Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: Розробка програмної системи для обліку інвентарю на підприємстві

Виконав: студент 4 курсу
групи 5ПІ-21Б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Альпашкін Максим Ігорович
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Бабюк Н.П.
(прізвище та ініціали)

Рецензент:. к.т.н., доц. каф. ОТ Савицька Л.А.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ Романюк О.Н.
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Директор КП «ВІЦ»
С. С. Бригадир
« 21 » березня 2025 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. М. Романюк
« 21 » березня 2025 р.

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Альпашкіну Максиму Ігоровичу

1. Тема роботи – Розробка програмної системи для обліку інвентарю на підприємстві.

2. Керівник роботи: Бабюк Наталя Петрівна затверджені наказом вищого навчального закладу від 20 березня 2025 р. № 97.

3. Строк подання студентом роботи 2 червня 2025 р.

4. Вихідні дані до роботи: є інформація про інвентарні одиниці, зокрема: назва, серійний номер, інвентарний номер, відповідальна особа, місце розташування, статус об'єкта, дата останнього оновлення, примітки, фотозображення, а також унікальний QR-код для кожної одиниці; додатково — дані про користувачів, сесії інвентаризації та журнал дій.

5. Перелік ілюстративного матеріалу: діаграми, схеми архітектури системи, інтерфейси користувача, таблиці з характеристиками функціональних модулів, а також графічні зображення алгоритмів обробки даних.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Бабюк Н.П., к.н.т., доцент каф. ПЗ	24.03.25р.	01.06.25р.

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану систем для проведення інвентаризації.	25.03.25-30.03.25	вик
2	Аналіз аналогів програмної системи для проведення інвентаризації на підприємстві.	01.04.25-15.04.25	вик
3	Розробка алгоритмів роботи системи	16.04.25-20.04.25	вик
4	Розробка програмних модулів системи	21.04.25-15.05.24	вик
5	Тестування системи	16.05.25-22.05.25	вик
6	Оформлення матеріалів до захисту БКР	22.05.25-30.05.25	вик

Студент _____ **М.І.Альпашкін**

(підпис) (ініціали та прізвище)

Керівник бакалаврської кваліфікаційної роботи _____ **Н.П.Бабюк**

(підпис) (ініціали та прізвище)

АНОТАЦІЯ

УДК 004.912.032.26

Альпашкін М.І. Розробка програмної системи для обліку інвентарю на підприємстві: бакалаврська кваліфікаційна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 137 с.

На укр. мові. Бібліогр. : 24я назв.; рис. : 55; табл. : 2.

У бакалаврській кваліфікаційній роботі проведено детальний аналіз стану систем для проведення інвентаризації. Запропоновано використання сучасних інформаційних технологій для покращення процесу проведення інвентаризації та обліку інвентарем. Розроблено систему, що дозволяє здійснювати додавання, редагування, видалення інвентарних одиниць, генерувати QR-коди для ідентифікації об’єктів. Запропонована система була реалізована з використанням мови програмування Python з основною бібліотекою Flask для серверної частини. HTML, CSS, JavaScript з фреймворками Tailwind CSS та Jinja2 для клієнтської частини. PostgreSQL використано в якості бази даних на проєкті. Система розроблена в середовищі PyCharm.

Ключові слова: інвентар, одиниця, інвентаризація, система.

ABSTRACT

UDC 004.912.032.26

Alpashkin M.I. Development of a software system for inventory accounting at an enterprise: bachelor's qualification work in the specialty 121 Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2022. 137 p.

In Ukrainian. Bibliography: 24 titles; Imgs. 55; Tables: 2.

In the bachelor's qualification work, a detailed analysis of the state of inventory systems is carried out. The use of modern information technologies to improve the process of inventory and inventory accounting is proposed. A system has been developed that allows adding, editing, deleting inventory items, and generating QR codes to identify objects. The proposed system was implemented using the Python programming language with the main Flask library for the server side. HTML, CSS, JavaScript with Tailwind CSS and Jinja2 frameworks for the client side. PostgreSQL is used as a database for the project. The system was developed in the PyCharm environment.

Keywords: inventory, unit, inventory, system.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкої цифровізації бізнес процесів та зростання обсягів матеріальних ресурсів ефективне управління інвентарем є одним із ключових факторів фінансової стійкості та конкурентоспроможності підприємств [1]. Надійний облік інвентарних одиниць дозволяє своєчасно оцінювати потреби в поповненні запасів, планувати закупівлі на основі реальних даних та зводити до мінімуму ризики дефіциту або надлишку ресурсів [2].

Традиційні ручні підходи до інвентаризації, які ґрунтуються на паперових журналах або електронних таблицях, характеризуються високою трудомісткістю, значними витратами часу на збір і обробку інформації, а також підвищеним ризиком помилок через людський фактор. Відсутність єдиних централізованих інструментів веде до фрагментації даних, дублювання операцій та затримок із формуванням звітності, що ускладнює прийняття оперативних управлінських рішень [3].

Впровадження спеціалізованого програмного забезпечення для управління інвентаризацією підвищує точність і швидкість операцій, автоматизує рутинні процеси та забезпечує єдине джерело правди для всіх зацікавлених сторін. Сучасні системи відстежують рух майна від моменту його надходження до компанії до списання або продажу, надають можливість планувати оптимальні обсяги замовлень, інтегруються з фінансовими модулями та ERP рішеннями для уникнення подвійного введення даних.

Особливу роль у підвищенні ефективності відіграє впровадження багаторівневої системи доступу з розподілом прав за ролями — адміністратора, бухгалтера та інвентаризатора. Адміністратор керує налаштуваннями системи, призначає ролі та контролює журнал дій, бухгалтер відповідає за внесення та редагування даних про інвентарні одиниці, а інвентаризатор виконує сканування на місцях зберігання і фіксує статус об’єктів під час інвентаризації. Завдяки такому розподілу обов’язків забезпечується прозорість процесу та мінімізується ризик несанкціонованого доступу або помилкового видалення даних.

Для ідентифікації кожної інвентарної одиниці все частіше використовуються QR коди, що дають змогу швидко та зручно отримувати інформацію за допомогою мобільних пристроїв. QR код містить унікальний посилання на сторінку з детальними даними про об’єкт (назва, серійний номер, інвентарний номер, відповідальна особа, примітки, фотознімки тощо), що дозволяє суттєво скоротити час пошуку та підвищити точність обліку [4].

Користувацький інтерфейс програмної системи має відповідати сучасним принципам юзабіліті: мінімалістичний дизайн, інтуїтивна навігація через бічну панель із розділами «Інвентар», «Звіти», «Налаштування» (доступне лише адміністратору), таблиці, а також форми з клієнт і сервер валідацією. Чіткі повідомлення про стан операцій (наприклад, «Об’єкт додано», «Інвентарний номер уже існує») сприяють зменшенню кількості помилок і підвищують довіру користувачів до системи [5].

Тому розробка програмної системи для обліку інвентарю на підприємстві є актуальною темою. Така система має забезпечити легкі процеси ведення обліку інвентарю та інвентаризації, можливість додавання, редагування, видалення інвентарних одиниць, генерування QR-кодів.

Метою роботи є підвищення ефективності та точності процесу інвентаризації на підприємствах за рахунок розробки і впровадження спеціальної інформаційної системи, яка дозволяє автоматизувати рутинні кроки та систематизувати інформацію.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- аналіз існуючих систем для обліку інвентарю та виявлення їхніх недоліків;
- визначення основних функціональних вимог до програмної системи;
- розробка архітектури та дизайну користувацького інтерфейсу системи;
- реалізація програмної системи з використанням сучасних технологій;
- тестування та оцінка ефективності розробленого програмного забезпечення;

Об'єктом дослідження є процеси розробки системи обліку інвентарю на підприємстві.

Предметом дослідження є система для обліку інвентар., який включає в себе функціональні можливості для додавання, редагування, видалення інвентарних одиниць, а також генерації QR-кодів до інвентарної одиниці.

Новизна роботи полягає у наступному:

Удосконалено метод взаємодії із користувачами, який полягає у поєднанні сучасних веб-технологій та підходів до створення користувацьких інтерфейсів систем для обліку інвентарю на підприємстві, що, порівняно із аналогами, полегшує виконання операцій із інвентарними одиницями (таких як додавання, редагування, перегляд, фіксація статусу через QR-коди) завдяки розробці інтерактивного веб-застосунку для системи обліку інвентарю на підприємстві.

Практичне значення отриманих результатів полягає у можливості застосування розробленої системи в реальних умовах функціонування підприємств різного масштабу. Це сприятиме підвищенню точності обліку матеріальних ресурсів, зменшенню витрат часу на проведення інвентаризації та покращенню контролю за майном. Крім того, розроблена система може бути використана як основа для подальших досліджень у сфері автоматизації облікових процесів та цифровізації підприємств.

Апробація та публікація результатів бакалаврської кваліфікаційної роботи. Результати проєкту, доповідалися на конференція і опубліковані в тезах За темою дослідження опубліковано 4 наукових праць [6-8, 17].

Для написання курсового проєкту використано 24 літературних джерела.

Також система була впроваджена на підприємстві, доказом цього є акт впровадження (додаток В).

1 ФОРМУЛЮВАННЯ ЗАВДАНЬ ТА АНАЛІЗ АКТУАЛЬНОСТІ РОЗРОБКИ ПРОГРАМНОЇ СИСТЕМИ ОБЛІКУ ІНВЕНТАРЮ НА ПІДПРИЄМСТВІ

1.1 Аналіз актуальності систем обліку інвентарю

На сучасних підприємствах, незалежно від їхнього розміру та галузі діяльності, одним із важливих аспектів ефективного управління є контроль та облік матеріальних активів. Навіть компанії які організовують повністю віддалену модель роботи можуть мати свої матеріальні одиниці, які числяться на цій компанії.

Інвентар – це сукупність матеріальних активів підприємства, що підлягають обліку, контролю та періодичній перевірці [9]. Кожна інвентарна одиниця є окремим об’єктом, який має унікальний ідентифікатор, такі характеристики, як назва, місцезнаходження, серійний номер, стан та вартість. Ефективне управління інвентарними одиницями забезпечує підприємству оптимальне використання ресурсів та мінімізацію втрат.

Інвентаризація основних засобів дозволяє контролювати їхнє місцезнаходження, стан, амортизацію та ефективність використання [10]. Проте традиційні методи ведення обліку, такі як паперові журнали або електронні таблиці, мають значні недоліки, зокрема високу ймовірність помилок, низьку швидкість оновлення даних та складність у взаємодії між співробітниками.

На сьогодні підприємства все частіше впроваджують автоматизовані системи для управління обліком інвентарю, що дозволяють зменшити ризики втрат, прискорити процеси обліку та забезпечити швидкий доступ до актуальної інформації. Проте багато організацій досі використовують застарілі або неефективні підходи, що ускладнює контроль активів і підвищує ризик зловживань.

Є певний перелік незручностей, які необхідно вирішити при розробці програмної системи для обліку інвентарю. Наведемо декілька з них:

1. Використання паперових журналів та електронних таблиць спричиняє неточності та втрату важливої інформації.

2. Відсутність централізованої бази та зручних фільтрів ускладнює ідентифікацію об'єктів.

3. Відсутність журналу подій ускладнює моніторинг змін та місцезнаходження активів.

4. Підприємства змушені витрачати значний час на створення звітів вручну, що уповільнює аналіз ефективності використання ресурсів.

5. Складні та застарілі інтерфейси зменшують ефективність роботи користувачів.

Представимо можливі програмні рішення для наведених незручностей:

1. Впровадження механізмів сканування QR-кодів або штрих-кодів для швидкого доступу до інформації.

2. Реалізація ефективних фільтрів та категоризації для зручного відображення списку активів.

3. Створення журналу подій для контролю за використанням та переміщенням активів.

4. Адаптацію до потреб користувачів без необхідності глибоких технічних знань.

У висновку, можна сказати, що розробка сучасної програмної системи для обліку інвентарю є важливим завданням, що сприятиме підвищенню ефективності підприємства, зменшенню витрат на інвентаризацію та мінімізації ризиків втрати активів.

1.2 Порівняльний аналіз існуючих систем

В умовах сучасних підприємницьких процесів автоматизація інвентаризації набуває все більшої популярності, оскільки дозволяє підвищити точність обліку активів та оптимізувати управлінські процеси. Розробка власних програмних засобів для інвентаризації сприяє впровадженню гнучкої і сучасної системи, що може адаптуватися під специфічні потреби підприємства. До

найбільш відомих систем у цій галузі відносять системи компаній Dilovod, SystemGroup, Intelpol, а також розроблювану нами систему.

Dilovod – український онлайн-сервіс Система розроблена для ведення управлінського, бухгалтерського обліку та складання звітності. Надає можливість контролювати надходження, рух всередині складу, вибуття товару і оплату за нього. А також оформляйте списання, оприбуткування надлишків, виконувати інвентаризацію в будь-який час. Програмний засіб від Dilovod орієнтований на підвищення точності обліку активів і скорочення часу, необхідного для проведення інвентаризації [11]. Dilovod спрощує роботу підприємців та бухгалтерів, робить її більш ефективною, допомагаючи тим самим звільнити час і зберегти баланс між роботою та особистим життям. У ньому оптимізовано відображення залишків товару за складом на дату та час заповнення документа "Інвентаризація", реалізовано угруповання товарів у списку при заповненні інвентаризації, покращено роботу зі штрих-кодами окремих партій товару, додано генератор серійних номерів до документа "Оприбуткування". Також, з'явилася можливість автоматичного встановлення партій при підборі товару за штрихкодом, друк серійних номерів у наклейках/етикетках, додані розгорнуті повідомлення про помилки при формуванні ТТН Нової Пошти. (див. рисунок 1.1).

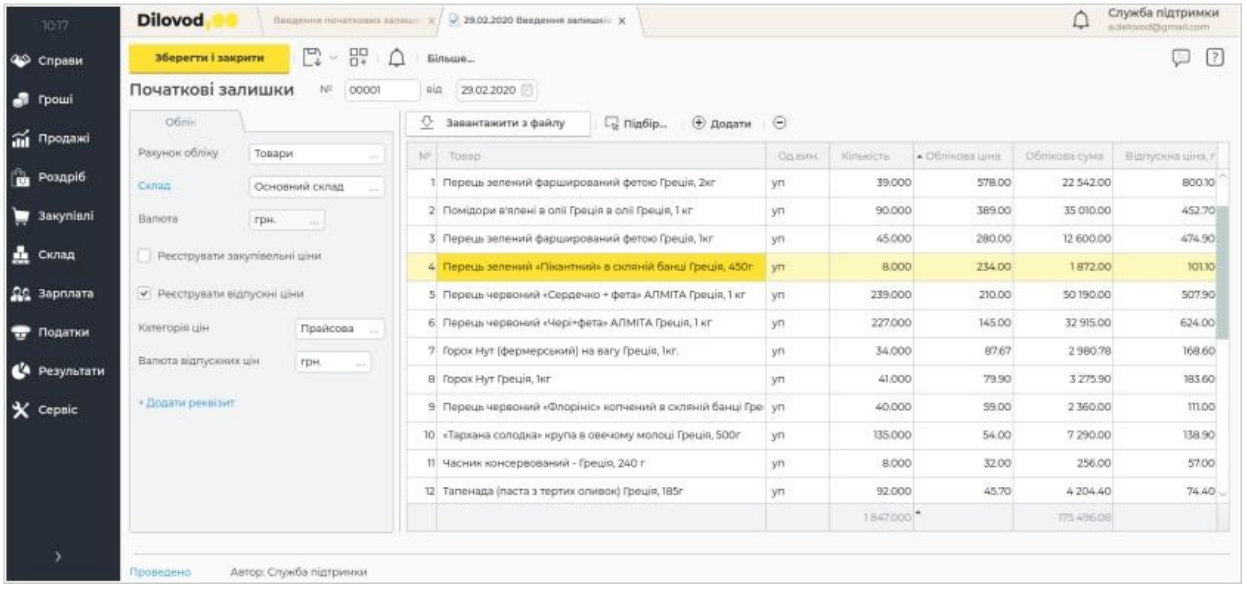


Рисунок 1.1 – Програмний засіб для інвентаризації від компанії Dilovod

Система від SystemGroup представляє собою комплексне рішення для управління товарами та цінами, яке включає модуль інвентаризації [12]. Програмний засіб дозволяє автоматизувати облік великої кількості позицій, проте з опису рішення не випливає підтримка вебдоступу або спеціальних інструментів для спрощення імплементації, що може обмежувати її гнучкість у деяких підприємствах (див. рисунок 1.2).

Рисунок 1.2 – Роздрукована через систему інвентаризації наклейка з QR-кодом від компанії SystemGroup

Рішення від Intelpol базується на використанні технології RFID для автоматизації інвентаризації, що дозволяє досягати високої точності при відстеженні руху товарів у режимі реального часу [13]. Проте, цей підхід вимагає застосування спеціалізованого обладнання (RFID-мітки та зчитувачі), що

підвищує витрати на впровадження і може стати перешкодою для малих та середніх підприємств (див. рисунок 1.3).

Рисунок 1.3 – Програмний засіб для інвентаризації від компанії Intelpol

Наша система вирізняється високою юзабіліті та легкою імплементацією: для її роботи достатньо стандартного ноутбука (або ПК) та смартфону будь-якої платформи. Система доступна у вебформаті, що забезпечує зручність впровадження в будь-яке підприємство, а розгортання в локальній мережі додає додатковий рівень безпеки. Основний функціонал включає можливість створення інвентарних одиниць із введенням обов’язкових даних (назва, серійний номер, відповідальна особа) та опціональних параметрів (ціна, фото, примітка), автоматичну генерацію інвентарного номера і QR-коду. Адміністративний модуль забезпечує ефективний менеджмент інвентаризації, а

користувацька частина дозволяє швидко отримати доступ до повної інформації про актив та відмітити його стан під час інвентаризації.

Результати порівняння аналогів зведено в табл. 1.1

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	Dilovod	SystemGroup	Intelpol	Розроблювана система
Вебдоступність	–	–	–	+
Інтеграція в локальну мережу	+	+	+	+
Легкість імплементації (без додаткових девайсів)	–	–	–	+
Висока юзабіліті	–	–	–	+
Генерація QR-коду	–	+	–	+
Можливість введення додаткових даних (ціна, фото, примітка)	–	+	–	+
Адміністративний модуль (менеджмент інвентаризації)	+	+	+	+
Використання інноваційних технологій (RFID)	–	–	+	–
Автоматизація обліку товарів	+	+	+	+
Загальна оцінка	33%	56%	44%	89%

Таким чином, проведений аналіз показує, що розробка власного програмного засобу для автоматизації інвентаризації з використанням сучасних

технологій та вебінтерфейсу є доцільною. Отриманий продукт дозволить усунути недоліки традиційних підходів, забезпечуючи високий рівень юзабіліті, легкість інтеграції без потреби додаткового обладнання та повноцінний функціонал для оперативного та точного обліку активів підприємства.

1.3 Постановка завдань до роботи

Постановка завдань є важливим етапом у розробці програмного забезпечення, оскільки вона визначає основні цілі та етапи, необхідні для успішної реалізації проєкту. Чітке формулювання завдань дає змогу організувати роботу команди, скоротити час розробки та забезпечити ефективне використання ресурсів..

Метою бакалаврської кваліфікаційної роботи є розробка веб-застосунку для обліку інвентарю на підприємстві, який включає функціонал для створення, редагування, перегляду та інвентаризації інвентарних одиниць з використанням QR-кодів. Система повинна забезпечити зручний і ефективний інтерфейс для адміністраторів, бухгалтерів та інвентаризаторів, а також безпечне збереження даних і повний аудит дій.

Для досягнення цієї мети необхідно вирішити наступні завдання:

1. Визначити основні функціональні вимоги, які повинні бути реалізовані у веб-застосунку: Це включає створення, редагування, перегляд та видалення інвентарних одиниць, автоматичну генерацію QR-кодів, фіксацію статусів під час інвентаризації та ведення журналу подій.
2. Розробити загальну архітектуру веб-застосунку: Створити структуру системи, що включатиме модулі керування інвентарем, генерації QR-кодів, логування дій користувачів, управління сесіями інвентаризації та рольовою авторизацією.
3. Створити базу даних для зберігання інформації про інвентарні одиниці: Розробити централізовану базу даних для зберігання всієї необхідної інформації: назви, серійні та інвентарні номери, фото, вартість, відповідальна особа, статус, історія перевірок тощо.

4. Реалізувати механізми безпеки для захисту даних та обмеження доступу: Забезпечити автентифікацію, авторизацію та розмежування прав доступу залежно від ролі користувача (адміністратор, бухгалтер, інвентаризатор), а також захист від несанкціонованого доступу до системи.
5. Розробити інтерфейс користувача для зручного управління інвентарем: Створити адаптивний, інтуїтивно зрозумілий інтерфейс із сучасним дизайном, фільтрами, формами з валідацією та інтерактивними елементами для швидкої і зручної взаємодії з даними.

1.4 Порівняльний аналіз методів розв'язання поставлених завдань

Розробка програмного забезпечення для автоматизації обліку інвентарю вимагає застосування ефективних методів обробки даних, які дозволять забезпечити швидкість, точність і безпеку управління активами підприємства. Традиційні підходи, такі як ведення обліку в паперових журналах або електронних таблицях, є застарілими, оскільки вони не дозволяють швидко оновлювати інформацію, ускладнюють пошук даних і створюють високий ризик людських помилок. Крім того, такі методи не надають можливості швидкого моніторингу змін і не підтримують централізоване зберігання історії переміщень інвентарних одиниць. У великих організаціях, де кількість об'єктів обліку може сягати тисяч одиниць, ручне ведення інвентаризації стає неефективним та потребує значних ресурсів.

Одним із популярних сучасних методів є використання RFID-міток [14], які дозволяють зчитувати інформацію про інвентарні одиниці безконтактним способом. Такі системи широко застосовуються в логістичних центрах і великих корпораціях, оскільки вони забезпечують високу точність обліку і дозволяють автоматично фіксувати переміщення активів. Однак їх впровадження потребує закупівлі додаткового обладнання, зокрема RFID-зчитувачів та спеціальних міток, що робить цей підхід дороговартісним і недоцільним для малих та середніх підприємств.

Більш доступним методом є використання QR-кодів для ідентифікації інвентарних одиниць (див. рисунок 1.4). На відміну від RFID, QR-коди можна генерувати автоматично та в подальшому друкувати їх, що знижує витрати на впровадження. Під час інвентаризації співробітники можуть використовувати смартфони або планшети для сканування QR-кодів, що миттєво надає доступ до актуальної інформації про об’єкт у вебінтерфейсі. Даний підхід дозволяє не лише скоротити час на проведення інвентаризації, а й інтегрувати функціонал оновлення статусу об’єкта в режимі реального часу.

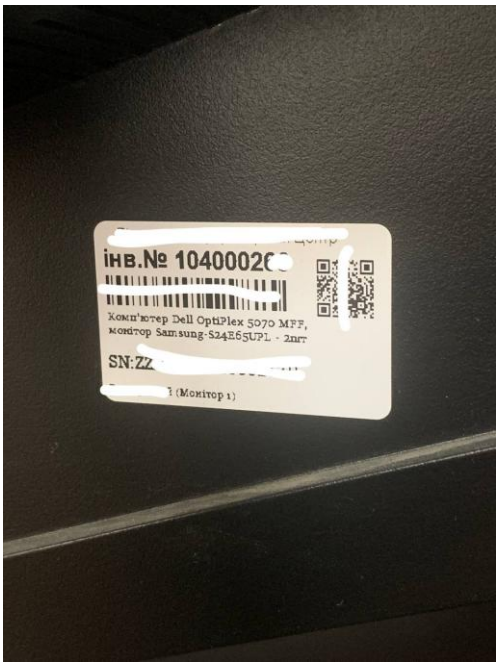


Рисунок 1.4 – Приклад термонаклейки з QR-кодом для індентифікації інвентарної одиниці

Ще одним важливим аспектом розробки є вибір архітектури програмного забезпечення. Використання вебдодатка з бекендом на основі Flask і базою даних PostgreSQL [15] дозволяє створити гнучку та масштабовану систему, яку можна адаптувати під специфічні потреби підприємства. Збереження даних у централізованій базі дозволяє уникнути дублювання записів та забезпечує їхню цілісність. Використання методу REST API допоможе розділити клієнтську та серверну частини, що дасть змогу легко масштабувати проєкт [16]. Інтеграція адміністративної панелі з можливістю додавання, редагування та видалення

інвентарних із QR-кодами забезпечують повноцінний цикл управління активами без необхідності використання стороннього програмного забезпечення.

Окрему увагу слід приділити питанню безпеки та доступу до даних. Оскільки система буде працювати у локальній мережі, вона не потребує додаткового захисту від зовнішніх атак, однак для запобігання внутрішнім порушенням необхідно передбачити розмежування прав доступу для різних ролей. Введення окремих облікових записів для адміністратора, бухгалтера та користувачів інвентаризації дозволить регулювати рівень доступу до функціоналу системи та мінімізує ризики несанкціонованих змін у базі даних.

Таким чином, оптимальним рішенням для автоматизації обліку інвентарю є використання вебдодатка, який інтегрує механізм генерації QR-кодів, централізовану базу даних та гнучку систему доступу. Такий підхід забезпечує економічність, простоту використання та високу точність обліку активів, що є критично важливим для підприємств будь-якого масштабу.

1.5 Висновок

У розділі розглядається концепція та значення автоматизації обліку та управління інвентарем на підприємстві. Система обліку інвентарних одиниць є важливим елементом для забезпечення ефективного, прозорого та точного контролю за матеріальними ресурсами організації. Впровадження сучасних інформаційних технологій дозволяє спростити процеси обліку, мінімізувати вплив людського фактора та підвищити загальну надійність збереження й обробки даних.

Основним завданням системи є забезпечення зручності для користувачів при роботі з інвентарем та зменшення часу, необхідного для виконання типових операцій. Це включає створення, редагування, перегляд та видалення записів, автоматичну генерацію інвентарних номерів і QR-кодів, а також фіксацію статусу одиниці під час інвентаризації. Враховуючи специфіку експлуатації подібних систем, доступ до функціоналу має бути розмежований відповідно до ролей — адміністратора, бухгалтера та інвентаризатора.

Процес автоматизації вимагає розробки ефективної системи управління даними, яка враховуватиме специфіку інвентарних процесів, можливість роботи в локальній мережі, дистанційного доступу для інвентаризаторів, а також оперативного оновлення інформації в централізованій базі даних. Система має бути гнучкою і підтримувати адаптацію до потреб підприємства, змін у структурі обліку та політиці доступу до даних.

Один із ключових аспектів — це безпека інформації. Дані про інвентарні одиниці, а також дії користувачів повинні бути надійно захищені від несанкціонованого доступу, що вимагає застосування сучасних механізмів автентифікації, розмежування прав доступу та створення аудиту подій. Крім того, система має бути стійкою до збоїв і забезпечувати швидке відновлення даних у разі виникнення технічних несправностей.

Аналіз існуючих систем управління інвентарем показує, що більшість систем зосереджені на реалізації окремих функцій — таких як ведення переліку об'єктів або створення звітів. Проте відсутність комплексного підходу, що включає повний цикл інвентаризації, логування змін, генерацію QR-кодів та адаптивний інтерфейс, обмежує їх ефективність у виробничих і комерційних умовах. Це створює потребу у створенні універсального інтегрованого рішення, здатного охопити всі етапи обліку інвентарю з мінімальними витратами часу та ресурсів.

Зокрема, слід враховувати можливість інтеграції з іншими корпоративними системами (наприклад, бухгалтерськими або ERP-рішеннями), що дозволить синхронізувати облік матеріальних цінностей з фінансовими даними та зменшити дублювання інформації. Такий підхід також відкриває шлях до використання аналітики та прогнозування потреб підприємства у матеріальних ресурсах.

Впровадження подібної системи вимагає всебічного аналізу наявних програмних продуктів, дослідження переваг і недоліків конкурентних аналогів, а також розробки детальної технічної архітектури із врахуванням можливостей масштабування в майбутньому. З огляду на швидкий розвиток цифрових

технологій та зміну регламентів у сфері звітності, важливо передбачити можливість розширення функціоналу без суттєвої перебудови всієї системи.

Таким чином, розробка та впровадження автоматизованої системи обліку інвентарю на підприємстві є необхідним кроком для підвищення ефективності облікових процесів, забезпечення прозорості управління активами та мінімізації ризиків втрати або недостовірності даних. Це дозволить не лише оптимізувати операційну діяльність, але й покращити загальну інформаційну безпеку та зручність роботи для всіх учасників процесу.

2 АНАЛІЗ СТРУКТУРИ АЛГОРИТМІВ ТА ФОРМУВАННЯ ПРОБЛЕМАТИКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз проблематики обліку інвентарних одиниць

Для успішної розробки програмної системи обліку інвентарю на підприємстві необхідно ретельно вивчити та проаналізувати наявні незручності в цій сфері. Важливо зрозуміти основні виклики, з якими стикаються працівники, відповідальні за інвентаризацію, а також адміністративний персонал, і виявити шляхи їх вирішення. Це включає всебічне дослідження технічних, функціональних та організаційних аспектів.

Перш за все необхідно проаналізувати функціональні труднощі, які виникають під час ведення обліку інвентарних одиниць. Наприклад, це може бути складність у пошуку актуальної інформації про об'єкт, дублювання записів, відсутність історії змін, складність з фіксацією місцезнаходження або відповідального, а також ненадійні методи перевірки наявності об’єктів на місцях.

Система обліку інвентарю є важливим інструментом для контролю матеріальних активів підприємства. У нашому випадку вона охоплює повний цикл роботи з інвентарними одиницями — від створення запису, автоматичної генерації QR-коду, фіксації та оновлення статусу під час інвентаризації, до ведення журналу дій та створення звітів. Для розробки ефективної системи важливо виявити всі вузькі місця поточного процесу обліку і врахувати їх при проєктуванні архітектури та логіки застосунку.

Однією з ключових задач є забезпечення точності й актуальності інвентарних даних. Необхідно гарантувати, що інформація про об’єкти зберігається в централізованій базі, автоматично оновлюється при зміні статусу або відповідальної особи, і є доступною для користувачів згідно з їх роллю. Важливим також є уникнення дублювання інвентарних номерів, що вимагає реалізації перевірки унікальності на етапі введення.

Ще однією задачею є фіксація статусу інвентарних одиниць під час проведення інвентаризації. У системах це зазвичай здійснюється вручну, що призводить до помилок і затримок. Використання QR-кодів спрощує цей процес: достатньо відсканувати код об'єкта мобільним пристроєм і одразу змінити його статус (присутній, відсутній, пошкоджений) із можливістю додати коментар. Така автоматизація підвищує швидкість і точність перевірок [17].

Не менш важливо забезпечити стабільність і продуктивність системи. Усі основні операції, як-от створення одиниць, зміна статусів, редагування даних, мають виконуватись швидко та без помилок, навіть при великій кількості користувачів. Також необхідно передбачити механізми резервного копіювання, логування дій і можливість відновлення після збоїв.

Інтерфейс системи повинен бути простим та інтуїтивно зрозумілим. Окремі ролі користувачів — адміністратор, бухгалтер, інвентаризатор — повинні мати доступ лише до релевантних функцій. Зручна навігація, фільтрація даних, зрозумілі повідомлення про результат дій (успіх або помилка) — усе це зменшує кількість помилок та підвищує ефективність роботи з системою.

Усі зазначені аспекти потребують ретельного вивчення й опрацювання для забезпечення розробки ефективної, надійної та масштабованої системи обліку інвентарю. Усвідомлення реальних потреб користувачів та системна оцінка незручностей дають змогу сформувати технічне завдання, яке максимально точно відповідатиме вимогам підприємства.

Підсумовуючи, всебічне дослідження потенціних задач є критично важливим для успішної реалізації проєкту. Ретельний аналіз дозволяє виявити основні виклики та знайти ефективні шляхи їх вирішення, що в підсумку сприятиме створенню конкурентоспроможної інформаційної системи обліку інвентарю, яка працюватиме стабільно, безпечно та зручно для кінцевого користувача.

2.2 Дослідження функціональних можливостей системи

Розробка програмної системи обліку інвентарю на підприємстві несе за собою створення зручного інструменту для ведення обліку матеріальних цінностей, організації інвентаризаційних перевірок та супутнього контролю змін. Основною метою є забезпечення централізованого управління інвентарними одиницями з можливістю їхньої швидкої ідентифікації, перегляду, оновлення статусів та створення звітів. Система сприяє покращенню координації між адміністраторами, бухгалтерами та інвентаризаторами, зменшенню навантаження на персонал та забезпеченню прозорості облікових процесів.

Функціональність системи охоплює кілька основних напрямків: створення й редагування інвентарних записів, генерація QR-кодів для ідентифікації об’єктів, ведення журналу змін і дій, фіксація результатів інвентаризації, а також формування звітів. Кожна інвентарна одиниця супроводжується такими атрибутами, як назва, серійний номер, інвентарний номер, відповідальна особа, місце розташування, вартість, поточний статус, фото, примітки та дата останньої перевірки. Це дозволяє здійснювати повний контроль над об’єктами та відстежувати історію змін.

Особлива увага приділяється модулям обліку та інвентаризації. При додаванні нового запису адміністратор або бухгалтер заповнює форму, після чого система автоматично генерує унікальний інвентарний номер і QR-код, що зберігається разом із записом. Сканування цього коду дозволяє інвентаризатору швидко перейти на сторінку відповідного об’єкта, переглянути його стан і зафіксувати новий статус (наприклад, "присутній", "відсутній", "пошкоджений") із коментарем.

Також реалізовано модуль керування статусами об’єктів. Усі зміни супроводжуються логуванням: фіксується дата, час, ID користувача та опис дії. Це дозволяє здійснювати аудит і формувати історію змін для кожної одиниці. Усі події доступні для перегляду адміністратора в окремому розділі — журналі дій.

Не менш важливою є функція обліку інвентаризаційних сесій. Система дозволяє створювати сесії перевірок, у рамках яких інвентаризатори фіксують статуси одиниць. Кожна сесія має власний ідентифікатор, дату створення,

відповідальних осіб та звіт про результати, що дозволяє чітко структуризувати процеси перевірки майна.

Інтерфейс системи структуровано для забезпечення швидкого доступу до основних функцій. Бокова навігація містить розділи: “Інвентар”, “Сесії”, “Звіти”, “Аудит”, “Налаштування”. На сторінці “Інвентар” реалізовано адаптивну таблицю з фільтрами, пошуком, пагінацією та кнопками дій. Кожен запис у таблиці має інвентарний номер, назву, відповідального, статус і дату останнього оновлення.

Для адміністраторів передбачено можливість архівації та редагування записів, додавання нових одиниць, зміни конфігурації системи та створення користувачів. Крім того, доступна функція експорту даних у форматі Excel для звітності або збереження локальних копій.

Система також підтримує фільтрацію записів за різними параметрами: місце зберігання, відповідальна особа, статус, вартість, дата оновлення тощо. Це дає змогу швидко знаходити потрібні записи, що особливо важливо при роботі з великими обсягами даних.

Щодо безпеки, система реалізує рольову модель доступу: адміністратори мають повний контроль над налаштуваннями, бухгалтери — доступ до облікових записів, а інвентаризатори — до фіксації статусів через QR-коди. Усі критичні операції супроводжуються автентифікацією, а доступ до клієнтської частини можливий лише в межах внутрішньої мережі підприємства, що забезпечує додатковий рівень безпеки.

Крім цього, реалізовано журнал змін — окремий механізм, який фіксує всі зміни в записах: від редагування атрибутів до оновлення статусів. Це дозволяє відновити дані у випадку помилкового редагування, провести аудит дій користувачів та забезпечити прозорість процесів.

Система також включає базовий механізм сповіщень: після успішного виконання дії (створення, редагування, фіксація статусу) користувач бачить підтвердження у вигляді повідомлення, що покращує взаємодію з інтерфейсом та зменшує кількість помилок.

Отже, функціональні можливості системи орієнтовані на спрощення процесу обліку, підвищення прозорості операцій і створення надійного інструменту для керування інвентарем. Цифрова форма обліку полегшує роботу облікових підрозділів підприємства, забезпечуючи точність, швидкодію і безпеку.

2.3 Аналіз та розробка інтерактивних можливостей програмної системи

Інтерактивні можливості програмної системи розроблені з урахуванням функціональних обов’язків основних ролей користувачів — інвентаризатора, бухгалтера та адміністратора. Система є інструментом ручного супроводу процесу обліку й інвентаризації майна, забезпечуючи зручну та контрольовану взаємодію між усіма учасниками процесу.

Інтерфейс системи побудований відповідно до ролей. Кожен користувач бачить лише ті функціональні елементи, які відповідають його повноваженням. Наприклад, інвентаризатор має доступ до перегляду інвентарних одиниць, сканування QR-кодів та фіксації статусу, тоді як бухгалтер може створювати й редагувати записи. Адміністратор керує обліковими записами користувачів, змінює налаштування системи та моніторить активність.

Інвентаризатори мабть доступ до списку об’єктів, які потрібно перевірити. За допомогою мобільного пристрою або сканера вони зчитують QR-код, що автоматично відкриває відповідну сторінку інвентарної одиниці. Через інтерактивну форму вони обирають статус об’єкта («присутній», «відсутній», «пошкоджений»), додають коментар та підтверджують дію.

Вся інформація про зміну статусу зберігається в журналі змін, до якого мають доступ адміністратор і бухгалтер. Кожна дія має часову позначку, вказівку на відповідального користувача та опис змін. Це дозволяє прозоро контролювати хід перевірки, виявляти помилки й за потреби — відновлювати хронологію змін.

Для зручності системою передбачено таблиці з фільтрацією та сортуванням за ключовими параметрами: назвою об'єкта, статусом,

місцезнаходженням, відповідальною особою, інвентарним номером або датою останнього оновлення. Це особливо важливо для великих підприємств, де кількість інвентарних позицій може сягати тисяч.

Бухгалтер, у свою чергу, має змогу переглядати повну інформацію про всі об’єкти, редагувати записи, прикріплювати зображення або змінювати відповідальних осіб. Також реалізовано функцію масового експорту обраних записів у форматі Excel, що спрощує звітність.

Адміністратор керує користувачами, розподіляє ролі, змінює паролі, деактивує записи або повністю видаляє інвентарні одиниці в разі списання. Він також має доступ до системних журналів та модулю налаштувань, де можна змінити назви категорій, структуру полів або активувати додаткові функції.

Журнал подій фіксує всі дії користувачів: створення, редагування, фіксацію статусу, зміну ролей, вхід у систему тощо. Завдяки цьому адміністратор може перевірити, хто і коли вніс зміни, що дозволяє швидко реагувати на порушення або непередбачені ситуації.

Оновлення таблиць і статусів у системі відбувається в режимі реального часу — інвентаризатор або бухгалтер одразу бачить внесені зміни без необхідності оновлення сторінки вручну. Це підвищує зручність роботи та зменшує ризик дублювання інформації.

Для запобігання помилкам система вбудовує попередження про відсутність обов’язкових даних: наприклад, якщо не вказано місцезнаходження або відповідальну особу. Такі поля позначені кольоровими індикаторами або повідомленням, що не дозволяє зберегти неповний запис.

У разі технічних несправностей адміністратор має змогу вручну змінити або деактивувати запис, а також закрити застарілі сесії інвентаризації для збереження чистоти робочого простору без втрати історичних даних.

Таким чином, функціональні можливості системи обліку інвентарю чітко узгоджені з логікою дій кожного учасника. Інтерфейс розроблений для мінімізації кроків при виконанні операцій, а поєднання ручних дій із системною автоматизацією забезпечує баланс між гнучкістю і надійністю.

2.4 Розробка загальної моделі програмного забезпечення

Загальна модель програмного забезпечення побудована за принципами інтуїтивної навігації, модульності та автоматизації обробки дій користувачів. Система є веборієнтованою, що дозволяє працювати з нею без потреби встановлення окремих програм — через будь-який сучасний браузер. Інтерфейс адаптовано під потреби трьох основних ролей — інвентаризатора, бухгалтера та адміністратора.

Розробка моделі системи спирається на багаторівневу архітектуру, що розділяє відповідальність презентаційного шару, бізнес-логіки та доступу до даних. Такий підхід забезпечує високу масштабованість за рахунок чітких меж між модулями, а також дозволяє централізовано впроваджувати механізми безпеки: автентифікацію, авторизацію та валідацію вхідних запитів. Бекенд реалізовано на Python із використанням Flask, що дає змогу швидко створювати легкі, але функціонально насичені RESTful API, які обробляють запити клієнтської частини, виконують бізнес-правила та гарантують цілісність транзакцій.

Основна структура вебзастосунку поділена на два рівні: верхнє горизонтальне меню (навігаційна панель) та основна зона контенту. У верхній панелі розміщено такі розділи:

- **Інвентар** — головна таблиця з усіма інвентарними об’єктами та їхніми параметрами (назва, інвентарний номер, статус, відповідальний).
- **Звіти** — генерація звітів про статуси одиниць, історію змін, результати інвентаризацій.
- **Аудит** — журнал дій користувачів у системі з деталізацією кожної операції.
- **Налаштування** — доступ до конфігурацій системи, створення користувачів, керування ролями (доступний лише адміністраторам).

У правій частині верхньої панелі розташовано меню користувача з відображенням поточного профілю (наприклад, "Ірина Сидоренко") та кнопкою

"Вийти". Також у деяких конфігураціях можливо реалізувати вибір підрозділу або філії підприємства.

На головній сторінці розміщено заголовок “Додати інвентар” і форму введення нової одиниці. Поля форми чітко структуровані:

- Назва об'єкта
- Інвентарний номер — вводиться вручну або генерується автоматично
- Серійний номер
- Категорія / тип майна
- Місце розташування
- Відповідальна особа
- Статус — “присутній”, “відсутній”, “пошкоджений” тощо
- Фото об’єкта (за бажанням)
- Коментарі або примітки

У нижній частині форми є кнопки: “Зберегти” — для внесення запису до бази та “Сканувати QR” — для переходу до об’єкта за його QR-кодом. Уся форма адаптована під швидке введення даних з мінімальною кількістю дій.

Після збереження нового об’єкта система може автоматично згенерувати PDF-файл з інвентарним QR-кодом, який можна роздрукувати та прикріпити до майна. Цей документ містить:

- QR-код для сканування мобільним пристроєм
- Назву об’єкта
- Інвентарний номер
- Серійний номер
- Відповідальну особу
- Дату створення запису
- Поточний статус

Діаграму діяльностей програмних засобів наведено на рисунку 2.1.

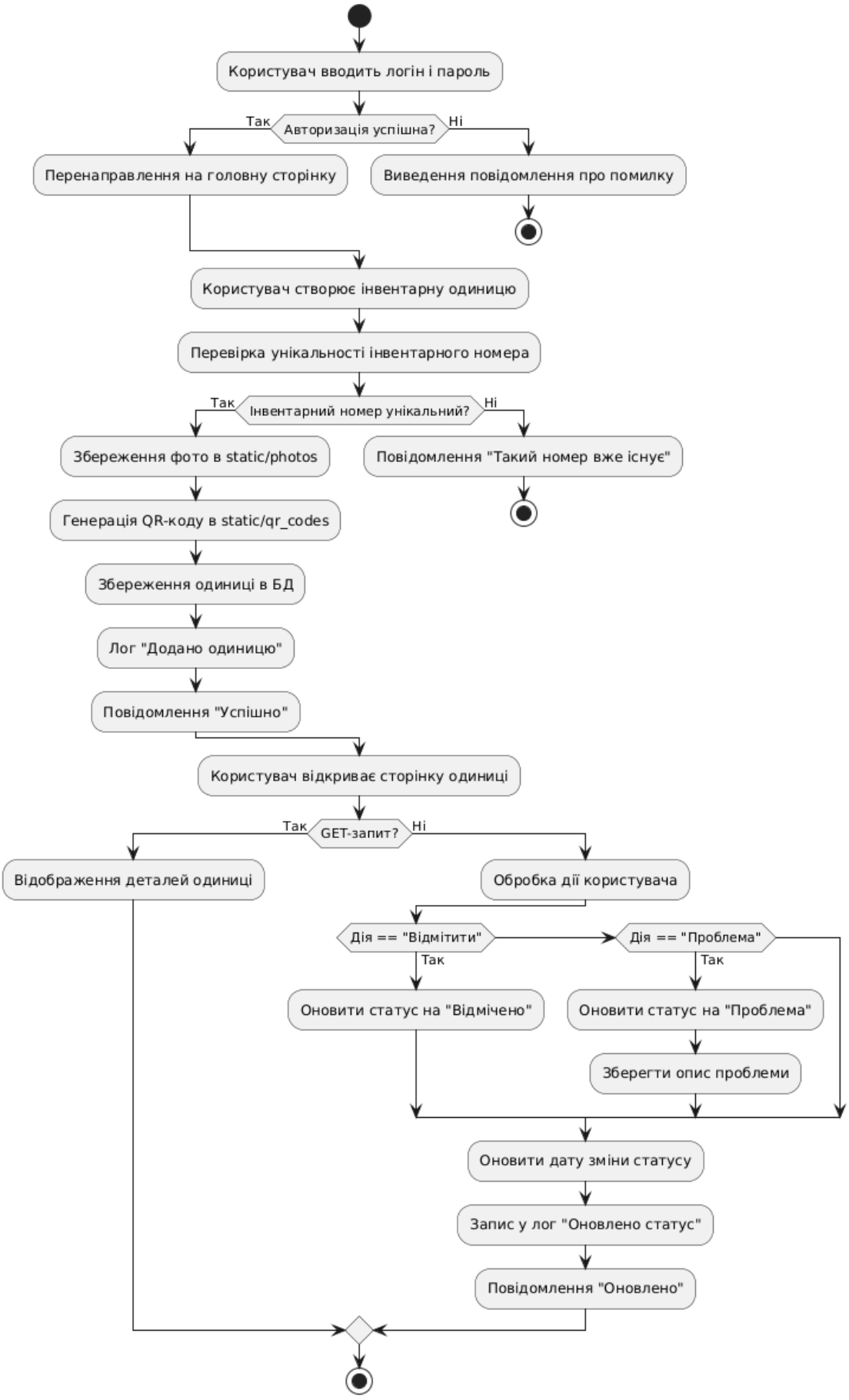


Рисунок 2.1 – Діаграма діяльності програмних засобів

Ця діаграма демонструє алгоритм роботи користувача із системою обліку інвентарю, починаючи з авторизації та завершуючи фіксацією змін до інвентарної одиниці. Вона відображає послідовність дій, зокрема: перевірку логіну та пароля, створення нової одиниці з перевіркою унікальності інвентарного номера, генерацію QR-коду, збереження даних у базі, а також подальшу роботу зі сторінкою об’єкта — перегляд, відмітку чи повідомлення про проблему.

2.5 Розробка алгоритмів роботи системи

Основою роботи системи є набір чітко виписаних алгоритмів, що автоматизують усі ключові кроки керування інвентарем. На етапі створення або реєстрації нової одиниці адміністратор чи бухгалтер через інтуїтивно зрозумілу форму вводить основні дані (назву, серійний номер, відповідального, опис тощо), після чого система автоматично формує унікальний інвентарний номер. Одночасно запускається модуль генерації QR-коду на Python (з використанням бібліотеки qrcode – легковагого та надійного рішення), який вказує на URL перегляду деталей саме цієї одиниці. Згенерований код разом із усіма атрибутами об’єкта зберігається в PostgreSQL, що гарантує цілісність транзакції та дозволяє за потреби відновити дані або виконати складні аналітичні запити.

При скануванні QR-коду мобільним пристроєм користувач із роллю інвентаризатора автоматично потрапляє на сторінку, де відображається повна інформація про актив: фотографія, історія змін, поточний статус та попередні відмітки. Інвентаризатор може миттєво позначити одиницю як «присутня», «відсутня» або «пошкоджена», додати коментар і підтвердити дію. Після цього сервер фіксує подію в журналі аудиту з точними мітками дати, часу та даними користувача, що дає змогу в будь-який момент побудувати повну історію операцій і відстежити відповідальних осіб для кожної зміни.

Алгоритми обробки запитів оптимізовано для високої продуктивності: усі запити на запис і читання виконуються через перевірені SQL-запити з індексами за ключовими полями (інвентарним номером, id сесії), а бізнес-логіка винесена в

окремі сервіси бекенду. Алгоритм роботи програмного застосунку зображено на рисунку 2.2.

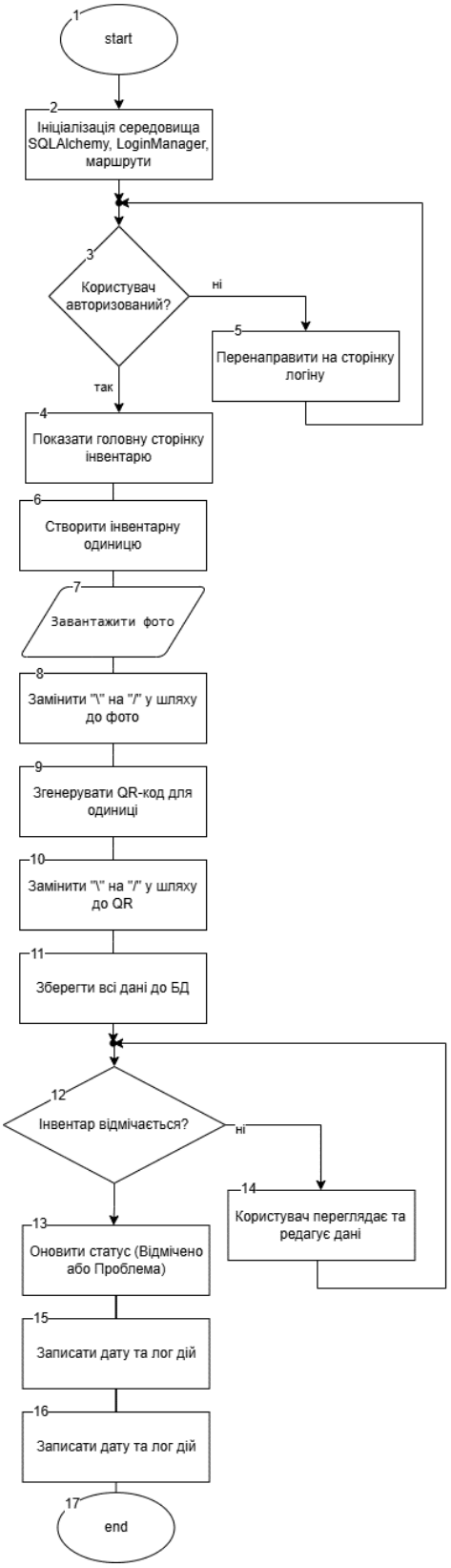


Рисунок 2.2 – Алгоритм роботи програмного застосунку

Важливою функцією роботи системи – є редагування вже доданих інвентарних одиниць. Візуалізований алгоритм даної функції зображено на рисунку 2.3

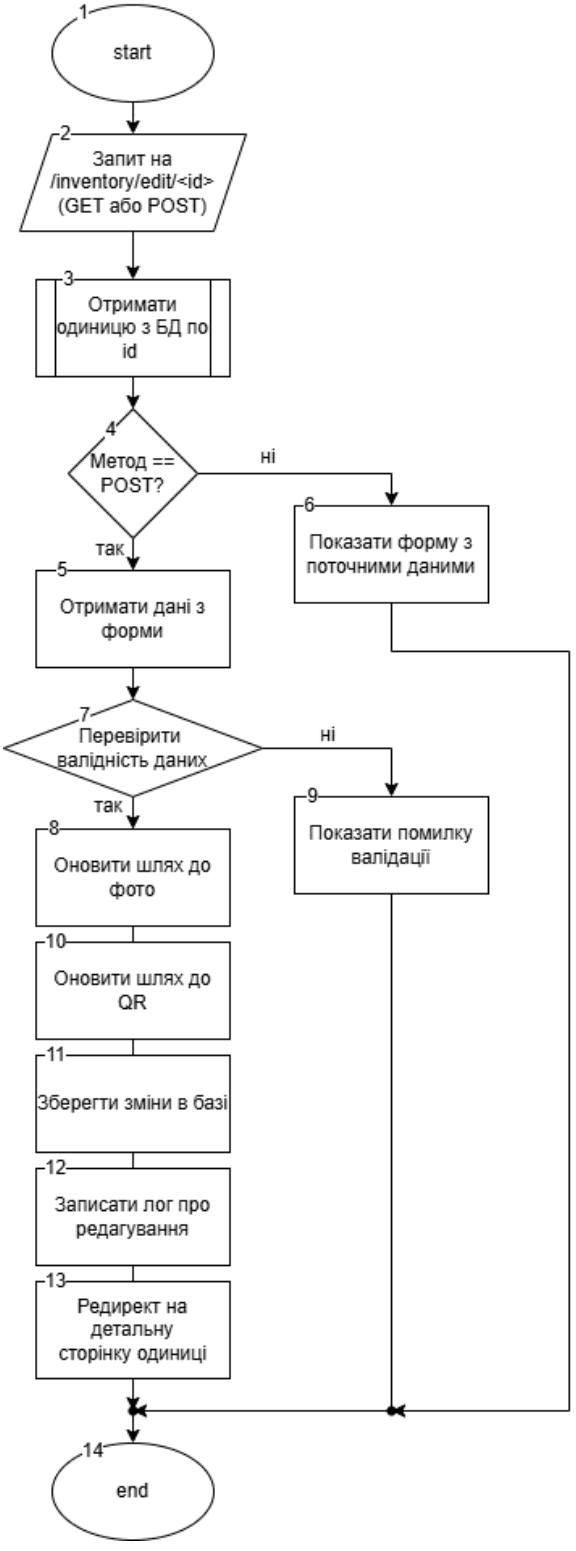


Рисунок 2.3 – Алгоритм редагування інвентарної одиниці

Крім того, система реалізує механізм кешування часто використовуваних даних, що дозволяє ще більше скоротити час відгуку.

Також архітектура системи має асинхронну обробку ресурсомістких операцій, таких як формування звітів або масове оновлення даних. Завдяки використанню черг завдань і фонової обробки користувачі не відчувають уповільнення інтерфейсу під час виконання складних обчислень. Наприклад, якщо бухгалтер ініціює генерацію великого звіту, система продовжує працювати швидко, а результат надходить у вигляді сповіщення або автоматичного збереження у відповідному розділі.

Додатковим плюсом є те, що система динамічно масштабується залежно від навантаження. Якщо кількість користувачів зростає, адміністратор може додати додаткові сервери або оптимізувати налаштування пулів з’єднань без необхідності зупинки роботи. Це робить систему гнучкою та адаптованою до потреб бізнесу, незалежно від його розміру чи сезонних коливань активності.

2.6 Висновки

У розділі 2 розглянуто аналіз структури алгоритмів та формування функцій, що лежать в основі розробки веб-застосунку для обліку інвентарю на підприємстві. У процесі дослідження було детально проаналізовано як архітектурні, так і логічні аспекти програмної системи, що дозволило обґрунтувати її технічну доцільність, відповідність поставленим вимогам та перспективність у реальному застосуванні в умовах підприємств різного масштабу.

Основною особливістю створеної системи є оптимізована багаторівнева структура, яка забезпечує чітке розмежування функціоналу між адміністративною панеллю, бухгалтерським модулем та інтерфейсом інвентаризатора. Це дозволяє ефективно організувати облік інвентарних одиниць, зменшити ризики дублювання записів та підвищити надійність під час перевірки матеріальних цінностей. Окремо було реалізовано модулі для інвентаризації, генерації QR-кодів, перегляду історії змін та створення звітів.

На основі аналізу потреб користувачів системи — працівників, відповідальних за ведення й перевірку інвентарю — було побудовано логічну модель, яка має збереження, обробку та оновлення даних у режимі реального часу. Технологічна база, що включає Python, Flask, HTML, CSS, Tailwind та JavaScript, забезпечує гнучкість розгортання, зручність масштабування, а також мінімальні витрати на технічне обслуговування.

У межах розділу було детально проаналізовано алгоритми інвентаризації, обробки змін статусів та оновлення об'єктів через сканування QR-коду. Запроваджено механізм ідентифікації об'єкта за унікальним інвентарним номером, що дозволяє швидко знаходити та оновлювати запис без потреби ручного пошуку. Це пришвидшує процес перевірок та знижує ризик помилок, пов’язаних із людським фактором.

Також розглянуто механізм логування дій користувачів — ключовий елемент для забезпечення прозорості системи. Усі зміни зберігаються у журналі дій з фіксацією часу, виконавця та типу операції. Це створює основу для внутрішнього аудиту, а також дає змогу відслідковувати історію змін при потребі відновлення або перевірки відповідальності.

Особливу увагу приділено безпеці: доступ до функцій обмежується відповідно до ролей, реалізовано перевірку автентичності сесії, шифрування паролів та захист від несанкціонованих змін. Завдяки цьому досягнуто високого рівня інформаційної безпеки без шкоди для зручності користування.

Результати проведеної роботи доводять, що обрана структура системи відповідає потребам сучасного підприємства. Система враховує специфіку облікових процедур, забезпечує простий та зрозумілий інтерфейс для кожного типу користувача, і може бути розширена або адаптована під інші галузі без суттєвого перепроєктування.

Окремо було розглянуто можливості оптимізації роботи системи через впровадження автоматичних фільтрів, пошуку, сортування та експорту даних. Це дозволяє скоротити час на виконання типових операцій та знизити навантаження на працівників підприємства.

Таким чином, проведений аналіз підтвердив доцільність запропонованого архітектурного рішення та алгоритмічного підходу для автоматизації обліку інвентарю. Система має потенціал для масштабування, інтеграції з іншими платформами та подальшого розвитку. Вона є надійною основою для цифрової трансформації процесів управління матеріальними ресурсами в умовах сучасного підприємства.

3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ПРОГРАМНОЇ СИСТЕМИ ОБЛІКУ ІНВЕНТАРЮ НА ПІДПРИЄМТСВІ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

При розробці програмного забезпечення для інвентаризації з використанням вебтехнологій важливо обрати ефективні засоби розробки як для клієнтської частини, так і для серверної логіки.

Python — основна мова програмування, яка відзначається читабельністю, зручним синтаксисом та активною спільнотою [18]. Саме Python ліг в основу усіх серверних модулів системи, забезпечуючи швидку розробку, модульність та легку підтримку коду.

Flask — легкий і гнучкий вебфреймворк для Python, що дозволяє створювати потужні вебзастосунки [19]. У цьому проєкті Flask використовується як основа для серверної частини, забезпечуючи обробку HTTP-запитів, маршрутизацію, роботу з сесіями користувача та авторизацією. Його мінімалістичний підхід надає розробнику повну свободу у побудові архітектури застосунку.

Tailwind CSS — це утилітарний CSS-фреймворк, який дозволяє створювати сучасні, адаптивні інтерфейси за допомогою класів, що описують стилі елементів без потреби писати додатковий CSS [20]. Завдяки цьому, інтерфейс користувача у застосунку стає гнучким, структурованим і легко змінюється відповідно до потреб користувача.

Jinja2 використовується як шаблонізатор для генерації HTML-сторінок на стороні сервера. Його перевага полягає у можливості створення динамічних вебсторінок із використанням змінних, циклів і умов, що забезпечує гнучке відображення даних з бази [21].

Для зберігання даних застосовано систему управління базами даних PostgreSQL — потужну об’єктно-реляційну СУБД, яка забезпечує високу продуктивність, надійність і підтримку складних запитів. PostgreSQL ідеально

підходить для структурованих даних, що мають складні зв’язки, як-от інвентарні одиниці, користувачі, сесії перевірок тощо. Робота з базою даних здійснюється через SQLAlchemy — ORM-бібліотеку для Python, яка забезпечує зручну взаємодію між Python-об’єктами та таблицями БД.

QR-коди реалізовані через бібліотеку qrcode, яка дозволяє генерувати унікальні QR-зображення для кожної інвентарної одиниці. Це спрощує процес сканування та переходу до сторінки конкретного елементу системи, що покращує інвентаризаційний процес.

JavaScript використовується для реалізації динамічної поведінки інтерфейсу, зокрема модальних вікон, асинхронної взаємодії з сервером (через fetch API), валідації форм та покращення юзабіліті системи.

Файл requirements.txt — це простий текстовий файл у Python-проєктах, який містить список усіх зовнішніх бібліотек та їхніх версій, необхідних для коректної роботи програми. Він створюється для того, щоб будь-який розробник або сервер міг швидко відтворити необхідне середовище для запуску проєкту, просто ввівши команду pip install -r requirements.txt. У цьому файлі кожен рядок відповідає певному пакету, наприклад Flask==2.0.1, де вказується не лише назва бібліотеки, а й бажана версія, що допомагає уникнути конфліктів через оновлення. Файл також може містити коментарі, посилання на бібліотеки з Git-репозиторіїв або бути розділеним на основні та додаткові залежності (наприклад, для розробки чи тестування). Використання requirements.txt спрощує розгортання проєкту, забезпечує стабільність роботи та дозволяє легко керувати залежностями, особливо в командній розробці або при переносі програми на інший сервер. Хоча для складніших проєктів іноді використовують інші інструменти, такі як Poetry або pyproject.toml, requirements.txt залишається найпопулярнішим і найзручнішим варіантом для більшості випадків.

Усі необхідні бібліотеки для функціонування системи винесені в окремий файл requirements.txt (див. рисунок 3.1).

requirements.txt × app__init__.py

```
1  blinker==1.9.0
2  click==8.1.8
3  colorama==0.4.6
4  et_xmlfile==2.0.0
5  Flask==3.1.0
6  Flask-Login==0.6.3
7  Flask-SQLAlchemy==3.1.1
8  greenlet==3.1.1
9  itsdangerous==2.2.0
10 Jinja2==3.1.6
11 MarkupSafe==3.0.2
12 openpyxl==3.1.5
13 pillow==11.2.1
14 psycopg2-binary==2.9.10
15 qrcode==8.1
16 SQLAlchemy==2.0.40
17 typing_extensions==4.13.2
18 Werkzeug==3.1.3
19
```

Рисунок 3.1 – Вміст файлу requirements.txt

Таким чином, поєднання Python, Flask, Python, Tailwind CSS, Jinja2, PostgreSQL, SQLAlchemy, JavaScript та технології QR-кодування дозволяє створити інтуїтивно зрозумілий, швидкий і надійний вебзастосунок для інвентаризації. Такий підхід забезпечує зручність у роботі для інвентаризаторів та адміністрації, дозволяючи ефективно управляти інвентарем у реальному часі.

3.2 Розробка модуля керування інвентарем

Модуль керування інвентарем є одним із ключових компонентів серверної частини системи інвентаризації. Його основне призначення — обробка запитів, пов’язаних із створенням, редагуванням, переглядом, пошуком та видаленням інвентарних одиниць. Модуль побудований з використанням Flask у поєднанні з SQLAlchemy для роботи з базою даних PostgreSQL.

У системі кожна інвентарна одиниця представлена як окремий запис у відповідній таблиці бази даних.

Через спеціальну утиліту для PostgreSQL PgAdmin 4 [22], можна переглянути усю потрібну нам таблицю з усіма її полями. Зображення частини таблиці inventory_items можна побачити на рисунку 3.2.

	id [PK] integer	name character varying (120)	serial_number character varying (120)	inventory_number character varying (120)	responsible character varying (120)	notes text
1	8	Монітор Dell E2314H	SN_0003	0003	Бачков	каб. 223
2	9	Ноутбук HP Pavilion 15	SN_0004	0004	Кадило	
3	10	ПК Dell Optiplex 3020 SFF	SN_0005	0005	Кадило	Знята плашка оперативної пам'яті

Рисунок 3.2 – Таблиця інвентарних одиниць в PgAdmin 4

Структура моделі включає такі основні поля:

– id: унікальний ідентифікатор інвентарної одиниці (первинний ключ);
– name: назва об'єкта;
– inventory_number: інвентарний номер (унікальний у межах системи);
– location: місце розташування;
– responsible_person: особа, відповідальна за об'єкт;

– status: поточний стан одиниці;

– last_checked: дата останньої інвентаризації;

– qr_code_path: шлях до згенерованого зображення QR-коду;

– description: додатковий опис або коментар;

– created_at, updated_at: часові мітки створення та оновлення.

Модель реалізується як клас Python, що наслідує базовий клас SQLAlchemy db.Model (див. рисунок 3.3).

```
class InventoryItem(db.Model):  17 usages  ntl89
    __tablename__ = 'inventory_items'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(120), nullable=False)
    serial_number = db.Column(db.String(120), nullable=False)
    inventory_number = db.Column(db.String(120), unique=True, nullable=False)
    responsible = db.Column(db.String(120))
    notes = db.Column(db.Text)
    price = db.Column(db.Float)
    photo = db.Column(db.String(255))
    qr_code = db.Column(db.String(255))
    current_status = db.Column(db.String(50), nullable=False, default="Не перевірено")
    status_updated_at = db.Column(db.DateTime, nullable=True)
    issue_description = db.Column(db.Text, nullable=True)

    def mark_checked(self):  ntl89
        from datetime import datetime
        self.current_status = "Відмічено"
        self.status_updated_at = datetime.utcnow()
        self.issue_description = None

    def report_issue(self, description):  ntl89
        from datetime import datetime
        self.current_status = "Проблема"
        self.status_updated_at = datetime.utcnow()
        self.issue_description = description
```

Рисунок 3.3 – Клас InventoryItem

Для забезпечення консистентності та автоматичної фіксації часу змін використовуються тригери або Flask-сигнали.

Модуль підтримує повний набір CRUD-операцій:

1. Створення одиниці (POST /inventory/add): на сервер надходить запит зі всіма необхідними параметрами, які валідуються. Після збереження нового об'єкта в БД автоматично викликається генерація QR-коду.

2. Читання даних (GET /inventory/<id> або GET /inventory): сервер повертає або дані окремої одиниці за ID, або список усіх об'єктів, із можливістю фільтрації за параметрами (місце, відповідальний, статус тощо).

3. Оновлення одиниці (POST/PUT /inventory/edit/<id>): система перевіряє, чи існує об'єкт, після чого застосовує зміни до відповідних полів. Зміни фіксуються у полі updated_at та логуються.

4. Видалення одиниці (DELETE /inventory/delete/<id>): сервер здійснює фізичне або логічне видалення (залежно від реалізації), попередньо перевіривши права користувача.

Кожен запит обробляється із попередньою валідацією вхідних даних. Якщо дані некоректні (наприклад, дублюється інвентарний номер або не заповнені обов’язкові поля), сервер повертає відповідне повідомлення про помилку. Всі виключення логуються в системний журнал.

3.3 Розробка модуля генерації та збереження QR-кодів

Для генерації QR-кодів у модулі використовується бібліотека qrcode, яка надає простий інтерфейс для створення кодів у форматі PNG з можливістю налаштування розміру, кольору та інших параметрів. Додатково застосовується бібліотека Pillow (PIL) для обробки зображень – це дозволяє оптимізувати QR-коди, змінювати їх роздільну здатність або додавати логотипи при необхідності. Кожен QR-код містить унікальне URL-посилання, що веде на сторінку перегляду конкретної інвентарної одиниці в системі. Наприклад, для об'єкта з ID=8 (який зберігається в базі даних) генерується код із назвою item_8.png, де закодовано шлях типу /inventory/item/8. Усі QR-коди систематизовано зберігаються у спеціальній файловій структурі (див. рисунок 3.4), що дозволяє швидко знаходити та оновлювати їх при змінах. Цей підхід не лише спрощує ідентифікацію об'єктів, але й дозволяє інтегрувати систему з мобільними

додатками – користувач може просто відсканувати код камерою та миттєво отримати доступ до потрібної інформації.

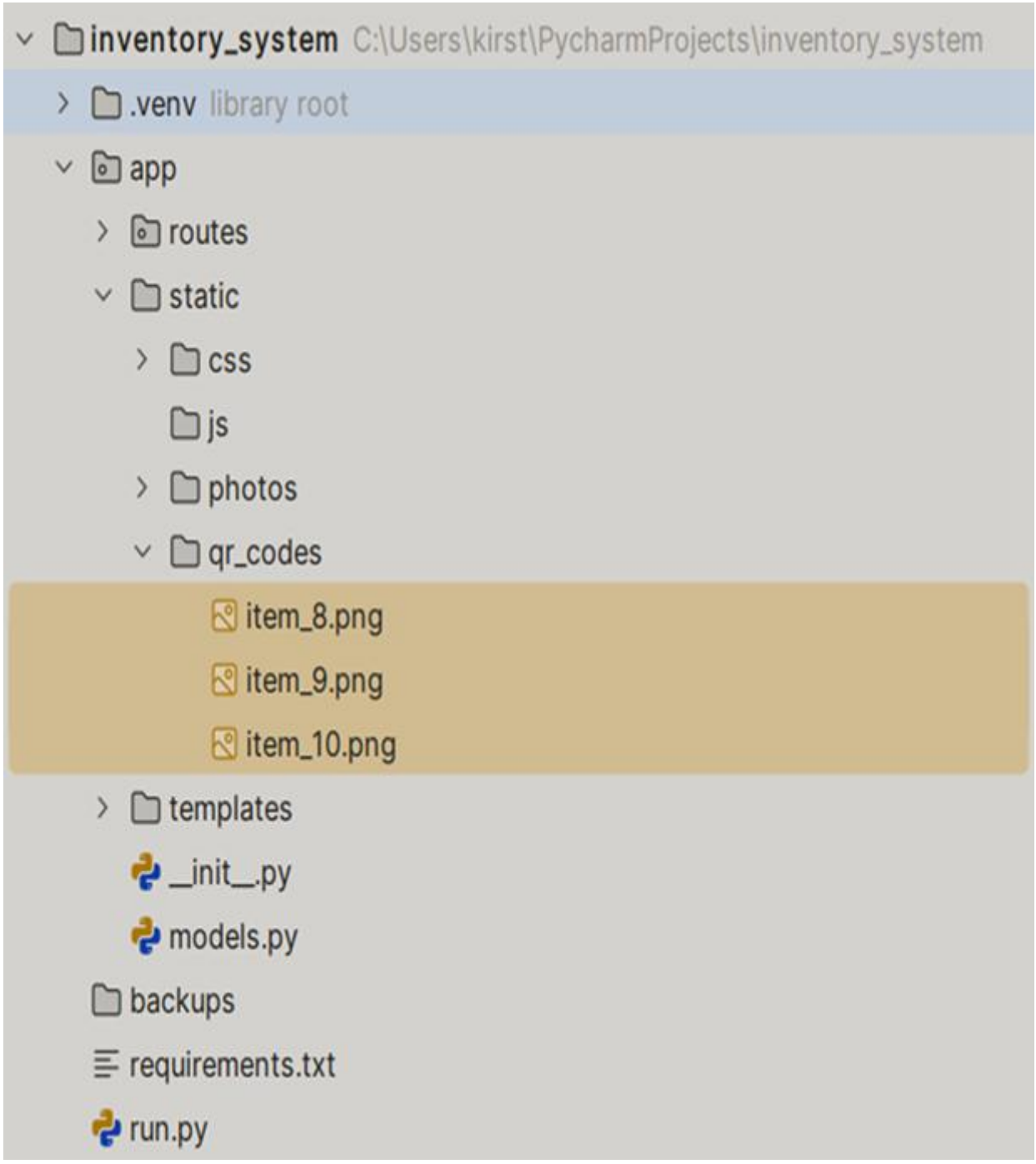


Рисунок 3.4 – Директорія з зображеннями qr-кодів.

QR-код item_8.png буде мати посилання на 8-у створену в базі одиницю інвентарю, тому буде вести по: http://127.0.0.1:5000/inventory/8 (див. рисунок 3.5).

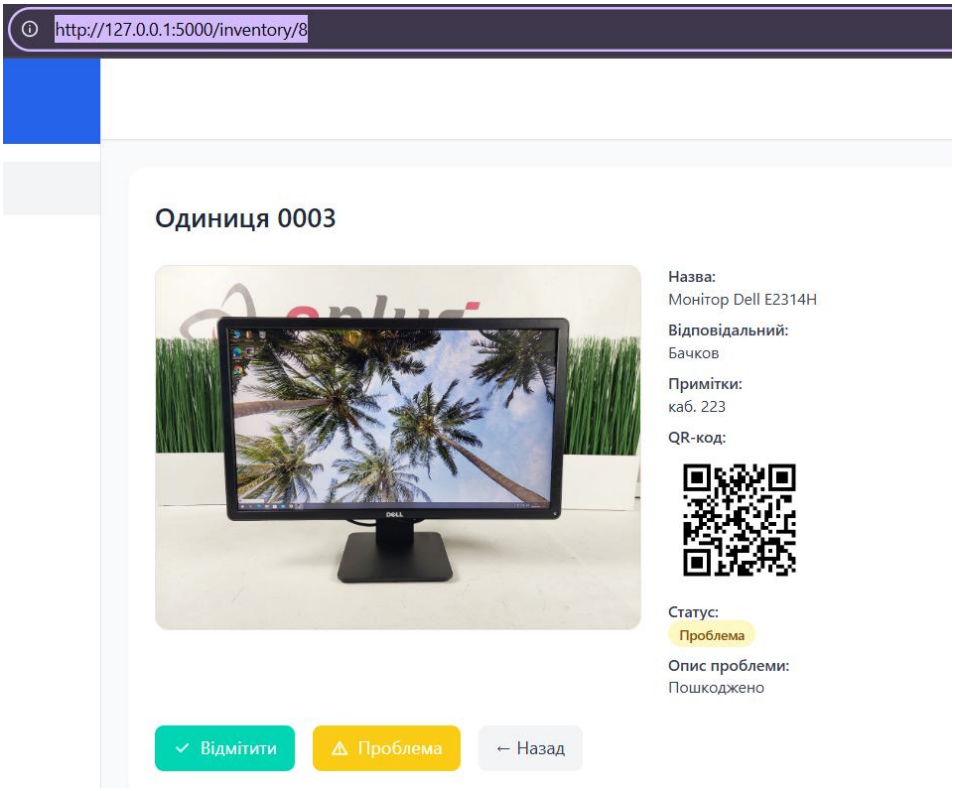


Рисунок 3.5 – Приклад посилання яке містить qr-код

Основні функції модуля:

1. generate_qr_code(data: str) -> Image. Генерує QR-код із переданими даними. Повертає об’єкт зображення.

2. save_qr_code(image: Image, filename: str) -> str. Зберігає зображення QR-коду у визначене місце в файловій системі (static/qr_codes/) Повертає шлях до збереженого файлу для запису в базу даних.

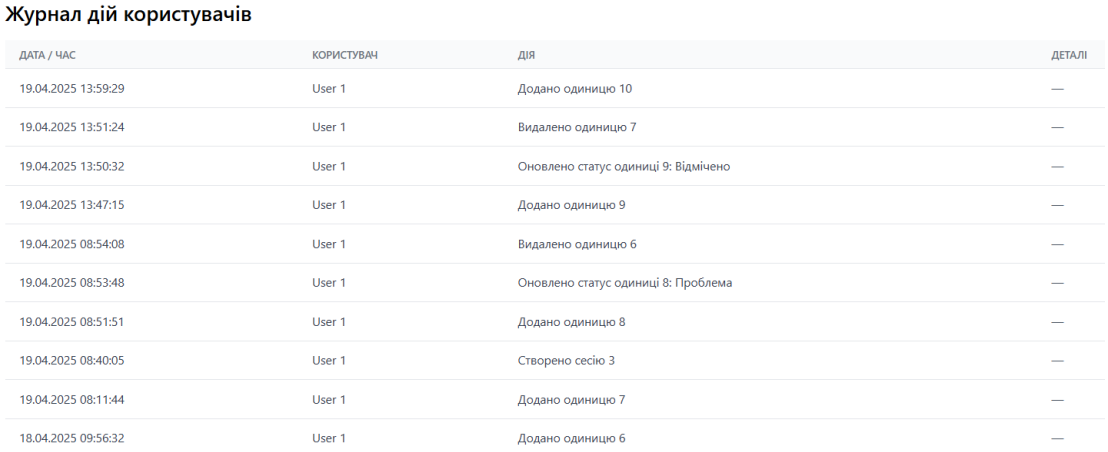
3. get_or_generate_qr(inventory_id: int) -> str. Допоміжна функція, яка перевіряє, чи вже існує QR-код для вказаної одиниці. Якщо так — повертає наявний шлях, інакше — генерує новий код, зберігає його та повертає шлях.

Модуль не взаємодіє напряму з базою даних — він лише створює і зберігає зображення, а шлях до файлу вже передається до модуля керування інвентарем, який зберігає його в БД.

Таким чином, сканування коду дозволяє швидко перейти до конкретної одиниці системи.

3.4 Розробка модуля логування подій

Модуль логування відповідає за фіксацію усіх значних подій у системі — від створення чи редагування інвентарних одиниць до запуску резервного копіювання та управління користувачами. У центрі цього модуля лежить SQLAlchemy-модель AuditLog, що містить поля id, user_id, action, details і timestamp. Зв’язок із таблицею users реалізовано через db.relationship, що дозволяє у шаблонах та у коді одразу отримувати ім’я чи email користувача, який виконав дію (див. рисунок 3.6).



Журнал дій користувачів

ДАТА / ЧАС	КОРИСТУВАЧ	ДІЯ	ДЕТАЛІ
19.04.2025 13:59:29	User 1	Додано одиницю 10	—
19.04.2025 13:51:24	User 1	Видалено одиницю 7	—
19.04.2025 13:50:32	User 1	Оновлено статус одиниці 9: Відмічено	—
19.04.2025 13:47:15	User 1	Додано одиницю 9	—
19.04.2025 08:54:08	User 1	Видалено одиницю 6	—
19.04.2025 08:53:48	User 1	Оновлено статус одиниці 8: Проблема	—
19.04.2025 08:51:51	User 1	Додано одиницю 8	—
19.04.2025 08:40:05	User 1	Створено сесію 3	—
19.04.2025 08:11:44	User 1	Додано одиницю 7	—
18.04.2025 09:56:32	User 1	Додано одиницю 6	—

Рисунок 3.6 – Журнал дій користувачів

Записи до журналу створюються щоразу, коли в коді викликається функція log_action(user_id, action, details=None). Вона інкапсулює додавання нового об’єкта AuditLog до сесії SQLAlchemy і коміт, автоматично ставить часову мітку. Виклик log_action розміщено в ключових місцях:

– при CRUD-операціях над інвентарем (додавання, редагування, видалення);

– під час оновлення статусу одиниці у сесії інвентаризації;

– при керуванні користувачами в адмін панелі;

Для виводу історії подій використовується окремий маршрут GET /reports/audit, який через Flask-Blueprint вантажить усі записи AuditLog (від останнього до найстарішого) і передає їх у Jinja2-шаблон reports_audit.html.

Завдяки такому підходу адміністратори та аудитори отримують читабельний журнал подій без зайвих JSON-структур.

3.5 Розробка вебінтерфейсу програмного застосунку

При розробці веб-інтерфейсу програмного застосунку було приділено особливу увагу простоті, консистентності та зручності. В інтерфейсі використано стриману палітру (див. рисунок 3.7):

1. Основний акцент: #00D6B4 (primary).
2. Фон сторінки: #F9FAFB (bg-gray-50).
3. Білий інтерфейс: #FFFFFF (bg-white).
4. Текст (основний): #374151, #1F2937 (text-gray-700/800).
5. Успіх: #D1FAE5 / #065F46 (bg-green-100, text-green-800).
6. Попередження: #FEF3C7 / #92400E (bg-yellow-100, text-yellow-800).
7. Помилка: #FEE2E2 / #991B1B (bg-red-100, text-red-800).
8. Інформація: #DBEAFE / #1E40AF (bg-blue-100, text-blue-800).

Рисунок 3.7 – Кольорова палітра інтерфейсу

На кожній сторінці зберігається єдина структура: бокове меню з навігацією, верхня панель з назвою і кнопками дій та основна зона контенту. Першим етапом користувацької взаємодії є сторінка входу (/auth/login). Тут користувач вводить облікові дані. Застосовано мінімалістичну форму з фокусованим стилем інпутів і плавним анімаційним появам (див. рисунок 3.8).

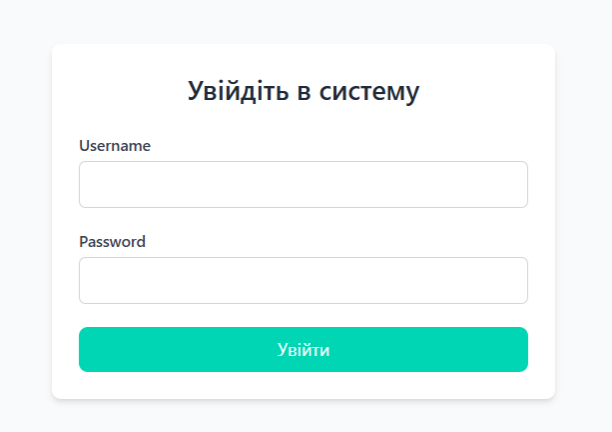


Рисунок 3.8 – Вікно входу користувача

Після входу користувача зустрічає вертикальне меню з головними розділами: Інвентар, Сесії, Звіти, Аудит, Вийти. Меню фіксоване, що дозволяє швидко переключатися між модулями (див. рисунок 3.9).

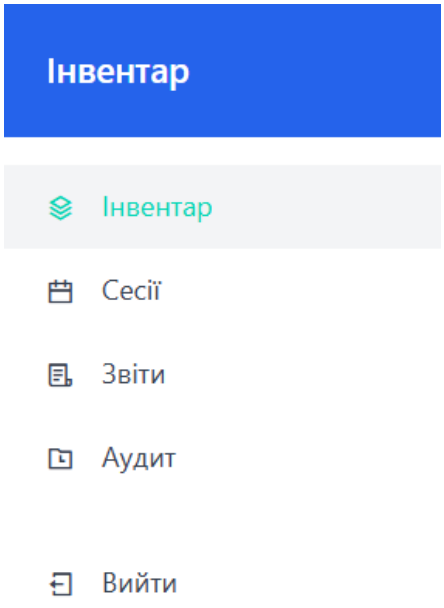


Рисунок 3.9 – Бокове меню навігації

На сторінці “Інвентар” розміщено рядок пошуку, селектори фільтрацій та таблицю з пагінацією. Кожен рядок містить Інв. №, назву, статус, дату останнього оновлення й кнопки дій. Таблиця адаптивна, із горизонтальною прокруткою на малих екранах (див. рисунок 3.10).

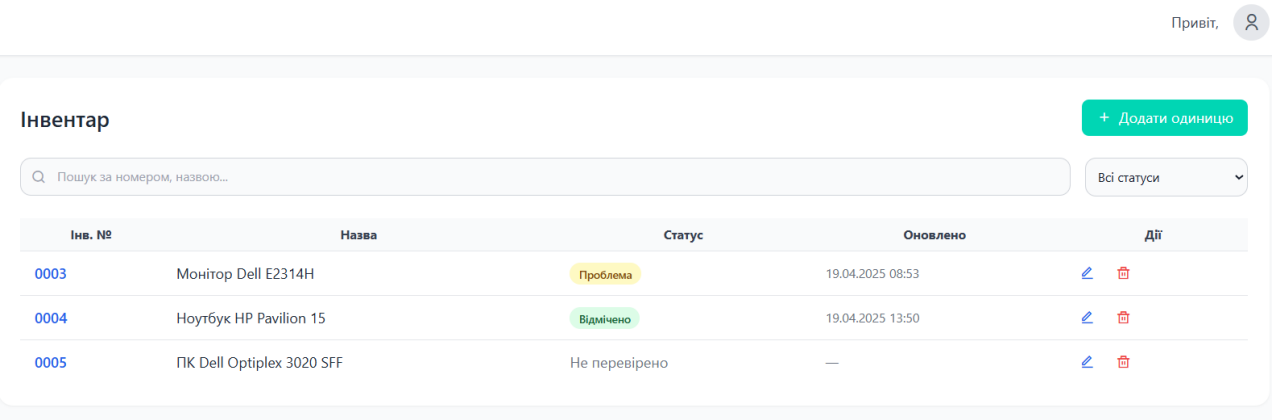


Рисунок 3.10 – Сторінка “Інвентар”

При кліку на Інв. № відкривається сторінка одиниці: ліворуч фото, праворуч — таблиця атрибутів (назва, сер. ном., відповідальний, вартість, примітки, QR-код) (див. рисунок 3.11). Нижче дві кнопки — “Відмітити” і “Проблема” — що викликає модальне вікна для вибору статусу й введення коментаря (див. рисунок 3.12). На цю сторінку також можна потрапити через сканування відповідного QR-коду.

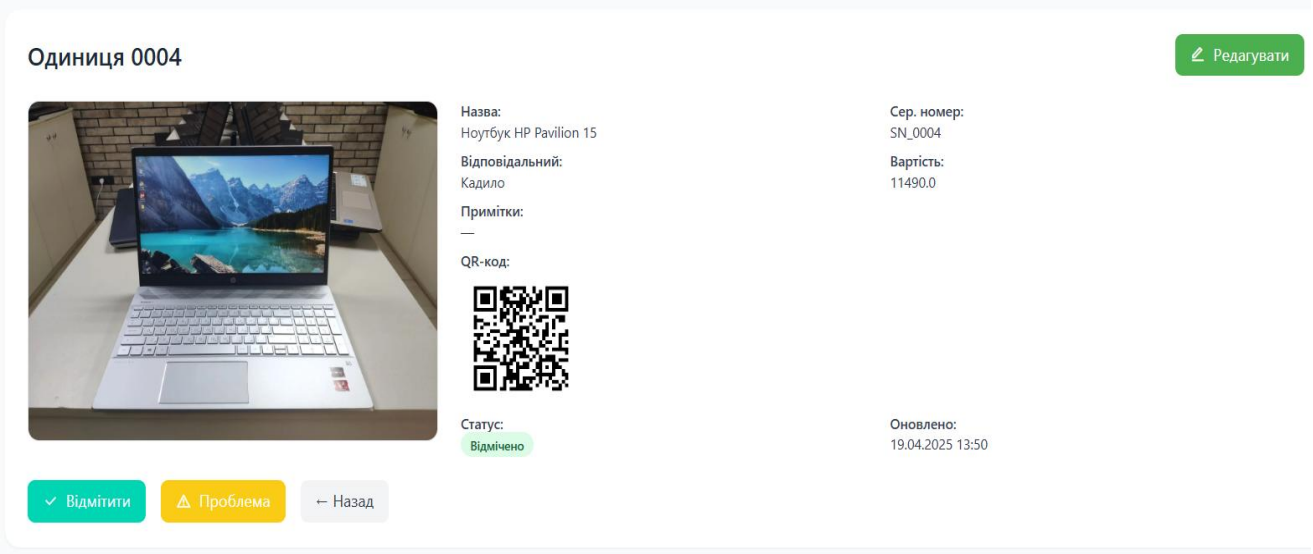


Рисунок 3.11 – Сторінка з детальною інформацією про одиницю

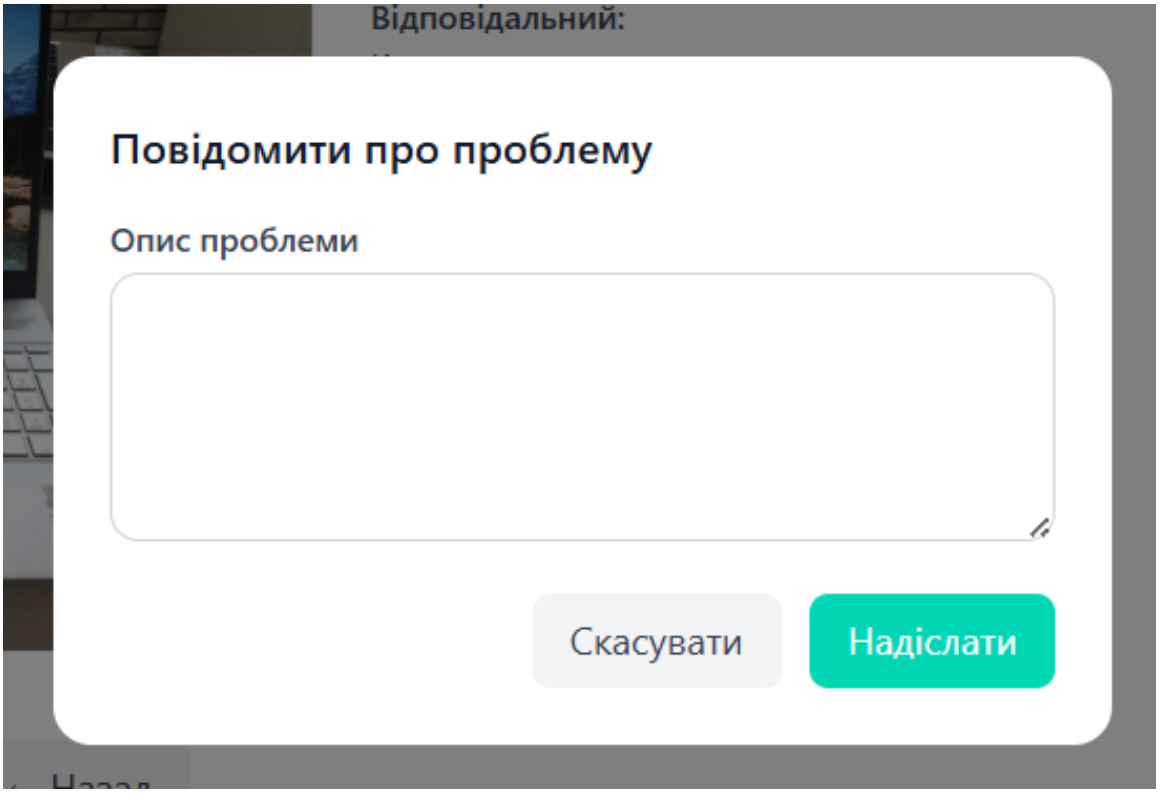


Рисунок 3.12 – Модальне вікно для опису проблеми

Журнал дій (Аудит) надає табличний огляд усіх подій: дата/час, користувач, тип дії та деталі (див. рисунок 3.13).

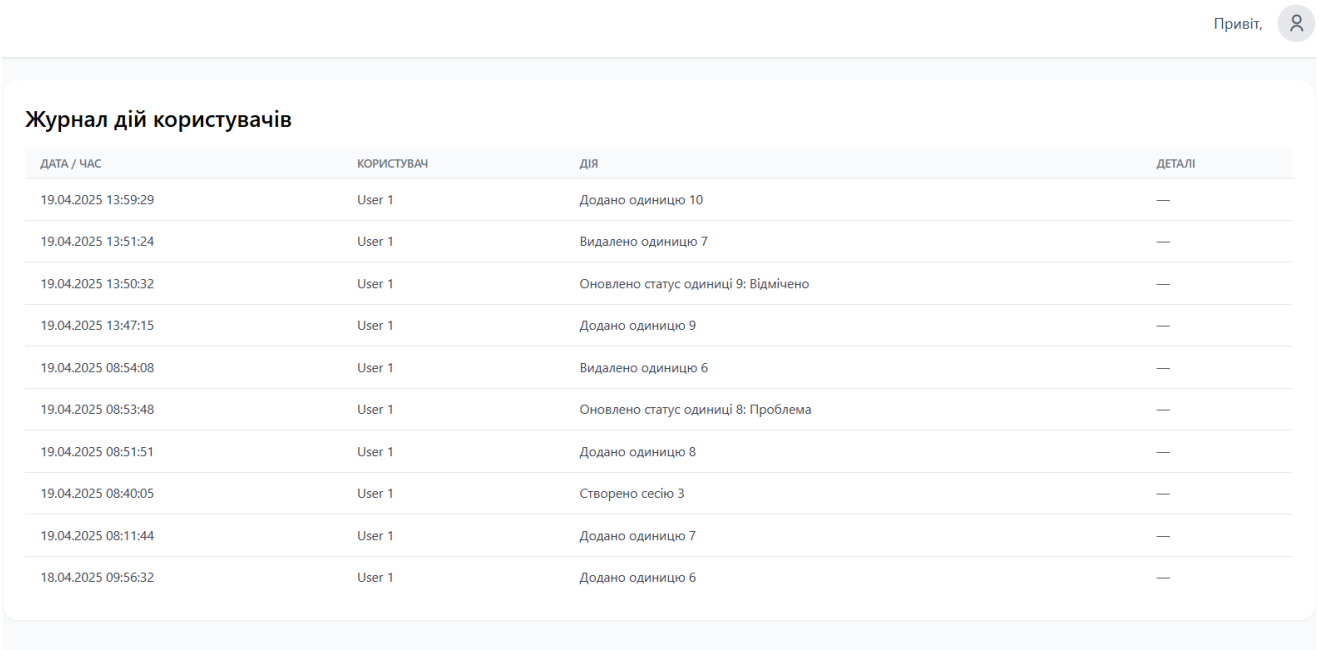


Рисунок 3.13 – Сторінка журналу дій користувачів

Сторінка додавання одиниці має форму, яку користувач заповнює, що б додати у базу даних нову інвентарну одиницю (див. рисунок 3.14)

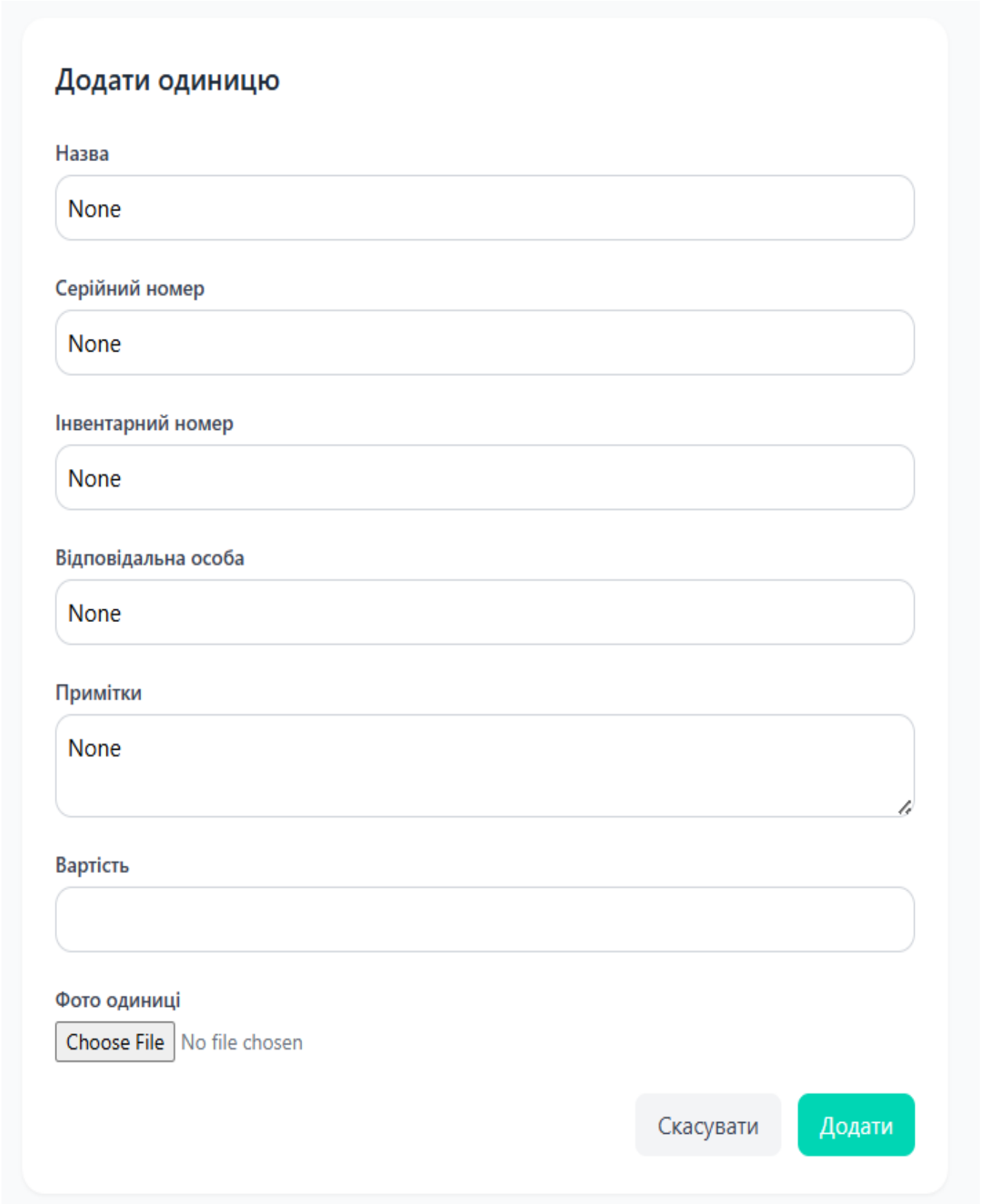


Рисунок 3.14 – Форма додавання нової інвентарної одиниці

Розробка веб-інтерфейсу здійснювалася з використанням Jinja2 (шаблонізатор у складі Flask), Tailwind CSS (сучасний утилітарний підхід до стилізації) та JavaScript (з використанням Fetch API для асинхронних запитів). Такий стек технологій дозволив створити швидкий, адаптивний та інтуїтивно зрозумілий інтерфейс при мінімальній кількості зовнішніх залежностей.

Jinja2 забезпечив гнучке формування HTML на стороні сервера, дозволяючи динамічно генерувати сторінки з урахуванням ролей користувачів та поточних даних. Tailwind CSS спростив розробку адаптивного дизайну завдяки готовим утилітарним класам, що дозволило уникнути роздутого CSS-коду та забезпечити кросплатформенність. Для покращення користувацького досвіду був використаний JavaScript з Fetch API, що дало можливість реалізувати плавне оновлення даних без перезавантаження сторінки (наприклад, при фільтрації таблиць або відправці форм).

Окремо варто відзначити продуману архітектуру клієнтської частини, де логіка інтерфейсу була чітко поділена на модулі (взаємодія з API, рендеринг компонентів, обробка подій). Це дозволило легко додавати новий функціонал та підтримувати існуючий код.

3.6 Висновок

Розділ 3 присвячений детальному аналізу та обґрунтуванню концептуальних підходів до реалізації веб-застосунку для автоматизації обліку інвентарю на підприємстві. Після вивчення різних аспектів проєктування та програмної реалізації системи можна зробити низку висновків, які підтверджують ефективність і доцільність обраної архітектури та технологічної бази.

Одним із ключових аспектів реалізації є ретельне планування архітектури веб-застосунку з урахуванням вимог до масштабованості, надійності та зручності у використанні. Застосування мікрофреймворку Flask у поєднанні з Python забезпечує легкість розробки серверної частини, високу швидкість обробки запитів та можливість інтеграції з іншими інструментами (наприклад, системами

звітності чи CRM).

Важливим досягненням є створення інтерфейсу, орієнтованого на простоту й логічність для трьох ключових категорій користувачів: адміністратора, бухгалтера та інвентаризатора. Завдяки використанню HTML, Tailwind CSS і JavaScript реалізовано адаптивний і сучасний дизайн, який дозволяє швидко освоїти функціонал системи без попередньої технічної підготовки. Інтерфейс включає таблиці з фільтрацією, форму для додавання інвентарних одиниць, сканування QR-кодів та модальні вікна для редагування даних.

Окрему увагу приділено розробці модуля логування дій користувачів. Усі критичні операції — створення, редагування, зміна статусів, запуск сесій перевірки — фіксуються в журналі змін із зазначенням часу, користувача, IP-адреси та опису події. Це дозволяє забезпечити повний аудит усіх дій у системі, посилити інформаційну безпеку та забезпечити прозорість процесів інвентаризації.

Чітке розмежування прав доступу є ще одним важливим компонентом системи. Реалізовано рольову модель, яка забезпечує розподіл функціоналу між адміністративною частиною, бухгалтерським модулем та інтерфейсом інвентаризатора. Такий підхід дозволяє уникнути конфліктів у доступі до ресурсів і знижує ризик людських помилок.

З урахуванням специфіки обліку інвентарного майна, система дозволяє ефективно відстежувати статус кожної одиниці, історію змін, відповідальну особу та місце зберігання. Також реалізовано підтримку QR-кодів, що спрощує ідентифікацію об’єктів під час інвентаризації.

Принцип модульності, який було покладено в основу системи, забезпечує простоту її подальшого масштабування. Нові функції можуть бути додані без зміни існуючої структури, що важливо у довгостроковій перспективі: система може адаптуватися під специфіку конкретного підприємства чи галузі.

Завдяки проведеному тестуванню (модульному та на основі реальних сценаріїв) було виявлено й усунено низку потенційних проблем. Це дозволило підвищити стабільність системи, забезпечити коректну обробку помилок і

гарантувати безвідмовну роботу в умовах різного рівня навантаження.

Розроблений застосунок здатен обробляти великі обсяги даних завдяки використанню PostgreSQL і ORM-бібліотеки SQLAlchemy. Архітектура запитів оптимізована для швидкої вибірки, сортування й фільтрації, що гарантує високу продуктивність навіть при тисячах записів.

Безпека системи досягається за рахунок автентифікації користувачів, захисту форм від CSRF-атак, контролю сесій та використання сучасних методів шифрування паролів. Усі ці заходи унеможливлюють несанкціонований доступ до чутливої інформації та відповідають сучасним вимогам до захисту даних.

У підсумку можна стверджувати, що розроблена система автоматизації обліку інвентарю є ефективним і масштабованим рішенням, яке може бути впроваджене на підприємствах різного типу. Вона сприяє покращенню процесів інвентаризації, підвищенню прозорості обліку та зниженню адміністративних витрат.

Завдяки гнучкій архітектурі, продуманій логіці взаємодії користувачів і надійним технічним рішенням система має значний потенціал для подальшого розвитку та інтеграції в корпоративну IT-інфраструктуру.

4. ТЕСТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ ОБЛІКУ ІНВЕНТАРЮ НА ПІДПРИЄМСТВІ

4.1 Тестування програмної системи

Тестування програмного забезпечення є важливим етапом розробки, який дозволяє перевірити його функціональність, стабільність та відповідність очікуванням користувачів [23]. Основною метою тестування є виявлення помилок і забезпечення надійної роботи системи інвентаризації перед її використанням у реальних умовах.

Перед початком тестування було розгорнуто тестову версію застосунку на сервері з операційною системою Windows, а доступ до клієнтської частини здійснювався через браузер (Google Chrome). База даних PostgreSQL також була попередньо налаштована з тестовими записами інвентарних одиниць.

Перший етап тестування полягав у перевірці основної функціональності, що включає перегляд списку інвентарних одиниць, відкриття сторінки конкретного об’єкта через QR-код і зміну його статусу.

Наступним етапом є тестування на відмовостійкість, метою якого є виявлення дефектів та несправностей у програмному забезпеченні. Під час цього етапу аналізується здатність системи відновлювати роботу після збоїв або критичних помилок. Тестові сценарії можуть передбачати:

1. Навмисне введення коректованих даних для перевірки обробки помилок.

2. Використання неочікуваних вхідних параметрів.

3. Імітацію роботи в екстремальних умовах, зокрема при відсутності зв’язку з іншими компонентам и системи.

Після успішного проходження функціонального тестування та перевірки на відмовостійкість система переходить до етапу тестування продуктивності. На цьому кроці аналізуються показники швидкодії програми: час виконання ключових операцій (наприклад пошук інвентарних одиниць), реакція на одночасні запити користувачів та стабільність роботи під навантаженням.

Під час першого етапу тестування програмних модулів, було протестовано:

1. Завантаження таблиці інвентарю та її відображення на головній сторінці.
2. Додавання інвентарної одиниці.
3. Редагування інвентарної одиниці.
4. Видалення інвентарної одиниці.
5. Роботу пошуку за назвою інвентарної одиниці.
6. Поведінку системи при скануванні QR-коду, що веде до конкретної інвентарної одиниці.
7. Функціональність кнопки "Відмітити як перевірено", яка змінює статус об’єкта та зберігає дату перевірки.
8. Обробку натискання на кнопку "Повідомити про проблему", яка також змінює статус і зберігає дату останньої дії.

Сам вигляд таблиці інвентарю одразу закладений на сторінку. Контент в таблицю надходить з бази даних, та становить собою інвентарні одиниці. Клієнт посилає на базу GET запит коли заходить на сторінку з інвентарем, або перезавантажує її. В результаті перевірки, та численних перезавантажеть сторінки ніяких збоїв виявлення не було, в усіх випадках в таблицю завантажується контент (див. рисунок 4.1).

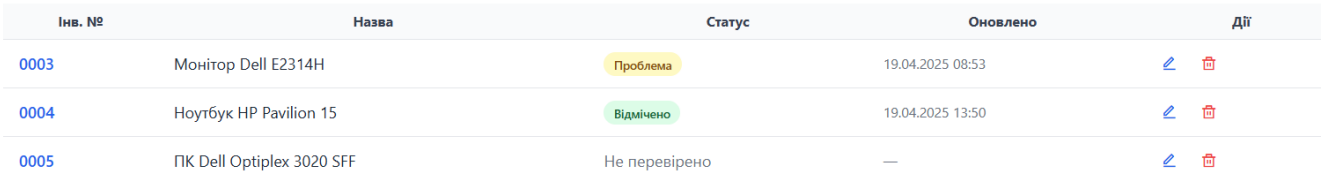


Рисунок 4.1 – Таблиця інвентарних одиниць

Додавання інвентарної одиниці відбувається зокрема на клієнтській частині системи, коли користувач натискає кнопку “Додати одиницю” та заповнює відповідну форму (див. рисунок 4.2).

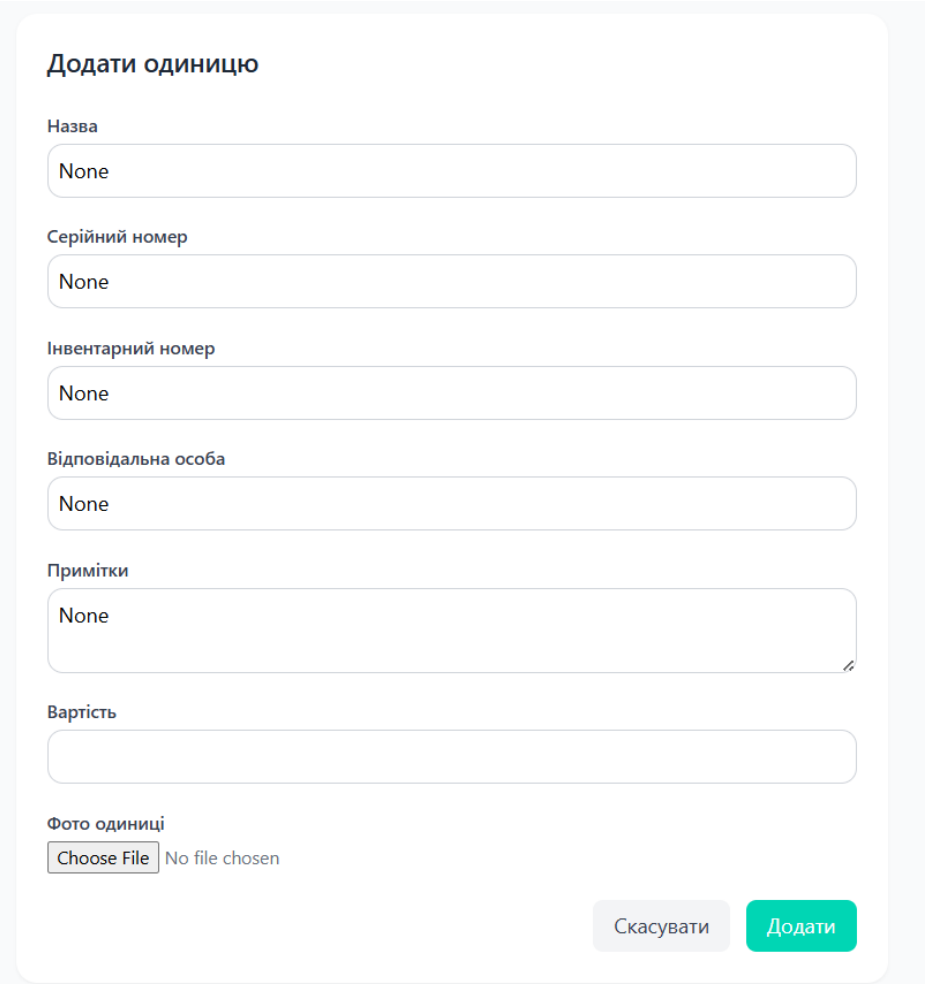


Рисунок 4.2 – Форма додавання інвентарних одиниць

Форма містить як обов’язкові поля, такі як серійний та інвентарний номери, так і не обов’язкові, такі як фото одиниці, вартість, примітки. При спробі додати інвентарну одиницю, не вказуючи при цьому серійний номер, система видасть помилку, повідомить про це користувача, та не запише в базу нову одиницю. Якщо ж користувач введе в поле інвентарного номеру значення, яке вже існувало та приписане до іншої інвентарної одиниці, то система повідомить йому про неможливість додавання нової інвентарної одиниці з вже існуючим інвентарним номером (див. рисунок 4.3).

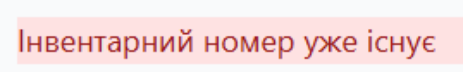


Рисунок 4.3 – Помилка “Інвентарний номер уже існує”

Функція редагування інвентарної одиниці несе за собою можливість змінити наявну інформацію по вже існуючому запису. Наприклад, візьмемо інвентарну одиницю з інвентарним номером 0004. В полі “Примітки” по цій одиниці не містилось ніякої інформації (див. рисунок 4.4).

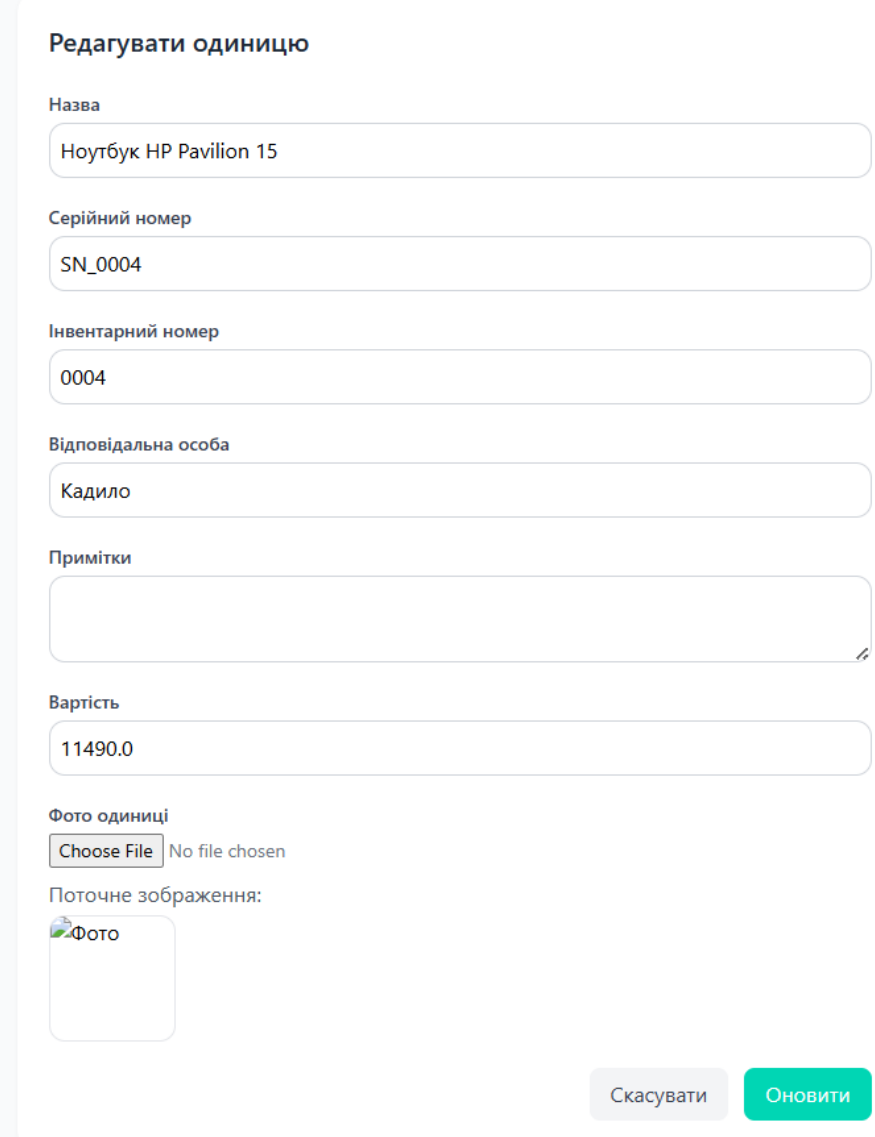


Рисунок 4.4 – Вікно редагування інвентарної одиниці

Напишемо тестові дані поле “Примітки” та завантажимо нову фотографію для цього засобу. Внаслідок редагування інвентарної одиниці, інформація про неї була успішно змінена без жодних помилок (див. рисунок 4.5). Також система за допомогою надпису дає користувачам чітке розуміння що їх дія пройшла успішно.

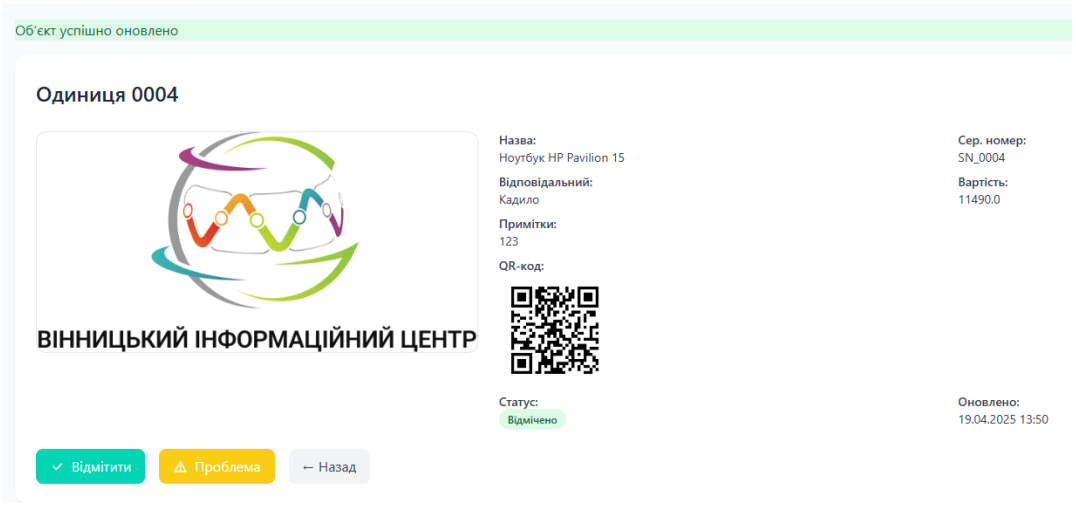


Рисунок 4.5 – Успішне редагування одиниці зі зміною зображення та поля приміток

Функція видалення інвентарної одиниці містить собою посилання DELETE запиту з клієнтської частини на сервер, а у висновку видалення з бази інвентарної одиниці. Бачимо в базі 3 інвентарних одиниці (див. рисунок 4.6).

	id [PK] integer	name character varying (120)	serial_number character varying (120)	inventory_number character varying (120)
1	8	Монітор Dell E2314H	SN_0003	0003
2	9	Ноутбук HP Pavilion 15	SN_0004	0004
3	10	ПК Dell Optiplex 3020 SFF	SN_0005	0005

Рисунок 4.6 – Таблиця БД з 3-а інвентарними одиницями

Після спроби через користувацький інтерфейс видалити одиницю з інвентарниим номером 0004, інвентарна одиниця була успішно видалена з бази (див. рисунок 4.7).

	id [PK] integer	name character varying (120)	serial_number character varying (120)	inventory_number character varying (120)
1	8	Монітор Dell E2314H	SN_0003	0003
2	10	ПК Dell Optiplex 3020 SFF	SN_0005	0005

Рисунок 4.7 – Таблиця БД після видалення інвентарної одиниці 0004

Система також повідомляє користувача, що одиниця була видалена надписом: “Об’єкт успішно видалено”.

Пошук по таблиці в інтерфейсі здійснюється через клієнтську частину системи. Після тестування визначено, що функція пошуку працює справно, та шукає інвентарні одиниці з відповідною назвою або інвентарним номером. Результат тестування можна побачити на рисунках 4.8 та 4.9.

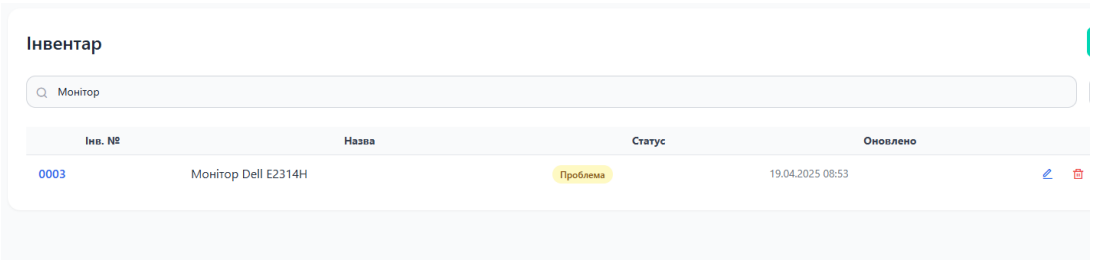


Рисунок 4.8 – Результат пошуку за запитом “Монітор”

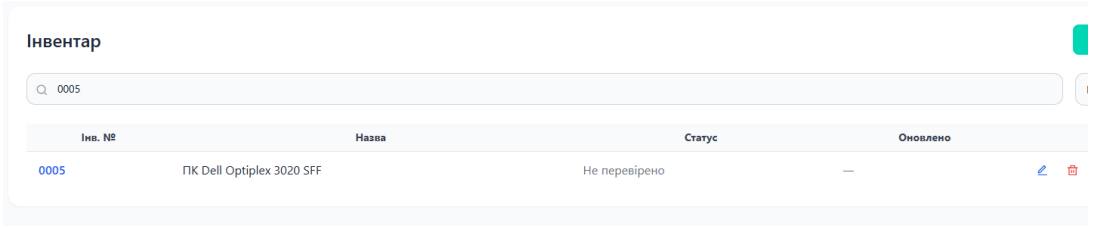


Рисунок 4.9 – Результат пошуку за запитом “0005”

QR-коди мають містити посилання на саме ту одиницю інвентарю, для якої він створювався, що б коли працівники роздрукували його, вони б могли одразу його приклеїти до того чи іншого засобу. Перевірка правильності qr-кодів була здійснена через сервіс MEQRScanner [24]. QR код можна отримати або через сторінку з детальною інформацією про одиницю, або безпосередньо через директорію де ці коди зберігаються (див. рисунок 4.10).

Рисунок 4.10 – Зображення QR-коду в веб інтерфейсі системи

Внаслідок сканування QR-коду, було виявлено, що він веде за адресою http://127.0.0.1:5000/inventory/10, що відповідає місцерозташуванню нашої одиниці з інвентарним номером 0005 (див. рисунок 4.11).

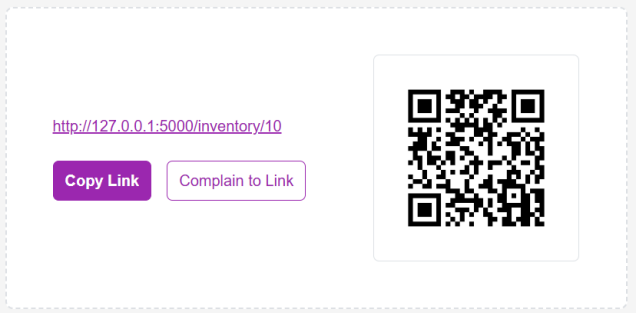


Рисунок 4.11 – Результат сканування QR-коду

Хоч і в посиланні написано 10, але сторінки формуються по ідентифікатору, та немають нічого спільного з інвентарним номером, тому шлях вказано вірно, інформація про одиницю відповідає дійсності (див. рисунок 4.12).

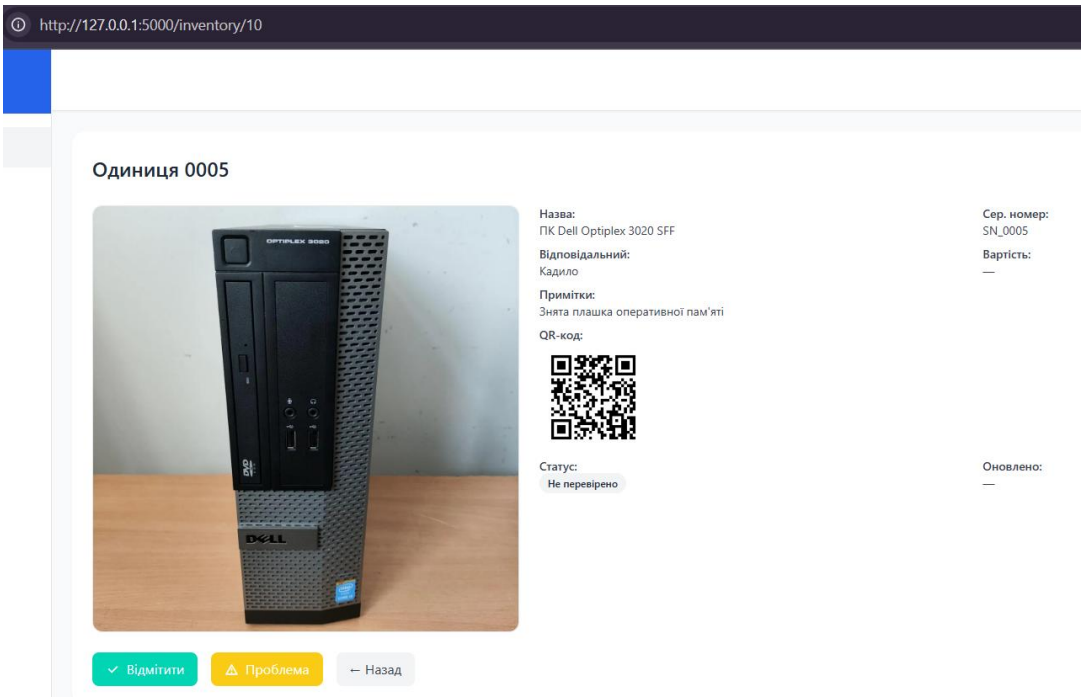


Рисунок 4.12 – Сторінка на яку веде QR-код

Важливою функцією системи – є сам процес інвентаризації, який заключає в собі відмітку тих інвентарних одиниць, які є. А також повідомлення про

проблеми, такі як пошкодження. Кожна одиниця має свій статус, їх може бути всього 3: “Не перевірено”, “Відмічено” та “Проблема” (див. рисунок 4.13).

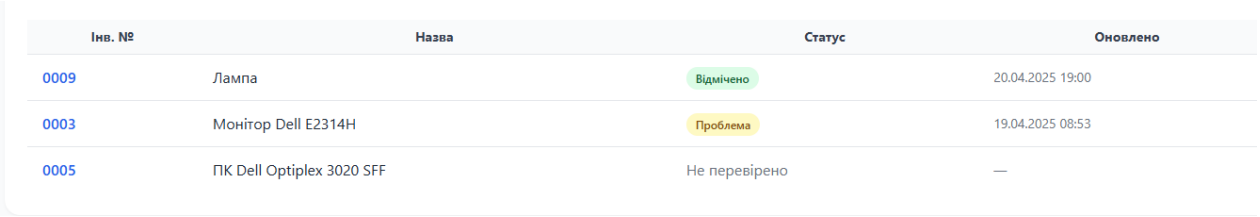


Рисунок 4.13 – Інвентарні одиниці з їх статусами

Також, в полі “Оновлено” міститься інформація про останній час коли було відмічено, або повідомлено про проблему по інвентарній одиниці. Після натискання кнопки відмітити на одиниці з інвентарним номером 0005, можна побачити, що її статус успішно оновився, та отримав значення “Відмічено” з точним до хвилини часом коли це відбулось (див. рисунок 4.14)

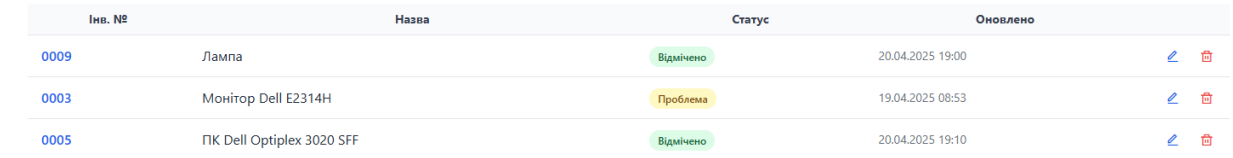


Рисунок 4.14 – Інвентарні одиниці після відмітки 0005

Отже функціонал відмітки працює правильно та без збоїв. Аналогічним чином протестовано функцію повідомлення про проблему по інвентарному засобу. На цей раз статус змінено одиниці з інвентарним номером 0009. (див. рисунок 4.15).

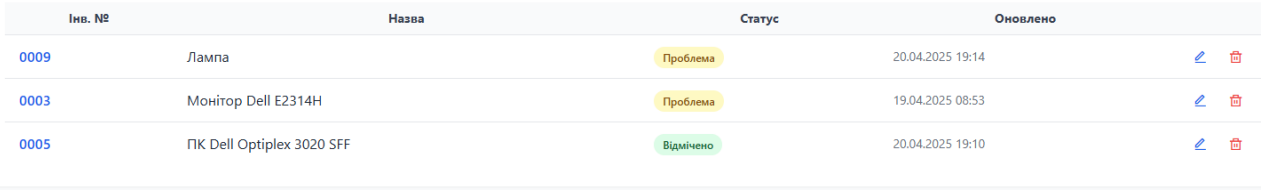


Рисунок 4.15 – Інвентарні зі зміною статусу на проблема для 0009

Що б дізнатись як саме користувач описав проблему, достатньо просто зайти на детальну інформацію по інвентарній одиниці, там будуть відповідні поля, де можна побачити статус та опис проблеми (див. рисунок 4.16).

Рисунок 4.16 – Опис проблеми для інвентарної одиниці з номером 0009

Окремо було протестовано інтерфейс користувача, його зручність та відповідність принципам юзабіліті. Зокрема:

1. Інтерфейс адаптивно відображається на різних пристроях (ноутбуки, планшети, телефони).

2. Кнопки мають чіткі підписи та візуальний зворотний зв’язок при взаємодії.

3. Колірна схема сприяє фокусуванню уваги користувача на основних діях.

Також проведено перевірку доступності інтерфейсу для користувачів із кольоровою сліпотою [6]. Завдяки проведеному аудиту, було визначено, що система відповідає не тільки принципам юзабіліті, але й інклюзивності, завдяки використанню не лише кольору, а й іконок для статусів.

Проведено тестування на реальних користувачах, бухгалтерах підприємства. Суть цього тестування – це визначити, чи правильно побудований інтерфейс, та чи можуть реальні користувачі з легкістю здійснити потрібні дії по

користувацькому шляху [7]. Тестування було здійснено за допомогою інструменту Hotjar, в результаті тестуваннння отримано теплову карту, яка показує найбільші взаємодії користувачів та перебування їх курсору миші (див. рисунок 4.17).

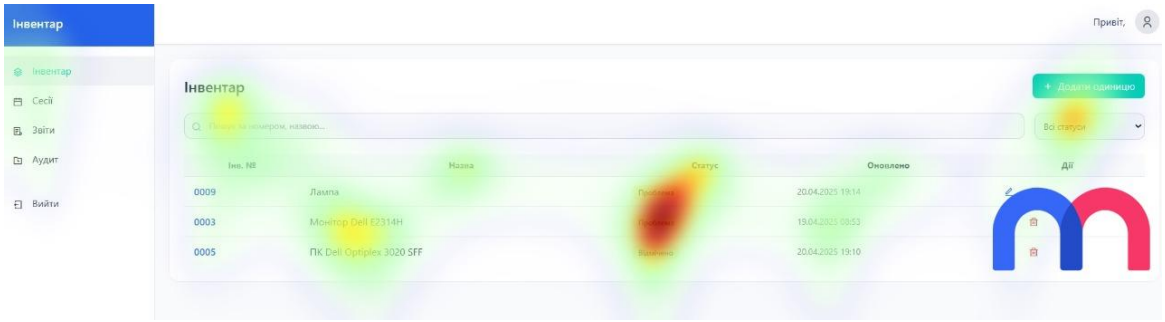


Рисунок 4.17 – Результат проведеного тестування у вигляді heatmap

Завданням для користувачів було звірити відмічені інвентарні одиниці вручну в системі. Всі респонденти успішно виконали завдання, та виявили що всі відмічені інвентарні одиниці також відмічені і в системі. По тепловій карті [8] можна зрозуміти, що респонденти акцентували увагу саме на тих елементах інтерфейсу, які потрібні для виконання задачі. А саме: пошук, фільтри, статуси, назва, оновлено. Отже, можна зазначити, що інтерфейс виконує свою задачу, та забезпечує зручу та зрозумілу взаємодію з користувачами.

Було проведено базове тестування швидкодії при завантаженні списку інвентаря та оновленні статусів. На тестових даних (до 500 записів) час відгуку інтерфейсу на основні дії (завантаження таблиці, натискання кнопок) не перевищував 0.3–0.5 секунди, що свідчить про оптимальну швидкодію інтерфейсу.

4.2 Створення інструкції для користувачів

Інструкція користувача містить опис технічних умов, необхідних для успішного запуску програмного забезпечення. Мінімальні та рекомендовані параметри апаратного та програмного середовища детально вказані у таблиці 4.1, що дозволяє користувачам переконатися у відповідності систем цим вимогам.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Вимоги до конфігурування	Рекомендовано	Мінімально
Процесор	Intel Core i5 або AMD Ryzen 5	Intel Core i3
Оперативна пам'ять	8 ГБ RAM	4 ГБ RAM
Вільний простір на диску	64 ГБ	12 ГБ
Операційна система	Windows 10/11, Ubuntu 20.04+, macOS 12+	Windows 7, Ubuntu 18.04, macOS 10.14
Браузер	Google Chrome, Mozilla Firefox, Edge	Будь-який сучасний браузер із підтримкою JavaScript
Додатково	Наявність камери або сканера QR-кодів	—

Після встановлення залежностей (pip install -r requirements.txt), налаштування з’єднання з PostgreSQL та запуску Flask-сервера командою python run.py користувач відкриває у браузері адресу http://localhost:5000/inventory/. На екрані відображається список існуючих інвентарних одиниць із їхніми унікальними номерами, назвами та поточними статусами. У верхній частині сторінки розташовано поле пошуку та випадний список фільтрації за статусом, що дозволяють миттєво звузити результати без перезавантаження сторінки.

Додавання нової одиниці здійснюється кнопкою «+ Додати одиницю», розташованою праворуч над таблицею. Після натискання відкривається форма, де необхідно вказати назву, серійний номер, інвентарний номер та за бажанням – фото і додаткові примітки. Поле «Фото» підтримує завантаження файлів і одразу відображає прев’ю малюнка. Після натискання «Додати» нова одиниця з’являється вгорі списку та отримує згенерований QR-код.

Для перегляду деталей конкретного об’єкта користувач може натиснути на його інвентарний номер у таблиці або відсканувати QR-код, який приведе на сторінку .../inventory/<id>. Тут відображаються всі атрибути одиниці, фото (якщо було завантажено) і дві кнопки дій: «Відмітити» та «Повідомити про проблему». Перша призначена для фіксації факту перевірки — після кліку статус змінюється на «Перевірено» та зберігається мітка дати й часу. Друга відкриває поле для

введення коментаря про несправність або відсутність і позначає одиницю як «Проблема» з прив’язкою до тексту повідомлення.

Редагування існуючої одиниці відбувається через кнопку “Олівець” у рядку таблиці — вона відкриває ту ж саму форму, що й при додаванні, але із заповненими полями. Після внесення змін слід натиснути «Оновити», і таблиця автоматично відобразить оновлені дані. Видалення відбувається через кнопку “Смітник”, після чого з’являється модальне вікно підтвердження дії. Підтвердження видалення назавжди прибирає запис із бази та позначається в журналі аудиту.

Журнал дій доступний у бічному меню розділом «Аудит». Веб-сторінка /reports/audit виводить табличний перелік усіх подій, що фіксуються модулем логування: дата й час, ім’я користувача та опис виконаної операції. Це дозволяє відстежити історію змін та швидко знайти інформацію про додавання, редагування, маркування чи видалення одиниць.

Протягом роботи користувач може у будь-який момент повертатися до головного списку, застосовувати нові значення пошуку та фільтрувати дані за статусом для оперативного контролю. Інтерфейс адаптовано під різні розміри екранів, від десктопів до планшетів та телефонів.

Таким чином, інструкція відображає весь наявний функціонал системи інвентаризації: перегляд, пошук, додавання, редагування, маркування одиниць, видалення та аудит подій.

4.4 Висновок

У розділі 4 було детально розглянуто процес реалізації та технічного забезпечення веб-застосунку для автоматизованого обліку інвентарю на підприємстві, зокрема визначено основні вимоги до його функціонування та інфраструктури. Формування правильних технічних параметрів є критично важливим, адже саме вони впливають на надійність, продуктивність системи та якість взаємодії з користувачем.

Для забезпечення стабільної роботи веб-застосунку, особливо при обробці

великої кількості інвентарних одиниць і одночасному доступі кількох користувачів, необхідно враховувати як мінімальні, так і рекомендовані технічні вимоги. Це дозволяє забезпечити безперебійне функціонування системи навіть у складних умовах експлуатації.

Застосунок, як основний інструмент для працівників підприємства, повинен бути оптимізований для швидкого обміну інформацією та оновлення даних у режимі реального часу. Це досягається завдяки правильно налаштованому серверному середовищу, ефективному управлінню запитами та зниженню навантаження на клієнтську частину. Використання сучасного технологічного стеку, зокрема фреймворку Flask, мови Python та бази даних PostgreSQL, забезпечує масштабованість, швидкодію та гнучкість розгортання.

Успішне визначення вимог до інтерфейсу та логіки користувацької взаємодії дозволяє створити простий та інтуїтивно зрозумілий інтерфейс. Це дає змогу користувачам швидко орієнтуватися у системі, зменшуючи кількість помилок і підвищуючи ефективність виконання основних дій. Тестування на різних пристроях і браузерах допомагає гарантувати кросплатформену сумісність і стабільність роботи незалежно від середовища використання.

Особливу увагу слід приділяти роботі в умовах нестабільного або повільного інтернет-з'єднання. Визначення мінімально необхідної швидкості та покращення запитів дозволяють зберегти працездатність системи навіть за слабких мережевих умов. Це важливо для підприємств, які мають розподілену структуру або працюють у регіонах із низькою якістю зв’язку.

Не менш значущим є правильний вибір інструментів та середовищ розробки, які забезпечують відповідність технічним вимогам проєкту та можливість подальшої адаптації системи до змін. Технології Flask, Python, HTML, Tailwind CSS і JavaScript використовуються для створення гнучкого, продуктивного й надійного інтерфейсу з мінімальним навантаженням на серверну частину.

У підсумку, визначення технічних вимог є фундаментальним етапом при розробці веб-застосунку, що має відповідати реальним потребам підприємства.

Саме завдяки чіткому технічному плануванню забезпечується висока продуктивність, стабільність і зручність використання системи обліку інвентарю.

Технічні вимоги повинні бути адаптовані до особливостей проєкту — з урахуванням типу користувачів, кількості об’єктів, що підлягають обліку, а також особливостей інфраструктури. Правильне їх визначення на ранньому етапі дозволяє мінімізувати ризики в процесі запуску системи та підвищити рівень задоволеності користувачів.

Таким чином, ретельне формування технічних вимог є запорукою стабільної, ефективної та конкурентоспроможної системи. Саме ці чинники визначають якість веб-застосунку в умовах практичного використання, сприяючи його успішному впровадженню та масштабуванню в майбутньому.

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи було розглянуто та всебічно проаналізовано основні аспекти, що стосуються створення системи для автоматизованого обліку інвентарю на підприємстві. Завданням роботи було не лише розробити веб-застосунок, але й визначити технічні параметри, побудувати логічну структуру даних та запропонувати ефективні рішення для забезпечення стабільної роботи системи.

Робота почалася з вивчення теоретичних підходів і аналізу сучасних технологій веб-розробки. Основною метою було створити систему, яка дозволяє безпечно та ефективно працювати з даними інвентаризації, забезпечує доступність через браузер і відповідає специфічним вимогам підприємства.

Однією з основних задач, яку вирішував даний проєкт, було формулювання апаратних і програмних вимог до забезпечення швидкої та надійної роботи системи. Для цього було проведено аналіз технічних характеристик, що дозволило підібрати оптимальну конфігурацію для різних умов експлуатації — від локальних серверів до хмарного хостингу.

Важливим етапом розробки стало визначення структури бази даних, яка зберігає інформацію про інвентарні об'єкти, їх характеристики, статуси, місця розташування та відповідальних осіб. Особливу увагу приділено оптимізації запитів, що забезпечує швидкий доступ до даних навіть при великій кількості записів.

Ще одним важливим аспектом роботи була розробка інтерфейсу веб-застосунку. Оскільки система призначена для працівників різних посадових рівнів, інтерфейс було реалізовано максимально доступним і простим у використанні. Забезпечено швидку навігацію, чітку структуру дій і зрозумілу візуалізацію даних, що покращує загальний користувацький досвід.

Веб-застосунок був реалізований із використанням сучасних технологій, таких як Flask (Python), HTML, CSS (Tailwind), JavaScript і PostgreSQL.

Обраний стек технологій дозволяє досягти високої продуктивності, забезпечити масштабованість і спростити подальший супровід. Також було впроваджено базові засоби захисту: автентифікацію користувачів, контроль доступу за ролями, логування подій та захист конфіденційних даних.

Однією з найбільших складностей на етапі реалізації стало забезпечення стабільної роботи інтерфейсу в умовах обмежених технічних ресурсів і низької швидкості з’єднання. Для вирішення цієї задачі було впроваджено адаптивну верстку, а також оптимізовано завантаження візуального контенту, що дозволяє зберігати швидкодію навіть на слабших пристроях.

Завдяки чітко визначеним технічним вимогам і ретельно продуманій архітектурі система показала високу стабільність, надійність і зручність у використанні. Вона забезпечує швидкий доступ до інформації, чіткий розподіл ролей і злагоджену роботу з великим масивом інвентарних даних.

Також необхідно відзначити, що робота включає проведення серії тестувань, які дозволили виявити потенційні несправності на ранніх етапах та усунути їх. Проведене модульне й інтеграційне тестування дало змогу підтвердити стабільність роботи системи в умовах, наближених до реального середовища експлуатації.

В цілому, робота продемонструвала важливість точного формулювання вимог, обґрунтованого вибору технологій та системного підходу до тестування. Це дозволило створити програмне рішення, яке повністю відповідає поставленим цілям, є технічно грамотним і зручним у повсякденному використанні.

У ході виконання роботи були отримані не тільки практичні результати, але й набуті теоретичні знання з розробки інформаційних систем, структурування даних і управління програмними проєктами.

Отже, виконана бакалаврська кваліфікаційна робота є результатом успішного втілення сучасних технологій у розробці програмного забезпечення. Створена система задовольняє актуальні вимоги щодо функціональності, безпеки та зручності й може слугувати базою для

подальших досліджень і впроваджень у сфері автоматизованого обліку інвентарного майна.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Enterprise Inventory Management [Електронний ресурс] – Режим доступу до ресурсу: https://budibase.com/blog/inside-it/enterprise-inventory-management (дата звернення: 25.03.2025)

2. Importance of Inventory Management: Key Components and Best Practices [Електронний ресурс] – Режим доступу до ресурсу: https://ezo.io/ezofficeinventory/blog/importance-of-inventory-management-in-an-organization (дата звернення: 25.03.2025)

3. Your Guide to Inventory Management Software [Електронний ресурс] – Режим доступу до ресурсу: https://www.businessnewsdaily.com/15868-what-is-inventory-management-software.html (дата звернення: 25.03.2025)

4. Використання qr кодів у бізнесі [Електронний ресурс] – Режим доступу до ресурсу: https://remonline.ua/blog/qr-code/ (дата звернення: 26.03.2025)

5. Експертна оцінка юзабіліті вашого сайту та 10 евристик Якоба Нільсена [Електронний ресурс] – Режим доступу до ресурсу: https://goldwebsolutions.com/uk/blog/ekspertna-otsinka-yuzabiliti-vashogo-sajtu-ta-10-evristik-yakoba-nilsena/ (дата звернення: 26.03.2025)

6. Альпашкін М.І., Романюк О. Н., Романюк О.В. (2023). Дизайн для людей з особливими потребами: як зробити свій вебсайт доступним для всіх. В матеріалах Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (випуск 74) (07.02.2023)

7. Альпашкін М.І., Мазур В. В., Романюк О. Н. (2023). Редизайн цифрових продуктів та його важливість. В матеріалах Молодь в науці: дослідження, проблеми, перспективи (МН-2023) (ст. 687-689). Вінниця: ВНТУ

8. Alpashkin M.I., Dmytriiev V.G., Romaniuk O.V. (2023). Usability testing and why is it important. In Materials of the XV All-Ukrainian scientific and practical conference “Actual Problems of Computer Science APSCN-2023" (pp. 243-244). Khmelnytsky: KhNU

9. Бухгалтерський облік інвентарю на підприємстві [Електронний ресурс] – Режим доступу до ресурсу: https://www.mogol-alfa.com.ua/ua/blog/bukhhalterskyy-oblik-inventaryu-na-pidpryyemstvi/ (дата звернення: 27.03.2025)

10. Інвентаризація на підприємстві: необхідність, порядок та складнощі проведення [Електронний ресурс] – Режим доступу до ресурсу: https://ordage.com/blog/inventarizaciya-na-pidpriyemstvi-neobhidnist-poryadok-ta-skladnoshi-provedennya (дата звернення: 30.03.2025)

11. Сучасна облікова програма для бізнесу та бухгалтерії [Електронний ресурс] – Режим доступу до ресурсу: https://dilovod.ua/ (дата звернення: 01.04.2025)

12. EVIDEI - Програма для інвентаризації та обліку активів [Електронний ресурс] – Режим доступу до ресурсу: https://systemgroup.com.ua/uk/rishennya-ta-pz/identyfikaciya-ta-oblik-tovariv/evidei-programa-dlya-inventaryzaciyi-ta-obliku-aktyviv (дата звернення: 02.04.2025)

13. WMS система: складський облік і інвентаризація товару [Електронний ресурс] – Режим доступу до ресурсу: https://intelpol.ua/ua/rfid-rishennia/avtomatyzatsiia-obliku-i-inventaryzatsii-tovariv (дата звернення: 09.04.2025)

14. Що таке RFID-мітка [Електронний ресурс] – Режим доступу до ресурсу: https://idcard.com.ua/ua/blog/chto-takoe-sistema-rfid-v-chem-ee-osobennosti-ispolzovaniya (дата звернення: 12.04.2025)

15. Що таке PostgreSQL і для чого використовується? [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/postgresql-shcho-tse/ (дата звернення: 16.04.2025)

16. REST API як спосіб спілкування компонент вебдодатків [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/shcho-take-rest-api/ (дата звернення: 16.04.2025)

17. Альпашкін М.І., Бабюк Н. П. (2025). Автоматизація процесу інвентаризації на підприємстві засобами вебзастосунку з QR-ідентифікацією. В

матеріалах Молодь в науці: дослідження, проблеми, перспективи (МН-2025). Вінниця: ВНТУ

18. Що таке Python? [Електронний ресурс] – Режим доступу до ресурсу: https://acode.com.ua/intro-python/ (дата звернення: 16.04.2025)

19. Flask [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Flask (дата звернення: 22.04.2025)

20. Що таке Tailwind CSS? [Електронний ресурс] – Режим доступу до ресурсу: https://create.t3.gg/uk/usage/tailwind/#overview (дата звернення: 22.04.2025)

21. Jinja [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/Jinja2/ (дата звернення: 02.05.2025)

22. pgAdmin 4 (Windows) [Електронний ресурс] – Режим доступу до ресурсу: https://www.pgadmin.org/download/pgadmin-4-windows/ (дата звернення: 14.05.2025)

23. Що таке тестування ПЗ, його етапи, види, інструменти [Електронний ресурс] – Режим доступу до ресурсу: https://university.sigma.software/what-is-software-testing/ (дата звернення: 16.05.2025)

24. MeQrScanner[Електронний ресурс] – Режим доступу до ресурсу: https://me-qr-scanner.com/ru/qr-scanner (дата звернення: 21.05.2024)

71

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти та науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Директор КП «ВІЦ»
_____С. С. Бригадир
«21» березня 2025р.

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.н.т., професор
_____О.Н.Романюк
«21» березня 2025р.

Технічне завдання
на бакалаврську кваліфікаційну роботу «Розробка програмної системи для обліку інвентарю на підприємстві» за спеціальністю 121-Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.н.т, доцент Н.П.Бабюк
Виконав:
_____студент гр. М.І.Альпашкін
«20» березня 2025р.

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка програмної системи для обліку інвентарю на підприємстві».

Галузь застосування – веб-застосунок.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №79 від «20» березня 2025 р. ректора ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки.

Метою бакалаврської кваліфікаційної роботи є розробка веб-застосунку для обліку інвентарю на підприємстві, який забезпечить ефективну організацію процесів інвентаризації, зручний доступ до інформації про матеріальні ресурси та підвищення якості управління обліковими даними. Застосунок має на меті оптимізувати процеси створення, редагування та перевірки інвентарних одиниць, забезпечуючи при цьому захист даних і простоту у використанні для всіх категорій користувачів.

Призначення роботи – створення та реалізація веб-застосунку для автоматизованого обліку інвентарного майна, який дозволить цифровізувати процеси, пов’язані з веденням, інвентаризацією та контролем об’єктів. Застосунок надасть доступ до ключових функцій, таких як перегляд інвентарних записів, фіксація статусів через QR-коди, генерація звітів та логування дій користувачів. Система має бути зручною як для інвентаризаторів і бухгалтерів, так і для адміністраторів, забезпечуючи швидку роботу з базою даних і високий рівень надійності.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР:

1. Enterprise Inventory Management [Електронний ресурс] – Режим доступу до ресурсу: https://budibase.com/blog/inside-it/enterprise-inventory-management (дата звернення: 25.03.2025)

2. Importance of Inventory Management: Key Components and Best Practices [Електронний ресурс] – Режим доступу до ресурсу: https://ezo.io/ezofficeinventory/blog/importance-of-inventory-management-in-an-organization (дата звернення: 25.03.2025)

3. Your Guide to Inventory Management Software [Електронний ресурс] – Режим доступу до ресурсу: https://www.businessnewsdaily.com/15868-what-is-inventory-management-software.html (дата звернення: 25.03.2025)

4. Alpashkin M.I., Dmytriiev V.G., Romaniuk O.V. (2023). Usability testing and why is it important. In Materials of the XV All-Ukrainian scientific and practical conference “Actual Problems of Computer Science APSCN-2023" (pp. 243-244). Khmelnytsky: KhNU

5. Альпашкін М.І., Мазур В. В., Романюк О. Н. (2023). Редизайн цифрових продуктів та його важливість. В матеріалах Молодь в науці: дослідження, проблеми, перспективи (МН-2023) (ст. 687-689). Вінниця: ВНТУ

6. Альпашкін М.І., Романюк О. Н., Романюк О.В. (2023). Дизайн для людей з особливими потребами: як зробити свій вебсайт доступним для всіх. В матеріалах Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (випуск 74) (07.02.2023)

7. Бухгалтерський облік інвентарю на підприємстві [Електронний ресурс] – Режим доступу до ресурсу: https://www.mogol-alfa.com.ua/ua/blog/bukhhalterskyy-oblik-inventaryu-na-pidpryyemstvi/ (дата звернення: 27.03.2025)

8. Інвентаризація на підприємстві: необхідність, порядок та складнощі проведення [Електронний ресурс] – Режим доступу до ресурсу: https://ordage.com/blog/inventarizaciya-na-pidpriyemstvi-neobhidnist-poryadok-ta-skladnoshi-provedennya (дата звернення: 30.03.2025)

5. Технічні вимоги

Тип програми – веб-застосунок; Модель розробки – ітеративна; Засоби зберігання даних – PostgreSQL; Вхідні дані: дані про інвентарні одиниці (назва, серійний номер, інвентарний номер, місце зберігання, відповідальна особа), статуси перевірок, журнал дій користувачів; Вихідні дані: зведення про інвентар, QR-коди для об'єктів; Середовище розробки – PyCharm, PostgreSQL, Flask, Tailwind CSS; Мови програмування – Python, HTML, CSS, JavaScript; Інструменти для тестування веб-застосунків – Chrome Developer Tools, Postman.

У межах цієї роботи було створено веб-застосунок для обліку інвентарю на підприємстві. Система забезпечує зручний і адаптивний інтерфейс, що дозволяє користувачам переглядати, редагувати та контролювати стан інвентарних одиниць. Застосунок підтримує функції генерації QR-кодів, які дають змогу швидко переходити до об’єкта під час інвентаризації, а також ведення логів дій користувачів.

Усі дані зберігаються в реляційній базі даних PostgreSQL, що гарантує цілісність і доступність інформації у будь-який момент. Система також має багаторівневу авторизацію, що забезпечує розмежування доступу для адміністраторів, бухгалтерів та інвентаризаторів. Адміністративна частина дозволяє керувати користувачами, налаштовувати параметри системи та переглядати аудит змін.

Таким чином, розроблений веб-застосунок відповідає технічним вимогам для автоматизованого обліку інвентарного майна, забезпечуючи надійне збереження даних, швидкий обмін інформацією та високий рівень зручності в роботі з великим обсягом записів.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БКР.
2. Технічне завдання.
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану систем для проведення інвентаризації.	25.03.25-30.03.25
2	Аналіз аналогів програмної системи для проведення інвентаризації на підприємстві.	01.04.25-15.04.25
3	Розробка алгоритмів роботи системи	16.04.25-20.04.25
4	Розробка програмних модулів системи	21.04.25-15.05.24
5	Тестування системи	16.05.25-22.05.25
6	Оформлення матеріалів до захисту БКР	22.05.25-30.05.25

10. Порядок контролю та прийняття

Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

76

Додаток Б. Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка програмної системи для обліку інвентарю на підприємстві

Тип роботи: Бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра ПЗ, ФІТКІ 5ПІ-21Б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 77 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

(прізвище, ініціали, посада) (підпис)

(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник (підпис) к.т.н., доц. каф. ПЗ Бабюк Н. П.
(прізвище, ініціали, посада)

Здобувач (підпис) Альпашкін М.І.
(прізвище, ініціали)

77

Додаток В. Акт впровадження

УЗГОДЖЕНО
Директор КП "Вінницький
Інформаційний центр"
С.С. Бригадир
«28» травня 2025 року

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор О.Н. Романюк
«28» травня 2025 року

АКТ ВПРОВАДЖЕННЯ № _____
результатів науково-дослідних робіт

Замовник КП "Вінницький інформаційний центр"
(найменування організації)

Цим актом підтверджується, що результати роботи – Розробка програмної системи для обліку інвентарю на підприємстві.
(найменування теми)

що виконав студент гр. 5ПІ – 21б ВНТУ Альпашкін М.І.
(виконавець)

за трудовим договором з КП «ВІЦ», на посаді Оператора з обробки інформації та програмного забезпечення відділу інформаційних технологій та системного адміністрування

виконана з 25.03.2025 по 28.05.2025.
(строки виконання)

впроваджено у Комунальному підприємстві "Вінницький інформаційний центр"
(найменування організації, де здійснювалося впровадження)

1. Вид впроваджених результатів: експлуатація програмного забезпечення
(експлуатація виробу, роботи, технології)
2. Характеристика масштабу впровадження: одиничне
(унікальне, одиничне, партія, масове, серійне)
3. Форма впровадження: дослідний зразок
4. Новизна результатів науково-дослідної роботи: якісно нові
(піонерські, принципово нові, якісно нові, модифікації, модернізація старих розробок)
5. Впроваджені: у відділі інформаційних технологій та системного адміністрування
6. Соціальний та науково-технічний ефект: спеціальне призначення
(охорона навколишнього середовища, поліпшення й оздоровлення умов праці, удосконалення структури керування, науково-технічних напрямків, спеціальне призначення)

Від виконавця:
студент групи 5ПІ-21Б
_____ М. І. Альпашкін

Від КП "Вінницький
інформаційний центр
_____ С. С. Бригадир

Керівник: к.т.н., доцент кафедри ПЗ _____ Н. П. Бабюк

Додаток Г. Лістинг програми

```
# run.py
from app import create_app, db
from app.models import User

app = create_app()

with app.app_context():
    db.create_all()

if __name__ == '__main__':
    app.run(debug=True)

# app/models.py
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
from werkzeug.security import generate_password_hash, check_password_hash



db = SQLAlchemy()

class User(db.Model):
    __tablename__ = 'users'
    id            = db.Column(db.Integer, primary_key=True)
    username      = db.Column(db.String(80), unique=True, nullable=False)
    password_hash = db.Column(db.String(128), nullable=False)
    role          = db.Column(db.String(20), nullable=False)  # 'admin','accountant','inventory'

    def set_password(self, password):
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password_hash, password)
```

```
class InventoryItem(db.Model):
    __tablename__ = 'inventory_items'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(120), nullable=False)
    serial_number = db.Column(db.String(120), nullable=False)
    inventory_number = db.Column(db.String(120), unique=True, nullable=False)
    responsible = db.Column(db.String(120))
    notes = db.Column(db.Text)
    price = db.Column(db.Float)
    photo = db.Column(db.String(255))
    qr_code = db.Column(db.String(255))
    current_status = db.Column(db.String(50), nullable=False, default="Не перевірено")
    status_updated_at = db.Column(db.DateTime, nullable=True)
    issue_description = db.Column(db.Text, nullable=True)

    def mark_checked(self):
        from datetime import datetime
        self.current_status = "Відмічено"
        self.status_updated_at = datetime.utcnow()
        self.issue_description = None

    def report_issue(self, description):
        from datetime import datetime
        self.current_status = "Проблема"
        self.status_updated_at = datetime.utcnow()
        self.issue_description = description

class AuditLog(db.Model):
    __tablename__ = 'audit_logs'
    id = db.Column(db.Integer, primary_key=True)
    # Зовнішній ключ до таблиці користувачів
```

```
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    action = db.Column(db.Text, nullable=False)
    timestamp = db.Column(db.DateTime, default=datetime.utcnow)

# Нові моделі для сесій інвентаризації

class InventorySession(db.Model):
    __tablename__ = 'inventory_sessions'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(120), nullable=False)  # Наприклад, "Інвентаризація складу 01.11.2023"
    date = db.Column(db.Date, nullable=False, default=datetime.utcnow)
    description = db.Column(db.Text)
    statuses = db.relationship('SessionItemStatus', backref='session', lazy=True)

class SessionItemStatus(db.Model):
    __tablename__ = 'session_item_statuses'
    id = db.Column(db.Integer, primary_key=True)
    session_id = db.Column(db.Integer, db.ForeignKey('inventory_sessions.id'), nullable=False)
    inventory_item_id = db.Column(db.Integer, db.ForeignKey('inventory_items.id'), nullable=False)
    status = db.Column(db.String(50), nullable=False)  # Наприклад, "присутній", "відсутній", "пошкоджений"
    comment = db.Column(db.Text)

# app/__init__.py
from flask import Flask
from flask_login import LoginManager
from app.models import db, User

login_manager = LoginManager()
login_manager.login_view = 'auth.login'

def create_app():
    app = Flask(__name__)
```

```
    app.config['SQLALCHEMY_DATABASE_URI']      = 'postgresql+psycopg2://
@localhost/inventory_db'
    app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
    app.config['SECRET_KEY']

    db.init_app(app)
    login_manager.init_app(app)

    # Імпорти blueprints
    from app.routes.inventory import bp as inventory_bp
    from app.routes.sessions import bp as sessions_bp
    from app.routes.reports import bp as reports_bp
    from app.routes.auth import bp as auth_bp
    from app.routes.admin import bp as admin_bp
    from app.routes.scan import bp as scan_bp

    # Реєструємо blueprints
    app.register_blueprint(inventory_bp)
    app.register_blueprint(sessions_bp)
    app.register_blueprint(reports_bp)
    app.register_blueprint(auth_bp)
    app.register_blueprint(admin_bp)
    app.register_blueprint(scan_bp)

    @login_manager.user_loader
    def load_user(user_id):
        return User.query.get(int(user_id))

    @app.context_processor
    def inject_current_user():
        # current_user тепер із Flask-Login
        from flask_login import current_user
        return dict(current_user=current_user)
```

```
    return app

# app/routes/admin.py
from flask import Blueprint, render_template, request, redirect, url_for, flash
from flask_login import login_required, current_user
from app.models import db, User

bp = Blueprint('admin', __name__, url_prefix='/admin')

def admin_required(func):
    from functools import wraps
    from flask import abort
    @wraps(func)
    def wrapper(*args, **kwargs):
        if not current_user.is_authenticated or current_user.role != 'admin':
            abort(403)
        return func(*args, **kwargs)
    return wrapper

@bp.route('/users')
@login_required
@admin_required
def list_users():
    users = User.query.all()
    return render_template('admin_users.html', users=users)

@bp.route('/users/new', methods=['GET','POST'])
@login_required
@admin_required
def new_user():
    if request.method=='POST':
        u = User(username=request.form['username'], role=request.form['role'])
        u.set_password(request.form['password'])
```

```
        db.session.add(u); db.session.commit()
        flash('Користувача додано','success')
        return redirect(url_for('admin.list_users'))
    return render_template('admin_user_form.html', user=None)

@bp.route('/users/<int:id>/edit', methods=['GET','POST'])
@login_required
@admin_required
def edit_user(id):
    user = User.query.get_or_404(id)
    if request.method=='POST':
        user.username = request.form['username']
        user.role     = request.form['role']
        if request.form.get('password'):
            user.set_password(request.form['password'])
        db.session.commit()
        flash('Користувача оновлено','success')
        return redirect(url_for('admin.list_users'))
    return render_template('admin_user_form.html', user=user)

@bp.route('/users/<int:id>/delete', methods=['POST'])
@login_required
@admin_required
def delete_user(id):
    user = User.query.get_or_404(id)
    db.session.delete(user); db.session.commit()
    flash('Користувача видалено','warning')
    return redirect(url_for('admin.list_users'))

# app/routes/auth.py
from flask import Blueprint, render_template, redirect, url_for, flash, request
from flask_login import login_user, logout_user, login_required
from app.models import User
```

```
bp = Blueprint('auth', __name__, url_prefix='/auth')

@bp.route('/login', methods=['GET','POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = User.query.filter_by(username=username).first()
        if user and user.check_password(password):
            login_user(user)
            flash('Успішний вхід', 'success')
            # Переходимо назад, або на інвентар за замовчуванням
            return redirect(request.args.get('next') or url_for('inventory.list_items'))
        flash('Невірне ім’я або пароль', 'danger')
    return render_template('login.html')

@bp.route('/logout')
@login_required
def logout():
    logout_user()
    flash('Ви вийшли', 'info')
    return redirect(url_for('auth.login'))

# app/routes/backup.py
import os
import datetime
from flask import Blueprint, jsonify, current_app, send_from_directory
import subprocess
from urllib.parse import urlparse

bp = Blueprint('backup', __name__, url_prefix='/backup')
```

```
@bp.route('/manual', methods=['GET'])
def backup_manual():
    # Отримуємо рядок підключення з конфігурації
    db_uri = current_app.config.get('SQLALCHEMY_DATABASE_URI')
    if db_uri.startswith("postgresql+psycopg2://"):
        conn_str = db_uri.replace("postgresql+psycopg2://", "postgresql://")
    elif db_uri.startswith("postgresql://"):
        conn_str = db_uri
    else:
        return jsonify({"error": "Невірний формат URI бази даних"}), 400

    # Розбираємо рядок підключення
    parsed = urlparse(conn_str)
    username = parsed.username
    password = parsed.password
    host = parsed.hostname
    port = parsed.port if parsed.port else 5432
    dbname = parsed.path.lstrip('/')

    # Формуємо ім'я файлу для резервної копії
    backup_dir = os.path.join(os.getcwd(), "backups")
    os.makedirs(backup_dir, exist_ok=True)
    timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    backup_filename = f"backup_{timestamp}.sql"
    backup_filepath = os.path.join(backup_dir, backup_filename)

    # Формуємо команду для pg_dump із зазначенням параметрів
    command = f"pg_dump -U {username} -h {host} -p {port} -d {dbname} -f
\"{backup_filepath}\""

    # Створюємо копію оточення із встановленою змінною PGPASSWORD
    env = os.environ.copy()
    env["PGPASSWORD"] = password if password else ""
```

```
    try:
        # Використовуємо subprocess.run для захоплення виводу
        result = subprocess.run(command, shell=True, env=env, capture_output=True, text=True)
        if result.returncode != 0:
            return jsonify({
                "error": "Помилка резервного копіювання",
                "details": f"Return code: {result.returncode}\nstdout: {result.stdout}\nstderr:
{result.stderr}"
            }), 500
    except Exception as e:
        return jsonify({"error": "Exception during backup", "details": str(e)}), 500


    return jsonify({
        "message": "Резервну копію створено",
        "backup_file": backup_filename,
        "download_url": f"/backup/download/{backup_filename}"
    })



@bp.route('/download/<filename>', methods=['GET'])
def download_backup(filename):
    backup_dir = os.path.join(os.getcwd(), "backups")
    return send_from_directory(backup_dir, filename, as_attachment=True)
```

```
import os
import qrcode
from flask import (
    Blueprint, request, render_template, jsonify,
    redirect, url_for, flash, current_app
)
from sqlalchemy import or_, asc, desc
from app.models import db, InventoryItem, SessionItemStatus, AuditLog, User
```

```
bp = Blueprint('inventory', __name__, url_prefix='/inventory')


def get_default_user():
    """Ensure there is at least one user (id=1) for audit logs."""
    user = User.query.get(1)
    if not user:
        user = User(username="admin", password_hash="default", role="admin")
        db.session.add(user)
        db.session.commit()
    return user


def generate_qr_code(rel_path, data):
    """
    Generate a QR code for the given data, save it under
    static/<rel_path> and return the same rel_path.
    """
    full_path = os.path.join(current_app.static_folder, rel_path)
    os.makedirs(os.path.dirname(full_path), exist_ok=True)
    qr = qrcode.QRCode(version=1, box_size=10, border=4)
    qr.add_data(data)
    qr.make(fit=True)
    img = qr.make_image(fill_color="black", back_color="white")
    img.save(full_path)
    return rel_path


@bp.route('/', methods=['GET'])
def list_items():
    search = request.args.get('search', '').strip()
    status_f = request.args.get('status', '').strip()
    sort = request.args.get('sort', 'name_asc')
```

```
page = request.args.get('page', 1, type=int)
per_page = 10

query = InventoryItem.query

if search:
    query = query.filter(or_(
        InventoryItem.name.ilike(f'%{search}%'),
        InventoryItem.inventory_number.ilike(f'%{search}%')
    ))

if status_f:
    query = query.filter(InventoryItem.current_status == status_f)

if sort == 'name_desc':
    query = query.order_by(desc(InventoryItem.name))
elif sort == 'date_asc':
    query = query.order_by(asc(InventoryItem.status_updated_at))
elif sort == 'date_desc':
    query = query.order_by(desc(InventoryItem.status_updated_at))
else:  # name_asc or default
    query = query.order_by(asc(InventoryItem.name))

pagination = query.paginate(page=page, per_page=per_page, error_out=False)
items = pagination.items

return render_template(
    'inventory.html',
    items=items,
    pagination=pagination,
    search=search,
    status=status_f,
    sort=sort
```

```
    )


@bp.route('/new', methods=['GET', 'POST'])
def new_item():
    if request.method == 'POST':
        data = request.form
        if InventoryItem.query.filter_by(inventory_number=data['inventory_number']).first():
            flash('Інвентарний номер уже існує', 'danger')
            return redirect(url_for('inventory.new_item'))

        # Handle photo upload
        photo_file = request.files.get('photo')
        photo_rel = None
        if photo_file and photo_file.filename:
            photo_rel = os.path.join('photos', photo_file.filename).replace('\\', '/')
            full_path = os.path.join(current_app.static_folder, photo_rel)
            os.makedirs(os.path.dirname(full_path), exist_ok=True)
            photo_file.save(full_path)

        item = InventoryItem(
            name=data['name'],
            serial_number=data['serial_number'],
            inventory_number=data['inventory_number'],
            responsible=data.get('responsible'),
            notes=data.get('notes'),
            price=float(data['price']) if data.get('price') else None,
            photo=photo_rel
        )
        db.session.add(item)
        db.session.commit()

        # Generate QR code
```

```
        qr_rel = os.path.join('qr_codes', f'item_{item.id}.png').replace('\\', '/')
        url = url_for('inventory.view_item', item_id=item.id, _external=True)
        item.qr_code = generate_qr_code(qr_rel, url)
        db.session.commit()

        # Audit log
        user = get_default_user()
        db.session.add(AuditLog(user_id=user.id, action=f"Додано одиницю {item.id}"))
        db.session.commit()

        flash('Об’єкт успішно додано', 'success')
        return redirect(url_for('inventory.list_items'))

    return render_template('inventory_form.html', item=None)


@bp.route('/<int:item_id>/edit', methods=['GET', 'POST'])
def edit_item_form(item_id):
    item = InventoryItem.query.get_or_404(item_id)
    if request.method == 'POST':
        data = request.form
        item.name = data['name']
        item.serial_number = data['serial_number']
        item.inventory_number = data['inventory_number']
        item.responsible = data.get('responsible')
        item.notes = data.get('notes')
        item.price = float(data['price']) if data.get('price') else None

        photo_file = request.files.get('photo')
        if photo_file and photo_file.filename:
            photo_rel = os.path.join('photos', photo_file.filename).replace('\\', '/')
            full_path = os.path.join(current_app.static_folder, photo_rel)
            os.makedirs(os.path.dirname(full_path), exist_ok=True)
```

```
            photo_file.save(full_path)
            item.photo = photo_rel

        db.session.commit()

        user = get_default_user()
        db.session.add(AuditLog(user_id=user.id, action=f"Оновлено одиницю {item.id}"))
        db.session.commit()

        flash('Об’єкт успішно оновлено', 'success')
        return redirect(url_for('inventory.view_item', item_id=item.id))

    return render_template('inventory_form.html', item=item)


@bp.route('/<int:item_id>/delete', methods=['POST'])
def delete_item_form(item_id):
    item = InventoryItem.query.get_or_404(item_id)
    db.session.delete(item)
    db.session.commit()

    # Optional: remove files
    if item.photo:
        try:
            os.remove(os.path.join(current_app.static_folder, item.photo))
        except:
            pass
    if item.qr_code:
        try:
            os.remove(os.path.join(current_app.static_folder, item.qr_code))
        except:
            pass
```

```
    user = get_default_user()
    db.session.add(AuditLog(user_id=user.id, action=f"Видалено одиницю {item.id}"))
    db.session.commit()

    flash('Об’єкт успішно видалено', 'warning')
    return redirect(url_for('inventory.list_items'))


@bp.route('/<int:item_id>', methods=['GET', 'POST'])
def view_item(item_id):
    item = InventoryItem.query.get_or_404(item_id)

    if request.method == 'POST':
        action = request.form.get('action')
        if action == 'mark':
            item.current_status = 'Відмічено'
            from datetime import datetime
            item.status_updated_at = datetime.utcnow()
            item.issue_description = None
        elif action == 'report':
            desc = request.form.get('issue_description', '').strip()
            if not desc:
                flash('Будь ласка, опишіть проблему', 'danger')
                return redirect(url_for('inventory.view_item', item_id=item.id))
            item.current_status = 'Проблема'
            from datetime import datetime
            item.status_updated_at = datetime.utcnow()
            item.issue_description = desc
        db.session.commit()

        user = get_default_user()
        db.session.add(AuditLog(
            user_id=user.id,
```

```
            action=f"Оновлено статус одиниці {item.id}: {item.current_status}"
        ))
        db.session.commit()

        flash('Статус успішно оновлено', 'success')
        return redirect(url_for('inventory.view_item', item_id=item.id))

    return render_template('unit_detail.html', item=item)


# JSON-ендпоінти (якщо потрібні)
@bp.route('/add', methods=['POST'])
def add_item_json():
    data = request.form
    if InventoryItem.query.filter_by(inventory_number=data['inventory_number']).first():
        return jsonify({'error': 'Інвентарний номер уже існує'}), 400
    # … (подібно до new_item) …
    return jsonify({'message': 'Об’єкт додано', 'item_id': item.id}), 201


@bp.route('/edit/<int:item_id>', methods=['POST'])
def edit_item_json(item_id):
    # … (як у edit_item_form) …
    return jsonify({'message': 'Об’єкт оновлено'})


@bp.route('/delete/<int:item_id>', methods=['DELETE'])
def delete_item_json(item_id):
    # … (як у delete_item_form) …
    return jsonify({'message': 'Об’єкт видалено'})

# app/routes/reports.py
from flask import Blueprint, send_file, request, render_template
```

```
from app.models import InventorySession, SessionItemStatus, InventoryItem, AuditLog
import openpyxl, tempfile

bp = Blueprint('reports', __name__, url_prefix='/reports')


@bp.route('/session/<int:session_id>/excel', methods=['GET'])
def report_session_excel(session_id):
    # Отримання сесії
    session = InventorySession.query.get_or_404(session_id)
    statuses = SessionItemStatus.query.filter_by(session_id=session_id).all()

    # Створення нового Excel файлу
    wb = openpyxl.Workbook()
    ws = wb.active
    ws.title = "Звіт за сесією"

    # Заголовок звіту
    ws.append(["ID сесії", "Назва сесії", "Дата", "Опис"])
    ws.append([
        session.id,
        session.name,
        session.date.strftime("%Y-%m-%d"),
        session.description or ""
    ])
    ws.append([])

    # Таблиця статусів
    ws.append(["Інвентарний №", "Назва", "Статус", "Коментар"])
    for record in statuses:
        item = InventoryItem.query.get(record.inventory_item_id)
        ws.append([
            item.inventory_number if item else "N/A",
```

```
            item.name            if item else "N/A",
            record.status,
            record.comment or ""
        ])

    # Збереження тимчасового файлу
    tmp = tempfile.NamedTemporaryFile(delete=False, suffix='.xlsx')
    wb.save(tmp.name)
    tmp.seek(0)

    return send_file(
        tmp.name,
        as_attachment=True,
        download_name=f"report_session_{session_id}.xlsx",
        mimetype='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet'
    )


@bp.route('/audit', methods=['GET'])
def report_audit():
    # Отримуємо всі записи аудиту, останні зверху
    logs = AuditLog.query.order_by(AuditLog.timestamp.desc()).all()
    return render_template('reports_audit.html', logs=logs)

# app/routes/scan.py
from flask import Blueprint, request, redirect, url_for, abort
from flask_login import login_required, current_user

bp = Blueprint('scan', __name__)

@bp.route('/scan')
@login_required
def scan_item():
```

```
    # Дістаємо з URL сесію та item_id
    session_id = request.args.get('session', type=int)
    item_id    = request.args.get('item', type=int)
    if current_user.role != 'inventory':
        abort(403)
    # Перевіримо, що сесія і одиниця існують (за бажанням)
    return redirect(url_for('sessions.item_status', session_id=session_id, item_id=item_id))

# app/routes/sessions.py
from flask import Blueprint, request, jsonify, render_template
from app.models import db, InventorySession, SessionItemStatus, AuditLog, User
from datetime import datetime

bp = Blueprint('sessions', __name__, url_prefix='/sessions')


def get_default_user():
    # Функція перевіряє, чи є користувач з id=1, і повертає його
    user = User.query.get(1)
    if not user:
        user = User(username="admin", password_hash="default", role="admin")
        db.session.add(user)
        db.session.commit()
    return user


@bp.route('/', methods=['GET'])
def list_sessions():
    sessions = InventorySession.query.all()
    return render_template('sessions.html', sessions=sessions)


@bp.route('/create', methods=['POST'])
```

```
def create_session():
    data = request.form
    # Перетворення рядка дати у об'єкт date (якщо вказано, інакше використовується поточна дата)
    try:
        date_obj = datetime.strptime(data.get('date'), '%Y-%m-%d').date()
    except Exception:
        date_obj = datetime.utcnow().date()

    new_session = InventorySession(
        name=data.get('name'),
        date=date_obj,
        description=data.get('description')
    )
    db.session.add(new_session)
    db.session.commit()

    default_user = get_default_user()
    audit = AuditLog(user_id=default_user.id, action=f"Створено сесію {new_session.id}")
    db.session.add(audit)
    db.session.commit()

    return jsonify({'message': 'Сесію створено', 'session_id': new_session.id}), 201



@bp.route('/update_status', methods=['POST'])
def update_status():
    data = request.form
    session_id = data.get('session_id')
    inventory_item_id = data.get('inventory_item_id')
    status = data.get('status')
    comment = data.get('comment', '')

    # Спроба знайти існуючий запис статусу для цієї сесії й одиниці
```

```
    record = SessionItemStatus.query.filter_by(session_id=session_id,
inventory_item_id=inventory_item_id).first()
    if record:
        record.status = status
        record.comment = comment
    else:
        record = SessionItemStatus(
            session_id=session_id,
            inventory_item_id=inventory_item_id,
            status=status,
            comment=comment
        )
        db.session.add(record)

    db.session.commit()

    default_user = get_default_user()
    audit = AuditLog(user_id=default_user.id,
                     action=f"Оновлено статус для одиниці {inventory_item_id} у сесії {session_id}")
    db.session.add(audit)
    db.session.commit()

    return jsonify({'message': 'Статус оновлено'})
```

```
{% extends 'base.html' %}
{% set active = 'users' %}
{% block title %}{{ user and 'Редагувати' or 'Додати' }} користувача{% endblock %}

{% block content %}
<div class="max-w-lg mx-auto bg-white rounded-lg shadow-sm p-6">
  <h2 class="text-xl font-semibold text-gray-800 mb-6">
    {{ user and 'Редагувати' or 'Додати' }} користувача
  </h2>
  <form method="post" class="space-y-5">
```

```
  <div>
    <label class="block text-sm font-medium text-gray-700">Username</label>
    <input name="username" required
           class="mt-1 block w-full border border-gray-300 rounded-md p-2 focus:ring-primary focus:border-primary"
           value="{{ user and user.username }}">
  </div>
  <div>
    <label class="block text-sm font-medium text-gray-700">Роль</label>
    <select name="role"
            class="mt-1 block w-full border border-gray-300 rounded-md p-2 focus:ring-primary focus:border-primary">
      {% for r in ['admin','accountant','inventory'] %}
      <option value="{{r}}" {% if user and user.role==r %}selected{% endif %}>{{ r }}</option>
      {% endfor %}
    </select>
  </div>
  <div>
    <label class="block text-sm font-medium text-gray-700">
      Password {% if user %}(залишити порожнім){% endif %}
    </label>
    <input name="password" type="password" {% if not user %}required{% endif %}
           class="mt-1 block w-full border border-gray-300 rounded-md p-2 focus:ring-primary focus:border-primary">
  </div>
  <div class="flex justify-end space-x-3">
    <a href="{{ url_for('admin.list_users') }}"
       class="px-4 py-2 bg-gray-100 text-gray-700 rounded-button hover:bg-gray-200">Скасувати</a>
    <button type="submit"
            class="px-4 py-2 bg-primary text-white rounded-button hover:bg-opacity-90">
      {{ user and 'Оновити' or 'Додати' }}
    </button>
  </div>
</form>
```

```
</div>
{% endblock %}


{% extends 'base.html' %}
{% set active = 'users' %}
{% block title %}Користувачі{% endblock %}


{% block content %}
<div class="bg-white rounded-lg shadow-sm p-6">
  <div class="flex justify-between items-center mb-6">
    <h1 class="text-2xl font-semibold text-gray-800">Користувачі</h1>
    <a href="{{ url_for('admin.new_user') }}"
       class="bg-primary text-white px-4 py-2 rounded-button flex items-center hover:bg-opacity-
90">
      <i class="ri-add-line mr-2"></i>Додати користувача
    </a>
  </div>

  <div class="table-container">
    <table class="min-w-full">
      <thead class="bg-gray-50 border-b border-gray-200">
        <tr>
          <th class="text-sm text-gray-700 px-4 py-2">ID</th>
          <th class="text-sm text-gray-700 px-4 py-2">Username</th>
          <th class="text-sm text-gray-700 px-4 py-2">Role</th>
          <th class="text-sm text-gray-700 px-4 py-2">Дії</th>
        </tr>
      </thead>
      <tbody class="divide-y divide-gray-200">
        {% for u in users %}
        <tr>
          <td class="px-4 py-2 text-sm text-gray-800">{{ u.id }}</td>
          <td class="px-4 py-2 text-sm text-gray-800">{{ u.username }}</td>
          <td class="px-4 py-2 text-sm text-gray-800">{{ u.role }}</td>
```

```
        <td class="px-4 py-2">
          <div class="flex space-x-2">
            <a href="{{ url_for('admin.edit_user', id=u.id) }}"
               class="w-8 h-8 flex items-center justify-center rounded-full text-blue-600 hover:bg-blue-50"
               title="Редагувати">
              <i class="ri-edit-line"></i>
            </a>
            <form method="post" action="{{ url_for('admin.delete_user', id=u.id) }}"
                  onsubmit="return confirm('Видалити користувача?');">
              <button type="submit"
                      class="w-8 h-8 flex items-center justify-center rounded-full text-red-500 hover:bg-red-50"
                      title="Видалити">
                <i class="ri-delete-bin-line"></i>
              </button>
            </form>
          </div>
        </td>
      </tr>
      {% else %}
      <tr><td colspan="4" class="py-4 text-center text-gray-500">Немає користувачів</td></tr>
      {% endfor %}
    </tbody>
  </table>
 </div>
</div>
{% endblock %}

<!DOCTYPE html>
<html lang="uk">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
  <title>{% block title %}Авторизація{% endblock %}</title>

  <!-- TailwindCSS + ваша конфігурація -->
  <script src="https://cdn.tailwindcss.com/3.4.16"></script>
  <script>
    tailwind.config = {
      theme: {
        extend: {
          colors: { primary: "#00D6B4", secondary: "#4CAF50" },
          borderRadius: { button: "8px" }
        }
      }
    };
  </script>

  <!-- Шрифти та іконки -->
  <link
    href="https://fonts.googleapis.com/css2?family=Inter&display=swap"
    rel="stylesheet"
  />
  <link
    href="https://cdn.jsdelivr.net/npm/remixicon@4.5.0/fonts/remixicon.css"
    rel="stylesheet"
  />

  {% block head %}{% endblock %}
</head>
<body class="bg-gray-50 min-h-screen flex items-center justify-center font-inter">
  <div class="w-full max-w-md bg-white p-6 rounded-lg shadow-md">
    {% block content %}{% endblock %}
  </div>
  {% block scripts %}{% endblock %}
</body>
```

```
</html>

<!DOCTYPE html>
<html lang="uk">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>{% block title %}Inventory System{% endblock %}</title>

    <!-- TailwindCSS + ваша конфігурація -->
    <script src="https://cdn.tailwindcss.com/3.4.16"></script>
    <script>
      tailwind.config = {
        theme: {
          extend: {
            colors: {
              primary: "#00D6B4",
              secondary: "#4CAF50",
            },
            borderRadius: {
              none: "0px",
              sm: "4px",
              DEFAULT: "8px",
              md: "12px",
              lg: "16px",
              xl: "20px",
              "2xl": "24px",
              "3xl": "32px",
              full: "9999px",
              button: "8px",
            },
          },
        },
```

```
    };
  </script>

  <!-- Шрифти та іконки -->
  <link rel="preconnect" href="https://fonts.googleapis.com" />
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
  <link
    href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap"
    rel="stylesheet"
  />
  <link
    href="https://cdn.jsdelivr.net/npm/remixicon@4.5.0/fonts/remixicon.css"
    rel="stylesheet"
  />

  <!-- Додаткові head-вставки із конкретних сторінок -->
  {% block head %}{% endblock %}
 </head>
 <body class="bg-gray-50 min-h-screen font-inter">
  <div class="flex min-h-screen">

    <!-- Sidebar -->
    <aside class="w-64 bg-white border-r border-gray-200 flex-shrink-0">
      <div class="p-6 bg-blue-600 text-white">
        <h2 class="text-xl font-semibold">Інвентар</h2>
      </div>
      <nav class="mt-4">
        <ul>
          <li>
            <a
              href="{{ url_for('inventory.list_items') }}"
              class="flex items-center px-6 py-3 text-gray-700 hover:bg-gray-100 hover:text-primary
{% if active=='inventory' %}bg-gray-100 text-primary{% endif %}"
            >
```

```
            <i class="ri-stack-line w-5 h-5 mr-3"></i>
            <span>Інвентар</span>
          </a>
        </li>
        <li>
          <a
            href="{{ url_for('sessions.list_sessions') }}"
            class="flex items-center px-6 py-3 text-gray-700 hover:bg-gray-100 hover:text-primary
{% if active=='sessions' %}bg-gray-100 text-primary{% endif %}"
          >
            <i class="ri-calendar-line w-5 h-5 mr-3"></i>
            <span>Сесії</span>
          </a>
        </li>
        <li>
          <a
            href="{{ url_for('reports.report_session_excel', session_id=1) }}"
            class="flex items-center px-6 py-3 text-gray-700 hover:bg-gray-100 hover:text-primary
{% if active=='reports' %}bg-gray-100 text-primary{% endif %}"
          >
            <i class="ri-file-list-3-line w-5 h-5 mr-3"></i>
            <span>Звіти</span>
          </a>
        </li>
        <li>
          <a
            href="{{ url_for('reports.report_audit') }}"
            class="flex items-center px-6 py-3 text-gray-700 hover:bg-gray-100 hover:text-primary
{% if active=='audit' %}bg-gray-100 text-primary{% endif %}"
          >
            <i class="ri-folder-history-line w-5 h-5 mr-3"></i>
            <span>Аудит</span>
          </a>
        </li>
```

```
        {% if current_user.role == 'admin' %}
        <li>
          <a
            href="{{ url_for('admin.list_users') }}"
            class="flex items-center px-6 py-3 text-gray-700 hover:bg-gray-100 hover:text-primary
{% if active=='users' %}bg-gray-100 text-primary{% endif %}"
          >
            <i class="ri-user-settings-line w-5 h-5 mr-3"></i>
            <span>Користувачі</span>
          </a>
        </li>
        {% endif %}
        <li class="mt-6">
          <a
            href="{{ url_for('auth.logout') }}"
            class="flex items-center px-6 py-3 text-gray-700 hover:bg-gray-100 hover:text-primary"
          >
            <i class="ri-logout-box-line w-5 h-5 mr-3"></i>
            <span>Вийти</span>
          </a>
        </li>
      </ul>
    </nav>
  </aside>

  <!-- Main Content -->
  <div class="flex-1 flex flex-col">
    <!-- Header -->
    <header class="bg-white shadow-sm border-b border-gray-200">
      <div class="flex justify-between items-center px-6 py-4">
        <div><!-- тут можна додати breadcrumps або інші елементи --></div>
        <div class="flex items-center">
          <span class="mr-4 text-gray-700">Привіт, {{ current_user.username }}</span>
          <div
```

```
            class="w-10 h-10 rounded-full bg-gray-200 flex items-center justify-center"
          >
            <i class="ri-user-line text-xl text-gray-600"></i>
          </div>
        </div>
      </div>
    </header>

    <!-- Page Content -->
    <main class="p-6 flex-1 overflow-y-auto">
      {% with messages = get_flashed_messages(with_categories=true) %}
        {% for category, msg in messages %}
          <div class="notification mb-4
                {% if category=='success' %}bg-green-100 text-green-800{% endif %}
                {% if category=='danger'  %}bg-red-100   text-red-800{% endif %}
                {% if category=='warning' %}bg-yellow-100 text-yellow-800{% endif %}
                {% if category=='info'    %}bg-blue-100  text-blue-800{% endif %}">
            {{ msg }}
          </div>
        {% endfor %}
      {% endwith %}

      {% block content %}{% endblock %}
    </main>
  </div>
</div>

<!-- Додаткові скрипти із конкретних сторінок -->
{% block scripts %}{% endblock %}
</body>
</html>

{% extends 'base.html' %}
```

```
{% set active = 'inventory' %}
{% block title %}Інвентар{% endblock %}

{% block content %}
<div class="bg-white rounded-lg shadow-sm p-6 mb-6">
  <!-- Заголовок + Додати -->
  <div class="flex justify-between items-center mb-6">
    <h1 class="text-2xl font-semibold text-gray-800">Інвентар</h1>
    <a href="{{ url_for('inventory.new_item') }}"
       class="bg-primary text-white px-4 py-2 rounded-button hover:bg-opacity-90 flex items-center">
      <i class="ri-add-line mr-2"></i>Додати одиницю
    </a>
  </div>

  <!-- Пошук/фільтри -->
  <form method="get" class="flex flex-wrap gap-4 mb-6">
    <div class="relative flex-1 min-w-[240px]">
      <i class="ri-search-line absolute left-3 top-1/2 -translate-y-1/2 text-gray-400"></i>
      <input
        type="text"
        name="search"
        value="{{ request.args.get('search', '') }}"
        placeholder="Пошук за номером, назвою..."
        class="search-input bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-md w-full pl-10 p-2.5"
      />
    </div>
    <div class="min-w-[180px]">
      <select
        name="status"
        class="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-md w-full p-2.5">
        <option value="">Всі статуси</option>
```

```
        <option value="available" {% if request.args.get('status')=='available'%}selected{% endif %}>Наявний</option>
        <option value="missing" {% if request.args.get('status')=='missing'%}selected{% endif %}>Відсутній</option>
        <option value="damaged" {% if request.args.get('status')=='damaged'%}selected{% endif %}>Пошкоджений</option>
      </select>
    </div>
  </form>

  <!-- Таблиця -->
  <div class="table-container">
    <table class="min-w-full">
      <thead class="bg-gray-50 border-b border-gray-200">
        <tr>
          <th class="text-sm text-gray-700 px-4 py-2">Інв. №</th>
          <th class="text-sm text-gray-700 px-4 py-2">Назва</th>
          <th class="text-sm text-gray-700 px-4 py-2">Статус</th>
          <th class="text-sm text-gray-700 px-4 py-2">Оновлено</th>
          <th class="text-sm text-gray-700 px-4 py-2">Дії</th>
        </tr>
      </thead>
      <tbody class="divide-y divide-gray-200">
        {% for item in items %}
        <tr>
          <td class="px-4 py-2 text-blue-600 font-medium">
            <a href="{{ url_for('inventory.view_item', item_id=item.id) }}">
              {{ item.inventory_number }}
            </a>
          </td>
          <td class="px-4 py-2 text-gray-800">{{ item.name }}</td>
          <!-- Статус -->
          <td class="px-4 py-2">
            {% if item.current_status == 'Відмічено' %}
```

```
            <span class="px-2.5 py-1 text-xs font-medium rounded-full bg-green-100 text-green-800">
              Відмічено
            </span>
          {% elif item.current_status == 'Проблема' %}
            <span class="px-2.5 py-1 text-xs font-medium rounded-full bg-yellow-100 text-yellow-
800">
              Проблема
            </span>
          {% else %}
            <span class="text-gray-500">Не перевірено</span>
          {% endif %}
        </td>
        <!-- Дата оновлення статусу -->
        <td class="px-4 py-2 text-sm text-gray-500">
          {{ item.status_updated_at
             and item.status_updated_at.strftime('%d.%m.%Y %H:%M')
             or '—' }}
        </td>
        <td class="px-4 py-2">
          <div class="flex space-x-2">
            <a href="{{ url_for('inventory.edit_item_form', item_id=item.id) }}"
               class="w-8 h-8 flex items-center justify-center rounded-full text-blue-600 hover:bg-blue-
50"
               title="Редагувати">
              <i class="ri-edit-line"></i>
            </a>
            <form method="post"
                  action="{{ url_for('inventory.delete_item_form', item_id=item.id) }}"
                  onsubmit="return confirm('Видалити одиницю?');">
              <button type="submit"
                      class="w-8 h-8 flex items-center justify-center rounded-full text-red-500 hover:bg-
red-50"
                      title="Видалити">
                <i class="ri-delete-bin-line"></i>
```

```
            </button>
          </form>
        </div>
      </td>
    </tr>
    {% else %}
    <tr>
      <td colspan="5" class="py-4 text-center text-gray-500">Нічого не знайдено</td>
    </tr>
    {% endfor %}
  </tbody>
 </table>
</div>
</div>
{% endblock %}

{% extends 'base.html' %}
{% set active = 'inventory' %}
{% block title %}{{ item and 'Редагувати' or 'Додати' }} одиницю{% endblock %}

{% block content %}
<div class="max-w-2xl mx-auto bg-white rounded-lg shadow-sm p-6">
 <h2 class="text-xl font-semibold text-gray-800 mb-6">
  {{ item and 'Редагувати' or 'Додати' }} одиницю
 </h2>
 <form method="post" enctype="multipart/form-data" class="space-y-5">
  <!-- Назва -->
  <div>
   <label class="block text-sm font-medium text-gray-700">Назва</label>
   <input name="name" required
       class="mt-1 block w-full border border-gray-300 rounded-md p-2 focus:ring-primary
focus:border-primary"
       value="{{ item and item.name }}">
  </div>
```

```
    <!-- Серійний № -->
    <div>
      <label class="block text-sm font-medium text-gray-700">Серійний номер</label>
      <input name="serial_number" required
            class="mt-1 block w-full border border-gray-300 rounded-md p-2"
            value="{{ item and item.serial_number }}">
    </div>
    <!-- Інвентарний № -->
    <div>
      <label class="block text-sm font-medium text-gray-700">Інвентарний номер</label>
      <input name="inventory_number" required
            class="mt-1 block w-full border border-gray-300 rounded-md p-2"
            value="{{ item and item.inventory_number }}">
    </div>
    <!-- Відповідальний -->
    <div>
      <label class="block text-sm font-medium text-gray-700">Відповідальна особа</label>
      <input name="responsible"
            class="mt-1 block w-full border border-gray-300 rounded-md p-2"
            value="{{ item and item.responsible }}">
    </div>
    <!-- Примітки -->
    <div>
      <label class="block text-sm font-medium text-gray-700">Примітки</label>
      <textarea name="notes"
              class="mt-1 block w-full border border-gray-300 rounded-md p-2">{{ item and
item.notes }}</textarea>
    </div>
    <!-- Вартість -->
    <div>
      <label class="block text-sm font-medium text-gray-700">Вартість</label>
      <input name="price" type="number" step="0.01"
            class="mt-1 block w-full border border-gray-300 rounded-md p-2"
            value="{{ item and item.price }}">
```

```
    </div>
    <!-- Фото -->
    <div>
      <label class="block text-sm font-medium text-gray-700">Фото одиниці</label>
      <input name="photo" type="file" accept="image/*"
            class="mt-1 block w-full text-sm text-gray-500">
      {% if item and item.photo %}
      <p class="mt-2 text-gray-600">Поточне зображення:</p>
      <img src="/{{ item.photo }}" class="mt-1 w-24 h-24 object-cover rounded-md border"
alt="Фото">
      {% endif %}
    </div>

    <!-- Кнопки -->
    <div class="flex justify-end space-x-3">
      <a href="{{ url_for('inventory.list_items') }}"
        class="px-4 py-2 bg-gray-100 text-gray-700 rounded-button hover:bg-gray-
200">Скасувати</a>
      <button type="submit"
            class="px-4 py-2 bg-primary text-white rounded-button hover:bg-opacity-90">
        {{ item and 'Оновити' or 'Додати' }}
      </button>
    </div>
  </form>
</div>
{% endblock %}

{% extends 'base.html' %}
{% block title %}Інвентар{% endblock %}

{% block content %}
<div class="bg-white rounded-lg shadow-sm p-6 mb-6">
  <div class="flex justify-between items-center mb-6">
    <h1 class="text-2xl font-semibold text-gray-800">Інвентар</h1>
```

```
    <a href="{{ url_for('inventory.add') }}"
       class="bg-primary hover:bg-opacity-90 text-white px-4 py-2 rounded-button flex items-center whitespace-nowrap">
      <i class="ri-add-line mr-2"></i> Додати одиницю
    </a>
  </div>

  <!-- Пошук і фільтрація -->
  <form method="get" class="flex flex-wrap gap-4 mb-6">
    <div class="relative flex-1 min-w-[240px]">
      <i class="ri-search-line absolute left-3 top-1/2 -translate-y-1/2 text-gray-400"></i>
      <input type="text" name="search" value="{{ request.args.get('search', '') }}"
             placeholder="Пошук за номером, назвою..."
             class="search-input bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-md block w-full pl-10 p-2.5">
    </div>
    <div class="min-w-[180px]">
      <select name="status" class="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-md block w-full p-2.5">
        <option value="">Всі статуси</option>
        <option value="active" {% if request.args.get('status') == 'active' %}selected{% endif %}>Активний</option>
        <option value="marked" {% if request.args.get('status') == 'marked' %}selected{% endif %}>Відмічено</option>
        <option value="inactive" {% if request.args.get('status') == 'inactive' %}selected{% endif %}>Неактивний</option>
      </select>
    </div>
  </form>

  <!-- Таблиця -->
  <div class="table-container">
    <table class="min-w-full">
      <thead class="bg-gray-50 border-b border-gray-200">
        <tr>
          <th class="text-sm text-gray-700">Інв. №</th>
```

```
        <th class="text-sm text-gray-700">Назва</th>
        <th class="text-sm text-gray-700">Статус</th>
        <th class="text-sm text-gray-700">Оновлено</th>
        <th class="text-sm text-gray-700">Дії</th>
      </tr>
    </thead>
    <tbody class="divide-y divide-gray-200">
      {% for item in items %}
      <tr>
        <td class="text-sm text-blue-600 font-medium">{{ item.inventory_number }}</td>
        <td class="text-sm text-gray-700">{{ item.name }}</td>
        <td>
          {% if item.status == 'active' %}
            <span class="px-2.5 py-1 text-xs font-medium rounded-full bg-primary bg-opacity-10 text-
primary">Активний</span>
          {% elif item.status == 'marked' %}
            <span class="px-2.5 py-1 text-xs font-medium rounded-full bg-green-100 text-green-
800">Відмічено</span>
          {% else %}
            <span class="px-2.5 py-1 text-xs font-medium rounded-full bg-red-100 text-red-
800">Неактивний</span>
          {% endif %}
        </td>
        <td class="text-sm text-gray-500">{{ item.updated_at.strftime('%d.%m.%Y %H:%M')
}}</td>
        <td>
          <div class="flex space-x-2">
            <a href="{{ url_for('inventory.edit', item_id=item.id) }}"
               class="w-8 h-8 flex items-center justify-center rounded-full text-blue-600 hover:bg-blue-
50" title="Редагувати">
              <i class="ri-edit-line"></i>
            </a>
            <button type="button" data-item-id="{{ item.id }}"
                    class="w-8 h-8 flex items-center justify-center rounded-full text-red-500 hover:bg-
red-50 delete-button"
                    title="Видалити">
```

```
            <i class="ri-delete-bin-line"></i>
          </button>
        </div>
      </td>
    </tr>
    {% else %}
    <tr>
      <td colspan="5" class="text-center text-sm text-gray-500 py-4">Немає результатів</td>
    </tr>
    {% endfor %}
  </tbody>
 </table>
</div>

<!-- Пагінація -->
{% if pagination %}
<div class="flex justify-between items-center mt-6">
 <div class="text-sm text-gray-700">
  Показано <span class="font-medium">{{ pagination.start_index }}</span>–
  <span class="font-medium">{{ pagination.end_index }}</span> з
  <span class="font-medium">{{ pagination.total }}</span> записів
 </div>
 <div class="flex space-x-1">
  {% for page_num in pagination.iter_pages() %}
   {% if page_num %}
    <a href="{{ url_for('inventory.list', page=page_num, **request.args) }}"
       class="pagination-button {% if page_num == pagination.page %}active{% endif %} w-8 h-8 flex items-center justify-center rounded text-gray-700">
      {{ page_num }}
    </a>
   {% else %}
    <span class="w-8 h-8 flex items-center justify-center text-gray-400">…</span>
   {% endif %}
  {% endfor %}
```

```
    </div>
  </div>
  {% endif %}
</div>

<!-- Модалка підтвердження видалення -->
<div id="deleteModal" class="modal">
  <div class="modal-content">
    <h3 class="text-lg font-medium text-gray-900 mb-4">Підтвердження видалення</h3>
    <p class="text-gray-600 mb-6">Ви впевнені, що хочете видалити цей елемент? Ця дія не може бути скасована.</p>
    <form id="deleteForm" method="post" action="">
      <div class="flex justify-end space-x-3">
        <button type="button" id="cancelDelete"
                class="px-4 py-2 bg-gray-100 text-gray-700 rounded-button hover:bg-gray-200">
          Скасувати
        </button>
        <button type="submit"
                class="px-4 py-2 bg-red-500 text-white rounded-button hover:bg-red-600">
          Видалити
        </button>
      </div>
    </form>
  </div>
</div>

<script>
  document.addEventListener("DOMContentLoaded", function () {
    const deleteButtons = document.querySelectorAll(".delete-button");
    const deleteModal = document.getElementById("deleteModal");
    const cancelDelete = document.getElementById("cancelDelete");
    const deleteForm = document.getElementById("deleteForm");

    deleteButtons.forEach(button => {
```

```
    button.addEventListener("click", () => {
      const itemId = button.dataset.itemId;
      deleteForm.action = `/inventory/delete/${itemId}`;
      deleteModal.style.display = "flex";
    });
  });

  cancelDelete.addEventListener("click", () => {
    deleteModal.style.display = "none";
  });

  window.addEventListener("click", (event) => {
    if (event.target === deleteModal) {
      deleteModal.style.display = "none";
    }
  });
 });
</script>
{% endblock %}

{% extends 'auth_base.html' %}

{% block title %}Увійти{% endblock %}

{% block content %}
<h2 class="text-2xl font-semibold text-gray-800 mb-6 text-center">Увійдіть в систему</h2>

<form method="post" class="space-y-5">
 <div>
  <label class="block text-sm font-medium text-gray-700">Username</label>
  <input
   type="text"
   name="username"
```

```
      required
      class="mt-1 block w-full border border-gray-300 rounded-md p-2 focus:ring-primary focus:border-primary"
    />
  </div>
  <div>
    <label class="block text-sm font-medium text-gray-700">Password</label>
    <input
      type="password"
      name="password"
      required
      class="mt-1 block w-full border border-gray-300 rounded-md p-2 focus:ring-primary focus:border-primary"
    />
  </div>
  <button
    type="submit"
    class="w-full flex justify-center py-2 px-4 bg-primary text-white rounded-button hover:bg-opacity-90 transition"
  >
    Увійти
  </button>
</form>
{% endblock %}


{# app/templates/reports_audit.html #}
{% extends 'base.html' %}
{% set active = 'audit' %}
{% block title %}Журнал дій{% endblock %}

{% block content %}
<div class="bg-white rounded-lg shadow-sm p-6">
  <h1 class="text-2xl font-semibold mb-4">Журнал дій користувачів</h1>
  <div class="table-container overflow-x-auto">
```

```
    <table class="min-w-full divide-y divide-gray-200">
      <thead class="bg-gray-50">
        <tr>
          <th class="px-4 py-2 text-left text-xs font-medium text-gray-500 uppercase">Дата / час</th>
          <th class="px-4 py-2 text-left text-xs font-medium text-gray-500
uppercase">Користувач</th>
          <th class="px-4 py-2 text-left text-xs font-medium text-gray-500 uppercase">Дія</th>
          <th class="px-4 py-2 text-left text-xs font-medium text-gray-500 uppercase">Деталі</th>
        </tr>
      </thead>
      <tbody class="bg-white divide-y divide-gray-200">
        {% for log in logs %}
        <tr>
          <td class="px-4 py-3 text-sm text-gray-700">
            {{ log.timestamp.strftime('%d.%m.%Y %H:%M:%S') }}
          </td>
          <td class="px-4 py-3 text-sm text-gray-700">
            {{ log.user.username if log.user else 'User ' ~ log.user_id }}
          </td>
          <td class="px-4 py-3 text-sm text-gray-700">
            {{ log.action }}
          </td>
          <td class="px-4 py-3 text-sm text-gray-600">
            {{ log.details or '—' }}
          </td>
        </tr>
        {% else %}
        <tr>
          <td colspan="4" class="px-4 py-3 text-center text-gray-500">Немає записів</td>
        </tr>
        {% endfor %}
      </tbody>
    </table>
  </div>
```

```
</div>
{% endblock %}


{% extends 'base.html' %}
{% set active = 'sessions' %}
{% block title %}{{ session.name }} – {{ item.inventory_number }}{% endblock %}


{% block content %}
<div class="bg-white rounded-lg shadow-sm p-6 max-w-md mx-auto">
  <h1 class="text-2xl font-semibold text-gray-800 mb-4">
    {{ session.name }} – Одиниця {{ item.inventory_number }}
  </h1>
  <img src="/{{ item.photo or 'static/images/placeholder.png' }}"
       class="w-full h-auto rounded-md mb-4" alt="Фото">
  <form method="post" class="space-y-4">
    <div>
      <label class="block text-sm font-medium text-gray-700 mb-1">Статус</label>
      <select name="status" required
              class="block w-full border border-gray-300 rounded-md p-2 focus:ring-primary
focus:border-primary">
        <option value="present"   {% if record and record.status=='present'   %}selected{% endif
%}>Присутній</option>
        <option value="missing"   {% if record and record.status=='missing'   %}selected{% endif
%}>Відсутній</option>
        <option value="damaged"   {% if record and record.status=='damaged'   %}selected{% endif
%}>Пошкоджений</option>
      </select>
    </div>
    <div>
      <label class="block text-sm font-medium text-gray-700 mb-1">Коментар</label>
      <textarea name="comment"
                class="block w-full border border-gray-300 rounded-md p-2 focus:ring-primary
focus:border-primary"
                rows="3">{{ record.comment if record else '' }}</textarea>
    </div>
```

```
    <div class="flex justify-end space-x-3">
      <a href="{{ url_for('sessions.list_sessions') }}"
         class="px-4 py-2 bg-gray-100 text-gray-700 rounded-button hover:bg-gray-
200">Скасувати</a>
      <button type="submit"
              class="px-4 py-2 bg-primary text-white rounded-button hover:bg-opacity-90">
        Зберегти
      </button>
    </div>
  </form>
</div>
{% endblock %}

{% extends 'base.html' %}
{% set active = 'sessions' %}
{% block title %}Сесії{% endblock %}

{% block content %}
<div class="bg-white rounded-lg shadow-sm p-6 mb-6">
  <h1 class="text-2xl font-semibold text-gray-800 mb-4">Сесії інвентаризації</h1>
  <form action="{{ url_for('sessions.create_session') }}" method="POST"
        class="grid grid-cols-1 md:grid-cols-3 gap-4 mb-6">
    <input name="name" required placeholder="Назва сесії"
           class="border border-gray-300 rounded-md p-2 focus:ring-primary focus:border-primary">
    <input name="date" placeholder="YYYY-MM-DD"
           class="border border-gray-300 rounded-md p-2 focus:ring-primary focus:border-primary">
    <button type="submit"
            class="bg-primary text-white px-4 py-2 rounded-button hover:bg-opacity-90">
      Створити
    </button>
  </form>

  <ul class="divide-y divide-gray-200">
    {% for session in sessions %}
```

```
    <li class="py-4 flex justify-between items-center">
      <div>
        <p class="font-medium text-gray-800">{{ session.name }}</p>
        <p class="text-sm text-gray-500">{{ session.date }}</p>
      </div>
      <a href="{{ url_for('sessions.item_status', session_id=session.id, item_id=0) }}"
         class="bg-secondary text-white px-4 py-2 rounded-button hover:bg-opacity-90">
        Перейти
      </a>
    </li>
    {% else %}
    <li class="py-4 text-center text-gray-500">Немає сесій</li>
    {% endfor %}
  </ul>
</div>
{% endblock %}


{% extends 'base.html' %}
{% set active = 'inventory' %}
{% block title %}Одиниця {{ item.inventory_number }}{% endblock %}

{% block content %}
<div class="bg-white rounded-lg shadow-sm p-6">
  <div class="flex justify-between items-center mb-6">
    <h1 class="text-2xl font-semibold text-gray-800">
      Одиниця {{ item.inventory_number }}
    </h1>
    <a href="{{ url_for('inventory.edit_item_form', item_id=item.id) }}"
       class="bg-secondary text-white px-4 py-2 rounded-button hover:bg-opacity-90 flex items-
center">
      <i class="ri-edit-line mr-2"></i>Редагувати
    </a>
  </div>
```

```
 <div class="grid grid-cols-1 md:grid-cols-3 gap-6 mb-6">
  <!-- Фото -->
  <div>
   <img
    src="{{ url_for('static', filename=item.photo) }}"
    class="w-full h-auto rounded-md border"
    alt="Фото одиниці"
   />
  </div>
  <!-- Детальна інформація -->
  <div class="md:col-span-2">
   <dl class="grid grid-cols-2 gap-x-4 gap-y-2 text-sm text-gray-700">
    <div><dt class="font-medium">Назва:</dt><dd>{{ item.name }}</dd></div>
    <div><dt class="font-medium">Сер. номер:</dt><dd>{{ item.serial_number }}</dd></div>
    <div><dt class="font-medium">Відповідальний:</dt><dd>{{ item.responsible or '—'
}}</dd></div>
    <div><dt class="font-medium">Вартість:</dt><dd>{{ item.price or '—' }}</dd></div>
    <div class="col-span-2">
     <dt class="font-medium">Примітки:</dt><dd>{{ item.notes or '—' }}</dd>
    </div>
    <div class="col-span-2">
     <dt class="font-medium">QR-код:</dt>
     <dd>
      <img
       src="{{ url_for('static', filename=item.qr_code) }}"
       class="w-32 h-32"
       alt="QR-код"
      />
     </dd>
    </div>
    <div>
     <dt class="font-medium">Статус:</dt>
     <dd>
      <span class="px-2.5 py-1 text-xs font-medium rounded-full
```

```
          {% if item.current_status=='Відмічено'   %}bg-green-100 text-green-800
          {% elif item.current_status=='Проблема'   %}bg-yellow-100 text-yellow-800
          {% else %}bg-gray-100 text-gray-800{% endif %}">
          {{ item.current_status }}
        </span>
      </dd>
    </div>
    <div>
      <dt class="font-medium">Оновлено:</dt>
      <dd>{{ item.status_updated_at
            and item.status_updated_at.strftime('%d.%m.%Y %H:%M')
            or '—' }}</dd>
    </div>
    {% if item.issue_description %}
    <div class="col-span-2">
      <dt class="font-medium">Опис проблеми:</dt>
      <dd>{{ item.issue_description }}</dd>
    </div>
    {% endif %}
  </dl>
 </div>
</div>

<!-- Кнопки дій та модалки — як було раніше -->
<div class="flex space-x-4">
 <button
  class="bg-primary text-white px-4 py-2 rounded-button hover:bg-opacity-90"
  onclick="document.getElementById('markModal').classList.remove('hidden')">
  <i class="ri-check-line mr-2"></i>Відмітити
 </button>
 <button
  class="bg-yellow-400 text-white px-4 py-2 rounded-button hover:bg-opacity-90"
  onclick="document.getElementById('issueModal').classList.remove('hidden')">
```

```
      <i class="ri-alert-line mr-2"></i>Проблема
    </button>
    <a href="{{ url_for('inventory.list_items') }}"
       class="px-4 py-2 bg-gray-100 text-gray-700 rounded-button hover:bg-gray-200">
      ← Назад
    </a>
  </div>
</div>

{% include 'unit_mark_modal.html' %}
{% include 'unit_issue_modal.html' %}

{% endblock %}
{% block scripts %}
<script>
['markModal','issueModal'].forEach(id=>{
  const modal = document.getElementById(id);
  modal.addEventListener('click', e=>{
    if(e.target===modal) modal.classList.add('hidden');
  });
});
</script>
{% endblock %}

<!-- app/templates/unit_issue_modal.html -->
<div id="issueModal" class="fixed inset-0 hidden bg-black bg-opacity-50 flex items-center justify-center">
  <div class="bg-white rounded-lg p-6 w-full max-w-md">
    <h3 class="text-lg font-medium text-gray-900 mb-4">Повідомити про проблему</h3>
    <form method="post">
      <div class="mb-4">
        <label class="block text-sm font-medium text-gray-700 mb-1">Опис проблеми</label>
        <textarea
          name="issue_description"
```

```
        required
        class="w-full border border-gray-300 rounded-md p-2 focus:ring-primary focus:border-primary"
        rows="4"
       ></textarea>
      </div>
      <div class="flex justify-end space-x-3">
       <button
        type="button"
        class="px-4 py-2 bg-gray-100 text-gray-700 rounded-button hover:bg-gray-200"
        onclick="document.getElementById('issueModal').classList.add('hidden')"
       >
        Скасувати
       </button>
       <button
        type="submit"
        name="action"
        value="report"
        class="px-4 py-2 bg-primary text-white rounded-button hover:bg-opacity-90"
       >
        Надіслати
       </button>
      </div>
    </form>
  </div>
</div>


<!-- app/templates/unit_mark_modal.html -->
<div id="markModal" class="fixed inset-0 hidden bg-black bg-opacity-50 flex items-center justify-center">
  <div class="bg-white rounded-lg p-6 w-full max-w-md">
    <h3 class="text-lg font-medium text-gray-900 mb-4">Відмітити одиницю</h3>
    <p class="text-gray-600 mb-6">
      Ви дійсно хочете відмітити цю одиницю як «Відмічено»?
```

```
    </p>
    <form method="post" class="flex justify-end space-x-3">
      <button
        type="button"
        class="px-4 py-2 bg-gray-100 text-gray-700 rounded-button hover:bg-gray-200"
        onclick="document.getElementById('markModal').classList.add('hidden')"
      >
        Скасувати
      </button>
      <button
        type="submit"
        name="action"
        value="mark"
        class="px-4 py-2 bg-primary text-white rounded-button hover:bg-opacity-90"
      >
        Так
      </button>
    </form>
  </div>
</div>
```

Додаток Д. Ілюстративна частина

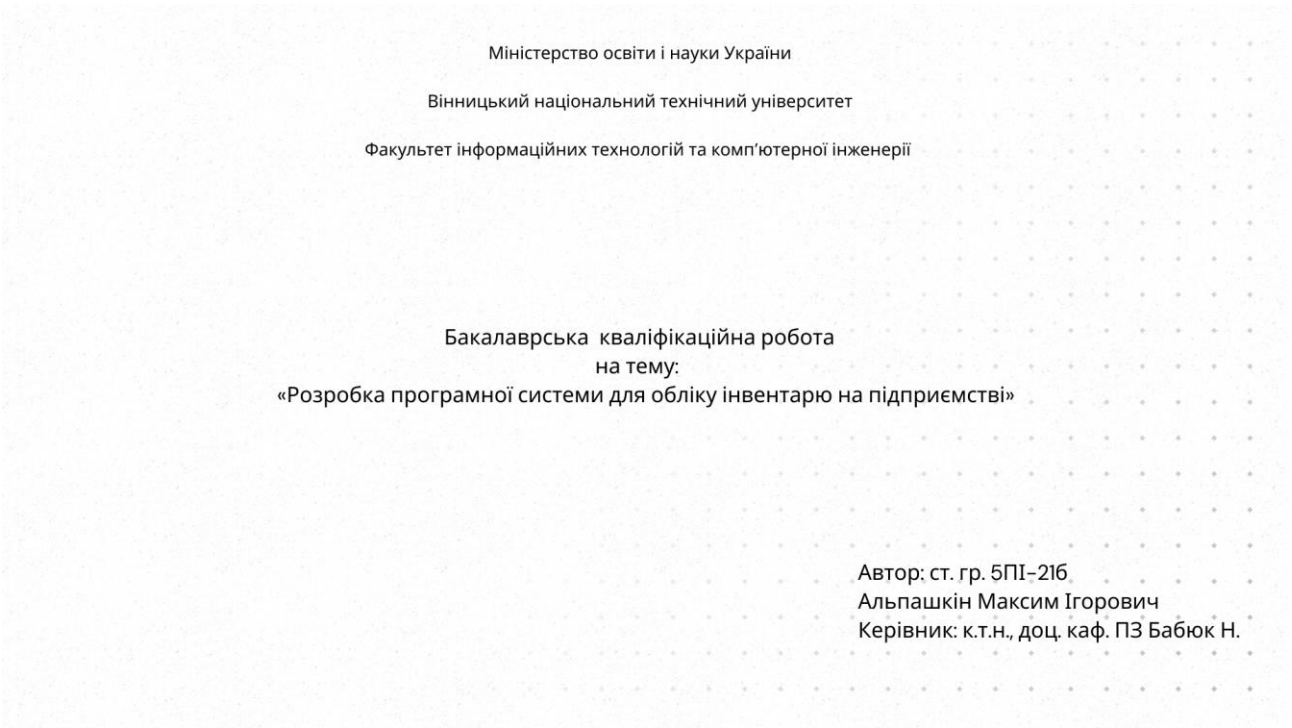


Рисунок 1 – Слайд 1

Рисунок 3 – Слайд 2

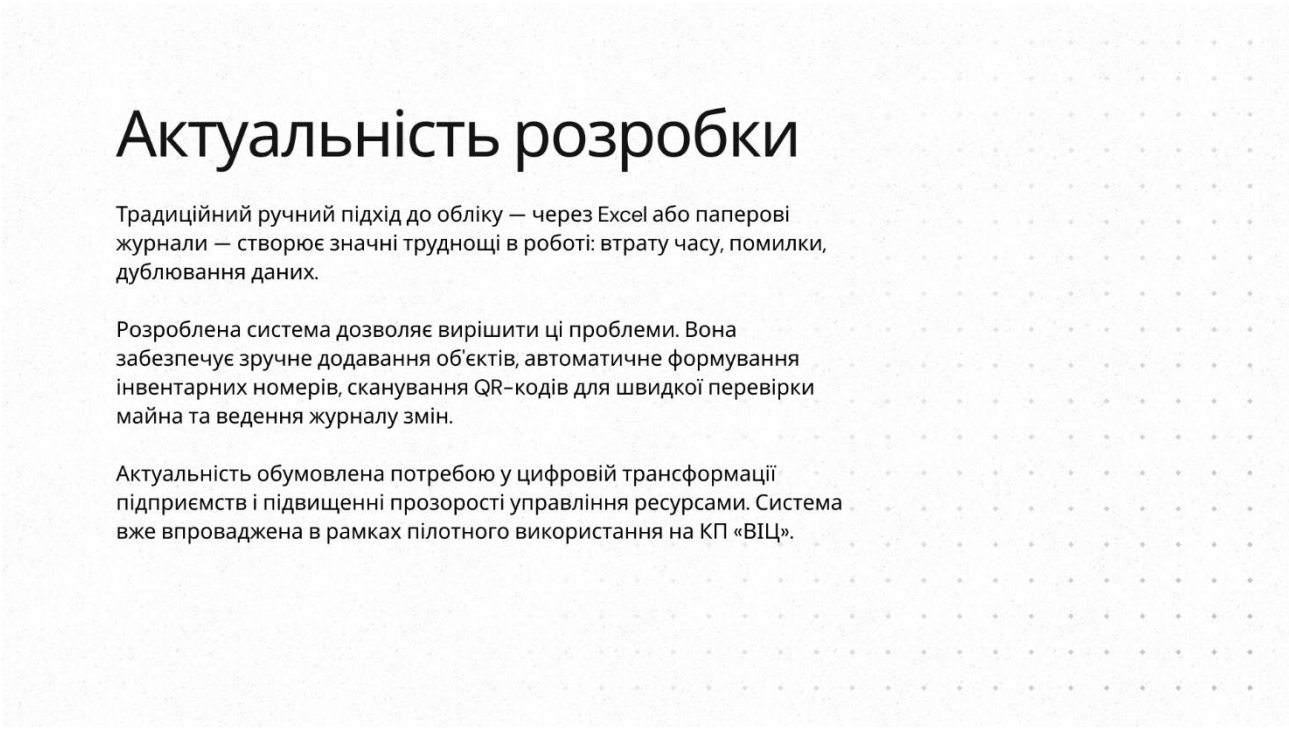


Рисунок 3 – Слайд 3

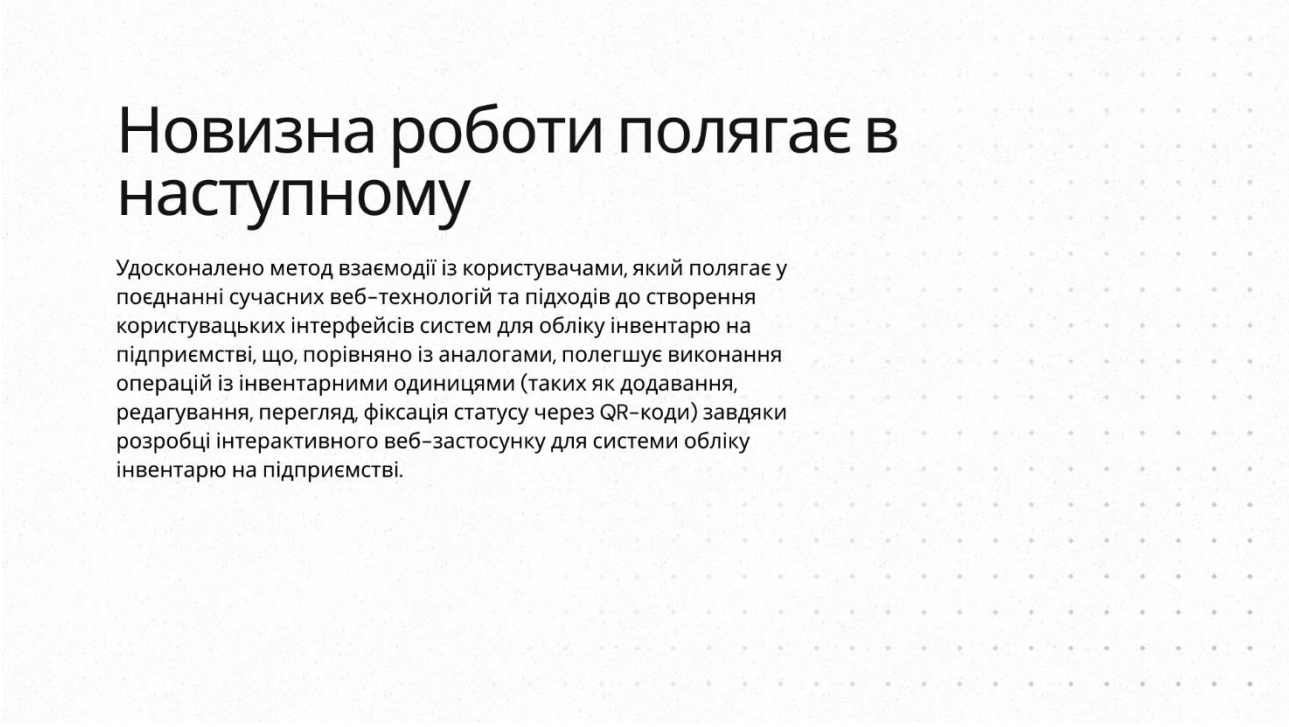


Рисунок 4 – Слайд 4

Порівняльний аналіз аналогів

Було здійснено порівняльний аналіз аналогів, який допоміг встановити актуальність розроблюваної системи та отримати цінні ідеї для розробки

Деякі з них мають суттєві обмеження — або складне впровадження, або обмежений функціонал, або закритість доступу.

Критерії	Dilovod	SystemGroup	Intelpol	Розроблювана система
Вебдоступність	–	–	–	+
Інтеграція в локальну мережу	+	+	+	+
Легкість імплементації (без додаткових девайсів)	–	–	–	+
Висока юзабіліті	–	–	–	+
Генерація QR-коду	–	+	–	+
Можливість введення додаткових даних (ціна, фото, примітка)	–	+	–	+
Адміністративний модуль (менеджмент інвентаризації)	+	+	+	+
Використання інноваційних технологій (RFID)	–	–	+	–
Автоматизація обліку товарів	+	+	+	+
Загальна оцінка	33%	56%	44%	89%

Рисунок 5 – Слайд 5

Вибір засобів реалізації

Python та фреймворк Flask для бекенду

PostgreSQL як основна СУБД

HTML, Tailwind CSS, JavaScript — клієнтська частина

Jinja2 — генерація сторінок

Рисунок 6 – Слайд 6

Діаграма діяльності програмних засобів

Ця діаграма демонструє алгоритм роботи користувача із системою обліку інвентарю, починаючи з авторизації та завершуючи фіксацією змін до інвентарної одиниці. Вона відображає послідовність дій, зокрема: перевірку логіну та пароля, створення нової одиниці з перевіркою унікальності інвентарного номера, генерацію QR-коду, збереження даних у базі, а також подальшу роботу зі сторінкою об'єкта — перегляд, відмітку чи повідомлення про проблему.

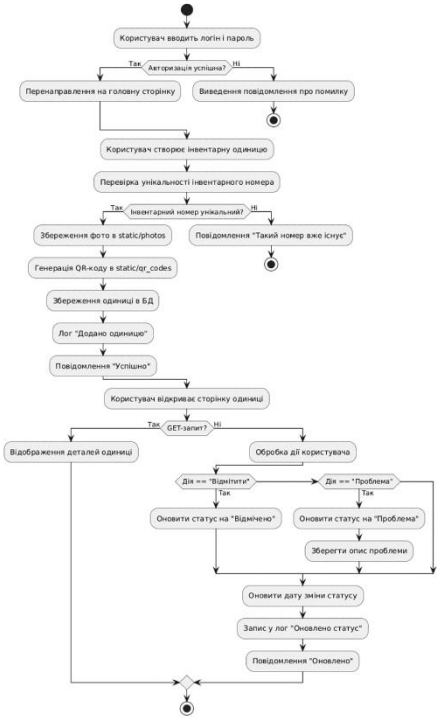
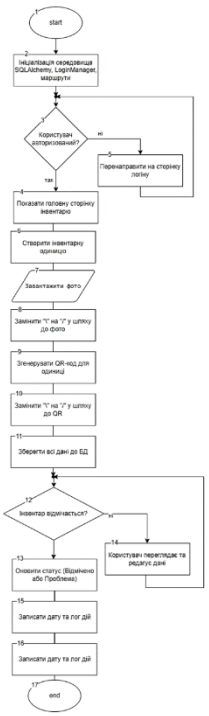


Рисунок 7 – Слайд 7

Алгоритми загальної роботи програмного застосунку та алгоритм редагування інвентарної одиниці



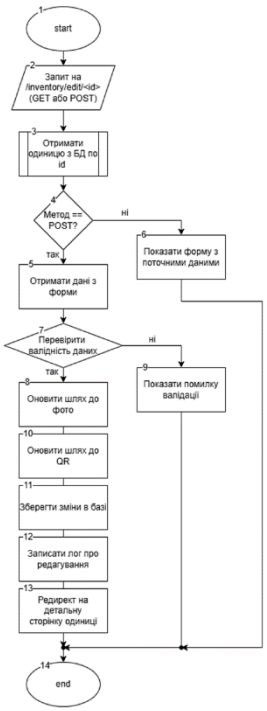


Рисунок 8 – Слайд 8

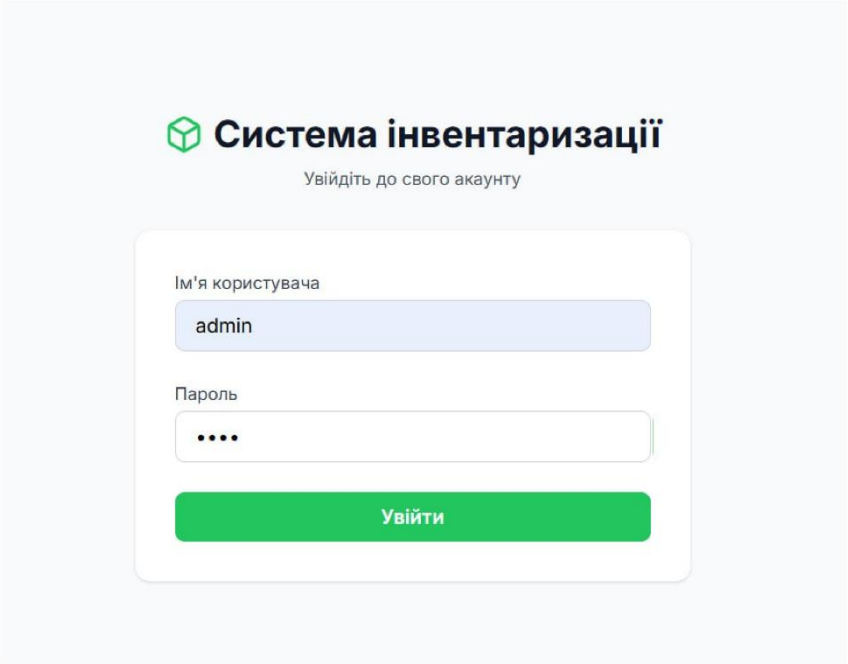


Рисунок 9 – Слайд 9

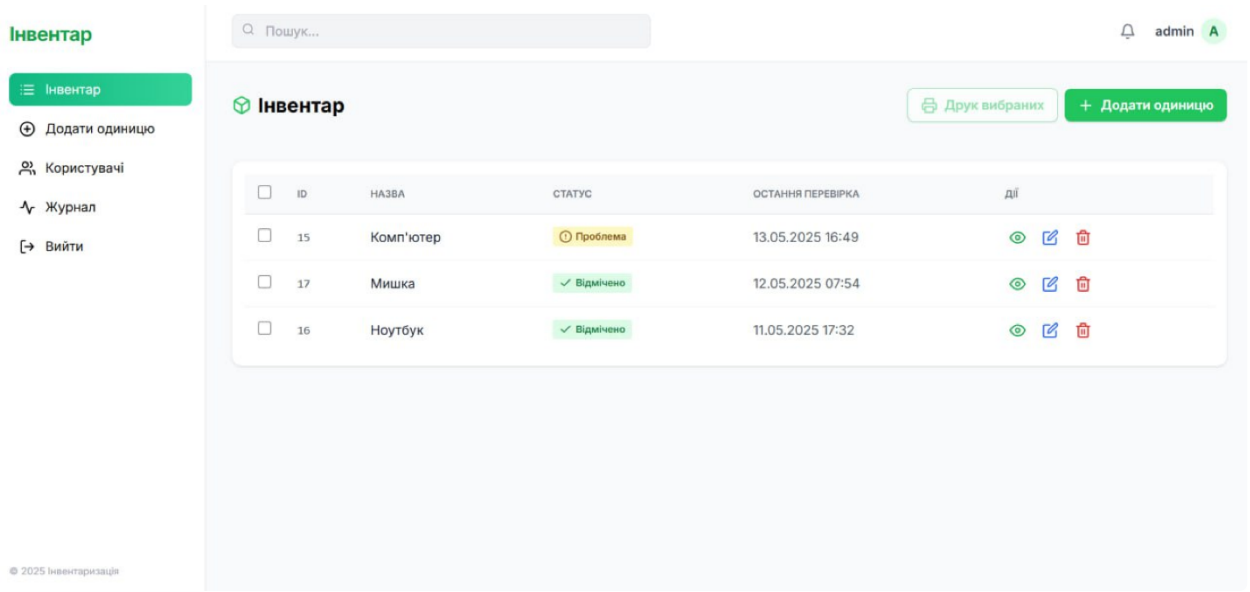


Рисунок 10 – Слайд 10

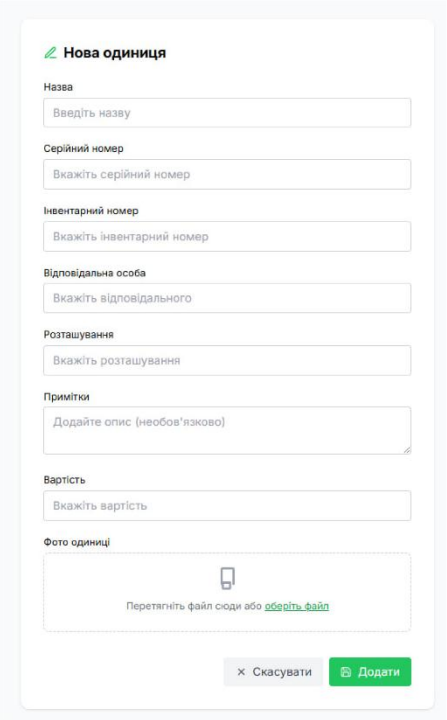


Рисунок 11 – Слайд 11

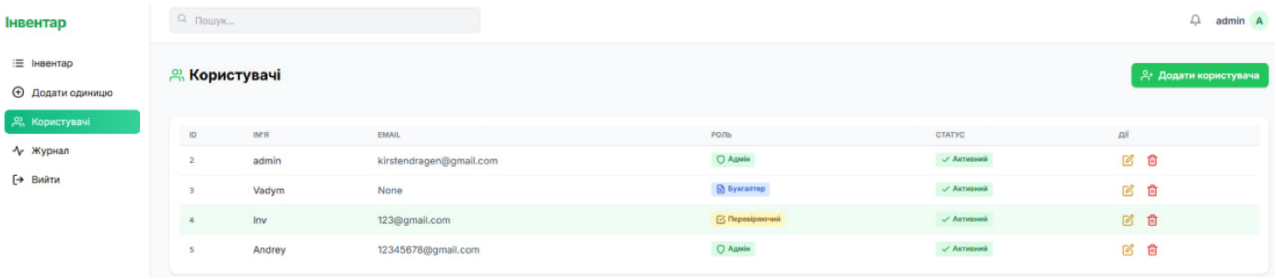


Рисунок 12 – Слайд 12

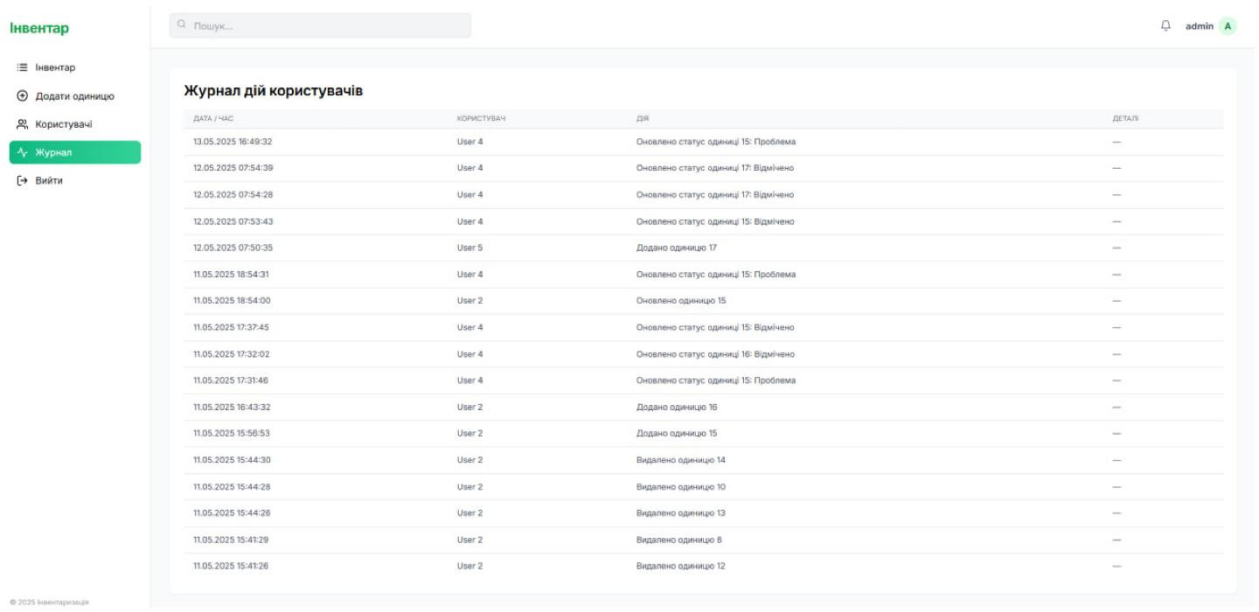


Рисунок 13 – Слайд 13

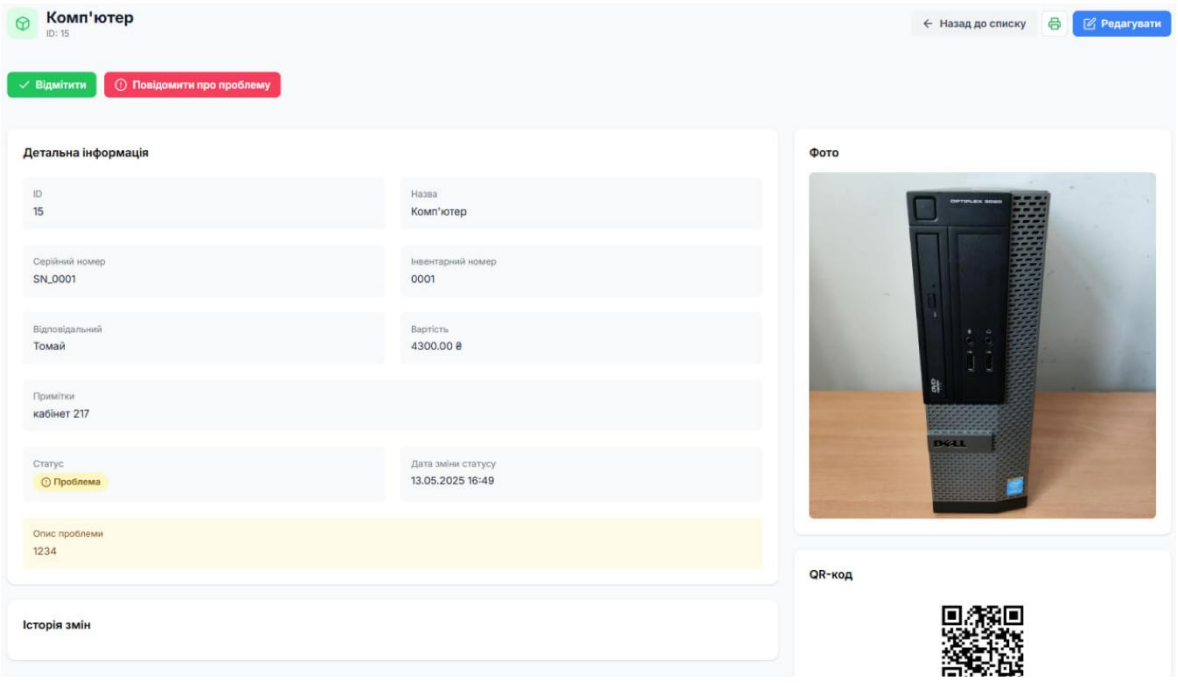


Рисунок 14 – Слайд 14

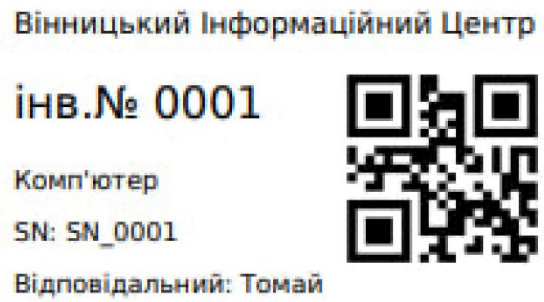


Рисунок 15 – Слайд 15

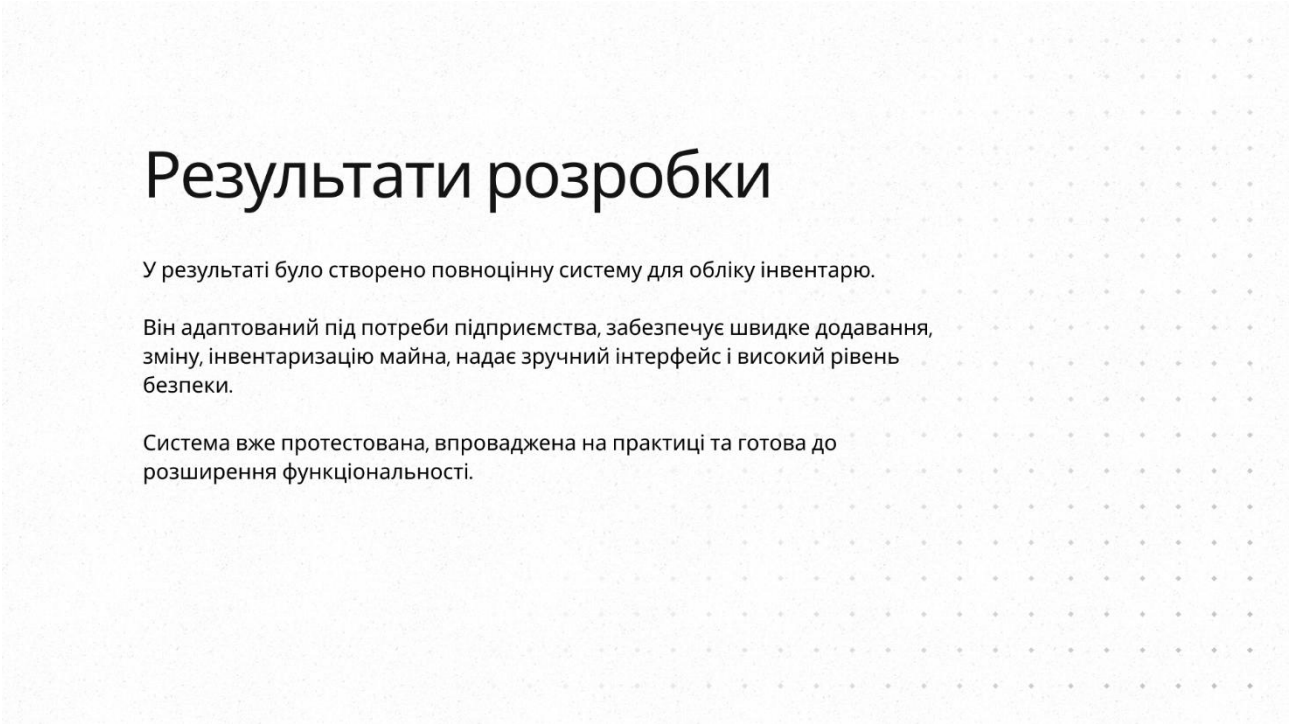


Рисунок 16 – Слайд 16

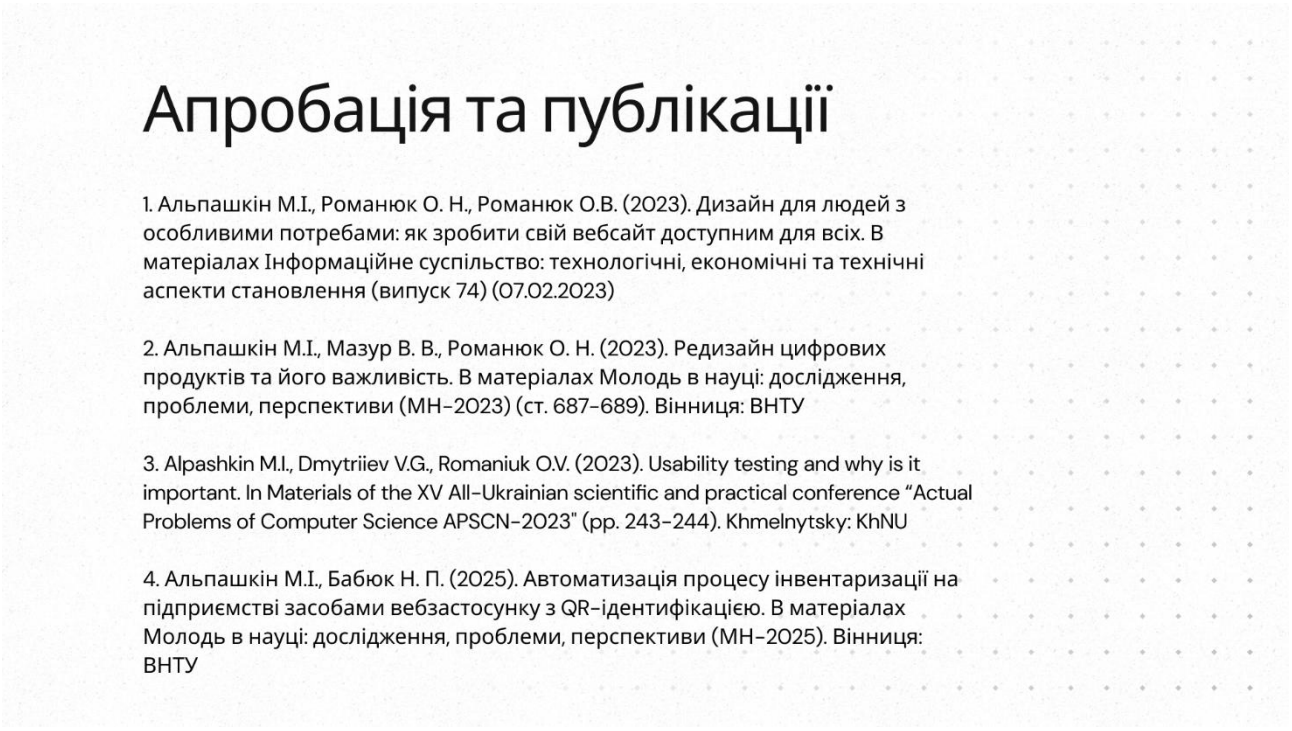


Рисунок 17 – Слайд 17

Рисунок 18 – Слайд 18