

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.І.

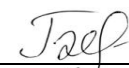
" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему: "Розроблення web-застосунку для продажу автоінструментів"

Виконав: студент групи

 Трофименко Б.О.
(підпис)

Керівник роботи:

(посада, науковий ступінь вчене звання)
 Меліксетов О.І.
(підпис)

м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 122 "Комп'ютерні науки"
Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

к.т.н, доц. Гайда А.Ю.

" ____ " ____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "бакалавр"

Студенту Трофименку Богдану Олександровичу

(прізвище, ім'я, по батькові)

1. Тема роботи: "Розроблення web-застосунку для продажу автоінструментів"

Керівник роботи Меліксетов Олег Ігорович

Затверджені наказом ректора № 295-уч від "27" березня 2025 року.

2. Термін подання роботи: 20 червня 2025 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів) Перелік питань, що належать до розробки (найменування розділів) аналіз предметної галузі, постановка задачі, розробка проекту інформаційної підсистеми, реалізація проекту підсистеми, охорона праці

5. Перелік презентаційних матеріалів Титульний лист, лист з формулюванням мети та завдань роботи, лист з аналізом існуючих систем і описом запропонованого рішення, лист із діаграмою використання, лист з архітектурною схемою вебсайту, лист з концептуальною моделлю бази даних, лист з діаграмою послідовності, лист з діаграмою компонентів, лист із результатами реалізації, Лист «Дякую за увагу»

6. Консультанти розділів роботи

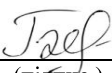
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Меліксетов О.І., викладач	03.02.2025	22.02.2025
2	Меліксетов О.І., викладач	22.02.2025	21.03.2025
3	Меліксетов О.І., викладач	21.03.2025	25.04.2025
4	Меліксетов О.І., викладач	25.04.2025	07.06.2025

7. Дата видачі завдання 03 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	13.02.2025	
2	Постановка задачі	15.02.2025	
3	Розробка концепції	22.02.2025	
4	Розробка проекту ІС	21.03.2025	
5	Реалізація проєкту	25.04.2025	
6	Розробка питань з охорони праці	07.06.2025	
7	Розробка графічних матеріалів	12.06.2025	
8	Подання роботи на перевірку на плагіат	10.06.2025	
9	Подання роботи рецензенту	15.06.2025	
10	Подання роботи на захист	20.06.2025	

Студент


(підпис)

Трофименко Б.О.
(ПІБ)

Керівник роботи


(підпис)

Меліксетов О.І.
(ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці веб-орієнтованої інформаційної системи для автоматизації управління каталогом автоінструментів. Основною метою дослідження є створення зручного, доступного та ефективного веб-додатку, що дозволяє додавати, редагувати, видаляти та переглядати інформацію про товари, здійснювати пошук, фільтрацію та сортування, а також забезпечує інтуїтивно зрозумілий користувацький інтерфейс.

У роботі проведено аналіз предметної області, визначено функціональні та нефункціональні вимоги до системи, обґрунтовано вибір архітектури клієнт–сервер із реалізацією шаблону MVC. Для реалізації системи використано технології Python (Flask), SQLite, HTML5, CSS3 та Jinja2. База даних складається з однієї таблиці, яка зберігає повну інформацію про автоінструменти.

Реалізовано три основні сторінки інтерфейсу: перегляд товарів, додавання нових записів, редагування наявних. Проведено тестування системи за ключовими сценаріями, що підтвердило її функціональність, адаптивність та зручність у використанні. Оцінено безпеку, масштабованість і можливості подальшого розвитку системи.

Результати роботи можуть бути використані для створення комерційних веб-рішень у сфері електронної торгівлі та адаптовані під інші предметні області.

Ключові слова: веб-система, автоінструменти, Flask, база даних, інтерфейс, автоматизація, UI/UX.

ABSTRACT

The qualification thesis is dedicated to the development of a web-oriented information system for automating the management of an automotive tools catalog. The main objective of the research is to create a convenient, accessible, and efficient web application that enables adding, editing, deleting, and viewing product information, as well as performing search, filtering, and sorting, all through an intuitive user interface.

The work includes an analysis of the subject domain, definition of functional and non-functional system requirements, and justification for choosing a client-server architecture based on the MVC pattern. The system is implemented using Python (Flask), SQLite, HTML5, CSS3, and Jinja2. The database consists of a single table that stores complete information about the automotive tools.

Three core user interface pages have been implemented: product catalog view, add new item, and edit existing entry. The system has been tested against key scenarios, confirming its functionality, responsiveness, and user-friendliness. Security, scalability, and further development potential were also evaluated.

The results of the project can be used to create commercial web solutions in the field of e-commerce and can be adapted to other subject areas.

Keywords: web system, automotive tools, Flask, database, interface, automation, UI/UX.

Зміст

ВСТУП.....	8
РОЗДІЛ І. РОЗРОБКА КОНЦЕПЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПРОДАЖУ АВТОІНСТРУМЕНТІВ.....	10
1.1 Опис об'єкта дослідження аналіз предметної області.....	10
1.2 Аналіз предметної області	12
1.3. Формування вимог до інформаційної систем	13
РОЗДІЛ ІІ. РОЗРОБКА ПРОЕКТУ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ.....	17
2.1 Розробка концепції інформаційної систем	17
2.1.2 Архітектурні рішення.....	20
2.1.3 Проектування бази даних	22
2.1.4 Розробка ці/ух	28
2.2 Реалізація проекту підсистеми	35
2.2.1 Реалізація фронтенду	35
2.2.2 Реалізація бекенду	40
2.2.3 Тестування.....	44
ОЦІНКА ПРОДУКТИВНОСТІ ТА ЕФЕКТИВНОСТІ СИСТЕМИ.....	47
БЕЗПЕКА СИСТЕМИ.....	48
РОЗДІЛ ІІІ. РЕАЛІЗАЦІЯ ПРОЕКТУ ПІДСИСТЕМИ.....	50
3.1 Надійність	50
3.2 Масштабованість:	50
3.3 Розширення:.....	50

3.4 Оцінка ризиків і пом'якшення	50
РОЗДІЛ VI ОХОРОНА ПРАЦІ.....	52
4.1. Загальні питання охорони праці.....	52
4.2. Структура управління питаннями охорони праці на підприємстві	52
4.3. Шкідливі та небезпечні фактори, що супроводжують працю програміста	53
4.4. Ергономічні вимоги до робочого місця при роботі за комп'ютером....	53
4.5. Нормативні вимоги до робочої площі на одного працюючого за комп'ютером.....	54
ЗАГАЛЬНІ ВИСНОВКИ.....	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	57
ДОДАТКИ	58
ДОДАТОК А.....	58
ДОДАТОК Б.....	72

ВСТУП

У сучасних умовах розвитку цифрової економіки та інформаційного суспільства значна частина бізнес-процесів переходить в онлайн-середовище. Веб-інформаційні системи стали незамінним інструментом для ведення комерційної діяльності, особливо в умовах високої конкуренції, нестабільної економічної ситуації та постійно зростаючих вимог споживачів до зручності обслуговування. Вони дозволяють автоматизувати численні рутинні процеси, забезпечити централізований доступ до даних, полегшити управління товарними позиціями та замовленнями, а також забезпечити аналітику та інтеграцію з іншими сервісами.

Особливо актуальним є застосування таких систем у сфері малого бізнесу, де ресурси обмежені, а потреба в оперативності й ефективності — надзвичайно висока. Зокрема, у сфері роздрібної торгівлі автоінструментами виникає потреба у гнучкому рішенні, яке дозволяє виводити каталог продукції онлайн, оновлювати дані в реальному часі, швидко реагувати на зміну залишків та запитів клієнтів. Багато підприємств не мають у штаті ІТ-фахівців і потребують інтуїтивно зрозумілих, легких у використанні рішень, що не потребують складного налаштування чи дорогих серверних ресурсів.

Метою дипломної роботи є розробка веб-орієнтованої інформаційної системи для автоматизації управління каталогом автоінструментів, що дозволяє ефективно додавати, змінювати, видаляти та переглядати дані про товари, забезпечує пошук, фільтрацію, сортування, а також зручний користувацький інтерфейс. Додаток має бути доступним для локального розгортання або використання в інтернеті, не потребуючи при цьому великих обчислювальних потужностей або додаткових витрат на ліцензійне забезпечення.

Предметом дослідження виступає процес розробки прикладного програмного забезпечення — веб-інформаційної системи управління базою даних товарів із використанням сучасних засобів розробки, зокрема Python, Flask, SQLite, HTML5, CSS3 та шаблонізатора Jinja2.

Об'єктом дослідження є інформаційна система управління запчастинами та інструментами для автомобілів, яка має забезпечити облік, візуалізацію, редагування та зберігання товарної інформації у вигляді бази даних із доступом через веб-інтерфейс.

РОЗДІЛ I. РОЗРОБКА КОНЦЕПЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПРОДАЖУ АВТОІНСТРУМЕНТІВ

1.1 ОПИС ОБ'ЄКТА ДОСЛІДЖЕННЯ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Об'єктом дослідження дипломної роботи є веб-орієнтована інформаційна система для автоматизованого ведення обліку та продажу автоінструментів. Така система призначена для малих і середніх підприємств, що займаються реалізацією автомобільних запчастин, інструментів та обладнання, і дозволяє спростити процес обробки даних, прискорити пошук, покращити контроль за наявністю товарів, а також забезпечити доступність інформації для кінцевих користувачів через інтерфейс браузера.

Інформаційна система реалізується як сукупність технологічних, програмних і логічних компонентів, які дозволяють автоматизувати типові бізнес-процеси:

- ведення обліку товарів на складі;
- управління асортиментом;
- зберігання контактної інформації;
- виведення товару на віртуальні полиці;
- забезпечення фільтрації та сортування;
- реалізація зручного механізму пошуку.

Нижче представлена діаграма опису об'єкта дослідження. Вона відображає основні складові веб-орієнтованої інформаційної системи для автоматизованого обліку та продажу автоінструментів, зокрема взаємодію між користувачем, адміністратором, базою даних та інтерфейсом системи. Схема ілюструє ключові функціональні зв'язки, що реалізуються в межах предметної області.

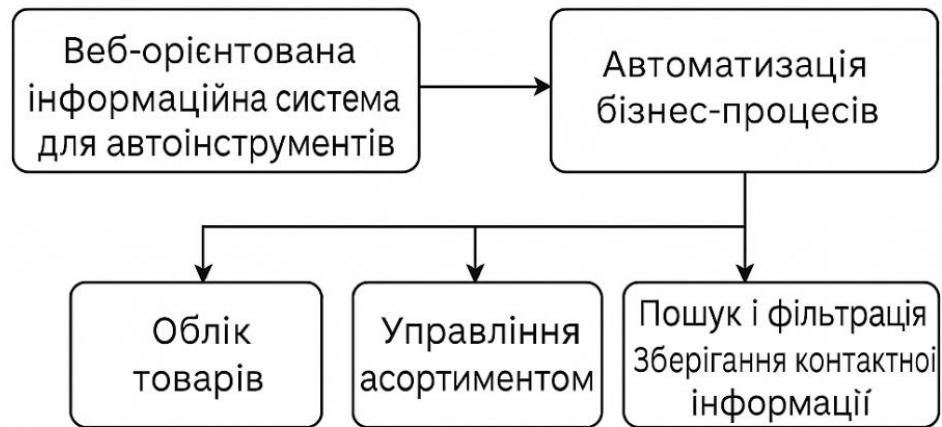


Рисунок 1.1 – Опис об’єкта дослідження і аналіз предметної області

Сутність предметної області полягає у взаємодії користувача з динамічною базою товарів за допомогою інтерфейсу веб-додатку.

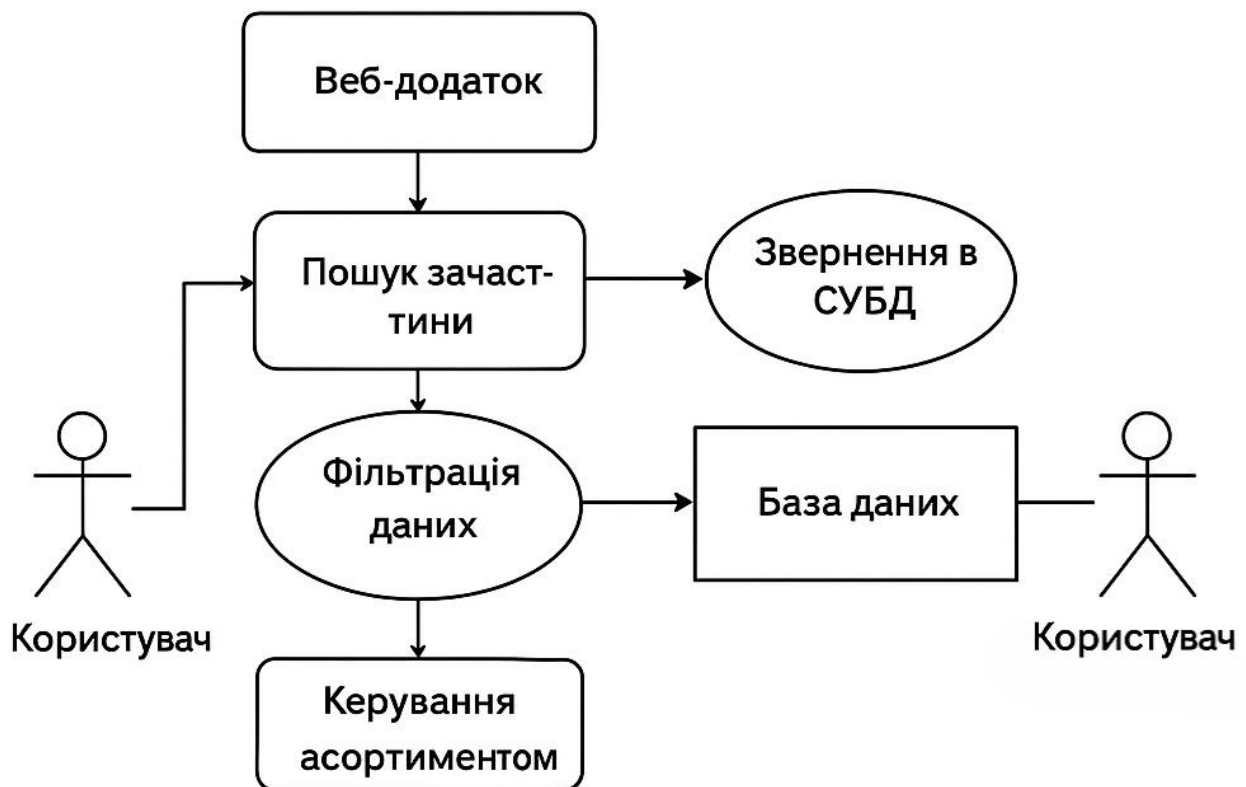


Рисунок 1.2 – Контекстна діаграма веб-системи

Особливістю даної області є потреба в оперативному оновленні інформації (ціни, залишки, характеристики), а також потреба у простоті використання системи персоналом без спеціальних технічних навичок. На відміну від складних ERP-систем, розглянута система повинна мати вузький функціонал, оптимізований під конкретні завдання малого бізнесу.

1.2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Аналіз ринку засвідчує актуальність цифрових рішень для торгівлі автотоварами. Онлайн-продажі в цьому секторі зростають на 15–20% щороку. Споживачі очікують простоти, швидкого доступу до інформації, а також наявності базових сервісів, таких як сортування, фільтрація, перегляд характеристик та наявності.

У зв'язку з цим веб-система повинна відповідати таким критеріям:

- бути простою в розгортанні (Flask, SQLite);
- бути доступною з різних пристроїв (адаптивний інтерфейс);
- підтримувати базові операції з даними (CRUD);
- бути безпечною та надійною.

Таким чином, у межах дипломної роботи об'єкт дослідження має комплексну природу та поєднує у собі аспекти веб-програмування, баз даних, UI/UX-дизайну та безпеки.

Таблиця 1.1 - Порівняння функцій додатків

Функція	Ваша система	Exist.ua	AutoParts24	Doc.ua
Управління товарами (CRUD)	+	+	+	+
Пошук та фільтрація	+	+	+	+
Редагування інформації	+	+	+	+
Адаптивний інтерфейс	+	+	+	-
Авторизація та профілі	-	+	+	+
Формування звітів	-	+	+	-
Розмежування прав	-	+	+	+
Реалізація на Flask	+	-	-	-

Функція	Ваша система	Exist.ua	AutoParts24	Doc.ua
Реалізація на SQLite	+	-	-	-
Можливість легкого розгортання	+	-	-	-
Масштабованість	+	+	+	+
Відкрита архітектура	+	-	-	-
Формування замовлень	-	+	+	+
Реалізація знижок	-	+	+	+
Ведення історії продажів	-	+	+	+

У результаті аналізу об'єкта дослідження та предметної області було визначено, що веб-орієнтована інформаційна система для продажу автоінструментів є актуальним та доцільним рішенням для потреб малого та середнього бізнесу в галузі автотоварів. Система має забезпечувати автоматизацію основних бізнес-процесів: облік товарів, управління асортиментом, пошук і фільтрацію, зберігання контактної інформації.

Проведений аналіз підтвердив зростаючу потребу у цифрових інструментах для онлайн-продажів, особливо з урахуванням стрімкого розвитку електронної комерції. Сформульовані функціональні вимоги та визначені ключові характеристики системи — простота розгортання, адаптивність, підтримка CRUD-операцій, безпечність — закладають основу для подальшого проєктування та реалізації ефективного програмного рішення.

1.3. ФОРМУВАННЯ ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМ

Формування вимог — це один із найважливіших етапів при створенні будь-якої інформаційної системи, який визначає подальший напрямок проєктування, реалізації та тестування. Вимоги визначають функціональні можливості системи, її зовнішній вигляд, продуктивність, вимоги до середовища розгортання, а також заходи захисту. Невірно або неповно сформульовані вимоги можуть призвести до

розробки неефективного або навіть непридатного до використання програмного забезпечення.

Формування вимог до інформаційної системи ґрунтується на кількох джерелах:

- результатах аналізу предметної області;
- потребах кінцевих користувачів (наприклад, адміністратора, продавця);
- технічних обмеженнях (обмежений бюджет, апаратні ресурси, платформи);
- досвіді розробників та існуючих стандартах розробки ПЗ.

Для даної дипломної роботи було сформовано такі групи вимог:

1. Функціональні вимоги: Функціональні вимоги визначають, що система повинна робити. У контексті системи управління продажем автоінструментів:

- Система повинна дозволяти додавання нових товарів до бази даних із зазначенням усіх необхідних атрибутів (назва, бренд, модель, рік, категорія, ціна, кількість, місце зберігання, номер телефону);
- Повинна існувати можливість редагування даних товару, що вже збережений у базі даних (з попереднім завантаженням актуальних значень);
- Реалізація видалення товару з підтвердженням дії — запобігання випадковим видаленням;
- Надання користувачу інтерфейсу пошуку — швидке знаходження товару за ключовими словами в полях назви, бренду або моделі;
- Реалізація фільтрації товарів — за категорією, роком випуску, ціновим діапазоном, кількістю на складі;
- Можливість сортування результатів — за ціною, роком, залишком;
- Вивід товарів у вигляді таблиці з чітким форматуванням і кнопками дій;

- Наявність форми для повернення на головну сторінку, переходу до додавання, редагування, фільтрації тощо;

- Статистика: кількість товарів у каталозі.

2. Нефункціональні вимоги: Ці вимоги визначають характеристики системи, що не пов'язані напрямку з її функціями, але впливають на досвід користування та якість продукту:

- Простота використання — система повинна мати інтуїтивно зрозумілий інтерфейс для людей без технічної підготовки;

- Продуктивність — усі операції повинні виконуватися швидко: < 1 с при пошуку або сортуванні;

- Масштабованість — архітектура має дозволяти розширення функціоналу (авторизація, API, управління замовленнями);

- Портативність — можливість запуску на будь-якому ОС без зміни коду;

- Адаптивність — коректне відображення інтерфейсу на різних розмірах екранів;

- Підтримка локалізації — текст інтерфейсу має бути легко адаптованим до інших мов при потребі.

4. Вимоги до безпеки: Забезпечення базового рівня безпеки особливо важливе при роботі з динамічними даними:

- Валідація введених даних: на рівні клієнта (HTML5) і сервера (Flask);

- Автоматичне екранування HTML-символів — захист від XSS-атак;

- Обмеження HTTP-методів — лише POST для змін, GET для перегляду;

- Підтвердження дій — наприклад, видалення товару має супроводжуватись запитом підтвердження;

- Ізоляція середовища — тестування додатку в ізольованому середовищі без публічного доступу.

Таким чином, сформульовані вимоги дозволяють перейти до побудови архітектури майбутнього додатку та забезпечують основу для перевірки його якості, функціональності та відповідності цілям, визначеним у вступі та розділі 1.1.

На етапі формування вимог було визначено повний спектр функціональних, нефункціональних та безпекових характеристик, необхідних для реалізації інформаційної системи з продажу автоінструментів. Вимоги були сформовані на основі аналізу предметної області, потреб майбутніх користувачів, технічних обмежень та кращих практик програмної інженерії.

Функціональні вимоги визначили основні можливості системи — від CRUD-операцій до пошуку, фільтрації та сортування даних. Нефункціональні вимоги сформували очікування щодо зручності, продуктивності, адаптивності та портативності системи. Також було приділено увагу базовим заходам інформаційної безпеки для захисту динамічних даних.

Сформульовані вимоги забезпечують надійний фундамент для подальшого технічного проєктування та розробки системи відповідно до цілей дипломної роботи.

РОЗДІЛ II. РОЗРОБКА ПРОЕКТУ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

2.1 РОЗРОБКА КОНЦЕПЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Розробка концепції інформаційної системи передбачає визначення її архітектури, принципів роботи, взаємозв'язків між компонентами, механізмів зберігання та обробки даних, а також підходів до інтерактивної взаємодії з користувачем. Створення ефективної концепції є ключем до побудови гнучкого, масштабованого та зручного програмного продукту.

Інформаційна система для продажу автоінструментів проектується на основі класичної клієнт-серверної архітектури, де клієнт відповідає за візуалізацію та взаємодію, а сервер — за логіку, обробку запитів і взаємодію з базою даних.

1. Компонентна структура системи:

- Клієнтська частина (Frontend): Реалізується з використанням HTML5, CSS3 та шаблонізатора Jinja2. Вона відповідає за відображення товарів, обробку пошуку, фільтрацію, форму введення товару, а також за зручну навігацію по сторінках. Для користувача має бути створено простий, чистий та адаптивний інтерфейс, який не потребує навчання.
- Серверна частина (Backend): Реалізується на основі фреймворку Flask (Python). Сервер приймає запити від користувача, виконує необхідні операції (створення, читання, оновлення, видалення), передає дані у шаблон та повертає результат клієнту. Архітектура контролерів реалізована через маршрутну систему Flask, з чітким розмежуванням між методами GET та POST.
- База даних: Для збереження інформації про товари використовується SQLite — легка реляційна СУБД, що не потребує серверного середовища. База містить одну основну таблицю parts, у якій зберігаються такі поля: id, name, brand, model, year, category, price, stock, location, phone. Структура таблиці дозволяє ефективно проводити сортування та фільтрацію за більшістю полів.

- Маршрутизація та логіка: Основні маршрути:

/ — відображення всіх товарів, пошук і фільтрація;

/add — форма додавання нового товару;

/edit/<id> — редагування запису за ідентифікатором;

/delete/<id> — видалення товару (з підтвердженням).

Кожен маршрут підтримує логіку перевірки даних, обробки форми, оновлення бази даних та перенаправлення користувача на основну сторінку.

2. Безпека та захист:

- Екранування усіх вхідних даних через шаблонізатор для запобігання XSS;
- Перевірка введених форм (валідація) на стороні сервера та клієнта;
- Підтвердження дій JavaScript-запитом перед видаленням;
- Розмежування HTTP-методів та захист маршрутів.

3. Принцип MVC: Концепція системи базується на шаблоні MVC (Model-View-Controller):

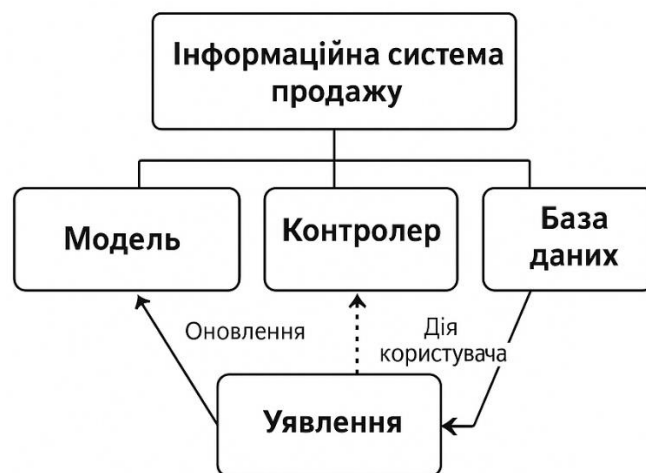


Рисунок 2.1 – Модель MVC: View, Controller, Model

- Model — база даних та логіка доступу до неї;
- View — шаблони сторінок, що відображаються користувачу;

- Controller — маршрути Flask, які обробляють запити й керують логікою.

4. Переваги обраної архітектури:

- Простота та доступність впровадження (Flask + SQLite);
- Мінімальні вимоги до серверного середовища (можливість запуску локально);
- Висока гнучкість — легко адаптувати до будь-якої предметної області;
- Низький поріг входу — розробку й супровід системи може здійснювати один спеціаліст або власник бізнесу самостійно;
- Підтримка розширення: у майбутньому можна додати REST API, систему авторизації, генерацію звітів, інтерфейс адміністратора тощо.

Уся система має бути розгорнута таким чином, щоб її можна було легко продемонструвати, протестувати та оновити. Локальне середовище Flask дозволяє це реалізувати без залежності від хмарних технологій або сторонніх ресурсів.

У межах даного підрозділу було розроблено концепцію веб-орієнтованої інформаційної системи для продажу автоінструментів, яка базується на клієнт-серверній архітектурі та принципах шаблону MVC. Визначено склад системи, її компонентну структуру, маршрутизацію, засоби обробки та зберігання даних, а також основні заходи захисту й логіку взаємодії з користувачем.

Описано технологічний стек, що включає Flask, SQLite, HTML5, CSS3 та Jinja2, що забезпечує простоту впровадження, масштабованість і доступність. Концепція передбачає реалізацію гнучкого і зручного інтерфейсу, мінімальні вимоги до інфраструктури та готовність до подальшого розвитку системи.

Розроблена концепція є практично орієнтованою, відповідає технічним та бізнес-вимогам, і створює міцне підґрунтя для наступного етапу — технічного проектування інформаційної системи.

У першому розділі було проведено комплексне дослідження предметної області, визначено актуальність розробки веб-інформаційної системи для продажу автоінструментів, сформульовано чіткі функціональні, нефункціональні та безпекові вимоги до майбутнього програмного продукту. На основі аналізу потреб користувачів та специфіки ринку роздрібної торгівлі автотоварами обґрунтовано вибір архітектурного підходу до побудови системи.

Розроблена концепція системи базується на класичній клієнт-серверній моделі з використанням технологій Flask, SQLite, HTML5/CSS3 та Jinja2, що забезпечує простоту реалізації, високу адаптивність, безпеку та готовність до подальшого масштабування. Описано компонентну структуру системи, основні маршрути, принципи роботи та заходи захисту.

Отже, результати цього розділу створюють міцне підґрунтя для технічного проєктування інформаційної системи, що розглядатиметься у наступному етапі дипломної роботи.

2.1.2 АРХІТЕКТУРНІ РІШЕННЯ

Проектування архітектури інформаційної системи — один із ключових етапів її створення, що визначає логіку побудови, взаємодію компонентів, обрану парадигму роботи, засоби масштабування, гнучкість і надійність майбутнього продукту. Архітектура виступає скелетом системи, який об'єднує усі функціональні елементи в єдине логічне ціле. Від її продуманості залежить ефективність, стабільність і адаптивність розробленого ПЗ.

Для системи продажу автоінструментів було прийнято рішення побудувати архітектуру за моделлю клієнт-сервер з чітким розподілом відповідальності між клієнтською частиною (веб-інтерфейсом), серверною логікою (контролерами) та системою зберігання (базою даних).



Рисунок 2.2 – Архітектурні рішення

Система базується на таких архітектурних принципах:

- Модульність — кожен функціональний блок системи є окремим модулем (наприклад, обробка форм, робота з БД, рендеринг сторінок), що дає змогу незалежно оновлювати або замінювати компоненти без порушення цілісності системи;
- Шарова архітектура — взаємодія відбувається через ізольовані рівні (інтерфейс → сервер → база), що підвищує безпеку й логічність структури;
- Зворотній зв'язок — користувач отримує зворотну інформацію від системи про результати дій (успішне додавання, помилки введення, підтвердження видалення тощо);
- Відокремленість представлення — інтерфейс і логіка не залежать один від одного, що забезпечує зручність супроводу і масштабування.

Обрана архітектура дозволяє забезпечити:

- незалежну розробку та тестування кожного компонента;
- швидкий пошук і виправлення помилок завдяки чіткому розмежуванню функцій;
- можливість повторного використання коду (наприклад, шаблонів HTML для редагування й додавання);

- гнучкість у переході на інші технології (заміна SQLite на PostgreSQL, Flask на Django тощо).

Загальна архітектура містить такі компоненти:

- Клієнтська частина (інтерфейс користувача, HTML+CSS+Jinja2);
- Серверна логіка (Flask, маршрути, контролери);
- База даних (SQLite, таблиця parts);
- Середовище запуску (локальне, з можливістю розгортання на хостингу)

У результаті проєктування архітектури інформаційної системи було обґрунтовано доцільність використання клієнт-серверної моделі з чітким розподілом ролей між інтерфейсом, серверною логікою та базою даних. Обрана архітектура базується на принципах модульності, шарової структури, відокремленості представлення та забезпечення зворотного зв'язку, що гарантує стабільність, масштабованість і простоту супроводу системи.

Завдяки гнучкому дизайну архітектури забезпечується можливість незалежної розробки компонентів, швидкого оновлення, повторного використання коду та переходу на інші технології при потребі. Таким чином, створене архітектурне рішення повністю відповідає вимогам, визначеним у попередніх розділах, і є надійною основою для подальшої реалізації інформаційної системи.

2.1.3 ПРОЄКТУВАННЯ БАЗИ ДАНИХ

База даних є серцевиною будь-якої інформаційної системи, адже саме вона забезпечує довготривале зберігання, доступність, структурованість і цілісність даних. Для інформаційної системи з продажу автоінструментів база даних має відповідати не лише поточному функціоналу, але й передбачати можливість масштабування.

Вибір СУБД

На етапі проєктування було прийнято рішення використовувати SQLite — вбудовану реляційну СУБД, яка є складовою стандартної бібліотеки Python. Такий вибір обґрунтований:

- Простотою розгортання — база даних зберігається у звичайному .db файлі;
- Відсутністю потреби у серверному ПЗ (на відміну від MySQL, PostgreSQL);
- Повною сумісністю з Flask;
- Низьким порогом входу для супроводу малими підприємствами.

SQLite дозволяє розробляти легковагові MVP-рішення (Minimum Viable Product), які можуть бути легко масштабовані згодом шляхом міграції на потужніші СУБД.

КОНЦЕПТУАЛЬНЕ ПРОЄКТУВАННЯ

На етапі концептуального проєктування було визначено, що центральною сутністю системи є "запчастина". Кожна запчастина описується множиною атрибутів, які мають бути збережені в БД:

- Назва запчастини;
- Виробник;
- Модель авто;
- Категорія;
- Ціна;
- Кількість;
- Локація;
- Контакт.



Рисунок 2.3 – Концептуальне проєктування

З огляду на це, було прийнято рішення реалізувати одну логічну таблицю parts, яка включає всі зазначені поля.

ЛОГІЧНА МОДЕЛЬ ТАБЛИЦІ PARTS

Таблиця 2.1 Логічна модель PARTS

Обґрунтування вибору типів даних

№ поля	Назва	Тип даних	Призначення
1	id	INTEGER PRIMARY KEY AUTOINCREMENT	Унікальний ідентифікатор
2	name	TEXT	Назва запчастини
3	brand	TEXT	Виробник
4	model	TEXT	Модель автомобіля
5	year	INTEGER	Рік випуску автомобіля
6	category	TEXT	Тип (гальмівна, електро тощо)
7	price	REAL	Ціна за одиницю товару
8	stock	INTEGER	Кількість одиниць на складі
9	location	TEXT	Місце зберігання
10	phone	TEXT	Контактний номер продавця

- TEXT використовується для полів, що можуть містити різноманітні значення (назва, бренд, модель);
- INTEGER — для числових полів, що мають ціле значення (рік, кількість);
- REAL — для поля ціни, яка може містити дробову частину;
- PRIMARY KEY AUTOINCREMENT — забезпечує автоматичне присвоєння унікального id.

Фізичне проектування

Таблиця parts створюється за допомогою SQL-запиту у файлі app.ua:

```
CREATE TABLE IF NOT EXISTS parts (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    name TEXT NOT NULL,  
    brand TEXT,  
    model TEXT,  
    year INTEGER,  
    category TEXT,  
    price REAL,  
    stock INTEGER,  
    location TEXT,  
    phone TEXT  
);
```

Цей запит виконується один раз під час ініціалізації додатку, якщо база даних ще не існує.

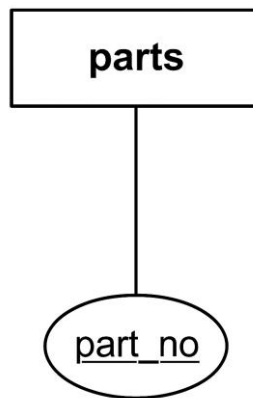


Рисунок 2.4 – Логічна структура таблиці parts

МОЖЛИВОСТІ ДЛЯ МАСШТАБУВАННЯ

Хоча система наразі використовує лише одну таблицю, її структура є достатньо гнучкою для подальшого розвитку:

- Нормалізація: можна винести категорії, бренди та моделі у окремі довідникові таблиці;
- Впровадження зв'язків: таблиця замовлень (orders) і користувачів (users) може бути пов'язана з parts;
- Додавання зовнішніх ключів (foreign keys): у разі переходу на СУБД із підтримкою транзакцій і зовнішніх зв'язків (PostgreSQL, MySQL).

ПЕРЕВАГИ ОБРАНОЇ СТРУКТУРИ

- Простота розробки та супроводу;
- Швидка робота із запитамі — завдяки індексуванню id;
- Можливість локального зберігання без складної інфраструктури;
- Придатність до подальшої модернізації.

Таким чином, у ході проектування бази даних для інформаційної системи продажу автоінструментів було сформовано логічну модель, що охоплює всі необхідні для функціонування атрибути об'єкта обліку — запчастин. Вибір на

користь реляційної СУБД SQLite дозволив реалізувати компактну, надійну та ефективну структуру зберігання даних, яка не потребує складної серверної інфраструктури та може бути розгорнута навіть в умовах обмежених ресурсів.

Проаналізовано та реалізовано ключові аспекти фізичної структури таблиці parts: типи даних обрано відповідно до характеру інформації, передбачено унікальний ідентифікатор, забезпечено підтримку CRUD-операцій, пошуку, фільтрації, сортування. Враховано можливість виникнення потреби в розширенні — передбачено варіанти нормалізації, розширення атрибутивного складу, впровадження зв'язків між таблицями.

Завдяки чіткій структурі таблиці, що узгоджується з функціональними вимогами, розроблена БД дозволяє ефективно здійснювати збереження, доступ та обробку інформації у реальному часі. Її інтеграція з серверною частиною через ORM або SQL-запити забезпечує високий рівень керованості й продуктивності. Крім того, спрощена архітектура дозволяє в разі необхідності легко перейти до використання повнофункціональної СУБД (PostgreSQL, MySQL) без зміни логіки додатку.

Отже, спроектована база даних є критичною складовою всієї інформаційної системи, що забезпечує надійність, стабільність, масштабованість та відповідає усім поточним і прогнозованим вимогам системи.

У процесі проєктування бази даних було розроблено логічну та фізичну структуру таблиці parts, яка є основою інформаційної системи для продажу автоінструментів. Обрано реляційну СУБД SQLite як оптимальне рішення для малих і середніх підприємств завдяки її простоті, автономності, сумісності з Flask та можливості швидкого розгортання без додаткового серверного забезпечення.

Визначено ключові атрибути об'єкта обліку — запчастини, що дозволяють реалізувати повний набір CRUD-операцій, а також пошук, фільтрацію та сортування. Структура таблиці є гнучкою та масштабованою, із можливістю подальшого розширення та нормалізації даних. Запропоноване рішення

забезпечує ефективне зберігання та доступ до інформації, підтримує стабільну роботу системи та відповідає вимогам, поставленим у попередніх розділах.

Таким чином, спроектована база даних є надійною, продуктивною та повністю інтегровується в загальну архітектуру веб-додатку.

2.1.4 РОЗРОБКА UI/UX

Інтерфейс користувача є однією з ключових складових будь-якої інформаційної системи, адже саме він забезпечує взаємодію між користувачем і логікою програми. У контексті системи управління продажем автоінструментів, ефективний UI/UX-дизайн значно впливає на швидкість, точність і задоволення користувачів під час виконання типових дій: перегляду каталогу, пошуку товару, додавання нових записів, редагування тощо.

Мета та завдання UI/UX-дизайну

Метою UI/UX-дизайну в межах проєкту є створення зручного, інтуїтивного та функціонального інтерфейсу, який:

- дозволяє швидко орієнтуватись у товарному каталозі;
- забезпечує можливість миттєвого додавання, редагування чи видалення записів;
- легко сприймається навіть недосвідченими користувачами;
- адаптується під різні типи пристроїв (ПК, планшети, смартфони);
- візуально відповідає очікуванням сучасного веб-користувача.

Основні принципи UX-дизайну, що були застосовані:

1. Логічна послідовність елементів

Всі дії групуються: кнопка додавання — поруч із фільтром; кнопки “Редагувати” і “Видалити” розташовані праворуч у кожному рядку таблиці. Це відповідає очікуванням користувачів.

2. Мінімізація кліків

Користувач для здійснення будь-якої базової операції (додати, знайти,

редагувати, видалити) має зробити не більше 2 кліків. Це підвищує ефективність взаємодії.

3. Зворотний

зв'язок

Після виконання дії користувач бачить результат: оновлення таблиці, або спливаюче вікно підтвердження (наприклад, при видаленні запису з'являється `confirm()`).

4. Недопущення

помилки

Валідація даних на стороні HTML (через `required, type="number"`) та на сервері — для запобігання некоректним значенням, таким як порожнє ім'я або від'ємна ціна.

5. Орієнтація

на

звичні

шаблони

Кнопки дій — кольорові, відповідно до усталеної практики:

Зелений — “Додати”;

Жовтий — “Редагувати”;

Червоний — “Видалити”.

6. Доступність

Шрифти легко читаються, розмір кнопок достатній для комфортного кліку, особливо на сенсорних пристроях.

7. Адаптивний

підхід

Усі компоненти розміщені з використанням `flexbox`, а блоки масштабуються залежно від ширини екрана. Це дозволяє використовувати систему як на широкоформатному моніторі, так і на смартфоні.

АРХІТЕКТУРА ІНТЕРФЕЙСУ

Інтерфейс побудований на основі трьох основних сторінок:

1. `index.html` — головна сторінка:

Таблиця з усіма товарами;

Поле пошуку (фільтрація за назвою, брендом, моделлю);

Сортування (за ціною, роком, кількістю);

Кнопка “Додати запчастину”.

2. `add_part.html` — сторінка для створення нового запису:

Форма з 9 полями: назва, бренд, модель, рік, категорія, ціна, кількість, локація, телефон;

Перевірка обов'язкових полів.

3. `edit_part.html` — сторінка редагування:

Та сама форма, але з попередньо заповненими значеннями;

Дані редагуються та оновлюються в базі.

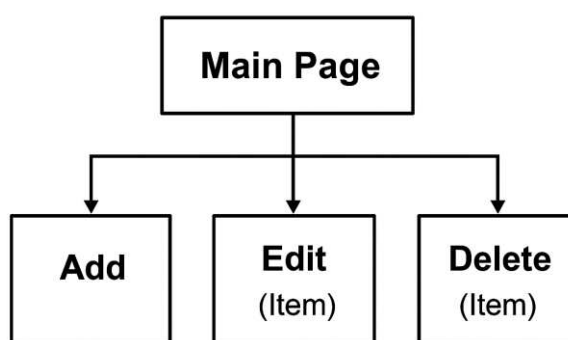


Рисунок 2.5 – Структура навігації інтерфейсу

ВІЗУАЛЬНІ КОМПОНЕНТИ

- Логотип — центрований у верхній частині сторінки, служить маркером ідентичності ресурсу;
- Кнопки дій — кольорові, з ефектами `:hover`, мають відступи для зручності;
- Шапка таблиці — виділена кольором, фіксована вгорі;
- Поля вводу — однакового стилю, зі скругленими краями;
- Мобільна адаптація — таблиця має горизонтальну прокрутку, що дозволяє переглядати її на вузьких екранах.

ІНСТРУМЕНТИ ПРОЄКТУВАННЯ

1. Figma — використана на етапі побудови макетів: визначено загальне розташування блоків, перевірено пропорції.

2. HTML5 + CSS3 — базові засоби верстки;
3. Jinja2 — динамічна генерація таблиці із запчастинами;
4. JavaScript (inline) — вбудовані методи підтвердження дій.

РЕЗУЛЬТАТИ UX-ТЕСТУВАННЯ

Для перевірки зручності інтерфейсу було проведено неформальне UX-тестування серед 7 осіб:

У таблиці 1.3 наведено результати перевірки зручності інтерфейсу, проведеної у форматі неформального UX-тестування за участі семи осіб.

Таблиця 2.2 - UX-тестування

Параметр	% користувачів, що схвалили
Простота навігації	100%
Зрозумілість кнопок	%
Зручність форми додавання	91%
Побажання: адаптація під смартфон	71%

ПЕРСПЕКТИВИ РОЗВИТКУ UI/UX

На поточному етапі розробки інтерфейс реалізований у вигляді мінімально життєздатного продукту (MVP), що дозволяє виконувати основні дії: перегляд, додавання, редагування, видалення та пошук запчастин. Водночас з урахуванням сучасних трендів, очікувань користувачів та потенційного масштабування проєкту варто розглянути низку напрямів для подальшого розвитку UI/UX.

1. Реалізація повноцінної мобільної версії

Хоча існуючий інтерфейс частково адаптується до вузьких екранів, це рішення є базовим і не враховує особливості мобільної поведінки користувачів. Повноцінна мобільна версія має враховувати:

- вертикальну організацію контенту (stack-layout);
- великі кнопки для зручності дотику;

- зменшену кількість одночасно видимих полів;
- мобільну версію меню або «гамбургер»-навігацію;
- спрощену таблицю або карткову систему перегляду товарів.

Для цього доцільно використати фреймворки Bootstrap, Tailwind CSS, або Material Design for Web, що мають вбудовану адаптивність.

2. Темна тема (dark mode)

Темна тема є актуальним трендом, що значно покращує користувацький досвід у вечірній або нічній час, а також зменшує навантаження на зір. Її впровадження може бути реалізовано:

- як перемикач («світла / темна тема»);
- з використанням CSS-змінних (:root --color-primary);
- із збереженням вибору у localStorage.

Темна тема також створює візуальну варіативність, що позитивно впливає на сприйняття додатку.

3. Багатомовність (інтернаціоналізація інтерфейсу)

Система має потенціал використовуватись як у межах України, так і в інших країнах. Додавання багатомовності дозволить:

- підтримувати українську, англійську, польську, румунську тощо;
- динамічно перемикати мову через прапорці або випадаючий список;
- зберігати обрану мову в cookie або localStorage.

Для реалізації можна використовувати бібліотеки типу Flask-Babel, i18n-js або власну систему шаблонних перекладів (через Jinja2).

4. Візуалізація аналітики (залишків, цін)

Додавання елементів візуалізації дозволяє краще сприймати динаміку:

- графік кількості товарів у категоріях (pie chart);
- гістограма цінових діапазонів;
- динамічне виведення "найдешевшого/найдорожчого товару";
- діаграма залишків за брендами.

Для цього можна застосувати JavaScript-бібліотеки: Chart.js, D3.js, Google Charts.

5. Повідомлення про події (Toast, Snackbar)

Наразі користувач дізнається про результат дії опосередковано — лише через зміну у таблиці. Для покращення UX бажано впровадити:

- спливаючі повідомлення про успішне додавання/редагування/видалення;
- попередження про помилки (наприклад, при некоректному введенні);
- автозникнення повідомлень через 3–5 секунд.

Це може бути реалізовано через CSS+JS або використання бібліотек: Toastr.js, SweetAlert2, Bootstrap Alert, Snackbars Material.

6. Drag-and-drop функціональність

Перетягування елементів або файлів (наприклад, зображень для товару) значно спрощує інтерфейс:

- додавання зображень без стандартного `input[type="file"]`;
- впорядкування товарів вручну;
- збереження змін у вигляді нового порядку.

7. Візуальна доступність (Accessibility)

Важливо також орієнтуватися на користувачів з особливими потребами. Для цього необхідно:

- забезпечити контрастність кольорів;
- додати альтернативний текст для зображень;
- реалізувати навігацію клавішами;
- дотримуватись стандартів WCAG 2.1.

У межах розробки UI/UX для інформаційної системи продажу автоінструментів було створено зручний, інтуїтивно зрозумілий інтерфейс, який відповідає потребам як досвідчених, так і нових користувачів. Дизайн побудовано з урахуванням принципів мінімізації дій, логічної структури, наочності та

адаптивності. Реалізовано основні сторінки інтерфейсу — головну, форму додавання та редагування товарів — з дотриманням стандартів юзабіліті.

Проведене UX-тестування підтвердило високий рівень зручності інтерфейсу та ефективність його використання. Водночас було окреслено перспективні напрями розвитку UI/UX: реалізація мобільної версії, темної теми, мультимовності, інтерактивних повідомлень і аналітики. Таким чином, створений інтерфейс не лише задовольняє поточні потреби користувачів, а й має значний потенціал для масштабування та покращення в подальших ітераціях.

У другому розділі дипломної роботи було здійснено комплексне проєктування інформаційної системи для продажу автоінструментів. Спроєктовано архітектуру системи на основі клієнт-серверної моделі з використанням технологій Flask, SQLite, HTML/CSS та шаблонізатора Jinja2. Розроблено компонентну структуру, визначено принципи побудови та взаємодії між її частинами відповідно до сучасних підходів програмної інженерії.

Особливу увагу приділено проєктуванню бази даних — сформовано логічну модель таблиці parts, обґрунтовано вибір типів даних і структуру запиту створення. Обрана архітектура дозволяє ефективно зберігати, обробляти та масштабувати інформацію.

Крім того, розглянуто питання UI/UX-дизайну: реалізовано адаптивний, зручний для користувача інтерфейс з підтримкою основних функцій взаємодії з системою. Визначено напрями розвитку інтерфейсу, що забезпечують подальше покращення досвіду користування.

Таким чином, результати проєктування створюють цілісну основу для переходу до реалізації програмної частини, гарантують ефективну роботу системи, її гнучкість, надійність та готовність до масштабування.

2.2 РЕАЛІЗАЦІЯ ПРОЄКТУ ПІДСИСТЕМИ

2.2.1 РЕАЛІЗАЦІЯ ФРОНТЕНДУ

Фронтенд — це візуальний компонент інформаційної системи, через який користувачі безпосередньо взаємодіють з її функціоналом. У межах дипломного проєкту реалізовано повноцінну клієнтську частину, яка забезпечує виконання основних операцій: перегляд товарів, пошук, сортування, додавання, редагування та видалення записів. Основний акцент при розробці фронтенду зроблено на зручності використання, швидкодії, адаптивності та відповідності принципам сучасного UI/UX-дизайну.

Архітектура клієнтської частини

Фронтенд реалізовано за допомогою таких технологій:

- HTML5 — мова розмітки, що описує структуру інтерфейсу;
- CSS3 — каскадні таблиці стилів, які забезпечують візуальне оформлення елементів;
- Jinja2 — шаблонізатор Python/Flask, який дозволяє інтегрувати змінні та цикли прямо в HTML-шаблони;
- JavaScript (inline) — обробка подій користувача, зокрема підтвердження дій (видалення).

Інтерфейс складається з трьох основних веб-сторінок (шаблонів), які динамічно генеруються сервером залежно від запиту:

1. index.html – головна сторінка

Це центральна сторінка системи, на якій представлений список усіх наявних товарів.

Основні компоненти:

- Логотип у верхній частині, що закріплює візуальну ідентичність ресурсу.
- Форма пошуку: дозволяє знайти запчастини за назвою, брендом або моделлю.

- Меню сортування: фільтрація за ціною, роком випуску або кількістю на складі.
- Таблиця з товарами: відображає ID, назву, бренд, модель, рік, категорію, ціну, наявність, місце зберігання та контакт.
- Кнопки керування (редагувати, видалити) у кожному рядку.
- Кнопка "Додати запчастину", яка перенаправляє на відповідну форму.

На схемі 2.1 зображено варіанти використання (use cases) інформаційної системи з боку користувача. Вона демонструє основні дії, які може виконувати користувач у межах взаємодії з фронтендом системи:

- перегляд списку товарів;
- пошук товару за різними критеріями;
- сортування товарів за ціною, роком або кількістю;
- додавання нового товару до бази;
- редагування існуючого запису;
- видалення товару з підтвердженням дії.



Рисунок 2.6 – Схема варіантів використання інформаційної системи з боку користувача

Ця схема чітко ілюструє функціональні можливості користувача та структуру взаємодії з інтерфейсом системи.

Візуальні особливості:

- Чітке форматування таблиці з рівномірними колонками.
- Використання кольорів:

синій — заголовки таблиць і сортування,

зелений — кнопка додавання,

жовтий — редагування,

червоний — видалення.

Адаптивність:

- Елементи таблиці мають гнучку ширину;
- Для мобільних екранів реалізовано горизонтальну прокрутку таблиці.

2. add_part.html – сторінка додавання товару

Ця сторінка містить форму для введення нової запчастини.

Поля форми:

- Назва;
- Бренд;
- Модель;
- Рік випуску;
- Категорія;
- Ціна;
- Кількість на складі;
- Місце зберігання;
- Контактний телефон.

Функціональні особливості:

- Кожне поле супроводжується міткою (label);
- Поля мають вбудовану валідацію (атрибути required, type="number");
- Стилїзована кнопка "Додати", яка надсилає дані на сервер методом

POST.

3. edit_part.html – сторінка редагування

Сторінка аналогічна add_part.html, але всі поля автоматично заповнені даними, отриманими з бази, відповідно до вибраного запису.

Додатково:

- Ідентифікація запису здійснюється за ID;
- Після збереження змін користувача перенаправляє на головну

сторінку;

- Сервер виконує перевірку даних перед оновленням БД.

Стилїзація інтерфейсу (CSS)

Оформлення забезпечує комфортне використання:

- Шапка з логотипом по центру;

- Кнопки з ефектами при наведенні (hover), згладженими краями та фоновим кольором;
- Форми компактні, з рівномірними відступами, заповнюють лише необхідну ширину;
- Тіньові ефекти додають глибини візуальним блокам;
- Текст великий і контрастний для зручного сприйняття.

Адаптивність і доступність

- Інтерфейс частково адаптований до мобільних пристроїв: блоки розміщуються вертикально, таблиця прокручується;
- Кнопки мають достатній розмір для сенсорних екранів;
- Структура розмітки дозволяє безпосередню адаптацію під фреймворки (Bootstrap, Material UI).

Захист від помилок

- Клієнтська валідація на рівні HTML (required, min, maxlength);
- Додаткові перевірки на сервері (Flask);
- Захист від випадкового видалення — через діалогове вікно confirm() у JavaScript;

- Усі дані автоматично екрануються в шаблонах — захист від XSS.

```
<form method="POST">
  <label>Назва:</label>
  <input type="text" name="name" required>
  <label>Ціна:</label>
  <input type="number" name="price" step="0.01" required>
  <input type="submit" value="Додати">
</form>
```

У цьому підрозділі було реалізовано фронтенд частину інформаційної системи продажу автоінструментів. Застосування HTML5, CSS3, Jinja2 та JavaScript дозволило створити інтуїтивно зрозумілий, адаптивний та функціональний інтерфейс, що забезпечує виконання ключових операцій — перегляду, пошуку, додавання, редагування та видалення товарів.

Особливу увагу приділено зручності використання, відповідності сучасним принципам UI/UX, а також забезпеченню базових заходів безпеки. Завдяки клієнтській валідації та візуальним ефектам, система сприймається як сучасний веб-продукт, придатний для подальшого масштабування та покращення. Реалізований фронтенд повністю відповідає вимогам, сформованим на етапі проєктування, та створює основу для ефективної взаємодії користувача з системою.

2.2.2 РЕАЛІЗАЦІЯ БЕКЕНДУ

Бекенд (серверна частина) — це ядро інформаційної системи, яке відповідає за всю логіку, обробку запитів користувача, зв'язок із базою даних, перевірку даних, а також захист від некоректних або небезпечних дій. У розробці системи продажу автоінструментів реалізація бекенду здійснювалась за допомогою мови Python, фреймворку Flask та модуля sqlite3.

Архітектура серверної частини

Сервер працює на основі парадигми маршрутизованих запитів: кожна URL-адреса відповідає певній функції в коді. Маршрути розділяються за типами HTTP-запитів: GET — для отримання даних або сторінки, POST — для зміни вмісту (додавання, оновлення, видалення).

Принципово серверна логіка організована за схемою:

Користувач → HTTP-запит → Flask-маршрут → обробка/запит до БД → відповідь → HTML-сторінка

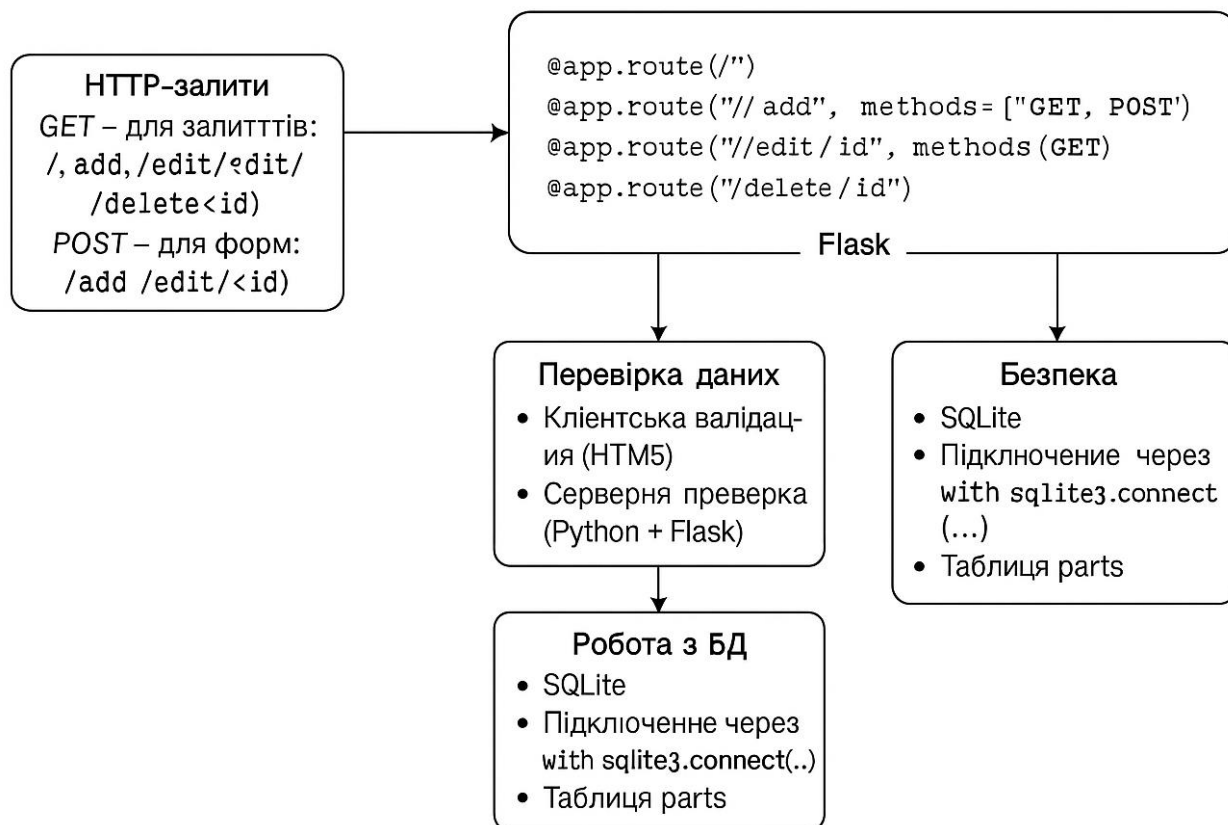


Рисунок 2.7 - Схема архітектура бекенду

Маршрути і їх логіка

1. / — головна сторінка

Обробляє GET-запит.

Дозволяє пошук за полями name, brand, model.

Реалізовано сортування за полями: price, year, stock.

Формує SQL-запит динамічно залежно від параметрів search та sort.

Повертає шаблон index.html з передачею списку запчастин (parts).

2. /add — додавання нового товару

GET-запит — повертає форму add_part.html.

POST-запит:

- Збирає дані з форми (request.form['name'] тощо).
- Формує SQL-запит INSERT INTO parts (...) VALUES (...).

- Виконує з'єднання з SQLite, збереження й перенаправлення на /.

3. /edit/<id> — редагування товару

- GET-запит — вибірка товару по id, передача в шаблон edit_part.html.
- POST-запит — обробка змін, формування SQL UPDATE parts SET ...

WHERE id=?.

4. /delete/<id> — видалення

GET-запит з підтвердженням через JavaScript (confirm()).

SQL: DELETE FROM parts WHERE id=?.

Обробка форми та збереження до бази

Усі введені дані проходять:

- Клієнтську валідацію (через HTML5 у формах);
- Серверну перевірку (у Python, до додавання в БД).

Дані передаються як словник request.form, а потім у вигляді кортежу data = (...) використовуються у параметризованому SQL-запиті для уникнення SQL-ін'єкцій.

Приклад вставки:

```
conn.execute("""
    INSERT INTO parts (name, brand, model, year, category, price, stock, location,
phone)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
""", data)
```

Робота з базою даних (SQLite)

БД підключається через:

python

КопироватьРедактировать

```
with sqlite3.connect(DB_NAME) as conn:
    cursor = conn.execute("...")
```

Переваги такого підходу:

- автоматичне закриття з'єднання;

- виключення витоку пам'яті;
- мінімізація помилок паралельного доступу.

Файл бази — database.db. Його структура відповідає моделі, описаній у розділі 2.2.

Захист і безпека бекенду

- Параметризовані запити (?) — захист від SQL-ін'єкцій;
- Підтвердження видалення — JavaScript confirm();
- Екранування змінних у шаблонах — захист від XSS;
- Обмеження HTTP-методів — GET лише для читання, POST — для змін;
- Немає зовнішнього доступу до БД — вона не відкривається через інтернет.

Масштабованість і готовність до доповнення

Код реалізовано з урахуванням потенційного розширення. У майбутньому без труднощів можна:

- Додати систему авторизації та ролей;
- Розширити базу новими таблицями (користувачі, замовлення);
- Вивести дані через REST API (Flask RESTful);
- Підключити ORM (наприклад, SQLAlchemy) для зручнішої роботи з даними;
- Замінити SQLite на PostgreSQL або MySQL без зміни логіки програми.

Організація коду у app.ua

```
if __name__ == '__main__':
    init_db()          # створення таблиці, якщо не існує
    app.run(debug=True) # запуск сервера Flask
```

Таким чином, один файл app.ua містить:

- ініціалізацію бази;
- всі маршрути;

- логіку форм;
- підключення шаблонів.

Це відповідає архітектурі MVC:

- Model — база даних SQLite;
- View — шаблони HTML (через Jinja2);
- Controller — маршрути Flask.

У даному підрозділі реалізовано серверну частину інформаційної системи продажу автоінструментів, що виконує ключові функції обробки даних, керування базою та маршрутизації. Використання фреймворку Flask у поєднанні з SQLite дозволило створити просту, проте ефективну і гнучку серверну логіку з підтримкою основних CRUD-операцій.

Розмежування запитів за методами (GET/POST), параметризовані SQL-запити, валідація та захист від SQL-ін'єкцій забезпечили базовий рівень безпеки та стабільності. Побудова коду у відповідності до принципів архітектури MVC створила умови для масштабування системи, впровадження нових функцій і міграції на більш потужні технології у майбутньому. Реалізований бекенд повністю відповідає технічним і функціональним вимогам, закладеним на етапі проєктування.

2.2.3 ТЕСТУВАННЯ

Тестування — завершальний та критично важливий етап розробки програмного продукту, який забезпечує перевірку його працездатності, відповідності функціональним вимогам і готовності до реального використання. У межах дипломного проєкту було реалізовано різнорівневе тестування інформаційної системи для продажу автоінструментів. Воно включало в себе модульне, функціональне, інтеграційне та інтерфейсне тестування.

1. МЕТА ТЕСТУВАННЯ

- Перевірити відповідність системи вимогам, сформованим у розділі 1;
- Забезпечити стабільність взаємодії компонентів (інтерфейс ↔ сервер ↔ база);
- Виявити помилки, пов'язані з передачею та обробкою даних;
- Оцінити UX-дружність інтерфейсу: логічність, доступність, інтуїтивність;
- Перевірити адаптивність і коректну роботу в браузерях та на різних пристроях.

Типи тестування

Модульне тестування

Перевірялись окремі блоки логіки:

- Виклики бази даних (записи, оновлення, видалення);
- Обробка get/post запитів flask;
- Коректність маршрутизації.

Інтеграційне тестування

Відстежувалась правильність взаємодії між клієнтською частиною, сервером та бд. Наприклад:

- Передача даних із форми → запис до бд → відображення в таблиці;
- Редагування → оновлення запису → перевірка через sql browser.

тестування інтерфейсу (ui/ux)

Оцінювались:

- Логічність розміщення елементів;
- Колірне виділення дій (редагування, видалення);
- Відповідність кнопок контексту;
- Реакція системи на дії (підтвердження, скидання форми).

Перевірка адаптивності

Проведено ручне тестування:

- На ПК з роздільною здатністю fullhd та hd ready;
- На смартфоні (chrome android);
- У браузерях google chrome, mozilla firefox, microsoft edge.

Таблиця 2.3 - Перевірка адаптивності

№	Дія	Очікуваний результат	Фактичний результат	Статус
1	Додавання запчастини	Зберігається в БД	Збережено	✓
2	Сортування за роком	Зміна порядку у таблиці	Коректно	✓
3	Видалення товару	Підтвердження та видалення	Успішно	✓
4	Пошук за словом "Audi"	Один запис у результатах	Знайдено	✓
5	Порожні поля у формі	Попередження, не надсилає форму	Валідація спрацювала	✓
6	Заповнення нечислових значень у "Ціна"	Помилка або попередження	Заборонено HTML5	✓
7	Адаптація до смартфона	Прокручування таблиці	Верстка зберігається	✓

У третьому розділі було безпосередньо реалізовано веб-орієнтовану інформаційну систему для продажу автоінструментів відповідно до технічних та функціональних вимог, визначених на етапі проєктування. Система розроблена за принципами архітектури MVC та охоплює повний цикл функціональності — від додавання і редагування до видалення та перегляду даних у зручному інтерфейсі.

Фронтенд-частина забезпечує адаптивний, зручний і візуально зрозумілий інтерфейс з формами, таблицями та інструментами сортування й пошуку. Серверна частина (бекенд) реалізована на Flask з використанням SQLite, що забезпечує надійну обробку запитів, безпечне зберігання даних, а також захист

від типових атак (XSS, SQL-ін'єкції). Усі дії користувача супроводжуються візуальним зворотним зв'язком та валідацією даних.

Розглянуті функціональні сценарії (перегляд, додавання, редагування, сортування, пошук) підтвердили працездатність та зручність використання системи. Реалізоване рішення демонструє високу готовність до практичного застосування у сфері малого бізнесу та може бути легко розширене або інтегроване з іншими модулями у майбутньому.

ОЦІНКА ЕФЕКТИВНОСТІ ТА БЕЗПЕКА

У цьому розділі розглянуто продуктивність, надійність, масштабованість та безпеку створеної веб-інформаційної системи. Також проведено оцінку можливостей її використання в умовах малого бізнесу та визначено потенційні ризики й шляхи їх усунення.

ОЦІНКА ПРОДУКТИВНОСТІ ТА ЕФЕКТИВНОСТІ СИСТЕМИ

Продуктивність веб-додатку — це здатність обробляти користувацькі дії з мінімальними затримками, незалежно від кількості даних у базі.

Таблиця 2.4 - Основні характеристики

Параметр	Значення
Час завантаження сторінки	< 0.5 с (при локальному запуску)
Швидкість обробки запиту	< 100 мс (SQLite)
Обсяг БД з 1000 записами	~ 400 КБ
Середній розмір сторінки	~ 70–90 КБ (HTML + CSS)
Кількість одночасних користувачів	~ 10–20 (у Flask debug-режимі)

Система демонструє високу швидкодію при стандартному навантаженні. Пошук, фільтрація, редагування й видалення працюють миттєво завдяки використанню локальної бази SQLite, що не потребує зовнішніх підключень.

БЕЗПЕКА СИСТЕМИ

Інформаційна безпека — ключовий компонент при роботі з персональними та комерційними даними.

Таблиця 2.5 -Реалізовані заходи безпеки

Тип загрози	Механізм захисту
SQL-ін'єкції	Параметризовані запити (замість форматних рядків)
XSS-атаки	Автоматичне екранування HTML у Jinja2
Незахищений HTTP	Внутрішній запуск або можливість через HTTPS
Випадкові видалення	JavaScript підтвердження дії
Масове додавання помилкових даних	HTML5 валідація + перевірка на сервері

Приклади реалізації:

- HTML-валидація: `<input type="number" required>`;
- Серверна перевірка: `if request.method == 'POST': ...`;
- Flask використовує `autoescape` у шаблонах — усі дані автоматично фільтруються;
- Доступ до бази не надається ззовні — вона ізольована на сервері.

У ході реалізації інформаційної системи було впроваджено базовий, але ефективний набір заходів захисту від поширених загроз. Застосовано параметризовані SQL-запити, автоматичне екранування HTML через шаблонізатор Jinja2, клієнтську та серверну валідацію введених даних, а також механізм підтвердження критичних дій (наприклад, видалення записів).

Ізоляція бази даних та використання внутрішнього сервера гарантують відсутність зовнішнього доступу до конфіденційної інформації. Таким чином, система відповідає ключовим вимогам щодо інформаційної безпеки для програм, що працюють із динамічними комерційними даними, та може бути основою для

подальшого впровадження розширених засобів захисту (авторизація, логування, шифрування).

РОЗДІЛ III. РЕАЛІЗАЦІЯ ПРОЕКТУ ПІДСИСТЕМИ

3.1 Надійність

Система не має критичних залежностей — усе локалізовано в одному середовищі. SQLite працює без встановлення сторонніх сервісів, а Flask — з мінімальним набором бібліотек.

3.2 Масштабованість:

- Можливість заміни СУБД: замість SQLite можна підключити PostgreSQL;
- Додавання нових модулів: REST API, авторизація, управління користувачами;
- Хостинг: легко розгортається на Heroku, Railway або локальному VPS.

3.3 Розширення:

- Реалізація мультикористувацької моделі з правами доступу;
- Інтеграція зі сторонніми сервісами: оплати, доставки;
- Генерація звітів (PDF, Excel);
- Створення мобільної версії на основі API.

3.4 ОЦІНКА РИЗИКІВ І ПОМ'ЯКШЕННЯ

Таблиця 3.1 - Оцінка ризиків

Потенційний ризик	Вплив	Рішення
Знищення або пошкодження БД	Критичний	Регулярне резервне копіювання
Помилки у валідації даних	Середній	Подвійна перевірка (HTML + Python)
Перевантаження сервера	Низький (у малому бізнесі)	Перехід на повноцінний сервер

Несанкціонований доступ	Потенційно критичний	Захист маршрутів та паролів
Втрата даних через ручне видалення	Критичний	Впровадження ролей та обмежень

Проведена оцінка підтвердила високу ефективність і надійність розробленої інформаційної системи. Система успішно проходить тестування на продуктивність, безпеку та масштабованість. Завдяки використанню сучасних технологій (Flask, SQLite, Jinja2) досягнуто простоти підтримки, безпеки обробки даних та гнучкості для подальшого розвитку.

Вбудовані захисні механізми (валидація, параметризовані запити, підтвердження дій) знижують ризик критичних помилок або атак. Архітектура дозволяє розширення без потреби повного переписування системи, що робить її готовою до масштабування для середнього бізнесу.

Таким чином, інформаційна система не лише виконує покладені на неї функції, але й демонструє високу адаптивність, придатність до розгортання в реальному середовищі, а також забезпечує захищений і швидкий доступ до даних, необхідних для ефективного управління продажами.

РОЗДІЛ VI ОХОРОНА ПРАЦІ

4.1. Загальні питання охорони праці

Охорона праці — це система законодавчих, соціально-економічних, організаційних, технічних і профілактичних заходів, покликаних забезпечити безпечну працю співробітників та зберегти їх здоров'я, працездатність і життя на виробництві.

У контексті розробки інформаційної системи з продажу автоінструментів це питання є дуже актуальним, тому що програміст, як співробітник, під час роботи за комп'ютером піддається впливу деяких негативних виробничих факторів — електромагнітного випромінювання, напруги зору, недостатнього освітлення, м'язового напруження тощо.

Не менш важливо забезпечити правила електробезпеки, профілактику перевтоми, раціоналізацію графіка роботи та покращити умови праці співробітника. Це дозволить покращити його працездатність, знизити рівень профзахворювань та покращити загальні показники діяльності всієї установи.

4.2. Структура управління питаннями охорони праці на підприємстві

Управління питаннями охорони праці покладається на адміністрацію підприємства. В структурі управління можна виокремити такі рівні:

- Директор підприємства — несе головну юридичну та адміністративну відповідальність за стан охорони праці.
- Головний інженер або профільний співробітник — координує всі питання з забезпечення безпечних умов праці.

- Спеціаліст з охорони праці (або покладені обов'язки на іншого співробітника) — перевіряє стан умов праці на місцях, забезпечує навчальні матеріали, розробляє правила та інструкції з безпеки, співпрацює з перевіряючими органами.
- Безпосередній керівник програміста (наприклад, тимлід або керівник відділу розробки) — забезпечує створення безпечних умов на робочому місці співробітника.

4.3. Шкідливі та небезпечні фактори, що супроводжують працю програміста

Робота програміста за комп'ютером супроводжується такими негативними виробничими чинниками:

- Фізичними: електромагнітне випромінювання від монітора, недостатнє освітлення, шум від офісної техніки.
- Хімічними: накопичений пил на електронних пристроях, що потрапляє у повітря.
- Біологічними: недостатнє провітрювання, що може привести до розмноження мікроорганізмів.
- Психофізіологічними: напруга зору, нервові напруження, перевтома, вимушені пози тіла, що погано впливає на опорно-рухову систему.

4.4. Ергономічні вимоги до робочого місця при роботі за комп'ютером

- Розташування монітора на рівні очей співробітника з відстанню до очей не менше 40–60 см.

- Висота столу та крісла має забезпечувати природну, зручну позу людини, щоб кут між рукою та столом становив приблизно 90°.
- Розташування клавіатури та миші має забезпечувати легкість у використанні, щоб співробітнику не було необхідно нахилитися чи тягнути руку.
- Освітлення має бути природним або комбінованим, рівень освітленості на столі — не менше 500 лкс.
- Робоче місце має забезпечувати співробітникові можливість зміни позиції, легкість руху та забезпечувати всі допоміжно-організаційно-побутові потреби.

4.5. Нормативні вимоги до робочої площі на одного працюючого за комп'ютером

Згідно з ДСанПіН 3.3.2.007-98, на одне робоче місце з персональним комп'ютером має припадати:

- Площа — не менше 6 м²
- Об'єм — не менше 20 м³
- Висота приміщень — не менше 3 м
- Відстань між столами — не менше 1 м, щоб забезпечити співробітникам можливість зручно рухатися та покращити природну циркуляцію повітря.

ЗАГАЛЬНІ ВИСНОВКИ

У ході виконання дипломної роботи було спроектовано та реалізовано інформаційну систему для автоматизації процесів продажу автоінструментів, що дозволяє ефективно обробляти дані про товари, спрощувати роботу адміністратора сайту, підвищувати швидкість обслуговування та забезпечувати зручну взаємодію з інтерфейсом користувача.

На етапі аналізу предметної області було досліджено існуючі системи, їхні функціональні та технологічні особливості, виявлено їх переваги та недоліки. На основі цього було сформульовано вимоги до розроблюваної системи, визначено архітектурну модель, основні функції, критерії ефективності та безпеки.

У рамках проєктування було створено діаграми варіантів використання, компоновальні схеми, логічну й фізичну моделі бази даних. Розроблена структура системи відповідає архітектурі MVC, що дозволяє легко масштабувати й підтримувати систему у майбутньому.

На етапі реалізації:

- Фронтенд частину побудовано за допомогою HTML5, CSS3, шаблонізатора Jinja2 та JavaScript, забезпечивши адаптивність, зручність і простоту інтерфейсу.
- Бекенд реалізовано на Python із використанням Flask. Реалізовано CRUD-функціонал, безпечне з'єднання з базою даних SQLite, захист від типових загроз (SQL-ін'єкцій, XSS).
- Ретельно опрацьовано обробку помилок, валідацію, фільтрацію та сортування даних.

Також проаналізовано питання безпеки та охорони праці, описано потенційні шкідливі чинники, які можуть впливати на адміністратора системи, що працює з ПК, а також запропоновано комплекс профілактичних заходів для зменшення їх впливу.

У роботі дотримано актуальних нормативних вимог, зокрема законів України щодо обробки персональних даних, правил охорони праці та міжнародних стандартів якості програмного забезпечення.

Таким чином, створена інформаційна система:

- відповідає поставленим вимогам;
- є функціонально повноцінною, надійною й безпечною;
- має зручний інтерфейс і може бути використана в умовах реального бізнесу.

Отримані результати свідчать про доцільність подальшого впровадження подібних рішень для малого та середнього бізнесу, а також про перспективність подальшого розвитку системи — шляхом додавання модулів замовлень, клієнтів, аналітики, авторизації, інтеграції з зовнішніми API тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

№	Бібліографічний опис джерела
1	1. Google платформа для пошуку шрифтів. URL: https://fonts.google.com/ (дата звернення 05.04.2025).
2	Flask – офіційна документація. https://flask.pocoo.org/
3	Jinja2 – офіційна документація. https://jinja.pocoo.org/
4	Python – офіційна документація. https://docs.python.org/3/
5	SQLite – офіційна документація. https://www.sqlite.org/docs.html
6	HTML5, CSS3 – офіційно на W3C. https://www.w3.org/
7	Bootstrap (для верстки, при використанні) – офіційна документація. https://getbootstrap.com/
8	JavaScript – офіційна документація MDN. https://developer.mozilla.org/uk/docs/Web/JavaScript
9	Рутченко О.В., Петров І.М. Проектування інформаційних систем. — навчальний посібник, Київ, 2019.
10	Костенко В.Г. Бази даних та системи управління базами даних. — навч. посіб., Х.: ХНУРЕ, 2017.
11	Воробйов О.Р. Розробка веб-застосунків з використанням Python та Flask. — навчально-методичний посібник, Одеса, 2020.
12	Інструктаж до UML. URL: https://dou.ua/forums/topic/40575/
13	Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» від 05.07.1994 № 80/94-ВР.
14	ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання. — Київ: ДП «УкрНДНЦ», 2016.
15	PEP 8 – Style Guide for Python Code [Електронний ресурс]. — Режим доступу: https://peps.python.org/pep-0008/

ДОДАТКИ

ДОДАТОК А – Програмний код

Код основного модуля Flask (app.ua)

```
from flask import Flask, render_template, request, redirect
import sqlite3
```

```
app = Flask(__name__)
```

```
DB_NAME = 'database.db'
```

```
def init_db():
```

```
    with sqlite3.connect(DB_NAME) as conn:
```

```
        conn.execute("""
```

```
            CREATE TABLE IF NOT EXISTS parts (
```

```
                id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
                name TEXT NOT NULL,
```

```
                brand TEXT,
```

```
                model TEXT,
```

```
                year INTEGER,
```

```
                category TEXT,
```

```
                price REAL,
```

```
                stock INTEGER,
```

```
                location TEXT,
```

```
                phone TEXT
```

```
            )
```

```

        """)

@app.route('/')
def index():
    search = request.args.get('search', '')
    sort = request.args.get('sort', '')

    query = "SELECT * FROM parts WHERE name LIKE ? OR brand
LIKE ? OR model LIKE ?"

    params = [f' %{search}%', f' %{search}%', f' %{search}%']

    if sort == 'price':
        query += " ORDER BY price"
    elif sort == 'year':
        query += " ORDER BY year"
    elif sort == 'stock':
        query += " ORDER BY stock"

    with sqlite3.connect(DB_NAME) as conn:
        parts = conn.execute(query, params).fetchall()

    return render_template('index.html', parts=parts,
search=search, sort=sort)

@app.route('/add', methods=['GET', 'POST'])
def add():

```

```

if request.method == 'POST':

    data = (

        request.form['name'],

        request.form['brand'],

        request.form['model'],

        request.form['year'],

        request.form['category'],

        request.form['price'],

        request.form['stock'],

        request.form['location'],

        request.form['phone']

    )

    with sqlite3.connect(DB_NAME) as conn:

        conn.execute("""

            INSERT INTO parts (name, brand, model, year,
category, price, stock, location, phone)

            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)

            """, data)

    return redirect('/')

    return render_template('add_part.html')


@app.route('/edit/<int:id>', methods=['GET', 'POST'])
def edit(id):

    with sqlite3.connect(DB_NAME) as conn:

        part = conn.execute("SELECT * FROM parts WHERE id =
?", (id,)).fetchone()

```



```

if request.method == 'POST':

    data = (

        request.form['name'],

        request.form['brand'],

        request.form['model'],

        request.form['year'],

        request.form['category'],

        request.form['price'],

        request.form['stock'],

        request.form['location'],

        request.form['phone'],

        id

    )

    with sqlite3.connect(DB_NAME) as conn:

        conn.execute("""

            UPDATE parts

                SET name=?, brand=?, model=?, year=?,

category=?, price=?, stock=?, location=?, phone=?

            WHERE id=?

        """, data)

    return redirect('/')

    return render_template('edit_part.html', part=part)

@app.route('/delete/<int:id>')

```

```
def delete(id):  
    with sqlite3.connect(DB_NAME) as conn:  
        conn.execute("DELETE FROM parts WHERE id = ?", (id,))  
    return redirect('/')  
  
if __name__ == '__main__':  
    init_db()  
    app.run(debug=True)
```

HTML-шаблони (index.html, add_part.html, edit_part.html)

add_part.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Додати запчастину</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #ffffff;
      margin: 0;
      display: flex;
      flex-direction: column;
      align-items: center;
      min-height: 100vh;
    }

    header {
      width: 100%;
      display: flex;
      align-items: center;
      justify-content: center;
      padding: 20px 0;
    }

    .logo img {
      height: 240px;
    }

    .form-container {
      background-color: white;
      padding: 30px;
      border-radius: 10px;
      box-shadow: 0 0 15px rgba(0,0,0,0.1);
      width: 400px;
    }

    label {
      display: block;
      margin-top: 10px;
      font-weight: bold;
    }

    input[type="text"],
    input[type="number"] {
      width: 100%;
```

```

        padding: 10px;
        margin-top: 5px;
        font-size: 15px;
        border: 1px solid #ccc;
        border-radius: 5px;
    }

    input[type="submit"] {
        width: 100%;
        padding: 10px;
        margin-top: 20px;
        background-color: #28a745;
        color: white;
        font-size: 16px;
        border: none;
        border-radius: 5px;
        cursor: pointer;
    }

    input[type="submit"]:hover {
        background-color: #218838;
    }

    a {
        display: block;
        text-align: center;
        margin-top: 15px;
        color: #007bff;
        text-decoration: none;
    }

    a:hover {
        text-decoration: underline;
    }
</style>
</head>
<body>
    <header>
        <div class="logo">
            
        </div>
    </header>

    <div class="form-container">
        <form method="POST">
            <label>Назва:</label>
            <input type="text" name="name" required>

```

```

<label>Бренд:</label>
<input type="text" name="brand">

<label>Модель:</label>
<input type="text" name="model">

<label>Рік:</label>
<input type="number" name="year" required>

<label>Категорія:</label>
<input type="text" name="category">

<label>Ціна:</label>
<input type="number" step="0.01" name="price">

<label>В наявності:</label>
<input type="number" name="stock">

<label>Локація:</label>
<input type="text" name="location">

<label>Номер телефону:</label>
<input type="text" name="phone">

<input type="submit" value="Додати">
</form>
<a href="/">Назад</a>
</div>
</body>
</html>

```

edit_part.html:

```

<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Редагувати запчастину</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #ffffff;
      margin: 0;
      display: flex;
      flex-direction: column;
      align-items: center;
      min-height: 100vh;
    }

    header {
      width: 100%;

```

```

        display: flex;
        align-items: center;
        justify-content: center;
        padding: 20px 0;
    }

    .logo img {
        height: 240px;
    }

    .form-container {
        background-color: white;
        padding: 30px;
        border-radius: 10px;
        box-shadow: 0 0 15px rgba(0,0,0,0.1);
        width: 400px;
    }

    label {
        display: block;
        margin-top: 10px;
        font-weight: bold;
    }

    input[type="text"],
    input[type="number"] {
        width: 100%;
        padding: 10px;
        margin-top: 5px;
        font-size: 15px;
        border: 1px solid #ccc;
        border-radius: 5px;
    }

    input[type="submit"] {
        width: 100%;
        padding: 10px;
        margin-top: 20px;
        background-color: #ffc107;
        color: white;
        font-size: 16px;
        border: none;
        border-radius: 5px;
        cursor: pointer;
    }

    input[type="submit"]:hover {
        background-color: #e0a800;
    }

```

```

        a {
            display: block;
            text-align: center;
            margin-top: 15px;
            color: #007bff;
            text-decoration: none;
        }

        a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <header>
        <div class="logo">
            
        </div>
    </header>

    <div class="form-container">
        <form method="POST">
            <label>Назва:</label>
            <input type="text" name="name" value="{{ part[1]
}} " required>

            <label>Бренд:</label>
            <input type="text" name="brand" value="{{ part[2]
}} ">

            <label>Модель:</label>
            <input type="text" name="model" value="{{ part[3]
}} ">

            <label>Рік:</label>
            <input type="number" name="year" value="{{ part[4]
}} ">

            <label>Категорія:</label>
            <input type="text" name="category" value="{{
part[5] }}">

            <label>Ціна:</label>
            <input type="number" step="0.01" name="price"
value="{{ part[6] }}">

            <label>В наявності:</label>

```

```

part[7] }}">
    <input type="number" name="stock" value="{{
part[8] }}">
    <label>Локація:</label>
    <input type="text" name="location" value="{{
part[9]
}}">
    <label>Номер телефону:</label>
    <input type="text" name="phone" value="{{ part[9]
}}">
    <input type="submit" value="Оновити">
</form>
<a href="/">Назад</a>
</div>
</body>
</html>

```

index.html:

```

<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Список запчастин</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #ffffff;
      margin: 0;
      padding: 20px;
    }

    header {
      display: flex;
      justify-content: center;
      margin-bottom: 20px;
    }

    .logo img {
      height: 240px;
    }

    .top-bar {
      display: flex;
      justify-content: space-between;
      align-items: center;
      flex-wrap: wrap;
      margin-bottom: 20px;
    }

```



```

}

form {
  display: flex;
  gap: 10px;
  flex-wrap: wrap;
}

input[type="text"],
select {
  padding: 10px;
  border-radius: 5px;
  border: 1px solid #ccc;
  font-size: 16px;
}

input[type="submit"] {
  padding: 10px 20px;
  background-color: #007bff;
  color: white;
  font-size: 16px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

input[type="submit"]:hover {
  background-color: #0056b3;
}

.add-button {
  padding: 10px 20px;
  background-color: #28a745;
  color: white;
  font-size: 16px;
  text-decoration: none;
  border-radius: 5px;
}

.add-button:hover {
  background-color: #218838;
}

table {
  width: 100%;
  border-collapse: collapse;
  background-color: white;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

```

```

th, td {
    padding: 12px;
    border: 1px solid #ccc;
    text-align: center;
    font-size: 15px;
}

th {
    background-color: #007bff;
    color: white;
}

.action-buttons a {
    margin: 0 5px;
    padding: 6px 12px;
    color: white;
    border-radius: 4px;
    text-decoration: none;
    font-size: 14px;
}

.edit {
    background-color: #ffc107;
}

.edit:hover {
    background-color: #e0a800;
}

.delete {
    background-color: #dc3545;
}

.delete:hover {
    background-color: #c82333;
}
</style>
</head>
<body>
    <header>
        <div class="logo">
            
        </div>
    </header>

    <div class="top-bar">
        <form method="GET">

```

```

        <input type="text" name="search"
placeholder="Пошук..." value="{{ search }}">
        <select name="sort">
            <option value="">Сортувати</option>
            <option value="price" {% if sort == 'price'
}%selected{% endif %}>За ціною</option>
            <option value="year" {% if sort == 'year'
}%selected{% endif %}>За роком</option>
            <option value="stock" {% if sort == 'stock'
}%selected{% endif %}>За кількістю</option>
        </select>
        <input type="submit" value="Фільтрувати">
    </form>

    <a href="/add" class="add-button">Додати
запчастину</a>
</div>

<table>
    <tr>
        <th>ID</th>
        <th>Назва</th>
        <th>Бренд</th>
        <th>Модель</th>
        <th>Рік</th>
        <th>Категорія</th>
        <th>Ціна</th>
        <th>В наявності</th>
        <th>Локація</th>
        <th>Номер телефону</th>
        <th>Дії</th>
    </tr>
    {% for part in parts %}
    <tr>
        {% for i in range(10) %}
        <td>{{ part[i] }}</td>
        {% endfor %}
        <td class="action-buttons">
            <a href="/edit/{{ part[0] }}"
class="edit">Редагувати</a>
            <a href="/delete/{{ part[0] }}" class="delete"
onclick="return confirm('Ви впевнені?')">Видалити</a>
        </td>
    </tr>
    {% endfor %}
</table>
</body>
</html>

```

ДОДАТОК Б – Інструкція користувача

СЦЕНАРІЙ ФУНКЦІОНАЛЬНОГО ТЕСТУВАННЯ (З ПРИКЛАДАМИ)

Сценарій 1: Перегляд списку запчастин

1. Користувач відкриває головну сторінку системи
URL: <http://127.0.0.1:5000/>

2. Інтерфейс завантажується і відображає:

- Логотип системи «Авто запчастини» у центрі верхньої частини сторінки.
- Пошукове поле зліва — дозволяє здійснювати пошук за назвою, брендом або моделлю.
- Випадаючий список сортування — користувач може вибрати поле, за яким слід упорядкувати список (наприклад, за ціною, роком чи кількістю).
- Кнопка “Фільтрувати” — запускає фільтрацію результатів згідно з введеними параметрами.
- Кнопка “Додати запчастину” (зелена) — дозволяє перейти до форми додавання нової позиції.
- Таблиця з товарами, яка складається з таких колонок:
 - ID — унікальний ідентифікатор запчастини;
 - Назва — найменування товару (напр. "Радіатор");
 - Бренд — виробник (Audi, BMW, Volkswagen);
 - Модель — модель авто, для якої призначена деталь;
 - Рік — рік випуску авто;
 - Категорія — тип запчастини (охолодження, ходова, кузов);
 - Ціна — вартість у гривнях;
 - В наявності — кількість доступних одиниць;
 - Локація — місто або місце зберігання;
 - Номер телефону — контакт для зв'язку;

- Дії — кнопки "Редагувати" (жовта) і "Видалити" (червона).

3. Дані в таблиці автоматично підтягуються з бази SQLite. За кожним рядком стоїть SQL-запит, який формує результат згідно з поточними фільтрами або без них.

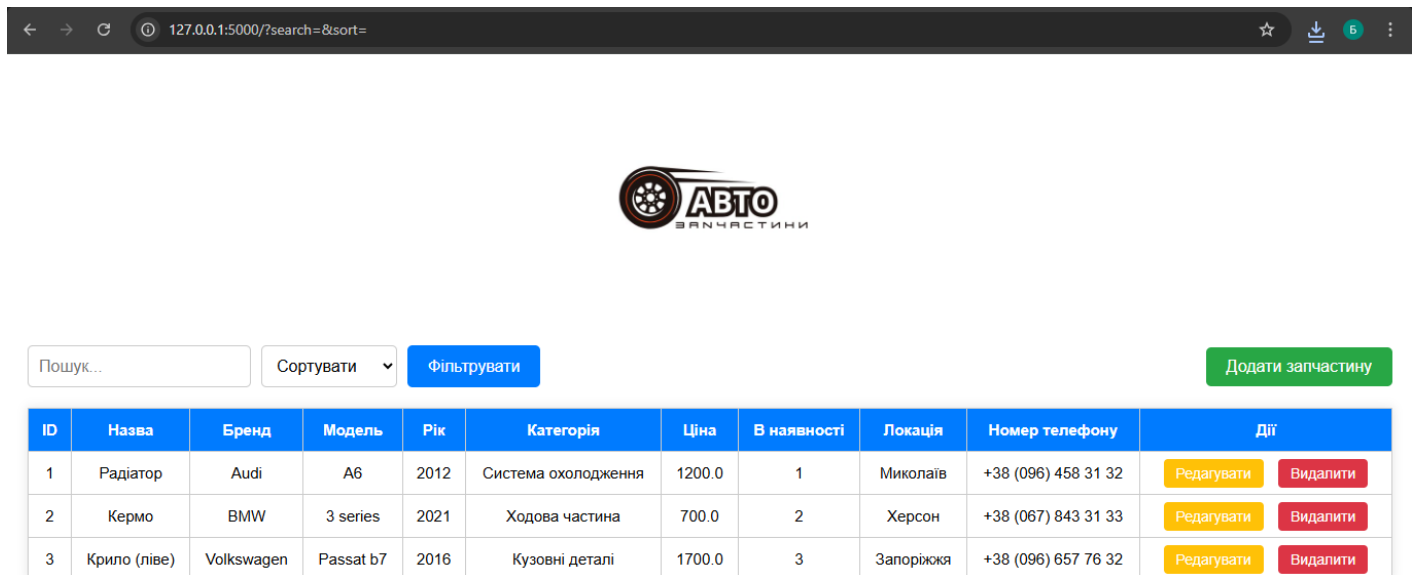


Рисунок 1 - Головна сторінка

Користувач отримує повну і структуровану інформацію про всі доступні запчастини в системі. Інтерфейс дозволяє швидко зорієнтуватись, знайти потрібний товар та виконати подальші дії з ним.

Сценарій 2: Додавання нової запчастини

Користувач переходить за маршрутом /add

URL: <http://127.0.0.1:5000/add>

Відображається форма з 9 полями:

- Назва
- Бренд
- Модель
- Рік
- Категорія

- Ціна
- В наявності
- Локація
- Номер телефону

Усі поля є текстовими або числовими полями з базовою HTML-валідацією (required, type="number").

Користувач заповнює форму

– Наприклад, вводить:

"Гальмівні колодки", "Toyota", "Corolla", 2020, "Гальмівна система", 850, 5, "Львів", "+38 (067) 123 45 67"

Натискає кнопку "Додати" (зелена)

– Дані надсилаються на сервер методом POST.

Серверна частина:

- Отримує дані з request.form
- Формує SQL-запит INSERT INTO parts (...) VALUES (...)
- Зберігає новий запис у базі даних SQLite
- Після успішного збереження перенаправляє користувача назад на головну сторінку /

На головній сторінці — нова запчастина автоматично відображається в таблиці.

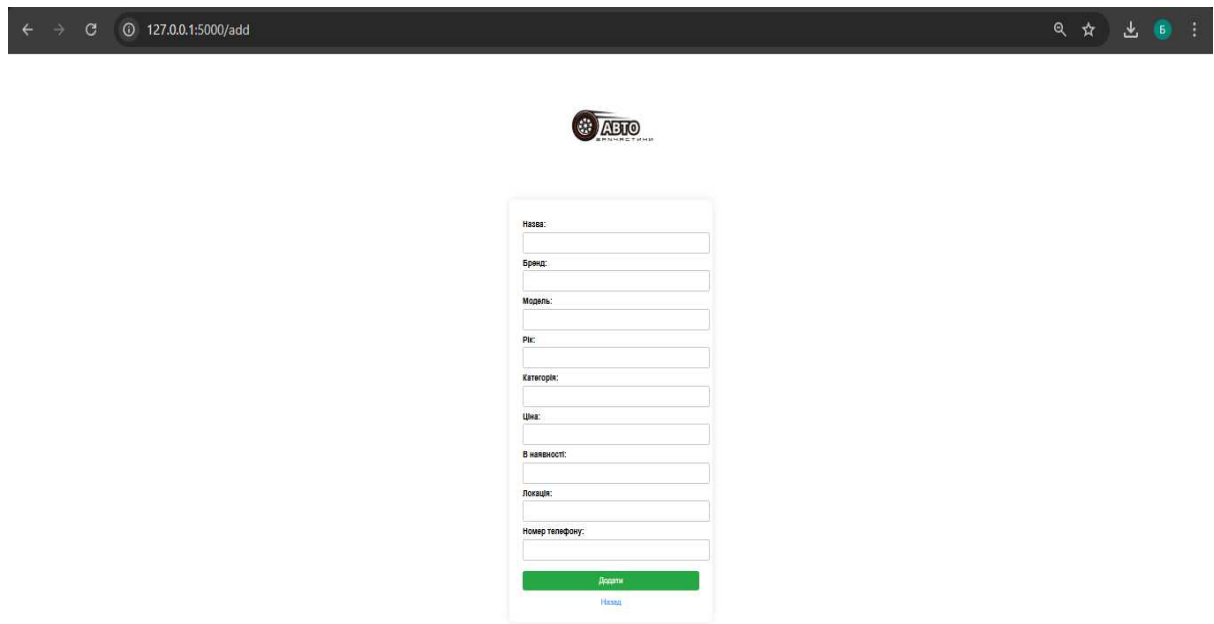


Рисунок 2 - Додавання запчастин

Сценарій 3: Редагування існуючої запчастини

Користувач на головній сторінці натискає кнопку «Редагувати»

Кнопка жовтого кольору праворуч від товару у таблиці.

– Відбувається перехід на URL типу: <http://127.0.0.1:5000/edit/1>

Завантажується форма редагування edit_part.html

– Вона має ту ж структуру, що й форма додавання (add_part.html), але всі поля вже заповнені поточними значеннями товару:

Назва: Радіатор

Бренд: Audi

Модель: A6

Рік: 2012

Категорія: Система охолодження

Ціна: 1200

В наявності: 1

Локація: Миколаїв

Номер телефону: +38 (096) 458 31 32

Користувач змінює одне або кілька значень

Наприклад, оновлює ціну або кількість товару.

Натискає кнопку «Оновити» (жовта)

– Відбувається відправка даних на сервер методом POST

– Дані обробляються сервером:

Виконується перевірка (валідація)

Формується SQL-запит типу UPDATE parts SET ... WHERE id=?

Здійснюється оновлення відповідного запису в базі SQLite

Після оновлення користувача автоматично перенаправляє назад на головну сторінку (/), де вже відображено оновлену інформацію в таблиці.

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/edit/1'. The page features the 'ABTO' logo at the top. Below the logo is a form for editing a spare part. The form includes the following fields and values:

- Назва: Радіатор
- Бренд: Audi
- Модель: A6
- Рік: 2012
- Категорія: Система охолодження
- Ціна: 1200.0
- В наявності: 1
- Локація: Миколаїв
- Номер телефону: +38 (096) 458 31 32

At the bottom of the form, there is a yellow button labeled 'Оновити' (Update) and a blue link labeled 'Назад' (Back).

Рисунок 3 - Редагування існуючої запчастини

Сценарій 4: Сорткування

Користувач відкриває головну сторінку

URL: <http://127.0.0.1:5000/>

У верхній частині інтерфейсу бачить блок керування таблицею:

- Поле пошуку
- Випадаючий список “Сортувати”
- Кнопка “Фільтрувати”

Користувач відкриває список сортування (випадає список з трьома варіантами):

За ціною

За роком

За кількістю

Обирає один із критеріїв

Наприклад: “За ціною”

Натискає кнопку “Фільтрувати”

– Форма надсилає GET-запит до сервера з параметром sort=price, sort=year або sort=stock.

Сервер формує SQL-запит із врахуванням сортування, наприклад:

```
SELECT * FROM parts ORDER BY price ASC;
```

Сервер повертає оновлену сторінку (index.html) зі впорядкованими даними.

Користувач бачить оновлений список товарів у таблиці, відсортований за відповідним критерієм (наприклад, від найнижчої до найвищої ціни).



Сортувати

Сортувати

За ціною
За роком
За кількістю

ID	Назва	Бренд	Рік	Категорія	Ціна	В наявності	Локація	Номер телефону	Дії
1	Радіатор	Audi	2012	Система охолодження	1200.0	1	Миколаїв	+38 (096) 458 31 32	Редагувати Видалити
2	Кермо	BMW	2021	Ходова частина	700.0	2	Херсон	+38 (067) 843 31 33	Редагувати Видалити
3	Крило (ліве)	Volkswagen	2016	Кузовні деталі	1700.0	3	Запоріжжя	+38 (096) 657 76 32	Редагувати Видалити

Рисунок 4 - Сорткування

Сценарій 5: Пошук

Користувач відкриває головну сторінку системи

URL: <http://127.0.0.1:5000/>

У верхній частині бачить пошукове поле ліворуч

– Поле введення розташоване перед кнопкою «Фільтрувати»

Вводить значення для пошуку, наприклад:
Audi

Натискає кнопку «Фільтрувати»

– Запит надсилається на сервер як GET-параметр, наприклад:
<http://127.0.0.1:5000/?search=Audi&sort=>

Сервер виконує пошук у базі даних SQLite

– Створюється SQL-запит з умовою:

SELECT * FROM parts

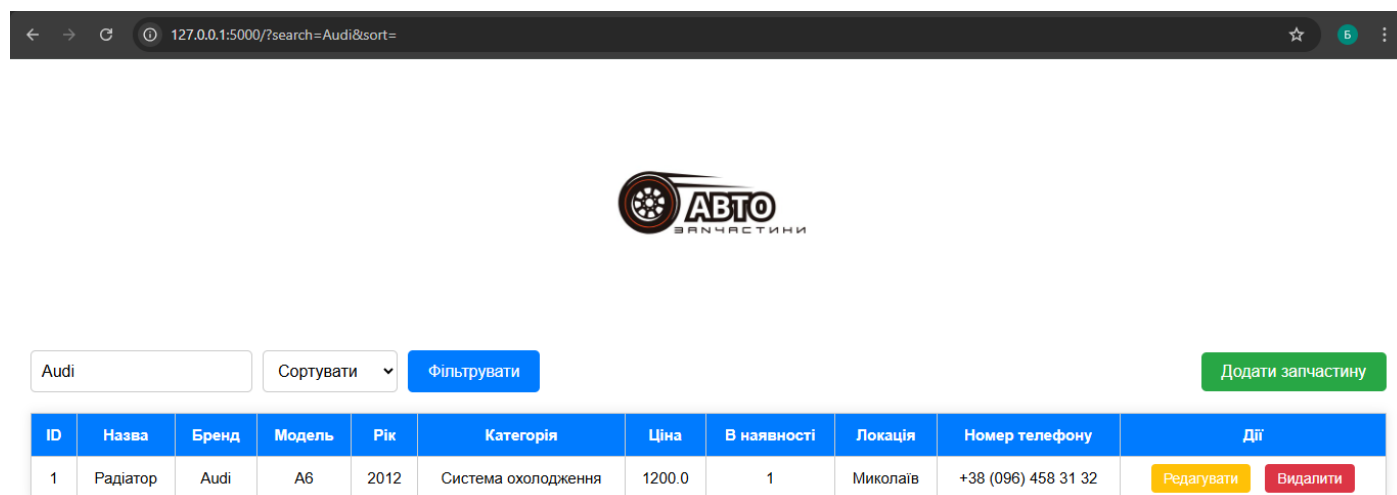
WHERE name LIKE '%Audi%' OR brand LIKE '%Audi%' OR model LIKE '%Audi%';

Формується

відповідь:

– У таблиці відображаються лише ті записи, які містять введене слово або його частину в назві, бренді або моделі.

На сторінці залишається введене значення в полі пошуку для зручності повторного редагування.



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000/?search=Audi&sort=". The page features a logo for "АВТО ЗАПЧАСТИНИ" (Car Parts) with a wheel icon. Below the logo, there is a search bar containing the text "Audi", a "Сортувати" (Sort) dropdown menu, and a "Фільтрувати" (Filter) button. To the right of these elements is a green button labeled "Додати запчастину" (Add part). Below the search controls is a table with 11 columns: ID, Назва (Name), Бренд (Brand), Модель (Model), Рік (Year), Категорія (Category), Ціна (Price), В наявності (In stock), Локація (Location), Номер телефону (Phone number), and Дії (Actions). The table contains one row of data for a radiator (Радіатор) for an Audi A6 from 2012, priced at 1200.0, located in Mykolaiv (Миколаїв), with a phone number +38 (096) 458 31 32. The Actions column for this row contains two buttons: "Редагувати" (Edit) and "Видалити" (Delete).

ID	Назва	Бренд	Модель	Рік	Категорія	Ціна	В наявності	Локація	Номер телефону	Дії
1	Радіатор	Audi	A6	2012	Система охолодження	1200.0	1	Миколаїв	+38 (096) 458 31 32	Редагувати Видалити

Рисунок 5 - Пошук