

БАКАЛАВРСЬКА РОБОТА

КРБ.ІСТ – 05.00.00.000 ПЗ

Група ІСТ-22-1

Юрій ГОРБАНЬ

2026

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Інститут інформаційних технологій
Кафедра інформаційно-телекомунікаційних технологій та систем
Горбань Юрій Олександрович

(прізвище, ім'я, по батькові)

УДК 004.89:004.65
(індекс)

БАКАЛАВРСЬКА РОБОТА

Розроблення інформаційної системи з інтеграцією ІІІ для пошуку
та рекомендацій електротехнічної продукції підприємства

(назва роботи)

Інформаційні системи та технології

(назва освітньої програми)

126 – Інформаційні системи та технології

(шифр і назва спеціальності)

**Робота містить результати власних досліджень, використання ідей, результатів і
текстів інших авторів мають посилання на відповідне джерело:**

Здобувач освітнього ступеня Горбань Ю.О.
(підпис, ініціали та прізвище здобувача)

Науковий керівник Белей О.І.
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
В.о завідувача кафедри

доц Заміховська О.Л.
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2026

Івано-Франківський національний технічний університет нафти і газу

(повне найменування закладу вищої освіти)

Факультет інформаційних технологій

Кафедра інформаційно-телекомунікаційних технологій та систем

Освітній рівень бакалавр

Спеціальність 126 – Інформаційні системи та технології

(шифр і назва)

ЗАТВЕРДЖУЮ

В.о завідувача кафедри ІТТС

Заміховська О.Л.

«___» _____ 2026 року

ЗАВДАННЯ НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Горбаню Юрію Олександровичу

(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення інформаційної системи з інтеграцією ШІ для пошуку та рекомендацій електротехнічної продукції підприємства

керівник роботи Белей О.І., доц. каф. ІТТС

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “07” 04 2026 року № 163/7

2. Строк подання студентом роботи «07» червня 2026 р.

3. Вихідні дані: мова програмування: Python 3.10+; фреймворк: Flask; база даних: NoSQL (MongoDB); технології ШІ: Large Language Models (LLM) через API для обробки природної мови; інтерфейс: Bootstrap 5, JavaScript; функціонал: інтелектуальний пошук, система ролей, кошик, генерація PDF-рахунків з QR-кодами.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

Аналіз предметної області та технічних засобів розроблення (огляд ринку електротехніки, порівняння Flask/Django, вибір MongoDB)

Проектування архітектури та бази даних системи (UML-діаграми, структура колекцій MongoDB, алгоритми роботи ШІ-асистента)

Програмна реалізація вебдодатку (опис Backend на Python, інтеграція ШІ-асистента, розроблення Frontend-частини, система авторизації та ролей)

Тестування та аналіз результатів (функціональне тестування, перевірка безпеки, аналіз швидкодії).

Висновки

Додатки (лістинг коду, специфікація бази даних)

5. Графічна частина: Варіанти використання вебдодатку “electro_advisor”. UML-діаграма; База даних вебдодатку “electro_advisor”. Схема структурна; Вебдодаток “electro_advisor”; Вебдодаток “electro_advisor”

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Белей О.І., доцент		
2	Белей О.І., доцент		
3	Белей О.І., доцент		
4	Белей О.І., доцент		

7. Дата видачі завдання « 10 » лютого 2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Термін виконання етапів роботи	Примітка
1	Збір матеріалів за темою роботи	до 10.04	викон.
2	Загальні відомості про інформаційні системи з інтеграцією ІІІ для пошуку та рекомендацій	до 10.04	викон.
3	Дослідження практичної доцільності використання	до 10.04	викон.
4	Аналіз існуючих аналогів	до 10.04	викон.
5	Розроблення інформаційної системи	до 14.06	викон.
6	Опис програмного забезпечення для розроблення, вимоги перегляду розробленої інформаційної системи	до 14.06	викон.
7	Опис інтерфейсу користувача	до 14.06	викон.
8	Оцінка функціонального та практичного призначення	до 14.06	викон.
9	Оформлення пояснювальної записки та графічних матеріалів	до 14.06	викон.

Студент _____ Горбань Ю.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Белей О. І.
(підпис) (прізвище та ініціали)

АННОТАЦІЯ

У бакалаврській роботі спроектовано інтелектуальну автоматизовану систему управління каталогом електротехнічної продукції, що є актуальною задачею в умовах цифрової трансформації ритейлу та складської логістики. Розроблена інформаційна система забезпечує ефективну взаємодію користувача з асортиментом товарів, автоматизацію обліку залишків та формування фінансової документації.

Науковою новизною є розроблення архітектури системи з інтеграцією великої мовної моделі (LLM) Google Gemini для надання інтелектуальних консультацій у режимі реального часу. Вперше застосовано поєднання гнучкої NoSQL-структури бази даних MongoDB та алгоритмів семантичного аналізу технічних характеристик товарів, що дозволяє системі не лише відображати дані, а й підбирати аналоги та надавати експертні поради користувачам на основі контексту бази даних.

Практична цінність полягає у створенні масштабованого програмного рішення, яке автоматизує процес продажу: від інтелектуального підбору обладнання до генерації PDF-інвойсів із вбудованими QR-кодами для оплати. Спроектовану систему можна адаптувати для використання у будь-яких галузях електронної комерції та дистрибуції, де необхідна експертна підтримка клієнтів без залучення додаткового персоналу.

РЕФЕРАТ

Бакалаврська робота: 104 сторінки, 70 рисунків, 5 таблиць, 25 посилань на джерела, 3 додатки.

Об'єкт дослідження – процес пошуку та підбору електротехнічної продукції на спеціалізованих підприємствах за допомогою сучасних вебтехнологій.

Предмет дослідження – методи розроблення інформаційної системи на мові Python із застосуванням ШІ-алгоритмів для автоматичного формування рекомендацій та управління процесом замовлень.

У бакалаврській роботі запропоновано розроблення інтелектуальної інформаційної системи, що поєднує можливості вебплатформи для електронної комерції та ШІ-консультанта. Реалізований застосунок спрямований на автоматизацію підбору специфічного електрообладнання та надання інтелектуальних рекомендацій користувачам. Функціонал забезпечує безперервний цикл закупівлі, що включає контроль актуальних залишків на складі та фіналізацію угоди через автоматичну генерацію звітних документів.

Застосування подібної системи оптимізує операційну діяльність відділів збуту, знижує навантаження на технічних консультантів та мінімізує помилки при виборі специфічної електротехнічної продукції. Використання NoSQL бази даних та хмарних моделей ШІ забезпечує високу масштабованість та гнучкість рішення.

Ключові слова: ВЕБДОДАТОК, PYTHON, FLASK, ШТУЧНИЙ ІНТЕЛЕКТ (ШІ), MONGODB, ЕЛЕКТРОТЕХНІЧНА ПРОДУКЦІЯ, СИСТЕМА РЕКОМЕНДАЦІЙ, АВТОМАТИЗАЦІЯ ПРОДАЖІВ.

ABSTRACT

Bachelor's thesis: 104 pages, 70 figures, 5 tables, 25 references, 3 appendices.

The object of research is the process of searching for and selecting electrical products at specialized enterprises using modern web technologies.

The subject of the study is methods for developing an information system in Python using AI algorithms for automatically generating recommendations and managing the ordering process.

The bachelor's thesis proposes the development of an intelligent information system that combines the capabilities of a web platform for e-commerce and an AI consultant. The implemented application is aimed at automating the selection of specific electrical equipment and providing intelligent recommendations to users. The functionality provides a continuous procurement cycle, which includes monitoring current stock levels and finalizing transactions through the automatic generation of reporting documents.

The use of such a system optimizes the operational activities of sales departments, reduces the workload on technical consultants, and minimizes errors when selecting specific electrical products. The use of NoSQL databases and cloud-based AI models ensures high scalability and flexibility of the solution.

Keywords: WEB APPLICATION, PYTHON, FLASK, ARTIFICIAL INTELLIGENCE (AI), MONGODB, ELECTRICAL ENGINEERING PRODUCTS, RECOMMENDATION SYSTEM, SALES AUTOMATION.

ЗМІСТ

ВСТУП.....	9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНИХ ПІДХОДІВ РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ІНТЕГРАЦІЄЮ ІІІ ДЛЯ ПОШУКУ ТА РЕКОМЕНДАЦІЙ ЕЛЕКТРОТЕХНІЧНОЇ ПРОДУКЦІЇ ПІДПРИЄМСТВА.....	11
1.1 Аналіз ринку електротехнічної продукції та потреб автоматизації торгових процесів.....	11
1.2 Огляд існуючих систем рекомендацій та пошуку на основі штучного інтелекту.....	13
1.3 Порівняльний аналіз технологій розроблення вебдодатків (Python frameworks: Flask vs Django).....	16
1.4 Обґрунтування вибору NoSQL бази даних MongoDB для зберігання каталогу та історії замовлень.....	19
Висновки до першого розділу	21
2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ВЕБДОДАТКУ.....	22
2.1 Визначення функціональних та нефункціональних вимог до інформаційної системи.....	22
2.2 Проєктування клієнт-серверної архітектури та логіки взаємодії модулів системи.....	23
2.3 Моделювання структури бази даних (колекції продуктів, користувачів та замовлень).....	24
2.4 Вибір та обґрунтування моделі ІІІ для обробки природної мови (NLP) у пошукових запитах.....	28
Висновки до другого розділу	29
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБДОДАТКУ.....	30

					КРБ.ІСТ – 05.00.00.000 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Розроблення інформаційної системи з інтеграцією ІІІ для пошуку та рекомендацій електротехнічної продукції підприємства	Літ.	Арк.	Аркушів
Розроб.		Горбань Ю.О.					7	104
Перевір.		Белей О.І.						
Реценз.								
Н. Контр.		Возний А.						
Затверд.		Заміховська О.Л.				ІФНТУНГ ІСТ-22-1		

3.1 Розроблення серверної логіки на Flask (Backend) та інтеграція з базою даних.....	30
3.2 Реалізація інтелектуального пошуку та системи рекомендацій за допомогою ІІІ (Gemini/OpenAI API).....	32
3.3 Розроблення модуля управління користувачами, ролями та безпекою (хешування паролів, сесії).....	34
3.4 Створення адміністративної панелі для моніторингу складу та історії замовлень.....	36
3.5 Реалізація клієнтського інтерфейсу (Frontend) та модуля генерації електронних інвойсів.....	38
Висновки до третього розділу.....	41
4 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ.....	43
4.1 Методика та стратегія тестування вебдодатку.....	43
4.2 Функціональне тестування основних модулів (вхід, кошик, адмін-панель).....	45
4.3 Тестування точності ІІІ-пошуку та рекомендацій на реальних даних.....	62
4.4 Аналіз інтерфейсу користувача (UI/UX) та оцінка швидкодії системи.....	66
Висновки до четвертого розділу.....	66
ВИСНОВКИ.....	68
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	69
ДОДАТОК А. Програмна реалізація вебдодатка «electro_advisor».....	72
ДОДАТОК Б. Специфікація бази даних.....	101
ДОДАТОК В. Сертифікат наукової публікації.....	103

ВСТУП

В умовах інтенсивної цифрової трансформації економічного сектору оптимізація торговельних операцій та складським обліком є критично важливим для підприємств електротехнічної галузі. Специфіка електротехнічної продукції полягає у великій номенклатурі товарів, складності технічних характеристик та необхідності точного підбору обладнання під конкретні потреби замовника. Традиційні методи пошуку в інтернет-магазинах часто не враховують контекст запиту користувача, що підвищує ризики прийняття хибних рішень під час здійснення закупівель.

Впровадження інструментарію штучного інтелекту (ШІ) та сучасних вебфреймворків на мові Python дозволяє створити інтелектуальні системи, які здатні не лише автоматизувати облік товарів, а й виступати в ролі «цифрового консультанта», розпізнаючи природно-мовні запити та надаючи релевантні рекомендації. Це зумовлює актуальність розроблення інформаційної системи, що поєднує в собі функції ERP-системи, електронної комерції та інтелектуального помічника.

Об'єктом дослідження є процеси автоматизації складського обліку та реалізації електротехнічної продукції через вебінтерфейс з використанням інтелектуальних систем підтримки прийняття рішень.

Предметом дослідження є методи та засоби розроблення інформаційної системи на базі фреймворку Flask, інтеграція великих мовних моделей (LLM) для обробки запитів та використання NoSQL бази даних MongoDB для зберігання складних структур даних.

Метою роботи є розроблення та впровадження інтелектуальної інформаційної системи для підприємства, що забезпечує автоматизований пошук, персоналізовані рекомендації електротехнічної продукції, ведення обліку залишків та автоматичне формування платіжних документів.

Задачі для виконання мети:

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

- провести аналіз ринку та існуючих технологічних рішень для створення систем пошуку та рекомендацій продукції;
- проектувати архітектуру системи на основі клієнт-серверної взаємодії та гнучкої структури бази даних MongoDB;
- реалізувати механізм інтеграції зі штучним інтелектом для обробки запитів природною мовою та надання технічних рекомендацій;
- розробити систему автентифікації та розмежування прав доступу (гість, користувач, адміністратор);
- створити модуль автоматичної генерації рахунків-фактур у форматі PDF з інтегрованими QR-кодами для оплати;
- провести тестування функціональних модулів системи та оцінити ефективність роботи ІІІ-асистента.

Науково-практична цінність полягає у проектуванні та імплементації програмного інструментарію, який може бути впроваджений на малих та середніх підприємствах електротехнічної галузі. Розроблена система дозволяє:

- скоротити час на пошук необхідного обладнання завдяки ІІІ-консультанту;
- мінімізувати вплив «людського фактора» при формуванні замовлень;
- забезпечити прозорий облік товарів та історію взаємодії з клієнтами в реальному часі.

Розроблені рішення можуть бути інтегровані в діяльність організацій, що займаються гуртовою та роздрібною торгівлею електротехнічним обладнанням, сервісними центрами, а також інжиніринговими компаніями для автоматизації внутрішніх процесів постачання та взаємодії з клієнтами.

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНИХ ПІДХОДІВ РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ІНТЕГРАЦІЄЮ ШІ ДЛЯ ПОШУКУ ТА РЕКОМЕНДАЦІЙ ЕЛЕКТРОТЕХНІЧНОЇ ПРОДУКЦІЇ ПІДПРИЄМСТВА

1.1 Аналіз ринку електротехнічної продукції та потреб автоматизації торгових процесів

Сучасний ринок електротехнічної продукції характеризується надзвичайно широким номенклатурним переліком, що охоплює тисячі найменувань – від пасивних компонентів та кабельно-провідникової продукції до складних систем автоматизації, промислових контролерів та силового обладнання [1], [2]. Кожна одиниця товару має унікальний набір технічних характеристик, таких як номінальна напруга, сила струму, клас захисту IP, матеріал виконання та кліматичне виконання. Така специфіка створює значний інформаційний бар'єр для кінцевого споживача, який не завжди володіє глибокими інженерними знаннями для точного підбору обладнання. У зв'язку з цим виникає критична потреба у переході від традиційних статичних електронних каталогів до динамічних інтелектуальних систем, які здатні інтерпретувати технічні запити користувачів.

Процес цифрової трансформації в електротехнічній галузі висуває нові вимоги до автоматизації торгових процесів. На сьогодні підприємства стикаються з проблемою високого навантаження на технічних консультантів, які змушені витрачати значний час на обробку типових запитів та перевірку наявності товарів на складі. Автоматизація дозволяє вирішити низку стратегічних завдань: забезпечення актуальності складських залишків у режимі реального часу, розмежування прав доступу для різних категорій контрагентів (B2B-клієнтів та роздрібних покупців) та мінімізацію помилок при оформленні первинної документації. Особливої ваги набуває можливість персоналізації

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		11

взаємодії з клієнтом, що реалізується через систему особистих кабінетів зі збереженням історії покупок та специфікацій [3].

Важливим аспектом сучасного підходу є впровадження інтелектуальних методів пошуку. Традиційні фільтри за параметрами часто виявляються малоефективними, коли користувач формулює запит на основі прикладної задачі, а не конкретного артикула. Потреба автоматизації торгових процесів у цьому контексті виходить за межі простої фіксації факту продажу. Вона передбачає створення екосистеми, де штучний інтелект виступає ланкою між складною базою даних продукції та природною мовою замовника.



Рисунок 1.1 – Автоматизація торгового процесу за допомогою ШІ [3]

Це дозволяє не лише пришвидшити цикл закупівлі, а й забезпечити автоматичне формування супровідних фінансових документів, таких як рахунки-фактури з QR-кодами для швидкої оплати, що значно скорочує час на фінансові операції [4]. Таким чином, наскрізна автоматизація перетворюється на фундаментальний фактор збереження ринкових позицій компанії, забезпечуючи можливість поєднати точність інженерних розрахунків із швидкістю сучасних вебсервісів.

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		12

1.2 Огляд існуючих систем рекомендацій та пошуку на основі штучного інтелекту

Сучасний етап розвитку інформаційних технологій характеризується переходом від простих алгоритмів пошуку за ключовими словами до складних інтелектуальних систем, що базуються на машинному навчанні та обробці природної мови [5]. Аналіз рішень глобальних лідерів електронної комерції дозволяє виділити ключові стратегії застосування ШІ, що представлені у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика інтелектуальних систем пошуку та рекомендацій

Компанія	Домінуюча технологія ШІ	Ключова перевага	Приклад застосування
Amazon [6]	LLM та Content Filtering	Експертна аналітика	Резюмування відгуків та аналіз трендів брендів
Alibaba [7]	Computer Vision (Alibaba Lens)	Швидкість ідентифікації	Пошук обладнання за фотографією чи ескізом
eBay [8]	Knowledge Graphs (Графи знань)	Структурування даних	Фільтрація за брендами та технічними категоріями

Основними методами побудови в таких системах виступають колаборативна та контентна фільтрації, а також їхні гібридні модифікації. Для електротехнічного сектору колаборативна фільтрація має обмеження, оскільки вибір обладнання частіше диктується технічними параметрами, а не вподобаннями інших покупців. Пріоритетною є контентна фільтрація, яка аналізує внутрішні властивості товарів. На прикладі платформи Amazon (див. рисунок 1.2-1.3) видно, як система аналізує ринкові показники брендів та групує технічно схожі товари (наприклад, коробки на різну кількість контактів).

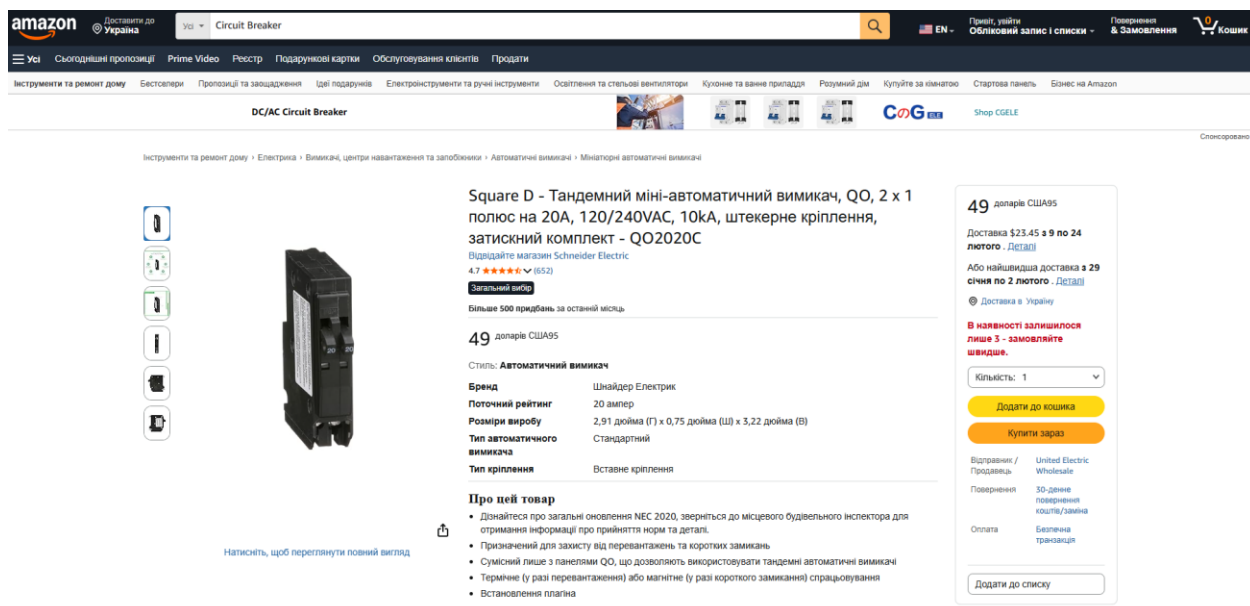


Рисунок 1.2 – Використання ШІ на Amazon для аналізу рейтингів брендів та групування товарів за технічними специфікаціями [6]

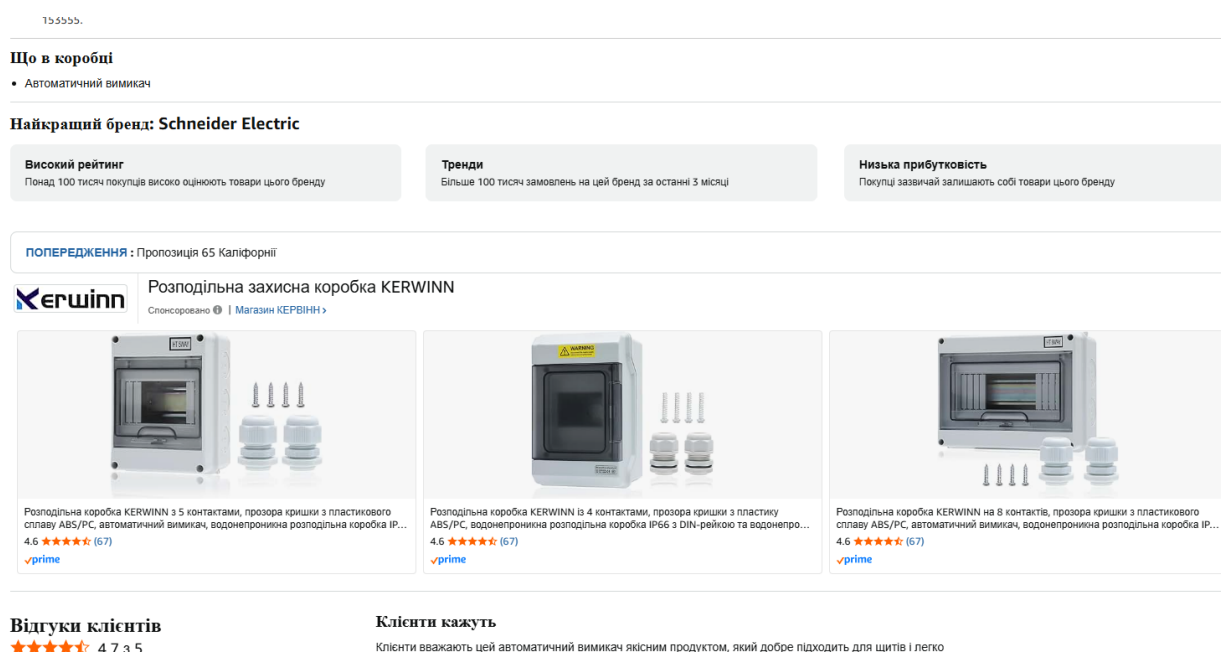


Рисунок 1.3 – Використання ШІ на Amazon для аналізу рейтингів брендів та групування товарів за технічними специфікаціями [6]

Важливим напрямком є візуальний пошук, де лідером виступає Alibaba [7]. Завдяки технології «Alibaba Lens» (див. рисунок 1.4), нейронні мережі здатні розпізнавати складні об'єкти на зображеннях, що дозволяє миттєво

знайти потрібний вузол або деталь у гігантській базі даних без знання точної назви.

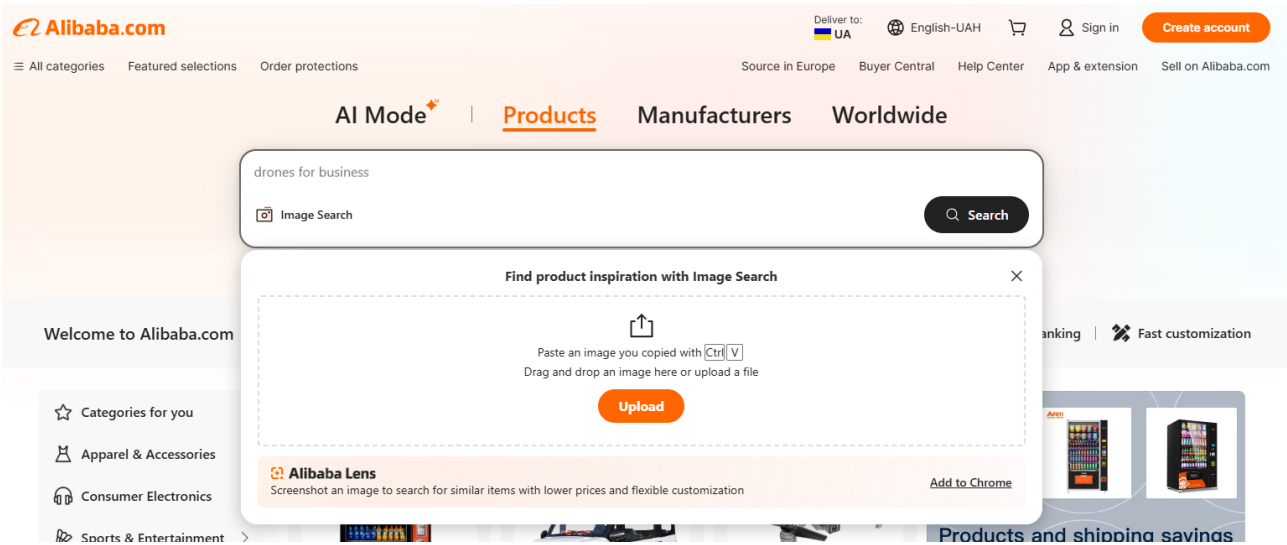


Рисунок 1.4 – Інтерфейс інструменту «Alibaba Lens» для візуального пошуку товарів на основі комп'ютерного зору [7]

Для роботи з великою кількістю неструктурованих оголошень від різних постачальників платформа eBay використовує графи знань [8]. Це дозволяє системі автоматично виділяти ключові сутності (бренди, типи двигунів, цінові діапазони) та формувати точні навігаційні фільтри (див. рисунок 1.5), що критично важливо для інженерного підбору.

Значний прорив також стався завдяки впровадженню семантичного аналізу та архітектури RAG (Retrieval-Augmented Generation) [9]. Це дозволяє поєднувати творчі можливості великих мовних моделей із точними даними зі складських баз даних. Таким чином, сучасні підходи зміщують акцент з пасивного очікування запиту на проактивну допомогу, де ШІ виступає в ролі експертного консультанта, здатного зрозуміти технічний контекст «захисту від перевантаження» навіть без дослівного згадування назви пристрою.

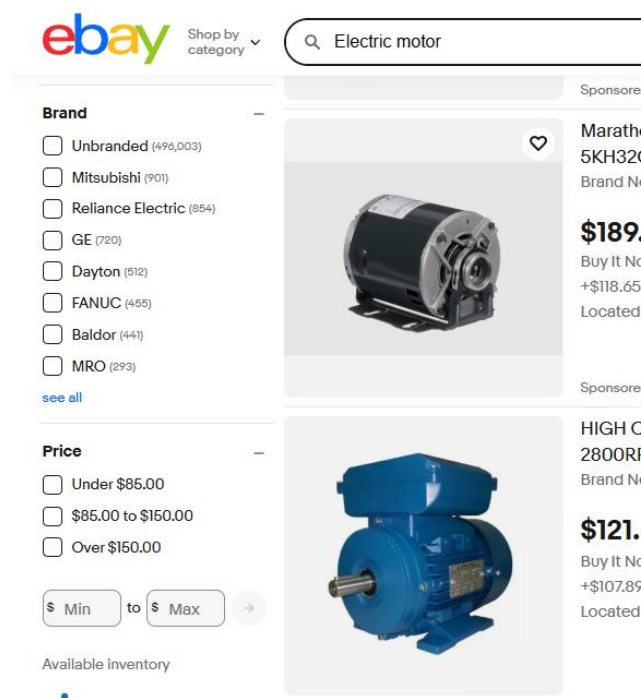


Рисунок 1.5 – Структурування технічних категорій та фільтрація за брендами за допомогою графів знань у системі eBay [8]

1.3 Порівняльний аналіз технологій розроблення вебдодатків (Python frameworks: Flask vs Django)

Вибір технологічного стеку для розроблення інтелектуального вебдодатку є визначальним фактором, що впливає на швидкість розгортання, масштабованість та легкість інтеграції зі сторонніми сервісами. Мова програмування Python де-факто стала стандартом для проєктів, що включають штучний інтелект, завдяки найбагатшій екосистемі бібліотек [10]. Однак вибір конкретного вебфреймворку зазвичай коливається між двома лідерами: Django та Flask.

Django позиціонується як високорівневий фреймворк із філософією «всі батареї в комплекті» (batteries-included) [11]. Він пропонує жорстку структуру проєкту, вбудовану систему автентифікації, адміністративну панель та потужну ORM, орієнтовану на реляційні бази даних. Хоча такий підхід пришвидшує розробку типових корпоративних порталів, він накладає значні

обмеження на гнучкість архітектури та створює надлишкову складність у проєктах, де не потрібен увесь базовий функціонал [13].

Control Flow Of MVT

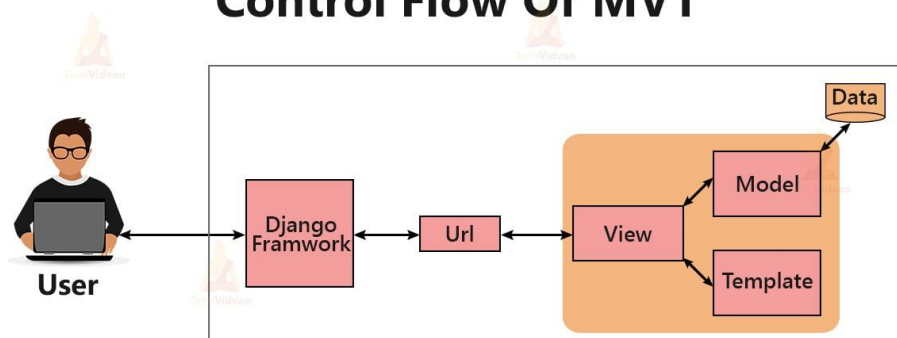


Рисунок 1.6 – Архітектура Django MVT (Model-View-Template) [12]

На противагу Django, Flask є мікрофреймворком, який надає розробнику лише базовий інструментарій для маршрутизації та обробки запитів, залишаючи вибір архітектурних рішень, методів роботи з базою даних та способів авторизації за автором. Така модульність є критично важливою для розроблення спеціалізованих систем з інтеграцією ІІІ. Flask дозволяє уникати «архітектурного шуму» та легко інтегрувати нестандартні рішення, такі як NoSQL база даних MongoDB, що є складним завданням для стандартної ORM у Django [14], [15].

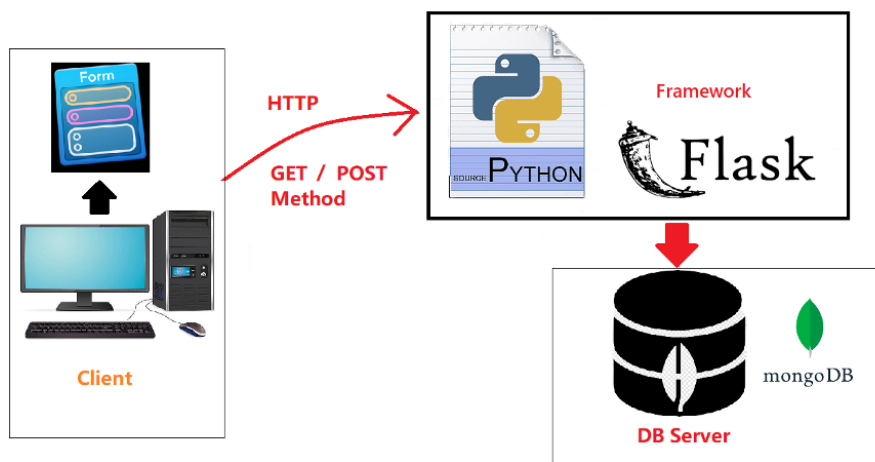


Рисунок 1.7 – Принцип роботи фреймворку Flask з користувачем та базою даних MongoDB [17]

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		17

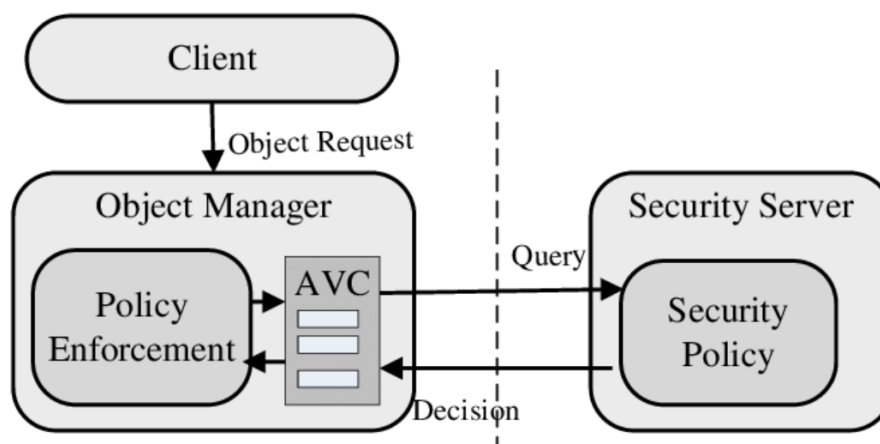


Рисунок 1.8 – Архітектура фреймворку Flask [16]

Оскільки інтелектуальний пошук та рекомендації потребують специфічної логіки взаємодії з API великих мовних моделей, легкість та прозорість Flask дозволяють реалізувати асинхронні запити та гнучку обробку даних без конфліктів із внутрішніми механізмами фреймворку [14].

Порівняльний аналіз показує, що для реалізації вебдодатку з пошуку електротехнічної продукції перевага Flask полягає у його здатності адаптуватися до специфічних вимог проекту. Використання Flask дозволяє створити максимально легку та швидку серверну частину, де кожен компонент – від генерації PDF-інвойсів до діалогового III-інтерфейсу – інтегрується як незалежний модуль. Це забезпечує вищу швидкість виконання запитів та спрощує подальше тестування системи. Таким чином, якщо Django є оптимальним для великих контент-орієнтованих систем із фіксованою структурою, то Flask стає ідеальним інструментом для інноваційних проєктів, де ключовим пріоритетом визначено гнучкість імплементації інноваційних рішень у сфері штучного інтелекту та висока швидкість ітерацій розробки.

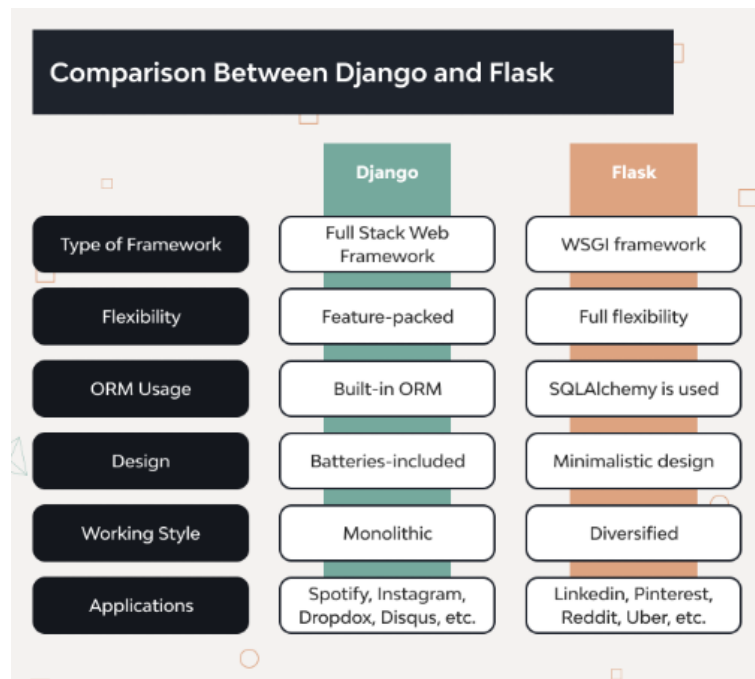


Рисунок 1.9 – Наглядне порівняння фреймворків Django та Flask [13]

1.4 Обґрунтування вибору NoSQL бази даних MongoDB для зберігання каталогу та історії замовлень

Вибір системи управління базами даних для проекту з великою кількістю технічно складних товарів зумовлений необхідністю гнучкого моделювання даних. Електротехнічна продукція відрізняється неоднорідністю характеристик: наприклад, для автоматичного вимикача критичними є номінальний струм та вимикаюча здатність, тоді як для кабельної продукції – переріз жил та тип ізоляції. Застосування класичних реляційних моделей (SQL) для вирішення цього завдання призвело б до створення надмірно складних схем із великою кількістю порожніх полів або громіздких таблиць зв'язків. Документоорієнтована NoSQL база даних MongoDB розв'язує цю проблему завдяки динамічній схемі (schemaless), що дозволяє зберігати кожен товар як окремий JSON-подібний документ із власним набором атрибутів без втрати продуктивності [18], [19], [25] (див. додаток В).

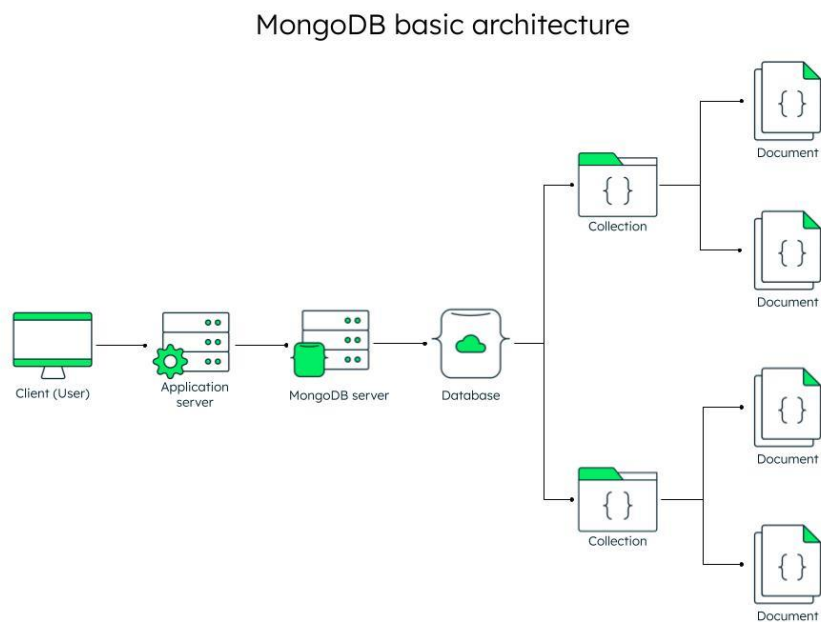


Рисунок 1.10 – Базова архітектура MongoDB [19]

Важливою перевагою MongoDB є ідентичність формату зберігання даних (BSON) і формату обміну даними в вебдодатку (JSON). Це значно спрощує розробку на Python за допомогою Flask, оскільки дані з бази безпосередньо мапяться на словники Python та об'єкти JavaScript на фронтенді, мінімізуючи витрати ресурсів на конвертацію типів. У контексті зберігання історії замовлень MongoDB дозволяє використовувати вкладені документи. Замість виконання складних операцій об'єднання (JOIN) декількох таблиць, як це прийнято в SQL, інформація про всі товари в чеку зберігається безпосередньо всередині документа замовлення. Це гарантує цілісність даних: навіть якщо ціна чи опис товару в каталозі зміняться в майбутньому, в історії замовлення збережеться саме той стан об'єкта, який був актуальним на момент покупки [17], [25] (див. додаток В).

Крім того, висока швидкість операцій читання та запису в MongoDB забезпечує миттєву реакцію інтелектуального пошуку. Оскільки штучний інтелект часто оперує неструктурованими описами та технічними характеристиками, формат документів MongoDB є найбільш природним середовищем для зберігання результатів семантичного аналізу. Можливість

легкого горизонтального масштабування та вбудована підтримка індексації великих масивів даних роблять MongoDB оптимальним вибором для підприємства, що планує постійне розширення асортименту та накопичення значних обсягів інформації про клієнтський досвід та комунікації [18], [20].

Висновки до першого розділу

На основі опрацьованого в першому розділі матеріалу встановлено наступне:

- специфіка ринку електротехніки з великою номенклатурою та складними характеристиками товарів потребує переходу від стандартних фільтрів до інтелектуальних систем підбору. Автоматизація торгових процесів є необхідною для підвищення точності замовлень та актуальності складського обліку;

- дослідження методів пошуку підтвердило перевагу семантичних ШІ-моделей (LLM) над класичним пошуком за ключовими словами. Це дозволяє реалізувати функцію «цифрового консультанта», що розуміє технічний контекст запитів природною мовою;

- обґрунтовано вибір мови Python та мікрофреймворку Flask. Його модульність і легкість забезпечують гнучку інтеграцію ШІ-інструментів та сервісів генерації документів, що є складнішим у монолітних фреймворках на кшталт Django;

- визначено доцільність використання NoSQL бази даних MongoDB. Її документоорієнтована модель ідеально підходить для зберігання неоднорідних технічних параметрів продукції та вкладених структур замовлень, забезпечуючи високу швидкість роботи та масштабованість системи.

Обраний технологічний стек створює оптимальні умови для реалізації функціонального та ефективного вебдодатка, що відповідає сучасним вимогам автоматизації підприємства.

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		21

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ВЕБДОДАТКУ

2.1 Визначення функціональних та нефункціональних вимог до інформаційної системи

Проєктування інтелектуального вебдодатка розпочинається з чіткого окреслення вимог, які визначають як внутрішню логіку роботи, так і зовнішню взаємодію з користувачем. Функціональні вимоги описують конкретні сервіси та можливості, які система повинна надавати для вирішення бізнес-задач підприємства. Насамперед це стосується управління каталогом електротехнічної продукції, де система має забезпечувати перегляд товарів, їх фільтрацію за технічними параметрами та виконання інтелектуального пошуку на основі природно-мовної обробки запитів через ШІ-асистента.

Для розмежування прав доступу система повинна підтримувати три ролі користувачів: гості (перегляд), авторизовані покупці (замовлення та історія) та адміністратори (повне керування складом та користувачами). Важливою функціональною частиною є механізм обробки замовлень, що включає роботу з «кошиком», автоматичне оновлення залишків у базі даних MongoDB та миттєву генерацію фінансових документів у форматі PDF із вбудованими QR-кодами для проведення безготівкових платежів. Адміністративна панель повинна надавати інструменти для CRUD-операцій (створення, читання, оновлення, видалення) товарів та перегляду детальної історії активності клієнтів [4].

До категорії нефункціональних вимог належать якісні параметри застосунку, зокрема показники продуктивності, стійкості до навантажень та рівень захищеності даних. Вимоги до продуктивності передбачають високу швидкість відгуку інтерфейсу та мінімальну затримку при зверненні до ШІ-моделей (не більше декількох секунд для формування рекомендації). З позиції захищеності даних та конфіденційності, система повинна гарантувати надійне зберігання конфіденційних даних через незворотне хешування паролів та захист сесій від несанкціонованого доступу. Надійність забезпечується

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		22

використанням транзакційної стійкості бази даних та коректною обробкою помилок при роботі із зовнішніми API [14], [22].

Масштабованість системи має дозволяти легке розширення асортименту продукції та збільшення кількості одночасних сесій без деградації швидкодії. Вимоги до юзабіліті (зручності використання) передбачають адаптивний дизайн на основі Bootstrap, що забезпечує коректне відображення інтерфейсу на мобільних та десктопних пристроях [21]. Ці вимоги формують цілісну структуру системи, де технічна реалізація на Flask та MongoDB безпосередньо відповідає потребам оперативності та гнучкості сучасного торгового підприємства.

2.2 Проєктування клієнт-серверної архітектури та логіки взаємодії модулів системи

Архітектура розроблюваного вебдодатка базується на класичній дворівневій моделі «клієнт-сервер», де чітко розмежовані інтерфейс користувача та логіка обробки даних. Фронтенд-складова системи реалізована через поєднання HTML5, CSS3 та JavaScript, де бібліотека Bootstrap 5 відповідає за коректне відображення на різних пристроях та динамічність взаємодії [21]. У свою чергу, бекенд системи побудовано на базі Flask – мікрофреймворку, що виступає центральним вузлом системи, обробляючи HTTP-запити, керуючи сесіями користувачів та забезпечуючи маршрутизацію до відповідних функціональних модулів [16].

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		23

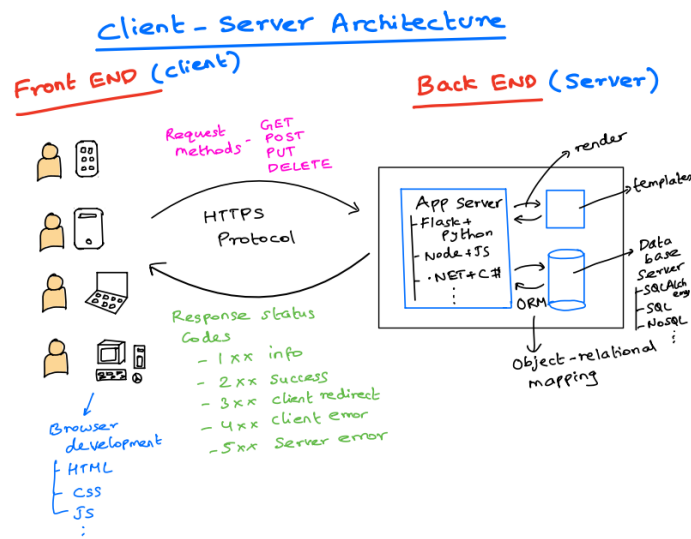


Рисунок 2.1 – Архітектура «Клієнт-Сервер» [23]

Взаємодія компонентів відбувається через механізм запитів та відповідей. При зверненні користувача сервер Flask ініціює запити до NoSQL-бази даних MongoDB для отримання інформації про товари або авторизації користувача (див. рисунок 1.7). Особливістю архітектури є інтеграція зовнішнього інтелектуального вузла – API III-моделі (Gemini/OpenAI), який підключається як окремий сервіс для обробки семантичних запитів [5]. Дані між клієнтом і сервером, а також між сервером і зовнішніми API передаються у форматі JSON, що забезпечує високу швидкість обміну та сумісність структур.

Модульна структура архітектури дозволяє ізолювати специфічні завдання, такі як генерація PDF-документів або обробка замовлень, у межах окремих функцій, що спрощує масштабування та технічну підтримку [4]. Таким чином, спроектована архітектура забезпечує цілісну взаємодію всіх підсистем: від введення запиту користувачем у браузері до отримання інтелектуальної рекомендації та збереження результату транзакції в базі даних.

2.3 Моделювання структури бази даних (колекції продуктів, користувачів та замовлень)

Моделювання бази даних для вебдодатка виконано з урахуванням гнучкості NoSQL-підходу, що дозволяє зберігати дані у вигляді JSON-подібних

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		24

документів (BSON) [18], [19]. Основна логіка системи зосереджена у трьох ключових колекціях MongoDB, які забезпечують роботу каталогу, системи безпеки та обліку продажів (див. рисунок 2.2-2.3).

Структурна схема бази даних, що відображає склад кожного документа та типи даних, зображена на рисунку 2.2. Специфікацію бази даних наведено у додатку Б.

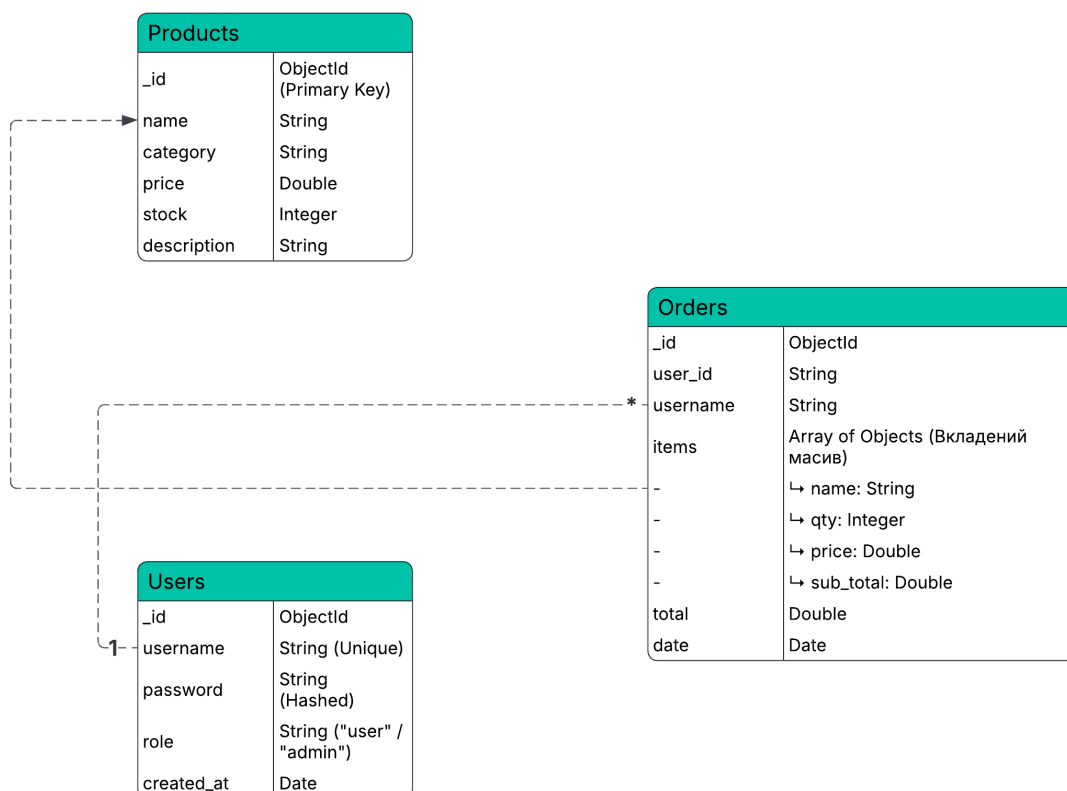


Рисунок 2.2 – Структурна схема бази даних додатку “electro_advisor”

Для візуалізації взаємодії об'єктів системи та логіки їхньої роботи, використовуючи програмне забезпечення Lucidchart [24], було побудовано UML-діаграму класів, яка зображена на рисунку 2.3. Вона відображає не лише структуру даних, а й ключові методи, такі як генерація інвойсів та взаємодія з ІІІ-модулем.

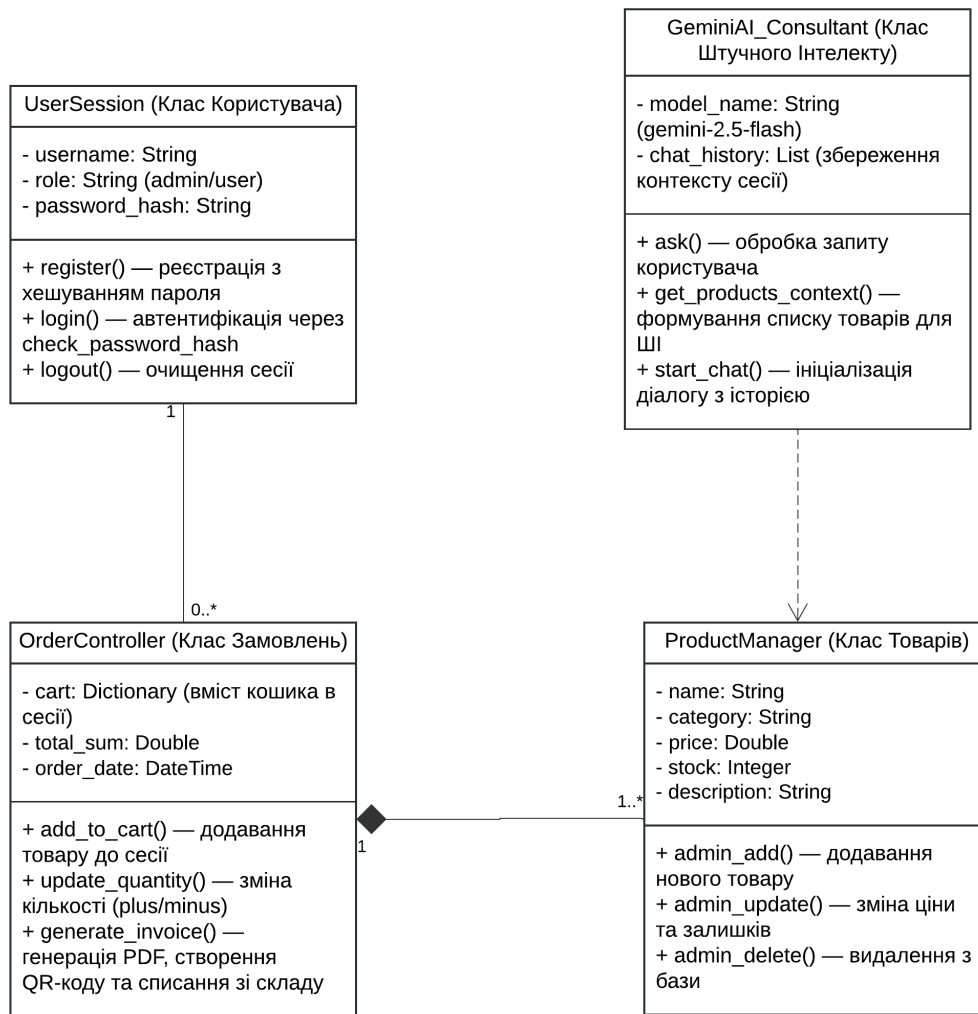


Рисунок 2.3 – UML-діаграма класів додатку “electro_advisor”

На відміну від статичної схеми бази даних (див. рисунок 2.2), UML-діаграма деталізує динамічні процеси системи:

- клас UserSession забезпечує безпеку через методи валідації та шифрування;
- клас OrderController реалізує алгоритм проведення транзакції, включаючи перевірку складських залишків та фіналізацію у вигляді PDF-інвойсу з вбудованим QR-кодом;

– клас GeminiAI_Consultant демонструє інтеграцію великої мовної моделі, яка використовує метод start_chat для підтримки контексту розмови та аналізу асортименту в режимі реального часу.

Такий підхід до моделювання дозволяє наочно продемонструвати логіку роботи бекенд-частини додатка та взаємозв'язок між бізнес-процесами торгівлі та штучним інтелектом».

Колекція Products спроектована за принципом динамічної схеми. Кожен документ містить обов'язкові поля: унікальну назву (name), категорію (category), ціну (price) та кількість одиниць на складі (stock). Важливою особливістю є поле опису (description), яке використовується ШІ-асистентом для семантичного аналізу та підбору аналогів. Завдяки відсутності жорсткої структури, документи цієї колекції можуть легко розширюватися додатковими технічними параметрами (наприклад, напруга, потужність, рівень захисту IP) без необхідності модифікації всієї бази даних.

Колекція Users відповідає за автентифікацію та розмежування прав доступу. Вона містить унікальне ім'я користувача (username), хешований пароль (password), що гарантує безпеку персональних даних, та поле ролі (role), яке набуває значень «user» або «admin». Також зберігається дата реєстрації (created_at), що дозволяє адміністратору відстежувати активність клієнтської бази та формувати статистику нових користувачів.

Колекція Orders реалізована за методом денормалізації даних для забезпечення цілісності історії продажів. Кожне замовлення зберігається як окремий документ, що включає посилання на клієнта (username), дату здійснення операції (date) та загальну суму (total). Ключовим елементом є вкладений масив об'єктів items, де зберігаються копії даних про товари (назва, ціна на момент купівлі та кількість) безпосередньо в момент транзакції. Такий підхід дозволяє зберігати коректну історію замовлень навіть у разі подальшої зміни ціни або видалення товару з основного каталогу, що є критично важливим для бухгалтерської звітності та генерації PDF-інвойсів.

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		27

Така структура бази даних мінімізує кількість складних операцій об'єднання (joins), забезпечуючи високу швидкість відгуку вебдодатка та легку масштабованість при збільшенні обсягів інформації.

2.4 Вибір та обґрунтування моделі ШІ для обробки природної мови (NLP) у пошукових запитах

Для реалізації інтелектуальної складової вебдодатка було обрано технологію великих мовних моделей (LLM), зокрема модель Gemini від Google, що інтегрується через прикладний програмний інтерфейс (API). Вибір саме такої моделі зумовлений її високою здатністю до контекстуального розуміння природної мови та можливістю оперувати складними технічними поняттями без необхідності додаткового навчання (fine-tuning) на специфічних наборах даних.

Обґрунтування використання LLM для пошуку електротехнічної продукції базується на переході від класичного лексичного пошуку до семантичного. Традиційні алгоритми пошуку в базах даних (наприклад, оператор \$regex у MongoDB) реагують лише на точні збіги символів, що є неефективним, коли користувач формує запит на основі інженерної задачі, а не конкретного артикулу [20]. Модель ШІ виконує роль інтелектуального прошарку, який інтерпретує запит користувача (наприклад, «щось для захисту двигуна від перегріву»), аналізує описи товарів у колекції products та знаходить релевантні відповіді (теплове реле чи автоматичний вимикач), навіть якщо вихідний запит не містив цих назв.

Технічна реалізація базується на принципі Zero-shot Learning, що дозволяє моделі класифікувати та підбирати товари на основі вбудованих знань про електротехніку [5]. Серверна частина на Flask передає системний промпт (інструкцію), який визначає роль ШІ як «експерта-консультанта з електрообладнання», та поточний контекст каталогу [9]. Це гарантує, що рекомендації будуть не лише лінгвістично правильними, а й технічно

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		28

обґрунтованими. Використання готової моделі через API забезпечує високу швидкість розробки, відсутність витрат на утримання потужних обчислювальних серверів та можливість миттєвого масштабування інтелектуальних функцій при розширенні асортименту підприємства. Таким чином, вибір моделі III забезпечує високу якість клієнтського досвіду, наближаючи взаємодію з вебдодатком до розмови з професійним живим консультантом.

Висновки до другого розділу

У другому розділі було проведено комплексне проєктування архітектури та внутрішньої логіки вебдодатка, що дозволило сформувати чіткий технічний план його реалізації. За результатами дослідження функціональних вимог було встановлено необхідність розмежування трьох рівнів доступу (гість, користувач, адміністратор) та автоматизації повного циклу обробки замовлень – від інтелектуального вибору товарів до генерації PDF-інвойсів із QR-кодами. Вибрана дворівнева клієнт-серверна архітектура на базі фреймворку Flask забезпечує стабільну взаємодію між адаптивним інтерфейсом користувача та серверною частиною, що обробляє бізнес-логіку.

Розроблена документоорієнтована модель бази даних у MongoDB, яка складається з колекцій products, users та orders, дозволяє ефективно працювати з неоднорідними технічними характеристиками електрообладнання та забезпечує цілісність історії покупок через механізм денормалізації даних. Аналіз потенціалу великих мовних моделей (LLM) довів їхню ефективність у роботі з природною мовою, зокрема для впровадження семантичного пошуку, що значно підвищує точність підбору продукції порівняно з традиційними методами. Таким чином, створений архітектурний проєкт є технологічно збалансованим, масштабованим та готовим до етапу безпосередньої програмної реалізації.

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		29

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБДОДАТКУ

3.1 Розроблення серверної логіки на Flask (Backend) та інтеграція з базою даних

Розроблення серверної частини (Backend) вебдодатка виконано мовою програмування Python з використанням мікрофреймворку Flask [16]. Основна перевага обраного підходу полягає у створенні модульної структури, де кожен маршрут (route) відповідає за конкретний бізнес-процес: від відображення каталогу до обробки адміністративних запитів.

Ініціалізація та підключення до бази даних. На початковому етапі розроблення було налаштовано зв'язок між сервером та NoSQL базою даних MongoDB за допомогою бібліотеки pymongo. Для забезпечення безпеки сесій користувачів було імплементовано секретний ключ шифрування. Фрагмент коду наведено на рисунку 3.1, а повний текст програмного коду представлено у додатку А.

```
electro_advisor > app.py > ...
1  import os
2  from flask import Flask, render_template, request, jsonify, session, send_file, redirect, url_for
3  from dotenv import load_dotenv
4  import google.generativeai as genai
5  from pymongo import MongoClient
6  from fpdf import FPDF
7  import io
8  import qrcode
9  from io import BytesIO
10 from werkzeug.security import generate_password_hash, check_password_hash
11 from datetime import datetime
12
13 load_dotenv()
14 app = Flask(__name__)
15 app.secret_key = "electrical_secret_key_123" # ключ для шифрування сесій
16 # Пароль для адмінки
17 ADMIN_PASSWORD = "yura_electro_2026123"
18
19 # Налаштування Gemini
20 genai.configure(api_key=os.getenv("GEMINI_API_KEY"))
21 model = genai.GenerativeModel('gemini-2.5-flash')
22
23 # налаштування MongoDB
24 client = MongoClient(os.getenv("MONGO_URI"))
25 db = client['electrical_db']
26 products_collection = db['products']
27 users_collection = db['users']
28 orders_collection = db['orders']
```

Рисунок 3.1 – Фрагмент вмісту файлу app.py

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		30

Реалізація логіки управління даними. Одним із технічно складних завдань стала інтеграція даних між MongoDB та Frontend-частиною. Оскільки MongoDB використовує специфічний тип даних ObjectId для ідентифікаторів, а стандарт JSON його не підтримує, було розроблено механізм серіалізації даних перед передачею в шаблон. Це критично важливо для коректної роботи адміністративної панелі та відображення історії замовлень.

```
electro_advisor > app.py > index
288
289 @app.route('/admin')
290 def admin_panel():
291     if session.get('role') != 'admin':
292         return redirect(url_for('index'))
293
294     all_prods = list(products_collection.find())
295     all_users = list(users_collection.find({}, {"password": 0}))
296
297     # Отримуємо замовлення
298     raw_orders = list(orders_collection.find().sort("date", -1))
299
300     serializable_orders = []
301     for order in raw_orders:
302         order['_id'] = str(order['_id'])
303         if 'date' in order:
304             order['date'] = order['date'].isoformat()
305         serializable_orders.append(order)
306
307     categories = sorted(list(set(p.get('category', 'Інше') for p in all_prods)))
308
309     return render_template('admin.html',
310                           products=all_prods,
311                           all_users=all_users,
312                           orders=serializable_orders,
313                           categories=categories)
314
```

Рисунок 3.2 – Фрагмент вмісту файлу app.py

Модуль безпеки та автентифікації. Для захисту системи реалізовано механізм хешування паролів за допомогою бібліотеки werkzeug.security. Логіка автентифікації базується на перевірці ролі користувача, що зберігається в об'єкті session. Це дозволяє серверу ідентифікувати права доступу при кожному запиті до захищених маршрутів, таких як оновлення цін або видалення товарів.

Адміністративна логіка. Серверна частина містить набір методів для управління складом. Кожна операція (додавання товару, зміна залишків) супроводжується валідацією прав адміністратора. Наприклад, оновлення

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		31

параметрів товару реалізовано через метод POST, який взаємодіє з колекцією products_collection за допомогою оператора \$set.

```
electro_advisor > app.py > admin_panel
322
323 @app.route('/admin/update', methods=['POST'])
324 def admin_update():
325     if session.get('role') != 'admin':
326         return "Access denied", 403
327
328     name = request.form.get('name')
329     price = float(request.form.get('price'))
330     stock = int(request.form.get('stock'))
331
332     products_collection.update_one(
333         {"name": name},
334         {"$set": {"price": price, "stock": stock}}
335     )
336     return redirect(url_for('admin_panel'))
337
```

Рисунок 3.3 – Фрагмент вмісту файлу app.py

Таким чином, розроблений бекенд забезпечує надійну обробку даних, повну інтеграцію з базою даних та гнучкість для підключення модуля штучного інтелекту, що робить систему цілісною та працездатною.

3.2 Реалізація інтелектуального пошуку та системи рекомендацій за допомогою ШІ (Gemini/OpenAI API)

Інтеграція штучного інтелекту в розроблений вебдодаток є його ключовою інноваційною перевагою, що трансформує стандартний пошук у повноцінну систему підтримки прийняття рішень. Технічна реалізація цього модуля базується на взаємодії серверної частини Flask із зовнішнім API моделі Gemini (або OpenAI), використовуючи підхід контекстного навчання.

Алгоритм роботи інтелектуального модуля. Процес формування рекомендації розділений на три етапи:

1. Сервер витягує актуальний перелік товарів із MongoDB;

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		32

2. Створюється складна інструкція для ШІ, що містить запит користувача та дані про наявну продукцію;
3. ШІ аналізує семантику запиту та повертає технічно обґрунтовану пораду.

Основним інструментом реалізації маршруту /ask є функція, що приймає POST-запит із фронтенду. Нижче наведено фрагмент коду, що демонструє логіку формування запиту до моделі:

```
electro_advisor > app.py > ask
72 @app.route('/ask', methods=['POST'])
73 def ask():
74     user_message = request.json.get("message").strip()
75
76     # ОБРОБКА КОМАНДИ /all
77     if user_message.lower() == '/all':
78         all_products = list(products_collection.find({}, {"_id": 0}))
79         return jsonify({"type": "table", "data": all_products})
80
81     # ЗАВАНТАЖЕННЯ ІСТОРИЇ ТА КОНТЕКСТУ
82     # отримуємо історію з сесії або створюємо порожню
83     history = session.get('chat_history', [])
84
85     all_products = list(products_collection.find({}, {"_id": 0}))
86     products_context = f"Ти консультант магазину електротехніки. Наш асортимент: {str(all_products)}. " \
87         f"Якщо користувач хоче купити товар, нагадай натиснути кнопку кошика 🛒 таблиці /all."
88
89     # ЗАПУСК ЧАТУ
90     # Перетворюємо історію з сесії назад у формат, який розуміє Gemini
91     chat = model.start_chat(history=history)
92
93     try:
94         # додаємо системну інструкцію до першого повідомлення, якщо історія порожня
95         full_prompt = user_message
96         if not history:
97             full_prompt = f"{products_context}\n\nКористувач запитує: {user_message}"
98
99         response = chat.send_message(full_prompt)
100
101         # ЗБЕРЕЖЕННЯ ІСТОРИЇ (Конвертація в JSON-формат)
102         new_history = []
103         for content in chat.history:
104             new_history.append({
105                 "role": content.role,
106                 "parts": [{"text": part.text} for part in content.parts]
107             })
108
109         # Зберігаємо тільки останні 10 повідомлень, щоб не переповнювати сесію
110         session['chat_history'] = new_history[-10:]
111         session.modified = True
```

Рисунок 3.4 – Фрагмент вмісту файлу app.py

```
112
113     return jsonify({"type": "text", "reply": response.text})
114
115 except Exception as e:
116     print(f"Помилка Gemini: {e}")
117     return jsonify({"type": "text", "reply": "Вибачте, сталася технічна помилка. Спробуйте ще раз."}), 500
118
```

Рисунок 3.5 – Фрагмент вмісту файлу app.py

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		33

На відміну від стандартних SQL-запитів, реалізований модуль використовує семантичну відповідність. Коли користувач вводить неточний технічний запит, наприклад, «потрібен захист від короткого замикання», ШІ-модель проводить асоціативний аналіз:

- аналізує опис товарів у базі (поле description);
- визначає, що автоматичні вимикачі відповідають цій функції;
- формує відповідь, вказуючи конкретні моделі, наявні на складі, та їхні ціни.

Для покращення користувацького досвіду на клієнтській стороні реалізовано асинхронне звернення до API через fetch у JavaScript. Це дозволяє уникнути перезавантаження сторінки під час очікування відповіді від ШІ. Такий підхід забезпечує інтерактивність та імітує реальний чат із технічним фахівцем, що значно підвищує якість обслуговування в інтернет-магазині електротехніки.

3.3 Розроблення модуля управління користувачами, ролями та безпекою (хешування паролів, сесії)

Модуль управління користувачами та забезпечення безпеки є фундаментом захищеності персональних даних та розмежування рівнів доступу в розробленому додатку. Програмна реалізація цього блоку базується на інтеграції механізмів автентифікації Flask із криптографічними інструментами бібліотеки Werkzeug [22].

```

electro_advisor > app.py > ask
9  from io import BytesIO
10 from werkzeug.security import generate_password_hash, check_password_hash
11 from datetime import datetime
12
13 load_dotenv()

```

Рисунок 3.6 – Фрагмент вмісту файлу app.py

Основним елементом безпеки є політика відмови від зберігання паролів у відкритому вигляді. При реєстрації нового користувача застосовується функція `generate_password_hash`, яка перетворює введений пароль у незворотний хеш із додаванням унікальної «солі». Це гарантує, що навіть у випадку порушення конфіденційності та отримання неправомірного доступу до середовища MongoDB, зломисник не зможе відновити реальні паролі клієнтів. Під час входу в систему додаток використовує функцію `check_password_hash`, яка порівнює введений користувачем рядок із хешем, що зберігається в колекції `users`.

Управління станом входу реалізовано через механізм серверних сесій (`session`), які працюють на основі криптографічно підписаних файлів `cookie`. Після успішної перевірки облікових даних сервер записує в сесію ідентифікатор користувача (`username`) та його рівень доступу (`role`). Це дозволяє системі «впізнавати» клієнта при переході між сторінками без повторного введення пароля. Рольова модель передбачає поділ на звичайних користувачів (`user`) та адміністраторів (`admin`).

Особлива увага приділена захисту адміністративної панелі та критичних функцій (видалення товарів, зміна цін). На рівні бекенд-логіки кожен захищений маршрут містить перевірку: `if session.get('role') != 'admin': return redirect(url_for('index'))`.

```
315 @app.route('/admin/delete/<name>', methods=['POST'])
316 def admin_delete(name):
317     if session.get('role') != 'admin':
318         return redirect(url_for('index'))
319
320     products_collection.delete_one({"name": name})
321     return redirect(url_for('admin_panel'))
322
```

Рисунок 3.7 – Фрагмент вмісту файлу `app.py`

Такий підхід унеможливлює доступ до функцій керування складом через пряме введення URL-адреси неавторизованими особами. Додатково реалізовано функцію `logout`, яка повністю очищує об'єкт сесії, забезпечуючи

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		35

безпечне завершення роботи з додатком, що є критично важливим для корпоративного використання системи на підприємстві.

3.4 Створення адміністративної панелі для моніторингу складу та історії замовлень

Центральним елементом управління вебдодатком є адміністративна панель, яка реалізована як захищений інтерфейс для повного контролю за товарними запасами та активністю клієнтів. Програмна логіка панелі базується на маршруті /admin, який виконує агрегацію даних із трьох колекцій MongoDB. Для забезпечення актуальності інформації при кожному завантаженні сторінки сервер виконує запити до бази даних, отримуючи повний перелік продукції, список зареєстрованих користувачів та архів замовлень, відсортований за часом від найновіших до найстаріших (див. рисунок 3.2).

Управління складом реалізовано через систему CRUD-операцій, що дозволяють адміністратору миттєво змінювати параметри товарів без прямого втручання в базу даних. У кожному рядку таблиці товарів інтегровано форми для редагування ціни та залишків, які використовують метод POST для передачі даних на сервер. Програмна обробка цих запитів використовує оператор \$set у MongoDB для точкового оновлення документів. Модуль додавання нових позицій підтримує інтелектуальні підказки категорій через елемент datalist, що забезпечує однотипність даних у базі, яка згодом аналізується ШІ-асистентом, як це показано на рисунку 3.3 та 3.8.

```
26 <div class="col-md-2">
27   <input type="text" name="category" list="categoryList" class="form-control" placeholder="Категорія" required>
28   <datalist id="categoryList">
29     {% for cat in categories %}
30       <option value="{{ cat }}">
31         {% endfor %}
32     </datalist>
33   </div>
34 </div class="col-md-2">
```

Рисунок 3.8 – Фрагмент вмісту файлу admin.html

Особливе технічне рішення впроваджене в модулі моніторингу історії замовлень. Оскільки дані про транзакції передаються з Python-бекенду в

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		36

JavaScript-середовище для відображення у модальних вікнах, було реалізовано спеціальний механізм попередньої обробки об'єктів. Через несумісність стандартного формату JSON із типами даних MongoDB, серверна частина виконує примусову серіалізацію ідентифікаторів `_id` у рядковий формат та конвертацію дат у стандарт ISO (див. рисунок 3.2). Це дозволяє адміністратору переглядати детальну історію кожного користувача — склад кошика, ціни на момент покупки та загальну суму – без перезавантаження сторінки, використовуючи клієнтську фільтрацію масиву замовлень (див. рисунок 3.9).

```
electro_advisor > templates > admin.html > html > body.bg-light > script > showHistory > userOrders.forEach() callback
2   <html lang="uk">
9   <body class="bg-light">
131
132   <script>
133     // передаємо дані замовлень з Python в JS
134     const allOrders = {{ orders|tojson|safe }};
135
136     function showHistory(username) {
137       document.getElementById('historyUser').innerText = username;
138       const body = document.getElementById('historyBody');
139
140       // фільтруємо замовлення тільки для цього користувача
141       const userOrders = allOrders.filter(o => o.username === username);
142
```

Рисунок 3.9 – Фрагмент вмісту файлу admin.html

Для контролю за клієнтською базою в панелі реалізовано блок моніторингу користувачів, де відображаються імена, дати реєстрації та поточні ролі. Кожен запис супроводжується кнопкою швидкого доступу до персональної історії замовлень, що дозволяє адміністратору аналізувати купівельну спроможність та вподобання клієнтів. Такий підхід до побудови адмін-панелі забезпечує високу оперативність управління бізнес-процесами підприємства, поєднуючи простоту інтерфейсу на основі Bootstrap із потужним функціоналом NoSQL-бази даних, як це показано на рисунку 3.10.

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		37

```

89     <table class="table table-sm">
90     <thead>
91     <tr>
92     <th>Користувач</th>
93     <th>Роль</th>
94     <th>Дія</th>
95     </tr>
96     </thead>
97     <tbody>
98     {% for u in all_users %}
99     <tr>
100    <td>{{ u.username }}</td>
101    <td>
102        <span class="badge {{ 'bg-danger' if u.role == 'admin' else 'bg-info text-dark' }}">
103            {{ u.role }}
104        </span>
105    </td>
106    <td>
107        <button class="btn btn-sm btn-outline-primary" onclick="showHistory('{{ u.username }}')">
108            <i class="bi bi-clock-history"></i> Історія
109        </button>
110    </td>
111    </tr>
112    {% endfor %}
113    </tbody>
114    </table>

```

Рисунок 3.10 – Фрагмент вмісту файлу admin.html

3.5 Реалізація клієнтського інтерфейсу (Frontend) та модуля генерації електронних інвойсів

Клієнтський інтерфейс додатка розроблений за принципами адаптивного дизайну з використанням фреймворку Bootstrap 5, що забезпечує коректне відображення на пристроях із різною діагоналлю екрана. Візуальна частина розділена на функціональні зони: навігаційну панель із доступом до профілю, сітку товарів із динамічними картками та інтерактивне вікно ШІ-консультанта. Основна взаємодія на стороні клієнта реалізована за допомогою мови JavaScript, яка забезпечує роботу кошика та асинхронне спілкування із сервером через Fetch API. Це дозволяє користувачеві додавати товари до замовлення та отримувати консультації від штучного інтелекту без повного перезавантаження сторінки, що значно покращує користувацький досвід (UX), як це показано на рисунку 3.11.

```

223 <script>
224   async function sendMessage() {
225     const input = document.getElementById('user-input');
226     const chat = document.getElementById('chat-window');
227     if (!input.value) return;
228     const msg = input.value;
229     chat.innerHTML += `<div class="user-msg">${msg}</div>`;
230     input.value = '';
231
232     const response = await fetch('/ask', {
233       method: 'POST',
234       headers: { 'Content-Type': 'application/json' },
235       body: JSON.stringify({ message: msg })
236     });
237     const data = await response.json();
238     if (data.type === "table") { renderTable(data.data); }
239     else { chat.innerHTML += `<div class="bot-msg">${data.reply}</div>`; }
240     chat.scrollTop = chat.scrollHeight;
241   }
242

```

Рисунок 3.11 – Фрагмент вмісту файлу index.html

Логіка формування замовлення базується на синхронізації фронтенд-даних із серверними сесіями Flask. Коли користувач ініціює оформлення покупки, система збирає ідентифікатори товарів із кошика, перевіряє їх наявність у MongoDB та розраховує підсумкову вартість. Після підтвердження транзакції запускається програмний модуль генерації електронних інвойсів, побудований на базі бібліотеки FPDF [4]. Процес генерації відбувається на сервері в режимі реального часу: дані про покупця, перелік артикулів та технічні параметри обладнання динамічно вписуються в заздалегідь підготовлений шаблон документа формату A4 (див. рисунок 3.12-3.13).

```

165 </div>
166 <div class="modal-footer">
167   <a href="/generate_invoice" class="btn btn-success">
168     <i class="bi bi-file-earmark-pdf"></i> Завантажити рахунок (PDF)
169   </a>
170 </div>
171 </div>

```

Рисунок 3.12 – Фрагмент вмісту файлу index.html

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		39

```

208 pdf = FPDF()
209 pdf.add_page()
210
211 try:
212     pdf.add_font('DejaVu', '', 'DejaVuSans.ttf', uni=True)
213     pdf.set_font('DejaVu', '', 14)
214 except:
215     pdf.set_font('Arial', '', 14)
216
217 # Заголовок
218 pdf.cell(200, 10, txt="РАХУНОК-ФАКТУРА: Electro-Tech Solutions", ln=True, align='C')
219 pdf.set_font('DejaVu', '', 10)
220 pdf.cell(200, 7, txt="Власник: Горбань Юрій | м. Івано-Франківськ", ln=True, align='C')
221 pdf.ln(10)
222
223 # Шапка таблиці
224 pdf.cell(80, 10, "Назва товару", border=1)
225 pdf.cell(30, 10, "К-сть", border=1)
226 pdf.cell(40, 10, "Ціна за од.", border=1)
227 pdf.cell(40, 10, "Сума", border=1)
228 pdf.ln()
229
230 # Наповнення таблиці та розрахунок суми
231 total_sum = 0
232 for item in products:
233     name = item['name']
234     price = item['price']
235     qty = cart.get(name, 0)
236     item_total = price * qty
237
238     pdf.cell(80, 10, name, border=1)
239     pdf.cell(30, 10, f"{qty} од.", border=1)
240     pdf.cell(40, 10, f"{price} грн", border=1)
241     pdf.cell(40, 10, f"{item_total} грн", border=1)
242     pdf.ln()
243     total_sum += item_total
244
245 pdf.ln(5)
246 pdf.set_font('DejaVu', '', 14)
247 pdf.cell(190, 10, txt=f"ЗАГАЛЬНА СУМА: {total_sum} грн", ln=True, align='R')

```

Рисунок 3.13 – Фрагмент вмісту файлу app.py

Ключовою технічною особливістю модуля інвойсів є автоматична інтеграція QR-коду для швидкої оплати. За допомогою бібліотеки qrcode система генерує зображення, що містить реквізити замовлення та суму до сплати, і вбудовує його безпосередньо в тіло PDF-файлу. Готовий документ зберігається в буфері пам'яті сервера та миттєво передається в браузер користувача як потік байтів, що дозволяє завантажити офіційний рахунок-фактуру одразу після натискання кнопки «Оплатити». Такий підхід повністю автоматизує паперову рутину торгового процесу, поєднуючи зручність вебінтерфейсу з юридичною чіткістю електронного документообігу (див. рисунок 3.14).

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		40

```

249 # ГЕНЕРАЦІЯ QR-КОДУ
250 payment_info = f"Оплата замовлення. Отримувач: Горбань Юрій. Сума: {total_sum} грн. Дякуємо!"
251 qr = qrcode.QRCode(version=1, box_size=10, border=5)
252 qr.add_data(payment_info)
253 qr.make(fit=True)
254 qr_img = qr.make_image(fill_color="black", back_color="white")
255
256 # Зберігаємо QR в тимчасовий буфер для картинки
257 qr_buffer = BytesIO()
258 qr_img.save(qr_buffer, format='PNG')
259 qr_buffer.seek(0)
260
261 # Вставляємо QR в PDF
262 pdf.ln(10)
263 pdf.set_font('DejaVu', '', 10)
264 pdf.cell(0, 10, "Відскануйте для оплати  отримання реквізитів:", ln=True, align='C')
265 # Малюємо QR-код по центру (x=80)
266 pdf.image(qr_buffer, x=80, y=pdf.get_y(), w=50)
267
268 # ФІНАЛІЗАЦІЯ: Збереження всього PDF в один буфер
269 output = io.BytesIO()
270 # використовуємо .encode('latin-1','replace') для сумісності fpdf
271 pdf_content = pdf.output(dest='S')
272 if isinstance(pdf_content, str):
273     pdf_content = pdf_content.encode('latin-1')
274
275 output.write(pdf_content)
276 output.seek(0)
277
278 session['cart'] = {}
279 session.modified = True
280
281 return send_file(output, download_name="invoice_Gorban_QR.pdf", as_attachment=True)
282

```

Рисунок 3.14 – Фрагмент вмісту файлу index.html

Висновки до третього розділу

У третьому розділі було успішно виконано програмну реалізацію всіх компонентів інтелектуального вебдодатка для автоматизації торгівлі електротехнічною продукцією. Завдяки використанню мови Python та мікрофреймворку Flask вдалося створити гнучку та продуктивну серверну архітектуру, що забезпечує стабільну взаємодію з NoSQL-базою даних MongoDB. Реалізовані механізми серіалізації даних дозволили ефективно опрацьовувати складні структури документів та коректно відображати їх у користувацькому інтерфейсі, нівелюючи обмеження стандартних форматів обміну даними.

Ключовим досягненням розробки стала успішна інтеграція великої мовної моделі Gemini через API, що дозволило впровадити функціонал семантичного пошуку та рекомендацій. Це рішення трансформувало процес

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		41

підбору обладнання з простого фільтрування за ключовими словами у формат інтелектуального консультування, що здатне інтерпретувати технічний контекст запитів користувачів. Паралельно з цим було забезпечено високий рівень безпеки системи шляхом незворотного хешування паролів та впровадження рольової моделі доступу на основі зашифрованих серверних сесій.

Розроблена адміністративна панель та адаптивний клієнтський інтерфейс на базі Bootstrap 5 сформували цілісне середовище для ефективного управління бізнес-процесами підприємства. Автоматизація завершального етапу торгівлі – генерації електронних PDF-інвойсів із вбудованими QR-кодами – продемонструвала можливість повної цифровізації документообігу. Таким чином, створений програмний продукт є технічно завершеним рішенням, яке поєднує інноваційні методи штучного інтелекту з надійними інструментами автоматизації, що підтверджує готовність системи до експлуатації та підсумкового тестування.

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		42

4 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ

4.1 Методика та стратегія тестування вебдодатку

Методика тестування розробленого вебдодатка базується на комплексному підході, що поєднує функціональне тестування (Black Box testing) та перевірку системної інтеграції всіх компонентів: від бази даних до модуля штучного інтелекту. Основною стратегією обрано сценарійне тестування (End-to-End), яке імітує реальний шлях користувача: від моменту авторизації та використання інтелектуального пошуку до фінальної генерації PDF-інвойсу. Основним інструментарієм для проведення перевірки є браузер Firefox (див. рисунок 4.1), який виконує роль не тільки графічної оболонки, а й виступає як діагностичний інструмент завдяки вбудованим засобам розробника (Firefox DevTools), як це показано на рисунку 4.2. Це дозволяє в реальному часі відстежувати час виконання запитів до серверної частини Flask та аналізувати структуру JSON-відповідей від моделі Gemini.

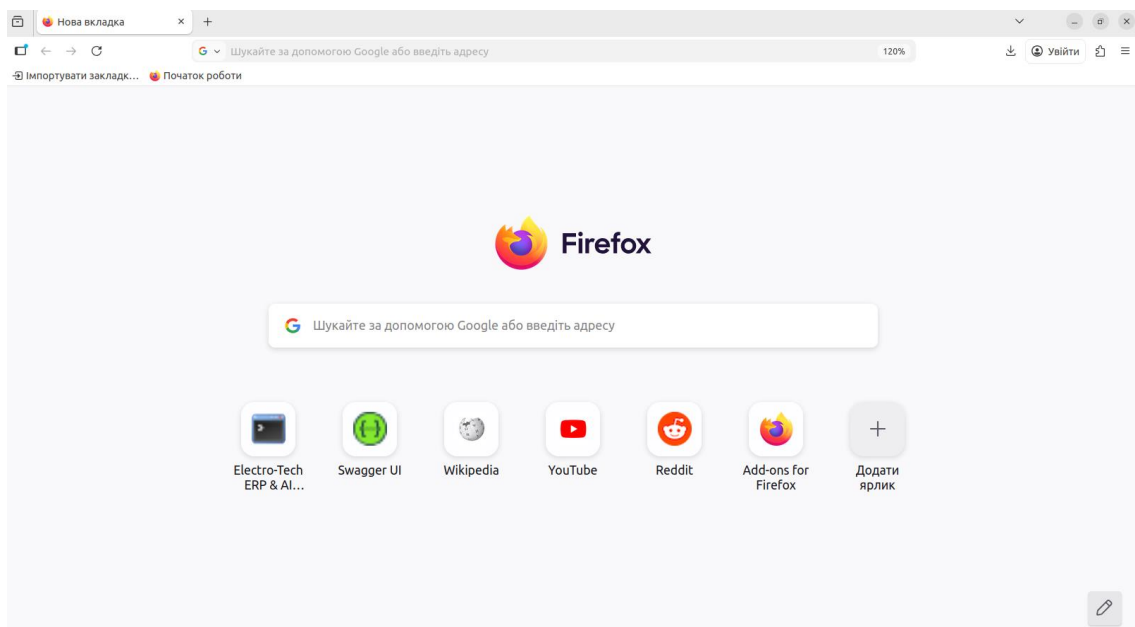


Рисунок 4.1 – Вікно браузера Firefox

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		43

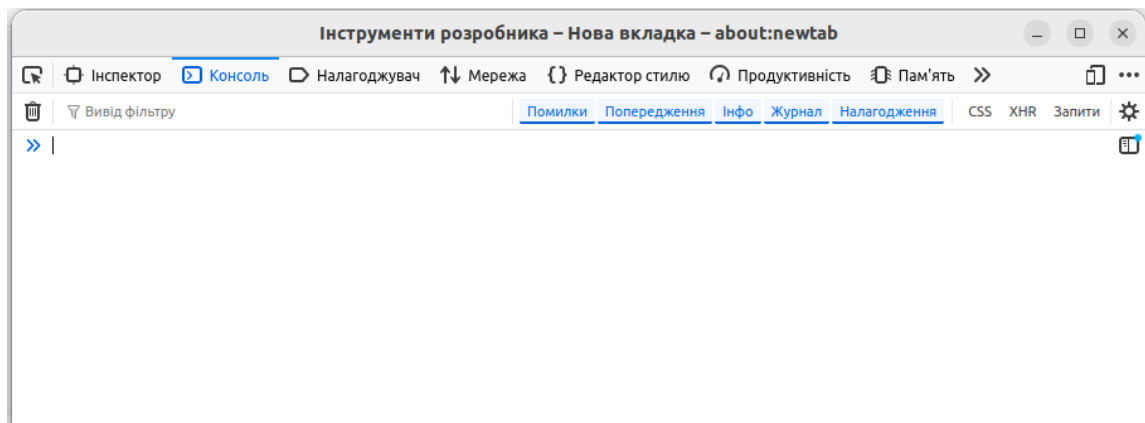


Рисунок 4.2 – Інструменти розробника в браузері Firefox (Firefox DevTools)

Процес тестування розділений на кілька критичних етапів, кожен з яких фокусується на окремому аспекті системи. На рівні функціонального тестування перевіряється коректність роботи модуля безпеки: надійність хешування паролів та стійкість сесій при спробі несанкціонованого доступу до адмін-панелі. Особлива увага приділяється валідації вводу даних при оновленні складу в MongoDB, щоб виключити помилки типів даних. Для оцінки ефективності інтелектуального пошуку застосовується методика порівняльного аналізу: подаються запити різного ступеня складності (прямі, технічні та непрямі описові), після чого оцінюється релевантність відповіді ШІ та швидкість її формування, що фіксується через вкладку «Мережа» (Network) у браузері Firefox.

Стратегія тестування також включає перевірку крос-браузерної адаптивності та коректності відображення графічних елементів Bootstrap 5. Окремим пунктом методики є стрес-тестування модуля генерації інвойсів, де перевіряється правильність вбудовування QR-кодів та читабельність тексту в генерованих PDF-файлах при різних обсягах замовлень. Використання консолі розробника Firefox дозволяє оперативно виявляти потенційні помилки в JavaScript-кодi, що відповідає за асинхронне додавання товарів у кошик. Такий системний підхід гарантує, що всі програмні модулі працюють як єдине ціле, а технічні рішення, закладені на етапі проєктування, повністю реалізовані та відповідають поставленим вимогам.

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		44

4.2 Функціональне тестування основних модулів (вхід, кошик, адмін-панель)

Для запуску додатку виконали команду “python3 app.py” в терміналі Linux (див. рисунок 4.4), попередньо створивши віртуальне середовище за допомогою команди “python3 -m venv venv” та активувавши його за допомогою команди “source venv/bin/activate” (див. рисунок 4.3).

```
Обробка тригерів gnome-menus (3.36.0-1ubuntu3)...\ngorban@gorban-VirtualBox:~/electro_advisor$ python3 -m venv venv\ngorban@gorban-VirtualBox:~/electro_advisor$ source venv/bin/activate
```

Рисунок 4.3 – Вікно терміналу Linux

```
(venv) gorban@gorban-VirtualBox:~/electro_advisor$ python3 app.py\n/home/gorban/electro_advisor/venv/lib/python3.10/site-packages/google/api_core/_\npython_version_support.py:275: FutureWarning: You are using a Python version (3.\n10.12) which Google will stop supporting in new releases of google.api_core once\nit reaches its end of life (2026-10-04). Please upgrade to the latest Python ve\nrsion, or at least Python 3.11, to continue receiving updates for google.api_cor\ne past that date.\n  warnings.warn(message, FutureWarning)\n/home/gorban/electro_advisor/app.py:4: FutureWarning:\n\nAll support for the `google.generativeai` package has ended. It will no longer b\n e receiving\nupdates or bug fixes. Please switch to the `google.genai` package as soon as pos\nsible.\nSee README for more details:\n\nhttps://github.com/google-gemini/deprecated-generative-ai-python/blob/main/READM\nE.md\n\n  import google.generativeai as genai\n  * Serving Flask app 'app'\n  * Debug mode: on\nWARNING: This is a development server. Do not use it in a production deployment.\nUse a production WSGI server instead.\n  * Running on all addresses (0.0.0.0)\n  * Running on http://127.0.0.1:5000\n  * Running on http://10.0.2.15:5000\nPress CTRL+C to quit\n  * Restarting with stat
```

Рисунок 4.4 – Вікно терміналу Linux

Та, запустивши браузер Firefox, перейшли за адресою “http://127.0.0.1:5000/” для відкриття головної сторінки нашого додатку (див. рисунок 4.5).

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		45

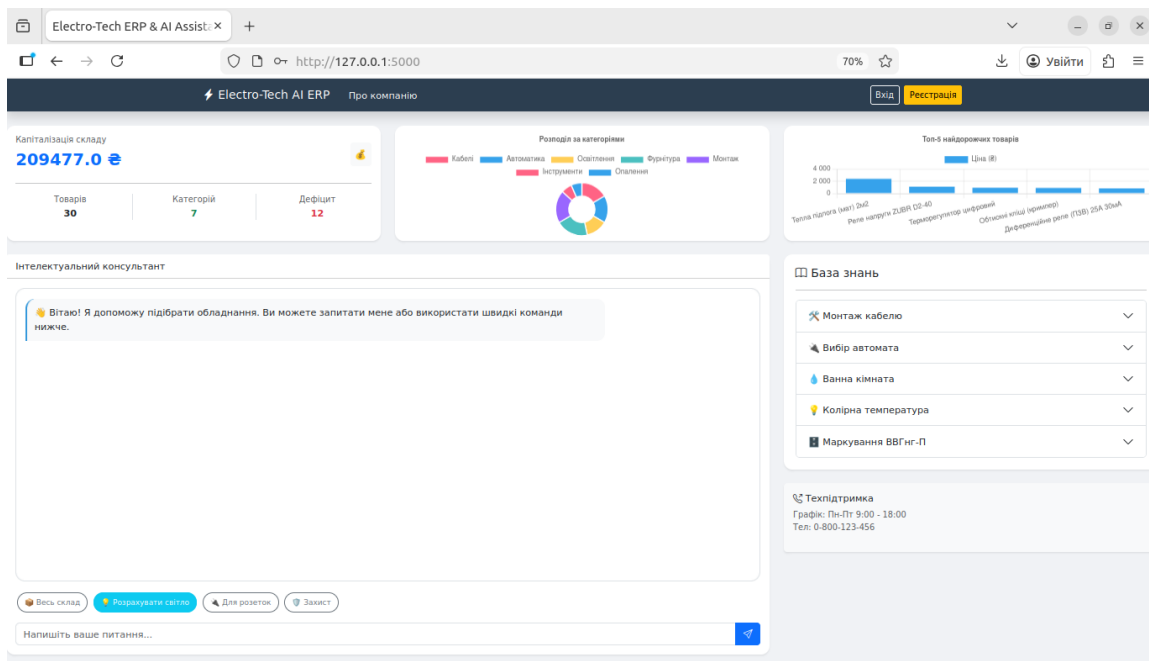


Рисунок 4.5 – Основне вікно додатку в браузері Firefox

Так як зараз ми фактично в статусі “Гість”, спробували протестувати основні функції додатку, як це показано на рисунку 4.6-4.8.

Тестування функцій в статусі “Гість”. Спочатку натиснули на “Весь склад” (функціонально команда /all) для перегляду повного списку доступних товарів (див. рисунок 4.6).

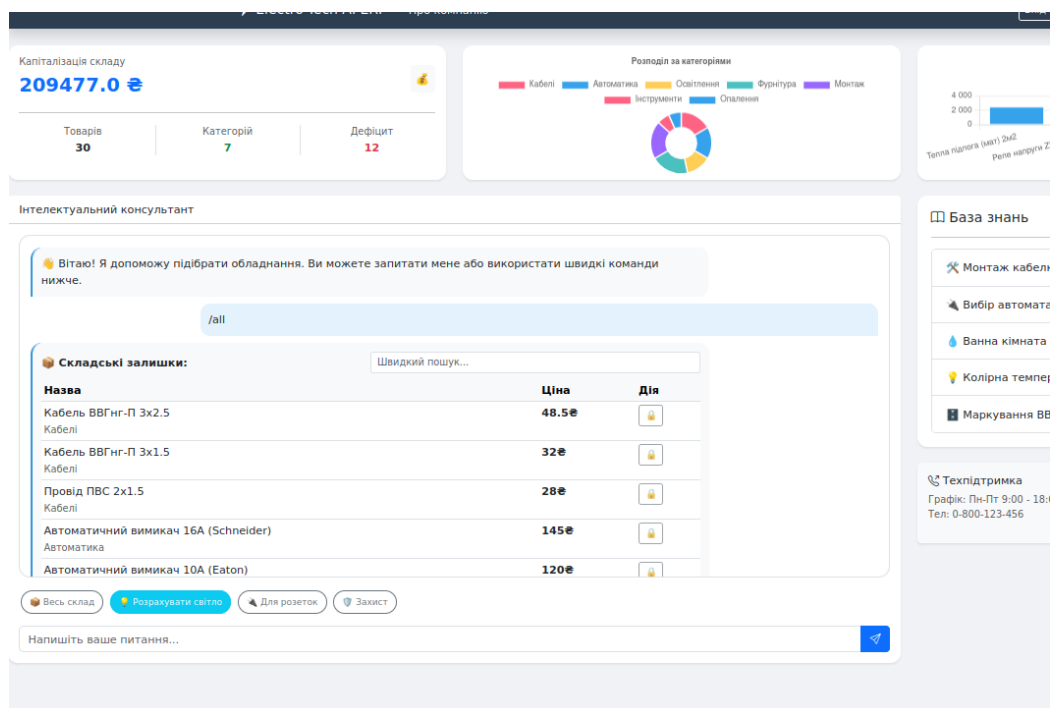


Рисунок 4.6 – Основне вікно додатку в браузері Firefox

Як бачимо, біля кожного товару графічний знак замка, через який ми не можемо додати товар до кошику і в подальшому оформити замовлення, адже зараз ми в статусі “Гість”.

Спробували ще поспілкуватись з інтелектуальним консультантом в чаті та переконались що це можливо для гостей (див. рисунок 4.7).

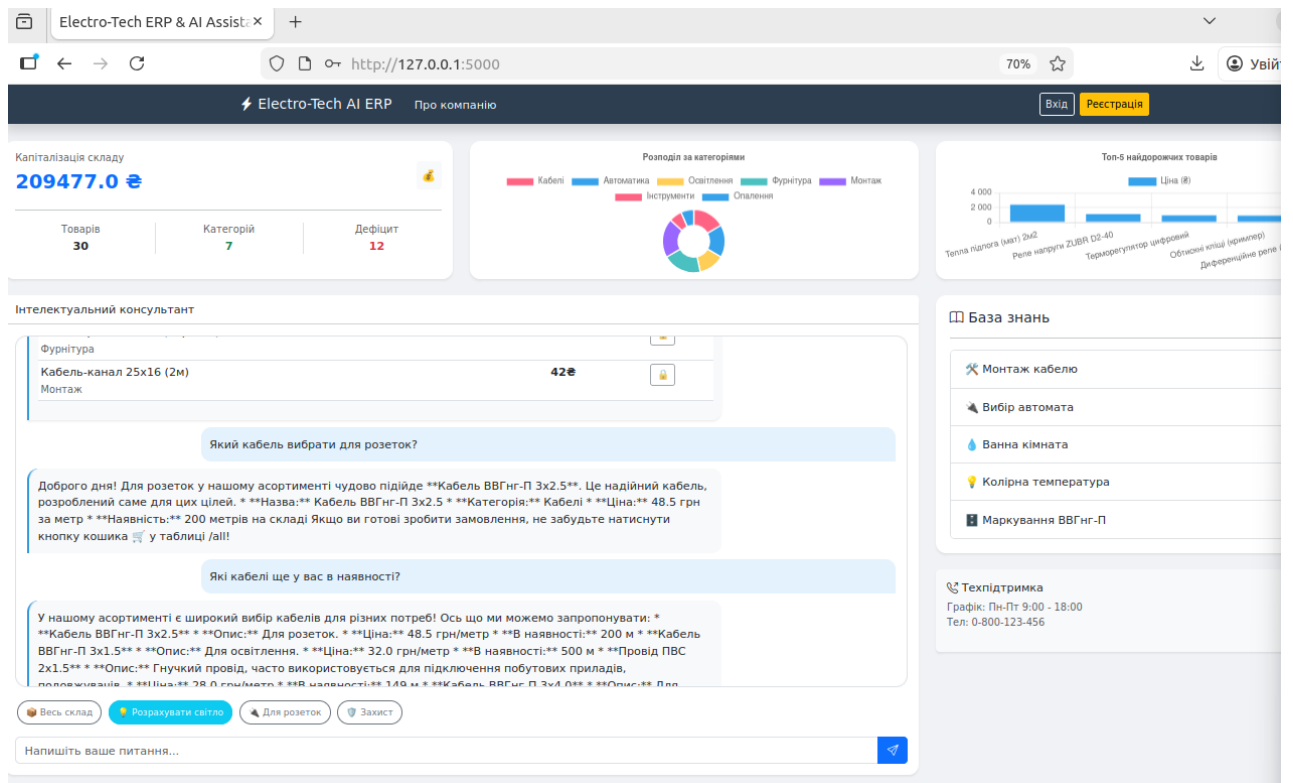


Рисунок 4.7 – Основне вікно додатку в браузері Firefox

Натиснули на кнопку “Про компанію” для перевірки доступу до неї в статусі “Гість”, як це показано на рисунку 4.8.

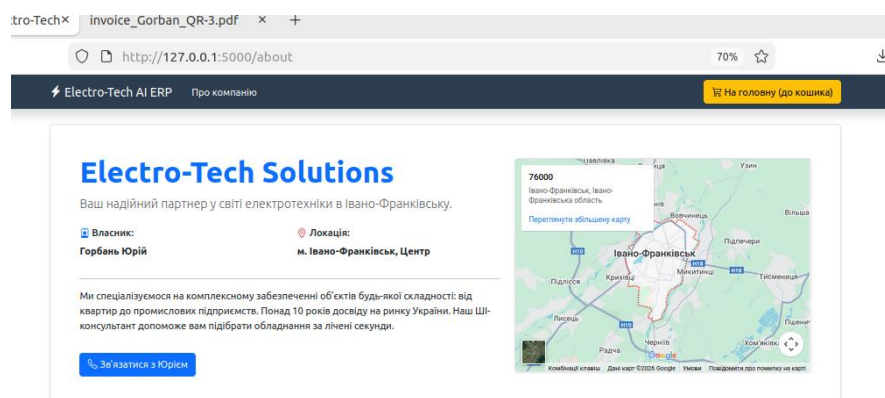


Рисунок 4.8 – Вікно сторінки “Про компанію” додатку в браузері Firefox

Як бачимо, тут зображена вся інформація про нашу фірму і ми маємо можливість її дізнатись, перебуваючи в статусі “Гість”.

Далі приступили до тестування функціональних можливостей статусу “Користувач”. Спочатку протестували функцію реєстрації.

Зареєструвались в системі під логіном “Yurii” та паролем “Yurii123”, натиснувши кнопку “Реєстрація” в правому верхньому куті сторінки (див. рисунок 4.9-4.10).

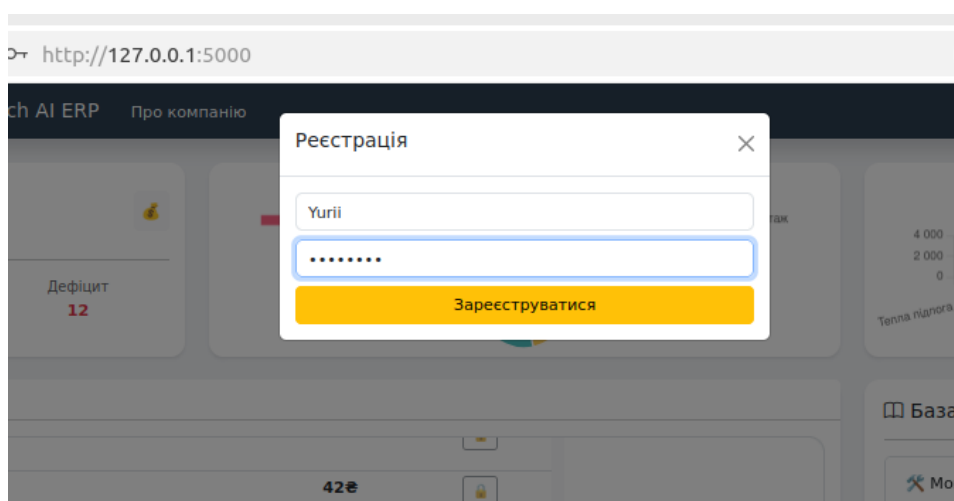


Рисунок 4.9 – Основне вікно додатку в браузері Firefox

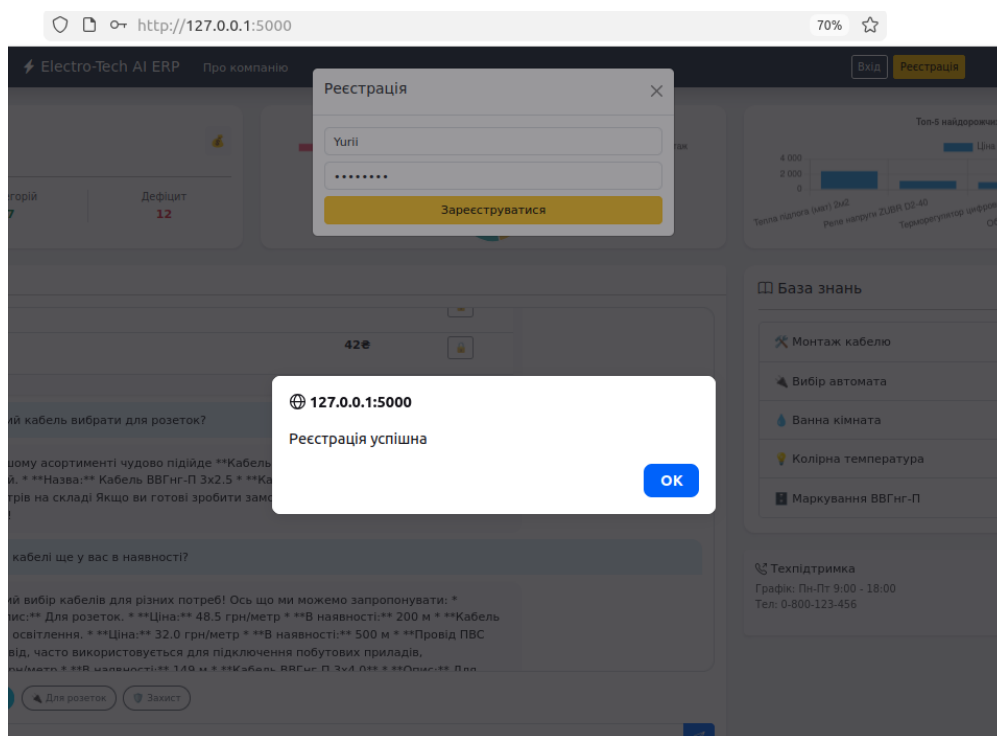


Рисунок 4.10 – Основне вікно додатку в браузері Firefox

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
						48
Зм.	Арк.	№ докум.	Підпис	Дата		

Отримали повідомлення про успішну реєстрацію. Далі натиснули на кнопку “Вхід” та ввели дані зареєстрованого користувача (див. рисунок 4.11-4.13).

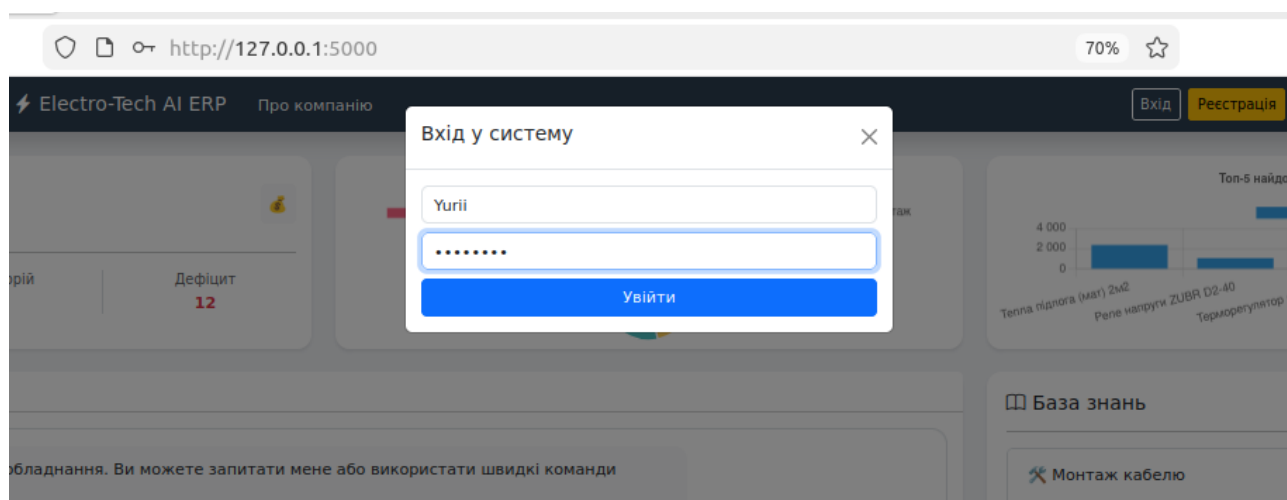


Рисунок 4.11 – Основне вікно додатку в браузері Firefox

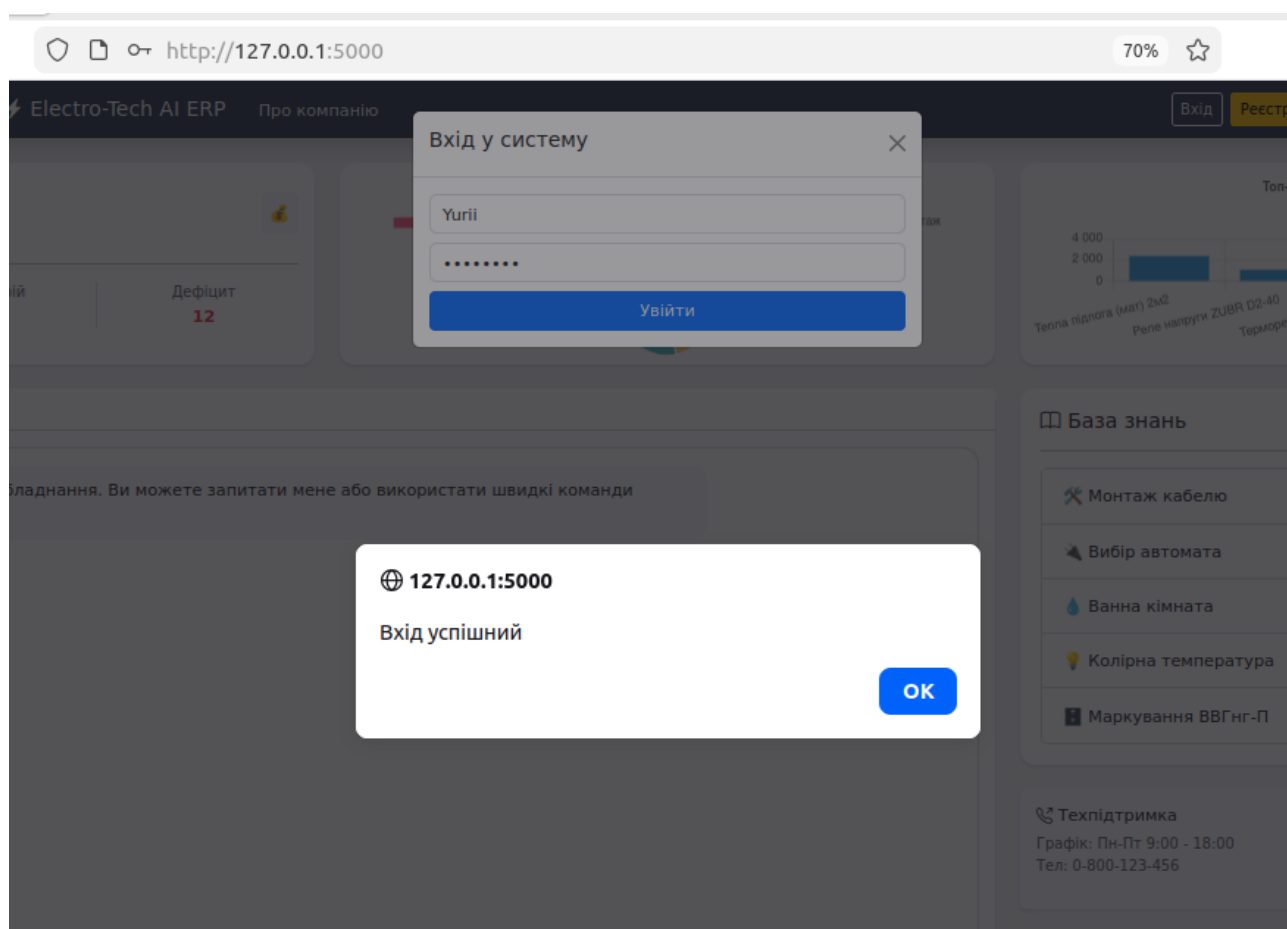


Рисунок 4.12 – Основне вікно додатку в браузері Firefox

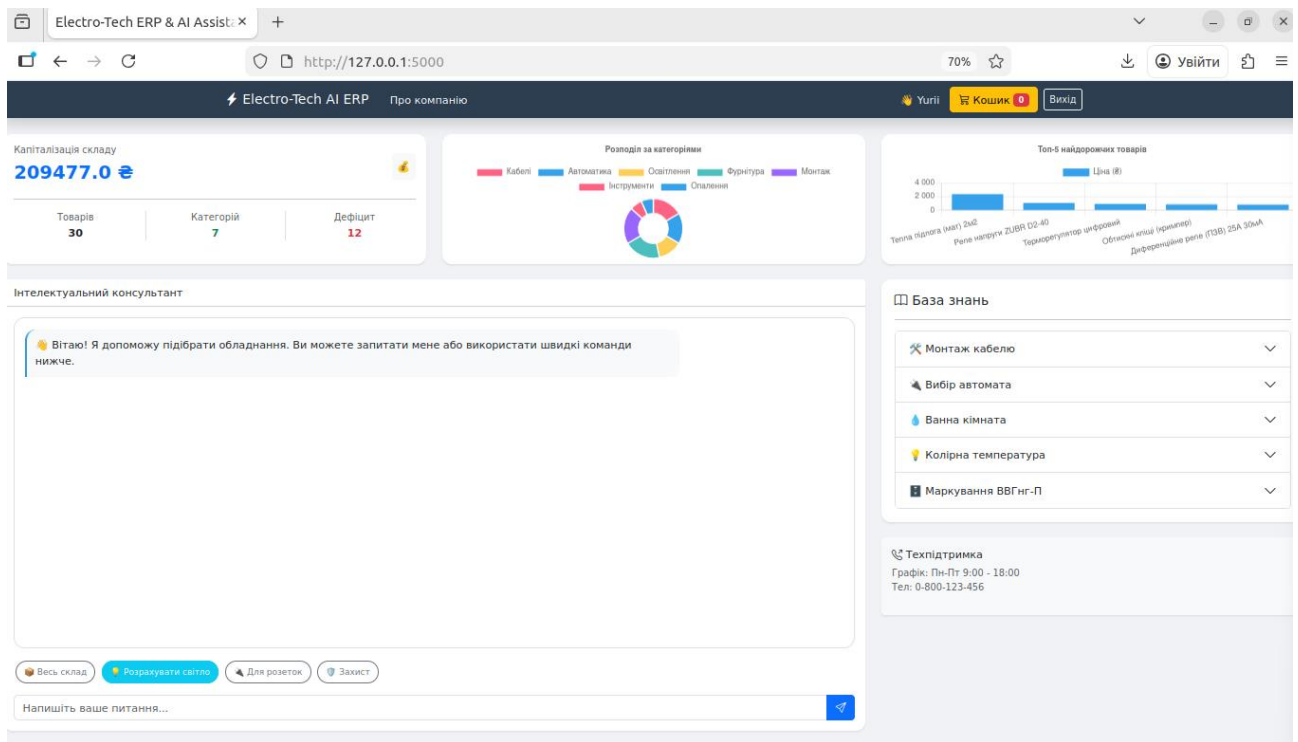


Рисунок 4.13 – Основне вікно додатку в браузері Firefox

Результати тестування модуля автентифікації та розмежування ролей у браузері Firefox підтвердили коректність роботи системи безпеки. При введенні правильних облікових даних сервер Flask успішно створює сесію, що було зафіксовано у вкладці «Сховище» (Storage) інструментів розробника (див. рисунок 4.14).

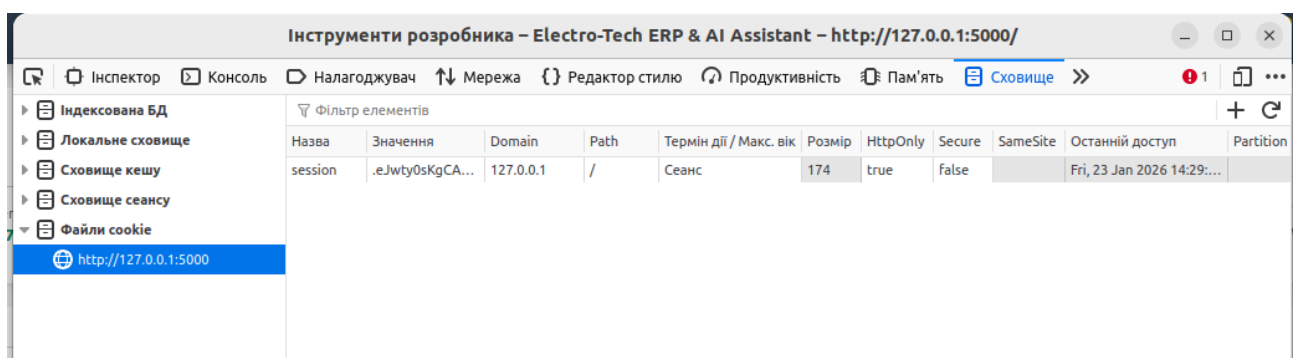


Рисунок 4.14 – Вікно інструментів розробника браузера Firefox

Далі натиснули знову кнопку в чаті “Весь склад” (функціонально команда /all), як це показано на рисунку 4.15.

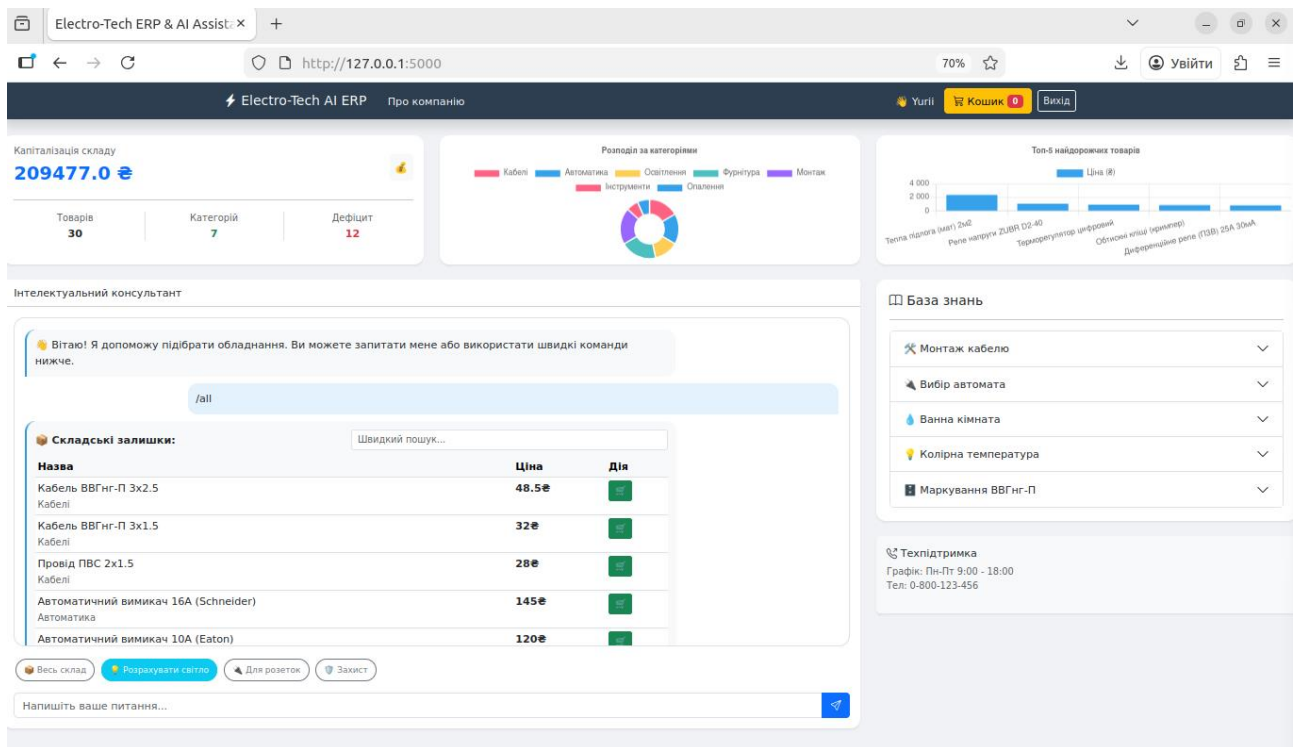


Рисунок 4.15 – Основне вікно додатку в браузері Firefox

Тепер ми можемо додати товари до кошика. Спробували додати такі товари: “Кабель ВВГнг-П 3x1.5”, “Автоматичний вимикач 10A (Eaton)” та “Вуличний прожектор LED 30W IP65” (див. рисунок 4.16).

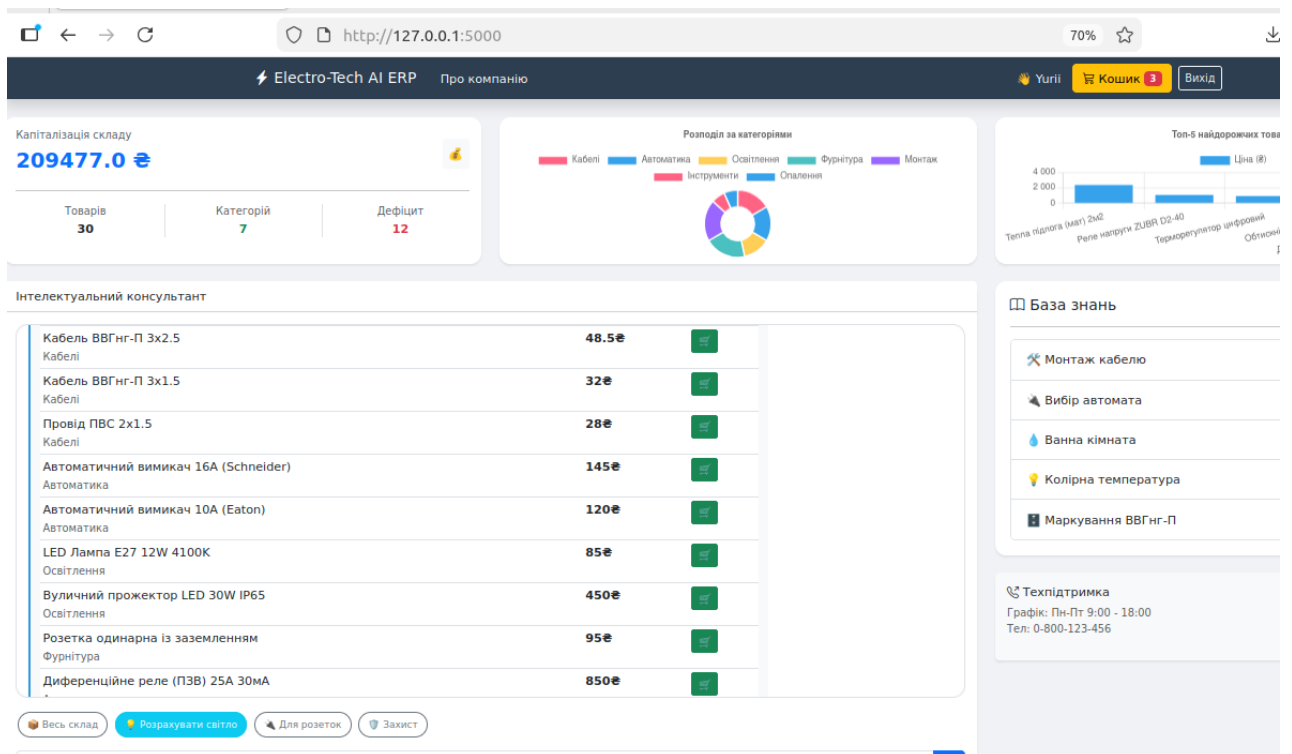


Рисунок 4.16 – Основне вікно додатку в браузері Firefox

Натиснули на кнопку Кошик зверху в правому куті, як це показано на рисунку 4.17.

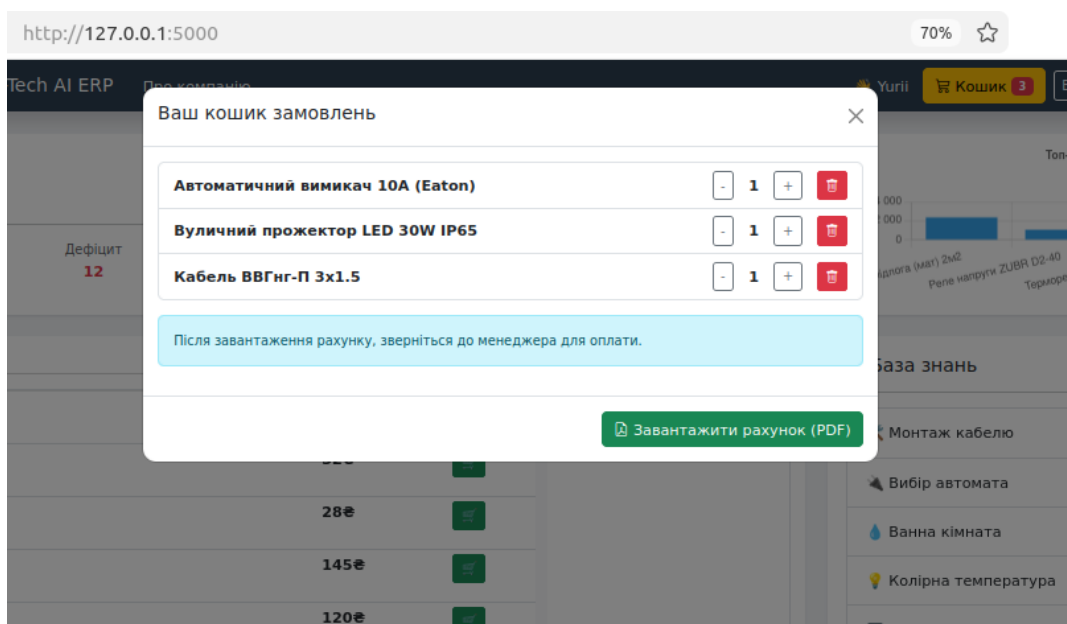


Рисунок 4.17 – Основне вікно додатку в браузері Firefox

Змінили кількість товару “Кабель ВВГнг-П 3х1.5” до 6 та видалили товар “Автоматичний вимикач 10А (Eaton)” (див. рисунок 4.18).

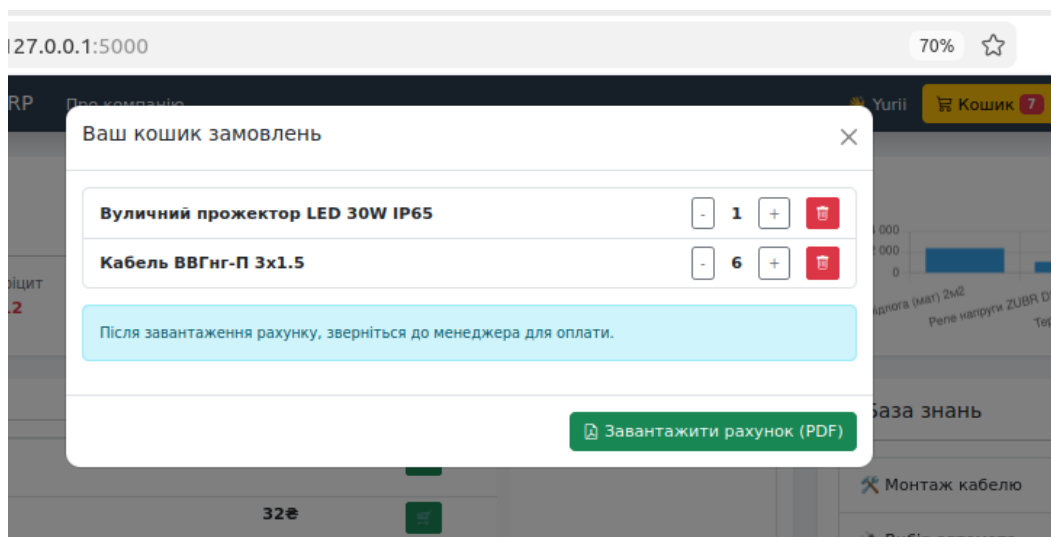


Рисунок 4.18 – Основне вікно додатку в браузері Firefox

Натиснули на кнопку “Завантажити рахунок (PDF)” для кінцевого оформлення замовлення (див. рисунок 4.19-4.20).

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		52

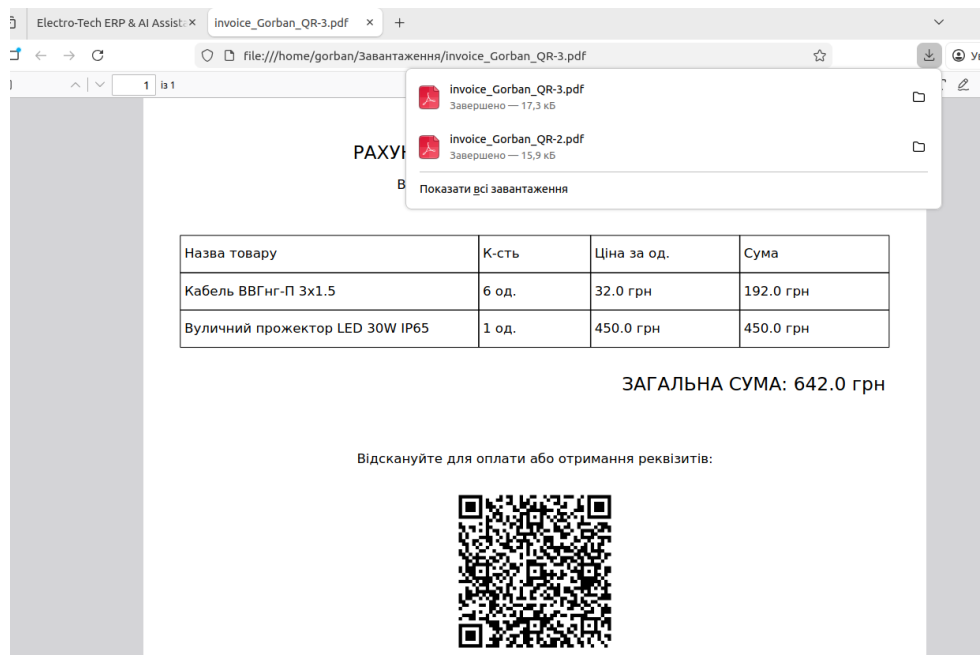


Рисунок 4.19 – Вміст завантаженого рахунку у вигляді PDF-файлу

Завантажився файл з ім'ям “invoice_Gorban_QR-3”, тому що це вже четвертий файл за період тестування.

Відповідно повний вигляд завантаженого файлу зображено на рисунку 4.20.

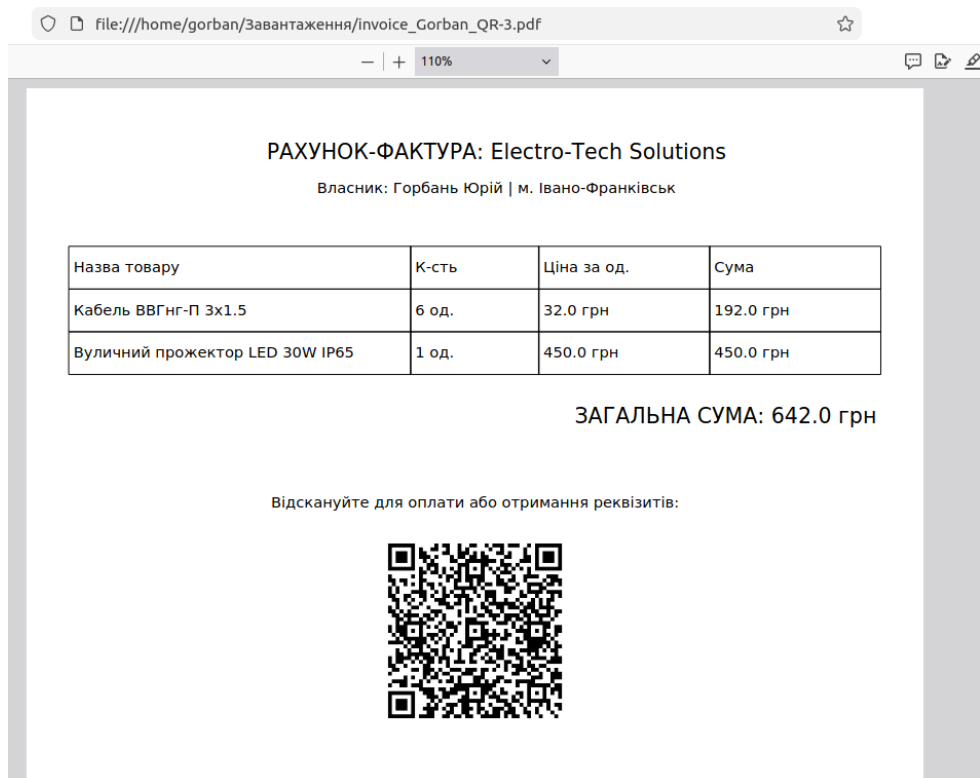


Рисунок 4.20 – Вміст завантаженого рахунку у вигляді PDF-файлу

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		53

Повернулись на головну сторінку (див. рисунок 4.21).

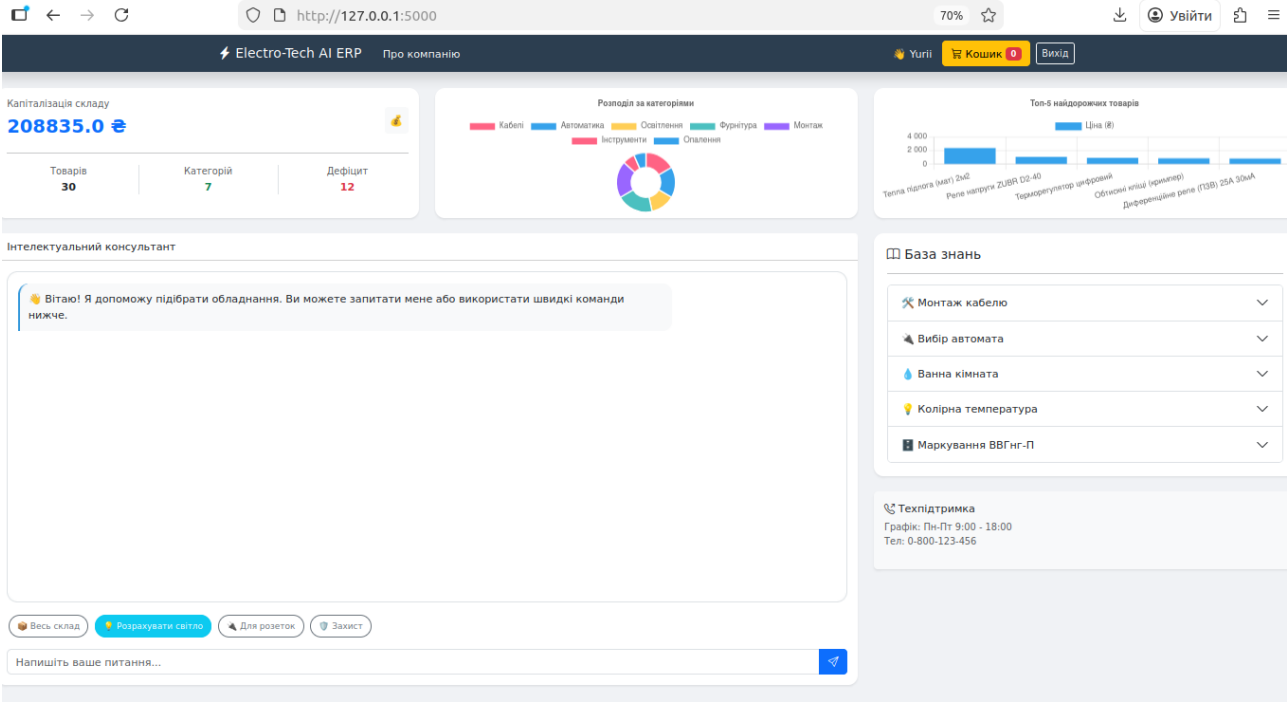


Рисунок 4.21 – Основне вікно додатку в браузері Firefox

Спроба доступу до маршруту /admin під обліковим записом зі статусом user призводила до негайного перенаправлення на головну сторінку, що свідчить про надійність реалізованих перевірок прав доступу (див. рисунок 4.22-4.23). Тестування форми реєстрації показало, що паролі в базі даних MongoDB зберігаються виключно у вигляді криптографічних хешів, унеможливлюючи їх компрометацію (див. рисунок 4.24).

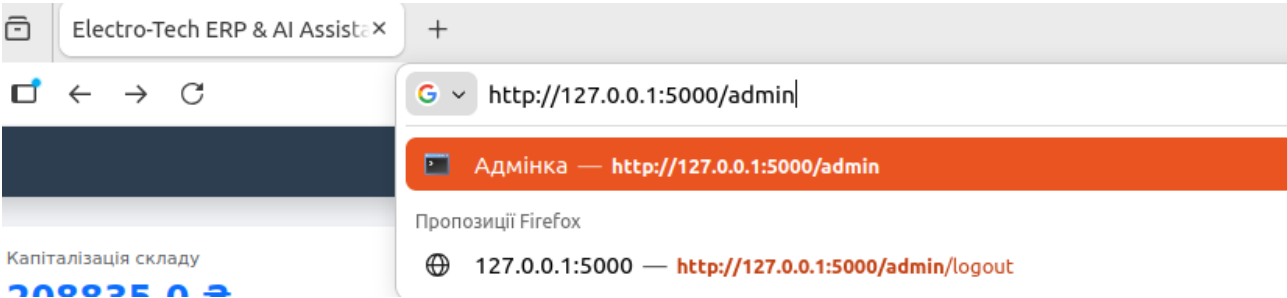


Рисунок 4.22 – Спроба перейти за адресою <http://127.0.0.1:5000/admin> в браузері Firefox

```

import google.generativeai as genai
* Debugger is active!
* Debugger PIN: 758-344-018
127.0.0.1 - - [23/Jan/2026 16:41:13] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [23/Jan/2026 16:41:14] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [23/Jan/2026 16:41:21] "POST /login HTTP/1.1" 200 -
127.0.0.1 - - [23/Jan/2026 16:41:22] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [23/Jan/2026 16:41:29] "GET /admin HTTP/1.1" 302 -
127.0.0.1 - - [23/Jan/2026 16:41:29] "GET / HTTP/1.1" 200 -

```

Рисунок 4.23 – Вигляд терміналу Linux при спробі перейти за адресою <http://127.0.0.1:5000/admin>, перебуваючи в статусі “Користувач”

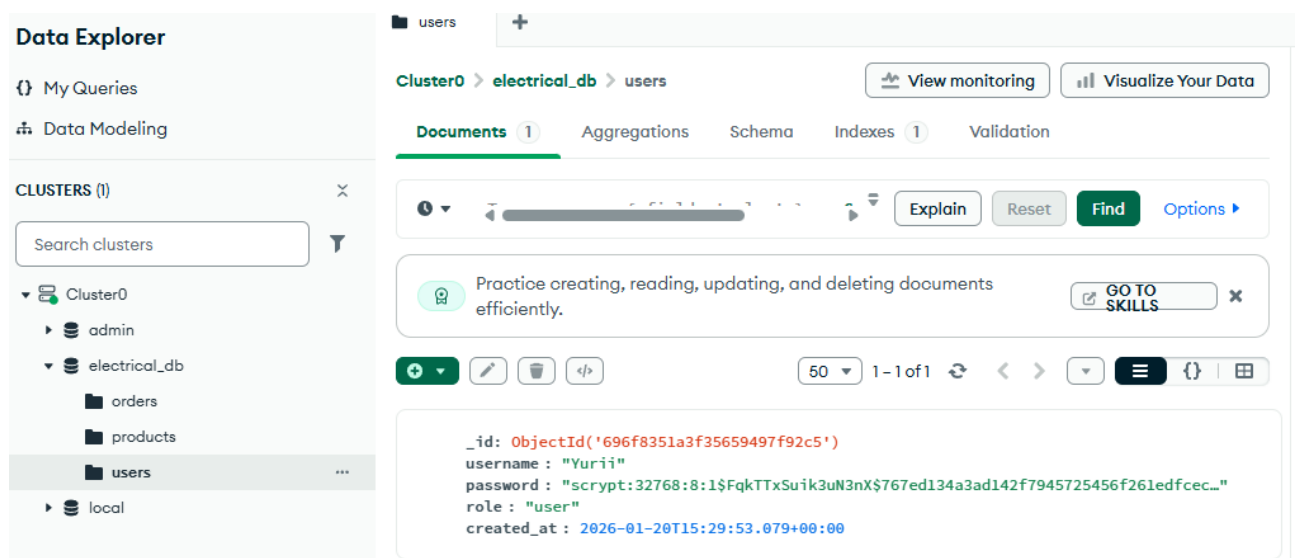


Рисунок 4.24 – Вигляд колекції “users” в базі даних “electrical_db” на сайті ПЗ MongoDB Atlas

Під час перевірки функціонала кошика було протестовано повний цикл вибору товарів, як це показано на рисунку 4.16. Додавання позицій відбувалося асинхронно через JavaScript; консоль Firefox підтвердила відсутність помилок виконання скриптів та коректність передачі ідентифікаторів товарів у форматі JSON. При зміні кількості одиниць у кошику загальна сума замовлення перераховувалася миттєво без перезавантаження сторінки (див. рисунок 4.17-4.18). Сценарій оформлення замовлення успішно завершувався списанням відповідної кількості товару зі складу в колекції products, що було перевірено шляхом прямого запиту до MongoDB після завершення тестової транзакції.

Далі приступили до тестування функціональних можливостей статусу “Адміністратор”.

Спочатку перейшли до бази даних “electrical_db” на сайті ПЗ MongoDB Atlas. Вибрали колекцію “users” та змінили роль (поле role) з “user” на “admin” нашого створеного акаунту з логіном “Yurii”, як це показано на рисунку 4.25.

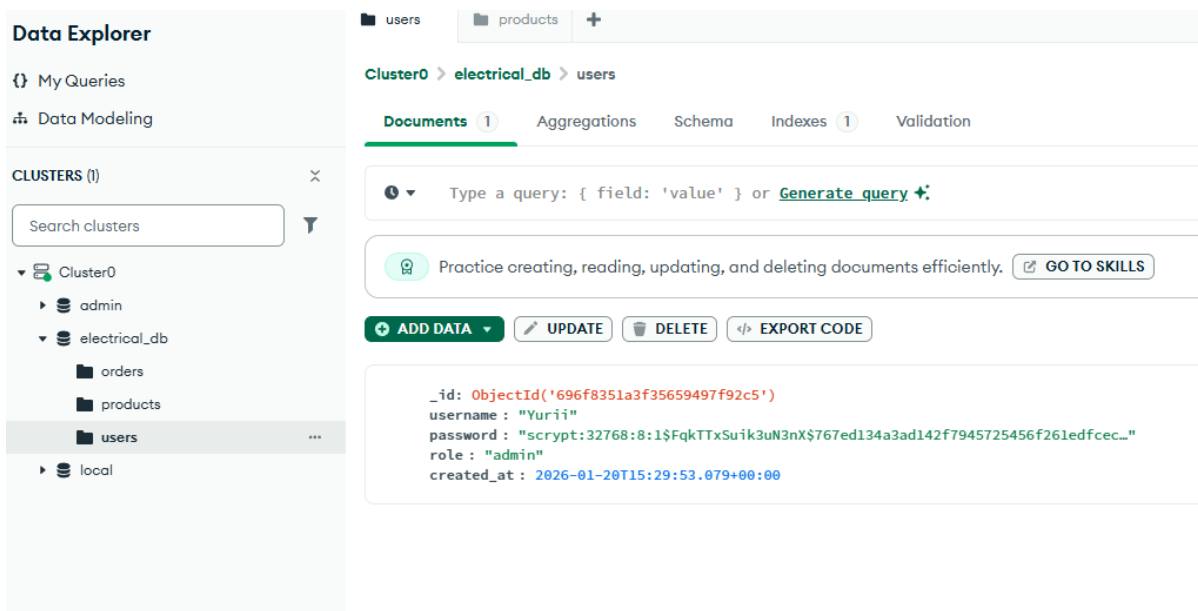


Рисунок 4.25 – Вигляд колекції “users” в базі даних “electrical_db” на сайті ПЗ MongoDB Atlas

Тепер повертаємось до нашого додатку. Знову увійшли в наш створений обліковий запис, як це показано на рисунку 4.26-4.27.

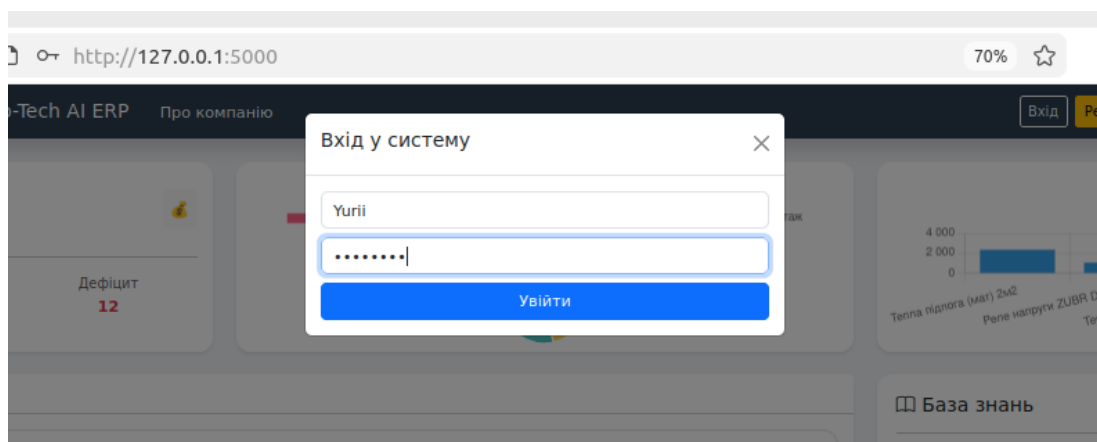


Рисунок 4.26 – Основне вікно додатку в браузері Firefox

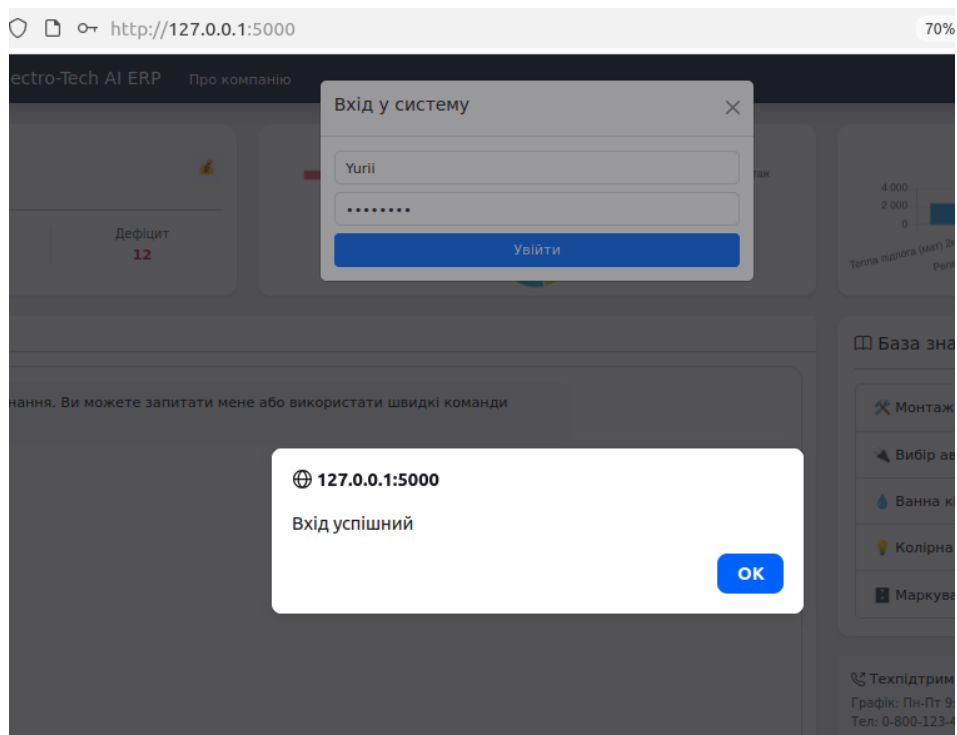


Рисунок 4.27 – Основне вікно додатку в браузері Firefox

Як бачимо, біля розділу “Про компанію” розташовується розділ “Адмін-панель” (див. рисунок 4.28).

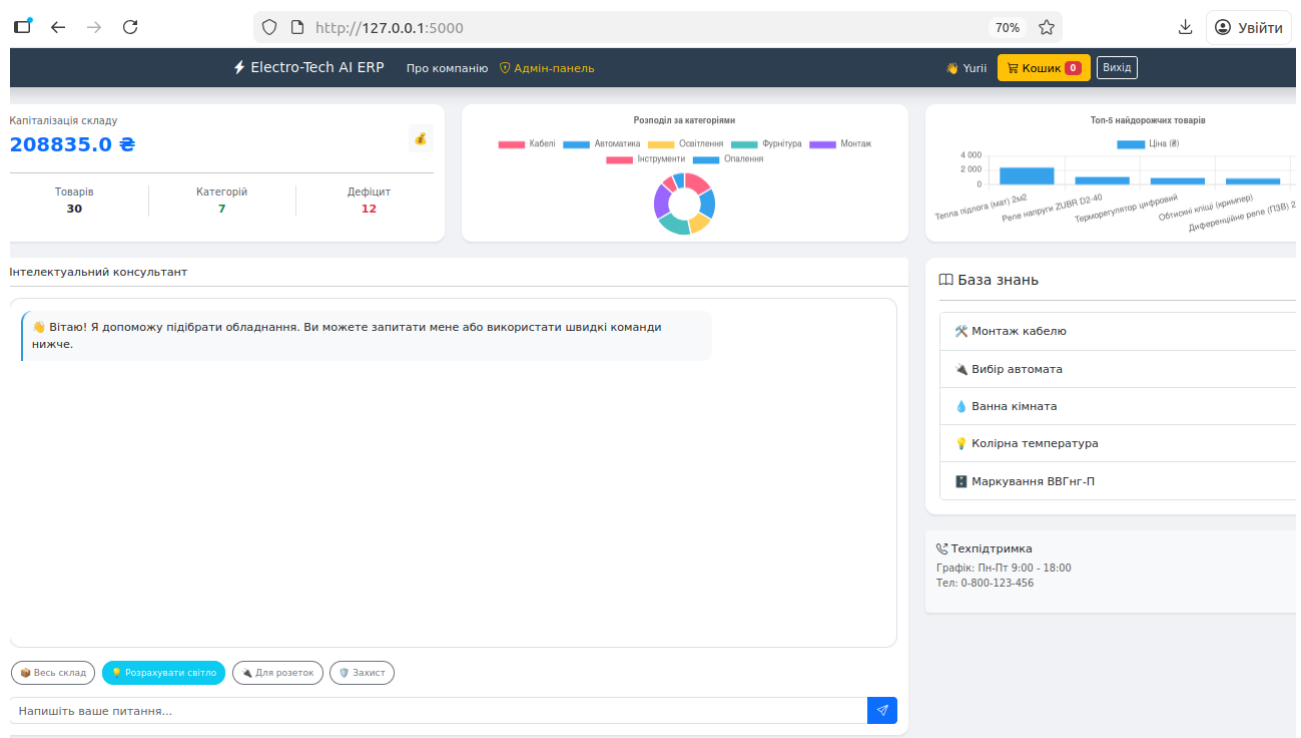


Рисунок 4.28 – Основне вікно додатку в браузері Firefox

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
						57
Зм.	Арк.	№ докум.	Підпис	Дата		

Натиснули на нього та перейшли на сторінку “Панель Адміністратора”, яка доступна за адресою <http://127.0.0.1:5000/admin> (див. рисунок 4.29).

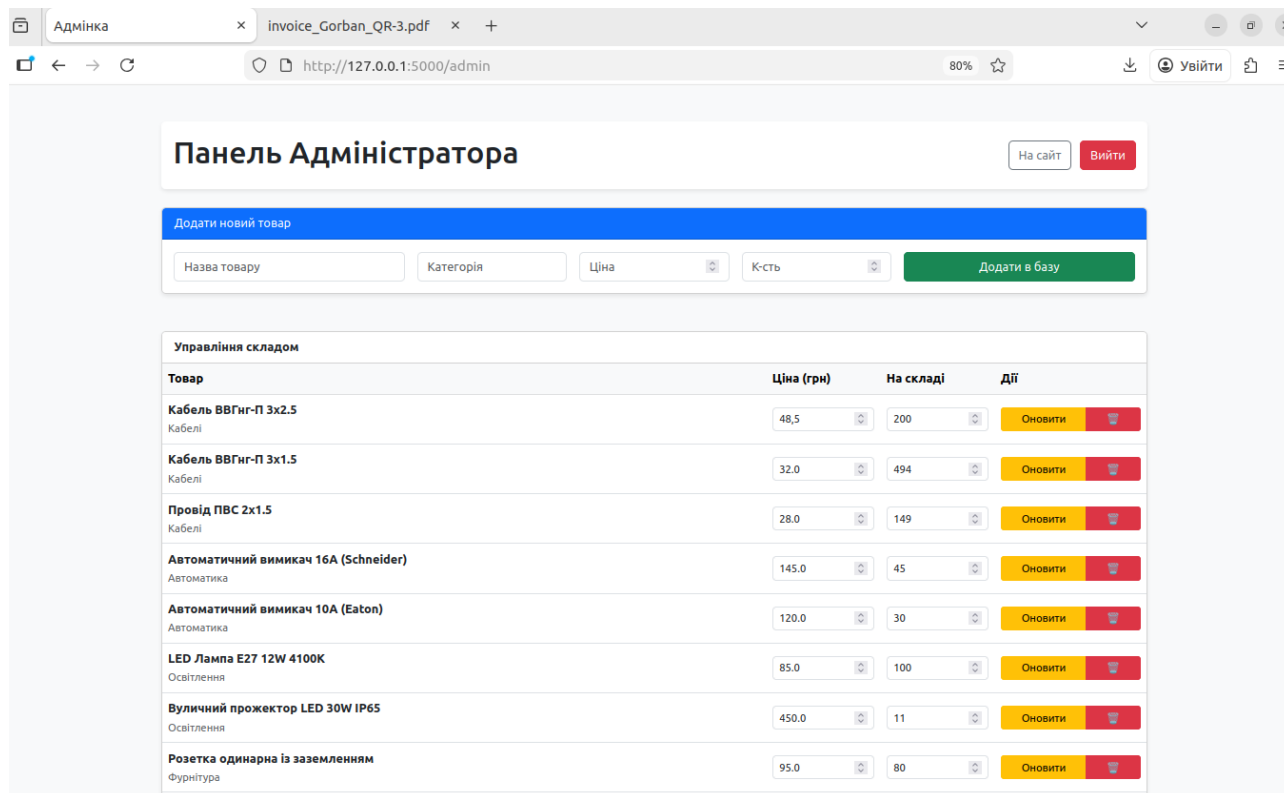


Рисунок 4.29 – Вікно сторінки адміністратора додатку в браузері Firefox

Спробували додати новий товар з такими характеристиками: Назва товару: Світильник світлодіодний лінійний 100W розсіювач призма 9500Lm 6500K 1,2m Ardero білий AL5020ARD; Категорія: Освітлення; Ціна: 420 грн.; Кількість: 14 (див. рисунок 4.30).

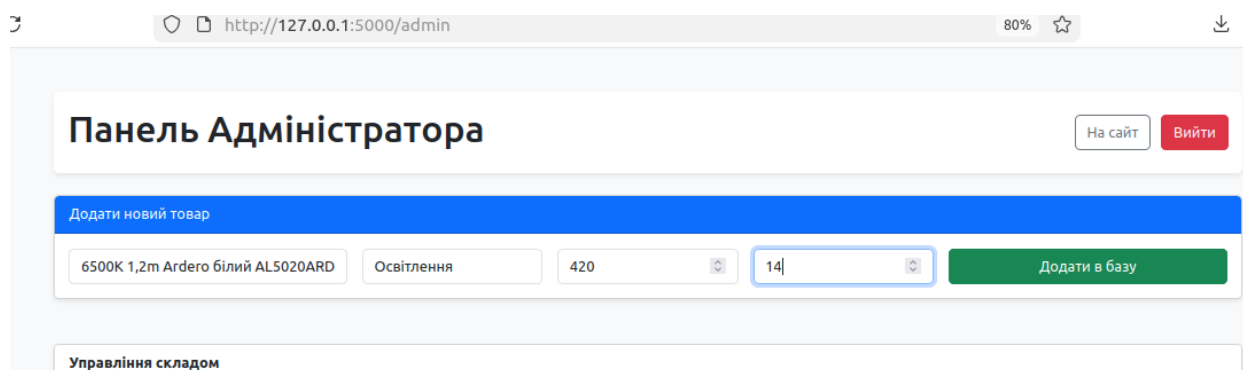


Рисунок 4.30 – Вікно сторінки адміністратора додатку в браузері Firefox

Натиснули кнопку “Додати в базу”. Прогорнули вниз сторінки та переконались, що товар додався успішно (див. рисунок 4.31).

Категорія	Назва	Ціна	Кількість	Дії
Фурнітура	Силовая розетка 32А (3-фазна)	480.0	10	Оновити
Монтаж	Кабель-канал 25х16 (2м)	42.0	150	Оновити
Освітлення	Светильник светодиодный линейный 100W розсіювач призма 9500Lm 6500K 1,2m Ardero білий AL5020ARD	420.0	14	Оновити

Зареєстровані користувачі		
Користувач	Роль	Дія
Yurii	admin	Історія

Рисунок 4.31 – Вікно сторінки адміністратора додатку в браузері Firefox

Спробували оновити кількість товару з 14-и на 20 і ціну на 500 грн., як це показано на рисунку 4.32.

Категорія	Назва	Ціна	Кількість	Дії
Фурнітура	Силовая розетка 32А (3-фазна)	480.0	10	Оновити
Монтаж	Кабель-канал 25х16 (2м)	42.0	150	Оновити
Освітлення	Светильник светодиодный линейный 100W розсіювач призма 9500Lm 6500K 1,2m Ardero білий AL5020ARD	500	20	Оновити

Зареєстровані користувачі		
Користувач	Роль	Дія
Yurii	admin	Історія

Рисунок 4.32 – Вікно сторінки адміністратора додатку в браузері Firefox

Натиснули кнопку “Оновити” та перевірили, перейшовши на сторінку бази даних “electrical_db” в ПЗ MongoDB Atlas (див. рисунок 4.33).

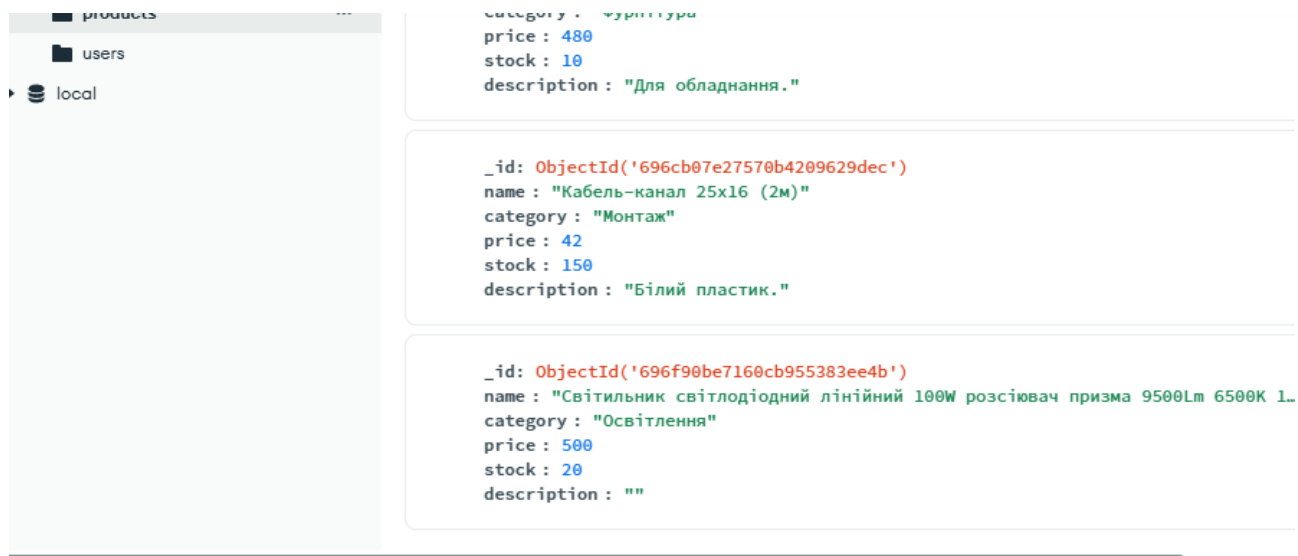


Рисунок 4.33 – Вигляд колекції “products” в базі даних “electrical_db” на сайті ПЗ MongoDB Atlas

В результаті інформація оновилась успішно.

Далі спробували видалити цей товар, натиснувши на червону кнопку смітника, як це показано на рисунку 4.34-4.35.

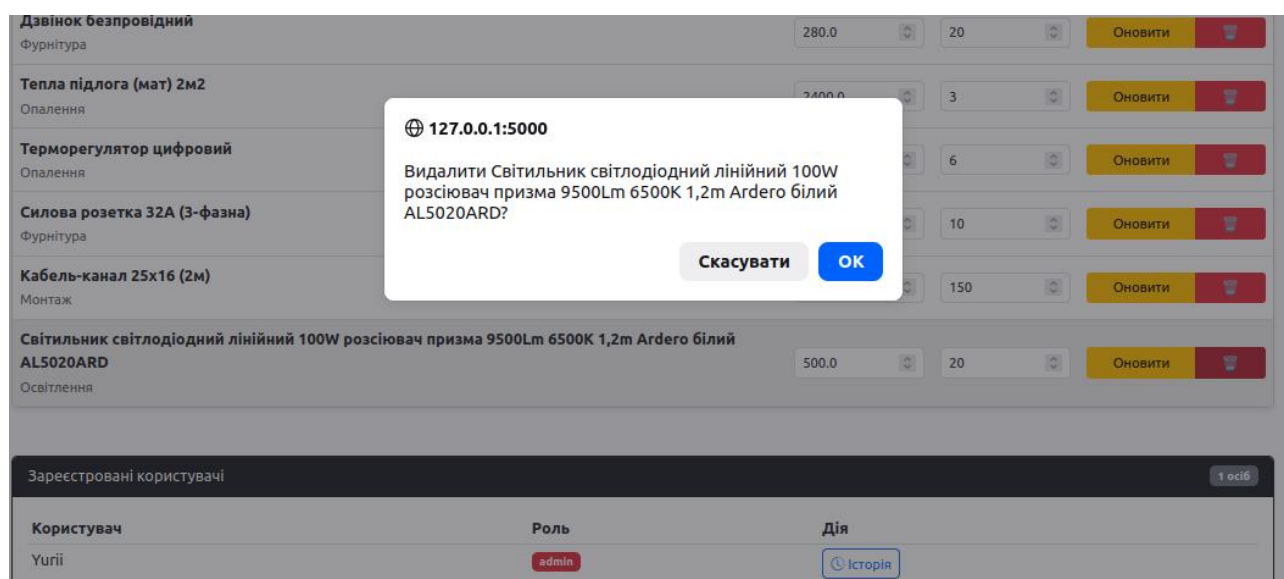


Рисунок 4.34 – Вікно сторінки адміністратора додатку в браузері Firefox

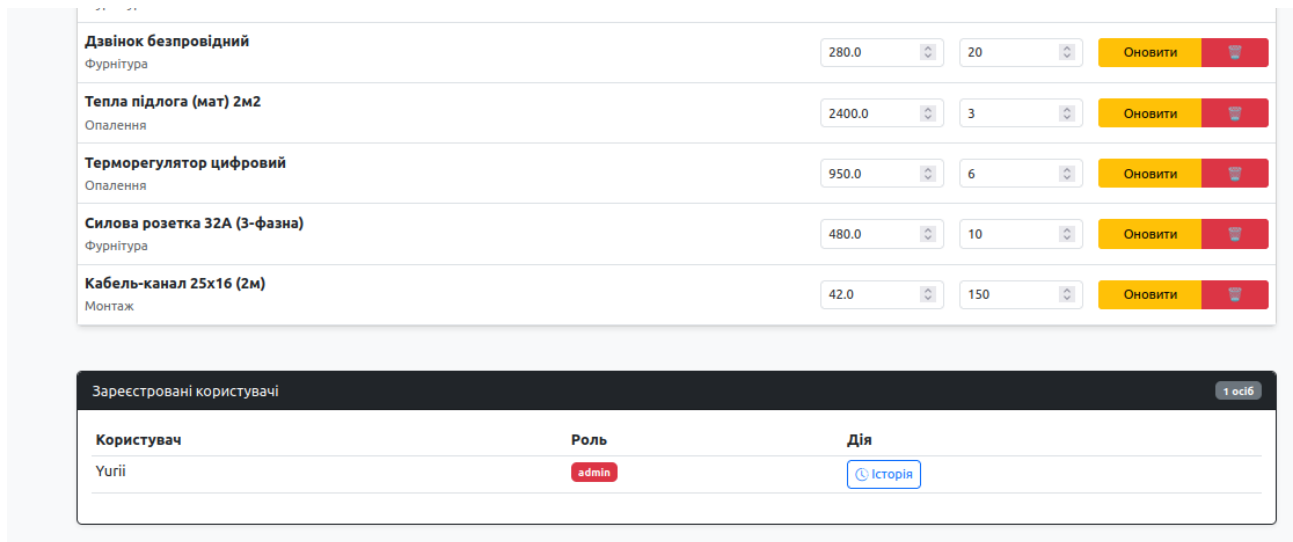


Рисунок 4.35 – Вікно сторінки адміністратора додатку в браузері Firefox

Товар успішно видалено.

Внизу сторінки бачимо розділ з назвою “Зареєстровані користувачі”, в якому наш обліковий запис є доданим також. Натиснули кнопку “Історія” в стовпчику “Дія” для перегляду списку всіх замовлень користувачів (див. рисунок 4.36).

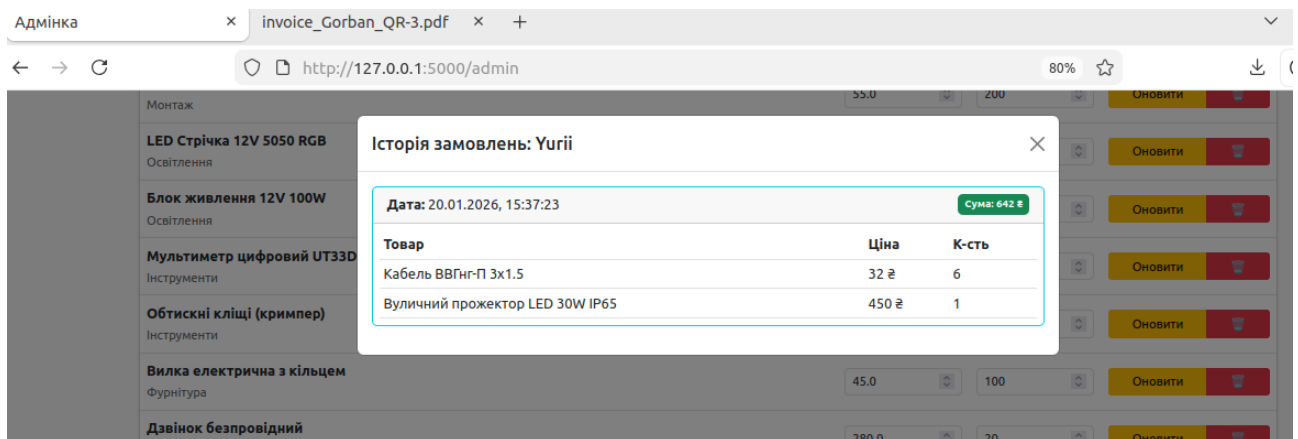


Рисунок 4.36 – Вікно сторінки адміністратора додатку в браузері Firefox

Як бачимо, в списку історії замовлень є наше, створене замовлення з потрібними деталями.

Тестування адміністративного інтерфейсу зосереджувалося на CRUD-операціях та відображенні історії замовлень (див. рисунок 4.32-4.36). Внесення змін у ціну та опис товару через форми адмін-панелі призводило до негайного

оновлення даних у базі, а вкладка «Мережа» (Network) у Firefox фіксувала успішні POST-запити зі статусом 200 OK, як це показано на рисунку 4.37.

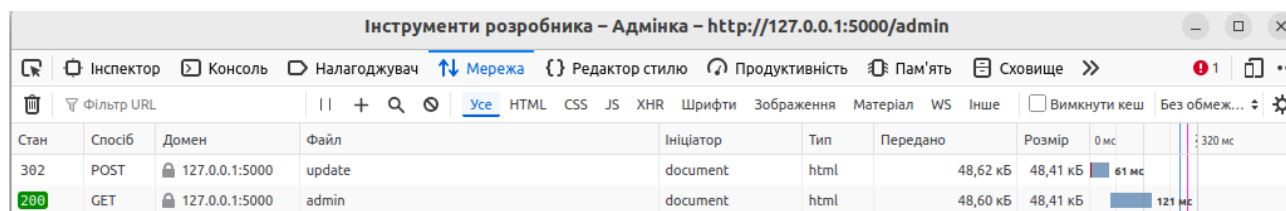


Рисунок 4.37 – Інструменти розробника в браузері Firefox

Особлива увага була приділена перегляду замовлень: завдяки реалізованій серіалізації ідентифікаторів ObjectId, дані про клієнтів та склад їхніх покупок відображалися у модальних вікнах коректно, без системних збоїв. Модуль генерації PDF-інвойсів успішно створював документи, які коректно відкривалися вбудованим переглядачем Firefox, зберігаючи структуру таблиць та згенерований QR-код (див. рисунок 4.20), що підтверджує повну працездатність системи в умовах, наближених до експлуатаційних.

4.3 Тестування точності ШІ-пошуку та рекомендацій на реальних даних

Аналіз роботи інтелектуального модуля проводився шляхом подання серії контрольних запитів, що охоплюють різні аспекти взаємодії клієнта з магазином: від простого пошуку залишків до отримання комплексних інженерних порад. Під час тестування у браузері Firefox було зафіксовано, що система стабільно опрацьовує неструктуровані дані, демонструючи високу релевантність відповідей завдяки прямому доступу до бази даних.

Перший етап тестування був спрямований на перевірку знання асортименту. На запит «Які кабелі у вас є в наявності та яка їхня ціна?» ШІ-консультант миттєво згенерував перелік товарів із категорії «Кабелі», вказавши актуальну вартість за метр, отриману безпосередньо з MongoDB (див. рисунок

4.38). Це підтвердило здатність моделі Gemini коректно ідентифікувати типи товарів у загальній базі.

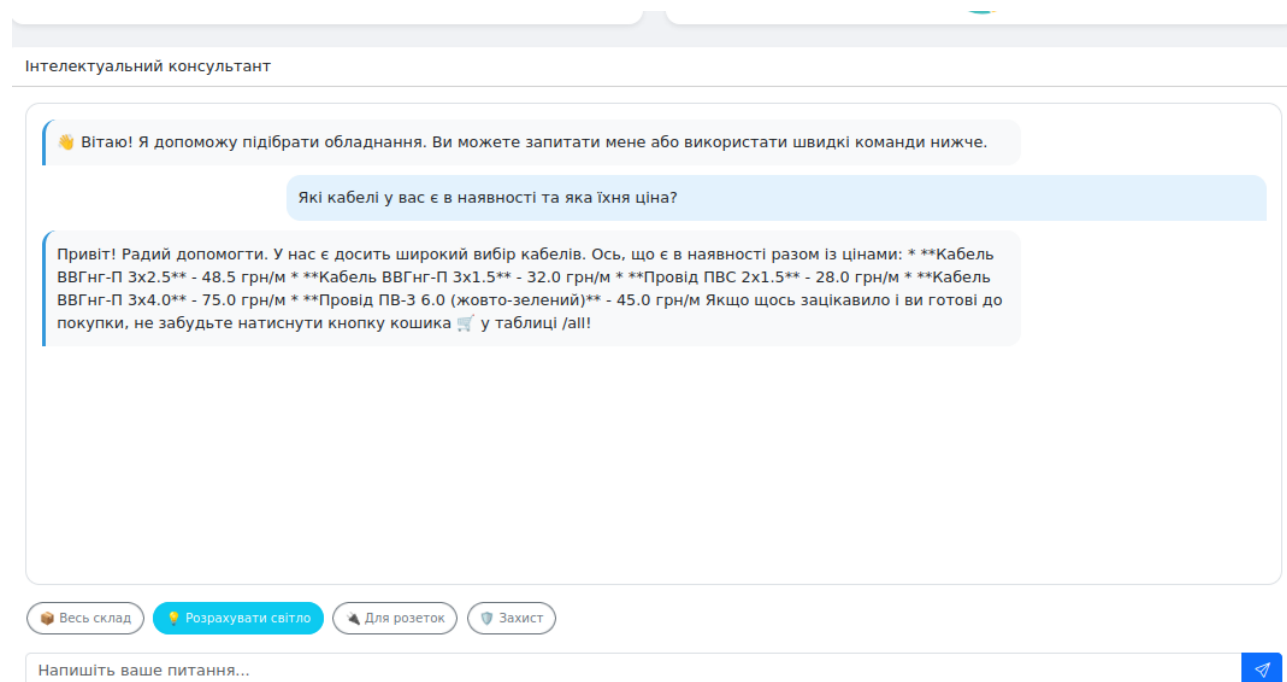


Рисунок 4.38 – Основне вікно додатку в браузері Firefox

Наступний тест стосувався професійної експертизи: на складний запит щодо підключення пральної машини (вибір кабелю та автоматичного вимикача) бот надав розгорнуту відповідь, рекомендуючи кабель ВВГнг-П 3х2.5, автоматичний вимикач на 16А, диференційне реле (ПЗВ) 25А 30мА та розетку одинарну із заземленням, що відповідає стандартам ПУЕ (див. рисунок 4.39). Це демонструє, що система не просто шукає слова, а аналізує технічну логіку задачі.

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		63

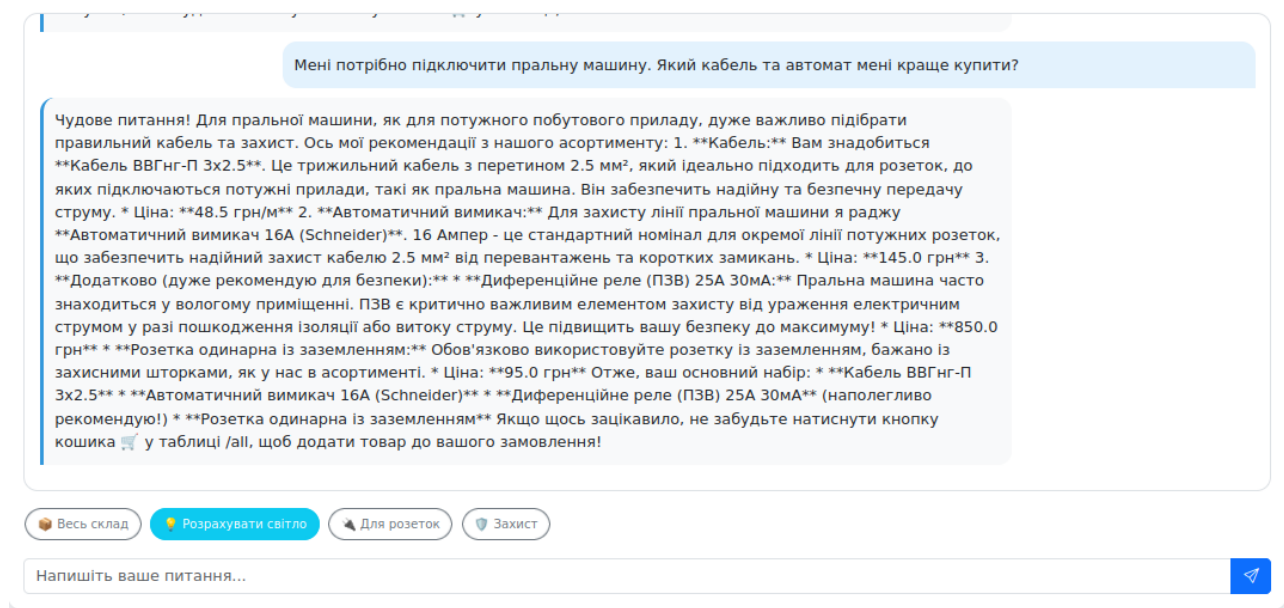


Рисунок 4.39 – Основне вікно додатку в браузері Firefox

Перевірка функцій фільтрації за неявними характеристиками проводилася через запит «Підкажи, які у вас є лампи з нейтральним світлом?». ШІ успішно проаналізував текстові описи товарів у колекції products, виокремив позиції з колірною температурою 4000-4500K та видав точний список, як це показано на рисунку 4.40.

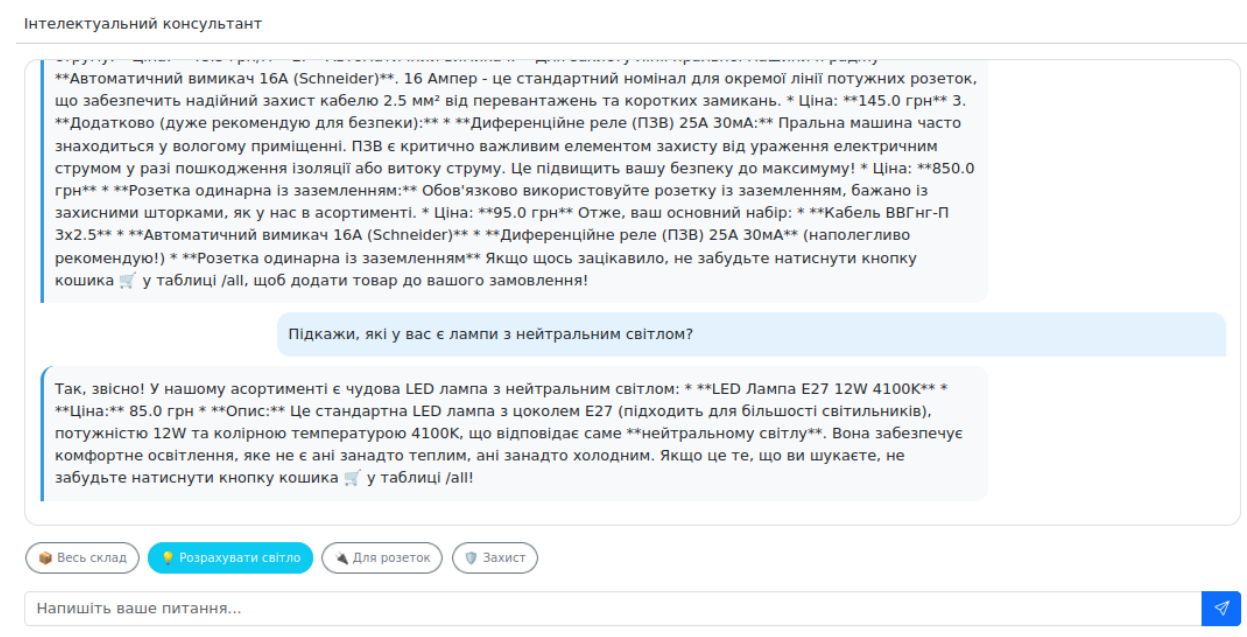


Рисунок 4.40 – Основне вікно додатку в браузері Firefox

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		64

Окремо було протестовано модуль безпеки та інструктажу. На запит «Як захистити дитину від ураження струмом біля розеток?» система видала докладну інструкцію щодо використання захисних шторкових механізмів та пристроїв захисного відключення (ПЗВ), що свідчить про високу інформативність бази знань моделі (див. рисунок 4.41).

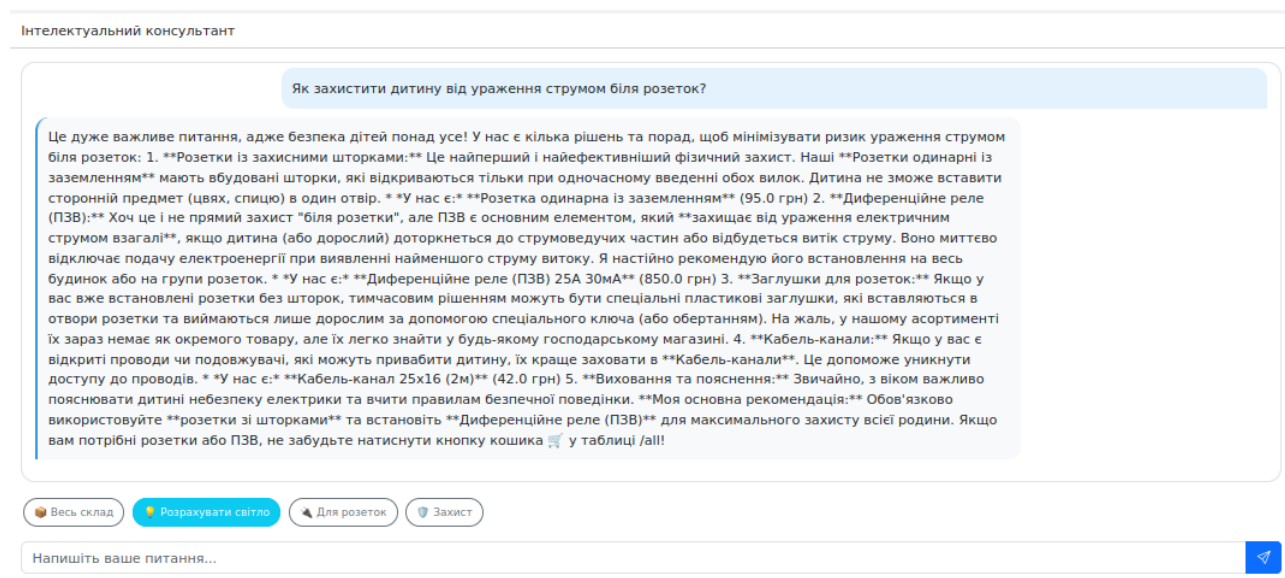


Рисунок 4.41 – Основне вікно додатку в браузері Firefox

Висока точність та «людяність» відповідей пояснюються впровадженою архітектурою RAG (Retrieval-Augmented Generation). Програмна логіка працює за триступеневим циклом: спочатку система витягує «сирі» дані про товари з MongoDB, потім передає цей актуальний контекст разом із запитом користувача до API Gemini, де ШІ аналізує базу «на льоту» і формує відповідь у ролі професійного живого консультанта. Такий підхід виключає галюцинації ШІ та гарантує, що клієнт отримує інформацію лише про ті товари, які дійсно є на складі в цей момент. Таким чином, результати тестування підтверджують, що інтегрований інтелект повністю замінює традиційні фільтри, забезпечуючи новий рівень сервісу в електротехнічній галузі.

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		65

4.4 Аналіз інтерфейсу користувача (UI/UX) та оцінка швидкодії системи

Підсумковий аналіз у браузері Firefox підтвердив високу ефективність обраного стеку технологій (див. рисунок 4.5-4.37). Використання Bootstrap 5 дозволило досягти повної адаптивності: інтерфейс залишається зручним як на десктопних моніторах, так і на мобільних пристроях, що є критичним для інженерів, які можуть працювати з каталогом безпосередньо на об'єктах монтажу. Користувацький досвід (UX) покращено завдяки мінімізації кліків для оформлення замовлення та інтеграції ШІ-чату, який замінює складні багаторівневі фільтри.

Вимірювання швидкодії через вкладку Network у Firefox показали такі результати:

- час відгуку сервера (TTFB) для стандартних сторінок становить 40-60 мс, що свідчить про високу продуктивність Flask та ефективність індексації в MongoDB;
- генерація PDF-інвойсу займає до 250 мс, що забезпечує миттєву видачу документа користувачеві;
- обробка запиту через Gemini API триває від 1.5 до 2.8 сек. Хоча це найповільніший вузол, асинхронне завантаження (AJAX) дозволяє інтерфейсу залишатися чуйним, не блокуючи роботу користувача.

Система продемонструвала стабільність та високу швидкість обробки даних, що в поєднанні з сучасним UI робить її готовою до впровадження в реальні бізнес-процеси підприємства.

Висновки до четвертого розділу

На основі даних, отриманих під час тестування та контрольних замірів ефективності, у четвертому розділі сформульовано такі висновки:

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		66

Комплексна перевірка вебдодатка у браузері Firefox підтвердила повну працездатність усіх функціональних модулів. Система продемонструвала надійність механізмів авторизації, стабільність роботи кошика та коректність CRUD-операцій в адміністративній панелі. Використання інструментів розробника Firefox дозволило підтвердити високу швидкість відгуку серверної частини на Flask (40–60 мс) та ефективність генерації PDF-інвойсів із вбудованими QR-кодами, що дозволяє кінцевому користувачеві отримувати результати без затримок (з мінімальною латентністю).

Визначним досягненням стала перевірка інтелектуального модуля на основі моделі Gemini. Впровадження архітектури RAG (Retrieval-Augmented Generation) дозволило III-консультанту вийти за межі звичайного пошуку: система не лише точно ідентифікує товари в базі MongoDB, а й надає професійні інженерні поради (наприклад, щодо підбору кабелів чи захисту електромереж). Хоча час очікування відповіді від III є вищим за стандартні запити, реалізована асинхронна модель взаємодії зберігає зручність інтерфейсу. Загалом, результати тестування довели, що поєднання сучасного UI на Bootstrap 5, швидкої бази даних та штучного інтелекту створює ефективний інструмент автоматизації, повністю готовий до практичної експлуатації.

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
						67
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході виконання бакалаврської роботи було успішно спроектовано та реалізовано інтелектуальну інформаційну систему для автоматизації процесів торгівлі електротехнічною продукцією. Розроблення довело ефективність поєднання сучасних вебтехнологій із методами штучного інтелекту для вирішення прикладних бізнес-задач. Використаний технологічний стек, що включає мову Python, мікрофреймворк Flask та NoSQL-базу даних MongoDB, дозволив створити гнучку систему, здатну опрацьовувати складні технічні характеристики товарів та гарантувати високу пропускну здатність системи.

Головним здобутком проєкту стала успішна інтеграція великої мовної моделі Gemini на основі архітектури Retrieval-Augmented Generation (RAG). Це рішення дозволило трансформувати традиційний пошук у повноцінну систему інтелектуального консультування, де ШІ аналізує актуальний стан складу в MongoDB та надає професійні поради щодо вибору обладнання, опираючись на реальний контекст запитів користувача. Паралельно з цим було повністю автоматизовано процес супроводження продажів: від надійної автентифікації користувачів із хешуванням паролів до миттєвої генерації електронних PDF-інвойсів із вбудованими QR-кодами для оплати.

Результати комплексного тестування у браузері Firefox підтвердили технічну досконалість розробленого продукту. Показники швидкодії серверної частини та безперебійне функціонування бази даних відповідають високим вимогам до сучасних корпоративних систем, а адаптивний інтерфейс на базі Bootstrap 5 гарантує зручність експлуатації на будь-яких пристроях. Реалізована адміністративна панель забезпечила зручний інструментарій для моніторингу замовлень та оперативного керування асортиментом. Створена система є завершеним програмним рішенням, яке не лише автоматизує рутинні операції, а й створює новий рівень клієнтського сервісу завдяки впровадженню інтелектуальних технологій, що підтверджує повну реалізацію поставленої мети роботи.

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		68

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Епіцентр К – Електрика та освітлення [Електронний ресурс]. – Режим доступу: <https://epicentrk.ua/ua/shop/elektryka-ta-osvitlennia/>
2. Rozetka – Освітлення [Електронний ресурс]. – Режим доступу: <https://build.rozetka.com.ua/ua/osveshchenie-dlia-doma/c230134/>
3. RPA: революційна ефективність для бізнесу [Електронний ресурс]. – Режим доступу: <https://blog.globalbilgi.com.ua/rpa-revoliutsiyna-efektyvnist-dlia-biznesu/>
4. FPDF Library Documentation for Python [Електронний ресурс]. – Режим доступу: <https://pyfpdf.github.io/fpdf2/>
5. Google Gemini API Documentation [Електронний ресурс]. – Режим доступу: <https://ai.google.dev/docs>
6. Amazon Official Site: Online Shopping for Electronics, Apparel, Computers & more [Електронний ресурс]. – Режим доступу: <https://www.amazon.com/>
7. Alibaba.com: Leading B2B Marketplace for Global Trade [Електронний ресурс]. – Режим доступу: <https://www.alibaba.com/>
8. eBay: Electronics, Cars, Fashion, Collectibles & More [Електронний ресурс]. – Режим доступу: <https://www.ebay.com/>
9. Retrieval-Augmented Generation (RAG) – Explained [Електронний ресурс]. – Режим доступу: <https://aws.amazon.com/what-is/retrieval-augmented-generation/>
10. Lutz M. Learning Python. 5th ed. – Sebastopol : O'Reilly Media, 2013. – 1648 p.
11. Фаулер М. Архітектура корпоративних додатків / пер. з англ. – Київ : Діалектика, 2020. – 560 с.
12. Django's MVT Architecture – A Detailed Guide [Електронний ресурс]. – Режим доступу: <https://techvidvan.com/tutorials/djangos-mvt-architecture/>

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		69

13. Flask vs. Django: Which Python Framework Should You Choose? [Електронний ресурс]. – Режим доступу: <https://djangostars.com/blog/flask-vs-django/>
14. Grinberg M. Flask Web Development: Developing Web Applications with Python. 2nd ed. – Sebastopol : O'Reilly Media, 2018. – 312 p.
15. Flask Documentation [Електронний ресурс]. – Режим доступу: <https://flask.palletsprojects.com/>
16. The Flask Architecture and Web Server Processing [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/figure/The-Flask-architecture_fig1_290715751
17. Integrating MongoDB with Flask: A Step-by-Step Guide [Електронний ресурс]. – Режим доступу: <https://medium.com/@sakshijadhav1792/integrating-mongodb-with-flask-204bb45e5a64>
18. Chodorow K. MongoDB: The Definitive Guide. 3rd ed. – Sebastopol : O'Reilly Media, 2019. – 514 p.
19. MongoDB Database Architecture Basics [Електронний ресурс]. – Режим доступу: <https://www.mongodb.com/resources/basics/databases/database-architecture>
20. MongoDB Documentation [Електронний ресурс]. – Режим доступу: <https://www.mongodb.com/docs/>
21. Bootstrap Documentation [Електронний ресурс]. – Режим доступу: <https://getbootstrap.com/docs/5.3/>
22. Werkzeug – The WSGI Web Application Library [Електронний ресурс]. – Режим доступу: <https://werkzeug.palletsprojects.com/>
23. Unlocking Insights: Exploring Client-Server Architecture [Електронний ресурс]. – Режим доступу: <https://medium.com/@shubavarma/unlocking-insights-exploring-client-server-architecture-for-business-and-data-science-78d4344b877d>
24. Lucidchart – Online Diagram Software & Visual Collaboration [Електронний ресурс]. – Режим доступу: <https://lucid.app/>

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		70

25. Обґрунтування вибору нереляційної моделі даних на базі MongoDB для обліку продукції та купівельної активності / Ю. О. Горбань, О. І. Белей, Л. О. Штаєр, Р. Б. Стасюк [Електронний ресурс]. – Режим доступу: <https://doi.org/10.36074/grail-of-science.20.03.2026.078>

					КРБ.ІСТ – 05.00.00.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		71

ДОДАТОК А. Програмна реалізація вебдодатка «electro_advisor»

А.1. Структура файлової системи та опис компонентів

Проект побудований за модульним принципом, де кожен файл відповідає за окрему частину бізнес-логіки системи (див. таблицю 5.1 та рисунок 5.1).

Таблиця 5.1 – Структура розробленого додатка “electro_advisor”

Файл/Папка	Опис
templates/	Шаблони інтерфейсу (index.html, admin.html, about.html)
.env	Конфігураційні ключі (API та БД)
app.py	Серверна логіка та інтеграція з Gemini AI
check_models.py	Модуль перевірки бази даних
DejaVuSans.ttf	Шрифт для кириличних PDF-інвойсів
requirements.txt	Перелік програмних залежностей
seed_db.py	Скрипт початкового наповнення MongoDB

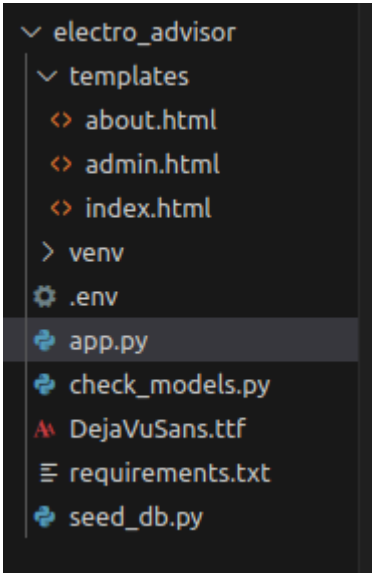


Рисунок 5.1 – Структура додатку в VS Code

А.2. Серверний модуль додатка (app.py)

Реалізує маршрутизацію, обробку сесій, взаємодію з MongoDB та архітектуру RAG для штучного інтелекту.

```
import os

from flask import Flask, render_template, request, jsonify, session, send_file,
redirect, url_for

from dotenv import load_dotenv

import google.generativeai as genai

from pymongo import MongoClient

from fpdf import FPDF

import io

import qrcode

from io import BytesIO

from werkzeug.security import generate_password_hash, check_password_hash

from datetime import datetime


load_dotenv()

app = Flask(__name__)

app.secret_key = "electrical_secret_key_123" # ключ для шифрування сесій

# Пароль для адмінки
ADMIN_PASSWORD = "yura_electro_2026123"


# Налаштування Gemini
genai.configure(api_key=os.getenv("GEMINI_API_KEY"))
model = genai.GenerativeModel('gemini-2.5-flash')


# налаштування MongoDB
client = MongoClient(os.getenv("MONGO_URI"))
db = client['electrical_db']
products_collection = db['products']
users_collection = db['users']
orders_collection = db['orders']


@app.route('/')

def index():
```

```
# очищуємо історію чату при вході на головну
session['chat_history'] = []

if 'cart' not in session or not isinstance(session['cart'], dict):
    session['cart'] = {}

all_prods = list(products_collection.find())

# Розрахунки для статистики та графіків
categories_map = {}
total_value = 0
low_stock_count = 0

for p in all_prods:
    # Рахуємо категорії
    cat = p.get('category', 'Інше')
    categories_map[cat] = categories_map.get(cat, 0) + 1

    # рахуємо загальну вартість
    total_value += (p.get('price', 0) * p.get('stock', 0))

    # рахуємо товари з малим залишком
    if p.get('stock', 0) < 20:
        low_stock_count += 1

top_products = sorted(all_prods, key=lambda x: x['price'], reverse=True)[:5]

# Збираємо ПОВНИЙ словник stats
stats = {
    "total_products": len(all_prods),
    "total_value": round(total_value, 2),
    "categories": len(categories_map),
    "low_stock": low_stock_count,
    "cat_labels": list(categories_map.keys()),
```

```

        "cat_values": list(categories_map.values()),
        "top_names": [p['name'] for p in top_products],
        "top_prices": [p['price'] for p in top_products]
    }

    return render_template('index.html', stats=stats)

@app.route('/ask', methods=['POST'])
def ask():
    user_message = request.json.get("message").strip()

    # ОБРОБКА КОМАНДИ /all
    if user_message.lower() == '/all':
        all_products = list(products_collection.find({}, {"_id": 0}))
        return jsonify({"type": "table", "data": all_products})

    # ЗАВАНТАЖЕННЯ ІСТОРІЇ ТА КОНТЕКСТУ
    # отримуємо історію з сесії або створюємо порожню
    history = session.get('chat_history', [])

    all_products = list(products_collection.find({}, {"_id": 0}))
    products_context = f"Ти консультант магазину електротехніки. Наш асортимент: {str(all_products)}. " \
        f"Якщо користувач хоче купити товар, нагадай натиснути кнопку кошика 🛒 у таблиці /all."

    # ЗАПУСК ЧАТУ
    # Перетворюємо історію з сесії назад у формат, який розуміє Gemini
    chat = model.start_chat(history=history)

    try:
        # додаємо системну інструкцію до першого повідомлення, якщо історія порожня
        full_prompt = user_message
        if not history:

```

```

        full_prompt = f"{products_context}\n\nКористувач запитує:
{user_message}"

    response = chat.send_message(full_prompt)

    # ЗБЕРЕЖЕННЯ ІСТОРИЇ (Конвертація в JSON-формат)
    new_history = []
    for content in chat.history:
        new_history.append({
            "role": content.role,
            "parts": [{"text": part.text} for part in content.parts]
        })

    # Зберігаємо тільки останні 10 повідомлень, щоб не переповнювати сесію
    session['chat_history'] = new_history[-10:]
    session.modified = True

    return jsonify({"type": "text", "reply": response.text})

except Exception as e:
    print(f"Помилка Gemini: {e}")
    return jsonify({"type": "text", "reply": "Вибачте, сталася технічна
помилка. Спробуйте ще раз."}), 500

@app.route('/add_to_cart', methods=['POST'])
def add_to_cart():
    name = request.json.get("name")
    product = products_collection.find_one({"name": name})

    if not product:
        return jsonify({"error": "Товар не знайдено"}), 404

    # гарантуємо, що кошик - це словник
    if 'cart' not in session or not isinstance(session['cart'], dict):
        session['cart'] = {}

```

```

current_qty = session['cart'].get(name, 0)

if current_qty < product.get('stock', 0):
    session['cart'][name] = current_qty + 1
    session.modified = True
else:
    return jsonify({"error": "Досягнуто ліміт складу"}), 400

    return jsonify({"cart_count": sum(session['cart'].values()), "cart":
session['cart']})

@app.route('/update_quantity', methods=['POST'])
def update_quantity():
    name = request.json.get("name")
    action = request.json.get("action") # 'plus' або 'minus'
    product = products_collection.find_one({"name": name})

    if name in session['cart']:
        if action == 'plus' and session['cart'][name] < product['stock']:
            session['cart'][name] += 1
        elif action == 'minus' and session['cart'][name] > 1:
            session['cart'][name] -= 1
        elif action == 'minus' and session['cart'][name] == 1:
            session['cart'].pop(name)

        session.modified = True

    return jsonify({"cart_count": sum(session['cart'].values()), "cart":
session['cart']})

@app.route('/remove_from_cart', methods=['POST'])
def remove_from_cart():
    name = request.json.get("name") # тепер отримуємо назву, а не індекс
    if 'cart' in session and name in session['cart']:
        session['cart'].pop(name)

```

```

        session.modified = True

    return jsonify({"cart_count": sum(session['cart'].values()), "cart":
session.get('cart', {}))})

@app.route('/generate_invoice')
def generate_invoice():
    if 'user_id' not in session:
        return "Тільки зареєстровані користувачі можуть купувати", 403

    cart = session.get('cart', {})
    if not cart: return "Кошик порожній", 400

    products = list(products_collection.find({"name": {"$in":
list(cart.keys())}}))

    total_sum = 0
    items_for_history = []

    for item in products:
        name = item['name']
        qty = cart.get(name, 0)
        price = item['price']
        item_total = price * qty
        total_sum += item_total

    items_for_history.append({
        "name": name,
        "qty": qty,
        "price": price,
        "subtotal": item_total
    })

    # списання зі складу
    products_collection.update_one(
        {"name": name},
        {"$inc": {"stock": -qty}}

```

```

    )

# Зберігаємо замовлення в БД
order_data = {
    "user_id": session['user_id'],
    "username": session['username'],
    "items": items_for_history,
    "total": total_sum,
    "date": datetime.now()
}

orders_collection.insert_one(order_data)

pdf = FPDF()
pdf.add_page()

try:
    pdf.add_font('DejaVu', '', 'DejaVuSans.ttf', uni=True)
    pdf.set_font('DejaVu', '', 14)
except:
    pdf.set_font('Arial', '', 14)

# Заголовок
pdf.cell(200, 10, txt="ПАХУНОК-ФАКТУРА: Electro-Tech Solutions", ln=True,
align='C')

pdf.set_font('DejaVu', '', 10)
pdf.cell(200, 7, txt="Власник: Горбань Юрій | м. Івано-Франківськ", ln=True,
align='C')

pdf.ln(10)

# Шапка таблиці
pdf.cell(80, 10, "Назва товару", border=1)
pdf.cell(30, 10, "К-сть", border=1)
pdf.cell(40, 10, "Ціна за од.", border=1)
pdf.cell(40, 10, "Сума", border=1)

pdf.ln()

```

```
# Наповнення таблиці та розрахунок суми

total_sum = 0

for item in products:

    name = item['name']

    price = item['price']

    qty = cart.get(name, 0)

    item_total = price * qty


    pdf.cell(80, 10, name, border=1)

    pdf.cell(30, 10, f"{qty} од.", border=1)

    pdf.cell(40, 10, f"{price} грн", border=1)

    pdf.cell(40, 10, f"{item_total} грн", border=1)

    pdf.ln()

    total_sum += item_total


pdf.ln(5)

pdf.set_font('DejaVu', '', 14)

pdf.cell(190, 10, txt=f"ЗАГАЛЬНА СУМА: {total_sum} грн", ln=True, align='R')


# ГЕНЕРАЦІЯ QR-КОДУ

payment_info = f"Оплата замовлення. Отримувач: Горбань Юрій. Сума: {total_sum} грн. Дякуємо!"

qr = qrcode.QRCode(version=1, box_size=10, border=5)

qr.add_data(payment_info)

qr.make(fit=True)

qr_img = qr.make_image(fill_color="black", back_color="white")


# Зберігаємо QR в тимчасовий буфер для картинки

qr_buffer = BytesIO()

qr_img.save(qr_buffer, format='PNG')

qr_buffer.seek(0)


# Вставляємо QR в PDF
```

```

pdf.ln(10)

pdf.set_font('DejaVu', '', 10)

pdf.cell(0, 10, "Відскануйте для оплати або отримання реквізитів:", ln=True,
align='C')

# Малюємо QR-код по центру (x=80)

pdf.image(qr_buffer, x=80, y=pdf.get_y(), w=50)


# ФІНАЛІЗАЦІЯ: Збереження всього PDF в один буфер
output = io.BytesIO()

# використовуємо .encode('latin-1','replace') для сумісності fpdf
pdf_content = pdf.output(dest='S')

if isinstance(pdf_content, str):
    pdf_content = pdf_content.encode('latin-1')

output.write(pdf_content)
output.seek(0)


session['cart'] = {}

session.modified = True


return send_file(output, download_name="invoice_Gorban_QR.pdf",
as_attachment=True)


@app.route('/about')
def about():
    # Передаємо кількість товарів у кошику, щоб вона відображалася і там
    cart_count = sum(session.get('cart', {}).values()) if
isinstance(session.get('cart'), dict) else 0

    return render_template('about.html', cart_count=cart_count)


@app.route('/admin')
def admin_panel():
    if session.get('role') != 'admin':
        return redirect(url_for('index'))

```

```

all_prods = list(products_collection.find())
all_users = list(users_collection.find({}, {"password": 0}))

# Отримуємо замовлення
raw_orders = list(orders_collection.find().sort("date", -1))

serializable_orders = []
for order in raw_orders:
    order['_id'] = str(order['_id'])
    if 'date' in order:
        order['date'] = order['date'].isoformat()
    serializable_orders.append(order)

categories = sorted(list(set(p.get('category', 'None') for p in all_prods)))

return render_template('admin.html',
                       products=all_prods,
                       all_users=all_users,
                       orders=serializable_orders,
                       categories=categories)

@app.route('/admin/delete/<name>', methods=['POST'])
def admin_delete(name):
    if session.get('role') != 'admin':
        return redirect(url_for('index'))

    products_collection.delete_one({"name": name})
    return redirect(url_for('admin_panel'))

@app.route('/admin/update', methods=['POST'])
def admin_update():
    if session.get('role') != 'admin':
        return "Access denied", 403

```

```

name = request.form.get('name')

price = float(request.form.get('price'))

stock = int(request.form.get('stock'))

products_collection.update_one(
    {"name": name},
    {"$set": {"price": price, "stock": stock}}
)

return redirect(url_for('admin_panel'))

@app.route('/admin/add', methods=['POST'])
def admin_add():
    if session.get('role') != 'admin':
        return "Access denied", 403

    new_prod = {
        "name": request.form.get('name'),
        "category": request.form.get('category'),
        "price": float(request.form.get('price')),
        "stock": int(request.form.get('stock')),
        "description": request.form.get('description', '')
    }

    products_collection.insert_one(new_prod)

    return redirect(url_for('admin_panel'))

@app.route('/admin/logout')
def admin_logout():
    session.clear()

    return redirect(url_for('index'))

@app.route('/register', methods=['POST'])
def register():
    data = request.json

    username = data.get('username')

```

```

password = data.get('password')

if users_collection.find_one({"username": username}):
    return jsonify({"error": "Користувач вже існує"}), 400

user_data = {
    "username": username,
    "password": generate_password_hash(password),
    "role": "user", # Ролі: guest, user, admin
    "created_at": datetime.now()
}

users_collection.insert_one(user_data)

return jsonify({"success": "Реєстрація успішна"})

@app.route('/login', methods=['POST'])
def login():
    data = request.json

    user = users_collection.find_one({"username": data.get('username')})

    if user and check_password_hash(user['password'], data.get('password')):
        session['user_id'] = str(user['_id'])
        session['username'] = user['username']
        session['role'] = user.get('role', 'user')
        session.modified = True

        return jsonify({"success": "Вхід успішний", "role": session['role']})

    return jsonify({"error": "Невірні дані"}), 401

@app.route('/logout')
def logout():
    session.clear()

    return redirect(url_for('index'))

if __name__ == '__main__':

```

```
app.run(debug=True, host='0.0.0.0', port=5000)
```

А.3. Інтерфейс користувача та адміністратора (Templates)

Нижче наведено об'єднаний лістинг основних HTML-файлів, що відповідають за візуальне представлення та клієнтську логіку.

— Головна сторінка та чат-інтерфейс (index.html):

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Electro-Tech ERP & AI Assistant</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.10.0/font/bootstrap-icons.css">
  <style>
    :root { --primary-color: #2c3e50; --accent-color: #3498db; }
    body { background-color: #f0f2f5; font-family: 'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif; }
    .navbar { background-color: var(--primary-color) !important;
color: white; }
    .stat-card { border-radius: 15px; border: none; transition:
transform 0.2s; }
    .stat-card:hover { transform: translateY(-5px); }
    #chat-window { height: 500px; overflow-y: auto; background: white;
border-radius: 15px; border: 1px solid #dee2e6; }
    .user-msg { background: #e3f2fd; border-radius: 15px 15px 0 15px;
padding: 12px; margin-bottom: 10px; margin-left: auto; max-width: 80%; }
    .bot-msg { background: #f8f9fa; border-radius: 15px 15px 15px 0;
padding: 12px; margin-bottom: 10px; max-width: 80%; border-left: 4px solid var(-
-accent-color); }
    .quick-btn { border-radius: 20px; font-size: 0.8rem; margin: 2px;
}
    .instruction-panel { background: white; border-radius: 15px;
padding: 20px; }
  </style>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
  <nav class="navbar navbar-dark mb-4 shadow">
    <div class="container">
      <div class="d-flex align-items-center">
        <a href="/" class="navbar-brand"><i class="bi bi-
lightning-charge-fill"></i> Electro-Tech AI ERP</a>
        <a href="/about" class="nav-link text-white ms-3">Про
компанію</a>

        {% if session.get('role') == 'admin' %}
        <a href="/admin" class="nav-link text-warning ms-3"><i
class="bi bi-shield-lock"></i> Адмін-панель</a>
        {% endif %}
      </div>

      <div class="d-flex align-items-center">
        {% if session.get('username') %}
```

```

        <span class="text-white me-3">👤 {{
session['username'] }}</span>
        <button class="btn btn-warning me-2" data-bs-
toggle="modal" data-bs-target="#cartModal">
            <i class="bi bi-cart3"></i> Кошик <span id="cart-
count" class="badge bg-danger">{{ session.get('cart', {}).values()|sum }}</span>
        </button>
        <a href="/logout" class="btn btn-outline-light btn-
sm">Вихід</a>
        {% else %}
        <button class="btn btn-outline-light btn-sm me-2"
data-bs-toggle="modal" data-bs-target="#loginModal">Вхід</button>
        <button class="btn btn-warning btn-sm" data-bs-
toggle="modal" data-bs-target="#registerModal">Реєстрація</button>
        {% endif %}
    </div>
</div>
</nav>

<div class="row mb-4">
    <div class="col-md-4">
        <div class="card shadow-sm border-0 p-3 h-100 bg-white"
style="border-radius: 15px;">
            <div class="d-flex justify-content-between align-items-
center">
                <div>
                    <h6 class="text-muted small
uppercase">Капіталізація складу</h6>
                    <h3 class="text-primary fw-bold">{{
stats.total_value }} ₪</h3>
                </div>
                <div class="bg-light p-2 rounded">📈</div>
            </div>
            <hr>
            <div class="row text-center">
                <div class="col-4">
                    <small class="text-muted d-block">Товарів</small>
                    <span class="fw-bold">{{ stats.total_products
}}</span>
                </div>
                <div class="col-4 border-start">
                    <small class="text-muted d-
block">Категорій</small>
                    <span class="fw-bold text-success">{{
stats.categories }}</span>
                </div>
                <div class="col-4 border-start">
                    <small class="text-muted d-block">Дефіцит</small>
                    <span class="fw-bold text-danger">{{
stats.low_stock }}</span>
                </div>
            </div>
        </div>
    </div>
    <div class="col-md-4">
        <div class="card p-2 shadow-sm border-0" style="border-radius:
15px;">
            <canvas id="categoryChart" style="max-height:
180px;"></canvas>
        </div>
    </div>
</div>
<div class="col-md-4">

```

```

        <div class="card p-2 shadow-sm border-0" style="border-radius:
15px;">
            <canvas          id="topPriceChart"          style="max-height:
180px;"></canvas>
        </div>
    </div>
</div>

<div class="row">
    <div class="col-lg-8">
        <div class="card shadow-sm border-0 mb-4" style="border-
radius: 15px;">
            <div class="card-header bg-white font-weight-
bold">Інтелектуальний консультант</div>
            <div class="card-body">
                <div id="chat-window" class="p-3 mb-3">
                    <div class="bot-msg">👋 Вітаю! Я допоможу
підібрати обладнання. Ви можете запитати мене або використати швидкі команди
нижче.</div>

                    </div>

                    <div class="mb-3">
                        <button class="btn btn-outline-secondary quick-
btn" onclick="quickAsk('/all')">📦 Весь склад</button>
                        <button class="btn btn-info text-white quick-btn"
data-bs-toggle="modal" data-bs-target="#lightModal">💡 Розрахувати
світло</button>

                        <button class="btn btn-outline-secondary quick-
btn" onclick="quickAsk('Який кабель вибрати для розеток?')">🔌 Для
розеток</button>

                        <button class="btn btn-outline-secondary quick-
btn" onclick="quickAsk('Що потрібно для захисту від перенапруги?')">🛡️
Захист</button>

                    </div>

                    <div class="input-group">
                        <input type="text" id="user-input" class="form-
control" placeholder="Напишіть ваше питання...">
                        <button class="btn btn-primary"
onclick="sendMessage()"><i class="bi bi-send"></i></button>
                    </div>
                </div>
            </div>
        </div>

    <div class="col-lg-4">
        <div class="instruction-panel shadow-sm mb-4">
            <h5><i class="bi bi-book"></i> База знань</h5>
            <hr>
            <div class="accordion" id="manuals">
                <div class="accordion-item">
                    <h2 class="accordion-header"><button
class="accordion-button collapsed" data-bs-toggle="collapse" data-bs-
target="#m1">🔧 Монтаж кабелю</button></h2>
                    <div id="m1" class="accordion-collapse collapse"
data-bs-parent="#manuals">
                        <div class="accordion-body small">Для розеток
– 2.5 мм², для світла – 1.5 мм². Глибина штробы – мінімум 2 см.</div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

        <h2
class="accordion-button
collapsed"
data-bs-toggle="collapse"
data-bs-target="#m2">⚡ Вибір автомата</h2>
        <div id="m2" class="accordion-collapse collapse"
data-bs-parent="#manuals">
            <div class="accordion-body
small">10А
(світло), 16А (розетки), 25А (потужні прилади). Не ставте 25А на кабель
1.5мм!</div>
            </div>
        </div>
        <div class="accordion-item">
            <h2
class="accordion-button
collapsed"
data-bs-toggle="collapse"
data-bs-target="#m3">⚡ Ванна кімната</h2>
            <div id="m3" class="accordion-collapse collapse"
data-bs-parent="#manuals">
                <div class="accordion-body
small">Обов'язково
ПЗВ (30мА) та розетки із захисними шторками (IP44).</div>
            </div>
        </div>
        <div class="accordion-item">
            <h2
class="accordion-button
collapsed"
data-bs-toggle="collapse"
data-bs-target="#m4">💡 Колірна температура</h2>
            <div id="m4" class="accordion-collapse collapse"
data-bs-parent="#manuals">
                <div class="accordion-body
small">2700К –
тепле світло, 4100К – нейтральне (денне), 6500К – холодне біле.</div>
            </div>
        </div>
        <div class="accordion-item">
            <h2
class="accordion-button
collapsed"
data-bs-toggle="collapse"
data-bs-target="#m5">📋 Маркування ВВГнг-П</h2>
            <div id="m5" class="accordion-collapse collapse"
data-bs-parent="#manuals">
                <div class="accordion-body
small">В – вініл, В
– вініл, Г – голий (без броне), нг – не горючий, П – плоский.</div>
            </div>
        </div>
    </div>
    <div>
        <div class="card shadow-sm p-3 bg-light border-0">
            <h6><i
class="bi
bi-telephone-outbound"></i>
Техпідтримка</h6>
            <p class="small text-muted">Графік: Пн-Пт 9:00 –
18:00<br>Тел: 0-800-123-456</p>
        </div>
    </div>
    <div>
        <div class="modal fade" id="cartModal" tabindex="-1">
            <div class="modal-dialog modal-lg">
                <div class="modal-content" style="border-radius: 15px;">
                    <div class="modal-header"><h5>Ваш кошик замовлень</h5><button
type="button" class="btn-close" data-bs-dismiss="modal"></button></div>
                    <div class="modal-body">
                        <div id="cart-items" class="list-group mb-3">Кошик
порожній</div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

        <div class="alert alert-info small">Після завантаження
рахунку, зверніться до менеджера для оплати.</div>
    </div>
    <div class="modal-footer">
        <a href="/generate_invoice" class="btn btn-success">
            <i class="bi bi-file-earmark-pdf"></i> Завантажити
рахунок (PDF)
        </a>
    </div>
</div>
</div>
</div>
</div>

```

```

<div class="modal fade" id="lightModal" tabindex="-1">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header"><h5>Калькулятор
освітлення</h5><button type="button" class="btn-close" data-bs-
dismiss="modal"></button></div>
            <div class="modal-body">
                <label>Площа кімнати (м²):</label>
                <input type="number" id="roomArea" class="form-control mb-
3" placeholder="Наприклад: 20">
                <label>Тип приміщення:</label>
                <select id="roomType" class="form-control mb-3">
                    <option value="150">Житлова кімната (150 Лк)</option>
                    <option value="300">Кухня/Кабінет (300 Лк)</option>
                    <option value="50">Коридор (50 Лк)</option>
                </select>
                <button class="btn btn-primary w-100"
onclick="calcLight()">Отримати рекомендацію</button>
                <div id="lightResult" class="mt-3 p-3 bg-light rounded d-
none"></div>
            </div>
        </div>
    </div>
</div>
</div>

```

```

<div class="modal fade" id="loginModal" tabindex="-1">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header"><h5>Вхід у систему</h5><button
type="button" class="btn-close" data-bs-dismiss="modal"></button></div>
            <div class="modal-body">
                <input type="text" id="login-user" class="form-control mb-
2" placeholder="Логін">
                <input type="password" id="login-pass" class="form-control
mb-2" placeholder="Пароль">
                <button class="btn btn-primary w-100"
onclick="authAction('login')">Увійти</button>
            </div>
        </div>
    </div>
</div>

```

```

<div class="modal fade" id="registerModal" tabindex="-1">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header"><h5>Реєстрація</h5><button
type="button" class="btn-close" data-bs-dismiss="modal"></button></div>

```

```

        <div class="modal-body">
            <input type="text" id="reg-user" class="form-control mb-2"
placeholder="Придумайте логін">
            <input type="password" id="reg-pass" class="form-control
mb-2" placeholder="Придумайте пароль">
            <button class="btn btn-warning w-100"
onclick="authAction('register')">Зареєструватися</button>
        </div>
    </div>
</div>
</div>

```

```

<script>
    async function sendMessage() {
        const input = document.getElementById('user-input');
        const chat = document.getElementById('chat-window');
        if (!input.value) return;
        const msg = input.value;
        chat.innerHTML += `<div class="user-msg">${msg}</div>`;
        input.value = '';

        const response = await fetch('/ask', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ message: msg })
        });
        const data = await response.json();
        if (data.type === "table") { renderTable(data.data); }
        else { chat.innerHTML += `<div class="bot-
msg">${data.reply}</div>`; }
        chat.scrollTop = chat.scrollHeight;
    }

    function quickAsk(text) {
        document.getElementById('user-input').value = text;
        sendMessage();
    }

    function renderTable(products) {
        const chat = document.getElementById('chat-window');
        const isGuest = ({ 'true' if not session.get('username') else 'false'
});
        let html = `
        <div class="bot-msg shadow-sm">
            <div class="d-flex justify-content-between align-items-center mb-
2">
                <b>📦 Складські залишки:</b>
                <input type="text" id="tableSearch" class="form-control form-
control-sm w-50"
                    placeholder="Швидкий пошук..." onkeyup="filterTable()">
            </div>
            <table class="table table-sm table-hover" id="productTable">
                <thead class="table-light">
                    <tr><th>Назва</th><th>Ціна</th><th>Дія</th></tr>
                </thead>
                <tbody>`;

        products.forEach(p => {
            let actionButton = isGuest
                ? `<button class="btn btn-sm btn-outline-secondary" disabled
title="Увійдіть, щоб купити">🔒</button>`

```

```

        : `<button onclick="addToCart('${p.name}')" class="btn btn-sm
btn-success">🛒</button>`;

        html += `<tr>
            <td>${p.name}                <br><small                class="text-
muted">${p.category}</small></td>
            <td><b>${p.price}</b></td>
            <td>${actionButton}</td>
        </tr>`;
    });
    html += `</tbody></table></div>`;
    chat.innerHTML += html;
    chat.scrollTop = chat.scrollHeight;
}

// функція фільтрації
function filterTable() {
    let                                input                                =
document.getElementById("tableSearch").value.toUpperCase();
    let table = document.getElementById("productTable");
    let tr = table.getElementsByTagName("tr");

    for (let i = 1; i < tr.length; i++) {
        let td = tr[i].getElementsByTagName("td")[0];
        if (td) {
            let textValue = td.textContent || td.innerText;
            tr[i].style.display = textValue.toUpperCase().indexOf(input) >
-1 ? "" : "none";
        }
    }
}

async function addToCart(name) {
    const res = await fetch('/add_to_cart', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ name: name })
    });
    const data = await res.json();
    if (data.error) {
        alert(data.error);
    } else {
        updateCartUI(data);
    }
}

async function removeFromCart(name) {
    const res = await fetch('/remove_from_cart', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ name: name })
    });
    const data = await res.json();
    updateCartUI(data);
}

async function changeQty(name, action) {
    const res = await fetch('/update_quantity', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ name: name, action: action })
    });
    const data = await res.json();
}

```

```

    if (data.error) {
        alert(data.error);
    } else {
        updateCartUI(data);
    }
}

function updateCartUI(data) {
    document.getElementById('cart-count').innerText = data.cart_count;
    const cartList = document.getElementById('cart-items');
    const cartEntries = Object.entries(data.cart || {});

    if (cartEntries.length === 0) {
        cartList.innerHTML = "Кошик порожній";
    } else {
        cartList.innerHTML = cartEntries.map(([name, qty]) => `
            <div class="list-group-item d-flex justify-content-between align-items-center">
                <div style="flex: 1;"><strong>${name}</strong></div>
                <div class="d-flex align-items-center">
                    <button class="btn btn-sm btn-outline-secondary px-2"
onclick="changeQty('${name}', 'minus')">-</button>
                    <span class="mx-3 fw-bold">${qty}</span>
                    <button class="btn btn-sm btn-outline-secondary px-2"
onclick="changeQty('${name}', 'plus')">+</button>
                </div>
                <button class="btn btn-sm btn-danger ms-3"
onclick="removeFromCart('${name}')"><i class="bi bi-trash"></i></button>
            </div>
        `).join('');
    }
}

function calcLight() {
    const area = document.getElementById('roomArea').value;
    const lux = document.getElementById('roomType').value;
    const totalLumens = area * lux;
    // припустимо, середня лампа дає 1000 Лм (12W LED)
    const lampsNeeded = Math.ceil(totalLumens / 1000);

    const resDiv = document.getElementById('lightResult');
    resDiv.classList.remove('d-none');
    resDiv.innerHTML = `Для площі ${area}м² потрібно приблизно
<b>${lampsNeeded}</b> ламп(и)</b> по 12W.<br>
                <button class="btn btn-sm btn-outline-primary mt-2"
onclick="quickAsk('Покажи мені LED лампи 12W')">Переглянути в
магазині</button>`;
}

// Дані з Flask в JS
const dashboardData = ({ stats|tojson|safe });

// перевіряємо, чи завантажилась бібліотека Chart.js перед використанням
if (typeof Chart !== 'undefined') {
    new Chart(document.getElementById('categoryChart'), {
        type: 'doughnut',
        data: {
            labels: dashboardData.cat_labels,
            datasets: [{
                data: dashboardData.cat_values,
                backgroundColor: ['#FF6384', '#36A2EB', '#FFCE56',
'#4BC0C0', '#9966FF']
            }]
        }
    });
}

```

```

    },
    options: {
        responsive: true,
        plugins: { title: { display: true, text: 'Розподіл за
категоріями' } }
    }
});

new Chart(document.getElementById('topPriceChart'), {
    type: 'bar',
    data: {
        labels: dashboardData.top_names,
        datasets: [{
            label: 'Ціна (€)',
            data: dashboardData.top_prices,
            backgroundColor: '#36A2EB'
        }]
    },
    options: {
        responsive: true,
        plugins: { title: { display: true, text: 'Топ-5 найдорожчих
товарів' } }
    }
});
} else {
    console.error("Бібліотека Chart.js не завантажена! Перевірте
підключення у <head>");
}

async function authAction(type) {
    const user = document.getElementById(type === 'login' ? 'login-user' :
'reg-user').value;
    const pass = document.getElementById(type === 'login' ? 'login-pass' :
'reg-pass').value;

    const response = await fetch(`/${type}`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ username: user, password: pass })
    });

    const data = await response.json();
    if (data.success) {
        alert(data.success);
        location.reload(); // перезавантажуємо сторінку, щоб побачити
зміни (ім'я користувача, кошик)
    } else {
        alert(data.error);
    }
}

</script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.j
s"></script>
</body>
</html>

```

— Панель адміністратора (admin.html):

```

<!DOCTYPE html>
<html lang="uk">

```

```

<head>
  <meta charset="UTF-8">
  <title>Адмінка</title>
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivrivr.net/npm/bootstrap-
icons@1.10.0/font/bootstrap-icons.css">
</head>
<body class="bg-light">
<div class="container py-5">
  <div class="d-flex justify-content-between align-items-center mb-4 p-3
bg-white shadow-sm rounded">
    <h1>Панель Адміністратора</h1>
    <div>
      <a href="/" class="btn btn-outline-secondary me-2">На сайт</a>
      <a href="/admin/logout" class="btn btn-danger">Вийти</a>
    </div>
  </div>

  <div class="card mb-5 shadow-sm">
    <div class="card-header bg-primary text-white">Додати новий
товар</div>
    <div class="card-body">
      <form action="/admin/add" method="post" class="row g-3">
        <div class="col-md-3">
          <input type="text" name="name" class="form-control"
placeholder="Назва товару" required>
        </div>
        <div class="col-md-2">
          <input type="text" name="category" list="categoryList"
class="form-control" placeholder="Категорія" required>
          <datalist id="categoryList">
            {% for cat in categories %}
            <option value="{{ cat }}">
            {% endfor %}
          </datalist>
        </div>
        <div class="col-md-2">
          <input type="number" step="0.01" name="price"
class="form-control" placeholder="Ціна" required>
        </div>
        <div class="col-md-2">
          <input type="number" name="stock" class="form-control"
placeholder="К-сть" required>
        </div>
        <div class="col-md-3">
          <button type="submit" class="btn btn-success w-
100">Додати в базу</button>
        </div>
      </form>
    </div>
  </div>

  <div class="card shadow-sm mb-5">
    <div class="card-header bg-white fw-bold">Управління складом</div>
    <div class="table-responsive">
      <table class="table table-hover align-middle mb-0">
        <thead class="table-light">
          <tr>
            <th>Товар</th>
            <th style="width: 150px;">Ціна (грн)</th>
            <th style="width: 150px;">На складі</th>

```

```

        <th style="width: 200px;">Дія</th>
    </tr>
</thead>
<tbody>
    {% for p in products %}
    <tr>
        <form      action="/admin/update"      method="post"
id="update-{{ loop.index }}">
            <input  type="hidden"  name="name"  value="{{
p.name }}">
            <td><span      class="fw-bold">{{      p.name
}}</span><br><small class="text-muted">{{ p.category }}</small></td>
            <td><input      type="number"      step="0.01"
name="price" value="{{ p.price }}" class="form-control form-control-sm"></td>
            <td><input      type="number"      name="stock"
value="{{ p.stock }}" class="form-control form-control-sm"></td>
            <td>
                <div class="btn-group w-100">
                    <button type="submit" class="btn btn-
sm btn-warning">Оновити</button>
                    <button
                        type="submit"
formaction="/admin/delete/{{ p.name }}"
                        class="btn btn-sm btn-danger"
                        onclick="return
confirm('Видалити {{ p.name }}?')">❌</button>
                </div>
            </td>
        </form>
    </tr>
    {% endfor %}
</tbody>
</table>
</div>
</div>

<div class="card shadow-sm border-dark">
    <div class="card-header bg-dark text-white d-flex justify-content-
between align-items-center">
        <span>Зареєстровані користувачі</span>
        <span class="badge bg-secondary">{{ all_users|length }}
ociб</span>
    </div>
    <div class="card-body">
        <table class="table table-sm">
            <thead>
                <tr>
                    <th>Користувач</th>
                    <th>Роль</th>
                    <th>Дія</th>
                </tr>
            </thead>
            <tbody>
                {% for u in all_users %}
                <tr>
                    <td>{{ u.username }}</td>
                    <td>
                        <span class="badge {{ 'bg-danger' if u.role ==
'admin' else 'bg-info text-dark' }}">{{ u.role }}
                    </span>
                    </td>
                    <td>

```

[illegible]

```

        <td>${i.price} ₾</td>
        <td>${i.qty}</td>
    </tr>
    `).join('')
</tbody>
</table>
</div>
</div>`;
    });
    body.innerHTML = html;
}

// відкриваємо модальку
new
bootstrap.Modal(document.getElementById('historyModal')).show();
}
</script>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

— Інформаційна сторінка (about.html):

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Про компанію - Electro-Tech</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.10.0/font/bootstrap-icons.css">
    <style>
        :root { --primary-color: #2c3e50; }
        .navbar { background-color: var(--primary-color) !important; }
    </style>
</head>
<body>

    <nav class="navbar navbar-dark mb-4 shadow">
        <div class="container">
            <div class="d-flex align-items-center">
                <a href="/" class="navbar-brand"><i class="bi bi-lightning-
charge-fill"></i> Electro-Tech AI ERP</a>
                <a href="/about" class="nav-link text-white ms-3 active">Про
компанію</a>
            </div>
            <button class="btn btn-warning position-relative"
onclick="window.location.href='/'">
                <i class="bi bi-cart3"></i> На головну (до кошика)
            </button>
        </div>
    </nav>

    <div class="container">
        <div class="bg-white border rounded shadow-sm p-5 mb-4">
            <div class="row align-items-center">
                <div class="col-md-7">

```

```

        <h1 class="display-5 fw-bold text-primary">Electro-Tech
Solutions</h1>
        <p class="lead text-muted">Ваш надійний партнер у світі
електротехніки в Івано-Франківську.</p>
        <div class="row mt-4">
            <div class="col-6">
                <h6><i class="bi bi-person-badge text-
primary"></i> Власник:</h6>
                <p class="fw-bold">Горбань Юрій</p>
            </div>
            <div class="col-6">
                <h6><i class="bi bi-geo-alt text-danger"></i>
Локація:</h6>
                <p class="fw-bold">м. Івано-Франківськ, Центр</p>
            </div>
        </div>
        <hr>
        <p>Ми спеціалізуємося на комплексному забезпеченні
об'єктів будь-якої складності: від квартир до промислових підприємств. Понад 10
років досвіду на ринку України. Наш ШІ-консультант допоможе вам підібрати
обладнання за лічені секунди.</p>
        <a href="tel:+380000000000" class="btn btn-primary mt-
3"><i class="bi bi-telephone"></i> Зв'язатися з Юрієм</a>
    </div>
    <div class="col-md-5">
        <div class="rounded shadow-sm overflow-hidden border"
style="height: 350px;">
            <iframe
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d83856.4172108785!2d24
.63973792945228!3d48.92429406085521!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!
1m2!1s0x4730c16c34b0381d%3A0xd2ad3cc8079c647!2z0IbQstCw0L3Qvi3QpNGA0LDQvdC60ZbQs
tGB0YzQuiwg0IbQstCw0L3Qvi3QpNGA0LDQvdC60ZbQstGB0YzQutCwINC-
0LHQu9Cw0YHRgtGMLCA3NjAwMA!5e0!3m2!1suk!2sua!4v1715431234567!5m2!1suk!2sua"
width="100%" height="100%" style="border:0;"
allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-when-downgrade">
            </iframe>
        </div>
    </div>
</div>
</div>
</div>
</div>
</body>
</html>

```

A.4. Скрипт ініціалізації бази даних (seed_db.py)

Скрипт призначений для автоматичного створення колекцій та наповнення їх початковими даними про електротовари.

```

import os
from pymongo import MongoClient
from dotenv import load_dotenv

load_dotenv()
client = MongoClient(os.getenv("MONGO_URI"))
db = client['electrical_db']
products_collection = db['products']

```

```
products_collection.delete_many({})
```

```
all_products = [  
    {"name": "Кабель ВВГнг-П 3х2.5", "category": "Кабелі", "price": 48.50,  
"stock": 200, "description": "Для розеток."},  
    {"name": "Кабель ВВГнг-П 3х1.5", "category": "Кабелі", "price": 32.00,  
"stock": 500, "description": "Для освітлення."},  
    {"name": "Провід ПВС 2х1.5", "category": "Кабелі", "price": 28.00, "stock":  
150, "description": "Гнучкий провід."},  
    {"name": "Автоматичний вимикач 16A (Schneider)", "category": "Автоматика",  
"price": 145.00, "stock": 45, "description": "Захист ліній."},  
    {"name": "Автоматичний вимикач 10A (Eaton)", "category": "Автоматика",  
"price": 120.00, "stock": 30, "description": "Захист світла."},  
    {"name": "LED Лампа E27 12W 4100K", "category": "Освітлення", "price":  
85.00, "stock": 100, "description": "Нейтральне світло."},  
    {"name": "Вуличний прожектор LED 30W IP65", "category": "Освітлення",  
"price": 450.00, "stock": 12, "description": "Вологозахисний."},  
    {"name": "Розетка одинарна із заземленням", "category": "Фурнітура",  
"price": 95.00, "stock": 80, "description": "Зі шторками."},  
    {"name": "Диференційне реле (ПЗВ) 25А 30мА", "category": "Автоматика",  
"price": 850.00, "stock": 8, "description": "Захист від струму."},  
    {"name": "Коробка розподільча 100х100 IP54", "category": "Монтаж", "price":  
42.00, "stock": 60, "description": "Для комутації."},  
  
    {"name": "Реле напруги ZUBR D2-40", "category": "Автоматика", "price":  
1100.0, "stock": 15, "description": "Захист від стрибків напруги."},  
    {"name": "Кабель ВВГнг-П 3х4.0", "category": "Кабелі", "price": 75.0,  
"stock": 100, "description": "Для електроплит."},  
    {"name": "Провід ПВ-3 6.0 (жовто-зелений)", "category": "Кабелі", "price":  
45.0, "stock": 300, "description": "Для заземлення."},  
    {"name": "Вимикач двоклавішний білий", "category": "Фурнітура", "price":  
110.0, "stock": 50, "description": "Для двох груп світла."},  
    {"name": "Диммер 600W", "category": "Фурнітура", "price": 320.0, "stock":  
10, "description": "Регулювання яскравості."},  
    {"name": "Щит електричний на 24 модулі", "category": "Автоматика", "price":  
680.0, "stock": 5, "description": "Внутрішній монтаж."},  
    {"name": "Шина нульова 7 отворів", "category": "Монтаж", "price": 45.0,  
"stock": 100, "description": "Для щитка."},  
    {"name": "Гофротруба d20 (50м)", "category": "Монтаж", "price": 350.0,  
"stock": 20, "description": "З протяжкою."},  
    {"name": "Клемник WAGO 221-413 (50шт)", "category": "Монтаж", "price":  
750.0, "stock": 15, "description": "Експрес-монтаж."},  
    {"name": "Стрічка ізоляційна 3М", "category": "Монтаж", "price": 55.0,  
"stock": 200, "description": "Професійна ізоляція."},  
    {"name": "LED Стрічка 12V 5050 RGB", "category": "Освітлення", "price":  
120.0, "stock": 60, "description": "Кольорова підсвітка."},  
    {"name": "Блок живлення 12V 100W", "category": "Освітлення", "price": 450.0,  
"stock": 12, "description": "Для LED стрічки."},  
    {"name": "Мультиметр цифровий UT33D", "category": "Інструменти", "price":  
520.0, "stock": 8, "description": "Вимірювання напруги."},  
    {"name": "Обтискні кліщі (кримпер)", "category": "Інструменти", "price":  
890.0, "stock": 5, "description": "Для наконечників."},  
    {"name": "Вилка електрична з кільцем", "category": "Фурнітура", "price":  
45.0, "stock": 100, "description": "Зручне виймання."},  
    {"name": "Дзвінок безпроводний", "category": "Фурнітура", "price": 280.0,  
"stock": 20, "description": "Радіус 100м."},  
    {"name": "Тепла підлога (мат) 2м2", "category": "Опалення", "price": 2400.0,  
"stock": 3, "description": "Під плитку."},  
    {"name": "Терморегулятор цифровий", "category": "Опалення", "price": 950.0,  
"stock": 7, "description": "Для теплої підлоги."},  
    {"name": "Силовa розетка 32A (3-фазна)", "category": "Фурнітура", "price":  
480.0, "stock": 10, "description": "Для обладнання."},  
]
```

```
        {"name": "Кабель-канал 25x16 (2м)", "category": "Монтаж", "price": 42.0,  
"stock": 150, "description": "Білий пластик."}  
]
```

```
products_collection.insert_many(all_products)  
print(f"Базу оновлено! Тепер у нас {len(all_products)} товарів.")
```

A.5. Конфігурація середовища та залежностей (requirements.txt)

```
flask  
pymongo  
google-generativeai  
python-dotenv  
fpdf  
qrcode  
pillow
```

ДОДАТОК Б. Специфікація бази даних

Б.1. Загальні відомості

Для реалізації системи обрано документоорієнтовану СУБД NoSQL-типу MongoDB (версія Cloud Atlas або локальний сервер). Назва бази даних – electrical_db. Дані зберігаються у форматі BSON (Binary JSON).

Б.2. Опис колекцій бази даних

Нижче наведено специфікації основних колекцій, що забезпечують функціонування вебдодатка.

Таблиця 5.2 – Специфікація колекції products (Каталог товарів)

Назва поля	Тип даних (BSON)	Опис призначення
_id	ObjectId	Унікальний ідентифікатор документа (Primary Key)
name	String	Повна назва електротехнічного виробу (унікальна)
category	String	Категорія товару (автомати, кабелі, щитки тощо)
price	Double	Актуальна роздрібна ціна за одиницю (грн)
stock	Int32	Кількість одиниць товару на складі
description	String	Технічні характеристики для аналізу III Gemini

Таблиця 5.3 – Специфікація колекції users (Користувачі системи)

Назва поля	Тип даних (BSON)	Опис призначення
_id	ObjectId	Унікальний ідентифікатор користувача
username	String	Логін користувача (унікальний індекс)
password	String	Хешований пароль (алгоритм PBKDF2/sha256)

role	String	Рівень доступу: user (клієнт) або admin (адмін)
created_at	Date	Дата та час реєстрації облікового запису

Таблиця 5.4 – Специфікація колекції orders (Журнал замовлень)

Назва поля	Тип даних (BSON)	Опис призначення
_id	ObjectId	Унікальний номер замовлення
user_id	String	Логічне посилання на _id з колекції users
username	String	Ім'я клієнта (для швидкої ідентифікації)
items	Array (Object)	Список куплених товарів (денормалізовані дані)
total	Double	Загальна сума замовлення
date	Date	Дата та час проведення транзакції

Б.3. Структура вкладеного об'єкта в масиві items

Для забезпечення цілісності історії продажів (навіть у разі зміни цін у майбутньому), у колекції orders використовується денормалізація. Кожен елемент масиву items містить такі поля:

- name (String) – назва товару на момент купівлі;
- qty (Int32) – кількість придбаних одиниць;
- price (Double) – ціна, що була актуальною в момент замовлення;
- subtotal (Double) – розрахована вартість позиції ($\text{price} * \text{qty}$).

ДОДАТОК В. Сертифікат наукової публікації

Результати проведеного дослідження щодо розроблення та впровадження системи на базі MongoDB були представлені на VI Міжнародній науково-практичній конференції «Science in Motion: Classic and Modern Tools and Methods in Scientific Investigations» (20 березня 2026 р.). За матеріалами доповіді опубліковано наукову статтю у фаховому журналі «Grail of Science» (№ 64), що підтверджується відповідним сертифікатом (рис. 5.2).



CERTIFICATE

OF PARTICIPATION AND PUBLICATION



Yurii Horban

participated in the VI Correspondence International Scientific and Practical Conference

Science in Motion: Classic and Modern Tools and Methods in Scientific Investigations

held on March 20th, 2026 by

NGO European Scientific Platform (Vinnytsia, Ukraine)
LLC International Centre Corporate Management (Vienna, Austria)

and published scientific paper

**ОБґРУНТУВАННЯ ВИБОРУ НЕРЕЛЯЦІЙНОЇ МОДЕЛІ ДАНИХ НА БАЗІ MONGODB
ДЛЯ ОБЛІКУ ПРОДУКЦІЇ ТА КУПІВЕЛЬНОЇ АКТИВНОСТІ**

in Periodical scientific journal «**GRAIL OF SCIENCE**»

№ **64**; ISSN 2710-3056; Media identifier R30-02704;
DOI 10.36074/grail-of-science.20.03.2026





0.6 ECTS credits (18 hours)
Recommended by the Academic Council of the «Institute of Scientific and Technical Integration and Cooperation».
Protocol № 10 from March 19th, 2026.

Head of the
NGO «European Scientific Platform»
Chairman of the Organizing committee
GOLDENBLAT MIRIAM



Head of Community Outreach at the
LLC «International Centre Corporate Management»
RACHAEL APARO







Рисунок 5.2 - Сертифікат наукової публікації

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи – «Розроблення інформаційної системи з інтеграцією ШІ для пошуку та рекомендацій електротехнічної продукції підприємства»

Обсяг пояснювальної записки в аркушах – 104

Перелік креслень графічної частини:

КРБ.ІСТ – 05.00.00.000 С1 – Варіанти використання вебдодатку “electro_advisor”. UML-діаграма (1 аркуш)

КРБ.ІСТ – 05.00.00.000 С1 – База даних вебдодатку “electro_advisor”.
Схема структурна (1 аркуш)

КРБ.ІСТ – 05.00.00.001 – Вебдодаток “electro_advisor”. Робочі вікна (1 аркуш)

КРБ.ІСТ – 05.00.00.002 – Вебдодаток “electro_advisor”. Робочі вікна (1 аркуш)

Дата закінчення бакалаврської роботи «07» червня 2026 р.

Студент _____ Горбань Ю.О.