Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматики
Кафедра системного аналізу та інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

ІНФОРМАЦІЙНА СИСТЕМА ІНВЕНТАРИЗАЦІЇ ОБЛАДНАННЯ НАВЧАЛЬНИХ ЗАКЛАДІВ

Виконав: студент 4 курсу, групи 2ІСТ-21б
спеціальності 126 – Інформаційні системи та технології
_____ Олег СЕВЕРИН

Керівник: к.т.н., доцент каф. САІТ
_____ Георгій ГОРЯЧЕВ
«27»05 2025 р.

Рецензент: к.ф.-м.н., доцент каф. КН
_____ Олександр ШЕВЧУК
«12»06 2025 р.

Допущено до захисту

Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«06»06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 126 Інформаційні системи та технології
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ к.т.н., проф. Віталій МОКІЙ
«28» 05 2025 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Северину Олегу Сергійовичу

1. Тема роботи: «Інформаційна система інвентаризації обладнання навчальних закладів».
   Керівник роботи: Горячев Г.В., к.т.н., доцент каф. САІТ
   Затверджені наказом ВНТУ від «20» 05 2025 року № 97
2. Термін подання студентом роботи 31 05 2025 року
3. Вихідні дані до роботи:
   Дані щодо організації та проведення процесу інвентаризації обладнання у закладах освіти.
4. Зміст текстової частини:
   1) Аналіз проблематики створення інформаційної системи інвентаризації обладнання навчальних закладів
   2) Проєктування інформаційної системи інвентаризації обладнання навчальних закладів;
   3) Програмна реалізація інформаційної системи інвентаризації обладнання навчальних закладів.
5. Перелік ілюстративного матеріалу:
   1) Логічна ER-діаграма бази даних інформаційної системи;
   2) Діаграма класів інформаційної системи;
   3) Блок-схема основного алгоритму функціонування інформаційної системи;
   4) Діаграма компонентів архітектури інформаційної системи інвентаризації;

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-3	Горячев Г.В., к.т.н., доцент каф. САІТ		

7. Дата видачі завдання 28.03.25р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Термін виконання		Примітки
		початок	закінчення	
1	Аналіз предметної області	01.04	15.04	вик
2	Порівняльний аналіз існуючих проєктних рішень	15.04	30.04	вик
3	Проєктування інформаційної системи інвентаризації обладнання навчальних закладів	01.05	10.05	вик
4	Програмна реалізація інформаційної системи інвентаризації обладнання навчальних закладів	10.05	20.05	вик
5	Оформлення матеріалів до захисту БКР	20.05	30.05	вик

Студент _____ (підпис) Олег СЕВЕРИН

Керівник роботи _____ (підпис) Георгій ГОРЯЧЕВ

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 75 сторінок формату А4, на яких наведено 29 рисунків, 6 таблиць, список використаних джерел містить 20 найменувань.

Метою роботи є розроблення інформаційної системи інвентаризації обладнання навчальних закладів. У підсумку реалізовано веборієнтовану систему на основі Flask (Python) та SQLite, що дозволяє здійснювати додавання, редагування, перегляд, облік та інвентаризацію обладнання. Система охоплює 8 ключових функціональних можливостей, що на 37,5% більше порівняно з найближчим аналогом.

Проведено аналіз предметної області, розроблено архітектуру бази даних і функціональні модулі з урахуванням специфіки навчальних закладів. Результати тестування підтвердили коректність роботи системи та відповідність заданим функціональним вимогам.

Ключові слова: інформаційна система, інвентаризація, база даних, обладнання, навчальний заклад.

ABSTRACT

The bachelor's thesis consists of 75 A4 pages, including 29 figures, 6 tables, and a list of 20 references.

The objective of the thesis is to enhance the functionality of the equipment inventory process in educational institutions by developing an information system. As a result, a web-based system was implemented using Flask (Python) and SQLite, which enables adding, editing, viewing, tracking, and inventorying equipment. The system incorporates 8 key functional features, which is 37.5% more compared to the closest existing analog.

The domain area was analyzed, and the database architecture and functional modules were developed with regard to the specific needs of educational institutions. Testing results confirmed the correct operation of the system and its compliance with the specified functional requirements.

Keywords: information system, inventory, database, equipment, educational institution.

ЗМІСТ

ВСТУП

Актуальність досліджень. У сучасних умовах цифрової трансформації закладів освіти особливої важливості набуває ефективне управління матеріальними ресурсами, зокрема обладнанням, яке використовується для забезпечення навчального процесу. Устаткування навчальних кабінетів, лабораторій, комп'ютерних класів та інших приміщень потребує постійного обліку, технічного обслуговування та своєчасного оновлення. Проте в багатьох навчальних закладах ці процеси досі здійснюються вручну або за допомогою застарілих інструментів, що не відповідають сучасним вимогам точності, швидкості та зручності.

Актуальність теми дослідження зумовлена необхідністю автоматизації процесу інвентаризації для підвищення прозорості управлінських рішень, запобігання втратам матеріальних ресурсів, зменшення навантаження на персонал і забезпечення надійного зберігання даних про обладнання. Інформаційна система, розроблена з урахуванням потреб конкретного освітнього закладу, дозволяє в режимі реального часу отримувати повну інформацію про наявне обладнання, його місцезнаходження, стан, відповідальних осіб, історію змін тощо.

Таким чином, створення інформаційної системи інвентаризації обладнання для навчальних закладів є важливим кроком у напрямі цифровізації управлінських процесів та забезпечення ефективного функціонування освітньої інфраструктури.

Метою дослідження є розроблення інформаційної системи інвентаризації обладнання навчальних закладів.

Для досягнення поставленої мети необхідно розв'язати такі задачі:

- провести аналіз проблематики проведення інвентаризації обладнання навчальних закладів;

- здійснити порівняльний аналіз існуючих інформаційних технологій;

- спроєктувати інформаційну систему інвентаризації обладнання навчальних закладів;

- реалізувати інформаційну систему інвентаризації обладнання навчальних закладів;

- провести тестування функціональності розробленої інформаційної системи.

Об’єктом дослідження є процес розробки інформаційної системи інвентаризації обладнання навчальних закладів.

Предметом дослідження є методи та технології розробки інформаційної системи інвентаризації обладнання навчальних закладів.

1 АНАЛІЗ ПРОБЛЕМАТИКИ СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНВЕНТАРИЗАЦІЇ ОБЛАДНАННЯ НАВЧАЛЬНИХ ЗАКЛАДІВ

1.1 Аналіз предметної області

Матеріально-технічне забезпечення є важливою складовою функціонування будь-якого навчального закладу – від шкіл і професійно-технічних училищ до закладів вищої освіти. Наявність, доступність та справність обладнання безпосередньо впливають на якість освітнього процесу, ефективність адміністративного управління, а також дотримання вимог безпеки та стандартів освіти.

Інвентаризація обладнання – це процес обліку, перевірки наявності, технічного стану, юридичного та фактичного статусу матеріальних активів, що належать навчальному закладу. Така інвентаризація проводиться згідно з внутрішніми регламентами закладу, а також відповідно до законодавчих актів та інструкцій, зокрема: положення про бухгалтерський облік в державних установах; наказів Міністерства освіти і науки України щодо звітності та матеріального забезпечення; нормативних вимог щодо проведення щорічної інвентаризації основних засобів, затвердженими наказами Міністерства фінансів [1, 2].

В межах освітньої установи інвентаризації підлягають різноманітні категорії обладнання: комп’ютерна техніка: стаціонарні комп’ютери, ноутбуки, монітори, принтери, сканери, мережеве обладнання; навчальне обладнання: інтерактивні дошки, проектори, лабораторні стенди, мікроскопи, хімічне та фізичне обладнання; меблі: столи, стільці, шафи, полиці, офісні меблі; побутове обладнання: холодильники, мікрохвильові печі, кондиціонери; інше устаткування: сейфи, аудіосистеми, засоби охорони, спортивне спорядження [3].

Кожна одиниця такого обладнання має набір атрибутів: інвентарний номер, найменування, серійний номер, тип, дата придбання, розташування,

відповідальний працівник, технічний стан, дата останньої інвентаризації, статус (використовується, списано, в ремонті, втрачено).

Разом з цим варто відзначити, що у багатьох навчальних закладах інвентаризація досі здійснюється вручну або за допомогою табличних редакторів. Це, у свою чергу, породжує низку проблем:

- людський фактор: помилки при введенні даних, дублювання записів, неповні або неактуальні відомості;

- відсутність централізованої бази даних: неможливо швидко отримати повну картину наявного обладнання;

- складність оновлення інформації: труднощі з фіксацією змін у розташуванні чи технічному стані;

- неможливість аналітики: відсутні інструменти для аналізу зносу, ремонту, оптимізації закупівель або заміни обладнання;

- велике навантаження на працівників: інвентаризація перетворюється на тривалий та трудомісткий процес.

У зв’язку з вищезазначеними проблемами постає потреба у створенні автоматизованої інформаційної системи інвентаризації, яка дозволить не лише створити централізовану базу обладнання а й надасть можливість зберігати повну історію змін щодо кожної одиниці майна; автоматично формувати інвентарні звіти; вести облік ремонту та списання; здійснювати фільтрацію та пошук обладнання за різними критеріями; призначати відповідальних осіб за збереження обладнання та забезпечити контроль доступу через систему ролей (наприклад, «адміністратор», «інвентаризатор», «користувач») [4].

Зазвичай, основними суб'єктами інвентаризаційного процесу є:

- адміністратор системи, який відповідає за створення облікових записів, налаштування доступу, редагування довідників типів обладнання та локацій;

- інвентаризатор (матеріально відповідальна особа), що вносить нові записи, оновлює інформацію, проводить інвентаризацію;

- користувач – переглядає список обладнання у межах своєї компетенції, залишає запити про несправності;

- керівництво закладу – використовує систему для перегляду звітів, статистики та прийняття управлінських рішень [5].

Враховуючи тенденції до цифровізації державних процесів та розвитку електронного документообігу, розробка інформаційної системи інвентаризації обладнання є не лише доцільною, а й необхідною. Вона, насамперед, дозволить підвищити прозорість та ефективність управління майном; зменшити витрати часу та людських ресурсів; мінімізувати фінансові втрати через неефективне використання або втрату обладнання та покращити обслуговування обладнання завдяки вчасному ремонту або заміні [6, 7].

1.2 Порівняльний аналіз існуючих проєктних рішень

Проаналізувавши вітчизняні програмні продукти, орієнтовані на автоматизацію процесів обліку та інвентаризації основних засобів, можна зробити висновок, що на ринку України вже існують рішення, які частково або повністю охоплюють функціональність електронного обліку майна. Серед таких програмних рішень варто відзначити «МІА: Електронна інвентаризація» [8], систему «M.E.Doc» [9], а також функціональний модуль «Облік основних засобів» у складі корпоративної ERP-платформи IT-Enterprise [10].

Проведення детального аналізу цих систем дозволить оцінити їхні технічні можливості, функціональні переваги та недоліки в контексті автоматизованої інвентаризації обладнання. Це, у свою чергу, надасть змогу сформувати обґрунтовані вимоги до створення нової інформаційної системи, яка буде мати розширений функціонал та краще адаптованою до специфіки роботи навчальних закладів, з урахуванням обмежених ресурсів, багаторівневої відповідальності та потреб у спрощеному адмініструванні.

Платформа «МІА: Електронна інвентаризація» була розроблена Міністерством внутрішніх справ України спільно з Міністерством цифрової трансформації в межах ініціативи цифровізації державних процесів [8]. Цей сервіс надає можливість обліку державного майна у режимі реального часу, що значно

підвищує прозорість та контроль за основними засобами, які належать державним установам. На рисунку 1.1 наведено скріншот головної сторінки сайту «MIA: Електронна інвентаризація» з описом основних функціональних можливостей.

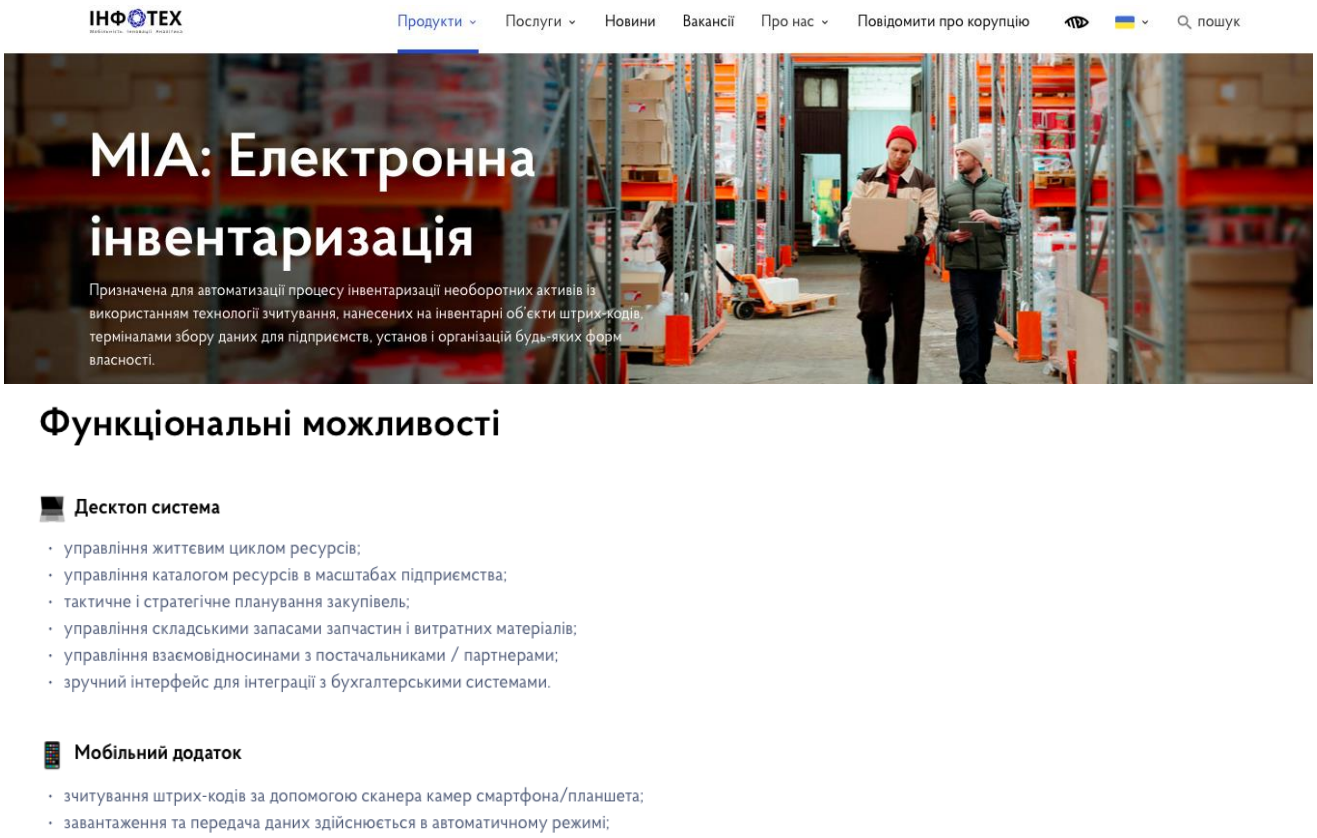

Рисунок 1.1 – Скріншот головної сторінки сайту «MIA: Електронна інвентаризація»

Особливістю даної системи є її централізований характер і тісна інтеграція з іншими державними реєстрами та платформами. Процес інвентаризації реалізовано через використання QR-кодів, що наносяться на об'єкти, а також через мобільний застосунок, який дозволяє працівникам сканувати маркери та передавати дані про стан та місцезнаходження майна до центральної бази. Система дозволяє фіксувати переміщення об'єктів, зміну відповідальних осіб, а також створює цифрові акти інвентаризації, які мають юридичну силу.

Проте дана платформа орієнтована насамперед на державні структури й не має широкої кастомізації під потреби освітніх закладів, особливо тих, які не є частиною підпорядкування МВС або інших центральних органів виконавчої влади.

Крім того, вона не завжди дозволяє гнучко адаптуватися до специфіки інвентаризації різних типів обладнання, що використовуються у навчальному процесі.

Програмний комплекс M.E.Doc відомий як система електронного документообігу та звітності, однак має окремі модулі, що підтримують облік основних засобів, включно з інвентаризаційними процедурами [9]. На рисунку 1.2 наведено скріншот головної сторінки сайту даного програмного комплексу.

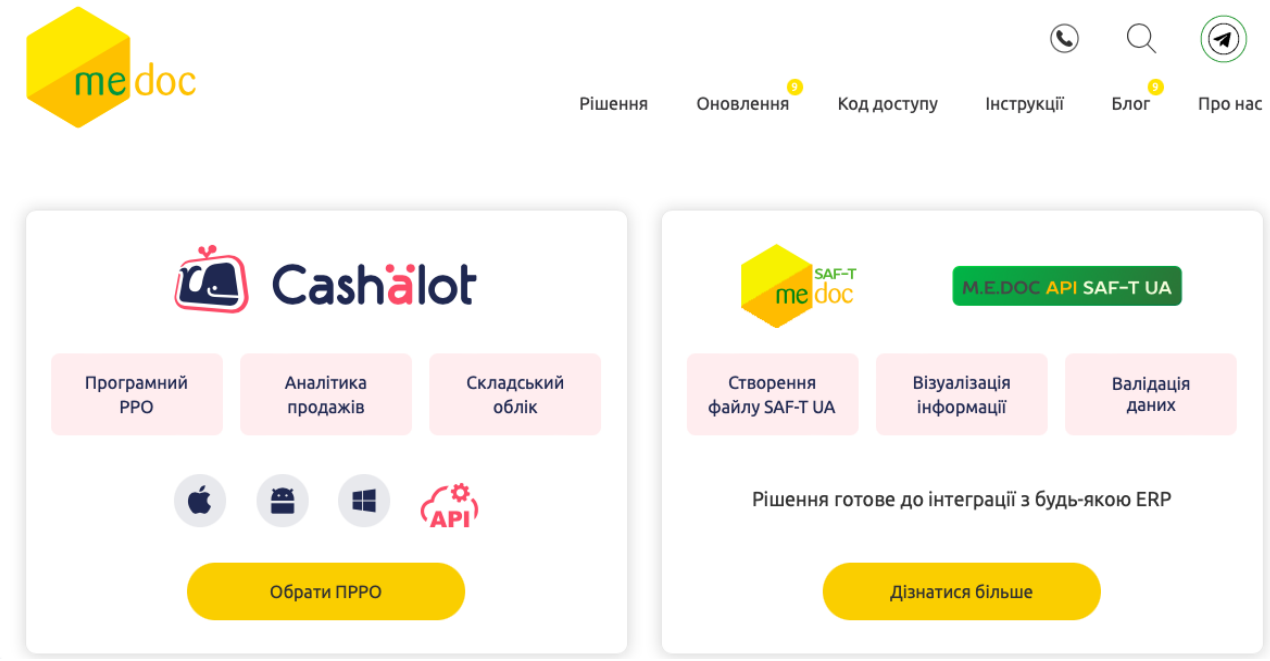

Рисунок 1.2 – Скріншот головної сторінки сайту програмного комплексу M.E.Doc

Основна увага в M.E.Doc приділяється документальній точності та відповідності вимогам податкового законодавства України, що робить систему зручною для бухгалтерських служб. У модулі інвентаризації передбачено можливість формування інвентаризаційних описів, актів виявлення нестач чи надлишків, а також протоколів результатів перевірки.

Всі операції супроводжуються відповідними бухгалтерськими проводками та звітністю. Проте варто зазначити, що система не надає широких можливостей для мобільної інвентаризації із застосуванням сканування QR-кодів або штрих-кодів, а

також не має зручного візуального інтерфейсу для оперативного внесення змін під час фізичного обстеження об'єктів.

Система орієнтована на професіоналів у сфері фінансового обліку, що може ускладнити її впровадження у навчальних закладах, де облік інвентарю часто здійснюється технічним персоналом без глибоких знань бухгалтерії. Крім того, доступ до M.E.Doc потребує окремої ліцензії та встановлення специфічного програмного забезпечення, що може створювати додаткові фінансові та організаційні труднощі для шкіл чи університетів.

Система IT-Enterprise з модулем «Облік основних засобів» є частиною великої ERP-платформи (рис. 1.3), що забезпечує комплексну автоматизацію управлінських та виробничих процесів на підприємствах різного профілю [10].

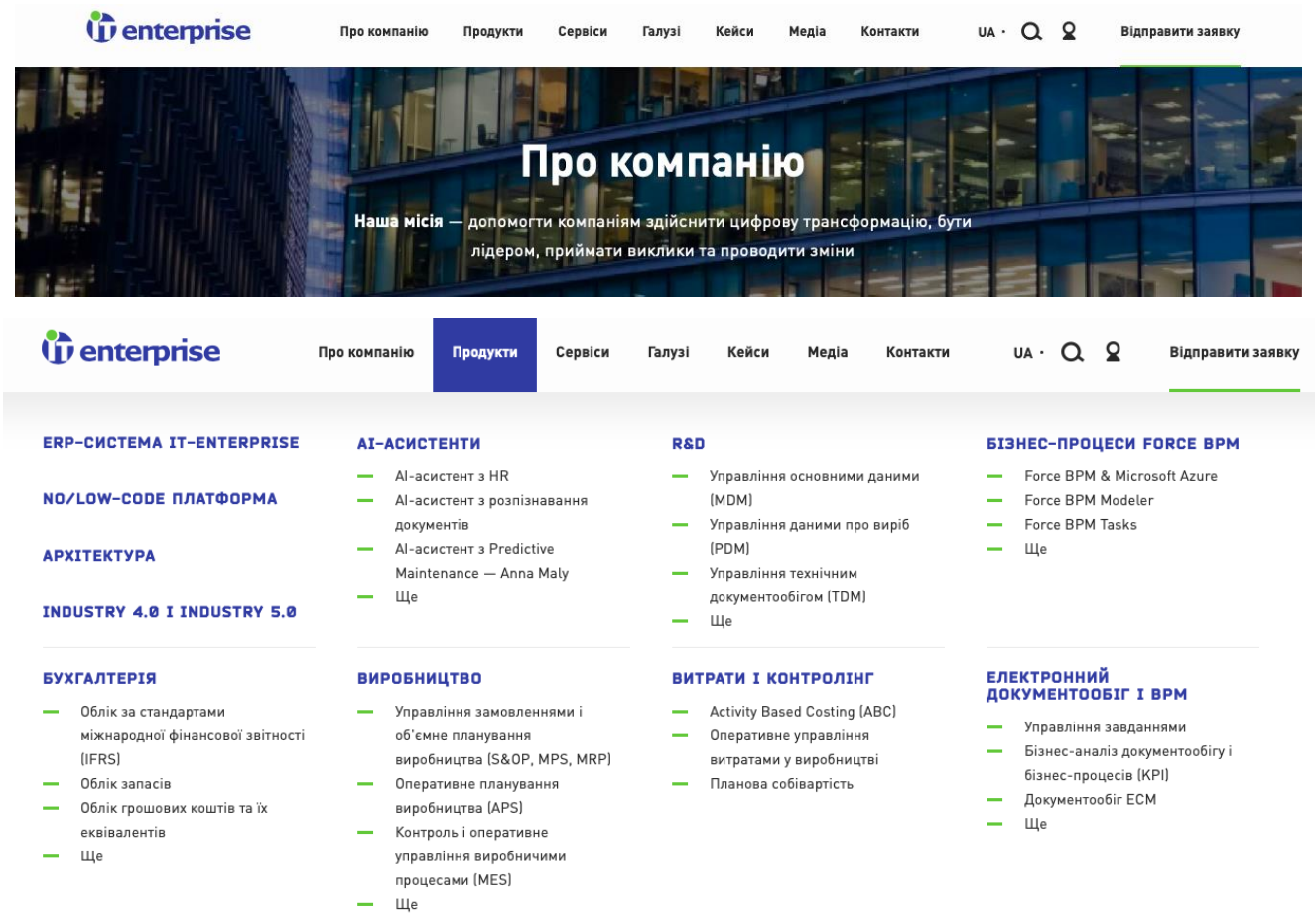

Рисунок 1.3 – Скріншот головної сторінки сайту інформаційної системи IT-Enterprise з переліком пропонованих програмних продуктів

Модуль обліку основних засобів у цій системі дозволяє проводити як первинний облік, так і автоматизовану інвентаризацію з урахуванням фактичного місцезнаходження, стану та змін вартості об'єктів. Система підтримує інтеграцію з мобільними пристроями, включно з функціями сканування ідентифікаторів, а також формування актів інвентаризації відповідно до встановлених стандартів.

Унікальною особливістю є можливість автоматичного оновлення даних про переміщення основних засобів, що особливо корисно для динамічних структур з великою кількістю обладнання.

Проте загальний функціонал системи є надзвичайно обширним і може бути надмірним для освітніх закладів, де інвентаризація має обмежене коло завдань. Також слід зазначити складність налаштування платформи, потребу в навчанні персоналу та значні витрати на впровадження, що обмежує її практичне використання у бюджетних установах без підтримки відповідного IT-відділу.

Таким чином, на основі проведеного аналізу можна відзначити, що кожна із розглянутих систем має свої переваги і недоліки. А саме, державна система МІА забезпечує високу прозорість та інтеграцію з реєстрами, однак є обмеженою у кастомізації. M.E.Doc підходить для точного документального обліку, але не враховує зручність польової інвентаризації. IT-Enterprise пропонує найширший функціонал, проте його складність і вартість роблять систему не завжди доцільною для потреб освітніх закладів.

Отже наведені фактори вказують на необхідність створення удосконаленої інформаційної системи інвентаризації обладнання навчального закладу, яка б мала розширений та адаптований під конкретну практичну область функціонал мобільного збору даних, забезпечувала зручність візуалізації та доступність використання при збереженні відповідності обліковим вимогам.

З метою визначення ключових напрямків удосконалення у таблиці 1.1 наведено порівняльну характеристику вітчизняних програмних продуктів інвентаризаційного обліку.

Таблиця 1.1 – Порівняння основних характеристик вітчизняних інформаційних систем автоматизованої інвентаризації

Критерій	MIA: Електронна інвентаризація	M.E.Doc	IT-Enterprise (Облік ОЗ)
Цільова аудиторія	Державні установи	Комерційні структури	Великі підприємства, організації
Інтеграція з державними реєстрами	Повна інтеграція	Обмежена	Можлива через додаткові модулі
Мобільна інвентаризація	Так, через QR-коди та мобільний застосунок	Ні	Так, з використанням мобільних пристроїв
Гнучкість налаштувань	Обмежена, централізована система	Низька, адаптована під фінансову звітність	Висока, але складна у налаштуванні
Підтримка візуального обліку	Частково, через додаток	Немає	Так, з можливістю інтеграції з графікою
Формування інвентаризаційних актів	Так, автоматично	Так, відповідно до бухгалтерського обліку	Так, з повною підтримкою стандартів
Простота впровадження	Висока для держустанов, низька для інших	Середня, вимагає підготовки бухгалтера	Низька, потребує IT-фахівців
Вартість впровадження	Безкоштовно для держустанов	Платна ліцензія	Висока
Адаптація під освітні заклади	Частково можлива	Малопридатна	Надлишкова складність
Підтримка маркування (QR/штрих-код)	Так	Ні	Так

Наведені у таблиці 1.1 результати порівнянь основних характеристик дозволяють сформувати можливі напрямки вдосконалення та розширення функціональних можливостей процесу проведення автоматизованої інвентаризації для навчальних закладів:

1. Простота та інтуїтивність інтерфейсу. Існуючі рішення, особливо IT-Enterprise і M.E.Doc, орієнтовані на користувачів зі спеціальною підготовкою. Для освітніх закладів доцільно розробити максимально просту у використанні систему з дружнім інтерфейсом, яка не потребуватиме тривалого навчання персоналу.

2. Гнучке налаштування структури обліку. На відміну від централізованих або бухгалтерських систем, нова система має дозволяти самостійно формувати структуру інвентаризації (корпус → аудиторія → категорія обладнання → одиниця обліку), додаючи специфічні поля, що враховують освітню специфіку (наприклад, «призначення обладнання» або «пов’язане програмне забезпечення»).

3. Механізми контролю змін і відповідальності. Система повинна передбачати автоматичний облік змін (хто змінив, коли, що саме), а також механізм закріплення відповідальних осіб за конкретним обладнанням чи групами майна. Це дозволить спростити проведення перевірок і зменшити кількість непогоджених переміщень.

4. Автоматизоване формування звітів. Доцільно забезпечити автоматичне створення інвентаризаційних описів, актів нестач, переміщення, списання тощо – відповідно до нормативної бази для бюджетних установ, із можливістю експорту в PDF, Excel.

5. Забезпечення багаторівневої системи доступу. З урахуванням ролей користувачів (адміністратор, інвентаризатор, відповідальний за об'єкт) необхідно реалізувати контроль доступу до різних функцій та частин бази даних. Це особливо актуально в контексті безпеки та точності обліку.

1.3 Постановка задачі

У результаті проведеного аналізу існуючих проєктних рішень для автоматизованої інвентаризації обладнання було виявлено низку недоліків, які обмежують їхнє ефективне застосування у закладах освіти. Серед основних проблем відзначається складність інтеграції із внутрішніми процесами навчального закладу, надмірна функціональна громіздкість або, навпаки, вузька спеціалізація, обмежені можливості гнучкого налаштування під потреби конкретного освітнього середовища.

Отже, з урахуванням проведеного аналізу існуючих рішень, сформульовано загальні вимоги до інформаційної системи інвентаризації, яка має забезпечити

централізований облік обладнання, його структуризацію за категоріями, відстеження розміщення в межах навчальних приміщень та контроль за станом. У системі повинна бути передбачена рольова модель доступу з можливістю розмежування прав між адміністраторами, інвентаризаторами та відповідальними особами. Важливо забезпечити фіксацію історії переміщень обладнання, його списання, а також результати перевірок у межах інвентаризаційних кампаній. З технічного боку система має бути реалізована як веб-додаток із трирівневою архітектурою, де логіка взаємодії з базою даних розміщується на серверному рівні, а інтерфейс користувача функціонує у браузері.

Основною задачею роботи є розробка простої, надійної та зручно керованої інформаційної системи, яка дозволить ефективно обліковувати матеріальні ресурси навчального закладу, автоматизувати процеси інвентаризації та створення аналітичних звітів. Розв’язання цієї задачі передбачає створення реляційної бази даних, розробку відповідного програмного забезпечення з функціоналом для внесення, редагування, перегляду та аналізу інформації.

1.4 Висновки

У першому розділі розглянуто передумови створення інформаційної системи інвентаризації обладнання навчальних закладів. Проведено аналіз актуальних проблем ведення обліку матеріальних ресурсів, серед яких – фрагментованість даних, відсутність прозорої системи контролю, трудомісткість ручної інвентаризації. Проаналізовано наявні вітчизняні програмні рішення, орієнтовані на автоматизацію інвентаризаційних процесів, виявлено їх обмеження щодо адаптивності, гнучкості й інтеграції з реаліями освітніх установ. На основі цього було сформульовано мету, вимоги та основні завдання проєкту, що включають побудову простої у використанні, розширюваної та функціональної системи, здатної забезпечити зберігання, оновлення й аналітичну обробку інформації про обладнання, користувачів та інвентаризаційні сесії.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНВЕНТАРИЗАЦІЇ ОБЛАДНАННЯ НАВЧАЛЬНИХ ЗАКЛАДІВ

2.1 Проєктування бази даних

Одним із ключових етапів створення інформаційної системи інвентаризації обладнання навчальних закладів є проєктування реляційної бази даних, що забезпечує надійне збереження, логічну структуру та коректні взаємозв’язки між усіма об’єктами обліку. База даних повинна бути масштабованою, гнучкою і оптимізованою для ефективної роботи з операціями додавання, редагування, пошуку, групування та фільтрації інформації про обладнання [11].

Для забезпечення повноти функціональності та підтримки специфіки інвентаризації у навчальних закладах у системі передбачено збереження таких основних сутностей:

- користувачі, що включають адміністраторів, інвентаризаторів та відповідальних осіб, які здійснюють різні операції у системі відповідно до своїх ролей;

- обладнання – одиниці інвентаря з унікальними інвентарними номерами, технічними характеристиками, датами придбання, поточним станом та можливістю збереження фотофіксації;

- приміщення, які включають інформацію про корпуси, поверхи і кабінети, де розміщене обладнання;

- категорії обладнання, що дозволяють логічно групувати одиниці інвентарю за типами (наприклад, комп’ютери, проектори, меблі тощо);

- інвентаризаційні сесії – періоди проведення кампаній обліку обладнання із фіксацією результатів перевірок;

- переміщення та списання – записи про зміну місця зберігання обладнання, його передачу або списання зі складу;

- журнал дій, що забезпечує протоколювання активності користувачів для аудиту та контролю.

Для структурування даних і опису взаємозв’язків між сутностями була розроблена інфологічна модель, на основі якої створено логічну ER-діаграму бази даних (рис. 2.1). Ця діаграма відображає таблиці, їх атрибути та типи зв’язків між ними, що в основному представлені у форматах «один до багатьох» та «багато до багатьох». Наприклад, один кабінет може містити багато одиниць обладнання, а один користувач може брати участь у кількох інвентаризаційних сесіях.

У таблиці 2.1 подано узагальнену інформацію про ключові таблиці бази даних, які реалізують основні сутності інформаційної системи інвентаризації. Кожна таблиця містить набір полів, що відповідають атрибутам відповідного об’єкта, та забезпечує зберігання, структуризацію й логічні зв’язки між даними.

Таблиця 2.1 – Ключові таблиці бази даних та їх основні поля

Назва таблиці	**Основні поля**
users	id, name, email, password_hash, role
equipment	id, name, category_id, room_id, status, acquisition_date, qr_code, image_url, responsible_id
rooms	id, name, building, floor
categories	id, name
inventory_sessions	id, name, start_date, end_date, status
inventory_results	id, session_id, equipment_id, actual_status, comments, checked_by
transfers	id, equipment_id, from_room_id, to_room_id, date, initiated_by
write_offs	id, equipment_id, reason, date, confirmed_by
activity_log	id, user_id, action, timestamp, details

Повна ER-діаграма, що візуалізує описані таблиці, їх атрибути та взаємозв’язки, наведена на рисунку 2.1. Вона є основою для розуміння структури бази даних та подальшої розробки системи.

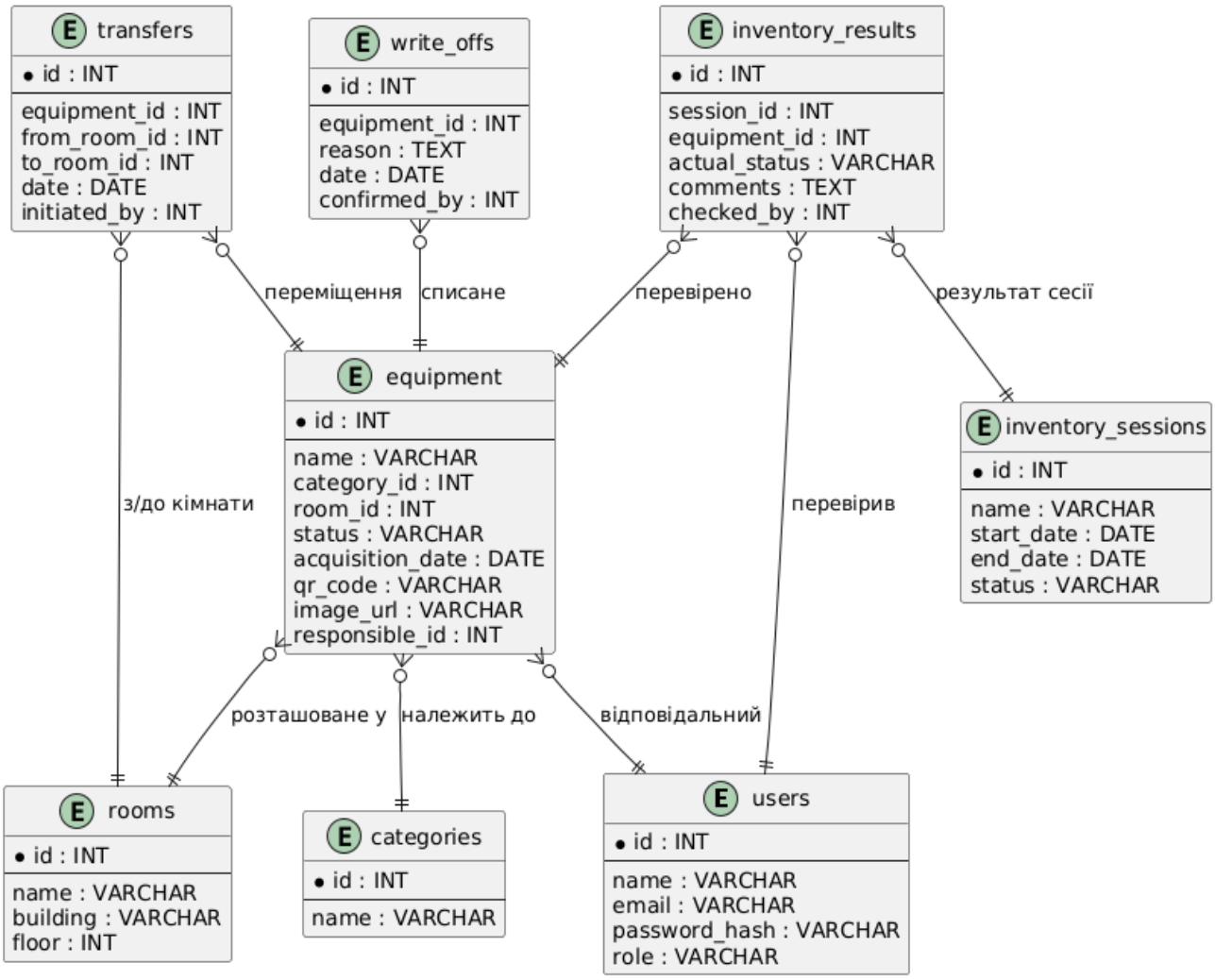

Рисунок 2.1 – Логічна ER-діаграма бази даних інформаційної системи інвентаризації обладнання

ER-діаграма бази даних відображає основні сутності системи та зв’язки між ними, що забезпечують коректне моделювання процесу інвентаризації обладнання.

Сутність users (Користувачі) містить інформацію про користувачів системи, зокрема їх унікальний ідентифікатор (id), ім’я, електронну адресу, захищений пароль та роль (адміністратор, інвентаризатор, відповідальна особа). Роль визначає права доступу користувача до функціональності системи.

Сутність equipment (Обладнання) представляє одиниці інвентарю, кожна з яких має унікальний інвентарний номер (id), назву, належність до категорії (category_id), розташування в певному приміщенні (room_id), статус (наприклад, «використовується», «пошкоджене», «списане»), дату придбання, можливість

зберігати посилання на фотографію, а також відповідального користувача (responsible_id).

Сутність rooms (Приміщення) містить дані про просторову структуру закладу: корпус, поверх і конкретний кабінет, де розміщене обладнання. Відношення між rooms та equipment – «один до багатьох», оскільки в одному приміщенні може знаходитись багато одиниць обладнання.

Сутність categories (Категорії обладнання) забезпечує логічну класифікацію інвентарю за типами (комп'ютери, меблі, проектори тощо). Кожна одиниця обладнання належить до однієї категорії.

Сутність inventory_sessions (Інвентаризаційні сесії) представляє окремі кампанії проведення інвентаризації з інформацією про період їх проведення та статус. Кожна сесія пов'язана з набором результатів перевірок.

Сутність inventory_results (Результати інвентаризації) зберігає фактичний стан обладнання під час конкретної інвентаризаційної сесії, включаючи відмітки про наявність, стан, коментарі та користувача, який здійснював перевірку.

Сутність transfers (Переміщення) відображає історію зміни розташування обладнання між приміщеннями, з фіксацією дати та особи, яка ініціювала переміщення.

Сутність write_offs (Списання) містить інформацію про списання обладнання із зазначенням причини, дати та користувача, який підтвердив списання.

Сутність activity_log (Журнал дій) фіксує всі важливі дії користувачів у системі для забезпечення аудиту та безпеки.

Зв'язки між сутностями підтримують цілісність даних і дозволяють виконувати комплексні запити для отримання аналітичної інформації, наприклад, відстежувати переміщення обладнання, аналізувати стан на різних етапах інвентаризації, контролювати відповідальних осіб.

Нормалізація бази даних проведена до третьої нормальної форми (3NF). Це дозволяє уникнути дублювання даних, мінімізувати надлишковість і підтримувати цілісність інформації, що є критично важливим для достовірності інвентаризації.

Для захисту інформації система передбачає авторизацію користувачів із

використанням безпечного хешування паролів, а також реалізує розмежування прав доступу відповідно до ролей користувачів. Усі дії користувачів фіксуються в журналі активності для забезпечення аудиту.

Запропонована структура бази даних є масштабованою і гнучкою: вона легко адаптується до збільшення кількості обладнання, а також дозволяє підключати нові корпуси або навчальні приміщення. Крім того, вона підтримує формування різноманітних аналітичних звітів за категоріями обладнання, періодами проведення інвентаризацій та відповідальними особами.

На рисунку 2.2 також наведено діаграму класів, що відображає сутності бази даних як класи з атрибутами, які відповідають полям таблиць.

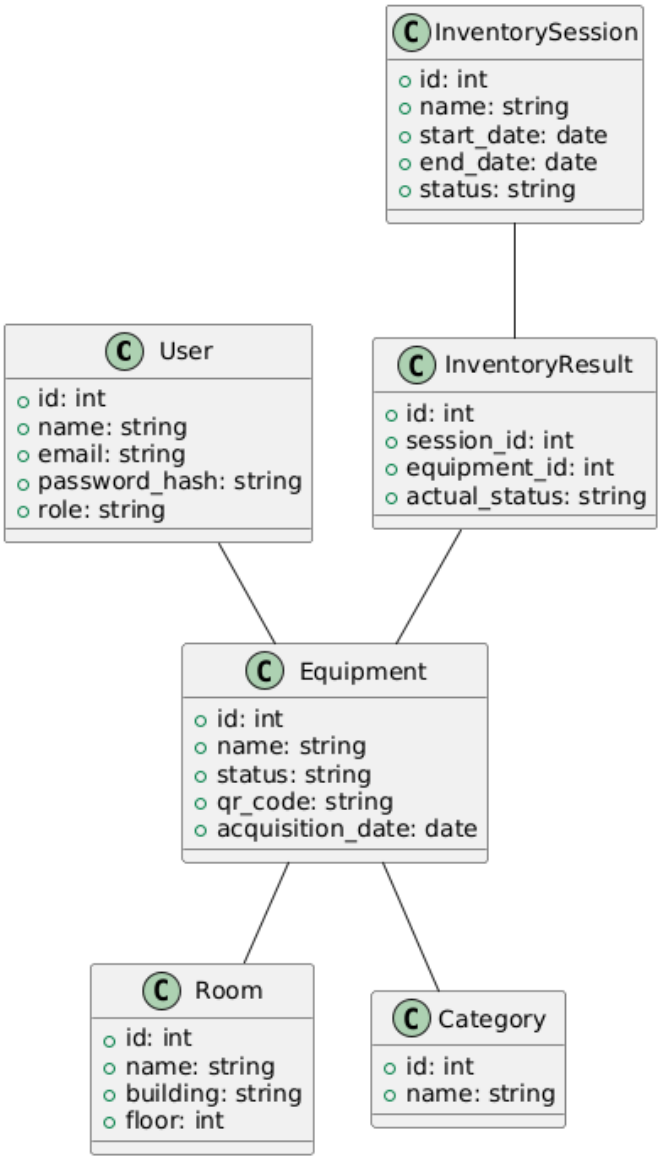

Рисунок 2.2 – Діаграма класів інформаційної системи інвентаризації обладнання

Діаграма класів наведена на рисунку 2.2 відображає сутності бази даних як класи з атрибутами, які відповідають полям таблиць. Зв’язки показані стрілками, що ілюструють типи відносин між класами:

- користувачі можуть створювати інвентаризаційні сесії, виконувати перевірки, ініціювати переміщення і списання, а також генерувати записи в журналі активності;

- обладнання прив’язане до категорій та розміщене у приміщеннях;

- результати інвентаризації пов’язують сесії і обладнання;

- переміщення і списання відносяться до конкретних одиниць обладнання;

Таким чином, розроблена база даних є надійною основою для функціональної інформаційної системи інвентаризації обладнання, що відповідає специфічним вимогам навчальних закладів і забезпечує зручність і ефективність роботи з даними.

2.2 Основний алгоритм функціонування системи

Для забезпечення цілісного уявлення про логіку роботи інформаційної системи інвентаризації обладнання навчальних закладів було розроблено блок-схему основного алгоритму її функціонування (рис. 2.3). Цей алгоритм демонструє загальну послідовність дій користувачів різних ролей у системі, а також внутрішню логіку обробки запитів, взаємодії з базою даних і формування результатів.

Алгоритм побудований з урахуванням трирівневої архітектури: користувач взаємодіє з веб-інтерфейсом, серверна частина обробляє логіку та звертається до бази даних для збереження або отримання інформації.

Основний алгоритм функціонування інформаційної системи інвентаризації починається із запуску веб-додатку, після чого користувач потрапляє на форму авторизації. Система очікує введення облікових даних і перевіряє їхню коректність. Якщо дані вірні, надається доступ до функціоналу, а користувачеві завантажується відповідний інтерфейс відповідно до його ролі у системі.

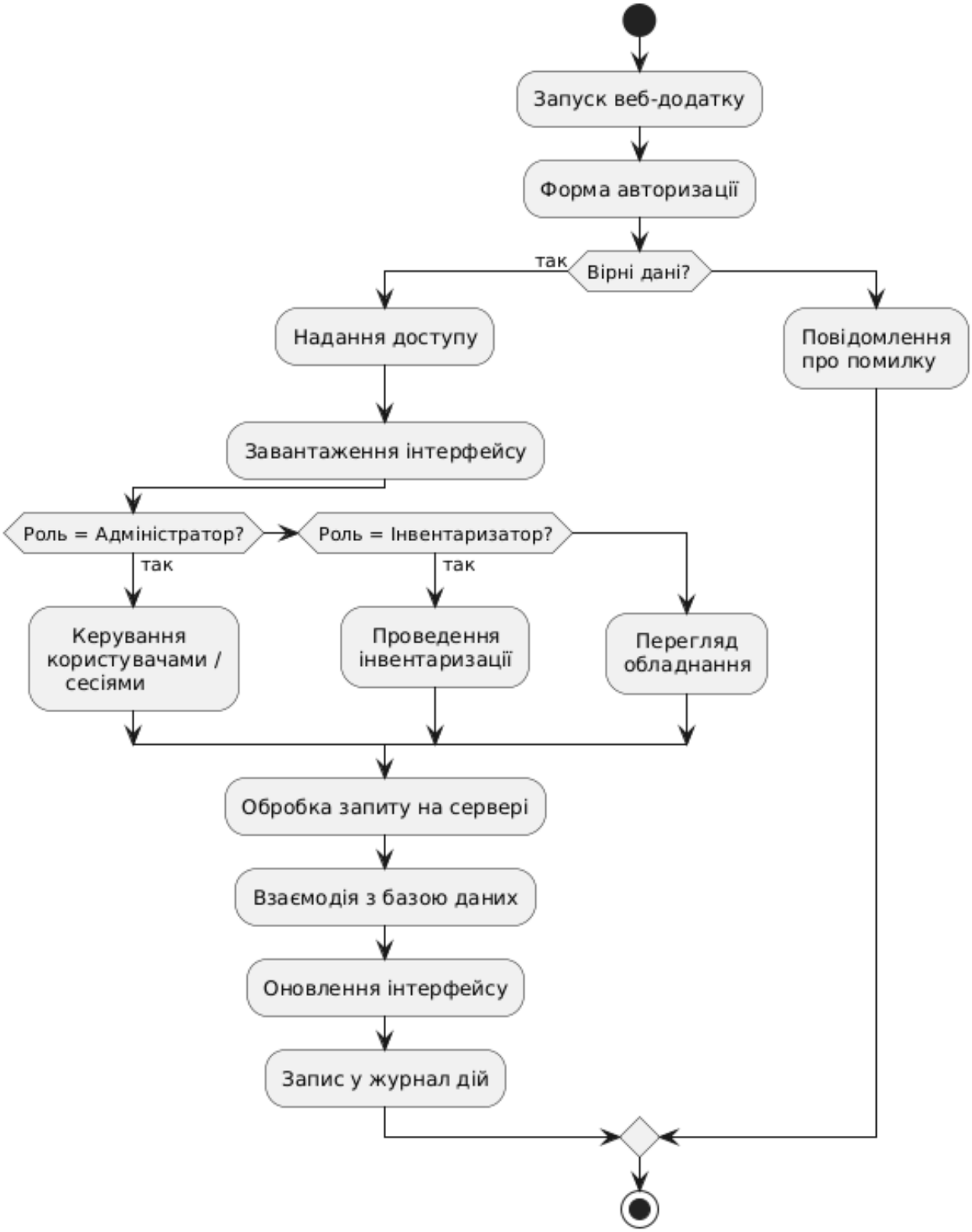

Рисунок 2.3 – Блок-схема основного алгоритму функціонування інформаційної системи інвентаризації обладнання

У разі, якщо користувач є адміністратором, йому відкривається можливість керування обліковими записами та створення або редагування інвентаризаційних

сесій. Якщо авторизується інвентаризатор, він отримує доступ до модулів проведення інвентаризації, включаючи роботу з результатами перевірки обладнання. У випадку, якщо користувач виконує роль відповідальної особи або іншого спостерігача, йому доступна функція перегляду інформації про обладнання.

Після вибору дії користувача, запит обробляється серверною частиною системи. Сервер взаємодіє з локальною базою даних SQLite для отримання або оновлення відповідної інформації. Результати операцій надсилаються назад до клієнта, і інтерфейс оновлюється відповідно до нових даних. Паралельно, усі ключові дії користувачів протоколюються у журналі активності для подальшого контролю та аудиту.

Якщо ж облікові дані виявляються невірними, система повідомляє про помилку авторизації без надання доступу до функціоналу.

2.3 Проєктування архітектури та UML-діаграм функціонування інформаційної системи

Архітектура системи. Інформаційна система інвентаризації обладнання навчальних закладів реалізується у вигляді веб-додатку з трирівневою архітектурою, яка забезпечує чітке розмежування відповідальностей між компонентами, що значно спрощує розробку, підтримку і масштабування системи (рис. 2.4).

Основні рівні архітектури (рис. 2.4):

1. Рівень бази даних. Локальна база даних SQLite, яка є легковаговою, простою в розгортанні та обслуговуванні. SQLite не вимагає налаштувань сервера, що спрощує впровадження системи у навчальних закладах з обмеженими ресурсами.

2. Серверна логіка. Цей рівень відповідає за обробку бізнес-логіки, взаємодію з базою даних, перевірку прав доступу та обробку запитів від клієнтської частини.

3. Клієнтська частина. Забезпечує зручний веб-інтерфейс користувача, через який здійснюється взаємодія з системою. Веб-інтерфейс підтримує роботу з

різними ролями користувачів – адміністраторами, інвентаризаторами та відповідальними особами.

Така архітектура дозволяє ефективно розділити функції системи, підвищити її надійність та гнучкість. Клієнтська частина фокусується на відображенні інформації та прийомі користувацьких дій, серверна логіка обробляє запити і взаємодіє з базою даних, що гарантує чіткий розподіл обов'язків і спрощує підтримку.

Рисунок 2.4 – Діаграма компонентів архітектури інформаційної системи інвентаризації

На рисунку 2.5 також представлено основні сценарії взаємодії користувачів із системою інвентаризації. Діаграма демонструє ролі основних акторів (адміністратора, інвентаризатора та відповідальної особи) та їхню взаємодію з ключовими функціональними можливостями системи, такими як керування користувачами, створення інвентаризаційних сесій, проведення інвентаризації, перегляд звітів і стану обладнання.

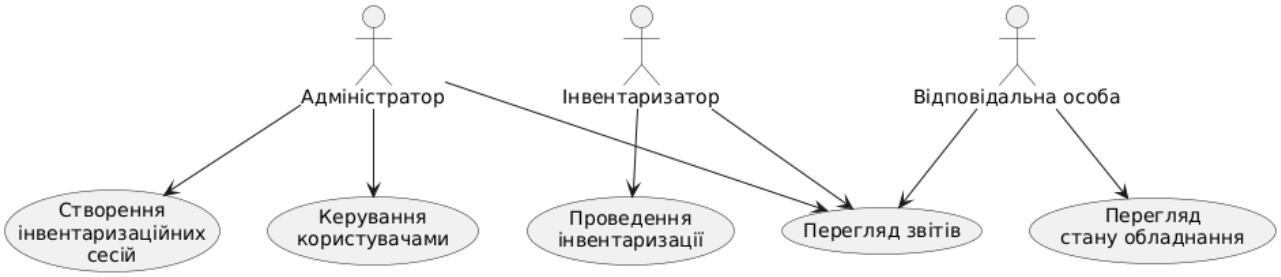

Рисунок 2.5 – Діаграма варіантів використання для інформаційної системи інвентаризації обладнання

Як бачимо з наведеного рисунку 2.5 користувачі взаємодіють з системою через веб-інтерфейс:

1. Адміністратор – має повний доступ до системи: створює/редагує користувачів, налаштовує параметри системи, формує інвентаризаційні сесії, переглядає звіти.

2. Інвентаризатор – проводить безпосередню перевірку наявності та стану обладнання, фіксує результати інвентаризації.

3. Відповідальна особа – контролює обладнання, переглядає стан та звіти по підпорядкованому обладнанню.

У межах проєктування інформаційної системи інвентаризації обладнання важливим аспектом також є моделювання динаміки взаємодії між об'єктами системи під час виконання ключових функціональних сценаріїв. Для цього доцільно використовувати UML-діаграми послідовності, які відображають порядок обміну повідомленнями між об'єктами у часі. Такі діаграми дозволяють краще зрозуміти логіку процесів у системі, визначити послідовність дій, а також забезпечити узгодженість між клієнтською, серверною та базовою логікою.

На рисунках 2.6-2.8 представлено UML-діаграми послідовності для ключових сценаріїв: створення інвентаризаційної сесії адміністратором (рис. 2.6), проведення процесу інвентаризації інвентаризатором (рис. 2.7) та сценарію перегляду звітів відповідальною особою або адміністратором (рис. 2.8).

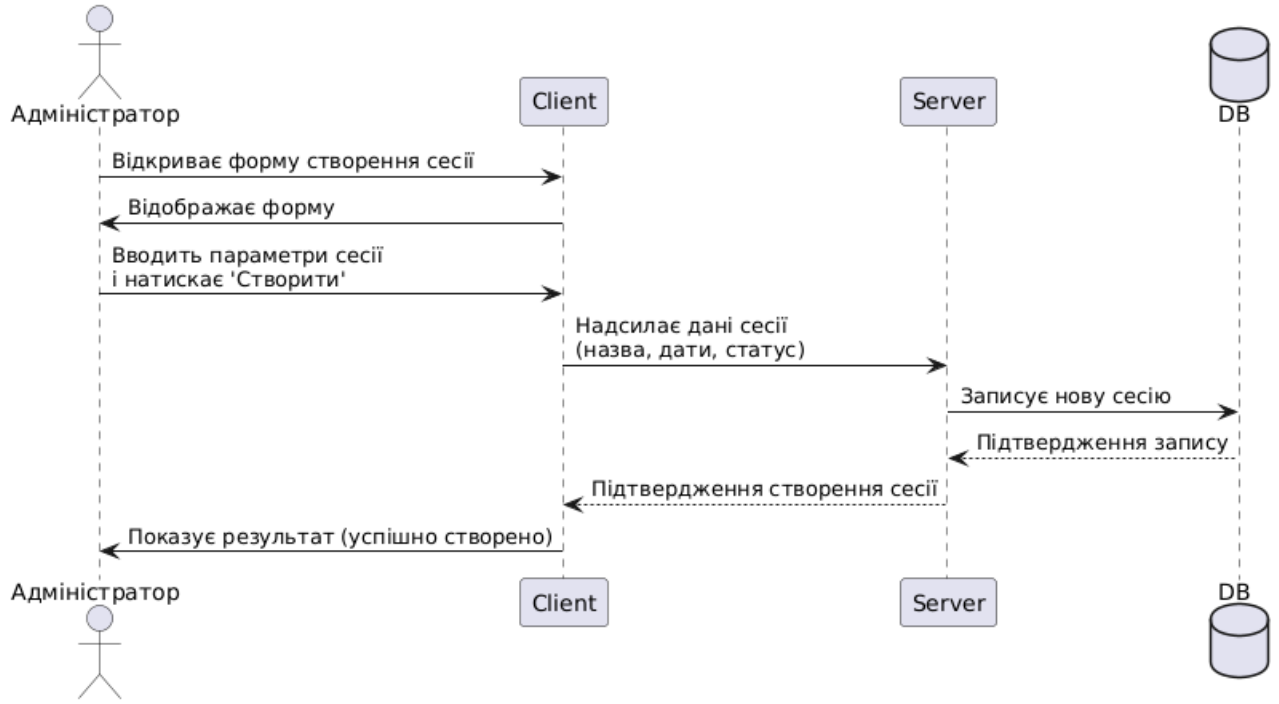

Рисунок 2.6 – Діаграма послідовності для сценарію «Створення інвентаризаційної сесії» (роль: Адміністратор)

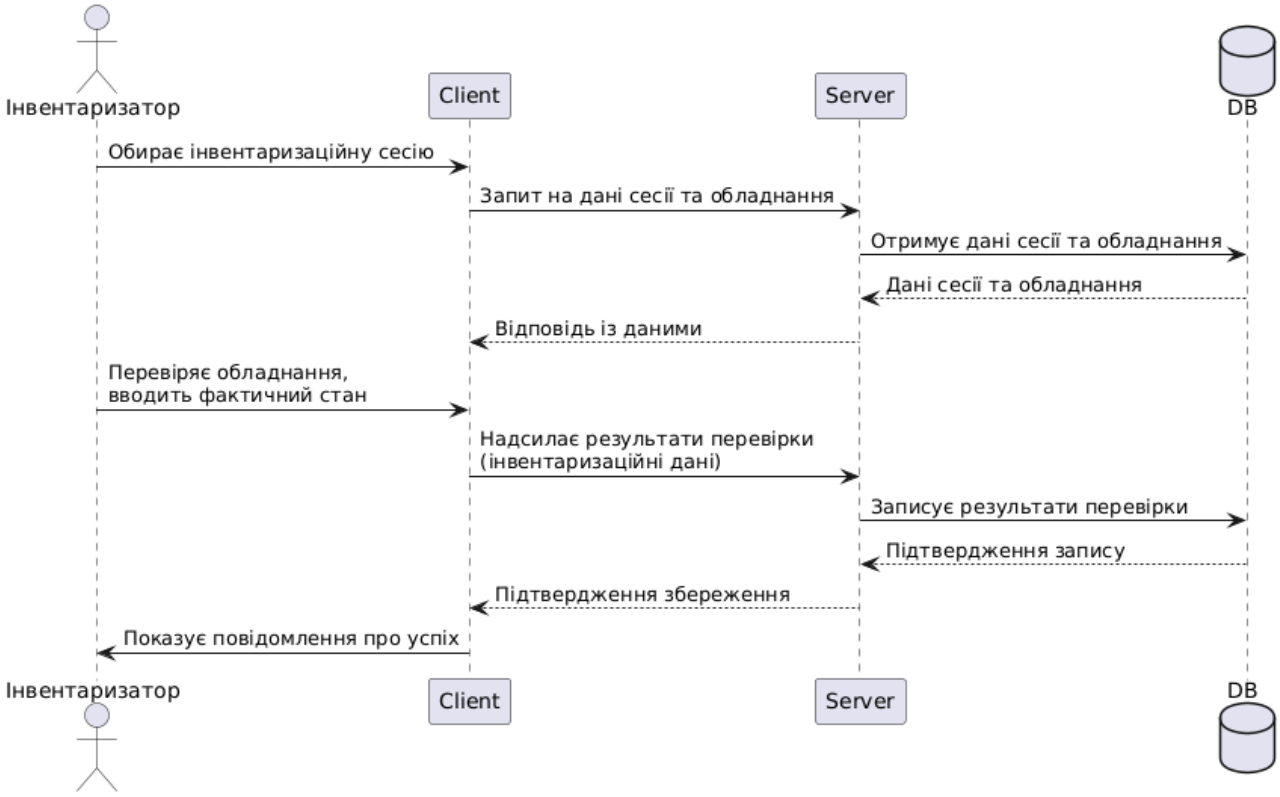

Рисунок 2.7 – Діаграма послідовності для сценарію «Проведення інвентаризації» (роль: Інвентаризатор)

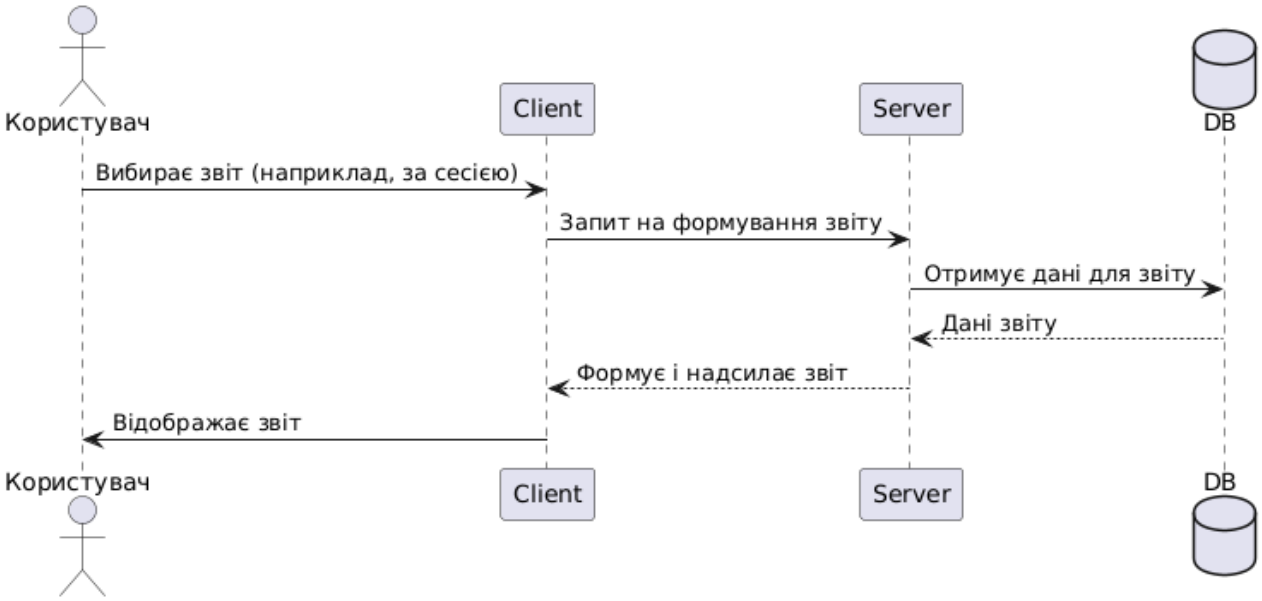

Рисунок 2.8 – Діаграма послідовності для сценарію «Перегляд звітів» (роль: Відповідальна особа)

2.4 Висновки

У другому розділі розроблено концепцію інформаційної системи інвентаризації обладнання навчальних закладів. Сформовано структуру бази даних, визначено основні сутності, їх атрибути та зв’язки між ними, що відображено в логічній ER-діаграмі та діаграмі класів. Запропоновано загальний алгоритм функціонування системи, який описує послідовність дій користувачів залежно від їх ролей. Також спроєктовано архітектуру програмного забезпечення з чітким розмежуванням клієнтської, серверної та базової частин. Використання UML-діаграм дало змогу деталізувати ключові сценарії взаємодії користувачів із системою. Представлений проєкт забезпечує структуровану основу для подальшої реалізації програмного модуля.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНВЕНТАРИЗАЦІЇ ОБЛАДНАННЯ НАВЧАЛЬНИХ ЗАКЛАДІВ

3.1 Обґрунтування вибору інструментів та засобів програмної реалізації

Розробка інформаційної системи інвентаризації обладнання навчальних закладів вимагає вибору інструментів, які забезпечують ефективну, гнучку та зручну реалізацію як серверної, так і клієнтської частини веб-додатку. При виборі засобів розробки було враховано низку критеріїв, зокрема: простота розгортання, швидкість розробки, масштабованість, доступність документації та спільноти підтримки, а також легкість інтеграції між компонентами [12].

Мова програмування для серверної частини. Під час розробки серверної логіки інформаційної системи інвентаризації було проаналізовано кілька популярних мов програмування, які широко використовуються у сфері веб-розробки, зокрема Python, PHP, JavaScript (Node.js) та Java [13]. Кожна з цих мов має свої переваги й недоліки, які слід враховувати з огляду на специфіку проєкту.

Python є мовою високого рівня з акцентом на читабельність коду та зручність розробки. Вона має простий синтаксис, що дозволяє швидко реалізовувати серверну логіку навіть без глибоких знань у сфері веб-програмування. Завдяки популярним фреймворкам Flask та Django, Python надає розробникам потужні інструменти для створення REST API, обробки запитів, авторизації, взаємодії з базами даних та логування. Python також добре підтримується в наукових і освітніх середовищах, що робить його логічним вибором для розробки освітніх або адміністративних систем [14].

PHP – це мова сценаріїв, яка спеціально створена для веб-розробки. Вона використовується у великій кількості готових CMS і підтримується більшістю веб-хостингів. У поєднанні з фреймворками Laravel або Symfony PHP може бути потужним інструментом для реалізації складних проєктів. Однак його синтаксис і принципи проєктування вважаються менш чистими порівняно з Python, а сучасні підходи до розробки іноді потребують додаткової конфігурації або бібліотек.

JavaScript (Node.js). Node.js – це середовище виконання JavaScript на стороні сервера, що дозволяє використовувати одну мову як на клієнті, так і на сервері [15]. Його перевагою є обробка великої кількості одночасних запитів завдяки неблокуючій моделі введення/виводу. Разом із фреймворком Express.js Node.js дозволяє гнучко будувати веб-архітектури, зокрема мікросервіси. Водночас розробка з Node.js вимагає ретельного підходу до структури коду та управління залежностями, що може бути зайвим у проєктах невеликого масштабу.

Java – це об’єктно-орієнтована мова програмування, що традиційно застосовується у великих корпоративних проєктах. Її основними перевагами є висока продуктивність, багатопотоковість і масштабованість. Java підтримує складні веб-додатки за допомогою таких фреймворків, як Spring Boot, які дають повний контроль над життєвим циклом програми. Проте Java має досить складний синтаксис, а її розгортання потребує більших ресурсів порівняно з Python або PHP, що не є оптимальним для легковагового проєкту з локальним розміщенням бази даних.

Порівняльний аналіз розглянутих мов програмування узагальнено представлено в таблиці 3.1.

Таблиця 3.1 – Порівняльний аналіз мов програмування

Критерій	Python	PHP	JavaScript (Node.js)	Java
Простота синтаксису	Висока	Середня	Середня	Низька
Крива навчання	Пологa	Помірна	Помірна	Крута
Швидкість розробки	Висока	Висока	Висока	Середня
Продуктивність	Середня	Середня	Висока	Висока
Підтримка REST API	Відмінна	Добра	Відмінна	Відмінна
Документація та спільнота	Дуже широка	Широка	Широка	Дуже широка
Легкість розгортання	Висока	Висока	Середня	Низька
Ресурси на запуск	Низькі	Низькі	Середні	Високі
Гнучкість у структурі коду	Висока	Середня	Висока	Середня

Отже, на основі вищенаведеного аналізу та даних табл. 3.1, для реалізації серверної частини було обрано мову Python. Це рішення зумовлене її зручним синтаксисом, високою швидкістю розробки, активною спільнотою та широким спектром бібліотек для взаємодії з базами даних, реалізації авторизації та побудови REST API. Python забезпечує зрозумілу архітектуру застосунку, ефективну обробку запитів і можливість розгортання на малопотужних пристроях без значного навантаження на систему, що є важливою перевагою для інформаційної системи локального рівня [14].

У якості фреймворку для серверної частини було обрано Flask – мікрофреймворк, написаний мовою Python, який надає розробнику мінімально необхідний набір інструментів для створення веб-додатків, залишаючи велику свободу у виборі архітектури, організації коду та сторонніх бібліотек [16].

На відміну від повнофункціональних фреймворків, таких як Django, Flask не нав’язує жорсткої структури проєкту, не вимагає дотримання шаблонів «усе включено» та не передбачає складної системи налаштувань. Завдяки цьому, він є особливо зручним у розробці невеликих або середніх за масштабом проєктів, які не потребують розширених компонентів на зразок вбудованої адмін-панелі, ORM за замовчуванням або обов’язкових шаблонів для реалізації авторизації та маршрутизації. Це дає змогу розробнику гнучко адаптувати структуру застосунку відповідно до специфіки проєкту, вибираючи тільки необхідні компоненти.

Для даної інформаційної системи інвентаризації така гнучкість є ключовою. Система складається з чітко визначених модулів – керування обладнанням, користувачами, приміщеннями та сесіями інвентаризації – що можуть розроблятись ізольовано й поступово доповнюватись. Flask ідеально підходить для реалізації подібної модульної архітектури завдяки можливості легко структурувати додаток у вигляді окремих частин логіки, підключати лише необхідні залежності та керувати запитами на рівні маршрутів.

Серед інших важливих переваг Flask можна відзначити швидкість розробки – за рахунок простоти API та низького порогу входу; підтримку великої спільноти – численні приклади, документація, готові розширення та мінімальне споживання

ресурсів, що важливо для локального хостингу з обмеженою продуктивністю.

У порівнянні з Django, який більше підходить для масштабних веб-проєктів з чітко визначеною структурою, системою прав доступу та великою кількістю інтегрованих компонентів, Flask є легшим, менш інвазивним і краще відповідає вимогам гнучкої, розширюваної, але нескладної інформаційної системи локального рівня.

Таким чином, вибір Flask як основи серверної логіки є обґрунтованим і дозволяє забезпечити баланс між функціональністю, масштабованістю та простотою реалізації.

У якості системи керування базами даних (СКБД) для реалізації інформаційної системи інвентаризації було обрано SQLite – легковагову вбудовану реляційну СКБД, що зберігає всі дані у вигляді одного локального файлу. Цей вибір зумовлений кількома технічними та практичними перевагами [17].

По-перше, SQLite не потребує окремого серверного оточення, що суттєво спрощує процес розгортання і супроводу системи. На відміну від класичних СКБД, таких як MySQL або PostgreSQL, SQLite не вимагає встановлення або налаштування серверного ПЗ, що особливо важливо для невеликих локальних або автономних проєктів.

По-друге, SQLite підтримує стандарт SQL-92, що забезпечує зручну побудову запитів, сумісність із типовими інструментами аналізу даних і гнучке управління структурою бази.

По-третє, вона повністю інтегрована у Python через стандартну бібліотеку sqlite3, що дозволяє безпосередньо працювати з базою даних без необхідності встановлення додаткових пакетів чи драйверів. Така інтеграція спрощує розробку серверної логіки, забезпечуючи швидкий доступ до даних з мінімальними затримками.

Крім того, важливими перевагами є висока продуктивність при низькому навантаженні, відсутність зовнішніх залежностей, а також портативність – базу даних можна легко перенести на інший пристрій разом з усіма даними, просто копіюючи файл.

У таблиці 3.2 представлено порівняння SQLite з іншими поширеними СКБД, що підтверджує доцільність вибору.

Таблиця 3.2 – Порівняльний аналіз SQLite з іншими популярними системами керування базами даних

Характеристика	SQLite	MySQL	PostgreSQL
Потреба у сервері	Ні	Так	Так
Формат зберігання	Один файл	Багатофайлова система	Багатофайлова система
Простота налаштування	Дуже проста	Середня	Складніша
Підтримка SQL	SQL-92	SQL-99	SQL-2008
Масштабованість	Низька (для невеликих систем)	Висока	Висока
Вбудована підтримка в Python	Так (sqlite3)	Через mysql-connector	Через psycopg2 або інші
Паралельна робота користувачів	Обмежена	Добра	Відмінна
Продуктивність для невеликих БД	Висока	Середня	Середня
Залежності	Відсутні	Потрібно встановити сервер	Потрібно встановити сервер

Для реалізації клієнтської частини обрано стандартні веб-технології HTML та CSS, що дає змогу створити інтуїтивно зрозумілий і зручний для користувача інтерфейс. HTML забезпечує семантичну структуру веб-сторінок, що сприяє правильному відображенню контенту у різних браузерах і сумісності з різноманітними пристроями. CSS відповідає за візуальне оформлення інтерфейсу, включно з розташуванням елементів, кольоровою палітрою, шрифтами та адаптивністю дизайну, що дозволяє підтримувати приємний та сучасний вигляд системи.

Використання саме цих технологій дозволяє досягти ефективної взаємодії користувача з додатком без необхідності встановлення додаткового програмного забезпечення або плагінів. Це сприяє більш широкому охопленню аудиторії, оскільки веб-додаток є доступним із будь-якого пристрою з сучасним браузером.

Такий підхід є оптимальним для реалізації базових функцій системи: користувачі можуть вводити та редагувати дані через форми, переглядати інформацію про обладнання, інвентаризаційні сесії, отримувати звіти та здійснювати навігацію між різними розділами системи. Водночас відсутність складних клієнтських скриптів знижує ймовірність технічних збоїв і помилок у роботі інтерфейсу, а також забезпечує простоту підтримки та розвитку проекту.

Якщо у подальшому виникне потреба розширення функціоналу, зокрема додавання динамічних елементів або інтерактивності, клієнтську частину можна буде доповнити JavaScript або іншими відповідними технологіями без кардинальної перебудови існуючої структури. Таким чином, обраний підхід є гнучким, масштабованим і відповідає вимогам сучасної веб-розробки.

Обґрунтування вибору середовища розробки.

Для розробки веб-додатку системи інвентаризації обладнання було проведено аналіз популярних середовищ розробки (IDE та редакторів коду), які відповідають вимогам проекту щодо зручності, функціональності та підтримки технологій, що використовуються [18, 19].

Головними критеріями вибору були: підтримка мови програмування Python, можливості для роботи з HTML/CSS, наявність інструментів для налагодження та тестування, інтеграція з системами контролю версій, а також легкість у налаштуванні та використанні.

У таблиці 3.3 наведено порівняння основних характеристик трьох популярних середовищ розробки – Visual Studio Code, PyCharm та Atom, з урахуванням їхньої підтримки Python, веб-технологій, можливостей налагодження, інтеграції з системою контролю версій Git та особливих функцій.

Як бачимо з наведених у табл. 3.3 даних, Visual Studio Code має найбільш збалансований набір можливостей, що включає розвинуту підтримку Python і веб-технологій, зручні інструменти налагодження, інтеграцію з Git та широкий вибір розширень. Це робить його оптимальним вибором для розробки, оскільки дозволяє ефективно працювати над серверною та клієнтською частинами системи, забезпечуючи при цьому комфортний і гнучкий робочий процес.

Таблиця 3.3 – Порівняльний аналіз основних середовищ розробки

Характеристика	Visual Studio Code	PyCharm	Atom
Платформи	Windows, Mac, Linux	Windows, Mac, Linux	Windows, Mac, Linux
Підтримка Python	Висока	Відмінна	Добра
Підтримка веб-технологій	Відмінна	Добра	Добра
Налагодження	Вбудоване	Відмінне	Плагіни
Інтеграція з Git	Вбудована	Вбудована	Вбудована
Особливості	Легка, розширювана, безкоштовна	Потужний IDE для Python	Відкритий код, розширюваний

3.2 Інфраструктурне забезпечення інформаційної системи інвентаризації обладнання навчальних закладів

IT-інфраструктура інформаційної системи інвентаризації обладнання навчальних закладів охоплює сукупність апаратних і програмних компонентів, які забезпечують стабільне функціонування, доступність, захист та масштабованість розробленого вебзастосунку. З огляду на вимоги до простоти розгортання, система проєктувалася так, щоб її можна було легко інсталювати як локально, так і на серверному або хмарному середовищі.

Апаратне забезпечення. Для ефективної роботи системи не потрібні значні обчислювальні ресурси. Мінімальні вимоги для серверної частини:

- процесор: Intel Core i3 або аналог;
- оперативна пам’ять: від 4 ГБ;
- накопичувач: від 20 ГБ вільного простору (SSD бажаний);
- мережеве підключення: Ethernet або Wi-Fi з доступом до локальної мережі чи Інтернету (за потреби).

Для кінцевих користувачів (адміністраторів, інвентаризаторів) достатньо персонального комп’ютера або ноутбука зі встановленим сучасним веббраузером.

Програмне забезпечення. Програмна частина реалізується з використанням

вільного та кросплатформного стеку технологій, який забезпечує гнучкість та легкість підтримки:

- операційна система: Windows 10/11 (для розробки), Ubuntu (для серверного розгортання);
- мова програмування: Python 3.10;
- фреймворк: Flask – мікрофреймворк для створення вебзастосунків;
- система управління базами даних: SQLite – зручна для локальних проєктів, з можливістю подальшої заміни;
- фронтенд: HTML, CSS, Bootstrap – для побудови адаптивного інтерфейсу користувача;
- система керування пакетами: pip;
- середовище розробки: Visual Studio Code.

Архітектура системи. Система реалізується за моделлю клієнт-серверної архітектури. Користувач працює через вебінтерфейс, який надсилає HTTP-запити до Flask-додатка, розгорнутого на сервері. Обробка запитів та взаємодія з базою даних відбувається на стороні сервера.

Така модель дозволяє легко масштабувати застосунок, змінюючи лише серверну частину, без зміни клієнтської логіки.

Захист даних і розмежування доступу. Інформаційна система передбачає реалізацію базових засобів безпеки, які забезпечують цілісність і конфіденційність даних:

- автентифікація користувачів через логін/пароль;
- авторизація відповідно до ролей (адміністратор, інвентаризатор, користувач);
- валідація вхідних даних як на клієнті, так і на сервері;
- обмеження доступу до адміністративних функцій і маршрутів;
- захист сесій (за потреби можлива інтеграція з JWT або Flask-Login).

Масштабованість і можливості підтримки. Архітектура інформаційної системи передбачає її розвиток у майбутньому. Серед можливих напрямів масштабування:

– можливість перенесення на хмарні платформи (PythonAnywhere, VPS);

– заміна SQLite на повнофункціональні СУБД (PostgreSQL, MySQL);

– інтеграція з LDAP або Active Directory для централізованого управління користувачами.

Таким чином, створена інфраструктура відповідає основним вимогам до сучасних вебсистем: вона проста у впровадженні, підтримує базові заходи безпеки та готова до подальшого розвитку.

3.3 Програмна реалізація серверної та клієнтської частини

Програмна реалізація системи інвентаризації обладнання навчальних закладів виконуємо відповідно до розроблених UML-діаграм, що дозволить забезпечити структурований та логічно зв’язаний код, а також відповідність функціональним вимогам.

Для початку створюємо файл app.py, який буде основним файлом запуску серверної частини. У ньому ініціалізується Flask-додаток, налаштовується підключення до бази даних та визначаються основні маршрути для обробки HTTP-запитів (рис. 3.1).

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///inventory.db'
app.config['SECRET_KEY'] = 'your_secret_key'

db = SQLAlchemy(app)
```

Рисунок 3.1 – Ініціалізація Flask-додатку та налаштування підключення до бази даних SQLite

У програмному коді наведеному на рисунку 3.1:

– Flask(__name__) створює екземпляр веб-додатку;

– SQLALCHEMY_DATABASE_URI визначає шлях до бази даних SQLite;

- SECRET_KEY необхідний для безпеки сесій і форм;
- SQLAlchemy(app) ініціалізує ORM для роботи з базою даних.

Далі необхідно створити основні моделі даних, які будуть відображати ключові таблиці бази даних згідно з ER-діаграмою (див. рис. 2.1). Для цього у файлі моделей визначаються класи, що успадковують від базового класу ORM (у випадку SQLAlchemy – db.Model). Кожен клас відповідає окремій таблиці, а атрибути класу – полям таблиці.

Принцип створення моделей полягає у визначенні класу, у якому кожне поле відображається у вигляді екземпляру відповідного типу колонки (наприклад, db.Integer, db.String, db.DateTime), а також налаштуванні зв'язків між таблицями (наприклад, зовнішніх ключів та відношень).

Для забезпечення коректності роботи системи необхідно реалізувати взаємозв'язки між моделями: наприклад, модель Equipment (рис. 3.2) матиме зовнішній ключ category_id, що посилається на Category, а також room_id – на Room. Аналогічно, InventoryResult зв'язується з InventorySession і Equipment. Це дозволяє ORM керувати зв'язками та полегшує виконання складних запитів.

```python
class Equipment(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(150), nullable=False)
    status = db.Column(db.String(50), nullable=False)
    qr_code = db.Column(db.String(100), unique=True)
    acquisition_date = db.Column(db.Date)

    category_id = db.Column(db.Integer, db.ForeignKey('category.id'), nullable=False)
    room_id = db.Column(db.Integer, db.ForeignKey('room.id'), nullable=False)
    responsible_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=True)

    inventory_results = db.relationship('InventoryResult', backref='equipment', lazy=True)

class InventoryResult(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    session_id = db.Column(db.Integer, db.ForeignKey('inventory_session.id'), nullable=False)
    equipment_id = db.Column(db.Integer, db.ForeignKey('equipment.id'), nullable=False)
    actual_status = db.Column(db.String(50))
    comments = db.Column(db.String(255))
    checked_by = db.Column(db.Integer, db.ForeignKey('user.id'))
```

Рисунок 3.2 – Фрагмент програмної реалізації класів Equipment, InventoryResult

Таким чином, створення моделей у вигляді класів згідно з ER-діаграмою забезпечує структуроване та прозоре представлення даних, що відповідає логіці роботи системи. Після створення моделей даних наступним кроком є формування маршрутів (routes) веб-додатку, які відповідають за обробку HTTP-запитів від клієнтської частини. У цих маршрутах реалізується логіка додавання, редагування, видалення та перегляду записів бази даних. Кожен маршрут зв'язується з відповідною функцією обробника, що взаємодіє з моделями через ORM SQLAlchemy. Приклад фрагмента коду з оголошенням маршруту для додавання обладнання наведено на рисунку 3.3.

Представлений фрагмент коду (рис. 3.3) відповідає за реалізацію маршруту /add_equipment, що забезпечує додавання нового обладнання до бази даних. У випадку, якщо запит має метод POST, дані з HTML-форми зчитуються через request.form, створюється новий екземпляр класу Equipment, який додається до сесії SQLAlchemy та зберігається в базі. Після цього виконується перенаправлення на головну сторінку за допомогою redirect(url_for('index')). Якщо запит має метод GET, повертається HTML-сторінка з формою для введення інформації про обладнання через шаблон add_equipment.html.

```python
@app.route('/add', methods=['GET', 'POST'])
def add_equipment():
    categories = Category.query.all()
    rooms = Room.query.all()
    persons = ResponsiblePerson.query.all()

    if request.method == 'POST':
        new_equipment = Equipment(
            name=request.form['name'],
            serial_number=request.form['serial_number'],
            acquisition_date=datetime.strptime(request.form['purchase_date'], '%Y-%m-%d'),
            status=request.form['status'],
            inventory_notes=request.form.get('inventory_notes'),
            last_inventory_date=datetime.strptime(request.form['last_inventory_date'], '%Y-%m-%d')
            if request.form['last_inventory_date'] else None,
            category_id=request.form['category'],
            room_id=request.form['room'],
            responsible_id=request.form['responsible']
        )
        db.session.add(new_equipment)
        db.session.commit()
        return redirect(url_for('index'))

    return render_template('add_equipment.html', categories=categories, rooms=rooms, persons=persons)
```

Рисунок 3.3 – Обробка маршруту додавання нового обладнання у Flask

Для реалізації клієнтської частини інформаційної системи використовуються HTML-шаблони, які відповідають за відображення інформації, отриманої із серверної частини, та забезпечення взаємодії користувача з інтерфейсом. Шаблони формуються за допомогою вбудованої у Flask системи шаблонів – Jinja2, що дозволяє інтегрувати Python-змінні безпосередньо в HTML-код.

Шаблон index.html (рис. 3.4) відповідає за відображення списку наявного обладнання в системі. Він отримує дані із серверної частини та виводить їх у вигляді таблиці. Також містить посилання для переходу до форми додавання нового обладнання.

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Обладнання</title>
    <style>
        body { font-family: Arial, sans-serif; background-color: #f9f9f9; padding: 40px; }
        table { width: 100%; border-collapse: collapse; background: white; border-radius: 6px; }
        th, td { border: 1px solid #ccc; padding: 10px; text-align: left; }
        th { background-color: #f2f2f2; }
        a.button {
            display: inline-block; margin-top: 20px; padding: 10px 20px; background: #007bff;
            color: white; text-decoration: none; border-radius: 4px;
        }
        a.button:hover { background-color: #0056b3; }
    </style>
</head>
<body>
    <h1>Список обладнання</h1>
    <table>
        <thead>
            <tr>
                <th>Назва</th>
                <th>Серійний номер</th>
                <th>Тип</th>
                <th>Локація</th>
                <th>Дата покупки</th>
                <th>Статус</th>
            </tr>
        </thead>
        <tbody>
```

Рисунок 3.4 – Фрагмент коду index.html

Окрім шаблону index.html, у клієнтській частині інформаційної системи було також реалізовано ще декілька HTML-шаблонів, кожен з яких виконує конкретну

функцію у рамках взаємодії користувача з інтерфейсом системи:

1. add_equipment.html – призначений для створення нового запису про обладнання. Містить форму для введення основних параметрів обладнання (назва, серійний номер, категорія, статус, локація, дата придбання тощо). Після надсилання форма обробляється сервером, і новий запис додається до бази даних.

2. edit_equipment.html – використовується для редагування наявного запису про обладнання. Шаблон заповнюється даними з бази даних та дозволяє користувачеві змінити параметри обраного пристрою.

3. view_equipment.html – призначений для перегляду детальної інформації про конкретне обладнання без можливості редагування. Може містити додаткову інформацію, таку як історія інвентаризацій або нотатки відповідальних осіб.

4. delete_confirmation.html – реалізує інтерфейс підтвердження видалення запису. Запитує у користувача підтвердження дії, що є важливою частиною захисту від випадкового видалення.

5. base.html – є базовим шаблоном, який використовується для уніфікації зовнішнього вигляду сторінок. Містить загальну структуру документа, спільні стилі, навігаційні елементи та заголовки. Інші шаблони наслідують його за допомогою механізму шаблонізації Jinja2.

Для забезпечення багаторівневого доступу до функціональності інформаційної системи інвентаризації обладнання навчального закладу було реалізовано також механізм реєстрації користувачів із призначенням ролей. Реєстрація нових користувачів здійснюється через веб-інтерфейс із валідацією вхідних даних. Фрагмент програмного коду наведено на рисунку 3.5.

Форма містить такі поля:

- Ім’я користувача (username) – унікальний ідентифікатор користувача в системі;

- Пароль (password) – захищене поле для введення паролю;

- Роль (role) – випадаючий список, який дозволяє вибрати одну з доступних ролей: admin – адміністратор системи; inventory – інвентаризатор; viewer – користувач із правами перегляду.

```python
@app.route('/register', methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('index'))

    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        role = request.form['role']

        if role not in ['admin', 'inventory', 'viewer']:
            flash('Некоректна роль.')
            return redirect(url_for('register'))

        existing_user = User.query.filter_by(username=username).first()
        if existing_user:
            flash('Користувач з таким ім'ям вже існує.')
            return redirect(url_for('register'))

        new_user = User(username=username, role=role)
        new_user.set_password(password)
        db.session.add(new_user)
        db.session.commit()
        flash('Реєстрація успішна. Тепер увійдіть у систему.')
        return redirect(url_for('login'))

    return render_template('register.html')
```

Рисунок 3.5 – Фрагмент програмного коду серверної частини реєстрації користувачів із призначенням ролі

HTML-форма оформлена за допомогою вбудованих стилів CSS для забезпечення зручного користувацького інтерфейсу. Реалізація форми базується на шаблонізаторі Jinja2, а дані передаються методом POST.

На стороні сервера запит обробляється у маршруті /register. Логіка реалізована у Flask-додатку та включає такі етапи:

1. Отримання та перевірка вхідних даних – виконується перевірка наявності всіх обов'язкових полів (username, password, role) і перевірка унікальності імені користувача.

2. Хешування пароля – перед збереженням у базі даних пароль хешується за допомогою функції generate_password_hash з бібліотеки werkzeug.security, що підвищує безпеку системи.

3. Створення облікового запису – дані нового користувача записуються в базу даних через ORM SQLAlchemy.

4. Виведення повідомлення – при успішній реєстрації або помилці виводиться відповідне повідомлення через flash.

Після реєстрації кожному користувачу призначається роль, яка використовується для обмеження доступу до окремих маршрутів програми. Наприклад, тільки адміністратор має доступ до сторінки створення нових користувачів, а переглядач не може змінювати записи. Для цього використовується перевірка ролі у відповідних маршрутах.

3.4 Тестування функціональності інформаційної системи

Для забезпечення надійної роботи інформаційної системи інвентаризації навчальних закладів було проведено комплексне тестування, яке включало перевірку основних функціональних модулів системи. Метою тестування було виявлення помилок, перевірка коректності роботи функцій та відповідності системи поставленим вимогам.

1. Перевірка процесу реєстрації нового користувача. Спочатку користувач переходить на головну сторінку системи, де знаходить посилання авторизацію або кнопку, що веде на сторінку реєстрації (рис. 3.6).

2. Для додавання нового користувача, після кліку на “Зареєструватися” відкривається форма реєстрації, яка містить поля для введення імені, пароля та ролі користувача (рис. 3.7). Користувач вводить відповідні дані, заповнюючи всі необхідні поля, після чого натискає кнопку для підтвердження реєстрації.

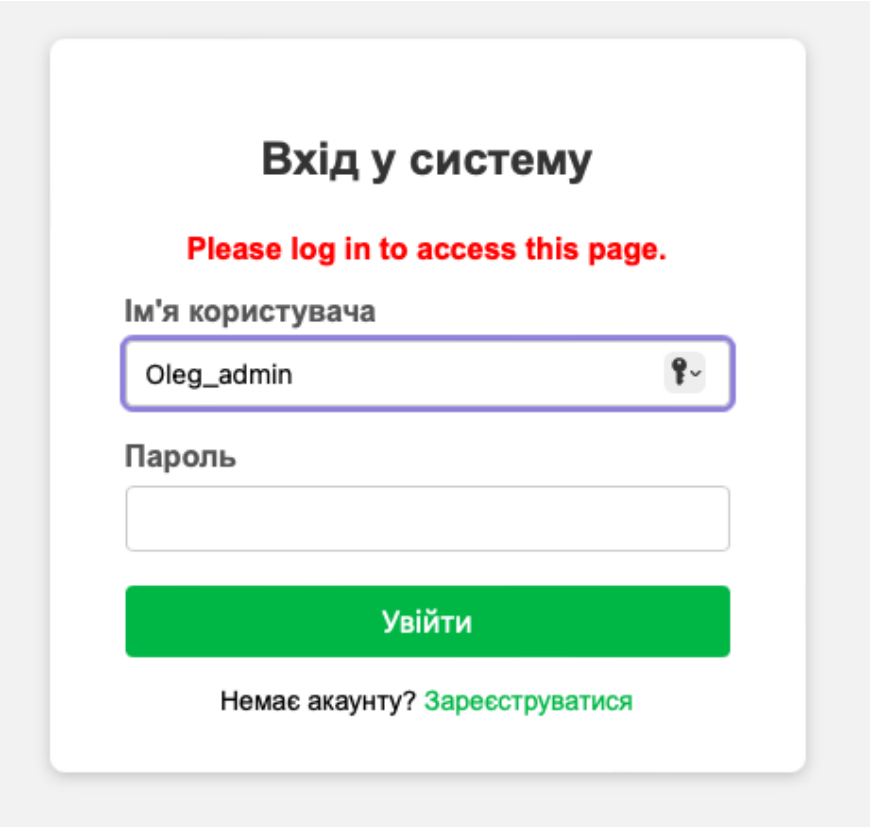

Рисунок 3.6 – Головна сторінка авторизації у системі

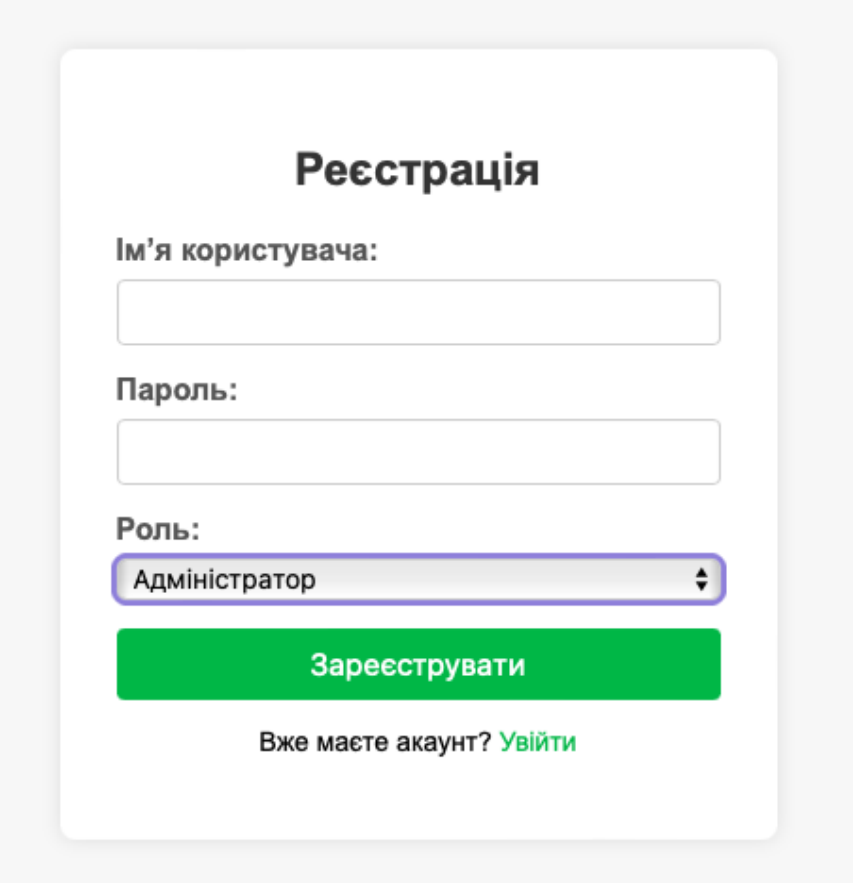

Рисунок 3.7 – Форма реєстрації у системі

3. Якщо всі дані коректні та унікальні, система створює нового користувача і перенаправляє його на сторінку входу. У випадку введення некоректних даних користувач побачить відповідне повідомлення про помилку (рис. 3.8).

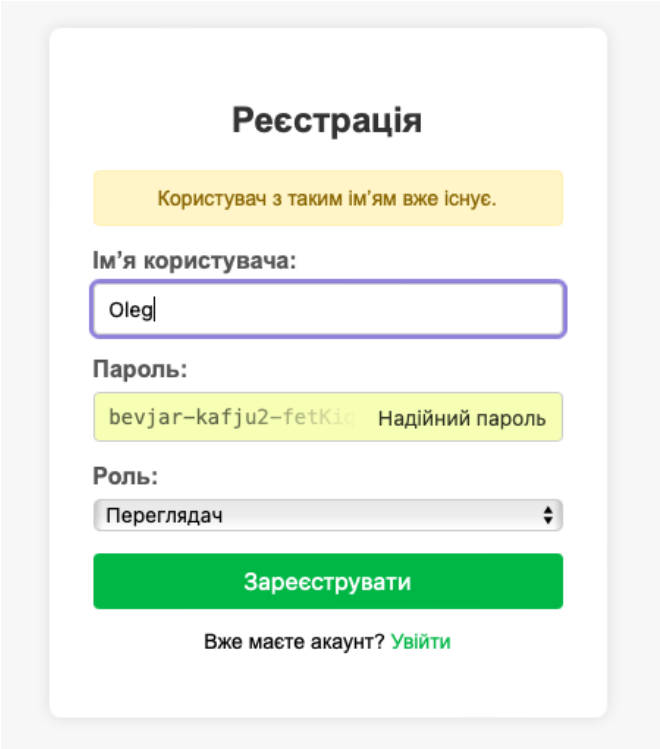

Рисунок 3.8 – Повідомлення про некоректні дані реєстрації користувача

4. На наступному етапі тестування виконано перевірку формування бази даних обладнання. Після успішної реєстрації або авторизації користувача система виконує перевірку облікових даних і при їх коректності надає доступ до основного функціоналу. На сторінці зі списком обладнання користувач має можливість додати новий запис, натиснувши кнопку "Додати обладнання" (рис. 3.9).

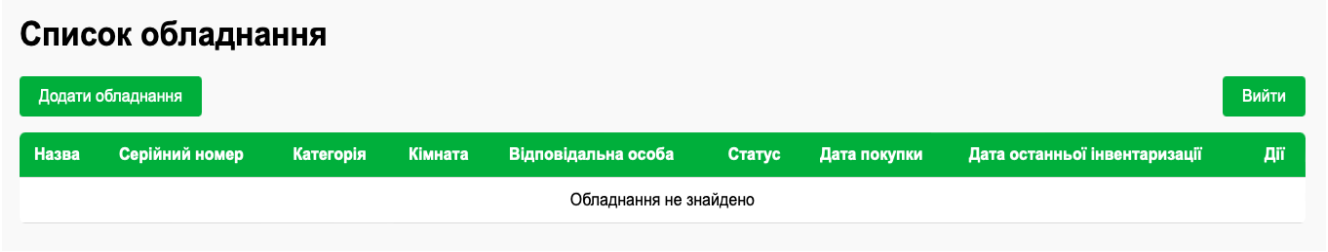

Рисунок 3.9 – Головна сторінка формування бази даних обладнання

5. Після натискання кнопки "Додати обладнання" відкривається форма, в якій користувач вказує всі необхідні параметри: назву обладнання, серійний номер, категорію, кімнату розташування, відповідальну особу, статус обладнання та дату його придбання (рис. 3.10).

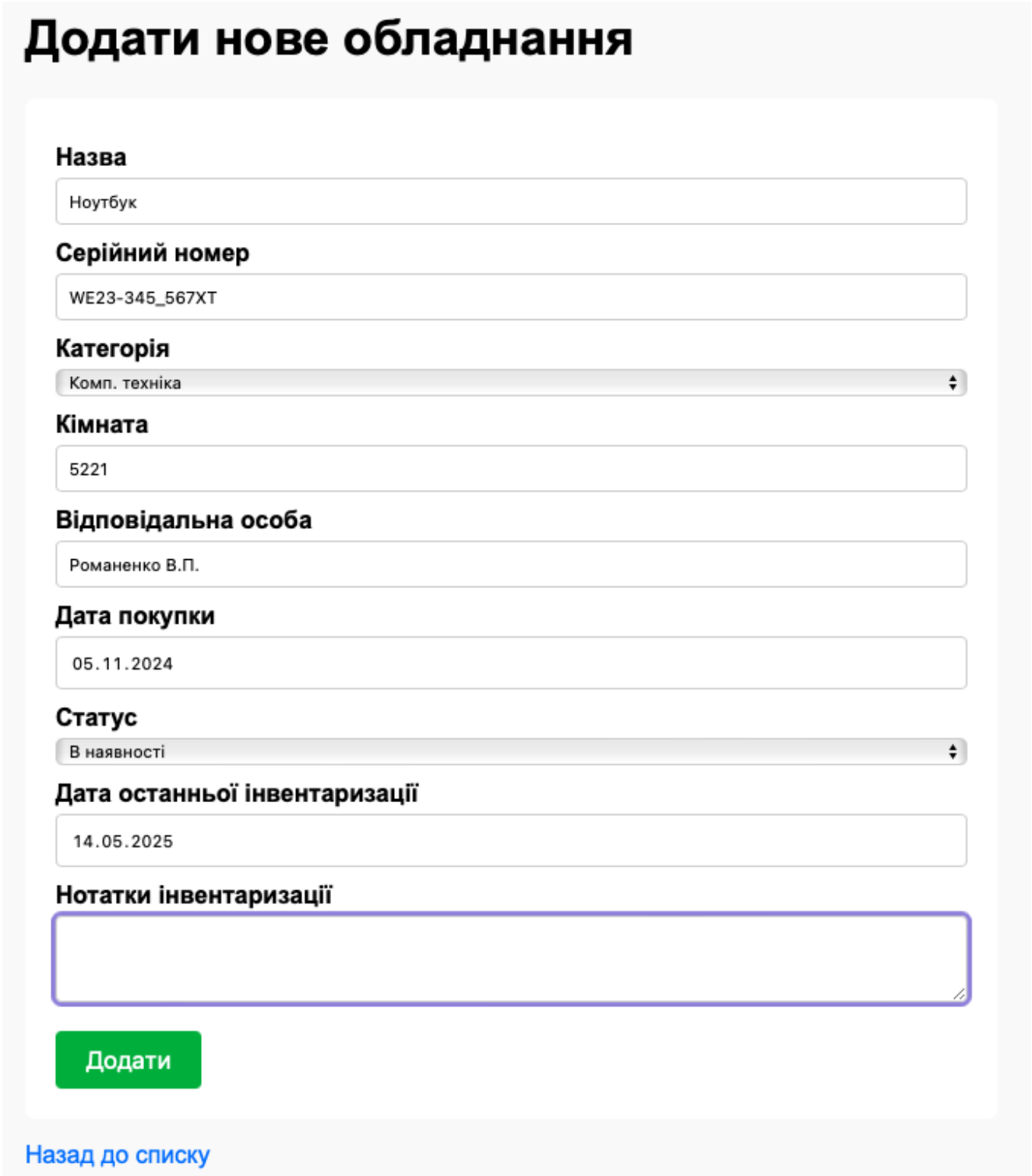

Рисунок 3.10 – Скріншот форми додавання нового обладнання

6. Після введення даних користувач натискає кнопку збереження. Якщо дані введені не коректно або не у повному обсязі система виводить відповідне повідомлення та вказує на ті поля форми, що залишилися ще не заповненими (рис. 3.11).

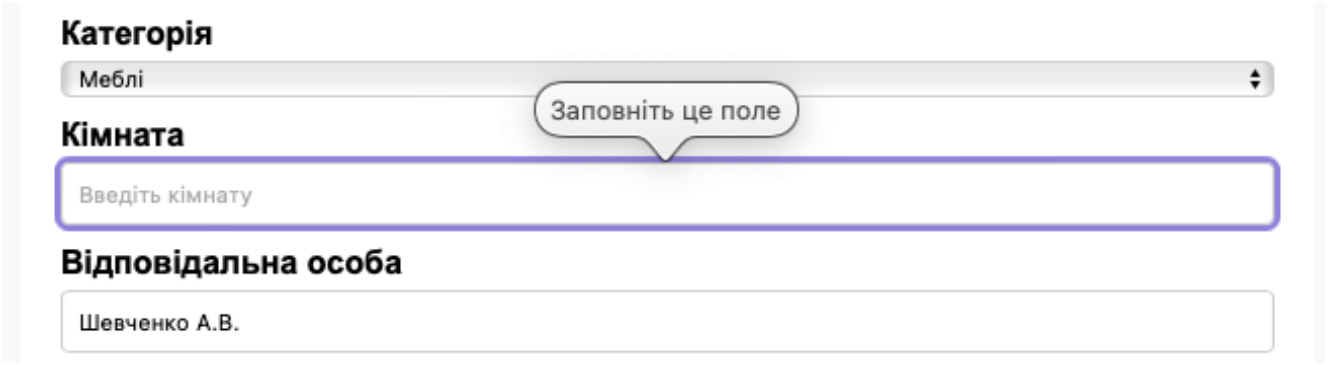

Рисунок 3.11 – Повідомлення про некоректність введення даних

7. Якщо всі поля заповнені коректно, система додає запис у базу даних і повертає користувача до списку обладнання, де новий запис відображається серед інших (рис. 3.12).

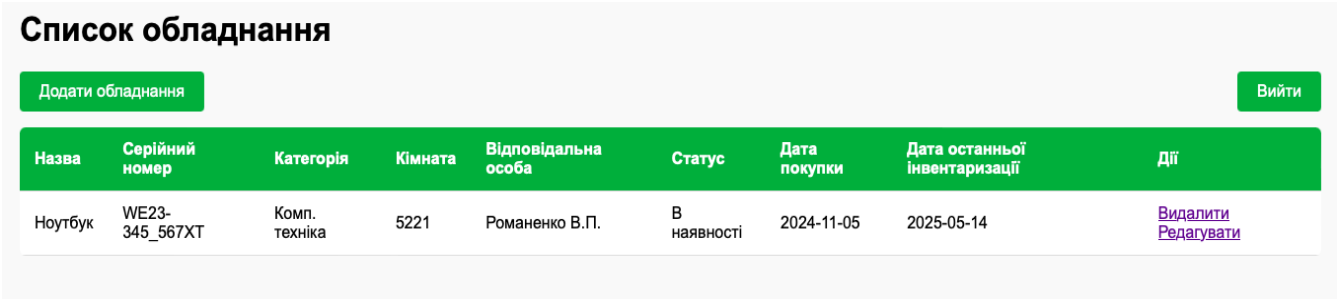

Рисунок 3.12 – Список обладнання після додавання нових даних

8. Також додатково було додано декілька інших категорій обладнання з різними параметрами для перевірки коректності формування бази даних та подальшого тестування процесу інвентаризації (рис. 3.13).

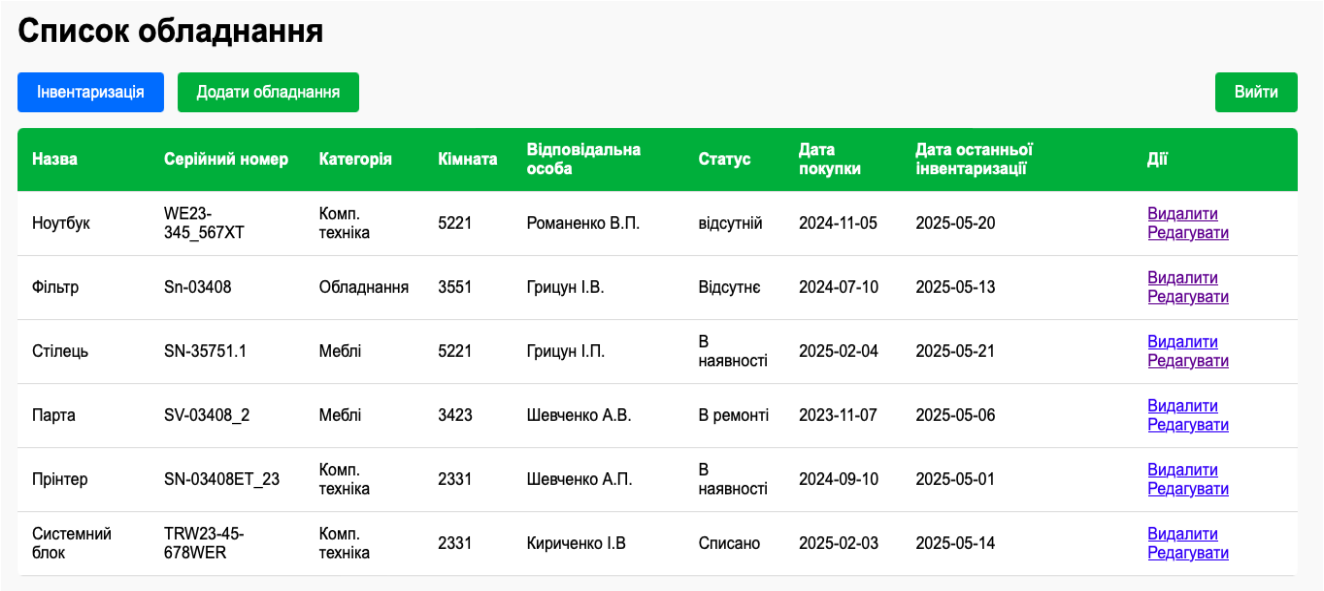

Рисунок 3.13 – Список обладнання для різних типів категорій

9. Процес тестування змін у базі даних охоплює функціональність редагування та видалення записів обладнання через користувацький інтерфейс.

Для перевірки редагування користувач відкриває сторінку зі списком обладнання, знаходить потрібний запис і натискає посилання «Редагувати» (рис. 3.13). Після цього відкривається форма з попередньо заповненими даними, у які користувач може вносити відповідні зміни – наприклад, серійний номер, категорію або відповідальну особу (рис. 3.14).

Після внесення змін натискається кнопка збереження, і система оновлює дані в базі. Успішність операції перевіряється візуально на сторінці списку – змінені значення відображаються без помилок.

10. Тестування видалення здійснюється аналогічно. Користувач натискає на посилання «Видалити» навпроти відповідного запису (рис. 3.13). Після цього система виводить запит на підтвердження (рис. 3.15). Після підтвердження елемент зникає зі списку, що свідчить про успішне видалення обладнання із сформованої бази даних.

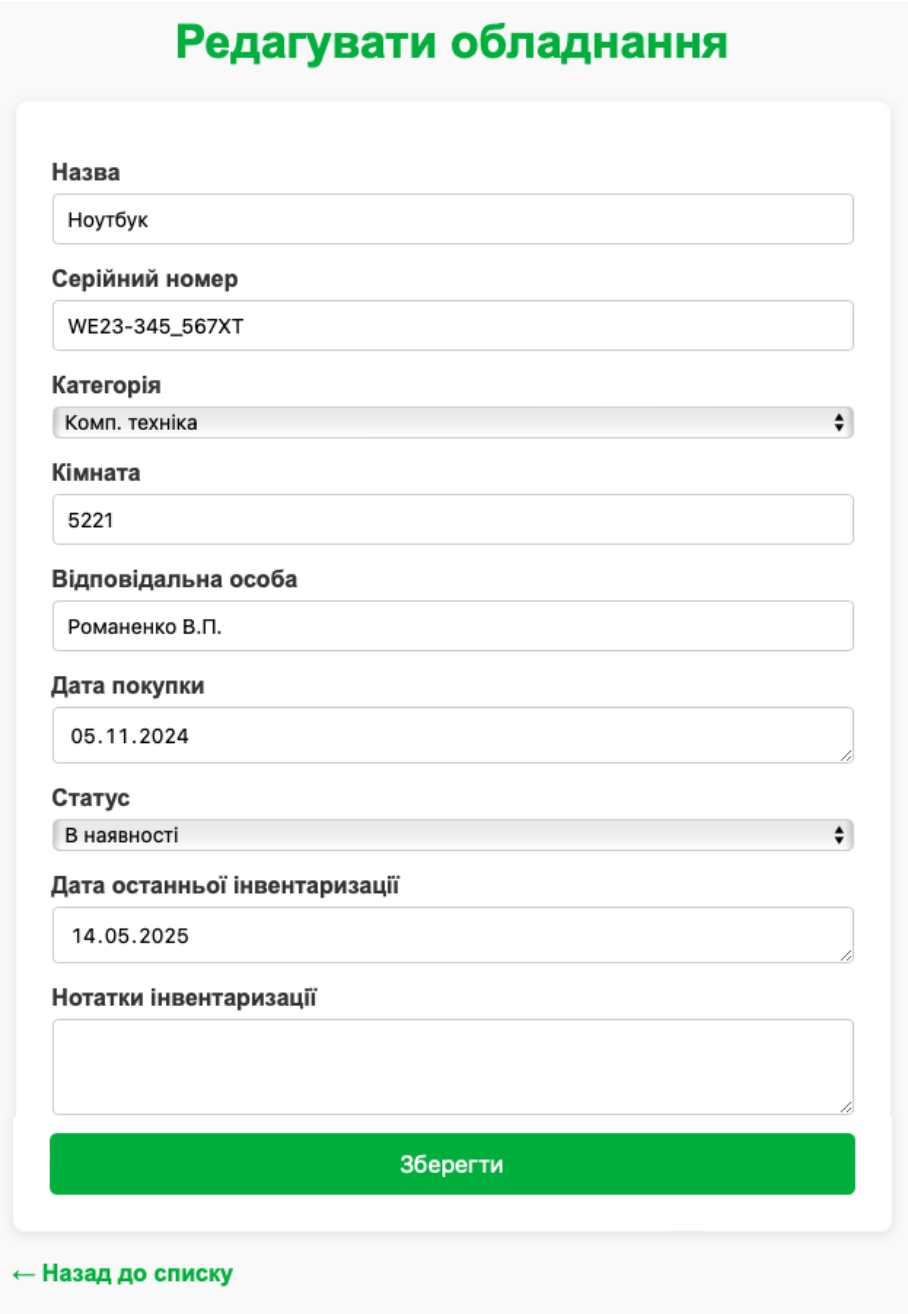

Рисунок 3.14 – Форма редагування обладнання

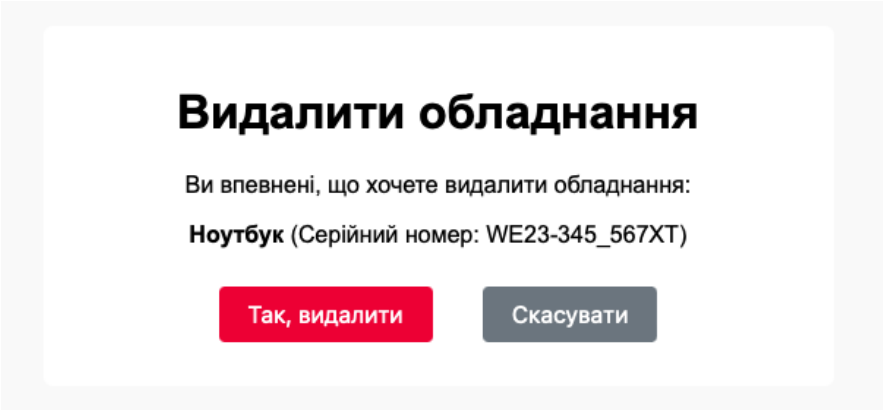

Рисунок 3.15 – Повідомлення про підтвердження видалення обладнання

11. Наступним етапом є тестування безпосередньо процесу інвентаризації. Для початку користувач переходить на спеціальну сторінку інвентаризації, натиснувши відповідне посилання “Інвентаризація” (рис. 3.13) в меню або на головній сторінці. На цій сторінці відображається список аудиторій (кімнат), які доступні для перевірки (рис. 3.16).

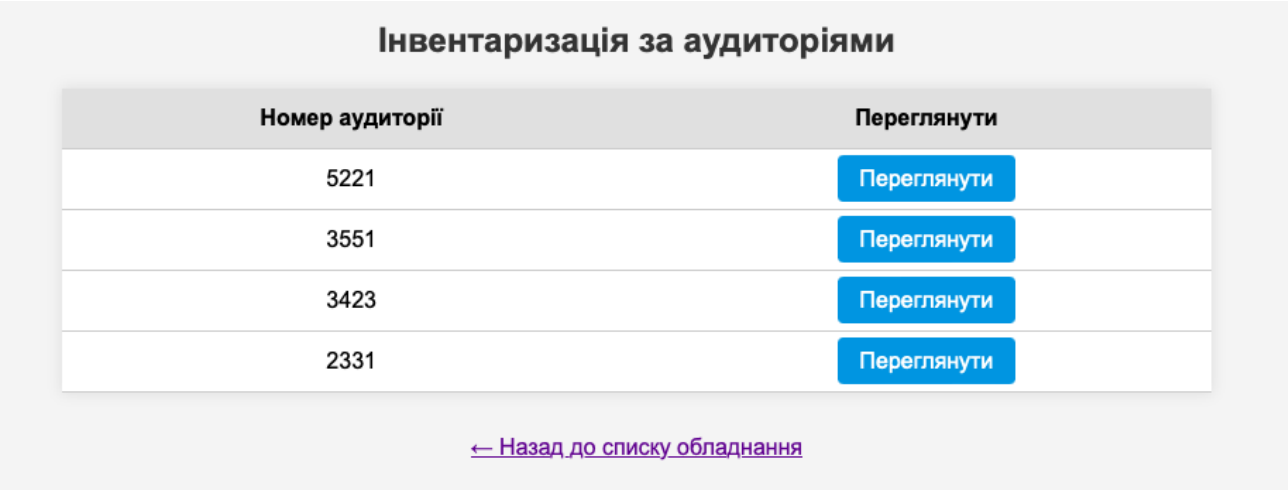

Рисунок 3.16 – Вибір аудиторії для проведення процесу інвентаризації

12. Користувач вибирає потрібну аудиторію, після чого відкривається список обладнання, яке знаходиться в цій кімнаті (рис. 3.17), (рис. 3.18).

Рисунок 3.17 – Список обладнання та проведення процесу інвентаризації в обраній аудиторії 5221

Для кожного елемента в списку (рис. 3.17) доступна опція зміни статусу, яка реалізована у вигляді випадаючого меню або окремої форми. Користувач вибирає актуальний статус обладнання, наприклад, «в наявності», «відсутній» або «в

ремонті», і підтверджує зміну. Після цього необхідно натиснути кнопку збереження, щоб система оновила інформацію в базі даних. У разі успішного оновлення статусів список обладнання на сторінці автоматично відображає актуальні дані.

Рисунок 3.18 – Список обладнання та проведення процесу інвентаризації в обраній аудиторії 2331

Таким чином, проведене тестування інформаційної системи інвентаризації обладнання навчальних закладів підтвердило її працездатність, відповідність функціональним вимогам, стабільність у типових сценаріях використання та зручність взаємодії з інтерфейсом. Система забезпечує точне внесення, редагування та видалення даних, успішно формує аналітичну інформацію, що дозволяє оптимізувати процеси обліку матеріальних ресурсів.

Також варто зазначити, що на відміну від переважної частини наявних рішень, які або мають надмірну комерціалізованість, або не орієнтовані на потреби освітніх установ, розроблений модуль враховує специфіку роботи навчальних закладів та дозволяє проводити інвентаризацію із гнучкою роллю користувачів та базовими засобами аналітики.

У таблиці 3.4 наведено порівняльний аналіз функціональних можливостей розробленої інформаційної системи відносно систем «MIA: Електронна інвентаризація», «M.E.Doc» та модуля «Облік основних засобів» корпоративної ERP-платформи IT-Enterprise.

Аналізуючи дані таблиці 3.4, можна відзначити, що розроблена інформаційна система вирізняється орієнтацією на потреби навчальних закладів, зокрема завдяки

простому, але ефективному інтерфейсу для проведення інвентаризації, ролям користувачів та можливості швидкого оновлення інформації щодо обладнання. На відміну від більшості існуючих рішень, система не перевантажена надлишковим функціоналом і забезпечує базові, але необхідні інструменти для реального обліку матеріальних ресурсів.

Таблиця 3.4 – Порівняння функціональних можливостей розробленої інформаційної системи з конкурентними системами

Критерій	Розроблена інформаційна система	MIA: Електронна інвентаризація	M.E.Doc	IT-Enterprise (облік ОЗ)
Орієнтація на освітні заклади	+	–	–	–
Багаторівнева система ролей	+	–	–	+
Формування звітів	+	+	+	+
Інтуїтивний інтерфейс	+	–	–	–
Редагування/видалення записів	+	+	+	+
Історія інвентаризації	+	+	–	+
Безкоштовне або бюджетне впровадження	+	–	–	–
Локальне розгортання / офлайн-доступ	+	+	–	+

Таким чином адаптація під освітню сферу, простота використання, підтримка багаторівневого доступу та інтуїтивне управління базою даних надають розробленій системі перевагу у використанні для інвентаризації саме в навчальних закладах, порівняно з універсальними або комерційними аналогами. Зокрема можна відзначити, що розроблена система реалізує 8 ключових функціональних можливостей, що на 3 більше (або на 37,5%) порівняно з найближчим аналогом – модулем «Облік основних засобів» у складі IT-Enterprise, який підтримує лише 5 із них (табл. 3.4).

3.5 Висновки

У третьому розділі розглянуто програмну реалізацію інформаційної системи інвентаризації обладнання навчальних закладів, що включала вибір мови програмування та інструментів розробки, створення бази даних, реалізацію функціоналу для керування обліком, авторизації користувачів і проведення інвентаризації. Обрана технологія Flask (Python) у поєднанні з SQLite забезпечила швидку розробку, масштабованість і простоту розгортання системи. Було реалізовано інтерфейс користувача, що дозволяє зручно управляти обладнанням, фіксувати зміни, а також проводити інвентаризацію. Ручне тестування підтвердило працездатність ключових функцій системи – реєстрації, додавання, редагування, видалення та проведення інвентаризації обладнання. У порівнянні з аналогічними рішеннями, система реалізує 8 функціональних можливостей, що на 37,5% більше, ніж у найближчого аналога, і є адаптованою до потреб саме освітніх закладів, забезпечуючи ефективний облік матеріальних ресурсів.

ВИСНОВКИ

Поставлені перед бакалаврською кваліфікаційною роботою задачі виконано в повному обсязі, а саме:

1. Здійснено аналіз проблематики проведення інвентаризації обладнання навчальних закладів, що дало змогу виявити ключові труднощі традиційного обліку: фрагментованість даних, низьку ефективність ручних процесів, відсутність централізованого контролю та прозорості в управлінні матеріальними ресурсами.

2. Проведено порівняльний аналіз вітчизняних інформаційних систем інвентаризації, зокрема «MIA: Електронна інвентаризація», «M.E.Doc» та модуля IT-Enterprise. Визначено обмеження їх функціональних можливостей у контексті потреб освітніх закладів та обґрунтовано напрямки вдосконалення.

3. Виконано проєктування інформаційної системи інвентаризації, яке включало розробку структури бази даних, побудову UML-діаграм (ER-діаграма, діаграма класів, варіантів використання та послідовності), що забезпечило створення логічно структурованої та масштабованої архітектури системи.

4. Реалізовано інформаційну систему, використовуючи мову програмування Python, вебфреймворк Flask та СУБД SQLite. Розроблений програмний продукт забезпечує управління обліком обладнання, авторизацію користувачів, формування й збереження інвентаризаційних сесій та обробку даних через зручний вебінтерфейс.

5. Проведено тестування функціональності інформаційної системи, що підтвердило правильність реалізації ключових операцій. Співставлення функціоналу з існуючими аналогами показало, що система реалізує на 37,5% більше функціональних можливостей, ніж найближчий конкурент.

Таким чином, мету бакалаврської кваліфікаційної роботи досягнуто – створено функціональну, адаптовану до специфіки освітніх закладів інформаційну систему інвентаризації обладнання, яка забезпечує автоматизацію обліку матеріальних ресурсів та підвищення прозорості управління.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Височан О. С., Височан О. О. Інвентаризація: сутність, класифікація, принципи, функції та завдання. *Науковий вісник Ужгородського національного університету. Серія: Міжнародні економічні відносини та світове господарство*, 2018. Вип. 22, ч. 1. С. 48–52. URL: https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/24253/1/ІНВЕНТАРИЗАЦІЯ%20СУТНІСТЬ%2c%20КЛАСИФІКАЦІЯ%2c.pdf (дата звернення 25.04.2025).

2. Шевців Л., Дарчук П. Особливості проведення інвентаризації на підприємствах України: стан і перспективи. *Scientific Collection «InterConf»*, 2022. № 132. С. 78–85. URL: https://archive.interconf.center/index.php/conference-proceeding/article/view/1615 (дата звернення 25.04.2025).

3. Лега О., Канцедал Н., Пешков А. Інвентаризація як інструмент забезпечення фінансової стійкості підприємства. *Підприємництво та інновації*, 2025. Вип. 34. С. 37–44. URL: https://doi.org/10.32782/2415-3583/34.5 (дата звернення 25.04.2025).

4. Фрундіна Л., Артюх О. Вдосконалення процесу інвентаризації шляхом застосування комп'ютерних технологій. *ЛОГОΣ. ОНЛАЙН*, 2020. №16. URL: https://www.ukrlogos.in.ua/10.11232-2663-4139.16.64.html (дата звернення 25.04.2025).

5. Івахненков С. В. Автоматизація бізнес–процесів та бухгалтерського обліку: «хмарна» революція? *Бухгалтерський облік і аудит*. 2018. № 5. С. 26–35.

6. Височан О., Височан О., Коркішко В. Інвентаризація запасів в умовах війни. *Таврійський науковий вісник. Серія: Економіка*. 2024. № 20. С. 182–189. URL: https://doi.org/10.32782/2708-0366/2024.20.21 (дата звернення 25.04.2025).

7. Беренда Н. І., Хабенко О. В. Особливості інвентаризації виробничих запасів на підприємствах харчової промисловості. *Формування ринкових відносин в Україні*. 2015. № 1 (164). С. 125–127.

8. Електронна інвентаризація MIA. URL: https://infotech.gov.ua/projects/mia-digital-inventory (дата звернення: 02.05.2025).

9. M.E.Doc – своєчасна звітність та простий обмін документами. URL: https://medoc.ua (дата звернення: 02.05.2025).

10. IT-Enterprise – your one-stop platform for digital transformation. URL: https://it.ua (дата звернення: 03.05.2025).

11. Савчук Т. О., Ольшанська О. В. Технології комп'ютерного проєктування : електронний навчальний посібник комбінованого (локального та мережного) використання. Вінниця : ВНТУ, 2024. 125 с.

12. Мокін Б. І., Мокін В. Б., Мокін О. Б. Методи та засоби комп'ютерних обчислень : навчальний посібник. Вінниця : ВНТУ, 2024. 155 с.

13. Топ-10 мов програмування, які будуть актуальні в 2025 // IT Jobs at SoftServe – Start tech career now. URL: https://career.softserveinc.com/uk-ua/stories/top-10-programming-languages-to-learn (дата звернення: 11.05.2025).

14. Python.org. Welcome to Python.org. URL: https://www.python.org (дата звернення: 11.05.2025).

15 Що таке Node.js. URL: https://uk.theastrologypage.com/node-js (дата звернення: 16.05.2025).

16 Flask. URL: https://flask.palletsprojects.com/ (дата звернення: 12.05.2025).

17 SQLite – The Most Widely Deployed Database Engine URL: https://www.sqlite.org/mostdeployed.html (дата звернення: 16.05.2025).

18. Популярні середовища розробки мов програмування – UA5.org. URL: https://ua5.org/osnprog/1600-populyarni-seredovyshha-rozrobky-mov-programuvannya.html (дата звернення: 21.05.2025).

19. Середовище розробки Python: що для себе обере професіонал – FoxmindEd. URL: https://foxminded.ua/seredovyshche-rozrobky-python/ (дата звернення: 21.05.2025).

20. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології») / уклад.: В.Б. Мокін, С.О. Жуков, О. М. Козачко. Вінниця : ВНТУ, 2023. 68 с.

Додаток А

(обов’язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
____ д.т.н., проф. Віталій МОКІН
2025 року

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу

ІНФОРМАЦІЙНА СИСТЕМА ІНВЕНТАРИЗАЦІЇ ОБЛАДНАННЯ НАВЧАЛЬНИХ ЗАКЛАДІВ

08-34.БКР.018.00.000 ТЗ

Керівник: к.т.н., доцент каф. САІТ
_____ Георгій ГОРЯЧЕВ
«___» _____2025 р.
Розробив студент гр. 2ІСТ-21б
_____ Олег СЕВЕРИН
«__» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт

Підставою для виконання роботи є наказ № ___ по ВНТУ від «___» _____ 2025 р., та індивідуальне завдання на БКР, затверджене протоколом № ___ засідання кафедри САІТ від «___» _____ 2025 р.

2. Джерела розробки:

1) Електронна інвентаризація MIA URL: https://infotech.gov.ua/projects/mia-digital-inventory;

2) Фрундіна Л., Артюх О. Вдосконалення процесу інвентаризації шляхом застосування комп'ютерних технологій. *ЛОГОС. ОНЛАЙН*, 2020. №16. URL: https://www.ukrlogos.in.ua/10.11232-2663-4139.16.64.html

3. Мета і призначення роботи.

Метою роботи є розширення функціональних можливостей процесу інвентаризації обладнання навчальних закладів за рахунок створення інформаційної системи.

4. Вихідні дані для проведення робіт.

Дані щодо організації та проведення процесу інвентаризації обладнання у закладах освіти.

5. Методи дослідження.

Методи системного аналізу, об'єктно-орієнтованого проєктування, структурного моделювання (UML).

6. Етапи роботи і терміни їх виконання

a) Аналіз предметної області _____ – _____

b) Порівняльний аналіз існуючих проєктних рішень _____ – _____

c) Проєктування інформаційної системи _____ – _____

d) Програмна реалізація інформаційної системи _____ – _____

e) Оформлення матеріалів до захисту БКР _____ – _____

7. Очікувані результати та порядок реалізації.

Отримання інформаційної системи інвентаризації обладнання навчальних закладів.

8. Вимоги до розробленої документації.

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «____» _____ 2025 р.

Початок розробки «____» _____ 2025 р.

Граничні терміни виконання БКР «____» _____ 2025 р.

Розробив студент групи 2ІСТ-21б _____ Олег СЕВЕРИН

Додаток Б

(обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Інформаційна система інвентаризації обладнання навчальних закладів »

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІТА, гр. 2ІСТ-21б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 1,03 %

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:

Віталій МОКІН, зав. каф. САІТ _____
(підпис)

Євгеній КРИЖАНОВСЬКИЙ, доц. каф. САІТ _____
(підпис)

Особа, відповідальна за перевірку _____ Сергій ЖУКОВ
(підпис)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Георгій ГОРЯЧЕВ, к.т.н., доц. каф. САІТ
(підпис)

Здобувач _____ Олег СЕВЕРИН
(підпис)

Додаток В

(довідниковий)

Фрагмент лістингу програми

Код файлу inventory.py інформаційної системи

```
from flask import Flask, render_template, request, redirect, url_for, flash
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
from flask_login import UserMixin, LoginManager, login_user, logout_user, login_required,
current_user
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///inventory.db'
app.config['SECRET_KEY'] = 'your_secret_key'

db = SQLAlchemy(app)

login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'

# --- Константи категорій ---
CATEGORIES = [
    'Комп. техніка',
    'Меблі',
    'Обладнання',
    'ПЗ',
    'Інше'
]

# --- Моделі ---

class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(100), unique=True, nullable=False)
    password_hash = db.Column(db.String(128), nullable=False)
    role = db.Column(db.String(50), nullable=False, default='user')

    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 Equipment(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
```

```
    serial_number = db.Column(db.String(100), nullable=False)
    acquisition_date = db.Column(db.Date, nullable=True)
    status = db.Column(db.String(50), nullable=False)
    qr_code = db.Column(db.String(100), nullable=True)
    inventory_notes = db.Column(db.Text, nullable=True)
    last_inventory_date = db.Column(db.Date, nullable=True)

    category = db.Column(db.String(50), nullable=False)  # Категорія як текст з обмеженням
    auditorium = db.Column(db.String(100), nullable=True)  # Замість Room
    responsible_person = db.Column(db.String(100), nullable=True)  # Замість ResponsiblePerson

# --- flask-login ---

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

# --- Маршрути ---

@app.route('/')
@login_required
def index():
    equipment_list = Equipment.query.all()
    return render_template('index.html', equipment=equipment_list)

@app.route('/add', methods=['GET', 'POST'])
@login_required
def add_equipment():
    if request.method == 'POST':
        category = request.form['category']
        if category not in CATEGORIES:
            flash('Виберіть коректну категорію.')
            return redirect(url_for('add_equipment'))

        new_equipment = Equipment(
            name=request.form['name'],
            serial_number=request.form['serial_number'],
            acquisition_date=datetime.strptime(request.form['purchase_date'], '%Y-%m-%d') if
request.form['purchase_date'] else None,
            status=request.form['status'],
            inventory_notes=request.form.get('inventory_notes'),
            last_inventory_date=datetime.strptime(request.form['last_inventory_date'], '%Y-%m-%d') if
request.form['last_inventory_date'] else None,
            category=category,
            auditorium=request.form.get('auditorium'),
            responsible_person=request.form.get('responsible_person')
        )
        db.session.add(new_equipment)
        db.session.commit()
        flash('Обладнання додано успішно.')
        return redirect(url_for('index'))
```

```
    return render_template('add_equipment.html', categories=CATEGORIES)

@app.route('/delete/<int:id>', methods=['GET', 'POST'])
@login_required
def delete_equipment(id):
    eq = Equipment.query.get_or_404(id)
    if request.method == 'POST':
        db.session.delete(eq)
        db.session.commit()
        flash('Обладнання видалено успішно.')
        return redirect(url_for('index'))
    return render_template('delete_equipment.html', equipment=eq)

@app.route('/register', methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('index'))

    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        role = request.form['role']

        if role not in ['admin', 'inventory', 'viewer']:
            flash('Некоректна роль.')
            return redirect(url_for('register'))

        existing_user = User.query.filter_by(username=username).first()
        if existing_user:
            flash('Користувач з таким ім’ям вже існує.')
            return redirect(url_for('register'))

        new_user = User(username=username, role=role)
        new_user.set_password(password)
        db.session.add(new_user)
        db.session.commit()
        flash('Реєстрація успішна. Тепер увійдіть у систему.')
        return redirect(url_for('login'))

    return render_template('register.html')

@app.route('/edit/<int:id>', methods=['GET', 'POST'])
@login_required
def edit_equipment(id):
    equipment = Equipment.query.get_or_404(id)

    if request.method == 'POST':
        equipment.name = request.form['name']
        equipment.serial_number = request.form['serial_number']
        equipment.category = request.form['category']
        equipment.auditorium = request.form.get('auditorium')
        equipment.responsible_person = request.form.get('responsible_person')
```

```
        purchase_date_str = request.form.get('purchase_date')
        if purchase_date_str:
            equipment.acquisition_date = datetime.strptime(purchase_date_str, '%Y-%m-%d').date()
        else:
            equipment.acquisition_date = None

        equipment.status = request.form['status']

        last_inventory_date_str = request.form.get('last_inventory_date')
        if last_inventory_date_str:
            equipment.last_inventory_date = datetime.strptime(last_inventory_date_str, '%Y-%m-
%d').date()
        else:
            equipment.last_inventory_date = None

        equipment.inventory_notes = request.form.get('inventory_notes')

        db.session.commit()
        flash('Обладнання успішно оновлено')
        return redirect(url_for('index'))

    return render_template('edit_equipment.html',
                           equipment=equipment,
                           categories=CATEGORIES)


@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('index'))

    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')
            return redirect(url_for('index'))
        else:
            flash('Невірний логін або пароль')
            return redirect(url_for('login'))

    return render_template('login.html')


@app.route('/logout')
@login_required
def logout():
    logout_user()
```

```
    flash('Ви вийшли з системи.')
    return redirect(url_for('login'))

@app.route('/inventory')
@login_required
def inventory_list():
    auditoriums = db.session.query(Equipment.auditorium).distinct().all()
    auditoriums = [a[0] for a in auditoriums if a[0]]  # Відкидаємо None
    return render_template('inventory_list.html', auditoriums=auditoriums)


@app.route('/inventory/<auditorium>', methods=['GET', 'POST'])
@login_required
def inventory_by_auditorium(auditorium):
    equipment_list = Equipment.query.filter_by(auditorium=auditorium).all()

    if request.method == 'POST':
        for item in equipment_list:
            notes = request.form.get(f'notes_{item.id}')
            inventory_date_str = request.form.get(f'date_{item.id}')
            if inventory_date_str:
                item.last_inventory_date = datetime.strptime(inventory_date_str, '%Y-%m-%d').date()
            item.inventory_notes = notes
        db.session.commit()
        flash('Інвентаризація оновлена.')
        return redirect(url_for('inventory_by_auditorium', auditorium=auditorium))

    return render_template('inventory_by_room.html', auditorium=auditorium,
equipment=equipment_list)

@app.route('/update_status/<int:id>', methods=['POST'])
@login_required
def update_status(id):
    equipment = Equipment.query.get_or_404(id)
    new_status = request.form.get('status')

    if new_status:
        equipment.status = new_status
        equipment.last_inventory_date = datetime.now().date()
        db.session.commit()
        flash('Статус оновлено успішно.')

    return redirect(url_for('inventory_by_auditorium', auditorium=equipment.auditorium))


# --- Ініціалізація бази даних ---
if __name__ == '__main__':
    with app.app_context():
        db.create_all()
    app.run(debug=True)
```

Код файлу edit.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
   <meta charset="UTF-8" />
   <title>Редагувати обладнання</title>
   <style>
      body { font-family: Arial, sans-serif; margin: 40px; background: #f9f9f9; }
      form { background: white; padding: 20px; border-radius: 6px; max-width: 600px; }
      label { display: block; margin-top: 10px; font-weight: bold; }
      input, select, textarea {
         width: 100%; padding: 8px; margin-top: 4px; border: 1px solid #ccc; border-radius: 4px;
         box-sizing: border-box;
      }
      button {
         margin-top: 15px; padding: 10px 20px; background-color: #007bff; border: none;
         color: white; font-size: 16px; border-radius: 4px; cursor: pointer;
      }
      button:hover { background-color: #0056b3; }
      a { display: inline-block; margin-top: 15px; color: #007bff; text-decoration: none; }
      a:hover { text-decoration: underline; }
   </style>
</head>
<body>
   <h1>Редагувати обладнання</h1>
   <form method="POST">
      <label for="name">Назва</label>
      <input type="text" id="name" name="name" value="{{ equipment.name }}" required>

      <label for="serial_number">Серійний номер</label>
      <input type="text" id="serial_number" name="serial_number" value="{{
equipment.serial_number }}" required>

      <label for="type">Тип</label>
      <input type="text" id="type" name="type" value="{{ equipment.type }}" required>

      <label for="location">Локація</label>
      <input type="text" id="location" name="location" value="{{ equipment.location }}" required>

      <label for="purchase_date">Дата покупки</label>
      <input type="date" id="purchase_date" name="purchase_date" value="{{
equipment.purchase_date }}" required>

      <label for="status">Статус</label>
      <select id="status" name="status" required>
         <option value="">Оберіть статус</option>
         <option value="В наявності" {% if equipment.status == "В наявності" %}selected{% endif
%}>В наявності</option>
         <option value="В ремонті" {% if equipment.status == "В ремонті" %}selected{% endif
%}>В ремонті</option>
```

```
        <option value="Списано" {% if equipment.status == "Списано" %}selected{% endif
%}>Списано</option>
    </select>

    <label for="last_inventory_date">Дата останньої інвентаризації</label>
    <input type="date" id="last_inventory_date" name="last_inventory_date" value="{{
equipment.last_inventory_date }}">

    <label for="inventory_notes">Нотатки інвентаризації</label>
    <textarea id="inventory_notes" name="inventory_notes" rows="3">{{
equipment.inventory_notes }}</textarea>

    <button type="submit">Зберегти</button>
  </form>
  <a href="/">Назад до списку</a>
</body>
</html>
```

Код файлу index.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8" />
  <title>Список обладнання</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 40px; background: #f9f9f9; }
    table {
      border-collapse: collapse;
      width: 100%;
      background: white;
      border-radius: 6px;
      overflow: hidden;
    }
    th, td {
      padding: 12px 15px;
      border-bottom: 1px solid #ddd;
      text-align: left;
    }
    th {
      background-color: #28a745;
      color: white;
    }
    tr:hover { background-color: #f1f1f1; }
    a.button {
      display: inline-block;
      margin-bottom: 15px;
      padding: 10px 20px;
      background-color: #28a745;
      color: white;
      text-decoration: none;
```

```
            border-radius: 4px;
        }
        a.button:hover { background-color: #218838; }
        .logout {
            float: right;
            background-color: #dc3545;
        }
        .logout:hover {
            background-color: #c82333;
        }
    </style>
</head>
<body>
    <h1>Список обладнання</h1>

    <a href="{{ url_for('inventory_list') }}" class="button" style="background-color: #007bff; margin-
right: 10px;">Інвентаризація</a>
    <a href="{{ url_for('add_equipment') }}" class="button">Додати обладнання</a>
    <a href="{{ url_for('logout') }}" class="button logout">Вийти</a>

    <table>
        <thead>
            <tr>
                <th>Назва</th>
                <th>Серійний номер</th>
                <th>Категорія</th>
                <th>Кімната</th>
                <th>Відповідальна особа</th>
                <th>Статус</th>
                <th>Дата покупки</th>
                <th>Дата останньої інвентаризації</th>
                <th>Дії</th>
            </tr>
        </thead>
        <tbody>
            {% for eq in equipment %}
            <tr>
                <td>{{ eq.name }}</td>
                <td>{{ eq.serial_number }}</td>
                <td>{{ eq.category }}</td>
                <td>{{ eq.auditorium }}</td>
                <td>{{ eq.responsible_person }}</td>
                <td>{{ eq.status }}</td>
                <td>{{ eq.acquisition_date.strftime('%Y-%m-%d') if eq.acquisition_date else '' }}</td>
                <td>{{ eq.last_inventory_date.strftime('%Y-%m-%d') if eq.last_inventory_date else ''
}}</td>
                <td>
                    <a href="{{ url_for('delete_equipment', id=eq.id) }}">Видалити</a>
                    <a href="{{ url_for('edit_equipment', id=eq.id) }}">Редагувати</a>
                </td>
            </tr>
            {% else %}
```

```
        <tr>
          <td colspan="9" style="text-align:center;">Обладнання не знайдено</td>
        </tr>
        {% endfor %}
      </tbody>
  </table>
</body>
</html>
```

Код файлу inventory_by_room.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
 <meta charset="UTF-8">
 <title>Інвентаризація — {{ auditorium }}</title>
 <style>
  body {
   font-family: Arial, sans-serif;
   background-color: #f8f8f8;
   padding: 20px;
  }

  h2 {
   text-align: center;
   margin-bottom: 30px;
  }

  table {
   width: 90%;
   margin: auto;
   border-collapse: collapse;
   background-color: #fff;
   box-shadow: 0 0 10px rgba(0,0,0,0.1);
  }

  th, td {
   border: 1px solid #ddd;
   padding: 12px;
   text-align: center;
  }

  th {
   background-color: #eaeaea;
  }

  select, input[type="submit"] {
   padding: 6px 10px;
   border-radius: 4px;
   border: 1px solid #ccc;
  }
```

```
    input[type="submit"] {
      background-color: #3498db;
      color: white;
      border: none;
      cursor: pointer;
    }

    input[type="submit"]:hover {
      background-color: #2980b9;
    }

    .back-link {
      text-align: center;
      margin-top: 20px;
    }

    .back-link a {
      color: #333;
      text-decoration: none;
    }
  </style>
</head>
<body>
  <h2>Інвентаризація в аудиторії {{ auditorium }}</h2>

  {% if equipment %}
  <table>
    <thead>
      <tr>
        <th>Назва</th>
        <th>Серійний номер</th>
        <th>Категорія</th>
        <th>Відповідальний</th>
        <th>Статус</th>
        <th>Оновити</th>
      </tr>
    </thead>
    <tbody>
      {% for item in equipment %}
      <tr>
        <form action="{{ url_for('update_status', id=item.id) }}" method="POST">
          <td>{{ item.name }}</td>
          <td>{{ item.serial_number }}</td>
          <td>{{ item.category }}</td>
          <td>{{ item.responsible_person or '—' }}</td>
          <td>
            <select name="status">
              <option value="в наявності" {% if item.status == 'в наявності' %}selected{% endif %}>в
наявності</option>
              <option value="відсутній" {% if item.status == 'відсутній' %}selected{% endif
%}>відсутній</option>
```

```
            <option value="потребує ремонту" {% if item.status == 'потребує ремонту' %}selected{%
endif %}>потребує ремонту</option>
          </select>
         </td>
         <td>
          <input type="submit" value="Зберегти">
         </td>
        </form>
      </tr>
      {% endfor %}
    </tbody>
  </table>
  {% else %}
    <p style="text-align:center;">У цій аудиторії немає обладнання.</p>
  {% endif %}

  <div class="back-link">
    <a href="{{ url_for('inventory_list') }}">← Назад до списку аудиторій</a>
  </div>
</body>
</html>
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА СИСТЕМА ІНВЕНТАРИЗАЦІЇ ОБЛАДНАННЯ НАВЧАЛЬНИХ ЗАКЛАДІВ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

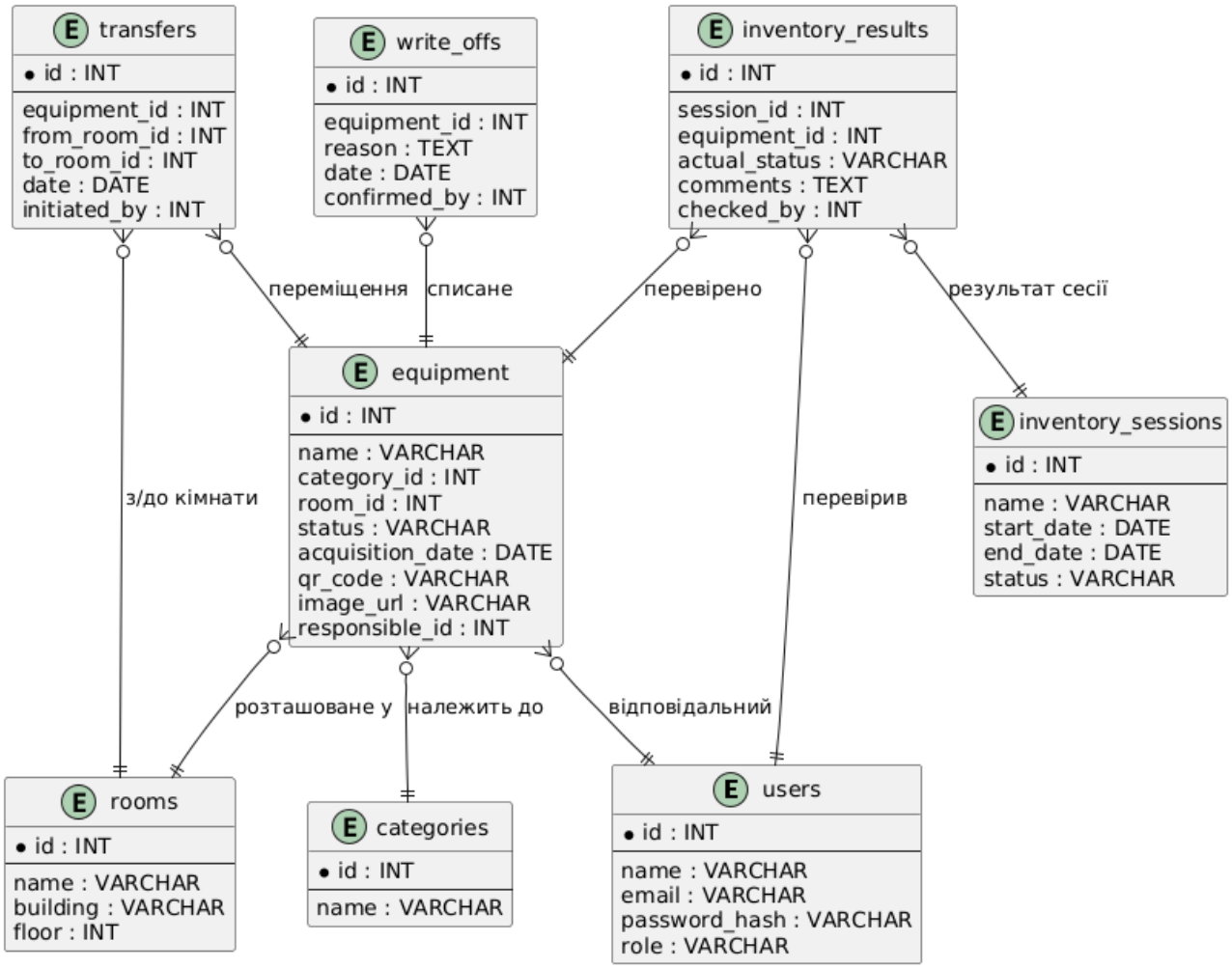

Рисунок Г.1 – Логічна ER-діаграма бази даних інформаційної системи інвентаризації обладнання

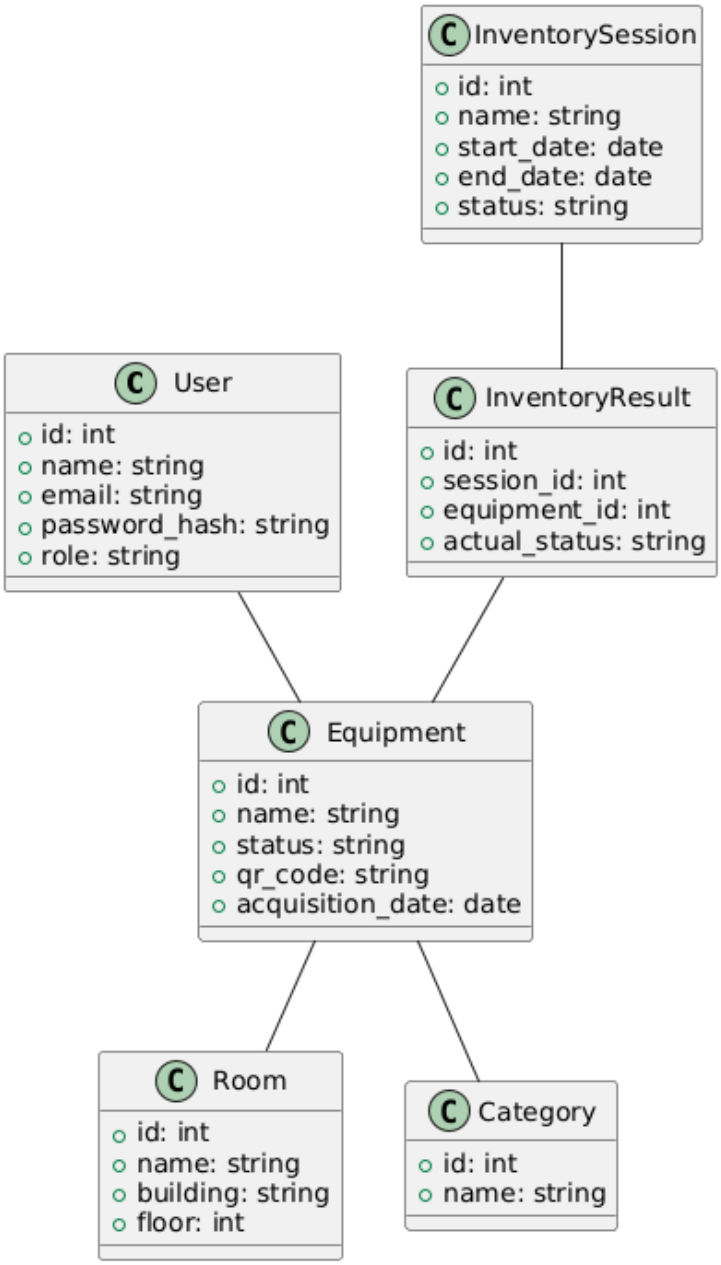

Рисунок Г.2 – Діаграма класів інформаційної системи інвентаризації обладнання

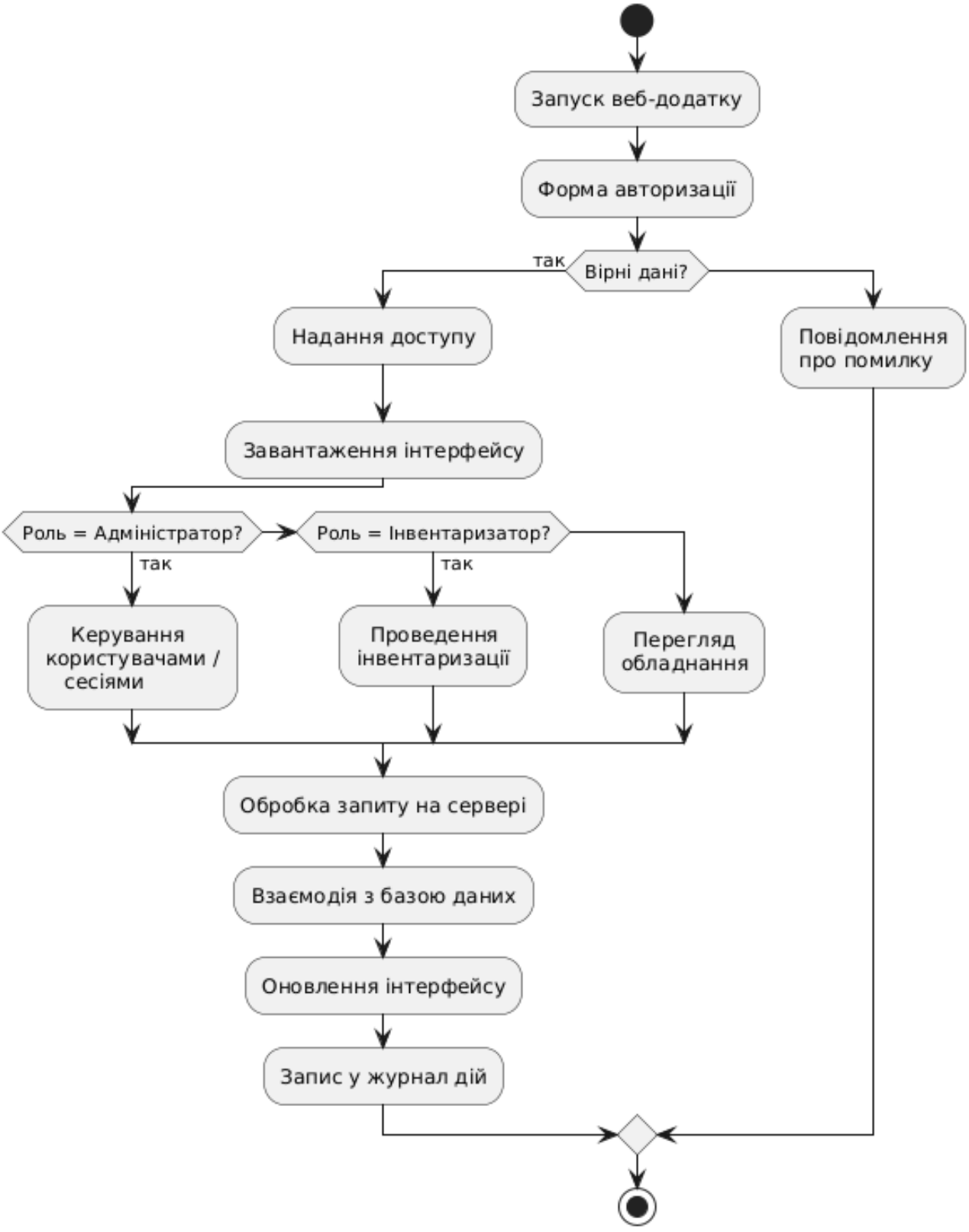

Рисунок Г.3 – Блок-схема основного алгоритму функціонування інформаційної системи інвентаризації обладнання

Рисунок Г.4 – Діаграма компонентів архітектури інформаційної системи інвентаризації

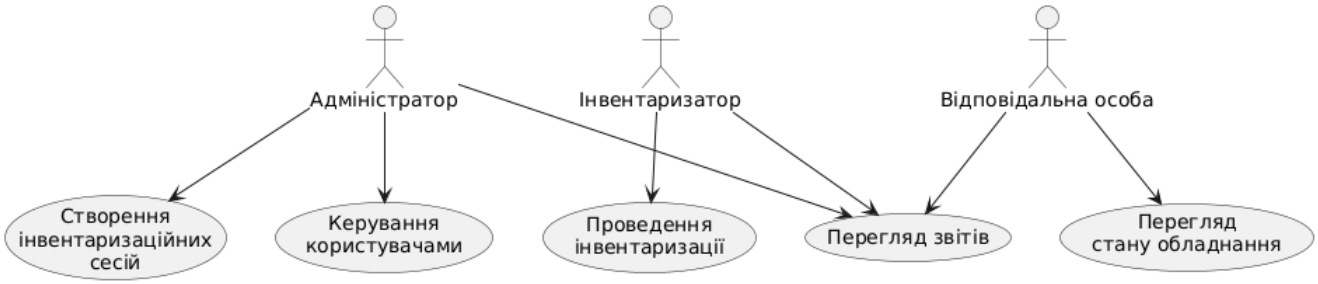

Рисунок Г.5 – Діаграма варіантів використання для інформаційної системи інвентаризації обладнання