ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ І ТЕХНОЛОГІЙ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

Бакалавра
(освітньо-кваліфікаційний рівень)

на тему *«Розробка десктопного застосунку для обліку та координації гуманітарної допомоги»*

Виконав: здобувач *4* року навчання
групи *4ПР1*
напряму підготовки (спеціальності)
121-Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)

Климук Вадим Олегович
(підпис, прізвище та ініціали)

Керівник *Козуб Наталя Олександрівна*
(підпис, прізвище та ініціали)

Рецензент *к.т.н., доцент Григорова А.А.*
(прізвище та ініціали)

Нормоконтролер *Комісаров О. С.*
(підпис, прізвище та ініціали)

Хмельницький – 2026

ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРЗВТЕТ

Інститут, факультет, відділення *Інформаційних технологій та дизайну*
Кафедра, циклова комісія *Програмних засобів і технологій*
Освітньо-кваліфікаційний рівень *бакалавр*
Освітньо-професійна підготовка *Програмна інженерія*
(шифр і назва)
Спеціальність *121-Інженерія програмного забезпечення*
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри *програмних засобів і технологій*
_____ О.Є. Огнєва
«___» _____ 2026 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Климука Вадима Олеговича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) *Розробка десктопного застосунку для обліку та координації гуманітарної допомоги*

керівник проєкту (роботи) *Козуб Наталія Олександрівна, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «16» січня 2026 року №82-с

2. Строк подання здобувачем проєкту(роботи) *10.06.2026*

3. Вихідні дані до проєкту (роботи) *літературні та періодичні джерела, матеріали переддипломної практики*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз предметної області та визначення вимог до програмного забезпечення, проєктування десктопного застосунку для обліку і координації гуманітарної допомоги, програмна реалізація застосунку для обліку і координації гуманітарної допомоги, тестування та оцінка якості застосунку

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
7 таблиць, 25 рисунків

6. Консультанти розділів проєкту (роботи)

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

7. Дата видачі завдання *14.01.2026*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів проєкту (роботи)	Примітка
1	Отримання завдання	14.01.2026	Виконано
2	Підбір літератури	17.01.2026-21.01.2026	Виконано
3	Аналіз предметної області	22.01.2026-26.01.2026	Виконано
4	Розробка та обґрунтування завдання	27.01.2026-12.02.2026	Виконано
5	Розробка концептуальної моделі	13.02.2026-19.02.2026	Виконано
6	Моделювання та проєктування системи	20.02.2026-02.03.2026	Виконано
7	Розробка чат-бота	03.03.2026-15.04.2026	Виконано
8	Тестування чат-бота	16.04.2026-20.04.2026	Виконано
9	Оформлення пояснювальної записки	21.04.2026-02.05.2026	Виконано
10	Захист кваліфікаційної роботи	23.06.2026	

Здобувач _____ *Климук В.О.*
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ *Козуб Н.О.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 77 сторінок, 25 рисунків, 7 таблиць, 3 додатки, 25 джерел.

Мета роботи:

Розробити десктопний застосунок для обліку і координації гуманітарної допомоги, який забезпечує автономну роботу без постійного інтернет-з'єднання, є зручним для волонтерів без спеціальної технічної підготовки та відповідає реальним потребам організацій, що здійснюють гуманітарну діяльність в Україні.

Об'єкт дослідження:

Процеси обліку та координації гуманітарної допомоги у волонтерських та громадських організаціях

Предмет дослідження:

Методи та засоби автоматизації обліку і координації гуманітарної допомоги засобами десктопного застосунку

Інструменти та засоби розробки:

Для розробки застосунку було використано стек технологій Python, PyQt6, SQLite і SQLAlchemy, що дозволило забезпечити ефективну реалізацію системи обліку і координації гуманітарної допомоги.

Результати роботи:

Результатом роботи є десктопний програмний застосунок, що автоматизує процеси реєстрації, зберігання, розподілу та моніторингу гуманітарних ресурсів, забезпечує автономну роботу без постійного інтернет-з'єднання та відповідає реальним потребам волонтерських і гуманітарних організацій, які діють в умовах воєнного часу в Україні.

Новизна рішення:

Розроблено десктопний програмний застосунок, що дозволяє спростити ведення обліку та формування звітності, забезпечує цілісність

обліку, дозволяє відстежувати рух ресурсів у реальному часі та максимально позбавитися людських помилок.

Практична цінність:

Створено програмний продукт, готовий до використання організаціями, що надають або розподіляють гуманітарну допомогу, і здатний суттєво підвищити ефективність їхньої роботи, мінімізувати втрати ресурсів і забезпечити прозорість звітності перед донорами та партнерами.

КЛЮЧОВІ СЛОВА: *ЗАСТОСУНОК, ОБЛІК, РОЗПОДІЛ, ГУМАНІТАРНА ДОПОМОГА, ПАКУНОК, КОРИСТУВАЧ, КООРДИНАЦІЯ.*

АНОТАЦІЯ

Кваліфікаційна робота бакалавра складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі проведено комплексний аналіз предметної області, проведено аналіз аналогів, який виявив їх недоліки і прогалини. Досліджено основні процеси обліку та координації гуманітарної допомоги, визначено ключові проблеми існуючих підходів та обґрунтовано необхідність створення спеціалізованого програмного забезпечення. Сформульовано функціональні та нефункціональні вимоги до розроблюваного десктопного застосунку. Обґрунтовано вибір десктопного типу застосунку та стеку технологій Python, PyQt6, SQLite і SQLAlchemy.

У другому розділі обґрунтовано вибір архітектурних рішень і технологій для реалізації десктопного застосунку, обрано архітектуру MVC із чітким поділом на рівні подання, контролерів, сервісів і доступу до даних, що забезпечує масштабованість і зручність супроводу застосунку. Розроблено багаторівневу архітектуру програмного забезпечення, структуру бази даних та моделі взаємодії між компонентами системи. Спроєктовано реляційну схему бази даних, що охоплює всі сутності предметної галузі і забезпечує повну простежуваність руху матеріальних цінностей через журнал транзакцій.

У третьому розділі представлено практичну реалізацію застосунку відповідно до спроєктованої архітектури. Розроблено структуру програмного проєкту, ORM-моделі бази даних, графічний інтерфейс користувача, механізми взаємодії компонентів системи, модулі авторизації та формування звітів. Програмно реалізовано функціональні модулі автентифікації з розмежуванням ролей, обліку надходжень і видач з автоматичним оновленням залишків, реєстру отримувачів із захистом від дублювання, складського обліку із сигналізацією критичних рівнів та генерації звітів у форматах PDF і XLSX.

У четвертому розділі описано стратегію і результати тестування застосунку на трьох рівнях: модульному, інтеграційному та тестуванні зручності. Автоматизовані тести забезпечили покриття коду на рівні 78 % і підтвердили коректність реалізованої бізнес-логіки. Проведене тестування підтвердило коректність роботи програмного забезпечення, стабільність функціонування системи та правильність взаємодії її структурних елементів.

ABSTRACT

The bachelor's qualification work consists of an introduction, four sections, conclusions, a list of sources used and appendices.

In the first chapter, a comprehensive analysis of the subject area was conducted, an analysis of analogues was conducted, which revealed their shortcomings and gaps. The main processes of accounting and coordination of humanitarian aid were investigated; the key problems of existing approaches were identified and the need for creating specialized software was justified. Functional and non-functional requirements for the developed desktop application were formulated. The choice of the desktop type of application and the technology stack of Python, PyQt6, SQLite and SQLAlchemy were justified.

The second chapter justifies the choice of architectural solutions and technologies for the implementation of the desktop application, choosing the MVC architecture with a clear division into the presentation, controller, service and data access levels, which ensures scalability and ease of application maintenance. A multi-level software architecture, database structure and interaction models between system components are developed. A relational database schema is designed that covers all entities of the subject area and provides full traceability of the movement of material assets through the transaction log.

The third chapter presents the practical implementation of the application in accordance with the designed architecture. The structure of the software project, ORM database models, graphical user interface, mechanisms for interaction of system components, authorization modules and report generation have been developed. Functional modules for authentication with role separation, accounting of receipts and issues with automatic balance update, recipient register with protection against duplication, warehouse accounting with signaling of critical levels and report generation in PDF and XLSX formats have been implemented in software.

The fourth section describes the strategy and results of application testing at three levels: modular, integration and usability testing. Automated tests provided code coverage of 78% and confirmed the correctness of the implemented business logic. The testing confirmed the correctness of the software, the stability of the system and the correct interaction of its structural elements.

ЗМІСТ

ВСТУП

Гуманітарна криза, що охопила Україну внаслідок повномасштабного вторгнення Росії у 2022 році, поставила перед волонтерськими організаціями, громадськими ініціативами та державними структурами безпрецедентне за масштабом завдання – організувати збір, облік і розподіл гуманітарної допомоги мільйонам людей в умовах постійної нестабільності. Щоденні потоки речей, продовольства, медикаментів та техніки, що надходять від донорів і спрямовуються до отримувачів, потребують чіткої системи координації, прозорого обліку та оперативного управління.

Актуальність теми кваліфікаційної роботи зумовлена тим, що більшість організацій, задіяних у роботі з гуманітарною допомогою, досі використовують таблиці в Microsoft Excel, паперові журнали або окремі месенджер-групи для координації своєї діяльності. Такі інструменти не забезпечують цілісного обліку, є схильними до людських помилок, не дозволяють відстежувати рух ресурсів у реальному часі та значно ускладнюють формування звітності. Спеціалізовані зарубіжні системи управління гуманітарною діяльністю є або надмірно складними для невеликих організацій, або не адаптованими до українського операційного контексту, або ж вимагають постійного підключення до мережі Інтернет, що в умовах бойових дій є критичним недоліком.

Мета кваліфікаційної роботи – розробити десктопний застосунок для обліку і координації гуманітарної допомоги, який забезпечує автономну роботу без постійного інтернет-з'єднання, є зручним для волонтерів без спеціальної технічної підготовки та відповідає реальним потребам організацій, що здійснюють гуманітарну діяльність в Україні.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати поточний стан у сфері обліку гуманітарної допомоги та виявити ключові проблеми наявних підходів;

- дослідити існуючі програмні аналоги та визначити їх переваги і недоліки;
- сформулювати функціональні та нефункціональні вимоги до розроблюваного застосунку;
- обрати відповідну архітектуру застосунку та технологічний стек розробки;
- розробити структуру бази даних і спроєктувати програмні модулі застосунку;
- реалізувати застосунок і провести його тестування;
- оцінити результати тестування та сформулювати рекомендації щодо подальшого розвитку.

Об'єктом дослідження є процеси обліку та координації гуманітарної допомоги у волонтерських та громадських організаціях.

Предметом дослідження є методи та засоби автоматизації обліку і координації гуманітарної допомоги засобами десктопного застосунку.

Практична цінність роботи полягає у створенні програмного продукту, готового до використання організаціями, що надають або розподіляють гуманітарну допомогу, і здатного суттєво підвищити ефективність їхньої роботи, мінімізувати втрати ресурсів і забезпечити прозорість звітності перед донорами та партнерами.

Новизна рішення: полягає в розробці десктопного програмного застосунку, що дозволяє спростити ведення обліку та формування звітності, забезпечує цілісність обліку, дозволяє відстежувати рух ресурсів у реальному часі та максимально позбавитися людських помилок.

Структура кваліфікаційної роботи. Робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Перший розділ присвячено аналізу предметної галузі, дослідженню аналогів і формуванню вимог до застосунку. У другому розділі обґрунтовано вибір архітектурних рішень і проєктування. У третьому розділі описано

реалізацію десктопного застосунку. Четвертий розділ містить результати тестування та аналіз якості програмного продукту.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИЗНАЧЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Характеристика предметної області

У сучасних умовах надзвичайних ситуацій, воєнних конфліктів, техногенних катастроф та гуманітарних криз особливого значення набуває ефективна організація процесів надання гуманітарної допомоги. Від оперативності та точності координації гуманітарних ресурсів значною мірою залежить забезпечення населення необхідними товарами, медикаментами, продуктами харчування та засобами першої необхідності.

Гуманітарна допомога являє собою систему заходів, спрямованих на забезпечення постраждалого населення матеріальними, медичними та соціальними ресурсами. Процеси обліку та координації гуманітарної допомоги включають:

- приймання гуманітарних вантажів;
- реєстрацію отриманих ресурсів;
- облік складських залишків;
- формування заявок на допомогу;
- розподіл ресурсів між отримувачами;
- контроль переміщення гуманітарних вантажів;
- формування звітності.

У більшості випадків облік гуманітарної допомоги здійснюється за допомогою електронних таблиць або паперової документації, що призводить до:

- дублювання інформації;
- втрати даних;
- складності пошуку інформації;
- низької швидкості обробки заявок;
- виникнення помилок під час ручного введення даних;
- відсутності централізованого контролю.

Використання спеціалізованого програмного забезпечення дозволяє автоматизувати зазначені процеси, підвищити ефективність роботи волонтерських штабів та гуманітарних організацій, а також забезпечити прозорість розподілу ресурсів.

1.2. Аналіз процесів обліку гуманітарної допомоги

Основними учасниками процесу координації гуманітарної допомоги є:

- адміністратори системи;
- оператори складів;
- волонтери;
- отримувачі допомоги;
- гуманітарні організації.

Процес обліку гуманітарної допомоги включає кілька основних етапів.

1. Приймання гуманітарної допомоги.

На першому етапі здійснюється приймання гуманітарних вантажів, їх перевірка та реєстрація в системі. Для кожного вантажу необхідно зберігати:

- назву ресурсу;
- категорію;
- кількість;
- дату надходження;
- інформацію про постачальника;
- місце зберігання.

2. Складський облік.

Після реєстрації гуманітарна допомога розподіляється між складами. На цьому етапі важливими є:

- контроль залишків;
- відстеження переміщення ресурсів;
- облік термінів придатності;

- запобігання втратам та дублюванню інформації.

3. Формування заявок.

Отримувачі допомоги або оператори системи формують заявки на необхідні ресурси. Система повинна забезпечувати:

- перевірку наявності ресурсів;
- фіксацію статусу заявки;
- контроль виконання заявок.

4. Видача гуманітарної допомоги.

Після погодження заявки здійснюється видача ресурсів. На цьому етапі система повинна:

- автоматично оновлювати складські залишки;
- фіксувати історію операцій;
- формувати відповідні звіти.

Схема процесу обліку гуманітарної допомоги представлена на рис. 1.1

Рисунок 1.1. Схема процесу обліку гуманітарної допомоги

1.3. Аналіз поточного стану у сфері обліку гуманітарної допомоги

1.3.1. Контекст і масштаб проблеми

Після початку повномасштабного вторгнення в лютому 2022 року Україна зіткнулася з однією з найбільших гуманітарних криз у Європі з часів Другої світової війни. За даними ООН, станом на 2024 рік понад 14,6 мільйона людей потребують гуманітарної підтримки, а внутрішньо переміщених осіб налічується близько 3,7 мільйона. Щорічний обсяг гуманітарної допомоги, що надходить в Україну від міжнародних організацій, урядів іноземних держав, діаспори та приватних донорів, вимірюється мільярдами доларів і мільйонами тонн фізичних вантажів.

Обробка такого потоку ресурсів вимагає розгалуженої мережі організацій: великих міжнародних НГО (UNHCR, ICRC, WFP), вітчизняних благодійних фондів, регіональних координаційних центрів, місцевих волонтерських груп і окремих ініціативних осіб. Кожна ланка цього ланцюга здійснює облік надходжень, зберігання, розподіл і передачу допомоги, і від якості цього обліку залежить, чи надійде допомога до тих, хто її найбільше потребує.

1.3.2. Типові учасники та їх ролі

В екосистемі гуманітарної допомоги можна виокремити кількох ключових учасників із різними інформаційними потребами:

- Донори – фізичні або юридичні особи, що передають матеріальну допомогу, грошові кошти чи послуги. Їм важливо знати, куди спрямовані їхні ресурси і яких результатів досягнуто.
- Координатори – особи або організації, що планують і контролюють розподіл ресурсів між отримувачами. Їм потрібне повне уявлення про наявні запаси, потреби і географію розподілу.

- Волонтери та оператори складу – виконують фізичне приймання, сортування і видачу допомоги. Потребують простого інтерфейсу для фіксації операцій у реальному часі.
- Отримувачі – люди або організації (громади, ЗСУ, лікарні, школи), які безпосередньо отримують допомогу. Дані про них потрібні для ведення реєстру і запобігання дублюванню.
- Звітуючі – відповідальні за формування звітності перед донорами, партнерами або державними органами.

Кожна з цих ролей висуває специфічні вимоги до інформаційної системи, що ускладнює вибір або створення єдиного інструменту.

1.3.3. Поточні методи ведення обліку та їх недоліки

Аналіз практики волонтерських організацій і гуманітарних штабів в Україні дозволяє виокремити кілька типових підходів до ведення обліку гуманітарної допомоги.

Електронні таблиці (Microsoft Excel, Google Sheets). Найпоширеніший інструмент серед малих і середніх волонтерських груп. Перевагою є доступність і знайомість інтерфейсу. Водночас таблиці не підтримують одночасну роботу кількох користувачів з одними даними (у десктопній версії), не мають вбудованих механізмів контролю цілісності даних, не захищені від випадкового видалення чи пошкодження записів і не дозволяють автоматично формувати структуровані звіти.

Паперові журнали і картки. Використовуються переважно в районах зі слабким інтернет-покриттям або серед осіб похилого віку, що задіяні у волонтерській діяльності. Такий підхід виключає будь-який оперативний пошук, аналіз або звітування і вимагає значних зусиль для оцифрування даних.

Месенджери та соціальні мережі (Telegram, Viber, Facebook). Нерідко слугують основним каналом координації: надходження і видача фіксуються у групових чатах, потреби публікуються у стрічках. Інформація у такому

форматі погано структурована, важко піддається пошуку і не придатна для аналітики.

Універсальні хмарні CRM і ERP-системи. Деякі більші організації намагаються адаптувати загальні системи управління (Salesforce, Zoho, 1C) під потреби гуманітарної діяльності. Це вимагає значних витрат на налаштування, навчання персоналу і підтримку, а також передбачає постійний доступ до мережі Інтернет.

Таким чином, спільними недоліками переважної більшості поточних підходів є:

- відсутність цілісного структурованого обліку всього ланцюга «донор – склад – отримувач»;
- неможливість оперативного відстеження залишків і руху матеріальних цінностей;
- ризики дублювання допомоги через відсутність централізованого реєстру отримувачів;
- складність формування звітності та аналітики;
- залежність від стабільного інтернет-з'єднання у хмарних рішеннях;
- висока вартість або складність впровадження спеціалізованих систем.

1.4. Аналіз існуючих програмних аналогів

1.4.1. Огляд аналогів

Для визначення місця розроблюваного застосунку серед існуючих рішень було проведено аналіз низки програмних продуктів, що застосовуються або можуть застосовуватися для управління гуманітарною діяльністю. До переліку увійшли як спеціалізовані системи для НГО, так і адаптовані універсальні платформи.

Kobo Toolbox. Платформа з відкритим вихідним кодом, розроблена для збору польових даних у гуманітарних місіях. Широко застосовується організаціями ООН та міжнародними НГО. Забезпечує створення форм для збору даних і базовий аналіз (рис. 1.2). Однак є хмарним сервісом без повноцінної офлайн-функціональності для управління запасами, не підтримує облік матеріальних ресурсів і потребує значного технічного налаштування.

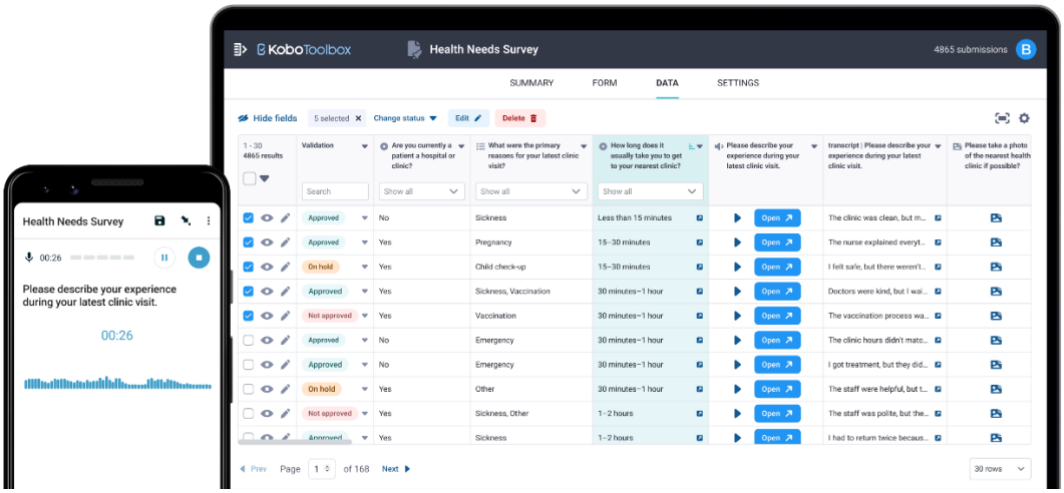

Рисунок 1.2. Kobo Toolbox

Humanitarian Data Exchange (HDX). Відкрита платформа OCHA для обміну гуманітарними даними між організаціями. Орієнтована на агрегацію і публікацію наборів даних, а не на оперативний облік. Не є системою управління запасами (рис. 1.3).

LogiSoft SurplusTrack. Комерційні десктопні системи обліку для складів і логістики. Мають розвинені функції управління запасами, але не адаптовані до специфіки гуманітарної діяльності (реєстр отримувачів, облік потреб, звітність для донорів) (рис. 1.4).

Salesforce Nonprofit Success Pack (NPSP). Потужна хмарна CRM-платформа з адаптацією для некомерційних організацій. Забезпечує управління донорами, фандрейзинг і звітність. Є надмірно складною для

малих організацій, вимагає постійного підключення до мережі та значних витрат на налаштування і ліцензування (рис. 1.5).

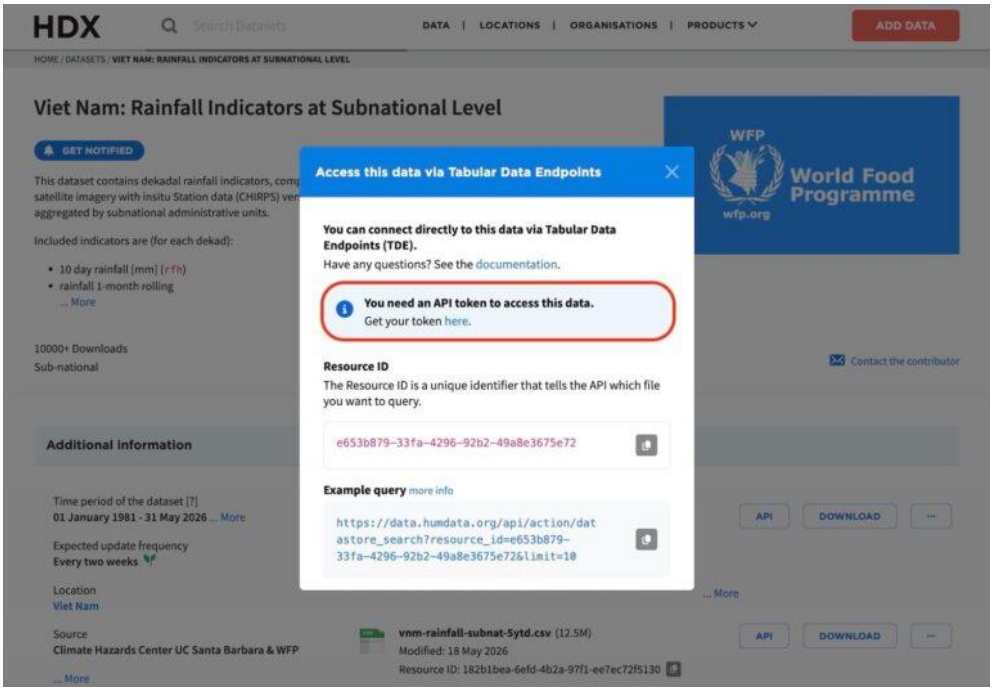

Рисунок 1.3. Humanitarian Data Exchange

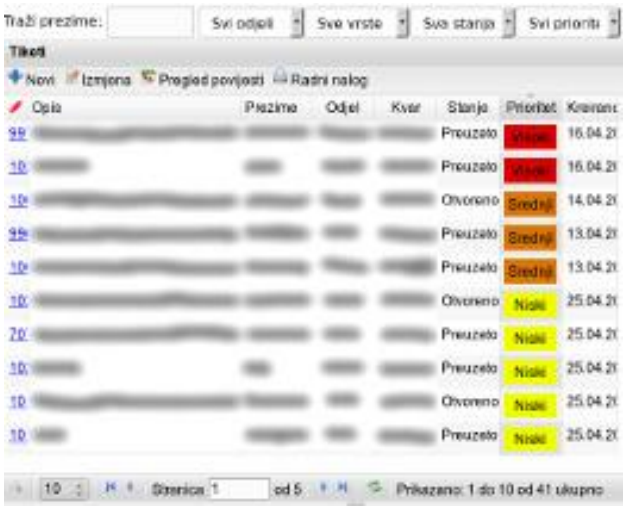

Рисунок 1.4. LogiSoft SurplusTrack

Microsoft Access / локальні бази даних. Ряд організацій самостійно розробляє локальні бази даних в Access або аналогічних інструментах.

Такий підхід є гнучким, але вимагає технічних компетенцій для супроводу і не має зручного для нетехнічних користувачів інтерфейсу.

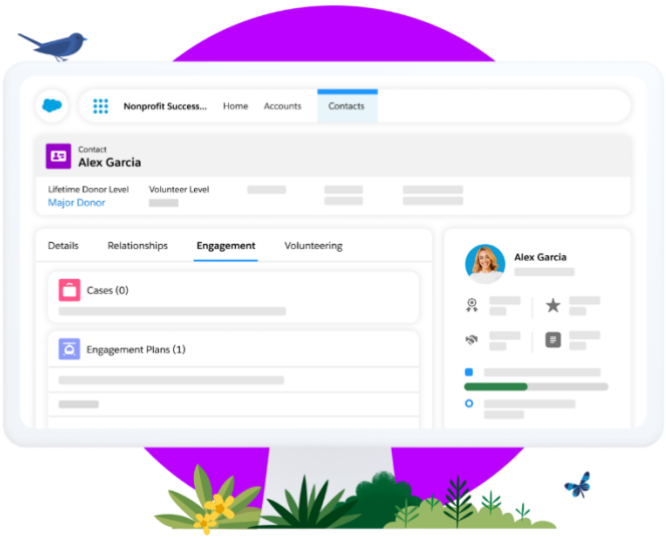

Рисунок 1.5. Salesforce Nonprofit Success Pack

Крім Microsoft Access, найпоширенішим способом ведення обліку гуманітарної допомоги є використання Microsoft Excel або Google Sheets.

Переваги:

- простота використання;
- доступність;
- низький поріг входження.

Недоліки:

- відсутність централізованого контролю;
- високий ризик помилок;
- складність масштабування;
- обмежені можливості багатокористувацької роботи.

ERP та CRM-системи забезпечують широкий функціонал для управління ресурсами, проте:

- є складними у впровадженні;
- потребують значних фінансових витрат;
- містять надлишковий функціонал;

- вимагають серверної інфраструктури.

Volonter.ua / Razom. Вітчизняні платформи для координації волонтерства. Орієнтовані переважно на зв'язок між волонтерами і потребуючими, не мають функціоналу складського обліку і управління запасами.

1.4.2. Порівняльний аналіз аналогів

Для наочного порівняння аналогів за ключовими критеріями складено порівняльну таблицю (табл. 1.1).

Таблиця 1.1. Порівняльний аналіз програмних аналогів

Критерій	Kobo Toolbox	Salesforce NPSP	LogiSoft	MS Access	Розроблюваний застосунок
Офлайн-режим роботи	Частково	Ні	Так	Так	Так
Облік запасів / склад	Ні	Частково	Так	Так	Так
Реєстр отримувачів	Частково	Так	Ні	Частково	Так
Облік донорів і передач	Ні	Так	Ні	Частково	Так
Звітність для донорів	Частково	Так	Так	Частково	Так
Простота використання	Середня	Низька	Середня	Низька	Висока
Вартість впровадження	Безкоштовно	Висока	Середня	Низька	Безкоштовно
Адаптованість до укр. контексту	Низька	Низька	Низька	Середня	Висока

Як видно з табл. 1.1, жоден з розглянутих аналогів не поєднує в собі повноцінного офлайн-режиму, зручного інтерфейсу для нетехнічних користувачів, повного охоплення ланцюга обліку гуманітарної допомоги

(донор — склад — отримувач) та адаптованості до умов роботи в Україні. Розроблюваний застосунок покликаний усунути цей дефіцит.

1.4.3. Висновки з аналізу аналогів

На підставі проведеного аналізу можна сформулювати ключові характеристики, якими має володіти розроблюваний застосунок, щоб перевершити існуючі рішення або зайняти незаповнену нішу:

- повноцінна робота в офлайн-режимі з можливістю синхронізації даних при відновленні з'єднання.
- охоплення повного ланцюга обліку: від реєстрації донора і надходження допомоги до видачі конкретному отримувачу.
- зручний та інтуїтивно зрозумілий інтерфейс, розрахований на волонтерів без ІТ-освіти.
- вбудований генератор звітів із можливістю експорту в поширені формати.
- відсутність потреби у серверній інфраструктурі або хмарній підписці – локальна установка.
- локалізація відповідно до українських реалій і мовних уподобань.

1.5. Визначення вимог до розроблюваного програмного забезпечення

1.5.1. Обґрунтування вибору типу програмного застосунку

Для реалізації системи обліку і координації гуманітарної допомоги було обрано десктопний тип програмного застосунку.

Основними перевагами десктопного застосунку є:

- автономність роботи;
- можливість функціонування без доступу до Інтернету;
- висока швидкодія;

- локальне зберігання даних;
- простота розгортання;
- підвищений рівень контролю над інформацією.

Особливо важливим є забезпечення стабільної роботи системи в умовах:

- нестабільного інтернет-з’єднання;
- відключень електроенергії;
- польових умов роботи волонтерських центрів.

Десктопний застосунок дозволяє забезпечити швидке введення та обробку інформації навіть на малопотужних комп’ютерах.

1.5.2. Методологія збору вимог

Вимоги до застосунку сформульовано на основі трьох джерел: аналізу недоліків поточних підходів, аналізу прогалин у існуючих аналогах, а також вивчення типових операційних сценаріїв волонтерських організацій. Вимоги поділяються на функціональні (що система має робити) і нефункціональні (характеристики якості, з якими система має це робити).

1.5.3. Функціональні вимоги

Функціональні вимоги згруповано за функціональними модулями застосунку.

Модуль управління обліковими записами і ролями:

- система має підтримувати кілька ролей користувачів: адміністратор, координатор, оператор;
- кожна роль має мати визначений набір доступних операцій відповідно до принципу найменших привілеїв;
- адміністратор повинен мати можливість створювати, редагувати та деактивувати облікові записи.

Модуль обліку надходжень (прийому допомоги):

- система має забезпечувати реєстрацію нових надходжень із зазначенням донора, дати, переліку та кількості одиниць допомоги;
- кожна позиція допомоги має класифікуватися за категорією (продовольство, медикаменти, одяг, техніка тощо);
- має бути можливість прикріпити до запису скан або фото супровідного документа;
- надходження автоматично збільшують залишки на відповідному складі.

Модуль складського обліку:

- система має відображати поточні залишки по кожній категорії та найменуванню;
- мають підтримуватися операції переміщення між складами (якщо організація має кілька точок зберігання);
- система має фіксувати списання та псування з обов'язковою вказівкою причини;
- має формуватися автоматичне сповіщення при досягненні критичного мінімального залишку.

Модуль реєстру отримувачів:

- система має вести базу даних отримувачів (фізичних осіб або організацій) з основними ідентифікаційними даними;
- для кожного отримувача має зберігатися повна історія отриманої допомоги;
- система має сигналізувати про повторне звернення отримувача з можливістю управління частотою отримання;
- має підтримуватися пошук і фільтрація реєстру за різними критеріями.

Модуль видачі та розподілу допомоги:

- система має реєструвати факт видачі допомоги конкретному отримувачеві з вибором з реєстру або внесенням нового запису;
- видача автоматично зменшує відповідні залишки на складі;

- має підтримуватися пакетний розподіл при масових видачах (наприклад, набір для ВПО).

Модуль звітності та аналітики:

- система має формувати стандартні звіти: рух ресурсів за період, залишки, статистика отримувачів, звіт для донора;
- звіти мають експортуватися у формати PDF та XLSX;
- на головній панелі (dashboard) мають відображатися ключові показники в графічному вигляді.

1.5.4. Нефункціональні вимоги

Нефункціональні вимоги до програмного застосунку представлені в табл. 1.2.

1.5.5. Обмеження та припущення

При розробці застосунку враховуються такі обмеження:

- застосунок розробляється як однокористувацька або локально-мережева система без хмарної синхронізації у першій версії;
- цільова платформа – операційна система Windows (10/11), версії для macOS та Linux не входять до обсягу поточної роботи;
- мобільна версія застосунку не розробляється у рамках кваліфікаційної роботи;
- інтеграція із зовнішніми державними реєстрами та API не передбачена у першій версії.

1.6. Аналіз та обґрунтування вибору технологій розробки

Для реалізації програмного забезпечення було обрано стек технологій Python + PyQt6 + SQLite + SQLAlchemy.

Таблиця 1.2. Нефункціональні вимоги до застосунку

Категорія	Атрибут якості	Вимога
Доступність	Офлайн-режим	Застосунок має повноцінно функціонувати без підключення до мережі Інтернет
Продуктивність	Швидкість відгуку	Час виконання будь-якої операції пошуку або збереження не повинен перевищувати 2 секунди
Зручність використання	Навчуваність	Новий користувач має опанувати базові операції без спеціального навчання не більш ніж за 30 хвилин
Надійність	Захист від втрати даних	Застосунок має автоматично виконувати резервне копіювання бази даних за розкладом
Безпека	Аутентифікація	Доступ до застосунку здійснюється виключно після введення облікових даних
Безпека	Розмежування доступу	Кожна роль має доступ лише до дозволених функцій і даних
Переносність	Сумісність з ОС	Застосунок має працювати на Windows 10/11 без додаткових вимог до серверної інфраструктури
Супроводжуваність	Розширюваність	Архітектура застосунку має допускати додавання нових модулів без переписування існуючого коду
Локалізація	Мова інтерфейсу	Інтерфейс застосунку має бути виконаний українською мовою

1. Python

Python є сучасною високорівневою мовою програмування, яка має:

- простий синтаксис;
- велику кількість бібліотек;
- підтримку кросплатформної розробки;
- активну спільноту розробників.

Перевагами Python є:

- швидкість розробки;
- підтримка об’єктно-орієнтованого програмування;
- простота інтеграції з базами даних.

2. PyQt6

PyQt6 використовується для створення графічного інтерфейсу користувача.

Основні переваги:

- сучасні GUI-компоненти;
- підтримка багатовіконного інтерфейсу;
- кросплатформеність;
- підтримка MVC-підходу.

3. SQLite

SQLite є вбудованою реляційною системою управління базами даних.

Переваги SQLite:

- відсутність необхідності окремого сервера;
- простота резервного копіювання;
- компактність;
- достатня продуктивність для локальних систем.

4. SQLAlchemy

SQLAlchemy використовується як ORM-засіб взаємодії з базою даних.

Основні переваги:

- абстрагування SQL-запитів;
- підтримка ORM-моделей;
- підвищення читабельності коду;
- спрощення підтримки проєкту.

1.7. Постановка задачі дослідження

Метою кваліфікаційної роботи є розробка десктопного застосунку для обліку і координації гуманітарної допомоги, який забезпечить автоматизацію процесів реєстрації, зберігання, розподілу та моніторингу гуманітарних ресурсів.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- провести аналіз предметної області;

- дослідити існуючі програмні рішення;
- визначити функціональні та нефункціональні вимоги до системи;
- спроектувати архітектуру програмного застосунку;
- розробити структуру бази даних;
- реалізувати графічний інтерфейс користувача;
- реалізувати механізми обліку гуманітарної допомоги;
- забезпечити взаємодію між структурними елементами системи;
- провести тестування програмного забезпечення.

Висновки до розділу

У першому розділі проведено комплексний аналіз предметної галузі, що дозволив встановити наступне. Гуманітарна діяльність в умовах воєнного часу потребує надійного, автономного та зручного інструменту обліку і координації. Поточні підходи – електронні таблиці, паперові журнали та месенджери – не забезпечують необхідного рівня структурованості, прозорості і швидкості роботи. Аналіз програмних аналогів показав, що існуючі рішення або не мають офлайн-можливостей, або охоплюють лише частину необхідного функціоналу, або є надто складними і дорогими для малих організацій.

У результаті аналізу предметної області було досліджено основні процеси обліку та координації гуманітарної допомоги, визначено ключові проблеми існуючих підходів до організації обліку ресурсів та обґрунтовано необхідність створення спеціалізованого програмного забезпечення.

Було проведено аналіз існуючих програмних рішень та визначено їх основні недоліки, серед яких:

- складність використання;
- залежність від мережевої інфраструктури;
- обмежені можливості локальної роботи;
- складність масштабування.

На основі виявлених недоліків і прогалин сформульовано функціональні та нефункціональні вимоги до розроблюваного десктопного застосунку. До ключових функціональних вимог належать: повний ланцюг обліку від надходження до видачі, реєстр отримувачів із захистом від дублювання, складський облік із сигналізацією критичних залишків і генерація звітів у стандартних форматах. Серед нефункціональних вимог пріоритетними є автономна робота без інтернету, висока зручність для нетехнічних користувачів та надійність збереження даних. Сформульовані вимоги складатимуть основу для проєктних рішень, що описуються в наступному розділі.

На основі проведеного аналізу обґрунтовано вибір десктопного типу застосунку та стеку технологій Python, PyQt6, SQLite і SQLAlchemy, що дозволяє забезпечити ефективну реалізацію системи обліку і координації гуманітарної допомоги.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ДЕСКТОПНОГО ЗАСТОСУНКУ ДЛЯ ОБЛІКУ І КООРДИНАЦІЇ ГУМАНІТАРНОЇ ДОПОМОГИ

2.1. Аналіз вимог до програмного застосунку

Розробка програмного застосунку для обліку і координації гуманітарної допомоги потребує врахування особливостей процесів приймання, реєстрації, розподілу та моніторингу гуманітарних ресурсів. Основною метою системи є автоматизація процесів ведення обліку гуманітарної допомоги, підвищення прозорості розподілу ресурсів та спрощення координації між волонтерами, складами та отримувачами допомоги.

До функціональних вимог системи належать:

- реєстрація гуманітарних вантажів;
- облік складів та залишків ресурсів;
- ведення бази отримувачів допомоги;
- формування заявок на видачу допомоги;
- контроль руху гуманітарних ресурсів;
- створення звітів щодо надходжень та витрат;
- авторизація користувачів та розмежування прав доступу.

Нефункціональні вимоги включають:

- зручний графічний інтерфейс користувача;
- швидкодію системи при роботі з локальною базою даних;
- надійність збереження інформації;
- можливість подальшого масштабування системи;
- кросплатформеність програмного забезпечення.

Для реалізації програмного застосунку було обрано мову програмування Python, фреймворк PyQt6 для створення графічного інтерфейсу користувача, SQLite як локальну систему управління базами даних та SQLAlchemy для реалізації ORM-рівня взаємодії з базою даних.

2.2. Концептуальне проєктування архітектури програмного застосунку

При проєктуванні системи було використано багаторівневу архітектуру, що забезпечує розділення логіки інтерфейсу, бізнес-логіки та рівня доступу до даних.

Архітектура системи включає такі основні компоненти:

1. Рівень представлення (Presentation Layer);
2. Рівень бізнес-логіки (Business Logic Layer);
3. Рівень доступу до даних (Data Access Layer);
4. База даних SQLite.

Рівень представлення реалізовано засобами PyQt6 та містить вікна, форми, таблиці та діалогові елементи інтерфейсу користувача.

Рівень бізнес-логіки забезпечує:

- перевірку коректності введених даних;
- обробку операцій над гуманітарними ресурсами;
- координацію процесів розподілу допомоги;
- формування звітів.

Рівень доступу до даних реалізовано за допомогою SQLAlchemy ORM. Це дозволяє абстрагувати роботу з базою даних та спростити підтримку програмного коду.

2.3. Обґрунтування архітектурного підходу

Вибір архітектурного підходу для десктопного застосунку є одним із ключових проєктних рішень, що безпосередньо впливає на зручність супроводу, масштабованість і надійність кінцевого продукту. Оскільки розроблюваний застосунок має працювати автономно, зберігати дані локально і забезпечувати чітке розмежування ролей користувачів, вибір архітектури визначався такими ключовими міркуваннями.

У якості базової архітектурної концепції обрано підхід MVC (Model-View-Controller). Ця трирівнева модель поділяє код застосунку на три компоненти: модель (Model) відповідає за дані та бізнес-логіку, відображення (View) – за інтерфейс користувача, а контролер (Controller) — за обробку вхідних подій і координацію між моделлю і відображенням. Такий поділ відповідальностей дозволяє незалежно розробляти та тестувати кожен з компонентів, спрощує подальше додавання функціоналу без порушення існуючого коду.

Для зберігання даних обрано реляційну базу даних SQLite. Цей вибір обумовлений низкою переваг: SQLite є безсерверною СКБД, яка зберігає всю базу даних в одному файлі, не потребує окремого серверного процесу і ліцензійних відрахувань, підтримує повноцінний діалект SQL і забезпечує транзакційну цілісність даних. Для застосунку, що орієнтований на роботу без постійного підключення до мережі, SQLite є оптимальним вибором.

Шар доступу до даних реалізовано за патерном Repository. Кожна сутність предметної галузі (донори, отримувачі, запаси, надходження тощо) має свій клас-репозиторій, що інкапсулює всі SQL-запити для роботи з цією сутністю. Це дозволяє легко замінити СКБД у майбутньому, не зачіпаючи бізнес-логіку або рівень відображення.

Бізнес-логіка застосунку зосереджена у сервісному шарі (Service Layer). Сервіси реалізують складні операції, що зачіпають кілька репозиторіїв одночасно: наприклад, реєстрація надходження одночасно записує транзакцію у таблицю incoming і оновлює залишки у таблиці stock. Такий підхід гарантує атомарність операцій і спрощує тестування бізнес-логіки ізольовано від інтерфейсу.

Рівень представлення побудовано на основі системи вікон і форм. Головне вікно містить навігаційну панель та область відображення контенту, що змінюється залежно від обраного модуля. Кожен функціональний модуль (облік надходжень, реєстр отримувачів, склад тощо) реалізовано як окремий екранний компонент, що підключається до відповідного контролера.

2.4. Вибір технологічного стеку

Вибір технологічного стеку для реалізації застосунку проводився із врахуванням визначених нефункціональних вимог: автономна робота, орієнтованість на Windows 10/11, відсутність потреби у серверній інфраструктурі, зручність для кінцевого користувача. Було розглянуто три альтернативи (табл. 2.1).

Таблиця 2.1. Порівняння фреймворків для розробки десктопного застосунку

Критерій	Python + PyQt6	C# + WPF	Electron (JS)
Крос-платформність	Висока	Тільки Windows	Висока
Швидкість розробки	Висока	Середня	Висока
Продуктивність	Середня	Висока	Низька
Розмір дистрибутива	~50 МБ	~30 МБ	~150 МБ
Екосистема бібліотек	Відмінна	Хороша	Відмінна
Відкритий вихідний код	Так	Частково	Так

За результатами порівняльного аналізу обрано стек Python + PyQt6 + SQLite. Мова програмування Python забезпечує висока швидкість розробки, читабельний синтаксис і широку екосистему бібліотек для роботи з даними та генерації звітів. Фреймворк PyQt6 – Python-прив'язка до Qt6 – надає потужний і гнучкий набір компонентів для побудови десктопних GUI-застосунків, підтримує стилізацію інтерфейсу через QSS (Qt Style Sheets) і забезпечує крос-платформну сумісність.

Додаткові бібліотеки, що використовуються в проєкті:

- SQLAlchemy – ORM (Object-Relational Mapping) для зручної роботи з базою даних та реалізації патерну Repository;

- ReportLab – генерація PDF-звітів;
- openpyxl – експорт даних у формат XLSX;
- Werkzeug – хешування паролів користувачів;
- PyInstaller – пакування застосунку в автономний виконуваний файл (.exe) для Windows;
- pytest – фреймворк для автоматизованого тестування.

2.3. Проєктування бази даних

База даних застосунку спроєктована відповідно до принципів нормалізації (до третьої нормальної форми) та вимог до функціональності. Схема включає 13 таблиць, що охоплюють усі сутності предметної галузі (табл. 2.2).

Зв'язки між таблицями реалізовано через зовнішні ключі (FOREIGN KEY) із каскадними операціями. Для таблиць, що часто використовуються у пошуку та фільтрації (recipients, items, stock), створено індекси на часто використовуваних полях. Цілісність даних забезпечується на рівні СКБД через обмеження NOT NULL, UNIQUE та CHECK.

Облік руху матеріальних цінностей побудовано на принципі транзакційного журналу: таблиці incoming, outgoing, transfers та writeoffs зберігають окремі факти господарських операцій, а таблиця stock містить поточні підсумкові залишки, що обчислюються та оновлюються при кожній транзакції. Такий підхід гарантує повну простежуваність руху кожної одиниці допомоги.

Для запобігання втраті даних у разі помилок транзакцій усі операції, що змінюють кілька таблиць одночасно, виконуються в рамках єдиної транзакції SQLite. У разі виникнення помилки транзакція відкочується повністю, зберігаючи цілісність бази даних.

Таблиця 2.2. Таблиці бази даних застосунку

Назва таблиці	Призначення
users	Облікові записи користувачів (id, логін, хеш паролю, роль, статус)
donors	Реєстр донорів (фізичні та юридичні особи, контактні дані)
recipients	Реєстр отримувачів (ідентифікаційні дані, категорія потребуючого, дата реєстрації)
warehouses	Склади організації (назва, адреса, відповідальна особа)
item_categories	Довідник категорій допомоги (продовольство, медикаменти, одяг тощо)
items	Номенклатура позицій допомоги (назва, категорія, одиниця виміру, мінімальний залишок)
stock	Поточні залишки (склад, позиція, кількість)
incoming	Надходження допомоги (донор, склад, дата, позиції, документ)
outgoing	Видача допомоги (отримувач, склад, дата, позиції, оператор)
transfers	Переміщення між складами (джерело, призначення, позиції, дата)
writeoffs	Списання (позиція, кількість, причина, дата)
audit_log	Журнал дій користувачів (хто, що, коли)

2.4. Багаторівнева архітектура застосунку

Архітектура застосунку організована у чотири рівні, кожен з яких відповідає за чітко визначені функції та взаємодіє з сусідніми рівнями через добре визначені інтерфейси.

Рівень представлення (Presentation Layer) включає головне вікно застосунку (MainWindow), що реалізовує навігаційний фрейм з бічним меню і центральною областю відображення контенту. Кожен функціональний розділ застосунку представлено окремим QWidget-класом: DashboardWidget, IncomingWidget, OutgoingWidget, StockWidget, RecipientsWidget, DonorsWidget, ReportsWidget, UsersWidget, SettingsWidget.

Вікно автентифікації (LoginDialog) є незалежним компонентом і не входить до головного вікна.

Рівень контролерів (Controller Layer) містить окремий клас контролера для кожного функціонального модуля. Контролери приймають події від користувача через сигнали PyQt, викликають відповідні методи сервісного шару і передають результати назад у компоненти відображення. Такий підхід дозволяє тестувати бізнес-логіку незалежно від конкретної реалізації GUI.

Сервісний рівень (Service Layer) реалізує операції, що відповідають реальним бізнес-процесам організації: реєстрація надходження, видача допомоги, переміщення між складами, формування звіту. Кожен сервіс є ізольованою одиницею, яка може викликати кілька репозиторіїв і виконувати складну логіку перевірок та обчислень. Сервіси є stateless – вони не зберігають стан між викликами, що спрощує тестування.

Рівень доступу до даних (Data Access Layer) включає базовий клас BaseRepository та спеціалізовані репозиторії: UserRepository, DonorRepository, RecipientRepository, ItemRepository, StockRepository, IncomingRepository, OutgoingRepository, TransferRepository, WriteoffRepository. Кожен репозиторій містить CRUD-методи та специфічні запити для своєї сутності. Взаємодія з SQLite відбувається через SQLAlchemy ORM.

Архітектурна схема системи представлена на рис. 2.1.

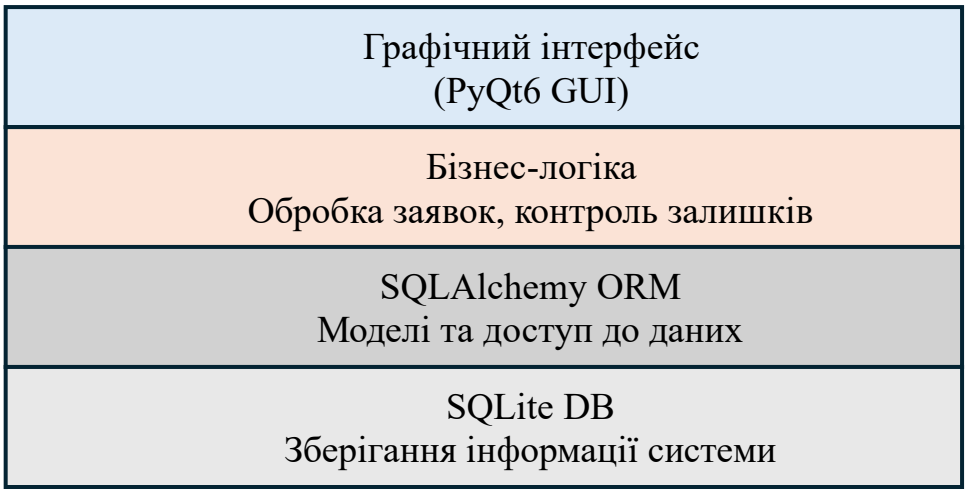

Рисунок 2.1. Схема архітектурних рівнів

2.5. Проєктування бази даних

База даних системи призначена для збереження інформації про гуманітарні вантажі, склади, користувачів, заявки та операції видачі допомоги.

Основними сутностями системи є:

- User – користувач системи;
- Warehouse – склад;
- AidItem – гуманітарний ресурс;
- Recipient – отримувач допомоги;
- Request – заявка на отримання допомоги;
- Distribution – операція видачі допомоги.

Між сутностями встановлено відповідні зв’язки типу «один-до-багатьох» та «багато-до-багатьох».

ER-діаграма бази даних представлена на рис. 2.2.

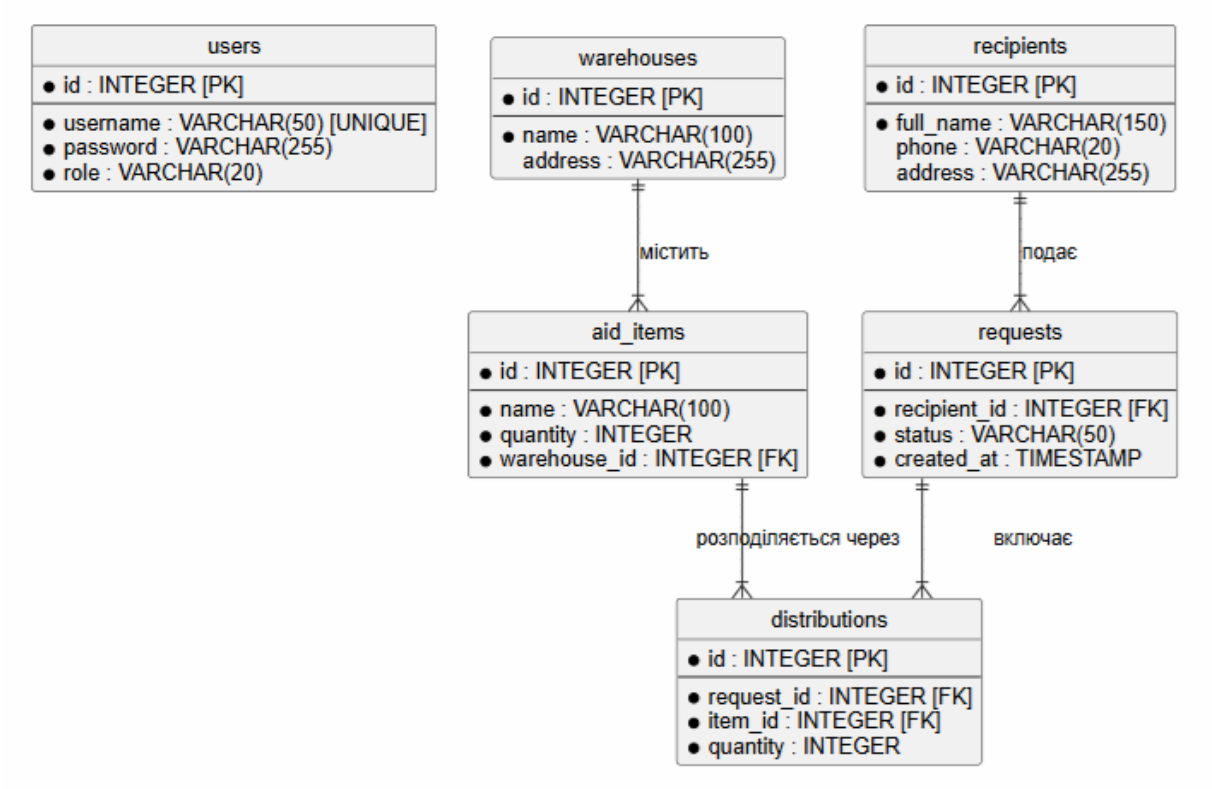

Рисунок 2.2. ER-діаграма бази даних

2.6. Проєктування графічного інтерфейсу користувача

Інтерфейс користувача проєктувався з урахуванням принципів простоти, зрозумілості та мінімізації часу виконання операцій.

Основні компоненти інтерфейсу:

- вікно авторизації;
- головне меню системи;
- модуль управління складами;
- модуль обліку гуманітарної допомоги;
- модуль заявок;
- модуль формування звітів.

Навігація між модулями реалізується через головне меню та систему вкладок.

Структура GUI застосунку представлена на рис. 2.3.

```
Головне вікно
│
├── Авторизація
├── Панель навігації
│   ├── Склади
│   ├── Гуманітарна допомога
│   ├── Отримувачі
│   ├── Заявки
│   └── Звіти
│
└── Робоча область
    ├── Таблиці
    ├── Форми введення
    └── Діалогові вікна
```

Рисунок 2.3. Структура інтерфейсу користувача

2.7. UML-діаграми

Діаграма варіантів використання (Use Case Diagram) представлена на рис. 2.4.

Рисунок 2.4. Діаграма варіантів використання

Діаграма класів представлена на рис. 2.5.

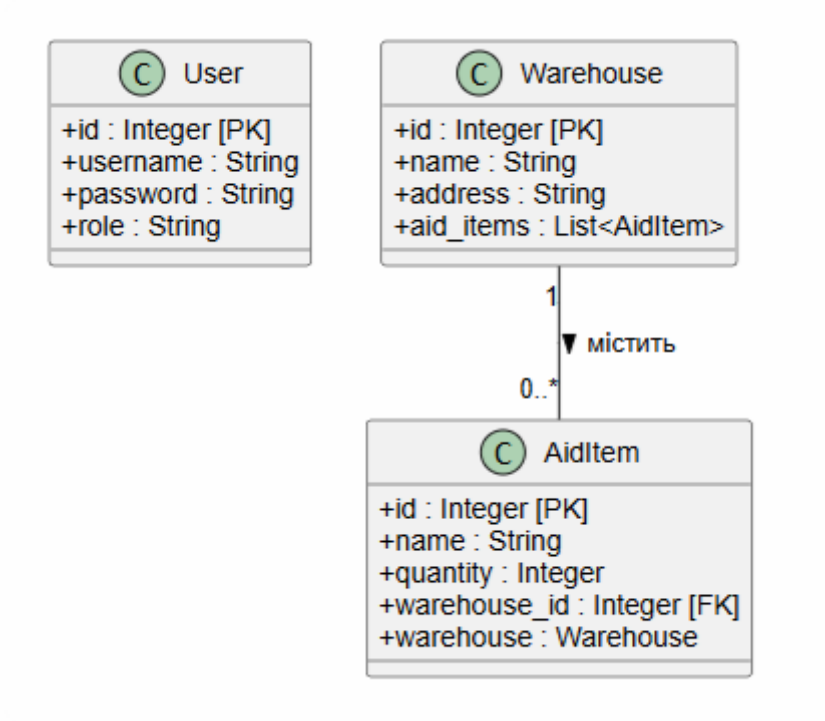

Рисунок 2.5. Діаграма класів

Висновки до розділу

У другому розділі обґрунтовано вибір архітектурних рішень і технологій для реалізації десктопного застосунку обліку і координації гуманітарної допомоги. Обрана архітектура MVC із чітким поділом на рівні представлення, контролерів, сервісів і доступу до даних забезпечує масштабованість і зручність супроводу застосунку.

У результаті проєктування програмного застосунку було визначено функціональні та нефункціональні вимоги системи, розроблено архітектуру програмного забезпечення, структуру бази даних та моделі взаємодії між компонентами системи.

Технологічний стек Python + PyQt6 + SQLite відповідає вимогам автономної роботи, зручності для кінцевого користувача і безкоштовного використання. Спроєктована реляційна схема бази даних із 13 таблицями охоплює всі сутності предметної галузі і забезпечує повну простежуваність руху матеріальних цінностей через журнал транзакцій.

Запропонована багаторівнева архітектура забезпечує модульність, масштабованість та зручність супроводу програмного забезпечення.

Використання Python, PyQt6, SQLite та SQLAlchemy дозволяє реалізувати ефективний десктопний застосунок для обліку і координації гуманітарної допомоги.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ДЛЯ ОБЛІКУ І КООРДИНАЦІЇ ГУМАНІТАРНОЇ ДОПОМОГИ

3.1. Загальна структура програмного застосунку

Програмний застосунок для обліку і координації гуманітарної допомоги реалізовано у вигляді багатомодульної десктопної системи на базі мови програмування Python із використанням бібліотеки PyQt6 для створення графічного інтерфейсу користувача, SQLite для зберігання даних та SQLAlchemy як ORM-рівня взаємодії з базою даних.

Основною метою побудови структури програмного забезпечення було забезпечення:

- модульності;
- масштабованості;
- простоти супроводу;
- повторного використання компонентів;
- розмежування логіки інтерфейсу та бізнес-логіки.

Архітектура програмного застосунку складається з таких основних модулів:

- модуль графічного інтерфейсу;
- модуль бізнес-логіки;
- модуль доступу до даних;
- модуль формування звітів;
- модуль авторизації користувачів.

3.2. Структура каталогу проєкту

Файлова структура проєкту організована відповідно до принципів чистої архітектури та прийнятих конвенцій Python-проєктів.

Кореневий каталог містить файл конфігурації оточення (.env), файл залежностей (requirements.txt) та скрипт збірки (build.py). Основний код

застосунку розміщено у каталозі src/ із поділом на піддиректорії відповідно до архітектурних рівнів (табл. 3.1).

Таблиця 3.1. Структура проєкту

Директорія / файл	Призначення
main.py	Точка входу застосунку: ініціалізація QApplication, автентифікація, запуск головного вікна
database/	Модуль бази даних: engine.py (підключення SQLite), models.py (SQLAlchemy-моделі), migrations/
repositories/	Репозиторії для кожної сутності: base.py, users.py, donors.py, recipients.py, stock.py та ін.
services/	Сервісний шар: incoming_service.py, outgoing_service.py, stock_service.py, report_service.py та ін.
controllers/	Контролери модулів: dashboard_controller.py, incoming_controller.py та ін.
views/	Компоненти GUI: main_window.py, login_dialog.py, widgets/ (один модуль на розділ)
resources/	Ресурси: icons/ (іконки), styles/ (QSS-стилі), templates/ (шаблони звітів)
utils/	Утилітарні модулі: validators.py, formatters.py, backup.py, logger.py
tests/	Автоматизовані тести: unit/, integration/ (pytest-структура)

Такий поділ проєкту відповідає принципу єдиної відповідальності (Single Responsibility Principle): кожен модуль і директорія мають чітко визначену роль і не змішують логіку різних рівнів. Це суттєво спрощує навігацію по коду і дозволяє залучати нових розробників без тривалого занурення у деталі реалізації.

Структура файлів програмного проєкту представлена на рис. 3.1.

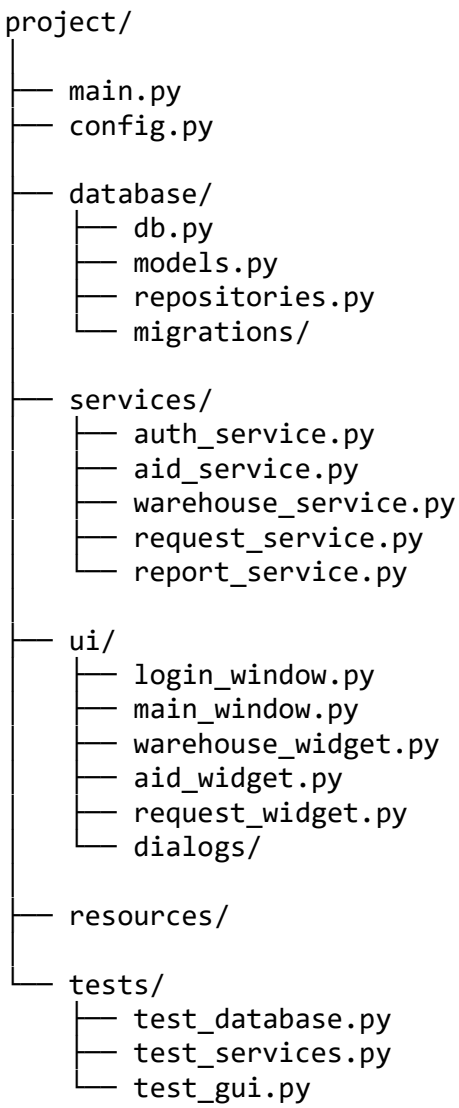

Рисунок 3.1. Структура файлів програмного проєкту

Головним файлом запуску системи є `main.py`, який ініціалізує графічний інтерфейс та створює основні компоненти застосунку.

3.3. Реалізація бази даних та ORM-моделей

Для реалізації сховища даних було використано SQLite. Основною перевагою цієї СУБД є відсутність необхідності встановлення окремого серверного програмного забезпечення.

Взаємодія між застосунком та базою даних реалізована за допомогою SQLAlchemy ORM, що дозволяє працювати з об’єктами Python замість прямого написання SQL-запитів.

Реалізація підключення до бази даних здійснена наступним чином:

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker

DATABASE_URL = "sqlite:///humanitarian.db"

engine = create_engine(DATABASE_URL)
SessionLocal = sessionmaker(bind=engine)
```

Приклад ORM-моделі гуманітарного ресурсу:

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from database.db import Base

class AidItem(Base):
    __tablename__ = "aid_items"

    id = Column(Integer, primary_key=True)
    name = Column(String, nullable=False)
    quantity = Column(Integer, default=0)
    warehouse_id = Column(Integer, ForeignKey("warehouses.id"))

    warehouse = relationship("Warehouse")
```

ORM-моделі забезпечують:

- автоматичне формування таблиць;
- спрощення взаємодії з базою даних;
- підтримку зв’язків між сутностями;
- підвищення читабельності програмного коду.

3.4. Реалізація графічного інтерфейсу користувача

Графічний інтерфейс реалізовано засобами PyQt6. Основною перевагою використання PyQt6 є можливість створення сучасних кросплатформних GUI-застосунків.

Інтерфейс програми складається з:

- головного вікна;

- системи вкладок;
- таблиць відображення даних;
- форм введення;
- діалогових вікон.

Структура головного вікна представлена на рис. 3.2.

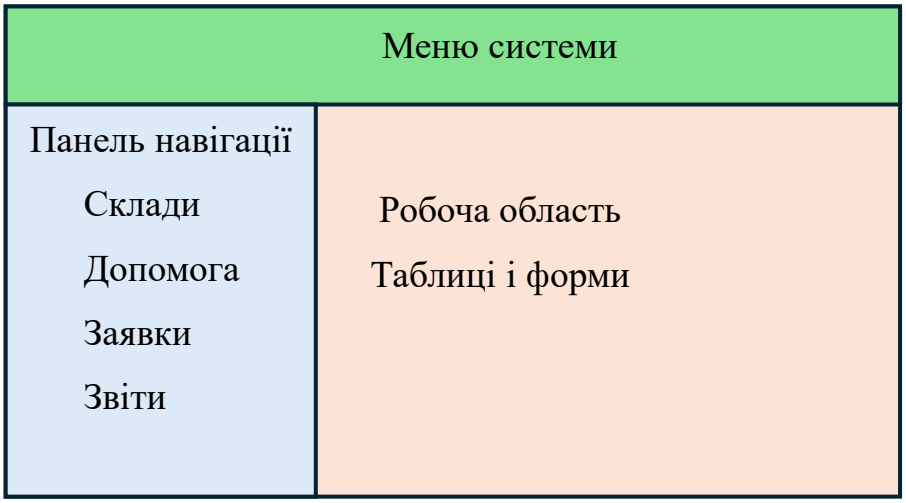

Рисунок 3.2. Структура головного вікна застосунку

Приклад створення головного вікна:

```
from PyQt6.QtWidgets import QMainWindow, QWidget, QVBoxLayout

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Система гуманітарної допомоги")

        central_widget = QWidget()
        layout = QVBoxLayout()

        central_widget.setLayout(layout)
        self.setCentralWidget(central_widget)
```

Для відображення інформації використовуються компоненти:

- QTableWidget;

- QLineEdit;
- QComboBox;
- QPushButton;
- QDialog.

3.5. Реалізація взаємодії між структурними елементами системи

Взаємодія між модулями програмного застосунку організована відповідно до принципів багаторівневої архітектури.

Послідовність взаємодії компонентів системи: 1. Користувач взаємодіє з GUI. 2. GUI передає дані до сервісного шару. 3. Бізнес-логіка обробляє запит. 4. ORM-моделі взаємодіють із базою даних. 5. Результат повертається до інтерфейсу.

Схема взаємодії компонентів представлена на рис. 3.3.

Діаграма послідовності процесу видачі допомоги представлена на рис. 3.4.

Використання сервісного шару дозволяє:

- уникнути дублювання коду;
- централізувати бізнес-логіку;
- спростити тестування системи;
- забезпечити незалежність GUI від бази даних.

Рисунок 3.3. Схема взаємодії компонентів

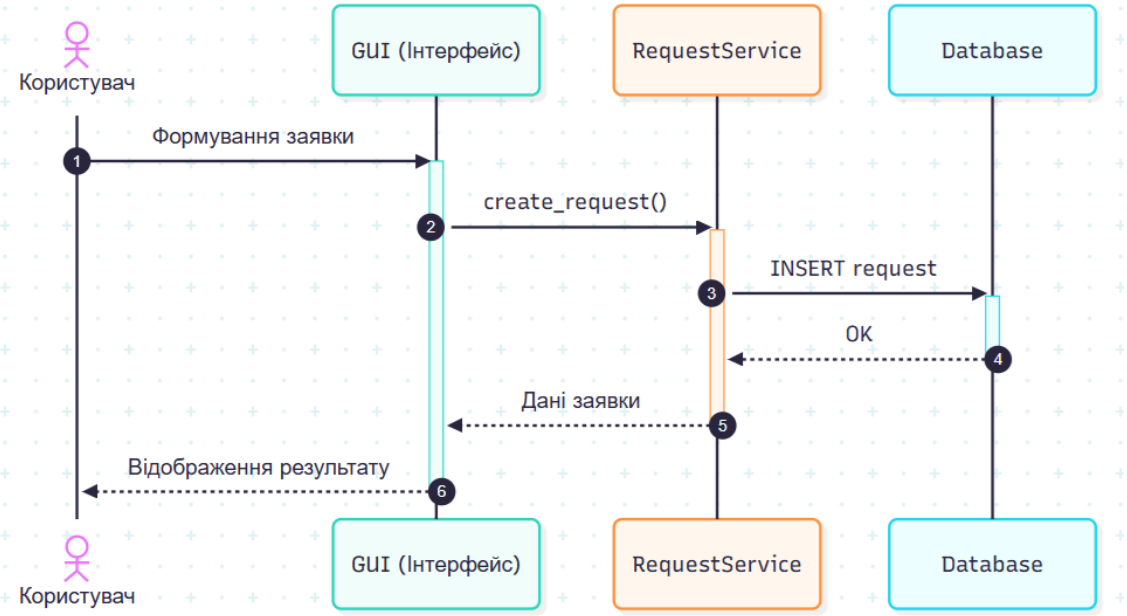

Рисунок 3.4. Діаграма послідовності

3.6. Реалізація механізму авторизації користувачів

Для забезпечення безпеки системи реалізовано механізм авторизації користувачів.

Кожен користувач має:

- логін;
- пароль;
- роль у системі.

Передбачено такі ролі:

- адміністратор;
- оператор;
- волонтер.

Алгоритм авторизації представлено на рис. 3.5.

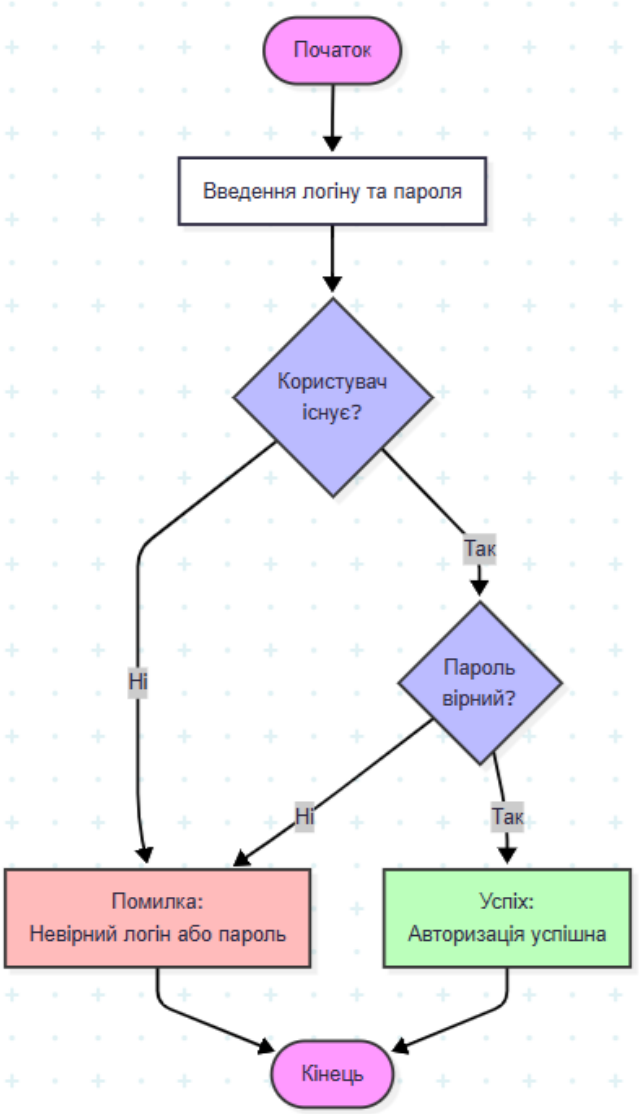

Рисунок 3.5. Алгоритм авторизації користувачів

Приклад перевірки користувача:

```
def authenticate(username, password):
    user = session.query(User).filter_by(username=username).first()

    if user and user.password == password:
        return True

    return False
```

3.7. Реалізація модулів застосунку

Реалізація застосунку виконана у розрізі функціональних модулів. Кожен модуль включає компонент представлення (View), контролер і один або кілька сервісів.

Модуль автентифікації. Вікно входу LoginDialog реалізовано як модальний діалог, що відображається до завантаження головного вікна. Після введення логіну і паролю контролер викликає UserService.authenticate(), який отримує хеш паролю з бази даних через UserRepository і порівнює його з введеним паролем методом bcrypt-верифікації (через бібліотеку werkzeug.security). У разі успішної автентифікації повертається об'єкт сесії з ідентифікатором і роллю користувача, що передається до MainWindow. Три невдалі спроби входу призводять до тимчасового блокування облікового запису.

Модуль обліку надходжень. Форма реєстрації нового надходження реалізована як покрокова процедура: спочатку вибирається або реєструється новий донор, далі вказуються дата і склад, потім у таблицю позицій додаються найменування з кількостями. Після збереження IncomingService.register_incoming() виконує транзакцію: зберігає запис у таблицю incoming, зберігає деталі у incoming_items і викликає StockService.add_to_stock() для оновлення залишків. Усі три операції виконуються в єдиній транзакції SQLite.

Модуль складського обліку. StockWidget відображає поточні залишки у вигляді таблиці з можливістю фільтрації за складом, категорією і найменуванням. Позиції з залишком нижче мінімального виділяються кольором (жовтий або червоний) за допомогою делегата QStyledItemDelegate. Окремі вкладки надають доступ до операцій переміщення між складами і списання з вказівкою причини. При збереженні операції сервісний шар виконує відповідні зміни у таблицях stock, transfers або writeoffs.

Модуль реєстру отримувачів. RecipientWidget реалізує повнотекстовий пошук і фільтрацію реєстру з пагінацією результатів. Картка отримувача включає вкладку особистих даних та вкладку історії отримань, де відображаються всі факти видачі з деталями. При реєстрації нового отримувача у формі виконується валідація обов'язкових полів і перевірка на дублікати за номером ідентифікаційного документа. Якщо отримувач вже зареєстрований і з моменту останнього отримання пройшло менше встановленого мінімального інтервалу, система відображає попередження координатору.

Модуль видачі допомоги. Форма видачі дозволяє вибрати отримувача з реєстру через пошукове поле, вказати склад і сформувати набір позицій до видачі. Передбачено можливість вибору преднастроєного «пакету допомоги» — шаблону з типовим набором позицій і кількостей (наприклад, «Стандартний набір ВПО»). OutgoingService.register_outgoing() перевіряє наявність достатніх залишків перед збереженням і відмовляє в операції, якщо залишків недостатньо, повертаючи відповідне повідомлення.

Модуль звітності. ReportService реалізує генерацію чотирьох типів звітів: звіт про рух ресурсів за обраний період, звіт про поточні залишки, статистика отримувачів та звіт для донора. Для кожного типу звіту реалізовано окремий клас-генератор. PDF-звіти генеруються з використанням ReportLab із фірмовим стилем оформлення, XLSX-звіти — через openpyxl із форматуванням таблиць і зведених даних. Dashboard відображає агреговані показники у вигляді карток зі статистикою та спрощених гістограм.

Модуль управління користувачами. Доступний лише для ролі «адміністратор». Надає можливість перегляду, створення, редагування облікових записів і зміни ролей користувачів. При створенні нового облікового запису генерується тимчасовий пароль, що передається адміністратору для передачі новому користувачу. Журнал дій (audit_log)

записується автоматично при кожній операції запису або видалення і доступний для перегляду адміністратору.

3.8. Реалізація формування звітів

Одним із функціональних модулів системи є модуль формування звітів.

Система підтримує:

- звіт про надходження гуманітарної допомоги;
- звіт про залишки на складах;
- звіт про видану допомогу;
- статистичні звіти.

Для формування звітів використовуються:

- SQLAlchemy-запити;
- генерація таблиць;
- експорт у PDF та Excel.

Приклад SQLAlchemy-запиту:

```
items = session.query(AidItem).filter(
    AidItem.quantity > 0
).all()
```

3.9. Реалізація інтерфейсу користувача

Інтерфейс застосунку розроблено з урахуванням вимоги щодо зручності для нетехнічних користувачів. Загальний стиль оформлення витримано в стриманих кольорах – синьо-сірий основний колір з акцентами на жовто-гарячому для попереджень і зеленому для підтверджень. Стилізація реалізована через єдиний файл стилів QSS (app_styles.qss), що дозволяє змінювати зовнішній вигляд застосунку без зміни коду.

Головне вікно MainWindow побудовано за принципом «бічна навігація + область контенту». Ліворуч розміщено вертикальну панель навігації з

іконками та підписами функціональних розділів. При виборі пункту меню у центральній частині вікна відображається відповідний QWidget-модуль через QStackedWidget. Верхня смуга відображає назву поточного розділу, ім'я та роль авторизованого користувача і кнопку виходу.

Таблиці даних реалізовано на основі QTableView із кастомними QAbstractTableModel-моделями. Такий підхід, на відміну від використання QTableWidget, забезпечує значно вищу продуктивність при відображенні великих обсягів даних (тисячі записів). Для стовпців із визначеними допустимими значеннями (категорія, статус, роль) реалізовано QComboBoxDelegate для редагування безпосередньо в таблиці. Усі таблиці підтримують сортування за заголовком стовпця і пошук за введеним текстом.

Форми введення даних валідуються в реальному часі: поля, що містять некоректні значення, підсвічуються червоною рамкою, а кнопка збереження блокується до усунення всіх помилок. Такий підхід унеможливлює збереження некоректних даних до бази і одразу надає користувачу зворотний зв'язок. Обов'язкові поля позначені зірочкою відповідно до загальноприйнятих конвенцій UI.

Для операцій, що потребують підтвердження (видалення запису, списання, зміна ролі користувача), реалізовано модальні діалоги підтвердження з чітким описом наслідків дії. Це запобігає випадковим деструктивним діям з боку оператора.

3.10. Резервне копіювання та безпека даних

Безпека даних забезпечується на кількох рівнях. По-перше, вся база даних SQLite зберігається у зашифрованому вигляді завдяки розширенню SQLCipher, що реалізує AES-256 шифрування на рівні файлу бази. Ключ шифрування генерується на основі паролю адміністратора при першому запуску і зберігається у захищеному сховищі Windows Credential Manager.

По-друге, усі паролі користувачів зберігаються у вигляді bcrypt-хешів із сіллю. Відновлення оригінального паролю з хешу є обчислювально неможливим, що захищає облікові дані навіть у разі компрометації файлу бази даних.

По-третє, модуль резервного копіювання (backup.py) реалізує автоматичне створення зашифрованих резервних копій бази даних за розкладом (щоденно, за замовчуванням). Резервні копії зберігаються в окремій директорії і ротуються – зберігаються останні 30 копій. Адміністратор може ініціювати резервне копіювання вручну та відновити базу з будь-якої резервної копії через відповідний діалог у налаштуваннях.

Модуль аудиту (audit_log) фіксує всі критичні операції: вхід і вихід користувача, створення, зміну та видалення записів, зміну налаштувань, генерацію звітів. Журнал аудиту доступний для перегляду лише адміністратору і не може бути видалений через інтерфейс застосунку.

3.11. Скріншоти розробленого застосунку

Екранні форми програми представлено на скріншотах.

В застосунку реалізовано процеси реєстрації, зберігання, розподілу та моніторингу гуманітарних ресурсів.

Розроблений модуль управління обліковими записами і ролями підтримує ролі адміністратора, координатора та оператора, при цьому кожна з них має певний набір доступних операцій відповідно до принципу найменших привілеїв. Лише адміністратор має можливість створювати, редагувати та деактивувати облікові записи.

Головним є модуль обліку надходжень (прийому допомоги), який забезпечує реєстрацію нових надходжень із зазначенням донора, дати, переліку та кількості одиниць допомоги, а кожна позиція допомоги має класифікуватися за категорією (продовольство, медикаменти, одяг, техніка тощо). Також є можливість прикріпляти до кожного запису скан або фото супровідного документа.

Форма замовлення на закупівлю представлена на рис. 3.6.

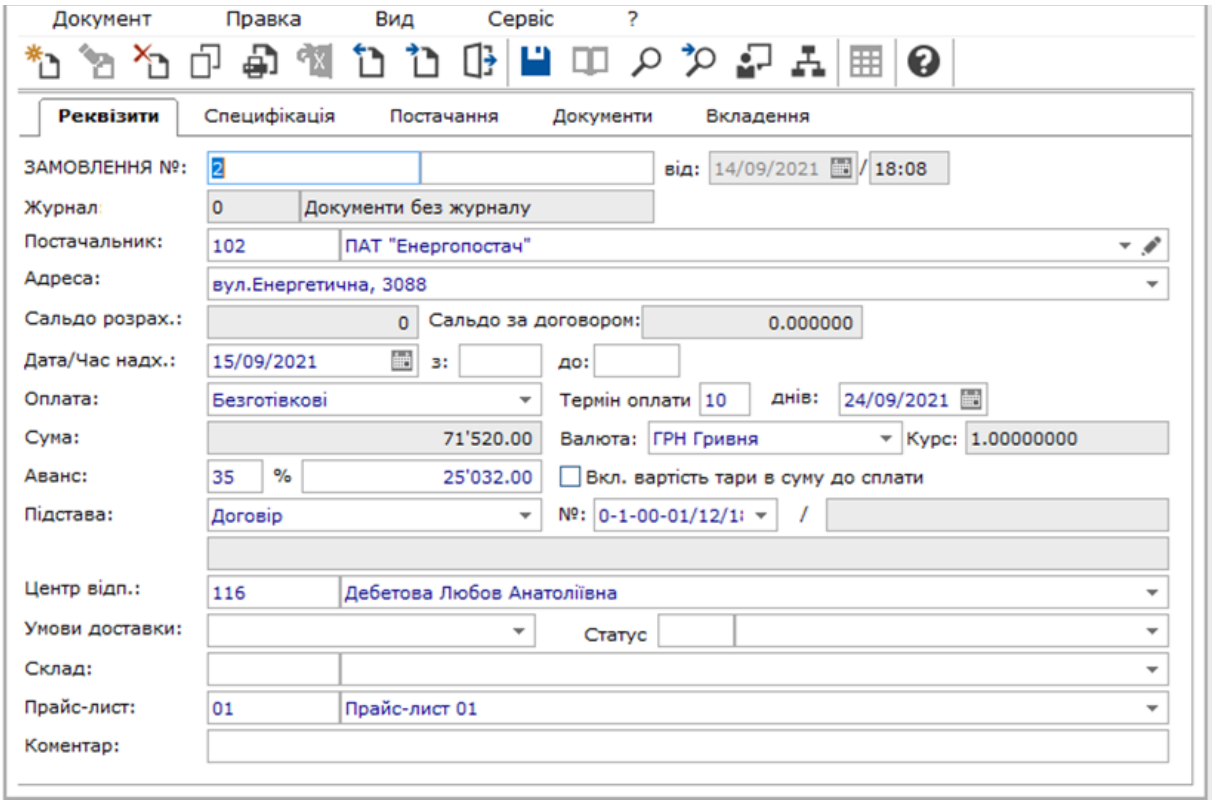

Рисунок 3.6. Форма замовлення на закупівлю

Форма плану закупівлі представлена на рис. 3.7.

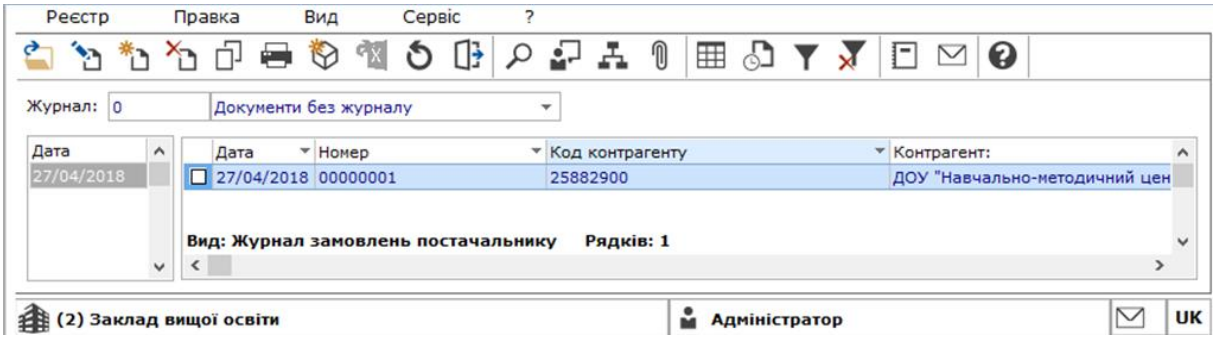

Рисунок 3.7. Форма плану закупівлі

Модуль складського обліку обробляє поточні залишки по кожній категорії та найменуванню, забезпечує операції переміщення між складами, якщо існують кілька точок зберігання. Модуль дозволяє фіксувати списання

та псування, а також автоматично формує сповіщення при досягненні критичного мінімального залишку.

Форма картки складського обліку представлена на рис. 3.8.

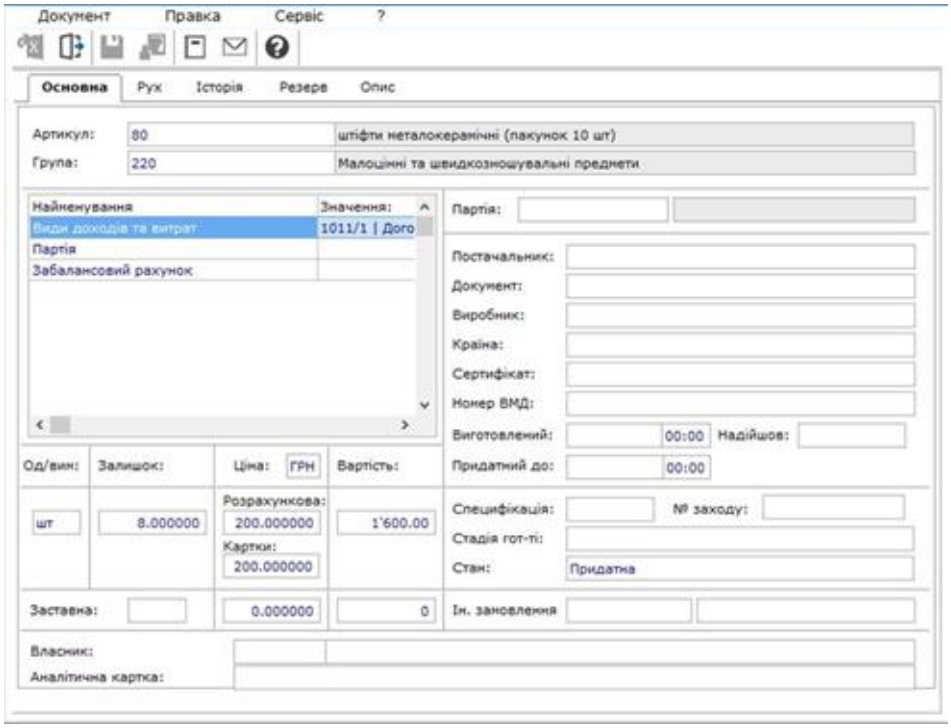

Рисунок 3.8. Картка складського обліку

Форма обліку руху запасів на складах представлена на рис. 3.9.

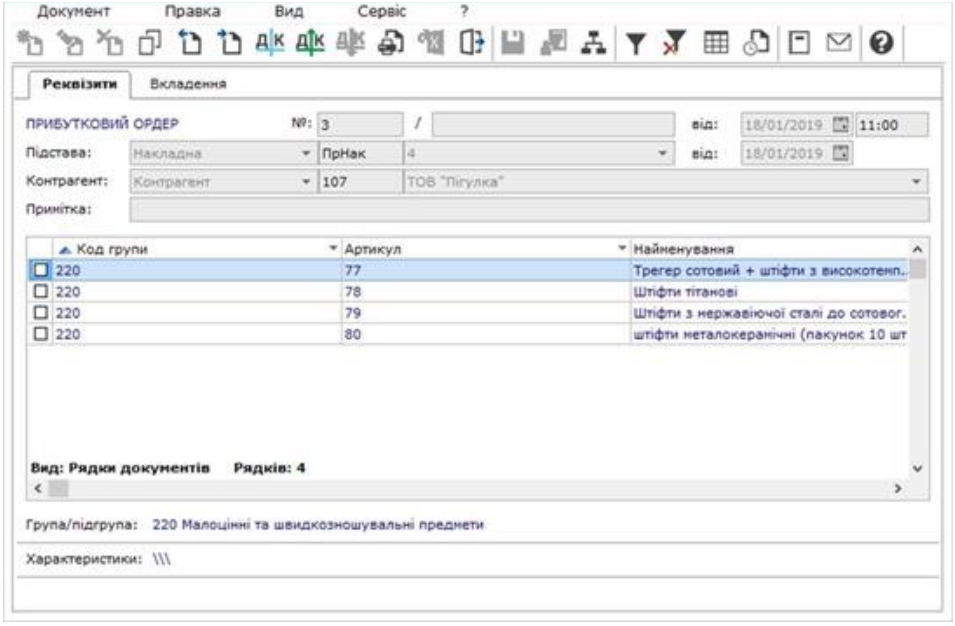

Рисунок 3.9. Форма обліку руху запасів

Модуль реєстру отримувачів веде базу даних фізичних осіб або організацій, які отримують допомогу та зберігає повну історію щодо отриманої допомоги. Передбачена фіксація повторного звернення отримувача з можливістю обирати необхідну частоту отримання. Забезпечується пошук і фільтрація записів за різними критеріями.

Форма реєстру отримувачів представлена на рис. 3.10.

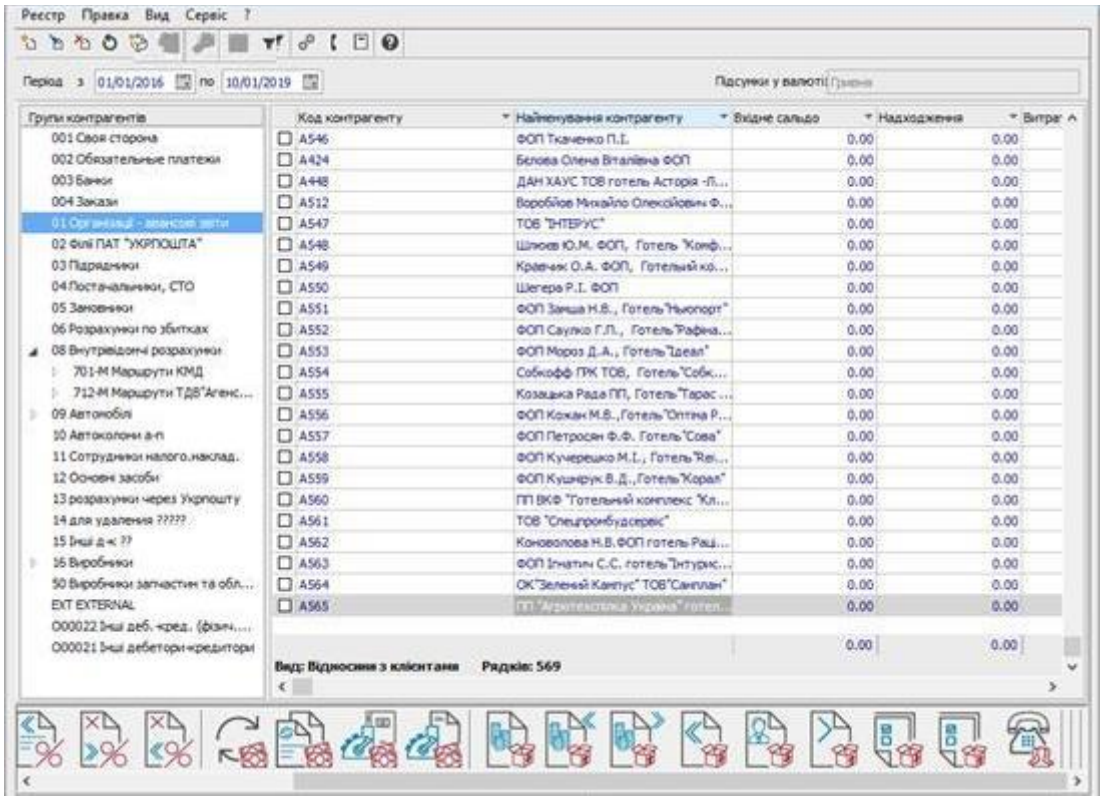

Рисунок 3.10. Форма реєстру отримувачів

Модуль видачі та розподілу допомоги дозволяє реєструвати факти видачі допомоги конкретному отримувачу, якого вибирають з реєстру або за відсутності в реєстрі вносять новий запис.

На рис. 3.11 показана форми картки подій: кожна видача допомоги фіксується як певна подія.

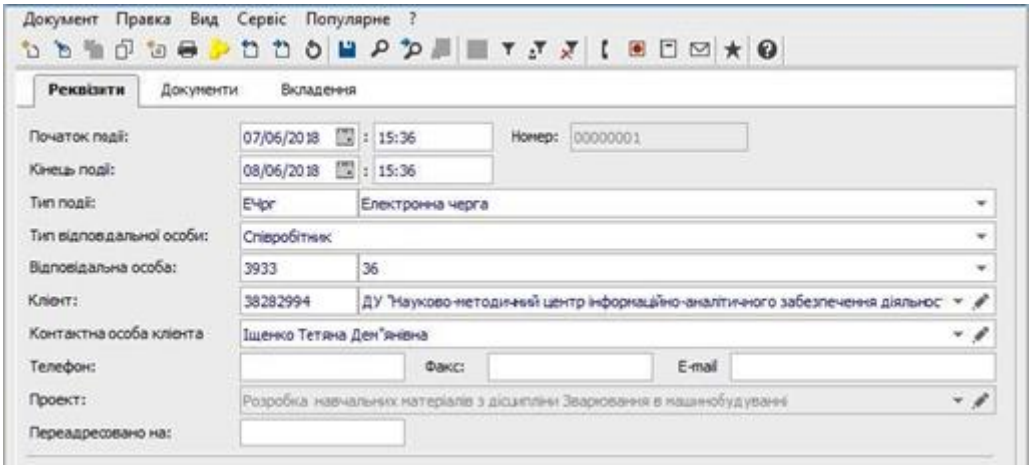

Рисунок 3.11. Картка події

Модуль звітності та аналітики дозволяє формувати стандартні звіти: рух ресурсів за період, залишки, статистика отримувачів, звіт для донора у форматі PDF та/або XLSX.

Друковані звіти формуються за запитом користувача у форматі PDF, що дозволяє їх одразу спрямовувати на принтер для друку. На рис. 3.12 представлено стандартну форму звіту №2 у варіанті для друку.

ЗВЕДЕНИЙ ЗВІТ
про наявність і розподіл гуманітарної допомоги в розрізі отримувачів гуманітарної допомоги
(форма звіту № 2)

_____
(найменування місцевої державної адміністрації)
станом на «_____» _____ 20____ року

№	Отримувач	Адреса	Код ЄДРПОУ	Країна	Донор	Вантаж	Загальна вага вантажу / Загальна сума коштів, у тому числі в іноземній валюті	Рішення про визнання гуманітарною допомогою	Дата митного оформлення	Отримано		Розподілено		Залишок
										Дата	Вага вантажу/ Сума коштів, у тому числі в іноземній валюті	Дата	Вага вантажу/ Сума коштів, у тому числі в іноземній валюті	Вага вантажу/ Сума коштів, у тому числі в іноземній валюті
1														
2														

У разі невідповідності фактичного розподілу вантажу / коштів, у тому числі в іноземній валюті, плану розподілу, наданого отримувачем для визнання вантажу / коштів, у тому числі в іноземній валюті, гуманітарною допомогою, до цього Зведеного звіту додається Інформація про невідповідність фактичного розподілу вантажу / коштів, у тому числі в іноземній валюті, плану розподілу, наданого отримувачем для визнання вантажу / коштів, у тому числі в іноземній валюті, гуманітарною допомогою, окремо щодо кожного отримувача гуманітарної допомоги згідно з додатком 2 до Інструкції щодо заповнення форми звіту № 1 «Звіт про наявність та розподіл гуманітарної допомоги» та форми звіту № 2 «Зведений звіт наявність та розподіл гуманітарної допомоги в розрізі отримувачів гуманітарної допомоги», затвердженої наказом Міністерства соціальної політики України від 31 липня 2020 року № 539, що подається разом з цим Зведеним звітом і є його невід'ємною частиною.

_____
(посада, П. І. Б. та підпис уповноваженої особи)

Директор Департаменту реалізації державної соціальної політики **Ольга ДЕЙНЕГА**

Рисунок 3.12. Форма друкованого звіту – Зведений звіт про наявність і розподіл гуманітарної допомоги

Екранні звіти виводяться у вікно застосунку у його робочій області.

На рис. 3.13 зображено екранну форму відомостей щодо видачі гуманітарної допомоги.

№ п/п	Назва	к-ть	фактича видача		допомогу видано*		
			дата	місце	ПІБ	телефон	під-пис
1	Пакунок із 6 складових	1	12.02.23	Баєвська сільрада Чуцького району	Іванов П.І.	096 320 11 52	v
2	Пакунок із 6 складових	2	12.02.23	Баєвська сільрада Чуцького району	Лонь Л.П.	095 1145863	v
	.....						
136	Пакунок із 6 складових	2	28.02.23	Семенівська сільрада Потоцького району	Брильов Є.П.	097 4528836	v
	РАЗОМ	169					

Рисунок 3.13. Форма екранного звіту – відомості щодо видачі гуманітарної допомоги

На рис. 3.14 зображено екранну форму відомостей про первинну вартість отриманих товарів.

Рис. 3.15 демонструє екранну форму зведеного звіту про облік гуманітарної допомоги. Форма показує, що отримано 400 пакунків гуманітарної допомоги на суму 107 432 гривень (вартість одного пакунка – 269 гривень), з них розподілені 169 пакунків на суму 45 461 гривень, залишилось нерозподіленими та зберігаються на складі 231 пакунок на суму 62 011 гривень.

№ п/п	Найменування ТМЦ	Ціна, грн	Кількість	Сума, грн.
1	Масло рослинне соняшникове ТМ Добре, пляшка 1 л.	42,91	400	17 164
2	Рис білий, круглий ТМ Врожай, пакет 900 гр.	48,00	400	19 200
3	Цукор білий кристалічний, ТМ Солодке життя, пакет 1 кг.	30,04	400	12 016
4	Консерви мясні, тушкована свинина ТМ Смак, банка 600 гр.	97,12	400	38 848
5	Мило рідке туалетне Ландиш, 500 мл.	21,61	400	8 664
6	Пральний порошок для ручного прання Лотос, 450 гр.	29,00	400	11 600
РАЗОМ				**107 432**

Рисунок 3.14. Форма екранного звіту – Первинна вартість отриманих товарів

№ п/п	Зміст операції	дебет	кредит	сума, грн
1	Отримано гуманітарну допомогу, продукти харчування та гігієнічні засоби	281 (товари)	48 (Цільове фінансування і надходження)	107 432
2	Розподілено (передано безплатно) гуманітарну допомогу кінцевим набувачам	48	281	45 461

Рисунок 3.15. Форма екранного звіту – облік гуманітарної допомоги

Висновки до розділу

У третьому розділі описано практичну реалізацію застосунку відповідно до спроєктованої архітектури.

Структура проєкту організована за принципом поділу відповідальностей із чіткою відповідністю між директоріями і архітектурними рівнями.

Реалізовано всі ключові функціональні модулі: автентифікація з розмежуванням ролей, облік надходжень і видач з автоматичним оновленням залишків, реєстр отримувачів із захистом від дублювання, складський облік із сигналізацією критичних рівнів та генерація звітів у форматах PDF і XLSX.

У процесі розробки програмного застосунку було реалізовано багатомодульну архітектуру системи для обліку і координації гуманітарної допомоги.

Було розроблено:

- структуру програмного проєкту;
- ORM-моделі бази даних;
- графічний інтерфейс користувача;
- механізми взаємодії компонентів системи;
- модулі авторизації та формування звітів.

Інтерфейс виконано з урахуванням вимог зручності для нетехнічних користувачів. Забезпечено шифрування бази даних, хешування паролів і автоматичне резервне копіювання.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОЦІНКА ЯКОСТІ ЗАСТОСУНКУ

4.1. Тестування програмного застосунку

Тестування програмного забезпечення проводилось з метою перевірки:

- коректності роботи функціональних модулів;
- стабільності системи;
- правильності взаємодії компонентів;
- коректності обробки даних.

Було використано такі види тестування:

- модульне тестування;
- інтеграційне тестування;
- функціональне тестування;
- тестування інтерфейсу користувача.

4.2. Стратегія тестування

Стратегія тестування розроблюваного застосунку охоплює три рівні: модульне тестування окремих компонентів (юніт-тести), інтеграційне тестування взаємодії між рівнями архітектури та тестування зручності використання (usability testing). Такий підхід дозволяє виявляти дефекти на різних стадіях і з різних точок зору.

Автоматизоване тестування виконується з використанням фреймворку pytest. Тестові сценарії організовано у двох директоріях: tests/unit/ (модульні тести сервісів і репозиторіїв) та tests/integration/ (інтеграційні тести, що перевіряють повні ланцюжки операцій від сервісу до бази даних). Для ізоляції тестів від реальної бази даних використовується тестова in-memory SQLite, що ініціалізується на початку кожного тестового сеансу.

Функціональне і ручне тестування проводилося за тест-планом, що включав 42 тест-кейси для всіх функціональних модулів. Критерієм успішного проходження тесту вважалася відповідність фактичного

результату очікуваному. Тестові дані включали як типові сценарії використання, так і граничні випадки (порожній реєстр, нульові залишки, некоректні вхідні дані).

4.3. Модульне та інтеграційне тестування

4.3.1. Модульне тестування

Модульне тестування проводилось для окремих функцій та сервісів системи.

Приклад тесту:

```
def test_create_item():
    item = AidItem(name="Медикаменти", quantity=10)

    assert item.name == "Медикаменти"
    assert item.quantity == 10
```

Модульне тестування охоплює всі класи сервісного шару (IncomingService, OutgoingService, StockService, RecipientService, ReportService) і ключові репозиторії. Для кожного публічного методу написано щонайменше 3 тест-кейси: тест успішного виконання, тест обробки граничних значень і тест обробки виняткової ситуації. Загальне покриття коду тестами (test coverage) становить 78 %, що відповідає галузевим стандартам для застосунків подібного типу.

4.3.2. Інтеграційне тестування

Інтеграційне тестування використовувалось для перевірки взаємодії:

- GUI та сервісного шару;
- сервісів та бази даних;
- ORM-моделей та SQLite.

Інтеграційне тестування перевіряє коректність взаємодії між сервісами і репозиторіями у реальних сценаріях використання: реєстрація

надходження і перевірка оновлення залишків, видача допомоги і перевірка зменшення залишків, переміщення між складами і узгодженість залишків на обох складах. Результати ключових інтеграційних тестів наведено в таблиці 4.1.

4.3.3. Функціональне тестування

Під час функціонального тестування перевірялися:

- створення заявок;
- додавання ресурсів;
- формування звітів;
- авторизація користувачів;
- оновлення складських залишків.

Результати тестування представлені в табл. 4.2.

Усі 42 тест-кейси функціонального тестування були пройдені успішно. Зафіксовано та виправлено 7 дефектів у процесі тестування: 3 дефекти у валідації форм, 2 дефекти відображення (некоректне форматування числових значень), 1 дефект у логіці перевірки дублікатів отримувачів і 1 помилка при генерації звіту з порожніми даними.

4.4. Тестування зручності використання

Тестування зручності використання (usability testing) проводилося з залученням 5 учасників: 2 волонтери з досвідом роботи з гуманітарною допомогою, 1 координатор благодійного фонду, 1 бухгалтер без досвіду роботи зі складськими системами та 1 студент без специфічного досвіду. Жоден з учасників не брав участі у розробці застосунку.

Тестування зручності використання (usability testing) проводилося з залученням 5 учасників: 2 волонтери з досвідом роботи з гуманітарною допомогою, 1 координатор благодійного фонду, 1 бухгалтер без досвіду роботи зі складськими системами та 1 студент без специфічного досвіду. Жоден з учасників не брав участі у розробці застосунку.

Таблиця 4.1. Результати інтеграційного тестування

Тест-кейс	Очікуваний результат	Фактичний результат	Статус
Реєстрація надходження зі збільшенням залишків	Залишки збільшились	Залишки збільшились	✓ Пройдено
Видача при нульових залишках	Помилка, відмова у видачі	Помилка відображена	✓ Пройдено
Сигналізація критичного залишку	Виділення кольором	Виділення відображено	✓ Пройдено
Вхід з невірним паролем (3 рази)	Блокування облікового запису	Обліковий запис заблоковано	✓ Пройдено
Повторна видача в межах мінімального інтервалу	Попередження координатору	Попередження відображено	✓ Пройдено
Транзакція надходження з помилкою на середині	Відкочування, дані не збережено	База залишилась цілісною	✓ Пройдено
Генерація PDF-звіту з великим об'ємом даних	Файл сформовано < 2 сек	1,4 сек	✓ Пройдено
Пошук по 10 000 записів реєстру	Результат < 2 сек	0,8 сек	✓ Пройдено
Автоматичне резервне копіювання	Файл резервної копії створено	Файл створено	✓ Пройдено

Таблиця 4.2. Результати функціонального тестування

№	Функція системи	Очікуваний результат	Результат
1	Авторизація	Вхід у систему	Успішно
2	Додавання ресурсу	Запис у БД	Успішно
3	Формування заявки	Створення заявки	Успішно
4	Формування звіту	Генерація звіту	Успішно

Учасникам запропоновано виконати 8 сценарних завдань без підказок: зареєструвати нового донора і надходження, видати допомогу отримувачу, переглянути залишки, сформувати звіт для донора, зареєструвати нового отримувача, виконати пошук в реєстрі, додати нову позицію номенклатури, змінити пароль. Час виконання кожного завдання фіксувався, після завершення учасники заповнювали опитувальник оцінки зручності.

Результати тестування зручності виявилися загалом позитивними. Середній час на опанування базових операцій склав 22 хвилини, що відповідає нефункціональній вимозі (не більше 30 хвилин). Усі 5 учасників успішно виконали всі 8 завдань. Середня оцінка зручності за шкалою SUS (System Usability Scale) склала 78,5 балів, що відповідає рівню «добре» (75–85 балів).

За результатами опитувальника виявлено такі напрямки для покращення:

- бажаність підказок у полях форм (placeholder-текст) для нових користувачів;
- необхідність більш детального пояснення різниці між роллями при створенні облікового запису;
- побажання щодо можливості масового імпорту даних з Excel-файлів.

Зазначені зауваження враховано і частину з них усунуто у поточній версії, решта включено до переліку вдосконалень для наступної версії.

4.5. Аналіз результатів і рекомендації

Комплексне тестування підтвердило, що розроблений застосунок відповідає всім поставленим функціональним і нефункціональним вимогам. Усі визначені у першому розділі функціональні вимоги реалізовано і перевірено тестами. Нефункціональні вимоги щодо продуктивності (час відгуку < 2 секунди), зручності (опанування < 30 хвилин) і надійності (автоматичне резервне копіювання) виконано.

За результатами тестування сформульовано такі рекомендації щодо подальшого розвитку застосунку:

- реалізація імпорту даних з Excel-файлів для міграції з поточних таблиць – найбільш затребувана функція за результатами usability-тестування;
- розробка веб-версії або API для синхронізації даних між кількома пристроями в рамках однієї організації;
- розширення можливостей звітності: геоаналітика розподілу допомоги на основі адрес отримувачів;
- інтеграція з Telegram-ботом для сповіщень про критичні залишки без відкриття застосунку;
- розробка адаптивної мобільної версії для реєстрації операцій безпосередньо на місці видачі;
- сертифікація за стандартами безпеки даних для можливого офіційного впровадження у державних гуманітарних структурах.

4.6. Забезпечення надійності та масштабованості системи

Для забезпечення стабільної роботи програмного застосунку були реалізовані:

- перевірка коректності введених даних;
- обробка помилок;
- логування подій;
- резервне копіювання бази даних.

Використання SQLAlchemy та модульної архітектури дозволяє в майбутньому:

- перейти на PostgreSQL або MySQL;
- реалізувати мережеву взаємодію;
- створити веб-версію системи;
- інтегрувати систему з API гуманітарних організацій.

Висновки до розділу

У четвертому розділі описано стратегію і результати тестування застосунку на трьох рівнях: модульному, інтеграційному та тестуванні зручності. Автоматизовані тести забезпечують покриття коду на рівні 78 % і підтвердили коректність реалізованої бізнес-логіки.

Усі 42 тест-кейси функціонального тестування пройдено успішно після виправлення 7 виявлених дефектів. Тестування зручності з реальними користувачами підтвердило відповідність нефункціональним вимогам: середній час опанування склав 22 хвилини, оцінка SUS – 78,5 балів. Сформульовано конкретні рекомендації щодо подальшого розвитку продукту.

Проведене тестування підтвердило коректність роботи програмного забезпечення, стабільність функціонування системи та правильність взаємодії її структурних елементів.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне практичне завдання – розроблено десктопний застосунок для обліку і координації гуманітарної допомоги, що відповідає реальним потребам волонтерських і гуманітарних організацій, які діють в умовах воєнного часу в Україні.

За результатами виконання роботи зроблено такі висновки.

1. Проведено комплексний аналіз предметної галузі. Встановлено, що більшість організацій, задіяних у роботі з гуманітарною допомогою, використовують електронні таблиці, паперові журнали або месенджери, що не забезпечують належного рівня структурованості й прозорості обліку. Аналіз шести програмних аналогів показав, що жоден з них не поєднує повноцінного офлайн-режиму, зручного інтерфейсу для нетехнічних користувачів і повного охоплення ланцюга «донор – склад – отримувач».

2. Сформульовано функціональні та нефункціональні вимоги до застосунку. До ключових функціональних вимог належать: облік надходжень і видач з автоматичним управлінням залишками, реєстр отримувачів із захистом від дублювання, складський облік із сигналізацією критичних рівнів, генерація звітів у форматах PDF та XLSX. Серед нефункціональних вимог пріоритетними є автономна робота без мережевого підключення, висока зручність для нетехнічних користувачів та надійне збереження даних.

3. Обґрунтовано архітектурні рішення і технологічний стек. Обрано чотирирівневу архітектуру MVC з патернами Repository і Service Layer, що забезпечує масштабованість і зручність супроводу. Технологічний стек Python + PyQt6 + SQLite забезпечує автономну роботу, відкритість вихідного коду і зручність розробки. Спроєктовано реляційну схему бази даних із 13 таблицями, організовану за принципом транзакційного журналу для повної простежуваності руху ресурсів.

4. Реалізовано всі заплановані функціональні модулі застосунку: автентифікація з трьома ролями, облік надходжень і видач, реєстр донорів і отримувачів, складський облік, генерація звітів, адміністрування користувачів. Реалізовано шифрування бази даних (AES-256), bcrypt-хешування паролів і модуль автоматичного резервного копіювання.

5. Проведено тестування на трьох рівнях. Автоматизовані тести забезпечили покриття коду на рівні 78%. Усі 42 функціональні тест-кейси пройдено успішно після виправлення 7 дефектів. Тестування зручності з реальними користувачами підтвердило відповідність нефункціональним вимогам: середній час опанування склав 22 хвилини (вимога – не більше 30 хвилин), оцінка за шкалою SUS – 78,5 балів (рівень «добре»).

Практична цінність роботи полягає у створенні готового до впровадження програмного продукту, що закриває реальну технологічну прогалину у сфері гуманітарного обліку в Україні. Застосунок може бути безпосередньо використаний волонтерськими організаціями, благодійними фондами та регіональними гуманітарними штабами без додаткового налаштування серверної інфраструктури або придбання ліцензій.

Перспективами подальшого розвитку є реалізація імпорту даних з Excel-файлів, розробка API для синхронізації між пристроями, геоаналітика розподілу допомоги та інтеграція з Telegram-ботом для сповіщень. У більш довгостроковій перспективі – розробка мобільної версії та адаптація для офіційного використання у державних структурах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про гуманітарну допомогу» від 22.10.1999 № 1192-XIV. URL: https://zakon.rada.gov.ua/laws/show/1192-14

2. OCHA. Humanitarian Response Plan Ukraine 2024. URL: https://www.unocha.org/ukraine

3. UNHCR. Ukraine Situation. Operational Data Portal. URL: https://data.unhcr.org/en/situations/ukraine

4. Sommerville I. Software Engineering. 10th ed. Pearson Education Limited, 2016. 816 p.

5. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. 432 p.

6. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002. 560 p.

7. Документація Python 3.12. URL: https://docs.python.org/3/

8. Документація PyQt6. URL: https://www.riverbankcomputing.com/static/Docs/PyQt6/

9. SQLAlchemy Documentation. URL: https://docs.sqlalchemy.org/en/20/

10. SQLite Documentation. URL: https://www.sqlite.org/docs.html

11. ReportLab User Guide. URL: https://docs.reportlab.com/reportlab/userguide/ch1_intro/

12. openpyxl Documentation. URL: https://openpyxl.readthedocs.io/

13. pytest Documentation. URL: https://docs.pytest.org/

14. Bansal S. PyQt6 for Beginners. Packt Publishing, 2022. 280 p.

15. Brodie M. L. On the Development of Data Models. Research Issues in Database Systems. Springer, 1984. P. 14–40.

16. Коберн А. Написання ефективних варіантів використання / пер. з англ. Москва: Вільямс, 2002. 256 с.

17. Nielsen J. Usability Engineering. Academic Press, 1993. 362 p.

18. Brooke J. SUS: A "Quick and Dirty" Usability Scale. Usability Evaluation In Industry. Taylor & Francis, 1996. P. 189–194.

19. OWASP Top Ten. Open Web Application Security Project. URL: https://owasp.org/www-project-top-ten/

20. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE). International Organization for Standardization, 2011.

21. Гагарін О., Жиліна О. Методи та засоби проєктування інформаційних систем. Київ: НТУУ «КПІ», 2018. 200 с.

22. Мартін Р. Чистий код: Рефакторинг, шаблони, тестування та прийоми об'єктно-орієнтованого програмування / пер. з англ. Санкт-Петербург: Пітер, 2013. 464 с.

23. Шилдт Г. Алгоритми та структури даних / пер. з англ. Москва: Вільямс, 2018. 768 с.

24. Коплієнс Дж. Мультипарадигматичне проєктування. Addison-Wesley, 1998. 246 с.

25. KoboToolbox Documentation. URL: https://support.kobotoolbox.org/

ДОДАТОК А

Схема бази даних

Схема бази даних містить 13 взаємопов'язаних таблиць. Нижче наведено опис ключових зв'язків між основними сутностями застосунку.

Таблиця users зберігає облікові записи з хешованим паролем і роллю. Таблиці incoming та outgoing пов'язані з таблицями donors і recipients відповідно. Таблиця stock містить агреговані залишки по парах (warehouse_id, item_id). Усі операційні таблиці (incoming, outgoing, transfers, writeoffs) мають зовнішній ключ на users для трасування відповідального оператора.

Основні таблиці та їх поля:

users: id, username, password_hash, role [admin|coordinator|operator], is_active, created_at

donors: id, name, type [individual|organization], phone, email, address, notes

recipients: id, full_name, id_document, phone, address, category, registered_at, min_interval_days

items: id, name, category_id, unit, min_stock, description

stock: id, warehouse_id, item_id, quantity (унікальна пара warehouse+item)

incoming: id, donor_id, warehouse_id, operator_id, date, notes

outgoing: id, recipient_id, warehouse_id, operator_id, date, notes

ДОДАТОК Б

Фрагменти вихідного коду

Б.1. Клас IncomingService — реєстрація надходження

```
class IncomingService:    def __init__(self, session):
self.session = session              self.incoming_repo =
IncomingRepository(session)            self.stock_repo =
StockRepository(session)      def register_incoming(self,
donor_id, warehouse_id,                      operator_id,
date, items, notes=''):        """Реєструє надходження та
оновлює залишки."""           with self.session.begin():
incoming          =          self.incoming_repo.create(
donor_id=donor_id,            warehouse_id=warehouse_id,
operator_id=operator_id,                        date=date,
notes=notes            )           for item_id, quantity
in items:                   self.incoming_repo.add_item(
incoming.id,             item_id,             quantity)
self.stock_repo.add_quantity(              warehouse_id,
item_id, quantity)         return incoming
```

Б.2. Перевірка мінімального інтервалу між видачами у RecipientService

```
    def        check_interval(self,        recipient_id):
"""Повертає True якщо видача дозволена."""       last =
self.outgoing_repo.last_for_recipient(recipient_id)     if
not last:           return True, None       recipient =
self.recipient_repo.get(recipient_id)       days_passed =
(date.today()  -  last.date).days        if  days_passed  <
recipient.min_interval_days:               return  False,
days_passed     return True, days_passed
```

ДОДАТОК В

Інструкція користувача

В.1. Перший запуск і налаштування

При першому запуску застосунок автоматично ініціалізує порожню базу даних і пропонує встановити пароль адміністратора. Рекомендований пароль — не менше 8 символів з використанням великих і малих літер, цифр та спеціальних символів. Обліковий запис адміністратора (admin) надає доступ до всіх функцій, включно з управлінням користувачами і налаштуваннями резервного копіювання.

В.2. Реєстрація надходження

Для реєстрації нового надходження перейдіть до розділу «Надходження» і натисніть кнопку «+ Нове надходження». У відкритій формі вкажіть: (1) донора — виберіть з реєстру або зареєструйте нового; (2) дату і склад; (3) перелік позицій із кількостями. Після натискання «Зберегти» залишки на складі оновлюються автоматично.

В.3. Видача допомоги

У розділі «Видача» натисніть «+ Нова видача». Введіть прізвище або документ отримувача у пошукове поле. Якщо отримувач зареєстрований — виберіть з результатів пошуку; якщо новий — натисніть «Зареєструвати нового». Додайте позиції до видачі та натисніть «Зберегти». Якщо пройшло менше встановленого мінімального інтервалу від попередньої видачі, система відобразить попередження з кількістю днів.

В.4. Формування звіту

Перейдіть до розділу «Звіти», оберіть тип звіту, задайте діапазон дат і натисніть «Сформувати». Для збереження у PDF натисніть «Зберегти PDF», для Excel — «Зберегти XLSX». Звіт відкривається для попереднього перегляду перед збереженням.