

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 126 «Інформаційні системи та технології»
(освітня програма «Системи бізнес-аналітики»)

на тему:

«Платформа для онлайн-продажу автозапчастин та аксесуарів»

Виконав студент групи ІСТ-20-1
КОЗАК Юрій Олександрович

Керівник роботи:
КУЗЬМЕНКО Олександр Вікторович

Рецензент:
СВІНЦИЦЬКА Олександра Миколаївна

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

«ЗАТВЕРДЖУЮ»

Завідувач кафедри комп'ютерних наук
Марина ГРАФ
«25» січня 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 126 «Інформаційні системи та технології»
(освітня програма «Системи бізнес-аналітики»)

Здобувач вищої освіти: **КОЗАК Юрій Олександрович**

Керівник роботи: **КУЗЬМЕНКО Олександр Вікторович**

Тема роботи: **«Платформа для онлайн-продажу автозапчастин та аксесуарів»**,
затверджена Наказом закладу вищої освіти від **«25» січня 2024 р., №24/с**

Вихідні дані для роботи: **Об'єктом дослідження є платформа для продажу автозапчастин та аксесуарів через Інтернет. Предметом дослідження є розробка механіки продажу товарів через онлайн платформу.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	КУЗЬМЕНКО О.В	05.02.2024	06.02.2024
2	КУЗЬМЕНКО О.В	12.03.2024	14.03.2024
3	КУЗЬМЕНКО О.В	08.04.2024	12.04.2024

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	02 березня 2024 – 05 квітня 2024	Виконано
2	Проектування структури системи	6 квітня 2024 – 12 квітня 2024	Виконано
3	Написання програмного коду	13 квітня 2024 – 26 квітня 2024	Виконано
4	Тестування	27 квітня 2024 – 04 травня 2024	Виконано
5	Підготовка пояснювальної записки	5 травня 2024 – 09 червня 2024	Виконано
6	Попередній захист	10 червня 2024	Виконано

Здобувач вищої освіти **Юрій КОЗАК**
Керівник **Олександр КУЗЬМЕНКО**

РЕФЕРАТ

Випускна робота бакалавра складається з backend та frontend частини веб-сайту для продажу запчастин для автомобілів та пояснювальної записки. Пояснювальна записка до випускної роботи містить 60 сторінку, 24 ілюстрацій.

Метою роботи є розробка веб-застосунку для продажу автомобільних запчастин та аксесуарів.

В роботі визначено основні завдання на розробку системи, проаналізовано аналоги розробленої системи. Наведено сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис алгоритмів взаємодії окремих модулів системи, опис об'єктної структури системи (діаграма класів). Також наводяться результати тестування системи.

Ключові слова: CarParts, есомерсе, веб-сайт, продаж автомобільних компонентів.

					ІСТ.КР.Б – 126 – 24 – ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Платформа для онлайн-продажу автозапчастин та аксесуарів Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Ю.О. КОЗАК					4	46
Керівник		О.В. КУЗЬМЕНКО				Житомирська політехніка, група ІСТ-20-1		
Рецензент		О.М. СВИНЦИЦЬКА						
Н. контр.								
Зав. каф.		М.С. ГРАФ						

ABSTRACT

The bachelor's thesis consists of the backend and frontend parts of a website for selling car parts, along with an explanatory note. The explanatory note for the thesis contains 60 pages, 24 illustrations.

The purpose of the work is the development of a web application for the sale of car spare parts and accessories.

The thesis defines the main tasks for the development of the system and analyzes the analogs of the developed system. It provides usage scenarios for the software complex, the general structure of the software complex, a description of the algorithms for the interaction of individual system modules, and a description of the object structure of the system (class diagram). The results of system testing are also presented.

Keywords: CarParts, website, car component sales, ecommerce.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		5

Зміст

Зміст.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	10
1.1. Постановка задачі	10
1.2. Аналіз аналогів програмного продукту	11
1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	14
Висновки до першого розділу	17
РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	19
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	19
2.2. Розробка бази даних системи	24
2.3. Проектування та реалізація алгоритмів роботи системи.....	26
2.4. Реалізація додатку.....	34
Висновки до другого розділу.....	40
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ДОДАТКОМ	42
3.1. Структура інтерфейсу. Інтерфейс та порядок роботи з системою	42
3.2. Тестування роботи системи	50
Висновки до третього розділу	55
ВИСНОВКИ.....	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59

ВСТУП

У сучасному світі онлайн-покупок і глобалізованого ринку ця діяльність стає найбільш значущою. Те ж саме можна сказати і про вузькоспеціалізовані товари, такі як автомобільні запчастини. Не існує жодного іншого сегмента ринку, де існує більш жорстка конкуренція, ніж у випадку з автозапчастинами. Зростаюча кількість транспортних засобів у всьому світі стимулює попит на всі види компонентів. Відповідно, попит стимулює розвиток інтернет-магазинів та різноманітних платформ для їх продажу.

Автозапчастини - це специфічний товар, який вимагає більшої уваги до деталей, точної класифікації та надійних постачальників. Щодо традиційних способів продажу запчастин через фізичні магазини та склади, то для такого способу розвитку бізнесу існують суттєві обмеження: мінімальний асортимент, географічна прив'язка, високі витрати на утримання великих запасів товару. Всі ці проблеми вирішують онлайн-платформи: доступ до величезного асортименту запчастин і простота їх пошуку та замовлення, економія часу і коштів як для продавців, так і для покупців.

Метою даної роботи є розробка ефективного, зручного та інтуїтивно зрозумілого віртуального ресурсу для торгівлі запасними частинами до транспортних засобів; такий сайт буде зручно надавати користувачам доступ до лінійки доступних товарів, надійної системи пошуку, інструментів фільтрації, а також доступ до зручних інструментів для управління замовленнями та залишками. Для досягнення цієї мети Для досягнення цієї мети необхідно було вирішити низку завдань. Найпершим кроком у цьому напрямку є аналіз ринку автомобільних запчастин та існуючих на ньому рішень для онлайн-продажів. На основі цього аналізу слід визначити вимоги до функціональності та архітектури обраної платформи з урахуванням потреб її потенційних користувачів. Тепер необхідно розробити "технічну специфікацію платформи, включаючи її основні компоненти та модулі". Також необхідно розробити прототипи користувацького інтерфейсу, який би враховував принципи зручності та ергономіки.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		7

Потім реалізуються основні функції платформи інтернет-магазину, включаючи каталог товарів, систему пошуку та фільтрації, кошик замовлень, систему реєстрації та авторизації користувачів. Потім відбувається тестування створеної платформи з метою виявлення та усунення можливих недоліків. І завершальним етапом є оцінка ефективності роботи платформи та розробка рекомендацій щодо її подальшого розвитку та вдосконалення.

Для виконання роботи були використані наступні методи дослідження: аналіз літератури, порівняльний аналіз існуючих рішень, проектування архітектури системи та прототипів інтерфейсу, програмування функціональних модулів платформи, тестування роботи системи та емпіричний аналіз зворотного зв'язку з користувачами.

Робота складається з кількох основних розділів. У вступі описано актуальність теми, сформульовано мету та завдання, методи дослідження та структуру роботи. Далі в роботі проводиться огляд літератури та аналогів, в якому проаналізовано існуючі рішення та дослідження у сфері онлайн-продажу запчастин для автомобілів. Блок проектування платформи включає технічне завдання, архітектуру системи та прототипи інтерфейсу платформи. Реалізація - опис процесу розробки платформи, використаних технологій та інструментів.

Тестування та результати - аналіз проведених тестів, знайдених проблем, а також їх вирішення. Робота завершується висновками та рекомендаціями, які включають загальні висновки по роботі, оцінку досягнення мети та поставлених завдань, пропозиції щодо подальшого розвитку платформи та поставлених завдань, а також пропозиції щодо подальшого розвитку платформи.

Електронні майданчики з продажу автозапчастин є одним з перспективних кроків сучасного бізнесу, який відповідає як викликам ринку, так і потребам споживача. Робота над розробкою онлайн-платформи для продажу автозапчастин передбачає ускладнення підходу, що складається з ета-

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		8

пів аналізу ринку, проектування, програмування та тестування. В даній дипломній роботі буде зроблена спроба створити таке функціональне та зручне рішення, яке забезпечить ефективний процес купівлі-продажу автозапчастин.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1. Постановка задачі

Інтернет-магазин автозапчастин повинен забезпечувати зручну для користувачів і системних адміністраторів процедуру купівлі або продажу товарів у найбільш зручний і ефективний спосіб. Для цього необхідно сформулювати як функціональні, так і нефункціональні вимоги в рамках даної платформи.

Користувацький Інтерфейс:

- Реєстрація та авторизація користувачів;
- Профіль користувача з можливістю редагування особистої інформації;
- Каталог товарів з можливістю перегляду деталей кожного товару;
- Пошук та фільтрація товарів за різними критеріями;
- Кошик замовлень з можливістю додавання/видалення товарів, перегляду загальної вартості замовлення;
- Оформлення замовлення з вибором способу оплати та доставки;
- Відгуки та рейтинг товарів.

Адміністративний інтерфейс:

- Управління користувачами;
- Управління товарами;
- Управління замовленнями (перегляд, зміна статусу, обробка) ;
- Управління відгуками та рейтингами.

Таким чином, створення інтернет-магазину для торгівлі автозапчастинами передбачає реалізацію набору функціональних компонентів, які повинні забезпечити, з одного боку, простоту та зручність інтерфейсу користувача, з іншого - простоту адміністрування системи та надійність бекенду в частині зберігання та обробки даних. У свою чергу, кожен з цих компонентів повинен бути реалізований на сучасній, масштабованій та безпечній платформі, яка б відповідала потребам ринку та забезпечувала ефективний онлайн-продаж та купівлю автозапчастин.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		10

1.2. Аналіз аналогів програмного продукту

Аналіз проводиться на основі системних вимог та особливостей предметної області. Результати цього аналізу будуть основою для обґрунтування вибору платформи, технології та середовища для розробки програмного забезпечення. Нижче наведено конкуренти, які впливали на прийняття основних рішень щодо обґрунтування вибору:

<https://www.autodoc.co.uk/> – Це інтернет-магазин автозапчастин, де обговорюються автозапчастини та аксесуари для різних транспортних засобів. На сайті продаються запчастини для багатьох марок автомобілів, включаючи BMW, Mercedes-Benz і Audi. Вони також мають підручники та посібники, які допоможуть з ремонтом автомобіля.

Переваги:

- Величезний асортимент автозапчастин та аксесуарів: Autodoc пропонує величезний асортимент автозапчастин та аксесуарів для широкого спектру марок автомобілів;
- Конкурентоспроможні ціни: Користувач може отримати дешевші ціни на автозапчастини в інтернет-магазинах;
- Доступна інформація: Даний інтернет-магазин часто має досить багато інформації про автозапчастини, які продає.

Недоліки:

- Складність в оцінці якості;
- Ризик отримати не ту деталь;
- Не вказують терміни доставки;
- Повернення може бути проблематичним.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		11

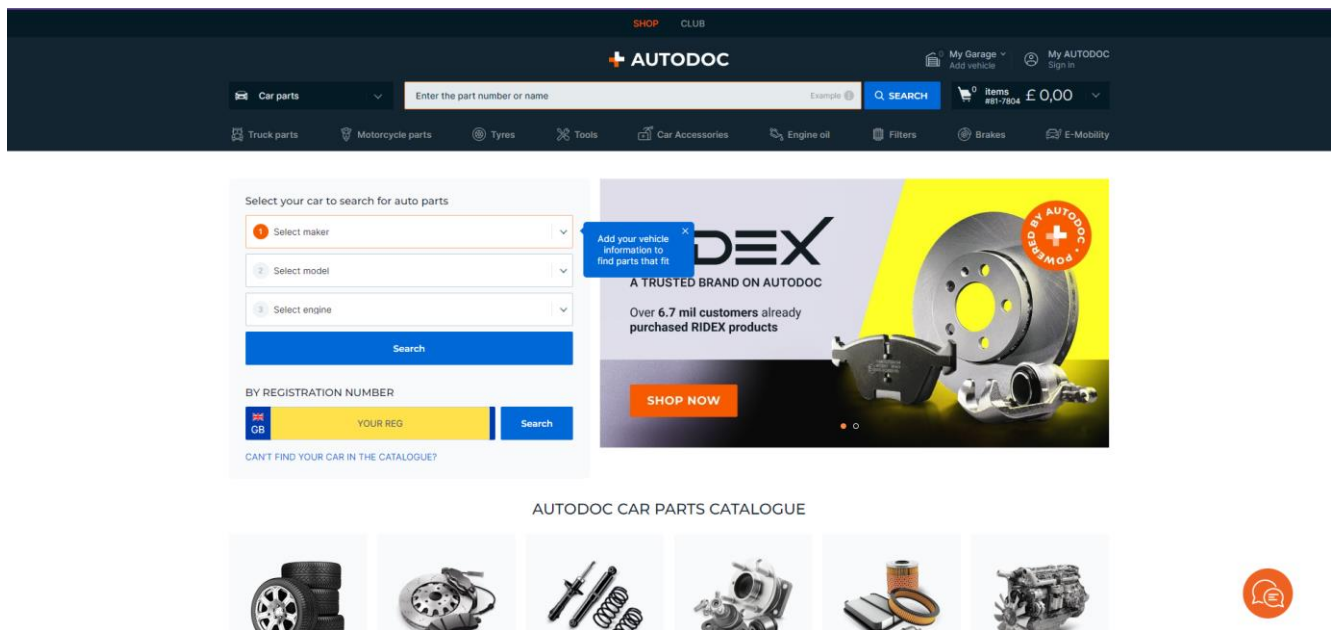


Рис. 1.1. AutoDoc

<https://www.onlinecarparts.co.uk/> – інтернет-магазин автозапчастин, який вирізняється великим вибором деталей, конкурентними цінами та зручними способами здійснення покупок. На сайті продаються запчастини для багатьох марок автомобілів. Onlinecarparts.co.uk пропонує безкоштовну доставку замовлень на суму понад 140 фунтів стерлінгів у межах Великобританії, за деякими винятками.

Переваги:

- Широкий вибір деталей;
- Конкурентні ціни;
- Зручний сайт;
- Безкоштовна доставка при замовленні на суму понад 140 фунтів стерлінгів.

Недоліки:

- Вартість доставки може стягуватися за замовлення на суму до 140 фунтів стерлінгів або за великогабаритні товари.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		12

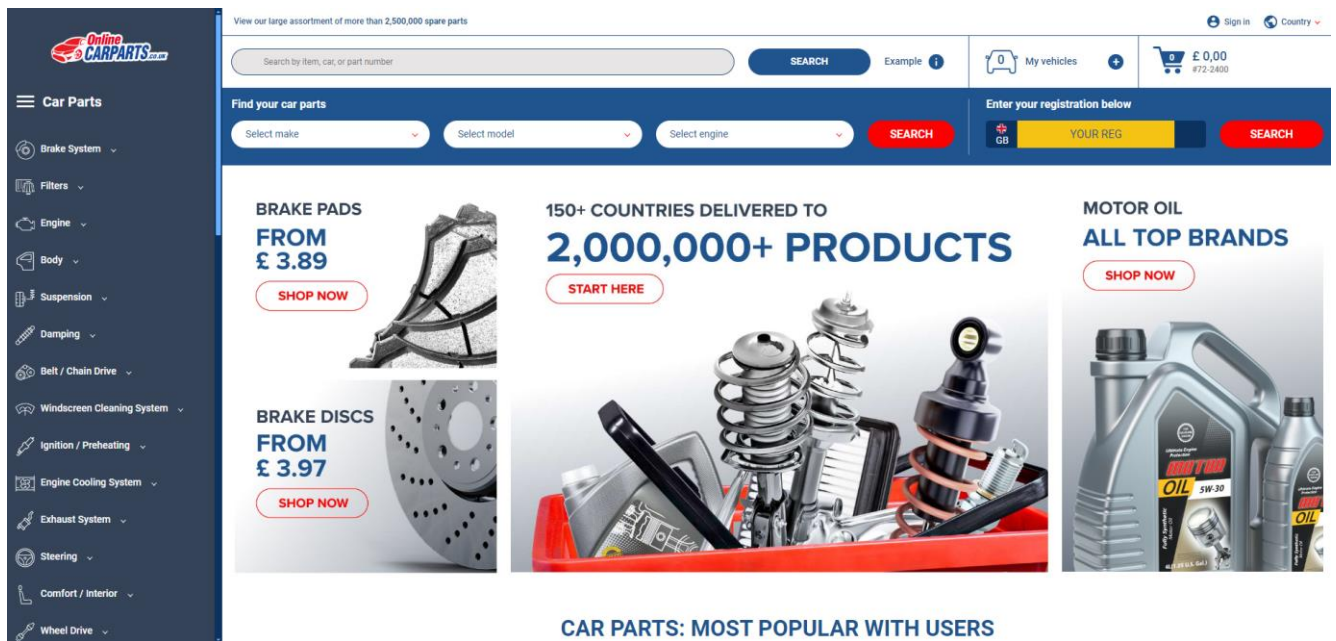


Рис. 1.2. onlinecarparts

<https://www.buycarparts.co.uk/> – це магазин автозапчастин. У ній йдеться про купівлю автозапчастин онлайн. Магазин пропонує широкий вибір запчастин, включаючи гальма, підвіску, фільтри та деталі двигуна. Вони також продають засоби для догляду за автомобілем. Buycarparts.co.uk гарантує високоякісні запчастини за доступними цінами. Вони пропонують швидку доставку більшості товарів.

Переваги:

- Широкий асортимент запчастин, включаючи гальма, підвіску, фільтри та деталі двигуна;
- Гарантія високої якості запчастин за доступними цінами;
- Швидка доставка більшості товарів;
- Зручний сайт, що дозволяє легко знайти потрібні запчастини;

Недоліки:

- Відгуки про Buy Car Parts;
- Порівняння цін на автозапчастини;
- Посібник з купівлі автозапчастин.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		13

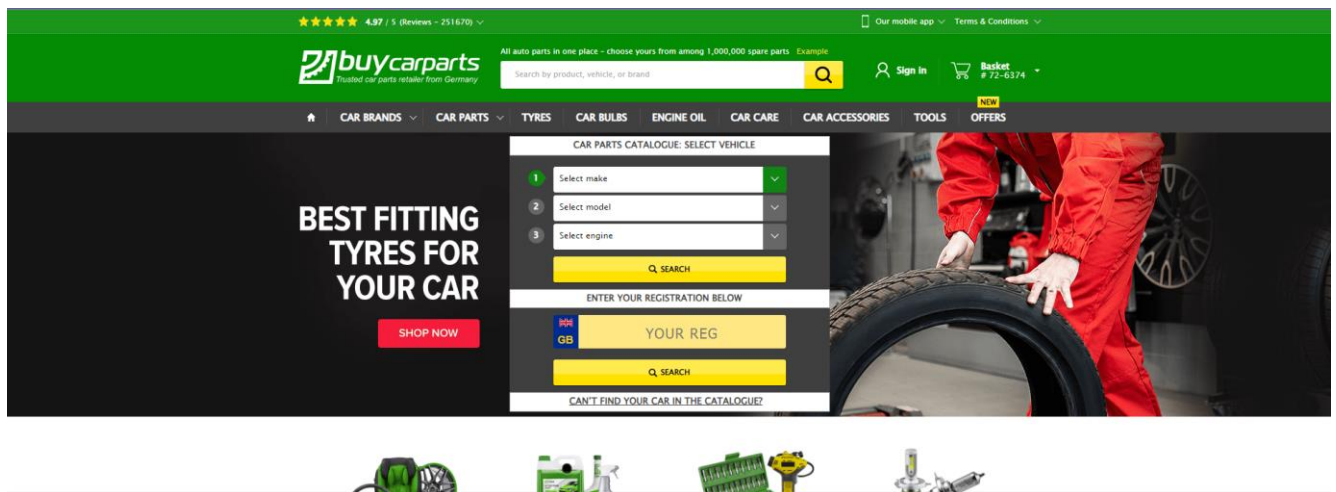


Рис. 1.3. buycarparts

Таблиця 1.1. Стек конкурентних додатків

Веб-сторінка	Розробка	Мова програмування	Кількість розробників
autodoc	Django CRM	Python	15-20
buycarparts	Wordpress	PHP	8-12
onlinecarparts	-	—	1

1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вибираючи мову розробки, слід звернути увагу на кількість літератури, документації та спільноти, які можуть допомогти вам вирішити проблеми, коли інші джерела не дають відповіді на проблему, яка є неочевидною.

Під час вибору стеку, вибір впав на мову програмування Python

1.3.1. Python:

Python-одна з найпопулярніших і універсальних мов програмування, що відрізняється високою продуктивністю і простотою розробки. Є багато причин вибрати Python як основну мову для розробки онлайн-платформи для продажу автозапчастин.

Python має простий та зрозумілий синтаксис, що дозволяє швидко розробляти і підтримувати код. Це важливо для забезпечення гнучкості та швидкості розробки, а також для можливості легкої адаптації та розширення платформи в майбутньому.

Python має велику кількість бібліотек і фреймворків, таких як Django, Flask і FastAPI, які дозволяють швидко й ефективно створювати веб-додатки.

Ці інфраструктури забезпечують високий рівень абстракції та включають готові рішення для багатьох завдань, таких як використання бази даних, маршрутизація запитів, керування сесіями користувачів і безпека.

Python надає багато бібліотек для роботи з різними типами баз даних, наприклад SQLAlchemy для реляційних баз даних (наприклад, PostgreSQL, MySQL) і PyMongo для баз даних документів (наприклад, MongoDB).

Це полегшує інтеграцію бази даних у платформу та забезпечує ефективне зберігання й обробку даних.

Python надає зручні інструменти для інтеграції з іншими системами та службами через API.

Це важливо для забезпечення підключення платформи до зовнішніх сервісів, таких як платіжні системи, служби доставки тощо.

Асинхронні можливості Python (особливо через бібліотеки `asyncio` та `aiohttp`) забезпечують ефективну обробку великої кількості одночасних запитів, що важливо для високонавантажених веб-платформ.

1.3.2. Django

Django є одним із найпопулярніших і найпотужніших веб-фреймворків для мови програмування Python. Він пропонує багато варіантів для створення складних і потужних веб-додатків, що робить його ідеальним вибором для розробки онлайн-платформи для продажу автозапчастин. Вибір Django виправданий декількома ключовими факторами, які визначають його перевагу в розробці сучасних веб-додатків.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Однією з головних переваг Django є його зосередженість на швидкому розвитку. Завдяки інтегрованим інструментам адміністрування та різноманітним готовим до використання рішенням розробники можуть значно прискорити процес створення веб-додатків. Django дозволяє автоматично створювати багато аспектів вашої веб-програми, такі як форми введення даних, маршрутизація URL-адрес і базові панелі адміністратора. Це особливо важливо на ранніх етапах розробки, коли важливо швидко створити робочий прототип платформи.

Django розроблено за принципом «батарейка включена» (все необхідне є в комплекті). Це означає, що він поставляється з багатьма вбудованими інструментами та бібліотеками, які забезпечують функціональність, необхідну для розробки веб-додатків. Ці інструменти включають потужний ORM (Object Relational Mapping) для роботи з базами даних, системами обробки моделей, механізмами автентифікації та авторизації користувачів, а також інструменти для створення API RESTful.

Завдяки цьому розробники можуть зосередитися на логіці програми, не витрачаючи час на дослідження та інтеграцію додаткових бібліотек.

Розробка веб-додатків, наприклад, тих, що обробляють персональні дані користувачів, а також тих, що здійснюють фінансові транзакції, вимагає безпеки; отже, важливо подумати про захист таких веб-додатків, які розробляються. Django поставляється з вбудованим високим рівнем безпеки, який захищає його від найпоширеніших вразливостей, таких як SQL ін'єкції, Cross Site Scripting (XSS), CSRF та інших. Крім того, Django регулярно оновлює свої версії, щоб задовольнити нові виявлені вразливості та відповідати сучасним тенденціям у сфері захисту даних.

Висока масштабованість та продуктивність роблять Django ідеальним інструментом для розробки платформ з високим трафіком. Асинхронні процеси та можливість інтеграції з різними системами кешування, зокрема Memcached та Redis, дозволяють Django ефективно обробляти численні одночасні запити. Це особливо важливо для онлайн-платформ, які повинні

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		16

мати можливість одночасно обслуговувати багато користувачів і замовлень з швидким часом відгуку.

Django пропонує високий ступінь гнучкості та розширюваності, що дозволяє компаніям модифікувати платформу відповідно до їхніх конкретних потреб. Розробники можуть легко додавати нові функції, модулі, змінювати існуючі або навіть підключати сторонні сервіси. В результаті платформу можна поступово розширювати, додаючи нові функції, зберігаючи при цьому існуючі функції максимально недоторканими, не витрачаючи час на розробку.

Django має велику і активну спільноту розробників, яка надає доступ до широкого спектру ресурсів, таких як форуми, блоги, навчальні посібники та бібліотеки. Це значно полегшує процес розробки, дозволяючи розробникам швидко знаходити відповіді на питання та отримувати допомогу у вирішенні технічних проблем. Крім того, Django має детальну і добре структуровану документацію, яка охоплює всі аспекти розробки, від початкового налаштування до розгортання на виробничому сервері.

Висновки до першого розділу

Інтернет-магазин автозапчастин повинен забезпечувати як зручність користування для клієнтів, так і ефективність адміністрування для системних адміністраторів. Це досягається шляхом реалізації як функціональних, так і нефункціональних вимог. Користувачі повинні мати змогу реєструватися, авторизуватися, редагувати профілі, переглядати каталог товарів, здійснювати пошук і фільтрацію, керувати кошиком замовлень, оформлювати замовлення, а також залишати відгуки та оцінки товарів. Адміністратори, своєю чергою, повинні мати доступ до управління користувачами, товарами, замовленнями та відгуками. Важливим аспектом є вибір технологічного стеку, який забезпечить масштабованість, безпеку та ефективність платформи. Аналіз аналогів, таких як Autodoc, Onlinecarparts та Buycarparts, показав важливість надання широкого асортименту, конкурентних цін та зру-

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		17

чного користувацького інтерфейсу. Вибір Python як основної мови програмування, а також фреймворку Django обґрунтований їхньою популярністю, продуктивністю, безпекою та можливістю швидкого розвитку. Django, з його вбудованими інструментами і високим рівнем безпеки, є ідеальним вибором для створення сучасного та надійного інтернет-магазину автозапчастин.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		18

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗЕБЕЗПЕЧЕННЯ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Під час проектування існує певний етап загального проектування, який включає важливі елементи, що сприяють розробці програмного продукту. На цьому етапі здійснюється детальне планування та опис проекту, що забезпечує його якісну та успішну реалізацію. У вступній частині цього розділу зазвичай містяться такі елементи:

Тут надається детальний опис вимог і функціональних можливостей програмного продукту. Воно включає інформацію про цільову аудиторію, функціональні вимоги, необхідні інтеграції та інші деталі, що допомагають правильно зрозуміти проект.

2.1.1. Технічне завдання

Сформовано технічне завдання для проекту. Це дасть розуміння в продовж розробки частини групового проекту.

1. Загальні положення

1.1. Назва проекту: Розробка онлайн-платформи для продажу автомобільних запчастин.

1.2. Мета проекту: Створення веб-платформи, що дозволяє користувачам купувати та продавати автомобільні запчастини онлайн, забезпечуючи зручний інтерфейс та високий рівень безпеки транзакцій.

1.3. Основні задачі:

- Реалізація функціоналу для реєстрації та авторизації користувачів;
- Розробка каталогу товарів з можливістю пошуку та фільтрації;
- Забезпечення кошика покупок та системи оформлення замовлень;
- Створення адміністративного інтерфейсу для управління товарами, замовленнями та користувачами;
- Інтеграція з платіжними системами та службами доставки;

2. Вимоги до функціональності

2.1. Реєстрація та авторизація користувачів:

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		19

- Користувачі повинні мати можливість реєструватися на платформі за допомогою електронної пошти та пароля;
- Повинна бути реалізована функція авторизації користувачів для доступу до особистого кабінету;

2.2. Профіль користувача:

- Користувач повинен мати можливість переглядати та редагувати особисту інформацію;
- Реалізація можливості перегляду історії замовлень;

2.3. Каталог товарів:

- Каталог повинен містити інформацію про всі доступні товари, включаючи фото, опис, ціни та наявність;
- Повинна бути реалізована функція пошуку товарів за ключовими словами;
- Потрібно передбачити можливість фільтрації товарів за категоріями, ціною, виробником та іншими параметрами;

2.4. Кошик покупок:

- Користувачі повинні мати можливість додавати товари до кошика, переглядати його вміст та видаляти товари;
- Повинна бути реалізована функція обчислення загальної вартості замовлення;

2.5. Оформлення замовлення:

- Користувачі повинні мати можливість оформляти замовлення, вибираючи спосіб оплати та адреси доставки;
- Повинна бути реалізована можливість вибору способу доставки та розрахунку вартості доставки;

2.6. Адміністративний інтерфейс:

- Адміністратори повинні мати можливість додавати, редагувати та видаляти товари;

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		20

- Потрібно забезпечити функціонал для управління користувачами та перегляду їх активності;
- Адміністратори повинні мати можливість переглядати та обробляти замовлення, змінювати їх статус;

2.7. Безпека:

- Платформа повинна забезпечувати захист від основних видів кіберзагроз, таких як SQL-ін'єкції, XSS, CSRF;
- Дані користувачів повинні бути захищені за допомогою шифрування та інших методів безпеки;

3. Вимоги до продуктивності

3.1. Швидкодія:

- Платформа повинна забезпечувати швидкий час відгуку для користувачів, навіть при високих навантаженнях;
- Сторінки повинні завантажуватися не довше ніж за 2 секунди при середньому навантаженні;

3.2. Масштабованість:

- Платформа повинна бути здатна обробляти великий обсяг запитів та користувачів одночасно;
- Потрібно забезпечити можливість горизонтального масштабування;

4. Вимоги до інтерфейсу

4.1. Зручність використання:

- Інтерфейс платформи повинен бути інтуїтивно зрозумілим та зручним для користувачів;
- Потрібно передбачити можливість доступу до платформи з різних пристроїв, включаючи мобільні телефони та планшети;

4.2. Дизайн:

- Дизайн платформи повинен бути сучасним та привабливим;

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		21

- Потрібно дотримуватись єдиного стилю для всіх сторінок платформи;

5. Вимоги до технічного середовища

5.1. Мова програмування та фреймворк:

- Основною мовою програмування обирається Python;
- Для розробки веб-додатку використовується фреймворк Django;

5.2. База даних:

- Використовується реляційна база даних SQLite;

5.3. Сервер та хостинг:

- Сервер повинен мати не менше 4 ядер процесора та 8 ГБ оперативної пам'яті;

Використовуються SSD-накопичувачі для зберігання даних.

5.4. Інструменти розробки:

- Використовується система контролю версій Git;
- PyCharm як IDE.

5.5. Середовище розгортання:

Використовується Windows Server.

6. План розробки

6.1. Етапи розробки:

- Збір вимог та аналіз;
- Проектування архітектури системи;
- Розробка користувацького та адміністративного інтерфейсу;
- Розгортання та запуск платформи.

В процесі проектування можуть використовуватися різні типи діаграм, такі як UML-діаграми (наприклад, діаграми варіантів використання, діаграми класів, діаграми послідовності) або інші візуальні засоби для моделювання різних аспектів системи.

У контексті нашого дипломного проекту, який присвячений розробці онлайн-платформи для продажу автозапчастин, діаграма варіантів використання

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		22

(Use Case Diagram) відіграє ключову роль у визначенні основних сценаріїв взаємодії користувачів з платформою. Ця діаграма допомагає деталізувати всі етапи взаємодії користувача з системою, починаючи від реєстрації та входу в систему і закінчуючи оформленням та оплатою замовлення.

Діаграма дозволяє чітко визначити, які саме функції будуть доступні користувачам платформи, та які дії вони можуть виконувати. Це забезпечує зрозумілу структуру для подальшої розробки та тестування системи.

Завдяки детальному опису всіх можливих взаємодій користувача з системою, розробники можуть оптимізувати інтерфейс користувача, зробивши його більш інтуїтивним та зручним.

Діаграма варіантів використання допомагає команді розробників визначити пріоритети та етапи реалізації функціоналу, що забезпечує ефективне планування проекту та управління ресурсами.

Завдяки візуалізації всіх взаємодій, команда може завчасно виявити та вирішити потенційні проблеми в логіці роботи системи або у взаємодії її компонентів.

Діаграма є важливим елементом документації, яка може бути використана для пояснення роботи системи зацікавленим сторонам, включаючи замовників, інвесторів та нових членів команди.

.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		23

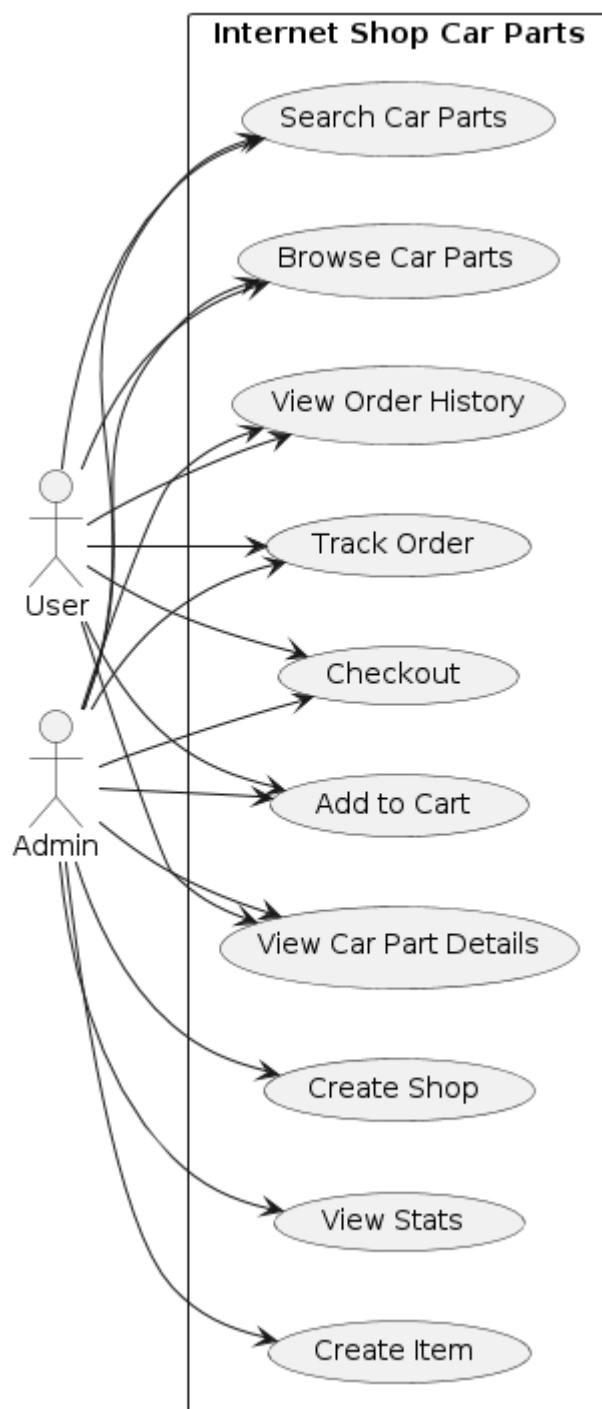


Рис. 2.1. Діаграма варіантів використання

2.2. Розробка бази даних системи

Наша база даних розроблена для підтримки функціонування високоефективної онлайн-платформи, яка спеціалізується на повному циклі продажу автомобільних запчастин. Вона включає комплекс ключових моделей, що забезпечують ретельне зберігання та оптимізовану обробку інформації про товари,

		Ю.О.КОЗАК		
		О.В.КУЗЬМЕНКО		
Змн.	Арк.	№ докум.	Підпис	Дата

профілі користувачів, замовлення, відгуки та динамічні зображення. Основні моделі бази даних забезпечують:

- Модель Category відповідає за зберігання категорій товарів, що продаються на платформі.
- Модель Users зберігає інформацію про користувачів платформи, таку як електронна адреса, ім'я користувача, пароль та адреса.
- Модель Shop представляє магазини, що продають автомобільні запчастини. Кожен магазин має назву, дату створення та власника.
- Модель Product зберігає інформацію про кожен товар на платформі, таку як назва, кількість на складі, опис, ціна, характеристики, категорія та магазин, що продає товар.
- Модель Orders зберігає дані про замовлення користувачів, включаючи інформацію про користувача, магазин, товар, кількість, статус замовлення, метод оплати та дату.
- Модель Reviews відображає відгуки користувачів про кожен товар. Кожен відгук має текст, оцінку товару, а також посилання на користувача, товар та замовлення.
- Модель Images забезпечує зберігання зображень для кожного товару на платформі. Кожне зображення пов'язане з певним товаром.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		25

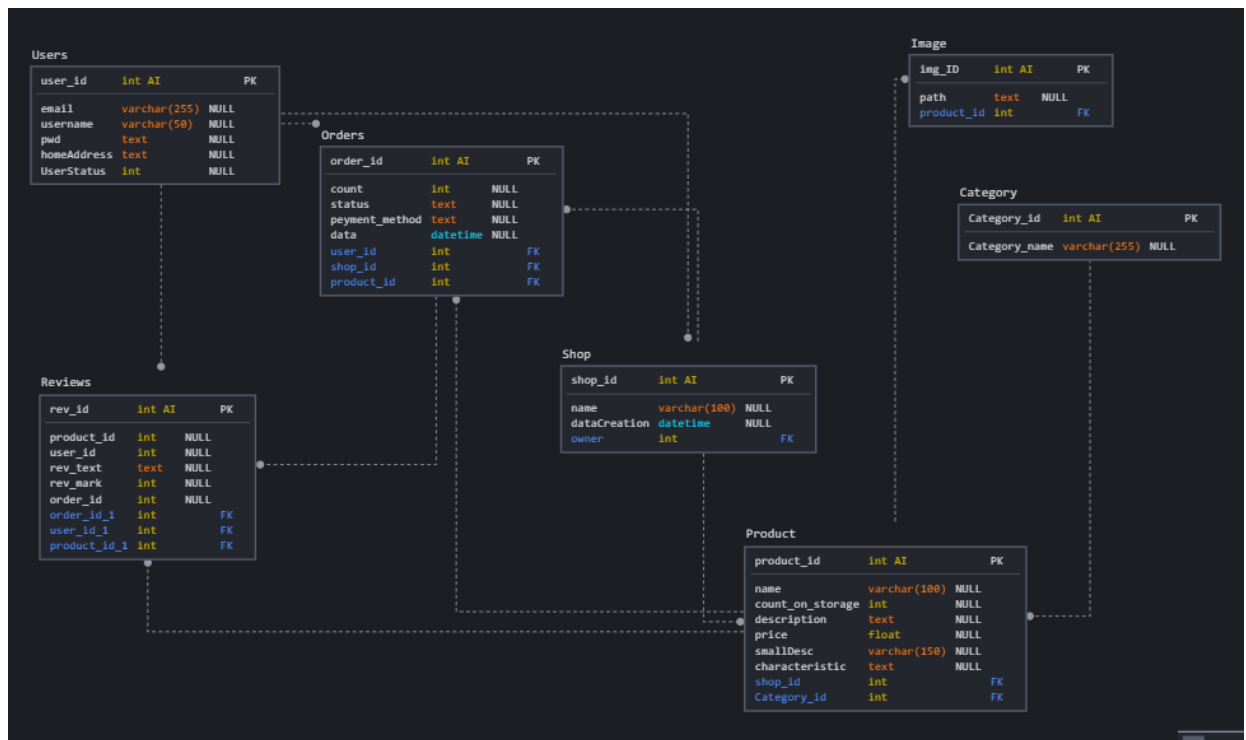


Рис 2.2. логічна ERD діаграма

2.3. Проектування та реалізація алгоритмів роботи системи

Побудова ефективних алгоритмів для роботи системи є критично важливим етапом у розробці будь-якої програмної системи. Ці алгоритми визначають логіку та поведінку системи, вирішують завдання, обробляють дані та

Діаграма активностей (Activity Diagram) в рамках дипломної роботи грає важливу роль у візуалізації та аналізі процесів, що відбуваються у системі. Вона дозволяє детально уявити послідовність дій, які здійснюють користувачі або системні компоненти, включаючи умови, рішення та взаємодію між ними.

Переваги використання діаграми активностей полягають у її здатності чітко відображати логіку процесів і сприяти розумінню роботи системи. Це особливо важливо в контексті розробки онлайн-платформи для продажу автозапчастин, де ретельне моделювання кожного кроку взаємодії з користувачем є ключовим аспектом успішної реалізації проекту.

Аналізуючи діаграму активностей, можна ідентифікувати можливі логічні або послідовні помилки, які можуть виникнути під час взаємодії з системою. Це дозволяє уникнути проблем під час реалізації та тестування програмного забезпечення.

В процесі створення діаграми активностей можна уточнити та відпрацювати вимоги до програмного забезпечення замовника, що допомагає покращити розуміння і взаємодію між всіма учасниками проекту.

Вона допомагає точно визначити послідовність операцій, що потрібно виконати для реалізації кожної функції платформи, включаючи авторизацію, роботу з каталогом товарів, оформлення замовлень та інші важливі дії.

Ця діаграма показує послідовність дій, які можуть бути виконані користувачем платформи в залежності від того, чи він адміністратор чи звичайний користувач:

- Login (Вхід): Початок процесу. Користувач або адміністратор входять до системи.
- Admin Menu (Меню адміністратора): Якщо користувач - адміністратор, він має доступ до спеціального меню адміністратора.
- Select Action (Вибір дії): Адміністратор обирає дію, яку потрібно виконати (наприклад, керування товарами або користувачами).
- Perform Action (Виконання дії): Виконання обраної адміністратором дії.
- User Menu (Меню користувача): Якщо користувач не є адміністратором, він бачить меню для користувача.
- Browse Car Parts (Перегляд запчастин): Користувач переглядає доступні на платформі автомобільні запчастини.
- Select Car Part (Вибір запчастини): Вибір конкретної запчастини для подальших дій.
- View Details (Перегляд деталей): Перегляд детальної інформації про обрану запчастину.
- Add to Cart (Додавання до кошика): Додавання обраної запчастини до кошика покупок.
- Checkout (Оформлення замовлення): Перехід до процесу оформлення замовлення.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		27

- Payment (Оплата): Введення даних для оплати обраних запчастин.
- Order Confirmation (Підтвердження замовлення): Підтвердження успішної операції оформлення замовлення.

Ця діаграма демонструє типовий процес взаємодії користувача з платформи продажу автомобільних запчастин, починаючи від входу до системи і закінчуючи оформленням та підтвердженням замовлення.

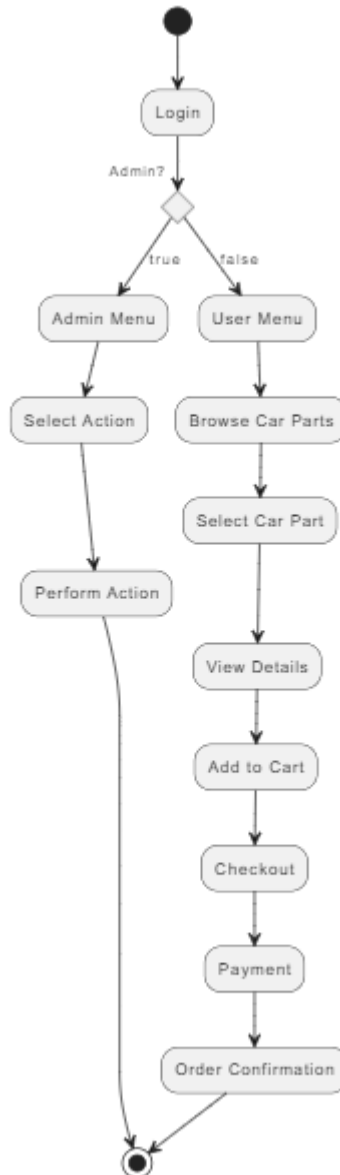


Рис. 2.3. Діаграма активностей

Діаграма класів є структурним типом діаграми в області моделювання програмного забезпечення. Вона використовується для візуалізації структури системи шляхом представлення класів, їх атрибутів, методів та взаємозв'язків між ними.

Наша діаграма класів представляє моделі даних для веб-додатку, розробленого у функціональному стилі на фреймворку Django. Django використовує підходи моделювання, що сприяють створенню структурованих та ефективних баз даних. Отже, кожна модель Django представляється як клас, який відображено на діаграмі.

Опис елементів діаграми класів:

- Класи моделей: Кожен клас (наприклад, Category, Users, Shop, Product, Orders, Reviews, Images) відображає відповідну модель Django, яка визначає структуру та взаємозв'язки між об'єктами даних.
- Атрибути класів: Кожен клас має свої атрибути (наприклад, ідентифікатори, назви, тексти описів, ціни тощо), які відображають поля бази даних, визначені у відповідній моделі Django.
- Методи класів: Деякі класи мають методи, які використовуються для представлення об'єктів у текстовому форматі (`__str__()` методи), що спрощує їх виведення та використання в програмі.
- Взаємозв'язки: Взаємозв'язки між класами (наприклад, асоціації, залежності, агрегації) показують, як об'єкти взаємодіють між собою в контексті системи.

Розробка на функціональному стилі Django підкреслює використання моделей як основного елемента для взаємодії з базою даних, забезпечуючи зручність та ефективність в роботі з даними.

Діаграма класів, розроблена для веб-додатку на фреймворку Django, ілюструє структуру та організацію моделей даних. Вона допомагає команді розробників та зацікавленим сторонам краще розуміти взаємозв'язки між об'єктами та сприяє ефективній розробці та підтримці системи.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		29

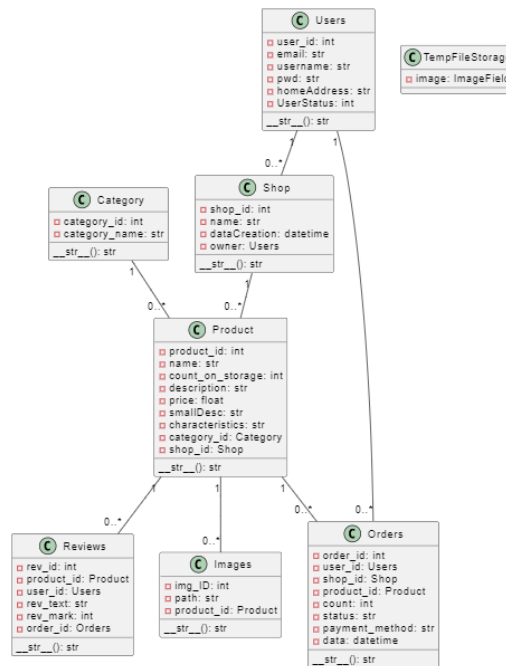


Рис 2.4. Загальна діаграма класів

Діаграма послідовностей у рамках дипломної роботи відіграє ключову роль у візуалізації взаємодії між об'єктами системи під час виконання конкретного сценарію або функціональної операції. Цей інструмент дозволяє ілюструвати порядок передачі повідомлень та залежності між об'єктами, що допомагає розробникам краще розуміти логіку програми.

Основна мета діаграми послідовностей полягає в тому, щоб відобразити, як об'єкти взаємодіють один з одним під час виконання заданого процесу. Це дозволяє виявляти помилки та проблеми у взаємодії між об'єктами, а також знаходити можливості для оптимізації та вдосконалення продукту.

У контексті дипломної роботи діаграма послідовностей особливо корисна при проектуванні або розробці систем з великою кількістю взаємодіючих компонентів або при виконанні складних операцій, які вимагають узгодженого виконання кількох об'єктів. Вона допомагає виявити послідовність кроків у виконанні функціональних вимог, узагальнити поведінку системи та спростити розуміння взаємодії між компонентами.

Крім того, діаграма послідовностей може виступати як ефективний інструмент для комунікації між учасниками команди розробки, якщо всі сторони

розуміють обрану нотацію. Вона дозволяє зображати послідовність подій у зрозумілій формі, що спрощує спілкування та спільне розуміння між учасниками проекту, підвищуючи ефективність роботи команди.

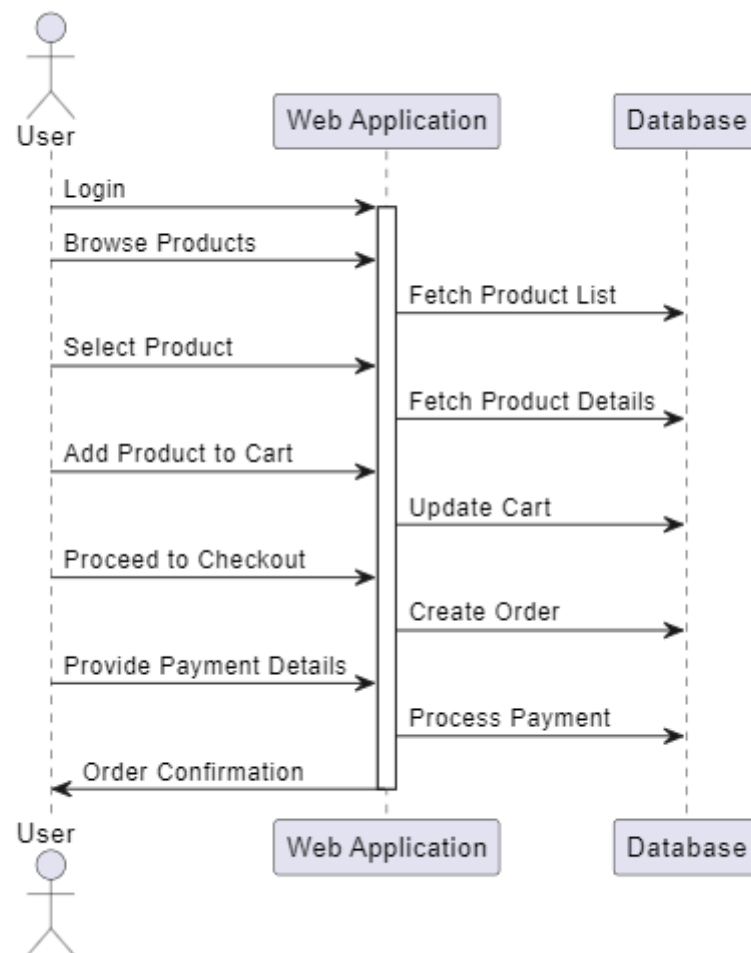


Рис. 2.5. Діаграма послідовності

Діаграма компонентів в рамках дипломної роботи є невід'ємною частиною проектування і візуалізації архітектури веб-додатку. Цей інструмент дозволяє детально представити структуру додатку, включаючи компоненти системи, їх взаємозв'язки та залежності.

Основна мета діаграми компонентів полягає в тому, щоб показати, які конкретні частини додатку складаються з окремих компонентів, як вони взаємодіють між собою та які функції вони виконують. Це сприяє кращому розу-

мінню структури проекту і дозволяє виявляти можливі проблеми чи неузгодженості в архітектурі, що може значно поліпшити організацію коду та управління проектом.

Діаграма компонентів також допомагає зрозуміти, які саме компоненти відповідають за реалізацію певних функцій додатку, що сприяє кращій організації коду та підвищує загальне розуміння структури проекту. Вона включає такі ключові елементи, як рушій додатку, системи управління, аудіо- та графічні двигуни, інтерфейси користувача та інші важливі модулі, що спрощує комунікацію і спільне розуміння між усіма учасниками проекту.

Вона дозволяє однозначно уявити, які саме компоненти використовуються, як вони взаємодіють між собою та які зміни можна вносити для покращення системи, що сприяє ефективній спільній роботі і направляє розвиток проекту в потрібному напрямку.

Орієнтовність діаграми компонентів також дозволяє з легкістю внести необхідні зміни в систему під час її розвитку, що підтримує гнучкість і масштабованість проекту. Цей підхід дозволяє ефективно впроваджувати нові функціональності та зміни без порушення загальної архітектури додатку.

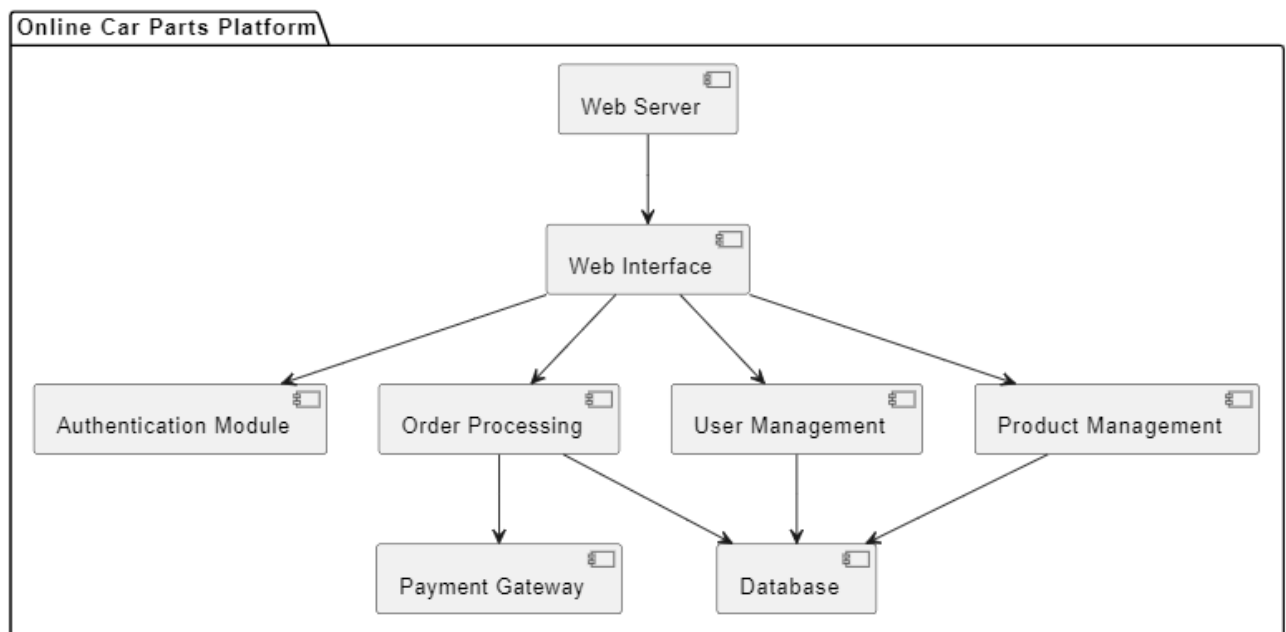


Рис. 2.6. Діаграма компонентів

Діаграма розгортання яка використовується для візуалізації фізичного розташування компонентів системи та залежностей між ними на різних вузлах апаратного забезпечення. У контексті додатку, діаграма розгортання допомагає представити, як додаток та його компоненти розгортані на фізичних серверах, комп'ютерах або пристроях.

Основна мета діаграми розгортання у дипломному проєкті полягає в тому, щоб показати, як фізичні ресурси (сервери, комп'ютери, мобільні пристрої тощо) використовуються для розгортання компонентів додатку. Діаграма дозволяє визначити, на яких вузлах або пристроях розташовані різні компоненти додатку, як вони спілкуються між собою та які залежності існують між ними.

У додатку діаграма розгортання може включати фізичні сервери, хмарні платформи, клієнтські комп'ютери, мобільні пристрої або інші пристрої, на яких додаток розгортається. Воно відображає, які компоненти додатку знаходяться на кожному вузлі або пристрої, як вони взаємодіють між собою та які мережеві зв'язки використовуються для обміну даними.

Діаграма розгортання є типом структурної діаграми, яка використовується для візуалізації фізичного розташування компонентів системи та залежностей між ними на різних вузлах апаратного забезпечення. У контексті додатку, діаграма розгортання допомагає представити, як додаток та його компоненти розгортані на фізичних серверах, комп'ютерах або пристроях.

Основна мета діаграми розгортання у додатку полягає в тому, щоб показати, як фізичні ресурси (сервери, комп'ютери, мобільні пристрої тощо) використовуються для розгортання компонентів додатку. Діаграма дозволяє визначити, на яких вузлах або пристроях розташовані різні компоненти додатку, як вони спілкуються між собою та які залежності існують між ними.

Даний додаток використовує діаграму розгортання, що допомагає зрозуміти що може включати фізичні сервери, хмарні платформи, клієнтські комп'ютери, мобільні пристрої або інші пристрої, на яких додаток відтворюється. Вона відображає, які компоненти додатку знаходяться на кожному вузлі або

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		33

пристрої, як вони взаємодіють між собою та які мережеві зв'язки використовуються для обміну даними.

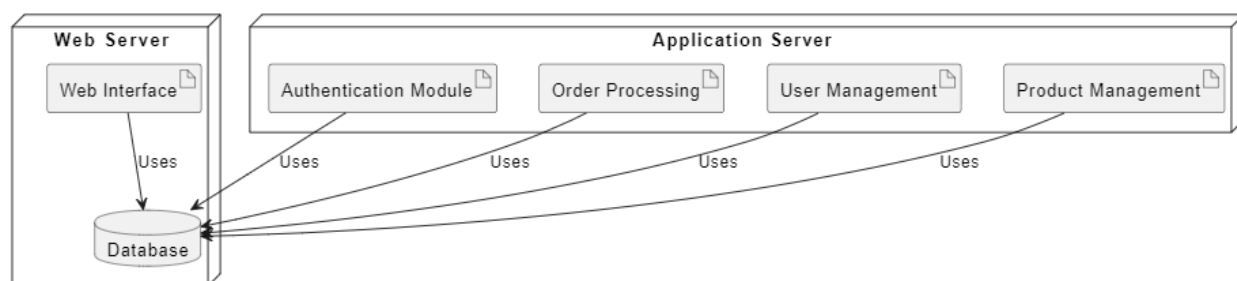


Рис. 2.7. Діаграма розгортання

2.4. Реалізація додатку

Розробка проекту Django починається зі створення віртуального середовища для ізоляції пакетів. Після цього встановлюється Django, що надає необхідні інструменти для розробки веб-додатків. Далі створюється основний проект Django за допомогою команди `django-admin startproject project_name`, що ініціалізує каталоги та основні файли проекту.

Ключові компоненти проекту включають `settings.py`, де зберігаються налаштування проекту, `urls.py` для визначення маршрутизації URL-адрес та каталоги `apps/`, `templates/` і `static/`, де розміщуються код додатків, HTML-шаблони та статичні файли відповідно.

Додатки розробляються окремо за допомогою команди `python manage.py startapp app_name` і містять власні моделі, представлення та шаблони. Після розробки компонентів проводиться їх тестування для перевірки коректності роботи. Готовий проект може бути розгорнутий на сервері після налаштування веб-сервера та інших необхідних параметрів Django.

Створення суперадміністратора в Django є важливим кроком для управління додатком через інтегровану адміністративну панель. Для цього використовується команда `python manage.py createsuperuser`. Після виконання цієї команди система запропонує ввести ім'я користувача, електронну пошту та пароль. Після успішного створення облікового запису суперадміністратора, цей

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		34

користувач отримує доступ до адміністративної панелі, де може управляти усіма аспектами додатку, такими як користувачі, групи, моделі даних та інше.

У Django файли `views.py`, `models.py` та `urls.py` взаємодіють між собою для створення функціональності веб-додатку. Моделі, які описуються у файлі `models.py`, визначають структуру даних додатку. Вони визначають, які поля та типи даних будуть використовуватися, а також встановлюють зв'язки між різними моделями. Наприклад, модель для зберігання інформації про продукти може містити поля для назви, опису, ціни та кількості на складі.

Файл `views.py` містить функції або класи, які відповідають за обробку запитів від користувачів та формування відповідей. Представлення у `views.py` отримують дані з моделей, обробляють їх та передають у шаблони для відображення. Наприклад, представлення може отримати список всіх продуктів з бази даних, обробити ці дані та відобразити їх на веб-сторінці.

Файл `urls.py` визначає маршрутизацію URL-адрес у додатку. Він містить набір шляхів, кожен з яких вказує на відповідне представлення у `views.py`. Коли користувач переходить за певною URL-адресою, Django використовує маршрути, визначені у `urls.py`, щоб знайти відповідне представлення та обробити запит. Наприклад, URL-адреса `/products/` може відповідати представленню, яке відображає список продуктів.

Таким чином, у процесі розробки веб-додатку на Django, файли `models.py`, `views.py` та `urls.py` взаємодіють між собою. Моделі визначають структуру даних, представлення обробляють ці дані та формують відповіді, а маршрутизація забезпечує коректний напрямок запитів до відповідних представлень. Ця взаємодія дозволяє створювати функціональні та структуровані веб-додатки з чітко розподіленими ролями між компонентами системи.

У Django посилання визначаються у файлі `urls.py`, де вони відповідають певним шляхам URL-адрес і зв'язуються з відповідними функціями у вигляді подань (`views`). На початку імпортується функція `path` з модуля `django.urls`, а також файл `views` з поточного каталогу. Далі встановлюється ім'я Django-

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		35

додатку за допомогою змінної `app_name`, що дозволяє уникнути конфліктів імен між різними додатками проекту. Приклад коду з файлу `cart/urls.py`:

```
from django.urls import path
from . import views

app_name = 'cart'

urlpatterns = [
    path('add/<int:product_id>/', views.cart_add, name='cart_add'),
]
```

У списку `urlpatterns` визначаються конкретні URL-шляхи. В даному випадку шлях `'add/int:product_id/'` відповідає функції `cart_add` у модулі `views`. Вираз `<int:product_id>` вказує, що частина URL, яка з'явиться після `'add/'`, буде цілим числом, яке передається як аргумент у функцію `cart_add`. Ім'я цього шляху `'cart_add'` може використовуватися для побудови URL-адрес або у HTML-шаблонах для створення посилань на цей шлях. Таким чином, коли користувач переходить за цим URL, викликається відповідна функція `cart_add` з переданим параметром `product_id`, що дозволяє виконувати відповідну дію, пов'язану з додаванням продукту до кошика.

Коли сервер отримує запит, він спочатку аналізує URL-адресу, зазначену в запиті. Django порівнює цей URL з шаблонами, визначеними у файлі `urls.py` кожного встановленого додатка. Пошук відбувається послідовно, починаючи з першого шаблону і продовжуючи до тих пір, поки не знайдеться відповідний шаблон.

Для кожного посилання в системі створюється обробник, обробник розміщений у файлі `views`, у кожному додатку в проекті на Django має свій файл `views`, приклад коду з файлу `main/views.py`:

```
def usercabinet(request):
    k = User(request).user
    if k:
        try:
            user = get_object_or_404(Users, username=k['username'])
            shop = Shop.objects.select_related().filter(owner=user)
            orders = Orders.objects.select_related().filter(user_id=user)
        except:
            shop = None
        kwarg = {
            "title": "User Cabinet",
            "category": Category.objects.all(),
            "user": k,
            "orders": orders,
```

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		36

```

"shops": shop
}
return render(request, 'main/cabinet.html', context=kwargs)
else:
return HttpResponseRedirect('<h1>Page not found, go back to <a href="/">main</a></h1>')

```

В основному обробники використовуються тільки для посилань з цього джанго-додатку, проте ніхто не забороняє використовувати й з іншого.

Обробник запиту має один обов'язковий параметр, request, який містить у собі інформацію запит на основі функції зазвичай включає функцію подання, яке обробляє HTTP-запит і повертає HTTP-відповідь. Ці подання призначені для обробки конкретних веб-запитів і створення відповідних відповідей, які можуть включати HTML-сторінки, JSON-дані, перенаправлення або повідомлення про помилки. Згідно від типу запиту (наприклад GET або POST), може містити в собі й інші дані, наприклад дані форми, які передаються у словнику request.POST. Окрім цього request, приймає усі HTTP заголовки які клієнт відправляє з клієнту, а при рендері чи відповіді нові заголовки записуються у request та повертаються клієнту.

За потреби можна в функцію-подання (view function) додавати й інші параметри, наприклад id необхідного товару, при GET-запиті, тощо.

При роботі з запитами часто потрібно діставати чи записувати дані з бази даних, проте спочатку потрібно створити модель бази даних.

Моделі в Django є основою для зберігання даних. Вони дозволяють визначити структуру бази даних, а також надають інструменти для роботи з цими даними. Моделі визначають поля і поведінку збережених даних. Приклад з main/models.py

```

class Reviews(models.Model):
    rev_id = models.AutoField(verbose_name='id', auto_created=True, primary_key=True)
    product_id = models.ForeignKey(Product, on_delete=models.CASCADE)
    user_id = models.ForeignKey(Users, on_delete=models.CASCADE)
    rev_text = models.TextField('RevText')
    rev_mark = models.IntegerField("rev_mark", null=False, blank=False, default=1)
    order_id = models.ForeignKey(Orders, on_delete=models.CASCADE, default=None)

    def __str__(self):
        return self.rev_text

```

Кожне поле в моделі є екземпляром класу, який визначає тип даних цього поля.

Після створення або зміни моделі, необхідно створити та застосувати міграції. Міграції - це спосіб зберігання змін у структурі бази даних, які потім можна застосувати до самої бази даних.

Таким чином можна впливати на дані, не використовуючі SQL коду.

Приклад з файлу main/views.py

```
user = get_object_or_404(Users, username=k['username'])
shop = Shop.objects.select_related().filter(owner=user)
orders = Orders.objects.select_related().filter(user_id=user)
```

Таким чином можна діставати об'єкти, дістававти та робити вибірку об'єктів, при цьому ми будемо отримувати дані у форматі QuerySet або ж Object, взаємності від способу маніпуляції даних.

Далі потрібно показати користувачу дані та запропонувати йому переглянути і купити товари на магазині, для цього використовується зв'язка HTML+JS+CSS. Але кожен сторінку на таких веб-сайтах ніхто не буде створювати, для цього буде використовуватись один з наявних шаблонізаторів, основним у Python це Jinja2. Приклад з файлу `main/templates/main/index.html`:

```
{% extends 'base.html' %}
{% block title %}{{ title }}{% endblock %}
{% block content %}
    {% load static %}
    <script src="{% static 'jQuery.js' %}"></script>

    {% if sort == True %}
        <div class="d-flex flex-wrap justify-content-around">
            <div>
                {% if search == False %}
                    <form action="/" method="post">
                {% else %}
                    <form action="/search" method="post">
                {% endif %}
                {% csrf_token %}
                {% if search == True %}
                    <input type="search" value="{{ searchinfo }}" name="search" style="display: none">
                {% endif %}
                <table class="table">
                    <tbody>
                        <td>Sort by price</td>
                        <td><select name="sort-type-price">
                            <option value="price-ascendant">Ascendant</option>
                            <option value="price-descendent">descendent</option>
                        </select></td>
                        {% if shop_inc_sort == True %}
                            <td>and by shop<i>(optional)</i></td>
                            <td><select name="sort-type-shop">
                                <option value="No"></option>
                                {% for i in shops %}
                                    <option value="{{ i.shop_id }}">{{ i.name }}</option>
                                {% endfor %}
                            </select></td>
                        {% else %}
                            <td></td>
                        {% endif %}
                    </tbody>
                </table>
            </div>
        </div>
    {% else %}
        <div class="d-flex flex-wrap justify-content-around">
            <div>
                <div class="text-center">
                    <h2>No results found</h2>
                </div>
            </div>
        </div>
    {% endif %}
{% endblock %}
```

```

        {% endfor %}
    </select>
</td>
{% endif %}
<td><input type="submit" value="submit" class="btn btn-warning"></td>
</tbody>
</table>
</form>
</div>
{% if search != True %}
<div>
    <form action="/" method="post">
        {% csrf_token %}
        <table class="table">
            <tbody>
                <tr>
                    <td>Sort by category</td>
                    <td><select name="sort-type-category">
                        {% for i in category %}
                            <option value="{{ i.category_id }}">{{ i.category_name }}</option>
                        {% endfor %}
                    </select>
                </td>
                <td>
                    <input type="submit" value="submit" class="btn btn-warning"></td>
                </td>
            </tr>
        </tbody>
    </table>
</form>
</div>
{% endif %}
</div>
{% endif %}
<div class="d-flex flex-wrap justify-content-around">
    {% for i in product %}
        <div class="m-3">
            <a class="col mb-4 text-center overflow-hidden textDec" href="/product/{{ i.product_id }}/">
                <div class="card overflow-hidden">
                    <div class="hider mt-2">
                        {% for k in images %}
                            {% if k.product_id == i %}
                                
                            {% endif %}
                        {% endfor %}
                    </div>
                    <div class="card-body">
                        <h5 class="card-title">{{ i.name }}</h5>
                        <p class="card-text"><strong class="text-muted">${{ i.price }}</strong></p>
                        <p class="card-text text-truncate">{{ i.smallDesc }}</p>
                    </div>
                </div>
            </a>
        </div>
    {% endfor %}
</div>
<script>
    let x = document.getElementsByClassName("hider");
    for (let i = 0; i < x.length; i++) {
        while (x[i].childNodes.length > 2) {
            x[i].removeChild(x[i].lastChild);
        }
    }
</script>

```

```

    }
  }
</script>
{% endblock %}

```

Таким чином можна побачити як в проєкті використовується HTML код, разом з кодом Jinja2 та JavaScript-ом.

В Jinja2 є декілька маркерів:

- `{% %}` означають застосування функції, умовної конструкції, циклу та будь-чого що не імпортує в HTML код дані напряму, при цьому у всіх варіантах окрім функцій цей маркер має закриваючу пару (наприклад `{% if shop_inc_sort == True %}{% endif %}`);
- `{{ }}` означають імпорт значень в HTML код, тобто частіш за все дані з бази даних

Звичайно Jinja сама пособі має багато додаткових маркерів наприклад `|safe`, якій застовується під час встановлення даних зі змінної, проте у даній роботі вони виокристовувались дуже рідко.

Висновки до другого розділу

У другому розділі нашої роботи було детально розглянуто процес розробки платформи продажу автомобільних запчастин на базі Django. Ми дослідили, як створюється віртуальне середовище для ізоляції залежностей проєкту та встановлення Django, що забезпечує необхідний набір інструментів для розробки. Було також розглянуто, як ініціалізується проєкт та створюються основні компоненти, такі як налаштування, маршрутизація, шаблони та статичні файли.

Особливу увагу було приділено опису взаємодії між моделями, представленнями та URL-адресами. Моделі визначають структуру даних і зв'язки між ними, представлення обробляють запити користувачів, отримують дані з моделей і передають їх у шаблони для відображення, а маршрутизація спрямовує запити на відповідні представлення. Ця чітко структурована взаємодія між

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		40

компонентами забезпечує зручну та ефективну організацію коду, що полегшує підтримку і розвиток додатку.

Було також розглянуто процес створення суперадміністратора, який відіграє ключову роль у керуванні додатком через адміністративну панель. Цей інструмент дозволяє легко управляти користувачами, групами та іншими моделями даних, що значно спрощує адміністрування та підтримку платформи.

Загалом, другий розділ надав всебічний огляд технологічних та організаційних аспектів розробки веб-додатку на Django. Отримані знання і методи дозволяють ефективно планувати та реалізовувати проекти, використовуючи найкращі практики розробки, що забезпечують високу якість кінцевого продукту.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		41

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ДОДАТКОМ

3.1. Структура інтерфейсу. Інтерфейс та порядок роботи з системою

Після переходу на сайт користувач опиняється на головній сторінці, де йому пропонується ознайомитися з актуальним каталогом товарів, що були додані різними продавцями за останній період часу. Цей каталог є відображенням останніх оновлень і нових пропозицій, доступних на сайті. Відвідувач може вільно переглядати різноманітні товари, використовуючи зручні фільтри та сортування, щоб знайти саме те, що йому потрібно. Такий підхід дозволяє користувачам швидко знаходити й вибирати оптимальні варіанти покупок з великого асортименту товарів на сайті. (рис 3.1).

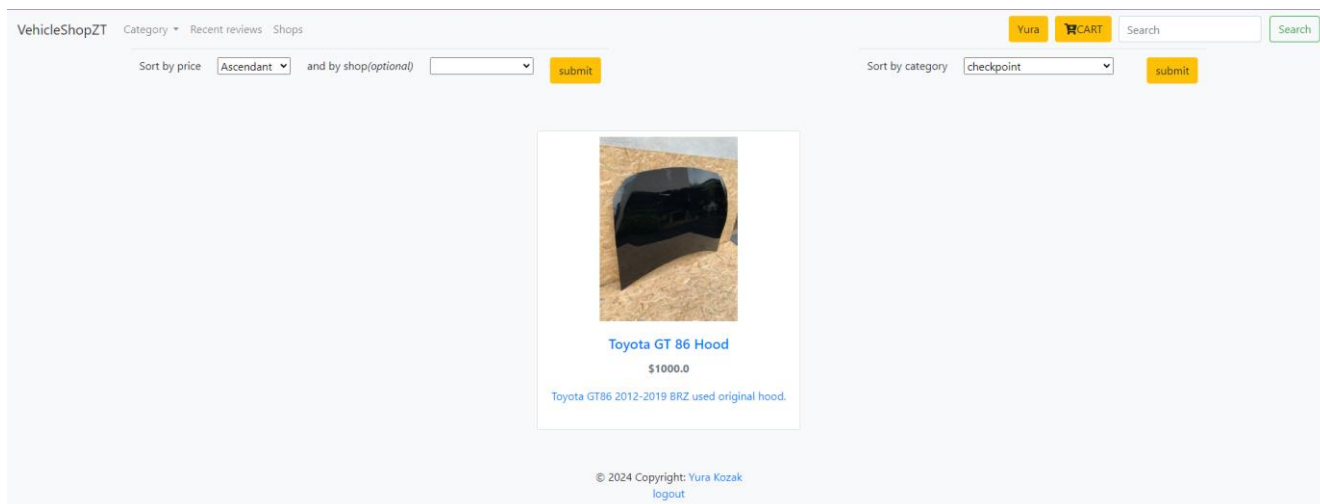


Рис 3.1. Головна сторінка

На головній сторінці сайту також присутні зручні опції сортування товарів за різними критеріями, можливість швидкого пошуку за частковим співпадінням і зручний перегляд списку категорій.

Користувач, перейшовши на сторінку конкретного товару, може детально ознайомитися з усією інформацією про нього: від визначення кількості необхідних одиниць товару до ознайомлення з його повним описом. Крім того, на цій сторінці користувач має можливість переглянути вже залишені відгуки інших покупців або залишити власний відгук, якщо він раніше придбав цей товар (рис. 3.2).

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		42

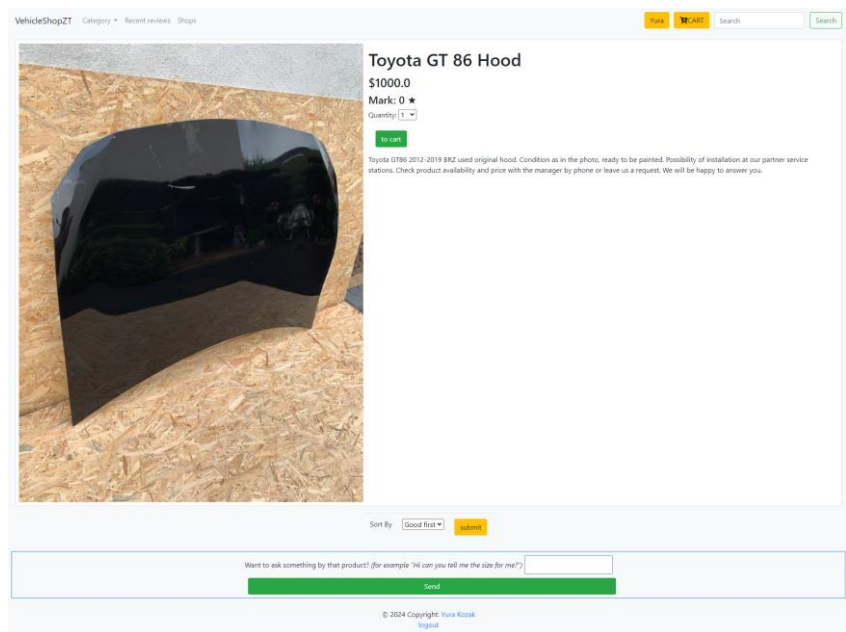


Рис 3.2. Сторінка товару

Також на сайті присутній важливий функціонал авторизації, який необхідний користувачам для здійснення покупок, залишення відгуків, створення власних магазинів і розміщення товарів у продаж. Для цього існує спеціальний модуль авторизації та реєстрації, який забезпечує безпеку і зручність взаємодії з сайтом для кожного користувача. (рис 3.3-4).

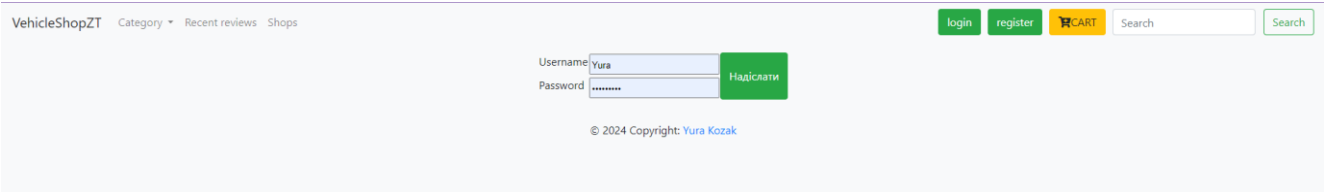


Рис 3.3. Авторизація

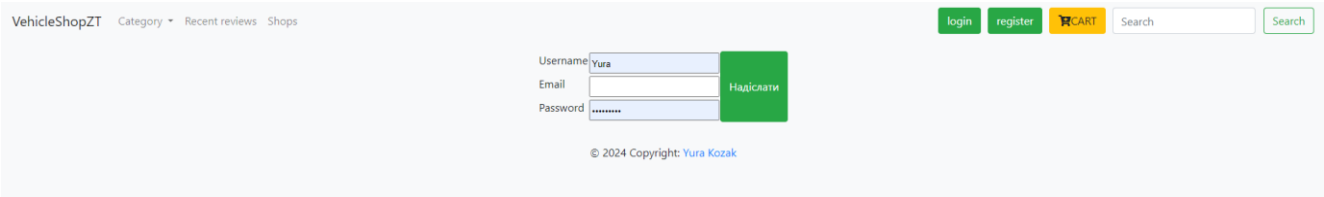


Рис 3.4. Реєстрація

У кожного користувача є свій особистий кабінет на сайті, де вони можуть зручно управляти своїми особистими даними, зокрема переглядати і змінювати реєстраційні дані за необхідністю. Крім того, в кабінеті є можливість залишати відгуки на придбані товари, що дозволяє іншим користувачам оці-

нювати якість продукції. Також у кабінеті доступна функція перегляду товарів, щоб зручно відстежувати їхню наявність та характеристики. Крім того, користувачі мають можливість перевіряти статус своїх замовлень, щоб бути завжди в курсі з їхнім виконанням. Не остання можливість кабінету полягає у створенні власного магазину, де користувачі можуть виставляти свої товари на продаж, а також переглядати статистику продажів та керувати адміністративними функціями магазину. (Рис. 3.5.)

Рис 3.5. Особистий кабінет користувача

На сторінці статистики власник магазину зможе здійснити глибокий аналіз свого бізнесу. Він матиме можливість докладно оцінити ефективність своїх товарів, включаючи товари з найкращими або найгіршими відгуками, а також найбільш популярні або непопулярні товари на основі показників продажів і рейтингів. Крім того, він зможе переглянути деталі щодо комісії сервісу і загального прибутку за визначений період часу, включаючи аналіз за тиждень або за весь період існування магазину.

За допомогою налаштовуваних рамок часу власник може змінювати періоди аналізу та отримувати актуальну інформацію про продуктивність свого бізнесу. Також доступна можливість перегляду загальної статистики за конкретний магазин, що дозволяє зробити індивідуальні висновки і приймати

управлінські рішення на основі даних (Рис. 3.6.)

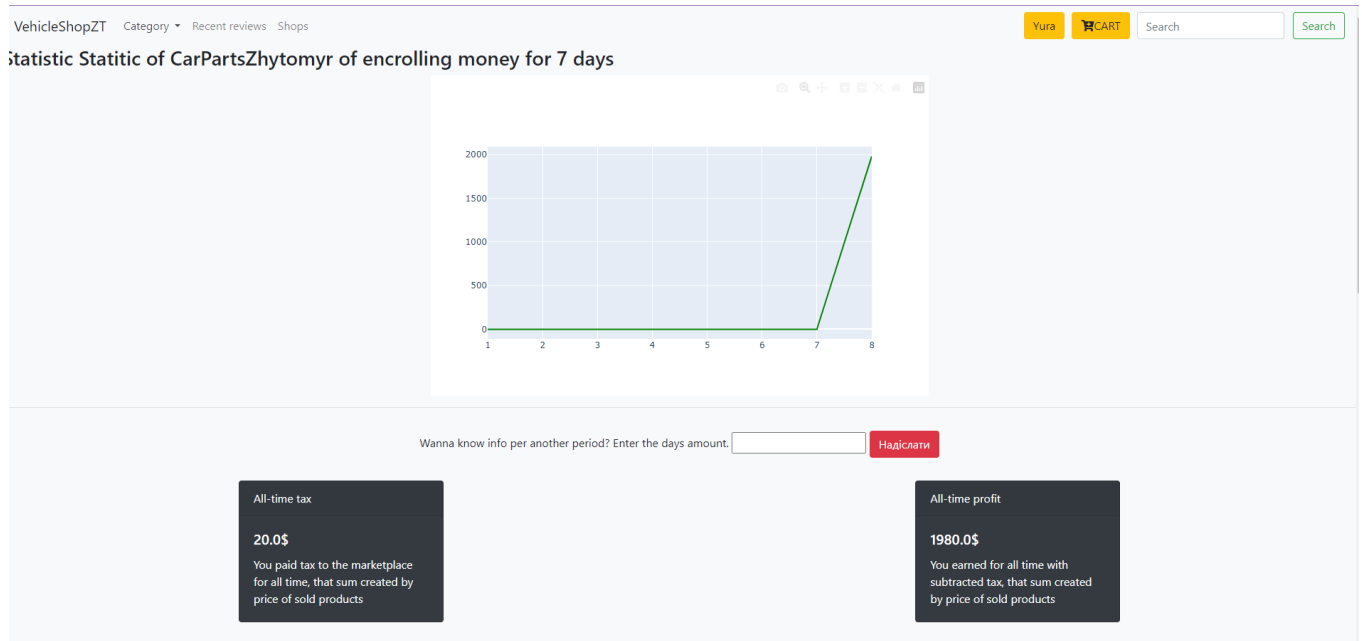


Рис 3.6. Сторінка статистики конкретного магазину

На адміністративній сторінці магазину користувач має широкі можливості для управління своїм бізнесом. Він може легко додавати нові товари, а також редагувати існуючі, щоб підтримувати актуальність асортименту і відповідати потребам клієнтів. Крім цього, на адміністративній панелі доступна можливість написати повідомлення адміністраторам магазину для швидкого вирішення питань і вироблення рішень. Користувач також може ефективно керувати замовленнями, відстежуючи їх статус і забезпечуючи вчасну обробку та доставку продукції.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		45

VehicleShopZTCategoryRecent reviewsShopsYuraCARTSearchSearch

Admin page: CarPartsZhytomyr

Add new product

name

count on storage

description

small description

Characteristics

Price

Category

Submit

Toyota GT 86 Hood

\$1000.0

Toyota GT86 2012-2019 BRZ used original hood.

Edit

Delete

Send message to superadmin (for example "Hi please add new category")

Send

Look at order user with username:

Send

Product	Count	Status	Payment method	User	Button
Toyota GT 86 Hood	1	Wait for payment	Cart number: 4149 4978 4107 5322, Cart date: 10/12, cvv: 321	Yura yura@gmail.com	Submit
cancel that order					

Product	Count	Status	Payment method	User	Button
Toyota GT 86 Hood	1	Wait for payment	Cart number: 4149 4978 4107 5322, Cart date: 10/12, cvv: 321	Yura yura@gmail.com	Submit
cancel that order					

© 2024 Copyright: Yura Kozak
logout

Рис 3.7. Адміністративна сторінка магазину

Також користувач може додати товари до кошику, та в ньому ж змінити деякі дані замовлення, наприклад кількість товару, або видалити позицію з каталогу.

VehicleShopZTCategoryRecent reviewsShopsYuraCARTSearchSearch

CART

Product	Update	Delete	Price(for one)	Fully Price
Toyota GT 86 Hood	2update	Remove	2	\$1000.0 \$2000.0

back to mainmake an order

© 2024 Copyright: Yura Kozak
logout

Рис 3.8. Адміністративна сторінка магазину

На сторінці оплати користувач може ввести свої дані для оплати, та оплатити товар

VehicleShopZT
Category
Recent reviews
Shops
Yura
CART
Search
Search

Payment Information

card number
0123 4567 8910 1112

cardholder name
JOHN DOE

expiration
01/23

Name

Card Number

Expiration (mm/yy)

Security Code

Рис 3.9. Адміністративна сторінка магазину

На адміністративній сторінці усього сайту доступні розширені можливості для управління базою даних. Користувач може переглянути повний список SQL-таблиць, що знаходяться в базі даних, що дозволяє йому зробити детальний огляд структури даних і їх взаємозв'язків. Крім того, доступний перегляд списку записів в кожній таблиці, що дозволяє користувачеві переглядати конкретні дані і виконувати їх редагування або додавання нових записів безпосередньо через адміністративний інтерфейс сайту. Такий функціонал забезпечує зручний і ефективний контроль за базою даних, що є критичним для успішного управління сайтом і його функціоналом. (Рис. 3.10-12) .

Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups
Add
Change

Users
Add
Change

MAIN

Categories
Add
Change

Images
Add
Change

Orders
Add
Change

Products
Add
Change

Reviews
Add
Change

Shops
Add
Change

Users
Add
Change

Recent actions

My actions

- id: 16, name: glass
Category
- id: 15, name: batteries
Category
- id: 14, name: Heating, ventelations
Category
- id: 13, name: Body, external components
Category
- id: 12, name: Filters, Oils
Category
- id: 11, name: Ropes
Category
- id: 10, name: Suspension
Category
- id: 9, name: Accessories
Category
- id: 8, name: Cooling system
Category
- id: 7, name: fuel system
Category

Рис. 3.10 Адміністративна панель – перегляд таблиць в базі даних

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		47

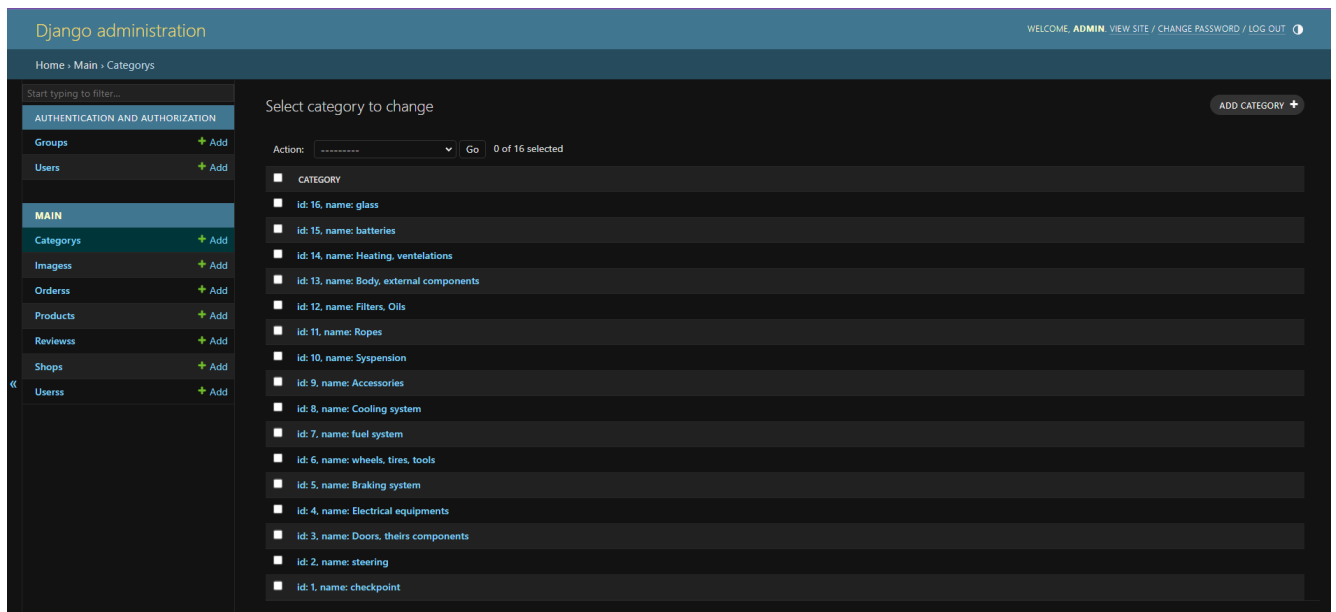


Рис. 3.11 Адміністративна панель – перегляд записів в таблиці

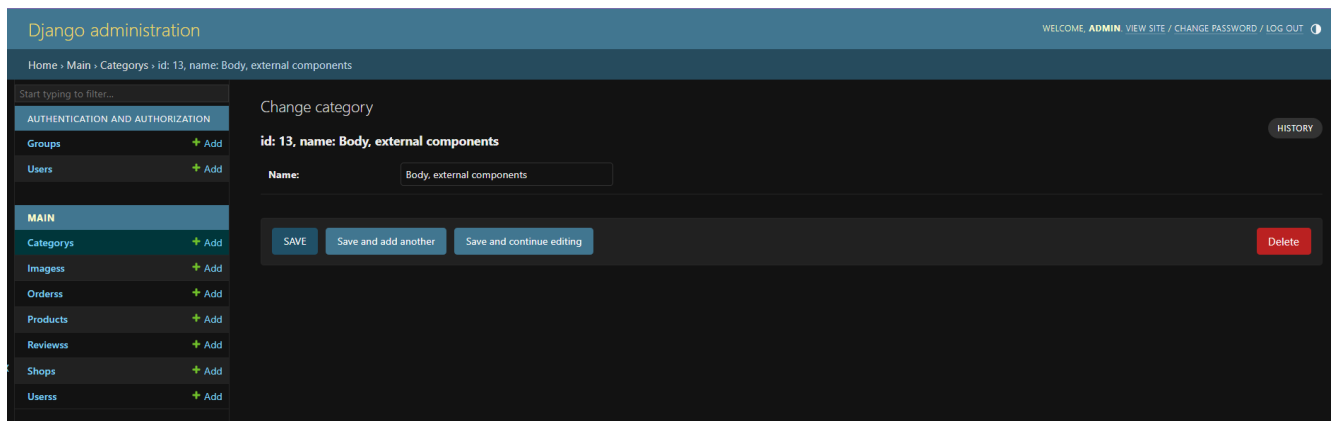


Рис. 3.12 Адміністративна панель – редагування запису

Також якщо зайти під акаунтом суперадміна, не в адмін панель, а на інтернет-магазин, при переході на особистий кабінет, модератор побачить кнопку переходу на модераційно-адміністративну сторінку. (рис. 3.13)

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		48

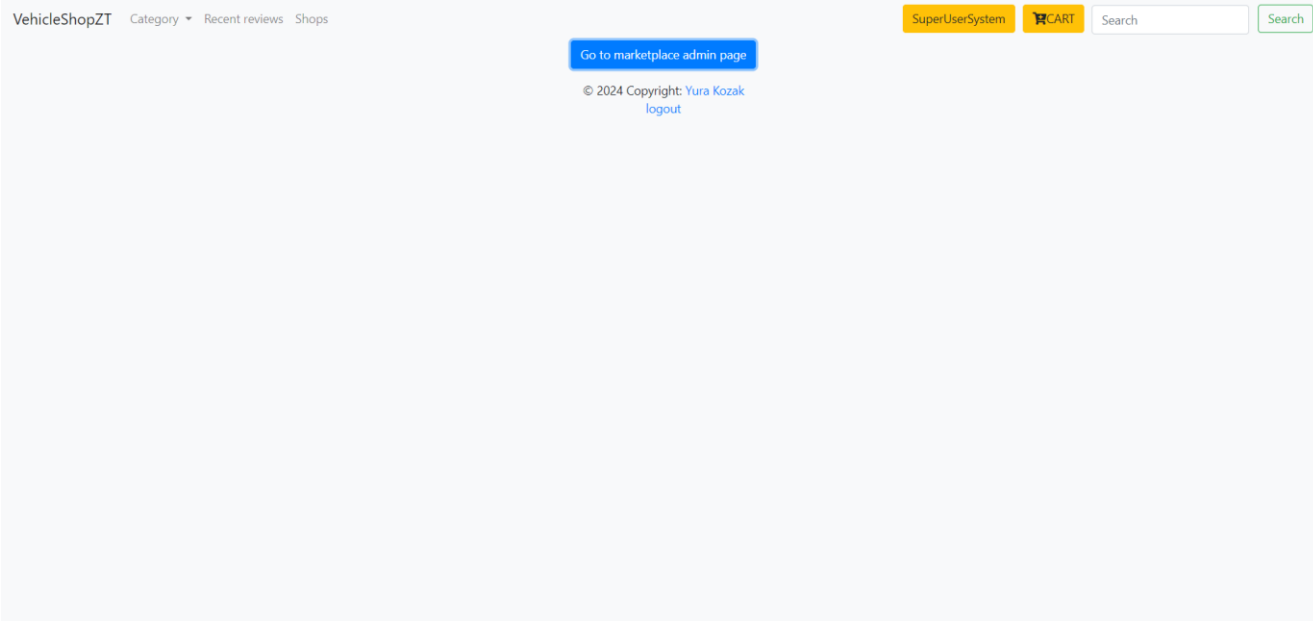


Рис. 3.13 Кабінет суперадміна

Після переходу на модераційно-адміністративну сторінку, також видно деяку статистику, проте її менше. Так само можна побачити статистику за різний період. З цієї ж сторінки можна створювати нові категорії, перейти на адміністративну панель будь-якого магазину з парвами адміна, видалити будь-який магазин. Також на цій же сторінці можна створити бекап, який на справді є дампом пам'яті згідно до специфіки бази, та відновити базу з цього останнього дампу. Також можна побачити прибутки та виплачену комісію магазинів за весь час. (рис. 3.14)

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		49



Рис. 3.14 модерaційно-адміністрaтивнa сторінкa

3.2. Тестування роботи системи

Перевірка на працездатність – необхідна дія під час створення програмного коду для перевірки працездатності системи. Геймплейне тестування відрізняється в певних аспектах від тестування веб-ресурсів, мобільних додатків та інших продуктів.

Тестування веб додатків містить такі основні види:

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		50

- тестування функціональності;
- тестування контенту;
- тестування покупки;
- користувацьке тестування.

Тестування функціональності. Нижче наведено тест-кейси роботи.

Тестування авторизації:

Таблиця 3.1. Тест-кейс авторизація

Test name: User Authorization (Issue id #1)			
Status	Passed		
Created date	2024-06-05	Tested at	2024-06-07
Reporter	-	Assigned to	Yuri Kozak
Pre Conditions: register new account			
Steps	Description	Expected results	
1	Identify the user or entity requesting access.	User or entity information is obtained.	
2	Verify the authenticity of the user or entity.	User or entity is authenticated successfully.	
3	Determine the level of access required.	Access level requirements are identified.	
4	Retrieve the user or entity's access permissions.	User or entity's permissions are retrieved from a database or access control system.	
5	Compare the required access level with the user or entity's permissions.	Access level requirements are compared against the permissions.	
6	Grant or deny access based on the comparison result.	Access is granted if the user or entity's permissions meet the requirements; otherwise, access is denied.	

Тестування покупки:

Таблиця 3.2. Тестування покупки

Test name: Create order (Issue id #2)			
Status	Passed		
Created date	2024-06-05	Created date	2024-06-05
Reporter	-	Reporter	Yuri Kozak
Pre Conditions: Authorization/Registration, cart filled			
Steps	Description	Expected results	
1	Click on buy in cart	Should redirect on pay page	
2	Fill form with payment data	Should check data on inputs	

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		51

3	Hit buy	Should redirect to cabinet page
4	Receive order confirmation on Email	Gmail account should send data about order to settled mail
5	Check data in user cabinet	Order should be rendered at order page

Тестування створення нового товару:

Таблиця 3.3. Тестування створення нового товару

Test name: Create item (Issue id #3)			
Status	Passed		
Created date	2024-06-05	Created date	2024-06-05
Reporter	-	Reporter	Yuri Kozak
Pre Conditions: Authorization/Registration, cart filled			
Steps	Description	Expected results	
1	Go to the product management section.	Open the page	
2	Click on the "Add NewProduct" button.	Request should be sent, and in item table should be new item	
3	Fill an image	Should add item	
4	Click the "Save" button.	Page should redirect	
5	Open main page	An item should be in the first position	

Тестування не заповненості полей:

Таблиця 3.4. Тестування створення нового товару 2

Test name: Missing Required Fields (Issue id #4)			
Status	Passed		
Created date	2024-06-05	Created date	2024-06-05
Reporter	-	Reporter	Yuri Kozak
Pre Conditions: Authorization/Registration, cart filled			
Steps	Description	Expected results	
1	to the product management section	Open the page	
2	Click on the "Add NewProduct" button.	Request should be sent, and in item table should be new item	
3	Leave required fields (e.g., name, description) blank	Should add item	

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		52

4	Click the "Save" button.	Error message appears indicating missing required fields. The item is not created
5	Open main page	An item should not be in the first position

Тестування повторних елементів:

Таблиця 3.5. Тестування створення повторних елементів

Test name: Create duplicate item (Issue id #5)			
Status	Passed		
Created date	2024-06-05	Created date	2024-06-05
Reporter	-	Reporter	Yuri Kozak
Pre Conditions: Authorization/Registration, cart filled			
Steps	Descripriion	Expected results	
1	Go to the product management section	Open the page	
2	Click on the "Add NewProduct" button.	Request should be sent, and in item table should be new item	
3	Click on the "Add New Product" button	Should add item	
4	Fill in all required fields with data that exactly matches an existing item	Page should redirect	
5	Click the "Save" button	An item should be in the first position	

Тестування входу з неправильними даними:

Таблиця 3.6. Тестування входу з неправильними даними

Test name: Login with Invalid Credentials (Issue id #6)			
Status	Passed		
Created date	2024-06-05	Created date	2024-06-05
Reporter	-	Reporter	Yuri Kozak
Pre Conditions: Authorization/Registration, cart filled			
Steps	Descripriion	Expected results	
1	Enter valid username and invalid password the login form	Open the page	

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		53

2	Click the "Login" button	Error message appears indicating invalid credentials. Access is denied.
3	Enter valid password and invalid username the login form	Should add item
4	Click the "Login" button	Error message appears indicating invalid credentials. Access is denied.

Тестування вичерпання сесії:

Таблиця 3.7. Тестування вичерпання сесії

Test name: Expired User Session (Issue id #7)			
Status	Passed		
Created date	2024-06-05	Created date	2024-06-05
Reporter	-	Reporter	Yuri Kozak
Pre Conditions: Expired User Session			
Steps	Descripion	Expected results	
1	User logs in to the system	User gains access	
2	User remains inactive for a set period exceeding the session timeout	N/A	
3	User tries to access a protected resource	User is redirected to the login page due to session expiration	

Тестування користувача із недостатніми правами доступу:

Таблиця 3.8. Тестування користувача із недостатніми правами доступу

Test name: User with Insufficient Permissions (Issue id #8)			
Status	Passed		
Created date	2024-06-05	Created date	2024-06-05
Reporter	-	Reporter	Yuri Kozak
Pre Conditions: Create user accounts with different permission levels.			
Steps	Descripion	Expected results	

1	Open web-page on login page	Rendered login page
2	Login with a user account with limited permissions	N/A
3	Try to access a resource restricted to users with higher permission levels	Error message appears indicating insufficient permissions, or access is denied

Тестування магазинів було проведено і охопило всі аспекти їхньої роботи. У процесі тестування було вивчено основні функції магазинів, включаючи роботу з продуктами, процеси оплати, входу користувачів та реєстрації. Всі виявлені помилки та спірні моменти були ретельно аналізовані та виправлені, а також була проведена оптимізація системи. У результаті тестування було досягнуто стан, коли значних проблем не залишилося, і можна стверджувати, що роботу магазинів можна вважати завершеною і готовою до використання користувачами..

Висновки до третього розділу

У даному розділі було проведено тестування розробленого додатку та порядок роботи з нею. Процес тестування дозволив виявити та виправити різноманітні помилки, недоліки та проблеми. Тестування включало в себе перевірку функціональності різних аспектів, перевірку на відповідність специфікаціям та вимогам проекту, а також оцінку загального досвіду користувача.

В результаті тестування були виявлені та виправлені ряд технічних проблем, таких як баги, помилки у логіці додатку, проблеми з продуктивністю та оптимізацією. Крім того, в процесі тестування було звернуто увагу на зручність та інтуїтивність.

Встановлений порядок роботи з додатком включає, режстрацію, створення замовлень, створення магазинів, написання відгуків.

Завдяки проведеному тестуванню та встановленому порядку роботи з грою було досягнуто високої якості. Розроблений додаток виявилася функціональним та стабільним.

В цілому, третій розділ був спрямований на тестування, що дозволило виявити та виправити проблеми.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		56

ВИСНОВКИ

У даній роботі було розглянуто процес розробки платформи для продажу автомобільних запчастин, яка побудована на основі фреймворку Django.

Мета дослідження полягала у створенні ефективного та зручного веб-додатку, що дозволяє користувачам здійснювати пошук, вибір та купівлю запчастин.

Вступ надав загальний огляд важливості інформаційних технологій у сучасному світі та обґрунтував вибір теми дослідження. Ми визначили, що розробка платформи продажу автомобільних запчастин є актуальною і затребуваною задачею, що сприяє підвищенню ефективності бізнес-процесів та зручності для користувачів.

В роботі було здійснено аналіз напрямків використання інформаційних технологій, постановці задачі, аналізу аналогів програмного продукту та обґрунтуванню вибору інструментальних засобів. Було проведено детальний аналіз існуючих рішень на ринку, що дозволило визначити їхні сильні та слабкі сторони. На основі цього аналізу були сформульовані вимоги до нашого додатку та обґрунтовано вибір Python та Django як основних технологій для його розробки.

Робота охоплює процес проектування програмного забезпечення. Було визначено варіанти використання та об'єктно-орієнтовану структуру системи, розроблено технічне завдання та здійснено проектування бази даних. Також детально описано алгоритми роботи системи та їх реалізацію у вигляді коду. Завершальний підрозділ присвячений безпосередній розробці додатку, де детально описали кроки створення проекту, структуру файлів та взаємодію між компонентами системи.

Після розробки, робота зосереджувалась на інтерфейсі та порядку роботи з додатком. Було описано структуру інтерфейсу користувача, функціональні можливості та порядок взаємодії користувачів із системою. Також приділили увагу тестуванню роботи системи, щоб переконатися у її коректності та надійності.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		57

На основі проведеного дослідження можна зробити висновок, що розробка платформи для продажу автомобільних запчастин на основі Django є ефективним рішенням.

Завдяки проведеному аналізу, проектуванню та реалізації системи було досягнуто поставлених цілей, а розроблена платформа готова до впровадження та подальшого використання. Сподіваємося, що результати даної роботи будуть корисними для інших розробників та сприятимуть подальшому розвитку подібних проектів.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		58

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mccarthy A. Django documentation¶ [Електронний ресурс] / Andrew Mccarthy // Django. – 2021. – Режим доступу до ресурсу: <https://docs.djangoproject.com/en/3.2/>.
2. SQLite Documentation [Електронний ресурс] // SQLite. – 2021. – Режим доступу до ресурсу: <https://www.sqlite.org/docs.html>.
3. Python 3.9.5 documentation [Електронний ресурс] // Python org. – 2021. – Режим доступу до ресурсу: <https://docs.python.org/3/>.
4. The Django Book [Електронний ресурс] // Matt Behrens. Revision 694e4a37.. – 2012. – Режим доступу до ресурсу: <https://django-book.readthedocs.io/en/latest/chapter05.html>.
5. Django for Professionals: Production Websites with Python & Django – los angeles: Independently published, 2019. – 380 с. – (Williams College). – (978; кн. 1081582166).
6. SQLite Forensics – San-Francisco: Independently published, 2018. – 315 с. – (Paul Sanderson). – (978; 1980293071).
7. Python and SQLite Development – London: Microsoft MVP, 2021. – 48 с. – (B08SKK4F99).
8. SQLite Tutorial [Електронний ресурс] // SQLite Tutorial. – 2021. – Режим доступу до ресурсу: <https://www.sqlitetutorial.net/>.
9. Python Django Web Framework - Full Course for Beginners [Електронний ресурс] // freeCodeCamp.org. – 2019. – Режим доступу до ресурсу: <https://youtu.be/F5mRW0jo-U4>.
10. Тести і практика: Django [Електронний ресурс] // 2021 – Режим доступу до ресурсу: <https://itproger.com/practice/django>.
11. Docs to SQLite [Електронний ресурс] // 15. – 2023. – Режим доступу до ресурсу: <https://www.sqlite.org/docs.html>.
12. Working with AJAX in Django [Електронний ресурс] // testdriven. – 2021. – Режим доступу до ресурсу: <https://testdriven.io/blog/django-ajax-xhr/>.

13. djangoproject [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://www.djangoproject.com/>.

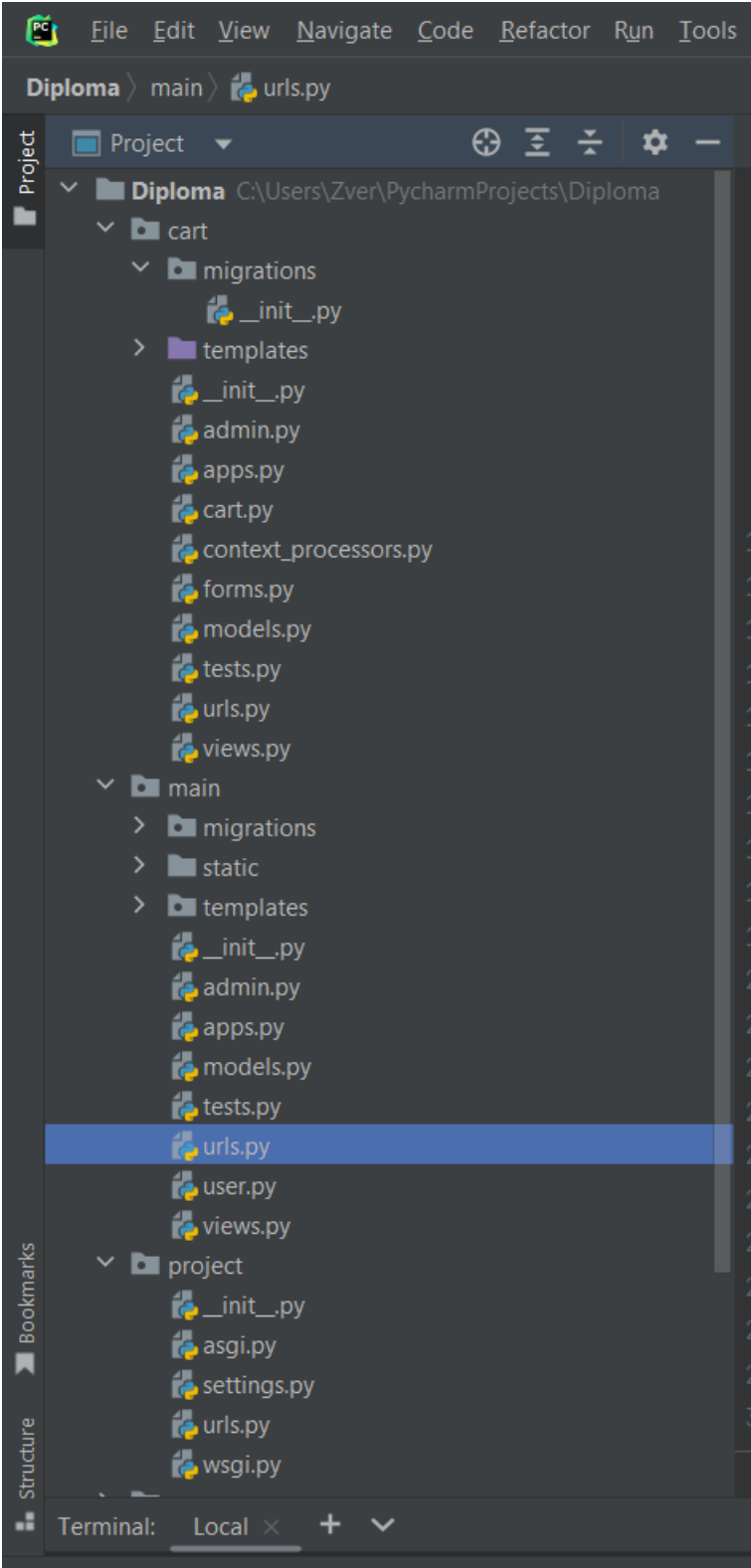
14. metanit [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://metanit.com/python/django/1.2.php>.

15. Что же такое Django? [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://habr.com/ru/articles/747234/>.

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		60

ДОДАТКИ

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		61



Лістинг коду класу Cart:

```

class Cart(object):
    def __init__(self, request):
        """
        initialization of cart
        """
        self.session = request.session
        cart = self.session.get(settings.CART_SESSION_ID)
        if not cart:
            # save empty into session
            cart = self.session[settings.CART_SESSION_ID] = {}
        self.cart = cart

    def __len__(self):
        return sum(item['quantity'] for item in self.cart.values())

    def __iter__(self):
        """
        foreach item in cart get product from db
        """
        product_ids = self.cart.keys()
        products = Product.objects.filter(product_id__in=product_ids)

        cart = self.cart.copy()
        for product in products:
            cart[str(product.product_id)]['product'] = product

        for item in cart.values():
            item['price'] = Decimal(item['price'])
            item['total_price'] = item['price'] * item['quantity']
            yield item

    def __str__(self):
        return str(self.cart)

    def add(self, product, quantity=1, update_quantity=False):
        product_id = product.product_id
        if product_id not in self.cart:
            self.cart[str(product_id)] = {'quantity': 0,
                                           'price': str(product.price)}
        if update_quantity:
            self.cart[str(product_id)]['quantity'] = quantity
        else:
            self.cart[str(product_id)]['quantity'] += quantity
        self.save()

    def save(self):
        self.session.modified = True

    def remove(self, product):
        """remove product from cart"""
        product_id = product.product_id
        print(self.cart)
        if str(product_id) in self.cart:
            self.cart.pop(str(product_id))
            print(self.cart)
            self.save()

    def get_total_price(self):

```

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		63

```

return sum(Decimal(item['price']) * item['quantity'] for item in self.cart.values())

def clear(self):
    del self.session[settings.CART_SESSION_ID]
    self.save()

```

Лістинг коду класу User:

```

from django.conf import settings
from main.models import Users

class User(object):
    def __init__(self, request, user_var=None):
        self.session = request.session
        user = self.session.get(settings.USER_SESSION_ID)
        if not user:
            # save empty into session
            if user_var != None:
                user = self.session[settings.USER_SESSION_ID] = user_var
        self.user = user

    def logout(self):
        del self.session[settings.USER_SESSION_ID]
        self.save()

    def save(self):
        self.session.modified = True

```

Лістинг файлу /main/models:

```

from django.db import models
import datetime

class Category(models.Model):
    category_id = models.AutoField(verbose_name='id', auto_created=True, primary_key=True)
    category_name = models.CharField("name", max_length=50)

    def __str__(self):
        return f'id: {self.category_id}, name: {self.category_name}'

class Users(models.Model):
    user_id = models.AutoField(verbose_name='id', auto_created=True, primary_key=True)
    email = models.CharField("email", max_length=200, null=False, blank=False)
    username = models.CharField("username", max_length=50, null=False, blank=False)
    pwd = models.TextField("pwd", null=False, blank=False)
    homeAddress = models.TextField("homeAddress", null=False, blank=False)
    UserStatus = models.IntegerField("UserStatus", null=False, blank=False, default=0)

    def __str__(self):
        return f'user {self.username}, email {self.email}'

class Shop(models.Model):
    shop_id = models.AutoField(verbose_name='id', auto_created=True, primary_key=True)
    name = models.CharField('name', max_length=100)
    dataCreation = models.DateTimeField()
    owner = models.ForeignKey(Users, on_delete=models.CASCADE)

    def __str__(self):

```

		Ю.О.КОЗАК			ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
		О.В.КУЗЬМЕНКО				
Змн.	Арк.	№ докум.	Підпис	Дата		64

```

    return f"name {self.name}, shop_id {self.shop_id}"

class Product(models.Model):
    product_id = models.AutoField(verbose_name='id', auto_created=True, primary_key=True)
    name = models.CharField('name', max_length=100)
    count_on_storage = models.IntegerField("count_on_storage", blank=False, null=False)
    description = models.TextField("description", blank=True, null=False)
    price = models.FloatField("price", blank=False, null=False)
    smallDesc = models.CharField("smallDesc", max_length=150)
    characteristics = models.TextField("characteristics", blank=True, null=True)
    category_id = models.ForeignKey(Category, on_delete=models.CASCADE)
    shop_id = models.ForeignKey(Shop, on_delete=models.CASCADE)

    def __str__(self):
        return f"name {self.name}, id {self.product_id}, price {self.price}"

class Orders(models.Model):
    order_id = models.AutoField(verbose_name='id', auto_created=True, primary_key=True)
    user_id = models.ForeignKey(Users, on_delete=models.CASCADE)
    shop_id = models.ForeignKey(Shop, on_delete=models.CASCADE)
    product_id = models.ForeignKey(Product, on_delete=models.CASCADE)
    count = models.IntegerField(default=1)
    status = models.TextField('status', default="data is being verified")
    payment_methond = models.TextField()
    data = models.DateTimeField(default=(datetime.datetime.now()).replace(hour=0, minute=0, second=0, microsecond=0), auto_created=True)

    def __str__(self):
        return f"order_id {self.order_id}, user_id {self.user_id}"

class Reviews(models.Model):
    rev_id = models.AutoField(verbose_name='id', auto_created=True, primary_key=True)
    product_id = models.ForeignKey(Product, on_delete=models.CASCADE)
    user_id = models.ForeignKey(Users, on_delete=models.CASCADE)
    rev_text = models.TextField('RevText')
    rev_mark = models.IntegerField("rev_mark", null=False, blank=False, default=1)
    order_id = models.ForeignKey(Orders, on_delete=models.CASCADE, default=None)

    def __str__(self):
        return self.rev_text

class Images(models.Model):
    img_ID = models.AutoField(verbose_name='id', auto_created=True, primary_key=True)
    path = models.TextField('path')
    product_id = models.ForeignKey(Product, on_delete=models.CASCADE)

    def __str__(self):
        return f"{self.img_ID} {self.path} {self.product_id}"

class TempFileStorage(models.Model):
    image = models.ImageField(null=False, blank=False)

```