунком.

2. Сервер, який має кілька компонентів:

– login.blade.php – файл який містить сторінку авторизації, з якої користувач відправляє запит на авторизацію;

– home.blade.php – файл який містить головну сторінку вебсайту, на якій користувач має доступ до інших частин програми;

– AdminAccess.php (middleware) – компонент, який відповідає за права доступу користувачів. Якщо користувач не має ролі «Завідувач», його можливості стають обмеженими;

– Controllers – модулі, що відповідають за обробку даних, виведення сторінок, взаємодію з базою даних;

– Database – база даних в якій зберігається інформація вебзастосунку.

2.3 Проект вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

2.3.1 Ескізний проект вебзастосунку

Діаграма діяльності варіанту використання – це графічна модель у мові UML, яка деталізує внутрішні кроки, логіку та потоки конкретного процесу (прецеденту). На відміну від звичайної діаграми варіантів використання, що лише показує, хто і що робить, діаграма діяльності показує, як саме це працює крок за кроком [7].

Діаграма діяльності для варіанту використання «Додавання препарату» наведена на рисунку 2.4.

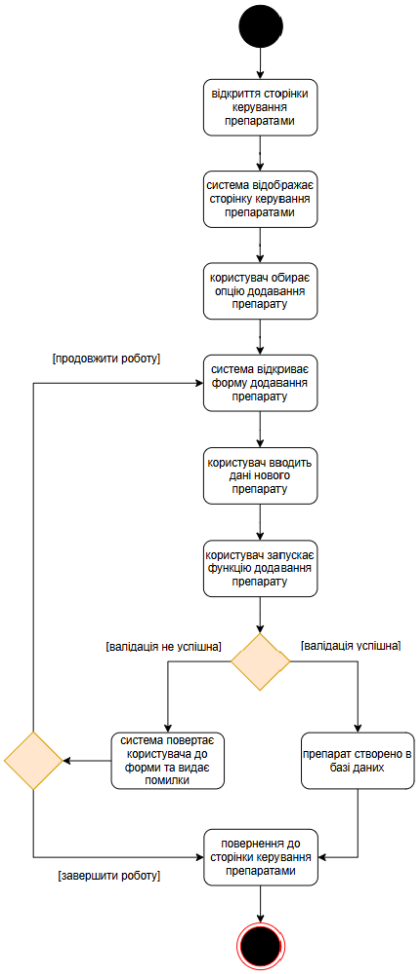


Рисунок 2.4 – Діаграма діяльності варіанту використання «Додавання препарату»

Діаграма діяльності для варіанту використання «Формування звітів» наведена на рисунку 2.5.

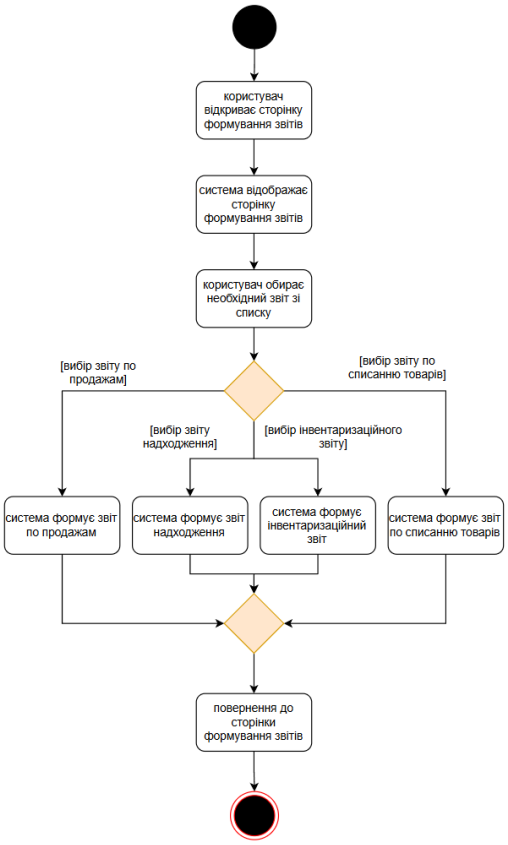


Рисунок 2.5 – Діаграма діяльності варіанту використання «Формування звітів»

Розробка концептуальної моделі

Сутність – це базовий об’єкт, процес або поняття з реального світу, інформація про який має зберігатися в базі даних [8].

Атрибут – це характеристика або властивість, яка описує сутність.

В роботі можна визначити наступні сутності та їх атрибути:

– користувачі: id, ім’я, прізвище, email, пароль, посада;

– препарат: id, назва, кількість, дата надходження, ціна, термін придатності, рецептурний препарат, категорія, форма, виробник, постачальник;

– обладнання: id, назва, кількість, дата надходження, ціна, термін придатності, категорія, тип живлення, виробник, постачальник;

– продажі: id, продавець, товари, кількість, дата продажу, ціна, статус.
– категорії препаратів: id, назва;
– категорії обладнання: id, назва;
– форми препаратів: id, назва;
– типи живлення: id, назва;
– виробники: id, назва;
– постачальники: id, назва.

Концептуальна модель даних вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові» наведена на рисунку 2.6.

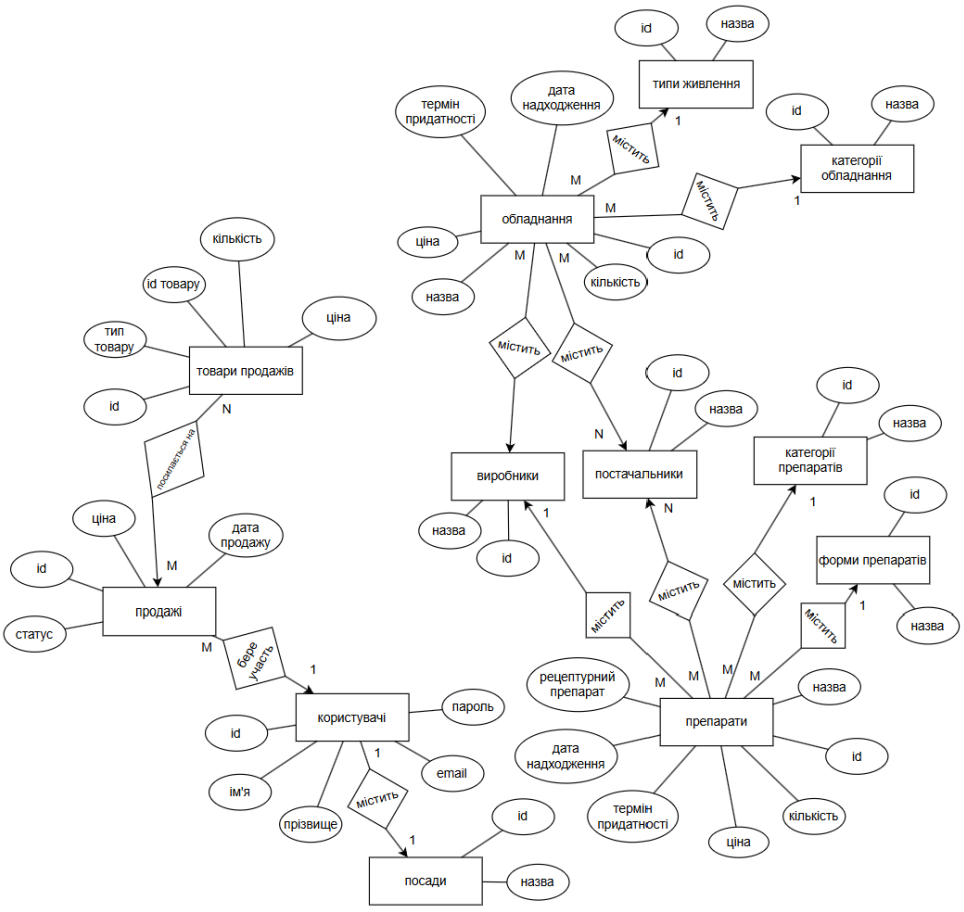


Рисунок 2.6 – Концептуальна модель даних вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

Зв’язки між сутностями наведені в таблиці 2.3.

Таблиця 2.3 – Зв’язки між сутностями

Сутності	Зв’язки	Пояснення
Користувачі – Продажі	1:M	У багатьох продажах може бути лише один продавець
Продажі – Товари продажу	M:N	У багатьох продажах можуть бути участь декілька товарів
Користувачі – Посади	1:1	Один користувач може мати лише одну посаду
Категорії препаратів – Препарати	1:M	У багатьох препаратів може бути тільки одна категорія
Форми препаратів – Препарати	1:M	У багатьох препаратів може бути тільки одна форма
Постачальники – Препарати	M:N	У багатьох препаратів може бути кілька постачальників
Виробники – Препарати	1:M	У багатьох препаратів може бути тільки один виробник
Категорії обладнання – Обладнання	1:M	У багатьох обладнань може бути тільки одна категорія
Тип живлення – Обладнання	1:M	У багатьох обладнань може бути тільки один тип живлення
Постачальники – Обладнання	M:N	У багатьох обладнань може бути кілька постачальників
Виробники – Обладнання	1:M	У багатьох обладнань може бути тільки один виробник

Концептуальна модель дозволяє визначити основні сутності системи, їх атрибути та взаємозв’язки, що забезпечує цілісне уявлення про структуру майбутнього вебзастосунку. Використання концептуальної моделі спрощує процес проєктування бази даних і програмних компонентів, сприяє виявленню можливих недоліків на ранніх етапах розробки та забезпечує основу для подальшого створення логічної й фізичної моделей системи.

2.3.2 Технічний проект вебзастосунку

У своїй роботі я використовував мову програмування PHP, мову розмітки HTML та фреймворк Laravel. Було розроблено діаграму класів, яка відображає взаємозв’язки та структуру класів програми.

Діаграма класів представлена на рисунку 2.7.

Специфікація класів виконана лише для контролерів додатку, у зв’язку з тим, що моделі є досить простими та подібними один до одного.

Специфікація класів

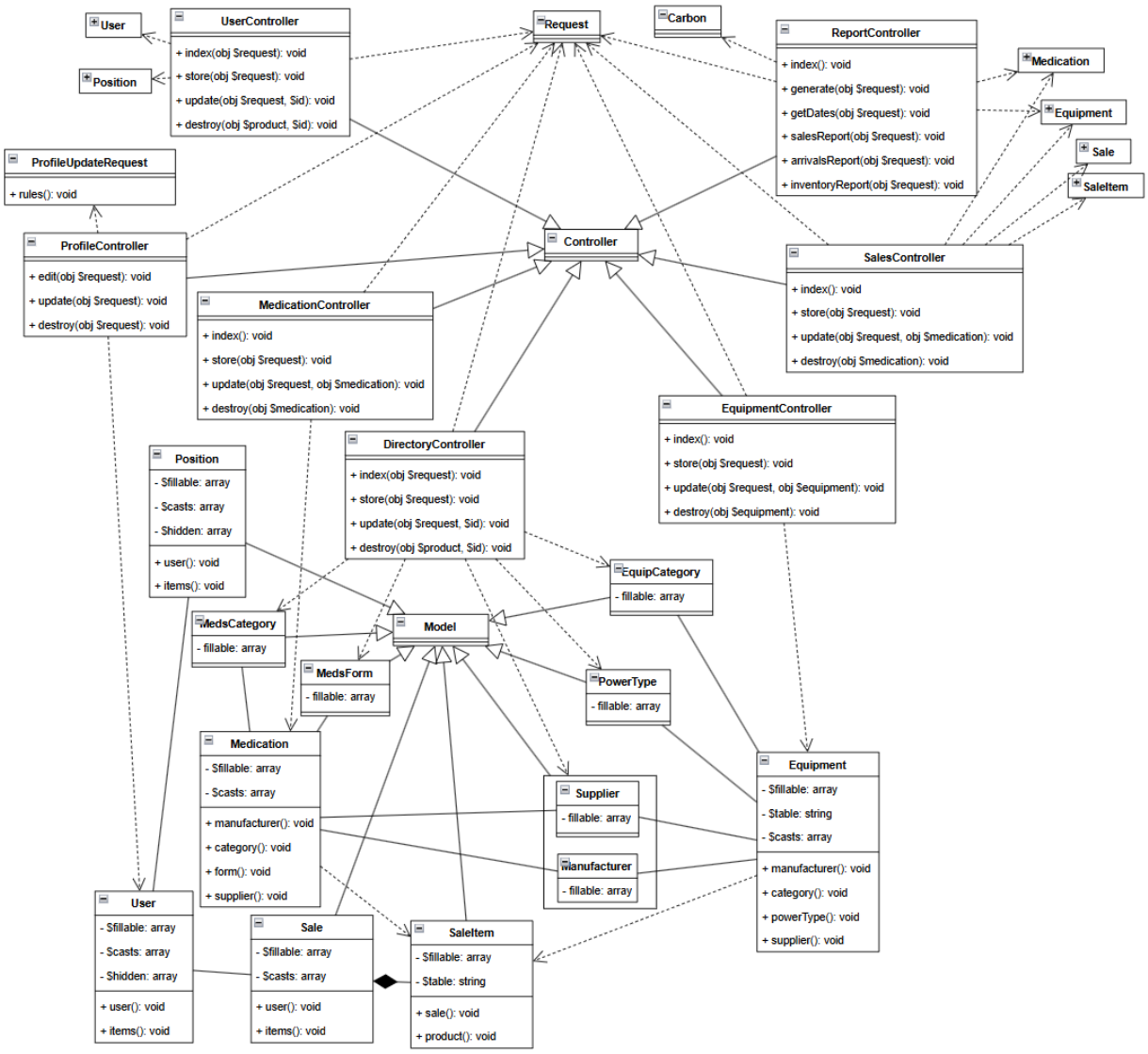


Рисунок 2.7 – Діаграма класів вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

Специфікація класу «Controller»

Ім’я: Controller.

Відповідальність: клас зазвичай не містить бізнес-логіки, а слугує базою, яку наслідують інші контролери. Він відповідає за:

- базову логіку всіх контролерів у Laravel;
- групування спільного функціоналу для інших контролерів (ProductController, EquipmentController тощо);
- підключення middleware, трейтів та методів, які можуть використовувати всі контролери;

- визначення структури контролерів, що обробляють HTTP-запити;
- забезпечення архітектурного поділу між маршрутом, бізнес-логікою та моделями.

Атрибути: у стандартному класі Controller атрибути відсутні. Основні атрибути зазвичай успадковуються від батьківських класів PHP або додаються в дочірніх контролерах.

Операції зі специфікаціями нетривіальних операцій:

Стандартний Laravel Controller зазвичай має лише трейти AuthorizesRequests та ValidatesRequests. Тому він повністю успадковє свій функціонал.

Специфікація класу «Request»

Ім’я: Request.

Відповідальність: клас отримує всі дані, що прийшли від клієнта (GET-параметри, POST-дані, JSON, файли, cookie, заголовки, маршрутні параметри), надає API для доступу до цих даних та забезпечує валідацію даних через Form Request.

Атрибути:

$query – колекція параметрів GET-запиту;

$request – колекція параметрів POST-запиту;

$files – масив завантажених файлів;

$cookies – колекція cookies, надісланих клієнтом;

$headers – HTTP-заголовки поточного запиту;

$server – змінні середовища (SERVER, REMOTE_ADDR, REQUEST_METHOD тощо);

$attributes – внутрішні службові атрибути, які можуть використовуватися $middleware, маршрутизатором та іншими компонентами.

Операції зі специфікаціями нетривіальних операцій:

input($name, $default = null) – метод спрацьовує при спробі отримати значення параметра, незалежно від того, чи він прийшов через GET, POST або JSON. У результаті запуску даного методу повертається значення поля, яке

клієнт надіслав у запиті, або default, якщо такого поля немає.

filled($name) – метод використовується для перевірки, чи присутнє поле в запиті і чи воно не є пустим. У результаті метода повертається true, якщо параметр існує та містить значення, і false – у протилежному випадку.

has($name) – метод спрацьовує при перевірці, чи взагалі присутній параметр у вхідних даних (навіть якщо він порожній). У результаті запуску повертається true, якщо ключ є у запиті.

all() – метод спрацьовує при необхідності отримати всі дані запиту (GET, POST, JSON). У результаті повертає асоціативний масив усіх доступних вхідних параметрів.

json($key = null) – метод використовується для отримання значень із JSON-тіла запиту. У результаті повертається або весь JSON як колекція, або окреме поле.

Специфікація класу «DirectoryController»

Ім’я: DirectoryController.

Відповідальність: клас DirectoryController відповідає за

- відображення списку довідників різних типів;
- створення нових елементів довідників;
- оновлення існуючих записів;
- видалення елементів довідників;
- роботу з конфігураційною картою config('directories'), яка визначає модель та параметри кожного довідника.

Контролер реалізує універсальну логіку для різних типів довідників без дублювання коду.

Атрибути: клас не має власних атрибутів, оскільки працює як HTTP-контролер.

Операції зі специфікаціями нетривіальних операцій:

index(Request $request) – відображає сторінку довідника конкретного типу.

store(Request $request) – створює новий запис у вибраному довіднику.

update(Request $request, $id) – оновлює існуючий запис довідника.

destroy(Request $request, $id) – видаляє запис із довідника.

Специфікація класу «EquipmentController»

Ім’я: EquipmentController.

Відповідальність: EquipmentController відповідає за:

- отримання списку обладнання разом із пов’язаними довідниками;
- створення нового обладнання;
- оновлення існуючих записів обладнання;
- видалення обладнання;
- передачу довідникових даних (категорії, виробники, постачальники тощо) у представлення.

Контролер реалізує повний CRUD для сутності Equipment та забезпечує зв’язок із довідниками системи.

Атрибути: клас не містить власних атрибутів.

Операції зі специфікаціями нетривіальних операцій:

index() – відображає сторінку керування обладнанням.

store(Request $request) – створення нового запису обладнання.

update(Request $request, Equipment $equipment) – оновлення існуючого обладнання.

destroy(Equipment $equipment) – видалення обладнання з системи.

Специфікація класу «MedicationController»

Ім’я: MedicationController.

Відповідальність: MedicationController відповідає за:

- отримання списку лікарських засобів разом із пов’язаними довідниками;
- створення нових медикаментів;
- оновлення існуючих записів;
- видалення медикаментів;
- передачу довідникової інформації для форми (категорії, форми випуску, виробники, постачальники).

Контролер реалізує повний CRUD для сутності Medication.

Атрибути: клас не містить власних атрибутів.

Операції зі специфікаціями нетривіальних операцій:

index() – відображення сторінки керування лікарськими засобами.

store(Request $request) – створення нового лікарського засобу.

update(Request $request, Medication $medication) – оновлення існуючого лікарського засобу.

destroy(Medication $medication) – видалення лікарського засобу.

Специфікація класу «ProfileController»

Ім’я: ProfileController.

Відповідальність: ProfileController відповідає за керування профілем автентифікованого користувача. А саме:

- відображення сторінки редагування профілю;
- оновлення даних користувача;
- видалення облікового запису користувача;
- забезпечення безпеки під час зміни критичних даних (email, пароль).

Атрибути: у класі відсутні власні атрибути.

Операції зі специфікаціями нетривіальних операцій:

edit(Request $request) – повертає сторінку редагування профілю користувача.

update(ProfileUpdateRequest $request) – оновлює дані профілю користувача.

destroy(Request $request) – видаляє акаунт користувача з системи.

Специфікація класу «ReportController»

Ім’я: ReportController.

Відповідальність: ReportController відповідає за формування звітів.

Клас обробляє HTTP-запити користувача, збирає необхідні дані з бази даних, формує звіти та зберігає їх у файловій системі, після чого повертає користувачу файл для завантаження.

Атрибути: клас не має власних атрибутів.

Операції зі специфікаціями нетривіальних операцій:

index() – відображає головну сторінку формування звітів.

generate(Request $request) – маршрутизує запит до відповідного типу звіту.

getDates(Request $request) – формує часовий інтервал для звіту.

salesReport(Request $request) – формує звіт по продажах.

arrivalsReport(Request $request) – формує звіт по надходженню товарів.

inventoryReport() – формує інвентаризаційний звіт.

Специфікація класу «SaleController»

Ім’я: SaleController.

Відповідальність: SaleController відповідає за повний цикл роботи з продажами в системі аптеки:

- відображення доступних товарів для продажу;
- створення нових продажів;
- формування позицій продажу;
- списання залишків товарів зі складу;
- перегляд історії продажів;
- зміна статусу продажу (активний/скасований).

Атрибути: клас не має власних атрибутів.

Операції зі специфікаціями нетривіальних операцій:

index() – відображає сторінку створення продажу.

store(Request $request) – створює новий продаж і всі пов’язані позиції.

history() – показує історію всіх продажів.

toggleStatus(Sale $sale) – змінює статус продажу.

Специфікація класу «UserController»

Ім’я: UserController.

Відповідальність: UserController відповідає за управління користувачами в системі аптеки. А саме:

- перегляд списку користувачів;
- створення нових користувачів;
- редагування даних користувачів;
- видалення користувачів;
- прив’язка користувачів до посад.

Атрибути: клас не має власних атрибутів.

Операції зі специфікаціями нетривіальних операцій:

index() – відображає список користувачів системи.

store(Request $request) – створює нового користувача.

update(Request $request, User $user) – оновлює дані користувача.

destroy(User $user) – видаляє користувача з системи.

Динамічна модель програмного забезпечення

Для моделювання роботи користувачів вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові» динамічна модель системи може бути представлена у вигляді діаграм послідовностей. Даний тип UML-діаграм належить до групи діаграм взаємодії та використовується для відображення процесу обміну повідомленнями між об’єктами системи в часовій послідовності.

Діаграми послідовностей дозволяють наочно представити порядок взаємодії користувачів із програмною системою, а також взаємодію між окремими компонентами вебзастосунку під час виконання певних функцій. За допомогою таких діаграм можна простежити послідовність викликів методів, передачу даних між клієнтською частиною, сервером та базою даних, а також визначити логіку виконання окремих бізнес-процесів [9].

Діаграма послідовності варіанту використання «Додавання препарату», наведена на рисунку 2.8.

Перш ніж переходити до фізичного проєктування бази даних та створення таблиць у СУБД, необхідно сформувати її логічну модель.

Логічна модель даних вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові» представлена на рисунку 2.9.

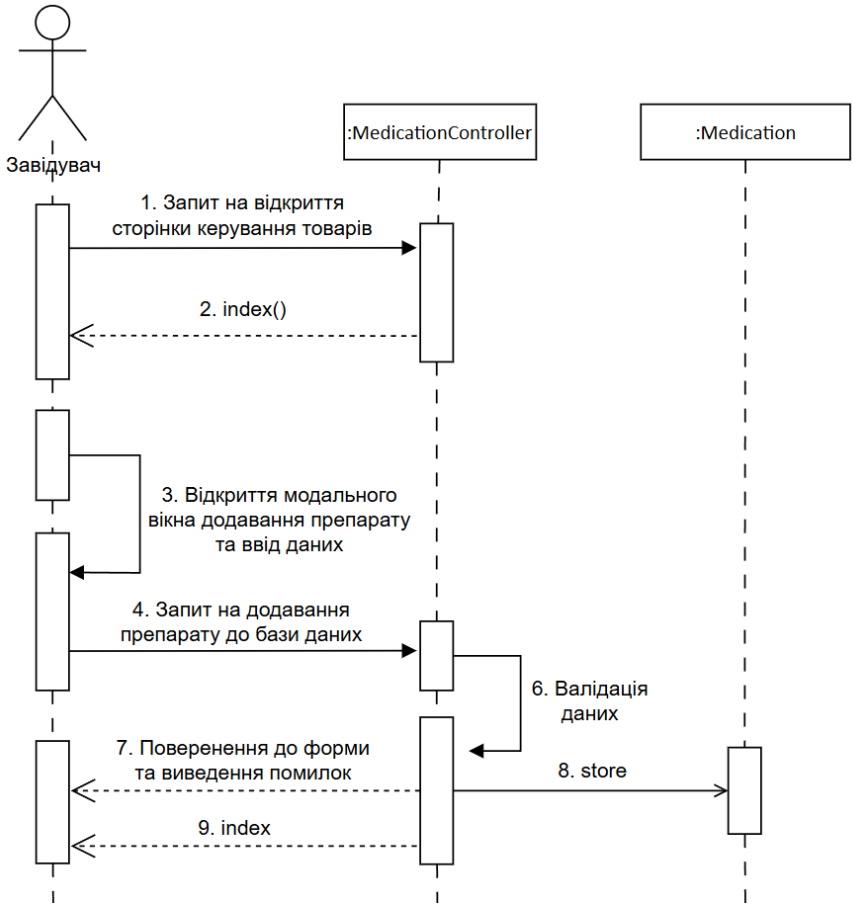


Рисунок 2.8 – Діаграма послідовності вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

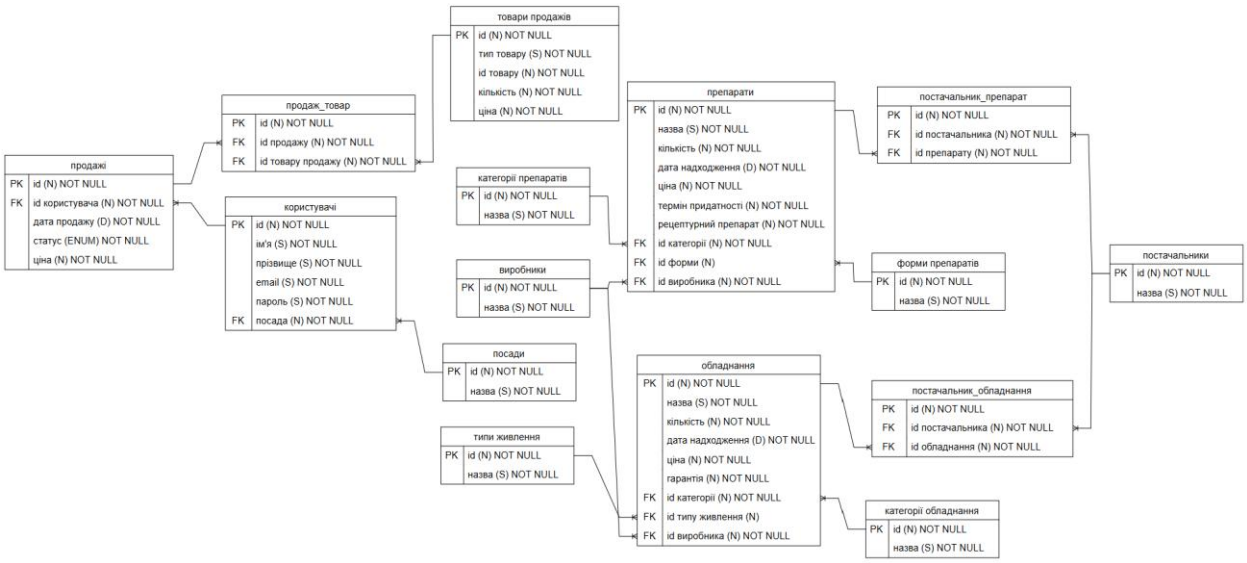


Рисунок 2.9 – Логічна модель даних вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

Для вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові» логічне проєктування бази даних ґрунтується на розробленій концептуальній моделі предметної області. Логічна модель даних являє собою абстрактне представлення інформації, незалежне від конкретної системи керування базами даних та особливостей її реалізації. На цьому рівні визначаються основні сутності системи, їх атрибути, а також зв’язки між ними.

У розробленій системі логічна модель охоплює дані про користувачів, посади, лікарські препарати, медичне обладнання, категорії товарів, виробників, постачальників, типи живлення, форми препаратів та продажі. Для кожної сутності визначено набір характеристик, необхідних для зберігання та обробки інформації. Крім того, встановлено зв’язки між сутностями, що забезпечують цілісність даних та підтримують виконання бізнес-процесів аптеки.

2.3.3 Робочий проект вебзастосунку

Фізична модель бази даних

Фізична модель даних є деталізованим представленням структури бази даних, яке описує спосіб зберігання даних у конкретній системі керування базами даних. Вона містить таблиці, поля, типи даних, первинні та зовнішні ключі, індекси, а також зв’язки між таблицями. Фізична модель даних створюється на основі логічної моделі [10].

Фізична модель бази даних вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові» представлена на рисунку 2.10.

Вибір мови програмування та СКБД

Для розробки вебзастосунку було обрано середовище розробки PhpStorm. PhpStorm – це професійне інтегроване середовище розробки (IDE) для мови програмування PHP, яке надає широкий набір інструментів для створення, тестування та супроводження вебзастосунків. Середовище підтримує автоматичне доповнення коду, рефакторинг, навігацію по проєкту, аналіз якості коду та засоби налагодження програм [11].

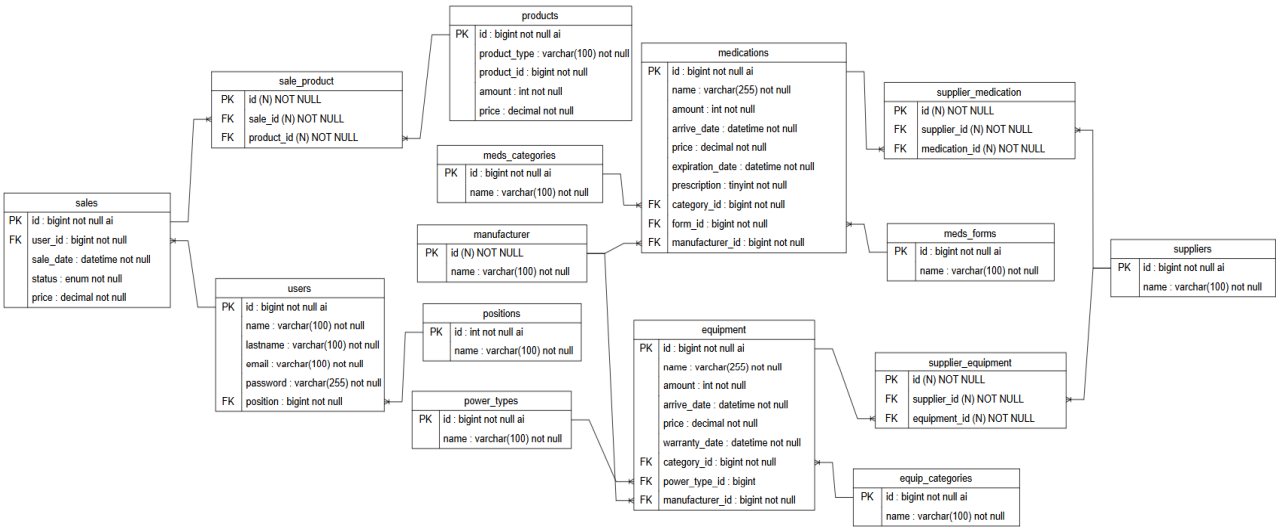


Рисунок 2.10 – Фізична модель бази даних вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

PhpStorm має вбудовану підтримку сучасних вебтехнологій, зокрема HTML, CSS, JavaScript і SQL, а також популярних фреймворків, таких як Laravel, Symfony та CodeIgniter. Крім того, середовище інтегрується із системами контролю версій Git, засобами керування базами даних та інструментами автоматизації розробки, що значно спрощує процес створення програмного забезпечення.

Для локальної розробки та тестування вебзастосунку використовувалося середовище Open Server Panel. Open Server Panel є програмним комплексом для операційної системи Windows, який надає готове середовище для запуску вебпроєктів без необхідності окремого встановлення та налаштування вебсервера, інтерпретатора PHP і системи керування базами даних. До складу Open Server Panel входять вебсервер Apache або Nginx, різні версії PHP, СУБД MySQL та інші інструменти, необхідні для розробки вебзастосунків [12].

Як СУБД було обрано MySQL. MySQL – це реляційна система керування базами даних, яка використовується для зберігання, обробки та керування структурованими даними. Вона забезпечує надійне зберігання інформації, підтримує мову структурованих запитів SQL та дозволяє ефективно виконувати операції додавання, пошуку, оновлення й видалення даних [13].

Для адміністрування бази даних використовувався вебінтерфейс phpMyAdmin. Це спеціалізований програмний засіб для роботи з базами даних MySQL через браузер, який дозволяє виконувати більшість операцій без написання SQL-запитів вручну.

За допомогою phpMyAdmin здійснювалося створення та перегляд таблиць бази даних, аналіз структури даних, контроль зв'язків між таблицями, перегляд та редагування записів, а також виконання SQL-запитів під час тестування роботи системи. Використання phpMyAdmin значно спростило процес адміністрування бази даних та дозволило оперативно контролювати правильність збереження інформації про користувачів, товари, продажі та інші об'єкти системи [14].

Завдяки широкій поширеності, сумісності з різними операційними системами та інтеграції з популярними мовами програмування і вебфреймворками, MySQL є одним із найпоширеніших рішень для реалізації серверної частини інформаційних систем. У межах даної роботи MySQL використовується для організації бази даних вебзастосунку та зберігання інформації про товари, продажі, користувачів та інші дані, необхідні для функціонування системи.

Для розробки вебзастосунку автоматизації роботи аптеки серед різних PHP-фреймворків, таких як Symfony, CodeIgniter та Yii, було обрано Laravel – один із найпопулярніших сучасних фреймворків для створення вебзастосунків. Laravel надає потужний набір інструментів для реалізації серверної логіки, роботи з базами даних, маршрутизації, автентифікації користувачів та обробки HTTP-запитів. Завдяки архітектурному шаблону MVC (Model-View-Controller) фреймворк забезпечує чіткий поділ відповідальностей між компонентами системи, що спрощує підтримку та подальший розвиток програмного забезпечення [15].

Для взаємодії з базою даних використовується ORM-система Eloquent, яка дозволяє працювати з таблицями бази даних у вигляді об’єктів та реалізовувати зв’язки між сутностями без написання складних SQL-запитів. Для управління

структурою бази даних застосовуються механізми міграцій та сідерів, що забезпечують контроль версій схеми даних та автоматичне наповнення початковими записами.

Для реалізації системи автентифікації та авторизації користувачів використовується Laravel Breeze – офіційний стартовий пакет Laravel. Breeze надає готові механізми входу до системи, виходу з облікового запису, управління профілем користувача, відновлення пароля та захисту маршрутів від несанкціонованого доступу.

Інтерфейс користувача реалізовано за допомогою Blade – шаблонізатора Laravel, який забезпечує зручне створення динамічних вебсторінок із використанням серверного рендерингу. Blade дозволяє використовувати умовні конструкції, цикли, компоненти та шаблони повторного використання, що сприяє структурованості та читабельності коду представлень.

Для оформлення інтерфейсу використовується CSS-фреймворк Tailwind CSS. На відміну від традиційних CSS-фреймворків, Tailwind пропонує набір готових утилітарних класів, які дозволяють швидко створювати адаптивний та сучасний дизайн без написання великої кількості власних стилів. Це забезпечує єдиний стиль оформлення сторінок, високу швидкість розробки та зручність подальшої підтримки інтерфейсу.

Для реалізації інтерактивних елементів інтерфейсу, таких як модальні вікна, випадаючі списки та динамічне керування станом сторінки, використовується бібліотека Alpine.js. Вона дозволяє додавати клієнтську логіку без створення складних JavaScript-застосунків та добре інтегрується з Blade і Tailwind CSS.

Для формування звітів у форматі PDF використовується бібліотека barryvdh/laravel-dompdf, яка інтегрується з Laravel та дозволяє генерувати PDF-документи на основі Blade-шаблонів. За її допомогою реалізовано створення звітів про продажі, надходження товарів та інвентаризаційних звітів аптеки.

Для написання вебзастосунку було використано такі засоби розробки:

– Мови програмування: PHP 8.1, JavaScript (ES6+).

– IDE / редактор коду: PhpStorm.
– Фреймворк для бекенду: Laravel 10.5.
– ORM для роботи з базою даних: Eloquent ORM.
– Система керування базою даних: MySQL.
– Середовище розробки: Open Server Panel.
– Інструмент адміністрування БД: phpMyAdmin.

Розглянемо контролер роботи з довідниками DirectoryController. Представимо код контролеру клієнта який буде тестуватися.

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
class DirectoryController extends Controller
{
   public function index(Request $request)
   {
      $type = $request->get('type', 'medications');
      $directories = config('directories');
      abort_if(!isset($directories[$type]), 404);
      $directory = $directories[$type];
      return view('directories.index', [
         'type' => $type,
         'title' => $directory['title'],
         'items' => $directory['model']::all(),
      ]);
   }
   public function store(Request $request)
   {
      $request->validate([
            'name' => 'required|string|max:255',
            'type' => 'required|string',
         ],
         [],
         [
            'name' => 'назви',
         ]
      );
      $directories = config('directories');
      abort_if(!isset($directories[$request->type]), 404);
      $model = $directories[$request->type]['model'];
      $model::create([
         'name' => $request->name,
```

```
        ]);

        return redirect()->back();
    }
    public function update(Request $request, $id)
    {
        $request->validateWithBag('editCategory', [
                'name' => 'required|string|max:255',
                'type' => 'required|string',
            ],
            [],
            [
                'name' => 'назви',
            ]
        );
        $directories = config('directories');
        abort_if(!isset($directories[$request->type]), 404);
        $model = $directories[$request->type]['model'];
        $item = $model::findOrFail($id);
        $item->update([
            'name' => $request->name,
        ]);
        return redirect()->back();
    }
    public function destroy(Request $request, $id)
    {
        $directories = config('directories');
        abort_if(!isset($directories[$request->type]), 404);
        $model = $directories[$request->type]['model'];
        $item = $model::findOrFail($id);
        $item->delete();
        return redirect()->back();
    }
}
```

Наведений контролер відповідає за обробку HTTP-запиту на додавання, редагування та видалення одного з довідників та вивід їх іншормації.

Пояснення структури і логіки роботи:

1. Метод index()

Метод index() відповідає за відображення сторінки довідників.

На початку метод отримує параметр «type» із HTTP-запиту. Якщо параметр не передано, за замовчуванням використовується значення «medications».

Далі з конфігураційного файлу directories.php завантажується список доступних довідників.

За допомогою функції abort_if() перевіряється, чи існує довідник із вказаним типом. Якщо такого довідника немає, користувачу повертається помилка 404.

Після цього визначається модель, яка відповідає обраному довіднику, та отримуються всі записи з відповідної таблиці бази даних.

Наприкінці метод передає отримані дані у представлення directories.index, де вони відображаються користувачу у вигляді таблиці.

2. Метод store()

Метод store() використовується для додавання нового запису до довідника.

Спочатку виконується валідація введених користувачем даних – перевіряється, що поле назви заповнене та не перевищує допустиму довжину, а також передано тип довідника.

Після успішної перевірки визначається модель відповідного довідника через конфігураційний файл.

Новий запис створюється за допомогою методу create(). Після збереження користувач повертається на попередню сторінку.

3. Метод update()

Метод update() відповідає за редагування існуючого запису довідника.

Виконується перевірка вхідних. Використання окремого контейнера помилок («editCategory») дозволяє відображати помилки саме у модальному вікні редагування.

Після перевірки визначається потрібна модель та здійснюється пошук запису за ідентифікатором.

Якщо запис не знайдено, Laravel автоматично повертає помилку 404.

Далі виконується оновлення даних, після чого користувач повертається на попередню сторінку.

4. Метод destroy()

Метод destroy() реалізує видалення записів із довідника.

Спочатку визначається відповідна модель через конфігураційний файл та перевіряється її наявність. Потім знаходиться запис за його ідентифікатором та виконується його видалення.

Після завершення операції користувач повертається на сторінку довідників.

Основна частина коду представлена в додатку Б.

Реалізація та тестування основних класів еквівалентності вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

Тестування програмного забезпечення – це процес перевірки програмного продукту з метою виявлення помилок та підтвердження його відповідності встановленим вимогам. Проведення тестування дозволяє оцінити якість програмного забезпечення та його готовність до експлуатації.

Тестування функції методом розбиття на еквівалентності:

Класи еквівалентності функції «Додавання препарату» представлені в таблиці 2.4.

Таблиця 2.4 – Класи еквівалентності функції «Додавання препарату»

Вхідні дані	Правильні класи еквівалентності	Неправильні класи еквівалентності
назва	1. символи, кількість яких не перевищує 255	11. символи, кількість яких перевищує 255 та порожнє поле
кількість	2. цілі числа, включаючи нуль	12. від'ємні та дробові числа, символи, порожнє поле
дата надходження	3. дата, в форматі ДД.ММ.РРРР	13. дата в неправильному форматі або порожнє поле
ціна	4. дробові та цілі числа	14. символи або порожнє поле
термін придатності	5. дата, в форматі ДД.ММ.РРРР	15. дата в неправильному форматі або порожнє поле
рецептурний препарат	6. ціле число	16. порожнє поле
id категорії	7. ціле число	17. порожнє поле
id форми	8. ціле число	18. порожнє поле
id постачальника	9. ціле число	19. порожнє поле
id виробника	10. ціле число	20. порожнє поле

Тестування правильних класів еквівалентності наведено в таблиці 2.5. За результатами тестування, функція «Додавання препарату» працює коректно.

Результат тестування представлено на рисунку 2.11.

Таблиця 2.5 – Тести для правильних класів еквівалентності

№ тесту	№ покритого класу	Вхідні дані	Результат
1	1	Пр3п4р4т	Рисунок 2.11
	2	34	
	3	27.05.2026	
	4	350,15	
	5	20.08.2026	
	6	1	
	7	2	
	8	3	
	9	1	
	10	2	

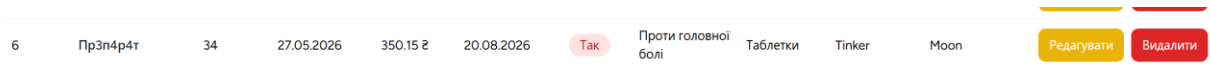


Рисунок 2.11 – Результат перевірки правильних класів еквівалентності

Неправильні класи еквівалентності представлені у таблиці 2.6. За результатами тестування функція «Додавання препарату» проводить правильну валідацію вхідних даних та виводить помилки. Результат наведено на рисунку 2.12.

Таблиця 2.6 – Тести для неправильних класів еквівалентності

№ тесту	№ покритого класу	Вхідні дані	Результат
2	11	порожнє поле	Рисунок 2.12
	12	-5	
	14	-550.57	
	19	порожнє поле	

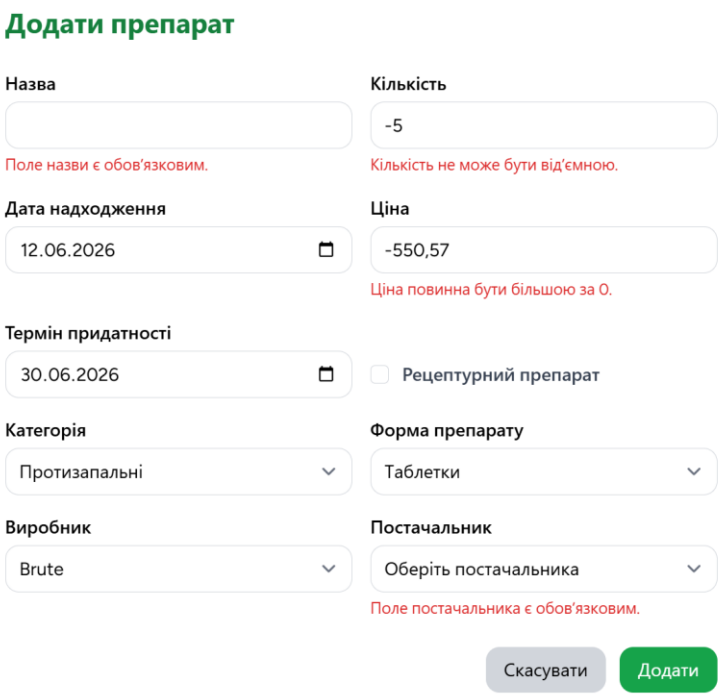


Рисунок 2.12 – Результат перевірки неправильних класів еквівалентності

Тестування вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові» виконувалося методом «чорного ящика». Тестування виконувалося за рахунок введення тестових наборів даних та аналізу отриманих результатів.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБЛІКУ ТОВАРІВ ТА ПРОДАЖУ В ПРИВАТНІЙ АПТЕЦІ «БУДЬТЕ ЗДОРОВІ»

Під час кваліфікаційної роботи була розв’язана практична задача інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробка вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові». Було розроблено вебзастосунок для автоматизації обліку товарів та продажу в приватній аптеці, що дозволить підвищити ефективність роботи персоналу, відповідального за ведення обліку лікарських засобів і оформлення продажів, а також зменшити кількість помилок під час обробки даних, контролю залишків та формування звітності.

Зовнішній вигляд сторінки керування товарами, наведено на рисунку 3.1.

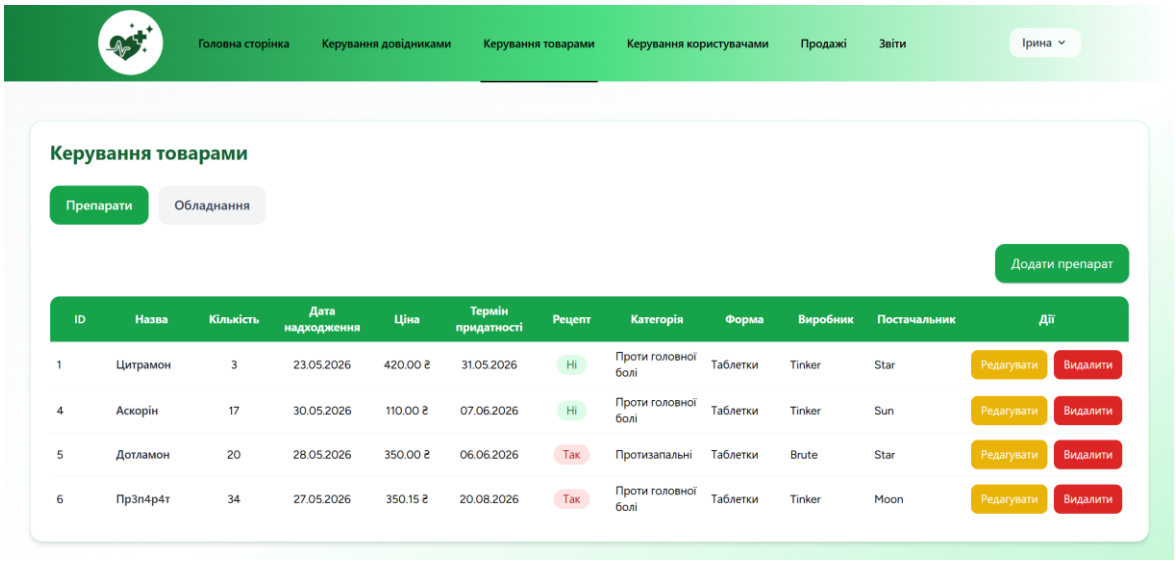


Рисунок 3.1 – Зовнішній вигляд сторінки керування товарами вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

Було проведено аналіз практичної задачі інженерії програмного

забезпечення, а саме досліджено особливості роботи приватної аптеки «Будьте здорові», пов’язані з обліком товарів, реєстрацією продажів та формуванням звітності. Також було проаналізовано існуючі програмні рішення для автоматизації діяльності аптечних закладів та сучасні підходи до автоматизації процесів обліку й продажу товарів. На основі проведеного аналізу було обґрунтовано доцільність розробки власного вебзастосунку, який забезпечить автоматизацію основних бізнес-процесів аптеки, підвищить ефективність роботи персоналу та зменшить ймовірність виникнення помилок під час обробки інформації.

У межах ескізного проєкту за допомогою мови моделювання UML було побудовано діаграми діяльності та концептуальну модель вебзастосунку. Також були визначені зв’язки між сутностями.

У межах технічного проєкту було побудовано діаграму класів, діаграму послідовності та логічну модель даних. Також було виконано специфікацію класів вебзастосунку.

У межах робочого проєкту було обрано та обґрунтовано інструменти розробки та системи управління базами даних, побудовано фізичну модель бази даних вебзастосунку, проведено тестування функції додавання препаратів.

Основною мовою програмування було обрано PHP. В якості СКБД для роботи з базою даних вебзастосунку виступила MySQL. Середовищем розробки було обрано PhpStorm.

При тестуванні та випробуванні вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові», було підтверджено коректність роботи реалізованих функціональних модулів та відповідність програмного забезпечення вимогам, визначеним у технічному завданні.

Також було розроблено комплект програмної документації, який включає технічне завдання, опис програми, інструкцію користувача, методику випробувань та інші документи, необхідні для впровадження, експлуатації та супроводження програмного забезпечення.

Вебзастосунок виконує наступні функції:

- авторизація користувача;
- продаж товарі;
- виконання CRUD-операцій з інформацією препаратів;
- виконання CRUD-операцій з інформацією обладнання;
- виконання CRUD-операцій з інформацією категорій товарів;
- виконання CRUD-операцій з інформацією форм препаратів;
- виконання CRUD-операцій з інформацією систем живлення;
- виконання CRUD-операцій з інформацією виробників;
- виконання CRUD-операцій з інформацією постачальників;
- виконання CRUD-операцій з інформацією користувачів;
- перегляд список продажів;
- скасування продажів;
- формування звітів.

Вебзастосунок було розроблено для приватної аптеки «Будьте здорові».

Програмне забезпечення впроваджено, акт впровадження наведено в додатку Е. Програму і методику випробувань наведено у Додатку Д.

Протокол випробувань

Тестування вебзастосунку виконувалося за методом «чорної скриньки», який передбачає перевірку функціональності системи без аналізу її внутрішньої структури та програмного коду. Враховуючи широкий функціонал розробленого програмного забезпечення, наведено результати тестування найбільш важливих функціональних модулів системи.

1. Перевірка відповідності розробленого вебзастосунку вимогам, визначеним у технічному завданні.

Перевірка роботи функції «Авторизація». При відкритті сайту, відкривається сторінка авторизації. У випадаючому списку обрати користувача та ввести пароль у відповідне поле. При натисканні кнопки керування «Увійти», враховуючи що пароль є валідним, виконується авторизація та відкривається головна сторінка вебсайту.

Перевірка роботи функції «Додавання препарат». При натисканні кнопки керування «Додати препарат» з’являється модальне вікно з формою для додавання препарату. Після заповнення всіх необхідних полів, натискається кнопка керування «Додати» та, якщо дані є валідними, препарат зберігається в базі даних.

2. Здійснено перевірку програмно-технічної сумісності вебзастосунку шляхом тестування його роботи на різних програмних та апаратних конфігураціях, що відповідають мінімальним вимогам, визначеним у технічному завданні. У процесі тестування було перевірено коректність функціонування системи в різних веббраузерах та операційних системах. Виявлені під час попередніх етапів тестування недоліки були усунуті та повторно перевірені.

3. Проведено завершальне тестування інтерфейсу користувача та функціональних можливостей системи. У ході перевірки виконано остаточне налагодження програмного забезпечення, спрямоване на виявлення та усунення незначних помилок, покращення зручності використання вебзастосунку та забезпечення стабільності його роботи.

4 ОХОРОНА ПРАЦІ

4.1 Огляд основних законодавчих і нормативних документів, які регламентують охорону праці в Україні

Охорона праці в Україні регулюється системою законодавчих і нормативно-правових актів, які визначають права та обов’язки працівників і роботодавців, а також встановлюють вимоги до безпечних і нешкідливих умов праці.

Основним нормативним документом у цій сфері є Закон України «Про охорону праці», який визначає загальні принципи державної політики у сфері охорони праці, гарантує право працівників на безпечні умови праці та покладає відповідальність на роботодавця за їх забезпечення. Закон регламентує організацію системи управління охороною праці, проведення навчання та інструктажів, а також контроль за дотриманням вимог безпеки [16].

Важливим нормативним актом є Кодекс законів про працю України, який містить окремий розділ, присвячений охороні праці. У ньому визначено обов’язок роботодавця створювати безпечні умови праці, а також встановлено вимоги щодо впровадження нових технологій і обладнання з урахуванням норм безпеки.

До основних законодавчих актів також належать:

– Закон України «Про загальнообов’язкове державне соціальне страхування», який регламентує соціальний захист працівників у разі нещасних випадків на виробництві [17];

– Закон України «Про пожежну безпеку», що визначає вимоги щодо запобігання пожежам та захисту працівників [18];

– інші нормативні документи, що регулюють окремі аспекти безпеки праці.

Окрім законів, важливу роль відіграють підзаконні нормативні акти,

зокрема:

– типові положення про службу охорони праці [19];

– положення про порядок навчання і перевірки знань з питань охорони праці;

– державні міжгалузеві та галузеві нормативні акти (правила, стандарти, інструкції), які є обов’язковими для виконання.

4.2 Структура управління питаннями охорони праці у приватній аптеці «Будьте здорові»

Система управління охороною праці на підприємстві є невід’ємною складовою загальної системи управління діяльністю організації та спрямована на забезпечення безпечних і нешкідливих умов праці, збереження життя і здоров’я працівників у процесі виконання ними трудових обов’язків. Для приватної аптеки «Будьте здорові», як суб’єкта господарювання у сфері роздрібної торгівлі лікарськими засобами, питання охорони праці є особливо важливими, оскільки пов’язані як з роботою персоналу, так і з обслуговуванням клієнтів.

Структура управління охороною праці на підприємстві визначається відповідно до чинного законодавства України та внутрішніх нормативних документів і включає систему організаційних заходів, розподіл обов’язків між посадовими особами, а також механізми контролю за дотриманням вимог безпеки праці.

Загальне керівництво системою охорони праці в аптеці здійснює керівник підприємства (завідувач аптеки), який несе повну відповідальність за створення безпечних і нешкідливих умов праці. До його основних обов’язків належать:

– організація роботи з охорони праці на підприємстві;

– забезпечення виконання вимог законодавчих і нормативних актів;

– створення належних умов праці для працівників;

– організація навчання та інструктажів з питань охорони праці;

– контроль за дотриманням працівниками встановлених правил і норм.

Завідувач аптеки також забезпечує розробку та впровадження внутрішніх нормативних документів, таких як інструкції з охорони праці, положення про порядок проведення інструктажів, правила пожежної безпеки та інші документи, що регламентують безпечну діяльність персоналу.

У невеликих підприємствах, до яких належить приватна аптека, функції служби охорони праці, як правило, покладаються на відповідальну особу, призначену наказом керівника. Така особа виконує функції організації та контролю заходів з охорони праці, зокрема:

– проведення вступного інструктажу з новими працівниками;

– організація та контроль проведення первинного, повторного та позапланового інструктажів;

– ведення відповідної документації з охорони праці;

– участь у розслідуванні нещасних випадків (у разі їх виникнення);

– контроль за дотриманням вимог безпеки праці на робочих місцях.

Працівники аптеки також є учасниками системи управління охороною праці та зобов’язані:

– дотримуватися вимог інструкцій з охорони праці;

– правильно використовувати обладнання та засоби праці;

– проходити навчання та інструктажі;

– повідомляти керівництво про виявлені небезпечні ситуації або несправності обладнання.

Важливим елементом системи управління охороною праці є організація навчання та інструктажів. Усі працівники аптеки проходять вступний інструктаж перед початком роботи, а також первинний інструктаж на робочому місці. У подальшому проводяться повторні інструктажі з метою закріплення знань та перевірки рівня обізнаності працівників щодо вимог безпеки. У разі змін у технологічному процесі або введення нового обладнання проводяться позапланові інструктажі.

Окрему увагу в аптечному закладі приділяють питанням пожежної

безпеки, оскільки в приміщенні можуть зберігатися легкозаймисті матеріали. Для цього розробляються відповідні інструкції, встановлюються засоби пожежогасіння, а працівники проходять навчання щодо дій у разі виникнення пожежі.

Контроль за станом охорони праці здійснюється на постійній основі. Він включає перевірку робочих місць, оцінку умов праці, контроль за дотриманням правил експлуатації обладнання та використанням засобів індивідуального захисту. У разі виявлення порушень вживаються заходи щодо їх усунення.

В умовах використання комп'ютерної техніки, що є невід'ємною частиною роботи сучасної аптеки, особливого значення набуває дотримання вимог безпеки при роботі з персональними комп'ютерами. Це включає правильну організацію робочого місця, дотримання режиму праці та відпочинку, а також забезпечення належного освітлення та ергономічних умов.

Таким чином, структура управління охороною праці в приватній аптеці «Будьте здорові» являє собою комплекс взаємопов'язаних заходів, спрямованих на забезпечення безпечних умов праці. Ефективне функціонування цієї системи досягається за рахунок чіткого розподілу обов'язків, постійного контролю та відповідального ставлення всіх працівників до питань безпеки.

4.3 Розробка заходів по зменшенню дії небезпечних і шкідливих факторів в приватній аптеці «Будьте здорові»

Діяльність працівників приватної аптеки «Будьте здорові» пов'язана з впливом різноманітних небезпечних і шкідливих виробничих факторів, які можуть негативно впливати на їх здоров'я та працездатність. До таких факторів належать фізичні, хімічні, психофізіологічні та ергономічні чинники. У зв'язку з цим важливим завданням є розробка комплексу заходів, спрямованих на зменшення або усунення їх негативного впливу.

До фізичних факторів належать недостатнє або надмірне освітлення, шум,

мікроклімат приміщення, а також електромагнітне випромінювання від комп’ютерної техніки.

Для забезпечення належних умов освітлення необхідно використовувати комбіноване освітлення, що включає природне та штучне. Робочі місця повинні бути розташовані таким чином, щоб уникнути прямих відблисків на екранах моніторів. Рівень освітленості має відповідати санітарним нормам і становити не менше 300–500 лк.

З метою зменшення шумового навантаження необхідно використовувати сучасне обладнання з низьким рівнем шуму, а також забезпечити раціональне розміщення технічних засобів у приміщенні.

Мікроклімат у приміщенні аптеки повинен підтримуватися на оптимальному рівні: температура повітря – 18–24°С, відносна вологість – 40–60%. Для цього необхідно використовувати системи вентиляції та кондиціонування повітря.

Зменшення впливу електромагнітного випромінювання досягається шляхом використання сучасної сертифікованої комп’ютерної техніки, дотримання нормативних відстаней між робочими місцями та правильного розташування обладнання.

У процесі діяльності аптеки працівники можуть контактувати з лікарськими засобами та хімічними речовинами, що може становити потенційну небезпеку.

Для зменшення впливу хімічних факторів необхідно:

– забезпечити зберігання лікарських засобів відповідно до встановлених вимог (температурний режим, вологість, захист від світла);

– використовувати герметичну упаковку та спеціальні контейнери для небезпечних речовин;

– дотримуватися правил особистої гігієни;

– застосовувати засоби індивідуального захисту (рукавиці, за потреби – маски);

– забезпечити наявність інструкцій щодо безпечного поводження з

лікарськими засобами.

Більшість працівників аптеки працює за комп’ютером або виконує операції, пов’язані з тривалим перебуванням у статичному положенні, що може призводити до перевтоми та захворювань опорно-рухового апарату.

Для мінімізації впливу ергономічних факторів необхідно:

– забезпечити правильну організацію робочого місця відповідно до ергономічних вимог;

– використовувати регульовані стільці та столи;

– розміщувати монітор на відстані 50–70 см від очей користувача;

– забезпечити правильне розташування клавіатури та миші;

– дотримуватися режиму праці та відпочинку (перерви кожні 1–2 години роботи).

До психофізіологічних факторів належать емоційне напруження, стрес, перевтома, що можуть виникати під час обслуговування клієнтів та виконання відповідальних операцій.

Для зниження впливу цих факторів необхідно:

– забезпечити раціональний режим праці та відпочинку;

– оптимізувати робоче навантаження;

– створити сприятливий психологічний клімат у колективі;

– впроваджувати сучасні програмні засоби, що спрощують виконання рутинних операцій (зокрема вебзастосунок для обліку товарів і продажу);

– проводити інструктажі та навчання працівників.

Оскільки в аптеці використовується електронне обладнання, важливим є дотримання вимог електробезпеки. Необхідно забезпечити:

– справний технічний стан електрообладнання;

– наявність заземлення;

– регулярну перевірку електромережі;

– проведення інструктажів з електробезпеки;

– заборону використання несправних електроприладів.

Для забезпечення пожежної безпеки необхідно:

– обладнати приміщення вогнегасниками;
– забезпечити наявність планів евакуації;
– проводити навчання персоналу діям у разі пожежі;
– дотримуватися правил зберігання легкозаймистих речовин;
– контролювати стан електропроводки.

Важливим заходом зниження впливу небезпечних факторів є впровадження сучасного програмного забезпечення. Використання вебзастосунку для автоматизації обліку товарів і продажу дозволяє:

– зменшити фізичне та психоемоційне навантаження на працівників;
– знизити ймовірність помилок при обробці даних;
– скоротити час виконання операцій;
– підвищити ефективність роботи персоналу.

Розробка та впровадження комплексу організаційних, технічних і санітарно-гігієнічних заходів дозволяє значно зменшити вплив небезпечних і шкідливих факторів у приватній аптеці «Будьте здорові». Дотримання вимог охорони праці, правильна організація робочих місць та використання сучасних інформаційних технологій сприяють створенню безпечних умов праці та підвищенню ефективності діяльності підприємства.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було вирішено практичну задачу інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме виконано розробку вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»:

– визначено та проаналізовано основні бізнес-процеси аптеки, що підлягають автоматизації;

– проведено аналіз існуючих програмних рішень для автоматизації обліку товарів і продажу, визначено їх переваги та недоліки;

– обґрунтовано доцільність розробки власного програмного забезпечення для автоматизації діяльності аптечного закладу;

– сформовано функціональні та нефункціональні вимоги до вебзастосунку;

– розроблено проєкт програмного забезпечення, який включає архітектуру системи, структуру бази даних, діаграми та інші проєктні рішення;

– реалізовано функціональні модулі керування товарами, обліку продажів, формування звітності та авторизації користувачів;

– забезпечено інтегровану роботу всіх компонентів системи для підтримки основних бізнес-процесів аптеки;

– проведено тестування вебзастосунку, результати якого підтвердили працездатність та відповідність програмного забезпечення встановленим вимогам.

У результаті виконання роботи створено вебзастосунок, який дозволяє автоматизувати процеси обліку товарів і продажу в приватній аптеці «Будьте здорові», підвищити ефективність роботи персоналу, зменшити кількість помилок під час обробки інформації та забезпечити швидкий доступ до необхідних даних.

Окремо було розглянуто питання охорони праці, проаналізовано структуру управління охороною праці в аптечному закладі та запропоновано заходи щодо зменшення впливу небезпечних і шкідливих факторів на працівників.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pharmacy Management System. URL: https://ijsred.com/volume8/issue3/IJSRED-V8I3P626.pdf (дата звернення: 12.03.2026).

2. Pharmacy management system Requirement Analysis and Elicitation Document. URL: https://www.slideshare.net/slideshow/pharmacy-management-system-requirement-analysis-and-elicitaion-document/49033178 (дата звернення: 12.03.2026).

3. Pharmacy Management Software: What, How, When, and More! URL: https://radixweb.com/blog/guide-on-pharmacy-management-software (дата звернення: 12.03.2026).

4. SwilERP is a software that seeks to provide intelligent business solutions for retail pharmacy shops. URL: https://www.trustradius.com/products/swilerp-software/reviews (дата звернення: (12.03.2026).

5. Unisolve software, developed by Softworld India Pvt Ltd's in-house development team, is a comprehensive management system tailored to meet the needs of pharmaceutical wholesale and distribution industries. URL: https://www.trustradius.com/products/unisolve-software/reviews (дата звернення: 12.03.2026).

6. Діаграма розгортання: Підручник з UML із прикладом. URL: https://www.guru99.com/uk/deployment-diagram-uml-example.html (дата звернення: 14.05.2026).

7. Діаграма діяльності в UML: символ, компоненти та приклад. URL: https://www.guru99.com/uk/uml-activity-diagram.html (дата звернення: 14.05.2026).

8. Побудова концептуальної моделі БД. URL: https://studfile.net/preview/5171241/page:4/ (дата звернення: 14.05.2026).

9. Діаграма послідовності (Sequence diagram). URL:

https://studfile.net/preview/9183836/page:25/ (дата звернення: 14.05.2026).

10. Проектування фізичної моделі даних. URL: https://studfile.net/preview/6329372/page:10/ (дата звернення: 15.05.2026).

11. Офіційний сайт PhpStorm. URL: https://www.jetbrains.com/phpstorm/ (дата звернення: 17.05.2026);

12. Офіційний сайт Open Server Panel. URL: https://ospanel.io/ (дата зверення: 17.05.2026);

13. Офіційний сайт MySQL. URL: https://www.mysql.com/ (дата звернення: 17.05.2026);

14. Офіційний сайт phpMyAdmin. URL: https://www.phpmyadmin.net/ (дата зверення: 17.05.2026);

15. Laravel Documentation. URL: https://laravel.com/docs/10.x/installation (дата звернення: 17.05.2026).

16. Про охорону праці: Закон України від 14.10.1992 № 2694-XII. Редакція від 12.09.2025. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 12.03.2026).

17. Про загальнообов’язкове державне соціальне страхування: Закон України від 23.09.1999 № 1105-XIV. Редакція від 01.01.2026. URL: https://zakon.rada.gov.ua/laws/show/1105-14/ed20260101#Text (дата звернення: 12.03.2026).

18. Про затвердження вимог щодо пожежної безпеки: Кодекс цивільного захисту України від 02.10.2012 №5403-VI. Редакція від 08.03.2026. URL: https://zakon.rada.gov.ua/laws/show/5403-17/ed20260308#Text (дата звернення: 12.03.2026).

19. Типове положення про службу охорони праці: Наказ державного комітету України з нагляду за охороною праці від 15.11.2004 № z1526-04. Редакція від 14.04.2017. URL: https://zakon.rada.gov.ua/laws/show/z1526-04#Text (дата звернення: 12.03.20206).

Додаток А Технічне завдання на розробку вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

Вступ

Повна назва продукту: вебзастосунок для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові». Область застосування – приватна аптека «Будьте здорові».

1. Підстава для розробки

Підставою для розробки даного програмного забезпечення є наказ про затвердження тем кваліфікаційних робіт бакалаврів за спеціальністю 121 «Інженерія програмного забезпечення» у Національному університеті кораблебудування імені адмірала Макарова, а також завдання на виконання кваліфікаційної роботи на тему «Розробка вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»».

2. Призначення розробки

2.1 Експлуатаційне призначення

Розроблюваний вебзастосунок призначений для використання в умовах приватної аптеки «Будьте здорові» з метою автоматизації основних бізнес-процесів, пов’язаних з обліком товарів та реалізацією продукції.

Використання програмного забезпечення сприятиме підвищенню ефективності роботи персоналу за рахунок зменшення часу на виконання рутинних операцій, зниження ймовірності помилок при обробці даних та оптимізації процесів обліку і продажу. Крім того, впровадження вебзастосунку дозволить покращити якість обслуговування клієнтів шляхом прискорення процесу продажу, забезпечення оперативного доступу до інформації про наявність товарів та підвищення загального рівня організації роботи аптеки.

2.2 Функціональне призначення

Розроблюваний вебзастосунок призначений для автоматизації основних функцій, пов’язаних з обліком товарів та здійсненням продажу в приватній аптеці «Будьте здорові».

Програмне забезпечення забезпечує виконання таких функцій:

– авторизація користувачів;
– продаж товару;
– виконання CRUD-операцій з інформацією препаратів;
– виконання CRUD-операцій з інформацією обладнання;
– виконання CRUD-операцій з інформацією категорій товарів;
– виконання CRUD-операцій з інформацією форм препаратів;
– виконання CRUD-операцій з інформацією систем живлення;
– виконання CRUD-операцій з інформацією виробників;
– виконання CRUD-операцій з інформацією постачальників;
– виконання CRUD-операцій з інформацією користувачів;
– перегляд списку продажів;
– формування звітів;
– вихід з облікового запису.

3. Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Вебзастосунок буде мати два типи користувачів: фармацевт та завідувач.

Фармацевт може лише переглядати та продавати товари, виконувати авторизацію.

Завідувач має доступ до всіх функцій застосунку, а саме:

− авторизація користувача;

Вхідні дані: логін, пароль.

Результат: вхід користувача до облікового запису певної ролі.

− продаж товару;

Вхідні дані: ID, ID товарів, дата продажу (хвилини, години, день, місяць, рік), ціна продажі, продавець, кількість, статус.

Результат: чек продажу в базі даних.

− перегляд препаратів;

Вхідні дані: ID препарату.

Результат: інформація певного препарату (ID, назва, дата надходження, кількість, ціна поставки, ціна продажу, термін придатності, категорія, форма, виробник, постачальник).

− додавання препаратів;

Вхідні дані: ID, назва, дата надходження, кількість, ціна поставки, ціна продажу, термін придатності, категорія, форма, виробник, постачальник.

Результат: новий препарат в базі даних.

− редагування препаратів;

Вхідні дані: назва, дата надходження, кількість, ціна поставки, ціна продажу, термін придатності, категорія, форма, виробник, постачальник.

Результат: препарат в базі даних зі зміненою інформацією.

− видалення препаратів;

Вхідні дані: ID препарату.

Результат: база даних без видаленого препарату.

− перегляд обладнання;

Вхідні дані: ID обладнання.

Результат: інформація певного обладнання (ID, назва, дата надходження, кількість, ціна поставки, ціна продажу, тип живлення, категорія, виробник, постачальник).

− додавання обладнання;

Вхідні дані: ID, назва, дата надходження, кількість, ціна поставки, ціна продажу, тип живлення, категорія, виробник, постачальник.

Результат: нове обладнання в базі даних.

− редагування обладнання;

Вхідні дані: назва, дата надходження, кількість, ціна поставки, ціна продажу, тип живлення, категорія, виробник, постачальник.

Результат: обладнання в базі даних зі зміненою інформацією.

− видалення обладнання;

Вхідні дані: ID обладнання.

Результат: база даних без видаленого обладнання.

− додавання категорії товарів;

Вхідні дані: ID, назва.

Результат: нова категорія в базі даних.

− редагування обладнання;

Вхідні дані: назва.

Результат: категорія в базі даних зі зміненою інформацією.

− видалення категорії;

Вхідні дані: ID категорії.

Результат: база даних без видаленої категорії.

− додавання форми;

Вхідні дані: ID, назва.

Результат: нова форма в базі даних.

− редагування форма;

Вхідні дані: назва.

Результат: форма в базі даних зі зміненою інформацією.

− видалення форми;

Вхідні дані: ID форми.

Результат: база даних без видаленої форми.

− додавання системи живлення;

Вхідні дані: ID, назва.

Результат: нова система живлення в базі даних.

− редагування системи живлення;

Вхідні дані: назва.

Результат: система живлення в базі даних зі зміненою інформацією.

− видалення системи живлення;

Вхідні дані: ID системи живлення.

Результат: база даних без видаленої системи живлення.

− додавання виробника;

Вхідні дані: ID, назва.

Результат: новий виробник в базі даних.

− редагування виробника;

Вхідні дані: назва.

Результат: виробник в базі даних зі зміненою інформацією.

− видалення виробника;

Вхідні дані: ID виробника.

Результат: база даних без видаленого виробника.

− додавання постачальника;

Вхідні дані: ID, назва.

Результат: новий постачальник в базі даних.

− редагування постачальника;

Вхідні дані: назва.

Результат: постачальник в базі даних зі зміненою інформацією.

− видалення постачальника;

Вхідні дані: ID постачальника.

Результат: база даних без видаленого постачальника.

− додавання користувача;

Вхідні дані: ID, ім’я, прізвище, пароль, посада.

Результат: новий користувач в базі даних.

− редагування користувача;

Вхідні дані: ім’я, прізвище, пароль, посада.

Результат: користувач в базі даних зі зміненою інформацією.

− видалення користувача;

Вхідні дані: ID користувача.

Результат: база даних без видаленого користувача.

− перегляд списку продажів.

Вхідні дані: ID продажу.

Результат: дані продажу (ID, ID товарів, дата продажу (хвилини, години, день, місяць, рік), ціна продажі, продавець, кількість).

− зміна статусу продажу.

Вхідні дані: ID продажу.

Результат: продаж в базі даних зі зміненим статусом.

В застосунку має бути можливість для завідувача створювати звіти:

– звіт по продажам – звіт, в якому буде інформація про кількість чеків, проданих товарів та виручку, отриману за певний період часу. Можна створювати за день, тиждень, місяць, або інший затребуваний період;

– звіт надходження – звіт, в якому буде інформація про товари, які були закуплені в певний період (дата надходження, назва, кількість, виробник, постачальник, ціна за якою було придбано товар);

– інвентаризаційний звіт – звіт, в якому буде інформація про товари, які на даний момент є в аптеці, та товари, термін придатності яких має скоро завершитись (унікальний номер, назва, кількість, термін придатності, виробник);

3.1.2 Вимоги до організації вхідних та вихідних даних

Зберігання даних здійснюється за допомогою реляційної бази даних, зокрема системи управління базами даних MySQL. Структура бази даних повинна забезпечувати цілісність, узгодженість та надійність збереження інформації про товари, операції продажу та користувачів вебзастосунку.

Вхідні дані:

Введення даних здійснюється користувачами за допомогою стандартних пристроїв введення, таких як клавіатура та миша. Дані вводяться через інтерфейс вебзастосунку у відповідні форми. Передбачено можливість як ручного введення інформації, так і вибору значень із випадаючих списків, що формуються на основі наявних даних у застосунку. Введені дані повинні проходити перевірку на коректність (валідацію) перед їх збереженням у базі даних.

Вихідні дані:

Виведення даних здійснюється на екран користувача через інтерфейс вебзастосунку у вигляді таблиць, форм і звітів. Інформація відображається в режимі реального часу відповідно до запитів користувача. Передбачено можливість перегляду, пошуку та фільтрації даних, а також формування звітної інформації у зручному для аналізу вигляді.

3.2 Вимоги до надійності

Розроблюваний вебзастосунок повинен забезпечувати надійну та безперебійну роботу під час виконання основних функцій, пов’язаних з обліком товарів та реєстрацією продажів.

Програма повинна забезпечувати:

– коректну обробку вхідних даних та запобігання виникненню помилок під час їх введення за рахунок використання механізмів валідації;

– збереження цілісності та узгодженості даних у базі даних при виконанні операцій додавання, редагування та видалення інформації;

– стійкість до помилок користувача, зокрема обробку некоректних або неповних даних без порушення роботи застосунку;

– захист від втрати даних у разі збоїв шляхом використання механізмів резервного копіювання бази даних;

– відновлення працездатності програми після збоїв без втрати критично важливої інформації;

– обмеження доступу до вебзастосунку через механізми авторизації та аутентифікації користувачів.

3.3 Вимоги до експлуатації

Розроблюваний вебзастосунок повинен забезпечувати зручну та безпечну експлуатацію в умовах роботи приватної аптеки. Програмна система має функціонувати у стандартних умовах використання персональних комп’ютерів та мобільних пристроїв за наявності доступу до мережі Інтернет.

Експлуатація програмного забезпечення не повинна вимагати спеціальної підготовки користувачів, окрім базових навичок роботи з комп’ютером і веббраузером. Інтерфейс вебзастосунку має бути інтуїтивно зрозумілим та забезпечувати швидке виконання основних операцій.

Програма повинна підтримувати безперервну роботу протягом робочого часу аптеки, а також забезпечувати можливість технічного обслуговування, оновлення та резервного копіювання даних без значного впливу на робочий процес.

3.4 Вимоги до складу та параметрів технічних засобів

Для забезпечення функціонування вебзастосунку необхідні серверні та клієнтські технічні засоби, а також мережеве з’єднання.

Серверна частина повинна працювати на комп’ютері або сервері з процесором не нижче 2.0 ГГц, оперативною пам’яттю не менше 4 ГБ та під керуванням операційної системи Windows (Windows 7 або вище), що підтримує роботу вебсерверного програмного забезпечення.

Клієнтська частина передбачає використання персональних комп’ютерів або мобільних пристроїв з доступом до мережі Інтернет, оперативною пам’яттю не менше 2 ГБ та сучасним веббраузером.

Технічні засоби повинні забезпечувати стабільну роботу програмної системи та можливість одночасного доступу кількох користувачів.

3.5 Вимоги до інформаційної та програмної сумісності

Розроблюваний вебзастосунок повинен забезпечувати інформаційну та програмну сумісність з сучасними програмними та технічними засобами.

Інформаційна сумісність передбачає використання уніфікованих форматів представлення та зберігання даних, що забезпечують їх цілісність, узгодженість та можливість подальшої обробки. Дані повинні зберігатися у реляційній базі даних та оброблятися відповідно до встановлених правил і обмежень. Обмін даними між клієнтською та серверною частинами повинен здійснюватися у стандартних форматах, зокрема JSON.

Програмна сумісність передбачає можливість функціонування вебзастосунку на різних операційних системах (Linux, Windows) та використання сучасних веббраузерів (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari або аналогічних). Програмне забезпечення повинно коректно працювати незалежно від програмного середовища клієнтського пристрою.

Розроблюване програмне забезпечення повинно забезпечувати можливість подальшого розширення та інтеграції з іншими програмними системами за необхідності.

4. Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

− технічне завдання;

− програму та методику тестувань;

− керівництво користувача;

− опис програми;

− код програми.

5 Техніко-економічні показники

У даній роботі техніко-економічні показники не розраховуються.

6 Етапи роботи

Етапи розробки та терміни виконання представлені у таблиці А.1

Таблиця А.1 – Стадії виконання робіт

Номер	Назва етапів роботи	Дати виконання
1.	Аналіз практичної задачі інженерії ПЗ	16.02.2026
2.	Аналіз та формування вимог ПЗ	27.03.2026
3.	Проєктування ПЗ	09.04.2026
4.	Розробка проекту ПЗ	14.04.2026
5.	Реалізація та тестування ПЗ	29.04.2026

7. Порядок контролю та приймання

На кожному етапі розробки програмного забезпечення здійснюється контроль процесів аналізу, проєктування та реалізації складових вебзастосунку з урахуванням вимог, визначених у технічному завданні. Кожен етап розробки повинен бути виконаний у встановлені терміни та узгоджений із керівником роботи або замовником.

Приймання програмного забезпечення проводиться відповідно до затвердженої методики та програми випробувань. Результати перевірки фіксуються у відповідних документах, зокрема протоколах тестування та перевірки працездатності програмного забезпечення.

У разі виявлення недоліків або помилок під час приймання програмного продукту складається акт, у якому зазначаються виявлені зауваження. Даний акт підписується відповідальними особами та є підставою для доопрацювання програмного забезпечення. Розробник зобов’язаний усунути виявлені недоліки у встановлені терміни та надати програмний продукт для повторного контролю.

Додаток Б Код вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

Код файлу EquipmentController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\EquipCategory;
use App\Models\Equipment;
use App\Models\Manufacturer;
use App\Models\PowerType;
use App\Models\Supplier;
use Illuminate\Http\Request;

class EquipmentController extends Controller
{
   public function index()
   {
      $equipment = Equipment::with([
         'category',
         'powerType',
         'manufacturer',
         'supplier',
      ])->get();
      $equipCategories = EquipCategory::all();
      $powerTypes = PowerType::all();
      $manufacturers = Manufacturer::all();
      $suppliers = Supplier::all();
      return view('products.index', compact(
         'equipment',
         'equipCategories',
         'powerTypes',
         'manufacturers',
         'suppliers'
      ));
   }

   public function store(Request $request)
   {
      $request->validate([
         'name' => 'required|string|max:255',
         'amount' => 'required|integer|min:0',
         'arrive_date' => 'required|date',
         'price' => 'required|numeric|min:0',
         'warranty_date' => 'nullable|date',
         'category_id' => 'required|exists:equip_categories,id',
```

```
            'power_type_id' => 'nullable|exists:power_types,id',
            'manufacturer_id' => 'required|exists:manufacturers,id',
            'supplier_id' => 'required|exists:suppliers,id',
        ]);

        Equipment::create($request->all());

        return redirect()->back();
    }

    public function update(Request $request, Equipment $equipment)
    {
        $request->validate([
            'name' => 'required|string|max:255',
            'amount' => 'required|integer|min:0',
            'arrive_date' => 'required|date',
            'price' => 'required|numeric|min:0',
            'warranty_date' => 'nullable|date',
            'category_id' => 'required|exists:equip_categories,id',
            'power_type_id' => 'nullable|exists:power_types,id',
            'manufacturer_id' => 'required|exists:manufacturers,id',
            'supplier_id' => 'required|exists:suppliers,id',
        ]);

        $equipment->update($request->all());

        return redirect()->back();
    }

    public function destroy(Equipment $equipment)
    {
        $equipment->delete();

        return redirect()->back();
    }
}
```

Код файлу MedicationController.php

```
<?php
namespace App\Http\Controllers;

use App\Models\Manufacturer;
use App\Models\Medication;
use App\Models\MedsCategory;
use App\Models\MedsForm;
use App\Models\Supplier;
use Illuminate\Http\Request;

class MedicationController extends Controller
```

```
{
    public function index()
    {
        $medications = Medication::with([
            'category',
            'form',
            'manufacturer',
            'supplier',
        ])->get();
        $medCategories = MedsCategory::all();
        $medForms = MedsForm::all();
        $manufacturers = Manufacturer::all();
        $suppliers = Supplier::all();
        return view('products.index', compact(
            'medications',
            'medCategories',
            'medForms',
            'manufacturers',
            'suppliers'
        ));
    }

    public function store(Request $request)
    {
        $request->validate(
            [
                'name' => 'required|string|max:255',
                'amount' => 'required|integer|min:0|max:999999',
                'arrive_date' => 'required|date',
                'price' => 'required|numeric|gt:0|max:999999.99',
                'expiration_date' => 'required|date',
                'prescription' => 'required|boolean',
                'category_id' => 'required|exists:meds_categories,id',
                'form_id' => 'required|exists:meds_forms,id',
                'manufacturer_id' => 'required|exists:manufacturers,id',
                'supplier_id' => 'required|exists:suppliers,id',
            ],
            [
                'amount.required' => 'Поле кількості є обов’язковим.',
                'amount.integer' => 'Кількість повинна бути цілим числом.',
                'amount.min' => 'Кількість не може бути від’ємною.',
                'amount.max' => 'Кількість перевищує допустиме значення.',

                'price.required' => 'Поле ціни є обов’язковим.',
                'price.numeric' => 'Ціна повинна бути числом.',
                'price.gt' => 'Ціна повинна бути більшою за 0.',
                'price.max' => 'Ціна перевищує допустиме значення.',
            ],
            [
```

```
            'name' => 'назви',
            'amount' => 'кількості',
            'arrive_date' => 'дати надходження',
            'price' => 'ціни',
            'expiration_date' => 'терміну придатності',
            'category_id' => 'категорії',
            'form_id' => 'форми препарату',
            'manufacturer_id' => 'виробника',
            'supplier_id' => 'постачальника',
        ]
    );
    Medication::create($request->all());
    return redirect()->back();
  }
  public function update(Request $request, Medication $medication)
  {
    $request->validate([
      'name' => 'required|string|max:255',
      'amount' => 'required|integer|min:0',
      'arrive_date' => 'required|date',
      'price' => 'required|numeric|min:0',
      'expiration_date' => 'required|date',
      'prescription' => 'required|boolean',
      'category_id' => 'required|exists:meds_categories,id',
      'form_id' => 'required|exists:meds_forms,id',
      'manufacturer_id' => 'required|exists:manufacturers,id',
      'supplier_id' => 'required|exists:suppliers,id',
    ]);
    $medication->update($request->all());
    return redirect()->back();
  }

  public function destroy(Medication $medication)
  {
    $medication->delete();
    return redirect()->back();
  }
}
```

Код файлу ProfileController.php

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\ProfileUpdateRequest;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Redirect;
```

```
use Illuminate\View\View;

class ProfileController extends Controller
{
    public function edit(Request $request): View
    {
        return view('profile.edit', [
            'user' => $request->user(),
        ]);
    }
    public function update(ProfileUpdateRequest $request): RedirectResponse
    {
        $request->user()->fill($request->validated());
        if ($request->user()->isDirty('email')) {
            $request->user()->email_verified_at = null;
        }
        $request->user()->save();
        return Redirect::route('profile.edit')->with('status', 'profile-updated');
    }
    public function destroy(Request $request): RedirectResponse
    {
        $request->validateWithBag('userDeletion', [
            'password' => ['required', 'current_password'],
        ]);
        $user = $request->user();
        Auth::logout();
        $user->delete();
        $request->session()->invalidate();
        $request->session()->regenerateToken();
        return Redirect::to('/');
    }
}
```

Код файлу ReportController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Equipment;
use App\Models\Medication;
use App\Models\Sale;
use Barryvdh\DomPDF\Facade\Pdf;
use Carbon\Carbon;
use Illuminate\Http\Request;

class ReportController extends Controller
{
    public function index()
    {
```

```
    return view('reports.index');
}

public function generate(Request $request)
{
    if ($request->report_type === 'sales') {
        return $this->salesReport($request);
    }
    if ($request->report_type === 'arrivals') {
        return $this->arrivalsReport($request);
    }
    return $this->inventoryReport();
}

private function getDates(Request $request)
{
    if ($request->period_type === 'week') {
        $start = now()->subWeek();
        $end = now();
    } else {
        $month = Carbon::parse($request->month);
        $start = $month->copy()->startOfMonth();
        $end = $month->copy()->endOfMonth();
    }
    return [$start, $end];
}

private function salesReport(Request $request)
{
    [$start, $end] = $this->getDates($request);
    $sales = Sale::with([
        'user',
        'items'
    ])
      ->whereBetween('sale_date', [$start, $end])
      ->get();
    $total = $sales->sum('total_price');
    $pdf = Pdf::loadView('reports.pdf.sales-pdf', [
        'sales' => $sales,
        'start' => $start,
        'end' => $end,
        'title' => 'Звіт по продажам',
        'total' => $total,
    ]);
    if (!file_exists(storage_path('app/reports'))) {
        mkdir(storage_path('app/reports'), 0777, true);
    }
    $fileName = 'sales-report-' . now()->format('Y-m-d_H-i-s') . '.pdf';
    $filePath = storage_path('app/reports/' . $fileName);
```

```
    $pdf->save($filePath);
    return response()->download($filePath);
}

private function arrivalsReport(Request $request)
{
    [$start, $end] = $this->getDates($request);
    $medications = Medication::with([
        'manufacturer',
        'supplier'
    ])
      ->whereBetween('arrive_date', [$start, $end])
      ->get();
    $equipment = Equipment::with([
        'manufacturer',
        'supplier'
    ])
      ->whereBetween('arrive_date', [$start, $end])
      ->get();
    $pdf = Pdf::loadView('reports.pdf.arrivals-pdf', [
        'title' => 'Звіт по надходженням',
        'medications' => $medications,
        'equipment' => $equipment,
        'start' => $start,
        'end' => $end,
    ]);
    if (!file_exists(storage_path('app/reports'))) {
        mkdir(storage_path('app/reports'), 0777, true);
    }
    $fileName = 'arrivals-report-' . now()->format('Y-m-d_H-i-s') . '.pdf';
    $filePath = storage_path('app/reports/' . $fileName);
    $pdf->save($filePath);
    return response()->download($filePath);
}

private function inventoryReport()
{
    $medications = Medication::with('manufacturer')
      ->orderByRaw('expiration_date IS NULL')
      ->orderBy('expiration_date')
      ->get();
    $equipment = Equipment::with('manufacturer')
      ->orderByRaw('warranty_date IS NULL')
      ->orderBy('warranty_date')
      ->get();
    $pdf = Pdf::loadView('reports.pdf.inventory-pdf', [
        'title' => 'Інвентаризаційний звіт',
        'medications' => $medications,
        'equipment' => $equipment,
```

```
        ]);
        $pdf->setPaper('a4', 'landscape');
        if (!file_exists(storage_path('app/reports'))) {
            mkdir(storage_path('app/reports'), 0777, true);
        }
        $fileName = 'inventory-report-' . now()->format('Y-m-d_H-i-s') . '.pdf';
        $filePath = storage_path('app/reports/' . $fileName);
        $pdf->save($filePath);
        return response()->download($filePath);
    }
}
```

Код файлу SaleController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Equipment;
use App\Models\Medication;
use App\Models\Sale;
use App\Models\SaleItem;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class SaleController extends Controller
{
    public function index()
    {
        $medications = Medication::where('amount', '>', 0)->get();
        $equipment = Equipment::where('amount', '>', 0)->get();
        return view('sales.index', compact(
            'medications',
            'equipment'
        ));
    }
    public function store(Request $request)
    {
        $request->validate([
            'medications' => 'nullable|array',
            'equipment' => 'nullable|array',

            'medication_quantities' => 'nullable|array',
            'equipment_quantities' => 'nullable|array',
        ]);
        if (
            empty($request->medications) &&
            empty($request->equipment)
        ) {
            return redirect()->back()
```

```
            ->with('error', 'Оберіть хоча б один товар');
        }
        $totalPrice = 0;
        $sale = Sale::create([
            'user_id' => Auth::id(),
            'sale_date' => now(),
            'total_price' => 0,
            'status' => 'active',
        ]);
        if ($request->medications) {
            foreach ($request->medications as $medicationId) {
                $medication = Medication::findOrFail($medicationId);
                $quantity = $request->medication_quantities[$medicationId] ?? 1;
                $totalPrice += $medication->price * $quantity;
                SaleItem::create([
                    'sale_id' => $sale->id,
                    'product_type' => 'medication',
                    'product_id' => $medication->id,
                    'amount' => $quantity,
                    'price' => $medication->price,
                ]);
                $medication->decrement('amount', $quantity);
            }
        }
        if ($request->equipment) {
            foreach ($request->equipment as $equipmentId) {
                $equipment = Equipment::findOrFail($equipmentId);
                $quantity = $request->equipment_quantities[$equipmentId] ?? 1;
                $totalPrice += $equipment->price * $quantity;
                SaleItem::create([
                    'sale_id' => $sale->id,
                    'product_type' => 'equipment',
                    'product_id' => $equipment->id,
                    'amount' => $quantity,
                    'price' => $equipment->price,
                ]);
                $equipment->decrement('amount', $quantity);
            }
        }
        $sale->update([
            'total_price' => $totalPrice,
        ]);
        return redirect()->back()
            ->with('success', 'Продаж успішно оформлено');
    }

    public function history()
    {
        $sales = Sale::with('user')
```

```
            ->latest()
            ->get();
        return view('sales.history', compact('sales'));
    }

    public function toggleStatus(Sale $sale)
    {
        $sale->status = $sale->status === 'active'
            ? 'cancelled'
            : 'active';
        $sale->save();
        return redirect()->back();
    }
}
```

Код файлу UserController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Position;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $users = User::with('position')
            ->where('id', '!=', auth()->id())
            ->get();
        $positions = Position::all();
        return view('users.index', compact(
            'users',
            'positions'
        ));
    }

    public function store(Request $request)
    {
        $request->validate([
            'name' => 'required|string|max:255',
            'lastname' => 'required|string|max:255',
            'email' => 'required|email|unique:users,email',
            'password' => 'required|string|min:6',
            'position_id' => 'required|exists:positions,id',
        ]);
        User::create([
```

```
            'name' => $request->name,
            'lastname' => $request->lastname,
            'email' => $request->email,
            'password' => $request->password,
            'position_id' => $request->position_id,
        ]);
        return redirect()->back();
    }

    public function update(Request $request, User $user)
    {
        $request->validate([
            'name' => 'required|string|max:255',
            'lastname' => 'required|string|max:255',
            'email' => 'required|email|unique:users,email,' . $user->id,
            'position_id' => 'required|exists:positions,id',
        ]);
        $data = [
            'name' => $request->name,
            'lastname' => $request->lastname,
            'email' => $request->email,
            'position_id' => $request->position_id,
        ];
        if ($request->filled('password')) {
            $data['password'] = Hash::make($request->password);
        }
        $user->update($data);
        return redirect()->back();
    }

    public function destroy(User $user)
    {
        $user->delete();

        return redirect()->back();
    }
}
```

Додаток В Опис вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

1. Загальні відомості

Розроблений вебзастосунок для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові» являє собою програму для завідувача та фармацевтів аптеки, які можуть продавати товари; виконувати операції додавання, редагування та видалення з інформацією товарів та довідників; формувати звіти.

Вебзастосунок призначений для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові», що дозволить підвищити ефективність роботи персоналу, відповідального за ведення обліку лікарських засобів, контроль товарних залишків і реєстрацію продажів, а також зменшити кількість помилок під час обробки інформації, формування звітності та виконання повсякденних операцій.

Програмне забезпечення, необхідне для функціонування вебзастосунку:

– серверне середовище виконання: PHP (8.0+);
– вебсервер: Apache HTTP Server (2.4+);
– фреймворк: Laravel (10.5+);
– система керування базами даних: MySQL (8.0+);
– інструменти керування залежностями: Composer (2.0+);
– клієнтська сторона: сучасний веббраузер (Google Chrome, Mozilla Firefox, Microsoft Edge).

Мови програмування, на яких написана програма:

– PHP 8.1 – реалізація серверної логіки вебзастосунку;
– HTML5 – створення структури сторінок користувацького інтерфейсу;
– JavaScript (ES6+) – реалізація динамічної взаємодії користувача з вебзастосунком.

2. Функціональне призначення

Функціональне призначення полягає в можливості користувачів виконувати такі функції:

– авторизація користувача;

– продаж товарів;

– перегляд та обробка інформації препаратів (додавання, редагування, видалення);

– перегляд та обробка інформації обладнання (додавання, редагування, видалення);

– перегляд та обробка інформації довідників (додавання, редагування, видалення);

– перегляд та обробка інформації користувачів (додавання, редагування, видалення);

– перегляд списку продажів;

– скасування продажів;

– формування звітів (звіт по продажам, по надходження та інвентаризаційний звіт).

3. Опис логічної структури

Розроблений вебзастосунок для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові» базується на фреймворку Laravel із використанням архітектурного шаблону MVC (Model-View-Controller), що забезпечує чіткий розподіл функціональності між окремими модулями системи. Такий підхід дозволяє ефективно організувати роботу з довідниками, управлінням товарами, обліком продажів, формуванням звітів та адмініструванням користувачів.

Для реалізації користувацького інтерфейсу застосовуються шаблони Blade, бібліотека Alpine.js та CSS-фреймворк Tailwind CSS, що забезпечують зручну взаємодію користувача із системою та адаптивний сучасний дизайн. Система

підтримує розмежування прав доступу між адміністраторами та фармацевтами, що підвищує рівень безпеки та контролю над виконанням операцій.

Для зберігання та обробки даних вебзастосунок використовує систему керування базами даних MySQL із нормалізованою структурою таблиць. База даних забезпечує надійне зберігання інформації про лікарські препарати, обладнання, постачальників, виробників, користувачів та здійснені продажі. Використання ORM Eloquent спрощує взаємодію між програмою та базою даних, забезпечуючи цілісність інформації та швидкий доступ до необхідних даних. Такий підхід дозволяє автоматизувати основні бізнес-процеси аптеки, підвищити ефективність роботи персоналу та забезпечити достовірність облікової інформації.

4. Технічні засоби

Для роботи з вебзастосунком необхідна система наступної мінімальної конфігурації:

Для персональних комп’ютерів:

– персональний комп’ютер з доступом до мережі Інтернет;
– процесор із тактовою частотою 1,5 ГГц;
– оперативна пам’ять обсягом в 2 ГБ;
– наявність сучасного веббраузера (Google Chrome, Mozilla Firefox, Microsoft Edge).

Для мобільних пристроїв:

– смартфон або планшет з доступом до мережі Інтернет;
– операційна система Android 8.0 або iOS 13;
– оперативна пам’ять обсягом в 2 ГБ;
– наявність сучасного мобільного веббраузера (Google Chrome, Safari або аналогічного).

5. Виклик та завантаження

При переході на URL вебзастосунку неавторизованому користувачу відкривається сторінка авторизації. На цій сторінці користувач може обрати необхідний йому обліковий запис зі списку та ввести пароль. При коректному паролі виконується авторизація в систему та користувач перенаправляється на головну сторінку. З цієї сторінки, в залежності від права доступу, користувачу доступні можливості роботи з препаратами, довідниками, продажами, користувачами а також формування звітів.

6. Вхідні та вихідні дані

Дані вебзастосунку зберігаються у базі даних MySQL, де міститься інформація про користувачів, товари, довідники та продажі.

Вхідні дані: введення даних здійснюється за допомогою пристроїв введення (клавіатура та миша) через вебінтерфейс системи. Користувач може вводити інформацію вручну у текстові поля, поля для числових значень і дат, а також обирати необхідні значення зі списків, що випадають, та встановлювати прапорці для вибору параметрів.

Вихідні дані: результати роботи системи відображаються на екрані монітора в режимі реального часу у вигляді таблиць, форм, повідомлень про виконання операцій та звітів. Крім того, система забезпечує можливість формування та завантаження звітів у форматі PDF для подальшого зберігання або друку.

Додаток Г Інструкція користувача вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

1. Призначення програми

Вебзастосунок призначений для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові». Після переходу за адресою вебзастосунку користувач потрапляє на захищену сторінку авторизації, де може увійти до системи, використовуючи власні облікові дані. Після успішної авторизації користувач отримує доступ до функціональних модулів відповідно до наданих прав доступу.

Інтерфейс вебзастосунку забезпечує зручну навігацію між розділами товарів, продажів, звітності та керування користувачами, дозволяючи швидко переглядати, додавати, редагувати та видаляти необхідні записи. Працівники аптеки можуть здійснювати облік товарів, контролювати залишки, реєструвати операції продажу та переглядати актуальну інформацію про наявність продукції.

Для завідувача аптеки передбачені додаткові можливості формування звітів, аналізу діяльності закладу та керування обліковими записами користувачів. Реалізовані механізми перевірки введених даних та обробки помилок забезпечують коректність виконання операцій, а централізоване зберігання інформації в базі даних гарантує її цілісність, актуальність і доступність у режимі реального часу.

2. Умови використання програми

2.1 Апаратні засоби

Для персональних комп’ютерів:

– персональний комп’ютер з ОС Windows 7 (або вище) та доступом до мережі Інтернет;

– процесор із тактовою частотою 1,5 ГГц;

– оперативна пам’ять обсягом в 2 ГБ.

Для мобільних пристроїв:

– смартфон або планшет з доступом до мережі Інтернет;

– операційна система Android 8.0 або iOS 13;

– оперативна пам’ять обсягом в 2 ГБ.

2.2. Програмні засоби

– PHP (8.1+);

– Фреймворк Laravel (10.5+);

– Система керування базами даних MySQL (8.0+);

– Веб-сервер Apache, що входить до складу Open Server Panel, або аналогічне серверне середовище;

– Сучасний веббраузер: Google Chrome, Mozilla Firefox, Microsoft Edge (актуальні версії).

3. Виконання програми

При переході на URL вебзастосунку неавторизованому користувачу відкривається сторінка авторизації. На цій сторінці користувач може обрати необхідний йому обліковий запис зі списку та ввести пароль. При коректному паролі виконується авторизація в систему та користувач перенаправляється на головну сторінку. Сторінка авторизації наведена на рисунку Г.1.

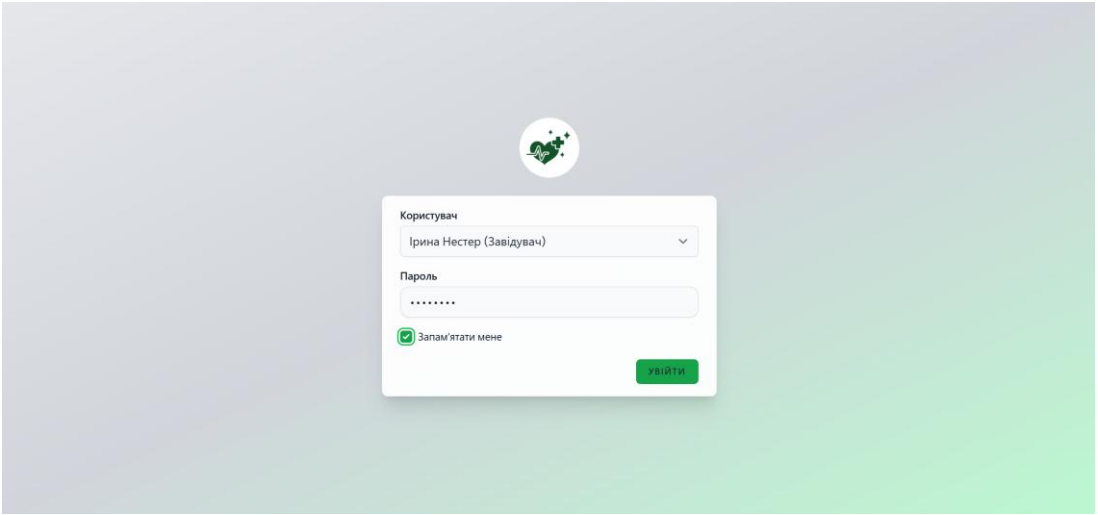


Рисунок Г.1 – Сторінка авторизації користувача вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

При успішній авторизації, користувач який має роль «Завідувач», потрапляє на головну сторінку сайту, яка наведена на рисунку Г.2.

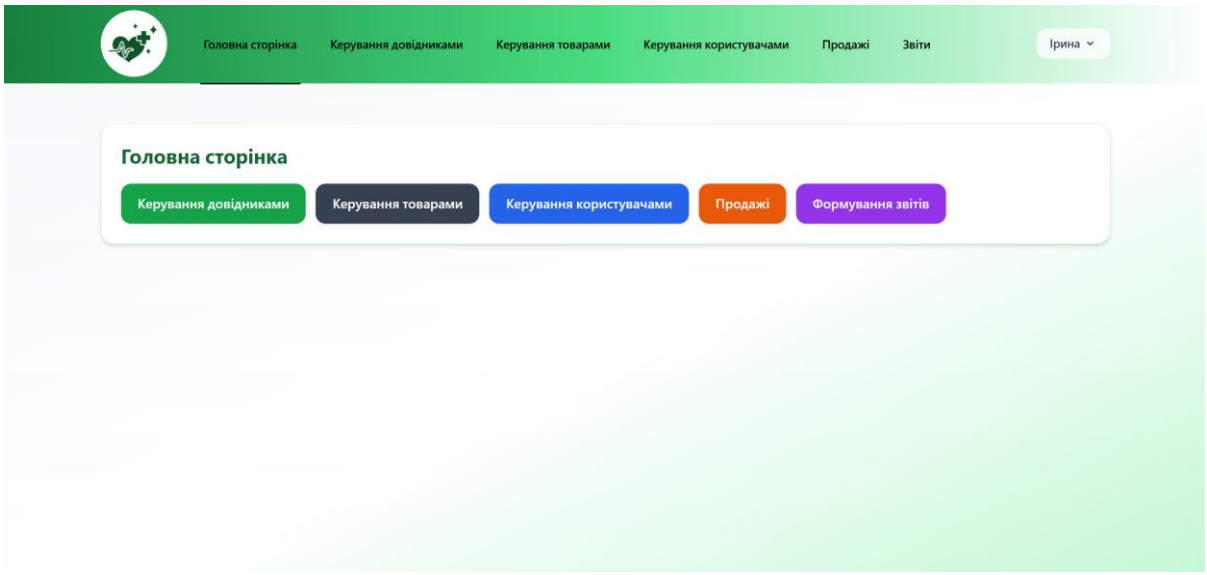


Рисунок Г.2 – Головна сторінка вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

При натисканні кнопки керування «Керування довідниками» завідувач потрапляє на сторінку, яка представлена на рисунку Г.3.

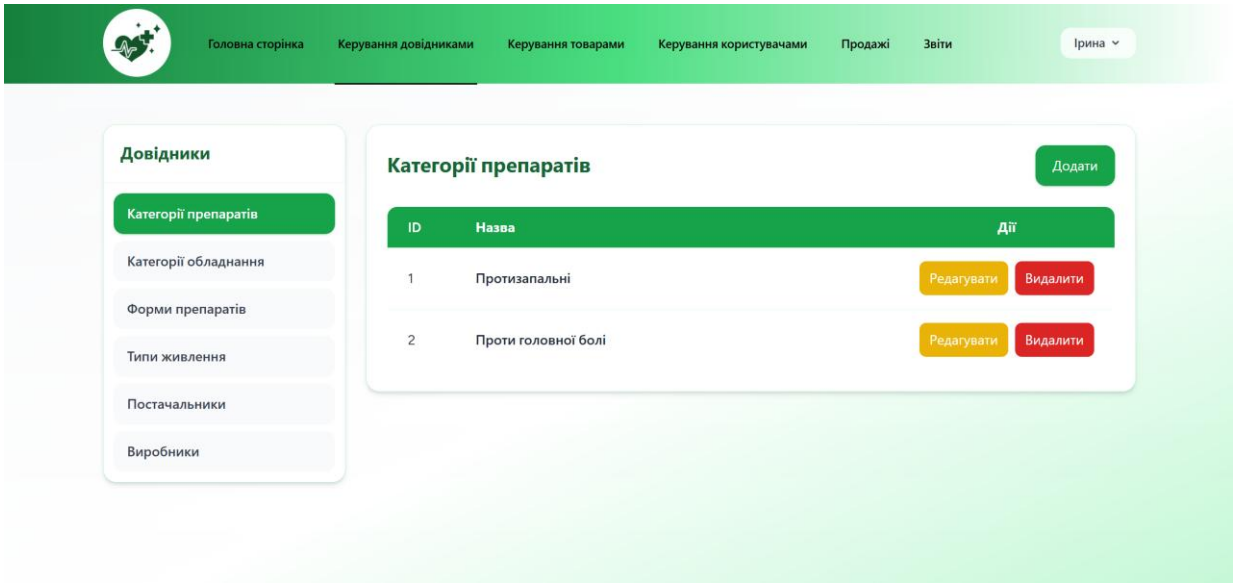


Рисунок Г.3 – Сторінка керування довідників вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

На сторінці керування довідників, завідувач може обрати з яким довідником він хоче працювати, обравши його зі списку в лівій частині сторінки. В правій частині сторінки наведено таблицю зі значеннями певного довідника. Завідувач може виконувати функції додавання, редагування та видалення значень, натискаючи кнопки керування «Додати», «Редагувати» та «Видалити» відповідно.

При натисканні кнопки керування «Керування товарами» користувач потрапляє на сторінку, яка представлена на рисунку Г.4.

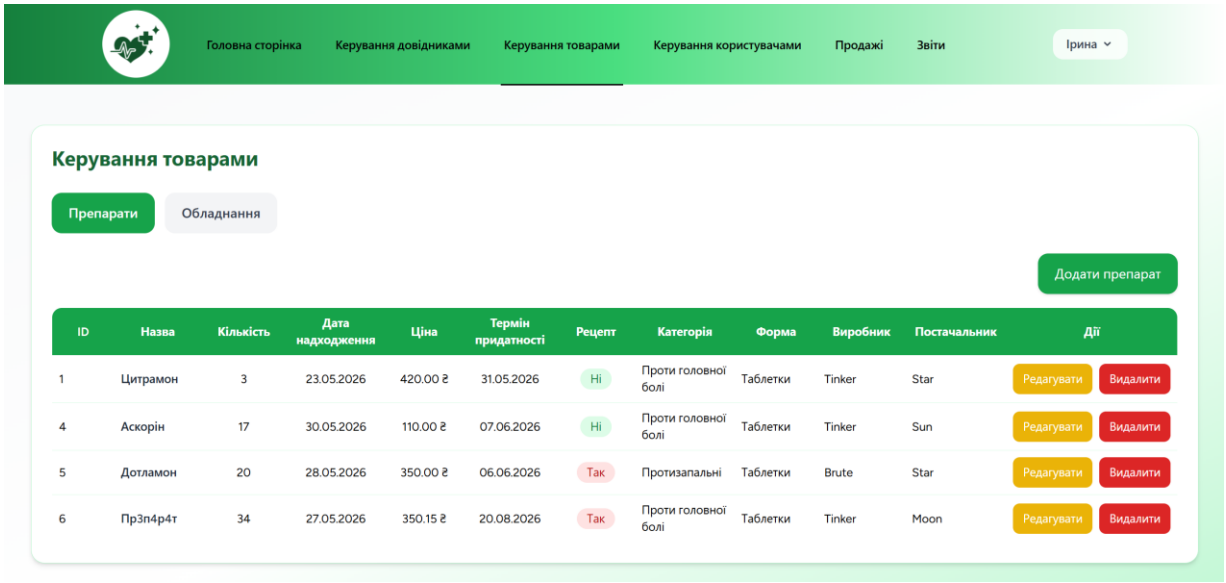


Рисунок Г.4 – Сторінка керування товарами вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

На сторінці керування товарами завідувач може обрати товари з якими він хоче працювати – препарати або обладнання. Препарати та обладнання мають окремі сторінки керування, на яких наведені таблиці з їх даними та кнопки керування для виконання операцій додавання, редагування та видалення.

Форма додавання препарату наведена на рисунку Г.5.

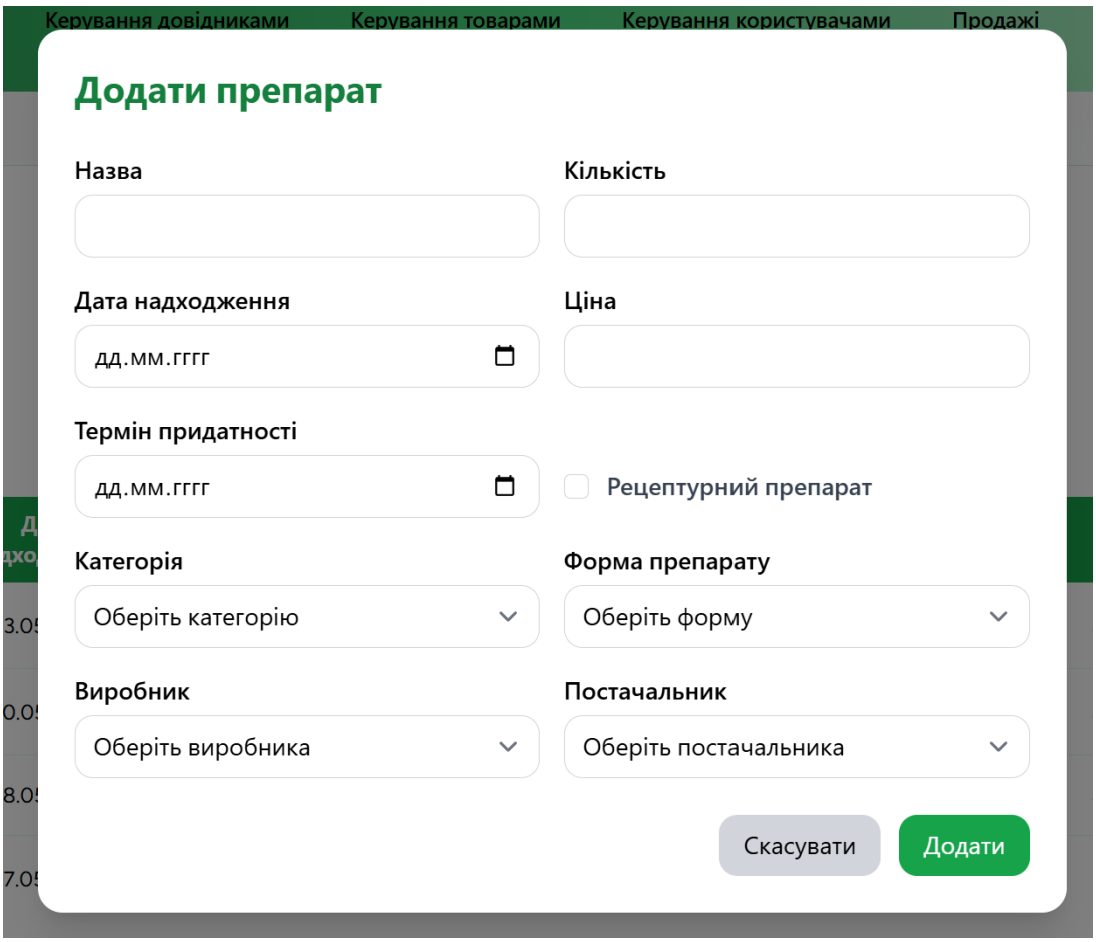


Рисунок Г.5 – Модальне вікно з формою додавання препарату.

При натисканні кнопки керування «Додати», виконується валідація даних. Якщо дані коректні, то коритувач повертається до сторінки керування на якій з’являється новий товар. Якщо дані некоректні – виводяться відповідні помилки.

Форма редагування обладнання наведена на рисунку Г.6.

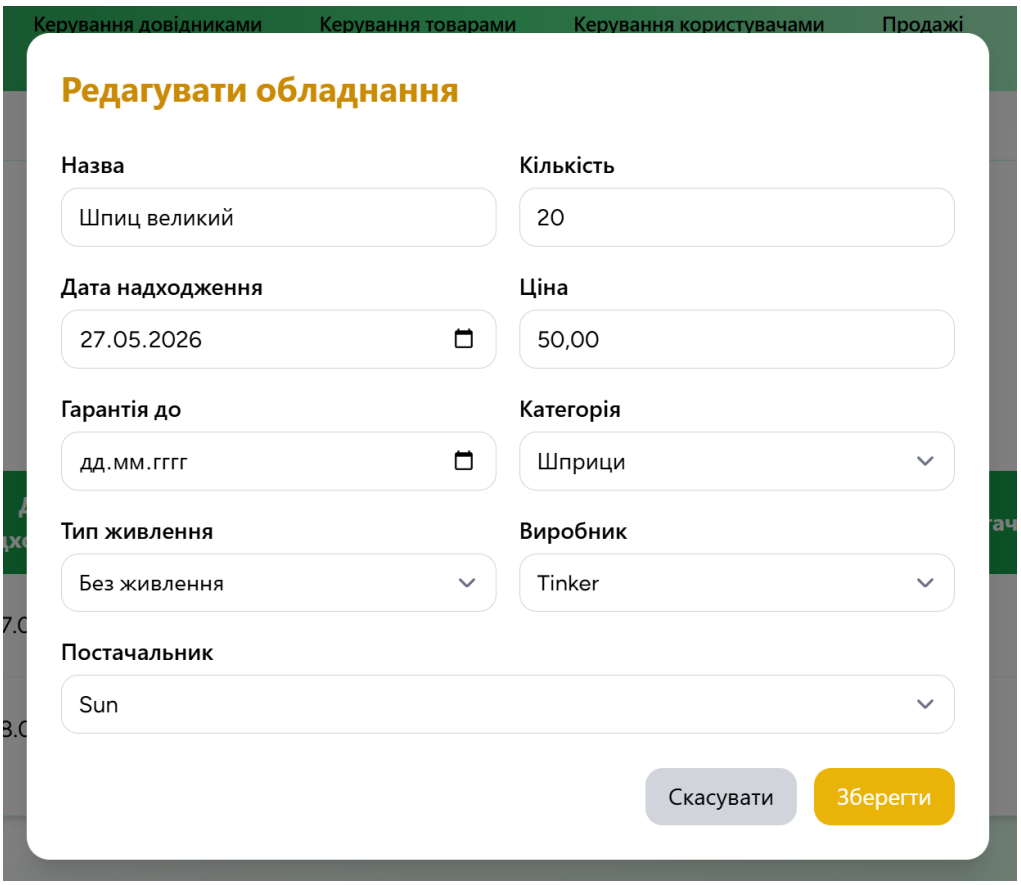


Рисунок Г.6 – Модальне вікно з формою редагування обладнання.

При натисканні кнопки керування «Зберегти», виконується валідація даних. Якщо дані коректні, то коритувач повертається до сторінки керування на якій редагований товар буде мати інші дані. Якщо дані некоректні – виводяться відповідні помилки.

При натисканні кнопки керування «Керування користувачами» завідувач потрапляє на сторінку, яка представлена на рисунку Г.7.

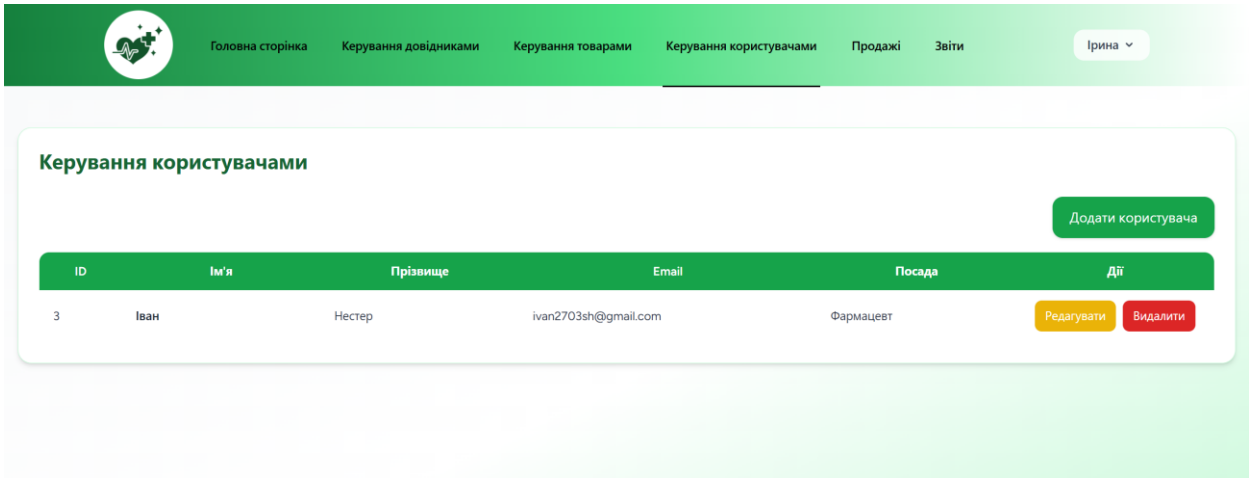


Рисунок Г.7 – Сторінка керування користувачами вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

На сторінці керування користувачами завідувач може переглядати користувачів та виконувати операції додавання, редагування та видалення користувачів вебсайту.

При натисканні кнопки керування «Продажі» завідувач потрапляє на сторінку, яка представлена на рисунку Г.8.

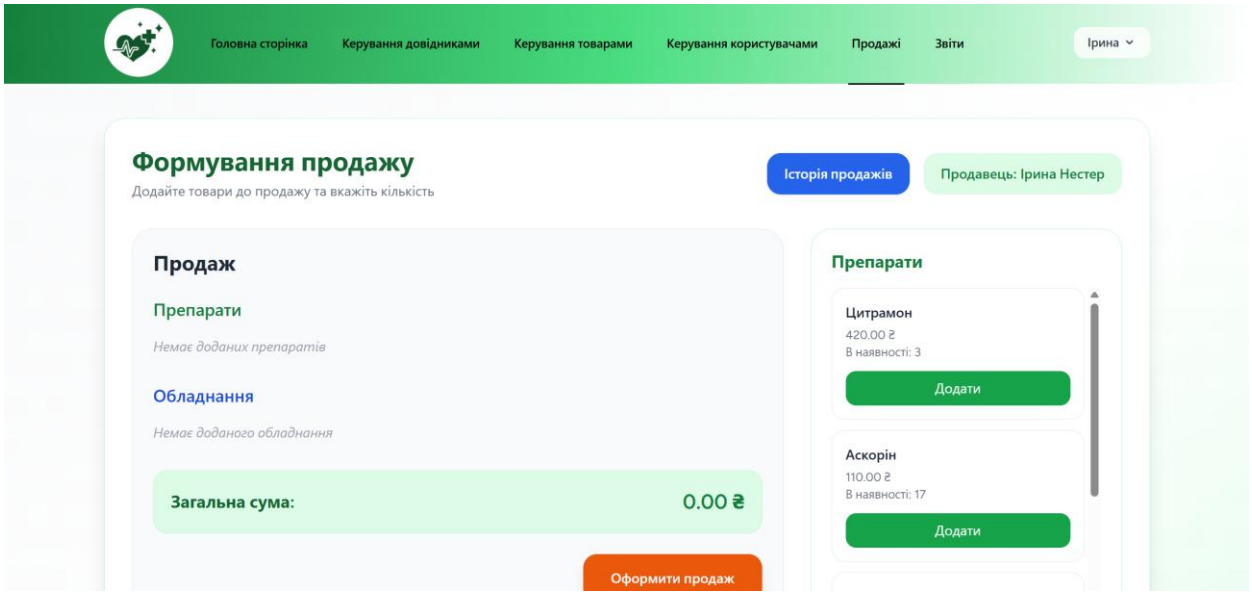


Рисунок Г.8 – Сторінка продажів вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

На сторінці продажів користувач може сформувати продаж обравши необхідні товари аптеки та натиснувши кнопку керування «Оформити продаж». У формі є можливість обрати кількість певного товару та відображається кінцева ціна продажу. Форма продажу наведена на рисунку Г.9.

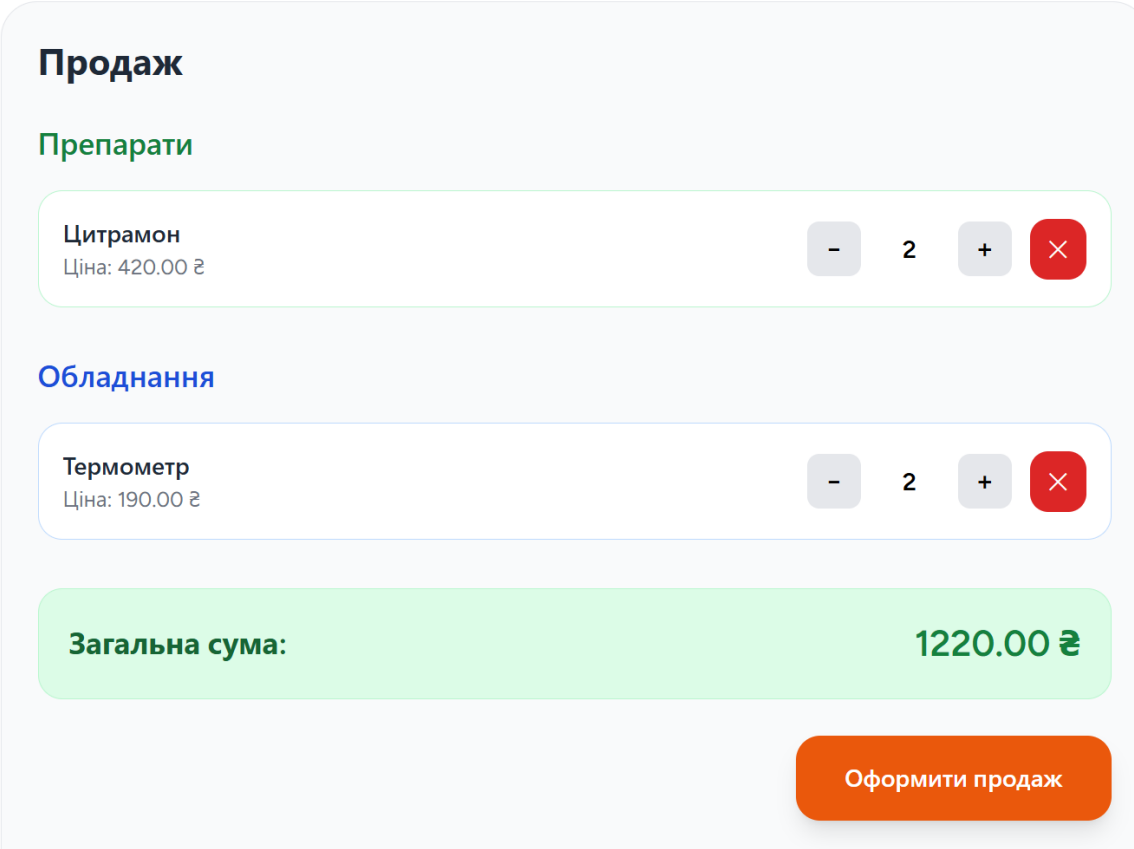


Рисунок Г.9 – Форма продажу

При натисканні кнопки керування «Історія продажів» відкривається сторінка, яка наведена на рисунку Г.10.

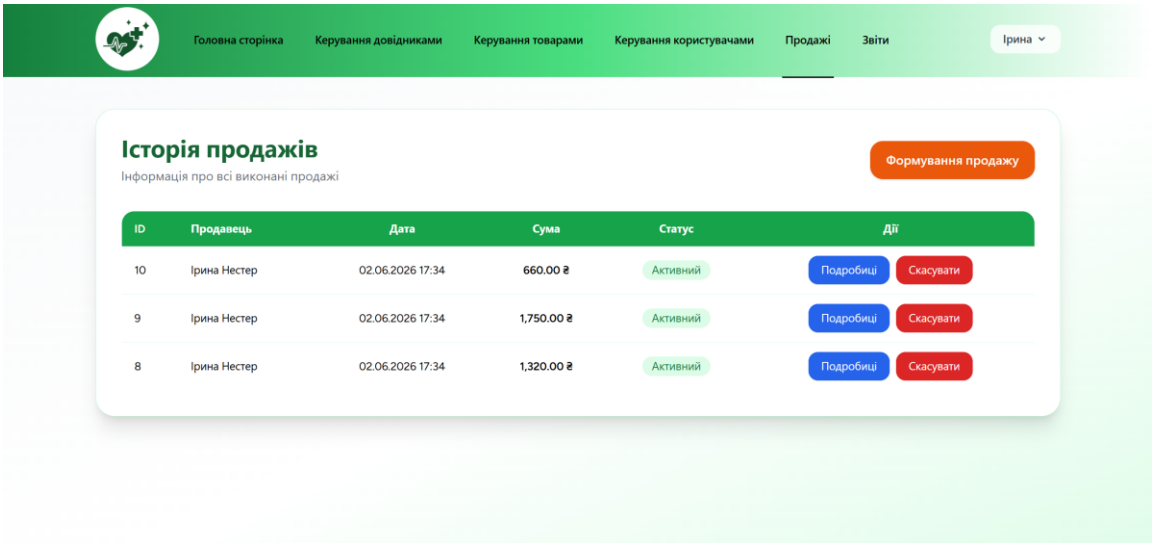


Рисунок Г.10 – Сторінка історії продажів вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

На сторінці з історією продажів користувач може змінити статус продажу натиснувши кнопку керування «Скасувати». Також, є можливість переглянути докладнішу інформацію продажу натиснувши кнопку керування «Подробиці». Докладніша інформація продажу наведена на рисунку Г.11.

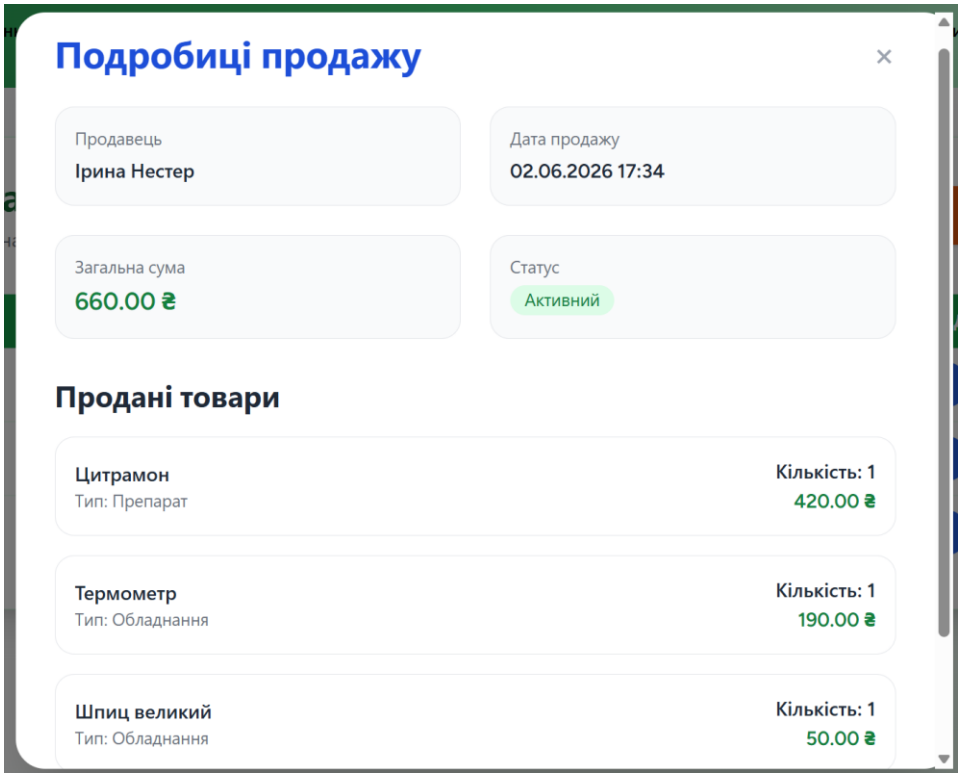


Рисунок Г.11 – Подробиці певного продажу

При натисканні кнопки керування «Формування звітів» відкривається сторінка, яка наведена на рисунку Г.12.

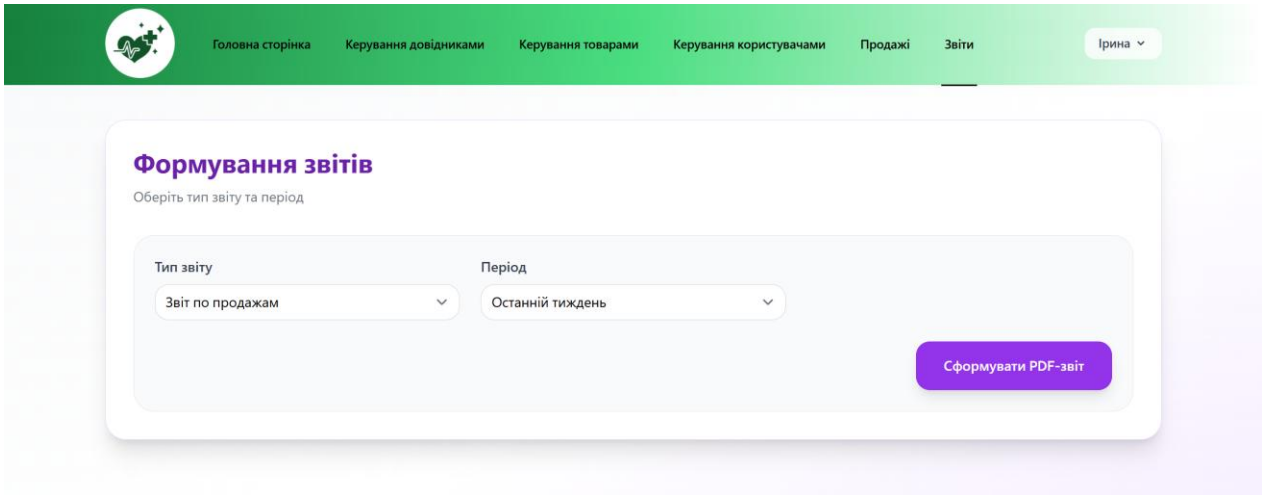


Рисунок Г.12 – Сторінка формування звітів вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

На цій сторінці можна обрати звіт який він хоче створити та період часу за яким він буде створений. Користувач може створити:

– звіт по продажам – звіт, в якому буде інформація про кількість чеків, проданих товарів та виручку, отриману за певний період часу;

– звіт надходження – звіт, в якому буде інформація про товари, які були закуплені в певний період.

– інвентаризаційний звіт – звіт, в якому буде інформація про товари, які на даний момент є в аптеці, та товари, термін придатності яких має скоро завершитись.

Можна створити звіти за певний місяць або за останній тиждень.

Приклад звіту по продажам наведено на рисунку Г.13.

Звіт по продажам

ID продажу	3
Продавець	Ірина Нестер
Дата продажу	26.05.2026 14:13
Статус	Активний
Сума	1,440.00 ₴
Товари	• Цитрамон — 2 шт. — 420.00 ₴ • Товар видалено — 3 шт. — 120.00 ₴ • Товар видалено — 2 шт. — 120.00 ₴

ID продажу	4
Продавець	Ірина Нестер
Дата продажу	26.05.2026 14:13
Статус	Активний
Сума	660.00 ₴
Товари	• Цитрамон — 1 шт. — 420.00 ₴ • Товар видалено — 1 шт. — 120.00 ₴ • Товар видалено — 1 шт. — 120.00 ₴

Рисунок Г.13 – Звіт про продажам в PDF-файлі

Завідувач може відкрити сторінку профілю натиснувши на кнопку керування «Профіль» у випадаючому списку на головній сторінці. Випадаючий список наведено на рисунку Г.14.

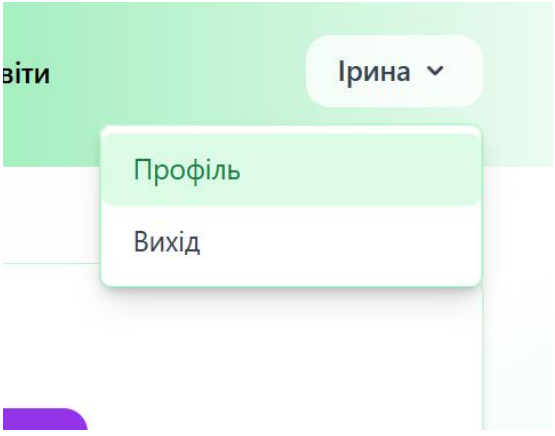


Рисунок Г.14 – Випадаючий список з кнопками керування «Профіль» та «Вихід».

На сторінці профілю користувач може змінити ім’я профілю, електронну пошту, пароль або видалити профіль взагалі. При розгортанні проекту, в базі даних автоматично створюється користувач з роллю «Завідувач».

Форми для зміни імені, пошти та паролю наведені на рисунках Г.15 та Г.16.

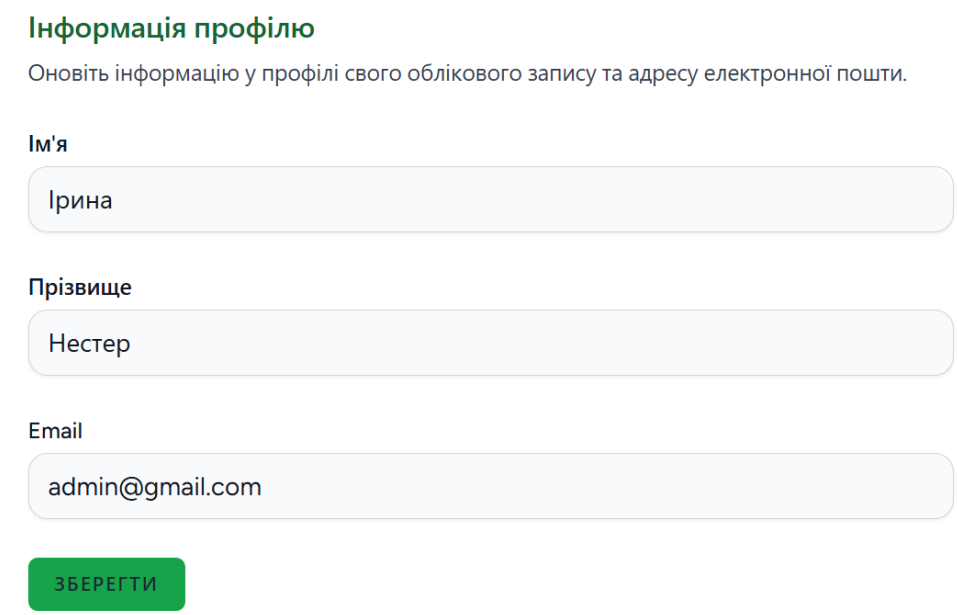


Рисунок Г.15 – Форма для зміни імені та електронної пошти

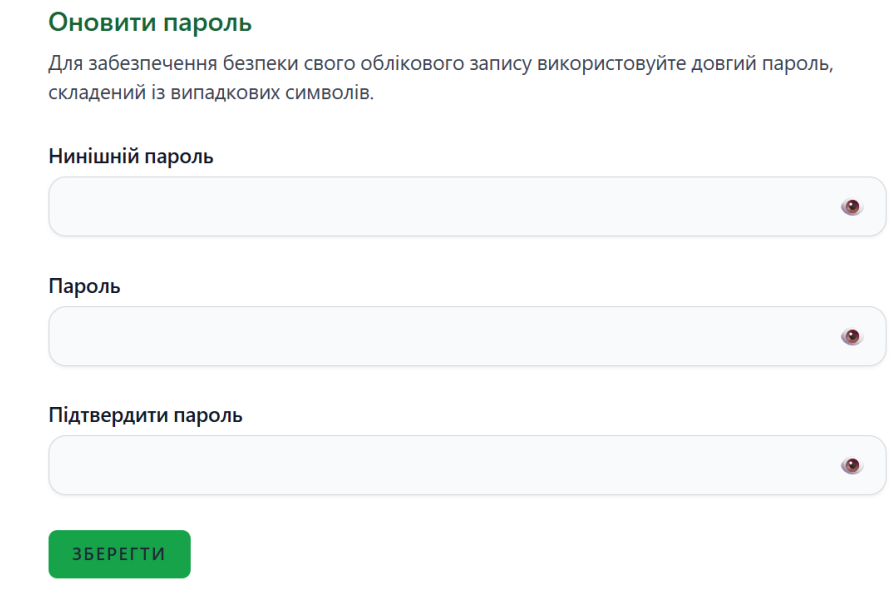


Рисунок Г.16 – Форма для зміни паролю

Кнопка видалення облікового запису наведена на рисунку Г.17.

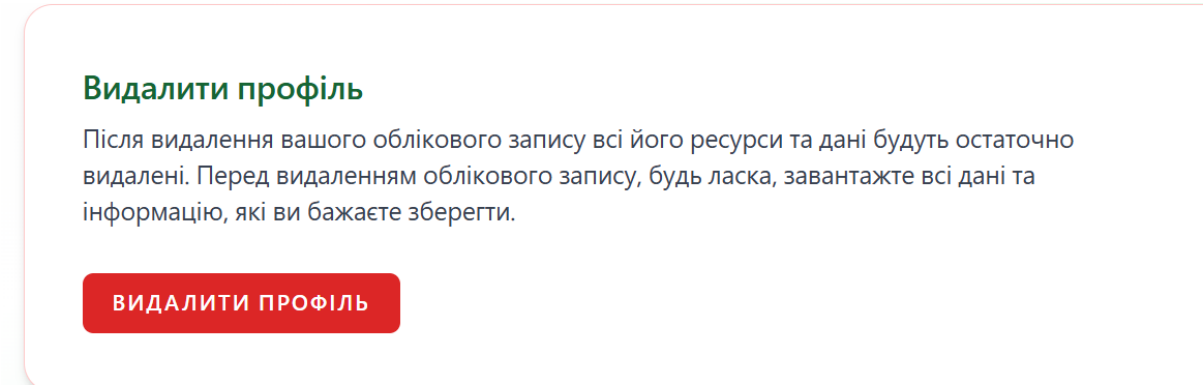


Рисунок Г.17 – Кнопка керування «Видалити профіль»

Для виходу з облікового запису, необхідно натиснути на кнопку керування «Вихід» у випадаючому списку на головні сторінці вебзастосунку.

Додаток Д Програма та методика випробування вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові»

1 Об’єкт випробувань

1.1 Назва об’єкту

Повна назва об’єкта випробувань: вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові».

Коротка назва об’єкта випробувань: вебзастосунок.

1.2 Область застосування:

Вебзастосунок призначений для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові», що дозволить підвищити ефективність роботи персоналу, відповідального за ведення обліку лікарських засобів, контроль товарних залишків і реєстрацію продажів, а також зменшити кількість помилок під час обробки інформації, формування звітності та виконання повсякденних операцій.

Функціональне призначення полягає в можливості користувачів виконувати такі функції:

– авторизація користувача;

– продаж товарів;

– перегляд та обробка інформації препаратів (додавання, редагування, видалення);

– перегляд та обробка інформації обладнання (додавання, редагування, видалення);

– перегляд та обробка інформації довідників (додавання, редагування, видалення);

– перегляд та обробка інформації користувачів (додавання, редагування, видалення);

– перегляд списку продажів;

– скасування продажів;

– формування звітів (звіт по продажам, по надходження та інвентаризаційний звіт).

2 Мета випробувань

Метою проведення випробувань є перевірка відповідності розробленого вебзастосунку для автоматизації обліку товарів та продажу в приватній аптеці «Будьте здорові» вимогам, визначеним у технічному завданні, наведеному в додатку А. Результати випробувань мають підтвердити коректність функціонування програмного забезпечення, його надійність та готовність до експлуатації.

3 Вимоги до програми

Вебзастосунок має виконувати функції, описані в технічному завданні, наведеному в додатку А.

4 Вимоги до програмної документації

Для проведення випробування має бути надана наступна документація:

– технічне завдання на розробку програмного забезпечення;
– текст програми;
– опис програми;
– інструкція користувача;
– програма і методика випробувань програмного забезпечення.

5 Засоби і порядок випробувань

Технічні засоби які були використані під час випробувань:

– Персональний комп’ютер: процесор Ryzen 7 5800HS, 16 ГБ ОЗП, SSD-накопичувач.
– Підключення до інтернету.

Програмні засоби, що використовуються під час випробувань:

– Середовище виконання: PHP (8.1+) + Apache.

– ОС Windows 11.

– СУБД: MySQL.

– Редактор коду: PhpStorm.

– Web-сервер та middleware: Apache HTTP Server, Laravel Middleware.

– Інструменти тестування API та БД: Postman, phpMyAdmin.

– Контроль версій: Git.

Порядок проведення випробувань:

– Підготовка тестового середовища (запуск вебсервера Apache та СУБД MySQL у середовищі Open Server Panel).

– Розробка тестових сценаріїв на основі функціональних вимог.

– Проведення функціонального тестування основних модулів системи.

– Перевірка коректності валідації введених даних у формах вебзастосунку та правильності обробки помилкових ситуацій.

– Аналіз отриманих результатів тестування, усунення виявлених помилок та документування результатів випробувань у звіті.

6 Методи випробувань

Відповідно до варіантів використання та функціональних вимог системи були розроблені тестові приклади для перевірки вебзастосунку за допомогою методу «Чорної скриньки». Показані приклади для основник варіантів використання:

Тестовий випадок 1

Ідентифікатор: TC- AUTH-01

Предмети тестування (Test items): Функція авторизації користувача в систему використовуючи коректні дані.

Специфікація вхідних даних (Input specifications):

– існуючий у системі користувач (Ірина Нестер);

– коректний пароль (admin123).

Специфікація результатів (Output specifications):

– система має надати доступ користувачу з правами в залежності від посади;

– перенаправлення на головну сторінку та збереження в сесії.

Тестове оточення (Environmental needs):

– форма авторизація яка дозволяє обрати обліковий запис та ввести пароль;

– база даних у якій існує користувач «Ірина Нестер» з паролем «admin123».

Спеціальні процедурні вимоги (Special procedural requirements):

– забезпечити валідність даних облікового запису (користувач/пароль) перед тестуванням.

Залежність з іншими тестами (Intercase dependencies): відсутня.

Кроки виконання (Typical test steps):

1. Відкриття форми авторизації;

2. Вибір облікового запису (Нестер Ірина) та ввід паролю (admin123);

3. Натискання кнопки керування «Вхід».

Очікуваний результат (Expected result):

– система авторизує користувача, створює сесію та перенаправляє його на головну сторінку сайту;

– користувач може виконувати всі функції відповідні до його прав доступу.

Тестовий випадок 2

Ідентифікатор: TC- AUTH-01

Предмети тестування (Test items): Функція авторизації користувача в систему використовуючи некоректні дані.

Специфікація вхідних даних (Input specifications):

– порожнє значення в полі вибору користувача;

АБО

– некоректний пароль (admin12345).

Специфікація результатів (Output specifications):

– система має повернути користувача до сторінки авторизації;

– відображення помилки про необхідність вибору облікового запису;

АБО

– відображення помилки про некоректність паролю.

Тестове оточення (Environmental needs):

– форма авторизація яка дозволяє обрати обліковий запис та ввести пароль;

– база даних у якій існує користувач «Ірина Нестер» з паролем «admin123».

Спеціальні процедурні вимоги (Special procedural requirements):

– перевірити що система не перевіряє пароль якщо користувач не обрав обліковий запис.

Залежність з іншими тестами (Intercase dependencies): відсутня.

Кроки виконання (Typical test steps):

1. Відкриття форми авторизації;

2. Ввести некоректні дані:

– залишити поле облікового запису порожнім та ввести пароль;

АБО

– обрати обліковий запис та ввести некоректний пароль (admin12345).

3. Натискання кнопки керування «Вхід».

Очікуваний результат (Expected result):

– система повертає користувача до сторінки авторизації;

– відображаються помилки про необхідність вибору облікового запису або некоректний пароль.

Тестовий випадок 3

Ідентифікатор: TC- ADD-01

Предмети тестування (Test items): Функція додавання препарату з коректними даними.

Специфікація вхідних даних (Input specifications):

– назва: Пр3па4р4т;

– кількість: 34;

– дата надходження: 27.05.2026;

– ціна: 350 грн.;

– дата придатності: 20.08.2026;

– рецептурний препарат: 1;

– категорія препарату: проти головної болі;

– форма: таблетки;

– виробник: Tinker;

– постачальник: Moon.

Специфікація результатів (Output specifications):

– система має повернути користувача до сторінки керування препаратами;

– новий препарат має зберегтись в базі даних.

Тестове оточення (Environmental needs):

– модальне вікно з формою додавання препарату;

– активне з’єднання з базою даних.

Спеціальні процедурні вимоги (Special procedural requirements):

– забезпечити валідність тестових даних препарату.

Залежність з іншими тестами (Intercase dependencies): виконання після TC-AUTH-01 та TC-AUTH-02.

Кроки виконання (Typical test steps):

1. Відкриття сторінки керування препаратами;

2. Натискання кнопки керування «Додати»;

3. Введення коректних даних препарату»;

4. Натискання кнопки керування «Зберегти».

Очікуваний результат (Expected result):

– система повертає користувача до сторінки керування препаратами;

– новий препарат зберігається в базі даних.

Тестовий випадок 4

Ідентифікатор: TC- ADD-02

Предмети тестування (Test items): Функція додавання препарату з некоректними даними.

Специфікація вхідних даних (Input specifications):

– назва: ;
– кількість: -5;
– дата надходження: 12.06.2026;
– ціна: -550,57 грн.;
– дата придатності: 30.06.2026;
– рецептурний препарат: 0;
– категорія препарату: протизапальні;
– форма: таблетки;
– виробник: Brute;
– постачальник: .

Специфікація результатів (Output specifications):

– система має повернути користувача до форми додавання препарату;
– виведення помилок для полів з некоректними даними.

Тестове оточення (Environmental needs):

– модальне вікно з формою додавання препарату;
– активне з’єднання з базою даних.

Спеціальні процедурні вимоги (Special procedural requirements):

– поля «Назва», «Кількість», «Ціна» та «Постачальник» мають мати некоректні дані.

Залежність з іншими тестами (Intercase dependencies): виконання після TC-AUTH-01 та TC-AUTH-02.

Кроки виконання (Typical test steps):

1. Відкриття сторінки керування препаратами;
2. Натискання кнопки керування «Додати»;
3. Введення некоректних даних препарату»;

4. Натискання кнопки керування «Зберегти».

Очікуваний результат (Expected result):

– система повертає користувача до форми додавання препарату;

– виведення помилок для полів «Назва», «Кількість», «Ціна» та «Постачальник»;

– новий препарат не зберігається в базі даних.

Тестовий випадок 5

Ідентифікатор: TC- EDIT-01

Предмети тестування (Test items): Функція редагування препарату з коректними даними.

Специфікація вхідних даних (Input specifications):

– назва: Препарат;

– кількість: 15;

– дата надходження: 27.05.2026;

– ціна: 350 грн.;

– дата придатності: 20.08.2026;

– рецептурний препарат: 0;

– категорія препарату: протизапальні;

– форма: таблетки;

– виробник: Brute;

– постачальник: Sun.

Специфікація результатів (Output specifications):

– система має повернути користувача до сторінки керування препаратами;

– дані препарату в базі даних мають змінитися.

Тестове оточення (Environmental needs):

– модальне вікно з формою редагування препарату;

– активне з’єднання з базою даних.

Спеціальні процедурні вимоги (Special procedural requirements):

– забезпечити валідність тестових даних препарату;

– наявність редагованого препарату в базі даних.

Залежність з іншими тестами (Intercase dependencies): виконання після TC-AUTH-01 та TC-AUTH-02.

Кроки виконання (Typical test steps):

1. Відкриття сторінки керування препаратами;

2. Натискання кнопки керування «Редагувати»;

3. Введення нових коректних даних препарату»;

4. Натискання кнопки керування «Зберегти».

Очікуваний результат (Expected result):

– система повертає користувача до сторінки керування препаратами;

– зміна даних препарату в базі даних.

Тестовий випадок 6

Ідентифікатор: TC- ADD-02

Предмети тестування (Test items): Функція редагування препарату з некоректними даними.

Специфікація вхідних даних (Input specifications):

– назва: Препарат;

– кількість: 999999999999999999999999;

– дата надходження: 05.15.2026;

– ціна: 250 грн.;

– дата придатності: 30.06.2026;

– рецептурний препарат: 1;

– категорія препарату: ;

– форма: таблетки;

– виробник: ;

– постачальник: Star.

Специфікація результатів (Output specifications):

– система має повернути користувача до форми редагування препарату;

– виведення помилок для полів з некоректними даними.

Тестове оточення (Environmental needs):

– модальне вікно з формою редагування препарату;

– активне з’єднання з базою даних.

Спеціальні процедурні вимоги (Special procedural requirements):

– поля «Кількість», «Дата надходження», «Категорія препарату» та «Виробник» мають мати некоректні дані;

– редагований препарат має існувати в базі даних.

Залежність з іншими тестами (Intercase dependencies): виконання після TC-AUTH-01 та TC-AUTH-02.

Кроки виконання (Typical test steps):

1. Відкриття сторінки керування препаратами;

2. Натискання кнопки керування «Редагувати»;

3. Введення нових некоректних даних препарату»;

4. Натискання кнопки керування «Зберегти».

Очікуваний результат (Expected result):

– система повертає користувача до форми редагування препарату;

– виведення помилок для полів Кількість», «Дата надходження», «Категорія препарату» та «Виробник»;

– зміна даних препарату в базі даних не відбувається.

Тестовий випадок 7

Ідентифікатор: TC- DEL-01

Предмети тестування (Test items): Функція видалення препарату з бази даних.

Специфікація вхідних даних (Input specifications): id препарату.

Специфікація результатів (Output specifications):

– система має повернути користувача до сторінки керування препаратами;

– обраний препарат має зникнути з бази даних.

Тестове оточення (Environmental needs):

– сторінка керування препаратами з кнопкою «Видалити» біля необхідного препарату;

– активне з’єднання з базою даних.

Спеціальні процедурні вимоги (Special procedural requirements):

– наявність препарату який видаляється в базі даних.

Залежність з іншими тестами (Intercase dependencies): виконання після TC-AUTH-01 та TC-AUTH-02.

Кроки виконання (Typical test steps):

1. Відкриття сторінки керування препаратами;

2. Натискання кнопки керування «Видалити»;

3. Підтвердження дії видалення у спливаючому вікну.

Очікуваний результат (Expected result):

– система повертає користувача до сторінки керування препаратами;

– обраний препарат видаляється з бази даних.