

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Навчально-науковий інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___» _____ 2025 р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Автоматизована система контролю замовлень в мережі
ресторанів на основі бази даних»

Виконала: студентка IV курсу, групи КА-15
Бондарева Діана Володимирівна

Керівник:
Професор, д.т.н., Мухін Вадим Євгенович

Консультант з економічного розділу:
доцент, к.е.н. Рощина Надія Василівна

Консультант з нормоконтролю:
к.ф.-м.н. Статкевич Віталій Михайлович

Рецензент: к.т.н., доцент, доцент кафедри
цифрових технологій в енергетиці НН ІАТЕ КП
ім. Ігоря Сікорського Шаповалова Світлана Ігорівна

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студентка _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___» _____ 2025 р.

ЗАВДАННЯ

на дипломну роботу студентці

Бондаревій Діані Володимирівні

1. Тема роботи «Автоматизована система контролю замовлень в мережі ресторанів на основі бази даних», керівник роботи Мухін Вадим Євгенович, д.т.н., професор, затверджені наказом по університету від 26. 05. 2025р. №1759-с.
2. Термін подання студентом роботи
3. Вихідні дані до роботи: акти звірки, накладні, структура бази даних для обліку продукції, інформація з торгових точок, вимоги до структури зведених даних.
4. Зміст роботи: 1. Аналіз баз даних для контролю обліку замовлень. 2. Загальні принципи розробки системи звірки замовлень. 3. Проектування та реалізація системи звірки та обліку замовлень. 4. Оцінка ефективності та аналіз можливостей програмного рішення.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	доцент, к.е.н. Рощина Н.В.		

7. Дата видачі завдання: 14.04.2025

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання та затвердження теми роботи	17.04.2025	Виконано
2	Збір інформації та пошук літератури	25.04.2025	Виконано
3	Аналіз існуючих підходів до систем обліку і звірки замовлень	28.04.2025	Виконано
4	Проектування архітектури системи та вибір методів для аналізу даних	03.05.2025	Виконано
5	Розробка основних алгоритмів програмного забезпечення	07.05.2025	Виконано
6	Розробка інтерфейсу користувача та інтеграція з базами даних	14.05.2025	Виконано
7	Тестування розробленого програмного продукту, виправлення помилок	20.05.2025	Виконано
8	Оформлення дипломної роботи та презентації	29.05.2025	Виконано

Студентка

Діана БОНДАРЕВА

Керівник

Вадим МУХІН

РЕФЕРАТ

Дипломна робота: 112 с., 23 рис., 6 табл., 2 дод., 16 джерел.

АВТОМАІЗОВАНА СИСТЕМА КОНТРОЛЮ ЗАМОВЛЕНЬ В МЕРЕЖІ РЕСТОРАНІВ НА ОСНОВІ БАЗИ ДАНИХ

Об'єкт дослідження – процес обробки та контролю замовлень у мережі ресторанів.

Предмет дослідження – інформаційна система для автоматизованого контролю замовлень на основі бази даних.

Мета роботи – розробка автоматизованої системи контролю замовлень у мережі ресторанів, яка забезпечує ефективне управління даними про замовлення, ціни та постачальників із використанням бази даних.

У процесі дослідження було проаналізовано існуючі системи автоматизації ресторанного бізнесу. Спроековано структуру бази даних для зберігання інформації про замовлення, ціни та постачальників. Реалізовано користувацький інтерфейс для обробки замовлень та їхнього контролю.

Використано мову програмування Python та бібліотеку Tkinter для побудови графічного інтерфейсу користувача. Для зберігання та обробки даних застосовано систему управління базами даних MySQL. Система дозволяє здійснювати пошук, додавання, редагування та видалення записів, створювати корисну аналітику приходів, а також забезпечує контроль обліку замовлень.

Результатом роботи є працездатна система, яка може бути впроваджена в мережі ресторанів для оптимізації внутрішніх бізнес-процесів та полегшення роботи працівників.

ABSTRACT

Diploma thesis: 112 p., 23 fig., 6 tabl., 2 app., 16 ref.

AUTOMATED ORDER CONTROL SYSTEM IN A RESTAURANT CHAIN BASED ON A DATABASE

Object of research – the process of processing and controlling orders in a restaurant chain.

Subject of research – an information system for automated order control based on a database.

Purpose of the work – to develop an automated order control system for a restaurant chain that ensures effective management of data on orders, prices, and suppliers using a database.

During the research, existing restaurant business automation systems were analyzed. A database structure was designed to store information about orders, prices, and suppliers. A user interface for order processing and control was implemented.

The Python programming language and the Tkinter library were used to build the graphical user interface. The MySQL database management system was used for data storage and processing. The system allows searching, adding, editing, and deleting records, generating useful analytics on supply flows, and provides order accounting control.

The result of the work is a functional system that can be implemented in restaurant chains to optimize internal business processes and facilitate the work of employees.

ЗМІСТ

ВСТУП.....	9
РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ	12
1.1 Функціонування і структура роботи мережі ресторанів	12
1.2 Актуальність автоматизації контролю замовлень у ресторанному бізнесі.....	16
1.3 Аналіз існуючих рішень і доцільність розробки власної системи	21
1.4 Переваги побудови системи на основі бази даних	26
Висновки до розділу 1	28
РОЗДІЛ 2 ТЕОРЕТИЧНИЙ ОПИС ПРОГРАМНИХ ІНСТРУМЕНТІВ.....	29
2.1 Таблиці та їх структура.....	29
2.1.1 Типи даних у MySQL	31
2.1.2 Обмеження цілісності	32
2.1.3 Індексація.....	33
2.1.4 SQL – запити.....	33
2.1.5 Нормалізація та засоби адміністрування	34
2.2 Створення вкладеного інтерфейсу за допомогою віджета Notebook модуля Tkinter	34
2.2.1 Процес створення інтерфейсу.....	35
Висновки до розділу 2	36
РОЗДІЛ 3 ОПИС РЕЗУЛЬТАТІВ РЕАЛІЗАЦІЇ ПРОГРАМИ	38
3.1 Архітектура застосунку та логіка роботи	38
3.2 Опис створеної бази даних	41
3.3 Ініціалізація графічного інтерфейсу.....	46

	7
3.4 Опис вкладки «Звірка».....	48
3.4.1 Загальне призначення та принцип роботи	48
3.4.2 Структура інтерфейсу та логіка взаємодії	50
3.5 Опис вкладки «Зведена таблиця»	52
3.5.1 Загальне призначення	52
3.5.2 Структура інтерфейсу та логіка взаємодії	53
3.6 Опис вкладки «Аномалії»	55
3.6.1 Загальне призначення та принцип роботи	55
3.6.2 Структура інтерфейсу та логіка взаємодії	55
3.7 Опис вкладки «Графік цін».....	57
3.7.1 Загальне призначення вкладки	57
3.7.2 Структура інтерфейсу та логіка взаємодії	58
Висновки до розділу 3	60
РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО	
ПРОДУКТУ	62
4.1. Постановка задачі проектування	63
4.2. Обґрунтування функцій програмного продукту	64
4.3. Обґрунтування системи параметрів програмного продукту	68
4.4. Аналіз експертного оцінювання параметрів.....	71
4.5. Аналіз рівня якості варіантів реалізації функцій.....	74
4.6. Економічний аналіз варіантів розробки ПП	76
4.7. Вибір кращого варіанту ПП техніко-економічного рівня.....	81
Висновки до розділу 4	82

	8
ВИСНОВКИ	84
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	86
ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ.....	88
ДОДАТОК Б. ГРАФІЧНІ МАТЕРІАЛИ	104

ВСТУП

Сучасні умови розвитку ресторанного бізнесу в Україні вимагають від підприємців не лише високої якості обслуговування, а й ефективного управління внутрішніми процесами. Громадське харчування є однією з найдинамічніших галузей торгівлі, що обумовлює зростаючу конкуренцію між кафе, барами та ресторанами. Для того щоб утримувати конкурентні позиції, необхідно інвестувати в сучасні інструменти підтримки й розвитку бізнесу, серед яких автоматизація займає провідне місце.

Ведення ресторанного бізнесу може приносити більше задоволення і менше вимагати зусиль, якщо його організувати грамотно: з чітким плануванням, прозорою звітністю та оптимізацією всіх процесів. Важливу роль при цьому відіграє повний контроль над рухом товарно-матеріальних цінностей, грошових потоків, зменшення виробничих витрат, попередження зловживань з боку персоналу та створення привабливої атмосфери для відвідувачів [1].

Однією з найскладніших задач у ресторанному бізнесі є облік товарно-матеріальних цінностей. Продукти, які закупаються, проходять через кілька стадій переробки, перетворюючись у напівфабрикати, з яких згодом готуються страви. Без автоматизованого обліку практично неможливо точно визначити фактичні витрати продуктів, залишки на кухні чи складі, а також виявити крадіжки або нелегальні продажі. Саме тому сучасна автоматизована система, що спирається на технологічні карти та калькуляційні дані, є необхідністю.

Автоматизація закладу громадського харчування дозволяє оптимізувати документообіг, забезпечити порядок на складі, координувати дії персоналу при змінному графіку роботи, а також централізовано зберігати і обробляти всю необхідну інформацію. Це особливо актуально у великих містах і мережах ресторанів, де обсяг інформації та кількість операцій значно зростає.

Сучасна система автоматизації ресторану – це багатофункціональний інструмент для управління, який дозволяє підвищити якість сервісу, прискорити обслуговування клієнтів, мінімізувати зловживання персоналу, зменшити навантаження на працівників, а також забезпечити можливість гнучкого розширення функціональності в майбутньому. Впровадження таких систем дає змогу не лише автоматизувати облік, а й створити умови для якісної аналітики, прогнозування й оперативного прийняття рішень.

В умовах цифрової трансформації бізнес-процесів особливу актуальність набуває цифровізація управлінських процесів. Ефективне управління замовленнями – один із критично важливих аспектів функціонування ресторанної мережі. Воно охоплює формування потреб, постачання товарів, облік залишків, узгодження між закладами та контроль виконання.

Традиційні методи управління, що базуються на паперових носіях або електронних таблицях, вичерпали себе, особливо в умовах масштабування діяльності. У таких умовах зростає потреба у впровадженні централізованих автоматизованих систем, які дозволяють не лише знижувати навантаження на персонал, а й покращувати планування закупівель, зменшувати ризики логістичних збоїв, оптимізовувати витрати та вивільняти ресурси для вирішення складніших завдань.

Особливо актуальним є створення єдиного механізму контролю замовлень для мережі ресторанів, що дає змогу не лише ефективно обслуговувати клієнтів у кожному закладі, але й узгоджено керувати поставками, залишками, попитом та аналітикою в масштабі всієї мережі. Відсутність такої системи часто призводить до втрати даних, помилок, затримок у обслуговуванні та зниження конкурентоспроможності бізнесу.

Окрім економічної ефективності, автоматизація має й значну соціальну значущість. Вона підвищує рівень організації праці, створює прозору систему обліку, знижує залежність результатів роботи від людського фактору та сприяє

прийняттю обґрунтованих управлінських рішень. У контексті цифрової трансформації економіки України та інтеграції у світові ринки, такі технологічні ініціативи відповідають загальнонаціональному курсу на цифровізацію і модернізацію галузей [2].

Метою даної дипломної роботи є проектування та реалізація автоматизованої системи контролю замовлень у мережі ресторанів на основі реляційної бази даних. Система повинна забезпечувати централізований облік, підтримку ключових операцій, зручний інтерфейс користувача та можливість масштабування й розширення функціоналу в майбутньому.

Об'єктом дослідження є процес інформаційного забезпечення управління замовленнями в ресторанній мережі. Предметом дослідження виступає база даних та програмна реалізація засобів автоматизації обліку замовлень.

У процесі виконання роботи вирішуються такі основні завдання:

- 1) вивчення специфіки предметної області та обґрунтування потреби в автоматизації;
- 2) аналіз існуючих програмних рішень та визначення їхніх обмежень;
- 3) проектування логічної структури бази даних з урахуванням вимог цілісності, узгодженості й продуктивності;
- 4) розробка користувацького інтерфейсу для ведення щоденного обліку та аналітики;
- 5) тестування й демонстрація роботи системи на прикладі типових сценаріїв ресторанної мережі.

Результатом дипломної роботи є програмний продукт, який дозволяє ефективно контролювати замовлення та інтегруватися у чинні бізнес-процеси підприємства. Запропоноване рішення є універсальним та може бути адаптоване до інших галузей, що мають подібну логістику обліку поставок та обробки замовлень.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Функціонування і структура роботи мережі ресторанів

Мережа ресторанів – це сукупність різних закладів харчування, що мають спільне управління, бренд та, можливо, меню. Функціонування мережі передбачає координацію роботи окремих ресторанів, логістику поставок, маркетинг, а також забезпечення єдиних стандартів якості та обслуговування. Структура мережі може бути різною:

- 1) централізоване управління – у таких мережах є центральний офіс, який відповідає за маркетинг, закупівлі, стандартизацію процесів та навчання персоналу;
- 2) децентралізоване управління – у децентралізованих мережах окремі ресторани мають більшу автономію в управлінні та прийнятті рішень;
- 3) франшизне управління – франшизні мережі передбачають спільне використання бренду, стандартизованих процесів та навчання, але окремі ресторани мають певну автономію в управлінні та маркетингу.

В даній роботі ми працюватимемо з мережею з централізованим видом управління, де функціонування сучасного ресторанного підприємства ґрунтується на чіткій взаємодії низки ключових структурних підрозділів, кожен з яких виконує визначені управлінські, адміністративні та операційні функції. Узгодженість їхньої діяльності є запорукою досягнення стратегічних і тактичних цілей компанії, забезпечення стабільності бізнес-процесів та формування сталого конкурентного середовища. Оскільки всі області управління тісно між собою пов'язані, вони мають одну спільну базу даних, яка слугує повноцінною

системою з розширеним функціоналом. Аби покращити роботу цієї системи потрібно глибше зануритись в систему управління мережею ресторанів.

Однією з основних складових внутрішньої організаційної системи є кадровий напрям, відповідальний за підбір, адаптацію, навчання та мотивацію персоналу. В межах цього напрямку формуються системи оцінювання результативності праці, визначаються стандарти взаємодії з гостями, а також закладаються основи корпоративної культури. Якість кадрової роботи прямо впливає на рівень клієнтського сервісу, продуктивність обслуговування та стабільність функціонування підрозділів у динамічному середовищі ресторанної діяльності.

Бухгалтерська служба виконує функції фінансового обліку, контролю витрат, адміністрування податкових зобов'язань, а також формує базу для прийняття рішень у частині бюджетування та оптимізації грошових потоків. Надійність облікових даних і своєчасність фінансової звітності створюють умови для обґрунтованого планування й управління рентабельністю бізнесу.

Маркетинговий підрозділ відповідає за створення ринкової стратегії, реалізацію заходів із просування бренду, управління комунікаціями та формування лояльності цільової аудиторії. У межах цього напрямку проводяться рекламні кампанії, розробляються програми лояльності, досліджується споживча поведінка. Ефективність маркетингової діяльності безпосередньо впливає на рівень заповнюваності залів, середній чек та частоту повторних візитів.

Операційне управління зосереджене на контролі повсякденної діяльності окремих закладів або філій. У межах цього напрямку здійснюється координація роботи директорів ресторанів, дотримання стандартів обслуговування, впровадження внутрішніх процедур, а також реалізація політик, що забезпечують єдиний підхід до організації сервісу та операційної дисципліни.

Технічний відділ відповідає за матеріально-технічне забезпечення, включно з експлуатацією обладнання, обслуговуванням інженерних мереж,

меблюванням та виконанням ремонтних робіт. Високий рівень технічної підтримки знижує ймовірність збоїв у процесах обслуговування, сприяє раціональному використанню ресурсів і формує передумови для ефективної роботи виробничих підрозділів.

Логістичний напрям забезпечує своєчасне постачання сировини, управління товарними запасами та взаємодію з постачальниками. У межах цього напрямку здійснюється контроль за якістю продукції, ведеться планування закупівель, проводиться оптимізація складських запасів. Раціональне логістичне управління є визначальним чинником у досягненні стабільного рівня обслуговування та контролі собівартості продукції.

Юридична служба реалізує функції правового супроводу діяльності компанії, включно з договірною роботою, супроводом господарських операцій, веденням претензійно-позовної роботи та забезпеченням відповідності чинному законодавству. Її діяльність знижує правові ризики та підвищує прозорість взаємодії із зовнішніми контрагентами.

Аналітична служба функціонує як інтеграційна складова, що забезпечує всі структурні напрями релевантною інформацією для ухвалення управлінських рішень. У межах цього напрямку здійснюється збір, оброблення й інтерпретація даних, розробка статистичних і динамічних моделей, а також моніторинг ключових показників ефективності діяльності. Аналітика дозволяє виявляти відхилення, формувати прогностичні сценарії та забезпечувати управлінську підтримку на основі об'єктивних числових і якісних параметрів. Таким чином, злагоджена робота усіх структурних підрозділів ресторанної компанії формує цілісну систему управління, де кожен відділ виконує визначену роль, а ефективна міжфункціональна взаємодія створює основу для стійкого розвитку та підвищення конкурентоспроможності на ринку гостинності.

У процесі внутрішнього аналізу організаційної структури підприємств ресторанної сфери було встановлено, що логістичний напрям демонструє

високий рівень функціональної ефективності та забезпечує стабільне виконання ключових завдань, пов'язаних із поставками продукції, товарного забезпечення, контролю залишків та взаємодії з постачальниками. Водночас, незважаючи на результативність і відповідальність відповідного відділу, особливості організації його роботи на сучасному етапі зумовлюють надмірне навантаження на персонал, що переважно спричинене ручним характером обробки значного обсягу інформації.

Найбільш трудомісткими процесами виявляються створення, погодження та адміністрування замовлень на постачання, планування графіків доставки, а також звірка фактичних поставок із внутрішніми обліковими даними та інформацією від постачальників. Останній етап є особливо критичним, оскільки навіть незначні розбіжності у кількісних або вагових показниках можуть істотно вплинути на точність формування собівартості готових страв, на достовірність показників списання, а також на подальший аналіз рентабельності меню-позицій. Саме тому процес контролю та звірки виконано з максимальною точністю, що вимагає додаткового часу та високої концентрації уваги з боку працівників.

У більшості випадків ключові операції в логістичному блоці виконуються за допомогою електронних таблиць та програм загального призначення, які, хоча й забезпечують базовий рівень обліку, не дають змоги досягти необхідної гнучкості, швидкості обробки даних і рівня аналітичної глибини, що вимагаються в умовах масштабного ресторанного бізнесу. Наявна система не інтегрована з іншими бізнес-напрямами, що ускладнює як оперативне прийняття рішень, так і побудову цілісної аналітичної моделі, зокрема для оцінювання впливу постачань на загальні фінансові показники [3].

З урахуванням результатів структурного аналізу трудових витрат і часових затрат на виконання логістичних функцій, а також обґрунтованих потреб у підвищенні прозорості та точності даних, було сформульовано висновок про доцільність поетапної автоматизації логістичного напрямку. Запровадження

спеціалізованого програмного забезпечення, здатного забезпечити централізоване управління замовленнями, автоматичне оновлення залишків, електронну верифікацію поставок та інтеграцію з аналітичними модулями, є стратегічно обґрунтованим рішенням. Такі зміни не лише зменшують навантаження на персонал, а й сприяють зниженню операційних ризиків, оптимізації управлінських витрат і покращенню якості управлінських даних.

Автоматизація зазначених процесів, окрім підвищення ефективності, також відкриває перспективи для організаційної оптимізації структури підрозділу, зменшення витрат на рутинні операції та перерозподілу трудових ресурсів на виконання більш складних завдань, пов'язаних із плануванням, контролем та стратегічним управлінням ланцюгами постачання. У сукупності це формує передумови для переходу на якісно новий рівень логістичної підтримки ресторанного бізнесу в умовах цифрової трансформації [4].

1.2 Актуальність автоматизації контролю замовлень у ресторанному бізнесі

У сучасних умовах розвитку ресторанної галузі спостерігається істотне ускладнення внутрішніх бізнес-процесів, зокрема тих, що пов'язані з організацією логістики, контролем замовлень і обліком постачання продукції. Підвищення динаміки споживчого попиту, розширення мереж закладів та зростання обсягів операційної діяльності створюють потребу у впровадженні більш ефективних засобів управління, серед яких автоматизація контролю замовлень посідає особливе місце [5].

У багатьох випадках процеси створення та узгодження заявок, облік залишків, звірка поставок з фактичними даними, а також подальше формування звітності залишаються залежними від ручної праці. Навіть за використання електронних таблиць і базових цифрових інструментів, що вже впроваджені на підприємствах, спостерігається брак єдиної інтегрованої системи, яка б забезпечувала точну, узгоджену і масштабовану роботу в межах усієї організаційної структури ресторанної мережі.

Наявні підходи, як правило, не здатні забезпечити належний рівень прозорості, оперативності й аналітичної точності, що стає критичним у контексті контролю за собівартістю продукції, коректністю обліку товарних залишків і фінансовим плануванням. Відсутність автоматизованих засобів веде до перевитрат часу, збільшення навантаження на працівників, а також до потенційних помилок, які виникають через людський фактор. Це ускладнює не лише щоденну операційну діяльність, а й стратегічне управління підприємством загалом.

У зв'язку з активним розвитком ресторанного господарства, з підвищенням конкуренції, інтерес до автоматизації управління постійно зростає. Автоматизація системи контролю поставок продукції для ресторанів має на меті оптимізувати процес постачання та управління складським запасом, забезпечуючи точний облік та мінімізацію витрат. Ця система має бути інтегрована з базою даних, де зберігається інформація про товари, складські залишки, замовлення та поставки, а також дозволяти відстежувати залишки на кожному ресторані в мережі.

Основні функції такої системи:

1. Централізований облік складських залишків. Тобто усі складські залишки в мережі ресторанів згруповані в єдиній базі даних, що дозволяє відстежувати запаси в реальному часі, також інтеграція з POS системами на кожному ресторані для автоматичного відображення змін в запасі, і можливість

відстежувати переміщення товарів між різними локаціями (з центрального складу до ресторанів, з ресторану до ресторану).

2. Аналіз та оптимізація. Тобто: зведення даних про споживання товарів для кожного ресторану, що дає можливість оптимізувати замовлення та знизити витрати; аналіз залишків, що дозволяє виявляти надлишки та дефіцити товарів; можливість створення звітів про ефективність роботи доставщиків та логістики.

3. Інтеграція з іншими системами. Тобто можливість інтеграції з CRM-системою для відстеження замовлень клієнтів, а також інтеграція з ERP-системою для управління фінансовими потоками та бухгалтерським обліком.

З огляду на вищезазначене, впровадження автоматизованих систем контролю замовлень є нагальною потребою ресторанного бізнесу. Такі системи забезпечують централізоване управління процесами постачання, дозволяють здійснювати облік і звірку даних у режимі реального часу, а також інтегруються з аналітичними модулями для формування точних прогнозів і звітів. Вони не лише знижують ризики логістичних порушень і помилок, а й підвищують ефективність використання ресурсів, надаючи можливість перерозподілу кадрового потенціалу на завдання вищого рівня складності.

Вітчизняний ресторанний бізнес автоматизує свої внутрішні бізнес-процеси. При цьому під інтелектуальною автоматизацією процесів підприємства громадського харчування розуміємо інтеграцію технологічних застосунків, що забезпечують оперативний бізнес-аналіз і розширення можливостей з обслуговування, які дають змогу підвищити аналітичність інформації, її оперативність доведення до власника та інших користувачів, у тому числі користувачів фінансової звітності.

Необхідність в автоматизації бізнес-процесів зумовлена такими основними причинами:

- 1) розширенням асортименту реалізованих страв і сировини, що закуповується для їх виробництва;

- 2) необхідністю оптимізації складських операцій з метою зниження фінансових втрат від псування вчасно невикористаної сировини;
- 3) гострою потребою в підвищенні швидкості пошуку надійного постачальника продукції та укладення договору з ним щодо термінів поставки необхідного товару;
- 4) попитом споживачів на різні способи оплати отриманого товару;
- 5) законодавчо обумовленою необхідністю ведення електронного бухгалтерського обліку, надсилання в електронний спосіб статистики до контролюючих органів, обов'язком звітувати за кожну одиницю проданої алкогольної або молочної продукції;
- 6) потребою в побудові ефективної маркетингової стратегії.

Перелік потреб ресторанів в автоматизації бізнес-процесів досить широкий. На сьогодні ці потреби задовольняються наявними на ринку системами автоматизації, більша частина з яких дає змогу автоматизувати відразу кілька бізнес-процесів [6].

У загальному вигляді схему автоматизації ресторанного бізнесу можна побачити на рис. 1.1

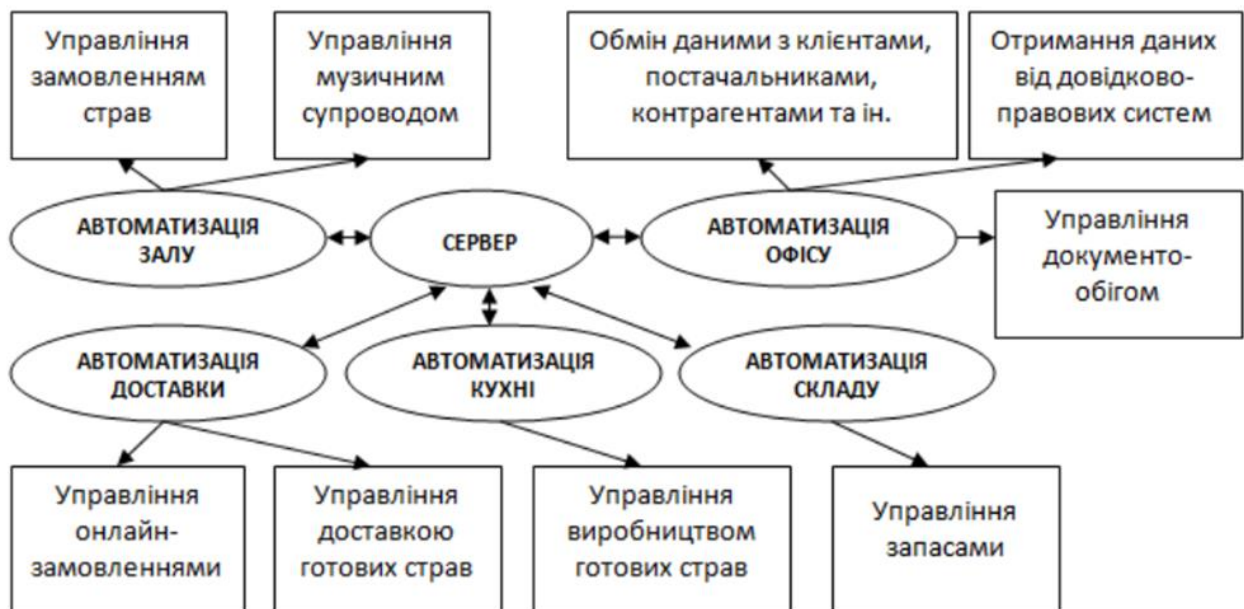


Рисунок 1.1 Основні напрями автоматизації ресторанного бізнесу

Центром автоматизованих робочих місць є сервер, на якому розгортаються відповідні системи автоматизації ресторанного бізнесу. Безпосередньо автоматизація здійснюється в процесі обміну даними між основними підрозділами підприємства, з одного боку, постачальниками та контрагентами – з іншого, за допомогою сервісів, які надають встановлені системи автоматизації.

Кожна з дій супроводжується формуванням звіту у відповідній системі автоматизації ресторанного бізнесу, що допомагає аналізувати проходження бізнес-процесів, оцінювати їхню ефективність, генерувати коригувальні заходи, формувати маркетингові та інші програми, якщо відповідний функціонал наявний у встановлених системах автоматизації.

Формування комплексних систем інтелектуальної автоматизації на підприємствах ресторанної галузі дає змогу максимально спростити механізми управління низкою бізнес-процесів:

- 1) скорочення та оптимізація витрат, збільшення швидкості обробки та надходження інформації;
- 2) встановлення повного або часткового контролю бізнес-процесів;
- 3) раціональне використання робочого часу працівників;
- 4) оперативне планування закупівлі шляхом використання різних простих у розумінні та наочних звітів;
- 5) мінімізація впливу людського фактору на результати діяльності.

Загалом, інтелектуальна автоматизація відіграє важливу роль у підвищенні ефективності та конкурентоспроможності українського ресторанного бізнесу, проте важливо розглядати її в контексті конкретних потреб і можливостей кожного закладу [7].

Слід зазначити, що функціональність систем автоматизації та вартість їх встановлення, обслуговування і використання визначають ринкові позиції таких систем. Наявні на підприємстві системи комплексної автоматизації, як правило,

значно дорожчі за системи, що мають менший функціонал. Однак останні можуть конкурувати на ринку не тільки за ціною, а й за ступенем опрацювання наявного в них функціоналу. Грамотно поєднуючи різні системи автоматизації, менеджмент підприємства не тільки підвищує ефективність процесу управління підприємством, а й підвищує якість обслуговування гостей, мінімізує зловживання з боку персоналу.

Таким чином, автоматизація контролю замовлень постає як ключовий чинник підвищення загальної ефективності функціонування ресторанної мережі, формування конкурентних переваг і забезпечення стійкості в умовах зростаючих вимог до точності, швидкості та прозорості управлінських рішень [8].

1.3 Аналіз існуючих рішень і доцільність розробки власної системи

Застосування сучасних інформаційних технологій у ресторанному бізнесі є важливим інструментом оптимізації процесів обслуговування, обліку та управління.

Це дає змогу суттєво оптимізувати бізнес, підвищити якість обслуговування та спростити управлінські процеси. На ринку представлені готові рішення, що дозволяють автоматизувати роботу ресторану, від онлайн-замовлень та онлайн-реєстрації до управління запасами та розрахунків.

Готові програмні продукти орієнтовані на автоматизацію операцій закладів громадського харчування. Серед найпоширеніших рішень виділяються такі системи, як R-Keeper, Poster, Iiko, 1C, а також низка мобільних POS-систем.

Ці системи належать до таких класів як:

– «фронт-офіс» (автоматизація процесів продажів) – r_Keeper;

– «бек-офіс» (автоматизація товарооблікової системи) – 1С: Громадське харчування, StoreHouse;

– комплексні системи автоматизації ресторанного бізнесу, що поєднують «фронт-офіс» і «бек-офіс» (автоматизація складського, управлінського та фінансового обліків) - Iiko.

Однак цей поділ доволі умовний. Це пояснюється таким чином: коли система має велику кількість додаткових модулів, застосування яких дає змогу автоматизувати ті чи інші підрозділи ресторану, система автоматизація в кожному конкретному випадку може трансформуватися з фронт- або бек-офісної у комплексну систему автоматизації бізнесу [9]. Як приклад можна розглянути склад модулів системи r_Keeper, який можемо побачити на рис. 1.2.

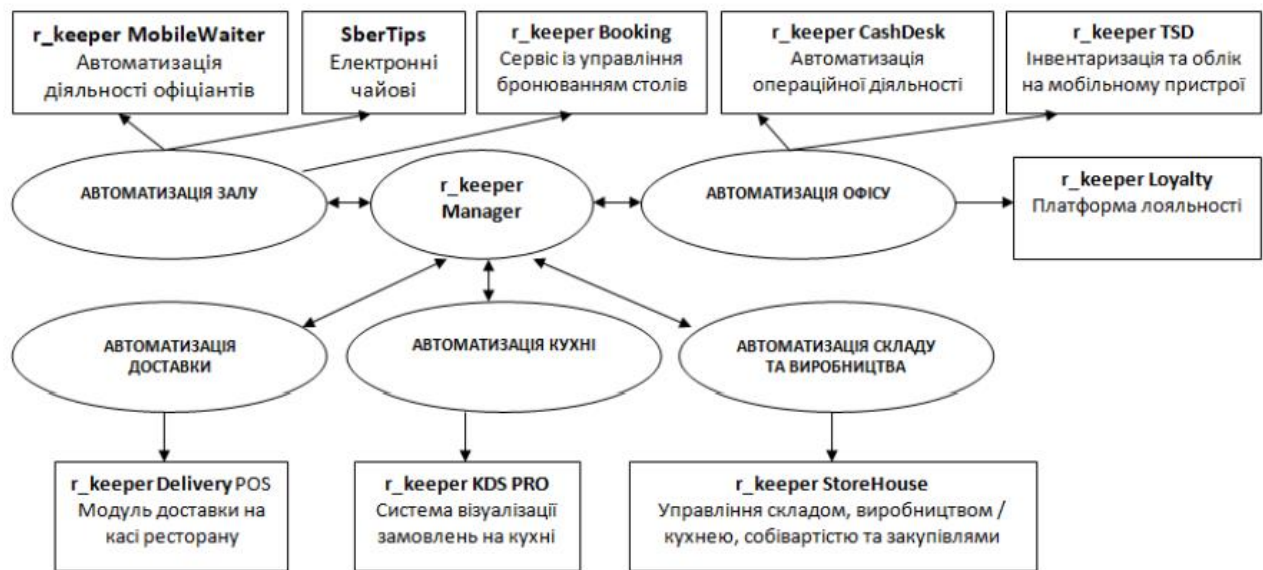


Рисунок 1.2 Система автоматизації ресторанного бізнесу R-Кеєр

Система r_Keeper містить як елементи бек-офісу (r_keeperTSD, r_keeper StoreHouse), так і елементи фронт-офісу (r_keeper MobileWaiter). При цьому такі модулі, як r_keeper KDS PRO, r_keeper TSD, r_keeper Delivery POS, r_keeper MobileWaiter, r_keeper Loyalty, SberTips і r_keeper Booking, не входять у базову поставку, а замовляються для встановлення додатково. Набір необхідних

додаткових модулів ресторан визначає самостійно, виходячи зі своїх можливостей і необхідності вирішення тих чи інших бізнес-завдань.

Система R-Keeper позиціонується як комплексне рішення для автоматизації великих ресторанів і мереж. Вона охоплює широкий спектр функцій: прийом замовлень, облік страв, контроль персоналу, аналітику тощо. Проте, її замкнута архітектура, складність налаштування та висока вартість обмежують можливості її адаптації під індивідуальні потреби, особливо в умовах середніх або регіональних мереж.

Poster – це хмарна POS-система, яка забезпечує базову автоматизацію замовлень, складу, каси та звітності. Вона вирізняється простотою у використанні й порівняно швидким впровадженням.

POS-система є сучасним технологічним рішенням, яке революціонізувало процес обслуговування клієнтів у ресторанах. Це потужний інструмент, що об'єднує функції каси, інвентаризації та управління персоналом. Завдяки POS-системі, офіціант може швидко та зручно вводити замовлення безпосередньо біля столика клієнта. Замовлення миттєво передається на кухню, що значно скорочує час очікування та підвищує ефективність роботи персоналу.

Окрім прийому замовлень, POS-системи забезпечують комплексний контроль над усіма аспектами роботи ресторану, дозволяючи:

- 1) управляти запасами – відстежувати кількість інгредієнтів, автоматично формувати замовлення постачальникам та мінімізувати втрати від псування продуктів;
- 2) оптимізувати фінансові процеси – здійснювати різні види оплати (готівкою, картою), формувати чеки, генерувати звіти про доходи та витрати;
- 3) збирати аналітичні дані – отримувати інформацію про популярність страв, середній чек, час обслуговування клієнтів та іншу важливу статистику для прийняття управлінських рішень.

Однак функціонал щодо контролю поставок, обліку постачальників та централізованої звірки товарів по мережі є дуже обмеженим. Poster більше зосереджений на процесах продажу, а не на логістичних або аналітичних задачах.

Більшість мобільних POS-рішень орієнтовані на малі заклади і мають переважно фронтову функціональність: прийом замовлень, виставлення чеків, зберігання історії продажів. У таких системах немає повноцінного механізму аналітичного контролю поставок, формування гнучких звітів, звірки фактичних та очікуваних даних або інтеграції з іншими внутрішніми модулями мережі [10].

Крім того, існуючі рішення часто вимагають щомісячної абонентської плати, обмежують можливості користувача у зміні логіки обліку або структури бази даних, а також потребують тривалого навчання персоналу. Більшість із них створено з урахуванням потреб західних ринків або великих франчайзингових мереж, що ускладнює або робить надмірно витратним їхнє впровадження у невеликих або регіональних закладах.

З практичної точки зору, такі обмеження створюють серйозні виклики для ефективного управління поставками. У випадку, коли йдеться не про облік продажів, а про контроль над прийомом товару, звірку з накладними, фіксацію обсягів і вартості поставок. Більшість із наявних систем не забезпечують необхідного функціоналу. У великих мережах ресторанів відсутність централізованої системи, що охоплює всі етапи поставки, призводить до втрати контролю, дублювання інформації, складнощів у координації логістики та помилок у плануванні закупівель.

Результати проведеного дослідження підтверджують необхідність автоматизації бізнес-процесів ресторанного підприємства як стратегічної перспективи та інструменту конкурентоспроможності на ринку ресторанних послуг. У цьому контексті доцільним і стратегічно обґрунтованим є створення власної автоматизованої системи контролю обліку товарів, що надходять від постачальників. Така система дозволяє адаптувати архітектуру та

функціональність під специфіку бізнес-процесів конкретної мережі, інтегруватися з бухгалтерськими модулями або CRM, вести централізовану базу даних з усіх філій, у режимі реального часу контролювати залишки та формувати звіти на різних рівнях.

Основою системи може слугувати реляційна база даних, яка забезпечує структуроване зберігання інформації, підтримує транзакційність, унікальність записів і взаємозв'язки між сутностями. Вона дозволяє виконувати запити будь-якої складності, будувати звіти, контролювати відхилення у поставках, а також масштабуватися при розширенні мережі. Завдяки поєднанню з мовою програмування Python система може забезпечити як введення даних, так і їх перевірку, візуалізацію та аналіз.

Крім технічної гнучкості, власна система дозволяє зменшити витрати на обслуговування, оскільки побудована на відкритих або безкоштовних інструментах. Можливість гнучкої модифікації логіки, без залежності від сторонніх постачальників ПЗ, є додатковою перевагою.

Отже, аналіз ринку ПЗ для ресторанної галузі показує, що попри велику кількість готових рішень, більшість із них не пристосовані для повноцінного контролю поставок у масштабі мережі ресторанів. В умовах зростаючої конкуренції, потреби в точному обліку, гнучкій аналітиці та централізованому контролі, розробка власної системи контролю постачань постає як ефективне, економічно виправдане та технологічно обґрунтоване рішення [9].

1.4 Переваги побудови системи на основі бази даних

У сучасному ресторанному бізнесі ефективне управління закупівлями та замовленнями є критично важливим фактором успіху. Особливо це актуально для мережі ресторанів, де потрібно координувати роботу багатьох закладів. Використання бази даних як основи для побудови системи контролю дає змогу значно підвищити точність, прозорість і швидкість прийняття рішень.

1. Оптимізація закупівель – система аналізує історію замовлень і прогнозує потреби кожного ресторану, що дозволяє закуповувати товари раціонально, уникаючи надлишків і дефіцитів.

2. Контроль витрат і бюджетування дає можливість порівнювати планові й фактичні витрати, а також можливість виявлення неефективних закупівель та перевищення бюджету на ранніх етапах [12].

3. Покращення роботи з постачальниками, а саме: збереження інформації про всі угоди, ціни, умови доставки.

4. Аналітика та звітність, а саме: формування статистичних звітів і графіків щодо закупівель, витрат, залишків, а також допомога у прийнятті стратегічних управлінських рішень на основі даних.

5. Масштабованість – система легко адаптується до розширення мережі, нові ресторани можуть швидко інтегруватися в єдину платформу управління.

6. Підвищення ефективності персоналу – менше ручної роботи, зниження навантаження на адміністраторів і менеджерів.

Завдяки впровадженню програмного забезпечення на основі бази даних відкриваються нові можливості для оптимізації процесів, підвищення продуктивності праці та зменшення витрат. Актуальність створення власної автоматизованої системи обумовлена не лише економічною доцільністю, але й потребою у гнучкості, безпеці даних та відповідності до конкретних вимог

закладу. Крім того, з огляду на доступність інструментів розробки, таких як Python та MySQL, можливо створити ефективне рішення без значних фінансових витрат, при цьому зберігаючи можливість масштабування і подальшого розвитку.

У теоретичному розумінні побудова такої системи ґрунтується на принципах автоматизації облікових процесів, інтеграції компонентів через базу даних, модульної структури програмного забезпечення та дотримання загальноприйнятих стандартів проектування інтерфейсів користувача. Важливим аспектом є також врахування досвіду впровадження аналогічних систем в інших галузях обліку та обслуговування, що дозволяє уникнути типових помилок і забезпечити практичну цінність розробки.

Сучасні підприємства вимагають найновіших методів управління. Ухвалення управлінських рішень в умовах постійно змінного, динамічного середовища вимагає не тільки аналізу, оцінки і прогнозування внутрішнього розвитку підприємства, але і забезпечення відповідності між зовнішнім мікросередовищем, зовнішнім макросередовищем і результатами фінансово-господарської діяльності підприємства, що, у свою чергу, висуває високі вимоги до інформаційного та інших видів забезпечення управління підприємством [11].

Таким чином, створення автоматизованої системи контролю замовлень для мережі ресторанів є актуальним технічним завданням, вирішення якого має на меті підвищення ефективності управління, покращення сервісу та зниження операційних ризиків. Така система стає не лише інструментом обліку, але й стратегічним засобом підвищення конкурентоспроможності ресторанного бізнесу.

Висновки до розділу 1

У першому розділі було розглянуто основи функціонування мережі ресторанів з централізованим типом управління, що передбачає єдину систему обліку та управління, спільні стандарти й уніфіковану інфраструктуру. Підтверджено, що в умовах зростання кількості закладів, обсягів замовлень та ускладнення логістики, критичною потребою стає автоматизація процесів контролю замовлень. Проведений аналіз ринку показав, що хоча існують потужні програмні продукти, як-от R-Keeper, Poster, Iiko, вони мають низку обмежень, пов'язаних з вартістю, закритістю архітектури або обмеженими можливостями адаптації. Саме це обґрунтовує доцільність розробки власної системи, адаптованої до конкретних потреб мережі. Запропонована система базуватиметься на централізованій базі даних, що забезпечить точність, оперативність, інтеграцію ключових функцій (замовлення, аналітика, облік) та можливість масштабування. Таким чином, створення власного рішення дозволить підвищити ефективність управління мережею ресторанів та оптимізувати витрати.

РОЗДІЛ 2 ТЕОРЕТИЧНИЙ ОПИС ПРОГРАМНИХ ІНСТРУМЕНТІВ

2.1 Таблиці та їх структура

MySQL реалізує реляційну модель, у якій інформація зберігається у вигляді таблиць (relations), що складаються з рядків (records) та стовпців (fields). Кожен стовпець характеризується певним типом даних, а кожен рядок є унікальним записом. Реляційна модель дозволяє використовувати мову структурованих запитів SQL для доступу, зміни та аналізу інформації, подібну структуру можемо побачити на рис. 2.1.

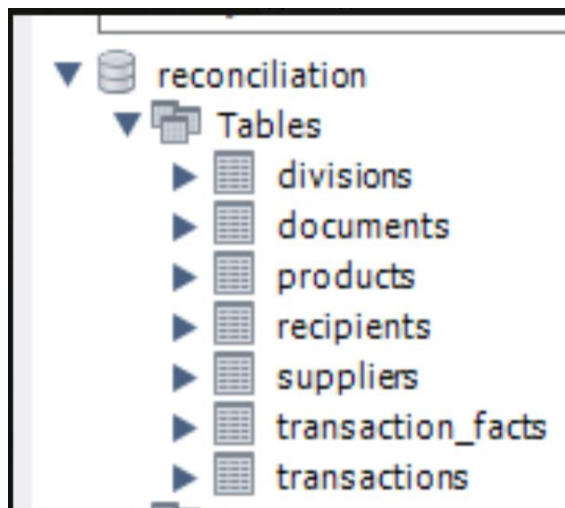


Рисунок 2.1 Структура таблиць бази даних

У проектуванні бази даних для автоматизованої системи контролю постачання товарів важливу роль відіграє логічне структурування інформації. Дані організовуються у вигляді таблиць, кожна з яких представляє окремий тип об'єктів або подій: наприклад, інформацію про постачальників, категорії товарів, обсяги поставок, чи факти приймання продукції. Такий підхід дозволяє зберігати

дані у впорядкованому вигляді, що спрощує як повсякденну роботу з системою, так і виконання аналітичних запитів.

Кожна таблиця має унікальний **первинний ключ (PRIMARY KEY)** – атрибут, що однозначно ідентифікує кожен запис. Це забезпечує уникнення дублювання та спрощує зв'язування даних між різними сутностями. Для встановлення взаємозв'язків між таблицями використовуються **зовнішні ключі (FOREIGN KEY)**, які дають змогу логічно поєднувати пов'язані записи. Наприклад, окремі поставки можуть бути пов'язані з відповідними постачальниками, а номенклатура товарів – з категоріями або одиницями виміру.

Застосування **реляційної моделі** дозволяє уникнути надмірного дублювання інформації, забезпечити узгодженість даних і гарантувати збереження логічних зв'язків між таблицями. Це особливо важливо у випадках, коли одні й ті самі товари можуть постачатися різними контрагентами, або коли потрібно контролювати повторювані поставки протягом тривалого періоду.

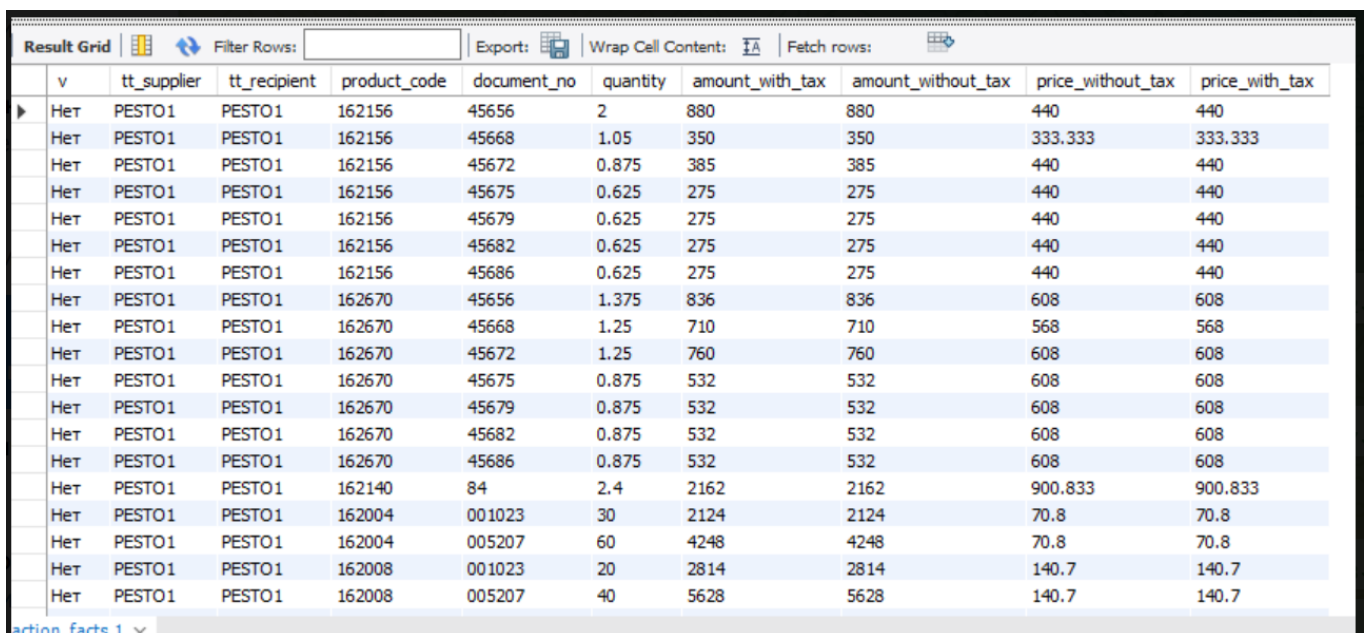
Завдяки такій структурі дані залишаються гнучкими та масштабованими: систему легко доповнювати новими функціями або звітами без суттєвих змін у вже наявній архітектурі. Наприклад, можливо без зусиль додати підтримку ведення залишків, перевірок якості, або фіксації невідповідностей у поставках.

Ще однією перевагою структурованої моделі є можливість ефективного використання SQL-запитів. Об'єднання таблиць за допомогою операторів JOIN, використання фільтрації, агрегації, сортування та умовних вибірок дозволяє формувати звіти й контролювати динаміку обігу товарів за різними критеріями. Наприклад, адміністрація ресторанної мережі може оперативно проаналізувати обсяги постачання певного товару за обраний період, відстежити зміну цін або частоту співпраці з конкретним постачальником.

Крім зв'язків і ключів, у проектуванні таблиць важливо обирати коректні типи даних: наприклад, для кількісних показників – DECIMAL або FLOAT, для дат – DATE, а для статусів чи категорій – ENUM або VARCHAR з обмеженням

переліком значень. Це підвищує надійність перевірки даних на етапі введення та зменшує ймовірність помилок [12].

Таким чином, правильно спроектована структура таблиць є основою стабільної та ефективної роботи всієї системи контролю. Вона забезпечує як зручність у щоденному використанні, так і можливість автоматизованої обробки, аналітики та розширення функціоналу в майбутньому – без втрати цілісності та керованості даних. На рисунку 2.2 можемо побачити одну з таблиць, яка містить всю необхідну інформацію, упорядковану належним чином.



v	tt_supplier	tt_recipient	product_code	document_no	quantity	amount_with_tax	amount_without_tax	price_without_tax	price_with_tax
Her	PESTO1	PESTO1	162156	45656	2	880	880	440	440
Her	PESTO1	PESTO1	162156	45668	1.05	350	350	333.333	333.333
Her	PESTO1	PESTO1	162156	45672	0.875	385	385	440	440
Her	PESTO1	PESTO1	162156	45675	0.625	275	275	440	440
Her	PESTO1	PESTO1	162156	45679	0.625	275	275	440	440
Her	PESTO1	PESTO1	162156	45682	0.625	275	275	440	440
Her	PESTO1	PESTO1	162156	45686	0.625	275	275	440	440
Her	PESTO1	PESTO1	162670	45656	1.375	836	836	608	608
Her	PESTO1	PESTO1	162670	45668	1.25	710	710	568	568
Her	PESTO1	PESTO1	162670	45672	1.25	760	760	608	608
Her	PESTO1	PESTO1	162670	45675	0.875	532	532	608	608
Her	PESTO1	PESTO1	162670	45679	0.875	532	532	608	608
Her	PESTO1	PESTO1	162670	45682	0.875	532	532	608	608
Her	PESTO1	PESTO1	162670	45686	0.875	532	532	608	608
Her	PESTO1	PESTO1	162140	84	2.4	2162	2162	900.833	900.833
Her	PESTO1	PESTO1	162004	001023	30	2124	2124	70.8	70.8
Her	PESTO1	PESTO1	162004	005207	60	4248	4248	70.8	70.8
Her	PESTO1	PESTO1	162008	001023	20	2814	2814	140.7	140.7
Her	PESTO1	PESTO1	162008	005207	40	5628	5628	140.7	140.7

Рисунок 2.2 Основна таблиця з даними

2.1.1 Типи даних у MySQL

Для кожного поля таблиці MySQL дозволяє вказати тип даних, що визначає характер та обсяг збереженої інформації. Основні категорії:

- 1) числові типи: INT, DECIMAL, FLOAT, BIGINT;

- 2) рядкові типи: VARCHAR, TEXT, CHAR;
- 3) дата та час: DATE, DATETIME, TIMESTAMP;
- 4) логічні значення: BOOLEAN або TINYINT(1).

У рамках розроблюваної системи широко застосовуються VARCHAR для найменувань, INT для ідентифікаторів, DECIMAL(10,2) для фінансових значень та DATE для полів із датами замовлень [12].

2.1.2 Обмеження цілісності

Щоб уникнути неузгодженостей у даних, у MySQL передбачено ряд обмежень:

- 1) NOT NULL – поле обов’язкове до заповнення;
- 2) UNIQUE – значення повинне бути унікальним;
- 3) PRIMARY KEY – визначає унікальний ідентифікатор запису;
- 4) FOREIGN KEY – вказує на зв’язок між таблицями;
- 5) DEFAULT – встановлює значення за замовчуванням.

Такі обмеження використовуються для забезпечення логічної цілісності структури бази даних [12].

2.1.3 Індексація

Індекс (INDEX) – це структура, що прискорює пошук за певними полями. У невеликих системах кількість індексів може бути обмежена, проте доцільно індексувати поля, за якими відбувається сортування або частий фільтр (наприклад, дати замовлення чи коди товарів). Створення індексу не впливає на структуру таблиці, але значно покращує швидкість запитів SELECT [12].

2.1.4 SQL – запити

У MySQL використовується стандарт SQL, що включає такі базові команди:

- 1) SELECT – вибірка даних;
- 2) INSERT – додавання записів;
- 3) UPDATE – зміна існуючих записів;
- 4) DELETE – видалення;
- 5) JOIN – об'єднання таблиць;
- 6) GROUP BY, ORDER BY – агрегація й сортування.

У рамках системи контролю замовлень переважно використовуються запити з фільтрами за датою, назвами товарів, постачальниками або фільтрація за сумами [12].

2.1.5 Нормалізація та засоби адміністрування

Нормалізація – це процес упорядкування структури бази даних з метою зменшення надлишковості та підвищення узгодженості. У системі використовується третя нормальна форма (3NF), яка передбачає, що всі атрибути таблиці залежать лише від первинного ключа. Це дозволяє уникнути дублювання однакової інформації та полегшує обслуговування БД.

Для взаємодії з базою даних можуть використовуватись як консольні інструменти (mysql, mysqldump), так і графічні середовища, наприклад, **phpMyAdmin** або **MySQL Workbench**. Вони дозволяють зручно виконувати структурування, імпорт/експорт, перевірку запитів та резервне копіювання бази [12].

2.2 Створення вкладеного інтерфейсу за допомогою віджета Notebook модуля Tkinter

У розробці графічних інтерфейсів на Python доцільно використовувати модуль Tkinter, який є вбудованим інструментом для створення GUI-додатків. Розширення tkinter.ttk надає доступ до сучасніших і візуально привабливіших віджетів, зокрема Notebook, що дозволяє будувати інтерфейси з багатьма вкладками [13].

Модуль ttk включає 18 базових віджетів, з яких 12 є оновленням стандартних компонентів (Button, Checkbutton, PanedWindow, Radiobutton, Entry,

Frame, Label, LabelFrame, Menubutton, Scale, Scrollbar, Spinbox) і 6 нових: Combobox, Notebook, Sizegrip, Progressbar, Separator, Treeview.

Використання `ttk.Notebook` доцільне у випадках, коли потрібно реалізувати інтерфейс із вкладеною структурою. Цей віджет керує набором вікон, кожне з яких відповідає окремій вкладці. При цьому одночасно відображається лише одна з них, і користувач може перемикатись між вкладками для доступу до різних розділів інтерфейсу [14].

2.2.1 Процес створення інтерфейсу

Процес створення такого інтерфейсу передбачає кілька послідовних етапів:

1. Спочатку імпортується модуль `tkinter` та його розширення `ttk`, що забезпечує доступ до покращених елементів керування.
2. Далі створюється головне вікно, ініціалізоване класом `Tk`, якому доцільно задати заголовок за допомогою відповідного методу.
3. На наступному етапі створюється сам елемент `Notebook`, що слугує контейнером для вкладок. Він приймає параметри, такі як `height`, `padding` та `width`, а також посилання на батьківське вікно.
4. Для реалізації вкладок створюються об'єкти типу `Frame`, які є контейнерами для групування інших елементів.
5. Кожна вкладка додається до `Notebook` із зазначенням текстової назви.
6. Весь елемент `Notebook` розміщується в головному вікні за допомогою менеджера розташування `pack`, де параметри `expand=1` та `fill="both"` забезпечують максимальне використання доступного простору.

7. На вкладках розміщуються віджети Label, які використовуються для виведення текстових повідомлень. Ці віджети підтримують набір параметрів, таких як anchor, font, padx, pady, text, width, height тощо, що дозволяє гнучко налаштовувати їх відображення.
8. Для розміщення Label застосовується метод grid, який поділяє батьківський контейнер на таблицю зі стовпцями і рядками, дозволяючи точно визначати позицію кожного елемента за допомогою параметрів column, row, padx, pady, sticky тощо.
9. На завершальному етапі графічний застосунок запускається за допомогою головного циклу mainloop, що забезпечує безперервну роботу інтерфейсу та реагування на дії користувача.

Загалом, використання віджета Notebook у поєднанні з іншими елементами ttk надає широкі можливості для створення інтуїтивно зрозумілих і функціональних інтерфейсів, особливо актуальних у розробці прикладного програмного забезпечення для автоматизації бізнес-процесів [15,16].

Висновки до розділу 2

Ефективне проєктування бази даних є основоположним фактором для побудови надійної автоматизованої системи контролю замовлень у мережі ресторанів. Результати опрацювання особливостей MySQL продемонстрували, що реляційна модель, за умови правильного структурування таблиць і застосування обмежень цілісності, дозволяє створити гнучку й масштабовану інформаційну основу для майбутньої системи.

Забезпечення логічних зв'язків між сутностями, чітке визначення типів даних та дотримання нормалізації сприяють підвищенню точності обліку, мінімізації помилок і дублювання, а також спрощують обслуговування бази у процесі експлуатації. Важливим чинником є і підтримка індексації, що дозволяє зберегти високу швидкодію при виконанні складних вибірок, що особливо актуально для оперативної аналітики та формування звітів у реальному часі.

Особливу увагу заслуговує можливість поступового розширення функціональності без втрати керованості даними. Це відкриває перспективи для подальшої автоматизації суміжних процесів: обліку залишків, оцінки надійності постачальників, аналізу тенденцій замовлень тощо.

Таким чином, база даних у MySQL виконує не лише роль сховища інформації, а й стає критичним елементом, що визначає ефективність, прозорість та керованість усієї системи замовлень. Її грамотне проектування закладає передумови для надійної роботи системи в умовах змін, навантаження й подальшої інтеграції з іншими компонентами.

РОЗДІЛ 3 ОПИС РЕЗУЛЬТАТІВ РЕАЛІЗАЦІЇ ПРОГРАМИ

3.1 Архітектура застосунку та логіка роботи

У рамках практичної частини проєкту було реалізовано прототип автоматизованої системи контролю замовлень у мережі ресторанів. Основна мета – забезпечити ефективну обробку транзакцій, автоматизовану звірку облікових даних, аналітику цінової динаміки та виявлення аномальних значень, що можуть свідчити про помилки або потенційні порушення.

Для реалізації програмного забезпечення було обрано технологічний стек, що поєднує зручність мови програмування Python, можливості обробки табличних даних за допомогою Pandas, просту побудову графічного інтерфейсу з Tkinter, а також потужність реляційної СУБД MySQL для зберігання даних. Такий вибір обґрунтовано наступними чинниками:

Python – мова, яка широко використовується в аналітиці даних, має зрозуманий синтаксис, велику кількість бібліотек та активну спільноту. Вона дозволила швидко реалізувати як бізнес-логіку, так і інтерфейс користувача.

Tkinter – стандартний GUI-інструмент у складі Python, що забезпечує створення настільних застосунків без потреби у встановленні додаткових модулів.

Pandas – потужна бібліотека для обробки табличних даних, яка дозволяє виконувати сортування, фільтрацію, агрегацію, зведення та аналіз великих обсягів інформації.

MySQL – популярна система керування базами даних, що забезпечує надійне зберігання та ефективний доступ до транзакційної інформації. Через бібліотеку `mysql.connector` реалізовано зручну інтеграцію з Python.

Архітектура застосунку має модульну структуру та складається з трьох основних програмних модулів на Python і однієї централізованої бази даних MySQL. Модулі системи:

1. **app.py** – графічний інтерфейс та основна логіка взаємодії з користувачем. Цей модуль відповідає за ініціалізацію вікна програми та реалізацію ключових сценаріїв роботи:

- 1) створення головного вікна з чотирма вкладками: звірка файлів, зведена таблиця, аномалії та графік цін;
- 2) завантаження Excel файлів за допомогою діалогових вікон;
- 3) запуск процесу звірки даних із виводом результатів у вигляді інтерактивної таблиці;
- 4) можливість експорту оброблених результатів до нового Excel файлу.

2. **pivottable.py** – модуль побудови агрегованих звітів. Модуль виконує підключення до бази даних, отримання інформації з таблиці транзакцій та побудову аналітичного звіту:

- 1) групування даних за товарними одиницями (номенклатурою) та обчислення сумових показників;
- 2) визначення найчастіше зустрічаючихся цін — це дозволяє відслідковувати сталість або коливання вартості;
- 3) обчислення середньої вартості одиниці товару, що особливо важливо для контролю стабільності закупівель;
- 4) побудова та візуалізація зведеної таблиці, з можливістю подальшого аналізу та вивантаження.

3. **anomalies.py** – модуль виявлення статистичних відхилень. Цей компонент відповідає за аналіз транзакцій з метою виявлення підозрілих записів:

- 1) отримання даних з MySQL, групування за позиціями товарів;
- 2) розрахунок Z-оцінок, що дозволяє виявляти статистичні аномалії у цінах – записи, які суттєво відрізняються від середнього значення ($|Z| > 3$);
- 3) побудова списку підозрілих транзакцій для перегляду та подальшого аналізу або верифікації.

База даних. У MySQL реалізовано структуру, яка включає основну таблицю transactions, що зберігає дані по всіх транзакціях, а також допоміжні довідники: постачальників, отримувачів, номенклатури товарів, пов'язані документи. Завдяки цій структурі забезпечується цілісність даних, зв'язність та можливість виконання складних запитів для подальшого аналізу. Крім того, централізована база відкриває перспективу розширення системи до багатокористувацького режиму або інтеграції з зовнішніми обліковими системами.

Переваги обраної архітектури. Перше – це продуктивність, бо використання Pandas та SQL запитів дозволяє швидко опрацьовувати великі масиви даних. Друге – гнучкість, бо модульна структура дозволяє легко масштабувати застосунок, додаючи нові функції. Також надійність, бо централізована база даних гарантує збереження, резервування та консистентність даних. І, звісно, автономність, бо застосунок працює локально, без потреби в підключенні до інтернету, що важливо для захисту конфіденційної інформації.

У результаті реалізована система дозволяє суттєво скоротити час, необхідний для ручного звіряння фінансових даних, мінімізувати людський фактор, своєчасно виявляти відхилення та аномалії, а також генерувати звіти для керівництва.

3.2 Опис створеної бази даних

Для забезпечення повноцінної роботи застосунку розроблено структуру бази даних, що є основним джерелом інформації для аналізу. Програма побудована таким чином, що здійснює обробку та аналіз даних, збережених у базі даних MySQL. Такий підхід забезпечує гнучкість, масштабованість і зручність в обслуговуванні системи. Водночас архітектура застосунку дозволяє легко адаптувати його до роботи з іншими програмними продуктами, такими як Profit, Poster POS, r_keeper та інші системи обліку й управління, що підтримують взаємодію з базами даних.

Розроблена база даних складається з головної таблиці, яку бачили раніше на рис. 2.2, в якій міститься вся потрібна інформація, і 6 маленьких, що зберігають окрему інформацію, для зручності.

Таблиця «products», яку можемо побачити на рис. 3.1, потрібна для акумулювання всіх товарів без повторень, вона включає в себе стовпці:

- product_code (Код);
- product_group (Група);
- product_name (Назва);
- unit (Од.).

До цієї таблиці відбувається звернення під час формування списку всіх товарів без повторень у вкладці «Графік цін», а також для складання зведеної таблиці у вкладці «Зведена таблиця».

	product_code	product_group	product_name	unit
	162573	сировина та матеріали	гастрономія панчетта кг	кг
	162608	сировина та матеріали	Мука Марія кг	кг
	162737	сировина та матеріали	Бакалея Уксус малиновий л	лтр
	162799	сировина та матеріали	Сир пармезан тертий кг	кг
	162828	сировина та матеріали	гастрономія грудинка карпаччо порц	кг
	44	сировина та матеріали	Соус кетчуп томатний кг	кг
	133	сировина та матеріали	Соус соєвий концентрат	лтр
	189	сировина та матеріали	Соус гірчиця	кг
	102001	сировина та матеріали	олія соняшникова	лтр
	162146	сировина та матеріали	Конс. Ананас	кг

Рисунок 3.1 Таблиця «products»

Таблиця «documents», яку можемо побачити на рис. 3.2, потрібна для акумулювання всіх документів без повторень, вона включає в себе стовпці:

- document_no (Документ);
- document_date (Дата);
- comment (Коментар).

До цієї таблиці створюється запит під час формування звіту про аномальні ціни у вкладці «Аномалії» та під час формування графіка цін.

Result Grid			
		Filter Rows:	
		Export:	
document_no	document_date	comment	
45749	2025-04-02 00:00:00		
45752	2025-04-05 00:00:00		
455756	2025-04-09 00:00:00		
45759	2025-04-12 00:00:00		
45763	2025-04-16 00:00:00		
45766	2025-04-19 00:00:00		
45770	2025-04-23 00:00:00		
45773	2025-04-26 00:00:00		
00878	2025-04-08 00:00:00		
067925	2025-04-03 00:00:00		
068867	2025-04-08 00:00:00		
0720225	2025-04-17 00:00:00		
073214	2025-04-22 00:00:00		
075867	2025-04-30 00:00:00		
2209849461	2025-04-08 00:00:00		
2209862723	2025-04-17 00:00:00		
2209869268	2025-04-22 00:00:00		
2209864903	2025-04-18 00:00:00		
10564	2025-04-02 00:00:00		
12333	2025-04-16 00:00:00		
13930	2025-04-30 00:00:00		
4504-0186	2025-04-01 00:00:00		
16035	2025-04-01 00:00:00		
16767	2025-04-04 00:00:00		
17394	2025-04-08 00:00:00		
18033	2025-04-11 00:00:00		
18819	2025-04-15 00:00:00		

Рисунок 3.2 Таблица «documents»

Таблица «suppliers», яку можемо побачити на рис. 3.3, потрібна для акумулювання всіх постачальників без повторень, вона включає в себе стовпці:

- tt_supplier (ТТ постачальник);
- supplier_name (Постачальник).

	tt_supplier	supplier_name
▶	PESTO1	ринок
	PESTO1	НЕСТЛЕ ЧАЙ
	PESTO1	ПРОДМАРКЕТ
	PESTO1	КОКА-КОЛА
	PESTO1	БАЯДЕРА
	PESTO1	суші-ХАУЗ
	PESTO1	ПІТАНИУМ
	PESTO1	ОВОА
	PESTO1	РУДЬ
	PESTO1	ГРИГОРЕНКО ІТАЛІЯ
	PESTO1	ІТА
	PESTO1	ХОЗтовари "ТО РЕСТ"
	PESTO1	хозсервіс

Рисунок 3.3 Таблица «suppliers»

Таблиця «recipients» , яку можемо побачити на рис. 3.4, потрібна для акумулювання всіх отримувачів без повторень, вона включає в себе стовпці:

- tt_recipient (ТТ отримувач);
- recipient_name (Отримувач).

tt_recipient	recipient_name
PESTO1	СПД-ФОП
PESTO1	ООО ТПЕСТО СТАР
PESTO3	СПД-ФОП
PESTO3	ООО ТПЕСТО СТАР
PESTO4	СПД-ФОП
PESTO4	ООО ТПЕСТО СТАР
PESTO5	СПД-ФОП
PESTO5	ООО ТПЕСТО СТАР
PESTO6	СПД-ФОП
PESTO6	ООО ТПЕСТО СТАР
PESTO7	СПД-ФОП
PESTO7	ООО ТПЕСТО СТАР
PESTO8	СПД-ФОП
PESTO8	ООО ТПЕСТО СТАР
PESTO9	СПД-ФОП
PESTO9	ООО ТПЕСТО СТАР
PESTO10	СПД-ФОП
PESTO10	ООО ТПЕСТО СТАР
MISTER T...	СПД-ФОП
MISTER T...	ООО ТПЕСТО СТАР
PESTO11	СПД-ФОП
PESTO11	ООО ТПЕСТО СТАР
PESTO12	СПД-ФОП

Рисунок 3.4 Таблиця «recipients»

Таблиця «divisions», яку можемо побачити на рис. 3.5, потрібна для акумулювання всіх підрозділів без повторень, вона включає в себе стовпці:

- tt_supplier (ТТ постачальник);
- supplier_division (Підрозділ постачальник);
- tt_recipient (ТТ отримувач);
- recipient_division (Підрозділ отримувач).

tt_supplier	supplier_division	tt_recipient	recipient_division
PESTO18		PESTO18	УПАКОВКА
PESTO18		PESTO18	КАРАОКЕ
PESTO19		PESTO19	КУХНЯ
PESTO19		PESTO19	УПАКОВКА
PESTO19		PESTO19	БАР
PESTO2		PESTO2	КУХНЯ
PESTO2		PESTO2	БАР

Рисунок 3.5 Таблиця «divisions»

Після створення довідкових таблиць реалізована таблиця «transaction_facts» (можемо побачити її на рис. 3.6), у якій зберігається інформація про поставки та всі операції, вона включає в себе стовпці:

- v (V);
- tt_supplier (ТТ постачальник);
- tt_recipient (ТТ отримувач);
- product_code (Код);
- document_no (Документ);
- quantity (Кіль-сть);
- amount_with_tax (Сума с податком);
- amount_without_tax (Сума без податку);
- price_without_tax (Ціна без податку);
- price_with_tax (Ціна з податком);

Ця таблиця є ключовим елементом аналітичної моделі, оскільки об'єднує в собі значущі поля всіх довідкових таблиць і дозволяє легко здійснювати агрегацію, фільтрацію та обчислення на основі операційних даних.

	v	tt_supplier	tt_recipient	product_code	document_no	quantity	amount_with_tax	amount_without_tax	price_without_tax	price_with_tax
▶	Нет	PESTO1	PESTO1	162156	45749	0.5	220	220	440	440
	Нет	PESTO1	PESTO1	162156	45752	0.625	275	275	440	440
	Нет	PESTO1	PESTO1	162156	455756	0.25	110	110	440	440
	Нет	PESTO1	PESTO1	162156	45759	1	608	608	608	608
	Нет	PESTO1	PESTO1	162156	45763	0.625	275	275	440	440
	Нет	PESTO1	PESTO1	162156	45766	0.375	165	165	440	440
	Нет	PESTO1	PESTO1	162156	45770	0.75	330	330	440	440
	Нет	PESTO1	PESTO1	162156	45773	0.5	220	220	440	440
	Нет	PESTO1	PESTO1	162670	45749	1	608	608	608	608
	Нет	PESTO1	PESTO1	162670	45752	1	608	608	608	608
	Нет	PESTO1	PESTO1	162670	455756	0.875	532	532	608	608
	Нет	PESTO1	PESTO1	162670	45759	0.25	110	110	440	440
	Нет	PESTO1	PESTO1	162670	45763	1.25	760	760	608	608
	Нет	PESTO1	PESTO1	162670	45766	0.75	456	456	608	608
	Нет	PESTO1	PESTO1	162670	45770	1.125	684	684	608	608

Рисунок 3.6 Таблиця «transaction_facts»

3.3 Ініціалізація графічного інтерфейсу

Графічний інтерфейс користувача (GUI) реалізовано у модулі `app.py` із використанням бібліотеки Tkinter – стандартного інструменту Python для створення настільних застосунків. При проєктуванні інтерфейсу було зроблено акцент на простоті, зручності та інтуїтивно зрозумілій навігації, що є особливо важливим у задачах автоматизації для бухгалтерського або аналітичного персоналу, який може не мати технічної підготовки.

Основні параметри головного вікна. Під час ініціалізації застосунку створюється головне вікно розміром 1000×600 пікселів, яке є достатнім для комфортного відображення таблиць, графіків і елементів керування. Вікно має фіксований розмір для уникнення спотворення вмісту при зміні масштабу. Також встановлено заголовок вікна з назвою застосунку. Побачити це можна на рис. 3.7.

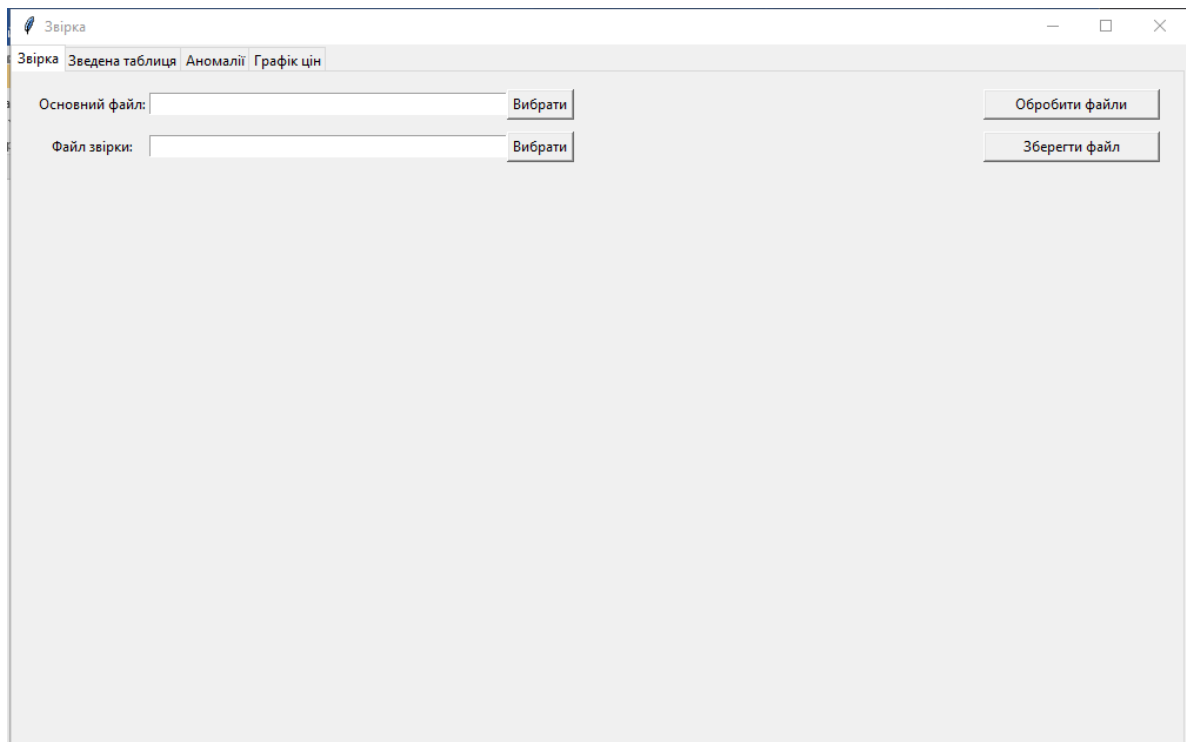


Рисунок 3.7 Головне вікно застосунку з вкладками.

Використання ttk.Notebook. Для реалізації багатосторінкового інтерфейсу використовується елемент `ttk.Notebook`, який дозволяє створити вкладки (tabs) у межах одного вікна. Це забезпечує логічну структуру взаємодії та дає змогу чітко розділити функціональні модулі системи. Такий підхід робить інтерфейс зручним для користувача, дозволяючи швидко перемикатися між різними аналітичними задачами без перевантаження екрана.

Кожна вкладка є окремим фреймом (`ttk.Frame`), що додається до `Notebook`, та містить власні елементи інтерфейсу, специфічні для виконання певного завдання. Опис вкладок інтерфейсу:

1. «Звірка». Ця вкладка надає можливість завантаження та порівняння двох Excel файлів з метою виявлення розбіжностей у фінансових документах. Вона призначена для бухгалтерської звірки між внутрішніми обліковими системами або між різними юридичними особами.

2. «Зведена таблиця». У цій вкладці користувач може переглядати агреговані дані з бази, зведені за товарними категоріями, з розрахунками сум, середніх цін, найчастіших значень тощо. Це дає змогу отримати узагальнене уявлення про структуру закупівель і зміну цін.

3. «Аномалії». Призначена для виявлення статистичних відхилень у даних. Вкладка реалізує автоматичний аналіз на основі Z-оцінки та дозволяє відстежувати підозрілі транзакції, які можуть бути результатом помилок або аномальних змін на ринку.

4. «Графік цін». Візуалізує часові ряди зміни цін на окремі товари або групи товарів. Це важливий інструмент для моніторингу постійних коливань на ринку та допомагає у виявленні трендів, сезонності або цінових сплесків.

Узагальнення. Завдяки використанню бібліотеки Tkinter у поєднанні з `ttk.Notebook`, вдалося створити інтуїтивно зрозумілий багатовіконний інтерфейс, який є функціональним, стабільним і легко адаптується під розширення у майбутньому. Кожна вкладка відповідає за окремий етап аналізу або обробки

даних, що дозволяє користувачеві ефективно виконувати завдання без потреби взаємодіяти з кодом або базами даних безпосередньо.

3.4 Опис вкладки «Звірка»

3.4.1 Загальне призначення та принцип роботи

Вкладка "Звірка" призначена для реалізації функціоналу автоматизованого порівняння облікових даних між двома незалежними джерелами у форматі електронних таблиць Excel. Цей функціонал дає змогу виявляти розбіжності у фінансових або товарних операціях шляхом прямого порівняння відповідних записів за ключовими параметрами. Застосування вкладки дозволяє скоротити час на рутинну звірку, знизити ймовірність людських помилок та забезпечити об'єктивність процесу перевірки. Процес звірки орієнтовано на бухгалтерський та фінансовий персонал підприємств, які працюють з великими обсягами транзакційних даних, що надходять з різних інформаційних систем, зокрема з внутрішніх ERP або зовнішніх платформ постачальників.

Для виконання програми необхідно надати два файли формату .xls або .xlsx. Перший файл повинен містити в собі основні дані поставок, а другий файл - дані, які необхідно перевірити на збіг. Застосунок зчитує документи стовпця "Документи" першого файлу та зі стовпця "Номер документа" другого файлу, а потім перевіряє дані другого орієнтуючись на дані з першого файлу. Всі дані файлів наводяться до одного формату.

Наступним кроком програма створює словник відповідності “Документ – ТТ отримувач” з першого файлу, і на його основі визначає значення “ТТ одержувач” для кожного запису у другому файлі. Також виконується звірка сум: для кожного документа знаходиться відповідна сума з першого файлу, після чого перевіряється, чи не відрізняється вона від суми у другому файлі більш ніж на 1%. У разі невідповідності користувач отримує відповідне повідомлення у стовпці “Перевірка суми”. Схему процесу звірки можемо побачити на рис. 3.8.

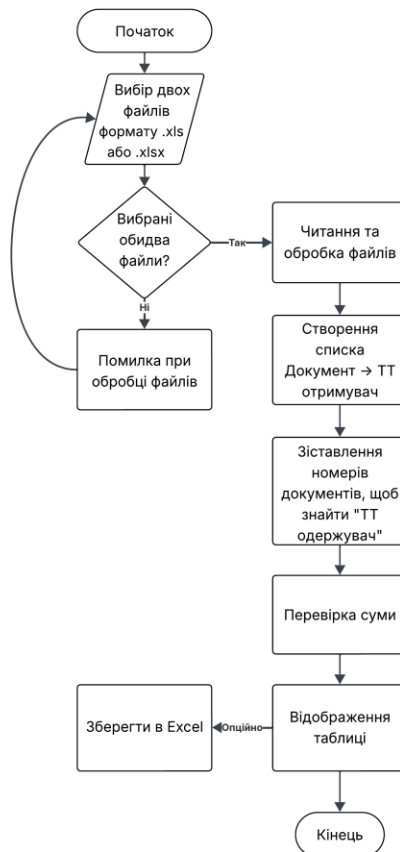


Рисунок 3.8 Схема процесу звірки

Результат обробки відображається у вигляді таблиці у графічному інтерфейсі. Користувач може зберегти отриману таблицю у файл Excel, натиснувши кнопку “Зберегти файл”. При збереженні автоматично виконується вирівнювання ширини колонок для зручності подальшого перегляду.

3.4.2 Структура інтерфейсу та логіка взаємодії

Інтерфейс вкладки реалізовано з використанням стандартних віджетів бібліотеки Tkinter та організовано за принципами вертикального та горизонтального компоновання через менеджери геометрії pack і grid. Функціональні елементи розміщено у логічній послідовності, яка відповідає типовому порядку дій користувача — від завантаження файлів до перегляду та збереження результатів. До основних елементів інтерфейсу належать:

Надписи Label. Мітки label1 та label2 відображають підказки до полів вибору файлів. Вони містять відповідно написи "Основний файл" та "Файл звірки", які орієнтують користувача у процесі взаємодії.

Поля вводу Entry. Текстові поля entry_file1 та entry_file2 відображають абсолютні шляхи до вибраних Excel файлів. Поля є недоступними для ручного редагування та оновлюються автоматично після вибору файлу.

Кнопки Button. Кнопки btn_file1 та btn_file2 відкривають стандартне діалогове вікно для вибору файлу за допомогою `filedialog.askopenfilename()`. Кнопка btn_process запускає основну процедуру звірки файлів — виклик функції `process_files()`, яка виконує обробку даних, порівняння ключових параметрів і формування результату. Кнопка btn_export надає можливість експортувати результати звірки у формат Excel для подальшого збереження або аналізу.

Таблиця результатів pandastable Table. Для відображення звірених даних використовується таблиця, реалізована через віджет `pandastable.Table`, вбудований у відповідний фрейм. Цей компонент забезпечує інтерактивне представлення результатів, включаючи фільтрацію, прокручування та сортування записів.

У таблиці відображаються такі основні колонки: Дата документа, Номер документа, Сума, ТТ одержувач, Перевірка суми.

Якщо у файлі звірки присутній запис, що відсутній у первинному файлі, у полі "ТТ одержувач" з'являється позначка "Помилка". Колонка "Перевірка суми" заповнюється лише для тих записів, де відносне відхилення між сумами в обох файлах перевищує 1 відсоток. Це дозволяє швидко виявляти критичні розбіжності без необхідності повного ручного перегляду.

Загалом вкладка "Звірка", яку зображено на рис. 3.9, реалізує автоматизовану процедуру контролю достовірності облікових даних із можливістю інтерактивного аналізу та експорту результатів, що значно підвищує ефективність аудиту та моніторингу фінансових операцій. Результат роботи можемо побачити на рис. 3.10.

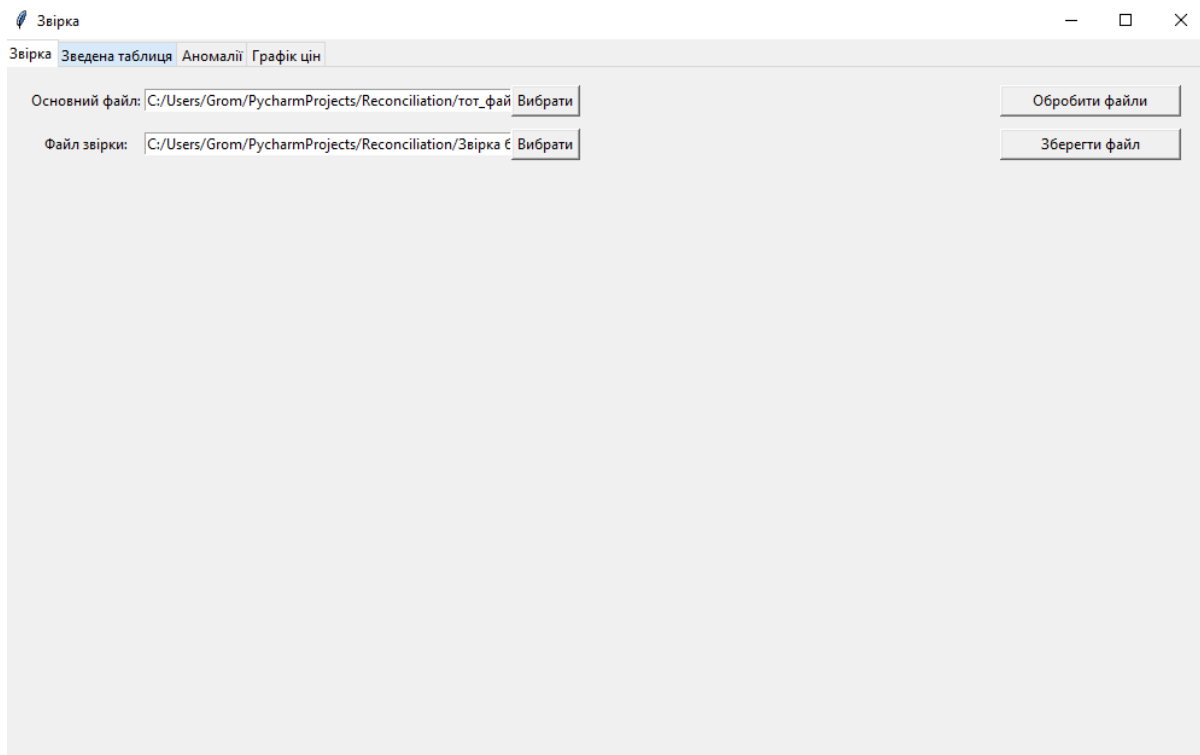


Рисунок 3.9 Вкладка «Звірка» до аналізу.

Звірка

Зведена таблиця Аномалії Графік цін

Основний файл: C:/Users/Grom/PycharmProjects/Reconciliation/тот_файл Вибрати

Файл звірки: C:/Users/Grom/PycharmProjects/Reconciliation/файл от Вибрати

Обробити файли

Зберегти файл

	Дата документа	Номер докумен	Сума	ТТ одержувач	Перевірка суми
3	2025-01-03	2209728401	1000.36	MISTER TWISTER	Сума не збігається (3753.359999999999)
4	2025-01-03	2209728086	4246.56	Помилка	Помилка
5	2025-01-03	2209728413	1407.12	PESTO5	
6	2025-01-03	2209728414	2018.88	PESTO5	
7	2025-01-03	2209728379	2408.40	PESTO7	
8	2025-01-03	2209728380	1407.12	PESTO7	
9	2025-01-07	2209732432	2331.36	PESTO17	
10	2025-01-07	2209732227	455.52	PESTO7	
11	2025-01-07	2209732228	3092.40	PESTO7	
12	2025-01-07	2209732052	7738.56	PESTO14	
13	2025-01-07	2209731986	951.60	PESTO13	
14	2025-01-07	2209731987	5771.52	PESTO13	
15	2025-01-07	2209732164	1998.00	PESTO 20	
16	2025-01-07	2209732124	959.40	PESTO8	
17	2025-01-07	2209732125	4511.52	PESTO8	
18	2025-01-07	2209732263	8444.16	EPI 1	
19	2025-01-07	2209732264	3131.28	PESTO18	
20	2025-01-07	2209731441	455.52	PESTO9	
21	2025-01-07	2209731442	6338.16	PESTO9	

97 rows x 5 columns

Рисунок 3.10 Вкладка «Звірка» після аналізу.

3.5 Опис вкладки «Зведена таблиця»

3.5.1 Загальне призначення

Вкладка "Зведена таблиця" призначена для інтерактивного перегляду та аналізу агрегованих метрик за всіма товарами на основі даних із бази даних. Цей функціональний блок дозволяє користувачеві в реальному часі оновити дані з СУБД, сформувати аналітичну зведену таблицю та зберегти її у форматі Excel. Таким чином забезпечується можливість гнучкого аналізу фінансових показників, обсягу закупівель та цінової політики по номенклатурі товарів у мережі ресторанів.

3.5.2 Структура інтерфейсу та логіка взаємодії

Інтерфейс вкладки реалізовано за допомогою менеджера геометрії `pack` і логічно поділено на дві основні області:

- 1) панель керування з інтерактивними елементами управління;
- 2) область відображення аналітичної таблиці.

У верхній частині вкладки розташовано дві основні кнопки:

1. Кнопка «Оновити дані» (`btn_refresh`) викликає функцію `show_pivot_table()`, яка ініціює звернення до функції `generate_pivot()` із модуля `pivottable.py`. У процесі виконання запиту здійснюється підключення до бази даних MySQL, отримання даних із таблиці `transactions`, SQL-групування інформації за найменуванням товарів і обчислення агрегованих показників.
2. Кнопка «Експорт в Excel» (`btn_export`) дає змогу зберегти сформовану зведену таблицю в окремий файл формату Excel для подальшого аналізу або звітності.

У нижній частині вкладки відображається таблиця результатів, реалізована за допомогою віджета `pandastable.Table`. Цей компонент дає змогу взаємодіяти з табличними даними у графічному режимі: переглядати, прокручувати, копіювати та аналізувати структуру даних безпосередньо в межах застосунку. Таблиця містить п'ять ключових колонок:

- 1) назва – найменування товару відповідно до записів у базі;
- 2) кількість – сумарний обсяг замовленого товару (агрегований показник);
- 3) сума з податком – повна вартість закупівель із урахуванням податку;
- 4) середня ціна – середнє значення ціни, що розраховується як відношення суми з податком до кількості, округлене до двох знаків після коми;
- 5) топ 3 ціни – три найчастіше вживані ціни для кожного товару, виведені у форматі списку, розділеного комами.

Формування поля "Топ 3 ціни" реалізовано шляхом підрахунку моди значень ціни за допомогою функціоналу бібліотеки Pandas. У разі рівнозначної частоти кількох значень у топ потрапляють усі, що відповідають цій частоті, з обмеженням до трьох позицій. Таким чином, вкладка "Зведена таблиця", яку зображено на рис 3.11, виступає в ролі аналітичного інструменту для оперативного моніторингу цінової політики та фінансових витрат у мережі ресторанів. Автоматизований підхід до побудови таблиці дозволяє підвищити ефективність прийняття управлінських рішень, зменшити вплив людського фактора та прискорити аналіз ключових показників діяльності підприємства.

Звірка Зведена таблиця Аномалії Графік цін					
Оновити дані		Експорт в Excel			
	Назва	Кількість	Сума з пода	Середня цін	Топ-3 ціни
1	Сир моцарелла	2996.63	660793.43	220.51	165.00, 286.00, 572.00
2	олія соняшникова	3252.80	275117.60	84.58	97.40, 64.19, 69.77
3	вершки	2084.70	264689.80	126.97	117.90, 232.33, 121.05
4	Сир пармезан тертий	389.80	214653.50	550.67	550.00, 600.00, 0.00
5	Зелень Салат Романо	715.35	214392.79	299.70	275.00, 243.75, 400.00
6	Овочі перець болгарск	791.85	190314.45	240.34	235.75, 253.00, 241.50
7	суші креветки тигрові	415.00	189663.30	457.02	457.02
8	Молоко	2892.00	148857.12	51.47	52.23, 55.13, 49.62
9	С/м Овочі картофель	1932.50	133791.00	69.23	69.00, 82.80
10	Овочі гриби шампінй	1040.77	130435.87	125.33	121.55, 123.90, 128.80
11	Фрукти апельсин	1901.74	129419.85	68.05	68.05, 68.06, 68.66
12	Овочі картофель чищ	1839.27	116305.83	63.23	63.80, 62.10, 47.85
13	Пиво Хугарден кег	810.00	112560.00	138.96	140.70, 93.80
14	Пиво Стелла Артуа к	1560.00	110448.00	70.80	70.80
15	соління Томати в влас	1320.00	109200.00	82.73	84.00, 0.00
16	Яловичий антрикот	317.06	106911.28	337.20	337.20
17	Вода кока-кола ж/б шт	5376.00	105663.44	19.65	19.68, 18.90, 16.87
18	м'ясо стейк рибай кг	149.16	104412.00	700.00	700.00
19	Вино Castelnovo Pros	271.25	103079.70	380.02	378.62, 378.63, 340.76
20	Фрукти лимон	997.10	100530.58	100.82	87.75, 115.50, 112.35
21	морозиво	394.00	100195.14	254.30	270.41, 229.15, 252.72
22	Вакумований лосось с	116.56	90916.80	780.00	780.00
23	Упаковка Сети на сто	100.00	90000.00	900.00	900.00

Рисунок 3.11 Вкладка «Зведена таблиця» після аналізу.

3.6 Опис вкладки «Аномалії»

3.6.1 Загальне призначення та принцип роботи

Вкладка "Аномалії" призначена для автоматизованого виявлення статистичних викидів у даних про ціни товарів на основі аналізу транзакцій з бази даних. Цей інструмент дозволяє ідентифікувати атипові значення, які суттєво відрізняються від характерного цінового рівня по кожному товару, що є важливим етапом у виявленні помилок, зловживань або неочікуваних змін у закупівельній політиці.

Щоб побудувати цю таблицю здійснюється SQL-запит, який об'єднує таблиці `transaction_facts`, `products` та `documents`, щоб отримати наступну інформацію:

- номер документа (`document_no`),
- дату операції (`document_date`),
- назву товару (`product_name`),
- ціну з податком (`price_with_tax`),
- ТТ постачальника (`tt_supplier`).

3.6.2 Структура інтерфейсу та логіка взаємодії

Інтерфейс вкладки побудовано з використанням геометричного менеджера `pack` із вертикальним розміщенням елементів. Основні компоненти користувацького інтерфейсу наведено нижче:

Кнопки Button. Кнопка «Оновити дані» (btn_refresh_anom) викликає функцію show_anomaly_table(). Вона звертається до функції generate_anomalies() з модуля anomalies.py, яка підключається до бази даних MySQL, виконує SQL-запит для отримання транзакцій та розраховує z-оцінки для кожної ціни. Z-оцінка дає змогу оцінити ступінь відхилення ціни від середнього значення у відповідній товарній групі. Записи, у яких модуль z-оцінки перевищує заданий поріг (наприклад, $|z| > 3$), визначаються як аномальні. Кнопка «Експорт в Excel» (btn_export_anom) дає змогу зберегти результати виявлених аномалій у файл формату Excel для подальшого аналізу або документування.

Таблиця результатів pandastable.Table. Віджет бібліотеки pandastable вбудовано у фрейм вкладки та використано для виводу інтерактивної таблиці з результатами аналізу. Користувач має змогу переглядати дані, прокручувати таблицю та візуально ідентифікувати аномальні значення. Таблиця містить такі колонки: Документ, Дата, ТТ постачальник, Назва, Ціна з податком, z_оцінка, які можемо побачити на рис. 3.12. Завдяки цьому функціоналу користувач може оперативно виявляти транзакції, які виходять за межі статистичної норми, та вживати відповідних управлінських заходів. Автоматизоване виявлення аномалій сприяє покращенню внутрішнього контролю, виявленню потенційних порушень і загалом підвищує прозорість фінансового обліку в межах мережі ресторанів.

	Документ	Дата	ТТ постачальник	Назва	Ціна з подат	Z_оцінка
112	13932-12796	2025-04-30	PEST06	віскі Johnnie Walker Red Label л.	882.43	4.00
113	13226-12103	2025-04-23	PEST06	віскі Johnnie Walker Red Label л.	882.43	4.00
114	10568/9675	2025-04-02	PEST06	віскі Johnnie Walker Red Label л.	882.43	4.00
115	1847	2025-04-24	PEST08	Овочі аспарагус кг	1733.33	3.99
116	24254	2025-04-25	PEST010	Соус кетчуп томатний кг	182.29	3.99
117	21037	2025-04-08	PEST012	Соус кетчуп томатний кг	182.29	3.99
118	1568	2025-04-07	PEST09	В_Пф Шпинат бейби	487.60	3.88
119	5108051	2025-04-02	ЕРІ 2 ЧАБАНИ	Вино кухня/гінтвейн, л	222.76	3.88
120	1812	2025-04-22	PEST08	Сметана	198.00	3.86
121	1841	2025-04-23	PEST08	Сметана	198.00	3.86
122	1568	2025-04-07	PEST09	Сметана	198.00	3.86
123	1530	2025-04-05	PEST019	Сметана	198.00	3.86
124	1529	2025-04-05	PEST013	Хлеб Батон	92.32	3.86
125	11968	2025-04-29	СКЛАД	Соус майонез	84.77	3.86
126	9879	2025-04-08	СКЛАД	Соус майонез	84.77	3.86
127	9862	2025-04-08	PEST018	Соус майонез	84.77	3.86
128	512601	2025-04-10	PEST013	Вино Castelnovo Prosecco Spumante Ex	425.83	3.78
129	512601	2025-04-10	PEST013	нове вино Pinot Grigio delle Venezi л	283.50	3.78
130	18035	2025-04-11	PEST03	Сир твердий	550.00	3.66
131	16039	2025-04-01	PEST03	Сир твердий	550.00	3.66
132	21040	2025-04-25	PEST03	Сир твердий	550.00	3.66

Рисунок 3.12 Вкладка «Аномалії» після аналізу.

3.7 Опис вкладки «Графік цін»

3.7.1 Загальне призначення вкладки

Вкладка "Графік цін" призначена для візуалізації змін цін конкретного товару в часі на основі даних, збережених у реляційній базі даних MySQL. Інструмент дозволяє користувачеві відстежувати динаміку закупівельних цін, ідентифікувати тренди, сезонні коливання або аномальні стрибки, що є важливою складовою фінансового контролю та прийняття управлінських рішень. Блок схему процесу візуалізації динаміки цін вибраного товару можна побачити на рис. 3.13.

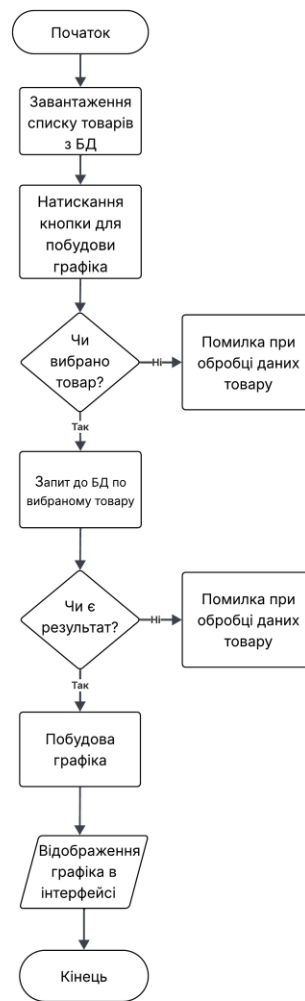


Рисунок 3.13 Схема процесу візуалізації динаміки цін

3.7.2 Структура інтерфейсу та логіка взаємодії

Інтерфейс вкладки організовано за допомогою геометричного менеджера раск із поділом на дві основні області — панель керування зверху та область виведення графіка внизу. Усі елементи розташовані по вертикалі для забезпечення зручного доступу до функціоналу.

Надпис Label. Мітка "Товар" використовується для позначення випадającego списку вибору.

Випадаючий список ttk.Combobox. Єдиний випадający список `combo_product` розміщено на панелі керування. При ініціалізації вкладки викликається функція `load_product_list()`. Вона встановлює з'єднання з MySQL, виконує SQL-запит для отримання унікальних назв товарів, сортує результат і передає його до поля `combo_product['values']`. Користувач має змогу обрати будь-який товар для побудови графіка.

Кнопка Button. Кнопка "Побудувати графік" (`btn_plot`) запускає функцію `show_price_plot()`, якій передається вибране значення з `Combobox`. Функція виконує SQL-запит до таблиці транзакцій, фільтрує записи за назвою товару, витягує дані про дати та ціни, і на їх основі будує графік за допомогою бібліотеки `matplotlib`.

Фрейм для графіка Frame. Контейнер `plot_frame` займає весь залишковий простір під панеллю керування. У ньому відображається лінійний графік із цінами на вибраний товар у хронологічному порядку. Графік автоматично масштабується під розмір вікна та оновлюється після кожного виклику побудови (зображено на рис. 3.14). Такий підхід дозволяє користувачеві ефективно аналізувати часові ряди, оцінювати стабільність поставчань, виявляти аномальні цінові піки або падіння та приймати обґрунтовані рішення щодо закупівель.

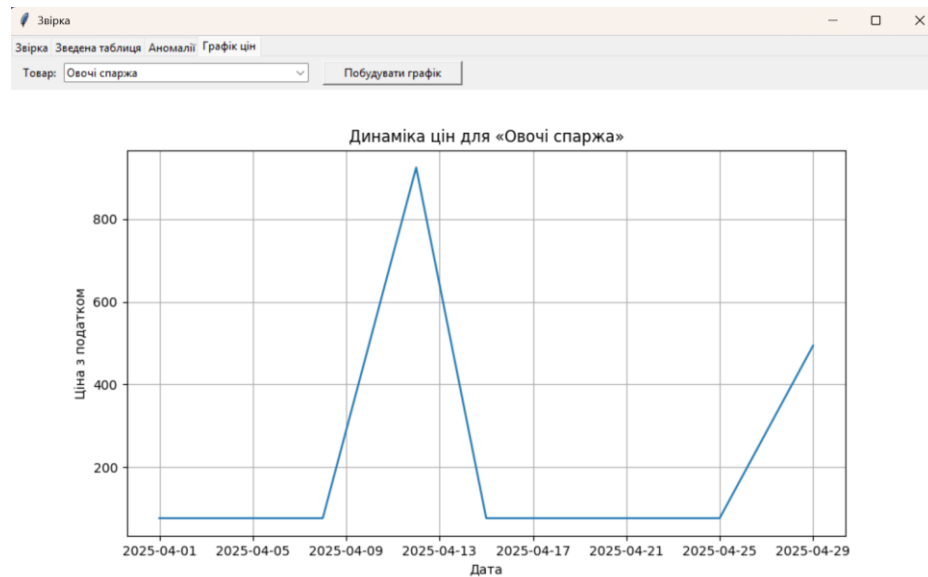


Рисунок 3.14 Вкладка «Графік цін».

Висновки до розділу 3

У цьому розділі було висвітлено процес розробки автономного програмного застосунку на мові Python для автоматизованої звірки, обробки та візуалізації фінансових транзакцій. Описано архітектуру програмного забезпечення, що складається з трьох функціональних модулів `app.py`, `pivottable.py`, `anomalies.py` та реляційної бази даних MySQL, які забезпечують повний цикл обробки – від завантаження Excel файлів до побудови звітів і графіків.

У межах графічного інтерфейсу на основі бібліотеки Tkinter реалізовано чотири логічно відокремлені вкладки. Вкладка "Звірка" надає користувачеві інструмент для порівняння двох незалежних облікових джерел. "Зведена таблиця" дозволяє переглядати агреговану аналітику по товарах із можливістю експорту результатів. "Аномалії" забезпечують статистичне виявлення відхилень за допомогою z-оцінки. "Графік цін" візуалізує динаміку зміни цін у часі для вибраного товару.

Застосування бібліотеки Pandas для обробки таблиць і Pandastable для інтерактивного відображення результатів забезпечило високу гнучкість та інформативність користувацького інтерфейсу. Інтеграція з MySQL дозволила зберігати й опрацьовувати великі обсяги даних із гарантованою узгодженістю.

Загалом розроблений застосунок поєднує автоматизацію, візуалізацію та аналітику у зручному середовищі, що підвищує ефективність обліку та контролю фінансових операцій.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі автоматизована система контролю замовлень у мережі ресторанів розглядається не лише як інженерно-програмний продукт, а й як економічно доцільне рішення, що може підвищити ефективність обслуговування, знизити витрати та покращити управління внутрішніми процесами.

Розроблений програмний продукт може бути впроваджений у закладах ресторанного господарства для централізованого збору, обробки та аналізу інформації про замовлення. Така система є особливо актуальною для середніх та великих мереж, де обсяг операцій щоденно зростає, а помилки ручного обліку можуть спричиняти фінансові втрати.

Для оцінки доцільності впровадження системи використано функціонально-вартісний аналіз (ФВА) – метод, що дозволяє системно оцінити функціональні можливості продукту та пов'язані з ним витрати. ФВА є дієвим інструментом у розробці програмного забезпечення, оскільки дозволяє визначити критично важливі функції, оцінити їх значущість і співвідношення з витратами, а також обґрунтувати прийняття рішень щодо архітектури системи, вибору технологій та способу реалізації.

У межах аналізу буде виконано:

- 1) постановку завдання проектування з економічної точки зору;
- 2) визначення базових функцій системи та їх вартісної оцінки;
- 3) розрахунок витрат на розробку, впровадження та супровід;
- 4) обґрунтування економічного ефекту від впровадження.

Такий підхід дозволяє забезпечити оптимальний баланс між функціональністю розробки, витратами на її реалізацію та очікуваними результатами у вигляді підвищення продуктивності та зменшення операційних витрат.

4.1. Постановка задачі проєктування

Застосування методу функціонально-вартісного аналізу (ФВА) дозволило провести комплексну оцінку техніко-економічних аспектів створення автоматизованої системи контролю замовлень у мережі ресторанів. Ця система орієнтована на централізовану обробку, збереження та аналіз даних щодо постачань, вартості продукції та виявлення аномалій, що дозволяє підвищити точність обліку та зменшити ризик фінансових втрат.

Аналіз охоплює всі підсистеми та функціональні компоненти продукту, які забезпечують автоматизацію ключових бізнес-процесів у сфері ресторанного обслуговування. Особлива увага приділяється доцільності впровадження системи з погляду її впливу на ефективність управління логістикою, постачаннями та витратами.

Визначено наступні функціонально-економічні вимоги до програмного продукту:

Автоматизоване збирання та обробка даних про всі вхідні замовлення в зручному форматі (таблиці, графіки), з можливістю сортування за товарами, постачальниками, цінами та датами.

Наявність візуальних інструментів (наприклад, вкладки "Аномалії", "Зведена таблиця", "Графік цін"), що дозволяють оперативно виявляти нестандартні коливання цін, зміни в асортименті або порушення логістичних ланцюгів.

Інтуїтивно зрозумілий інтерфейс, створений за допомогою модуля Tkinter та віджета Notebook, що забезпечує легкий доступ до всіх функцій без потреби спеціального навчання персоналу.

Можливість інтеграції з базою даних MySQL, яка забезпечує збереження великого обсягу даних, підтримує запити з фільтрами та дозволяє формувати точну аналітику в реальному часі.

Економічна ефективність розробки: застосування вільних інструментів (Python, Pandas, MySQL) дозволяє мінімізувати витрати на розробку, впровадження та обслуговування програмного продукту. Тобто, задачі проектування спрямовані на досягнення максимальної функціональності при збереженні низького рівня витрат, що відповідає вимогам сучасного ресторанного бізнесу до цифрових рішень.

4.2. Обґрунтування функцій програмного продукту

Головна функція F_0 – створення автоматизованої системи контролю замовлень у мережі ресторанів, що дозволяє централізовано збирати, обробляти та аналізувати дані про постачання товарів і формувати зведену інформацію для управлінських рішень. Система має бути простою у використанні, ефективною та наочною.

Ця головна функція деталізується на такі ключові підфункції:

F_1 – Вибір програмного стеку для реалізації системи.

F_2 – Визначення способів обробки, фільтрації та візуалізації даних.

F_3 – Реалізація користувацького інтерфейсу.

F_4 – Організація структури бази даних та підтримка її цілісності.

F_5 – Виявлення аномалій і контроль цін у динаміці.

Кожна з цих функцій може мати кілька альтернатив реалізації:

Функція F_1 (вибір програмного стеку):

а) Python + MySQL

б) JavaScript + MongoDB

в) C# + SQL Server

Функція F_2 (аналіз і обробка даних):

- а) Pandas для агрегацій та фільтрації
- б) SQL-запити з групуванням та сортуванням
- в) Інтеграція з Excel-аналітикою або Google Sheets

Функція F₃ (інтерфейс користувача):

- а) Tkinter з вкладками Notebook
- б) PyQt5 із сучасними UI-компонентами
- в) Вебінтерфейс на Flask з HTML+CSS

Функція F₄ (організація бази даних):

- а) Нормалізована структура у 3NF
- б) Застосування обмежень цілісності та індексації
- в) Резервне копіювання та адміністрування через phpMyAdmin

Функція F₅ (виявлення аномалій):

- а) Статистичний аналіз відхилень цін
- б) Порівняння з середніми значеннями по постачальнику
- в) Візуалізація коливань цін за допомогою графіків

Морфологічна карта системи (рис. 4.1) відображає перелік можливих реалізацій кожної з функцій та дозволяє обрати найдоцільніші варіанти з урахуванням цілей розробки, вартості та ефективності.

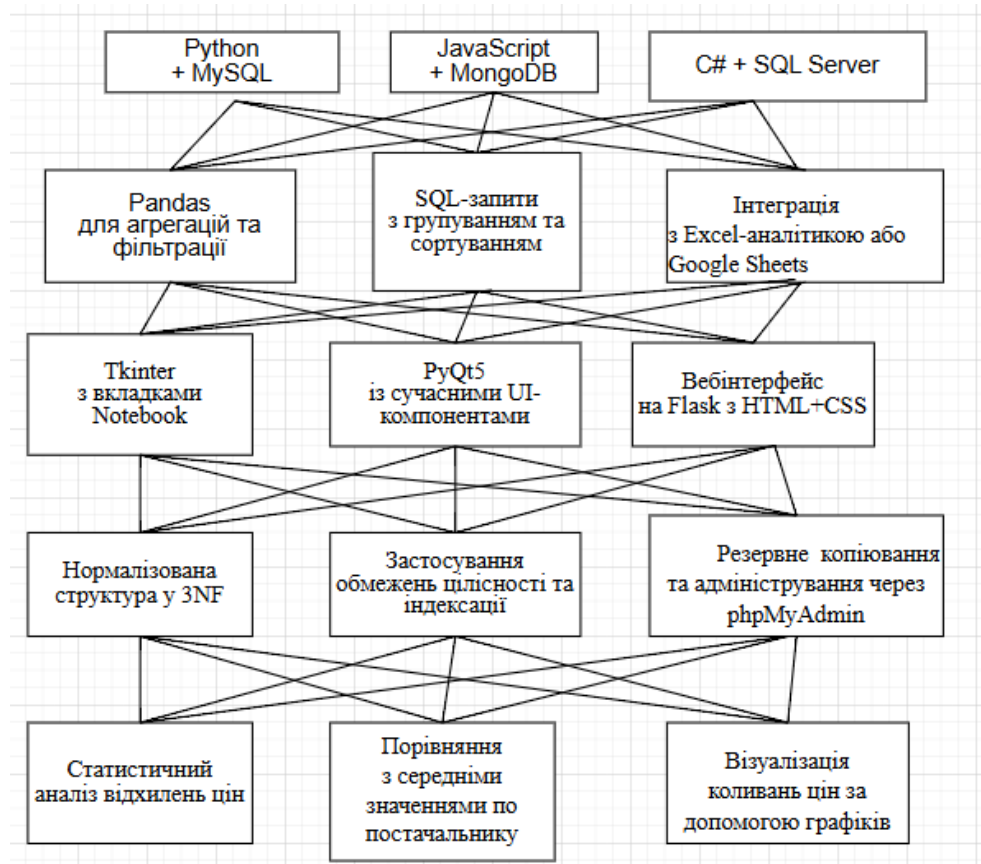


Рисунок 4.1 Морфологічна карта

За допомогою морфологічної карти є можливість побудувати позитивно-негативну матрицю з відображенням усіх варіантів основних функцій (табл. 4.1).

Таблиця 4.1 – Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F ₁ – Вибір програмного стеку	а) Python + MySQL	Безкоштовний, популярний, багато бібліотек	Вимоги до оптимізації при масштабуванні
	б) JavaScript + MongoDB	Гнучкий стек, швидка розробка	Складна логіка звітності
	в) C# + SQL Server	Висока продуктивність, enterprise-рівень	Платна ліцензія, складність налаштування
F ₂ – Обробка даних	а) Pandas + SQL	Гнучкий аналіз, візуалізація	Високе навантаження на пам'ять
	б) SQL-запити	Швидко, безпосередньо в БД	Менше аналітичних можливостей

Продовження таблиці 4.1

	в) Excel/Google Sheets	Доступно користувачам	Обмежена автоматизація
F ₃ – Інтерфейс користувача	а) Tkinter + Notebook	Вбудований, простий, легкий у використанні	Обмеження по сучасному дизайну
	б) PyQt5	Більше елементів управління, гарна візуалізація	Складніша розробка
	в) Flask + HTML	Веб-доступ, масштабування	Потребує більше ресурсів
F ₄ – Організація БД	а) Третя нормальна форма (3NF) + цілісність	Мінімум дублювання, висока структурованість	Більша складність запитів при багатьох зв'язках
	б) Денормалізована модель	Швидкі запити для читання	Можливе дублювання, складність актуалізації
	в) Використання графічного інтерфейсу адміністрування (phpMyAdmin)	Зручність налаштування, візуальний контроль	Може бути обмеженою при складних операціях
F ₅ – Виявлення аномалій і контроль цін	а) Порівняння з середніми значеннями	Простий підхід, швидке впровадження	Нестабільність при малій вибірці
	б) Графіки коливань	Наглядність для користувачів	Потрібна інтерпретація
	в) Поріг для сповіщень	Автоматизація виявлення	Потребує налаштування і тестування

Висновки на основі аналізу:

F₁: Python + MySQL (а) – оптимальне поєднання з позиції вартості, простоти та підтримки.

F₂: Pandas + SQL (а) – ефективно для гнучкої обробки.

F₃: Tkinter + Notebook (а) – мінімальні вимоги, швидка реалізація.

F₄: 3NF + обмеження цілісності (а) – забезпечує якісну структуру з мінімальними помилками.

F₅: Поєднання графіків (б) та порогів (в) – забезпечує і наочність, і автоматизацію.

Рекомендована конфігурація:

F_{1a} – F_{2a} – F_{3a} – F_{4a} – F_{5бв}

Ця комбінація дозволяє:

1. Реалізувати систему на безкоштовному стеку;
2. Отримати повноцінний інтерфейс та функціональність;
3. Знизити витрати при високій якості результатів.

4.3. Обґрунтування системи параметрів програмного продукту

Під час проектування автоматизованої системи контролю замовлень у ресторанній мережі важливо врахувати технічні характеристики, які впливають на ефективність, надійність та економічну доцільність впровадження продукту. На основі функціонально-вартісного аналізу були визначені ключові параметри, що дозволяють об'єктивно оцінити технічний рівень програмного рішення, з урахуванням особливостей середовища, інструментів і технологій, які застосовуються.

Враховуючи доступні варіанти реалізації (Python, MySQL, Tkinter, Pandas, різні методи обробки даних), а також необхідність гнучкої взаємодії з базою даних, виявлення аномалій та побудови інтерфейсів, для порівняльного аналізу обрано такі параметри:

X₁ – обсяг коду програмного продукту (рядків коду);

X₂ – час обробки одного запиту або формування звіту (у секундах);

X₃ – швидкість реакції інтерфейсу при роботі з великою базою даних (у мілісекундах);

X_4 – точність виявлення цінових аномалій (у відсотках).

Ці параметри, які зображено в таблиці 4.2, дозволяють оцінити як ефективність функціональних компонентів (обробка запитів, фільтрація, вивід), так і зручність для користувача (реакція інтерфейсу, швидкість дій).

Таблиця 4.2 – Основні параметри програмного продукту

Назва параметра	Умовне позначення	Одиниця виміру	Гірші значення	Середні значення	Кращі значення
Об'єм коду	X_1	рядків	2500	1700	1200
Час обробки запиту	X_2	секунди (с)	3	1.8	0.5
Швидкість реакції інтерфейсу	X_3	мілісекунди (мс)	1000	500	200
Точність обчислень при складних сценаріях	X_4	відсотків (%)	70	85	95

На рис. 4.2-4.5 можемо побачити гіпотетичний об'єм коду програми, час навчання даних, час, витрачений на виконання поставленого завдання, точність моделі.

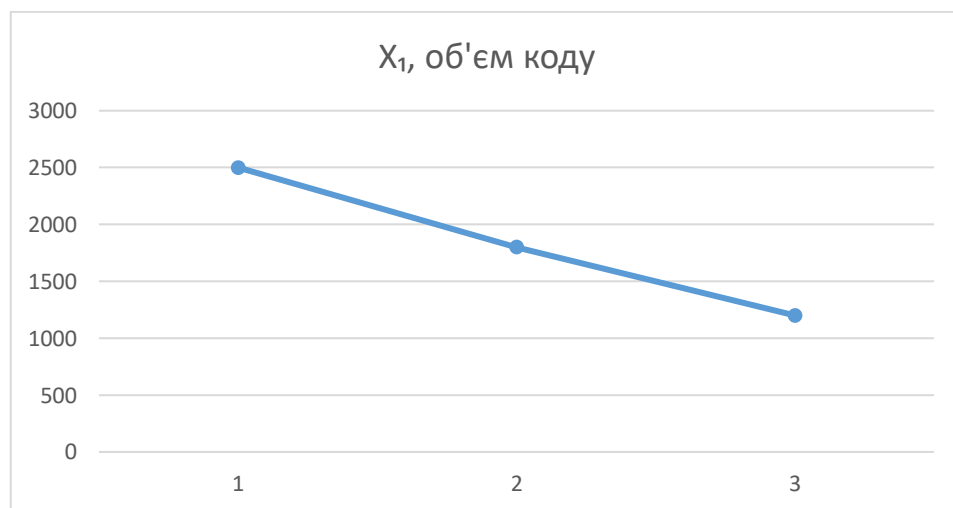


Рисунок 4.2 X_1 , Гіпотетичний об'єм коду програми

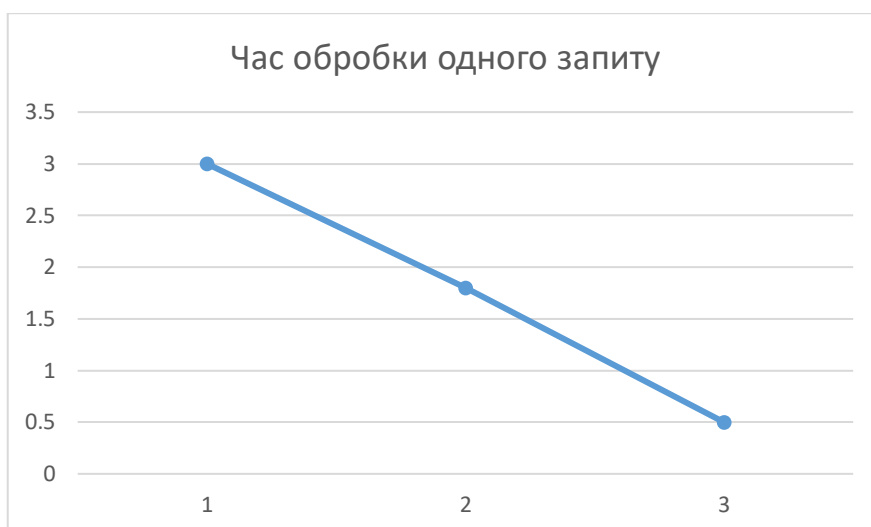


Рисунок 4.3 X2, Час навчання даних

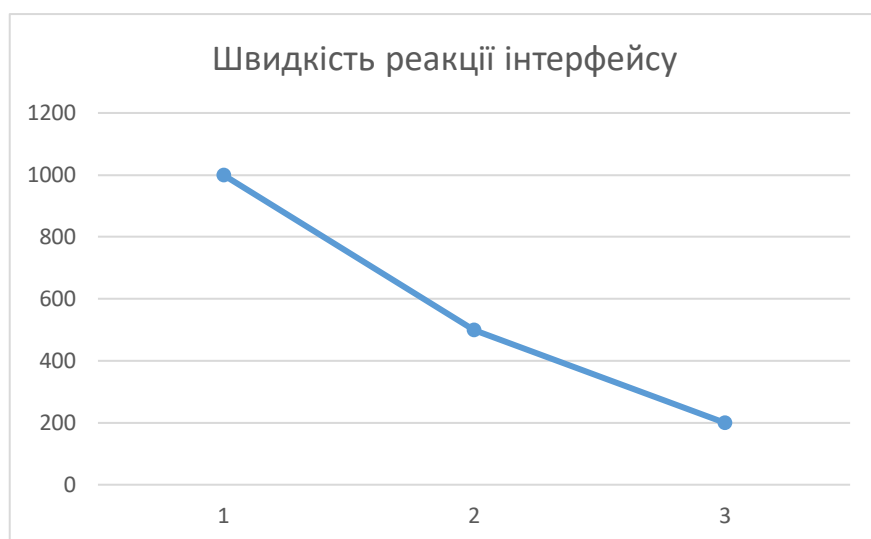


Рисунок 4.4 X3, Час витрачений на виконання поставленого завдання

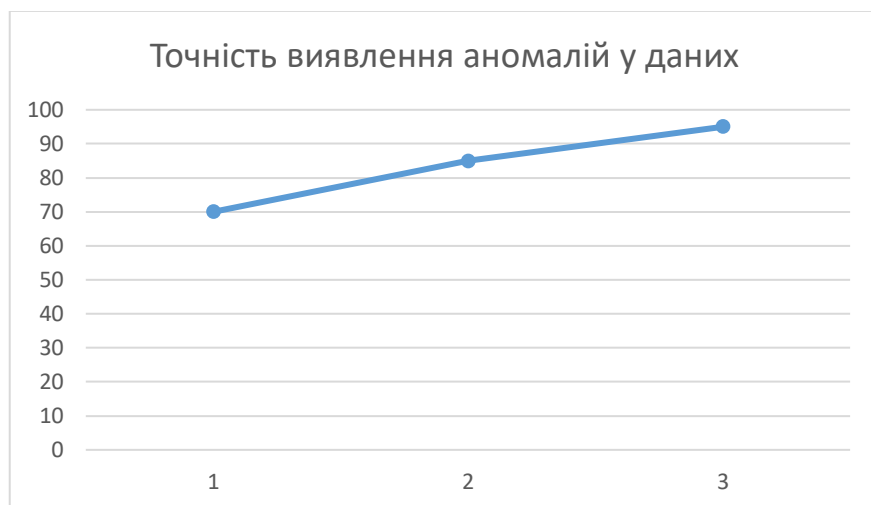


Рисунок 4.5 X4, Точність моделі.

4.4. Аналіз експертного оцінювання параметрів

На основі попередньо отриманих результатів попарного порівняння параметрів, проведеного експертами, було сформовано матрицю переваг параметрів, що дозволяє визначити ступінь впливу кожного параметра на загальну ефективність програмного продукту. Оцінка проводиться за методикою Сааті.

Результати рангової оцінки експертами наведені у таблиці 4.3.

Таблиця 4.3 - Результати ранжування параметрів

Позначення	Назва параметра	Одиниці виміру	Ранги (експерти 1–7)	ΣR_i	Δ_i	Δ_i^2
X ₁	Обсяг коду програми	Кількість рядків	3, 2, 3, 4, 4, 3, 4	23	3.5	12.25
X ₂	Час обробки одного запиту	Секунди	1, 3, 2, 1, 2, 2, 1	12	-7.5	56.25
X ₃	Швидкість реакції інтерфейсу	Мілісекунди	4, 4, 4, 3, 3, 4, 4	26	6.5	42.25
X ₄	Точність виявлення аномалій	Відсотки	2, 1, 1, 2, 1, 1, 2	10	-9.5	90.25

Разом = 201

Щоб оцінити достовірність експертних рейтингів, визначаємо такі параметри:

а) сумарний ранг кожного показника та загальний ранг усіх показників:

$$R_i = \sum_{j=1}^N r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \quad (4.1)$$

де N – кількість експертів,

n – число параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17,5 \quad (4.2)$$

в) різниця між рангами кожного параметра та середньої суми рангів:

$$\Delta_i = R_i - T. \quad (4.3)$$

Сума, що визначає відхилення кожного параметру повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^N \Delta_i^2 = 201. \quad (4.4)$$

Коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 201}{7^2(4^3 - 4)} = 0,820 > W_k = 0,56. \quad (4.5)$$

Отримані результати ранжування можна вважати надійними, адже розрахований коефіцієнт узгодженості перевищує нормативний показник, значення якого дорівнює 0,56. Таким чином утворимо таблицю 4.4, де попарно порівняємо всі параметри.

Таблиця 4.4 - Попарне порівняння параметрів.

Параметри	Експерт 1	2	3	4	5	6	7	Кінцева оцінка	Числове значення
X ₁ і X ₂	>	<	>	<	<	<	<	<	0.5
X ₁ і X ₃	>	>	>	>	<	>	>	>	1.5
X ₁ і X ₄	<	>	<	<	<	<	<	<	0.5
X ₂ і X ₃	>	>	<	>	>	>	>	>	1.5
X ₂ і X ₄	<	>	<	<	<	>	<	<	0.5
X ₃ і X ₄	<	<	<	<	<	<	<	<	0.5

Ступінь переваги i -го параметра над j -тим визначається ячисловим параметром a_{ij} і визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості K_{bi} за наступними формулами:

$$K_{bi} = \frac{b_i}{\sum_{i=1}^n b_i} \quad (4.7)$$

$$b_i = \sum_{j=1}^N a_{ij} \quad (4.8)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

	X ₁	X ₂	X ₃	X ₄	b _i	K _i ¹	b _i '	K _i ²
X ₁	1	0.5	1	0.5	3	0.21	11	0.2113
X ₂	1.5	1	2	1	5	0.36	18	0.3363
X ₃	1	0.5	1	0.5	3	0.21	12	0.2141
X ₄	1.5	1	2	1	5	0.36	23	0.2383
Σ					16	1	64	1

4.5. Аналіз рівня якості варіантів реалізації функцій

Для вибору оптимального варіанту реалізації автоматизованої системи контролю замовлень у мережі ресторанів необхідно порівняти альтернативні рішення за узгодженими техніко-економічними параметрами. Кожен із варіантів реалізації оцінюється за рядом параметрів, визначених у попередніх підрозділах, з урахуванням їхньої вагомості.

Абсолютні значення параметрів:

- X₁ (Обсяг коду),
- X₂ (Час обробки запиту),
- X₃ (Швидкодія інтерфейсу),
- X₄ (Точність виявлення аномалій)

відповідають умовам практичного застосування та технічним вимогам до розроблюваної системи.

Варіанти реалізації функцій F_1 , F_2 , F_3 розглядаються як найбільш імовірні для впровадження.

Розрахунок коефіцієнта технічного рівня (KK_j) виконується за формулою:

$$K_K(j) = \sum_{i=1}^n K_{ei,j} B_{i,j}, \quad (4.11)$$

де n – кількість параметрів;

K_{ei} – коефіцієнт вагомості i -го параметра;

B_i – оцінка i -го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації	Параметр	Абсолютне значення	Бальна оцінка	Коеф. вагомості	Коеф. рівня якості
F_1	A (Python + MySQL)	X_1	1300 рядків	18	0.21	3.78
F_2	A (Pandas + SQL)	X_4	93 %	32	0.24	7.68
	Б (SQL-запити)	X_4	83 %	26	0.24	6.24
F_3	A (Tkinter + Notebook)	X_3	400 мс	22	0.21	4.62
Σ					16	1

За даними з таблиці 4.6 за формулою:

$$K_K = K_{Ty}[F_{1k}] + K_{Ty}[F_{2k}] + \dots + K_{Ty}[F_{zk}], \quad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 3.78 + 7.68 + 4.62 = 16.08;$$

$$K_{K2} = 3.78 + 6.24 + 4.62 = 14.64.$$

Як видно з розрахунків, кращим є 1 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6. Економічний аналіз варіантів розробки ПП

Для оцінки доцільності вибору конкретного варіанту реалізації програмного продукту доцільно виконати економічний аналіз, що включає розрахунок трудомісткості та вартості робіт. Кожен варіант передбачає виконання двох ключових завдань:

- Розробка проєктної структури системи обліку та контролю замовлень;
- Реалізація графічного інтерфейсу з інтеграцією бази даних.

За класифікацією:

Завдання 1 має степінь новизни Б та групу складності алгоритму 2;

Завдання 2 – степінь новизни В, група складності 3.

Вхідна інформація для обох завдань представлена у вигляді структурованих таблиць, що впливає на коефіцієнти складності та стандартності.

Загальна трудомісткість обчислюється як:

$$T_0 = T_P \cdot K_{\Pi} \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.13)$$

де T_p – трудомісткість розробки ПП;

K_{Π} – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

K_M – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Завдання 1 (архітектура + логіка):

$$T_1 = 64 \cdot 1.05 \cdot 1 \cdot 0.65 = 43.68 \text{ людино-днів.}$$

Завдання 2 (інтерфейс + MySQL):

$$T_2 = 28 \cdot 1.03 \cdot 1 \cdot 0.6 = 17,30 \text{ людино-днів.}$$

Загальна трудомісткість:

$$T_0 = (43.68 + 17,30) \cdot 8 = 487.8 \text{ людино-годин.}$$

Розрахунок оплати праці

До команди залучені:

– Програміст з окладом 68 000 грн

– Аналітик із досвідом у ресторанному бізнесі — 64 000 грн

Середня погодинна ставка:

$$C_q = \frac{M}{T_m \cdot t} \text{ грн.,} \quad (4.14)$$

де M – місячний оклад працівників;

T_m – кількість робочих днів на місяць;

t – кількість робочих годин в день.

$$C_q = \frac{68000 + 64000}{2 \cdot 20 \cdot 8} = 412.5 \text{ грн.} \quad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$C_{зп} = C_q \cdot T_i \cdot K_d, \quad (4.16)$$

де C_q – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

K_d – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$C_{зп} = 487.8 \cdot 412.5 = 201727.5 \text{ грн.}$$

Відрахування на єдиний соціальний внесок становить 22%:

$$C_{вд} = C_{зп} \cdot 0.22 = 201727.5 \cdot 0.22 = 44380 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години. (C_m)

Так як одна ЕОМ обслуговує одного програміста з окладом 36000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_{\Gamma} = 12 \cdot M \cdot K_3 = 12 \cdot 36000 \cdot 0.2 = 86400 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{ЗП} = C_{Г} \cdot (1 + K_3) = 86400 \cdot (1 + 0.2) = 103680 \text{ грн.}$$

Відрахування на соціальний внесок:

$$C_{ВД} = C_{ЗП} \cdot 0.22 = 103680 \cdot 0.22 = 22809,6 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 49000 грн.

$$C_A = K_{ТМ} \cdot K_A \cdot Ц_{ПР} = 1.23 \cdot 0.25 \cdot 49000 = 15067,5 \text{ грн.,}$$

де $K_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

K_A – річна норма амортизації;

$Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{ТМ} \cdot Ц_{ПР} \cdot K_P = 1.23 \cdot 49000 \cdot 0.03 = 1808,1 \text{ грн.,}$$

де K_P – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$\begin{aligned} T_{ЕФ} &= (D_K - D_B - D_C - D_P) \cdot t_3 \cdot K_B = (365 - 105 - 21 - 16) \cdot 8 \cdot 0.87 = \\ &= 1552,08 \text{ години,} \end{aligned}$$

де D_K – календарна кількість днів у році;

D_B, D_C – відповідно кількість вихідних та святкових днів;

D_p – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день;

K_B – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{\text{ЕЛ}} = T_{\text{ЕФ}} \cdot N_{\text{С}} \cdot K_3 \cdot \text{Ц}_{\text{ЕН}} = 1552,08 \cdot 0,3 \cdot 0,5 \cdot 9,43 = 2195,45 \text{ грн.},$$

де $N_{\text{С}}$ – середньо-споживча потужність приладу;

K_3 – коефіцієнтом зайнятості приладу;

$\text{Ц}_{\text{ЕН}}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_{\text{Н}} = \text{Ц}_{\text{ПР}} \cdot 0,67 = 49000 \cdot 0,67 = 32830 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{\text{ЕКС}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{А}} + C_{\text{Р}} + C_{\text{ЕЛ}} + C_{\text{Н}}, \quad (4.17)$$

$$C_{\text{ЕКС}} = 103680 + 22809,6 + 24676,7 + 1808,1 + 2195,45 + 32830 = 187999.85 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{\text{М-Г}} = C_{\text{ЕКС}} / T_{\text{ЕФ}} = 187999.85 / 1552,08 = 121.17 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов'язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_M = C_{M-\Gamma} \cdot T, \quad (4.18)$$

$$C_M = 121.17 \cdot 487,8 \approx 59136.63 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$C_H = C_{ЗП} \cdot 0,67, \quad (4.19)$$

$$\text{І. } C_H = 201727,5 \cdot 0,67 = 135157,43 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_M + C_H, \quad (4.20)$$

$$C_{ПП} = 201727,50 + 44380,05 + 59136.63 + 135157,43 = 439731,69 \text{ грн.}$$

4.7. Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{ТЕР}j} = K_{Kj} / C_{\Phi j}, \quad (4.21)$$

$$K_{\text{TEP1}} = 16,08 / 439731,69 = 3,66 \cdot 10^{-5},$$

На основі проведених розрахунків встановлено, що загальна трудомісткість реалізації основних функцій системи становить 487.8 людино-годин. Це включає:

- 43.68 людино-днів на розробку архітектури та логіки роботи системи;
- 17.30 людино-днів на створення графічного інтерфейсу та підключення бази даних.

Розрахована повна собівартість розробки програмного продукту становить 439731,69 грн, включаючи:

- оплату праці;
- соціальні внески;
- вартість машинного часу;
- накладні витрати.

Оцінений коефіцієнт техніко-економічного рівня ($K_{\text{TEP}} = 3.66 \times 10^{-5}$) свідчить про достатньо високу вартість розробки відносно технічної ефективності продукту. Це обґрунтовує доцільність подальшого оптимізаційного аналізу, зокрема:

- пошуку можливостей повторного використання модулів;
- впровадження безкоштовних або спрощених інструментів;
- поетапної реалізації окремих функцій.

Висновки до розділу 4

У четвертому розділі дипломної роботи було здійснено техніко-економічну оцінку варіантів реалізації автоматизованої системи контролю замовлень у мережі ресторанів. Проведений функціонально-вартісний аналіз дозволив

виявити найбільш ефективні шляхи реалізації ключових функцій програмного продукту з урахуванням вимог до точності, швидкодії та зручності користування.

На основі методу попарного порівняння та ранжування параметрів було встановлено вагомість кожного критерію (об'єм коду, час навчання, точність моделі, час виконання завдання) для обґрунтування вибору оптимального варіанту реалізації. В результаті сформовано матрицю пріоритетів, обчислено коефіцієнти вагомості та технічного рівня, що дозволило кількісно порівняти можливі альтернативи.

Проведено деталізований економічний аналіз розробки програмного продукту, включаючи оцінку трудомісткості, заробітної плати розробників, накладних витрат, соціальних відрахувань, вартості машинного часу та енергоспоживання. Розрахована повна собівартість реалізації базового варіанту дозволила встановити орієнтовну вартість розробки в межах 440 тис. грн.

На завершення проведено оцінку техніко-економічного рівня системи, що дозволило інтегрувати технічні показники з економічними витратами. Отримане значення коефіцієнта КТЕР свідчить про потребу у подальшій оптимізації витрат, проте загалом обраний варіант має високий потенціал до впровадження в ресторанній сфері завдяки поєднанню технічної ефективності, масштабованості та адаптивності до різних форматів бізнесу.

Таким чином, запропонована система може стати ефективним інструментом для покращення обліку та контролю замовлень у мережі ресторанів, знижуючи людський фактор та підвищуючи якість обслуговування клієнтів. У подальшій роботі доцільно зосередитись на мінімізації витрат через використання готових модулів, хмарних сервісів або open-source рішень.

ВИСНОВКИ

У дипломній роботі було розроблено автоматизовану систему контролю замовлень, що орієнтована на використання в мережі ресторанів. Основною метою проекту стало підвищення ефективності внутрішнього документообігу, швидкості обробки замовлень, зменшення впливу людського фактора та забезпечення прозорого контролю над постачанням продукції й обліком витрат. Реалізована система базується на використанні реляційної бази даних MySQL, що дозволяє організувати надійне та масштабоване середовище для зберігання та обробки структурованих даних.

У процесі дослідження було проведено аналіз існуючих проблем у ресторанній галузі, зокрема в частині обробки замовлень і взаємодії з постачальниками. Здійснено огляд програмних рішень, які вже використовуються на ринку автоматизації ресторанного бізнесу. За результатами аналізу встановлено, що більшість таких систем мають вузький функціонал, обмежені можливості налаштування або значну вартість впровадження. Це обумовило необхідність створення власної, адаптивної та відкритої до масштабування інформаційної системи, яка могла б повністю відповідати вимогам конкретного бізнесу.

Розроблена система включає кілька логічних компонентів: модуль управління базою даних, модуль взаємодії з користувачем (інтерфейс), а також аналітичний модуль, який відповідає за формування звітів та візуалізацію ключових показників. Застосування мови програмування Python у поєднанні з бібліотекою Tkinter дозволило реалізувати простий, інтуїтивно зрозумілий графічний інтерфейс користувача, що надає змогу співробітникам ресторану без спеціальної технічної підготовки працювати з програмою.

У системі реалізовано функціонал з додавання, редагування, видалення та пошуку замовлень. Також реалізовано можливість фільтрації записів за заданими параметрами, збереження історії змін, контроль відповідності цінової політики постачальників, генерація щотижневих звітів та виведення даних у вигляді графіків і таблиць. Структура бази даних розроблена таким чином, щоб у подальшому її можна було легко модифікувати або доповнювати, наприклад, для інтеграції з POS-системами.

Проведене тестування на основі реальних сценаріїв показало, що система працює стабільно, коректно обробляє дані, забезпечує швидкий доступ до необхідної інформації та відповідає вимогам до продуктивності в умовах середнього навантаження. Програма здатна працювати як на окремому робочому місці, так і в локальній мережі, що дозволяє використовувати її в ресторанах різних масштабів.

У результаті виконання дипломної роботи було досягнуто поставленої мети – створено працездатний, функціональний і адаптивний програмний продукт, що дозволяє автоматизувати контроль замовлень у мережі ресторанів. Впровадження такої системи дає змогу не лише зменшити витрати часу та ресурсів на обробку замовлень, а й забезпечити кращий контроль за дотриманням внутрішніх стандартів, підвищити точність обліку та сприяти прийняттю обґрунтованих управлінських рішень. У перспективі можливе подальше розширення функціоналу, зокрема інтеграція з CRM-системами, веб-сервісами для онлайн-замовлень та системами бізнес-аналітики.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hoffman T. How to Start, Run & Grow a Successful Restaurant Business: A Lean Startup Guide, 2011.
2. Завадинська О., Ніколайко Г., Огороднік М. Дослідження інноваційних рішень для оновлення існуючих бізнес-моделей і сучасних сервісних технологій у ресторанному бізнесі. Ресторанний і готельний консалтинг. Інновації, 2022. Т. 5, № 2. С. 229–238.
3. Friedman A. Running a Restaurant For Dummies, Wiley Publishing, 2011.
4. Кіш Г. В. Інноваційні технології в діяльності готельно-ресторанних підприємств. Актуальні питання у сучасній науці, 2023. Вип. №6(12). С. 65–77.
5. Camillo A. A. Strategic International Restaurant Development: From Concept to Production, 2021.
6. Процак К., Передрій М. Ресторанний бізнес в умовах кризи: проблеми та напрямки розвитку. Економіка та суспільство, 2022. № 44. 6 с.
7. Fernandez T. Pandemic Reflections: What Lessons Has the Restaurant Industry Learned?, 2023.
8. Постова В.В. Перспективи впровадження інтелектуальної автоматизації в ресторанний бізнес в умовах післявоєнного відновлення України. Економіка та суспільство, 2023. № 49. 6 с. URL: <https://economyandsociety.in.ua/index.php/journal/article/view/2331/2253> (дата звернення: 15.05.2025).
9. Коркач Г., Котузаки О., Никітчина Т., Шунько Г. Технологічні інновації в ресторанному бізнесі для забезпечення конкурентоспроможності закладу. Економіка харчової промисловості, 2021. Т. 13, № 4. С. 69–74.
10. Рябенка М. О. Впровадження інформаційних комп'ютерних технологій у ресторанному бізнесі. Економіка та суспільство, 2022. Вип. 41. 6 с.

URL: <https://sel.vtei.edu.ua/repository/g.php?fname=28739.pdf> (дата звернення: 22.04.2025).

11. Grinberg M. Flask Web Development: Developing Web Applications with Python, O'Reilly Media, 2018.

12. MySQL Documentation. URL: <https://dev.mysql.com/doc/> (дата звернення: 25.04.2025).

13. Python Documentation. URL: <https://docs.python.org/3/> (дата звернення: 20.05.2025).

14. Python Tkinter Library. URL: <https://docs.python.org/3/library/tkinter.html> (дата звернення: 30.04.2025).

15. Creating Tabbed Widget With Python-Tkinter. URL: <https://www.geeksforgeeks.org/creating-tabbed-widget-with-python-tkinter/> (дата звернення: 19.04.2025).

16. Python Tkinter TTK Library. URL: <https://docs.python.org/3/library/tkinter.ttk.html> (дата звернення: 17.05.2025).

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

Файл `app.py`

```

import matplotlib
matplotlib.use('TkAgg')
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import mysql.connector
import pandas as pd
import tkinter as tk
from tkinter import ttk, filedialog, messagebox
from pandastable import Table
from pivottable import adjust_column_widths, generate_pivot,
export_to_excel
from anomalies import generate_anomalies, export_anomalies
from pivottable import config

table_pt = None
pivot_pt = None

def load_file(entry):
    filename = filedialog.askopenfilename(filetypes=[("Excel
Files", "*.xls;*.xlsx")])
    entry.delete(0, tk.END)
    entry.insert(0, filename)

def process_files():
    global df2
    file1 = entry_file1.get()
    file2 = entry_file2.get()
    if not file1 or not file2:
        messagebox.showerror("Помилка", "Виберіть обидва файли")

```

```

return

try:
    df1 = pd.read_excel(file1)
    df2 = pd.read_excel(file2)

    df1['Дата'] = pd.to_datetime(df1['Дата']).dt.date
    df2['Дата документа'] = pd.to_datetime(df2['Дата
документа']).dt.date

    df1['Документ'] = df1['Документ'].astype(str)

    sum_df1 = df1.groupby('Документ', as_index=False)['Сума с
податком'].sum()
    tt_mapping = df1.set_index('Документ')['ТТ
отримувач'].to_dict()
    print(tt_mapping)
    df2['Номер документа'] = df2['Номер
документа'].astype("Int64").astype(str)

    def find_tt(doc_number, mapping):
        for doc, tt in mapping.items():
            if doc_number in str(doc):
                return tt
        return "Помилка"

    df2['ТТ одержувач'] = df2['Номер документа'].apply(lambda
x: find_tt(x, tt_mapping))

    sum_mapping = sum_df1.set_index('Документ')['Сума с
податком'].to_dict()
    print(sum_mapping)
    def check_sum(row, sum_mapping):

```

```

        doc_number = row['Номер документа']
        sum_df2 = row['Сума']
        matched_docs = [doc for doc in sum_mapping if
doc_number in str(doc)]
        if not matched_docs:
            return 'Помилка'
        else:
            sum_df1_value = sum_mapping[matched_docs[0]]
            if abs(sum_df1_value - sum_df2) /
max(sum_df1_value, sum_df2) > 0.01:
                return f'Сума не збігається ({sum_df1_value})'
            return ""

        df2['Перевірка суми'] = df2.apply(lambda row:
check_sum(row, sum_mapping), axis=1)

        display_table(df2)
    except Exception as e:
        messagebox.showerror("Помилка", str(e))

def display_table(df):
    global table_pt
    if table_pt is not None:
        table_pt.model.df = df
        table_pt.redraw()
    else:
        for widget in frame_table.wininfo_children():
            widget.destroy()
        table_pt = Table(frame_table, dataframe=df,
showtoolbar=True, showstatusbar=True)
        table_pt.show()

```

```

def save_file():
    global df2
    if df2 is None or df2.empty:
        messagebox.showerror("Помилка", "Немає даних для
збереження. Обробіть файли спочатку.")
        return
    filename =
filedialog.asksaveasfilename(defaultextension=".xlsx",
filetypes=[("Excel Files", "*.xlsx")])
    if filename:
        try:
            df2.to_excel(filename, index=False)
            adjust_column_widths(filename)
            messagebox.showinfo("Готово", f"Файл збережено:
{filename}")
        except Exception as e:
            messagebox.showerror("Помилка при збереженні", str(e))

root = tk.Tk()
root.title("Звірка")
root.geometry("1000x600")

notebook = ttk.Notebook(root)
notebook.pack(fill=tk.BOTH, expand=True)

# Перша вкладка
tab1 = ttk.Frame(notebook)
notebook.add(tab1, text="Звірка")

# Друга вкладка
tab2 = ttk.Frame(notebook)
notebook.add(tab2, text="Зведена таблиця")

```

```

# Елементи першої вкладки
header_frame = tk.Frame(tab1)
header_frame.pack(side=tk.TOP, fill=tk.X, pady=10)

left_buttons = tk.Frame(header_frame)
left_buttons.pack(side=tk.LEFT, padx=20)

file1_frame = tk.Frame(left_buttons)
file1_frame.pack(pady=5)
tk.Label(file1_frame, text="Основний файл:").pack(side=tk.LEFT)
entry_file1 = tk.Entry(file1_frame, width=50)
entry_file1.pack(side=tk.LEFT)
btn_file1 = tk.Button(file1_frame, text="Вибрати", command=lambda:
load_file(entry_file1))
btn_file1.pack(side=tk.LEFT)

file2_frame = tk.Frame(left_buttons)
file2_frame.pack(pady=5)
tk.Label(file2_frame, text="Файл звірки:",
padx=12).pack(side=tk.LEFT)
entry_file2 = tk.Entry(file2_frame, width=50)
entry_file2.pack(side=tk.LEFT)
btn_file2 = tk.Button(file2_frame, text="Вибрати", command=lambda:
load_file(entry_file2))
btn_file2.pack(side=tk.LEFT)

right_inputs = tk.Frame(header_frame)
right_inputs.pack(side=tk.RIGHT, padx=20)

btn_process = tk.Button(right_inputs, text="Обробити файли",
command=process_files, width=20)
btn_process.pack(pady=5)

```

```

btn_save = tk.Button(right_inputs, text="Зберегти файл",
command=save_file, width=20)
btn_save.pack(pady=5)

frame_table = tk.Frame(tab1)
frame_table.pack(fill=tk.BOTH, expand=True)

# Елементи другої вкладки
def show_pivot_table():
    global pivot_pt
    try:
        df = generate_pivot()
        if pivot_pt is not None:
            pivot_pt.model.df = df
            pivot_pt.redraw()
        else:
            for widget in pivot_frame.winfo_children():
                widget.destroy()
            pivot_pt = Table(pivot_frame, dataframe=df,
showtoolbar=True, showstatusbar=True)
            pivot_pt.show()
    except Exception as e:
        messagebox.showerror("Помилка", f"Неможливо завантажити
дані: {str(e)}")

pivot_controls = tk.Frame(tab2)
pivot_controls.pack(fill=tk.X, pady=5)
btn_refresh = tk.Button(pivot_controls, text="Оновити дані",
width=20, command=show_pivot_table)
btn_refresh.pack(side=tk.LEFT, padx=10)
btn_export = tk.Button(pivot_controls, text="Експорт в Excel",
width=20, command=lambda: export_to_excel())
btn_export.pack(side=tk.LEFT)

```

```

pivot_frame = tk.Frame(tab2)
pivot_frame.pack(fill=tk.BOTH, expand=True)

show_pivot_table()

# Третя вкладка
tab3 = ttk.Frame(notebook)
notebook.add(tab3, text="Аномалії")

anomaly_controls = tk.Frame(tab3)
anomaly_controls.pack(fill=tk.X, pady=5)

def show_anomaly_table():
    global anomaly_pt
    try:
        df = generate_anomalies()
        if df.empty:
            messagebox.showinfo("Інформація", "Аномалій не знайдено.")
            return
        if anomaly_pt is not None:
            anomaly_pt.model.df = df
            anomaly_pt.redraw()
        else:
            for widget in anomaly_frame.winfo_children():
                widget.destroy()
            anomaly_pt = Table(anomaly_frame, dataframe=df,
showtoolbar=True, showstatusbar=True)
            anomaly_pt.show()
    except Exception as e:
        messagebox.showerror("Помилка", f"Помилка при завантаженні аномалій:\n{str(e)}")

```

```

btn_refresh_anom = tk.Button(
    anomaly_controls,
    text="Оновити дані",
    width=20,
    command=lambda: show_anomaly_table()
)
btn_refresh_anom.pack(side=tk.LEFT, padx=10)

btn_export_anom = tk.Button(
    anomaly_controls,
    text="Експорт в Excel",
    width=20,
    command=lambda: export_anomalies()
)
btn_export_anom.pack(side=tk.LEFT)

anomaly_frame = tk.Frame(tab3)
anomaly_frame.pack(fill=tk.BOTH, expand=True)

anomaly_pt = None

show_anomaly_table()

# Четверта вкладка
tab4 = ttk.Frame(notebook)
notebook.add(tab4, text="Графік цін")

plot_controls = tk.Frame(tab4)
plot_controls.pack(fill=tk.X, pady=5)

tk.Label(plot_controls, text="Товар:").pack(side=tk.LEFT,
padx=(10,0))

```

```

combo_product = ttk.Combobox(plot_controls, state="readonly",
width=40)
combo_product.pack(side=tk.LEFT, padx=5)

btn_plot = tk.Button(
    plot_controls,
    text="Побудувати графік",
    width=20,
    command=lambda: show_price_plot(combo_product.get())
)
btn_plot.pack(side=tk.LEFT, padx=10)

plot_frame = tk.Frame(tab4)
plot_frame.pack(fill=tk.BOTH, expand=True)

def load_product_list():
    conn = None
    try:
        conn = mysql.connector.connect(**config)
        df = pd.read_sql("SELECT DISTINCT product_name FROM
products", conn)
        products = sorted(df['product_name'].dropna().tolist())
        combo_product['values'] = products
        if products:
            combo_product.current(0)
    except Exception as e:
        messagebox.showerror("Помилка DB", str(e))
    finally:
        if conn and conn.is_connected():
            conn.close()

def show_price_plot(product_name):

```

```

if not product_name:
    messagebox.showwarning("Увага", "Оберіть товар зі списку.")
    return

for w in plot_frame.wininfo_children():
    w.destroy()

conn = None
try:
    conn = mysql.connector.connect(**config)
    query = """
        SELECT
            d.document_date AS Дата,
            tf.price_with_tax AS Ціна_з_податком
        FROM transaction_facts tf
        JOIN products p ON tf.product_code = p.product_code
        JOIN documents d ON tf.document_no = d.document_no
        WHERE p.product_name = %s
        ORDER BY d.document_date
    """

    df = pd.read_sql(query, conn, params=(product_name,))
except Exception as e:
    messagebox.showerror("Помилка DB", str(e))
    return

finally:
    if conn and conn.is_connected():
        conn.close()

if df.empty:
    messagebox.showinfo("Інформація", f"Даних для товару  
«{product_name}» не знайдено.")
    return

```

```

df['Дата'] = pd.to_datetime(df['Дата'])
df = df.sort_values('Дата')

fig = Figure(figsize=(6, 4))
ax = fig.add_subplot(111)
ax.plot(df['Дата'], df['Ціна_з_податком'])
ax.set_xlabel('Дата')
ax.set_ylabel('Ціна з податком')
ax.set_title(f'Динаміка цін для «{product_name}»')
ax.grid(True)

canvas = FigureCanvasTkAgg(fig, master=plot_frame)
canvas.draw()
canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

load_product_list()

root.mainloop()

Файл pivottable.py

import mysql.connector
import pandas as pd
from openpyxl import load_workbook
from openpyxl.utils import get_column_letter

config = {
    'user': 'root',
    'password': '12345678',
    'host': 'localhost',
    'database': 'reconciliation',
    'raise_on_warnings': True
}

```

```

def top_prices(series):
    price_counts = series.round(2).value_counts().head(3)
    return price_counts.index.tolist()

def generate_pivot():
    cnx = None
    try:
        cnx = mysql.connector.connect(**config)

        query = """
            SELECT
                p.product_name AS Назва,
                tf.quantity,
                tf.amount_with_tax,
                tf.price_with_tax
            FROM transaction_facts tf
            JOIN products p ON tf.product_code = p.product_code
        """

        df = pd.read_sql(query, con=cnx)

        grouped = df.groupby('Назва').agg(
            Кількість=('quantity', 'sum'),
            Сума_з_податком=('amount_with_tax', 'sum'),
            Топ_3_цени=('price_with_tax', top_prices)
        ).reset_index()

        grouped['Середня_ціна'] = grouped['Сума_з_податком'] /
        grouped['Кількість']

        grouped['Топ_3_цени'] = grouped['Топ_3_цени'].apply(
            lambda x: ', '.join(f'{price:.2f}' for price in x) if x

```

```

else 'немає даних'
    )

    grouped['Середня_ціна'] = grouped['Середня_ціна'].round(2)
    grouped['Сума_з_податком'] =
grouped['Сума_з_податком'].round(2)
    grouped = grouped.sort_values(by='Сума_з_податком',
ascending=False)

    grouped = grouped.rename(columns={
        'Назва': 'Назва',
        'Кількість': 'Кількість',
        'Сума_з_податком': 'Сума з податком',
        'Середня_ціна': 'Середня ціна',
        'Топ_3_цени': 'Топ-3 ціни'
    })

    return grouped[['Назва', 'Кількість', 'Сума з податком',
'Sередня ціна', 'Топ-3 ціни']]

except mysql.connector.Error as err:
    print(f"Помилка підключення: {err}")
    return pd.DataFrame()

finally:
    if cnx and cnx.is_connected():
        cnx.close()

def adjust_column_widths(filename):
    wb = load_workbook(filename)
    for sheet in wb.sheetnames:
        ws = wb[sheet]
        for col in ws.columns:

```

```

        max_length = 0
        col_letter = get_column_letter(col[0].column)
        for cell in col:
            try:
                if cell.value:
                    max_length = max(max_length,
len(str(cell.value)))
            except:
                pass
        ws.column_dimensions[col_letter].width = max_length + 2
wb.save(filename)

```

```

def export_to_excel():
    df = generate_pivot()
    if not df.empty:
        df.to_excel('зведена таблиця.xlsx',
                    sheet_name='Зведена таблиця',
                    index=False,
                    header=['Назва', 'Кількість', 'Сума з
податком', 'Середня ціна', 'Топ-3 ціни'])
        adjust_column_widths('зведена таблиця.xlsx')
        print("Експорт завершено успішно!")
    else:
        print("Немає даних для експорту")

```

Файл anomalies.py

```

import mysql.connector
import pandas as pd
from pivottable import adjust_column_widths, config

```

```

def generate_anomalies():
    conn = None

```

```

try:
    conn = mysql.connector.connect(**config)
    query = """
        SELECT
            d.document_no AS Документ,
            d.document_date AS Дата,
            p.product_name AS Назва,
            tf.price_with_tax AS `Ціна з податком`,
            tf.tt_supplier AS `ТТ постачальник`
        FROM transaction_facts tf
        JOIN products p ON tf.product_code = p.product_code
        JOIN documents d ON tf.document_no = d.document_no
    """
    df = pd.read_sql(query, conn)

    df['Дата'] = pd.to_datetime(df['Дата'])

    counts = df.groupby('Назва')['Ціна з
податком'].transform('count')
    df = df[counts > 1].copy()

    df['mean_price'] = df.groupby('Назва')['Ціна з
податком'].transform('mean')
    df['std_price'] = df.groupby('Назва')['Ціна з
податком'].transform('std')

    df = df[df['std_price'] > 0].copy()
    df['z_оцінка'] = (df['Ціна з податком'] - df['mean_price'])
/ df['std_price']

    anomalies = df[df['z_оцінка'].abs() > 3].copy()
    anomalies['Дата'] = anomalies['Дата'].dt.date

```

```

        return anomalies[[
            'Документ',
            'Дата',
            'ТТ постачальник',
            'Назва',
            'Ціна з податком',
            'z_оцінка'
        ]].sort_values(by='z_оцінка', ascending=False)

    except mysql.connector.Error as e:
        print(f"DB Error: {e}")
        return pd.DataFrame()
    finally:
        if conn and conn.is_connected():
            conn.close()

def export_anomalies():
    df = generate_anomalies()
    if not df.empty:
        df.to_excel('аномалії.xlsx',
                    sheet_name='Аномалії',
                    index=False,
                    header=['Документ', 'Дата',
                           'ТТ постачальник',
                           'Назва', 'Ціна з податком',
                           'z_оцінка'])
        adjust_column_widths('аномалії.xlsx')
        print("Експорт завершено успішно!")
    else:
        print("Немає даних для експорту")

print(generate_anomalies())
export_anomalies()

```

ДОДАТОК Б. ГРАФІЧНІ МАТЕРІАЛИ



Автоматизована система звірки поставок в мережі ресторанів на основі бази даних

Виконала:
студентка групи КА-15
Бондарева Діана Володимирівна

Науковий керівник:
Професор, д.т.н., Мухін Вадим Євгенович

Мета та об'єкт дослідження. Основні завдання.

Мета та об'єкт дослідження:

Метою даної роботи є розробка програмного продукту, що забезпечує контроль за замовленнями в мережі ресторанів.

Об'єктом є сам процес обробки замовлень, а предметом — інформаційна система на базі реляційної бази даних.

Для досягнення мети були поставлені такі завдання:

- Аналіз предметної області;
- Вивчення існуючих програмних рішень;
- Проектування структури бази даних;
- Розробка інтерфейсу користувача;
- Тестування функціоналу на прикладах.

Чому автоматизація звірки поставок необхідна для мереж ресторанів?

Зниження людських помилок

Ручна звірка схильна до помилок. Автоматизація мінімізує неточності. Це забезпечує точність даних.

Економія часу та ресурсів

Швидка обробка великих обсягів даних. Звільнення персоналу для інших завдань. Зниження операційних витрат.

Підвищення ефективності

Оптимізація складських процесів. Краще планування замовлень. Покращення управління запасами.

3



Існуючі рішення для автоматизації поставок: Аналіз ринку

Локальні POS-системи –

встановлюються на обладнанні закладу.

Переваги: працюють без інтернету, висока стабільність.

Недоліки: складне оновлення, обмеженість у віддаленому контролі.

Приклади: *r_keeper*, *Tillypad*, *KeepinCRM*.

Хмарні POS-системи – працюють онлайн з доступом до даних через вебінтерфейс.

Переваги: централізоване управління мережею, швидке оновлення, зручно для франшиз.

Недоліки: залежність від інтернету. Приклади: *Poster*, *iikoCloud*, *KeepinCRM*.

Гібридні POS-системи – поєднують локальне встановлення з можливістю хмарного зберігання та синхронізації.

Переваги: працюють навіть при нестабільному інтернеті, дозволяють віддалене управління, забезпечують резервне копіювання.

Недоліки: вища вартість впровадження, складніша технічна підтримка. Приклади: *iiko*, *Syrve*, *r_keeper Cloud*.

4



В чому необхідність створення такого рішення?



Відсутність ідентичного продукту на ринку.



Простота використання та гнучкість. Адаптування під будь-яку систему або самостійне використання.



Зменшення втрат від неточностей, крадіжок, раптових підвищень цін.

5

Пропонована система: Концепція та архітектура



Централізована база даних

Зберігання всіх даних про поставки та замовлення.



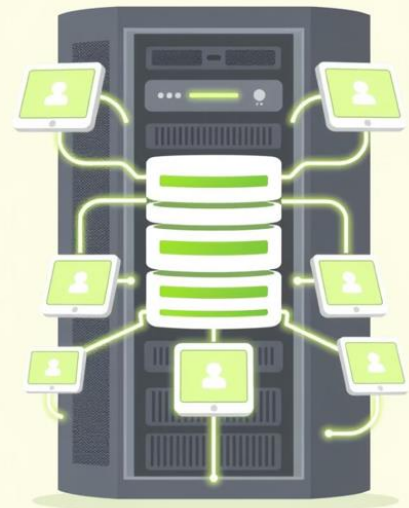
Призначення

Розширена аналітика: звірка приходів, зведені таблиці, аномалії та візуалізація цін



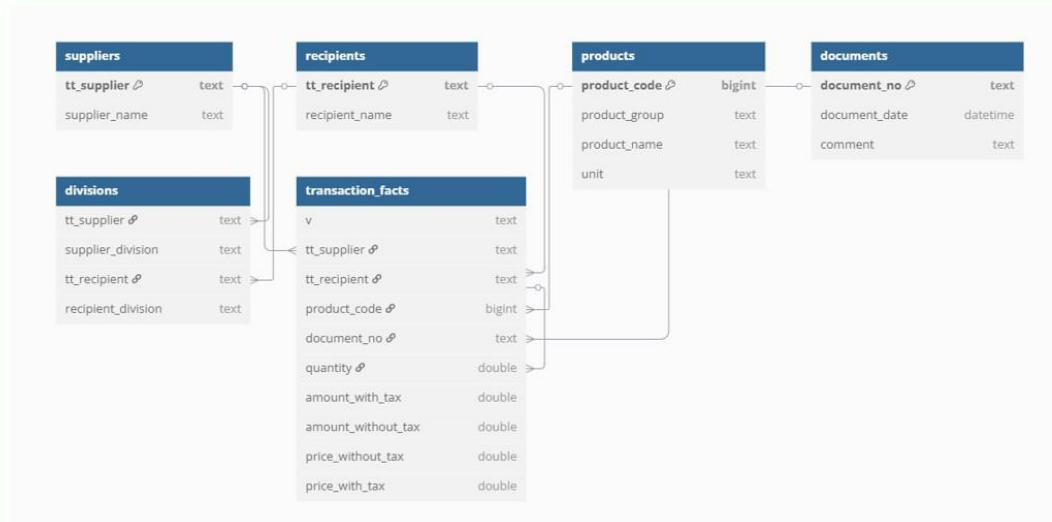
Інтерфейс

Зручний інтерфейс для звірки та аналізу даних



6

Запропонована база даних



7

Реалізація програми: Технології та підходи

Backend

- Python
- MySQL (база даних)
- Tkinter

Програмне рішення

- Зручне використання з ноутбуку
- Сумісність з різними системами
- Офлайн-режим

8

```

Version control
Current File
app.py x pivottable.py
181 anomaly_controls = tk.Frame(tab3)
182 anomaly_controls.pack(fill=tk.X, pady=5)
183
184 def show_anomaly_table(): 2 usages
185     global anomaly_pt
186     try:
187         df = generate_anomalies()
188         if df.empty:
189             messagebox.showinfo(title="Інформація", message="Аномалій")
190             return
191         if anomaly_pt is not None:
192             anomaly_pt.model.df = df
193             anomaly_pt.redraw()
194         else:
195             for widget in anomaly_frame.winfo_children():
196                 widget.destroy()
197             anomaly_pt = Table(anomaly_frame, dataframe=df, showtools=True)
198             anomaly_pt.show()
199     except Exception as e:
200         messagebox.showerror(title="Помилка", message=f"Помилка при зчитуванні даних")
201
202 btn_refresh_anom = tk.Button(
203     text="Оновити",
204     command=show_anomaly_table,
205     width=10,
206     height=1,
207     font=font,
208     bg="white",
209     border=1,
210     borderwidth=1,
211     relief="solid",
212     state="normal",
213     cursor="hand2",
214     activebackground="white",
215     activeborder="black",
216     activeborderwidth=1,
217     activefont=font,
218     activestate="normal",
219     default="normal",
220     overriderule="normal",
221     style="normal",
222     textcolor="black",
223     type="button",
224     winfo_children=[],
225     winfo_depth=1,
226     winfo_geometry="100x100+100+100",
227     winfo_height=100,
228     winfo_width=100,
229     x=100,
230     x2=200,
231     y=100,
232     y2=200,
233     z=1,
234     _root_=root,
235     _winfo_children_=[],
236     _winfo_depth_=1,
237     _winfo_geometry_='100x100+100+100',
238     _winfo_height_=100,
239     _winfo_width_=100,
240     _x_=100,
241     _x2_=200,
242     _y_=100,
243     _y2_=200,
244     _z_=1,
245     _root_=root,
246     _winfo_children_=[],
247     _winfo_depth_=1,
248     _winfo_geometry_='100x100+100+100',
249     _winfo_height_=100,
250     _winfo_width_=100,
251     _x_=100,
252     _x2_=200,
253     _y_=100,
254     _y2_=200,
255     _z_=1,
256     _root_=root,
257     _winfo_children_=[],
258     _winfo_depth_=1,
259     _winfo_geometry_='100x100+100+100',
260     _winfo_height_=100,
261     _winfo_width_=100,
262     _x_=100,
263     _x2_=200,
264     _y_=100,
265     _y2_=200,
266     _z_=1,
267     _root_=root,
268     _winfo_children_=[],
269     _winfo_depth_=1,
270     _winfo_geometry_='100x100+100+100',
271     _winfo_height_=100,
272     _winfo_width_=100,
273     _x_=100,
274     _x2_=200,
275     _y_=100,
276     _y2_=200,
277     _z_=1,
278     _root_=root,
279     _winfo_children_=[],
280     _winfo_depth_=1,
281     _winfo_geometry_='100x100+100+100',
282     _winfo_height_=100,
283     _winfo_width_=100,
284     _x_=100,
285     _x2_=200,
286     _y_=100,
287     _y2_=200,
288     _z_=1,
289     _root_=root,
290     _winfo_children_=[],
291     _winfo_depth_=1,
292     _winfo_geometry_='100x100+100+100',
293     _winfo_height_=100,
294     _winfo_width_=100,
295     _x_=100,
296     _x2_=200,
297     _y_=100,
298     _y2_=200,
299     _z_=1,
300     _root_=root,
301     _winfo_children_=[],
302     _winfo_depth_=1,
303     _winfo_geometry_='100x100+100+100',
304     _winfo_height_=100,
305     _winfo_width_=100,
306     _x_=100,
307     _x2_=200,
308     _y_=100,
309     _y2_=200,
310     _z_=1,
311     _root_=root,
312     _winfo_children_=[],
313     _winfo_depth_=1,
314     _winfo_geometry_='100x100+100+100',
315     _winfo_height_=100,
316     _winfo_width_=100,
317     _x_=100,
318     _x2_=200,
319     _y_=100,
320     _y2_=200,
321     _z_=1,
322     _root_=root,
323     _winfo_children_=[],
324     _winfo_depth_=1,
325     _winfo_geometry_='100x100+100+100',
326     _winfo_height_=100,
327     _winfo_width_=100,
328     _x_=100,
329     _x2_=200,
330     _y_=100,
331     _y2_=200,
332     _z_=1,
333     _root_=root,
334     _winfo_children_=[],
335     _winfo_depth_=1,
336     _winfo_geometry_='100x100+100+100',
337     _winfo_height_=100,
338     _winfo_width_=100,
339     _x_=100,
340     _x2_=200,
341     _y_=100,
342     _y2_=200,
343     _z_=1,
344     _root_=root,
345     _winfo_children_=[],
346     _winfo_depth_=1,
347     _winfo_geometry_='100x100+100+100',
348     _winfo_height_=100,
349     _winfo_width_=100,
350     _x_=100,
351     _x2_=200,
352     _y_=100,
353     _y2_=200,
354     _z_=1,
355     _root_=root,
356     _winfo_children_=[],
357     _winfo_depth_=1,
358     _winfo_geometry_='100x100+100+100',
359     _winfo_height_=100,
360     _winfo_width_=100,
361     _x_=100,
362     _x2_=200,
363     _y_=100,
364     _y2_=200,
365     _z_=1,
366     _root_=root,
367     _winfo_children_=[],
368     _winfo_depth_=1,
369     _winfo_geometry_='100x100+100+100',
370     _winfo_height_=100,
371     _winfo_width_=100,
372     _x_=100,
373     _x2_=200,
374     _y_=100,
375     _y2_=200,
376     _z_=1,
377     _root_=root,
378     _winfo_children_=[],
379     _winfo_depth_=1,
380     _winfo_geometry_='100x100+100+100',
381     _winfo_height_=100,
382     _winfo_width_=100,
383     _x_=100,
384     _x2_=200,
385     _y_=100,
386     _y2_=200,
387     _z_=1,
388     _root_=root,
389     _winfo_children_=[],
390     _winfo_depth_=1,
391     _winfo_geometry_='100x100+100+100',
392     _winfo_height_=100,
393     _winfo_width_=100,
394     _x_=100,
395     _x2_=200,
396     _y_=100,
397     _y2_=200,
398     _z_=1,
399     _root_=root,
400     _winfo_children_=[],
401     _winfo_depth_=1,
402     _winfo_geometry_='100x100+100+100',
403     _winfo_height_=100,
404     _winfo_width_=100,
405     _x_=100,
406     _x2_=200,
407     _y_=100,
408     _y2_=200,
409     _z_=1,
410     _root_=root,
411     _winfo_children_=[],
412     _winfo_depth_=1,
413     _winfo_geometry_='100x100+100+100',
414     _winfo_height_=100,
415     _winfo_width_=100,
416     _x_=100,
417     _x2_=200,
418     _y_=100,
419     _y2_=200,
420     _z_=1,
421     _root_=root,
422     _winfo_children_=[],
423     _winfo_depth_=1,
424     _winfo_geometry_='100x100+100+100',
425     _winfo_height_=100,
426     _winfo_width_=100,
427     _x_=100,
428     _x2_=200,
429     _y_=100,
430     _y2_=200,
431     _z_=1,
432     _root_=root,
433     _winfo_children_=[],
434     _winfo_depth_=1,
435     _winfo_geometry_='100x100+100+100',
436     _winfo_height_=100,
437     _winfo_width_=100,
438     _x_=100,
439     _x2_=200,
440     _y_=100,
441     _y2_=200,
442     _z_=1,
443     _root_=root,
444     _winfo_children_=[],
445     _winfo_depth_=1,
446     _winfo_geometry_='100x100+100+100',
447     _winfo_height_=100,
448     _winfo_width_=100,
449     _x_=100,
450     _x2_=200,
451     _y_=100,
452     _y2_=200,
453     _z_=1,
454     _root_=root,
455     _winfo_children_=[],
456     _winfo_depth_=1,
457     _winfo_geometry_='100x100+100+100',
458     _winfo_height_=100,
459     _winfo_width_=100,
460     _x_=100,
461     _x2_=200,
462     _y_=100,
463     _y2_=200,
464     _z_=1,
465     _root_=root,
466     _winfo_children_=[],
467     _winfo_depth_=1,
468     _winfo_geometry_='100x100+100+100',
469     _winfo_height_=100,
470     _winfo_width_=100,
471     _x_=100,
472     _x2_=200,
473     _y_=100,
474     _y2_=200,
475     _z_=1,
476     _root_=root,
477     _winfo_children_=[],
478     _winfo_depth_=1,
479     _winfo_geometry_='100x100+100+100',
480     _winfo_height_=100,
481     _winfo_width_=100,
482     _x_=100,
483     _x2_=200,
484     _y_=100,
485     _y2_=200,
486     _z_=1,
487     _root_=root,
488     _winfo_children_=[],
489     _winfo_depth_=1,
490     _winfo_geometry_='100x100+100+100',
491     _winfo_height_=100,
492     _winfo_width_=100,
493     _x_=100,
494     _x2_=200,
495     _y_=100,
496     _y2_=200,
497     _z_=1,
498     _root_=root,
499     _winfo_children_=[],
500     _winfo_depth_=1,
501     _winfo_geometry_='100x100+100+100',
502     _winfo_height_=100,
503     _winfo_width_=100,
504     _x_=100,
505     _x2_=200,
506     _y_=100,
507     _y2_=200,
508     _z_=1,
509     _root_=root,
510     _winfo_children_=[],
511     _winfo_depth_=1,
512     _winfo_geometry_='100x100+100+100',
513     _winfo_height_=100,
514     _winfo_width_=100,
515     _x_=100,
516     _x2_=200,
517     _y_=100,
518     _y2_=200,
519     _z_=1,
520     _root_=root,
521     _winfo_children_=[],
522     _winfo_depth_=1,
523     _winfo_geometry_='100x100+100+100',
524     _winfo_height_=100,
525     _winfo_width_=100,
526     _x_=100,
527     _x2_=200,
528     _y_=100,
529     _y2_=200,
530     _z_=1,
531     _root_=root,
532     _winfo_children_=[],
533     _winfo_depth_=1,
534     _winfo_geometry_='100x100+100+100',
535     _winfo_height_=100,
536     _winfo_width_=100,
537     _x_=100,
538     _x2_=200,
539     _y_=100,
540     _y2_=200,
541     _z_=1,
542     _root_=root,
543     _winfo_children_=[],
544     _winfo_depth_=1,
545     _winfo_geometry_='100x100+100+100',
546     _winfo_height_=100,
547     _winfo_width_=100,
548     _x_=100,
549     _x2_=200,
550     _y_=100,
551     _y2_=200,
552     _z_=1,
553     _root_=root,
554     _winfo_children_=[],
555     _winfo_depth_=1,
556     _winfo_geometry_='100x100+100+100',
557     _winfo_height_=100,
558     _winfo_width_=100,
559     _x_=100,
560     _x2_=200,
561     _y_=100,
562     _y2_=200,
563     _z_=1,
564     _root_=root,
565     _winfo_children_=[],
566     _winfo_depth_=1,
567     _winfo_geometry_='100x100+100+100',
568     _winfo_height_=100,
569     _winfo_width_=100,
570     _x_=100,
571     _x2_=200,
572     _y_=100,
573     _y2_=200,
574     _z_=1,
575     _root_=root,
576     _winfo_children_=[],
577     _winfo_depth_=1,
578     _winfo_geometry_='100x100+100+100',
579     _winfo_height_=100,
580     _winfo_width_=100,
581     _x_=100,
582     _x2_=200,
583     _y_=100,
584     _y2_=200,
585     _z_=1,
586     _root_=root,
587     _winfo_children_=[],
588     _winfo_depth_=1,
589     _winfo_geometry_='100x100+100+100',
590     _winfo_height_=100,
591     _winfo_width_=100,
592     _x_=100,
593     _x2_=200,
594     _y_=100,
595     _y2_=200,
596     _z_=1,
597     _root_=root,
598     _winfo_children_=[],
599     _winfo_depth_=1,
600     _winfo_geometry_='100x100+100+100',
601     _winfo_height_=100,
602     _winfo_width_=100,
603     _x_=100,
604     _x2_=200,
605     _y_=100,
606     _y2_=200,
607     _z_=1,
608     _root_=root,
609     _winfo_children_=[],
610     _winfo_depth_=1,
611     _winfo_geometry_='100x100+100+100',
612     _winfo_height_=100,
613     _winfo_width_=100,
614     _x_=100,
615     _x2_=200,
616     _y_=100,
617     _y2_=200,
618     _z_=1,
619     _root_=root,
620     _winfo_children_=[],
621     _winfo_depth_=1,
622     _winfo_geometry_='100x100+100+100',
623     _winfo_height_=100,
624     _winfo_width_=100,
625     _x_=100,
626     _x2_=200,
627     _y_=100,
628     _y2_=200,
629     _z_=1,
630     _root_=root,
631     _winfo_children_=[],
632     _winfo_depth_=1,
633     _winfo_geometry_='100x100+100+100',
634     _winfo_height_=100,
635     _winfo_width_=100,
636     _x_=100,
637     _x2_=200,
638     _y_=100,
639     _y2_=200,
640     _z_=1,
641     _root_=root,
642     _winfo_children_=[],
643     _winfo_depth_=1,
644     _winfo_geometry_='100x100+100+100',
645     _winfo_height_=100,
646     _winfo_width_=100,
647     _x_=100,
648     _x2_=200,
649     _y_=100,
650     _y2_=200,
651     _z_=1,
652     _root_=root,
653     _winfo_children_=[],
654     _winfo_depth_=1,
655     _winfo_geometry_='100x100+100+100',
656     _winfo_height_=100,
657     _winfo_width_=100,
658     _x_=100,
659     _x2_=200,
660     _y_=100,
661     _y2_=200,
662     _z_=1,
663     _root_=root,
664     _winfo_children_=[],
665     _winfo_depth_=1,
666     _winfo_geometry_='100x100+100+100',
667     _winfo_height_=100,
668     _winfo_width_=100,
669     _x_=100,
670     _x2_=200,
671     _y_=100,
672     _y2_=200,
673     _z_=1,
674     _root_=root,
675     _winfo_children_=[],
676     _winfo_depth_=1,
677     _winfo_geometry_='100x100+100+100',
678     _winfo_height_=100,
679     _winfo_width_=100,
680     _x_=100,
681     _x2_=200,
682     _y_=100,
683     _y2_=200,
684     _z_=1,
685     _root_=root,
686     _winfo_children_=[],
687     _winfo_depth_=1,
688     _winfo_geometry_='100x100+100+100',
689     _winfo_height_=100,
690     _winfo_width_=100,
691     _x_=100,
692     _x2_=200,
693     _y_=100,
694     _y2_=200,
695     _z_=1,
696     _root_=root,
697     _winfo_children_=[],
698     _winfo_depth_=1,
699     _winfo_geometry_='100x100+100+100',
700     _winfo_height_=100,
701     _winfo_width_=100,
702     _x_=100,
703     _x2_=200,
704     _y_=100,
705     _y2_=200,
706     _z_=1,
707     _root_=root,
708     _winfo_children_=[],
709     _winfo_depth_=1,
710     _winfo_geometry_='100x100+100+100',
711     _winfo_height_=100,
712     _winfo_width_=100,
713     _x_=100,
714     _x2_=200,
715     _y_=100,
716     _y2_=200,
717     _z_=1,
718     _root_=root,
719     _winfo_children_=[],
720     _winfo_depth_=1,
721     _winfo_geometry_='100x100+100+100',
722     _winfo_height_=100,
723     _winfo_width_=100,
724     _x_=100,
725     _x2_=200,
726     _y_=100,
727     _y2_=200,
728     _z_=1,
729     _root_=root,
730     _winfo_children_=[],
731     _winfo_depth_=1,
732     _winfo_geometry_='100x100+100+100',
733     _winfo_height_=100,
734     _winfo_width_=100,
735     _x_=100,
736     _x2_=200,
737     _y_=100,
738     _y2_=200,
739     _z_=1,
740     _root_=root,
741     _winfo_children_=[],
742     _winfo_depth_=1,
743     _winfo_geometry_='100x100+100+100',
744     _winfo_height_=100,
745     _winfo_width_=100,
746     _x_=100,
747     _x2_=200,
748     _y_=100,
749     _y2_=200,
750     _z_=1,
751     _root_=root,
752     _winfo_children_=[],
753     _winfo_depth_=1,
754     _winfo_geometry_='100x100+100+100',
755     _winfo_height_=100,
756     _winfo_width_=100,
757     _x_=100,
758     _x2_=200,
759     _y_=100,
760     _y2_=200,
761     _z_=1,
762     _root_=root,
763     _winfo_children_=[],
764     _winfo_depth_=1,
765     _winfo_geometry_='100x100+100+100',
766     _winfo_height_=100,
767     _winfo_width_=100,
768     _x_=100,
769     _x2_=200,
770     _y_=100,
771     _y2_=200,
772     _z_=1,
773     _root_=root,
774     _winfo_children_=[],
775     _winfo_depth_=1,
776     _winfo_geometry_='100x100+100+100',
777     _winfo_height_=100,
778     _winfo_width_=100,
779     _x_=100,
780     _x2_=200,
781     _y_=100,
782     _y2_=200,
783     _z_=1,
784     _root_=root,
785     _winfo_children_=[],
786     _winfo_depth_=1,
787     _winfo_geometry_='100x100+100+100',
788     _winfo_height_=100,
789     _winfo_width_=100,
790     _x_=100,
791     _x2_=200,
792     _y_=100,
793     _y2_=200,
794     _z_=1,
795     _root_=root,
796     _winfo_children_=[],
797     _winfo_depth_=1,
798     _winfo_geometry_='100x100+100+100',
799     _winfo_height_=100,
800     _winfo_width_=100,
801     _x_=100,
802     _x2_=200,
803     _y_=100,
804     _y2_=200,
805     _z_=1,
806     _root_=root,
807     _winfo_children_=[],
808     _winfo_depth_=1,
809     _winfo_geometry_='100x100+100+100',
810     _winfo_height_=100,
811     _winfo_width_=100,
812     _x_=100,
813     _x2_=200,
814     _y_=100,
815     _y2_=200,
816     _z_=1,
817     _root_=root,
818     _winfo_children_=[],
819     _winfo_depth_=1,
820     _winfo_geometry_='100x100+100+100',
821     _winfo_height_=100,
822     _winfo_width_=100,
823     _x_=100,
824     _x2_=200,
825     _y_=100,
826     _y2_=200,
827     _z_=1,
828     _root_=root,
829     _winfo_children_=[],
830     _winfo_depth_=1,
831     _winfo_geometry_='100x100+100+100',
832     _winfo_height_=100,
833     _winfo_width_=100,
834     _x_=100,
835     _x2_=200,
836     _y_=100,
837     _y2_=200,
838     _z_=1,
839     _root_=root,
840     _winfo_children_=[],
841     _winfo_depth_=1,
842     _winfo_geometry_='100x100+100+100',
843     _winfo_height_=100,
844     _winfo_width_=100,
845     _x_=100,
846     _x2_=200,
847     _y_=100,
848     _y2_=200,
849     _z_=1,
850     _root_=root,
851     _winfo_children_=[],
852     _winfo_depth_=1,
853     _winfo_geometry_='100x100+100+100',
854     _winfo_height_=100,
855     _winfo_width_=100,
856     _x_=100,
857     _x2_=200,
858     _y_=100,
859     _y2_=200,
860     _z_=1,
861     _root_=root,
862     _winfo_children_=[],
863     _winfo_depth_=1,
864     _winfo_geometry_='100x100+100+100',
865     _winfo_height_=100,
866     _winfo_width_=100,
867     _x_=100,
868     _x2_=200,
869     _y_=100,
870     _y2_=200,
871     _z_=1,
872     _root_=root,
873     _winfo_children_=[],
874     _winfo_depth_=1,
875     _winfo_geometry_='100x100+100+100',
876     _winfo_height_=100,
877     _winfo_width_=100,
878     _x_=100,
879     _x2_=200,
880     _y_=100,
881     _y2_=200,
882     _z_=1,
883     _root_=root,
884     _winfo_children_=[],
885     _winfo_depth_=1,
886     _winfo_geometry_='100x100+100+100',
887     _winfo_height_=100,
888     _winfo_width_=100,
889     _x_=100,
890     _x2_=200,
891     _y_=100,
892     _y2_=200,
893     _z_=1,
894     _root_=root,
895     _winfo_children_=[],
896     _winfo_depth_=1,
897     _winfo_geometry_='100x100+100+100',
898     _winfo_height_=100,
899     _winfo_width_=100,
900     _x_=100,
901     _x2_=200,
902     _y_=100,
903     _y2_=200,
904     _z_=1,
905     _root_=root,
906     _winfo_children_=[],
907     _winfo_depth_=1,
908     _winfo_geometry_='100x100+100+100',
909     _winfo_height_=100,
910     _winfo_width_=100,
911     _x_=100,
912     _x2_=200,
913     _y_=100,
914     _y2_=200,
915     _z_=1,
916     _root_=root,
917     _winfo_children_=[],
918     _winfo_depth_=1,
919     _winfo_geometry_='100x100+100+100',
920     _winfo_height_=100,
921     _winfo_width_=100,
922     _x_=100,
923     _x2_=200,
924     _y_=100,
925     _y2_=200,
926     _z_=1,
927     _root_=root,
928     _winfo_children_=[],
929     _winfo_depth_=1,
930     _winfo_geometry_='100x100+100+100',
931     _winfo_height_=100,
932     _winfo_width_=100,
933     _x_=100,
934     _x2_=200,
935     _y_=100,
936     _y2_=200,
937     _z_=1,
938     _root_=root,
939     _winfo_children_=[],
940     _winfo_depth_=1,
941     _winfo_geometry_='100x100+100+100',
942     _winfo_height_=100,
943     _winfo_width_=100,
944     _x_=100,
945     _x2_=200,
946     _y_=100,
947     _y2_=200,
948     _z_=1,
949     _root_=root,
950     _winfo_children_=[],
951     _winfo_depth_=1,
952     _winfo_geometry_='100x100+100+100',
953     _winfo_height_=100,
954     _winfo_width_=100,
955     _x_=100,
956     _x2_=200,
957     _y_=100,
958     _y2_=200,
959     _z_=1,
960     _root_=root,
961     _winfo_children_=[],
962     _winfo_depth_=1,
963     _winfo_geometry_='100x100+100+100',
964     _winfo_height_=100,
965     _winfo_width_=100,
966     _x_=100,
967     _x2_=200,
968     _y_=100,
969     _y2_=200,
970     _z_=1,
971     _root_=root,
972     _winfo_children_=[],
973     _winfo_depth_=1,
974     _winfo_geometry_='100x100+100+100',
975     _winfo_height_=100,
976     _winfo_width_=100,
977     _x_=100,
978     _x2_=200,
979     _y_=100,
980     _y2_=200,
981     _z_=1,
982     _root_=root,
983     _winfo_children_=[],
984     _winfo_depth_=1,
985     _winfo_geometry_='100x100+100+100',
986     _winfo_height_=100,
987     _winfo_width_=100,
988     _x_=100,
989     _x2_=200,
990     _y_=100,
991     _y2_=200,
992     _z_=1,
993     _root_=root,
994     _winfo_children_=[],
995     _winfo_depth_=1,
996     _winfo_geometry_='100x100+100+100',
997     _winfo_height_=100,
998     _winfo_width_=100,
999     _x_=100,
1000    _x2_=200,
1001    _y_=100,
1002    _y2_=200,
1003    _z_=1,
1004    _root_=root,
1005    _winfo_children_=[],
1006    _winfo_depth_=1,
1007    _winfo_geometry_='100x100+100+100',
1008    _winfo_height_=100,
1009    _winfo_width_=100,
1010    _x_=100,
1011    _x2_=200,
1012    _y_=100,
1013    _y2_=200,
1014    _z_=1,
1015    _root_=root,
1016    _winfo_children_=[],
1017    _winfo_depth_=1,
1018    _winfo_geometry_='100x100+100+100',
1019    _winfo_height_=100,
1020    _winfo_width_=100,
1021    _x_=100,
1022    _x2_=200,
1023    _y_=100,
1024    _y2_=200,
1025    _z_=1,
1026    _root_=root,
1027    _winfo_children_=[],
1028    _winfo_depth_=1,
1029    _winfo_geometry_='100x100+100+100',
1030    _winfo_height_=100,
1031    _winfo_width_=100,
1032    _x_=100,
1033    _x2_=200,
1034    _y_=100,
1035    _y2_=200,
1036    _z_=1,
1037    _root_=root,
1038    _winfo_children_=[],
1039    _winfo_depth_=1,
1040    _winfo_geometry_='100x100+100+100',
1041    _winfo_height_=100,
1042    _w
```



Демонстрація системи: Інтерфейс та функціонал

Модуль звірки

Огляд поставок та перевірка на внесення в базу.

Модуль звірки

Перевірка на внесення в базу, виявлення розбіжностей.

Зведена таблиця

Топ продуктів, популярні ціни

Аномалії

Виявлення аномальних «скачків» цін

Графік цін

Візуалізація даних для відстежування динаміки.

9



Демонстрація системи: Інтерфейс та функціонал

Модуль звірки

Огляд поставок та перевірка на внесення в базу.

Модуль звірки

Перевірка на внесення в базу, виявлення розбіжностей.

Зведена таблиця

Топ продуктів, популярні ціни

Аномалії

Виявлення аномальних «скачків» цін

Графік цін

Візуалізація даних для відстежування динаміки.

10

Звірка

Зведена таблиця Аномалії Графік цін

Оновити дані Експорт в Excel

Файл звірки: C:\Users\Grom\PycharmProjects\Reconciliation\test_файл Вибрати

Обробити файл Зберегти файл

	Дата документа	Номер документа	Сума	ТТ одержувач	Перевірка суми
3	2025-01-03	2209728401	1000.38	MISTER TWISTER	Сума не збігається (3753.359999999999)
4	2025-01-03	2209728086	4246.56	Помилка	Помилка
5	2025-01-03	2209728413	1407.12	PEST05	
6	2025-01-03	2209728414	2018.88	PEST05	
7	2025-01-03	2209728379	2408.40	PEST07	
8	2025-01-03	2209728380	1407.12	PEST07	
9	2025-01-07	2209732432	2331.36	PEST017	
10	2025-01-07	2209732227	455.52	PEST07	
11	2025-01-07	2209732228	3092.40	PEST07	
12	2025-01-07	2209732052	7738.56	PEST014	
13	2025-01-07	2209731986	951.60	PEST013	
14	2025-01-07	2209731987	5771.52	PEST013	
15	2025-01-07	2209732164	1998.00	PEST020	
16	2025-01-07	2209732124	959.40	PEST08	
17	2025-01-07	2209732125	4511.52	PEST08	
18	2025-01-07	2209732263	8444.16	EP1	
19	2025-01-07	2209732264	3131.28	PEST018	
20	2025-01-07	2209731441	455.52	PEST09	
21	2025-01-07	2209731442	6338.16	PEST09	

97 rows x 5 columns

Демонстрація системи: Інтерфейс та функціонал

Модуль звірки

Огляд поставок та перевірка на внесення в базу.

Модуль звірки

Перевірка на внесення в базу, виявлення розбіжностей.

Зведена таблиця

Топ продуктів, популярні ціни

Аномалії

Виявлення аномальних «скачків» цін

Графік цін

Візуалізація даних для відстежування динаміки.

11

Звірка

Зведена таблиця Аномалії Графік цін

Оновити дані Експорт в Excel

	Назва	Кількість	Сума з пода	Середня ціна	Топ-3 ціни
1	Сир моцарелла	2996.63	660793.43	220.51	165.00, 286.00, 572.00
2	олія соняшникова	3252.80	275117.60	84.58	97.40, 64.19, 69.77
3	вершки	2084.70	264689.80	126.97	117.90, 232.33, 121.05
4	Сир пармезан тертий	389.80	214653.50	550.67	550.00, 600.00, 0.00
5	Зелень Салат Романо	715.35	214392.79	299.70	275.00, 243.75, 400.00
6	Овочі перець болгарск	791.85	190314.45	240.34	235.75, 253.00, 241.50
7	суші креветки тигрові	415.00	189663.30	457.02	457.02
8	Молоко	2892.00	148857.12	51.47	52.23, 55.13, 49.62
9	С/м Овочі картофель	1932.50	133791.00	69.23	69.00, 62.80
10	Овочі гриби шампінй	1040.77	130435.87	125.33	121.55, 123.90, 128.80
11	Фрукти апельсин	1901.74	129419.85	68.05	68.05, 68.06, 68.66
12	Овочі картофель чиц	1839.27	116305.83	63.23	63.80, 62.10, 47.85
13	Пиво Хугарден keg	810.00	112560.00	138.96	140.70, 93.80
14	Пиво Степла Артуа к	1560.00	110448.00	70.80	70.80
15	соління Томати в влас	1320.00	109200.00	82.73	84.00, 0.00
16	Яловичий антрикот	317.06	106911.28	337.20	337.20
17	Вода кока-кола ж/б шт	5376.00	105663.44	19.65	19.68, 18.90, 16.87
18	м'ясо стейк рибай кг	149.16	104412.00	700.00	700.00
19	Вино Castelnuovo Pros	271.25	103079.70	380.02	378.62, 378.63, 340.76
20	Фрукти лимон	997.10	100530.58	100.82	87.75, 115.50, 112.35
21	морозиво	394.00	100195.14	254.30	270.41, 229.15, 252.72
22	Вакумований лосось с/	116.56	90916.80	780.00	780.00
23	Упаковка Сети на сто	100.00	90000.00	900.00	900.00

492 rows x 5 columns

Демонстрація системи: Інтерфейс та функціонал

Модуль звірки

Огляд поставок та перевірка на внесення в базу.

Модуль звірки

Перевірка на внесення в базу, виявлення розбіжностей.

Зведена таблиця

Топ продуктів, популярні ціни

Аномалії

Виявлення аномальних «скачків» цін

Графік цін

Візуалізація даних для відстежування динаміки.

12

Демонстрація системи: Інтерфейс та функціонал

Звірка

Зведена таблиця Аномалії Графік цін

Оновити дані Експорт в Excel

	Документ	Дата	ТТ постачальник	Назва	Ціна з подат	З_оцінка
112	13932-12796	2025-04-30	PEST06	віскі Johnnie Walker Red Label n.	882.43	4.00
113	13226-12103	2025-04-23	PEST06	віскі Johnnie Walker Red Label n.	882.43	4.00
114	10568/9675	2025-04-02	PEST06	віскі Johnnie Walker Red Label n.	882.43	4.00
115	1847	2025-04-24	PEST08	Овочі аспарегус кг	1733.33	3.99
116	24254	2025-04-25	PEST010	Соус кетчуп томатний кг	182.29	3.99
117	21037	2025-04-08	PEST012	Соус кетчуп томатний кг	182.29	3.99
118	1568	2025-04-07	PEST09	В_ПФ Шлінат бейби	487.60	3.88
119	5108051	2025-04-02	ЕР12 ЧАБАНИ	Вино кухня/гінтвейн, п	222.76	3.88
120	1812	2025-04-22	PEST08	Сметана	198.00	3.86
121	1841	2025-04-23	PEST08	Сметана	198.00	3.86
122	1568	2025-04-07	PEST09	Сметана	198.00	3.86
123	1530	2025-04-05	PEST019	Сметана	198.00	3.86
124	1529	2025-04-05	PEST013	Хлеб Батон	92.32	3.86
125	11968	2025-04-29	СКЛАД	Соус майонез	84.77	3.86
126	9879	2025-04-08	СКЛАД	Соус майонез	84.77	3.86
127	9862	2025-04-08	PEST018	Соус майонез	84.77	3.86
128	512601	2025-04-10	PEST013	Вино Castelnuovo Prosecco Spumante Ex	425.83	3.78
129	512601	2025-04-10	PEST013	нове вино Pinot Grigio delle Venezi n	283.50	3.78
130	18035	2025-04-11	PEST03	Сир твердий	550.00	3.66
131	16039	2025-04-01	PEST03	Сир твердий	550.00	3.66
132	21040	2025-04-25	PEST03	Сир твердий	550.00	3.66

319 rows x 6 columns

Модуль звірки

Огляд поставок та перевірка на внесення в базу.

Модуль звірки

Перевірка на внесення в базу, виявлення розбіжностей.

Зведена таблиця

Топ продуктів, популярні ціни

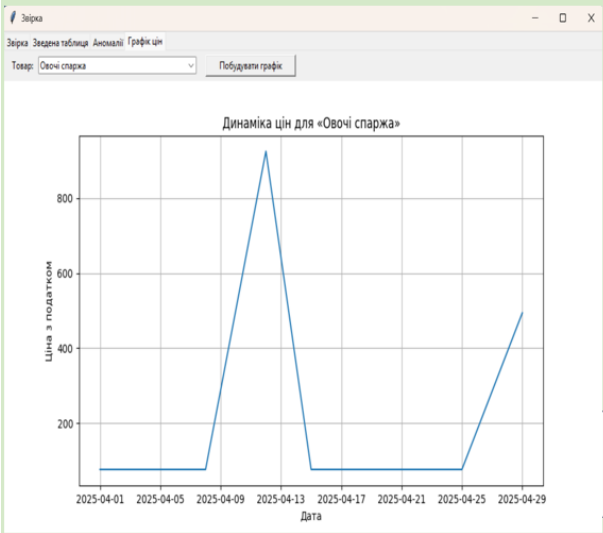
Аномалії

Виявлення аномальних «скачків» цін

Графік цін

Візуалізація даних для відстежування динаміки.

Демонстрація системи: Інтерфейс та функціонал



Модуль звірки

Огляд поставок та перевірка на внесення в базу.

Модуль звірки

Перевірка на внесення в базу, виявлення розбіжностей.

Зведена таблиця

Топ продуктів, популярні ціни

Аномалії

Виявлення аномальних «скачків» цін

Графік цін

Візуалізація даних для відстежування динаміки.



Яку користь ми отримуємо в результаті?



Прийом поставки

Автоматизована звірка приходів продукції. Порівняння отриманого з замовленим. Миттєве виявлення розбіжностей.



Контроль цін

Оперативне реагування на підвищення або зменшення вартості продукції.



Контроль персоналу

Моніторинг внесення приходів. Уникнення крадіжок або несанкціонованих списань.



Формування звітів

Статистика поставок. Аналіз ефективності та виявлення тенденцій.

15



Висновки та перспективи розвитку системи

1

Ключовий висновок

Автоматизація звірки поставок є необхідною для сучасної ресторанної мережі. Вона підвищує ефективність та знижує витрати.

2

Подальші плани

Інтеграція з POS-системами та CRM. Додавання модулів прогнозування попиту.

3

Майбутні можливості

Використання AI для оптимізації внесення приходів. Розширення на інші галузі.

16

Дякую за увагу!