МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра інформаційних технологій

«До захисту допущено»

В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: «Вебдодаток підтримки роботи магазину з продажу меблів»

Здобувача (ки) групи ІТ-02 Бобраніцький Назар Вадимович

(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Назар БОБРАНІЦЬКИЙ

(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доцент кафедри інформаційних технлогій, к.т.н., доц. Світлана ВАЩЕНКО

(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

В.о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
«__» _____ 202 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Бобраніцький Назар Вадимович

1 Тема роботи *Вебдодаток підтримки роботи магазину з продажу меблів*
керівник роботи *Ващенко Світлана Михайлівна, к.т.н., доцент,*

затверджені наказом по університету від « 07 » травня 2024 р. №0482-VI

2 Строк подання студентом роботи « 26 » червня 2024 р.

3 Вхідні дані до роботи технічне завдання на розробку вебдодатку підтримки роботи магазину з продажу меблів

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) вступ, аналіз предметної області, проектування вебдодатку, програмна реалізація вебдодатку, висновки, список використаних джерел.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність, постановка задачі, аналіз аналогів, вимоги, функціональне моделювання, моделювання використання вебдодатку, фізична модель даних, засоби реалізації, демонстрація роботи вебдодатку, висновки.

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 8 лютого 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Оформлення планування робіт	06.03.2024-17.03.2024	
2	Оформлення технічного завдання	20.03.2024-14.04.2024	
3	Проведення аналізу предметної області	17.04.2024-28.04.2024	
4	Проведення проектування вебдодатку	01.05.2024-12.05.2024	
5	Розробка та тестування вебдодатку	13.05.2024-05.06.2022	
6	Оформлення пояснювальної записки	06.03.2024-05.06.2024	

Студент _____ Назар БОБРАНІЦЬКИЙ
(підпис)

Керівник роботи _____ к.т.н., доц. Світлана ВАЩЕНКО
(підпис)

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра «Вебдодаток підтримки роботи магазину з продажу меблів».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 18 найменувань, додатків. Загальний обсяг роботи − 127 сторінок, у тому числі 57 сторінок основного тексту, 2 сторінки списку використаних джерел, 70 сторінок додатків.

У сучасному світі електронна комерція набуває все більшого значення, забезпечуючи зручність та доступність покупок. Розробка вебдодатку для продажу меблів для дому спрямована на полегшення процесу вибору та покупки меблів для різних приміщень, що є актуальним у зв'язку з розвитком технологій та змінами у споживацьких практиках.

Метою роботи є розробка вебдодатку організації діяльності магазину з продажу меблів для дому «TamTumba», який надає клієнтам можливість замовляти товари онлайн та переглядати їх у 3D форматі. Результати дослідження представлені у вигляді вебдодатку, спрямованого на поліпшення досвіду покупців та збільшення прибутковості меблевих магазинів.

Рекомендації щодо використання розробленого вебдодатку та результатів досліджень можуть стосуватися підвищення рейтингу та конкурентоспроможності магазину «TamTumba», а також збільшення прибутковості завдяки зручній онлайн платформі для покупок.

Ключові слова: електронна комерція, вебдодаток, меблів, онлайн магазин, 3D, каталог , сайт.

ЗМІСТ

ВСТУП

У сучасному світі електронна комерція набуває все більшого значення, забезпечуючи зручність та доступність покупок. У рамках цього дипломного проекту розглядається створення онлайн-магазину для продажу меблів для дому, який забезпечить зручний та ефективний вибір і придбання меблів для різних приміщень.

Цінність розробки виражається у покращенні досвіду покупців, їх здатності з легкістю знаходити необхідні товари, отримувати детальну інформацію та зручно здійснювати покупки в онлайн. Застосування онлайн-магазину для продажу меблів для дому має велику актуальність в умовах розвитку технологій та змін споживацьких практик.

Отже, метою дослідження є розробка вебдодатку для підтримки діяльності магазину з продажу меблів «TamTumba». Його використання забезпечить автоматизація таких процесів як, формування замовлень, доставка товару, інтерактивний перегляд товару у трьох вимірному вигляді.

Для досягнення мети проекту необхідно виконати такі задач :

– визначити актуальність роботи: задача передбачає аналіз поточного стану ринку меблів та його потреб. Потрібно визначити, чому саме створення вебдодатку для підтримки магазину меблів є важливим на сьогоднішній день, які проблеми це вирішить та які переваги це принесе;

– дослідити предметну область та провести аналіз існуючих аналогів онлайн сервісів: потрібно провести дослідження різних вебдодатків, що спеціалізуються на продажу меблів або пов'язаних з цим послуг. Важливо з'ясувати, які функції мають існуючі сервіси, їх переваги та недоліки, щоб врахувати це при розробці свого продукту;

– розробити структуру та необхідний функціонал програмного продукту : задача включає в себе створення концепції вебдодатку, визначення його основних

функцій (облік товарів, замовлень, оплати та інше), а також створення схеми взаємодії користувача з додатком;

– реалізувати структуру та необхідний функціонал програмного продукту: на основі попередньої структури розробляється вебдодаток. Розробники програмного забезпечення пишуть код програмного продукту та інтеграцію різних функцій та можливостей, щоб створити функціональний продукт;

– виконати тестування: проводяться тести, щоб перевірити роботу розробленого вебдодатку. Це включає в себе перевірку на відповідність вимогам, тестування на помилки та забезпечення коректної роботи всіх функцій. Також проводяться тести з використанням реальних користувачів для оцінки зручності інтерфейсу та функціональності.

Практичне значення цієї роботи полягає в розробці та впровадженні вебдодатку, який сприятиме оптимізації процесів продажу магазином меблів. Він дозволить автоматизувати такі процеси, як формування замовлень та збільшенню клієнтської бази.

Результати роботи представлено у доповіді на конференції «Інформатика. Математика. Автоматизація. 2024» (м. Суми, СумДУ).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

У сучасному світі не можна недооцінювати важливість присутності в Інтернеті для будь-якої установи, компанії чи підприємства. Візитною карткою будь-якого суб'єкта, який прагне привернути увагу та залучити клієнтів, є його вебсайт, який визначається як логічно пов'язана гіпертекстова інформація [2].

Вебсайт – це не лише інформаційний майданчик, а й інструмент взаємодії з аудиторією, відображення унікальності та створення позитивного іміджу. Лендінги, зокрема, – це інтернет платформа, що складаються з однієї великої сторінки, які стали популярними завдяки своїй простоті та здатності швидко привертати увагу.

Вміст сайту може включати графічні зображення, текст, відео та аудіо файли. Цілісність вебресурсу забезпечується гіперпосиланнями між сторінками платформи, що дозволяє створити зручний для користувача інтерфейс [3].

Важливо пам'ятати, що для ефективного просування та взаємодії з клієнтами вебсайт повинен не тільки існувати, але й бути унікальним і таким, що запам'ятовується. Привабливий дизайн, інноваційний підхід та достовірна інформація є одними з багатьох елементів, необхідних для надання послуг [4]. Важливо враховувати різні аспекти, такі як концепція закладу, адреса та години роботи, асортимент продукції та послуг, цінова політика, акції та заохочення.

В умовах сучасної бізнес-реальності інтернет платформи – це не лише онлайн-вітрина, а й ефективний інструмент взаємодії, який дозволяє користувачам здійснювати цілий ряд дій, починаючи від замовлення продукції і закінчуючи взаємодією з контентом. На даний момент 87% покупців вивчають інформацію в інтернеті, перш ніж зробити покупку. А за словами Nasdaq, до 2040 і зовсім 95% всіх продажів (незалежно від їх типу) відбуватимуться онлайн [5].

Тому, розробка вебдодатку організації діяльності магазину з продажу меблів, який би задовольнив усі вищезазначені вимоги та забезпечив умови для зростання прибутку є актуальною задачею за сучасних обставин ведення бізнесу.

1.2 Аналіз програмних продуктів – аналогів

Онлайн магазини для продажу меблів існують в онлайн-просторі вже давно. Однак більшість великих корпоративних сайтів навіть після розробки залишилися на примітивному рівні, характеризуючись застарілими, неякісними фотографіями та відсутністю сучасних функцій. Багатьом також бракує функціональних доповнень, наприклад, можливості зв’язатися з адміністратором.

Тому важливим фактором конкурентоспроможності онлайн-платформ для продажу меблів є адаптація до сучасних тенденцій розвитку. Для того, щоб визначити вимоги до майбутніх програмних продуктів, було проведено дослідження схожості між існуючими сайтами з продажу меблів, такими як «mebli.biz.ua» [6], «Artos» [7] та «DivanStar» [8].

На головній сторінці вебсайту інтернет-магазину «mebli.biz.ua» (рис. 1.1) розташований банер із зображеннями меблів та акційними пропозиціями, розділ із різноманітним асортиментом меблів, вказано структуру магазину (розділи з меблями для кімнат, кухні, вітальні, офісу, тощо), невеликий опис компанії, актуальні новини та соціальні мережі. Каталог товарів (рис. 1.2) функціональний та логічно зрозумілий.

Охарактеризувавши дизайн вебсайту коротко, можна сказати, що він виглядає не привабливо. Кольори сайту є зелений та білий. На сайті не має можливості зареєструватися, замовлення товару проводиться на сторінці яка зображена на рисунку 1.3. При оформленні замовлення вказується дані клієнта , адрес доставки та спосіб зв’язку через месенджер, тобто відразу оплатити замовлення не можна,

тільки після з’єднанню з менеджером . Отже, загальне враження від вебсайту не є приємним.

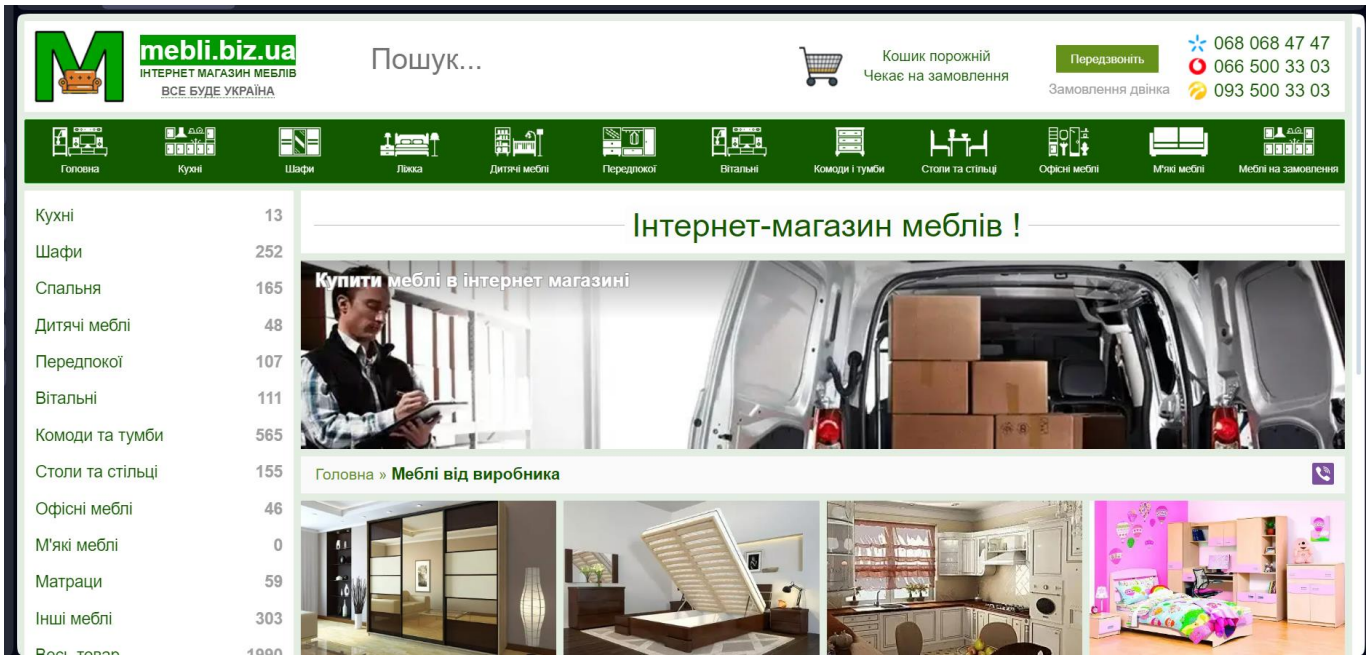


Рисунок 1.1 – Головна сторінка вебсайту «mebli.biz.ua»

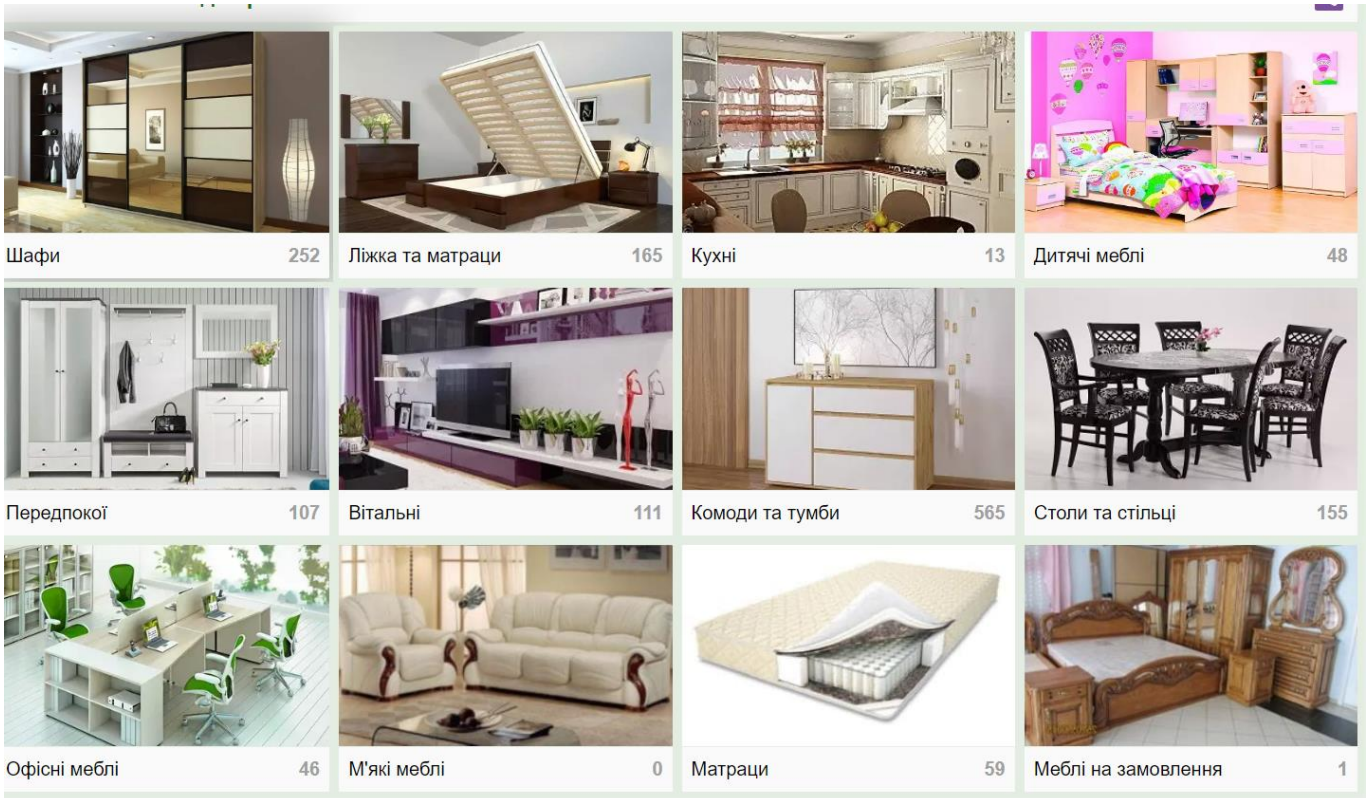


Рисунок 1.2 – Каталог товару

Перевагою є те, що можна оформити замовлення не реєструючись на сайті. Сторінка оформлення замовлення (рис. 1.3).

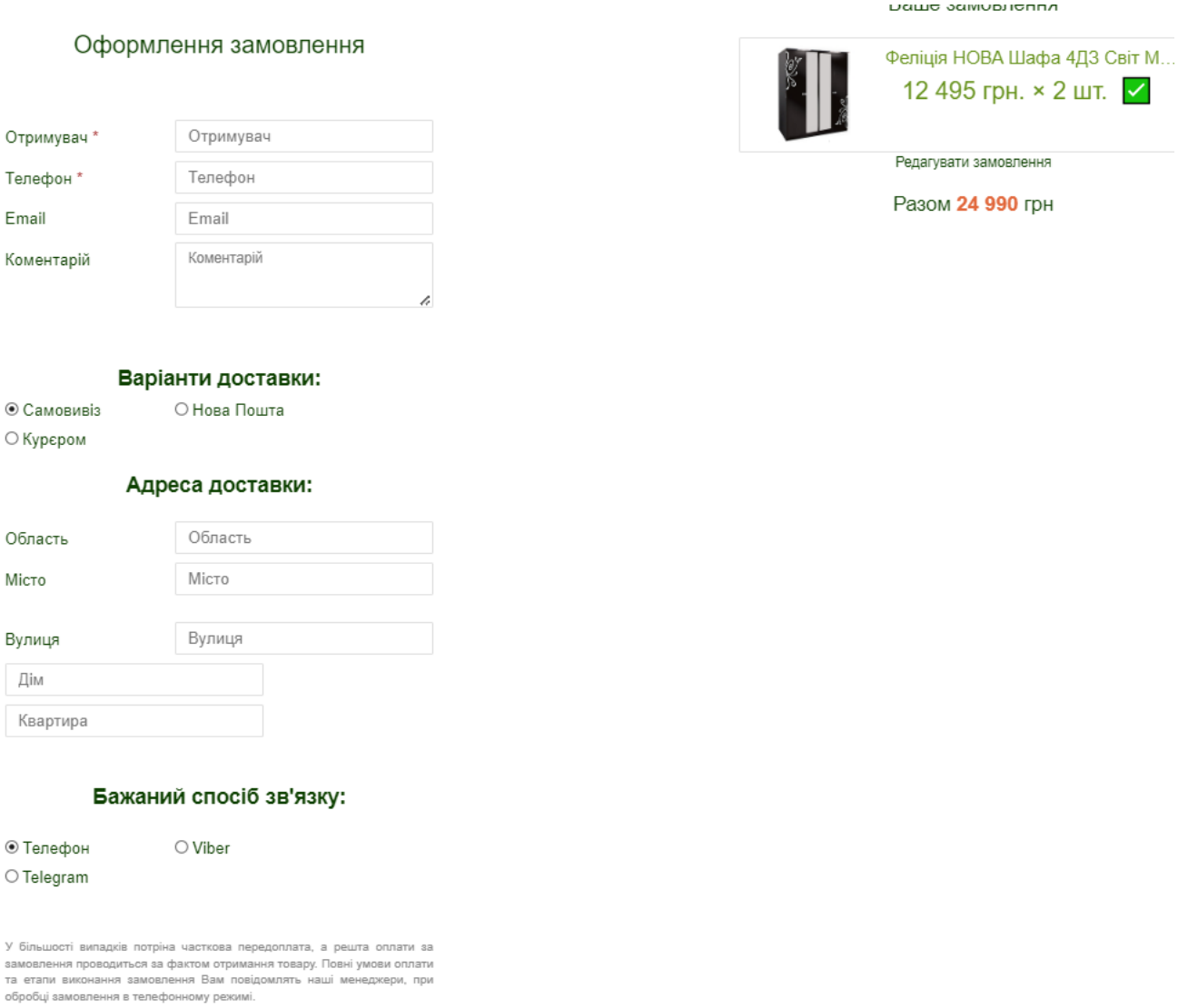


Рисунок 1.3 – Оформлення замовлення

Наступним проаналізуємо магазин DivanStar. Цей інтернет магазин виконаний у сучасному стилі, має приємний вигляд, виконаній у світлий кольорах. Сайт є інтуїтивно зрозумілим.

На головній сторінці можна побачити карусель (рис. 1.4), вона має 3 фото ,при натисканні на фото переносить користувача на відповідну частину сайту, таку як акції, доставка та оплата або же на товар дня.

Рисунок 1.4 – Головна сторінка вебсайту «DivanStar»

Замовлення товару (рис. 1.5). Для того щоб замовити товар потрібно вибрати його, а вже потім чекати дзвінка від менеджера, щоб його замовити, у свою чергу ,по статистиці, це є не зручно та застаріло.

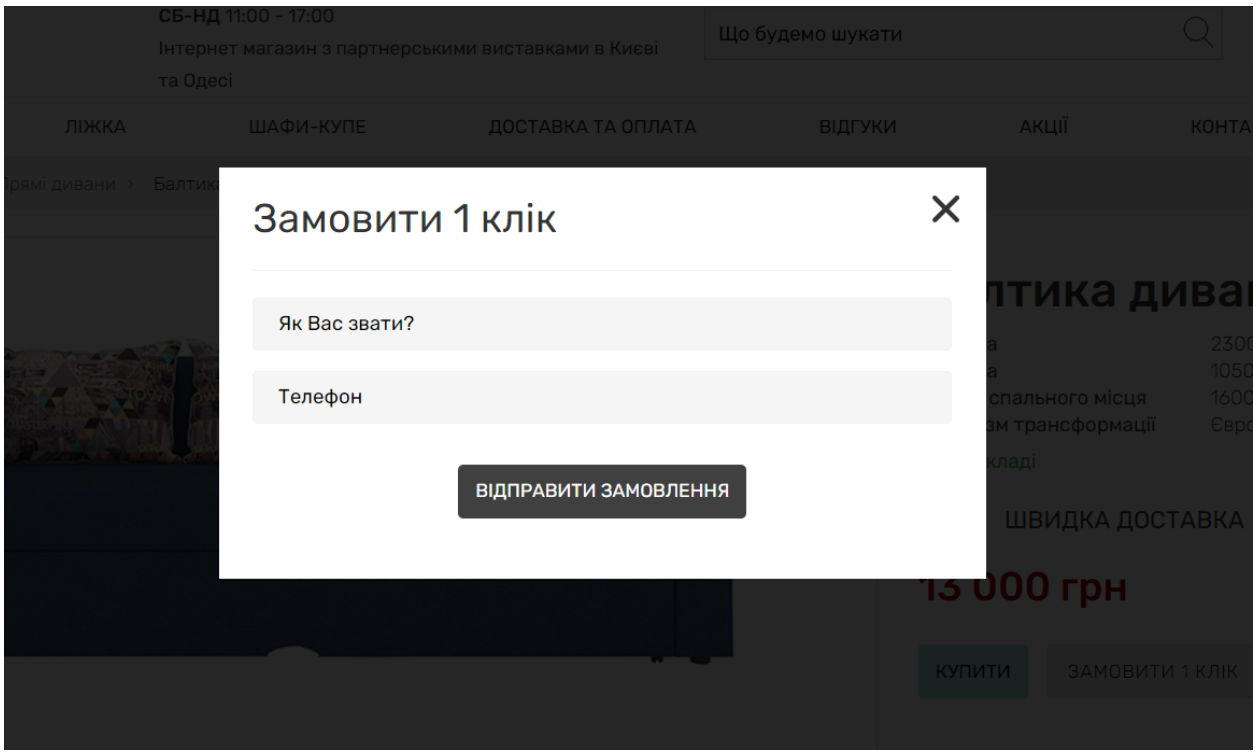


Рисунок 1.5 – Оформлення замовлення

Сторінка товару виглядає приємно (рис. 1.6). Вибір комплектації(категорії) по кольору та ціні, можна вибрати свою комбінації кольорів.

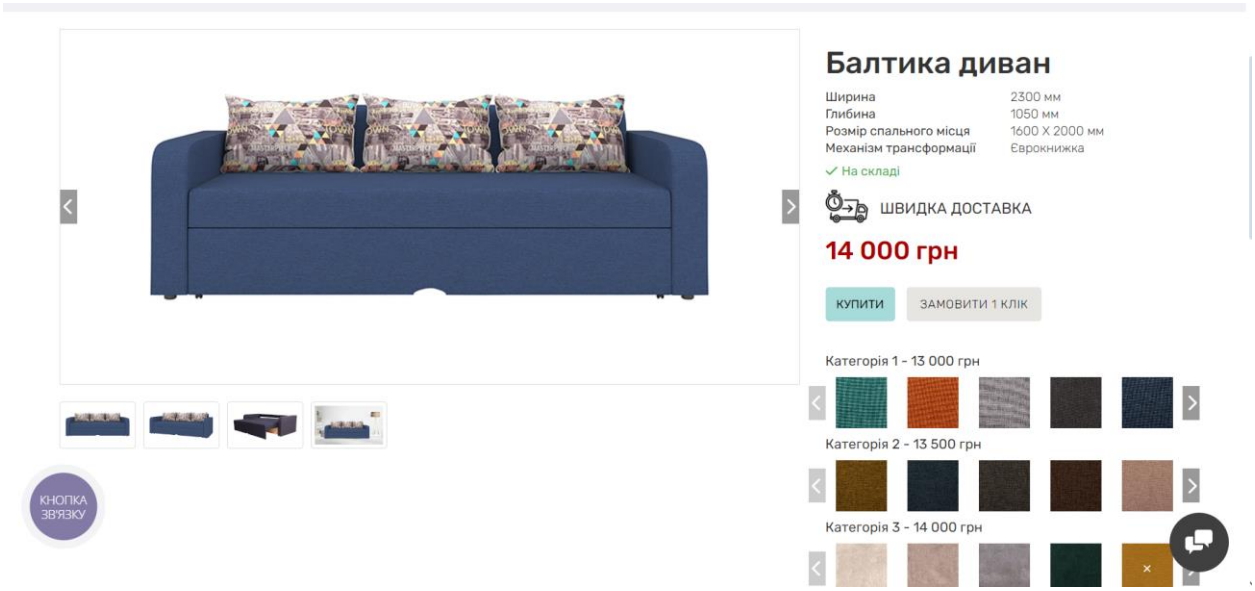


Рисунок 1.6 – Сторінка товару

Деякі товари які представлені в інтернет магазині мають відео ролик з детальним показом товару (рис. 1.7).

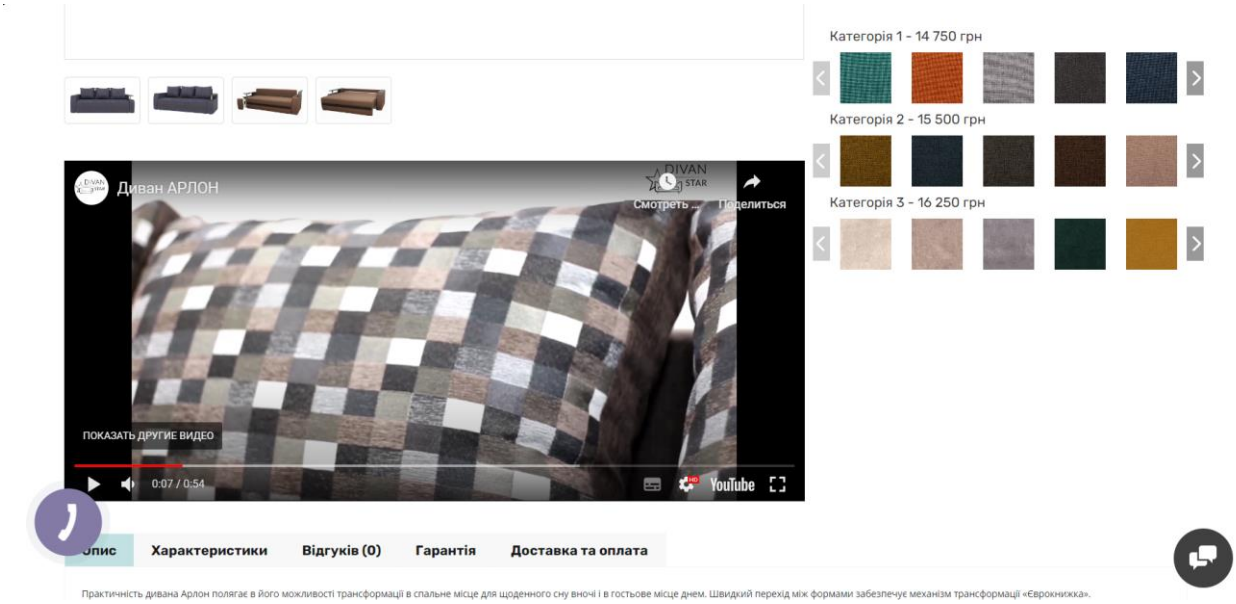


Рисунок 1.7 – Відео огляд товару

При аналізі інтернет магазину «Artos» (рис. 1.8) можна побачити що основні кольори це білий, чорний, золотий та червоний. Видно графік роботи, актуальні номери телефону для дзвінків. Відразу можна побачити всі категорії.

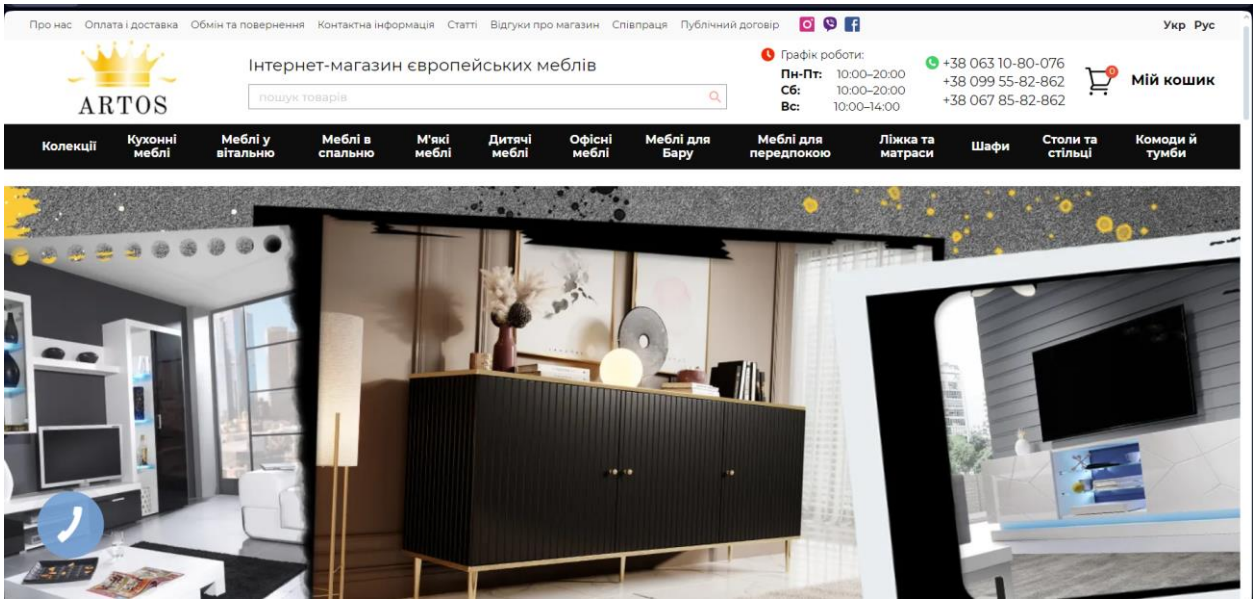


Рисунок 1.8 – Головна сторінка вебсайту «Artos»

При перегляді товарів у категоріях (рис. 1.9) можна побачити ,не натискаючи на товар, на скільки місяців можна взяти в кредит товар, в яких банках. Також знижки, новинки та хіт продажу. Цей функціонал спрощую знаходження потрібного товару.

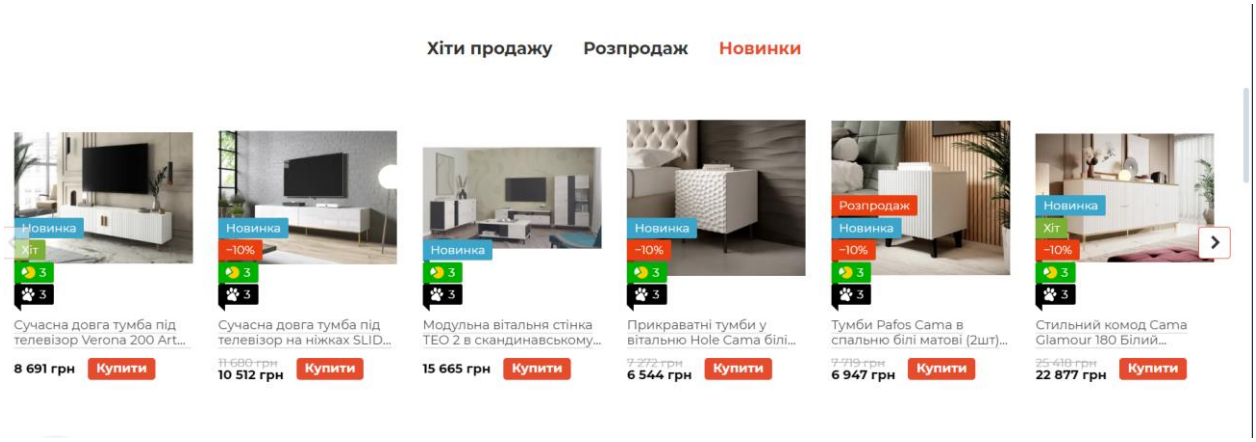


Рисунок 1.9 – Категорія новинки

Сторінка оформлення замовлень (рис. 1.10) є зручною. Є можливість оформити замовлення не реєструючись на сайті.

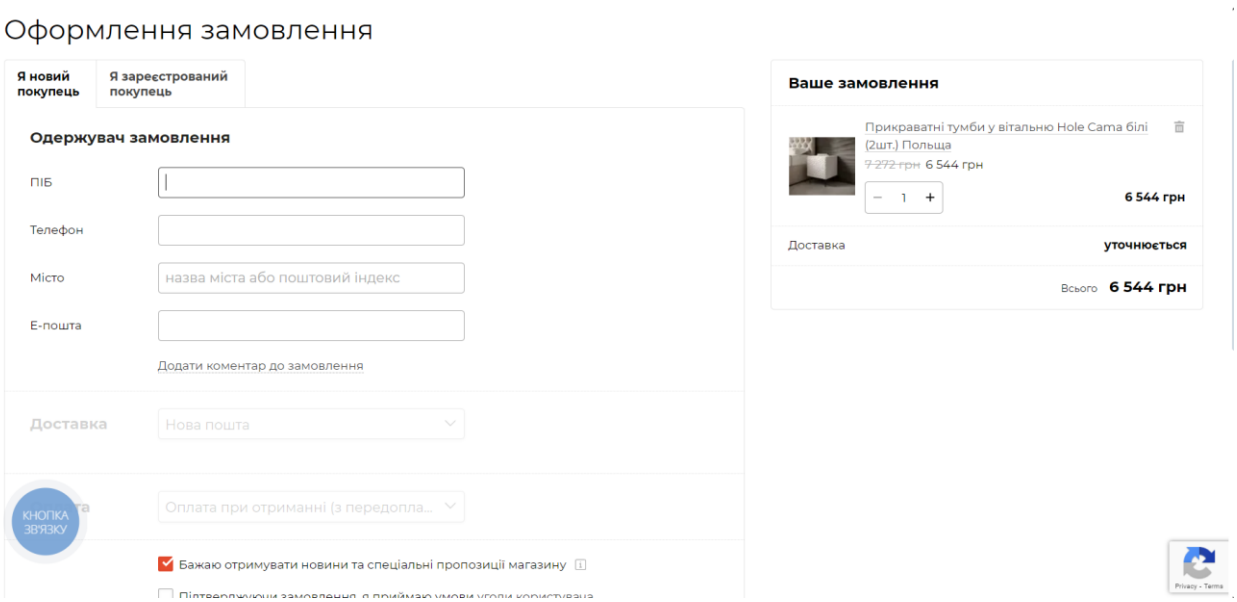


Рисунок 1.10 – Оформлення замовлення товару

Магазин має гнучкий фільтр товарів (рис. 1.11).

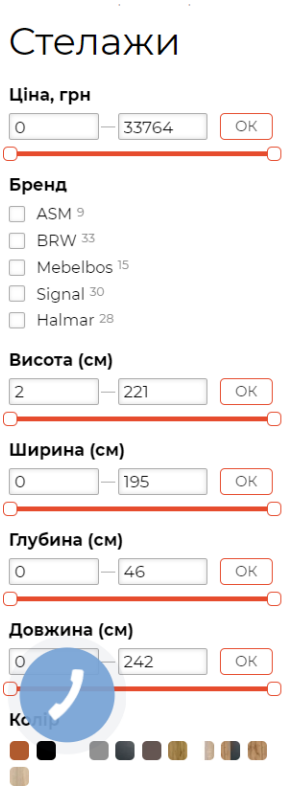


Рисунок 1.11 – Фільтр товарів

Після детального аналізу аналогів вебсайтів, було визначено їх переваги та недоліки. Його результати представлені в таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця характеристик аналогів вебсайтів.

Характеристика/ Заклад	«mebli.biz.ua»	«Artos»	«DivanStar»
Сучасний дизайн	-	+	+
Зручний інтерфейс	+	+	+
Інтерактивність	-	-	+
Функціональність	+	+	+
Навігація	+	+	+
Реєстрація користувачів	-	+	-
Можливість замовлення товару онлайн	+	+	+
Наявність відгуків	-	+	+

З даних, наведених у таблиці 1.1, можна відзначити цікаві доповнення, які можна використати, та недоліки, які необхідно подолати під час розробки. Програмний продукт повинен мати сучасний дизайн, зручну навігацію та інтерфейс, інтерактивність вебсторінок. Серед функціональних доповнень варто відзначити реєстрацію користувачів (але також повинна бути можливість оформлення замовлення не реєструючись), інтерактивність у вигляді перегляду товару в 3D.

1.3 Мета та задачі дослідження

Метою дослідження є розробка вебдодатку для підтримки діяльності магазину з продажу меблів «TamTumba». Його використання забезпечить автоматизація таких процесів як, формування замовлень, доставка товару, інтерактивний перегляд товару у трьох вимірному вигляді.

Основні вимоги до розроблюваного програмного продукту включають:

– організація зворотного зв’язку із адміністратором магазину за допомогою вебформи;

– створення вебформи для оформлення замовлень;

– реалізація інтерактивності, а саме перегляд товарів в 3D;

– зручний фільтр для пошуку товару/

Для досягнення цих завдань передбачено реалізацію таких етапів:

– аналіз актуальності роботи, визначення цільової аудиторії та вивчення предметної області;

– порівняльний аналіз аналогів інтернет магазинів з продажу меблів, визначення їх переваг та недоліків;

– проектування моделі та структури програмного продукту.

– вибір технологій для розробки сайту;

– створення прототипу та реалізація структури вебдодатку;

– розробка функціоналу для ефективної діяльності онлайн магазину;

– тестування продукту для впевненості в його ефективності;

Всі вимоги до проекту, структури вебдодатку та його функціонування детальніше описані у технічному завданні на розробку проекту (додаток А).

Для реалізації даного вебдодатку використовуються такі технології:

– фреймворк Django(Python) [9] для бекенду додатку. Фреймворк Django використовується для бекенду додатку через свою високу продуктивність, швидкодію та розширюваність. Django забезпечує широкі можливості для розробки

складних вебдодатків, включаючи вбудовану адміністративну панель, автентифікацію користувачів та безпеку даних;

– HTML для створення каркасу. HTML використовується для створення каркасу вебсторінок, оскільки це основна мова розмітки, яка дозволяє визначити структуру та зміст вмісту кожної сторінки;

– CSS для візуальної стилізації та адаптивності сайту [11]. Використовується для візуальної стилізації та адаптивності сайту. Ця технологія дозволяє задавати розміри, кольори, шрифти та інші візуальні параметри, щоб зробити сайт привабливим та зручним у використанні на різних пристроях.;

– JavaScript для динамічності та асинхронної взаємодії [12]. Використовується для динамічності та асинхронної взаємодії. За допомогою JavaScript можна реалізувати різноманітні ефекти, анімації, валідації форм, а також взаємодію з користувачем без перезавантаження сторінки.;

– SqlLite в якості СУБД для збереження та обробки даних. Використовується в якості системи управління базами даних для збереження та обробки даних. SQLite є простим у використанні, що робить його ідеальним вибором для невеликих до середніх вебдодатків.

2 ПРОЕКТУВАННЯ ВЕБДОДАТКУ

2.1 Структурно-функціональне моделювання

Функціональне моделювання для вебдодатку підтримки діяльності магазину з продажу меблів реалізується за методологією IDEF0 (Integrated Definition for Functional Modelling). Цей підхід базується на поділі системи на функціональні блоки та встановленні взаємозв'язків між ними. Кожен блок на діаграмі представляє окрему функцію, а стрілки вказують на інформаційні потоки та зв'язки між функціями [15].

Це моделювання дозволяє представити функції системи, їх взаємодію, вхідні/вихідні інформаційні потоки та учасників, які виконують ці функції. На рівні 0 весь процес відображається як функціональний блок з усіма пов'язаними з ним роботами та об'єктами управління. Така модель дозволяє аналізувати систему, ідентифікувати функції, визначати їх взаємозв'язки, оцінювати продуктивність та ефективність системи, виявляти можливі проблеми та вдосконалювати функції. На діаграмі також показані всі дані та вхідна інформація, необхідні для замовлення побутових меблів. На рисунку 2.1 представлено діаграму IDEF0.

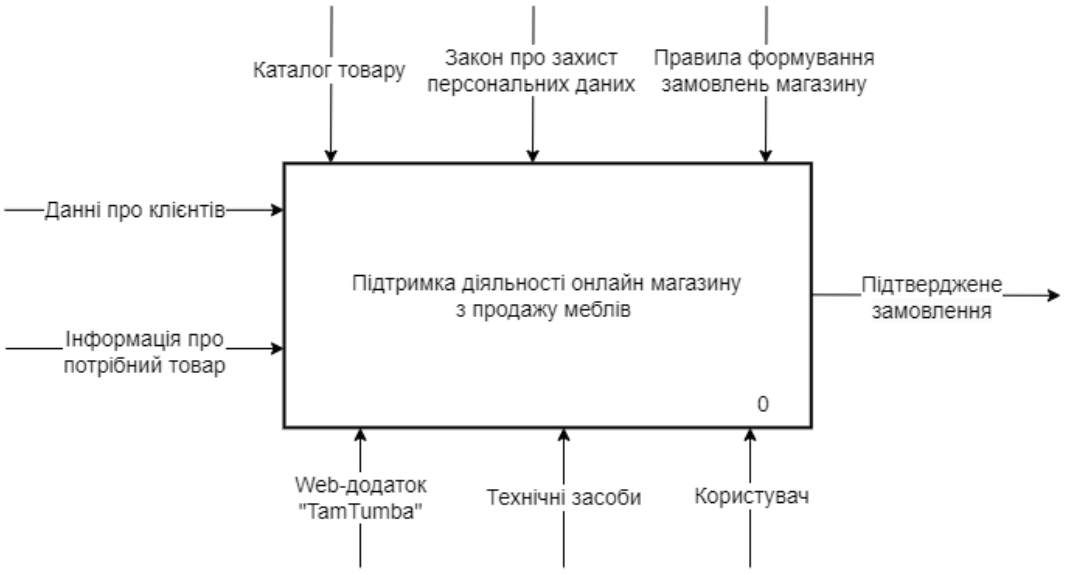


Рисунок 2.1 – Контекстна діаграма

Основний процес, що розглядається, це підтримка діяльності магазину з продажу меблів.

Вхідні дані.

Данні про клієнтів: Це може бути інформація про основні контактні дані клієнтів, такі як ім'я, адреса, номер телефону, або інші важливі дані, які визначають клієнта.

Інформація про потрібний товар: Включає в себе деталі про товар, який клієнт бажає придбати. Це може бути назва товару, кількість, характеристики і т.д.

Закон про захист персональних даних: Це обмеження встановлює вимоги та правила для збору, обробки та збереження особистих даних клієнтів. Система повинна дотримуватися цього законодавства, щоб гарантувати конфіденційність та безпеку персональної інформації.

Вихідні дані:

Підтверджене замовлення: Представляє собою результат обробки інформації про клієнтів та їхніх замовлень.

Механізми реалізації:

Каталог: Представляє собою обмеження або набір правил, які пов'язані з каталогом товарів. Може включати обмеження по наявності товарів, їх характеристики та інші параметри.

Правила формування замовлення магазину: Це набір правил магазину для коректного формування замовлень.

Вебдодаток “TamTumba” : Це інструмент або програмне забезпечення, яке надає можливість обробки інформації про клієнтів та їхні замовлення. Взаємодія з цим ресурсом дозволяє використовувати його функціонал для обробки даних та оформлення замовлень.

Технічні засоби: Включає у себе апаратне та програмне забезпечення, необхідне для функціонування вебдодатка «TamTumba».

Керуючі компоненти:

Користувач: Це особа або сутність, яка взаємодіє з системою, вводить дані про клієнтів та їх замовлення через вебдодаток «TamTumba».

Декомпозиція процесів, що мають виконуватися в додатку «TamTumba», є ключовим етапом для розуміння та ефективного впровадження функціоналу системи. Цей процес включає розділення процесів системи на чітко визначені етапи, щоб забезпечити логічну та ефективну роботу кожного компонента системи.

Декомпозиція функціональної моделі програмного продукту з продажу меблів для дому, яка представлена на рисунку 2.2, надає можливість відокремити та уточнити кожну функцію системи, розглядаючи її в контексті окремих етапів та взаємозв'язків з іншими функціями. Такий підхід сприяє кращому розумінню та управлінню процесами системи, що в свою чергу сприяє покращенню її функціональності та ефективності.

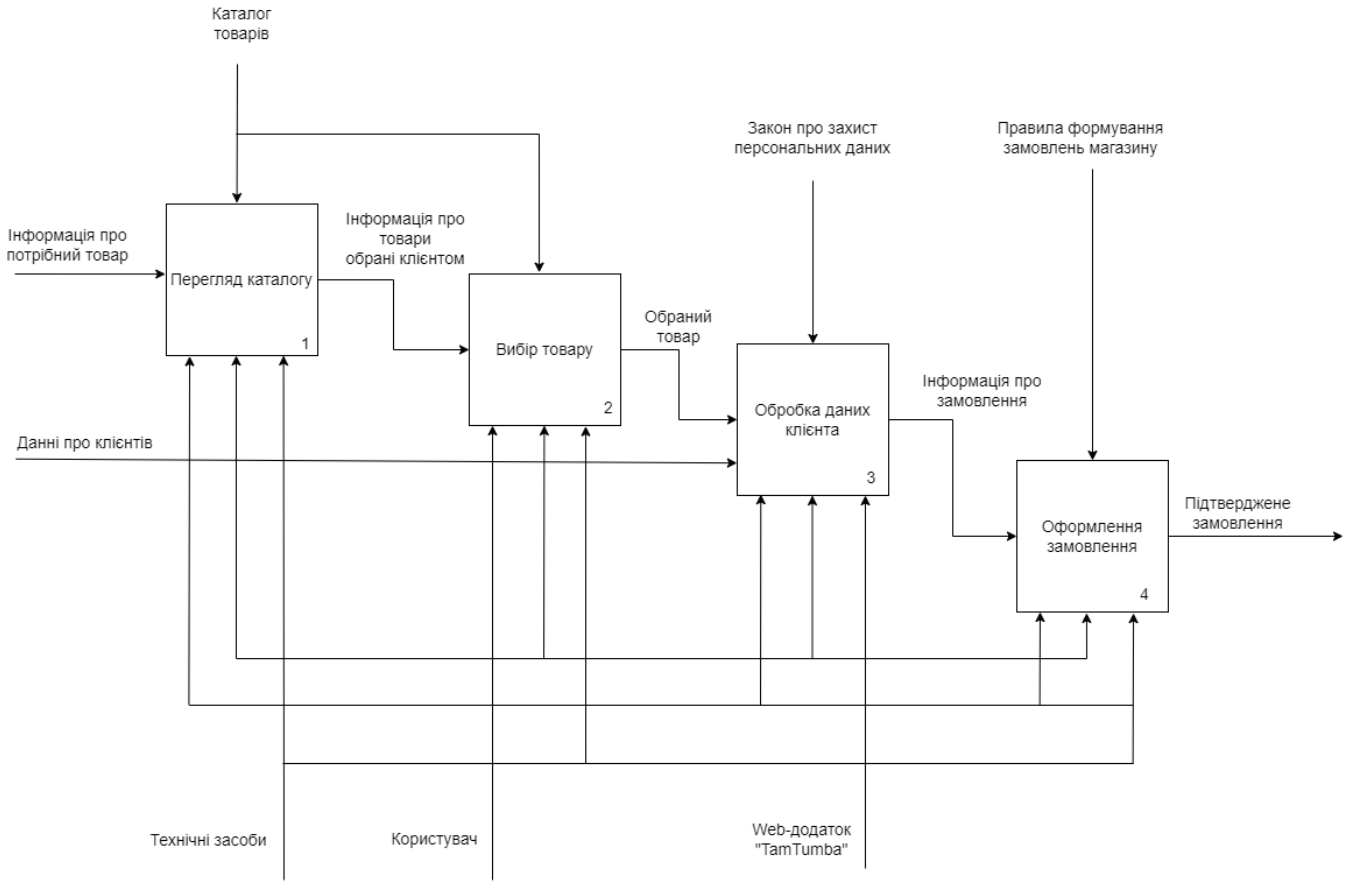


Рисунок 2.2 – Діаграма декомпозиції IDEF0 першого рівня

Нижче наведено деталі кожного процесу та його зв'язок з іншими аспектами системи.

Перегляд каталогу

Дія: Цей блок представляє процес перегляду каталогу товарів користувачем.

Входи:

Каталог товарів: Список доступних товарів.

Інформація про потрібний товар: Конкретні деталі або критерії, які шукає користувач.

Дані про клієнтів: Інформація про клієнта, яка може впливати на пропозиції або спосіб відображення товарів.

Виходи:

Інформація про товари обрані клієнтом: Деталі товарів, які зацікавили або були обрані клієнтом.

Вибір товару

Дія: Цей блок включає процес вибору товарів користувачем з каталогу.

Входи:

Інформація про товари обрані клієнтом: Деталі товарів, які зацікавили клієнта.

Дані про клієнтів: Деталі про клієнта, які можуть вплинути на процес вибору.

Виходи:

Обраний товар: Товар(и), який(і) був(ли) обраний(і) клієнтом для покупки.

Обробка даних клієнта

Дія: Цей блок займається обробкою даних клієнта відповідно до законодавства про захист персональних даних.

Входи:

Обраний товар: Товар, який обрав клієнт.

Дані про клієнтів: Інформація про клієнта.

Закон про захист персональних даних: Юридичні норми, які необхідно дотримуватися при обробці даних.

Виходи:

Інформація про замовлення: Оброблена інформація, готова для оформлення замовлення.

Оформлення замовлення

Дія: Цей блок представляє остаточний етап оформлення замовлення.

Входи:

Інформація про замовлення: Оброблена інформація, необхідна для завершення замовлення.

Правила формування замовлення магазину: Політики та процедури магазину щодо оформлення замовлень.

Виходи:

Підтверджене замовлення: Остаточне замовлення, яке підтверджено і готове до обробки магазином.

Додаткові елементи:

Технічні засоби: Представляє технічну інфраструктуру, що підтримує процес.

Користувач: Кінцевий користувач, який взаємодіє з системою.

Web-додаток "TamTumba": Веб-додаток, на платформі якого відбувається взаємодія.

Зв'язки між блоками

Блок 1 (Перегляд каталогу) забезпечує інформацію про товари обрані клієнтом, яка передається до блоку 2 (Вибір товару).

Блок 2 (Вибір товару) передає обраний товар до блоку 3 (Обробка даних клієнта).

Блок 3 (Обробка даних клієнта) надає інформацію про замовлення до блоку 4 (Оформлення замовлення).

Блок 4 (Оформлення замовлення) завершує процес, видаючи підтверджене замовлення.

2.2 UML МОДЕЛЮВАННЯ

Для реалізації конкретної фізичної системи потрібно мати всі елементи логічного опису представлені у вигляді конкретних матеріальних об'єктів. Фізичне подання моделі призначене для опису цих реальних елементів. У мові UML [16] це означає колекцію взаємопов'язаних елементів, яка включає програмне та апаратне забезпечення, а також персонал, необхідний для виконання спеціальних завдань. Для фізичного відображення моделей систем використовуються діаграми реалізації. Діаграма компонентів допомагає визначити архітектуру розроблюваної системи. Основними графічними елементами діаграми компонентів є компоненти, інтерфейси та їх взаємозалежності. Діаграма варіантів використання є вихідною концептуальною моделлю системи в процесі її проектування та розробки.

Діаграма варіантів використання «TamTumba» описує основні функції та можливості системи для користувачів. Кожен варіант використання представляє собою конкретний спосіб взаємодії користувача з системою. Діаграма варіантів використання в нотації UML представлена на рисунку 2.3.

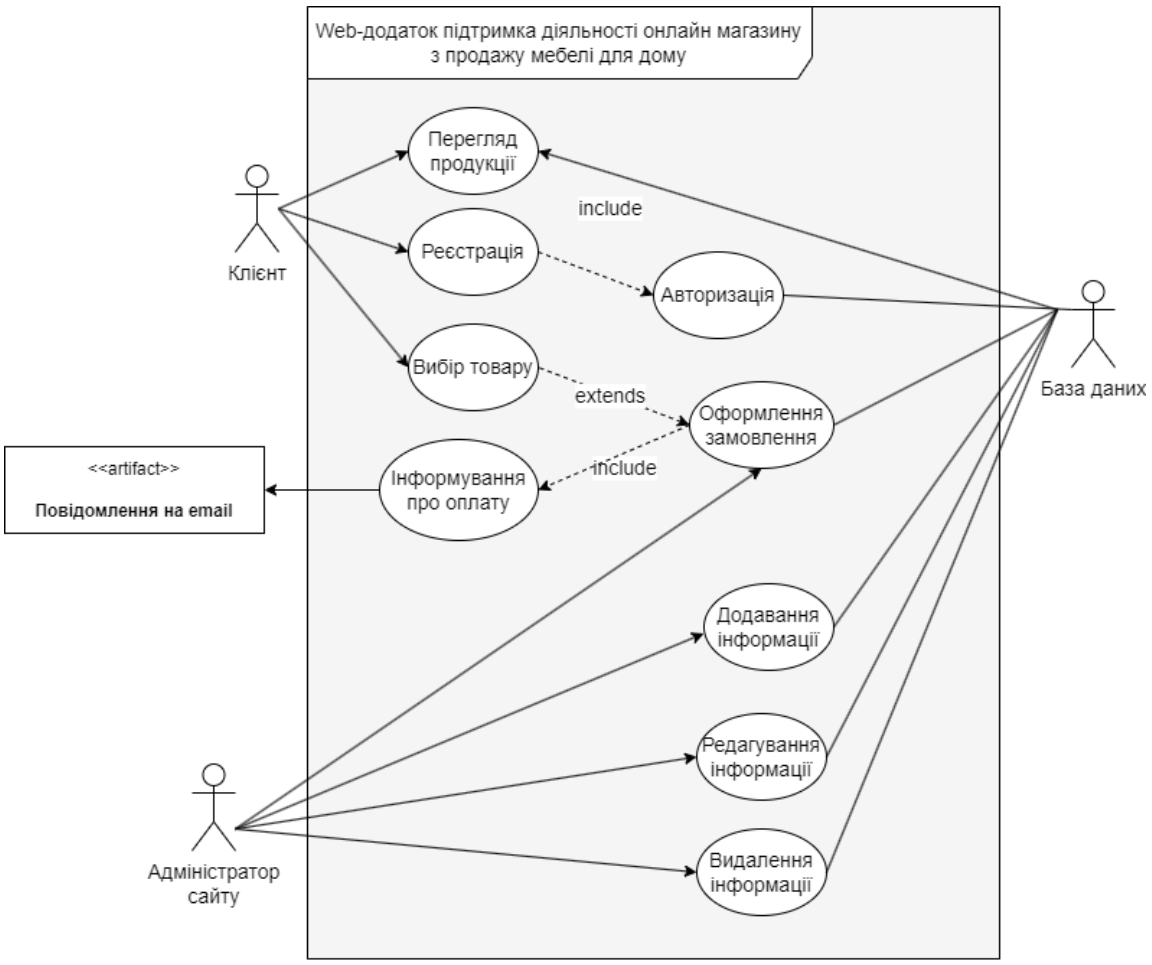


Рисунок 2.3 – Діаграма варіантів використання

Для перетворення абстрактної концепції системи у фізичну реалізацію необхідно мати опис всіх її складових у вигляді конкретних матеріальних об'єктів. У мові UML це відображається як сукупність зв'язаних елементів, що охоплюють програмне та апаратне забезпечення, а також персонал, необхідний для виконання певних завдань. Для такого фізичного опису використовуються діаграми реалізації, зокрема діаграми компонентів, які дозволяють визначити архітектуру системи через компоненти, інтерфейси та їх взаємозв'язки.

На рисунку 2.4 представлена діаграма компонентів вебдодатку.

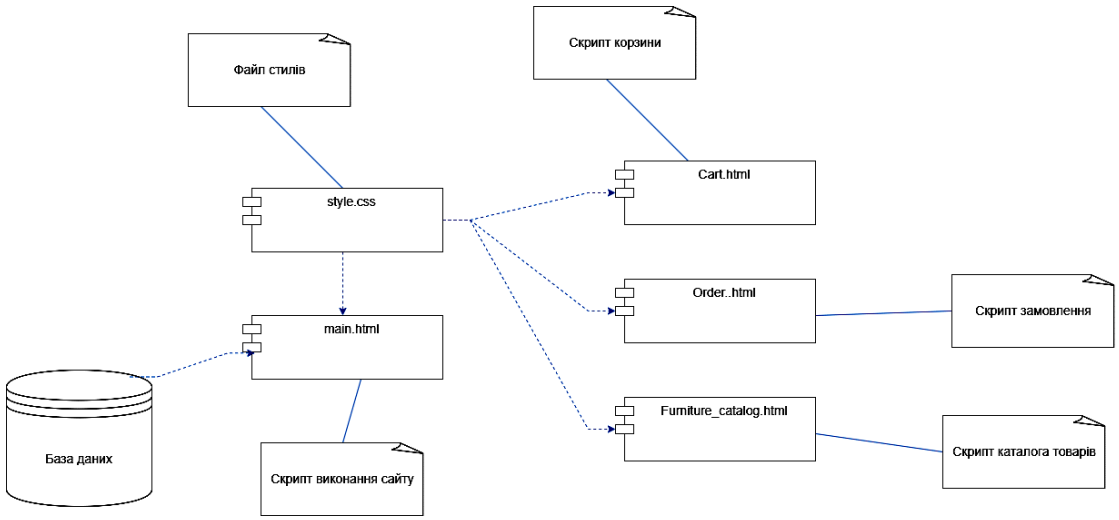


Рисунок 2.4 – Діаграма компонентів

2.3. Проектування логічної моделі даних проекту

Логічна модель даних визначає структуру та взаємозв'язки між об'єктами даних в системі [17]. На рисунку 2.5 представлена логічна модель даних вебдодатку для підтримки діяльності магазину з продажу меблів для дому «TamTumba».

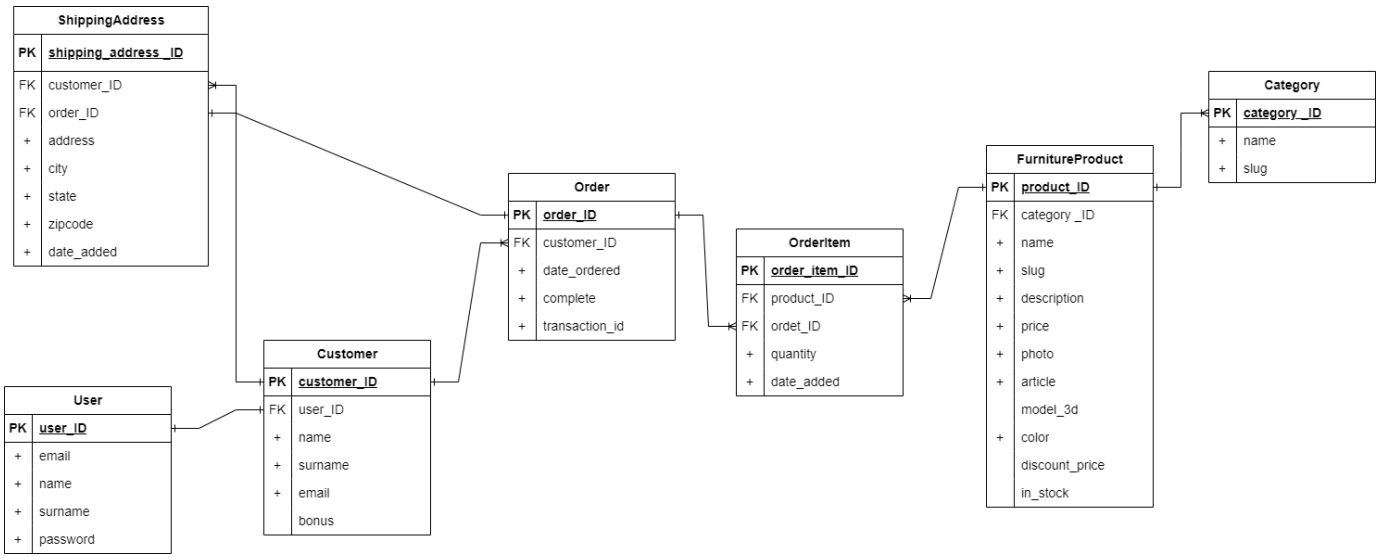


Рисунок 2.5 – Логічна модель даних

Таблиці бази даних:

- ShippingAddress: ця таблиця зберігає інформацію про адреси доставки для замовлень. Кожний запис містить унікальний ідентифікатор адреси доставки (shipping_address_ID), а також зв'язки з ідентифікаторами клієнта (customer_ID) та замовлення (order_ID). Поля таблиці включають адресу доставки (address), місто (city), штат (state), поштовий індекс (zipcode) та дату додавання (date_added);
- User: ця таблиця містить дані про користувачів системи. Кожний користувач ідентифікується унікальним ідентифікатором користувача (user ID). Поля таблиці включають електронну пошту користувача (email), ім'я (name) та прізвище (surname);
- Customer: ця таблиця зберігає дані про клієнтів, які є користувачами системи. Кожен клієнт має унікальний ідентифікатор клієнта (customer_ID), який пов'язаний з ідентифікатором користувача (user_ID). Поля таблиці включають ім'я (name), прізвище (surname), електронну пошту (email), бонуси (bonus) та пароль (password);
- Order: ця таблиця містить дані про замовлення, зроблені клієнтами. Кожне замовлення має унікальний ідентифікатор замовлення (order ID), який пов'язаний з ідентифікатором клієнта (customer_ID). Поля таблиці включають дату замовлення (date_ordered), статус замовлення (complete), ідентифікатор транзакції (transaction_id);
- FurnitureProduct: ця таблиця містить дані про меблеві товари. Кожен товар має унікальний ідентифікатор товару (product_ID), який пов'язаний з ідентифікатором категорії (category_ID);
- Category: ця таблиця містить дані про категорії товарів. Кожна категорія має унікальний ідентифікатор категорії (category ID). Поля таблиці включають назву категорії (name) та слаг (slug);
- OrderItem: кожен елемент має унікальний ідентифікатор елемента замовлення (order_item_ID). Поля таблиці включають назву (name), опис (description), ціну (price), фото (photo), кількість (quantity), дату додавання

(date_added), артикул (article), 3D модель (model_3d), колір (color), знижену ціну (discount_price) та наявність на складі (in_stock).

Зв'язки між таблицями:

- кожен Customer має один користувач (User), і кожен користувач може мати лише одного Customer;
- кожен FurnitureProduct належить до однієї Category, але кожна категорія може мати багато продуктів;
- кожен Order має одного Customer, але кожен клієнт може мати багато замовлень;
- кожен OrderItem пов'язаний з одним FurnitureProduct та одним Order, і кожен Order може мати багато елементів замовлення;
- кожна ShippingAddress пов'язана з одним Customer та одним Order, і кожен Order може мати одну адресу доставки.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБДОДАТКУ

3.1 Архітектура вебдодатку

Вебдодаток реалізований на основі фреймворку Django з використанням бази даних SQLite. Інтерфейс створено з використанням HTML, CSS та JavaScript. Для інтерактивного перегляду товарів у 3D використовується бібліотека Three.js. На рисунку 3.1 зображена архітектура вебдодатку.

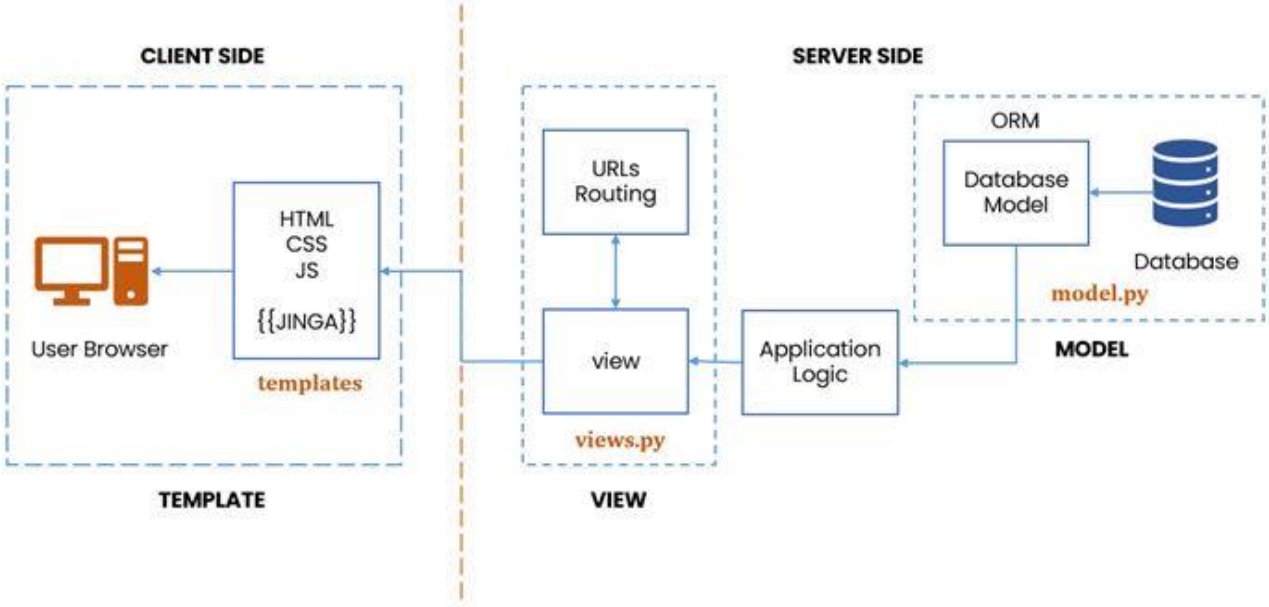


Рисунок 3.1 – Архітектура вебдодатку «TamTumba»

MTV означає Model-Template-View, який є шаблоном проектування, який використовується в Django. Це різновид шаблону Model-View-Controller (MVC), який розділяє програму на три взаємопов’язані компоненти для полегшення обслуговування та масштабованості[18].

У Django модель представляє схему бази даних і дані, які в ній зберігаються. Представлення відповідає за обробку HTTP-запитів і повернення відповідей. Шаблон використовується для створення HTML-сторінок, які надсилаються назад користувачеві.

Модель: визначає схему бази даних і містить методи для взаємодії з даними. Моделі зазвичай визначаються у файлі models.py програми.

Перегляд: обробляє HTTP-запити та повертає HTTP-відповіді. Представлення можна визначити як функції або класи, і зазвичай вони зберігаються у файлі views.py програми.

Шаблон: визначає структуру HTML-сторінок, які повертаються користувачеві. Шаблони зазвичай зберігаються в каталозі шаблонів програми.

Разом ці компоненти забезпечують чіткий розподіл завдань і полегшують підтримку та розширення програми Django.

3.2 Реалізація вебдодатку

Для того щоб розпочати створення вебдодатку потрібно встановити фреймворк Django в проект, це робиться за допомогою команди **pip install django.** На рисунку 3.2 продемонстровано процес встановлення пакету.

```
Collecting Django
  Downloading Django-4.0.3-py3-none-any.whl (8.0 MB)
     |████████████████████████████████| 8.0 MB 2.2 MB/s
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.4.2-py3-none-any.whl (42 kB)
Collecting asgiref<4,>=3.4.1
  Downloading asgiref-3.5.0-py3-none-any.whl (22 kB)
Collecting tzdata; sys_platform == "win32"
  Downloading tzdata-2021.5-py2.py3-none-any.whl (339 kB)
     |████████████████████████████████| 339 kB 6.4 MB/s
Installing collected packages: sqlparse, asgiref, tzdata, Django
Successfully installed Django-4.0.3 asgiref-3.5.0 sqlparse-0.4.2 tzdata-2021.5
WARNING: You are using pip version 20.2.3; however, version 22.3 is available.
You should consider upgrading via the 'C:\Users\Your Name\myworld\Scripts\python.exe -m pip install --upgrade
pip' command.
```

Рисунок 3.2 – Встановлення фреймворку Django

Після встановлення Django потрібно створити новий проект Django за допомогою команди django-admin.

```
(venv) PS F:\pythonProject1> django-admin startproject furnituresite
```

Рисунок 3.3 – Створення проекту

В Django працює з додатками, які є частинами проекту. Щоб створити новий додаток, потрібно використати наступну команду:

```
(venv) PS F:\pythonProject1> python manage.py startapp main
```

Рисунок 3.3 – Створення додатку

Після цього було створено та налаштовані інші необхідні додатки проекту. Результат попередніх налаштувань продемонстровано на рисунку 3.4

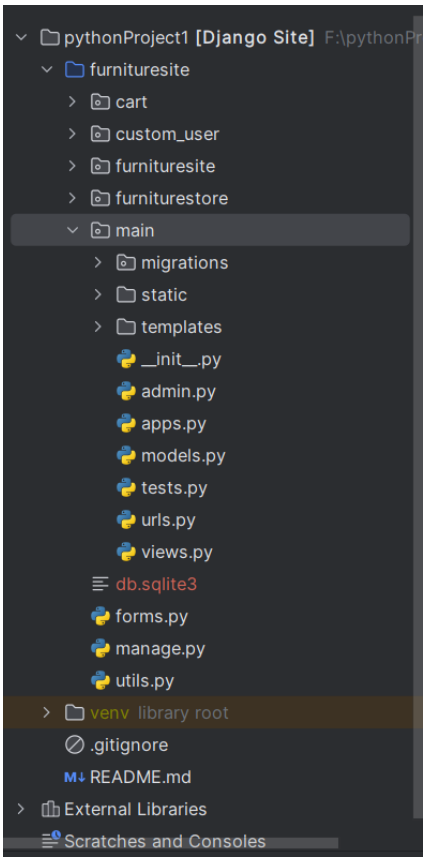


Рисунок 3.4 – Створення проекту

Проект складається з наступних елементів:

- migrations: призначена для зберігання міграцій – скриптів, які дозволяють синхронізувати структуру бази даних із визначенням моделей;
- _init__.py : вказує інтерпретатору python, що поточний каталог буде розглядатися як пакет;
- admin.py : призначений для адміністративних функцій, зокрема тут наводиться реєстрація моделей, які використовуються в інтерфейсі адміністратора;
- apps.py : визначає конфігурацію програми;
- models.py : зберігає визначення моделей, які описують дані, що використовуються в додатку;
- tests.py : зберігає тести програми;
- views.py : визначає функції, які отримують запити користувачів, обробляють їх та повертають відповідь;
- додаток cart : зберігає в собі функціонал корзини та обробки замовлення;
- додаток custome_user : відповідає за зміну функціоналу авторизації в адмін панель, та реєстрації на сайті, в основному змією те що потрібно реєструватися в додатку не за допомогою логіна та паролю, а пошти та паролю;
- додаток furniturestore : виконує функціонал каталогу товару;
- додаток main : зберігає в собі необхідні файли стилів, статичні файли ,такі як картинки, 3d моделі, .js файли.

Розберемо файл base.html який відповідає за базову розмітку сайту для всіх сторінок.

{% load static %}: Це тег шаблону Django, який дозволяє використовувати статичні файли, такі як CSS, JavaScript та зображення. Ці дані зберігається в папці main. На рисунку 3.5 зображено де зберігаються статичні файли.

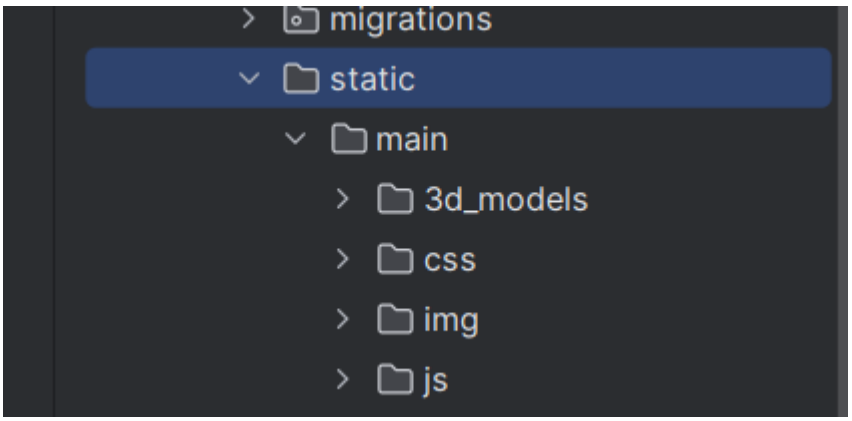


Рисунок 3.5 – Папка static

Далі переходимо в тег head. Це розділ заголовку, де ми встановлюємо загальні параметри сторінки, такі як заголовок, метатеги, посилання на зовнішні стилі і скрипти.

```
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, minimum-scale=1" />
<title>TamTumba</title>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.0/jquery.min.js"
        integrity="sha512-3gJwYpMe3QewGELv8k/BX9vcqhryRdzRMxVfq6ngyWXwo03GFEzjsUm8Q7RZcHPHksttq7/GFoxjCVUjkjvPdw=="
        crossorigin="anonymous" referrerpolicy="no-referrer"></script>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css"
      integrity="sha384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65"
      crossorigin="anonymous">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css"
      integrity="sha512-iecdLmaskl7CVkqkXNQ/ZH/XLlvWZOJyj7Yy7tcenmpD1ypASozpmT/E0iPtmFIB46ZmdtAc9eNBvH0H/ZpiBw=="
      crossorigin="anonymous" referrerpolicy="no-referrer" />
<link rel="stylesheet" href="{% static 'main/css/style.css' %}" >
<link rel="icon" href="{% static 'main/img/icon.ico' %}" type="image/png">
```

Рисунок 3.6 – Тег head

Далі було реалізовано скрипт, який відповідає за кукі. Основна ціль - дозволити обробляти корзину для неавторизованого користувача через кукі файли. Тобто, якщо користувач неавторизований додає товари до корзини , а потім вийде з сайту та зайде пізніше, якщо він не почистив файли кукі, то тоді в нього досі буде відображатися його додані товари в корзині.

```
<script type="text/javascript">
    var user='{{request.user}}'
    function getToken(name) {
        let cookieValue = null;
        if (document.cookie && document.cookie !== '') {
            const cookies = document.cookie.split(';');
            for (let i = 0; i < cookies.length; i++) {
                const cookie = cookies[i].trim();
                // Does this cookie string begin with the name we want?
                if (cookie.substring(0, name.length + 1) === (name + '=')) {
                    cookieValue = decodeURIComponent(cookie.substring(name.length + 1));
                    break;
                }
            }
        }
        return cookieValue;
    }
    const csrftoken = getToken('csrftoken');

    function getCookie(name){
        var cookieArr = document.cookie.split(";");
        for(var i = 0;i<cookieArr.length;i++){
            var cookiePair = cookieArr[i].split("=");

            if(name == cookiePair[0].trim()){
                return decodeURIComponent(cookiePair[1]);
            }
        }
        return null;
    }
    var cart = JSON.parse(getCookie('cart'))
    if(cart == undefined){
        cart = {}
        console.log('Cart was created')
        document.cookie = 'cart=' + JSON.stringify(cart) + ";domain=;path=/"
    }
    console.log('cart:',cart)
</script>
```

Рисунок 3.7 – Javascript функція для обробки кукі файлів

var user='{{request.user}}': Цей рядок JavaScript отримує ім'я користувача з Django через шаблон Django, яке вставляється у фрагмент JavaScript. Він використовується для отримання інформації про поточного користувача.

function getToken(name): Ця функція отримує значення кукі зазначеного імені. Вона перебирає всі кукі на сторінці та повертає значення, якщо ім'я кукі збігається з вказаним іменем.

const csrftoken = getToken('csrftoken'): Цей рядок отримує токен захисту від CSRF (Cross-Site Request Forgery), який зазвичай використовується для захисту форм від несанкціонованого відправлення. Цей токен може бути використаний при відправці AJAX-запитів.

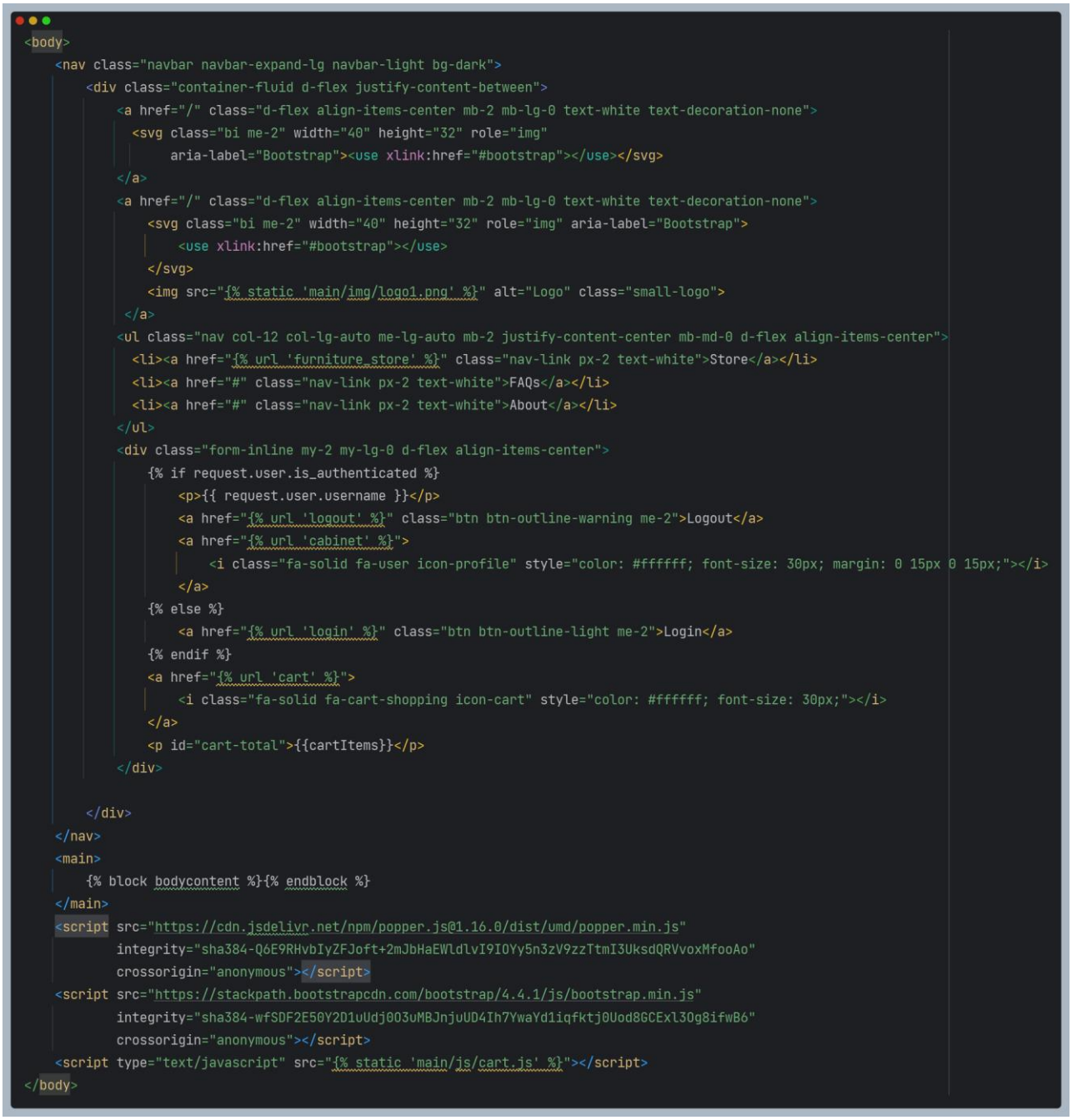
function getCookie(name): Ця функція схожа на попередню, але призначена для отримання будь-якого кукі за ім'ям.

var cart = JSON.parse(getCookie('cart')): Цей рядок отримує кукі з іменем 'cart' і перетворює їх з формату JSON у JavaScript-об'єкт. Кукі 'cart', ймовірно, містять інформацію про вміст корзини покупок користувача.

if(cart == undefined): Ця умова перевіряє, чи існує об'єкт корзини. Якщо корзина ще не існує (це може статися при першому відвідуванні сторінки або якщо користувач ще не додавав товари до корзини), то створюється порожня корзина і зберігається у вигляді кукі.

console.log('cart:',cart): Цей рядок виводить зміст корзини в консоль для налагодження та відладки.

Перейдемо до тегу body. Його приклад зображено на рисунку 3.8.



```
<body>
    <nav class="navbar navbar-expand-lg navbar-light bg-dark">
        <div class="container-fluid d-flex justify-content-between">
            <a href="/" class="d-flex align-items-center mb-2 mb-lg-0 text-white text-decoration-none">
              <svg class="bi me-2" width="40" height="32" role="img"
                   aria-label="Bootstrap"><use xlink:href="#bootstrap"></use></svg>
            </a>
            <a href="/" class="d-flex align-items-center mb-2 mb-lg-0 text-white text-decoration-none">
                <svg class="bi me-2" width="40" height="32" role="img" aria-label="Bootstrap">
                    <use xlink:href="#bootstrap"></use>
                </svg>
                <img src="{% static 'main/img/logo1.png' %}" alt="Logo" class="small-logo">
             </a>
            <ul class="nav col-12 col-lg-auto me-lg-auto mb-2 justify-content-center mb-md-0 d-flex align-items-center">
              <li><a href="{% url 'furniture_store' %}" class="nav-link px-2 text-white">Store</a></li>
              <li><a href="#" class="nav-link px-2 text-white">FAQs</a></li>
              <li><a href="#" class="nav-link px-2 text-white">About</a></li>
            </ul>
            <div class="form-inline my-2 my-lg-0 d-flex align-items-center">
                {% if request.user.is_authenticated %}
                    <p>{{ request.user.username }}</p>
                    <a href="{% url 'logout' %}" class="btn btn-outline-warning me-2">Logout</a>
                    <a href="{% url 'cabinet' %}">
                        <i class="fa-solid fa-user icon-profile" style="color: #ffffff; font-size: 30px; margin: 0 15px 0 15px;"></i>
                    </a>
                {% else %}
                    <a href="{% url 'login' %}" class="btn btn-outline-light me-2">Login</a>
                {% endif %}
                <a href="{% url 'cart' %}">
                    <i class="fa-solid fa-cart-shopping icon-cart" style="color: #ffffff; font-size: 30px;"></i>
                </a>
                <p id="cart-total">{{cartItems}}</p>
            </div>

        </div>
    </nav>
    <main>
        {% block bodycontent %}{% endblock %}
    </main>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
            integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
            crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
            integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
            crossorigin="anonymous"></script>
    <script type="text/javascript" src="{% static 'main/js/cart.js' %}"></script>
</body>
```

Рисунок 3.7 – Тег body

<nav class="navbar navbar-expand-lg navbar-light bg-dark">...</nav>: Цей розділ визначає навігаційне меню. Він використовує класи Bootstrap для стилізації та розширення. Меню розміщене в темному фоні.

...: Ці посилання ведуть на головну сторінку. Вони містять зображення, які використовуються для стилізації та іконка логотипу.

<ul class="nav col-12 col-lg-auto me-lg-auto mb-2 justify-content-center mb-md-0 d-flex align-items-center">...: Цей розділ містить список посилань на різні

розділи вашого сайту, такі як "Store", "FAQs" та "About". Він вирівняний по центру та містить класи Bootstrap для адаптивного розміщення.

... : Цей розділ містить елементи управління користувачем, такі як кнопки входу/виходу, логін, корзина тощо. Він також містить інформацію про кількість товарів у корзині.

<main>...</main>: Цей розділ вставляє основний вміст сторінки, який визначається через шаблонний тег Django {% block bodycontent %}{% endblock %}. Це місце, де буде відображатися зміст конкретної сторінки в залежності від того, який шаблон використовується.

<script>...</script>: Цей розділ містить посилання на зовнішні JavaScript-файли, такі як Popper.js, Bootstrap та ваш власний файл cart.js. Вони використовуються для додавання додаткової функціональності до вашого сайту, такої як анімації, модальні вікна тощо.

Цей код визначає загальну структуру вашої сторінки, включаючи навігаційне меню, основний вміст та підключення JavaScript-файлів для покращення функціональності сторінки.

Розглянемо реалізацію каталогу товарів.

На рисунку 3.8 зображено частину коду який визначає базову структуру сторінки каталогу товарів. Він розширює основний шаблон **main/base.html** і використовує блок **bodycontent** для вставки основного контенту сторінки.

```
{% extends 'main/base.html' %}
{% load static %}
{% block bodycontent %}
<div class="container">
    <div class="row">
        <!-- Sidebar for filters -->
        <div class="col-lg-1 col-md-2 col-sm-3 order-lg-first">
            ...
        </div>
        <!-- Product grid -->
        <div class="col-lg-10 col-md-10 col-sm-9 order-lg-last product-grid" style="margin-left: 50px">
            ...
        </div>
    </div>
</div>
...
{% endblock %}
```

Рисунок 3.8 – Базова структура каталогу товарів

Розділ коду, який представлено на рисунку 3.9, відповідає за фільтр за ціною. Використовується форма для введення мінімальної та максимальної ціни. Також використовується слайдер (створюється за допомогою noUiSlider) для зручного вибору цінового діапазону. Кнопка OK відправляє форму, але спочатку перевіряє правильність введених даних.

```
<form class="price-filter-form" method="get" action="{% url 'furniture_store' %}" onsubmit="return validatePriceFilter()">
    <div>
        <input type="text" id="min_price" name="min_price">
        - <input type="text" id="max_price" name="max_price">
    </div>
    <div id="priceSlider"></div>
    <button type="submit" id="filterButton" disabled>OK</button>
    {% if price_filter_form.errors %}
        <p class="error">{{ price_filter_form.errors.min_price }}</p>
        <p class="error">{{ price_filter_form.errors.max_price }}</p>
    {% endif %}
</form>
```

Рисунок 3.9 – Фільтр за ціною

Кожен товар відображається як окремий блок з зображенням, назвою, кнопкою "Add to Cart" або повідомленням "Out of Stock", кнопкою для перегляду деталей та ціною (зі знижкою або без). На рисунку 3.10 представлена частина коду для реалізації відображення товару.

```
<div class="col-lg-4">
    <img class="thumbnail" src="{{ product.imageURL }}">
    <div class="box-element product">
        <h6><strong>{{ product.name }}</strong></h6>
        <hr>
        {% if not product.in_stock %}
            <p style="display: inline-block; float: left; color: red; margin: 0px 10px 0px 0px;">Out of Stock :(</p>
        {%else%}
            <button data-product={{ product.id }} data-action="add" class="btn btn-outline-secondary add-btn update-cart">Add to Cart</button>
        {% endif %}
        <a class="btn btn-outline-success" href="{% url 'product_detail' product.slug %}">View</a>
        {% if product.discount_price %}
            <span style="display: inline-block; float: right">
                <del>${{ product.price|floatformat:0 }}</del>
            </span>
            <h4 style="display: inline-block; float: right; color: red;"><strong>${{ product.discount_price|floatformat:0 }}</strong></h4>
        {% else %}
        <h4 style="display: inline-block; float: right"><strong>${{ product.price|floatformat:0 }}</strong></h4>
        {% endif %}
    </div>
</div>
```

Рисунок 3.10 – Відображення товарів в каталозі

Для того, щоб відображати меблі у трьох вимірному вигляді, було встановлено бібліотеку Three.js на основі WebGL. Текст програми який зображений на рисунку 3.11 відповідає за завантаження необхідних модулів з бібліотеки Three.js для роботи з 3D-графікою, включаючи засоби керування камерою (OrbitControls), завантаження моделей (OBJLoader, MTLLoader) та освітлення (RectAreaLightHelper, RectAreaLightUniformsLib). Також цей код отримує параметри URL, щоб визначити URL-адресу 3D-моделі (3d_url) та прапорець view_3d, який вказує, чи потрібно ініціалізувати 3D-відображення.

```
import * as THREE from 'three';
import { OrbitControls } from 'OrbitControls';
import { OBJLoader } from 'OBJLoader';
import { MTLLoader } from 'MTLLoader';
import { RectAreaLightHelper } from 'RectAreaLightHelper';
import { RectAreaLightUniformsLib } from 'RectAreaLightUniformsLib';

const urlParams = new URLSearchParams(window.location.search);
const view3d_url = urlParams.get('3d_url');
const view3D = urlParams.get('view_3d');
let isRotating = false;
let container = document.querySelector('.container');
```

Рисунок 3.11 – Підключення файлів Three.js

Для того, щоб забезпечити оформлення замовлення, було створено корзину та сторінку з оплатою товарів.

Реалізації корзини товарів реалізована в файлі cart.html. На рисунку 3.12 продемонстровано частину коду яка відповідає за реалізації корзини. Використовуються шаблонні теги та фільтри Django для відображення динамічного вмісту, такого як товари в кошику, їх кількість та загальна сума.

```
<div class="container">
    <div class="row">
        <div class="col-lg-12">
            <div class="box-element">
                <a class="btn btn-outline-dark" href="{% url 'furniture_store' %}">&#x2190; Continue Shopping</a>
                <br>
                <br>
                <table class="table">
                    <tr>
                        <th><h5>Items: <strong>{{order.get_cart_items}}</strong></h5></th>
                        <th><h5>Total: <strong>{{order.get_cart_total|floatformat:2}} $</strong></h5></th>
                        <th>
                            {% if order.get_cart_items == 0 %}
                                <p style="color:red;">Your cart is empty.
                                    Please add items before proceeding to checkout.</p>
                            {% else %}
                                <a style="float:right; margin:5px;" class="btn btn-success" href="{% url 'checkout' %}">
                                    Checkout</a>
                            {% endif %}
                        </th>
                    </tr>
                </table>
            </div>
```

Рисунок 3.12 – Структура кошика

Контейнер: Використовується клас container для створення контейнера, що містить вміст кошика.

Кнопка "Continue Shopping": Посилання для повернення до магазину (furniture_store).

Таблиця з інформацією про замовлення: Відображає кількість товарів ({{order.get_cart_items}}) та загальну суму ({{order.get_cart_total|floatformat:2}} $). Якщо кошик порожній, відображається повідомлення про це, інакше з'являється кнопка переходу до оформлення замовлення (checkout).

На рисунку 3.13 продемонстровано реалізацію списку товарів у кошику.

```
<div class="box-element">
    <div class="cart-row">
        <div style="flex:2"></div>
        <div style="flex:2"><strong>Item</strong></div>
        <div style="flex:1"><strong>Price</strong></div>
        <div style="flex:1"><strong>Quantity</strong></div>
        <div style="flex:1"><strong>Total</strong></div>
    </div>
    {% for item in items %}
    <div class="cart-row">
        <div style="flex:2"><img class="row-image" src="{{item.product.imageURL}}"></div>
        <div style="flex:2"><p>{{item.product.name}}</p></div>
        {% if item.product.discount_price %}
            <div style="flex:1"><p>{{item.product.discount_price|floatformat:2}}</p></div>
        {% else %}
            <div style="flex:1"><p>{{item.product.price|floatformat:2}}</p></div>
        {% endif %}
        <div style="flex:1">
            <p class="quantity">{{item.quantity}}</p>
            <div class="quantity">
                <img data-product={{item.product.id}} data-action="add" class="chg-quantity update-cart"
                    src="{% static 'main/img/arrow-up.png' %}">

                <img data-product={{item.product.id}} data-action="remove" class="chg-quantity update-cart"
                    src="{% static 'main/img/arrow-down.png' %}">
            </div>
        </div>
        <div style="flex:1"><p>{{item.get_total}}</p></div>
    </div>
    {% endfor %}
</div>
```

Рисунок 3.13 – Список товарів у кошику

Заголовки стовпців: Визначені заголовки для товару, ціни, кількості та загальної вартості.

Перебір товарів у кошику: Використовується цикл {% for item in items %} для відображення кожного товару в кошику.

Зображення товару: Відображення зображення товару ({{item.product.imageURL}}).

Назва товару: Відображення назви товару ({{item.product.name}}).

Ціна товару: Відображення ціни товару, з урахуванням знижки ({{item.product.discount_price|floatformat:2}}), якщо вона є, або звичайної ціни ({{item.product.price|floatformat:2}}).

Кількість товару: Відображення кількості товару в кошику ({{item.quantity}}) та кнопок для зміни кількості (arrow-up.png, arrow-down.png).

Загальна вартість товару: Відображення загальної вартості товару ({{item.get_total}}).

3.3 Робота користувача з вебдодатком

При першому переході на сайт користувач бачить головну сторінку, яка зображена на рисунку 3.14. На цій сторінці зображено слайдер з основною інформацією про товари, також користувач бачить кнопку авторизації **Login,** корзину товарів , кількість товарів в корзині та інше.

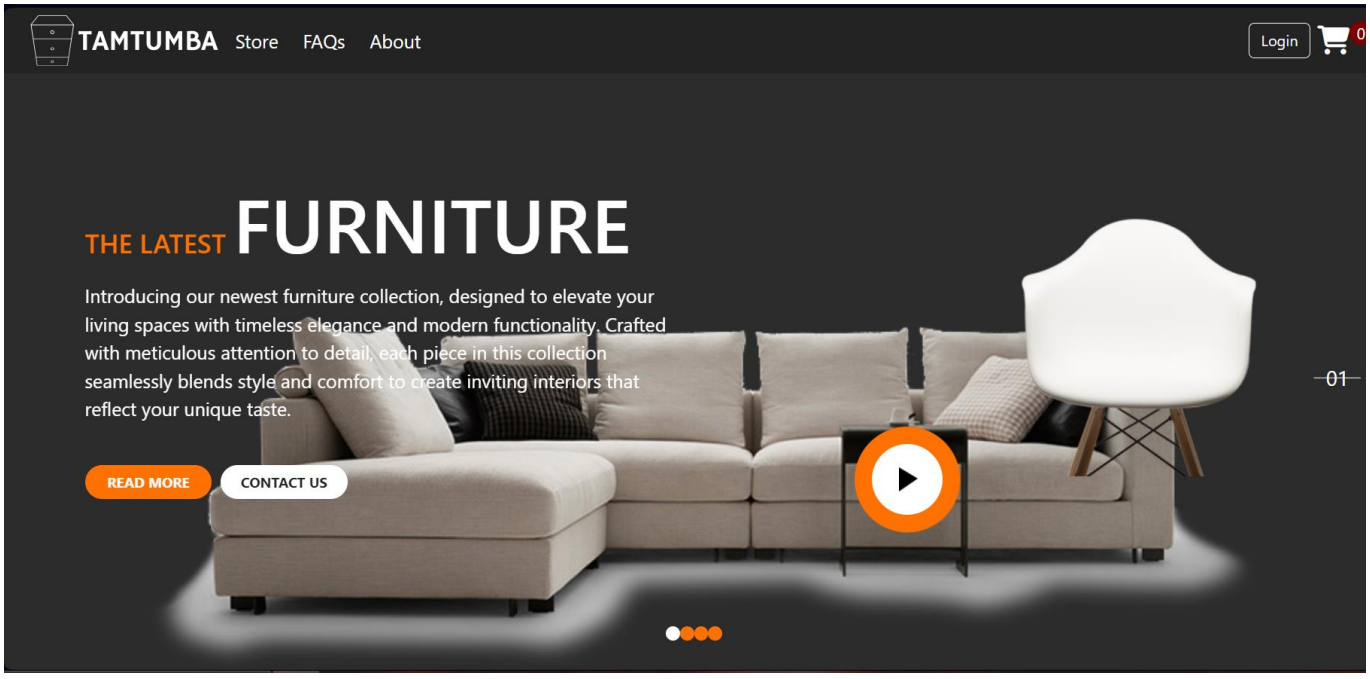


Рисунок 3.14 – Головна сторінка сайту

Натиснувши кнопку **Store** користувача переводить на сторінку з товарами. На рисунку 3.15 зображена сторінка з товарами. На цій сторінці зображено сайдбар в якому реалізовано механізм категорій та діапазон цін. Користувач може відразу додати товар в кошик, або ж перейти до детального огляду товару натиснувши кнопку **View**.

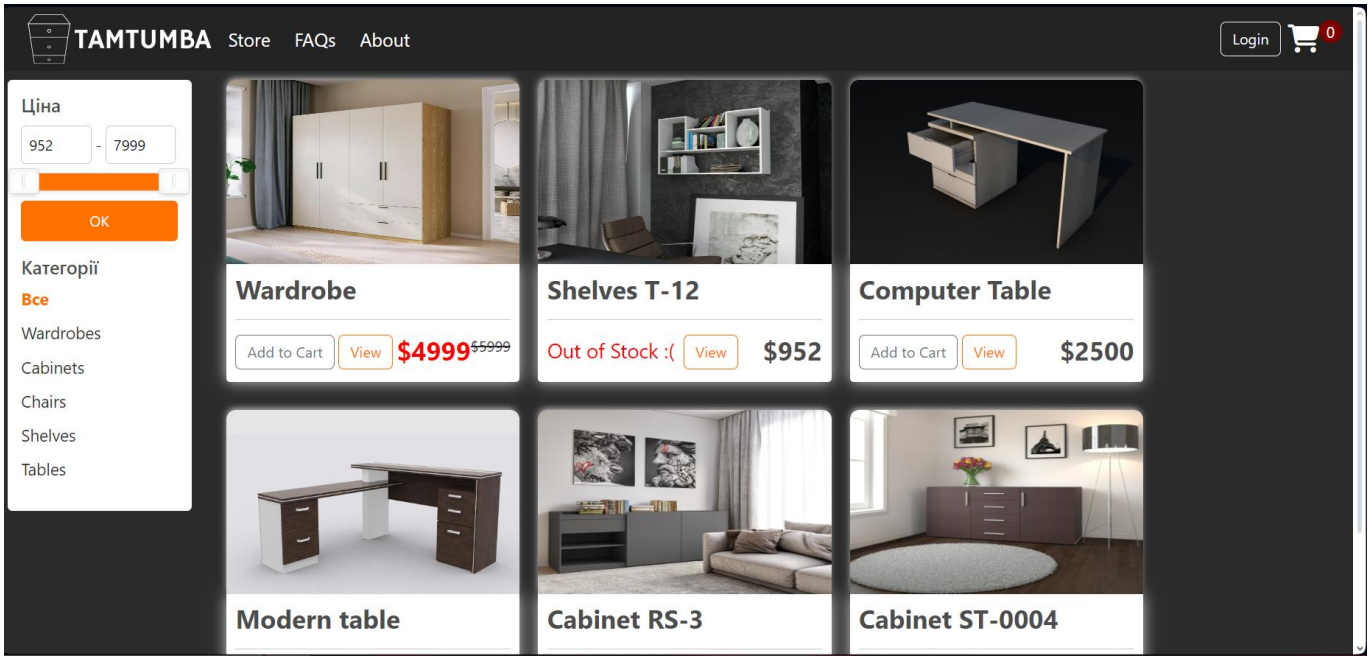


Рисунок 3.15 – Каталог товарів

Перейшовши на сторінку з детальним описом товару користувач бачить опис товару, додаткові фото, та , якщо є можливість, кнопку яка дозволяє переглянути товар у трьох вимірному вигляді. На рисунку 3.16 зображено сторінку з детальним описом товару.

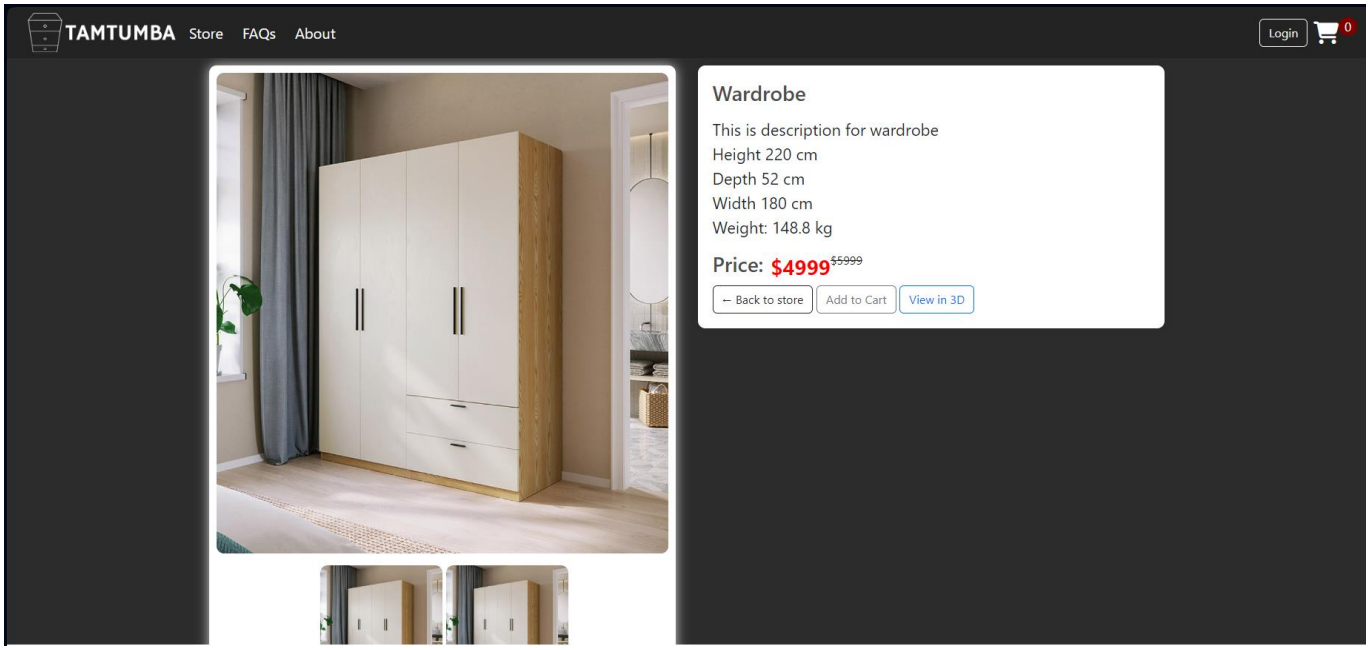


Рисунок 3.16 – Сторінка товару

Якщо користувач натисне на кнопку **View in 3D,** його перенесе на сторінку на якій користувач зможе переглянути товар у трьох вимірному вигляді. Приклад цієї сторінки продемонстровано на рисунку 3.17. З функціональної частини користувач може за допомогою миші крутити товар, за допомогою клавіши Ф(англійської A) користувач увімкне обертання навколо осі товару.

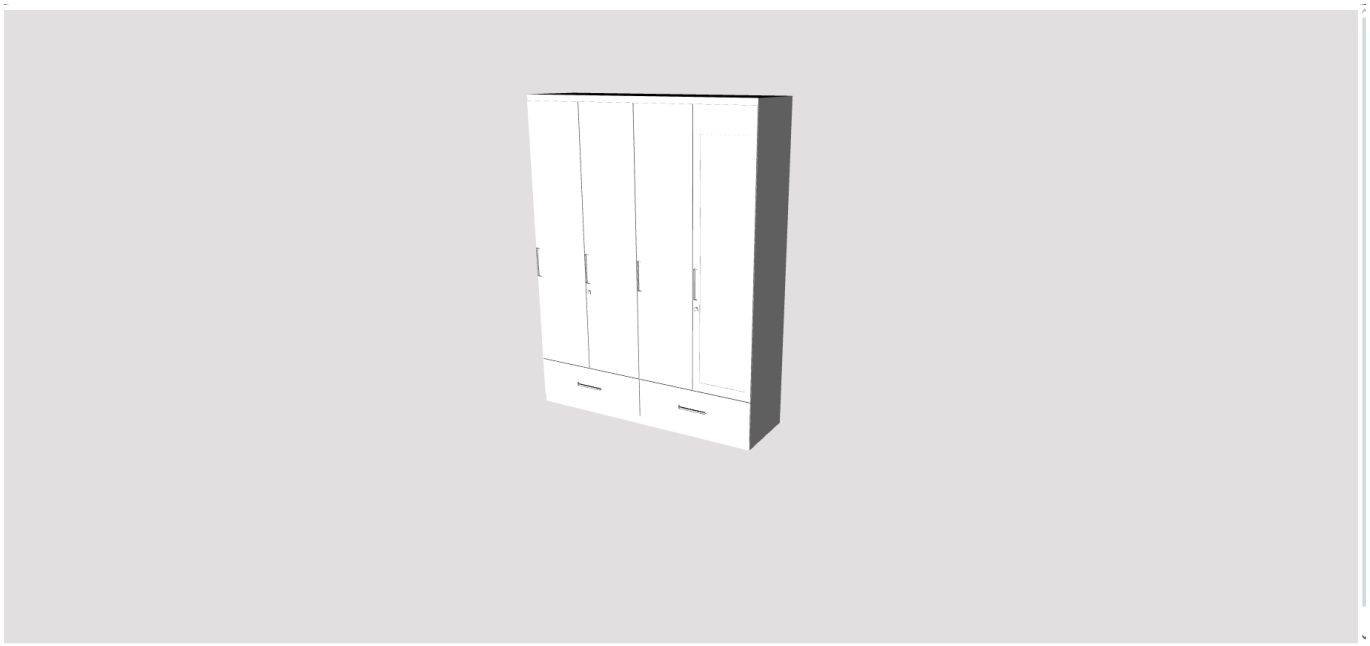


Рисунок 3.17 – 3D вигляд товару

Якщо користувач бажає авторизуватися то йому потрібно натиснути на кнопку **Login,** яка знаходиться у верхній панелі сайту. Перейшовши на сторінку яка зображена на рисунку 3.18 користувач може авторизуватися за допомогою своєї пошти та паролю, якщо ж користувач не зареєстрований то він натискає на кнопку **Sign up**, де йому потрібно буде заповнити необхідні поля, після чого потрібно буде йому авторизуватися.

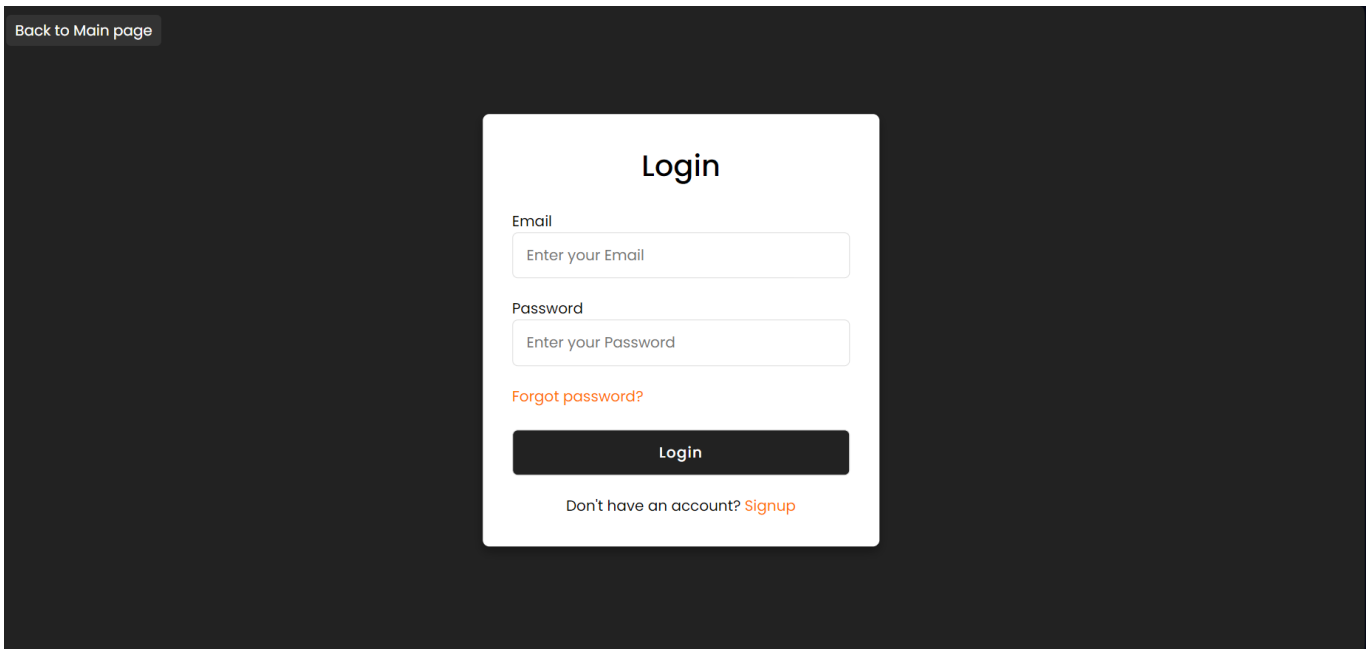


Рисунок 3.18 – Авторизація користувача

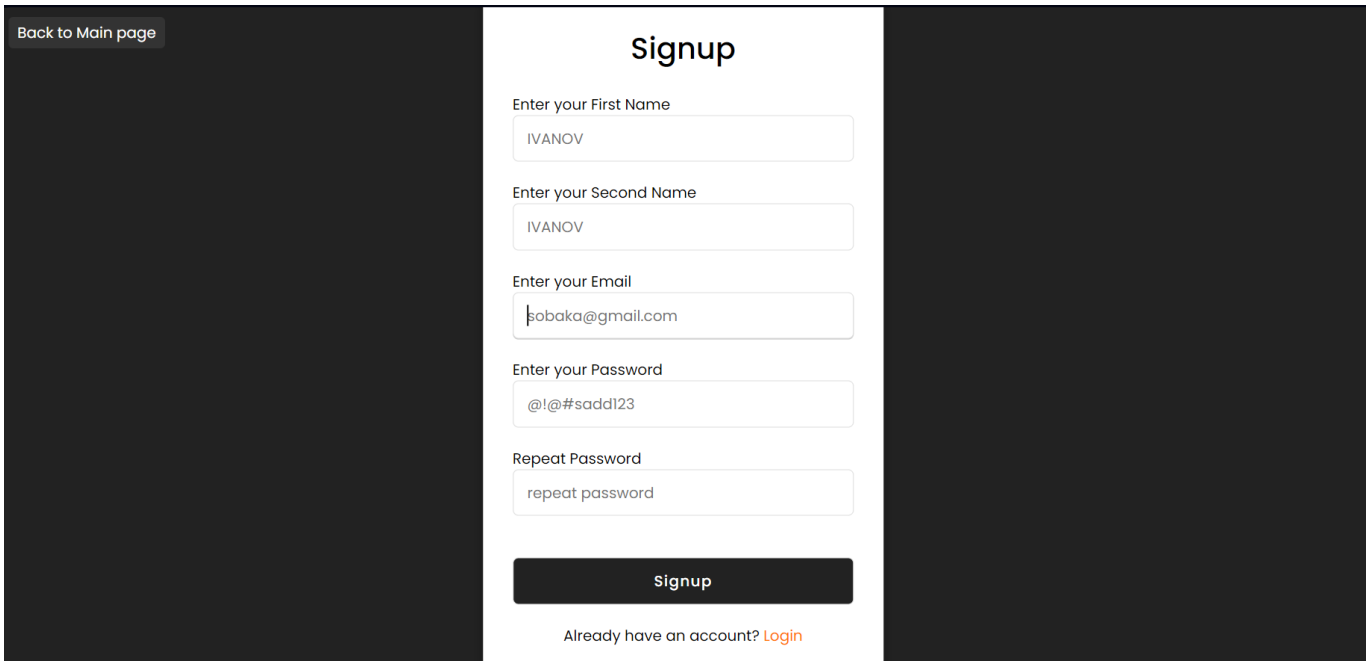


Рисунок 3.19 – Реєстрація користувача

Після авторизації користувачу буде доступно кнопка кабінету. В цьому кабінеті користувачу буде відображено його бонусні бали , які він отримає при замовлені товару у розмірі 5% від ціни. На рисунку 3.20 зображено кабінет товару.

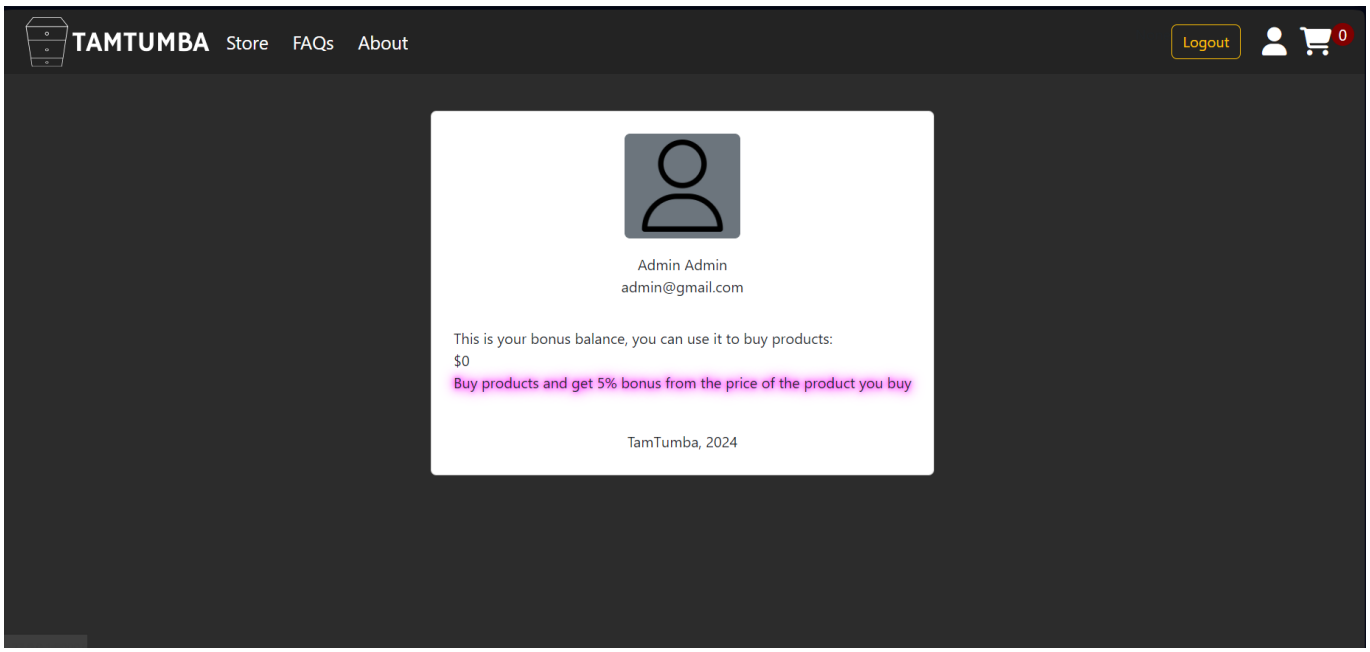


Рисунок 3.20 – Кабінет користувача

Додавши товари в корзину користувач може змінити кількість товарів, або ж оформити замовлення. На рисунку 3.21 продемонстровано корзину товарів.

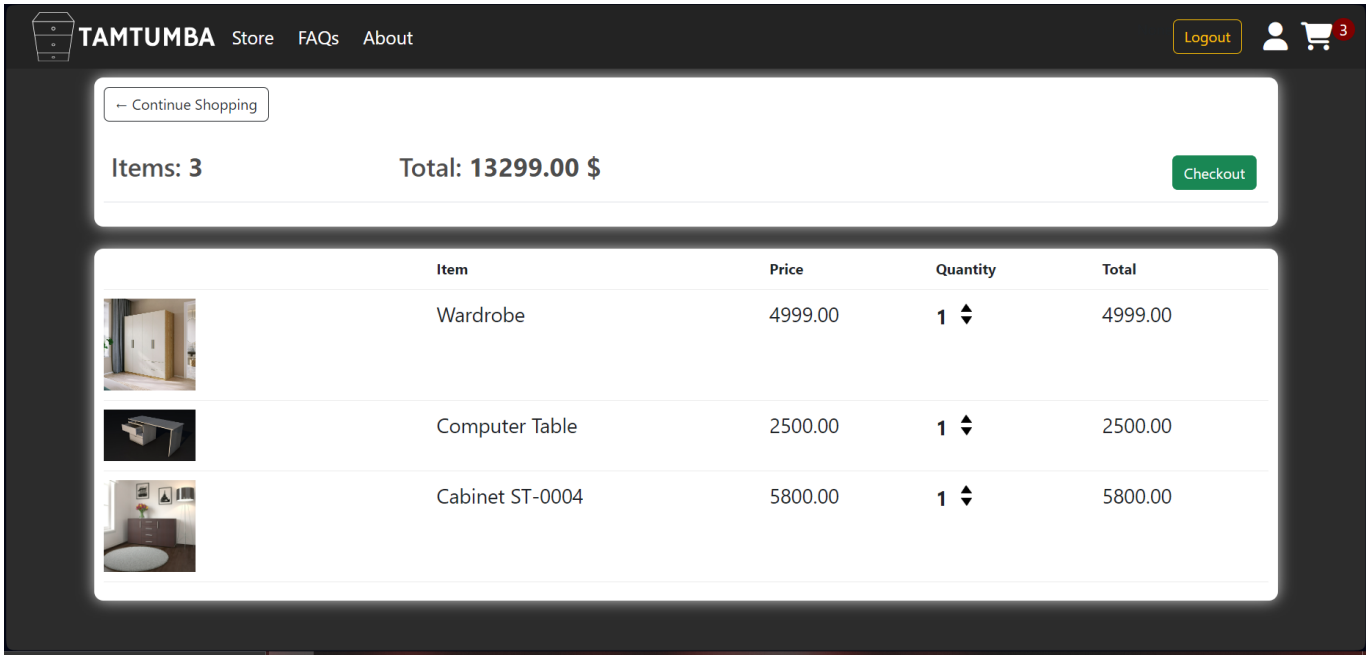


Рисунок 3.21 – Корзина товарів

Перейшовши на сторінку оформлення замовлення користувач, якщо він авторизований, повинен заповнити необхідні поля для доставки. На рисунку 3.22 зображено сторінку оформлення замовлення товарів.

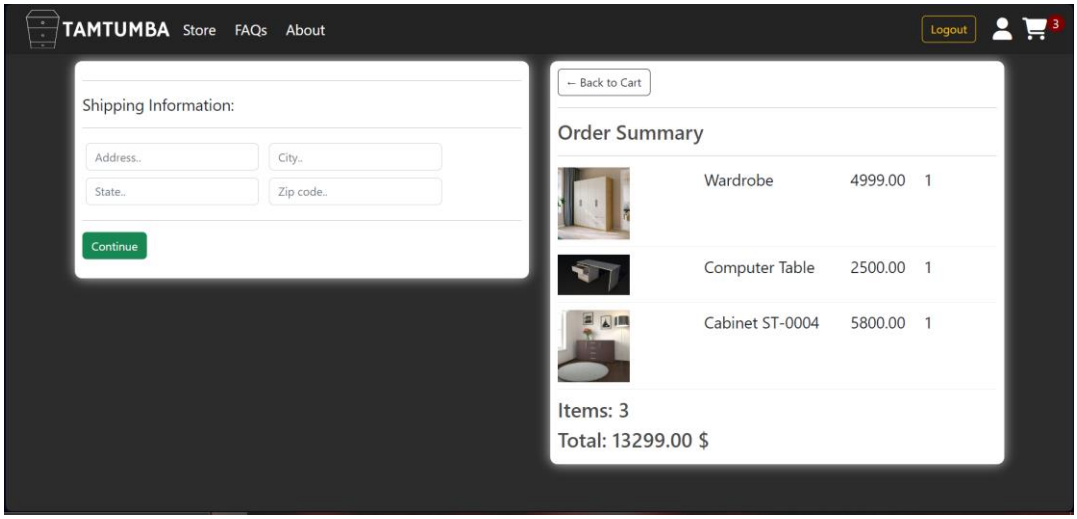


Рисунок 3.22 – Оформлення замовлення

Після оформлення замовлення користувачі приходить на електрону адресу лист з інформацією про замовлення товару. Приклад листа наведено на рисунку 3.23

Dear Nazar,

Thank you for your order!

Here are the details of your order:

Product: Computer Table
Price: 1600.00
Quantity: 2
Subtotal: 3200.00
Product: Modern chair
Price: 999.00
Quantity: 2
Subtotal: 1998.00

Total items: 4

Total amount: 5198.00

Order date: May 14, 2024, 11:37 p.m.

Regards,

The TamTumba Team

Рисунок 3.23 – Оформлення замовлення

3.4 Робота адміністратора з вебдодатком

Перейшов за посилання **/admin** адміністратор сайту повинен ввести свої дані для входу. На рисунку 3.24 зображена авторизація адміну сайту.

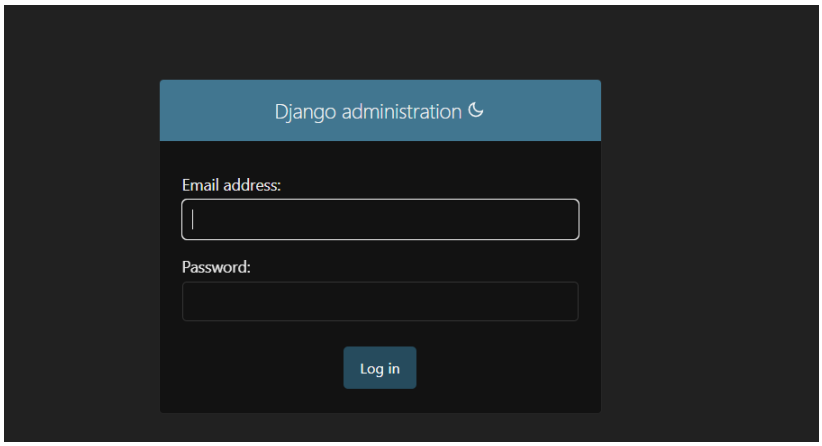


Рисунок 3.24 – Сторінка авторизації адміністратора

На рисунку 3.25 продемонстровано адмін-панель. На цій сторінці адміністратору доступно можливість до зміни, додаванню або видаленню товарів, перегляд який, та що замовив користувач , та адресу доставки.

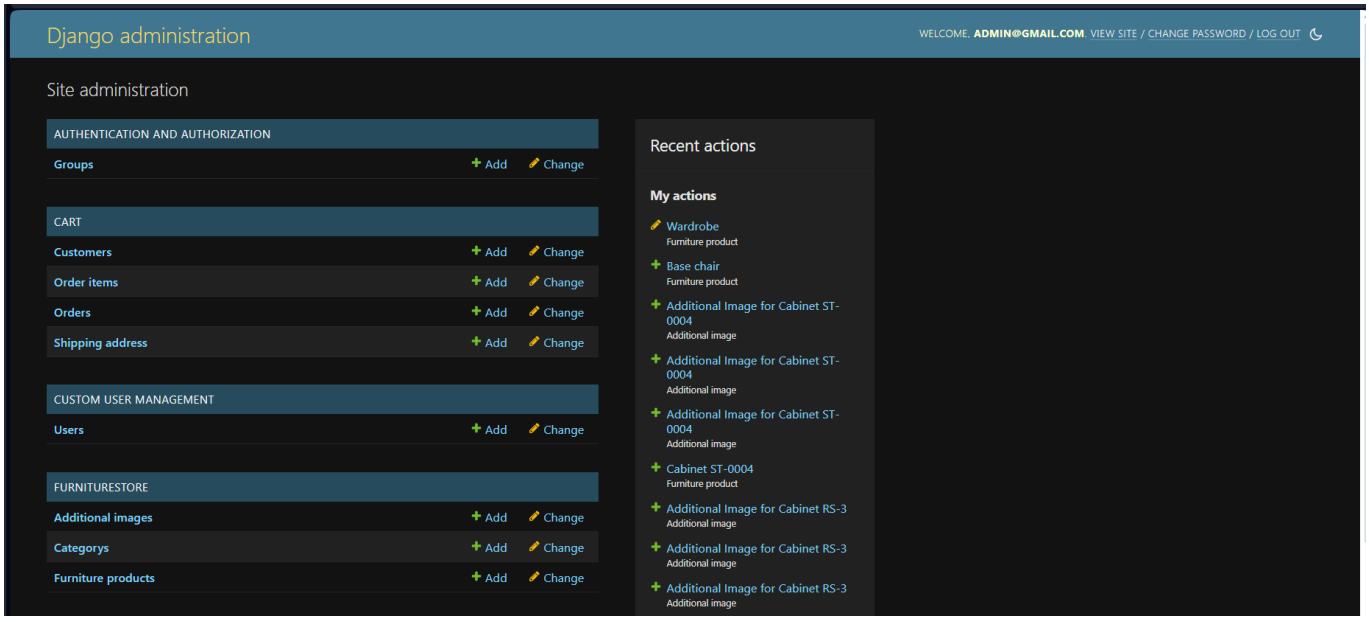


Рисунок 3.25 – Адмін-панель

Щоб додати новий товар адміністратор повинен натиснути знак плюс напроти **Furniture products**. На рисунку 3.26 показано сторінку по додаванню нових товарів. На цій сторінці адміністратор повинен ввести такі поля як: ім’я товару, опис, ціна, фото, 3D модель, якщо є , категорію, чи наявний товар в даний момент та знижку на ціну, якщо є.

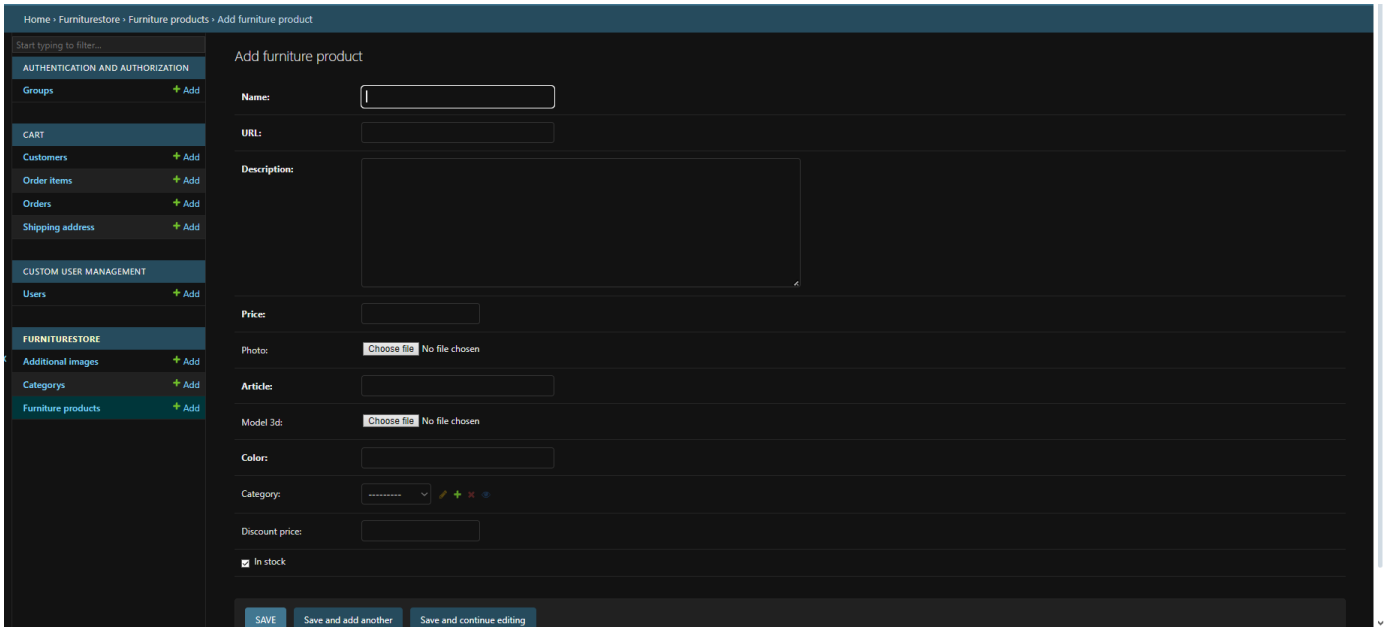


Рисунок 3.26 – Додавання нових товарів

Перейшов на вкладку **Orders** адміністратор може перевірити замовлення, а саме хто замови, статус замовлення та id транзакції. На вкладці **Order Items** зображено що саме було замовлено та яким користувачем. На рисунках 3.27 - 3.28 зображено сторінку з замовленнями та товарами на відправку.

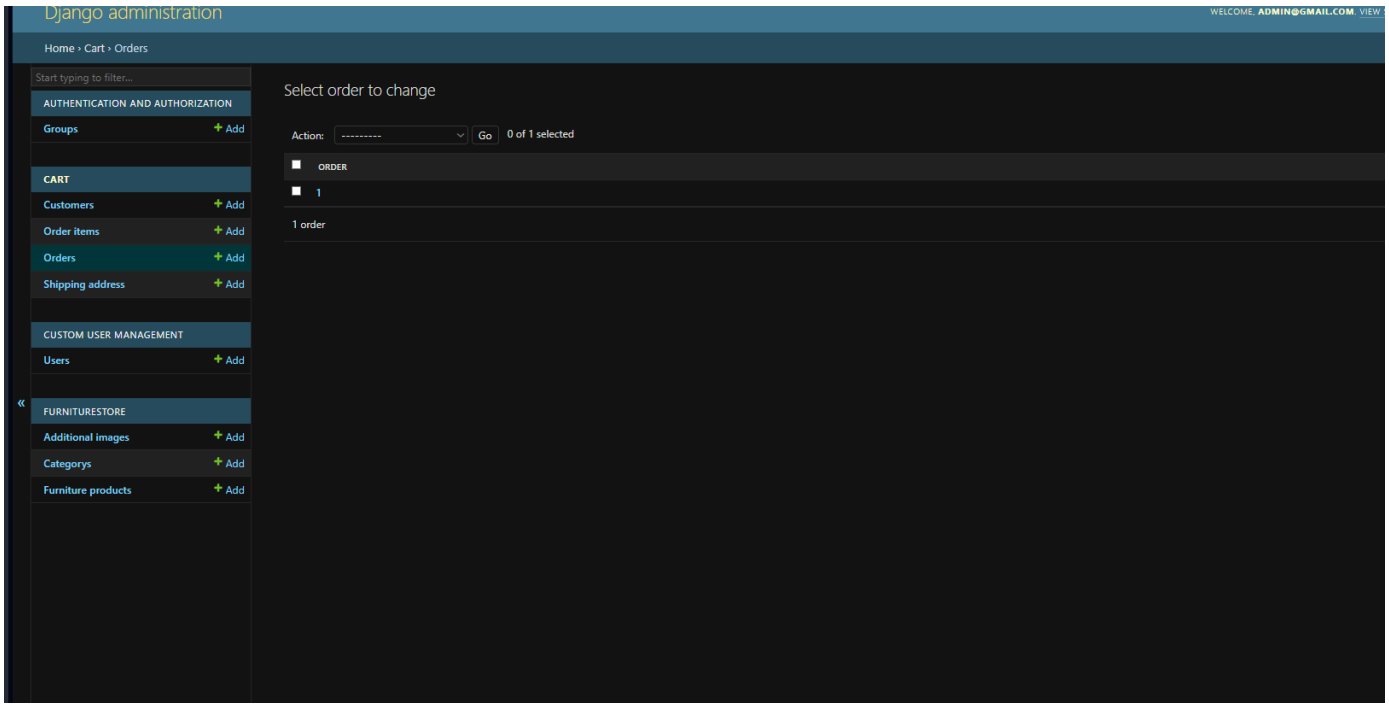


Рисунок 3.27 – Сторінка «Orders»

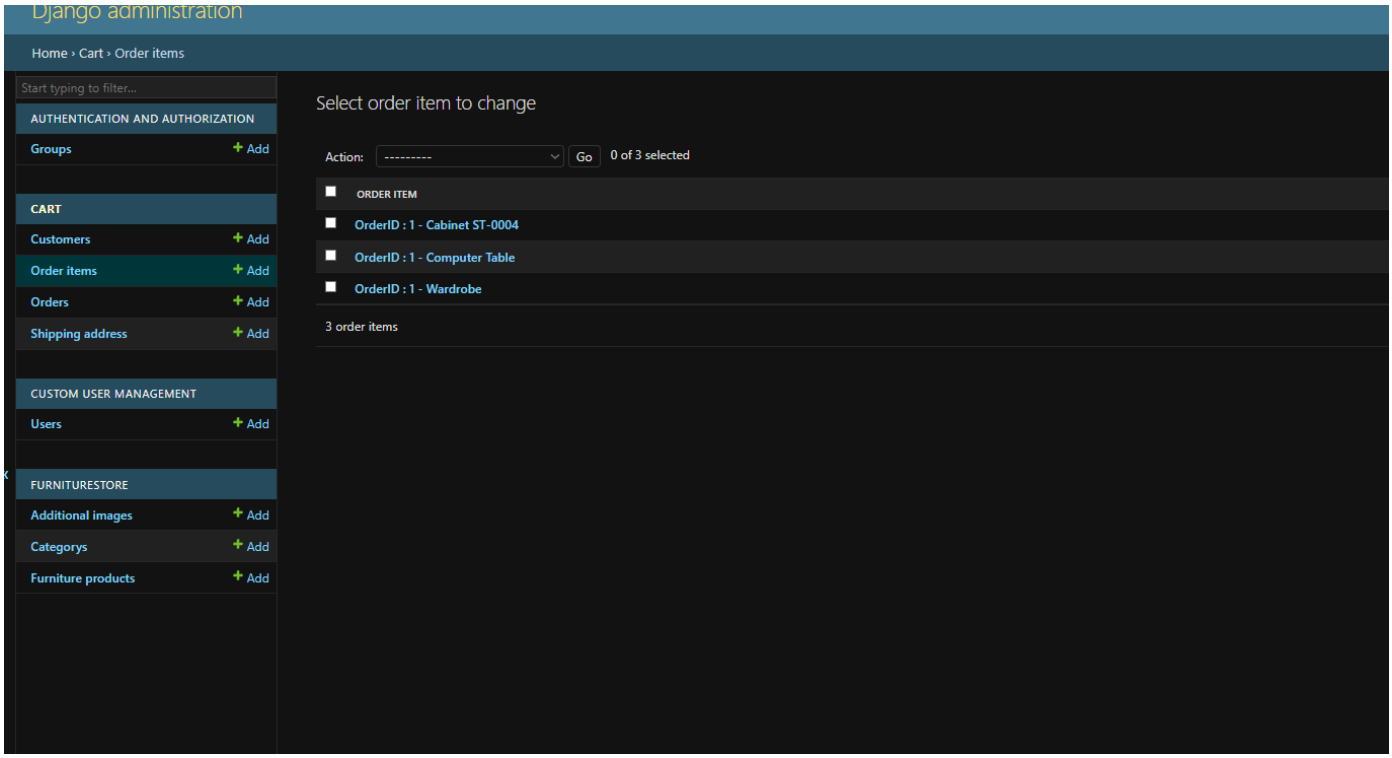


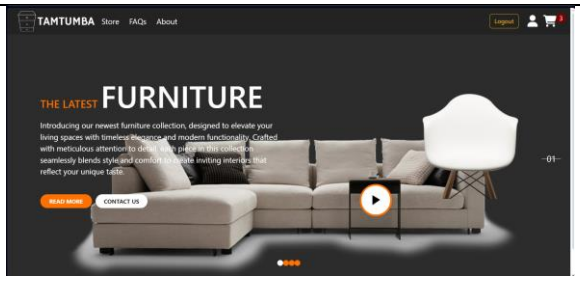
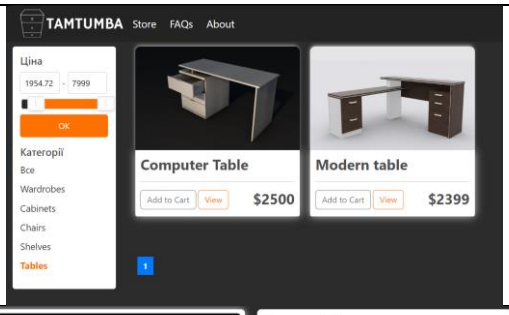
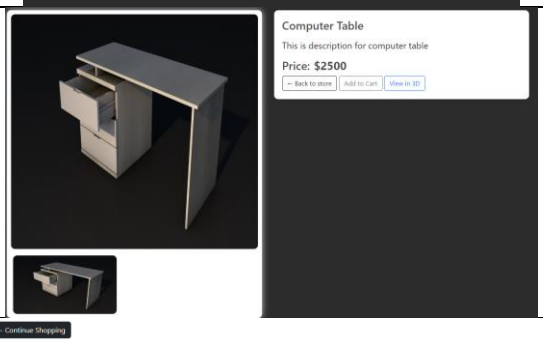
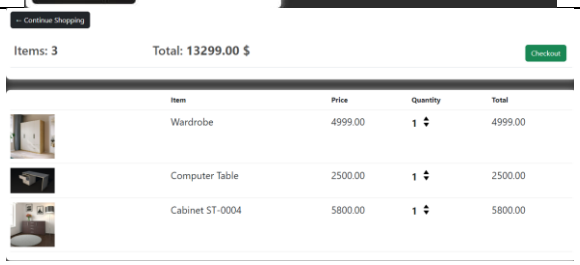
Рисунок 3.28 – Сторінка «Order items»

3.5 Результати тестування

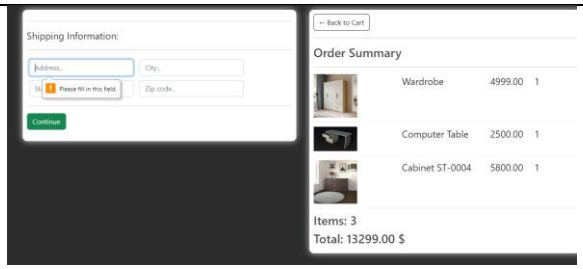
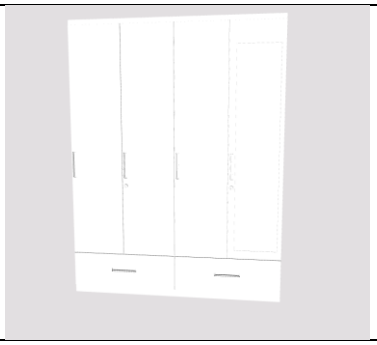
3.5.1 Модульне тестування моделі

Модульне тестування моделі перевіряє, чи створюються і зберігаються екземпляри моделі з правильними значеннями в базі даних. В таблиці 4.1 представлено модульне тестування системи

Таблиця 4.1 – Тестування вебдодатку

Назва	Очікуваний результат	Фактичний результат	0/1
Перевірка головної сторінки	Сторінка завантажується правильно, відображає інформацію про магазин, акції, основні пропозиції, навігаційне меню		1
Перевірка сторінки "Каталог меблів"	Відображає всі товари магазину, працює фільтрація за категоріями (Столи, Стільці, Дивани, Ліжка)		1
Перевірка сторінки "Товар"	Відображає деталі обраного товару: фотографії, опис, ціну, характеристики, можливість перегляду в 3D, додавання в корзину		1
Перевірка сторінки "Корзина товарів"	Показує список доданих товарів, дозволяє видаляти або змінювати кількість, містить кнопку переходу до оформлення замовлення		1

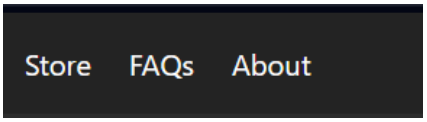
Продовження табл. 4.1

Назва	Очікуваний результат	Фактичний результат	0/1
Перевірка сторінки "Оформлення замовлення"	Дозволяє вказати адресу доставки, обрати спосіб оплати, переглянути вміст замовлення перед підтвердженням. Також перевіряється наповненість полів		1
Перевірка інтерактивності (перегляд товарів в 3D)	Користувач може переглядати товар у трьох вимірному вигляді		1

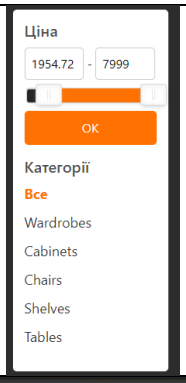
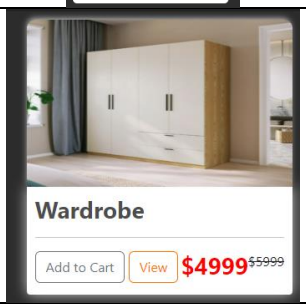
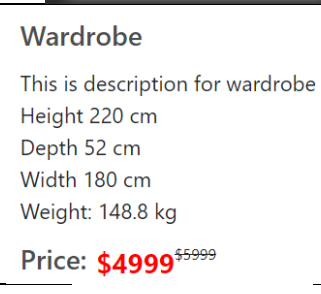
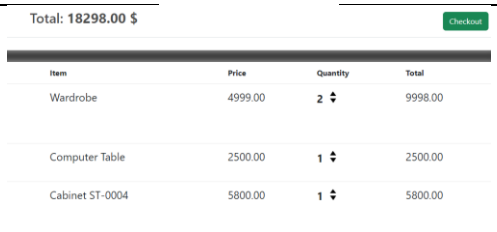
3.5.2 Тестування користувацького інтерфейсу

Тестування користувацького інтерфейсу (UI) включає перевірку зручності використання і правильності відображення елементів на сторінках. Ми перевіримо, чи відображаються основні елементи інтерфейсу правильно і чи працюють основні функції.

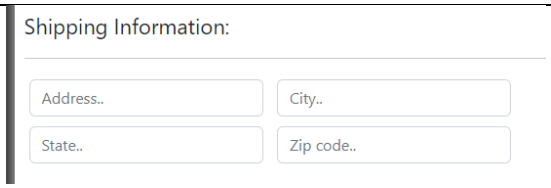
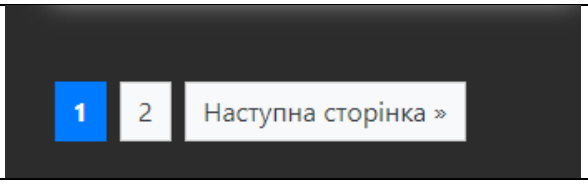
Таблиця 4.2 – Тестування користувацького інтерфейсу

Назва	Очікуваний результат	Фактичний результат	0/1
Перевірка логотипу	Логотип магазину відображається у верхньому лівому куті		1
Перевірка навігації	Меню навігації містить посилання на головні розділи сайту (Каталог, Контакти, Про нас), і ці посилання працюють		1

Продовження табл. 4.2

Назва	Очікуваний результат	Фактичний результат	0/1
Перевірка фільтрів	Фільтри для сортування товарів за категоріями, ціною та іншими параметрами працюють коректно		1
Перевірка відображення товарів	Всі товари відображаються з правильними зображеннями, назвами, цінами та описами		1
Перевірка інформації про товар	На сторінці товару відображається вся необхідна інформація (назва, опис, ціна, зображення, 3D модель, колір тощо)		1
Перевірка кнопки "Додати до кошика"	Кнопка "Додати до кошика" працює і товар додається до кошика		1
Перевірка зміни кількості товарів	Користувач може змінювати кількість товарів у кошику		1
Перевірка загальної суми	Загальна сума кошика оновлюється відповідно до змін кількості товарів		1

Продовження табл. 4.2

Назва	**Очікуваний результат**	**Фактичний результат**	**0/1**
Перевірка форми оформлення замовлення	Форма оформлення замовлення заповнюється і відправляється коректно		1
Перевірка пагінації	Відображається коректно кількість сторінок		1

Проведене тестування гарантує, що вебдодаток працює коректно і забезпечує позитивний користувацький досвід.

ВИСНОВКИ

Розробка вебдодатку для магазину меблів «TamTumba» є актуальним проектом, оскільки споживачі все більше використовують електронну комерцію для покупок. З метою покращення досвіду покупців щодо користування сайтом, важливо створити зручну і ефективну платформу, яка надає можливість легко знаходити потрібні товари, отримувати детальну інформацію та комфортно здійснювати покупки в онлайн-режимі.

Аналіз предметної області та програмних продуктів-аналогів показує, що конкурентоспроможність таких платформ залежить від їх адаптації до сучасних тенденцій, зокрема, від сучасного дизайну, зручної навігації, інтерактивності та функціональності, яка забезпечує зв'язок з адміністратором, зручність оформлення замовлень, інтерактивний перегляд товарів у тривимірному вигляді та зручний фільтр для пошуку товарів. Проведене порівняння аналогічних вебсайтів дозволило сформувати перелік вимог до розробки.

Детальне планування робіт, що включало в себе аналіз вимог до функціональності та дизайну, складання технічного завдання, оцінку необхідних ресурсів та термінів виконання. Крім того, був проведений аналіз ризиків, що дозволив ідентифікувати можливі загрози для успішного завершення проекту та розробити стратегії їх управління. Такий підхід сприяє забезпеченню ефективного виконання проекту та максимізації його успішності.

Проектування вебдодатку для магазину меблів «TamTumba» базується на функціональному моделюванні за методологією IDEF0 та UML. Це дозволило детально відобразити функції системи, їх взаємодію, інформаційні потоки та виконавців. Основний процес підтримки діяльності магазину охоплює збір і обробку даних про клієнтів та товари, дотримання законодавчих вимог щодо захисту персональних даних і формування підтверджених замовлень.

Архітектура вебдодатку включає клієнтську частину, реалізовану на основі HTML, CSS, JavaScript і React.js, серверну частину на Node.js, і базу даних MySQL.

Це забезпечує обробку запитів, управління сесіями, автентифікацію та зберігання даних.

Реалізація вебдодатку включала розробку інтерактивного інтерфейсу користувача, серверної логіки, структури бази даних та інтеграцію між клієнтською і серверною частинами. Користувачі можуть легко знаходити товари, переглядати їх характеристики та оформляти замовлення через зручний інтерфейс. Адміністратори використовують панель керування для управління товарами, замовленнями та користувачами.

Модульне тестування та тестування користувацького інтерфейсу забезпечили високу надійність і функціональність вебдодатку. Це дозволило створити надійну та зручну платформу для клієнтів та персоналу магазину, що підвищує ефективність діяльності магазину меблів «TamTumba».

Отже, розробка вебдодатку для магазину меблів «TamTumba» є перспективним проектом, який може позитивно вплинути на ефективність діяльності магазину.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бобраніцький Н.В., Ващенко С.М. Вебдодаток підтримки онлайн-продажів меблів. // «Інформатика, математика, автоматика»: матеріали та програма науково-технічної конференції, м. Суми, 24 – 26 квітня 2024 р. – Суми: Сумський державний університет, 2024.

2. Що таке вебсайт і як зробити сайт | hostkoss blog. hostkoss blog. URL: https://hostkoss.com/b/uk/what-is-website/ (дата звернення: 29.04.2024).

3. Інтернет, вебсайт та вебсторінки – урок. Інформатика НУШ, 3 клас. МійКлас. URL: і (дата звернення: 30.04.2024).

4. solomon4ik (2020) Що подобається відвідувачам на сайті і як їх залучити, PBB design Львів. Available at: https://pbb.lviv.ua/statti-i-novyny/statti-shchodo-stvorennia-saitu/shcho-pryvablue-klienta-na-sajti/#:~:text=На%20сайті%20клієнт%20повинен%20легко,ключові%20слова%20мають%20бути%20виділені. (Accessed: 30 April 2024).

5. Статистика Електронної Комерції у 2022 - Nexus. Nexus. URL: https://nxdigitalagency.com/blog-ua/statistika-elektronnoi-komercii-u-2022/ (дата звернення: 29.04.2024).

6. Меблі від виробника купити дешево в mebli.biz.ua. Інтернет-магазин mebli.biz.ua ™. URL: https://mebli.biz.ua (дата звернення: 29.04.2024).

7. Головна. Інтернет-магазин меблів Артос. URL: https://artos.in.ua (дата звернення: 23.04.2024).

8. Меблі в інтернет магазині Divan Star - купити меблі у Києві. URL: https://divanstar.com.ua (дата звернення: 23.04.2024).

9. Django documentation | Django documentation. Django Project. URL: https://docs.djangoproject.com/en/5.0/ (date of access: 23.04.2024).

10. Python 3.12.3 Documentation. URL: https://docs.python.org/3/ (date of access: 23.04.2024).

11. HTML і CSS довідник українською. *Html CSS довідник*. URL: https://html-css.co.ua (дата звернення: 23.04.2024).

12. JavaScript | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (date of access: 23.04.2024).

13. Solomono - How to create an online furniture store? Sale of furniture online | Solomono. osCommerce Create Your Own Online Store | Solomono. URL: https://solomono.net/how-to-create-an-online-furniture-store-in-2020-a-452.html (date of access: 29.04.2024).

14. Стратегії електронної комерції для онлайн-магазину меблів: рекомендації та підходи. SEO Evolution. (n.d.). https://seo-evolution.com.ua/blog/poleznye-sovety/strategiyi-elektronnoyi-komerciyi-dlya-onlajn-magazinu-mebliv

15. The Complete Guide To Understand IDEF Diagram | EdrawMax Online. Edrawsoft. URL: https://www.edrawmax.com/article/the-complete-guide-to-understand-idef-diagram.html (date of access: 16.05.2024).

16. What is Unified Modeling Language. *Lucidchart*. URL: https://www.lucidchart.com/pages/what-is-UML-unified-modeling-language (date of access: 16.05.2024).

17. Maxym Z. Моделювання даних (Data Modelling). Maxym Zosym. URL: https://www.maxzosim.com/data-modelling/ (дата звернення: 16.05.2024).

18. A Comprehensive Overview of How Python Django Works. Tutor Joe's Stanley. URL: https://www.tutorjoes.in/python_django_tutorial/how_python_django_works (date of access: 18.05.2024).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку

«Вебдодаток організації діяльності

магазину з продажу меблів для дому «TamTumba»

ПОГОДЖЕНО:

Доцент кафедри інформаційних технологій
_____ Ващенко С.М.

Студент групи ІТ-02
Бобраніцький Н.В.

Суми 2024

1. Призначення й мета вебдодатку організації діяльності магазину з продажу меблів для дому «TamTumba»

1.1 Призначення вебдодатку

Вебдодаток призначений для організації діяльності магазину з продажу меблів для дому «TamTubma».

1.2 Мета створення вебдодатку

Головна мета проекту – це збільшення прибутковості власників меблевих магазинів завдяки створенню зручного інтернет-магазину, який надає клієнтам змогу замовляти меблі та фурнітуру онлайн та переглядати їх у 3D форматі.

1.3 Цільова аудиторія

Цільовою аудиторією даного проекту є клієнти які бажають придбати нові меблі для дому

2 Вимоги до проекту

2.1 Вимоги до проекту в цілому

2.1.1 Вимоги до структури й функціонування

Вебдодаток організації діяльності магазину з продажу меблів для дому «TamTubma» повинен бути реалізований за допомогою вебінструментів та забезпечувати визначений набір функціональних можливостей.

Кінцевий продукт даного проекту має бути представлений вебдодатком, який містить якісне інформаційне наповнення та графічні матеріали.

2.1.2 Вимоги до персоналу

Персонал магазину не повинен мати особливих технічних навичок для роботи з вебдодатком і його підтримкою. Єдиною вимогою є наявність навичок користування персональним комп’ютером та веббраузером.

2.1.3 Вимоги до збереження інформації

Уся інформація надана у вебдодатку повинна зберігатися у базі даних реалізованій засобами системи управління базами даних SQLite.

2.1.4 Вимоги до розмежування доступу

Розроблюваний вебдодаток має бути загальнодоступним у мережі Інтернет. Права доступу до інформації розмежовані за групами користувачів: адміністратор, відвідувач та клієнт. Адміністратор має необмежений доступ до даних з правами перегляду, додавання, редагування та видалення. Доступ до адміністративної панелі надається за спеціальним логіном та паролем.

Відвідувач вебдодатку та клієнт мають спільний доступ до перегляду товарів та замовлення їх. Клієнт може авторизуватися якщо він уже створив акаунт. Реєстрація на сайті передбачає накопичувальну знижку на майбутні покупки товарів, тобто якщо відвідувач не реєструється на сайті то він не зможе накопичувати бонуси які при наступних замовленням знизять вартість замовлення.

2.2 Структура вебдодатку

2.2.1 Загальна інформація про структуру вебдодатку

До структури вебдодатку входять усі його вебсторінки, які є загальнодоступними, та адміністративна панель для користування персоналом меблевого магазину.

Перелік сторінок вебдодатку наступний:

— Головна сторінка містить інформацію про магазин, акції, та основні пропозиції. Включає навігаційне меню та можливість переходу до категорій товарів;

— Сторінка «Каталог меблів» розміщує усі товари магазину з можливістю фільтрації за категоріями (наприклад, «Столи», «Стільці», «Дивани», «Ліжка»).При кліку на товар переносить на сторінку конкретного товару;

— Сторінка «Товар» (для конкретного товару) відображає деталі про обраний товар: фотографії, опис, ціну та характеристики. Має можливість переглянути товар у 3D (якщо така функція доступна для товару).Дозволяє додати товар у корзину;

— Сторінка «Корзина товарів» показує список товарів, які користувач додав до кошика. Дозволяє видалити або змінити кількість товарів у кошику. Містить кнопку переходу до сторінки оформлення замовлення;

— Сторінка «Оформлення замовлення» дозволяє користувачам вказати адресу доставки, обрати спосіб оплати та переглянути вміст свого замовлення перед підтвердженням.

2.2.2 Навігаційне меню

Для зручної навігації повинно бути створене меню, що забезпечить швидке переміщення користувача по всім доступним сторінкам вебдодатку. Меню має бути закріплене і розташовуватися зверху (у шапці) на кожній сторінці.

2.2.3 Управління контентом

Управління контентом вебдодатку має здійснюватися за допомогою адміністративної панелі. Усе інформаційне наповнення вебдодатку має міститися у базі даних. Графічні матеріали та інформацію для наповнення надає Замовник.

2.2.4 Дизайн вебдодатку

Дизайн вебдодатку має бути виконаний у мінімалістичному та сучасному стилі. Корпоративними кольорами магазину закладу є чорний, червоний та палітра відтінків сірого кольору. Тому під час розробки вебдодатку треба використовувати саме ці кольори.

Види і розміри шрифтів повинні бути комфортними для перегляду. Інформаційні блоки, графічні матеріали та інші елементи вебсторінок повинні мати зручне і логічне розташування. Шаблон майбутнього програмного продукту зображено на рисунку А.1

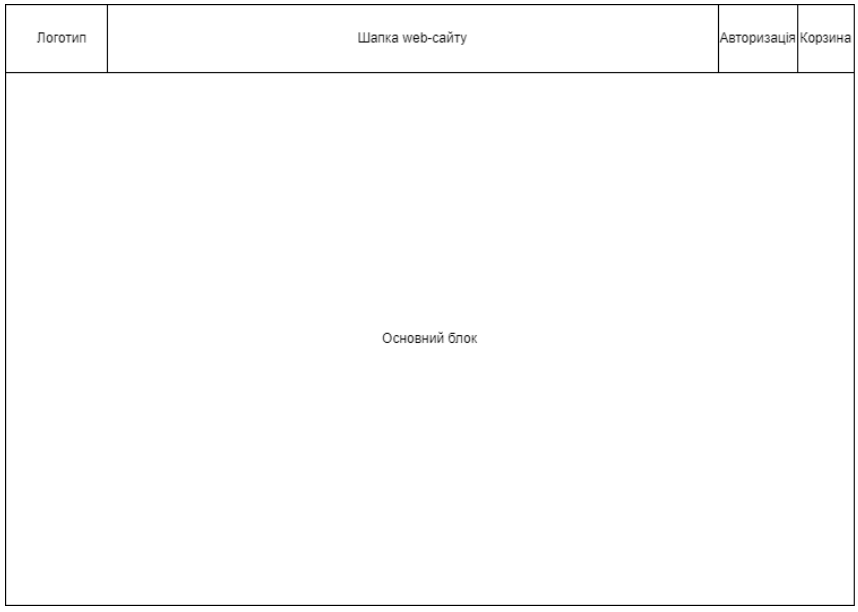


Рисунок А.1 – Схема головної сторінки

2.2.4 Система навігації (карта вебдодатку)

Карта вебдодатку зображена на рисунку А.2.

Рисунок А.2 – Система навігації

2.3 Вимоги до видів забезпечення

2.3.1 Вимоги до лінгвістичного забезпечення

Весь текст у вебдодатку має бути виконаний українською або англійською мовами.

2.3.2 Вимоги до програмного забезпечення

Для забезпечення стабільної роботи вебдодатку веббраузер має бути Internet Explorer 7.0 і вище, або Firefox 3.5 і вище, або Opera 9.5 і вище, або Safari 3.2.1 і вище, або Chrome 2 і вище.

2.4 Вимоги до функціонування системи

2.4.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	Потреби користувача	Джерело
UN-01	Перегляд продукції магазину	Клієнт,

		Відвідувач
UN-02	Замовлення продукції онлайн	Клієнт, Відвідувач
UN-03	Перегляд історії своїх замовлень	Клієнт
UN-04	Перегляд опису товарів	Клієнт, Відвідувач
UN-05	Перегляд акцій та оголошень	Клієнт, Відвідувач
UN-06	Перегляд інформації про інтернет магазин	Клієнт, Відвідувач
UN-07	Бронювання столу в закладі онлайн	Клієнт, Відвідувач
UN-08	Можливість перегляду товару в 3D	Клієнт, Відвідувач
UN-09	Редагування даних	Адміністратор

2.4.2 Функціональні вимоги

Проаналізувавши потреби користувачів та персоналу магазину було визначено наступні вимоги:

Клієнти магазину:

- наявність реєстрації та авторизації клієнтів;
- можливість інтерактивного перегляду товару в 3D
- пошук інформації про товар на вебдодатку;
- можливість оформлення онлайн-замовлень за допомогою форми;

Персонал магазину (адміністратори):

- наявність повної інформації про товар магазину;
- наявність панелі адміністратора для додавання, редагування та видаленні інформації з вебдодатку;

2.4.2 Системні вимоги

Для забезпечення стабільної роботи вебдодатку веббраузер має бути Internet Explorer 7.0 і вище, або Firefox 3.5 і вище, або Opera 9.5 і вище, або Safari 3.2.1 і вище, або Chrome 2 і вище.

3 Склад і зміст робіт зі створення вебдодатку з продажів меблів «TamTumba»

Детальний опис етапів створення вебдодатку наведено в таблиці А.2.

Таблиця А.2 – Етапи створення вебдодатку

№	Склад і зміст робіт	Строк розробки
1	Розробка шаблону вебдодатку	2 дні
2	Задання верстки сторінок вебдодатку	14 днів
3	Розробка функціоналу реєстрації та авторизації користувачів.	7 днів
4	Розробка каталогу меблів та предметів інтер'єру.	7 днів
5	Розробка функціоналу додавання товарів у кошик та оформлення замовлення.	7 днів
6	Розробка бази даних	8 днів
7	Розробка функціоналу обробки та відстеження замовлень.	5 днів
8	Розробка адміністративної панелі для управління контентом.	10 днів
9	Наповнення контентом вебдодатку	14 днів
10	Alpha-тестування	5 днів
11	Beta-тестування	8 днів
12	Розміщення на хостингу	1 день
13	Перевірка працездатності	2 дні
12	Написання супровідної документації	2 дні
13	Реліз вебдодатку	1 день
	Загальна тривалість робіт	93 днів

ДОДАТОК Б

Планування робіт

У сучасному світі зростає значення електронної комерції, яка відіграє важливу роль у зручності та доступності покупок. В рамках дипломного проекту розглядається розробка вебдодаток з продажу меблів для дому, спрямованого на забезпечення зручного та ефективного вибору та придбання меблів для різних приміщень.

Цінність розробки виражається у покращенні досвіду покупців, їх здатності з легкістю знаходити необхідні товари, отримувати детальну інформацію та зручно здійснювати покупки в онлайн режимі. Застосування вебдодатку для продажу меблів для дому має велику актуальність в умовах розвитку технологій та змін споживацьких практик.

Деталізація мети проєкту методом SMART. Для успішності та конкурентоспроможності проекту треба на концептуальному етапі правильно визначити його мету за допомогою SMART-методу. Вона має ширше формулювання. А саме: «Розробка вебдодатку організації діяльності магазину з продажу меблів для дому «TamTumba» на основі затвердженого технічного завдання для збільшення прибутковості власників меблевих магазинів завдяки створенню зручного вебдодатку, який надає клієнтам змогу замовляти меблі та фурнітуру онлайн та переглядати їх у 3D форматі до 1 червня 2024 року».

Результати деталізації мети методом SMART розміщені у таблиці Б.1.

Таблиця Б.1 – Деталізація мети проєкту методом SMART

Specific(конкретна)	Розробка вебдодатку організації діяльності магазину з продажу меблів для дому «TamTumba» з можливістю перегляду товарів у трьох вимірному форматі

Продовження табл. Б.1

Measurable (вимірювана)	Розроблений вебдодаток для підтримки діяльності магазину з продажу меблів для дому
Achievable (досяжна, узгоджена)	Мета досяжна, є затверджене технічне завдання від замовника та клієнтська база магазину «TamTumba»
Relevant (реалістична)	Для збільшення прибутковості за рахунок підвищення рейтингу та конкурентоспроможності магазину «TamTubma»
Time-framed (обмежена в часі)	Є конкретний термін – до 1 червня 2024 р.

Планування змісту робіт. WBS (Work Breakdown Structure – ієрархічна структура робіт) – це графічний вигляд елементів проекту, які згруповані ієрархією у єдине ціле з продуктом проекту. Структура декомпозиції робіт орієнтована на досконале виконання робіт по частинам і сама є ключовою частиною проекту, яка спрямована на організацію командної роботи. Елементами декомпозиції можуть бути продукти, дані та послуги. Більше того, WBS забезпечує необхідним каркасом для ретельної оцінки термінів та контролю та графіків роботи.

На найвищому (першому) рівні розміщений продукт проекту. Основні дії та заходи, що забезпечують досягнення мети проекту, зафіксовані на другому рівні декомпозиції. Декомпозиція робіт виконується до тих пір, поки вони не стануть елементарними (простими).

Елементарні роботи – це дії, які мають однозначний чіткий результат, на які призначена відповідальному одна конкретна особа, для якої можна обчислити витрати праці і тривалість виконання. На рисунку Б.1 представлено WBS з розробки вебдодатку організації діяльності магазину з продажу меблів для дому «TamTumba».

Планування структури виконавців. Наступним етапом після декомпозиції процесів є розробка організаційної структури виконавців або OBS, яка визначається як графічна структура відображення учасників або відповідальних осіб, які беруть участь у реалізації проєкту. У ролі відповідальних осіб виступають співробітники, що відповідають за організацію і виконання елементарної роботи, що зазначена у WBS. Кожну елементарну роботи можна розглядати як окремий проект. На рисунку Б.2

представлено організаційну структуру планування проєкту. Список виконавців, що функціонують в проекті описано в таблиці Б.2.

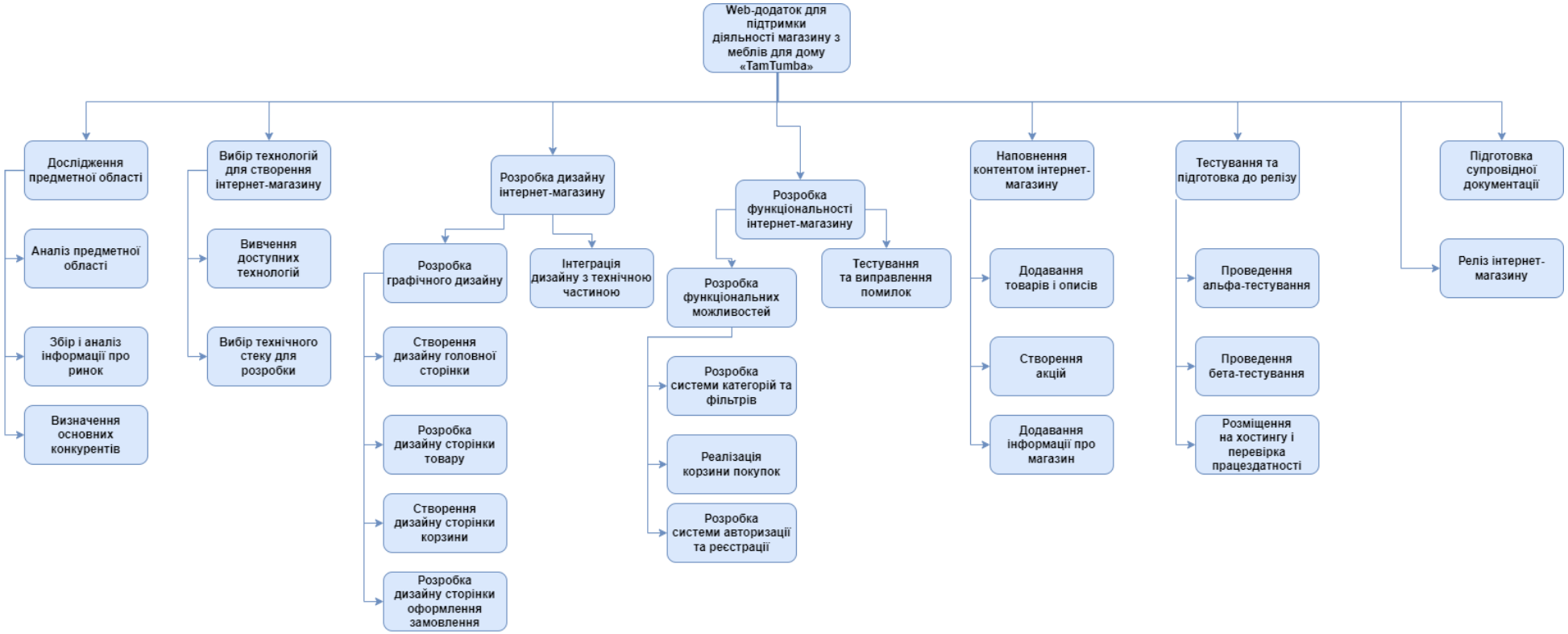


Рисунок Б.1 – WBS-структура робіт проекту

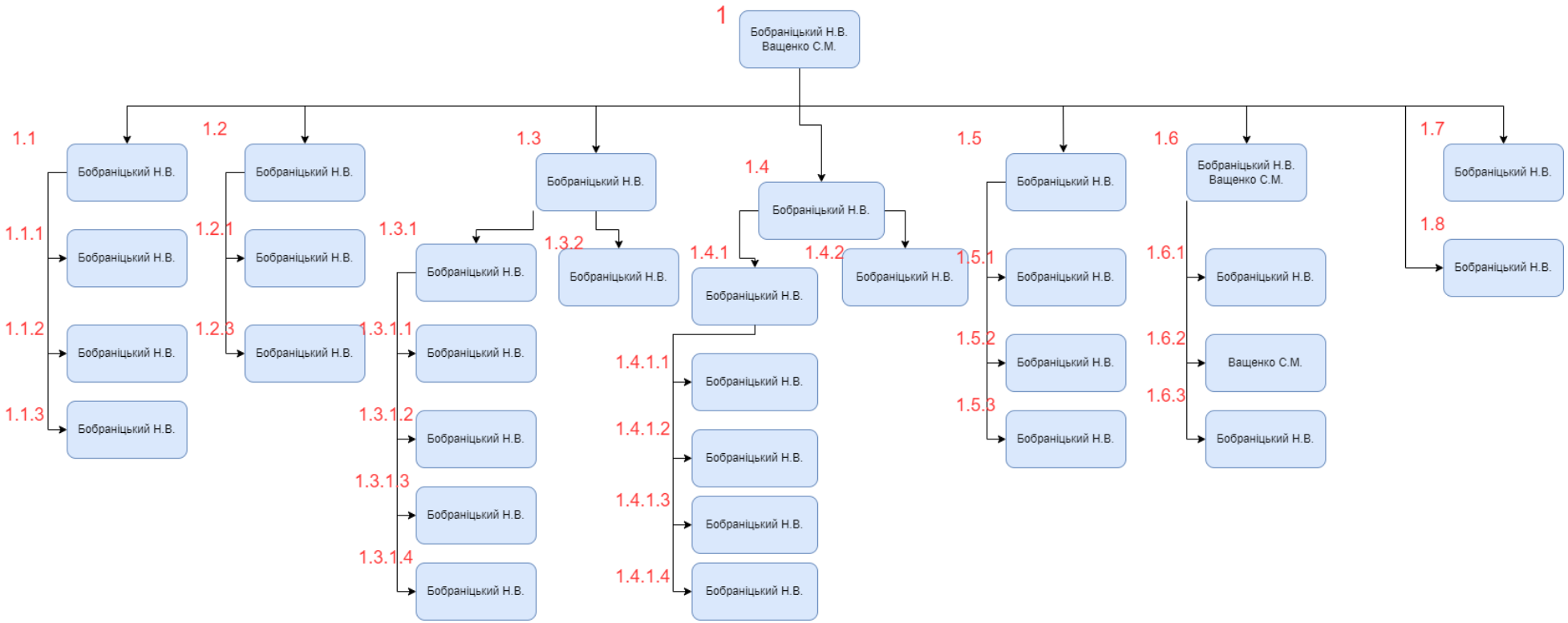


Рисунок Б.2 – OBS-структура робіт проекту

Таблиця Б.2 – Виконавці проєкту

Роль	ПІБ	Проектна роль
Розробник	Бобраніцький Н.В.	Виконує front-end та back-end розробку.
Проєктувальник	Бобраніцький Н.В.	Виконує проєктування бази даних та розробляє структуру вебдодатку.
Керівник проекту	Ващенко С.М.	Формує завдання на розробку проєкту.
Менеджер проєкту	Бобраніцький Н.В.	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками. Виконує збір та аналіз даних.
Тестувальник	Бобраніцький Н.В.	Відповідає за тестування функціоналу та дизайну вебдодатку

Діаграма Ганта. Побудова календарного графіку (діаграми Ганта) є одним з важливих етапів планування проєкту, що виглядає як розклад виконання робіт з реальним розподілом дат. Завдяки йому можна отримати достовірне уявлення про тривалість процесів з обмеженнями у ресурсах, урахуванням вихідних днів та свят. Календарний графік проєкту представлено на рисунках Б.3-Б.4.

WBS	Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Web-додаток організації діяльності магазину з продажу меблів для дому «TamTubma»**	**192 days**	**Tue 17.10.23**	**Wed 10.07.24**		**Developer;QA**
1.1	**Дослідження предм**	**20 days**	**Tue 17.10.23**	**Mon 13.11.23**		
1.1.1	Аналіз предметної с	10 days	Tue 17.10.23	Mon 30.10.23		Developer
1.1.2	Збір і аналіз інформ	5 days	Tue 31.10.23	Mon 06.11.23	3	Developer
1.1.3	Визначення основни	5 days	Tue 07.11.23	Mon 13.11.23	4	Developer
1.2	**Вибір технологій для створення інтернет-магазину**	**30 days**	**Tue 14.11.23**	**Mon 25.12.23**	**2**	
1.2.1	Вивчення доступни	15 days	Tue 14.11.23	Mon 04.12.23	5	Developer
1.2.2	Вибір технічного сте	15 days	Tue 05.12.23	Mon 25.12.23	7	Developer
1.3	**Розробка дизайну інтернет-магазину**	**37 days**	**Tue 26.12.23**	**Wed 14.02.24**	**6**	
1.3.1	**Розробка графічного**	**27 days**	**Tue 26.12.23**	**Wed 31.01.24**	**8**	
1.3.1.1	Створення дизайн	5 days	Tue 26.12.23	Mon 01.01.24	8	Developer
1.3.1.2	Розробка дизайну	5 days	Thu 11.01.24	Wed 17.01.24	11	Developer
1.3.1.3	Створення дизайн	5 days	Thu 18.01.24	Wed 24.01.24	12	Developer
1.3.1.4	Розробка дизайну	5 days	Thu 25.01.24	Wed 31.01.24	13	Developer
1.3.2	Інтеграція дизайну з	10 days	Thu 01.02.24	Wed 14.02.24	10	Developer
1.4	**Розробка функціональності інтернет-магазину**	**60 days**	**Thu 15.02.24**	**Wed 08.05.24**	**9**	
1.4.1	**Розробка функціональних можливостей**	**45 days**	**Thu 15.02.24**	**Wed 17.04.24**	**15**	
1.4.1.1	Розробка системи	15 days	Thu 15.02.24	Wed 06.03.24	15	Developer
1.4.1.2	Реалізація корзин	10 days	Thu 07.03.24	Wed 20.03.24	18	Developer
1.4.1.3	Розробка системи	10 days	Thu 21.03.24	Wed 03.04.24	19	Developer

Рисунок Б.3 – Календарний графік проекту

1.3.1.3	Створення дизайн	5 days	Thu 18.01.24	Wed 24.01.24	12	Developer
1.3.1.4	Розробка дизайну	5 days	Thu 25.01.24	Wed 31.01.24	13	Developer
1.3.2	Інтеграція дизайну з	10 days	Thu 01.02.24	Wed 14.02.24	10	Developer
1.4	**Розробка функціональності інтернет-магазину**	**60 days**	**Thu 15.02.24**	**Wed 08.05.24**	**9**	
1.4.1	**Розробка функціональних можливостей**	**45 days**	**Thu 15.02.24**	**Wed 17.04.24**	**15**	
1.4.1.1	Розробка системи	15 days	Thu 15.02.24	Wed 06.03.24	15	Developer
1.4.1.2	Реалізація корзин	10 days	Thu 07.03.24	Wed 20.03.24	18	Developer
1.4.1.3	Розробка системи	10 days	Thu 21.03.24	Wed 03.04.24	19	Developer
1.4.1.4	Реалізація перегл	10 days	Thu 04.04.24	Wed 17.04.24	20	Developer
1.4.2	Тестування та випр	15 days	Thu 18.04.24	Wed 08.05.24	17	QA
1.5	**Наповнення контентом інтернет-магазину**	**20 days**	**Thu 09.05.24**	**Wed 05.06.24**	**16**	
1.5.1	Додавання товарів і	10 days	Thu 09.05.24	Wed 22.05.24	21	Developer
1.5.2	Створення акцій та	5 days	Thu 23.05.24	Wed 29.05.24	24	Developer
1.5.3	Додавання інформа	5 days	Thu 30.05.24	Wed 05.06.24	25	Developer
1.6	**Тестування та підготовка до**	**23 days**	**Thu 06.06.24**	**Mon 08.07.24**	**23**	
1.6.1	Проведення бета-те	8 days	Thu 06.06.24	Mon 17.06.24	26	QA
1.6.2	Проведення альфа-	5 days	Fri 28.06.24	Thu 04.07.24	28	QA
1.6.3	Розміщення на хост	2 days	Fri 05.07.24	Mon 08.07.24	29	Developer
1.7	Підготовка супровідної	1 day	Tue 09.07.24	Tue 09.07.24	27	Developer
1.8	Реліз інтернет-магазину	1 day	Wed 10.07.24	Wed 10.07.24	31	Developer

Рисунок Б.4 – Продовження календарного графіку проєкту

Управління ризиками проєкту. Під час виконання якісної оцінки ризиків треба визначити ризики, які мають бути усунені якнайшвидше. В залежності від ступеня важливості ризику – реагування буде відповідне. Наступним етапом є виконання кількісного оцінювання ризиків. Кількісне та якісне оцінювання можуть виконувати одночасно або окремо, що залежить від ступеня забезпечення проєкту. У таблиці Б.3 надано перелік ризиків даного проєкту. Результати оцінки ризиків надано у таблиці Б.4.

Таблиця Б.3 – Ризики проекту

№ ризику	Назва (опис) ризику
1	Технічні проблеми розробки: Проблеми з програмним забезпеченням, архітектурою сайту або кодом можуть вплинути на функціональність та продуктивність вебсайту.
2	Затримка в розробці: Хвороба, відключення світла та інше.
3	Безпека даних: Недостатні заходи забезпечення безпеки можуть призвести до витоку конфіденційної інформації клієнтів.

4	Залежність від сторонніх сервісів: Використання сторонніх сервісів (наприклад, платіжних шлюзів або API) може призвести до проблем у випадку відмови або змін у цих сервісах.
5	Масштабування та продуктивність: Недостатнє масштабування сайту під високе навантаження може призвести до відмови сайту в пікові навантаження.
6	Зміни в технологіях: Швидкі зміни технологічних стандартів можуть призвести до необхідності постійного оновлення сайту.

Продовження табл. Б.3

№ ризику	Назва (опис) ризику
7	Незрозумілість вимог клієнта: Неясні вимоги майбутнього власника можуть призвести до непередбачуваних змін у розробці.
8	Недостатнє тестування: Недостатнє тестування може призвести до виявлення помилок після запуску та втрат клієнтів.
9	Пандемія
10	Відмова сервера: Перебої в роботі сервера можуть спричинити недоступність сайту для клієнтів.

Таблиця Б.4 – Результати визначення ймовірності, впливу та рангу ризиків проекту

№ ризику	Назва (опис) ризику	Ймовірність (0,1-0,9)	Вплив (0,05-0,8)	Ранг
1	Технічні проблеми розробки	0.3	0.4	0.12
2	Затримки в розробці	0.5	0.2	0.10
3	Безпека даних	0.1	0.8	0.08
4	Залежність від сторонніх сервісів	0.3	0.2	0.06
5	Масштабування та продуктивність	0.3	0.4	0.12
6	Зміни в технологіях	0.1	0.05	0.005
7	Незрозумілість вимог клієнта	0.3	0.2	0.06
8	Недостатнє тестування	0.3	0.4	0.12
9	Пандемія	0.1	0.4	0.04
10	Відмова сервера	0.3	0.2	0.06

Для того, щоб знизити негативний вплив ризиків на проект треба виконати планування реагування на них. До нього входить оцінка наслідків впливу на проект і розробка відповідних заходів. Аналіз виконується за показниками, які описані в таблиці Б.4. У результаті планування заходів

реагування на ризики проєкту було отримано матрицю ймовірності виникнення та впливу ризиків (таблиця Б.5. Зеленим кольором на матриці позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.

Таблиця Б.5 – Матриця ймовірності та впливу згідно проекту

Ймовірність ризику (Й)	Вплив загрози (ризику)				
	Дуже малий	Малий	Середній	Великий	Дуже великий
	0.05	0.1	0.2	0.4	0.8
0.9	0,045	0,09	0,18	0,36	0,72
0.7	0,035	0,07	0,14	0,28	0,56
0.5	0,025	0,05	0,10R2	0,20	0,40
0.3	0,015	0,03	0,06 R4,7,10	0,12R1,5,8	0.24
0.1	0,005 R6	0,01	0,02	0,04R9	0,08R3

Класифікація ризиків проєкту за рівнем, відповідно до отриманого значення індексу, представлена у таблиці Б.6. У таблиці Б.7 описано ризики та стратегії реагування на кожен із них.

Таблиця Б.6 – Шкала оцінювання ризику за рівнем

№	Назва	Межі	Ризики, які входять (номера)
1	Прийнятні	0,005≤R≤0,05	6,9
2	Виправдані	0,05<R≤0,14	1,2,3,4,5,7,8,10
3	Недопустимі	0,14<R≤0,72	

Таблиця Б.7 – Ризики проекту та стратегії реагування

ID ризику	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
1	Виправдний	Проблеми з програмним забезпеченням, архітектурою сайту або кодом можуть вплинути на функціональність та продуктивність вебсайту.	0.3	0.4	0.12	УХИЛЕННЯ	Вибір надійних інструментів та програмних бібліотек.(Python,HTML,CSS,JS,Django)	Розробка плану відновлення та відлагодження.
2	Виправдний	Хвороба, відключення світла та інше.	0.5	0.2	0.10	ЗМЕНШЕННЯ	Розробка гнучкого графіку та робочих процедур.	Запуск резервного плану роботи.
3	Виправдний	Недостатні заходи забезпечення безпеки можуть призвести до витоку конфіденційної інформації клієнтів.	0.1	0.8	0.08	ЗМЕНШЕННЯ	Розробка чіткої специфікації вимог.	Переговори та коригування вимог за потреби.
4	Виправдний	Використання сторонніх сервісів (наприклад, платіжних шлюзів або API) може призвести до проблем у випадку відмови або змін у цих сервісах.	0.3	0.2	0.06	ЗМЕНШЕННЯ	Використовувати надійне обладнання та мережеві резерви.	Використання альтернативних сервісів та розробка резервних методів.
5	Виправдний	Недостатнє масштабування сайту під високе навантаження може призвести до відмови сайту в пікові навантаження.	0.3	0.4	0.12	ЗМЕНШЕННЯ	Розробка гнучкої архітектури та планування масштабування.	Використання додаткових серверів та моніторинг навантаження.

ID ризику	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
6	Прийнятний	Швидкі зміни технологічних стандартів можуть призвести до необхідності постійного оновлення сайту.	0.1	0.05	0.005	ЗМЕНШЕННЯ	Регулярно вивчати та аналізувати технологічні тренди.	Розробка плану відновлення та відлагодження.
7	Виправдний	Неясні вимоги майбутнього власника можуть призвести до непередбачуваних змін у розробці.	0.3	0.2	0.06	ЗМЕНШЕННЯ	Розробка чіткої специфікації вимог.	Переговори та коригування вимог за потреби.
8	Виправдний	Недостатнє тестування може призвести до виявлення помилок після запуску та втрат клієнтів.	0.3	0.4	0.12	ЗМЕНШЕННЯ	Розробка плану тестування та контрольного списку помилок.	Виявлені помилки виправляти якнайшвидше та вивести в стабільний стан.
9	Прийнятний	Пандемія	0.1	0.1	0.01	ЗМЕНШЕННЯ	Розробити алгоритм запобіжних заходів для продовження роботи	-
10	Виправдний	Перебої в роботі сервера можуть спричинити недоступність сайту для клієнтів.	0.3	0.2	0.06	ЗМЕНШЕННЯ	Використовувати надійне обладнання та мережеві резерви.	Використовувати план відновлення та регулярно його перевіряти.

76

ДОДАТОК В

Лістинги основних модулів

base.html

```
{% load static %}
<html lang="en">
<head>
   <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, minimum-scale=1" />
   <title>TamTumba</title>
   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.0/jquery.min.js"
         integrity="sha512-3gJwYpMe3QewGELv8k/BX9vcqhryRdzRMxVfq6ngyWXwo03GFEzjsUm8Q7RZcHPHksttq7/GFoxjCVUjkjvPdw=="
         crossorigin="anonymous" referrerpolicy="no-referrer"></script>
   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css"
      integrity="sha384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65"
      crossorigin="anonymous">
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css"
      integrity="sha512-iecdLmaskl7CVkqkXNQ/ZH/XLlvWZOJyj7Yy7tcenmpD1ypASozpmT/E0iPtmFIB46ZmdtAc9eNBvH0H/ZpiBw=="
      crossorigin="anonymous" referrerpolicy="no-referrer" />
   <link rel="stylesheet" href="{% static 'main/css/style.css' %}" >
   <link rel="icon" href="{% static 'main/img/icon.ico' %}" type="image/png">
   <script type="text/javascript">
      var user='{{request.user}}'
      function getToken(name) {
         let cookieValue = null;
         if (document.cookie && document.cookie !== '') {
            const cookies = document.cookie.split(';');
            for (let i = 0; i < cookies.length; i++) {
               const cookie = cookies[i].trim();
               // Does this cookie string begin with the name we want?
               if (cookie.substring(0, name.length + 1) === (name + '=')) {
                  cookieValue = decodeURIComponent(cookie.substring(name.length + 1));
                  break;
               }
            }
         }
         return cookieValue;
      }
      const csrftoken = getToken('csrftoken');

      function getCookie(name){
         var cookieArr = document.cookie.split(";");
         for(var i = 0;i<cookieArr.length;i++){
```

```
            var cookiePair = cookieArr[i].split("=");

            if(name == cookiePair[0].trim()){
                return decodeURIComponent(cookiePair[1]);
            }
        }
        return null;
    }
    var cart = JSON.parse(getCookie('cart'))
    if(cart == undefined){
        cart = {}
        console.log('Cart was created')
        document.cookie = 'cart=' + JSON.stringify(cart) + ";domain=;path=/"
    }
    console.log('cart:',cart)
  </script>
</head>

<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-dark">
    <div class="container-fluid d-flex justify-content-between">
      <a href="/" class="d-flex align-items-center mb-2 mb-lg-0 text-white text-decoration-none">
        <svg class="bi me-2" width="40" height="32" role="img"
          aria-label="Bootstrap"><use xlink:href="#bootstrap"></use></svg>
      </a>
      <a href="/" class="d-flex align-items-center mb-2 mb-lg-0 text-white text-decoration-none">
        <svg class="bi me-2" width="40" height="32" role="img" aria-label="Bootstrap">
          <use xlink:href="#bootstrap"></use>
        </svg>
        <img src="{% static 'main/img/logo1.png' %}" alt="Logo" class="small-logo">
       </a>
      <ul class="nav col-12 col-lg-auto me-lg-auto mb-2 justify-content-center mb-md-0 d-flex align-
items-center">
        <li><a href="{% url 'furniture_store' %}" class="nav-link px-2 text-white">Store</a></li>
        <li><a href="#" class="nav-link px-2 text-white">FAQs</a></li>
        <li><a href="#" class="nav-link px-2 text-white">About</a></li>
      </ul>
      <div class="form-inline my-2 my-lg-0 d-flex align-items-center">
        {% if request.user.is_authenticated %}
          <p>{{ request.user.username }}</p>
          <a href="{% url 'logout' %}" class="btn btn-outline-warning me-2">Logout</a>
          <a href="{% url 'cabinet' %}">
            <i class="fa-solid fa-user icon-profile" style="color: #ffffff; font-size: 30px; margin: 0
15px 0 15px;"></i>
          </a>
        {% else %}
          <a href="{% url 'login' %}" class="btn btn-outline-light me-2">Login</a>
        {% endif %}
        <a href="{% url 'cart' %}">
          <i class="fa-solid fa-cart-shopping icon-cart" style="color: #ffffff; font-size: 30px;"></i>
        </a>
```

```
            <p id="cart-total">{{cartItems}}</p>
          </div>

        </div>
    </nav>
    <main>
        {% block bodycontent %}{% endblock %}
    </main>
          <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
          integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
          crossorigin="anonymous"></script>
          <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
          integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
          crossorigin="anonymous"></script>
    <script type="text/javascript" src="{% static 'main/js/cart.js' %}"></script>
</body>

</html>
```

main.html

```
{% extends 'main/base.html' %}
{% load static %}
{% block bodycontent %}

<div class="hero_area" style="background-image: url('{% static 'main/img/hero-bg.png' %}');">
<section class="slider_section ">
    <div class="play_btn">
      <a href="">
        <img src="{% static 'main/img/play.png' %}" alt="">
      </a>
    </div>
    <div class="number_box">
      <div>
        <ol class="carousel-indicators indicator-2">
          <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active">01</li>
          <li data-target="#carouselExampleIndicators" data-slide-to="1">02</li>
          <li data-target="#carouselExampleIndicators" data-slide-to="2">03</li>
          <li data-target="#carouselExampleIndicators" data-slide-to="3">04</li>
        </ol>
      </div>
    </div>
    <div class="container">
      <div id="carouselExampleIndicators" class="carousel slide" data-ride="carousel">
        <ol class="carousel-indicators">
          <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>
          <li data-target="#carouselExampleIndicators" data-slide-to="1"></li>
          <li data-target="#carouselExampleIndicators" data-slide-to="2"></li>
          <li data-target="#carouselExampleIndicators" data-slide-to="3"></li>
```

```
        </ol>
        <div class="carousel-inner">
          <div class="carousel-item active">
            <div class="row">
              <div class="col-md-6">
                <div class="detail-box">
                  <h1>
                    The Latest
                    <span>
                      Furniture
                    </span>
                  </h1>
                  <p style="font-size: 20px">
                    Introducing our newest furniture collection, designed to elevate your living spaces with timeless elegance and modern functionality. Crafted with meticulous attention to detail, each piece in this collection seamlessly blends style and comfort to create inviting interiors that reflect your unique taste.
                  </p>
                  <div class="btn-box">
                    <a href="" class="btn-1">
                      Read More
                    </a>
                    <a href="" class="btn-2">
                      Contact us
                    </a>
                  </div>
                </div>
              </div>
              <div class="col-md-6 img-container">
                <div class="img-box">
                  <img src="{% static 'main/img/slider-img.png' %}" alt="" style="margin-top: 10px; margin-left: -50px">
                </div>
              </div>
            </div>
          </div>
          <div class="carousel-item ">
            <div class="row">
              <div class="col-md-6">
                <div class="detail-box">
                  <h1>
                    Collection
                    <span>
                      <br>for you
                    </span>
                  </h1>
                  <p style="font-size: 20px">
                    From luxurious sofas upholstered in sumptuous fabrics to sleek coffee tables that serve as focal points in any room, our latest collection offers a diverse range of furniture to suit every aesthetic preference. Embrace the art of relaxation with our plush armchairs and ottomans, perfect for cozy reading nooks or evening unwind sessions.
                  </p>
```

```
            <div class="btn-box">
              <a href="" class="btn-1">
                Read More
              </a>
              <a href="" class="btn-2">
                Contact us
              </a>
            </div>
          </div>
        </div>
        <div class="col-md-6 img-container">
          <div class="img-box" style="">
            <img src="{% static 'main/img/slider-img2.png' %}" alt="" style="width: 150% ; height: 150%; margin-left: -100px">
          </div>
        </div>
      </div>
    </div>
    <div class="carousel-item ">
      <div class="row">
        <div class="col-md-6">
          <div class="detail-box">
            <h1>
              Style and quality
              <span>
                <br> for you
              </span>
            </h1>
            <p style="font-size: 20px">
              Infuse your dining area with sophistication and charm with our exquisite dining sets, featuring elegant tables and chairs crafted from high-quality materials for lasting durability. Whether you prefer classic designs or contemporary silhouettes, our collection has something to suit every dining space.
            </p>
            <div class="btn-box">
              <a href="" class="btn-1">
                Read More
              </a>
              <a href="" class="btn-2">
                Contact us
              </a>
            </div>
          </div>
        </div>
        <div class="col-md-6 img-container">
          <div class="img-box">
            <img src="{% static 'main/img/slider-img3.png' %}" alt="" style="margin-top: 50px; margin-left: -50px">
          </div>
        </div>
      </div>
```

```
          </div>
          <div class="carousel-item ">
            <div class="row">
              <div class="col-md-6">
                <div class="detail-box">
                  <h1>
                    Crafted with details
                    <span>
                      <br> for you
                    </span>
                  </h1>
                  <p style="font-size: 20px">
                    Discover the perfect accents to complete your home decor with our curated selection of accessories, including decorative mirrors, chic lamps, and statement-making rugs. Transform your space into a haven of style and relaxation with our latest furniture collection, where every piece is crafted to inspire and delight
                  </p>
                  <div class="btn-box">
                    <a href="" class="btn-1">
                      Read More
                    </a>
                    <a href="" class="btn-2">
                      Contact us
                    </a>
                  </div>
                </div>
              </div>
              <div class="col-md-6 img-container">
                <div class="img-box">
                  <img src="{% static 'main/img/slider-img4.webp' %}" alt="" style=" width: 110%; height: 110% ;margin-top: 50px; margin-left: -50px">
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </section>
  </div>

  <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.2.1/owl.carousel.min.js">
  </script>
  <script type="text/javascript">
    $(".owl-carousel").owlCarousel({
      loop: true,
      margin: 10,
      nav: true,
      navText: [],
      autoplay: true,
```

```
    autoplayHoverPause: true,
    responsive: {
      0: {
        items: 1
      },
      420: {
        items: 2
      },
      1000: {
        items: 5
      }
    }

  });
 </script>
 <script>
  var nav = $("#navbarSupportedContent");
  var btn = $(".custom_menu-btn");
  btn.click
  btn.click(function (e) {

    e.preventDefault();
    nav.toggleClass("lg_nav-toggle");
    document.querySelector(".custom_menu-btn").classList.toggle("menu_btn-style")
  });
 </script>
 <script>
  $('.carousel').on('slid.bs.carousel', function () {
    $(".indicator-2 li").removeClass("active");
    indicators = $(".carousel-indicators li.active").data("slide-to");
    a = $(".indicator-2").find("[data-slide-to='" + indicators + "']").addClass("active");
    console.log(indicators);
  })

  $('.play_btn a').click(function(e) {
    e.preventDefault();

    var carousel = $("#carouselExampleIndicators");

    carousel.carousel('next');
  });
 </script>
{% endblock %}
```

login.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>TamTumba</title>
  <link rel="stylesheet" href="{% static 'main/css/loginformstyle.css' %}">
  <link rel="icon" href="{% static 'main/img/icon.ico' %}" type="image/png">
</head>
<body>
<a href="{% url 'home' %}" class="home-button">Back to Main page</a>
  <div class="container">
    <input type="checkbox" id="check">
    <div class="login form">
      <header>Login</header>
      <form method="post">
        {% csrf_token %}
        <label class="input">Email</label>
          {{ form.email }}
        <label class="input"> Password</label>
          {{ form.password }}
         {% if form.errors %}
          {% for field in form %}
            {{ field.errors }}
          {% endfor %}
          {{ form.non_field_errors }}
        {% endif %}
        {% if invalid_login %}
          <p class="error" style="color:red; text-align:center">Invalid email or password.</p>
        {% endif %}
        <a href="#">Forgot password?</a>
        <input type="submit" class="button" value="Login">
      </form>
      <div class="signup">
        <span class="signup">Don't have an account?
           <a href="{% url 'signup' %}">Signup</a>
        </span>
      </div>
    </div>
  </div>
</body>
</html>
```

signup.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>TamTumba</title>
  <link rel="stylesheet" href="{% static 'main/css/loginformstyle.css' %}">
```

```
  <link rel="icon" href="{% static 'main/img/icon.ico' %}" type="image/png">
</head>
<a href="{% url 'home' %}" class="home-button">Back to Main page</a>
<body>
  <div class="container">
    <input type="checkbox" id="check">
    <div class="form">
      <header>Signup</header>
      <form method="post">
        {% csrf_token %}
        <label class="input">Enter your First Name</label>
          {{ customer_form.name }}
        <label class="input">Enter your Second Name</label>
          {{ customer_form.surname }}
        <label class="input">Enter your Email</label>
          {{ form.email }}
        <label class="input">Enter your Password</label>
          {{ form.password1 }}
        <label class="input">Repeat Password</label>
          {{ form.password2 }}
        {% if form.errors or form.non_field_errors %}
          {% for field in form%}
            {{ field.errors }}
          {% endfor %}
          {{ form.non_field_errors }}
        {% endif %}
        <input type="submit" class="button" value="Signup">
      </form>
      <div class="signup">
        <span class="signup">Already have an account?
         <a href="{% url 'login' %}">Login</a>
        </span>
      </div>
    </div>
  </div>
</body>
</html>
```

style.css

```
body{
    background: #2c2c2c;
}

.nav-link {
    font-size: 20px;
    margin-left: 10px;
}

.small-logo {
    width: 200px;
```

```
    height: auto;
    margin-left: -80px;
}

h1,h2,h3,h4,h5,h6{
        color:hsl(0, 0%, 30%);
}

.box-element{
        box-shadow:hsl(0, 0%, 80%) 0 0 16px;
        background-color: #fff;
        border-radius: 10px;
        padding: 10px;
        overflow: hidden;
}

.thumbnail{
        width: 100%;
        height: 200px;
        -webkit-box-shadow: -1px -3px 5px -2px rgba(214,214,214,1);
    -moz-box-shadow: -1px -3px 5px -2px rgba(214,214,214,1);
    box-shadow: -1px -3px 8px -2px rgba(214,214,214,1);
    border-top-left-radius: 10px;
    border-top-right-radius: 10px;
}
.product{
        border-radius: 0 0 4px 4px;
}
.product-grid {
    margin-left: 50px; /* Видаліть цей стиль, якщо він не потрібен */
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
}
.product-grid .row {
    width: 100%;
}

.product-grid .col-lg-4 {
    width: calc(33.33% - 20px);
    margin-bottom: 20px;
    padding: 0 10px;
}
.bg-dark{
        background-color: #232323 !important;
}

#cart-total{
        display: block;
        text-align: center;
        color:#fff;
```

```
        background-color: #800000;
        width: 25px;
        height: 25px;
        border-radius: 50%;
        font-size: 17px;
}


.col-lg-4, .col-lg-6, .col-lg-8, .col-lg-12{
        margin-top: 10px;
        padding: 0 15px;
        margin-bottom: 10px;
}

.btn{
        border-radius: 15;
}
.btn-outline-success{
   color: #FF7100;
   border-color: #FF7100;
}
.btn-outline-success:hover{
   background-color: #FF7100;
   border-color: #FF7100;
}
.row-image{
        width: 100px;
}

.form-field{
        width: 250px;
        display: inline-block;
        padding: 5px;
}

.cart-row{
        display: flex;
   align-items: flex-stretch;
   padding-bottom: 10px;
   margin-bottom: 10px;
   border-bottom: 1px solid #ececec;

}

.quantity{
        display: inline-block;
        font-weight: 700;
        padding-right:10px;
}

.chg-quantity{
```

```
        width: 12px;
        cursor: pointer;
        display: block;
        margin-top: 5px;
        transition:.1s;
}

.chg-quantity:hover{
        opacity: .6;
}

/*Detail product*/
.hidden{
        display: none!important;
}

.product-self {
    max-width: 100%;
    max-height: 100%;
    width: auto;
    height: auto;
    margin-bottom: 10px;
}

.product-image {
    max-width: 100%;
    height: auto;
    border-radius: 10px;
}
.product-info {
    padding: 20px;
    background-color: #fff;
    border-radius: 10px;
    box-shadow: 0 0 16px rgba(0, 0, 0, 0.1);
}
.box-element p {
    font-size: 18px;
}
/*Slider*/
.slider-for img{
    width: 100%;
    height: 80%
}
.slider-nav img {
    height: 20%;
    overflow: hidden;
    margin-left: 5px;
}

/* Category */
.category-sidebar {
```

```
    margin-left: -80px;
    margin-top: 10px;
    background-color: #fff;
    padding: 15px;
    border-radius: 5px;
    width: 200px;
    overflow-x: auto; /* Для горизонтальної прокрутки, якщо необхідно */
    white-space: nowrap; /* Якщо вибір категорій має багато елементів */
}

.category-sidebar h3 {
    font-size: 20px;
    margin-bottom: 10px;
}

.category-menu {
    list-style: none;
    padding: 0;
}

.category-menu li {
    margin-bottom: 10px;
}

.category-menu a {
    text-decoration: none;
    color: #333;
    font-size: 18px;
}

.category-menu a:hover {
    color: #FF7100;
}

.active a {
    font-weight: bold;
    color: #FF7100;
}

/*slider section*/
.slider_section {
  height: calc(90% - 70px);
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-align: center;
      -ms-flex-align: center;
          align-items: center;
}

.slider_section .container {
```

```
  position: relative;
  z-index: 2;
}

.slider_section .number_box {
  width: 50px;
  position: absolute;
  right: 25px;
  top: 46%;
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-orient: vertical;
  -webkit-box-direction: normal;
      -ms-flex-direction: column;
          flex-direction: column;
  -webkit-box-align: center;
      -ms-flex-align: center;
          align-items: center;
  color: #ffffff;
  -webkit-transform: translateY(-50%);
          transform: translateY(-50%);
  z-index: 2;
  font-size: 22px;
}

.slider_section .number_box::before {
  content: "";
  width: 50px;
  height: 1px;
  background-color: #ffffff;
  position: absolute;
  top: 50%;
  left: 50%;
  -webkit-transform: translate(-50%, -50%);
          transform: translate(-50%, -50%);
}

.slider_section .row {
  -webkit-box-align: center;
      -ms-flex-align: center;
          align-items: center;
}

.slider_section .detail-box h1 {
  color: #FF7100;
  text-transform: uppercase;
}

.slider_section .detail-box h1 span {
  font-size: 5rem;
```

```
  color: #fff;
}

.slider_section .detail-box p {
  margin-top: 15px;
  color: #ffffff;
}

.slider_section .detail-box .btn-box {
  margin-top: 45px;
}

.slider_section .detail-box .btn-box a {
  text-transform: uppercase;
  text-align: center;
  width: 135px;
  font-size: 15px;
}

.slider_section .detail-box .btn-box .btn-1 {
  display: inline-block;
  padding: 6px 0;
  background-color: #FF7100;
  color: #ffffff;
  -webkit-transition: all 0.3s;
  transition: all 0.3s;
  border: 1px solid transparent;
  border-radius: 20px;
  margin-right: 5px;
  text-decoration: none;
}

.slider_section .detail-box .btn-box .btn-1:hover {
  background-color: transparent;
  border-color: #FF7100;
  color: #FF7100;
}

.slider_section .detail-box .btn-box .btn-2 {
  display: inline-block;
  padding: 6px 0;
  background-color: #fff;
  color: #252525;
  -webkit-transition: all 0.3s;
  transition: all 0.3s;
  border: 1px solid transparent;
  border-radius: 20px;
  text-decoration: none;
}

.slider_section .detail-box .btn-box .btn-2:hover {
```

```
  background-color: transparent;
  border-color: #fff;
  background-color: #252525;
  color: #fff;
}

.slider_section .img-container {
  position: relative;
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-pack: end;
      -ms-flex-pack: end;
          justify-content: flex-end;
}

.slider_section .img-container .img-box {
  width: 250px;
}

.slider_section .img-container .img-box img {
  width: 100%;
  height: 100%;
}

.slider_section .play_btn {
  background-color: #FF7100;
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-align: center;
      -ms-flex-align: center;
          align-items: center;
  -webkit-box-pack: center;
      -ms-flex-pack: center;
          justify-content: center;
  border-radius: 100%;
  width: 75px;
  height: 75px;
  position: absolute;
  z-index: 3;
  top: 56%;
  right: 32%;
}

.slider_section .play_btn a {
  background-color: #ffffff;
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-align: center;
```

```
    -ms-flex-align: center;
        align-items: center;
  -webkit-box-pack: center;
    -ms-flex-pack: center;
        justify-content: center;
  border-radius: 100%;
  width: 75px;
  height: 75px;
  position: relative;
  z-index: 5;
}

.slider_section .play_btn img {
  width: 20px;
  margin-left: 3px;
}

.slider_section .play_btn::before {
  content: "";
  width: 100%;
  height: 100%;
  position: absolute;
  top: 50%;
  left: 50%;
  background-color: #FF7100;
  opacity: 1;
  border-radius: 100%;
  -webkit-transform: translate(-50%, -50%);
      transform: translate(-50%, -50%);
}

.slider_section .play_btn::before {
  z-index: 2;
  -webkit-animation: before-animation 1500ms infinite;
      animation: before-animation 1500ms infinite;
}

@-webkit-keyframes before-animation {
  0% {
    -webkit-transform: translateX(-50%) translateY(-50%) translateZ(0) scale(1);
        transform: translateX(-50%) translateY(-50%) translateZ(0) scale(1);
    opacity: 1;
  }
  100% {
    -webkit-transform: translateX(-50%) translateY(-50%) translateZ(0) scale(1.5);
        transform: translateX(-50%) translateY(-50%) translateZ(0) scale(1.5);
    opacity: 1;
  }
}

@keyframes before-animation {
```

```
  0% {
    -webkit-transform: translateX(-50%) translateY(-50%) translateZ(0) scale(1);
            transform: translateX(-50%) translateY(-50%) translateZ(0) scale(1);
    opacity: 1;
  }
  100% {
    -webkit-transform: translateX(-50%) translateY(-50%) translateZ(0) scale(1.5);
            transform: translateX(-50%) translateY(-50%) translateZ(0) scale(1.5);
    opacity: 1;
  }
}

.slider_section .carousel-indicators {
  margin: 0;
  -webkit-box-align: center;
      -ms-flex-align: center;
          align-items: center;
  -webkit-box-pack: center;
      -ms-flex-pack: center;
          justify-content: center;
  bottom: -150px;
}

.slider_section .carousel-indicators li {
  width: 15px;
  height: 15px;
  border-radius: 100%;
  background-color: #FF7100;
  opacity: 1;
  border: none;
  background-clip: unset;
}

.slider_section .carousel-indicators li.active {
  background-color: #fff;
}

.slider_section ol.carousel-indicators.indicator-2 {
  position: unset;
}

.slider_section ol.carousel-indicators.indicator-2 li {
  width: 25px;
  height: 25px;
  text-indent: 0px;
  text-align: center;
  line-height: 25px;
  color: #ffffff;
  margin: 0;
  background-color: transparent;
  opacity: 1;
```

```
  border: none;
  display: none;
}

.slider_section ol.carousel-indicators.indicator-2 li.active {
  display: block;
}

/*back ground section*/
.hero_area {
  height: 100vh;
  background-size: cover;
  position: relative;
  background-repeat: no-repeat;
  overflow: hidden;
  background-position: center 200%;
}

.hero_area::after {
  content: "";
  position: absolute;
  bottom: 0;
  right: 0;
  width: 45%;
  height: 100%;
  background-image: url('../img/hero-side-bg.png');
  background-size: cover;
}
/*Стили для фильтра по цене */
.price-filter-form label {
    display: block;
    margin-bottom: 10px;
    font-size: 16px;
}

.price-filter-form input {
    width: 45%;
    padding: 8px;
    border: 1px solid #ccc;
    border-radius: 5px;
    margin-bottom: 10px;
}

.price-filter-form button {
    margin-top: 10px;
    width:100%;
    padding: 10px;
    background-color: #FF7100;
    color: #fff;
    border: none;
    border-radius: 5px;
```

```
    cursor: pointer;
    font-size: 16px;
}

.price-filter-form button:disabled {
    background-color: #ccc;
    cursor: not-allowed;
}

.price-filter-form .error {
    color: red;
    font-size: 14px;
    margin-top: 5px;
}

/* Стили для фильтра по цене */
#priceSlider .noUi-connect {
    background: #FF7100;
}
#priceSlider.noUi-horizontal {
    height: 20px;
    background: #252525;
    width: 97%;
    margin-right: 5px;
    margin-left: 2px;

}
.noUi-handle:hover {
    background: #FF7100;
    box-shadow: 0 0 5px #FF7100;
}
/*  Pagination styles */
    .pagination {
        margin-top: 20px;
        text-align: center;
    }

    .pagination a, .pagination span {
        display: inline-block;
        padding: 5px 10px;
        margin: 0 2px;
        border: 1px solid #ccc;
        background-color: #f8f9fa;
        color: #343a40;
        text-decoration: none;
    }

    .pagination .current-page {
        font-weight: bold;
        background-color: #007bff;
        color: #fff;
```

```
        border-color: #007bff;
    }

    .pagination a:hover {
        background-color: #dee2e6;
    }

    .pagination span {
        pointer-events: none;
    }

.pagination-wrapper {
    margin-top: 10px;
    text-align: center;
}

.pagination {
    display: inline-block;
}

.pagination a, .pagination .current-page {
    display: inline-block;
    padding: 5px 10px;
    margin: 0 2px;
    border: 1px solid #ccc;
    background-color: #f8f9fa;
    color: #343a40;
    text-decoration: none;
}

.pagination .current-page {
    font-weight: bold;
    background-color: #007bff;
    color: #fff;
    border-color: #007bff;
}

.pagination a:hover {
    background-color: #dee2e6;
}



/* Додаткові адаптивні стилі для різних розмірів екрану */
/* Для екранів шириною менше 768px */
@media (max-width: 767px) {
    .nav-link {
        font-size: 16px;
        margin-left: 5px;
    }
    .small-logo {
```

```
        width: 150px;
        margin-left: -40px;
    }
    h1, h2, h3, h4, h5, h6 {
        font-size: 18px;
    }
    .box-element p {
        font-size: 16px;
    }
    .col-lg-4, .col-lg-6, .col-lg-8, .col-lg-12 {
        padding: 0 10px;
    }
    .row-image {
        width: 80px;
    }
    .form-field {
        width: 200px;
    }
}

/* Для екранів шириною від 768px до 991px */
@media (min-width: 768px) and (max-width: 991px) {
    .nav-link {
        font-size: 18px;
    }
    .small-logo {
        width: 170px;
        margin-left: -60px;
    }
    h1, h2, h3, h4, h5, h6 {
        font-size: 24px;
    }
    .box-element p {
        font-size: 20px;
    }
    .col-lg-4, .col-lg-6, .col-lg-8, .col-lg-12 {
        margin-top: 15px;
        padding: 0 10px;
    }
    .row-image {
        width: 90px;
    }
    .form-field {
        width: 220px;
    }
}

/* Для екранів шириною 992px і більше */
@media (min-width: 992px) {
    .nav-link {
        font-size: 20px;
```

```
    }
    .small-logo {
        width: 200px;
        margin-left: -80px;
    }
    h1, h2, h3, h4, h5, h6 {
        font-size: 28px;
    }
    .box-element p {
        font-size: 22px;
    }
    .col-lg-4, .col-lg-6, .col-lg-8, .col-lg-12 {
        margin-top: 10px;
        padding: 0 15px;
    }
    .row-image {
        width: 100px;
    }
    .form-field {
        width: 250px;
    }
}
@media (max-width: 1440px) {
    .category-sidebar {
        width: auto;
        overflow-x: auto;
    }
    .product-grid .col-lg-4 {
        width: calc(50% - 20px); /* Щоб розділити на 2 колонки */
    }
}

@media (max-width: 767px) {
    .product-grid .col-lg-4 {
        width: calc(100% - 20px); /* На маленьких екранах товари відображатимуться на весь екран */
    }
}

@media (max-width: 1400px) {
    .category-sidebar {
        position: relative;
        width: calc(100% - 20px);
        margin: 0 auto;
        margin-top: 20px; /* Додайте бажану відстань від nav menu */
    }

    .category-sidebar h3 {
        font-size: 16px; /* Зменште розмір тексту заголовків */
    }

    .category-sidebar ul.category-menu {
```

```
        margin-top: 10px;
    }

    .category-sidebar ul.category-menu li {
        margin-bottom: 5px;
    }
}

@media (max-width: 767px) {
    .category-sidebar {
        width: 100%;
    }

    .category-sidebar h3 {
        font-size: 14px; /* Додайте ще одне зменшення розміру тексту для мобільних пристроїв */
    }
}
@media (max-width: 1400px) {
    .category-sidebar {
        top: 0;
        bottom : 0;
        left: 100%; /* розміщуємо в центрі */
        transform: translateX(-50%); /* центруємо відносно батьківського блоку */
        width: 200px;
        z-index: 1; /* забезпечуємо, щоб sidebar був вище, ніж навігаційне меню */
    }

    .category-menu li {
    display: inline-block; /* Змінюємо стиль відображення на inline-block */
    margin-right: 10px; /* Додаємо відступ між елементами */
    }
    .price-filter-form label {
    font-size: 14px; /* Зменшуємо розмір шрифту міток */
    }

    .price-filter-form input {
        width: 40%; /* Зменшуємо ширину полів введення */
        padding: 6px; /* Зменшуємо внутрішній відступ */
    }

    .price-filter-form button {
        margin-top: 10px;
        width: 60%; /* Зменшуємо ширину кнопки */
        padding: 8px; /* Зменшуємо внутрішній відступ */
        font-size: 14px; /* Зменшуємо розмір шрифту кнопки */
    }

    .product-grid {
        margin-top: 10px; /* збільшуємо відступ товарів зверху, щоб вони не перекривали sidebar та навігаційне меню */
    }
```

```
    .navbar {
        position: relative;
        z-index: 0; /* забезпечуємо, щоб навігаційне меню було позаду sidebar */
    }
}

@media (max-width: 1024px) {
    .category-sidebar {
        top: 0;
        bottom : 0;
        left: 50%; /* розміщуємо в центрі */
        transform: translateX(-50%); /* центруємо відносно батьківського блоку */
        width: 250%;
        z-index: 1; /* забезпечуємо, щоб sidebar був вище, ніж навігаційне меню */
    }

    .category-menu li {
    display: inline-block; /* Змінюємо стиль відображення на inline-block */
    margin-right: 10px; /* Додаємо відступ між елементами */
    }
    .price-filter-form label {
    font-size: 14px; /* Зменшуємо розмір шрифту міток */
    }

    .price-filter-form input {
        width: 40%; /* Зменшуємо ширину полів введення */
        padding: 6px; /* Зменшуємо внутрішній відступ */
    }

    .price-filter-form button {
        margin-top: 10px;
        width: 60%; /* Зменшуємо ширину кнопки */
        padding: 8px; /* Зменшуємо внутрішній відступ */
        font-size: 14px; /* Зменшуємо розмір шрифту кнопки */
    }

    .product-grid {
        margin-top: 10px; /* збільшуємо відступ товарів зверху, щоб вони не перекривали sidebar та навігаційне меню */
    }

    .navbar {
        position: relative;
        z-index: 0; /* забезпечуємо, щоб навігаційне меню було позаду sidebar */
    }
}
@media (max-width: 768px) {
    .category-sidebar {
        top: 0;
        bottom : 0;
```

```
        left: 325%; /* розміщуємо в центрі */
        transform: translateX(-50%); /* центруємо відносно батьківського блоку */
        width: 250%;
        z-index: 1; /* забезпечуємо, щоб sidebar був вище, ніж навігаційне меню */
    }

    .category-menu li {
    display: inline-block; /* Змінюємо стиль відображення на inline-block */
    margin-right: 10px; /* Додаємо відступ між елементами */
    }
    .price-filter-form label {
    font-size: 14px; /* Зменшуємо розмір шрифту міток */
    }

    .price-filter-form input {
        width: 40%; /* Зменшуємо ширину полів введення */
        padding: 6px; /* Зменшуємо внутрішній відступ */
    }

    .price-filter-form button {
        margin-top: 10px;
        width: 60%; /* Зменшуємо ширину кнопки */
        padding: 8px; /* Зменшуємо внутрішній відступ */
        font-size: 14px; /* Зменшуємо розмір шрифту кнопки */
    }

    .product-grid {
        margin-top: 10px; /* збільшуємо відступ товарів зверху, щоб вони не перекривали sidebar та навігаційне меню */
    }

    .navbar {
        position: relative;
        z-index: 0; /* забезпечуємо, щоб навігаційне меню було позаду sidebar */
    }
}
@media (max-width: 450px) and (min-width: 320px){
    .category-sidebar {
        top: 0;
        bottom : 0;
        left: 50%; /* розміщуємо в центрі */
        transform: translateX(-50%); /* центруємо відносно батьківського блоку */
        width: 85%;
        z-index: 1; /* забезпечуємо, щоб sidebar був вище, ніж навігаційне меню */
    }

    .category-menu li {
    display: inline-block; /* Змінюємо стиль відображення на inline-block */
    margin-right: 10px; /* Додаємо відступ між елементами */
    }
    .price-filter-form label {
```

```
    font-size: 14px; /* Зменшуємо розмір шрифту міток */
    }

    .price-filter-form input {
        width: 40%; /* Зменшуємо ширину полів введення */
        padding: 6px; /* Зменшуємо внутрішній відступ */
    }

    .price-filter-form button {
        margin-top: 10px;
        width: 60%; /* Зменшуємо ширину кнопки */
        padding: 8px; /* Зменшуємо внутрішній відступ */
        font-size: 14px; /* Зменшуємо розмір шрифту кнопки */
    }

    .product-grid {
        margin-top: 10px; /* збільшуємо відступ товарів зверху, щоб вони не перекривали sidebar та навігаційне меню */
    }

    .navbar {
        position: relative;
        z-index: 0; /* забезпечуємо, щоб навігаційне меню було позаду sidebar */
    }
}
```

loginformstyle.css

```
/* Import Google font - Poppins */
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500;600;700&display=swap');

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}

body {
    min-height: 100vh;
    width: 100%;
    background: #222222;
}

.container {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    max-width: 430px;
```

```
    width: 90%; /* Adjusted for responsiveness */
    background: #fff;
    border-radius: 7px;
    box-shadow: 0 5px 10px rgba(0, 0, 0, 0.3);
}

.container .registration {
    display: none;
}

#check:checked ~ .registration {
    display: block;
}

#check:checked ~ .login {
    display: none;
}

#check {
    display: none;
}

.container .form {
    padding: 2rem;
}

.form header {
    font-size: 2rem;
    font-weight: 500;
    text-align: center;
    margin-bottom: 1.5rem;
}

.form input {
    height: 50px; /* Reduced height for responsiveness */
    width: 100%;
    padding: 0 15px;
    font-size: 16px; /* Reduced font size for responsiveness */
    margin-bottom: 1.3rem;
    border: 1px solid #ddd;
    border-radius: 6px;
    outline: none;
}

.form input:focus {
    box-shadow: 0 1px 0 rgba(0, 0, 0, 0.2);
}

.form a {
    font-size: 16px;
    color: #FF7100;
```

```
    text-decoration: none;
}

.form a:hover {
    text-decoration: underline;
}

.form input.button {
    color: #fff;
    background: #222222;
    font-size: 1rem; /* Reduced font size for responsiveness */
    font-weight: 500;
    letter-spacing: 1px;
    margin-top: 1.5rem; /* Adjusted margin for responsiveness */
    cursor: pointer;
    transition: 0.4s;
}

.form input.button:hover {
    background: #363636;
}

.signup {
    font-size: 16px; /* Reduced font size for responsiveness */
    text-align: center;
}

.signup label {
    color: #009579;
    cursor: pointer;
}

.signup label:hover {
    text-decoration: underline;
}

.home-button {
    position: absolute;
    top: 10px;
    left: 10px;
    text-decoration: none;
    background-color: #333;
    color: #fff;
    padding: 5px 10px;
    border-radius: 5px;
}
```

forms.py

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
```

```
from custom_user.models import User
from cart.models import *

class SignupForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['email', 'password1', 'password2']

    email = forms.CharField(widget=forms.EmailInput(attrs={
        'placeholder': 'sobaka@gmail.com',
        'class': '.form input',
    }))
    password1 = forms.CharField(widget=forms.PasswordInput(attrs={
        'placeholder': '@!@#sadd123',
        'class': '.form input',
    }))
    password2 = forms.CharField(widget=forms.PasswordInput(attrs={
        'placeholder': 'repeat password',
        'class': '.form input',
    }))


class CustomerForm(forms.ModelForm):
    class Meta:
        model = Customer
        fields = ['name', 'surname', 'email']

    name = forms.CharField(widget=forms.TextInput(attrs={
        'placeholder': 'IVANOV',
        'class': '.form input',
    }))
    surname = forms.CharField(widget=forms.TextInput(attrs={
        'placeholder': 'IVANOV',
        'class': '.form input',
    }))

    def save(self, commit=True):
        customer = super(CustomerForm, self).save(commit=False)
        customer.email = self.cleaned_data['email']

        if commit:
            customer.save()

        return customer

class LoginForm(forms.Form):
    email = forms.CharField(widget=forms.EmailInput(attrs={
        'placeholder': 'Enter your Email',
        'class': '.form input',}))
    password = forms.CharField(widget=forms.PasswordInput(attrs={
        'placeholder': 'Enter your Password',
```

```
        'class': '.form input', }))

class PriceFilterForm(forms.Form):
    min_price = forms.DecimalField(label='Min price', required=False)
    max_price = forms.DecimalField(label='Max price', required=False)
```

utils.py

```
import json
from furniturestore.models import *
from cart.models import *

def cookieCart(request):
    try:
        cart = json.loads(request.COOKIES['cart'])
    except:
        cart = {}

    items = []
    order = {'get_cart_total': 0, 'get_cart_items': 0}
    cartItems = order['get_cart_items']

    for i in cart:
        try:
            cartItems += cart[i]['quantity']

            product = FurnitureProduct.objects.get(id=i)
            if product.discount_price:
                total = (product.discount_price * cart[i]['quantity'])
            else:
                total = (product.price * cart[i]['quantity'])
            order['get_cart_total'] += total
            order['get_cart_items'] += cart[i]['quantity']
            item = {
                'product': {
                    'id': product.id,
                    'name': product.name,
                    'price': product.price,
                    'discount_price': product.discount_price,
                    'imageURL': product.imageURL
                },
                'quantity': cart[i]['quantity'],
                'get_total': total,
            }
            items.append(item)
        except:
            pass
    return {'cartItems':cartItems,'order':order,'items':items}

def cartData(request):
    if request.user.is_authenticated:
```

```
        customer = request.user.customer
        order,created = Order.objects.get_or_create(customer=customer, complete=False)
        items = order.orderitem_set.all()
        cartItems = order.get_cart_items
    else:
        coockieData = cookieCart(request)
        cartItems = coockieData['cartItems']
        order = coockieData['order']
        items = coockieData['items']
    return {'cartItems':cartItems,'order':order,'items':items}

def guestOrder(request,data):
    print('User is not logged in')
    print('COOKIES', request.COOKIES)
    name = data['form']['name']
    surname = data['form']['surname']
    email = data['form']['email']
    cookieData = cookieCart(request)
    items = cookieData['items']
    customer, created = Customer.objects.get_or_create(
        email=email,
    )
    customer.name = name
    customer.surname = surname
    customer.save()
    order = Order.objects.create(
        customer=customer,
        complete=False
    )
    for item in items:
        product = FurnitureProduct.objects.get(id=item['product']['id'])
        orderItem = OrderItem.objects.create(
            product=product,
            order=order,
            quantity=item['quantity']
        )
    return customer,order
```

views.py

```
from django.contrib.auth import authenticate,login,logout
from django.http import HttpResponse
from django.shortcuts import render,redirect
from django.contrib.auth.hashers import make_password
from cart.models import *
from .models import *
from utils import cookieCart,cartData
from forms import *
def home_page(request):

    data = cartData(request)
```

```
    cartItems = data['cartItems']
    order = data['order']
    items = data['items']

    products = FurnitureProduct.objects.all()
    context = {'products': products, 'cartItems': cartItems}
    return render(request,'main/main.html',context)


def user_signup(request):
    if request.method == 'POST':
        form = SignupForm(request.POST)
        customer_form = CustomerForm(request.POST)
        if form.is_valid() and customer_form.is_valid():
            # Создайте или получите пользователя по email
            email = form.cleaned_data['email']
            password = form.cleaned_data['password1']

            # Перед сохранением пароля хешируйте его с помощью make_password
            hashed_password = make_password(password)
            user, created = User.objects.get_or_create(email=email, defaults={'password': hashed_password})

            # Создайте объект Customer и свяжите его с пользователем
            customer = customer_form.save(commit=False)
            customer.user = user
            customer.save()

            return redirect('login')
    else:
        form = SignupForm()
        customer_form = CustomerForm()
    return render(request, 'main/signup.html', {'form': form, 'customer_form': customer_form})


# login page
def user_login(request):
    invalid_login = False
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            email = form.cleaned_data['email']
            password = form.cleaned_data['password']
            user = authenticate(request, email=email, password=password)
            if user is not None:
                if user.is_active:
                    login(request, user)
                    return redirect('home')
                else:
                    pass
            else:
                invalid_login = True
```

```
    else:
        form = LoginForm()
    return render(request, 'main/login.html', {'form': form, 'invalid_login': invalid_login})

# logout page
def user_logout(request):
    logout(request)
    return redirect('login')

def cabinet(request):
    data = cartData(request)
    cartItems = data['cartItems']
    user = request.user
    user_info = Customer.objects.get(user=user)
    if user_info.bonus is None:
        user_info.bonus = 0
        user_info.save()
    bonus_balance = user_info.bonus
    products = FurnitureProduct.objects.all()
    context = {'products': products, 'cartItems': cartItems, 'user': user_info, 'bonus_balance': bonus_balance}
    return render(request, 'main/cabinet.html', context)
```

urls.py

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


urlpatterns=[
    path('',home_page,name='home'),
    path('login/', user_login, name='login'),
    path('signup/', user_signup, name='signup'),
    path('logout/', user_logout, name='logout'),
    path('cabinet/',cabinet,name='cabinet'),
]
```

models.py

```
from django.db import models
from django.conf import settings
from furniturestore.models import *
from decimal import Decimal

class Customer(models.Model):
    user = models.OneToOneField(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, null=True, blank=True)
    name = models.CharField(max_length=200, null=True)
    surname = models.CharField(max_length=200, null=True)
    email = models.EmailField(max_length=200, null=True)
```

```
    bonus = models.DecimalField(max_digits=10, decimal_places=2, null=True, blank=True)

    def __str__(self):
        return self.name

class Order(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.SET_NULL, blank=True, null=True)
    date_ordered = models.DateTimeField(auto_now_add=True)
    complete = models.BooleanField(default=False, null=True, blank=False)
    transaction_id = models.CharField(max_length=200,null=True)

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

    @property
    def get_cart_total(self):
        orderitems = self.orderitem_set.all()
        total = sum([item.get_total for item in orderitems])
        return total

    @property
    def get_cart_items(self):
        orderitems = self.orderitem_set.all()
        total = sum([item.quantity for item in orderitems])
        return total

    def bonusCount(self):
        # Обчислюємо бонуси на основі суми замовлення (5%)
        bonus_amount = self.get_cart_total * Decimal('0.05')
        if bonus_amount > 0:
            if not self.customer.bonus:
                self.customer.bonus = bonus_amount
            else:
                self.customer.bonus += bonus_amount
            self.customer.save()

    def save(self, *args, **kwargs):
        super().save(*args, **kwargs)
        if self.complete:
            self.bonusCount()

class OrderItem(models.Model):
    product = models.ForeignKey(FurnitureProduct, on_delete=models.SET_NULL,null=True)
    order = models.ForeignKey(Order, on_delete=models.SET_NULL, null=True)
    quantity = models.IntegerField(default=0, null=True, blank=True)
    date_added = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f'OrderID    : {self.order_id} - {self.product.name}'

    @property
```

```
    def get_total(self):
        if self.product.discount_price:
            total = self.product.discount_price * self.quantity
        else:
            total = self.product.price * self.quantity
        return total

class ShippingAddres(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.SET_NULL ,null=True)
    order = models.ForeignKey(Order, on_delete=models.SET_NULL, null=True)
    address = models.CharField(max_length=200,null=False)
    city = models.CharField(max_length=200,null=False)
    state = models.CharField(max_length=200,null=False)
    zipcode = models.CharField(max_length=200, null=False)
    date_added = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.address
```

furniturestore.html

```
{% extends 'main/base.html' %}
{% load static %}
{% block bodycontent %}
<div class="container">
    <div class="row">
        <div class="col-lg-1 col-md-2 col-sm-3 order-lg-first">
            <div class="category-sidebar">
                <h3>Ціна</h3>
                <form class="price-filter-form" method="get" action="{% url 'furniture_store' %}"
onsubmit="return validatePriceFilter()">
                    <div>
                        <input type="text" id="min_price" name="min_price">
                        - <input type="text" id="max_price" name="max_price">
                    </div>
                    <div id="priceSlider"></div>
                    <button type="submit" id="filterButton" disabled>OK</button>
                    {% if price_filter_form.errors %}
                        <p class="error">{{ price_filter_form.errors.min_price }}</p>
                        <p class="error">{{ price_filter_form.errors.max_price }}</p>
                    {% endif %}
                </form>
                <h3>Категорії</h3>
                <ul class="category-menu">
                    <li {% if not current_category_slug %}class="active"{% endif %}>
                        <a href="{% url 'furniture_store' %}">Все</a>
                    </li>
                    {% for category in categories %}
                        <li {% if category.slug == current_category_slug %}class="active"{% endif %}>
                            <a href="{% url 'furniture_store_category' category.slug %}">{{ category.name }}</a>
                        </li>
```

```
            {% endfor %}
          </ul>
        </div>
      </div>
      <div class="col-lg-10 col-md-10 col-sm-9 order-lg-last product-grid" style="margin-left: 50px">
        <div class="row">
          {% for product in products %}
            <div class="col-lg-4">
              <img class="thumbnail" src="{{ product.imageURL }}">
              <div class="box-element product">
                <h6><strong>{{ product.name }}</strong></h6>
                <hr>
                {% if not product.in_stock %}
                  <p style="display: inline-block; float: left; color: red; margin: 0px 10px 0px 0px;">Out of Stock :(</p>
                {%else%}
                  <button data-product={{ product.id }} data-action="add" class="btn btn-outline-secondary add-btn update-cart">Add to Cart</button>
                {% endif %}
                <a class="btn btn-outline-success" href="{% url 'product_detail' product.slug %}">View</a>
                 {% if product.discount_price %}
                  <span style="display: inline-block; float: right">
                    <del>${{ product.price|floatformat:0 }}</del>
                  </span>

                  <h4 style="display: inline-block; float: right; color: red;"><strong>${{ product.discount_price|floatformat:0 }}</strong></h4>
                {% else %}
                <h4 style="display: inline-block; float: right"><strong>${{ product.price|floatformat:0 }}</strong></h4>
                {% endif %}
              </div>
            </div>
          {% endfor %}
        </div>
        <div class="pagination-wrapper" >
          <div class="pagination">
            {% if products.has_previous %}
              <a href="?page={{ products.previous_page_number }}">« Попередня сторінка</a>
            {% endif %}
            {% for num in products.paginator.page_range %}
              {% if products.number == num %}
                <span class="current-page">{{ num }}</span>
              {% else %}
                <a href="?page={{ num }}">{{ num }}</a>
              {% endif %}
            {% endfor %}
            {% if products.has_next %}
              <a href="?page={{ products.next_page_number }}">Наступна сторінка »</a>
            {% endif %}
```

```
            </div>
          </div>
        </div>
      </div>
</div>
<script                    src="https://cdnjs.cloudflare.com/ajax/libs/noUiSlider/15.7.1/nouislider.min.js"
integrity="sha512-
UOJe4paV6hYWBnS0c9GnIRH8PLm2nFK22uhfAvsTIqd3uwnWsVri1OPn5fJYdLtGY3wB11LGHJ4y
PU1WFJeBYQ==" crossorigin="anonymous" referrerpolicy="no-referrer"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/js-cookie/2.2.1/js.cookie.min.js"></script>
<link    rel="stylesheet"    href="https://cdnjs.cloudflare.com/ajax/libs/noUiSlider/15.7.1/nouislider.css"
integrity="sha512-
MKxcSu/LDtbIYHBNAWUQwfB3iVoG9xeMCm32QV5hZ/9lFaQZJVaXfz9aFa0IZExWzCpm7OWvp9
zq9gVip/nLMg==" crossorigin="anonymous" referrerpolicy="no-referrer" />
<script>
  document.addEventListener('DOMContentLoaded', function () {
  var priceSlider = document.getElementById('priceSlider');
  var minPriceInput = document.getElementById('min_price');
  var maxPriceInput = document.getElementById('max_price');
  var filterButton = document.getElementById('filterButton');
  var priceFilterForm = document.querySelector('.price-filter-form');

  // Получаем значения из куков
  var initialMinPrice = parseFloat(getCookie('min_price')) || {{ min_price }};
  var initialMaxPrice = parseFloat(getCookie('max_price')) || {{ max_price }};

  noUiSlider.create(priceSlider, {
    start: [initialMinPrice, initialMaxPrice],
    connect: true,
    range: {
      'min': {{ min_price }},
      'max': {{max_price}},
    },
  });

  // Event handler для обновления слайдера
  priceSlider.noUiSlider.on('update', function (values, handle) {
    var value = values[handle];

    if (handle === 0) {
      minPriceInput.value = value;
    } else {
      maxPriceInput.value = value;
    }

    // Enable кнопку "OK" после изменения значений
    filterButton.disabled = false;
  });

  // Event handler для отправки формы
  priceFilterForm.addEventListener('submit', function (event) {
```

```
        event.preventDefault();

        // Обновляем значения в куках
        setCookie('min_price', minPriceInput.value);
        setCookie('max_price', maxPriceInput.value);

        this.submit();
    });

    // Обновляем значения при загрузке страницы
    minPriceInput.value = initialMinPrice;
    maxPriceInput.value = initialMaxPrice;

    // Функция для установки значения в куки
    function setCookie(name, value) {
        document.cookie = name + '=' + value + '; path=/';
    }

    // Функция для получения значения из куков
    function getCookie(name) {
        var cookies = document.cookie.split(';');
        for (var i = 0; i < cookies.length; i++) {
            var cookie = cookies[i].trim();
            if (cookie.indexOf(name + '=') === 0) {
                return cookie.substring(name.length + 1);
            }
        }
        return null;
    }
});
</script>
{% endblock %}
```

product_detail.html

```
{% extends 'main/base.html' %}
{% load static %}
{% block bodycontent %}
<script src="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.8.1/slick.min.js" integrity="sha512-XtmMtDEcNz2j7ekrtHvOVR4iwwaD6o/FUJe6+Zq+HgcCsk3kj4uSQQR8weQ2QVj1o0Pk6PwYLohm206ZzNfubg==" crossorigin="anonymous" referrerpolicy="no-referrer"></script>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.8.1/slick-theme.min.css" integrity="sha512-17EgCFERpgZKcm0j0fEq1YCJuyAWdz9KUtv1EjVuaOz8pDnh/0nZxmU6BBXwaaxqoi9PQXnRWqlcDB027hgv9A==" crossorigin="anonymous" referrerpolicy="no-referrer" />
<div class="container">
    <div class="row">
        <div class="col-lg-6">
            <div class="box-element product-detail">
                <div class="slider-for">
```

```
            <img class="product-image" src="{{ product.imageURL }}" data-model-url="{{ product.modelURL|escapejs }}">
            {% for image in product.additional_images.all %}
              <img class="product-image" src="{{ image.image.url }}">
            {% endfor %}
          </div>
          <p></p>
          <div class="slider-nav">
            <img class="product-image" src="{{ product.imageURL }}">
            {% for image in product.additional_images.all %}
              <img class="product-image" src="{{ image.image.url }}">
            {% endfor %}
          </div>
        </div>
      </div>
      <div class="col-lg-6">
        <div class="box-element product-info">
          <h3>{{ product.name }}</h3>
          <p>{{ product.description|linebreaks }}</p>
          <div style="clear:both;">
            <h2 style="display: inline-block; float: left; margin-right: 10px;">Price:</h2>
            {% if product.discount_price %}
            <h4 style="display: inline-block; float: left; color: red; margin-top:4px; font-size:28px"><strong>${{ product.discount_price|floatformat:0 }}</strong></h4>
            <span style="display: inline-block; float: left; margin-right: 10px;">
              <del>${{ product.price|floatformat:0 }}</del>
            </span>
            {% else %}
              <h4 style="display: inline-block; float: left"><strong>${{ product.price|floatformat:0 }}</strong></h4>
            {% endif %}
          </div>
          <div style="clear:both;">
            <a class="btn btn-outline-dark" href="{% url 'furniture_store' %}">&#x2190; Back to store</a>
            {% if not product.in_stock %}
              <p style="display: inline-block; float: left; color: red; margin:5px 15px 0px 0px;">Out of Stock :(</p>
            {% else %}
              <button data-product="{{ product.id }}" data-action="add" class="btn btn-outline-secondary add-btn update-cart">Add to Cart</button>
            {% endif %}
            {% if product.modelURL %}
              <button id="view-3d-button" class="btn btn-outline-primary" data-3d-url="{% url '3d_view' product.slug %}?3d_url={{ product.modelURL }}&view_3d=true">View in 3D</button>
            {% endif %}
          </div>
        </div>
      </div>
    </div>
</div>
```

```
<script type="text/javascript">
   $(document).ready(function(){
      $('.slider-for').slick({
         slidesToShow: 1,
         slidesToScroll: 1,
         arrows: true,
         fade: true,
         asNavFor: '.slider-nav'
      });
      $('.slider-nav').slick({
         slidesToShow: 3,
         slidesToScroll: 1,
         asNavFor: '.slider-for',
         dots: false,
         centerMode: true,
         focusOnSelect: true
      });
   });
</script>

<script type="importmap">
      {
         "imports": {
            "three": "https://unpkg.com/three@0.139.0/build/three.module.js",
            "OrbitControls": "https://unpkg.com/three@0.139.0/examples/jsm/controls/OrbitControls.js",
            "GLTFLoader": "https://unpkg.com/three@0.139.0/examples/jsm/loaders/GLTFLoader.js",
            "OBJLoader" : "https://unpkg.com/three@0.139.0/examples/jsm/loaders/OBJLoader.js",
            "MTLLoader": "https://unpkg.com/three@0.139.0/examples/jsm/loaders/MTLLoader.js",
            "RectAreaLightHelper":
"https://unpkg.com/three@0.139.0/examples/jsm/helpers/RectAreaLightHelper.js",
            "RectAreaLightUniformsLib":
"https://unpkg.com/three@0.139.0/examples/jsm/lights/RectAreaLightUniformsLib.js"
         }
      }
</script>
<script>
   document.addEventListener("DOMContentLoaded", function () {
      var modelURLElement = document.querySelector('#view-3d-button');
      if(modelURLElement){
         var model3DUrl = modelURLElement.getAttribute("data-3d-url");
         modelURLElement.addEventListener("click", function () {
            // Navigate to the 3D model view page using model3DUrl
            window.location.href = model3DUrl;
         });
      }
   });
</script>
<script src="{% static 'main/js/3d.js' %}" type="module"></script>
{% endblock %}
```

3d_view.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>TamTumba</title>
   <link rel="icon" href="{% static 'main/img/icon.ico' %}" type="image/png">
</head>
<body>
   <div class="container"></div>
   <div id="model-container"></div>
   <script type="importmap">
      {
         "imports": {
            "three": "https://unpkg.com/three@0.139.0/build/three.module.js",
            "OrbitControls": "https://unpkg.com/three@0.139.0/examples/jsm/controls/OrbitControls.js",
            "GLTFLoader": "https://unpkg.com/three@0.139.0/examples/jsm/loaders/GLTFLoader.js",
            "OBJLoader" : "https://unpkg.com/three@0.139.0/examples/jsm/loaders/OBJLoader.js",
            "MTLLoader": "https://unpkg.com/three@0.139.0/examples/jsm/loaders/MTLLoader.js",
            "RectAreaLightHelper":
"https://unpkg.com/three@0.139.0/examples/jsm/helpers/RectAreaLightHelper.js",
            "RectAreaLightUniformsLib":
"https://unpkg.com/three@0.139.0/examples/jsm/lights/RectAreaLightUniformsLib.js"
         }
      }
   </script>
   <script src="{% static 'main/js/OBJLoader.js' %}"></script>
   <script src="{% static 'main/js/3d.js' %}" type="module"></script>
</body>
</html>
```

3d.js

```
import * as THREE from 'three';
import { OrbitControls } from 'OrbitControls';
import { OBJLoader } from 'OBJLoader';
import { MTLLoader } from 'MTLLoader';
import { RectAreaLightHelper } from 'RectAreaLightHelper';
import { RectAreaLightUniformsLib } from 'RectAreaLightUniformsLib';

const urlParams = new URLSearchParams(window.location.search);
const view3d_url = urlParams.get('3d_url');
const view3D = urlParams.get('view_3d');
let isRotating = false;
let container = document.querySelector('.container');

function init() {
   // Scene
   const scene = new THREE.Scene()
   scene.background = new THREE.Color("#E2DFE1");
```

```
    // Camera
    const camera = new THREE.PerspectiveCamera(45, window.innerWidth / window.innerHeight, 0.1,
3000);
    camera.position.set(0, 0.5, 1.5);
    camera.lookAt(0, 0, 0);

    // Renderer
    const renderer = new THREE.WebGLRenderer({ antialias: true })
    renderer.setSize(window.innerWidth, window.innerHeight)
    container.appendChild(renderer.domElement)

    let plain;
    {
        plain = new THREE.Mesh(
            new THREE.PlaneGeometry(1000, 1000),
            new THREE.MeshBasicMaterial({ color: "#E2DFE1" })
        )
        plain.receiveShadow = true;
        plain.position.set(0, -1, 0)
        plain.rotateX(-Math.PI / 2);
        scene.add(plain)
    }

    // Load your object using OBJLoader (assuming it's an OBJ file)
    let currentRotatingObject;

    function loadAndAddObject(url) {
        const loader = new OBJLoader();
        loader.load(url, obj => {
            // Remove previous rotating object if it exists
            if (currentRotatingObject) {
                scene.remove(currentRotatingObject);
            }
            currentRotatingObject = obj;
            scene.add(obj);
        },
        function (xhr) {
            console.log((xhr.loaded / xhr.total * 100) + '% loaded');
        },
        function (error) {
            console.log('Error ->' + error)
        });
    }

    // Model
    loadAndAddObject(view3d_url);

    // Lights
    const light1 = new THREE.DirectionalLight(0xffffff, 1)
    light1.position.set(-2, 0, 10)
```

```
light1.lookAt(0, -1, 0)
scene.add(light1)

const light2 = new THREE.DirectionalLight(0xffffff, 1)
light2.position.set(2, 0, 5)
light2.lookAt(0, 1, 0)
scene.add(light2)

// OrbitControls
const controls = new OrbitControls(camera, renderer.domElement);
controls.autoRotate = false;
controls.autoRotateSpeed = 5;
controls.enableDamping = true;

// Resize
window.addEventListener('resize', onWindowResize, false)

function onWindowResize() {
   camera.aspect = window.innerWidth / window.innerHeight;
   camera.updateProjectionMatrix();

   renderer.setSize(window.innerWidth, window.innerHeight)
}

// Keyboard controls for object rotation and camera movement
document.addEventListener('keydown', (event) => {
   switch (event.key) {
      case 'a': // Toggle auto-rotation on 'A' key
      case 'ф': // Toggle auto-rotation on 'A' key (Cyrillic)
         isRotating = !isRotating;
         break;
      case 'ArrowUp': // Move camera forward
         camera.position.y += 0.1;
         break;
      case 'ArrowDown': // Move camera backward
         camera.position.y -= 0.1;
         break;
      case 'ArrowLeft': // Move camera left
         camera.position.x -= 0.1;
         break;
      case 'ArrowRight': // Move camera right
         camera.position.x += 0.1;
         break;
   }
});

// Animate
function animate() {
   requestAnimationFrame(animate)
   controls.update();
   if (isRotating && currentRotatingObject) {
```

```
            currentRotatingObject.rotation.y += 0.01;
        }
        renderer.render(scene, camera)
    }
    animate();
}

if (view3D == 'true') {
    init();
}
```

cart.js

```
var updateBtns = document.getElementsByClassName('update-cart')


for(var i=0;i<updateBtns.length;i++){
    updateBtns[i].addEventListener('click',function(){
        var productId = this.dataset.product
        var action = this.dataset.action
        console.log('productId:',productId,'Action:',action)
        console.log('USER',user)

        if(user=='AnonymousUser'){
            addCookieItem(productId,action)
        }else{
            updateUserOrder(productId,action)
        }
    })
}


function addCookieItem(productId,action){
    console.log('User is not authenticated...')
    if(action == 'add'){
        if (cart[productId] == undefined){
            cart[productId] = {'quantity':1}
        }else{
            cart[productId]['quantity'] += 1
        }
    }
    if(action == 'remove'){
        cart[productId]['quantity'] -= 1
        if(cart[productId]['quantity']<=0){
            delete cart[productId]
        }
    }
    console.log('Cart:',cart)
    document.cookie = 'cart=' + JSON.stringify(cart) + ";domain=;path=/"
    location.reload()
}
```

```
function updateUserOrder(productId, action) {
    console.log('User is authenticated, sending data... ');

    var url = '/cart/update_item/';

    fetch(url, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-CSRFToken': csrftoken,
        },
        body: JSON.stringify({'productId': productId, 'action': action})
    })
    .then((response) => {

        return response.json();
    })
    .then((data) => {
        location.reload()
    });
}
```

cart.html

```
{% extends 'main/base.html' %}
{% load static %}
{% block bodycontent %}
        <div class="container">
                <div class="row">
                        <div class="col-lg-12">
                                <div class="box-element">
                                        <a class="btn btn-outline-dark" href="{% url 'furniture_store' %}">&#x2190; Continue Shopping</a>
                                        <br>
                                        <br>
                                        <table class="table">
                                                <tr>
                                                        <th><h5>Items: <strong>{{order.get_cart_items}}</strong></h5></th>
                                                        <th><h5>Total: <strong>{{order.get_cart_total|floatformat:2}} $</strong></h5></th>
                                                        <th>
                                                                {% if order.get_cart_items == 0 %}
                                                                        <p style="color:red;">Your cart is empty.
                                                                                Please add items before proceeding to checkout.</p>
                                                                {% else %}
```

```
                                                    <a style="float:right; margin:5px;"
class="btn btn-success" href="{% url 'checkout' %}">
                                                        Checkout</a>
                                                {% endif %}
                                            </th>
                                        </tr>
                                    </table>
                                </div>

                                <br>
                                <div class="box-element">
                                    <div class="cart-row">
                                        <div style="flex:2"></div>
                                        <div style="flex:2"><strong>Item</strong></div>
                                        <div style="flex:1"><strong>Price</strong></div>
                                        <div style="flex:1"><strong>Quantity</strong></div>
                                        <div style="flex:1"><strong>Total</strong></div>
                                    </div>
                                    {% for item in items %}
                                    <div class="cart-row">
                                        <div style="flex:2"><img class="row-image"
src="{{item.product.imageURL}}"></div>
                                        <div style="flex:2"><p>{{item.product.name}}</p></div>
                                        {% if item.product.discount_price %}
                                            <div
style="flex:1"><p>{{item.product.discount_price|floatformat:2}}</p></div>
                                        {% else %}
                                            <div
style="flex:1"><p>{{item.product.price|floatformat:2}}</p></div>
                                        {% endif %}
                                        <div style="flex:1">
                                            <p class="quantity">{{item.quantity}}</p>
                                            <div class="quantity">
                                                <img data-product={{item.product.id}} data-
action="add" class="chg-quantity update-cart"
                                                    src="{% static 'main/img/arrow-
up.png' %}">

                                                <img data-product={{item.product.id}} data-
action="remove" class="chg-quantity update-cart"
                                                    src="{% static 'main/img/arrow-
down.png' %}">
                                            </div>
                                        </div>
                                        <div style="flex:1"><p>{{item.get_total}}</p></div>
                                    </div>
                                    {% endfor %}
                                </div>
                            </div>
                        </div>
                    </div>
```

```
{% endblock %}
```

checkout.html

```
{% extends 'main/base.html' %}
{% load static %}
{% block bodycontent %}
<div class="container">
	<div class="row">
		<div class="col-lg-6">
			<div class="box-element" id="form-wrapper">
				<form id="form">
					<div id="user-info">
						<div class="form-field">
							<input required class="form-control" type="text" name="name" placeholder="Name..">
						</div>
						<div class="form-field">
							<input required class="form-control" type="text" name="surname" placeholder="Surname..">
						</div>
						<div class="form-field">
							<input required class="form-control" type="email" name="email" placeholder="Email..">
						</div>
					</div>

					<div id="shipping-info">
						<hr>
						<p>Shipping Information:</p>
						<hr>
						<div class="form-field">
							<input required class="form-control" type="text" name="address" placeholder="Address..">
						</div>
						<div class="form-field">
							<input required class="form-control" type="text" name="city" placeholder="City..">
						</div>
						<div class="form-field">
							<input required class="form-control" type="text" name="state" placeholder="State..">
						</div>
						<div class="form-field">
							<input required class="form-control" type="text" name="zipcode" placeholder="Zip code..">
						</div>

					</div>

					<hr>
```

```
                                        <input id="form-button" class="btn btn-success btn-block" type="submit" value="Continue">
                                    </form>
                                </div>

                                <br>
                                <div class="box-element hidden" id="payment-info">
                                    <small>Paypal Options</small>
                                    <button id="make-payment">Make payment</button>
                                </div>

                            </div>

                            <div class="col-lg-6">
                                <div class="box-element">
                                    <a  class="btn btn-outline-dark" href="{% url 'cart' %}">&#x2190; Back to Cart</a>
                                    <hr>
                                    <h3>Order Summary</h3>
                                    <hr>
                                    {% for item in items %}
                                    <div class="cart-row">
                                        <div style="flex:2"><img class="row-image" src="{{item.product.imageURL}}"></div>
                                        <div style="flex:2"><p>{{item.product.name}}</p></div>
                                        {% if item.product.discount_price %}
                                            <div style="flex:1"><p>{{item.product.discount_price|floatformat:2}}</p></div>
                                        {% else %}
                                            <div style="flex:1"><p>{{item.product.price|floatformat:2}}</p></div>
                                        {% endif %}
                                        <div style="flex:1"><p>{{item.quantity}}</p></div>
                                    </div>
                                    {% endfor %}
                                    <h5>Items: {{order.get_cart_items}}</h5>
                                    <h5>Total: {{order.get_cart_total|floatformat:2}} $</h5>
                                </div>
                            </div>
                        </div>
                    </div>

                    <script type="text/javascript">

                        var total = '{{order.get_cart_total}}'
                        if(user != 'AnonymousUser'){
                            document.getElementById('user-info').innerHTML =''
                        }
                    <!--    if(user != 'AnonymousUser'){-->
                    <!--        document.getElementById('payment-info').classList.remove('hidden')-->
                    <!--    }-->
```

```
var form = document.getElementById('form')
form.addEventListener('submit', function(e){
	e.preventDefault()
	console.log('Form submitted')
	document.getElementById('form-button').classList.add('hidden')
	document.getElementById('payment-info').classList.remove('hidden')
})

document.getElementById('make-payment').addEventListener('click', function(e){
	submitFormData()
})

function submitFormData(){
	console.log('Payment button clicked')

	var userFormData ={
		'name': null,
		'surname': null,
		'email': null,
		'total': total,
	}
	var shippingInfo ={
		'address': null,
		'city': null,
		'state': null,
		'zipcode': null,
		'total': total,
	}
	shippingInfo.address = form.address.value
	shippingInfo.city = form.city.value
	shippingInfo.state = form.state.value
	shippingInfo.zipcode = form.zipcode.value

	if(user == 'AnonymousUser'){
		userFormData.name = form.name.value
		userFormData.surname = form.surname.value
		userFormData.email = form.email.value
	}

	var url = 'processOrder/'
	fetch(url, {
		method: 'POST',
		headers: {
			'Content-Type': 'application/json',
			'X-CSRFToken': csrftoken,
		},
		body: JSON.stringify({'form': userFormData, 'shipping': shippingInfo})
	})
	.then((response) => response.json())
```

```
                .then((data) => {
                        console.log('Success:', data)
                        alert('Transaction completed')

                        cart = {}
                        document.cookie = 'cart=' + JSON.stringify(cart) + ";domain=;path=/"
                        window.location.href = "{% url 'furniture_store' %}"
                })

        }
</script>
{% endblock %}
```

order_confirmation.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Order Confirmation</title>
</head>
<body>
   <p>Dear {{ order.customer.name }},</p>
   <p>Thank you for your order!</p>
   <p>Here are the details of your order:</p>
   <ul>
      {% for item in order.orderitem_set.all %}
         <strong>Product:</strong> {{ item.product.name }}<br>
         <strong>Price:</strong> {{ item.product.price }}<br>
         <strong>Quantity:</strong> {{ item.quantity }}<br>
         <strong>Subtotal:</strong> {{ item.get_total }}<br>
      {% endfor %}
   </ul>
   <p>Total items: {{ order.get_cart_items }}</p>
   <p>Total amount: {{ order.get_cart_total }}</p>
   <p>Order date: {{ order.date_ordered }}</p>
   <p>Regards,</p>
   <p>The TamTumba Team</p>
</body>
</html>
```