НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2022 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою “ Комп’ютерні системи та мережі”
спеціальності 123 “ Комп’ютерна інженерія”

на тему: Web-додаток для продажу та реклами товарів канцелярського призначення

Виконала : студентка __4__ курсу, групи _ІВ-82_
(шифр групи)

_____Чуясова Тетяна Андріївна_____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____асистент Нікольський С.С._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) старший викладач, Сімоненко А. В._____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2022 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“ Комп’ютерні системи та мережі”

спеціальності 123 “ Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2022 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Чуясової Тетяни Андріївни

1. Тема проєкту Web-додаток для продажу та реклами товарів канцелярського призначення
керівник проєкту Нікольський Сергій Сергійович, асистент,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від _____

2. Термін здачі студентом закінченого проєкту *11 червня 2022 р.*

3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Аналіз предметної області.

Розділ 2. Вибір технології розробки та його обгругнування.

Розділ 3. Деталі розробки системи.

Розділ 4. Тестування та впровадження розробленої системи.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) структурна схема(структурна схема програмного забезпечення), функціональна схема (діаграма класів), принципова схема(алгоритм дій програмного забезпечення).

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко А. В.		

7. Дата видачі завдання «19» грудня 2021 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*10.12.2021-20.12.2021*	
2.	*Вивчення та аналіз завдання*	*21.12.2021-18.03.2022*	
3.	*Розробка архітектури та загальної структури системи*	*18.03.2022-25.03.2022*	
4.	*Розробка структур окремих підсистем*	*26.03.2022-29.04.2022*	
5.	*Програмна реалізація системи*	*30.04.2022-20.05.2022*	
6.	*Оформлення пояснювальної записки*	*21.05.2022-1.06.2022*	
7.	*Захист програмного продукту*	*27.05.2022*	
8.	*Передзахист*	*11.06.2022*	
9.	*Захист*	*.06.2022*	

Студент-дипломник _____ Тетяна ЧУЯСОВА
(підпис)

Керівник проєкту _____ Сергій НІКОЛЬСЬКИЙ
(підпис)

АНОТАЦІЯ

У ході виконання даної дипломної роботи були розглянуті ріноманітні види веб-додатків, технології їх створення та принципи роботи. Проаналізовано всі переваги та недоліки програмних засобів для реалізації систем заданого типу. На основі аналізу обрано стек технологій, що найкраще підходить для вирішення задачі створення веб-додатку для продажу товарів. Результатом роботи є розроблена система, яка являє собою веб-додаток для продажу та реклами товарів канцелярського призначення. Розроблений програмний продукт надає можливість власнику бізнесу швидко отримувати та оброблювати замовлення клінтів, які у свою чергу мають змогу комфортно робити покупки онлайн. Даний програмний продукт розроблено з використанням мов програмування Python та JavaScript, а також бібліотеки jQuery, мови розмітки HTML та каскадної таблиці стилів CSS.

Ключові слова: веб-додаток, Python, JavaScript.

ANNOTATION

During the implementation of this project for a Bachelor's Degree were considered various types of web applications, technologies for their creation and principles of work. All the advantages and disadvantages of software for the implementation of systems of a given type are analyzed. Based on the analysis, a technology stack was selected that is best suited to solve the problem of creating a web application for selling goods. The result is a developed system, which is a web application for the sale and advertising of stationery. The developed software product allows the business owner to quickly receive and process customer orders, which in turn can comfortably shop online. This software is developed using Python and JavaScript programming languages, as well as the jQuery library, HTML markup language and cascading CSS style sheet.

Keywords: web application, Python, JavaScript.

справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпля	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467100.002 ТЗ***	Web – додаток для продажу	4		
			та реклами товарів			
			канцелярського призначення			
			Технічне завдання			
	А4	***ІАЛЦ.467100.003 ПЗ***	Web – додаток для продажу	112		
			та реклами товарів			
			канцелярського призначення			
			Пояснювальна записка			
	А4	***ІАЛЦ.467100.004 Д1***	Web – додаток для продажу	1		
			та реклами товарів			
			канцелярського призначення			
			Структурна схема			
			системи			
	А4	***ІАЛЦ.4671008.005 Д2***	Web – додаток для продажу	1		
			та реклами товарів			
			канцелярського призначення			
			Функціональна схема			
			(діаграма класів)			
	А4	***ІАЛЦ.467100.006 Д3***	Web – додаток для продажу	1		
			та реклами товарів			
			канцелярського призначення			
			Принципова схема			
			(алгорим дії додатку)			
	А4	***ІАЛЦ.467100.007 Д4***	Web – додаток для продажу	14		
			та реклами товарів			
			канцелярського призначення			
			Текст програмного коду			

					ІАЛЦ.467100.001 ОА	
Зм	*Лист*	*№ докум.*	*Підп*	*Дата*		
Розроб		Чуясова Т.А.			*Web-додаток для продажу та реклами товарів канцелярського призначення* Опис альбому	Літ. / Аркуш 1 / Аркушів 1
Перев		Нікольський С.С.				
						НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІВ-82

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Web-додаток для продажу та реклами товарів канцелярського призначення*»

Київ – 2022

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку системи що являє собою веб-додаток для продаж товарів, а також на подальшу підтримку та вдосконалення розробленого веб-застосунку.

Областю застосування даної системи є онлайн шопінг, інтернет-магазини, інтернет-маркетинг.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп’ютерної інженерії», який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням даної роботи є розробка веб-додатку, який надаватиме можливість переглянути та придбати різні товари концелярського призначення.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки даного дипломного проекту є науково-технічна література, вофіційні документації, публікації, статті та відеоматеріали в мережі Інтернет на дану тему.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги:

- Зручний та інтуїтивно-зрозумілий інтерфейс користувача.
- Надати можливість користувачам застосовувати фільтри для пошуку бажаного товару.
- Надати користувачам можливість реєструватися.
- Надати можливість користувачам робити замовлення.
- Надати можливість користувачам стежити за статусом замовлення.
- Надати можливість адміністратору бачити список замовлень.
- Надати адміністратору можливість бачити список зареєстрованих користувачів.
- Надати адміністратору можливість змінювати статус замовлення.
- Надати адміністратору можливість бачити наявні на складі товари.
- Надати вичерпну та зрозумілу документацію для розробленого продукту.

5.2. Вимоги до програмного забезпечення

- ОС Windows, Mac чи Linux.
- Pycharm PyCharm Community Edition 2021.1.1 IDE версії або вище.
- Браузери Google Chrome версії 60 і вище, Internet Explorer версії 10 і вище, Firefox версії 4 і вище.

5.3. Вимоги до апаратної частини

- ЦП не менше ніж Intel® Core (TM) i3-2100T.
- ROM не менше ніж 64 ГБ.
- RAM не менше ніж 4 ГБ.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	10.12.2021-20.12.2021
Вивчення та аналіз завдання	21.12.2021-18.03.2022
Розробка архітектури та загальної структури системи	18.03.2022-25.03.2022
Розробка структур окремих частин системи	26.03.2022-29.04.2022
Програмна реалізація системи	30.04.2022-20.05.2022
Виправлення помилок	20.05.2022-21.05.2022
Оформлення пояснювальної записки	21.05.2022-1.06.2022

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Web-додаток для продажу та реклами товарів канцелярського призначення*»

Київ – 2022

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AJAX	(Asynchronous JavaScript And XML) Асинхронний підхід до побудови користувацьких інтерфейсів
API	(Application Programming Interface) Програмний інтерфейс застосунку
ASP	(Active Server Pages) Технологія створення веб-додатків
CLI	(Command Line Interface) Інтерфейс програмного рядка
CLR	(Common Language Runtime) Загальномовне виконуюче середовище
CMS	(Content Management System) Система керування контентом
CSS	(Cascading Style Sheets) Спеціальна мова стилю сторінок
ECMA	(ECMA Script) Стандарт мови програмування
HTML	(HyperText Markup Language) Мова розмітки гіпертексту
HTTP	(HyperText Transfer Protocol) Протокол передачі гіпертексту
HTTPS	(HyperText Transfer Protocol Secure) Протокол передачі гіпертексту з підтримкою шифрування
JSON	(JavaScript Object Notation) Текстовий формат обміну даними між комп'ютерами
MPA	(Multi-page Application) Багатосторінковий веб-додаток
MVC	(Model-View-Controller) Архітектурний шаблон виду модель–вигляд–контролер
PWA	(Progressive Web Application) Поступовий вебзастосунок
SPA	(Single-page Application) Односторінковий веб-додаток
SQL	(Structured Query Language) Мова структурованих запитів
SVG	(Scalable Vector Graphics) Масштабована векторна графіка
TCP/IP	(Transmission Control Protocol/Internet Protocol) Набір протоколів мережі Інтернет
UI	(User Interface) Інтерфейс користувача

URL	(Uniform Resource Locator) Уніфікований локатор ресурсу
UX	(User Experience) Досвід користувача
XML	(Extensible Markup Language) Стандарт побудови мов розмітки
XUL	(XML User Interface Language) Мова розмітки для створення графічних інтерфейсів користувача

ВСТУП

На сьогоднішній час, популярність та актуальність створення веб-застосунків зростає з кожним днем. За останні три роки світ кардинально змінився, під впливом пандемії та війни велика кількість сфер життя людей була перенесена в онлайн. Ми вже не уявляємо свій будній день без лекцій у віртуальній кімнаті, робочих мітингів у Тімсі, шопінгу в онлайн магазині або доставки смачної вечері через веб-застосунок. Така діджиталізація спростила шляхи існування та комунікації людей, водночас створивши шалений попит на розробку програмного забезпечення різноманітних веб-додатків та сервісів.

Окрему нішу у веб галузі почесно займають так звані інтернет-магазини. Такий вид магазину є неймовірно вигідним як для клієнта, так і для продавця, перший може робити покупки не виходячи з дому, а другий заощаджує величезну кількість грошей через відсутність потреби оренди приміщення, пожежних ліцензій, та навіть деяких податків.

Тож темою даної роботи було обрано саме створення веб-додатку для реклами та продажу товарів канцелярського призначення. Канцелярське приладдя обрано як основний товар магазину через шалений попит на нього серед дітей та підлітків, який з’явився під впливом західної культури естетичного робочого місця. Використовуючи даний веб-додаток також можна отримати додатковий дохід з реклами Інстаграм-сторінок інших магазинів, яка розміщується у відповідній секції.

Актуальність теми: зростання попиту на веб-додатки, через всесвітні пандемії та війну в Україні. Фізичні магазини та заклади не мають змоги працювати та сплачувати за оренду місця, покупці ,у свою чергу, не мають можливості відвідувати їх, у той час , використовуючи веб-додатки, такі вищезазначені проблеми зникають.

Мета розробки: створення веб-додатку, за допомогою якого користувач зможе обирати та замовляти канцелярське приладдя, а власники інших інтернет-магазинів замовляти рекламу.

Завдання розробки:

- Створення бази даних доступних товарів;
- Створення можливості користувача реєструватися та відстежувати зроблене замовлення;
- Створення можливості відображення на сторінці товарів, які відповідають втановленим користувачем фільтрів;
- Створення можливості робити замовлення зареєстрованим користувачем;
- Створення секції для розміщення реклами інших інтернет-магазинів;
- Створення бази даних зареєстрованих користувачів;
- Інтеграція веб-застосунку з вищезазначеною базою даних.

Практичне значення одержаних результатів: розроблений веб-додаток надасть можливість користувачу замовити товар та відстежувати статус замовлення.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Історична довідка

Історія створення та розробки веб-застосунків та веб-технологій в цілому дуже цікава та багатогранна, хоч , відносно всесвітньої історії, досить нова. Почнемо з того, хто ж все ж таки винайшов та запровадив перший веб-застосунок. Дізнатися відповідь на це питання можна повернувшись на початок 1990-х років, коли Інтернет був заповнений текстовими документами , які являли собою звичайні статичні HTML-сторінки. Лише пізніше з'явилася можливість додавати на веб-сторінки зображення, відео та навіть аудіофайли. Але вони, як і раніше, були статичними. Саме прагнення зробити HTML-сторінки динамічнішими призвело до розробки в 1995 році клієнтської мови програмування Java Script[1]. Він дозволяв відображати веб-сторінки з різними інтерактивними елементами, включаючи навіть векторну анімацію. Безповоротний перехід від статичних до динамічних веб-сторінок відбувся лише в 2005 році з введенням Ajax - нового підходу до адаптивного веб-дизайну та розробки веб-додатків. Термін веб-застосунку закріпив своє становище в історії Інтернету. Але цього було недостатньо, із стрімким розвитком технологій у 10-х роках 21-го сторіччя, все більше зростала потреба у новому рівні веб-додатків, тому в 2015 році Алекс Рассел і Френсіс Беррман представили Progressive Web Application (PWA), тобто “поступовий” веб-застосунок, що являв собою гібрид звичайних веб-сторінок та мобільних додатків [1]. Іншими словами, веб-додаток отримав функціональність нативної програми, і користувач не міг побачити жодної різниці між нативною програмою та PWA.

Технології постійно змінюються та вдосконалюються з кожним днем, тож як зміниться технології веб-застосунків найближчими роками ми зможемо побачити на власні очі вже зовсім скоро.

1.2 Класифікація веб-технологій

Всі технології, що застосовуються при створенні веб-систем, загалом можна поділити на два основні класи: виконувані на стороні клієнта засобами Інтернет-браузера (наприклад: HTML, CSS, JavaScript, Flash, ActiveX тощо) та виконувані на стороні сервера засобами веб-сервера (наприклад: SSI, PHP , ASP, C#, Perl, Python тощо) та пов'язаних з ним систем (MySQL, PostgreSQL, MSSQL тощо)[2].

Програми, що виконуються на сервері, практично нічим не обмежені за складністю: можуть виконувати будь-які перетворення інформації, а потім формувати потік даних, який може бути візуально представлений користувачеві засобами Інтернет-браузерів. У той час формати даних, які можуть бути оброблені на клієнті, обмежені досить вузьким набором технологій, стандартів та певними рамками, але саме це дозволяє уніфікувати робоче місце користувача та не вимагати від нього встановлення будь-якого додаткового програмного забезпечення, крім Інтернет-браузера.

Також Студопедія [3] пропонує наступну, більш розширену, класифікацію веб-технологій:

- Забезпечують статичне відображення сторінок:
    - Мова розмітки;
    - Мова стильових описів;

- Забезпечують інтерактивне відображення сторінок та взаємодію з ними;
    - На стороні клієнта;
    - На стороні сервера;

1.2.1 Визначення веб-технологій

Перш за все, потрібно зазначити, що будь-яка веб-технологія являє собою частину інформаційних технологій. Інформаційною технологією можна назвати сукупність певних методів та програмно-технічних засобів, які можуть бути інтегровані з метою ефективного опрацювання даних[4]. Що являє собою веб-технологія можна наглядно побачити на схемі (рис. 1.1).

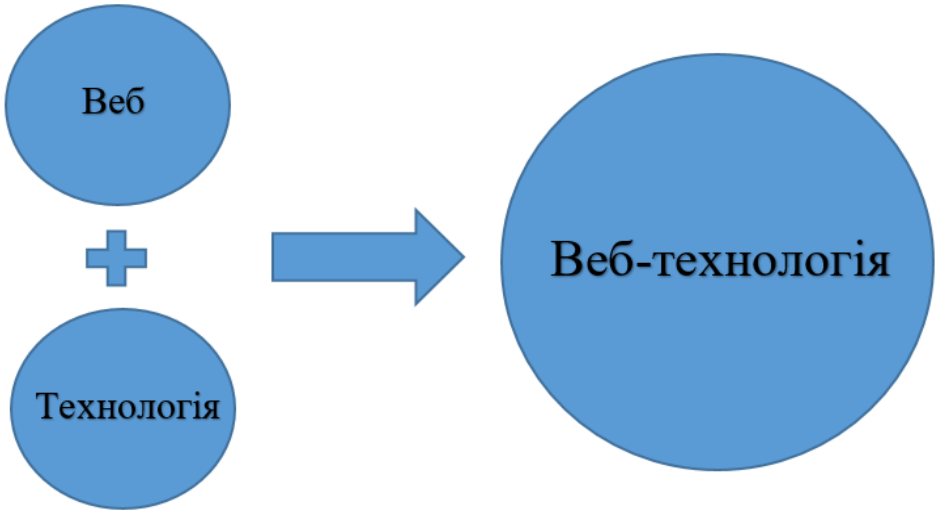


Рисунок 1.1 – Графічне представлення терміну “Веб-технологія”

Визначення веб-технології можна сформулювати наступним чином – це сукупність методів та програмно-технічних засобів, інтегрованих з метою ефективного опрацювання веб-ресурсів, що знаходяться у веб-просторі[4]. Веб-простір можна описати як глобальний інформаційний простір, що заснований на фізичній інфраструктурі Інтернету та протоколі передачі даних HTTP[4].

Веб-ресурси у свою чергу, являють собою віртуальні файли, котрі зберігаються в системі, кожен веб-ресурс має своє унікальне ім'я, яке можна додати до URL-адреси для отримання доступу до того чи іншого файлу[5]. Веб технології також можна поділити на так званий Фронтенд(рис. 1.2) - частина веб-сайту, з якою користувач безпосередньо взаємодіє, називається інтерфейсом[6]. Та Бекенд технології розробки(рис. 1.3) – серверна частина веб-сайту[6].

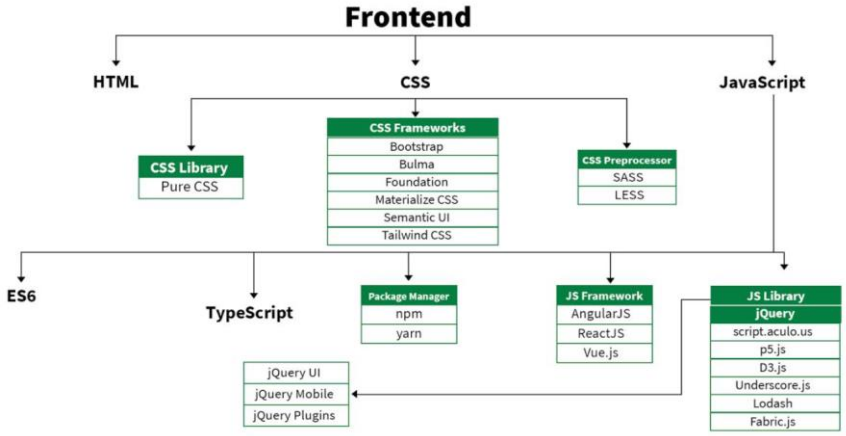


Рисунок 1.2 – Класифікація Frontend-технологій

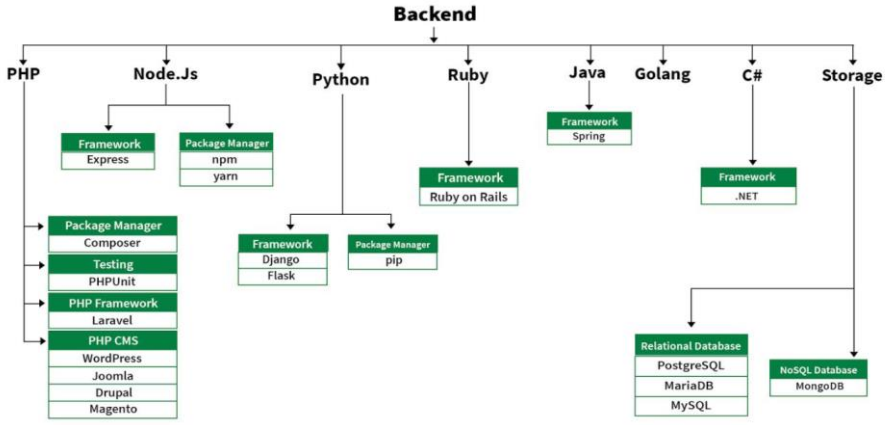


Рисунок 1.3 – Класифікація Backend-технологій

Його також називають «клієнтською стороною» програми.

1.2.2 Структура та принцип роботи Всесвітньої мережі Інтернет

На сьогоднішній день, жодна людина не може уявити своє життя без

Інтернету, але мало хто замислюється, що це не просто слово, а надскладна конструкція, яка поєднує у собі величезну кількість мереж. Керуючись трактуванням терміну “Інтернет” Джона Квотермана [7], можна зазначити, що Інтернет – це метамережа, що складається з багатьох мереж, які працюють згідно з протоколами сімейства TCP/IP, об'єднані через шлюзи і використовують єдиний адресний простір та простір імен. Середовище передачі даних в Internet не можна розглядати лише як величезну павутину проводів чи оптоволоконних ліній. Відцифровані дані пересилаються через маршрутизатори, які з'єднують мережі та за допомогою складних алгоритмів вибирають найкращі маршрути для інформаційних потоків (рис. 1.4).

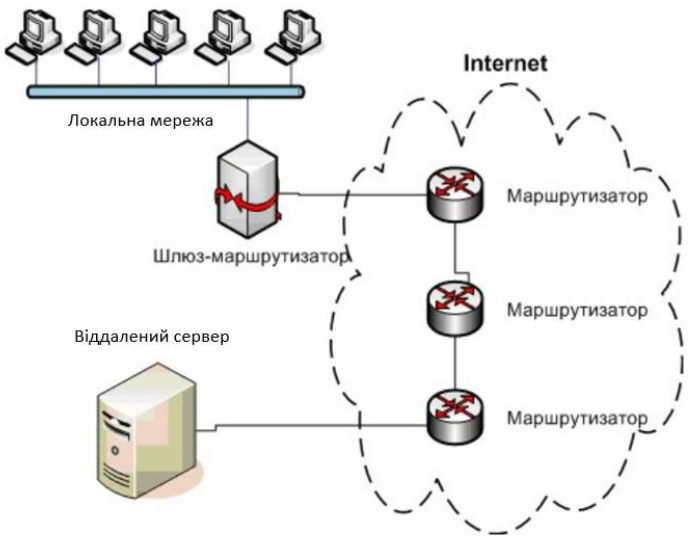


Рисунок 1.4 – Схема взаємодії у мережі Інтернет

На відміну від локальних мереж, у складі яких є свої високошвидкісні канали передачі інформації, глобальна мережа включає підмережу зв'язку до якої підключаються локальні мережі та інші окремі компоненти[8].

Підмережа зв'язку складається з каналів передачі інформації та комунікаційних вузлів, які призначені для передачі даних по мережі, вибору оптимального маршруту передачі інформації, комутації пакетів та реалізації

низки інших функцій за допомогою комп'ютера та відповідного програмного забезпечення. Комп'ютери, за якими працюють користувачі-клієнти, називаються робочими станціями, а комп'ютери, що є джерелами ресурсів мережі, що надаються користувачам, називаються серверами. Саме така структура мережі отримала назву вузлової[8].

Структуру мережі Інтернет можна описати наступним чином(рис. 1.5):

1. Магістральний рівень, тобто система пов'язаних високошвидкісних телекомунікаційних серверів.
2. Рівень мереж та точок доступу, що підключені до магістралі.
3. Рівень регіональних та інших мереж.
4. Користувачі.

Рисунок 1.5 – Структура мережі Інтернет

В основу архітектури мереж покладено багаторівневий принцип передачі повідомлень. Формування повідомлення здійснюється на верхньому рівні моделі ISO/OSI(рис. 1.6). При передачі воно послідовно проходить всі рівні системи до самого нижнього, де і передається по каналу зв'язку адресату. По мірі проходження кожного з рівнів системи повідомлення трансформується, розбивається на порівняно невеликі частини, які мають додаткові заголовки, що

забезпечують інформацією аналогічні рівні на вузлі адресата. У цьому вузлі повідомлення проходить від нижнього рівня до верхнього, знімаючи із себе заголовки. В результаті адресат приймає повідомлення у початковому вигляді.

Найбільш поширеним протоколом управління обміном даних є протокол TCP/IP. Головна відмінність мережі Інтернет від інших мереж полягає у її протоколах TCP/IP, що охоплюють ціле сімейство протоколів взаємодії між комп'ютерами мережі.

Рисунок 1.6 – Модель OSI

Протокол TCP/IP - це сімейство програмно реалізованих протоколів старшого рівня, що не працюють з апаратними перериваннями[8].

1.2.3 Веб-сервер

Термін «веб-сервер» може стосуватися не лише обладнання, а й(або) програмного забезпечення, навіть того й іншого, що працює разом одночасно.

З точки зору обладнання, веб-сервер — це комп'ютер, на якому

зберігається програмне забезпечення веб-сервера та файли компонентів веб-сайту (наприклад, документи HTML, зображення, таблиці стилів CSS та файли JavaScript). Веб-сервер підключається до Інтернету та підтримує фізичний обмін даними з іншими пристроями, підключеними до мережі[9].

Що стосується програмного забезпечення, веб-сервер – це поняття, що включає в собі кілька частин, які контролюють доступ веб-користувачів до розміщених файлів. Як мінімум, це сервер HTTP. У свою чергу HTTP-сервер – це програмне забезпечення, яке розуміє URL-адреси (веб-адреси) та HTTP протоколи (протокол, який Інтернет браузер використовує для перегляду веб-сторінок). Доступ до сервера HTTP можна отримати через доменні імена веб-сайтів, які він зберігає та доставляє вміст цих розміщених веб-сайтів на пристрій кінцевого користувача[9].

Підсумовуючи можна сказати, що щоразу, коли браузеру потрібен файл, розміщений на веб-сервері, браузер запитує файл через HTTP. Коли запит досягає апаратного веб-сервера, (програмний) HTTP-сервер приймає запит, знаходить запитаний документ і відправляє його назад до браузера, також через HTTP.

Щоб опублікувати веб-сайт, потрібно мати статичний або динамічний веб-сервер. Статичний веб-сервер або стек складається з апаратного забезпечення із HTTP-сервером. Такий сервер називається статичним, тому що відправляє розміщені файли у браузер як є, не оновлюючи їх вміст. Динамічний веб-сервер складається зі статичного веб-сервера та додаткового програмного забезпечення, найчастіше серверу додатків та бази даних[9]. Такий вид серверу зветься динамічним, тому що сервер додатків оновлює розміщені файли перед відправкою вмісту у браузер через HTTP-сервер.

1.2.4 Статичні та динамічні веб-технології

Статичні та динамічні веб-технології розробки дозволяють нам створювати відповідні веб-сайти та веб-додатки.

До статичних веб-технологій можна віднести мову розмітки HTML, а також CSS стилі оформлення, що забезпечує їх легкість та швидке завантаження. Статичні сайти складаються з незмінних сторінок. Це означає, що сайт має той самий зовнішній вигляд, а також те саме наповнення для всіх відвідувачів[10]. При запиті такого сайту в браузері сервер надає готовий HTML-документ у вихідному вигляді, в якому він і був створений. Найчастіше статичними бувають сайти з мінімальною кількістю сторінок або контентом, який не потрібно регулярно оновлювати, а саме сайти-візитки, каталоги продукції, довідники технічної документації. Однак, за допомогою сторонніх інструментів існує можливість додати на такі сторінки окремі динамічні елементи (коментарі, особистий кабінет для користувачів, пошук).

Динамічні веб-технології у свою чергу, забезпечують можливість розробки динамічних сайтів. Динамічні сайти мають сторінки, що змінюються, тобто адаптуються під конкретного користувача. Такі сторінки не розміщені на сервері у готовому вигляді, а збираються наново по кожному новому запиту. Спочатку сервер знаходить потрібний документ і відправляє його інтерпретатору, який виконує код з HTML-документа та звіряється з файлами та базою даних. Після цього документ повертається на сервер, а потім відображається в браузері. Для інтерпретації сторінок на серверній стороні використовуються мови програмування Java, PHP, ASP, Python та багато інших.

Найяскравішими прикладами динамічних сайтів є сторінки, створені з урахуванням систем управління контентом (CMS). Серед них найчастіше зустрічаються інтернет-магазини, а також форуми, сторінки з відгуками та інші ресурси із можливістю розміщення контенту відвідувачами.

1.3 Визначення веб-додатку

Веб-додаток - це прикладна програма, яка зберігається на віддаленому сервері і доставляється через інтерфейс Інтернет браузера. Веб-сервіси за визначенням є веб-програмами, які містяться у багатьох веб-сайтах, хоча і не у всіх. За словами Джарела Реміка[11], будь-який компонент веб-сайту, який виконує будь-яку функцію для користувача, вважається веб-програмою.

Веб-додатки можуть бути розроблені для різних цілей і можуть використовуватися будь-ким з багатьох причин. Зазвичай використовувані веб-програми можуть містити веб-пошту, онлайн-калькулятори або магазини електронної комерції. До деяких веб-програм можна отримати доступ лише через певний браузер; однак більшість із них доступні незалежно від браузера.

Веб-застосунки не потрібно завантажувати, оскільки доступ до них здійснюється через мережу. Користувачі можуть отримати доступ до веб-додатків через веб-браузер, такий як Google Chrome, Mozilla Firefox або Safari.

Для роботи веб-додатку йому необхідний веб-сервер, сервер програм та база даних. Веб-сервери керують запитами, що надходять від клієнта, а сервер програм виконує запитане завдання. База даних можна використовувати для зберігання будь-якої необхідної інформації[11].

Веб-застосунки зазвичай мають короткі цикли розробки і можуть створюватися невеликими групами розробників. Більшість веб-застосунків написано на JavaScript, HTML5 або каскадних таблицях стилів (CSS). Програмування сторони клієнта зазвичай використовує вищезазначені мови, які допомагають створювати інтерфейс програми. Програмування на стороні сервера виконується для створення сценаріїв, які використовуватиме веб-програма. Такі мови, як Python, Java та Ruby, зазвичай використовуються у серверному програмуванні[11]. Веб-додатки мають величезну кількість переваг: надання декільком користувачам доступу до однієї версії додатку, не

потребують встановлювання, доступ до веб-додатку можна отримати з різних платформ, доступ можливий з великої кількості різних браузерів.

1.3.1 Види веб-додатків

Загалом існує три основні види веб-додатків: односторінкові (SPA - Single Page Application), багатосторінкові (MPA - Multi Page Application) та прогресивні (PWA - Progressive Web Application)[12].

Односторінковий веб-додаток - це односторінкова веб-програма, яка завантажується на одну сторінку HTML. Завдяки динамічному оновленню (наприклад за допомогою JavaScript) під час використання не потрібно перезавантажувати або підвантажувати додаткові сторінки. На практиці це означає, що користувач бачить у браузері весь основний контент, а при прокручуванні або переходах на інші сторінки замість повного перезавантаження потрібні елементи просто підвантажуються.

Багатосторінковий веб-додаток – це багатосторінкова програма, яка працює за традиційною схемою. Тобто при кожній незначній зміні даних або завантаженні нової інформації оновлюється сторінка[12]. Такі програми важчі, ніж односторінкові, тому їх використання доцільно лише у випадках, коли потрібно відобразити велику кількість контенту.

Прогресивні веб-додатки або Progressive Web Application взаємодіють із користувачем, як програма. Вони можуть встановлюватися на головний екран смартфона, відправляти push-повідомлення та працювати в офлайн-режимі. Чудовим прикладом такого додатку є Google Docs[12].

При виборі типу веб-застосунку потрібно орієнтуватися на те, навіщо саме ви його створюєте. Багатосторінковий вид підійде інтернет-магазину з великою кількістю товарів та послуг, а якщо у вас, наприклад, інфобізнес, де можна викласти всю інформацію в рамках стисненого веб-простору - підійде односторінковий сайт. Наглядно переваги та недоліки кожного виду веб-

додатку можна побачити у таблиці 1.1. Крім технічної класифікації існує також класифікація на основі їх призначень, наприклад для бізнесу, навчання, медецини, тощо.

Таблиця 1.1 – Переваги та недоліки різних видів веб-додатків

Вид додатку	SPA	MPA	PWA
Переваги	1. Велика швидкість сайту досягається завдяки тому, що контент, який не змінюється, не потрібно перезавантажувати. 2.SPA ефективніше в роботі з кешем.	1.Можливість використання готових рішень, таких як OpenCart та WordPress. 2.SEO. Пошукові двигуни пристосовані для індексації багатосторінкових програм.	1.Працюють на будь-якому пристрої. Інтерфейс PWA підлаштовується під ширину екрана комп'ютера або будь-якого телефону бренду. 2.На домашньому екрані телефону можна закріплювати іконку.
Недоліки	1.Користувачі можуть відключити JavaScript у своїх браузерах. У цьому випадку програма може частково не працювати. 2. SPA більш уразливі для атак (XSS).	1.Швидкість взаємодії. MPA перезавантажують контент, коли користувач взаємодіє із ним. 2.Складність розробки. Потрібно окремо писати фронтенд та повноцінний бекенд.	1.На відміну від мобільних програм, PWA не представлені в магазинах AppStore та Google Play. 2.Споживання енергії вище, ніж у простого веб-сайту.

1.3.2 Відмінності веб-додатків та веб-сайтів

Веб-додаток - це частина програмного забезпечення, доступ до якого може отримати браузер. Браузер – це програма, яка використовується для роботи в Інтернеті. Веб-додаток потребує автентифікації. Веб-застосунок використовує комбінацію сценаріїв на стороні сервера та сценаріїв на стороні клієнта для представлення інформації. Також веб-додаток потребує сервера для керування

запитами користувачів[13]. Гарними прикладами веб-додатків можуть бути Google Apps, Amazon, YouTube.У свою чергу веб-сайт являє собою набір пов'язаних веб-сторінок, які містять зображення, текст, аудіо, відео тощо. Він може складатися з будь-якої кількості сторінок. Веб-сайт надає візуальний та текстовий контент, який користувачі можуть переглядати та читати[13]. Для перегляду веб-сайту потрібний браузер (наприклад Chrome, Firefox). Або форум чи блог. Нижче можна побачити наглядну різницю між веб-додатком та веб-сайтом(рис. 1.7).

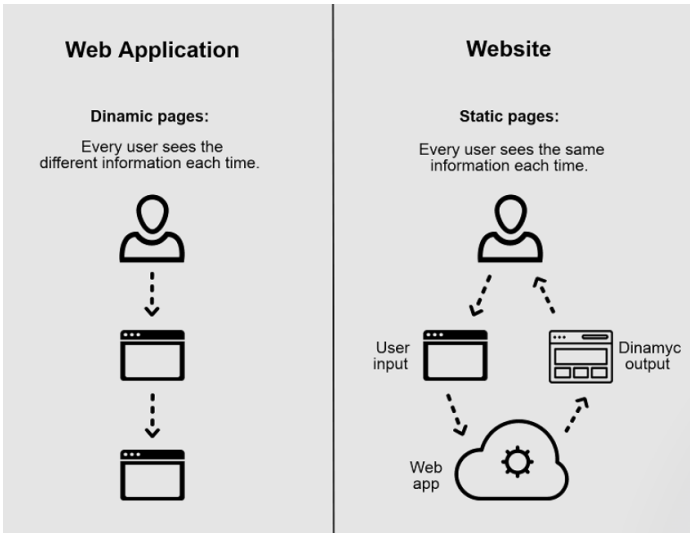


Рисунок 1.7 – Різниця між веб-сайтом та веб-додатком з точки зору користувача

Прикладом веб-сайту може бути веб-сторінка ресторану, де можна переглянути меню, години роботи тощо.

1.4 Огляд основних засобів створення веб-додатків

Сьогодні сучасні технології дозволяють використовувати величезну кількість різноманітних засобів для створення веб-застосунків різної складності. Але існує чотири базові інструменти без яких неможливе створення жодного веб-додатку та навіть звичайного веб-сайту.

1.4.1 HTML

HTML (мова гіпертекстової розмітки) – це код, який використовується для структурування веб-сторінки та її вмісту. Наприклад, контент може бути структурований як набір абзаців, список маркованих пунктів або з використанням зображень і таблиць даних. HTML – це мова розмітки, яка визначає структуру вашого контенту[14]. HTML складається з ряду елементів, які використовуються для включення або перенесення різних частин контенту, щоб він відображався або поводився певним чином. Приклад найпростішого HTML коду можна побачити нижче (рис. 1.8). Теги також містити різні атрибути, які надають елементу більше уніфікованих характеристик(рис. 1.9).

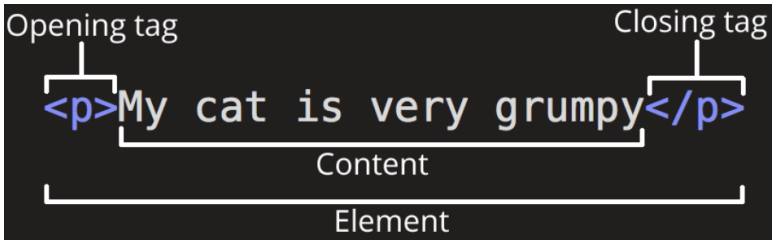


Рисунок 1.8 – Структура HTML елементу

Рисунок 1.9 – Атрибут HTML тегу

Навколишні теги можуть зробити слово або зображення гіперпосиланням, можуть виділяти слова курсивом, тощо[14].

1.4.2 CSS

CSS означає каскадні таблиці стилів. Це мова таблиць стилів, яка використовується для опису зовнішнього вигляду та форматування документа, написаного мовою розмітки. Він надає додаткову функцію для HTML[15]. Зазвичай він використовується з HTML для зміни стилю веб-сторінок та інтерфейсів користувача. Його також можна використовувати з будь-якими документами XML, включаючи звичайний XML, SVG і XUL. CSS використовується разом з HTML і JavaScript на більшості веб-сайтів для створення інтерфейсів для веб-додатків і багатьох мобільних додатків.

За допомогою CSS можна додати новий вигляд до своїх старих документів HTML або повністю змінити зовнішній вигляд свого веб-сайту, запровадивши лише кілька змін до коду CSS. До появи CSS такі теги, як шрифт, колір, стиль фону, вирівнювання елементів, межа та розмір, повинні були повторюватися на кожній веб-сторінці[15]. Це був дуже тривалий процес. Визначення стилів CSS зберігаються у зовнішніх файлах, тому можна змінити весь веб-сайт, створивши лише один файл. Синтаксис CSS є дуже простим та інтуїтивно зрозумілим для розробника (рис. 1.10).

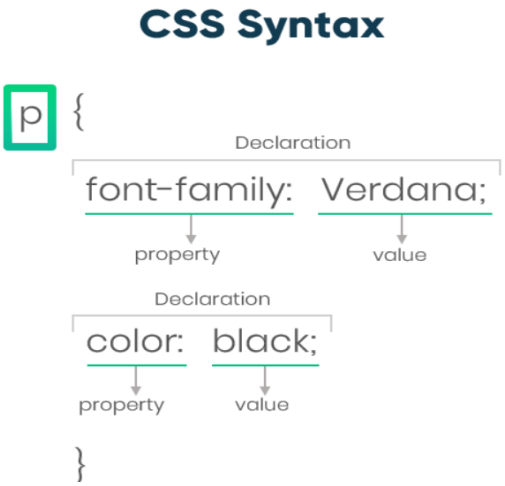


Рисунок 1.10 – Синтаксис CSS коду

CSS надає більш розширені атрибути, ніж звичайний HTML, тому зовнішній вигляд веб-сайту можна зробити набагато яскравішим.

1.4.3 UI/UX Дизайн

Дослівно “UI” в дизайні інтерфейсу означає “інтерфейс користувача(User Interface)”. Інтерфейс користувача — це графічне представлення програми[16]. Він складається з кнопок, на які натискають користувачі, тексту, який вони читають, зображень, повзунків, полів введення тексту та інших елементів, з якими взаємодіє користувач. Дизайн користувача включає в себе макет екрана, переходи, анімацію інтерфейсу і кожну мікровзаємодію(рис. 1.11).

“UX” означає “досвід користувача(User Experience)”. Досвід користувача у програмі визначається тим, як він взаємодіє з додатком[16]. Чи є досвід плавним та інтуїтивно зрозумілим чи незграбним та заплутаним, чи навігація додатку виглядає логічною чи ні. Користувальницький досвід визначається тим, наскільки легко або складно взаємодіяти з елементами інтерфейсу користувача(рис. 1.12).

Рисунок 1.11 – Приклад UI дизайну

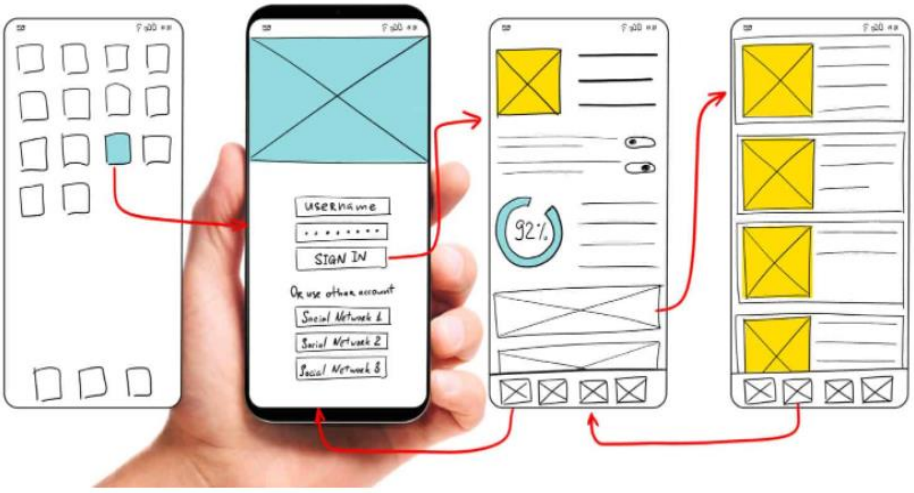


Рисунок 1.12 – Приклад UX дизайну

Якщо підсумувати, то можна сказати, що UX визначає як саме буде працювати дизайн, чи буде він зручним для користувача, у той час як UI вирішує питання як інтерфейс користувача буде вилядати. Частіше за все, ці два поняття непохитно пов’язані один з одним.

1.4.4 CMS

Система керування контентом, часто скорочено CMS, є програмним забезпеченням, яке допомагає користувачам створювати, керувати та змінювати контент на веб-сайті без необхідності спеціальних технічних знань. Простіше кажучи, система керування контентом – це інструмент, який допомагає створити веб-сайт без необхідності писати весь код із нуля[17]. Крім веб-сайтів, також існують системи керування контентом для інших функцій, таких як керування документами наприклад.

На технічному рівні система управління контентом складається з двох основних частин[17]:

- Додаток для керування контентом (CMA) – це та частина, яка дозволяє фактично додавати та керувати контентом на створюваному сайті.

- Додаток доставки контенту (CDA) – це внутрішній, закулісний процес, який приймає контент, що введений в CMA, правильно зберігає його та робить видимим для усіх відвідувачів сайту, що розробляється.

Разом ці дві системи полегшують обслуговування будь-якого сайту.

Найбільш популярними CMS системами на сьогоднішній день є[17]:

- WordPress(рис. 1.13);
- Joomla;
- Drupal;
- Magento;
- Squarespace;
- Wix;
- TYPO3;

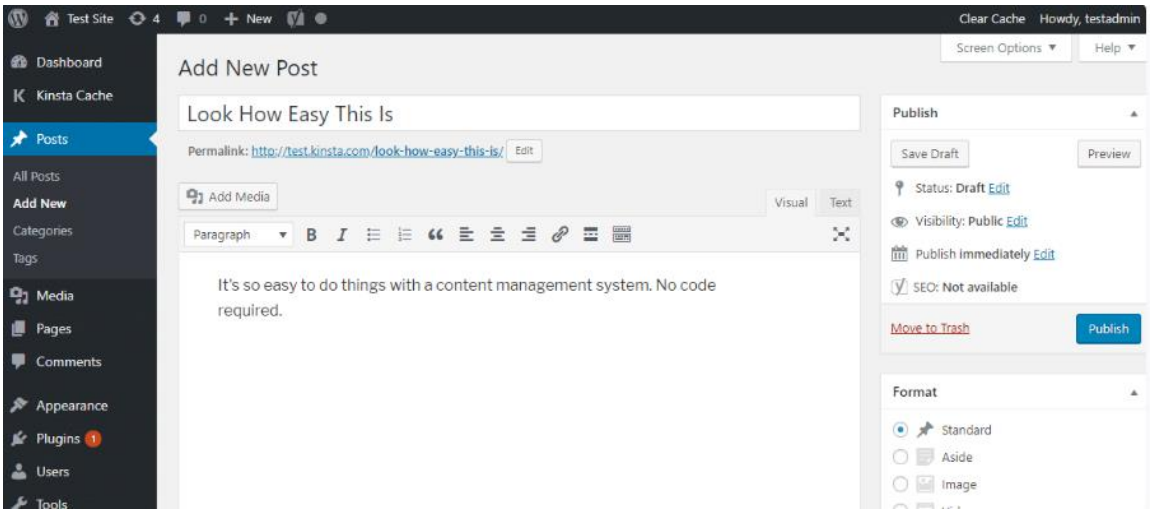


Рисунок 1.13 – Додавання нового запису на веб-сайт через CMS WordPress

Замість створення власної системи для розробки веб-сторінок, зберігання зображень та інших функцій система управління контентом обробляє всю цю базову інфраструктуру самостійно.

1.5 Проблеми створення та підтримки веб-додатків

Під час процесу створення, а потім підтримки, веб-додатків виникає певний ряд проблем, з якими стикається розробник.

Однією з таких проблем є неправильне планування розробки проекту. Чітке визначення цілей та вимог, правильне планування часу та ресурсів роботи – запорука створення успішного веб-додатку. Багато інших проблем, такі як продуктивність та швидкість додатків, можна майже повністю вирішити, зробивши обдуманий вибір на етапі планування[18].

Не менш важливою проблемою є вибір правильного стеку для розробки. Вони є комбінацією мов програмування, фреймворків, серверів, програмного забезпечення та інших інструментів, використовуваних розробниками. По суті це набір технологій для розробки веб-додатків або мобільних додатків[18]. Обраний стек технологій завжди має відповідати тим вимогам, які мають змогу вирішити поставлене завдання. Наприклад не потрібно обирати складний технічний стек для простого веб-додатка або технічний стек, який допоможе оптимізувати масштабованість, коли база користувача має сталий розмір.

Важливим є також правильний та відповідальний підхід до створення UI/UX дизайну. Адже якщо додаток складний у використанні, має велику кількість заплутаних навігаційних елементів, або застарілий дизайн інтерфейсу користувача, то малоймовірно, що користувач повернеться повторно до такого продукту або порекомендує його друзям.

При створенні веб-додатку також важливо акцентувати свою увагу на тому, що різна кількість користувачів буде створювати різну навантажуваність сторінки. Потрібно структурувати розробку таким чином, щоб при варіативному навантаженні веб-додатку, його швидкість роботи не змінювалась або змінювалась незначно.

Проблема масштабованості пов'язана з бажанням розвивати додаток з плином часу. При потребі створити застосунок сьогодні, необхідно знати якнайбільше про те, що потрібно від нього в майбутньому. Додаток може включати досить мізерний контент при першому запуску, але через рік або два розробник може планувати докладний, великий і багатий контентом продукт. Ось тут і з'являється масштабованість. Масштабованість - це проектування програми одразу для включення нових можливостей і функцій у майбутньому. Тому дуже важливо звертати на це увагу при розробці власного програмного продукту.

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі було досліджено історію створення веб-додатків ,їх види та засоби розробки. Проведено аналіз засобів створення веб-застосунків з використанням різноманітних веб-технологій, розглянуто теоретичні засади роботи веб-серверу та всесвітньої мережі Інтернет, виявлено основні проблеми при створенні веб-додатків. Завдяки проведеним вищезазначеним положенням, можна зробити наступні висновки, щодо веб-додатку, який буде розроблено:

- Веб-додаток потрібен мати чіткий, правильно розроблений UI та UX дизайн, для запобігання створення негативного користувацького досвіду;
- Веб-додаток має бути розроблений на правильному та найбільш доречному стеку технологій, для запобігання утворення різноманітних помилок;
- Веб-додаток має бути розроблений з використання динамічних веб-технологій;
- Веб-додаток має надавати користувачу можливість зареєструватися та зробити замовлення;
- Додаток має містити у собі базу даних для наявних товарів та зареєстрованих користувачів;

З вищезазначений висновків випливає необхідність створення веб-додатку, який надаватиме можливість переглянути та придбати різні товари концелярського призначення. Вибір найкраще пасуючих технології розробки такого додатку будуть розглянуті у наступному розділі.

РОЗДІЛ 2
ВИБІР ТЕХНОЛОГІЇ РОЗРОБКИ ТА ЙОГО ОБГРУНТУВАННЯ

2.1 Архітектура веб-додатку

В залежності від того, як логіка програми розподіляється між клієнтською та серверною сторонами, можна виділити кілька типів архітектури веб-застосунків. Найпоширеніші архітектури веб-додатків на сьогоднішній день це[19]:

1. Односторінкова архітектура
2. Багатосторінкова архітектура
3. Архітектура мікросервісів
4. Безсерверна архітектура
5. Прогресивна архітектура

На ринку розробки мають місце також і гібриди вещезгаданих архітектур.

2.1.1 Односторінкова архітектура

Односторінкова архітектура забезпечує створення додатку типу SPA - це веб-сайт або веб-додаток, який завантажує всю необхідну інформацію під час входу на сторінку. Односторінкова архітектура має одну істотну перевагу - вона

забезпечує приголомшливий інтерфейс користувача, оскільки користувачі не відчувають перезавантаження веб-сторінок. Односторінкові веб-додатки часто розробляються з використанням фреймворків JavaScript, таких як Angular, React та інших[19]. Даний тип архітектури має свої відповідні плюси та мінуси. До плюсів можна віднести швидкодію, гнучкість та відносну простоту розробки.

Мінусами у свою чергу є проблеми з оптимізацією, навантаження на браузер, необхідність підтримки JavaScript на стороні клієнта, поганий захист даних.

2.1.2 Багатосторінкова архітектура

Багатосторінкова архітектура вважається більш класичною. При її використанні кожна сторінка додатку надсилає запит на сервер та повністю оновлює всі дані, навіть якщо ці дані невеликі[20]. Таким чином витрачається продуктивність на відображення тих самих елементів. Відповідно це впливає на швидкість та продуктивність. Багато розробників, щоб підвищити швидкість і зменшити навантаження, використовують JavaScript/jQuery[20]. Наприклад, оновлення товарів без перезавантаження сторінки при використанні фільтрів в інтернет магазині. Плюсами даної архітектури є легка оптимізація, потребує меншого стеку технологій при розробці, має багато рішень. Мінуси – складність розробки мобільних додатків, важко розділити фронтенд та бекенд, адже вони дуже тісно взаємодіють між собою.

2.1.3 Безсерверна архітектура

Безсерверна архітектура – це спосіб створення та запуску додатків та сервісів без необхідності керування інфраструктурою. Програма буде працювати на серверах, але керування цими серверами AWS повністю бере на себе[21]. Розробнику не потрібно займатися виділенням ресурсів,

масштабуванням та обслуговуванням серверів для запуску програм, баз даних та систем зберігання даних. Зазвичай у безсерверних додатках використовуються функції у вигляді сервісу - керовані рішення для короткочасних обчислень без збереження стану [21], а також сервіси обміну повідомленнями, зберігання файлів, що повністю керуються оператором, баз даних, потокової передачі та аутентифікації - програми, що постають у вигляді сервісу. До плюсів даної архітектури можна віднести нижчі експлуатаційні трудовитрати, швидке розгортання завдяки використанню BaaS, практично безмежна масштабованість. Мінуси – втрата контролю над додатком, непередбачувані наслідки оновлень, складна процедура тестування.

2.1.4 Мікросерверна архітектура

Мікросервісна архітектура це повна протилежність монолітної архітектури[22]. Використовуючи такий підхід замість однієї великої програми, створюється набір невеликих слабозв'язаних і легко замінюваних модулів, які взаємодіють один з одним. Одна з головних переваг мікросервісної архітектури - можливість використовувати найкращий технічний стек для кожного окремого завдання. Такий підхід дозволяє розробникам створювати веб-додаток із набору невеликих сервісів. Розробники створюють та розгортають кожен компонент окремо. Перевагами даної архітектури є модульність, невеликий час тестування кожного окремого модулю, відносно малий час деплою, можливість масштабування. Недоліки - процес тестування зібраного воєдино ПЗ складний, тому що мікросервіси базуються на окремих доменах, міграція монолітної архітектури в мікросервіс може обійтися дуже дорого, керувати версіями ПЗ стає складніше. Наглядне представлення вигляду мікросерверної архітектури наведено нижче (рис. 2.1).

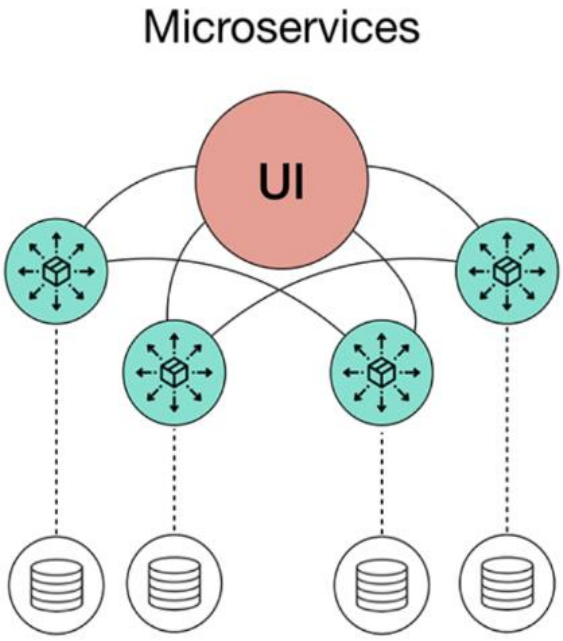


Рисунок 2.1 – Мікросерверна архітектура веб-додатку

Архітектура мікросервісів є вигідною для великих і складних проектів, оскільки кожен сервіс може бути змінений без шкоди для інших блоків[22].

2.2 Вибір технологій розробки частини користувача

Одним з найважливіших етапів створення веб-додатку є вибір найбільш зручної технології для розробки інтерфейсу користувача, так званого фронтенду. На сьогоднішній день існує купа технологій та фреймворків для його реалізації (рис. 2.2).

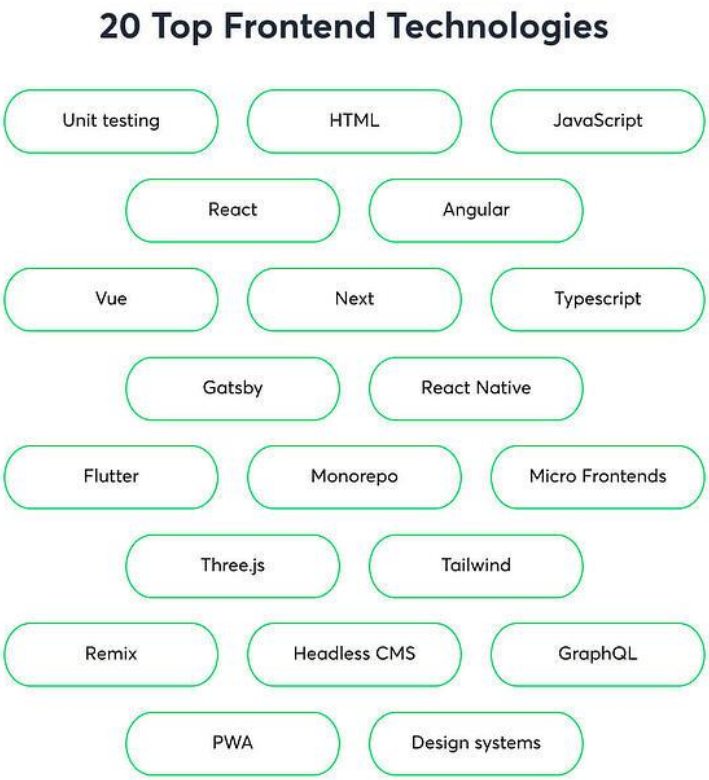


Рисунок 2.2 – Топ 20 технологій для розробки інтерфейсу користувача

Але звичайно базовими компонентами для розробки фронтенду залишається HTML та CSS, без їх використання неможливо зробити веб-додаток.

2.2.1 Javascript + jQuery

Почнемо з того, що JavaScript існує вже чверть століття[23]. Ця мова програмування відома тим, що зробила справжню революцію в мережі Інтернет завдяки своїм динамічним можливостям.

До появи JavaScript мережа загалом була статичною. Веб-сторінка являла собою просто блок тексту. Потім з'явився JavaScript, який забезпечив інтерактивність, таку як прокручування, натискання та багато іншого. Зараз JavaScript використовується на більшості веб-сайтів [23]. Дана мова програмування має величезну кількість переваг[24] :

- Швидкість. JavaScript дуже швидкий, тому що зазвичай запускається безпосередньо у браузері клієнта. Крім того, всі основні браузери підтримують компіляцію для JavaScript, що означає відсутність необхідності компілювати код перед його запуском.
- Простота. Синтаксис JavaScript має багато спільного з синтаксисисом Java і його відносно легко освоїти в порівнянні з іншими популярними мовами, такими як C++.
- Популярність - JavaScript всюди в Інтернеті. Останнім часом кількість проектів, що використовують JavaScript, неабияк зросла і очікується, що популярність, яку він набув останніми роками, буде тільки зростати.
- Взаємодія. JavaScript можна вставити на будь-яку веб-сторінку та використовувати в багатьох різних програмах завдяки підтримці інших мов, таких як Pearl та PHP.
- Навантаження на сервер. JavaScript виконується на стороні клієнта, тому в цілому він знижує навантаження на сервери, а простим програмам сервер може взагалі не знадобитися.
- Багаті інтерфейси. JavaScript можна використовувати для створення різноманітних функцій, які значно покращують інтерфейс користувача і зручність роботи з сайтом.
- Розширена функціональність. Розробники можуть розширити функціональність веб-сторінок, написавши фрагменти коду JavaScript.
- Універсальність. Існує багато способів розробити всю програму від початку до кінця, використовуючи тільки JavaScript.
- Оновлення. З моменту появи ECMAScript 5 (специфікація сценаріїв, на яку спирається JavaScript) ECMA International щорічно оновлює JavaScript.

До мінусів даної мови програмування можна віднести безпеку, оскільки код JavaScript виконується на стороні клієнта, помилки та упущення іноді можуть бути використані в зловмисних цілях, підтримка браузера - різні браузери іноді по-різному інтерпретують код JavaScript[24].

jQuery - це легка бібліотека JavaScript, яка має кредо - "пиши менше, роби більше"[25]. Ціль jQuery - спростити використання JavaScript на будь-якому веб-сайті. jQuery бере на себе безліч завдань, для виконання яких потрібно багато рядків коду JavaScript, і обертає їх у методи, які можна викликати за допомогою одного рядка коду. jQuery також спрощує багато складних речей з JavaScript, такі як виклики AJAX і маніпулювання DOM[25]. Існує безліч інших бібліотек JavaScript, але jQuery є найпопулярнішою(рис. 2.3), а також постійно розширюється. Найбільшими перевагами даної бібліотеки є: простота синтаксису, популярність, лігкість створення анімації, швидкість завантаження сторінок, кросбраузерність.

Top in JavaScript Libraries and Functions · Week beginning Dec 18th 2017

Name	10k	100k	Million	Entire Web
jQuery	↓6,636	↓67,707	↓699,998	↑73,007,390
Facebook for Websites	↓4,301	↓31,461	↓240,741	↑8,075,834
Facebook SDK	↓3,955	↓28,499	↓207,153	↑5,655,256
Google Hosted Libraries	↓3,553	↓28,206	↓183,111	↓11,117,932
html5shiv	↓3,203	↓26,016	↓196,088	↓8,786,122
Modernizr	↓2,823	↓21,506	↓145,635	↑6,940,562

Рисунок 2.3 – Топ найпопулярніших бібліотек JavaScript

З мінусів можна виділити лише складність використання при необхідності отримки даних з серверу через API[25].

2.2.2 Angular

Angular – це платформа та фреймворк для створення односторінкових клієнтських програм з використанням HTML та TypeScript. Angular написано

на TypeScript. Він реалізує основні та додаткові функції у вигляді набору бібліотек TypeScript, які можуть бути імпортовані до веб-програм[26]. Основними будівельними блоками фреймворку Angular є компоненти Angular, організовані в NgModules. NgModules збирають пов'язаний код у функціональні набори; Додаток Angular визначається набором NgModules. Програма завжди має принаймні кореневий модуль, що забезпечує початкове завантаження, і зазвичай має набагато більше функціональних модулів[26]. Фреймворк Angular спрощує розробку веб-програм. Поєднуючи використання залежностей, декларативні шаблони, комплексні інструменти та інтегровані передові практики, він вирішує практично всі проблеми при створенні веб-додатка.

До плюсів даного фреймворку можна віднести[27]:

- Реалізація архітектури MVC. Архітектура Модель-Представлення-Контролер не тільки підвищує цінність платформи при створенні клієнтської програми, але й закладає основу для інших функцій.
- Удосконалена архітектура проектування. Деякі великі веб-програми містять багато компонентів. Angular полегшує управління цими компонентами.
- Модулі. Модуль - це механізм, який групує пов'язані директиви, компоненти, канали та служби таким чином, щоб їх можна було поєднувати з іншими модулями для створення програми.
- Служби та впровадження залежностей (DI). Службі або компоненту іноді можуть знадобитися інші залежні служби для виконання завдання. Для виконання цих залежностей використається шаблон проектування Dependency Injection.
- Директиви користувача. Директиви користувача покращують функціональність HTML і підходять для динамічних клієнтських додатків.

Відповідно мінуси Angular [27] :

- Обмежені можливості SEO. Основним недоліком використання Angular є обмежені можливості SEO та погана доступність для пошукових роботів.
- Angular багатослівний і складний. Найбільша скарга на даний фреймворк це багатослівність цього інструменту.
- Складне навчання. Розробникам із знанням JavaScript дуже складно адаптуватися до використання Angular. Це пов'язано з тим, що коло тем та аспектів, які необхідно охопити, досить велике.
- У документації немає подробиць. Хоча командний рядок є дуже корисним для розробників Angular, неможливо знайти достатньо інформації в їх офіційній документації.

2.2.3 ASP.NET

ASP або Active Server Pages, розроблені Microsoft, - це технологія, що спрощує розробку інтерактивних веб-застосунків і створення багатофункціональних і динамічних веб-сайтів. Однією з основних переваг цієї технології є те, що вона може використовувати сценарії як на стороні клієнта, так і сервері. Щоб користуватися перевагами даної платформи, розробник повинен бути повністю знайомий із C#. ASP можна використовувати для керування вмістом будь-якої сторінки [28]. ASP.NET розширює платформу .NET(рис. 2.4) інструментами та бібліотеками, спеціально призначеними для створення веб-додатків. Як і згадані вище технології, ASP.NET має свої переваги та недоліки. Кажучи про переваги, це[28]:

- Можливість розділення роботи. ASP.NET слідує архітектурі MVC, яка дозволяє розділяти введення, обробку та виведення програми.

- Скорочення час кодування. Технологія фреймворку дуже допомагає скоротити час написання коду, особливо при створенні великих програм. Існують різні види код-рев'ю, тому майже немає шансів написати поганий код.
- Складається з деяких готових функцій. ASP.NET поставляється з такими функціями, як своєчасна компіляція, раннє зв'язування, вбудована оптимізація та служби кешування.
- Набір інструментів світового класу. Фреймворк поставляється з надзвичайно багатим набором інструментів через інтегроване середовище розробки Visual Studio.
- Налаштовуваність та розширюваність. Добре продумана архітектура фреймворку надзвичайно допомагає розробникам.
- Безпека. За допомогою даного фреймворку можливо розробляти безпечні програми за допомогою вбудованої автентифікації Windows та функцій конфігурації для кожної програми.
- Кросплатформеність.

Щодо мінусів ASP.NET, то це безперечно коштовність, у порівнянні з альтернативами з відкритим вихідним кодом, ASP.NET коштує дорого, оскільки у є витрати на ліцензії та інші продукти.

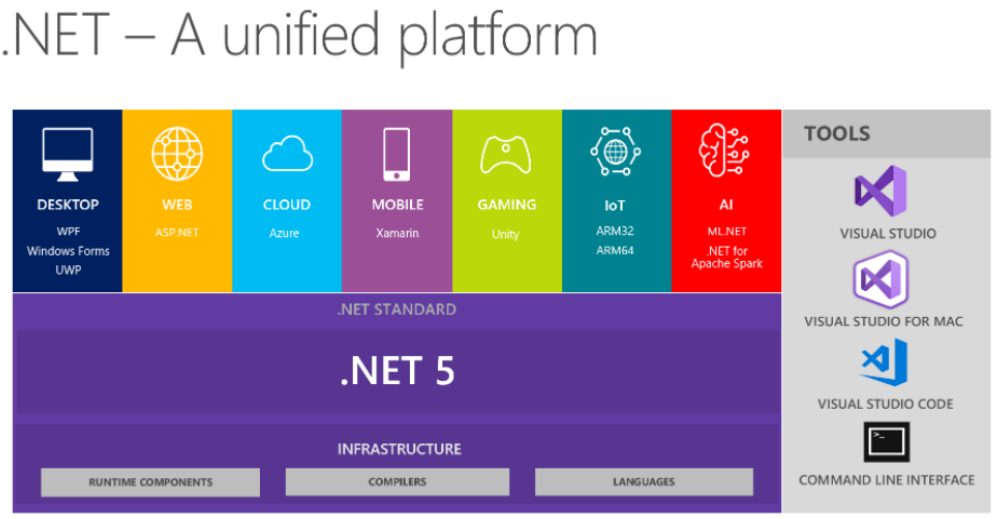


Рисунок 2.4 – Структура платформи .NET

Також важливими недоліками ASP.NET є погана документація фреймворку, внесені зміни до однієї програми можуть не запрацювати в іншій, також перенесення проекту з одного сервера на інший буде коштувати чималих грошей[28].

2.3 Вибір технологій розробки серверної частини

Важливим етапом у створенні веб-додатку є обрання правильного стеку технологій для розробки серверної частини. Адже саме від цього буде залежати швидкодія відповідей серверу додатка та ступінь складності його підтримки та адміністрування.

2.3.1 Python

Python відомий своїм простим синтаксисом та шорткодом. Це, у поєднанні з тим фактом, що існує безліч посібників щодо його використання, робить його досить простим у освоєнні. Більш того, Python, будучи надзвичайно добре спроектованим та універсальним, є незалежною від платформи мовою і може використовуватися в різних операційних системах. Таким чином, Python є ідеальною бекенд-мовою завдяки своїй простоті та узгодженості, де розробники можуть створювати надійні системи з великим набором бібліотек, що належать машинному навчанню, Keras, TensorFlow та Scikit-learn. Великий набір бібліотек та фреймворків Python може бути надзвичайно корисним та економити час, що призводить до скорочення часу роботи та підвищення продуктивності[29].

Численні ресурси Python представлені в багатьох формах, включаючи широкий спектр фреймворків для веб-додатків, таких як Django, Flask та інших. Крім того, найбільш поширеними варіантами використання Python є веб-розробка, штучний інтелект, машинне навчання та його підполя, наука про дані,

великі дані, Інтернет речей, вбудовані системи та багато іншого[29]. Python зменшує навантаження на код приблизно в 4 рази, значно зменшуючи кількість рядків коду(рис. 2.5), а також синтаксис мови є дуже зрозумілим та нагадує звичайну англійську(рис. 2.6).

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Рисунок 2.5 – Виведення привітання мовою Java

```
print("Hello, World")
```

Рисунок 2.6 – Виведення привітання мовою Python

Кажучи про Python, він також підтримує динамічне написання коду, легкий у вивченні та швидкодіючий.

Станом на 2022 рік, найпопулярнішим Python фреймворком для веб розробки звичайно є Django(рис. 2.7).

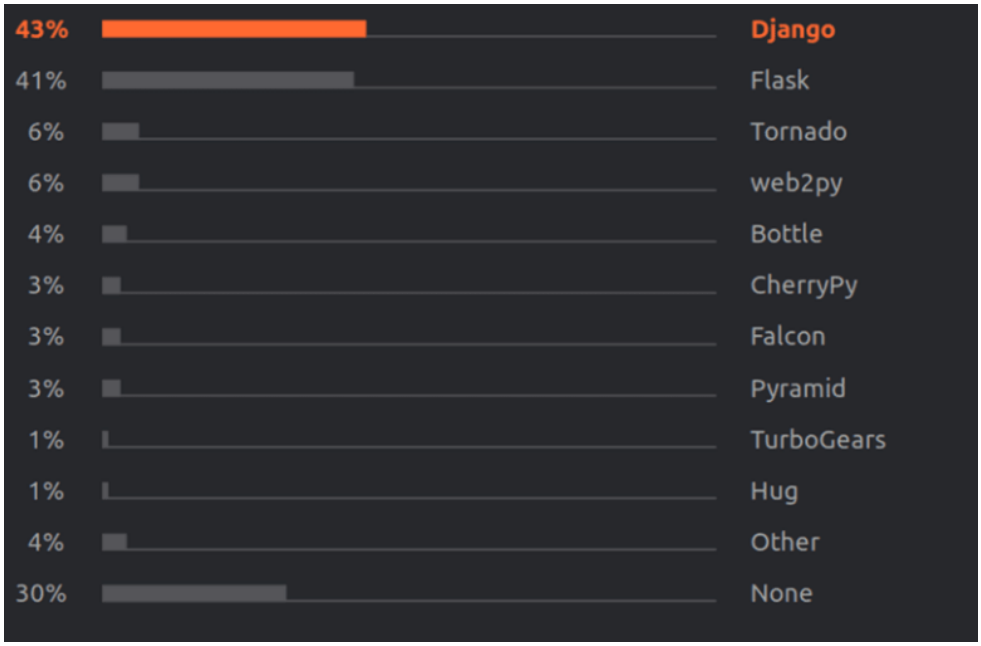


Рисунок 2.7 – Рейтинг Python веб фреймворків від JetBrains

Django – це високорівнева веб-інфраструктура Python, яка дозволяє швидко розробляти безпечні та зручні в обслуговуванні веб-додатки. Створений досвідченими розробниками, Django бере на себе велику частину

клопоту веб-розробки, тому можна зосередитися на написанні своєї програми, не винаходячи велосипед. Він безкоштовний і з відкритим вихідним кодом, має процвітаючу та активну спільноту, відмінну документацію та безліч варіантів безкоштовної та платної підтримки[30]. Django допомагає писати програмне забезпечення, яке є повним, універсальним, безпечним, має надзвичайно гарну масштабованість, ремонтопридатність та портативність[30]. Також цей фреймворк дуже простий(рис. 2.8) у використанні та навіть має власний вбудований шаблонізатор під назвою Jinja.

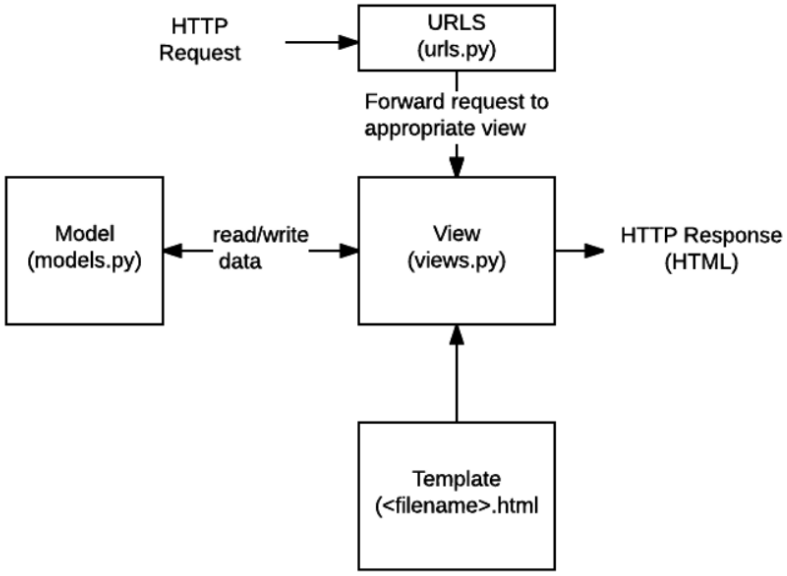


Рисунок 2.8 – Принцип роботи Django фреймворку

Завдяки Django можна створювати величезну кількість функцій для розширення веб-додатку, серед них[30]:

- Форми: HTML-форми використовуються для збору даних користувача та обробки їх на сервері.
- Аутентифікація користувачів та дозволи: Django включає надійну систему аутентифікації та дозволів користувачів, створену з урахуванням вимог безпеки.
- Кешування: Django забезпечує гнучке кешування, так що ви можете зберегти всю або частину сторінки, щоб вона не відображалася повторно, за винятком випадків, коли це необхідно.

- Сайт адміністрування: Сайт адміністрування Django включається за умовчанням при створенні програми з використанням базового скелета. Це спрощує надання адміністративної сторінки для адміністраторів сайту.
- Серіалізація даних: Django спрощує серіалізацію та обслуговування ваших даних у вигляді XML або JSON. Це може бути корисно при створенні веб-служби.

2.3.2 Java

Java має сильну підтримку для веб-розробки та часто використовується на стороні сервера. Веб-програми Java зазвичай не запускаються безпосередньо на сервері, а виконуються всередині веб-контейнера. Контейнер надає середовище виконання для веб-додатків Java. Контейнер - це те ж саме, що JVM (віртуальна машина Java) для локальних запущених програм Java[31]. Популярними веб-фреймворками Java є GWT, JavaServer Faces, Struts та Spring framework[31].

Java дуже популярна мова програмування, тож вона має ряд своїх переваг для обрання її при веб розробці. Java добре масштабується. Коли обробка або введення-виведення збільшуються, можна легко додавати ресурси та перерозподіляти навантаження. Компоненти Java легко доступні, що спрощує масштабування великих веб-програм. Мова є гнучкою, це надає можливість робити менш інвазивне кодування для покращення масштабованості[32]. Також Java являє собою кросплатформову мову програмування, що тягне за собою раяд переваг при веб розробці.

Автоматичне керування пам'яттю Java є значною перевагою. Java виділяє пам'ять стека для кожного потоку. У купі зберігаються фактичні об'єкти, змінні стеки посилаються на них. Пам'ять купи одна тільки в кожній JVM, тому вона поділяється між потоками[32]. Проте сама купа складається з кількох частин,

що спрощують складання сміття Java. Розміри стека та купи залежать від JVM.

Однією з найбільших переваг Java це її мультипотоковість, що добоможе створити унікальну структуру для веб-додатку. Для даної мови програмування також існує багато середовищ розробки, навчальних посібників та розробників, до яких можна звернутися за порадою.

Станом на 2021 рік найбільш популярними Java фреймворками(рис. 2.9) для веб розробки є:

- Spring.
- JSF.
- Vaadin.
- GWT

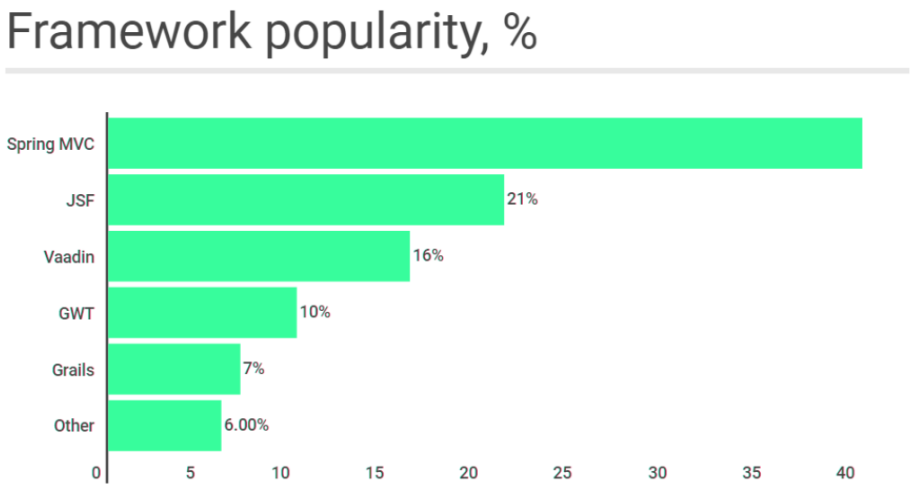


Рисунок 2.9 – Найпопулярніші фреймворки Java

Spring фреймворк надає комплексну модель програмування та конфігурації для сучасних корпоративних програм на основі Java на будь-якій платформі розгортання. Ключовим елементом Spring є інфраструктурна підтримка на рівні програм: Spring фокусується на «підключенні» корпоративних програм, щоб команди могли зосередитися на бізнес-логіці без непотрібних прив'язок до конкретних середовищ розгортання[33].

Особливості Spring фреймворку[33]:

- Чіткий поділ ролей.

- Потужна та проста конфігурація як фреймворку, так і класів додатків у вигляді JavaBeans.
- Адаптивність, ненав'язливість та гнучкість.
- Багаторазовий бізнес-код, немає потреби в дублюванні.
- Прив'язка, що налаштовується, і її перевірка.
- Відображення обробника і дозвіл перегляду.
- Гнучка передача моделі.
- Проста, але потужна бібліотека тегів JSP, відома як бібліотека тегів Spring, яка забезпечує підтримку таких функцій, як прив'язування даних та теми.
- Біни, чий життєвий цикл обмежений поточним запитом HTTP або сеансом HTTP.

2.3.3 C#

C# - це сучасна, об'єктно-орієнтована і типобезпечна мова програмування. C# дозволяє розробникам створювати безліч типів безпечних та надійних програм, що працюють у .NET. C# бере свій початок у сімействі мов C і буде відразу знайомf програмістам на C, C++, Java та JavaScript. C# - це об'єктно-орієнтована мова програмування, орієнтована на компоненти. C# надає мовні конструкції для безпосередньої підтримки цих концепцій, що робить C# природною мовою для створення та використання програмних компонентів. З моменту появи в C# були додані функції для підтримки нових робочих навантажень і нових методів проектування програмного забезпечення[34]. Програми на C# виконуються серед .NET, віртуальної системі виконання, званої CLR. CLR – це реалізація Microsoft загальномовної інфраструктури (CLI), міжнародного стандарту[34]. Інтерфейс командного рядка є основою для створення середовищ виконання та розробки, у яких мови та бібліотеки

працюють без проблем.

C# має кілька плюсів, тому він так часто використовується в більшості проектів розробки Windows.

По-перше, він добре інтегрується із Windows. Розробнику не потрібні спеціальні налаштування, щоб змусити програму C# працювати у середовищі Windows. Будь то веб-додаток, служба Windows або настільна програма, програми C# легко розгортаються в мережі. Доки цільовий сервер або робоча станція підтримує .NET, розгортання програми C# має бути плавним[35].

Далі, C# — одна з найпоширеніших мов, які вивчають програмісти. Він також має дуже близький Java синтаксис, тому зазвичай можна знайти розробника, який одночасно розуміє Java. Якщо сервер зламаний, зловмисник автоматично не зможе отримати доступу до вихідного коду[35].

Але у C# також є свої недоліки[35]:

- З ним набагато складніше працювати, тому що код повинен компілюватися щоразу, коли вноситься навіть незначна зміна. Це часто призводить до додавання нових помилок при тестуванні.
- Оскільки C# є частиною платформи .NET, сервер, на якому виконується програма, має бути Windows. Іншими словами, для виконання будь-якої програми .NET потрібна платформа Windows.
- Вам потрібний хостинг Windows для запуску .NET.
- Microsoft припиняє підтримку старих платформ .NET після кількох оновлень операційних систем.

Загалом C# є досить зручною мовою програмування та навіть входить у топ найпопулярніших мов для бекенд розробки(рис. 2.10).

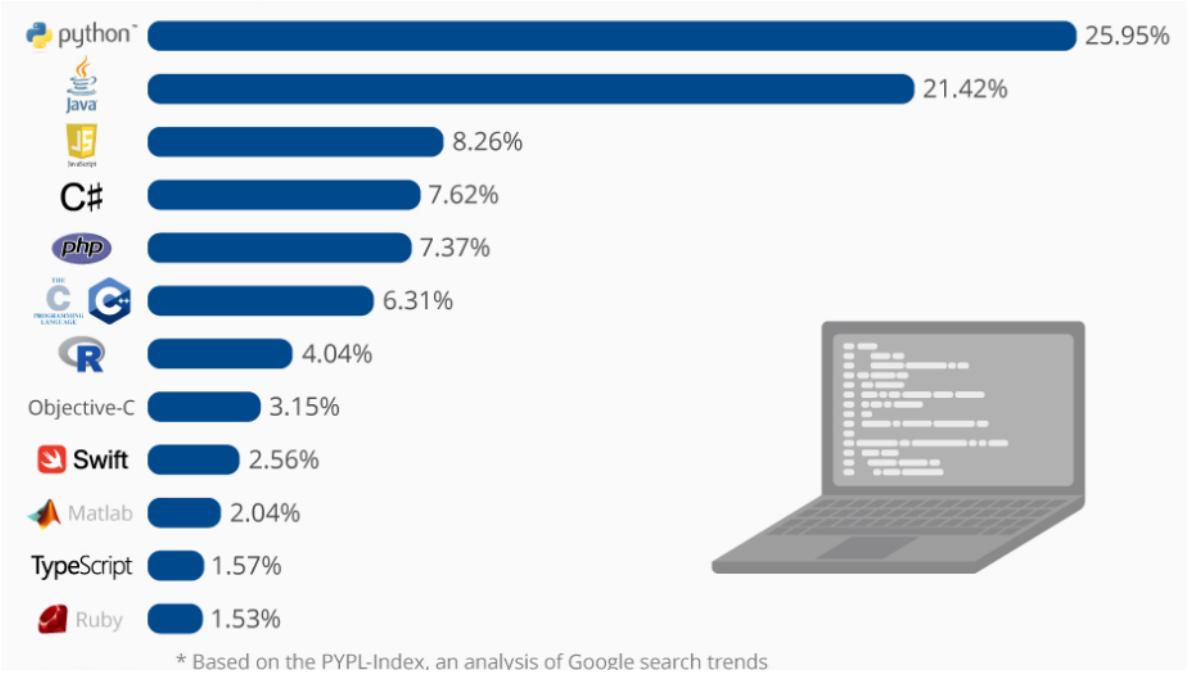


Рисунок 2.10 – Топ мов програмування для створення серверної частини веб-додатків

Нарешті, C# є мовою, що компілюється, а означає, що код, який зберігається на загальнодоступному сервері, знаходиться в двійковій формі.

ВИСНОВОК ДО РОЗДІЛУ 2

У даному розділі було розглянуто типи архітектури веб-додатків, а також різноманітні технології для створення серверної та клієнтської частини веб-застосунку.

Визначено найкращі випадки для застосування односторінкової, багатосторінкової, безсерверної та мікросерверної архітектур.

Визначено найпопулярніші мови програмування для фронтенд та бекенд розробки на сьогоднішній день.

Після проведення аналізу всіх переваг та недоліків вищезазначених компонентів було обрано наступний стек технологій для розробки власного веб-додатку:

- Гібрид багатосторінкової та односторікової архітектур;
- Мова програмування JavaScript та її бібліотека jQuery, мова розмітки HTML та каскадна таблиця стилів CSS для розробки клієнтської частини додатку (фронтенду);
- Мова програмування Python та її фреймворк Django для розробки серверної частини додатку (бекенду) ;
- Вбудована база даних sqllite3 у фреймворк Django.

РОЗДІЛ 3
ДЕТАЛІ РОЗРОБКИ СИСТЕМИ

Згідно з дослідженнями, проведеними у попередніх розділах, а також обраному стеку технологій, переходимо до безпосереньо розробки веб-застосунку. Щоб створити програмний продукт, що буде відповідати всім вимогам, зазначеним у попередніх розділах, потрібно детально описати його основні складові компоненти та способи їх взаємодії.

3.1 Розробка програмних компонентів частини користувача

У даному підрозділі буде описано основі етапи створення користувацької частини веб-застосунку, а саме: розробка дизайну інтерфейсу користувача, створення веб сторінок та різноманітних функціій для їх коректної роботи та взаємодії

3.1.1 Розробка дизайну

Першим етапом у розробці веб-додатку є обмірковування та створення макетів дизайну інтерфейсу користувача, адже візуальна компонента є першою на що користувач звертає увагу при першому користуванні застосунком.

Для створення приємного UI-дизану потрібно зосередитись на потребах користувача, поставити себе на його місце. Інтерфейс не має бути перевантажений зайвою інформацією або елементами, має дотримуватись певної ієрархії та послідовності, користувач повинен інтуїтивно розуміти “на які кнопки натискати”. Важливо також використовувати одну палітру кольорів на кожній сторінці, щоб дотримуватися загальної тематики веб-застосунку.

Для створення макетів основних сторінок веб-додатку використовувалась програма Photoshop, адже вона надає велику базу гнучких інструментів для роботи з візуальним контентом.

Загалом розроблено 10 макетів для ілюстрування вигляду основних компонентів веб-здодатку, серед них сторінки навігаційного меню, лендінг сторінка, сторінка корзини, регістації та авторизації користувача.

На рисунку 3.1 зображено макет головної сторінки веб-додатку, яка буде першою вітати користувачів.

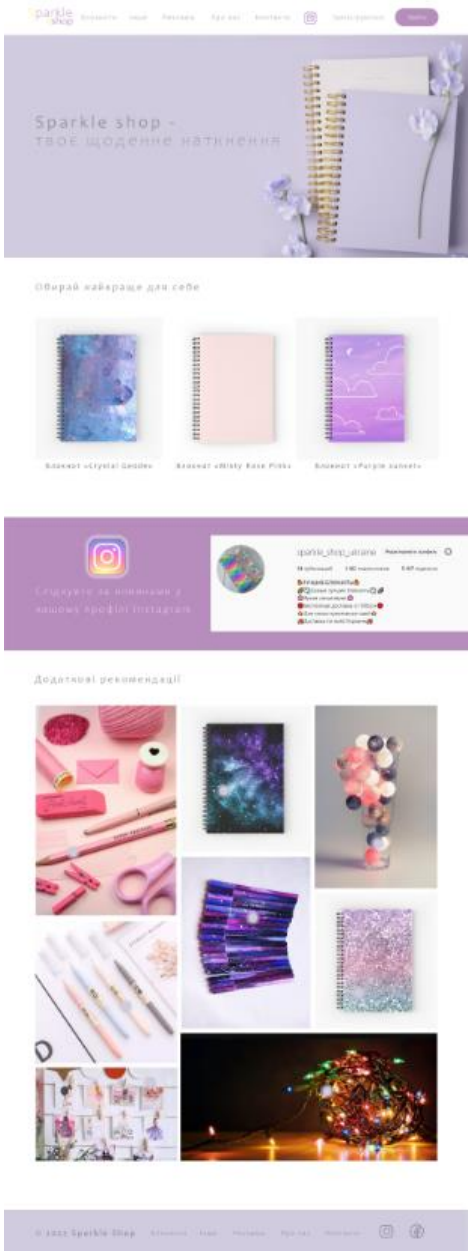


Рисунок 3.1 – Макет дизану головної сторінки веб-додатку

Важливо зазначити, що кожна компанія, веб-застосунок, тощо, має мати свій логотим, який у свою чергу зазвичай презентує їх назву. Тож даний інтернет-магазин вирішено назвати “Sparkle Shop”, далі ця назва часто буде зустрічатися на макетах дизайну інших сторінок.

Далі (рис. 3.2) зображено макет сторінки для відображення блокнотів, на якій можна побачити вигляд фільтрів, налаштовуючи які користувач зможе обирати потрібний йому товар.

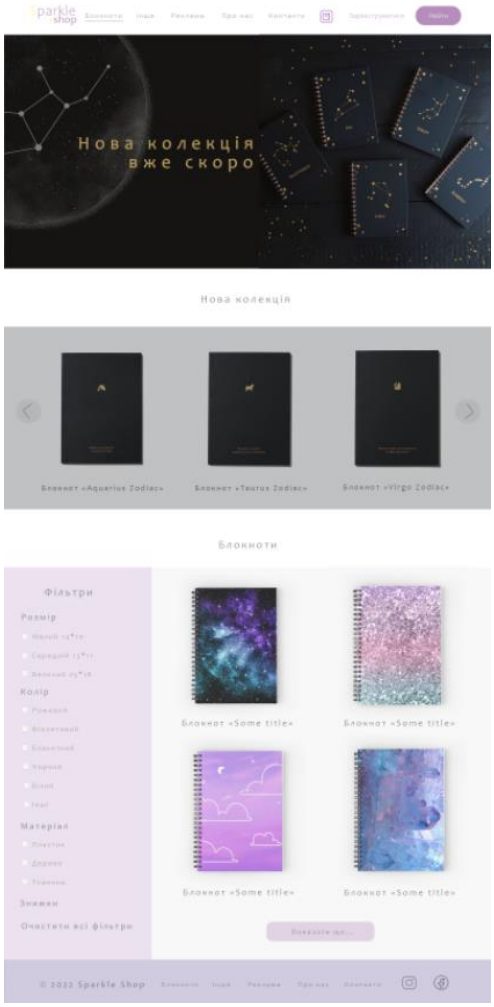


Рисунок 3.2 – Макет дизану сторінки “Блокноти” веб-додатку

Розглянемо наступний макет(рис. 3.3), який являє собою вигляд сторінки з іншими канцелярськими товарами, окрім блокнотів. Загалом, тут дотримано такого же принципу та концепції як і на попередній сторінці.

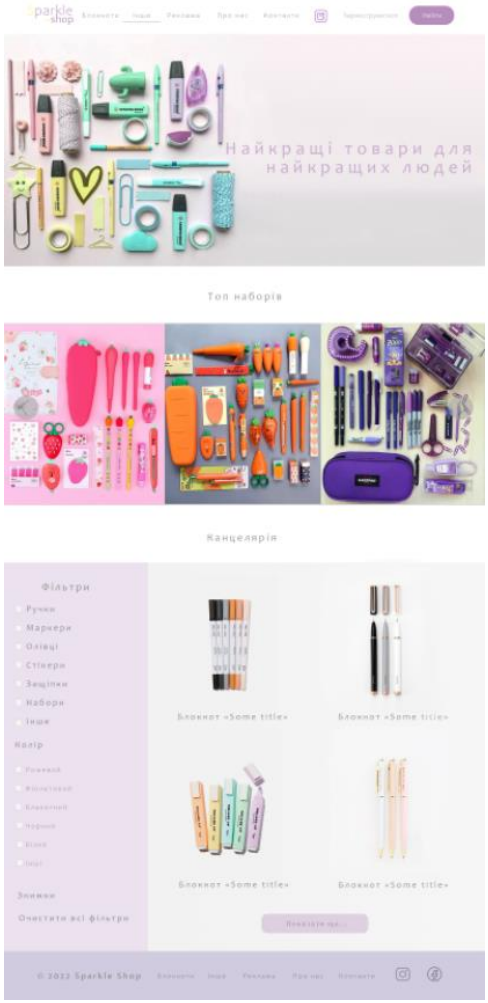


Рисунок 3.3 – Макет дизану сторінки “Інше” веб-додатку

Наступним є макет сторінки з рекламою інших магазинів(рис. 3.4), яку можна буде розміщувати саме тут. Відповідні секції з зображенням та описами магазинів будуть змінюватися від замовника. Працювати це буде наступним чином: власник інстаграм сторінки контактує з адміністратором даного веб-додатку через спеціальну поштову адресу, адміністратор висилає замовнику прайс на рекламу відповідно до терміну її перебування на сторінці веб-застосунку, якщо замовника влаштовують умови, він висилаю адміністратору фотографію для реклами, текст опису свого магазину та посилання на Інстаграм сторінку. Адміністратор у свою чергу розміщує вищезазначені дані у відповідній секції сторінки “Реклама”, замінюючи застарілу рекламу на нову.

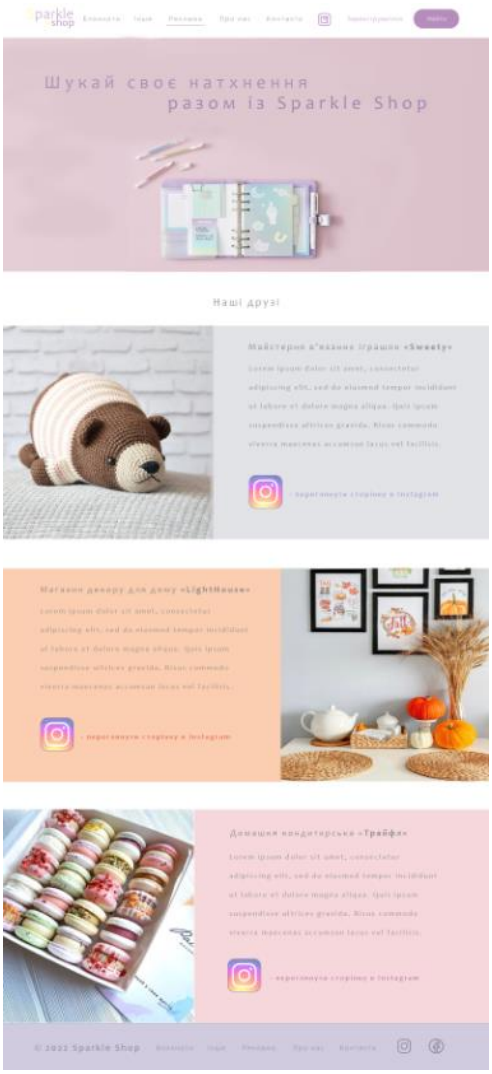


Рисунок 3.4 – Макет дизану сторінки “Реклама” веб-додатку

Далі зображено макет дизайн сторінки “Про нас”(рис. 3.5), яка є основним джерелом інформації про діяльність веб-застосунку для користувача, тобто основні принципи та мотивацію роботи.

Остання сторінка у навігаційному меню – це сторінка “Контакти”(рис. 3.6), за допомогою неї користувач може дізнатися як встановити контакт з адміністратором сайту через різноманітні способи “зв’язку”, у разі коли в нього виникли питання.

Рисунок 3.5 – Макет дизану сторінки “Про нас” веб-додатку

Рисунок 3.6 – Макет дизану сторінки “Контакти” веб-додатку

Наступним став макет сторінки корзини, у яку користувач буде додавати обрані товари. Корзина має два стани, перший – коли вона порожня(рис. 3.7), другий – коли наповнена(рис. 3.8), тому нижче представлено два макети дизайну, відповідно до кожного стану корзини.

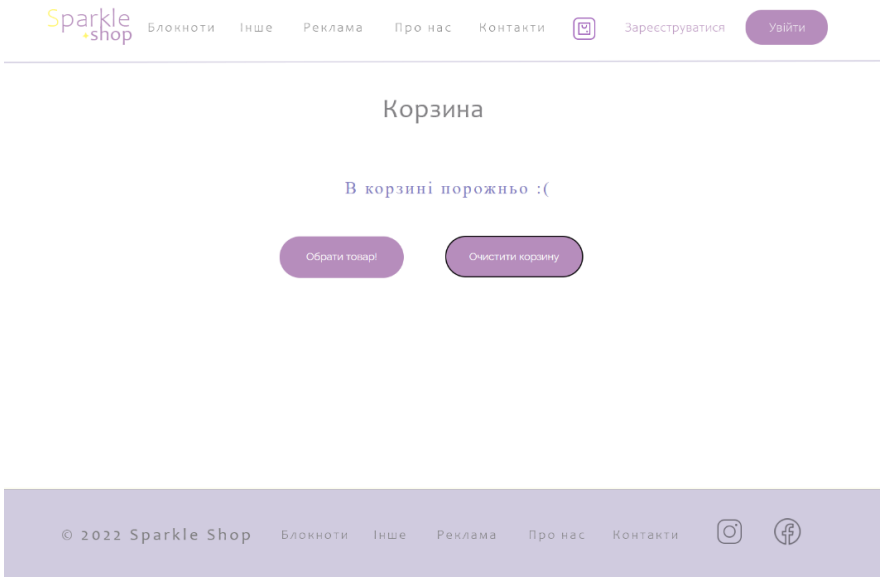


Рисунок 3.7 – Макет дизану сторінки “Корзина” веб-додатку, коли вона порожня

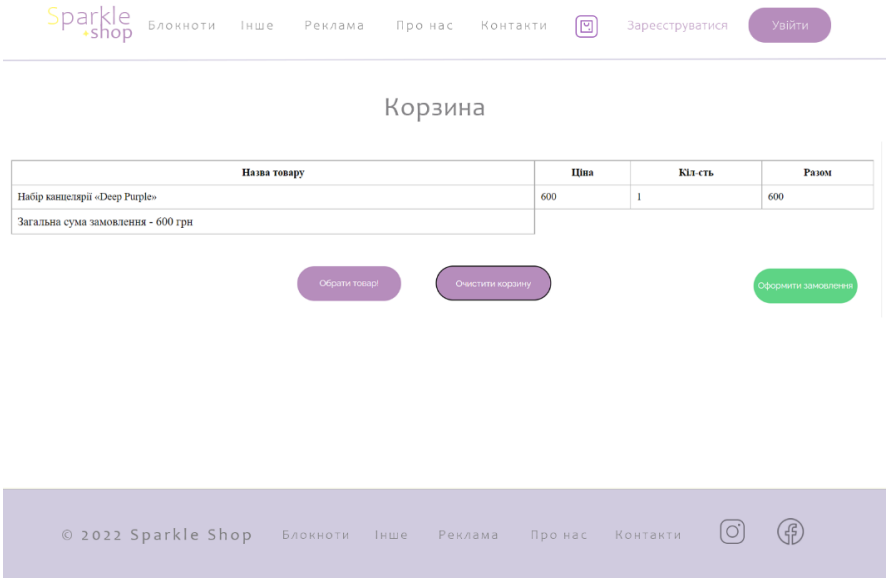


Рисунок 3.8 – Макет дизану сторінки “Корзина” веб-додатку, коли вона наповнена

Наступними макетами є дизайн сторінок реєстрації(рис. 3.9) та відповідно авторизаціїї(рис. 3.10) у веб-застосунку.

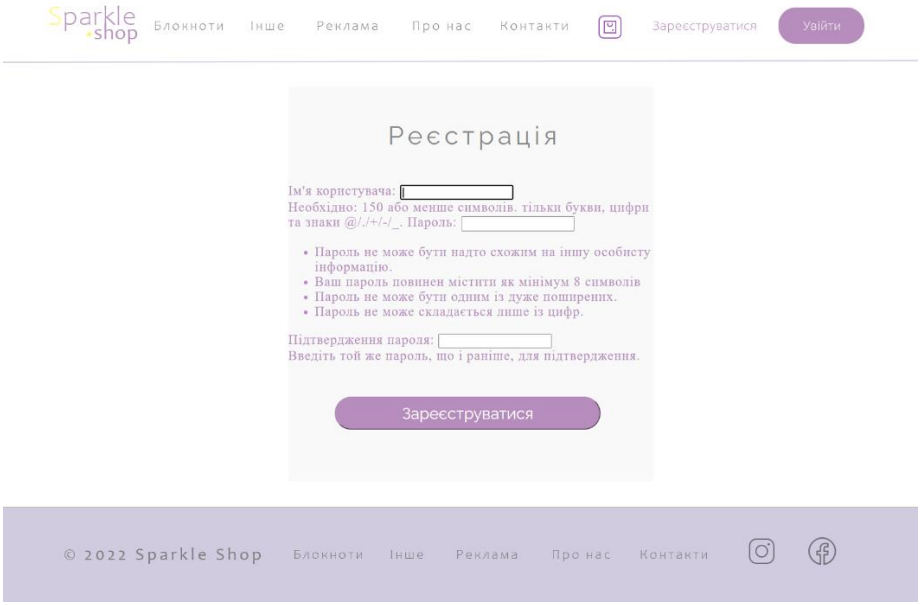


Рисунок 3.9 – Макет дизану сторінки “Реєстрація” веб-додатку

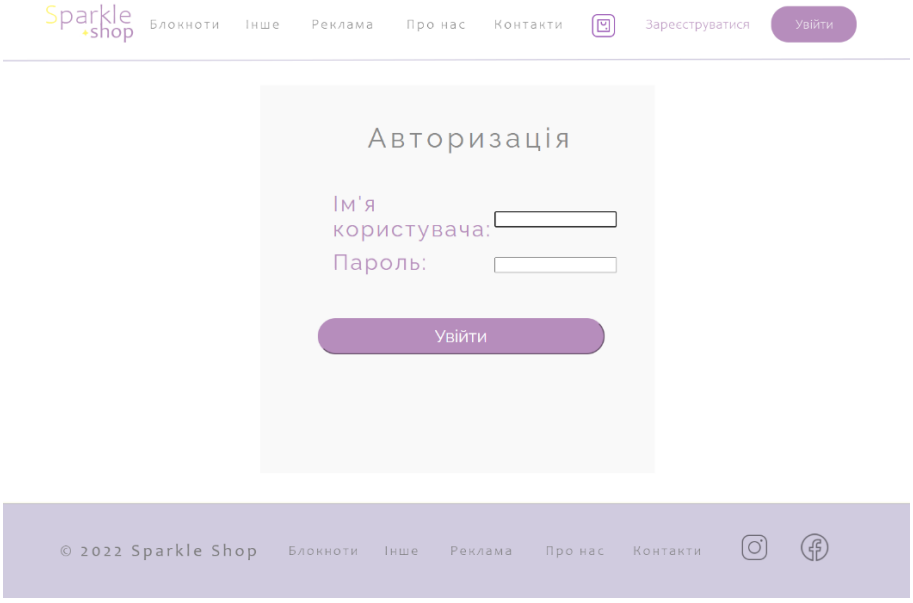


Рисунок 3.10 – Макет дизану сторінки “Авторизація” веб-додатку

Заключним етапом у розробці макетів стало створення дизайну сторінки продукту(рис. 3.11), залежно від обраної речі будуть змінюватись картинки та опис, а сама констрцкція сторінки залишатиметься сталою.

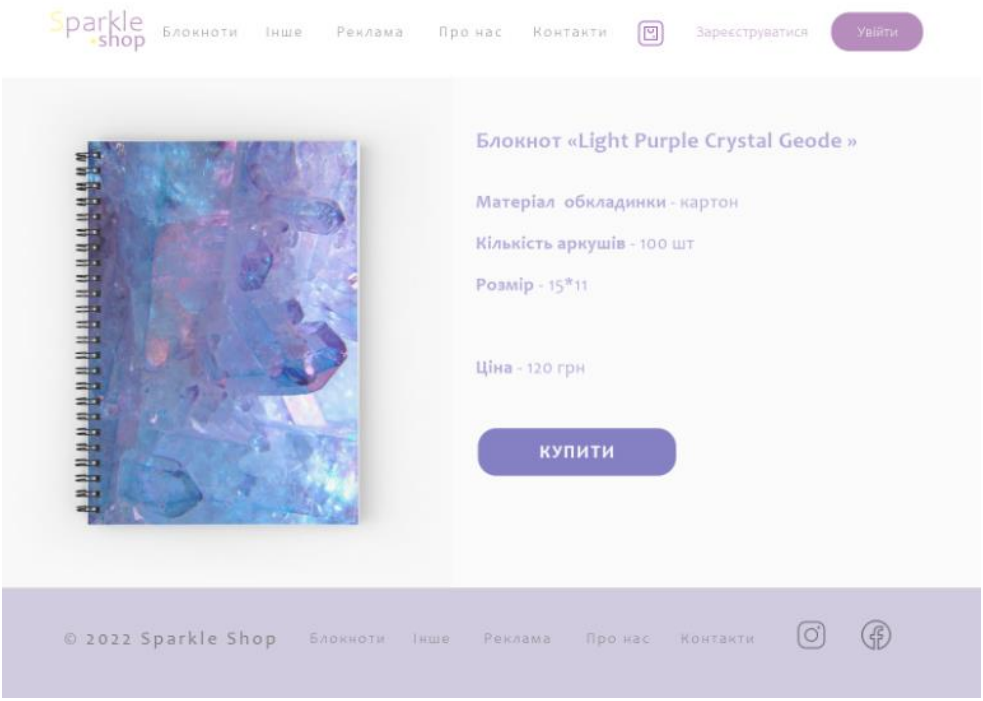


Рисунок 3.11 – Макет дизану сторінки продукту веб-додатку

Після розробки дизайну веб-додатку та обрання палітри кольорів, якої він буде дотримуватися, можна перейти до наступного етапу – створення відповідних сторінок, так званої “верстки”, за допомогою HTML та CSS.

3.1.2 Розробка веб-сторінок

Розробка веб-сторінок відбувається у такій же послідовності що і створення макетів дизайну, тобто спочатку це головна сторінка, потім сторінки навігаційного меню, потім сторіки реєстрації та аторизації і на кінець – сторінки усіх наявних на сайті продутків.

Створення веб-сторінок одразу розпочате у середовище розробки Pycharm для полегшення подальшої інтеграціїї з серверною частиною веб-додатку.

Всі веб-сторінки створені за допомогою мови розмітки HTML мають схожу структуру і однакову так звану “шапку”. Оскільки всі сторнки даного веб-додатку будуть мати однакову секцію навігації “header” (рис. 3.12), та

нижню підвальну секцію “footer” (рис. 3.13), їх код від сторінки до сторінки змінюватись не буде.

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Sparkle Shop</title>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link href="https://fonts.googleapis.com/css?family=Raleway:900|Raleway&amp;subset=cyrillic" rel="stylesheet">
    <link rel="stylesheet" href="css/main.css">
    <link rel="shortcut icon" href="img\logo.ico" type="image/x-icon">
    <link rel="stylesheet" href="css/animate.css">
</head>
<body>
    <header id="header" class="header">
        <div>
            <a href="index.html">
            <img src="img\logo.png" alt="logo" class="logo animated fadeInDown"></a>
        </div>

        <section id="navigation">
            <form class="menu animated fadeInDown">
                <style>
                    a {
                       text-decoration: none;
                       }
                    a:hover {
                       text-decoration: underline;
                       }
                </style>
                <a class="format_nv" href="Notebooks_page.html">Блокноти</a>
                <a class="format_nv" href="Staff_page.html">Інше</a>
                <a class="format_nv" href="Ads_page.html">Реклама</a>
                <a class="format_nv" href="About_page.html">Про нас</a>
                <a class="format_nv" href="Contacts_page.html">Контакти</a>
            </form>
        </section>
        <div class ="cart">
            <a href="Cart_page.html">
            <img src="img\cart.png" alt="cart" class="animated fadeInDown"></a>
        </div>
        <div class=" log_in_text_pos animated fadeInDown" >
            <a class="log_in_text" href="Registrarion.html" target="blank">Зареєструватися</a>
        </div>
        <form class="in_btn" action="Autorization_page.html" target="blank"><a href="Autorization_page.html"class="btn animated
        fadeInDown" id="btn" target="blank">Увійти</a>
        </form>
        <section id="Lending">
            <div class="lending">
                <img src="img\lending1.png" alt="lending" class="lending_img"></img>
            </div>
        </section>
    </header>
```

Рисунок 3.12 – Реалізація секції “header”

```
<section id="footer">

    <div class="footer">

        <div class="footer_title"> © 2022 Sparkle Shop </div>
        <form class="footer_text">
            <style>
                a {
                   text-decoration: none;
                   }
                a:hover {
                   text-decoration: underline;
                   }
            </style>
            <a class="format_nv" href="Notebooks_page.html">Блокноти</a>
            <a class="format_nv" href="Staff_page.html">Інше</a>
            <a class="format_nv" href="Ads_page.html">Реклама</a>
            <a class="format_nv" href="About_page.html">Про нас</a>
            <a class="format_nv" href="Contacts_page.html">Контакти</a>
        </form>

        <div id ="inst">
            <a href="https://www.instagram.com/sparkle_shop_ukraine/">
            <img src="img\icons8-instagram-50.png" alt="inst" class="inst_footer"></a>
        </div>



    </div>
    <div id ="facebook">
            <a href="https://www.facebook.com/sparkleshopukraine">
            <img src="img\icons8-facebook-50.png" alt="facebook" class="facebook_footer"></a>
    </div>

</section>
```

Рисунок 3.13 – Реалізація секції “footer”

Всі сторінки навігаційного меню, а також головна сторінка мають подбну структуру коду, всі вони розподілені на секції, для їх оптимізаії та покращення зручності керування. Головна сторінка поділена на наступні секції:

- Header;
- Lending;
- Choose the best;
- Instagram ads;
- Recommendations;
- Footer;

Сторінки “Блокноти” та “Інше” мають майже однакову струкуту та наступні секції:

- Header;
- Lending;
- New collection(“Блокноти”)/ Top pack(“Інше”);
- Notebooks(“Блокноти”)/ Stationery(“Інше”);
- Footer;

У відповідних секціях Notebooks та Stationery відбувається створення фільтрів(рис. 3.14) та відображення наявного товару.

Рисунок 3.14 – Розмітка фільтрів на сторінці “Інше”

Інші сторіки також поділені на відповідні секціїї, для сторінки “Реклама” це:

- Header;
- Lending;
- Our friends;
- Footer;

Сторінка “Про нас”:

- Header;
- Lending;
- About us;
- About us in icons;
- Our description;
- Footer;

Сторінка “Контакти”:

- Header;
- Lending;
- Our contacts;
- Footer;

Сторінка “Корзина”:

- Header;
- Cart;
- Footer;

І відповідно сторінка для відображення товару:

- Header;
- Item id;
- Footer;

Далі у таблиці 3.1 буде наведено список HTML тегів які були найчастіше використані при створенні веб-сторінок. Повний код розроблених сторінок можна знайти у Додатку 4.

Таблиця 3.1 – Найбільш вживані HTML теги при розробці даного веб-додатку

Назва тегу	Спосіб використання
<title>	Визначення заголовку сторінки
<link>	Підключення шрифтів, файлів стилів, іконок
	Створення блоку для контенту
<a>	Створення посилання
<img>	Додавання зображення
<section>	Виокремлення секції сторінки
<form>	Створення форми для контенту
<style>	Впровадження стилю для елемента
 	Перехід на наступний рядок
<p>	Створення абзацу
<button>	Створення кнопки
<script>	Підключення скрипту
<body>	Виокремлення тіла сторінки

Крім використання мови розмітки, до створений веб- сторіно також підключено декілька файлів стилів, що розроблено за допомогою CSS. Серед них два файли main.css – визначає стилі класів сторінок даного веб-додатку animate.css – впроваджує анімацію елементів на веб-сторінках.

Далі у таблиці 3.2 буде наведено список створених класів стилів у файлі main.css, за допомогою яких веб-сторінки виглядають таким же чином, як і на макетах дизайну.

Таблиця 3.2 – Стилі для класів сторінок веб-додатку

Назва	Опис
logo	Задає параметри положення логотипу на сторінці
menu	Задає параметри положення меню на сторінці
format_nv	Задає стиль для оформлення меню
cart	Задає положення іконки корзини на сторінці
log_in_text	Задає стиль тексту для кнопки “Увійти”
log_in_text_pos	Задає параметри положення тексту у кнопці “Увійти”
in_btn	Задає положення кнопки “Увійти” на сторінці
btn	Задає основний стиль для кнопок веб-додатку
in_btn:hover	Задає стиль кнопки “Увійти” принаведенні на неї
lending	Задає положення картинки лендінгу на сторінці
lending_img	Задає розміри лендінг-зображення
lending_title	Задає стиль для тексту-заголовку секцій
choose_img	Задає положення картинки з секції рекомендацій
choose_img_format	Задає розміри картинки з секції рекомендацій
choose_img_format:hover	Задає стиль картинки з секції рекомендацій при наведенні на неї
choose_img_text	Задає стиль текту з секції рекомендацій
choose_img_text_pos1	Задає позицію текту з секції рекомендацій
choose_img_text_pos2	Задає позицію текту з секції рекомендацій
inst_backg	Задає колір фону для секціїї з Інстаграмом
inst_page_format	Задає формат картинки для секціїї з Інстаграмом
inst_icon_format	Задає формат іконки для секціїї з Інстаграмом
inst_icon_format:hover	Задає формат іконки для секціїї з Інстаграмом при наведенні
inst_text	Задає стиль тексту для секціїї з Інстаграмом

Продовження таблиці 3.2– Стилі для класів сторінок веб-додатку

Назва	Опис
rec_img_1	Задає стиль та позицію 1-ї картинки з секції рекомендацій на головній сторінці
rec_img_2	Задає стиль та позицію 2-ї картинки з секції рекомендацій на головній сторінці
rec_img_3	Задає стиль та позицію 3-ї картинки з секції рекомендацій на головній сторінці
rec_img_4	Задає стиль та позицію 4-ї картинки з секції рекомендацій на головній сторінці
rec_img_5	Задає стиль та позицію 5-ї картинки з секції рекомендацій на головній сторінці
rec_img_6	Задає стиль та позицію 6-ї картинки з секції рекомендацій на головній сторінці
rec_img_7	Задає стиль та позицію 7-ї картинки з секції рекомендацій на головній сторінці
rec_img_8	Задає стиль та позицію 8-ї картинки з секції рекомендацій на головній сторінці
circle1	Задають стиль для вказівників круглої форми на картинках секції рекомендацій
circle2	
message1	Задають стиль і формат повідомлення та його тексту у секції рекомендацій
message2	
message1:hover	Задає стиль тексту повідомлення у секції рекомендацій при наведенні
message2:hover	Задає формат повідомлення у секції рекомендацій при наведенні
footer	Задає розмір футеру та його колір фону
footer_title	Задає формат для торгової марки у футері

Продовження таблиці 3.2 – Стилі для класів сторінок веб-додатку

Назва	Опис
footer_text	Задає стиль та положення тексту у футері
inst_footer	Задає формат іконки Інстаграму у футері
inst_footer:hover	Задає формат іконки Інстаграму у футері при наведенні
facebook_footer	Задає формат та положення іконки Фейсбуку у футері
facebook_footer:hover	Задає формат та положення іконки Фейсбуку у футері при наведенні
slides_notebook	Задає стиль для контейнеру зі слайдами на сторінці блокнотів
slider_btn1	Задає стиль для 1-ї кнопки контейнеру зі слайдами на сторінці блокнотів
slider_btn2	Задає стиль для 2-ї кнопки контейнеру зі слайдами на сторінці блокнотів
slider	Задає стиль для слайду на сторінці блокнотів
slider_btn1:hover	Задають стиль для 1-ї та 2-ї кнопок контейнеру зі слайдами на сторінці блокнотів при наведенні
slider_btn2:hover	
notebook_pink_area	Задає стиль та положенні для секції фільтрів
filter_title	Задає стиль заголовку для секції фільтрів
filter_titles	Задає стиль підзаголовків для секції фільтрів
filter_size	Задає розмір тексту для секції фільтрів
clear_filter	Задає стиль кнопи очистки для секції фільтрів
main	Задає стиль для секції відображення товарів
notebooks_format	Задає формат картинки товару у секції для відображення

Продовження таблиці 3.2 – Стилі для класів сторінок веб-додатку

Назва	Опис
notebooks_format:hover	Задає формат картинки товару у секції для відображення при наведенні
notebooks_description	Задає стиль тексту під картинкою товару у секції для відображення
top_packs	Задає формат зображень у секції топ наборів на сторінці “Інше”
top_packs:hover	Задає формат зображень у секції топ наборів на сторінці “Інше” при наведенні
our_friends_block	Задає стиль блоку для розміщенні реклами на сторінці “Реклама”
friend_pic	Задає формат зображення у блоці для розміщенні реклами на сторінці “Реклама”
friend_title	Задає стиль заголовку та тексту у блоці для розміщенні реклами на сторінці “Реклама”
friend_text	
inst_for_friends	Задає формат іконки Інстаграму у блоці для розміщенні реклами на сторінці “Реклама”
friend_pic2	Задає формат зображення у 2-гому блоці для розміщенні реклами на сторінці “Реклама”
our_friends_block2	Задає стиль 2-го блоку для розміщенні реклами на сторінці “Реклама”
about_circle1	Задають стиль та положення для кругів у секції про нас на сторінці “Про нас”
about_circle2	
about_circle2:hover	Задає стиль кругів у секції про нас на сторінці “Про нас” при наведенні
circle_text	Задає стиль тексту у кругах у секції про нас на сторінці “Про нас”

Кінець таблиці 3.2 – Стилі для класів сторінок веб-додатку

Назва	Опис
undercircle_text	Задає стиль тексту під кругами у секції про нас на сторінці “Про нас”
about_us_back	Задає формат фону для секції про нас на сторінці “Про нас”
icon_about	Задає формат іконок у секції про нас на сторінці “Про нас”
our_desc_text	Задає формат тексту для секції опису на сторінці “Про нас”
contact_back	Задає формат фону для секції контактів на сторінці “Контакти”
reg_backg	Задає формат фону на сторінці реєстрації
reg_inputs	Задає формат полів вводу на сторінці реєстрації
reg_text	Задає стиль тексту на сторінці реєстрації
reg_inputs_btn	Задає формат кнопки підтвердження на сторінці реєстрації
reg_inputs_btn:hover	Задає формат кнопки підтвердження на сторінці реєстрації при наведенні
item_conteiner	Задає формат фону сторінки товару
img_item_format	Задає формат картинки на сторінці товару
img-magnifier-glass	Задає стиль лупи на сторінці товару
text_item	Задає стиль тексту на сторінці товару
buy_button	Задає стиль кнопки “Купити” на сторінці товару
cart_content	Задає стиль тексту у корзині
shopping_list td shopping_list td tr	Задає формат рядків та стовпців у заповненій корзині

Після створення шаблонів всіх сторінок додатку та впровадження до них стилів, наступним етапом є зробити їх функціональними, а це вже задача для JavaScript.

3.1.3 Розробка функцій

Для коректного функціонування розроблених веб-сторінок додатку було створено п’ять програмних файлів JavaScript, з використанням бібліотеки jQuery, кожен з яких відповідає за певний функціонал.

Перший файл – main.js. Це перший JavaScript файл проекту, який реалізує в собі три функції: відображення слайдів та сторінці з блокнотами, їх перемикання, а також очистка всіх обрих фільтрів на сторінці.

На рис. 3.15 показано фрагмент коду з файлу main.js котрий являє собою функції відображення та перемикання слайдів.

```
var slideIndex = 1;
showDivs(slideIndex);
//////////////функція перемикання слайду
function plusDivs(n) {
  showDivs(slideIndex += n);
}
//////////////функція відображення слайдів
function showDivs(n) {
  var i;
  var x = document.getElementsByClassName("mySlides");
  if (n > x.length) {slideIndex = 1}
  if (n < 1) {slideIndex = x.length}
  for (i = 0; i < x.length; i++) {
    x[i].style.display = "none";
  }
  x[slideIndex-1].style.display = "block";
}
```

Рисунок 3.15 – Функції для слайдів у main.js

Функція для очистки фільтрів(рис. 3.16) надає користувачу можливість зняти галочки з усіх обрих чекбокс-фільтрів одночасно, замість того, щоб робити це з кожним окремо.

```
NodeList.prototype.forEach = Array.prototype.forEach;
//////////////функція очистки фільтрів
document.getElementById("clear").addEventListener("click", function(e) {
  var formBlock = document.getElementById("formId");
  var inputArr = formBlock.querySelectorAll("input[type=checkbox]");
  inputArr.forEach(function(el) {
    el.checked = false;
  });
});
```

Рисунок 3.16 – Функції для очистки фільтрів у main.js

Наступний створений Java Script має назву filter_notebooks.js та містить у собі функції, що реалізують процес фільтрування блокнотів за обраними параметрами, та відповідно відображення результатів фільтрування. Файл містить три функції: для присвоювання змінних чекбоксів(рис. 3.17), для оримання класу фультру(рис. 3.18) та для відображення результату(рис. 3.19).

```
//////////функція присвоювання змінних чекбоксів
function change() {
  var sizeCbs = document.querySelectorAll(".size input[type='checkbox']");
  var colorCbs = document.querySelectorAll(".color input[type='checkbox']");
  var materialCbs = document.querySelectorAll(".material input[type='checkbox']");
  var saleCbs = document.querySelectorAll(".sale input[type='checkbox']");
  var filters = {
    size: getClassOfCheckedCheckboxes(sizeCbs),
    color: getClassOfCheckedCheckboxes(colorCbs),
    material: getClassOfCheckedCheckboxes(materialCbs),
    sale: getClassOfCheckedCheckboxes(saleCbs)
  };

  filterResults(filters);
}
```

Рисунок 3.17 – Функція присвоювання змінних чекбоксів у filter_notebooks.js

```
//////////функція отримування класів чекбоксів
function getClassOfCheckedCheckboxes(checkboxes) {
  var classes = [];

  if (checkboxes && checkboxes.length > 0) {
    for (var i = 0; i < checkboxes.length; i++) {
      var cb = checkboxes[i];

      if (cb.checked) {
        classes.push(cb.getAttribute("rel"));
      }
    }
  }

  return classes;
}
```

Рисунок 3.18 – Функція оримання класу фультру у filter_notebooks.js

```
//////////функція відображення результатів фільтрації
function filterResults(filters) {
  var rElems = document.querySelectorAll(".result div");
  var hiddenElems = [];

  if (!rElems || rElems.length <= 0) {
    return;
  }

  for (var i = 0; i < rElems.length; i++) {
    var el = rElems[i];

    if (filters.size.length > 0) {
      var isHidden = true;

      for (var j = 0; j < filters.size.length; j++) {
        var filter = filters.size[j];

        if (el.classList.contains(filter)) {
          isHidden = false;
          break;
        }
      }

      if (isHidden) {
        hiddenElems.push(el);
      }
    }
```

Рисунок 3.19 – Функція відображення результату фільтрації у filter_notebooks.js

Далі було створено Java Script файл який має назву filter_stuff.js, він як і попередній складається із трьох функцій, які відповідають за присвоювання змінних чекбоксів, оримання класу фультру та для відображення результату. Відмінністю від попереднього файлу є те, що filter_stuff.js реалізує роботу фільтрів для ішних канцелярський товарів, а filter_notebooks.js – блокнотів.

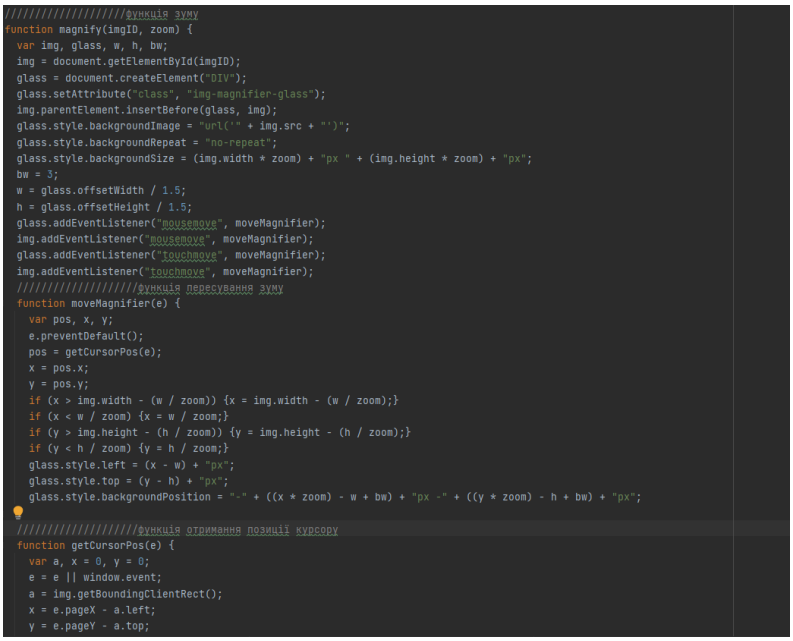
Принцип роботи в них залишається ідентичним, різняться лише змінні(рис. 3.20).

```
//////////функція присвоювання змінних чекбоксів
function change() {
  var pensCbs = document.querySelectorAll(".pens input[type='checkbox']");
  var markersCbs = document.querySelectorAll(".markers input[type='checkbox']");
  var pencilsCbs = document.querySelectorAll(".pencils input[type='checkbox']");
  var stickersCbs = document.querySelectorAll(".stickers input[type='checkbox']");
  var spinsCbs = document.querySelectorAll(".spins input[type='checkbox']");
  var packsCbs = document.querySelectorAll(".packs input[type='checkbox']");
  var stuffsCbs = document.querySelectorAll(".stuffs input[type='checkbox']");
  var colorCbs = document.querySelectorAll(".color input[type='checkbox']");
  var saleCbs = document.querySelectorAll(".sale input[type='checkbox']");
  var filters = {
    pens: getClassOfCheckedCheckboxes(pensCbs),
    markers: getClassOfCheckedCheckboxes(markersCbs),
    pencils: getClassOfCheckedCheckboxes(pencilsCbs),
    stickers: getClassOfCheckedCheckboxes(stickersCbs),
    spins: getClassOfCheckedCheckboxes(spinsCbs),
    packs: getClassOfCheckedCheckboxes(packsCbs),
    stuffs: getClassOfCheckedCheckboxes(stuffsCbs),
    color: getClassOfCheckedCheckboxes(colorCbs),
    sale: getClassOfCheckedCheckboxes(saleCbs)
  };

  filterResults(filters);
}
```

Рисунок 3.20 – Функція присвоювання змінних чекбоксів у filter_stuff.js

Наступним створено файл zoom.js який реалізує можливість користувача збільшувати певний фрагмент картинки, на сторінці відображенні товару, при наведенні на ньому курсором. Даний файл складається з однієї функції , що реалізує створення зуму на зображенні товару та викликається зі сторінки товару, та двох вкладених у неї, які відповідно відповідають за пересування зуму та отримання позиції курсору (рис. 3.21).



```
//////////////////функція зуму
function magnify(imgID, zoom) {
  var img, glass, w, h, bw;
  img = document.getElementById(imgID);
  glass = document.createElement("DIV");
  glass.setAttribute("class", "img-magnifier-glass");
  img.parentElement.insertBefore(glass, img);
  glass.style.backgroundImage = "url('" + img.src + "')";
  glass.style.backgroundRepeat = "no-repeat";
  glass.style.backgroundSize = (img.width * zoom) + "px " + (img.height * zoom) + "px";
  bw = 3;
  w = glass.offsetWidth / 1.5;
  h = glass.offsetHeight / 1.5;
  glass.addEventListener("mousemove", moveMagnifier);
  img.addEventListener("mousemove", moveMagnifier);
  glass.addEventListener("touchmove", moveMagnifier);
  img.addEventListener("touchmove", moveMagnifier);
  ////////////////////функція пересування зуму
  function moveMagnifier(e) {
    var pos, x, y;
    e.preventDefault();
    pos = getCursorPos(e);
    x = pos.x;
    y = pos.y;
    if (x > img.width - (w / zoom)) {x = img.width - (w / zoom);}
    if (x < w / zoom) {x = w / zoom;}
    if (y > img.height - (h / zoom)) {y = img.height - (h / zoom);}
    if (y < h / zoom) {y = h / zoom;}
    glass.style.left = (x - w) + "px";
    glass.style.top = (y - h) + "px";
    glass.style.backgroundPosition = "-" + ((x * zoom) - w + bw) + "px -" + ((y * zoom) - h + bw) + "px";
  }
  ////////////////////функція отримання позиції курсору
  function getCursorPos(e) {
    var a, x = 0, y = 0;
    e = e || window.event;
    a = img.getBoundingClientRect();
    x = e.pageX - a.left;
    y = e.pageY - a.top;
```

Рисунок 3.21 – Реалізація функцій у zoom.js

Останній створений файл функцій має назву cart.js він повністю відповідає за функціонування корзини. Тобто це і реалізація додавання товару в корзину, очистка корзина, запис та отримування даних з локального сховища, формування даних для виводу таблиці корзини, розрахунок загальної вартості замовлення та інше. На рис. 3.22 можна побачити частину коду яка відповідаає за отримування даних корзини з локального сховища.

```
//////////отримування даних з локалстореджу
function getCartData(){
  return JSON.parse(localStorage.getItem('cart'));
}
//////////запис даних у локалсторедж
function setCartData(o){
  localStorage.setItem('cart', JSON.stringify(o));
  return false;
}

//////////відображення даних у таблиці корзині
function viewDiv(){
  document.getElementById("divtest").style.display = "block";
  document.getElementById("cart_content").style.display = "none";
  document.getElementById("choose").style.display = "none";
  document.getElementById("clear_cart").style.display = "none";
```

Рисунок 3.22 – Функції роботи з localStorage у cart.js

Реалізувавши інтерфейс користувача та певні функції взаємодії з ним, можна переходити до розробки програмних компонентів серверної частини веб-додатку.

3.2 Розробка програмних компонентів серверної частини

У даному підроздірі буде описано процес створення серверної частини веб-додатку за допомогою технологій, що надає мова програмування Python та її бібліотека Django.

3.2.1 Створення Django проєкту

Першим етапом у створенні серверної частини веб-додатку є звичайно створення проекту. Оскільки для реалізації бекенду було обрано Python

фреймворк Django, починаємо з його встановлення та налаштування. Для роботи з проєктом обрано середовище розробки від JetBrains під назвою PyCharm Community Edition версії 2022.1.1.

Спочатку у середовищі розробки створюємо новий проєкт під назвою “Diploma”, саме у ньому буде далі відбуватися робота. Наступним кроком є встановлення Django фреймворку(рис. 3.23).

```
PS C:\Users\Acer\PycharmProjects\Diploma> pip install django
```

Рисунок 3.23 – Встановлення у проєкт Django фреймфорку

Після даної процедури встановлення, можна розпочинати роботу безпосереньо з Django проєктом(рис. 3.24), називаємо його SparkleShop, так само як і себ-додаток.

```
PS C:\Users\Acer\PycharmProjects\Diploma> django-admin startproject sparkleshop
```

Рисунок 3.24 – Початок роботи з Django проєктом

Після виконання даної команди у структурі проєкту з’являються всі необхідні файли. Також створюємо окремий додаток main, з яким надалі будемо працювати для розробки програмних компонентів серверної частини веб-додатку. Після виконання вищезазначених дій структура проєкту виглядає наступним чином(рис. 3.25).

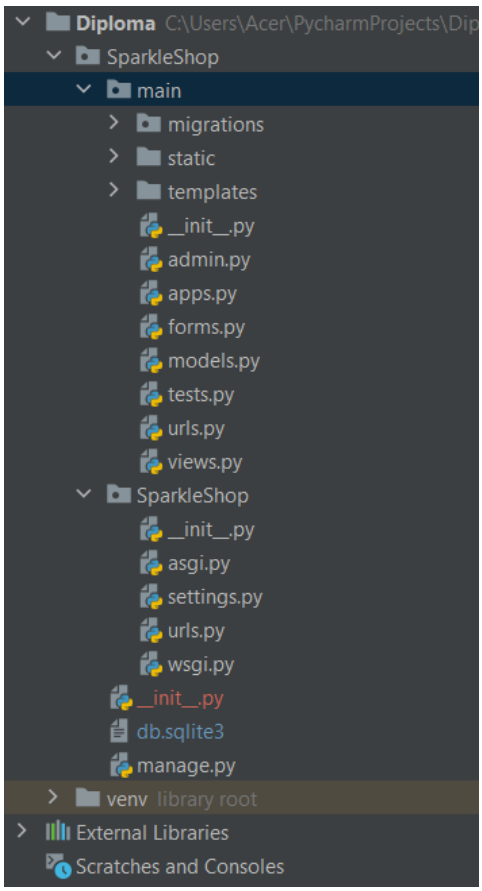


Рисунок 3.25 – Структура Django проєкту

3.2.2 Створення панелі адміністратора

Django фреймворк надає дуже зручні та гнучкі інструменти для розробки веб-додатків. Величезним плюсом є уже вбудована в нього панель адміністратора, все що потрібно зробити це додати її у проєкт, а також визначити логін і пароль для суперюзера, тобто адміністратора(рис. 3.26).

```
PS C:\Users\Acer\PycharmProjects\Diploma\sparkleshop> python manage.py createsuperuser
```

Рисунок 3.26 – Створення адміністратора веб-додатку Django проєкту

Після виконання даної команди можна зайти на локальний сервер з приставкою admin та ввівши всі раніше визначені дані потрапити до панелі адміністрування веб-додатку(рис. 3.27).

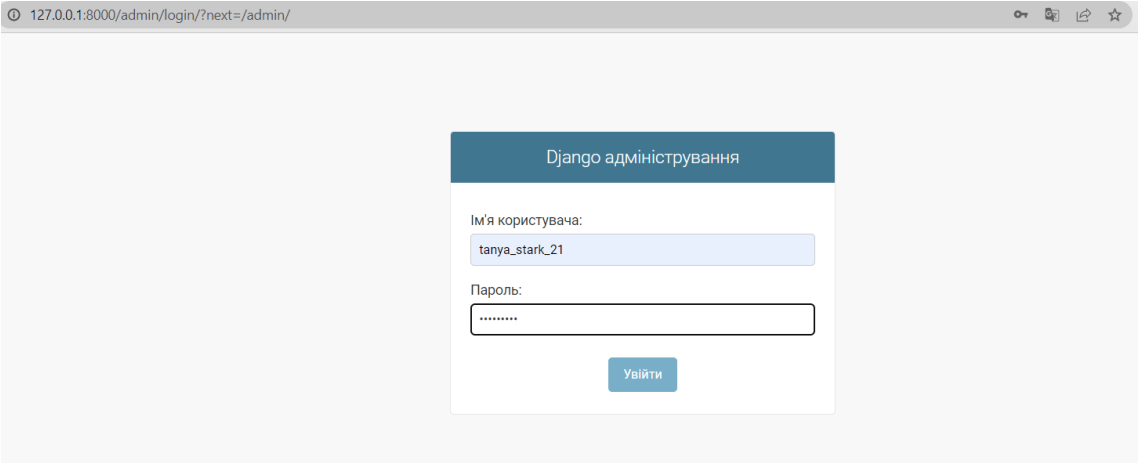


Рисунок 3.27 – Вхід до панелі адміністратора веб-додатку

Для розгортання локального серверу, який доречі теж вбудований у Django (та запускається командою python manage.py runserver), використовується веб-браузер Google Chrome. Передбачається, що на панелі адміністратора будуть відображатися зареєстровані користувачі, замовлення, що надійшли, інформація про замовника, місце та спосіб доставки, також наявні товари на складі. Адміністратор зможе змінювати статус замовлення в прогресі його виконання, а також змінювати кількість наявного на сладі товара, при перерахуванні, тощо.

3.2.3 Створення виглядів, форм та шляхів

Перш ніж перейти до створення форм, прописання шляхів до сторінок та виглядів, потрібно у додатку main створити дві папки: static – буде відповідати за статичні файли, тобто скрипти, стилі, зображення та інше, templates – буде містити у собі всі раніше створені html файли сторінок. Створивши вищезазначені папки, переміщуємо туди всі файли, які були створені під час верстки проєкту(рис. 3.28).

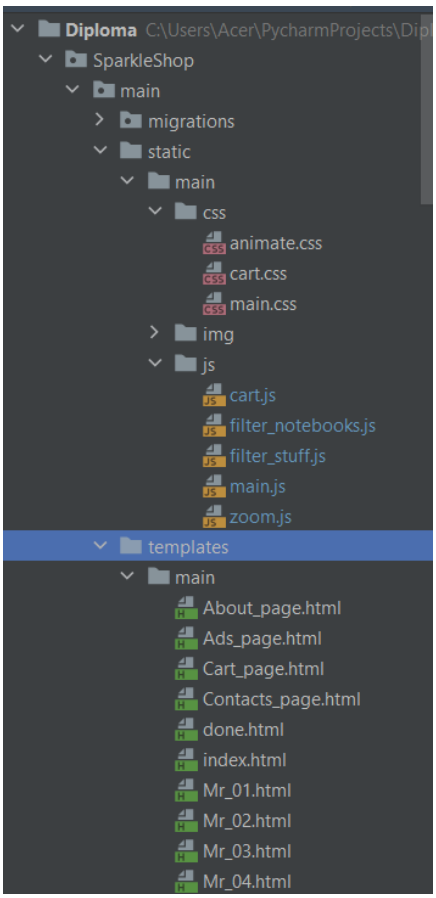


Рисунок 3.28 – Структура проєкту після додавання файлів верстки

У даному проєкті є два файли з назвою urls.py, перший знаходиться безпосередньо у папці SparkleShop та відповідає за декларацію шляхів до сторінки адмін-панелі та веб-сторінок(рис. 3.29), що знаходяться у додатку main, а другий urls.py розташований у вищезгаданому додатку та відповідає за декларацію шляхів до усіх html шаблонів(рис. 3.30).

Обидва файли urls.py мають однакову структуру – це імпорт бібліотек, список підключених шляхів urlpatterns та підключення статичних файлів. Підключення шляхів до веб-сторінок відбувається наступним чином, викликається метод *path*, який має три параметри: перший – назва html шаблону сторінки, до якої веде цей шлях, другий - метод із файлу *view*, що відповідає за відображення даного шаблону та третій – *name*, тобто ім’я, за яким можна звертатись до даного шаблону веб-сторінки.

```python
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('main.urls'))
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Рисунок 3.29 – Прописання шляхів у файлі urls.py

```python
from . import views, admin
from django.urls import path,include

urlpatterns = [
        path('admin/', admin.admin.site.urls),
        path('accounts/', include('django.contrib.auth.urls')),
        path('', views.index, name = 'index'),
        path('Notebooks_page',views.Notebooks_page , name = 'Notebooks_page'),
        path('done',views.done , name = 'done'),
        path('userpage',views.userpage , name = 'userpage'),
        path('Staff_page',views.Staff_page , name = 'Staff_page'),
        path('Ads_page',views.Ads_page , name = 'Ads_page'),
        path('About_page',views.About_page , name = 'About_page'),
        path('Contacts_page',views.Contacts_page , name = 'Contacts_page'),
        path('Cart_page',views.Cart_page , name = 'Cart_page'),
        path('Order',views.Order , name = 'Order'),
        path('Registrarion',views.SignUp.as_view() , name = 'Registrarion'),
        path('Autorization_page',views.Autorization_page , name = 'Autorization_page'),
        path('Order',views.Order , name = 'Order'),

        path('Mr_01',views.Mr_01 , name = 'Mr_01'),
        path('Mr_02',views.Mr_02 , name = 'Mr_02'),
        path('Mr_03',views.Mr_03 , name = 'Mr_03'),
        path('Mr_04',views.Mr_04 , name = 'Mr_04')
```

Рисунок 3.30 – Прописання шляхів у файлі urls.py додатку main

Останній параметр name є необов’язковим, але навіть при зміні url адресі сторінці, можно буде звертатися до неї по вищезазначеному імені та не змінювати його у всіх інших файлах.

Після декларації усіх шляхів до усіх веб-сторінок, що знаходяться у додатку, приступаємо до створення функцій відображення даних сторінок. Для цього потрібно працювати з файлом views.py, що знаходиться у додатку main. Всі методи для відображення сторінок мають подібну структуру(рис. 3.31): назва функції, параметр *request*, та метод *render* з відповідними параметрами(перший – це *request*, другий – шлях до шаблону у вигляді строки), що повертає функція.

```python
def index(request):
    return render(request, 'main/index.html')
```

Рисунок 3.31 – Приклад функції відображення веб-сторінки з файлу views.py

Далі у Таблиці 3.3 буде наведено список усіх створених функцій та класів у файлі views.py та їх опис. Також окремо потрібно виділити клас *SignUp*(рис. 3.32) , який відповідає за відображення форми реєстрації на сторінці реєстрації, та як параметр приймає CreateView, використовуючи надану Django стандартну форму реєстрації користувачів. Html шаблон сторінки реєстрації знаходиться у окремій папці додату main під назвою registration.

```python
class SignUp(CreateView):
    form_class = UserCreationForm
    success_url = reverse_lazy("login")
    template_name = "registration/Registrarion.html"
```

Рисунок 3.32 – Клас *SignUp* з файлу views.py

Таблиця 3.3 – Опис функцій відображення веб-сторінок

Назва	Опис
index	Реалізує відображення головної сторінки веб-додатку
Notebooks_page	Реалізує відображення сторінки “Блокноти” веб-додатку
Staff_page	Реалізує відображення сторінки “Інше” веб-додатку
Ads_page	Реалізує відображення сторінки “Реклама” веб-додатку
About_page	Реалізує відображення сторінки “Про нас” веб-додатку
Contacts_page	Реалізує відображення сторінки “Контакти” веб-додатку
done	Реалізує відображення сторінки веб-додатку, котра з’являється при успішному замовленні користувачем товару
userpage	Реалізує відображення сторінки веб-додатку, котра являє собою особистий кабінет користувача
Order	Реалізує відображення сторінки веб-додатку, котра з’являється при оформленні замовлення, завдяки ajax-запитам повертає також загальну суму замовлення та список замовлених товарів, реалізує зберігання форми замовлення та при успішному оформленні замовлення, переадресовує користувача на сторінку “done”
Cart_page	Реалізує відображення сторінки “Корзина” веб-додатку, за допомогою методу *request.POST.get*() отримує значення загальної суми замовлення та списку замовлених товарів та передає їх функції *Order*
Autorization_page	Реалізує відображення сторінки авторизації веб-додатку
Mr_01 . . Mr_08	Реалізують відображення всіх сторінок продуктів для маркерів

Продовження таблиці 3.3 – Опис функцій відображення веб-сторінок

Назва	Опис
Nb_01 . . Nb_30	Реалізують відображення всіх сторінок продуктів для блокнотів
Ot_01 . . Ot_11	Реалізують відображення всіх сторінок продуктів декору та різної канцелярії, що не має категорії
Pc_01 . . Pc_08	Реалізують відображення всіх сторінок продуктів для наборів канцелярії
Pn_01 . . Pn_10	Реалізують відображення всіх сторінок продуктів для ручок
Pnc_02 . . Pnc_08	Реалізують відображення всіх сторінок продуктів для олівців
Sp_01 . . Sp_06	Реалізують відображення всіх сторінок продуктів для защіпок
St_01 . . St_10	Реалізують відображення всіх сторінок продуктів для стікерів

Далі розгланемо створення форм у даному веб-додатку, їх було створено три. Перша та друга – для регістрації та авторизації відповідна, третя – форма оформлення замовлення. Форми для реєстрації та авторизації були створені без повно кастомізації, за допомогою класу *UserCreationForm*, який стандартно надає фреймворк Django. За допомогою вищезазначеного класу можна імпортувати у свій додаток вже готові форми з гарною валідацією для реєстрації та авторизації користувачів.

Форма для оформлення замовлення було створена кастомізовано, а також підключена до моделі замовлень у базі даних, про це буде йти мова у наступному підроздірі. Створена форма знаходиться у файлі forms.py додатку main та має назву *OrdersForm*(рис. 3.33), та по-суті являє собою клас, де параметром виступає відповідна модель у базі даних. Клас *OrdersForm* має підклас *Meta*, який ініціалізує поля заданої форми, та кастомізує їх і надає певні стилі за допомогою словнику *widgets*, де ключами являються відповідні поля форми, а значеннями їх параметри. підклас *Meta* також містить у собі список *fields* у якому задекларовані всі поля з відповідної моделі у базі даних, котрі мають бути відображеня на сторінці.

```
class OrdersForm(ModelForm):
    class Meta:
        model = Orders
        fields = ['username','orderer_items', 'total_sum', 'name', 'surname', 'phone_number', 'email',
                  'city', 'post_number', 'payment_method']


        widgets = {
            "username": TextInput(attrs={
                'class': 'to_order',
                'placeholder': "Введіть ваш username"
            }),

            "orderer_items": Textarea(attrs={

                'class': 'to_order',
                'placeholder': "Замовлені товари",
                'style': 'resize:none; font-size:14px; text-align:left;',
                'readonly': 'readonly'
            }),
            "total_sum": TextInput(attrs={
```

Рисунок 3.33 – Клас *OrdersForm* з файлу forms.py

Таблиця 3.4 надає наглідне представлення назви елементів *fields* та за які поля з моделі вони відповідають.

Таблиця 3.4 – Опис полів форми OrdersForm

Назва	Опис
username	Відображення поля “Username користувача”
order_items	Відображення поля “Замовлені товари”
total_sum	Відображення поля “Сума замовлення”
name	Відображення поля “Ім’я”

Продовження таблиці 3.4 – Опис полів форми OrdersForm

Назва	Опис
surname	Відображення поля “Username користувача”
phone_number	Відображення поля “Номер телефону”
email	Відображення поля “E-mail”
city	Відображення поля “Назва населеного пункту”
post_number	Відображення поля “Номер відділення Нової Пошти”
payment_method	Відображення поля “Спосіб оплати”

Після опису всіх форм, шляхів та методів для відображення веб-сторінок даного додатку можна переходити до роботи за базою даних.

3.3 Розробка програмних компонентів бази даних

У даному підрозділі буде описано створення Django моделей для взаємдії з базою під’єднаною базою даних sqlite3.

3.3.1 Реєстрація та авторизація

Django фреймворк надає велику кількість зручних та гнучкіх інструментів для розробки веб-додатку. Одним із таких інструментів є додаток *auth*, встановивши який під час створення проєкту, можна отримати доступ до вбудованого у Джанго функціоналу реєстрації та авторизвції користувачів. Щоб встановити вище згаданий додаток, потрібно у файлі налаштування проєкту прописати повну назву додатку, у секції для встановлених додатків(рис. 3.34).

```
# Application definition

INSTALLED_APPS = [
    'main',
    'django.contrib.admin',
    'django.contrib.auth',
```

Рисунок 3.34 – Встановлення додатку *auth*

Після виконання даної процедури розробник отримує доступ до вбудованих методів для роботи з авторизацією та реєстрацію нових користувачів у додатку.

Як раніше показувалося, у файлі для відображень створюється клас *SignUp,* де під'єднується Django форма для реєстрації користувачів, у цій же формі відбувається валідація всіх полів, а також її інтеграція з базою даних та паленню адміністратора. Все що потрібно, це створити html шаблон для виводу даної форми і якщо реєстрація нового користувача відбулася успішно, на панелі адміністрування це можна легко побачити(рис. 3.35).

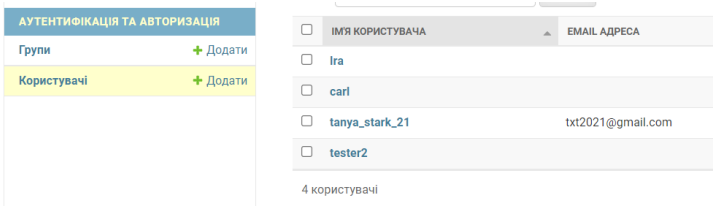


Рисунок 3.35 – Відображення зареєстрованих користувачів на палелі адміністрування

У тому ж самому класі *SignUp* є посилання на сторінку авторизації вже зареєстрованого користувача. Авторизація відбувається за допомогою вбудованих у Django методів, що перевіряють коректність введених користувачем даних, а також дають можливість вийти з системи. Для авторизації та виходу з облікового запису теж було створено окремі html шаблони, котрі знаходяться у окремій папці registration додатку main та мають назви відповідно *Registration, login, logged_out.*

При реєстрації перевіряється, чи немає користувача з обраним логіном вже у базі даних додатку, надійність пароля, його валідація та запрошується повторне його введення. Якщо при авторизації користувач ввів неправильні дані, він не зможе ввійти у систему.

Авторизовані користувачі мають змогу відвідувати особистий кабінет, в якому відображається інформація про зроблені ними замовлення(рис. 3.36).

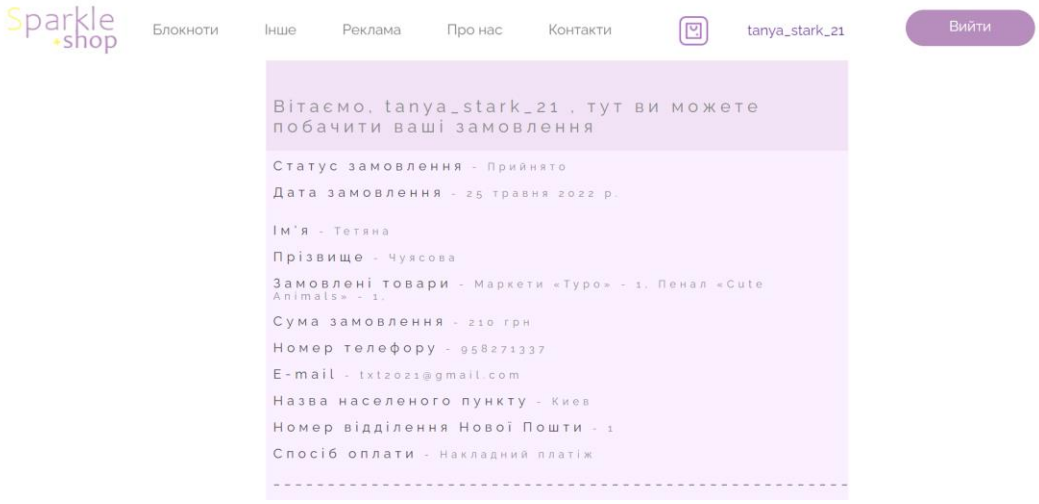


Рисунок 3.36 – Особистий кабінет користувача

Якщо користувач авторизований, то він має змогу здійснити замовлення. Якщо даного користувача немає в базі даних додатку, то він має створити свій обліковий запис для здійснення замовлення.

3.3.2 Модель замовлень

У Django за взаємодію з базою даних відповідають спеціальні моделі. По суті це класи, що створюються у спеціально файлі models.py і являють собою таблицю у базі баних. Після створення класу моделі генеруються файли так званої міграції, завдяки яким можна провести міграцію даної моделі з базою даних, внаслідок чого утвориться таблиці. Моделі також пов’язані з формами, які заповнюють користувачі.

Першою описаною моделлю буде модель замовлення – вона надає інформаію про замовлений користувачем товар. Для її створення потрібно у файлі models.py у додатку main створити клас *Orders* за параметром *models.Model*(рис. 3.37), ща вказує на те, що вищезазначений клас являє собою модель. Після створення класу переходимо до створення полів моделі, що будуть відображатися в таблиці бази даних. Дана модель замовлень буде мати дванадцять полів, кожне за яких буде відіповідати за наступні речі:

- *username* – ім’я користувача(логін) замовника;
- *status* – статус замовлення(відповідно від прогресу виконання замовлення адміністратор змінює його статус через адім панель, а користувач у свою чергу може відслідковувати його через особистий кабінет, всього є чьотири статусу замовлення: нове, прийнято, відправлено та доставлено);
- *orderer_items* – замовлені речі(повертає список замовлених речей);
- *total_sum* – загальна сума замовлення;
- *name* – ім’я замовника;
- *surname* – прізвище замовника;
- *phone_number* – номер телефону замовника;
- *email* – e-mail замовника;
- *city* – назва населеного пункту для доставки;
- *post_number* – номер відділення Нової Пошти;
- *payment_methon* – спосіб оплати замовлення(користувач при заповнені форми замовлення обирає метод оплати, це може бути або накладний платіж, або переказ на картку);
- *date* – дата замовлення;

Поле у моделі створюється наступним чином: створюється змінна у класі *Orders* з відповідним ім’ям, далі їй присвоюється значення що дорівнює відповідному методу з класу *models,* це може бути *CharField() TextField() DateField(),* тощо. Далі у цей метод передаються відповідні аргументи, перший – назва данного поля у текстовому форматі, другий – це максимальна довжина символів, що можна ввести в задане поле, третій – значення за замовчуванням, тощо.

```python
class Orders(models.Model):
    username = models.CharField("Username користувача" , max_length=100)
    status = models.CharField("Статус замовлення" , max_length=20, choices=ORDER_CHOICES, default='Новий')
    orderer_items = models.TextField("Замовлені товари" , max_length=1000)
    total_sum = models.CharField("Сума замовлення" , max_length=30_)
    name = models.CharField("Ім'я", max_length=30)
    surname = models.CharField("Прізвище", max_length=50)
    phone_number = models.IntegerField("Номер телефору" , max_length=25)
    email = models.EmailField("E-mail", max_length=50)
    city = models.CharField("Назва населеного пункту", max_length=50)
    post_number = models.IntegerField("Номер відділення Нової Пошти", max_length=25)
    payment_method = models.CharField("Спосіб оплати", max_length=18, default="Накладний платіж")
    date = models.DateField("Дата замовлення", default=datetime.date.today())

    def __str__(self):
        return self.surname

    class Meta:
        verbose_name ='Замовлення'
        verbose_name_plural = 'Замовлення'
```

Рисунок 3.37 – Клас *Orders* з файлу models.py

Також у класі моделі замовлень є функція *__str__* з параметром *self* , вона реалізує повернення прізвищ користувачів у відображенні таблиці замовлень у базі даних(рис. 3.38), а також підклас *Meta*, що встановлює назву таблиці у базу даних в множині та однині.

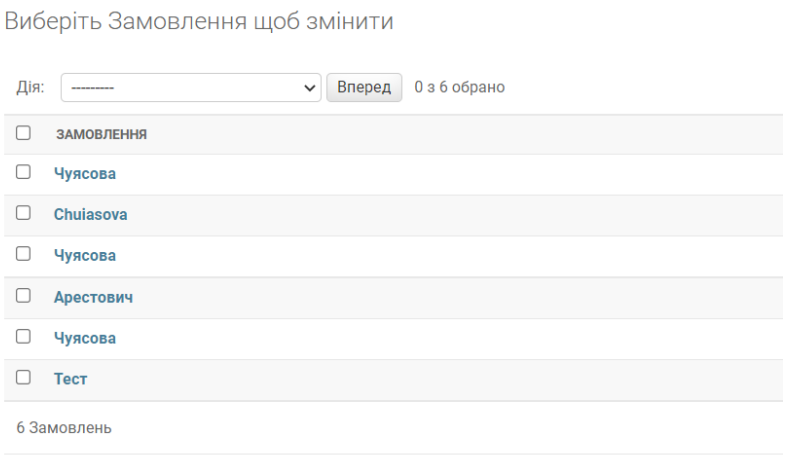


Рисунок 3.38 – Відображення таблиці замовлень на панелі адміністрування

На рис 3.39 можна побачити як виглядає замовлення окремого користувача, коли воно надходить до бази даних.

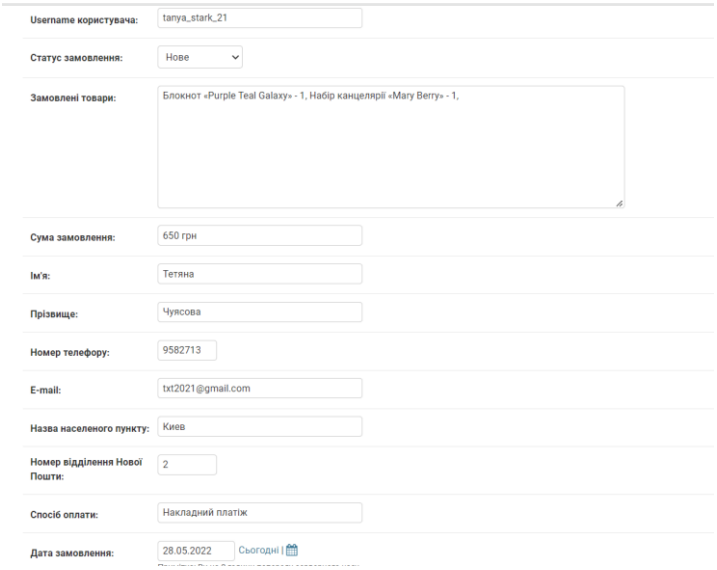


Рисунок 3.39 – Відображення замовленя користувача на панелі адміністрування

Адміністратор може змінювати статус замовлення, зберігати зміни у ньому та видаляти. Після надходження замовлення до бази даних, адміністратор зв’язується з замовленням через мобільний телефон, уточнює всі деталі замовлення та перевіряє їх коректність, після чього поченається комплектація та відправка. Якщо замовник вказав накладний платіж, то оплачувати замовлення він буде при отриманні його у відділенні Нової Пошти, з відсотком пошти. Якщо користувач вказав оплату шляхом переказу на картку, то адміністратор надсилає замовнику реквізити на оплату, чекає підтвердження у вигляді скріншоту від замовника, тоді після цього починається процес комплектації та відправки замовлення.

3.3.3 Модель речей

Дана модель була створена для можливості додавати нові товари у базу даних, прибирати старі, а також змінювати кількість товару при перерахунку на складі, тощо. Модель речей також являє собою клас під назвою *Items* у файлі models.py(рис. 3.40). Вона має подібну структу до моделі замовлень, тобто це поля моделі, що будуть відображатися в таблиці бази даних, функція *__str__* з

параметром *self* , що повертає прізвища користувачів при відображенні таблиці замовлень у базі даних, а також підклас *Meta*, котрий назву таблиці у базу даних в множині та однині. Дана модель має лише чотири поля:

- item_id – унікальний номер товару;
- name – назва товару;
- price – ціна товару;
- amount – кількість даного товару на складі;

```
class Items(models.Model):
    item_id = models.CharField("Код товару", max_length=10)
    name = models.CharField("Назва товару", max_length=300)
    price = models.IntegerField("Ціна за одиницю")
    amount = models.IntegerField("Кількість на складі" )

    def __str__(self):
        return self.name

    class Meta:
        verbose_name ='Товар'
        verbose_name_plural = 'Товари'
```

Рисунок 3.40 – Клас *Items* з файлу models.py

На рис. 3.41 показано як кожен товар відображається на панелі адміністрування

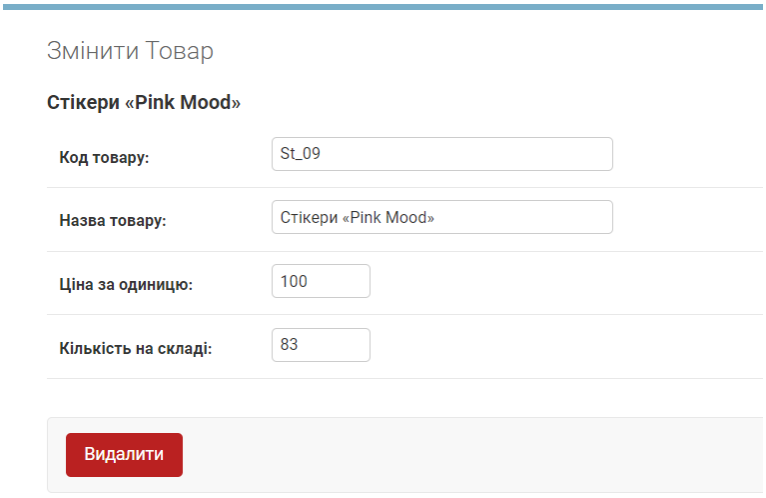


Рисунок 3.41 – Відображення інформації про товар на панелі адміністратора

Після створення кожної моделі, або внесенні до них змін створюються міграційні файли (команда *python manage.py makemigrations*) та проводяться міграції(команда *python manage.py migrate*). Завдяки цьому зміни відображаються у базі даних.

Щоб нова модель, яка була створена, відображалась на панелі адміністратора, її обов'язково потрібно зареєструвати у файлі admin.py додатку main. Робиться це наступним чином(рис. 3.42).

```
from django.contrib import admin
from .models import Orders,Items

admin.site.register(Orders)
admin.site.register(Items)
```

Рисунок 3.42 – Реєстрація моделі у файлі admin.py

Після проведення даної операції моделі відображаються на панелі адміністратора(рис. 3.43), де можна додавати до них нові записи.

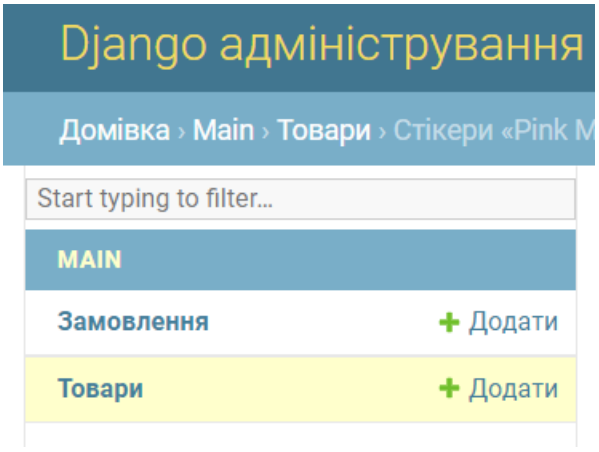


Рисунок 3.43 – Відображення моделей на панелі адміністрування

Створивши всі моделі зв'язавши їх з базою даних та панеллю адміністрування, можна переходити до тестування додатку.

ВИСНОВОК ДО РОЗДІЛУ 3

Результатом роботи над даним розділом є розробка системи, яка являє собою веб-додаток для продажу та реклами товарів канцелярського призначення та відповідає всім вимогам, описаним раніше у попередніх розділах. У даному розділі було розроблено:

- Дизайн веб-додатку, наведо скріншоти макетів UI дизайну.
- Клієнтську частину додатку, всі веб-сторінки, їх розмітка, стилі, функції та зв’язки між собою, наведено приклади коду у вигляді скріншотів, а також описано створені методи та функції у відповідних таблицях.
- Серверну частину веб-додатку, моделі, вигляди, форми, прописано шляхи доступу до сорінок, налаштовано додаток на підтримку української мови, наведено код реалізації відповідних класів у вигляді скріншотів коду, скріншотів панелі адміністрування, атакож таблиць опису методів.
- Базу даних для роботи з замовленнями користувачів, реалізовано відображення відповідних таблиць у панелі адміністрвання, функціонал реєстрації та авторизації користувачів, описано класи моделів у відповідних підрозділах, наведено приклади роботи панелі адміністрування та коду у вигляді скріншотів.

РОЗДІЛ 4
ТЕСТУВАННЯ ТА АНАЛІЗ РОБОТИ РОЗРОБЛЕНОЇ СИСТЕМИ

Тестування розробленої системи є дуже важливою частиною розробки, саме це допомагає оцінити якість створеного програмного продукту. У даному розділі буде описано методи тестування розробленого додатку та його результати, також проведено аналіз роботи веб-застосунку.

4.1 Тестування та результати роботи веб-додатку

Для аналізу роботи розробленого веб-додатку має бути проверено функціональне тестування, яке покриватиме більшу частину функціоналу застосунку. Основні аспекти веб-додатку які мають бути протестовані у користувацькій частині це:

- Реєстрація користувача;
- Авторизація користувача;
- Фільтрація товарів на сторінці “Блокноти”;
- Фільтрація товарів на сторінці “Інше”;
- Можливість користувача додати товари в корзину;
- Можливість користувача очистити корзину;
- Можливість користувача оформити замовлення;
- Можливість користувача зайти до особистого кабінету;
- Можливість користувача вийти зі свого облікового запису;

Основні аспекти веб-додатку які мають бути протестовані у частині адміністрування це:

- Відображення зроблених користувачем замовлень;

– Відображення всіх зареєстрованих користувачів;
– Відображений наявних у продажу товарів;
– Можливість адміністратора змінювати статус замовлення;
– Можливість адміністратора видалити замовлення;
– Можливість адміністратора видалити користувача з системи;
– Можливість адміністратора змінити кількість наявного певного товару;
– Можливість адміністратора додати новий товар у систему;

Для покриття тестуванням всього вищезазначеного функціоналу, потрібно написати тесткейси з вказанням передумов тестування, кроків відтворення та очікуваного результату. Під час тестувння буде виконано System testing з використанням GreyBox і Blackbox технік, а також Smoke тестування та Critical Path тестування. Буде проведенно positive тестування системи, адже для negative testing потребує значно більших ресурсів часу та робочих рук. У нижченаведених таблицях 4.1 – 4.17 створено тесткейси для покриття тестуванням вищезазначеного функціоналу створеного веб-додатку.

Таблиця 4.1 – TC001

Назва тесткейсу	Реєстрація користувача
Передумови виконання	Користувач зайшов у систему як гість, не авторизований
Кроки відтворення	1. Натиснути кнопку “Зареєструватися”. 2. Ввести валідний логін. 3. Ввести валідний пароль. 4. Повторити пароль. 5. Натиснути на кнопку “Зареєструватися”.
Очікуваний результат	Відкривається сторінка авторизації

Таблиця 4.2 – ТС002

Назва тесткейсу	Авторизація користувача
Передумови виконання	Користувач зайшов у систему як гість, не авторизований, пройшов процедуру реєстрації, відкрито сторінку авторизації
Кроки відтворення	1. Ввести валідне ім’я користувача. 2. Ввести валідний пароль. 3. Натиснути на кнопку “Увійти”.
Очікуваний результат	Відкривається головна сторінка додатку, біля іконки корзини написано ім’я користувача

Таблиця 4.3 – ТС003

Назва тесткейсу	Фільтрація товарів на сторінці “Блокноти”
Передумови виконання	Користувач авторизувався у системі, відкрито головну сторінку додатку
Кроки відтворення	1. Натиснути на “Блокноти” у меню навігації 2. Обрати фільтр “Чорний”
Очікуваний результат	У полі для відображенні товарів показано лише блокноти чорного кольору

Таблиця 4.4 – ТС004

Назва тесткейсу	Фільтрація товарів на сторінці “Інше”
Передумови виконання	Користувач авторизувався у системі, відкрито головну сторінку додатку
Кроки відтворення	1. Натиснути на “Інше” у меню навігації 2. Обрати фільтр “Стікери”
Очікуваний результат	У полі для відображенні товарів показано лише стікери

Таблиця 4.5 – ТС005

Назва тесткейсу	Додавання товарів в корзину
Передумови виконання	Користувач авторизувався у системі, відкрито головну сторінку додатку
Кроки відтворення	1. Натиснути на “Інше” у меню навігації 2. Обрати фільтр “Стікери” 3. Натиснути на зображення першого товару 4. Натиснути кнопку “Купити” 5. Натиснути на іконку корзини
Очікуваний результат	Відкривається сторінка корзини, у ній відображено назву, вартість та кількість раніше обраного товару, а також загальну кількість замовлення

Таблиця 4.6 – ТС006

Назва тесткейсу	Очистка корзини
Передумови виконання	Користувач авторизувався у системі, відкрито сторінку корзини, у корзині є товар
Кроки відтворення	1. Натиснути на кнопку “Очистити корзину”
Очікуваний результат	Всі речі з корзини видалено

Таблиця 4.7 – ТС007

Назва тесткейсу	Оформлення замовлення
Передумови виконання	Користувач авторизувався у системі, відкрито сторінку корзини, у корзині є товар
Кроки відтворення	1. Натиснути на кнопку “Оформити замовлення” 2. Ввести валідні дані у форму, що відкрилась 3. Натиснути кнопку “Підтвердити замовлення”
Очікуваний результат	Відкривається сторінка з підтвердженням успішного оформлення замовлення

Таблиця 4.8 – ТС008

Назва тесткейсу	Вхід до особистого кабінету
Передумови виконання	Користувач авторизувався у системі, відкрито головну сторінку додатку
Кроки відтворення	1. Натиснути на кнопку ім’я користувача біля іконки корзини
Очікуваний результат	Відкривається сторінка персонального кабінету користувача

Таблиця 4.9 – ТС009

Назва тесткейсу	Вихід з облікового запису
Передумови виконання	Користувач авторизувався у системі, відкрито головну сторінку додатку
Кроки відтворення	1. Натиснути на ім’я користувача біля іконки корзини 2. Натиснути на кнопку “Вийти”
Очікуваний результат	Сторінка оновлюється, біля іконки корзини з’являється напис “Зареєструватися” замість імені користувача

Таблиця 4.10 – ТС010

Назва тесткейсу	Відображення зроблених користувачем замовлень
Передумови виконання	Адміністратор зайшов до панелі адміністрування, відкрито головну сторінку Django адміністрування
Кроки відтворення	1. Натиснути на “Замовлення” у секції “main”
Очікуваний результат	Відкривається таблиця з усіма замовленнями, зробленими користувачами

Таблиця 4.11 – ТС011

Назва тесткейсу	Відображення всіх зареєстрованих користувачів
Передумови виконання	Адміністратор зайшов до панелі адміністрування, відкрито головну сторінку Django адміністрування
Кроки відтворення	1. Натиснути на “Користувачі” у секції “Аутентифікація та авторизація”
Очікуваний результат	Відкривається таблиця з усіма зареєстрованими в системі користувачами

Таблиця 4.12 – ТС012

Назва тесткейсу	Відображення наявних у продажу товарів
Передумови виконання	Адміністратор зайшов до панелі адміністрування, відкрито головну сторінку Django адміністрування
Кроки відтворення	1. Натиснути на “Товари” у секції “ main ”
Очікуваний результат	Відкривається таблиця з усіма наявними в системі товарами

Таблиця 4.13 – ТС013

Назва тесткейсу	Зміна статусу замовлення
Передумови виконання	Адміністратор зайшов до панелі адміністрування, відкрито головну сторінку Django адміністрування
Кроки відтворення	1. Натиснути на “Замовлення” у секції “main” 2. Натиснути на перше прізвище у таблиці 3. Натиснути на “Статус замовлення” та обрати “Прийнято” 4. Натиснути на “Зберегти” 5. Натиснути на перше прізвище у таблиці
Очікуваний результат	Статус замовлення змінився на “Прийнято”

Таблиця 4.14 – ТС014

Назва тесткейсу	Видалення замовлення
Передумови виконання	Адміністратор зайшов до панелі адміністрування, відкрито головну сторінку Django адміністрування
Кроки відтворення	1. Натиснути на “Замовлення” у секції “main” 2. Натиснути на перше прізвище у таблиці 3. Натиснути на кнопку “Видалити” 4. Натиснути на кнопку “Так, я впевнений”
Очікуваний результат	У таблиці замовлень зникло видалене замовлення

Таблиця 4.15 – ТС015

Назва тесткейсу	Видалення користувача
Передумови виконання	Адміністратор зайшов до панелі адміністрування, відкрито головну сторінку Django адміністрування
Кроки відтворення	1. Натиснути на “Користувачі” у секції “Аутентифікація та авторизація” 2. Натиснути на перше ім’я користувача у таблиці 3. Натиснути на кнопку “Видалити” 4. Натиснути на кнопку “Так, я впевнений”
Очікуваний результат	У таблиці “Користувачі” зникло видалене ім’я користувача

Таблиця 4.16 – ТС016

Назва тесткейсу	Зміна кількості наявного товару
Передумови виконання	Адміністратор зайшов до панелі адміністрування, відкрито головну сторінку Django адміністрування
Кроки відтворення	1. Натиснути на “Товари” у секції “ main ” 2. Натиснути на перший товар у таблиці 3. Збільшити на одиницю кількість товару у полі кфлькість на “складі”

Продовження таблиці 4.16 – ТС016

Кроки відтворення	4. Натиснути на кнопку “Зберегти” 5. Натиснути на перший товар у таблиці “Товари”
Очікуваний результат	Кількість наявного на складі даного товару збільшилась на одиницю

Таблиця 4.17 – ТС017

Назва тесткейсу	Додавання товару
Передумови виконання	Адміністратор зайшов до панелі адміністрування, відкрито головну сторінку Django адміністрування
Кроки відтворення	1. Натиснути на “Товари” у секції “main” 2. Натиснути на кнопку “Додати товар” 3. Заповнити всі поля валідними даними 4. Натиснути на кнопку “Зберегти” 5. Повернутися до таблиці “Товари”
Очікуваний результат	У таблиці “Товари” відображається доданий товар

Таблиця 4.18 – ТС018

Назва тесткейсу	Видалення товару
Передумови виконання	Адміністратор зайшов до панелі адміністрування, відкрито головну сторінку Django адміністрування
Кроки відтворення	1. Натиснути на “Товари” у секції “main” 2. Натиснути на перший товар у таблиці 3. Натиснути на кнопку “Видалити” 4. Натиснути на кнопку “Так, я впевнений”
Очікуваний результат	У таблиці “Товари” зник видалений товар

За описаними вище тесткейсами було проведено тестування веб-додатку, результати даного тестування наведено у таблиці 4.19, де статус тесткейсу “Pass” означає, що він пройшов успішно.

Таблиця 4.19 – Результати тестувння веб-додатку

Номер тесткейсу	Очікуваний результат	Реальний результат	Статус тесткейсу
TC001	Відкривається сторінка авторизації	Відкривається сторінка авторизації	Pass
TC002	Відкривається головна сторінка додатку, біля іконки корзини написано ім’я користувача	Відкривається головна сторінка додатку, біля іконки корзини написано ім’я користувача	Pass
TC003	У полі для відображенні товарів показано лише блокноти чорного кольору	У полі для відображенні товарів показано лише блокноти чорного кольору	Pass
TC004	У полі для відображенні товарів показано лише стікери	У полі для відображенні товарів показано лише стікери	Pass
TC005	Відкривається сторінка корзини, у ній відображено назву, вартість та кількість раніше обраного товару, а також загальну кількість замовлення	Відкривається сторінка корзини, у ній відображено назву, вартість та кількість раніше обраного товару, а також загальну кількість замовлення	Pass
TC006	Всі речі з корзини видалено	Всі речі з корзини видалено	Pass
TC007	Відкривається сторінка з підтвердженням успішного оформлення замовлення	Відкривається сторінка з підтвердженням успішного оформлення замовлення	Pass
TC008	Відкривається сторінка персонального кабінету користувача	Відкривається сторінка персонального кабінету користувача	Pass
TC009	Сторінка оновлюється, біля іконки корзини з’являється напис “Зареєструватися” замість імені користувача	Сторінка оновлюється, біля іконки корзини з’являється напис “Зареєструватися” замість імені користувача	Pass
TC010	Відкривається таблиця з усіма замовленями, зробленими користувачами	Відкривається таблиця з усіма замовленями, зробленими користувачами	Pass

Продовження таблиці 4.19 – Результати тестувння веб-додатку

Номер тесткейсу	Очікуваний результат	Реальний результат	Статус тесткейсу
TC011	Відкривається таблиця з усіма зареєстрованими в системі користувачами	Відкривається таблиця з усіма зареєстрованими в системі користувачами	Pass
TC012	Відкривається таблиця з усіма наявними в системі товарами	Відкривається таблиця з усіма наявними в системі товарами	Pass
TC013	Статус замовлення змінився на “Прийнято”	Статус замовлення змінився на “Прийнято”	Pass
TC014	У таблиці замовлень зникло видалене замовлення	У таблиці замовлень зникло видалене замовлення	Pass
TC015	У таблиці “Користувачі” зникло видалене ім’я користувача	У таблиці “Користувачі” зникло видалене ім’я користувача	Pass
TC016	Кількість наявного на складі даного товару збільшилась на одиницю	Кількість наявного на складі даного товару збільшилась на одиницю	Pass
TC017	У таблиці “Товари” відображається доданий товар	У таблиці “Товари” відображається доданий товар	Pass
TC018	У таблиці “Товари” зник видалений товар	У таблиці “Товари” зник видалений товар	Pass

Як можна побачити з вищенаведеної таблиці, всі тесткейси пройшли успішно, отже розроблений веб-додаток має достатній рівень якості.

4.2 Аналіз роботи та захист веб-додатку

Як видно з попереднього підрозділу, розроблений веб-додаток відповідає зазначеним раніше вимогам та проходить позитивне тестування.

Розроблений інтерфейс приємний, користувачу легко працювати з застосунком, адміністратор, у свою чергу без проблем має змогу приймати та керувати замовленнями.

Веб-застосунок працює швидко та виконує всі свої основні функції. Нижче буде приведено зображення інтерфейсу веб-додатку (рис.4.1 – 4.13) при специфічних діях користувача.

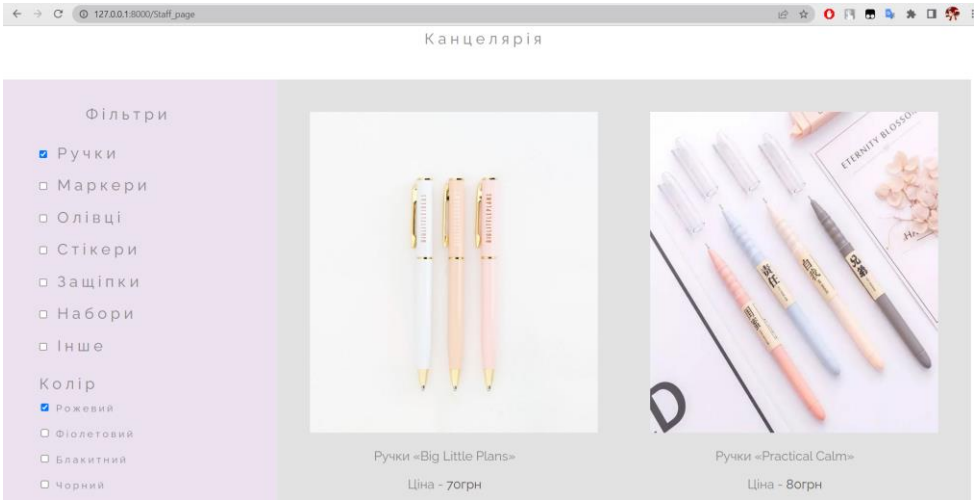


Рисунок 4.1 – Обрання декількох фільтрів

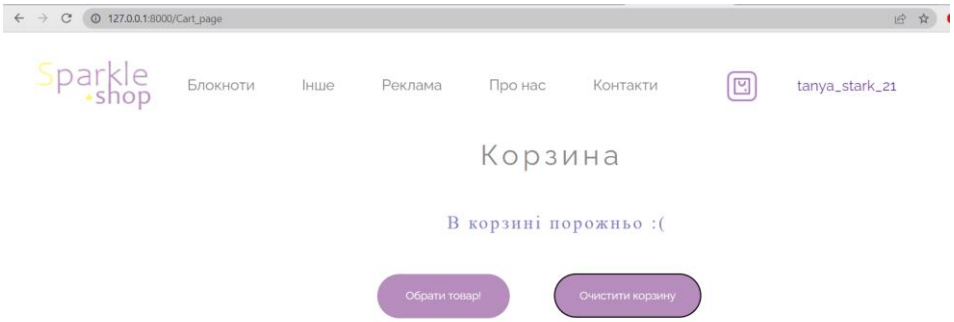


Рисунок 4.2 – Відсутність товару в корзині

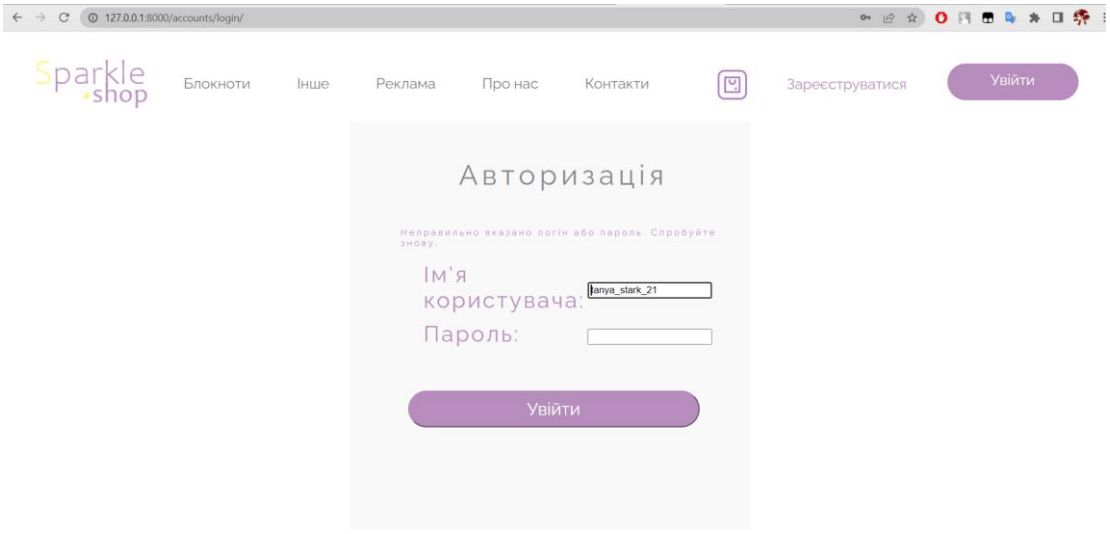


Рисунок 4.3 – Неправильно вказано дані для авторизації

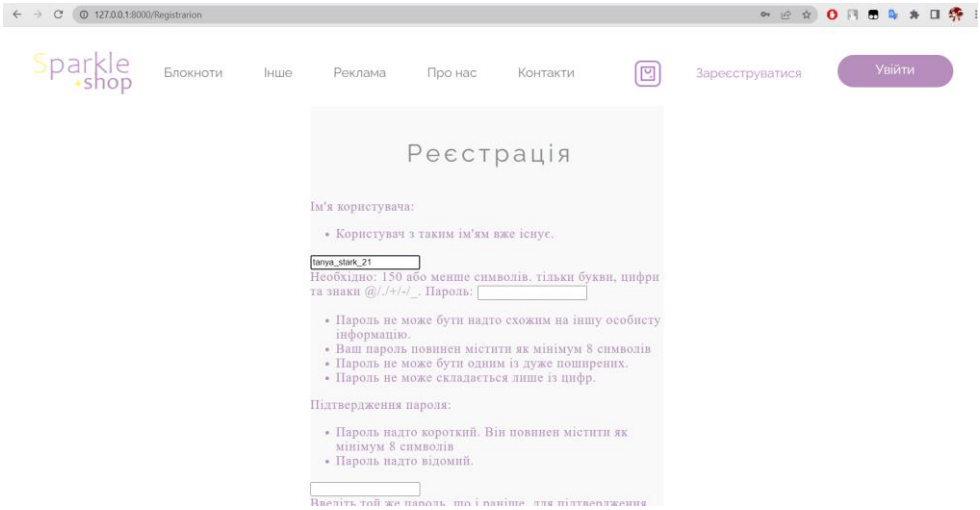


Рисунок 4.4 – Введення невалідних даних у поля реєстрації

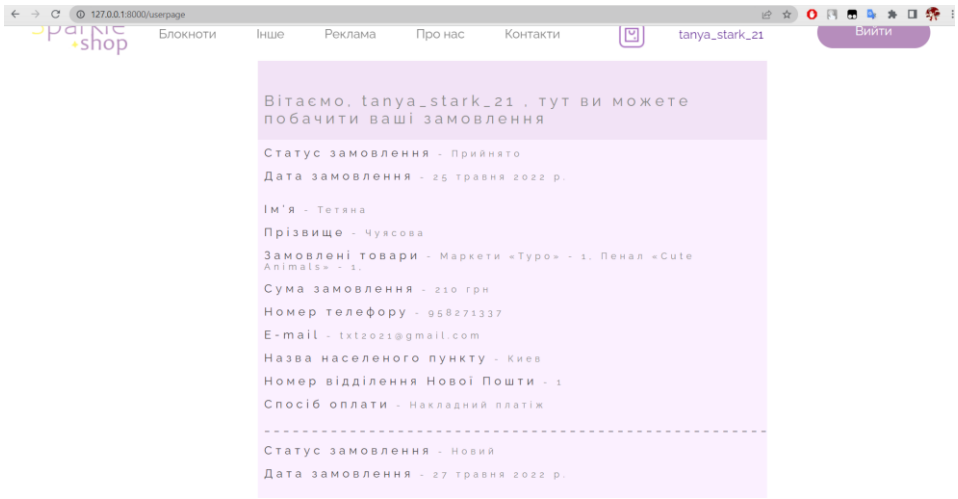


Рисунок 4.5 – Особистий кабінет користувача при оформленні кількох замовлень

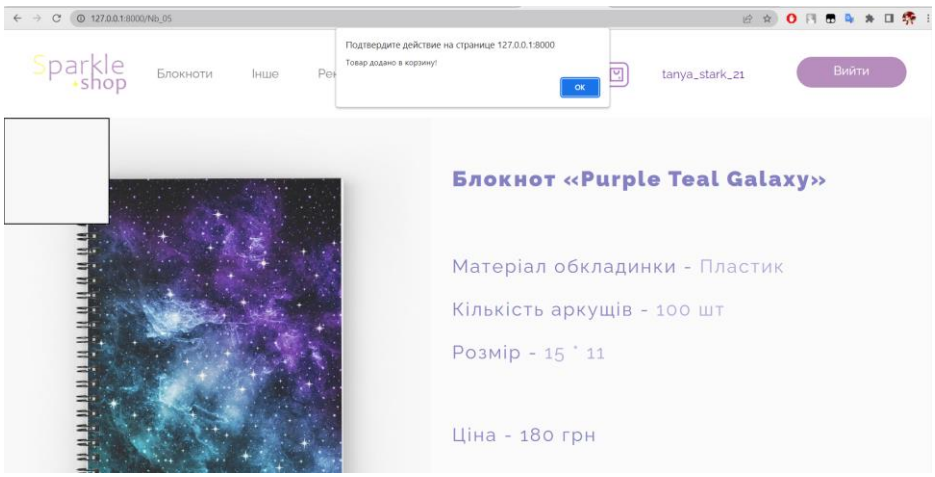


Рисунок 4.6 – Стан сторінки товару при додаванні його в корзину

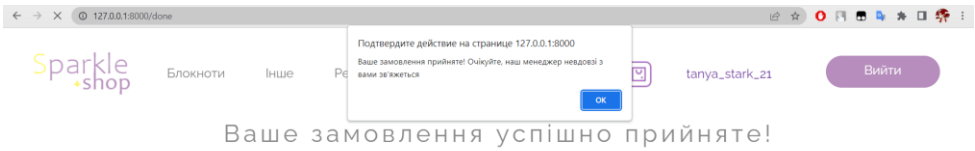


Рисунок 4.7 – Сторінка підтвердження оформлення замовлення

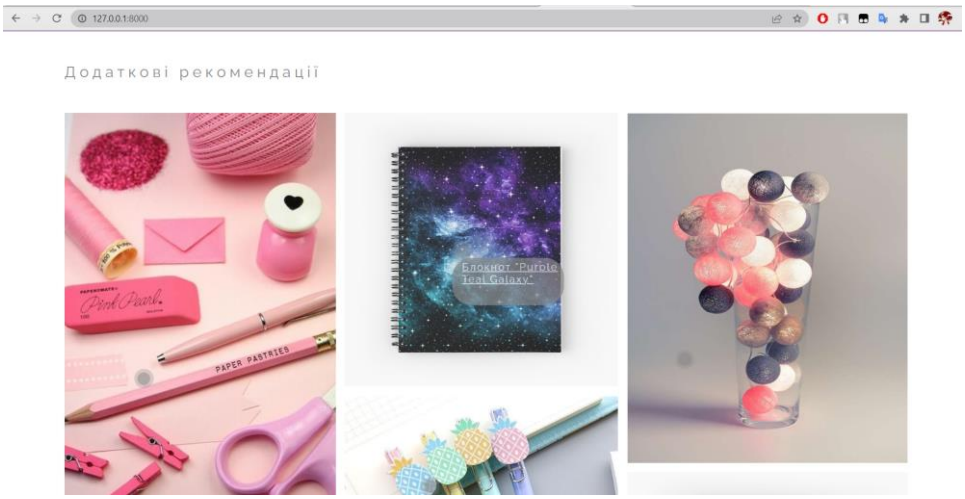


Рисунок 4.8 – Поява повідомлення з посиланням на сторінку товару після наведення на знак-підказку на зображенні

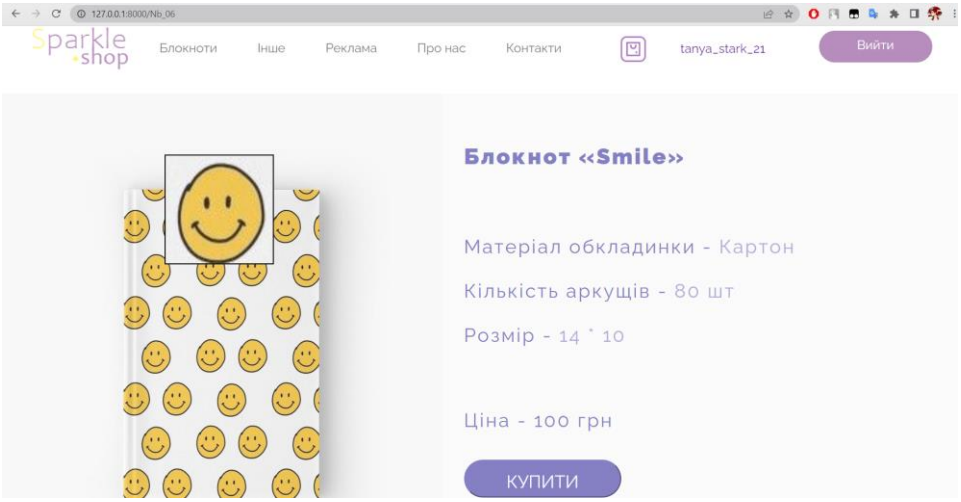


Рисунок 4.9 – Приклад роботи зума на зображенні товару

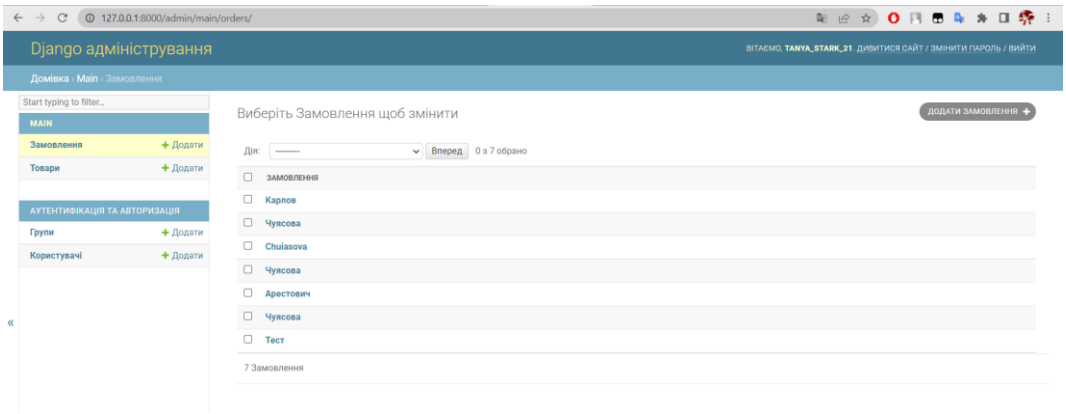


Рисунок 4.10 – Таблиця замовлень на панелі адміністрування

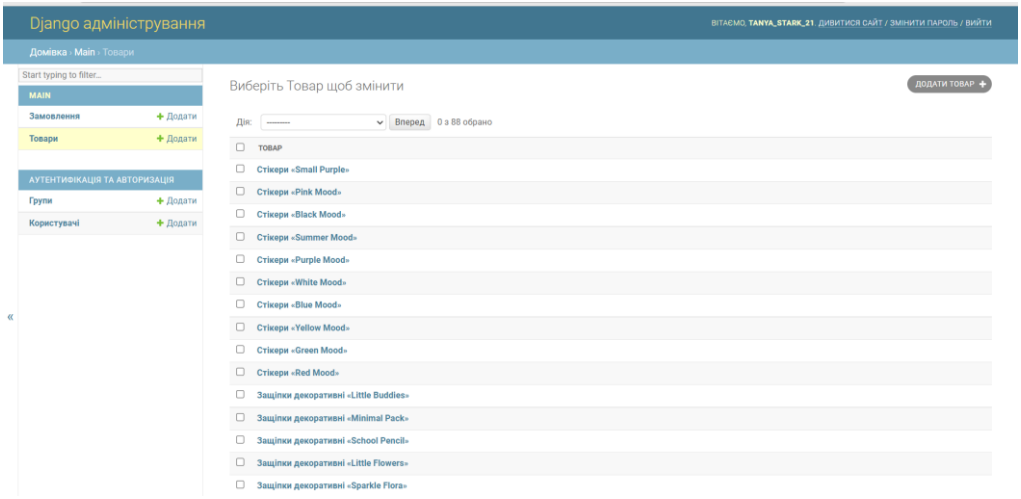


Рисунок 4.11 – Таблиця товарів на панелі адміністрування

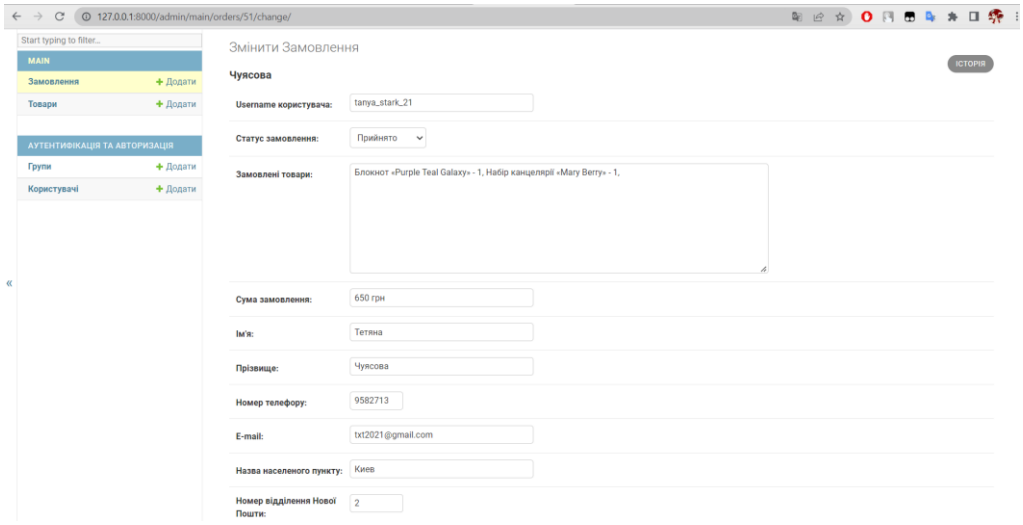


Рисунок 4.12 – Форма замовлення на стороні адміністратора

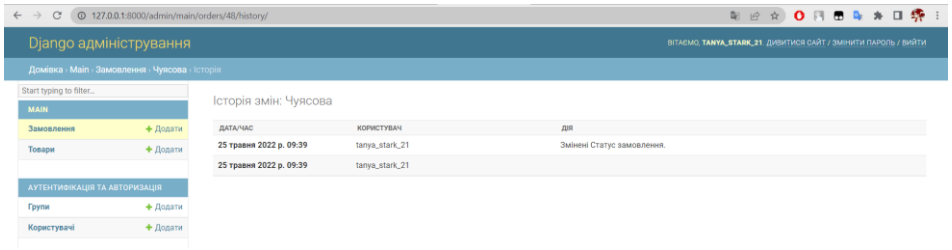


Рисунок 4.13 – Історія зміни замовлення

Аналізуючи проведену роботу по створенню даного веб-застосунку можна сказати, що він має складну багаторівневу структуру, всі компоненти якої гарно співпрацюють між собою. Використовувати даний веб-застосунок можна як інтернет-магазин з інтуїтивно зрозумілим інтерфейсом та порядком дій.

Можна додати декілька слів про захист даних у створеному веб-додатку. Django надає велику кількість вбудованих захисних механізмів, які використовуються у розроблеому застосунку, серед них:

– Система шаблонів з захистом від міжсайтового скриптингу, тобто зловнисники не зможуть внідрити шкідливий скрипт через форми на сторінках сайту;
– Захист від міжсайтової підробки запистів, котрий реалізується за допомогою додавання у форми {% csrf_token %}. Даний механізм не дозволить зловмиснику надісліти певний запит від чужого імені;

- Механізм захисту від sql ін'єкцій не дозволить зловмиснику виконати довільний sql-код та отримати доступ до бази даних;
- Механізм захисту від Клікджекінгу, тобто перехвату вводу даних на сторніці, та перенаправлення на іншу, яку контролює зловмисний. Реалізований за допомогою проміжного програмного забезпечення X-Frame-Options;
- Механізм шифрування трафіку між сервером та користувачем, використовуючи SSL/HTTPS;
- Шифрування явного вигляду паролів користувчів на адмін-панелі.

4.3 Рекомендації щодо розвитку та вдосконалення додатка

Розроблений у даному проєкті веб-додаток має гарний потенціал та можливості для вдосконалення. Завдяки використанню могутнього фреймворку Django можна запроваджувати велику кількість корисних додаткових функцій. Нариклад, можна додати пошук по додатку або наприклад додаткові можливості корзини чи фільтри товарів. Даний веб-додаток знаходиться на локальному сервері, котрий предоставляє Django фреймворк, тому як вдосконалення можна запропонувати розташування додатку на віддаленому сервері, тощо.

Даний програмний продукт також містить швидкодіючу базу-даних, котру теж можна розширити, наприклад додавши нові категорії товару, або додаткові форми чи можливості для створення та оформлення користувачем замовлення.

З розвитком модних тенденцій дизайну інтерфейсу користувача, можна також змінювати зовнішній вигляд веб-додатку, а під час свят чи знижок запроваджувати відповідні святкові картинки на лендинг-банерах.

ВИСНОВОК ДО РОЗДІЛУ 4

У даному розділі представлено роботу повністю готового веб-застосунку, що розроблювався протягов всього виконання дипломної роботи.

Проведено тестування застосунку, використано для цього різні техніки. Наведено результати тестування веб-додатку відповідно створеним наведеними тесткейсам. Основним завданням даного застосунку було реалізація оформлення замовлення різних товарів для користувачів та відповідно менеджмент цих замовлень для адміністратора. Веб-застосунок безперечно виконую вищезазначену задачу.

Наведено скріншоти роботи веб-додатку у різних випадках його використання, які можуть виникати при користувацькому досвіді. Проведено аналіз роботи застосунку та його відповідність вимогам, котрі були визначені на початку роботи.

Наприкінці даного розділу описано перспективи розвитку даного веб-додатку, різноманітні ідеї для його вдосконалення та використання у майбутньому. Запропоновано рекомендації щодо вдосконалення розробленого веб-застосунку.

ВИСНОВКИ

У ході виконання даної дипломної роботи було розроблено систему, яка являє собою веб-додаток для продажу та реклами товарів канцелярського призначення. Даний застосунок дає змогу користувачам робити замовлення, а адміністратору оперувати та сладкувати за цими замовленнями, також вести обіг товарів та зареєстрованих користувачів. Розроблений веб-додаток також можна використовувати для розміщення на окремій веб-сторінці реклами рзноманітних Інстаграм магазинів.

У першому розділі було розглянуто поняття веб-додатку в цілому, описано їх види та принципи роботи, актуальність різноманітних веб-додатків на сьогоднішній день. Описано структуру веб-застосунків, їх взаємодію з Всесвітньою мережею Інтернет. Проведено огляд основних засобів для створення веб-додатків, а також висвітлено основні проблеми їх створення та підтримки.

У другому розділі описано різноманітні технології, що найчастіше використовуються сьогодні для розробки веб-застосунків. Визначено їх переваги та недоліки, показано які технології краще підходять для розробки користувацької частини додатку, а які для серверної. Обрано стек технологій для розробки власного веб-застосунку та обгрунтовано даний вибір.

У третьому розділі описано розроблення веб-додатку по заданій темі з використання раніше обраного стеку технологій. Детально показано етапи створення веб-застосунку, описано створення користувацької та серверної частини додатко. Надано опис різноманітних класів та функцій, а також предоставлено скріншоти з демонструванням поетапного створення проєкту.

У четвертому описано процес проведення тестування розробленого веб-застосунку. Описано техніки тестування, які використовувалися, створено тесткейси для різноманітних випадків використання додатку. Протестовано

застосунок та наведено його результати по всім окремо написаним тесткейсам. Проведено аналіз роботи вже готового застоснку, наведено результати роботи веб-додатку у вигляді скріншотів під час різноманітних випадків його використання. Описано різноманітні механізми захисту даних, які використовуються у розробленому веб-додатку. Також надано рекомендаціїї щодо подальшого розвитку даного веб-додатку, та ідеї щодо його покращення у майбутньому.

Отже готовий програмний продукт, що розроблено у даній роботі, повністю відповідає початково визначеним вимогам, та виконує свої основні функції, заради яких його було створено.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Uryutin O. A brief history of web app [Електронний ресурс] / Oleg Uryutin – Режим доступу до ресурсу: https://oleg-uryutin.medium.com/a-brief-history-of-web-app-50d188f30d (дата звернення 21.05.2022).
2. Классификация Web-технологий [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/6008269/page:3/ (дата звернення 21.05.2022).
3. Основні поняття та класифікація Web-технологій [Електронний ресурс] – Режим доступу до ресурсу: https://studopedia.ru/16_7807_osnovni-ponyattya-ta-klasifikatsiya-Web-tehnologiy.html (дата звернення 21.05.2022).
4. Інформаційні технології [Електронний ресурс] – Режим доступу до ресурсу: https://pidru4niki.com/1497110247709/informatika/informatsiyni_tehnologiyi (дата звернення 21.05.2022).
5. Створення та редагування веб-ресурсів для розширення застосунку [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/uk-ua/dynamics365/customerengagement/on-premises/customize/create-edit-web-resources?view=op-9-1 (дата звернення 21.05.2022).
6. Web Technology [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/web-technology/ (дата звернення 21.05.2022).
7. John S. Quoterman. The Matrix: Computer Networks and Conferencing Systems Worldwide. – 1990. (дата звернення 21.05.2022).

8. Структура и основные принципы работы сети Интернет [Електронний ресурс] – Режим доступу до ресурсу: http://mir.it-karma.ru/set-internet/lekcii/struktura-i-osnovnye-principy-raboty-seti-internet (дата звернення 21.05.2022).
9. What is the web server? [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server (дата звернення 21.05.2022).
10. Статические и динамические сайты сегодня: какие лучше и почему [Електронний ресурс] – Режим доступу до ресурсу: https://jino.ru/journal/articles/staticheskie-dinamicheskie-sayty/ (дата звернення 21.05.2022).
11. Web application (web app) [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app (дата звернення 22.05.2022).
12. Одностраничные(spa) и многостраничные(pwa) веб-приложения [Електронний ресурс] – Режим доступу до ресурсу: https://vc.ru/seo/108149-odnostranichnye-spa-i-mnogostranichnye-pwa-veb-prilozheniya (дата звернення 22.05.2022).
13. Difference Between Web application and Website [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/difference-between-web-application-and-website/ (дата звернення 22.05.2022).
14. HTML basics [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics (дата звернення 22.05.2022).

15. What is CSS [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/what-is-css (дата звернення 22.05.2022).
16. What is UI design? What is UX design? UI vs UX: What's the difference [Електронний ресурс] – Режим доступу до ресурсу: https://uxplanet.org/what-is-ui-vs-ux-design-and-the-difference-d9113f6612de (дата звернення 22.05.2022).
17. What Is Content Management System (CMS)? [Електронний ресурс] – Режим доступу до ресурсу: https://kinsta.com/knowledgebase/content-management-system/ (дата звернення 22.05.2022)
18. 7 challenges in web application development [Електронний ресурс] – Режим доступу до ресурсу: https://tillerdigital.com/blog/7-challenges-in-web-application-development/ (дата звернення 22.05.2022).
19. Архитектура веб приложения: компоненты, слои и типы [Електронний ресурс] – Режим доступу до ресурсу: https://itanddigital.ru/webapplications (дата звернення 22.05.2022).
20. SINGLE PAGE APPLICATION (SPA) И MULTI PAGE APPLICATION (MPA): ПРЕИМУЩЕСТВА И НЕДОСТАТКИ [Електронний ресурс] – Режим доступу до ресурсу: https://merehead.com/ru/blog/single-page-application-vs-multi-page-application/ (дата звернення 22.05.2022).
21. Создание приложений с бессерверными архитектурами [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ru/lambda/serverless-architectures-learn-more/ (дата звернення 23.05.2022).
22. Sanderzinewicz A. Микросервисный подход в веб-разработке: micro frontends [Електронний ресурс] / Alex Sanderzinewicz – Режим доступу до ресурсу: https://medium.com/@aleksanderzinewicz/ (дата звернення 23.05.2022).

23. 11 Best Front-End Technologies To Use In 2021 [Електронний ресурс] – Режим доступу до ресурсу: https://www.trio.dev/blog/front-end-technologies (дата звернення 23.05.2022).
24. The Advantages and Disadvantages of JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/the-advantages-and-disadvantages-of-javascript/ (дата звернення 23.05.2022).
25. jQuery Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/jquery/jquery_intro.asp (дата звернення 23.05.2022).
26. Introduction to Angular concepts [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/guide/architecture (дата звернення 23.05.2022).
27. What are the Advantages and Disadvantages of Angular? [Електронний ресурс] – Режим доступу до ресурсу: https://www.edureka.co/blog/advantages-and-disadvantages-of-angular/ (дата звернення 23.05.2022).
28. Advantages and Disadvantages of ASP.NET [Електронний ресурс] – Режим доступу до ресурсу: https://www.software-developer-india.com/advantages-and-disadvantages-of-asp-net/ (дата звернення 23.05.2022).
29. Python for backend development[Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/techvariable/python-for-backend-development-bbdd4c0cf041 (дата звернення 23.05.2022).
30. Django Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction (дата звернення 23.05.2022).

31. Introduction to Java Web development - Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.vogella.com/tutorials/JavaWebTerminology/article.html (дата звернення 23.05.2022).
32. Why Should You Use Java For Your Backend Infrastructure? [Електронний ресурс] – Режим доступу до ресурсу: https://www.devteam.space/blog/why-should-you-use-java-for-your-backend-infrastructure/ (дата звернення 23.05.2022).
33. Spring Framework [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/projects/spring-framework (дата звернення 23.05.2022).
34. A tour of the C# language [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ (дата звернення 23.05.2022).
35. Pros and Cons of Using C# as Your Backend Programming Language [Електронний ресурс] – Режим доступу до ресурсу: https://www.agilites.com/pros-and-cons-of-using-c-as-your-backend-programming-language.html (дата звернення 23.05.2022).

ДОДАТОК 1

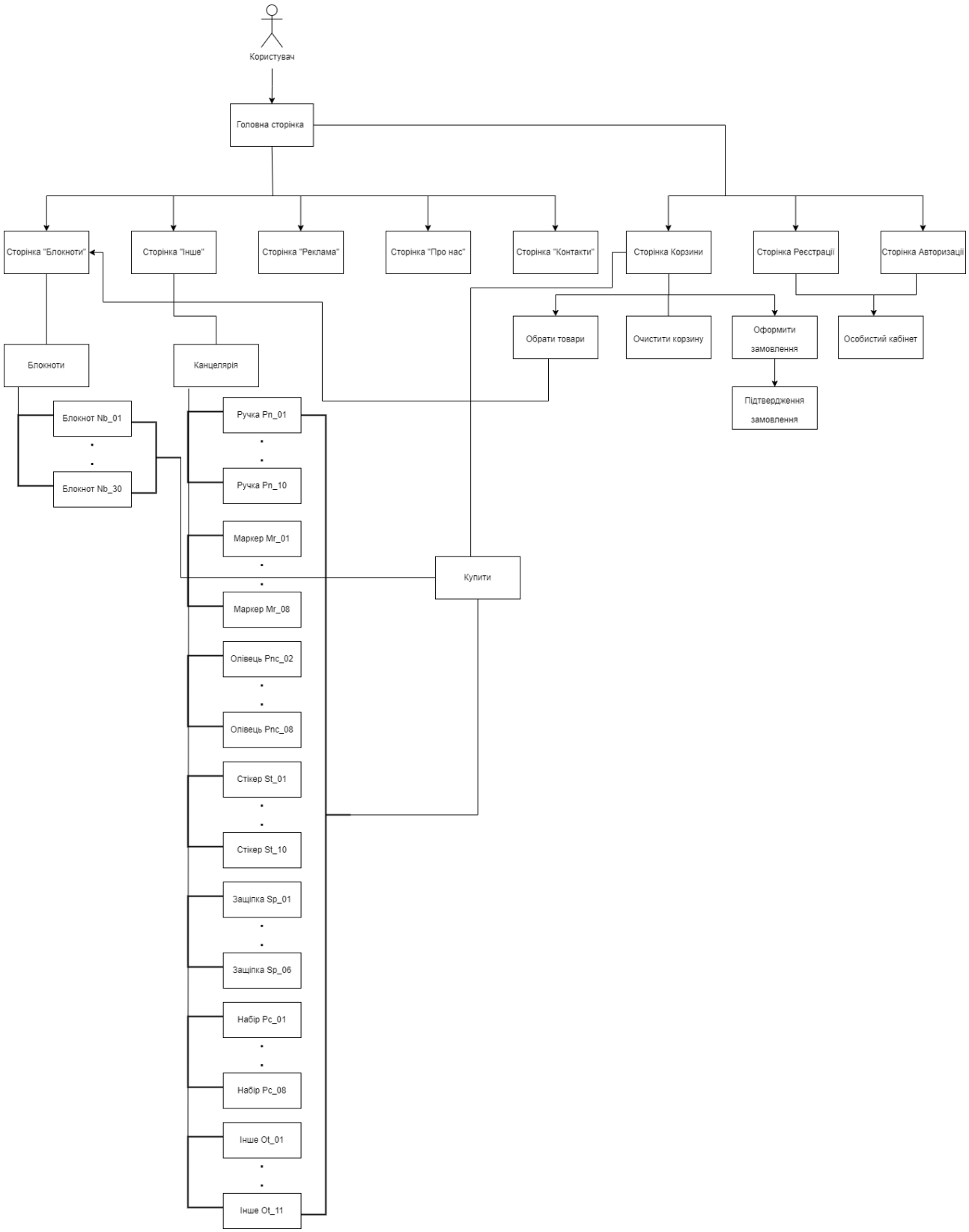
Web-додаток для продажу та реклами товарів канцелярського призначення

Структурна схема системи

ІАЛЦ.467100.004 Д1

Аркушів 1

Київ 2022 р



					ІАЛЦ.467100.004 Д1			
		№ докум.	Підпис	Дата				
Розробив		Чуясова Т. А.			Web-додаток для продажу та реклами товарів канцелярського призначення Структурна схема системи	Літ.	Аркуш	Аркушів
Перевірив		Нікольський С. С.					1	1
						НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІВ-82		
Н. Контр.		Сімоненко А. В.						
Затвердив								

ДОДАТОК 2

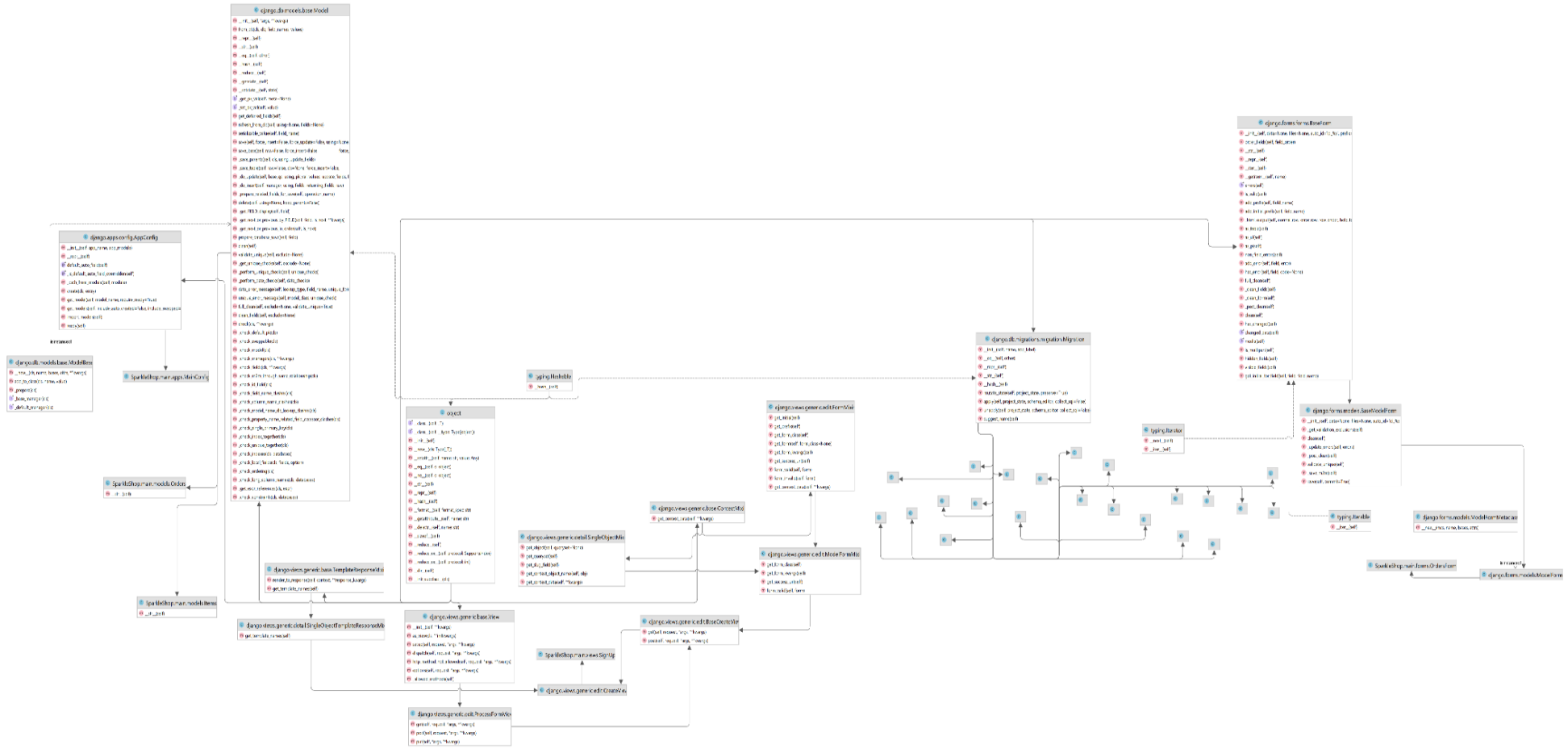
Web-додаток для продажу та реклами товарів канцелярського призначення

Функціональна схема (діаграма класів)

ІАЛЦ.467100.005 Д2

Аркушів 1

Київ 2022 р



					ІАЛЦ.467100.005 Д2	
		№ докум.	Підпис	Дата		
Розробив		Чуясова Т. А.			Web-додаток для продажу та реклами товарів канцелярського призначення Функціональна схема (діаграма класів)	Літ. / Аркуш / Аркушів 1 / 1
Перевірив		Нікольський С. С.				НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІВ-82
Н. Контр.		Сімоненко А. В.				
Затвердив						

ДОДАТОК З

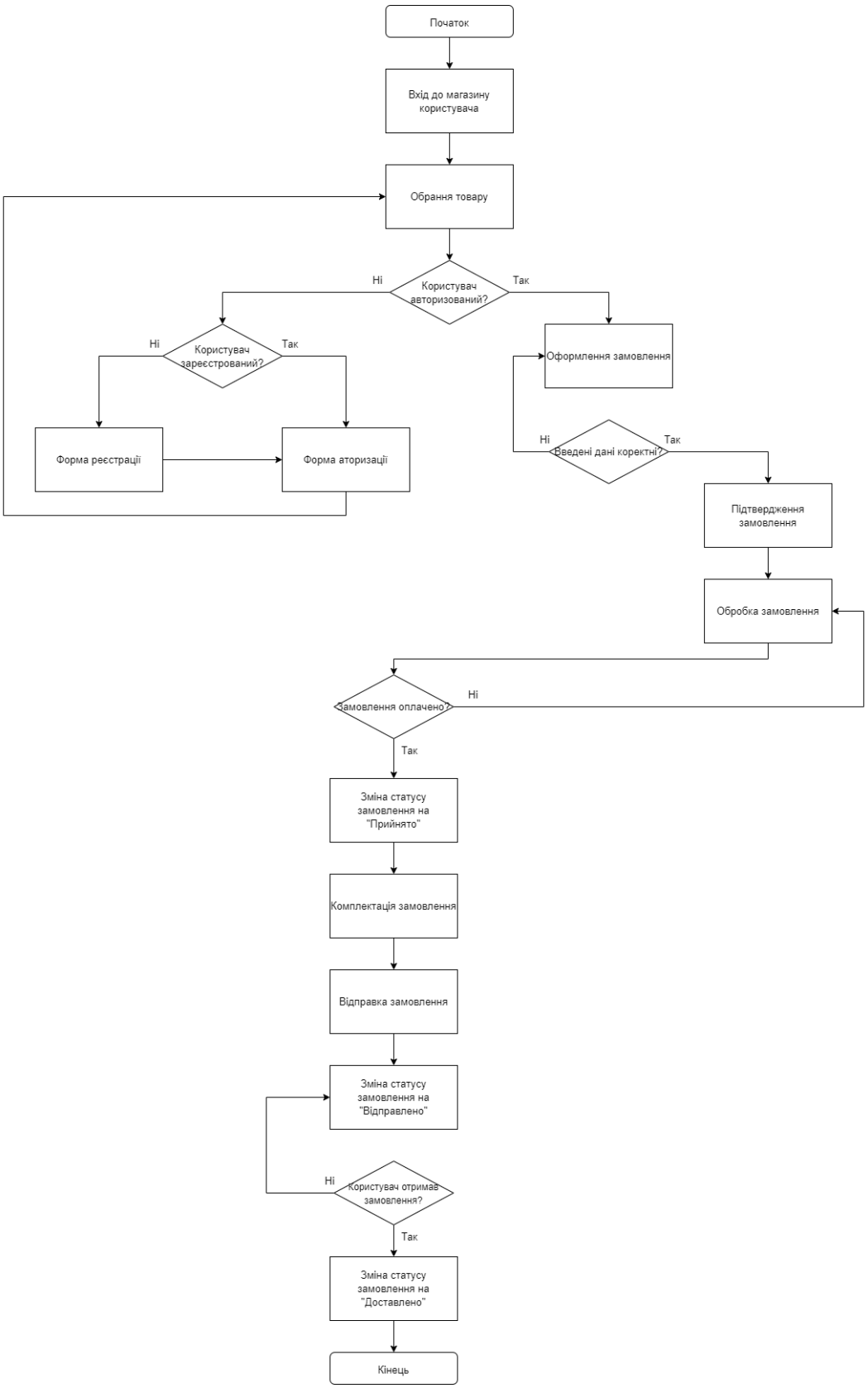
Web-додаток для продажу та реклами товарів канцелярського призначення

Принципова схема(алгоритм дій додатку)

ІАЛЦ.467100.006 Д3

Аркушів 1

Київ 2022 р



					ІАЛЦ.467100.006 Д3	
		№ докум.	Підпис	Дата		
Розробив		Чуясова Т. А.			Web-додаток для продажу та реклами товарів канцелярського призначення	Літ. / Аркуш / Аркушів
Перевірив		Нікольський С. С.				1 / 1
					Принципова схема (алгоритм дії додатку)	НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІВ-82
Н. Контр.		Сімоненко А. В.				
Затвердив						

ДОДАТОК 4

Web-додаток для продажу та реклами товарів канцелярського призначення

Текст програмного коду

ІАЛЦ.467100.007 Д4

Аркушів 14

Київ 2022 р

Settings.py

```
import os
from pathlib import Path
# Файл для налаштувань системи
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-pavj_4+%n%q^h2lp9_3iflof)+)3s719z@5lny8%r16ryawb@p'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []



# Application definition

INSTALLED_APPS = [
    'main',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'SparkleShop.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'SparkleShop.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation

# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators
```

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE_CODE = 'uk'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC_URL = '/static/'

STATICFILES_DIRS = [
    BASE_DIR / "static"
]

# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'


LOGIN_REDIRECT_URL = '/'
}
}
```

SparkleShop\Urls.py

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('main.urls'))
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Admin.py

```
from django.contrib import admin
from .models import Orders,Items

admin.site.register(Orders)
admin.site.register(Items)
```

Apps.py

```
from django.apps import AppConfig


class MainConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'main'
```

Forms.py

```
from .models import Orders,Items
from django.forms import ModelForm, TextInput, NumberInput, EmailInput, CharField, IntegerField, forms,
EmailField, \
    BooleanField, Textarea


class OrdersForm(ModelForm):
    class Meta:
        model = Orders
        fields = ['username','orderer_items', 'total_sum', 'name', 'surname', 'phone_number', 'email',
                  'city', 'post_number', 'payment_method']

        # def __init__(self, suma, *args, **kwargs):
        #  super(OrdersForm, self).__init__(*args, **kwargs)
        # self.fields[0].queryset = suma

        widgets = {
            "username": TextInput(attrs={
                'class': 'to_order',
                'placeholder': "Введіть ваш username"
            }),
           # "status": TextInput(attrs={

            #    'class': 'to_order_sum',
             #   'style': 'width:130px;  margin-left: 455px; margin-top: 10px; position-relative;'
              #            'cursor:not-allowed;',
               # 'readonly': 'readonly'

            #}),
            "orderer_items": Textarea(attrs={

                'class': 'to_order',
                'placeholder': "Замовлені товари",
                'style': 'resize:none; font-size:14px; text-align:left;',
                'readonly': 'readonly'
            }),
            "total_sum": TextInput(attrs={

                'class': 'to_order_sum',
                'style': 'width:130px; color:#000000; font-weight:bold; margin-left: 420px; margin-top: -1000px;
position-relative;'
                         'cursor:not-allowed;',
                'readonly': 'readonly'

            }),
            "name": TextInput(attrs={

                'class': 'to_order',
                'placeholder': "Ім'я"
            }),
            "surname": TextInput(attrs={
                'class': 'to_order',
                'placeholder': "Прізвище"
            }),
            "phone_number": NumberInput(attrs={
                'class': 'to_order',
                'placeholder': "Номер телефону"
            }),
            "email": EmailInput(attrs={
                'class': 'to_order',
                'placeholder': "E-mail"


            }),
            "city": TextInput(attrs={
                'class': 'to_order',
                'placeholder': "Назва населеного пункту"
            }),
            "post_number": NumberInput(attrs={
                'class': 'to_order',
                'placeholder': "Номер відділення Нової Пошти"
            }),
            "payment_method": TextInput(attrs={
                'class': 'to_order',
                'placeholder': "Спосіб оплати"
            })
        }
```

Models.py

```
from django.db import models
import datetime
```

```
ORDER_CHOICES = (
    ('Нове','Нове'),
    ('Прийнято', 'Прийнято'),
    ('Відправлено','Відправлено'),
    ('Доставлено','Доставлено')
)

class Orders(models.Model):
    username = models.CharField("Username користувача" , max_length=100)


    status = models.CharField("Статус замовлення" , max_length=20, choices=ORDER_CHOICES, default='Новий')
    orderer_items = models.TextField("Замовлені товари" , max_length=1000)
    total_sum = models.CharField("Сума замовлення" , max_length=30 )
    name = models.CharField("Ім'я", max_length=30)
    surname = models.CharField("Прізвище", max_length=50)
    phone_number = models.IntegerField("Номер телефору" , max_length=25)
    email = models.EmailField("E-mail", max_length=50)
    city = models.CharField("Назва населеного пункту", max_length=50)
    post_number = models.IntegerField("Номер відділення Нової Пошти", max_length=25)
    payment_method = models.CharField("Спосіб оплати", max_length=18, default="Накладний платіж")
    date = models.DateField("Дата замовлення", default=datetime.date.today())

    def __str__(self):
        return self.surname

    class Meta:
        verbose_name ='Замовлення'
        verbose_name_plural = 'Замовлення'

class Items(models.Model):
    item_id = models.CharField("Код товару", max_length=10)
    name = models.CharField("Назва товару", max_length=300)
    price = models.IntegerField("Ціна за одиницю")
    amount = models.IntegerField("Кількість на складі" )

    def __str__(self):
        return self.name

    class Meta:
        verbose_name ='Товар'
        verbose_name_plural = 'Товари'
```

Urls.py

```
from . import views, admin
from django.urls import path,include

urlpatterns = [
        path('admin/', admin.admin.site.urls),
        path('accounts/', include('django.contrib.auth.urls')),
        path('', views.index, name = 'index'),
        path('Notebooks_page',views.Notebooks_page , name = 'Notebooks_page'),
        path('done',views.done , name = 'done'),
        path('userpage',views.userpage , name = 'userpage'),

        path('Staff_page',views.Staff_page , name = 'Staff_page'),
        path('Ads_page',views.Ads_page , name = 'Ads_page'),
        path('About_page',views.About_page , name = 'About_page'),
        path('Contacts_page',views.Contacts_page , name = 'Contacts_page'),
        path('Cart_page',views.Cart_page , name = 'Cart_page'),
        path('Order',views.Order , name = 'Order'),
        path('Registrarion',views.SignUp.as_view() , name = 'Registrarion'),
        path('Autorization_page',views.Autorization_page , name = 'Autorization_page'),
        path('Order',views.Order , name = 'Order'),

        path('Mr_01',views.Mr_01 , name = 'Mr_01'),
        path('Mr_02',views.Mr_02 , name = 'Mr_02'),
        path('Mr_03',views.Mr_03 , name = 'Mr_03'),
        path('Mr_04',views.Mr_04 , name = 'Mr_04'),
        path('Mr_05',views.Mr_05 , name = 'Mr_05'),
        path('Mr_06',views.Mr_06 , name = 'Mr_06'),
        path('Mr_07',views.Mr_07 , name = 'Mr_07'),
        path('Mr_08',views.Mr_08 , name = 'Mr_08'),

        path('Nb_01',views.Nb_01 , name = 'Nb_01'),
        path('Nb_02',views.Nb_02 , name = 'Nb_02'),
        path('Nb_03',views.Nb_03 , name = 'Nb_03'),
        path('Nb_04',views.Nb_04 , name = 'Nb_04'),
        path('Nb_05',views.Nb_05 , name = 'Nb_05'),
        path('Nb_06',views.Nb_06 , name = 'Nb_06'),
        path('Nb_07',views.Nb_07 , name = 'Nb_07'),
        path('Nb_08',views.Nb_08 , name = 'Nb_08'),
        path('Nb_09',views.Nb_09 , name = 'Nb_09'),
        path('Nb_10',views.Nb_10 , name = 'Nb_10'),
        path('Nb_11',views.Nb_11 , name = 'Nb_11'),
```

```
        path('Nb_12',views.Nb_12 , name = 'Nb_12'),
        path('Nb_13',views.Nb_13 , name = 'Nb_13'),
        path('Nb_14',views.Nb_14 , name = 'Nb_14'),
        path('Nb_15',views.Nb_15 , name = 'Nb_15'),
        path('Nb_16',views.Nb_16 , name = 'Nb_16'),
        path('Nb_17',views.Nb_17 , name = 'Nb_17'),
        path('Nb_18',views.Nb_18 , name = 'Nb_18'),
        path('Nb_19',views.Nb_19 , name = 'Nb_19'),
        path('Nb_20',views.Nb_20 , name = 'Nb_20'),
        path('Nb_21',views.Nb_21 , name = 'Nb_21'),
        path('Nb_22',views.Nb_22 , name = 'Nb_22'),

        path('Nb_23',views.Nb_23 , name = 'Nb_23'),
        path('Nb_24',views.Nb_24 , name = 'Nb_24'),
        path('Nb_25',views.Nb_25 , name = 'Nb_25'),
        path('Nb_26',views.Nb_26 , name = 'Nb_26'),
        path('Nb_27',views.Nb_27 , name = 'Nb_27'),
        path('Nb_28',views.Nb_28 , name = 'Nb_28'),
        path('Nb_29',views.Nb_29 , name = 'Nb_29'),
        path('Nb_30',views.Nb_30 , name = 'Nb_30'),

        path('Ot_01',views.Ot_01 , name = 'Ot_01'),
        path('Ot_02',views.Ot_02 , name = 'Ot_02'),
        path('Ot_03',views.Ot_03 , name = 'Ot_03'),
        path('Ot_04',views.Ot_04 , name = 'Ot_04'),
        path('Ot_05',views.Ot_05 , name = 'Ot_05'),
        path('Ot_06',views.Ot_06 , name = 'Ot_06'),
        path('Ot_07',views.Ot_07 , name = 'Ot_07'),
        path('Ot_08',views.Ot_08 , name = 'Ot_08'),
        path('Ot_10',views.Ot_10 , name = 'Ot_10'),
        path('Ot_11',views.Ot_11 , name = 'Ot_11'),

        path('Pc_01',views.Pc_01 , name = 'Pc_01'),
        path('Pc_02',views.Pc_02 , name = 'Pc_02'),
        path('Pc_03',views.Pc_03 , name = 'Pc_03'),
        path('Pc_04',views.Pc_04 , name = 'Pc_04'),
        path('Pc_05',views.Pc_05 , name = 'Pc_05'),
        path('Pc_06',views.Pc_06 , name = 'Pc_06'),
        path('Pc_07',views.Pc_07 , name = 'Pc_07'),
        path('Pc_08',views.Pc_08 , name = 'Pc_08'),

        path('Pn_01',views.Pn_01 , name = 'Pn_01'),
        path('Pn_02',views.Pn_02 , name = 'Pn_02'),
        path('Pn_03',views.Pn_03 , name = 'Pn_03'),
        path('Pn_04',views.Pn_04 , name = 'Pn_04'),
        path('Pn_05',views.Pn_05 , name = 'Pn_05'),
        path('Pn_06',views.Pn_06 , name = 'Pn_06'),
        path('Pn_07',views.Pn_07 , name = 'Pn_07'),
        path('Pn_08',views.Pn_08 , name = 'Pn_08'),

        path('Pn_09',views.Pn_09 , name = 'Pn_09'),
        path('Pn_10',views.Pn_10 , name = 'Pn_10'),

        path('Pnc_02',views.Pnc_02 , name = 'Pnc_02'),
        path('Pnc_03',views.Pnc_03 , name = 'Pnc_03'),
        path('Pnc_04',views.Pnc_04 , name = 'Pnc_04'),
        path('Pnc_05',views.Pnc_05 , name = 'Pnc_05'),
        path('Pnc_06',views.Pnc_06 , name = 'Pnc_06'),
        path('Pnc_08',views.Pnc_08 , name = 'Pnc_08'),

        path('Sp_01',views.Sp_01 , name = 'Sp_01'),
        path('Sp_02',views.Sp_02 , name = 'Sp_02'),
        path('Sp_03',views.Sp_03 , name = 'Sp_03'),
        path('Sp_04',views.Sp_04 , name = 'Sp_04'),
        path('Sp_05',views.Sp_05 , name = 'Sp_05'),
        path('Sp_06',views.Sp_06 , name = 'Sp_06'),

        path('St_01',views.St_01 , name = 'St_01'),
        path('St_02',views.St_02 , name = 'St_02'),
        path('St_03',views.St_03 , name = 'St_03'),
        path('St_04',views.St_04 , name = 'St_04'),
        path('St_05',views.St_05 , name = 'St_05'),
        path('St_06',views.St_06 , name = 'St_06'),
        path('St_07',views.St_07 , name = 'St_07'),
        path('St_08',views.St_08 , name = 'St_08'),
        path('St_09',views.St_09 , name = 'St_09'),
        path('St_10',views.St_10 , name = 'St_10')


]
```

Views.py

```
from django.shortcuts import render, redirect
from django.urls import reverse_lazy
from django.contrib.auth.forms import UserCreationForm
from django.views.generic.edit import CreateView
from .forms import OrdersForm
from .models import Orders
```

```
class SignUp(CreateView):
    form_class = UserCreationForm
    success_url = reverse_lazy("login")
    template_name = "registration/Registrarion.html"


def index(request):
    return render(request, 'main/index.html')




def Notebooks_page(request):
    return render(request, 'main/Notebooks_page.html')


def Staff_page(request):
    return render(request, 'main/Staff_page.html')


def Ads_page(request):
    return render(request, 'main/Ads_page.html')


def About_page(request):
    return render(request, 'main/About_page.html')


def Contacts_page(request):
    return render(request, 'main/Contacts_page.html')

def done(request):
    return render(request, 'main/done.html')

def userpage(request):
    user = request.user.username
    order = Orders.objects.filter(username=user)

    return render(request, 'main/userpage.html', {'order': order})


def Order(request):
    vari = str(suma)
    vari2 = str(items)
    error = ''

    if request.method == 'POST':
        form = OrdersForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('done')

        else:
            error = 'Невірно введені дані'

    form = OrdersForm(initial={"total_sum": vari+" грн", "orderer_items" : vari2})
    data = {
        'form': form,
        'error': error
    }

    return render(request, 'main/Order.html',data)


def Cart_page(request):
    global suma, items
    suma = request.POST.get('suma')
    items = request.POST.get('items')
    #print("SUUUUUMAAAAA",suma)
    #print("itemsitemsitemsitems", items)

    return render(request, 'main/Cart_page.html')


#def Registrarion(request):
 #   return render(request, 'registration/Registrarion.html')


def Autorization_page(request):
    return render(request, 'registration/Autorization_page.html')


def Mr_01(request):
    return render(request, 'main/Mr_01.html')


def Mr_02(request):
    return render(request, 'main/Mr_02.html')
```

```
def Mr_03(request):
    return render(request, 'main/Mr_03.html')


def Mr_04(request):
    return render(request, 'main/Mr_04.html')


def Mr_05(request):
    return render(request, 'main/Mr_05.html')


def Mr_06(request):
    return render(request, 'main/Mr_06.html')


def Mr_07(request):
    return render(request, 'main/Mr_07.html')


def Mr_08(request):
    return render(request, 'main/Mr_08.html')


def Nb_01(request):
    return render(request, 'main/Nb_01.html')


def Nb_02(request):
    return render(request, 'main/Nb_02.html')


def Nb_03(request):
    return render(request, 'main/Nb_03.html')


def Nb_04(request):
    return render(request, 'main/Nb_04.html')


def Nb_05(request):
    return render(request, 'main/Nb_05.html')


def Nb_06(request):
    return render(request, 'main/Nb_06.html')


def Nb_07(request):
    return render(request, 'main/Nb_07.html')


def Nb_08(request):
    return render(request, 'main/Nb_08.html')


def Nb_09(request):
    return render(request, 'main/Nb_09.html')


def Nb_10(request):
    return render(request, 'main/Nb_10.html')


def Nb_11(request):
    return render(request, 'main/Nb_11.html')


def Nb_12(request):
    return render(request, 'main/Nb_12.html')


def Nb_13(request):
    return render(request, 'main/Nb_13.html')


def Nb_14(request):
    return render(request, 'main/Nb_14.html')


def Nb_15(request):
    return render(request, 'main/Nb_15.html')


def Nb_16(request):
    return render(request, 'main/Nb_16.html')


def Nb_17(request):
    return render(request, 'main/Nb_17.html')
```

```
def Nb_18(request):
    return render(request, 'main/Nb_18.html')


def Nb_19(request):
    return render(request, 'main/Nb_19.html')


def Nb_20(request):
    return render(request, 'main/Nb_20.html')


def Nb_21(request):
    return render(request, 'main/Nb_21.html')


def Nb_22(request):
    return render(request, 'main/Nb_22.html')


def Nb_23(request):
    return render(request, 'main/Nb_23.html')


def Nb_24(request):
    return render(request, 'main/Nb_24.html')


def Nb_25(request):
    return render(request, 'main/Nb_25.html')


def Nb_26(request):
    return render(request, 'main/Nb_26.html')


def Nb_27(request):
    return render(request, 'main/Nb_27.html')


def Nb_28(request):
    return render(request, 'main/Nb_28.html')


def Nb_29(request):
    return render(request, 'main/Nb_29.html')


def Nb_30(request):
    return render(request, 'main/Nb_30.html')



def Ot_01(request):
    return render(request, 'main/Ot_01.html')

def Ot_02(request):
    return render(request, 'main/Ot_02.html')

def Ot_03(request):
    return render(request, 'main/Ot_03.html')

def Ot_04(request):
    return render(request, 'main/Ot_04.html')

def Ot_05(request):
    return render(request, 'main/Ot_05.html')

def Ot_06(request):
    return render(request, 'main/Ot_06.html')

def Ot_07(request):
    return render(request, 'main/Ot_07.html')

def Ot_08(request):
    return render(request, 'main/Ot_08.html')

def Ot_10(request):
    return render(request, 'main/Ot_10.html')

def Ot_11(request):
    return render(request, 'main/Ot_11.html')



def Pc_01(request):
    return render(request, 'main/Pc_01.html')

def Pc_02(request):
    return render(request, 'main/Pc_02.html')
```

```
def Pc_03(request):
    return render(request, 'main/Pc_03.html')

def Pc_04(request):
    return render(request, 'main/Pc_04.html')

def Pc_05(request):
    return render(request, 'main/Pc_05.html')

def Pc_06(request):
    return render(request, 'main/Pc_06.html')

def Pc_07(request):
    return render(request, 'main/Pc_07.html')

def Pc_08(request):
    return render(request, 'main/Pc_08.html')



def Pn_01(request):
    return render(request, 'main/Pn_01.html')

def Pn_02(request):
    return render(request, 'main/Pn_02.html')

def Pn_03(request):
    return render(request, 'main/Pn_03.html')

def Pn_04(request):
    return render(request, 'main/Pn_04.html')

def Pn_05(request):
    return render(request, 'main/Pn_05.html')

def Pn_06(request):
    return render(request, 'main/Pn_06.html')

def Pn_07(request):
    return render(request, 'main/Pn_07.html')

def Pn_08(request):
    return render(request, 'main/Pn_08.html')

def Pn_09(request):
    return render(request, 'main/Pn_09.html')

def Pn_10(request):
    return render(request, 'main/Pn_10.html')



def Pnc_02(request):
    return render(request, 'main/Pnc_02.html')

def Pnc_03(request):
    return render(request, 'main/Pnc_03.html')

def Pnc_04(request):
    return render(request, 'main/Pnc_04.html')

def Pnc_05(request):
    return render(request, 'main/Pnc_05.html')

def Pnc_06(request):
    return render(request, 'main/Pnc_06.html')

def Pnc_08(request):
    return render(request, 'main/Pnc_08.html')



def Sp_01(request):
    return render(request, 'main/Sp_01.html')

def Sp_02(request):
    return render(request, 'main/Sp_02.html')

def Sp_03(request):
    return render(request, 'main/Sp_03.html')

def Sp_04(request):
    return render(request, 'main/Sp_04.html')

def Sp_05(request):
    return render(request, 'main/Sp_05.html')

def Sp_06(request):
    return render(request, 'main/Sp_06.html')
```

```
def St_01(request):
    return render(request, 'main/St_01.html')

def St_02(request):
    return render(request, 'main/St_02.html')

def St_03(request):
    return render(request, 'main/St_03.html')

def St_04(request):
    return render(request, 'main/St_04.html')

def St_05(request):
    return render(request, 'main/St_05.html')

def St_06(request):
    return render(request, 'main/St_06.html')

def St_07(request):
    return render(request, 'main/St_07.html')

def St_08(request):
    return render(request, 'main/St_08.html')

def St_09(request):
    return render(request, 'main/St_09.html')

def St_10(request):
    return render(request, 'main/St_10.html')
```

Manage.py

```
#!/usr/bin/env python
"""Файл менеджменту"""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'SparkleShop.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

Cart.js

```
var d = document,
    itemBox = d.querySelectorAll('.item_box'),
    cartCont = d.getElementById('cart_content');
//////////встановлення обробника івентів
function addEvent(elem, type, handler){
  if(elem.addEventListener){
    elem.addEventListener(type, handler, false);
  } else {
    elem.attachEvent('on'+type, function(){ handler.call( elem ); });
  }
  return false;
}
//////////отримування даних з локалстореджу
function getCartData(){
  return JSON.parse(localStorage.getItem('cart'));
}
//////////запис даних у локалсторедж
function setCartData(o){
  localStorage.setItem('cart', JSON.stringify(o));
  return false;
}

//////////відображення даних у таблиці корзині
function viewDiv(){
  document.getElementById("divtest").style.display = "block";
  document.getElementById("cart_content").style.display = "none";
  document.getElementById("choose").style.display = "none";
  document.getElementById("clear_cart").style.display = "none";
  document.getElementById("trash").style.display = "none";
  document.getElementById("toodrerbtn").style.display = "none";
  document.getElementById("totalprice").innerHTML = sum;
```

```
};



//////////додавання товару в корзину
function addToCart(e){
  this.disabled = true;
  var cartData = getCartData() || {},
      parentBox = this.parentNode,
      itemId = this.getAttribute('data-id'),

      itemTitle = parentBox.querySelector('.item_title').innerHTML,
      itemPrice = parentBox.querySelector('.item_price').innerHTML;
      itemTotal = parentBox.querySelector('.item_price').innerHTML;


  if(cartData.hasOwnProperty(itemId)){ // якщо таких товар в корзині, то додати +1 до кількості
    cartData[itemId][2] += 1;
    cartData[itemId][3]  = Number(cartData[itemId][2]) * Number(cartData[itemId][1]);
  } else { // якщо такого товару немає, додати об'єкт
    cartData[itemId] = [itemTitle, itemPrice, 1, itemTotal];


  }

  if(!setCartData(cartData)){
    this.disabled = false;
    setTimeout(function(){
      cartCont.innerHTML = '';
    }, 1000);
  }
  return false;
}

for(var i = 0; i < itemBox.length; i++){
  addEvent(itemBox[i].querySelector('.add_item'), 'click', addToCart);
}


var cartData = getCartData(),
      totalItems = '';
  console.log(JSON.stringify(cartData));

//////////формування даних таблиці для виводу
  if(cartData !== null){
    totalItems = '<table id="table" class="shopping_list" style="font-size:18px;"><tr><th>Назва
товару</th><th>Ціна</th><th>Кіл-сть</th><th class="together">Разом</th></tr>';
    for(var items in cartData){
      totalItems += '<tr>';
      document.getElementById("orderitem").innerHTML +=cartData[items][0] + " - " + cartData[items][2] + ", ";



      for(var i = 0; i < cartData[items].length; i++){

        totalItems += '<td>' + cartData[items][i] + '</td>';

      }
      totalItems += '</tr>';

    }
    totalItems += '<table>';
    cartCont.innerHTML = totalItems;
  } else {
    cartCont.innerHTML = '<div style="sans-serif; font-size: 25px; color: #8480c3;letter-spacing: 3px; margin-
left:590px;">В корзині порожньо :(</div>';
  }


//////////очистика корзини
addEvent(d.getElementById('clear_cart'), 'click', function(e){
  localStorage.removeItem('cart');
  cartCont.innerHTML = '<div style="sans-serif; font-size: 25px; color: #8480c3;letter-spacing: 3px; margin-
left:620px;">Корзина очищена</div>';
});

//////////розрахунок загальної суми замовлення
  var table = document.querySelector('#table');

  var trs = table.querySelectorAll('tr');
  var colsNum = trs[0].querySelectorAll('td').length;

  var resultTr = document.createElement('tr');
  table.appendChild(resultTr);



  for (var i = 3; i < 4; i++) {
    var tdNumber = i + 1;
    var tds = table.querySelectorAll('td:nth-child(' + tdNumber + ')');
```

```
    sum = 0;



    for (var j = 0; j < tds.length; j++) {
      sum += Number(tds[j].innerHTML);

    }

    var resultTd = document.createElement('td');
    resultTd.innerHTML = '<div style="font-size:20px; ">Загальна сума замовлення - ' + sum +' грн </div>';
    resultTr.appendChild(resultTd);
    localStorage.setItem(sum);




  };
```

Filter_notebooks.js

```
//////////функція присвоювання змінних чекбоксів
function change() {
  var sizeCbs = document.querySelectorAll(".size input[type='checkbox']");
  var colorCbs = document.querySelectorAll(".color input[type='checkbox']");
  var materialCbs = document.querySelectorAll(".material input[type='checkbox']");
  var saleCbs = document.querySelectorAll(".sale input[type='checkbox']");
  var filters = {
    size: getClassOfCheckedCheckboxes(sizeCbs),
    color: getClassOfCheckedCheckboxes(colorCbs),
    material: getClassOfCheckedCheckboxes(materialCbs),
    sale: getClassOfCheckedCheckboxes(saleCbs)
  };

  filterResults(filters);
}
//////////функція отримування класів чекбоксів
function getClassOfCheckedCheckboxes(checkboxes) {
  var classes = [];

  if (checkboxes && checkboxes.length > 0) {
    for (var i = 0; i < checkboxes.length; i++) {
      var cb = checkboxes[i];

      if (cb.checked) {
        classes.push(cb.getAttribute("rel"));
      }
    }
  }

  return classes;
}
//////////функція відображення результатів фільтрації
function filterResults(filters) {
  var rElems = document.querySelectorAll(".result div");
  var hiddenElems = [];

  if (!rElems || rElems.length <= 0) {
    return;
  }

  for (var i = 0; i < rElems.length; i++) {
    var el = rElems[i];

    if (filters.size.length > 0) {
      var isHidden = true;

      for (var j = 0; j < filters.size.length; j++) {
        var filter = filters.size[j];

        if (el.classList.contains(filter)) {
          isHidden = false;
          break;
        }
      }

      if (isHidden) {
        hiddenElems.push(el);
      }
    }

    if (filters.color.length > 0) {
      var isHidden = true;

      for (var j = 0; j < filters.color.length; j++) {
        var filter = filters.color[j];

        if (el.classList.contains(filter)) {
          isHidden = false;
```

```
          break;
        }
      }

      if (isHidden) {
        hiddenElems.push(el);
      }
    }


    if (filters.material.length > 0) {
      var isHidden = true;

      for (var j = 0; j < filters.material.length; j++) {
        var filter = filters.material[j];

        if (el.classList.contains(filter)) {
          isHidden = false;
          break;
        }
      }

      if (isHidden) {
        hiddenElems.push(el);
      }
    }

    if (filters.sale.length > 0) {
      var isHidden = true;

      for (var j = 0; j < filters.sale.length; j++) {
        var filter = filters.sale[j];

        if (el.classList.contains(filter)) {
          isHidden = false;
          break;
        }
      }

      if (isHidden) {
        hiddenElems.push(el);
      }
    }
  }

  for (var i = 0; i < rElems.length; i++) {
    rElems[i].style.display = "block";
  }

  if (hiddenElems.length <= 0) {
    return;
  }

  for (var i = 0; i < hiddenElems.length; i++) {
    hiddenElems[i].style.display = "none";
  }
}
```

Done.html

```
{% load static %}
<!DOCTYPE html>
<html lang="ru">
<head>
   <meta charset="UTF-8">
   <title>Sparkle Shop</title>
   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
   <link href="https://fonts.googleapis.com/css?family=Raleway:900|Raleway&amp;subset=cyrillic" rel="stylesheet">
   <link rel="stylesheet" href="{% static 'main/css/main.css' %}">
   <link rel="shortcut icon" href="{% static 'main/img\logo.ico' %}" type="image/x-icon">
   <link rel="stylesheet" href="{% static 'main/css/animate.css' %}">
</head>
<body>
   <header id="header" class="header">
      <div>
         <a href="{% url 'index' %}">
         <img src="{% static 'main/img\logo.png' %}" alt="logo" class="logo animated fadeInDown"></a>
      </div>

      <section id="navigation">
          <form class="menu animated fadeInDown">
            <style>
                   a {
                      text-decoration: none;
                      }
                   a:hover {
                      text-decoration: underline;
                      }
               </style>
            <a class="format_nv" href="{% url 'Notebooks_page' %}">Блокноти</a>
            <a class="format_nv" href="{% url 'Staff_page' %}">Інше</a>
            <a class="format_nv" href="{% url 'Ads_page' %}">Реклама</a>
```

```
            <a class="format_nv" href="{% url 'About_page' %}">Про нас</a>
            <a class="format_nv" href="{% url 'Contacts_page' %}">Контакти</a>
         </form>
      </section>
      <div class ="cart">
           <a href="{% url 'Cart_page' %}">
         <img src="{% static 'main/img\cart.png' %}" alt="cart" class="animated fadeInDown"></a>
      </div>

      <div class=" log_in_text_pos animated fadeInDown" >
         {% if user.is_authenticated %}
                <div class="log_in_text"><a href="{% url 'userpage' %}" >{{ user.get_username }}</a></div>
               <form class="in_btn"><a href="{% url 'logout'%}?next={{request.path}}" class="btn animated
fadeInDown" id="btn" target="blank">Вийти</a>
                 </form>

               {% else %}
                  <a class="log_in_text" href="{% url 'Registrarion' %}" target="blank">Зареєструватися</a>
                  <form class="in_btn" ><a href="{% url 'login'%}?next={{request.path}}" class="btn animated
fadeInDown" id="btn" target="blank">Увійти</a>
                   </form>

         {% endif %}


   </header>


   <section id="confirmation">
      <div class="lending_title_note" style="margin-left: 340px;  font-size: 38px;" id="trash">
         Ваше замовлення успішно прийняте!

      </div>
      <script src="{% static 'main/js/cart.js' %}"></script>
      <form class="in_btn" id="clear_cart" action="Notebooks_page.html" target="blank" style="margin-left: 560px;
margin-top: 50px;  "><a href="{% url 'Notebooks_page' %}"class="btn" id="btn" style="font-size: 18px; padding-top:
20px;width:400px; ">Повернутися до покупок!</a>
        </form>
      <script>
         alert("Ваше замовлення прийняте! Очікуйте, наш менеджер невдовзі з вами зв'яжеться");
      </script>


   </section>




   <section id="footer" style="margin-top:-300px;" >

      <div class="footer">

         <div class="footer_title"> © 2022 Sparkle Shop </div>
          <form class="footer_text">
            <style>
                   a {
                      text-decoration: none;
                      }
                   a:hover {
                      text-decoration: underline;
                      }
               </style>
            <a class="format_nv" href="{% url 'Notebooks_page' %}">Блокноти</a>
            <a class="format_nv" href="{% url 'Staff_page' %}">Інше</a>
            <a class="format_nv" href="{% url 'Ads_page' %}">Реклама</a>
            <a class="format_nv" href="{% url 'About_page' %}">Про нас</a>
            <a class="format_nv" href="{% url 'Contacts_page' %}">Контакти</a>
         </form>

         <div id ="inst">
              <a href="https://www.instagram.com/sparkle_shop_ukraine/">
            <img src="{% static 'main/img\icons8-instagram-50.png' %}" alt="inst" class="inst_footer"></a>
         </div>



      </div>
      <div id ="facebook">
             <a href="https://www.facebook.com/sparkleshopukraine">
            <img src="{% static 'main/img\icons8-facebook-50.png' %}" alt="facebook" class="facebook_footer"></a>
      </div>

   </section>
</body>
<script src="{% static 'main/js/cart.js' %}"></script>

</html>
```