МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота бакалавра

ІНТЕРНЕТ-МАГАЗИН З РЕАЛІЗАЦІЇ КУЛЬТУРНИХ РОСЛИН

Здобувач освіти
студент гр. ІН – 81 Олег АНІЩЕНКО

Науковий керівник Артем КОРОБОВ

Завідувач кафедри
доктор технічних наук, професор Анатолій ДОВБИШ

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2022 р.

ЗАВДАННЯ
до кваліфікаційної роботи

здобувача вищої освіти четвертого курсу, групи ІН-81 спеціальності «122 – Комп'ютерні науки» денної форми навчання Аніщенка Олега Олексійовича.

Тема: «ІНТЕРНЕТ-МАГАЗИН З РЕАЛІЗАЦІЇ КУЛЬТУРНИХ РОСЛИН»

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки: 1) аналіз предметної області; 2) вибір методів рішення задачі; 3) практична реалізація.

Дата видачі завдання «_____»_____ 2022 р.
Керівник роботи _____ Артем КОРОБОВ
Завдання прийняв до виконання _____ Олег АНІЩЕНКО

РЕФЕРАТ

Записка: 82 стор., 39 рис., 6 додатків, 13 джерел.

Об’єкт дослідження — сучасні методи розробки веб-ресурсів

Мета роботи — розробка інтернет-магазину з реалізації культурних.

Методи дослідження — методи спостереження, порівняння, аналізу.

Результати — розроблено веб-сайт для реалізації рослин, з дотримуванням сучасних тенденцій розробки веб-ресурсів. Сайт створений на базі мови Python з використання фреймворку Django.

ВЕБ-САЙТ, WEB, PYTHON, DJANGO, SESSIONS, POSTGRESQL, BOOTSTRAP, JAVASCRIPT

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

HTTP – HyperText Transfer Protocol – протокол передачі даних

TCP – Transmission Control Protocol – протокол керування передаванням

TLS – Transport Layer Security – протокол захисту транспортного рівня

HTML – HyperText Markup Language – мова гіпертекстової розмітки

CSS – Cascading Style Sheets – каскадні таблиці стилів

СУБД – Система Управління Базами Даних

БД – База Даних

IDE – Integrated Development Environment – інтегроване середовище розробки

SSH – Secure Shell – безпечна оболонка

UML – Unified Modeling Language – уніфікована мова моделювання

ERD – Entity Relation Diagram – діаграма сутностей та зв’язків

API – Application Programming Interface – програмний інтерфейс додатку

ВСТУП

З кожним останнім роком нас спіткають все більше проблем, які обмежують або й зовсім унеможливлюють фізичний контакт між людьми.

На мою думку, програмування існує для того, щоб нівелювати можливі проблеми та складнощі, які виникають в нашому житті та допомогти використовувати свій час та свої ресурси якнайбільш ефективно.

Ведення бізнесу зараз все частіше відбувається онлайн бо таким чином можна розширити географію своїх покупців, зекономити свій час та ресурси, які б при веденні фізичного бізнесу витрачалися на оренду, обслуговування та інше, набагато простіше та ефективніше вести просування свого бренду або товару, а також відкривати бізнес з невеликим стартовим капіталом. Ведення бізнесу таким, яким він був ще 10 років тому вже неможливо, або це буде занадто нераціональним використанням своїх ресурсів. Маючи так багато переваг, онлайн бізнес приваблює все більше і більше людей, охочий розпочати своє діло.

Більше того, реалії сьогодення такі, що все більше людей цікавляться тим, щоб самостійно у себе на ділянці вирощувати свої, екологічно чисті та корисні овочі та фрукти.

Якщо поєднати ці два фактори, перелічені вище, стає зрозумілим актуальність розробки інтернет-магазину з продажу культурних рослин.

Для ідеального виконання даної задачі, необхідно чітко визначитися з цілями, зробити аналіз вже існуючих рішень, виявити їхні переваги та недоліки та намагатися створити своє ідеальне рішення даної проблеми.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Огляд основних відомостей

Web це глобальний інформаційний простір, заснований на фізичній інфраструктурі Інтернету і протоколі передачі даних HTTP. Дану мережу утворюють сервери та комп'ютери по всьому світу, поєднані між собою [1]. Технологія Веб працює за принципом взаємодії клієнту з сервером. Клієнтом, в даній ситуації виступає браузер, наприклад Google Chrome, а сервером – певний сервер, наприклад Apache. Спілкування між клієнтською стороною та серверною відбувається за допомогою протоколу HTTP.

HTTP (Hyper Text Transfer Protocol) це набір певних правил для комунікації клієнта та сервера, а також для передачі інформації в мережі Інтернет. Дана комунікації відбувається за допомогою запитів. Основними запитами HTTP-клієнта є:

- GET, для отримання інформації;
- POST, для відправлення інформації;
- PUT, для заміни вже існуючої інформації;
- DELETE, для видалення з серверу.

В свою чергу, сервер надсилає клієнту HTTP відповіді, основними з яких є:

- 200: запит виконано успішно;
- 400: запит не був сформований належним чином;
- 404: вказаний в запиті ресурс не знайдений;
- 500: внутрішня помилка сервера;
- 502: служба недоступна. [2]

Наприклад, в нас є метод, при надсиланні запиту до якого, можна отримати деталі певного авто: GET : http://127.0.0.1:8000/api/v1/cars/car/detail/2/

У відповідь ми отримаємо код HTTP відповіді, та інформацію за даним автомобілем (рис. 1.1):

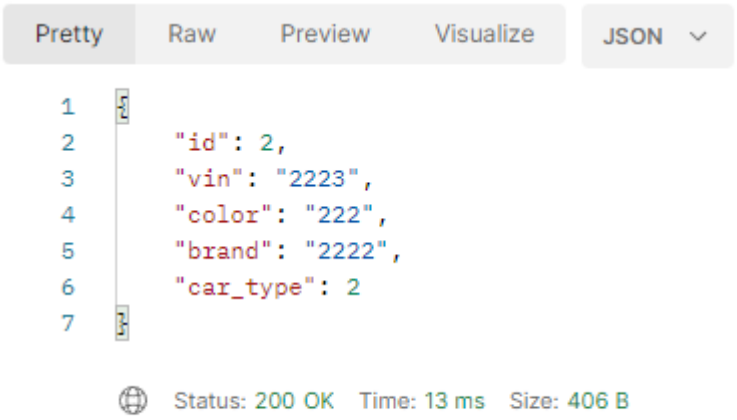

Рисунок 1.1 – Відповідь сервера на наш GET запит

Всі клієнтські запити, HTTP сервер приймає через відомий TCP-порт 80, або 443, якщо використовується HTTPS запит.

HTTPS – це безпечна версія протоколу HTTP, яка реалізує протокол HTTP за допомогою протоколу TLS для захисту базового під'єднання TCP. За винятком додаткової конфігурації, необхідної для налаштування TLS, використання HTTPS по суті не відрізняється від протоколу HTTP. [2]

Зараз, основна частина веб-сайтів використовує HTTPS, через його безпечність завдяки протоколу TLS, розробленим для забезпечення конфіденційності та безпеки даних для зв’язку через Інтернет.

1.2. Огляд існуючих рішень

Розглядаючи основні вимоги до інтернет-магазину можна виділити те, що він повинен мати інтуїтивно зрозумілий інтерфейс, був безпечним, вам основні функції для комфортного вибору необхідного продукту та оформлення замовлення. Нижче розглянемо та проаналізуємо деякі інтернет-магазини, які можуть бути використані для купівлі саджанців, насіння тощо.

1.2.1. Інтернет-магазин Green Market

Green Market це український онлайн-магазин, який дає змогу купувати товари для саду та городу. Спочатку розглянемо інтерфейс та дизайн сайту.

Домашня сторінка має привабливий вигляд, збалансовану кольорову гаму, зрозумілий інтерфейс. Є можливість використати бокове меню з категоріями для швидкого вибору необхідних товарів (рис. 1.2).

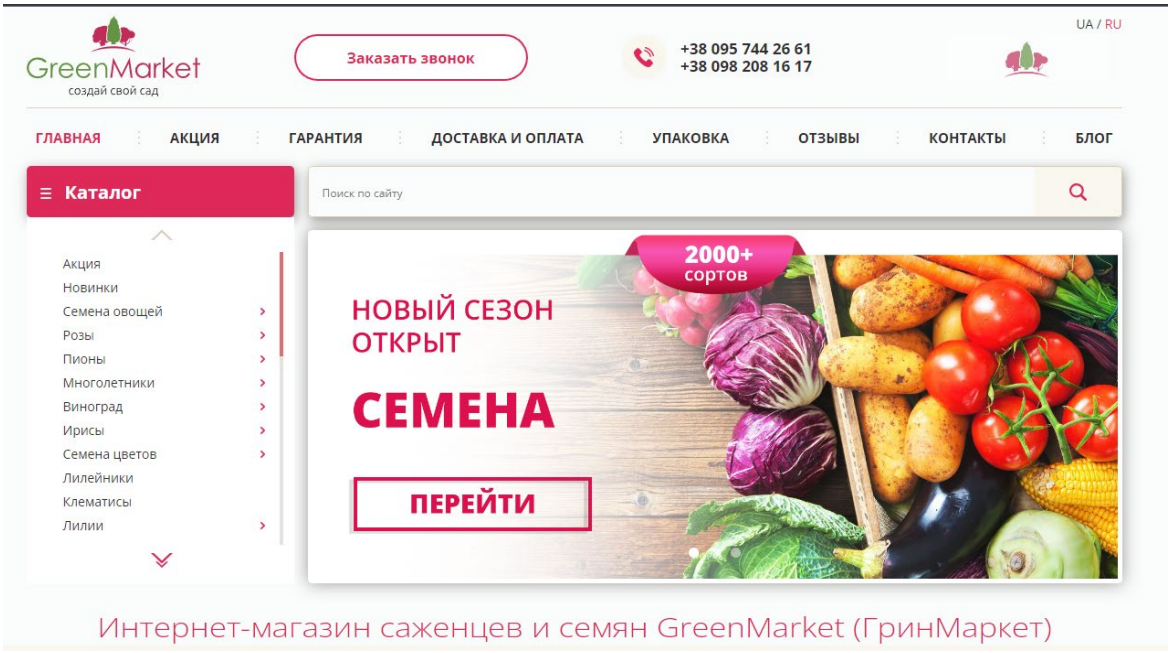

Рисунок 1.2 – Дизайн домашньої сторінки магазину Green Market

При перегляді каталогу товарів можна відсортувати товари за необхідними критеріями, продивитися короткий опис товару, при наведенні на нього мишкою, а також кнопку для додавання його в корзину (рис. 1.3).

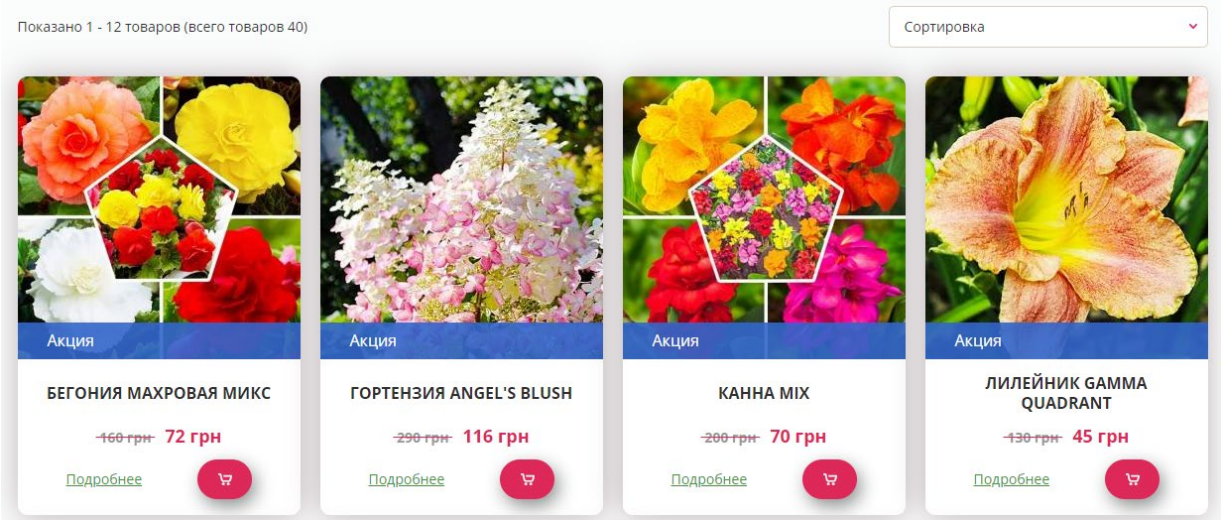

Рисунок 1.3 – Каталог товарів магазину Green Market

Проте, немає можливості одразу обрати необхідну кількість цього товару, що б прискорило процес покупки.

Також відсутня адаптивність на сайті, тому користування з мобільного пристрою ускладнене (рис. 1.4).

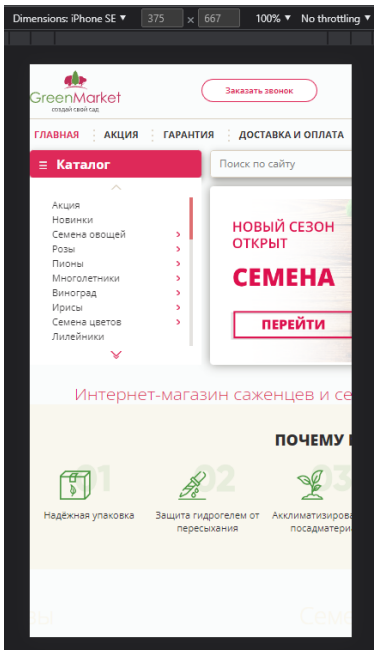

Рисунок 1.4 – Перегляд домашньої сторінки з телефону

Ще одним мінусом є швидкодія сайту. Для завантаження картинок товару, іконок корзини та інших візуальних елементів треба почекати щонайменше хвилину, а вже аж потім приступити до роботи з сайтом. Це відбувається практично на кожній сторінці, яка містить зображення чим дуже ускладнює процес купівлі необхідних товарів.

Кошик з товарами має наступний вигляд (рис. 1.5):

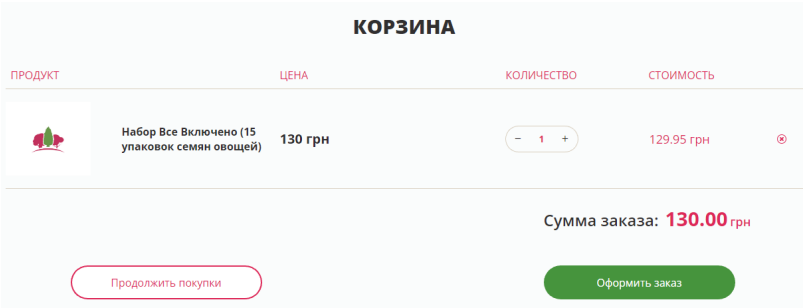

Рисунок 1.5 – Кошик магазину Green Market

Можемо помітити, що зображення не завантажилося, а також і необхідні скрипти для редагування товарів в кошику. Проте інтерфейс чіткий та зрозумілий.

Далі переходимо до сторінки з оформленням нашого замовлення, де можемо бачити просту форму для швидкого введення необхідних даних отримувача (рис. 1.6).

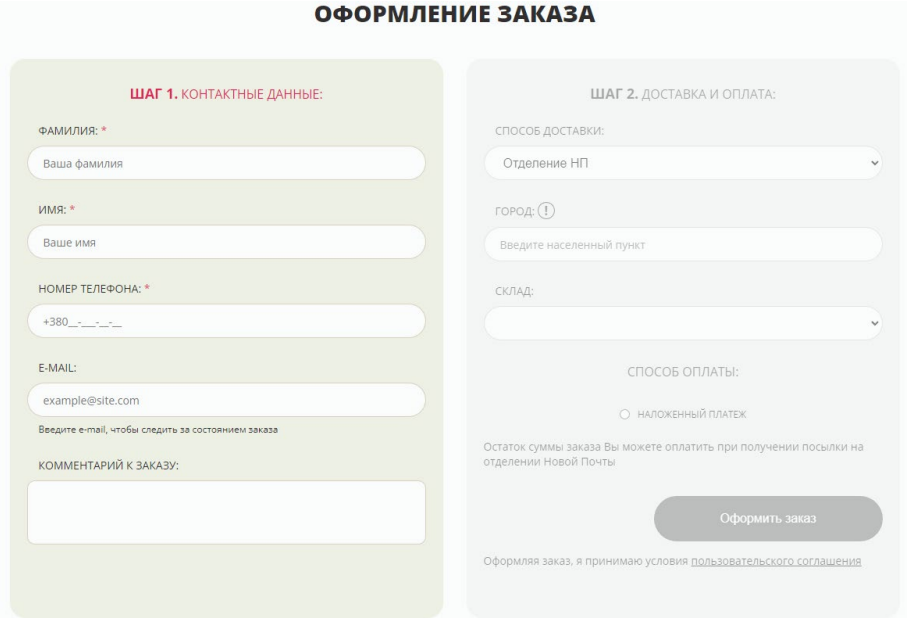

Рисунок 1.6 – Форма оформлення замовлення магазину Green Market

Підсумовуючи, даний сайт має гарний дизайн та зрозумілий інтерфейс, проте дуже погану швидкодію, відсутність реєстрації користувачів, відсутність можливості сплати онлайн і недостатню функціональність для більш комфортної роботи з магазином.

1.2.2. Інтернет-магазин Яскрава Клумба

Домашня сторінка має привабливий ненав'язливий дизайн. Тут можемо переглянути категорії товарів, виконати пошук необхідних товарів за допомогою відповідного поля, перейти в особистий кабінет, а також переглянути каталог товарів (рис. 1.7).

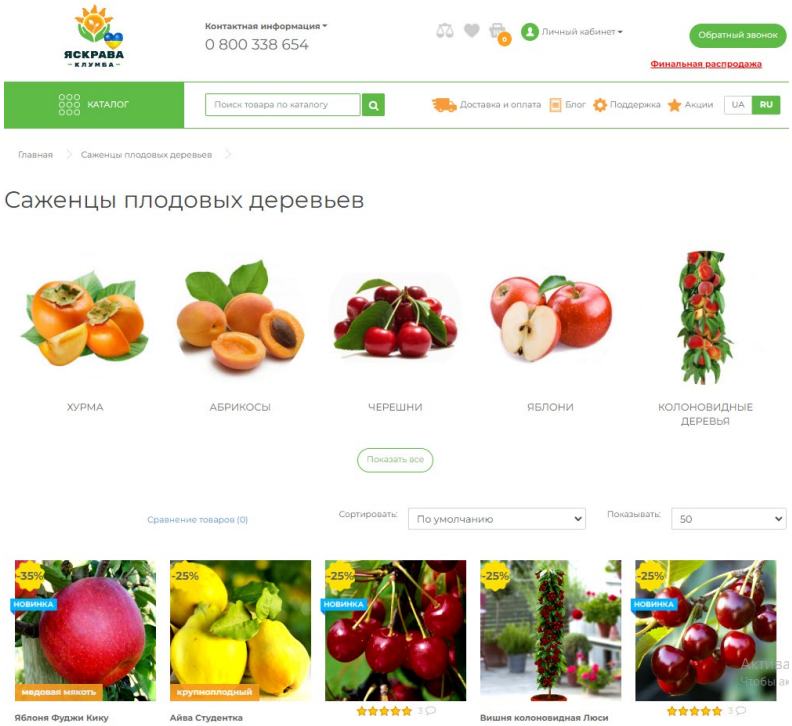

Рисунок 1.7 – Домашня сторінка магазину Яскрава Клумба

При перегляді товарів в каталозі, можна одразу додати його до кошика вказавши бажану кількість, а також переглянути особливості конкретної позиції (рис. 1.8).

Рисунок 1.8 – Вигляд товару в каталозі магазину Яскрава Клумба

При оформленні замовлення є можливість увійти у вже існуючий акаунт, або швидко його створити (рис. 1.9).

Після авторизації нас перенаправляє на сторінку з оформленням замовлення. Тут можна обрати способи оплати, серед яких є можливість сплатити онлайн за допомогою NovaPay, а також обрати спосіб доставки.

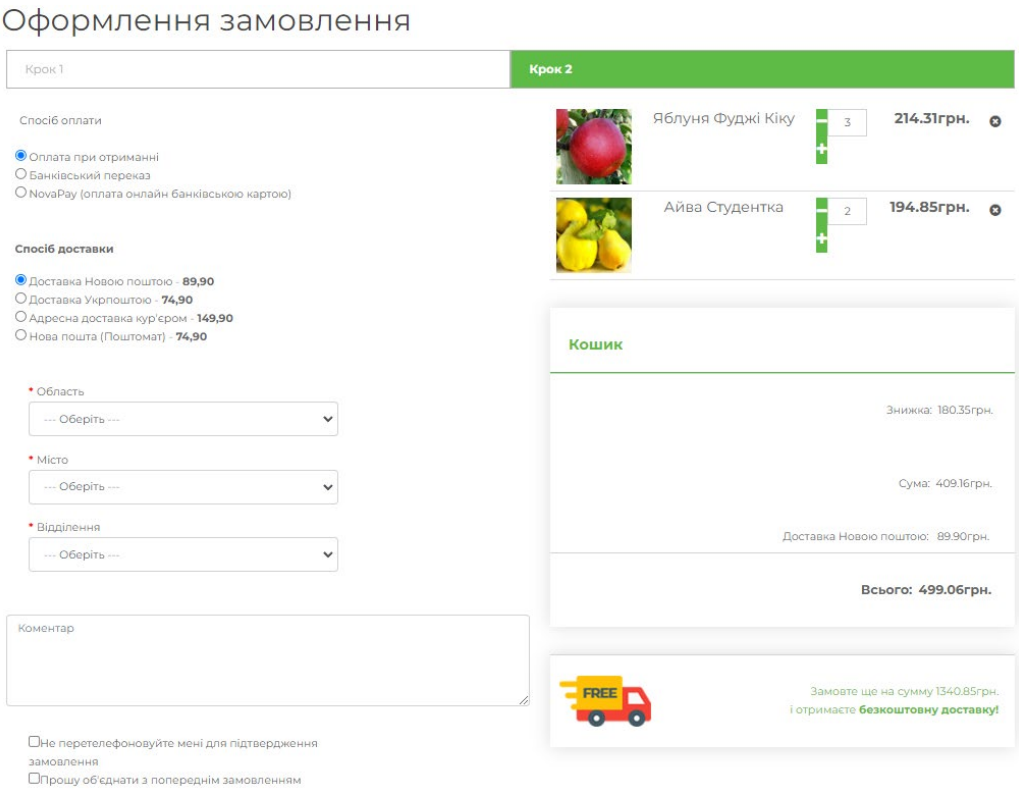

Рисунок 1.9 – Оформлення замовлення магазину Яскрава Клумба

Роблячи висновок, можна сказати, що даний сайт є дуже привабливим для покупця як з точки зору дизайну, так і з точки зору функціоналу. Має більшість необхідних функцій, серед яких можливість сортувати, фільтрувати та шукати товари, можливість реєструватися, платити онлайн, переглядати існуючі замовлення в особистому кабінеті, залишати відгуки та багато іншого. Явних недоліків не виявлено.

1.2.3. Інтернет-магазин Добродар

Домашня сторінка даного магазину має не надто привабливий інтерфейс, з великою кількістю різнокольорових банерів. Проте є меню з категоріями та поля для швидкого пошуку необхідного (рис. 1.10).

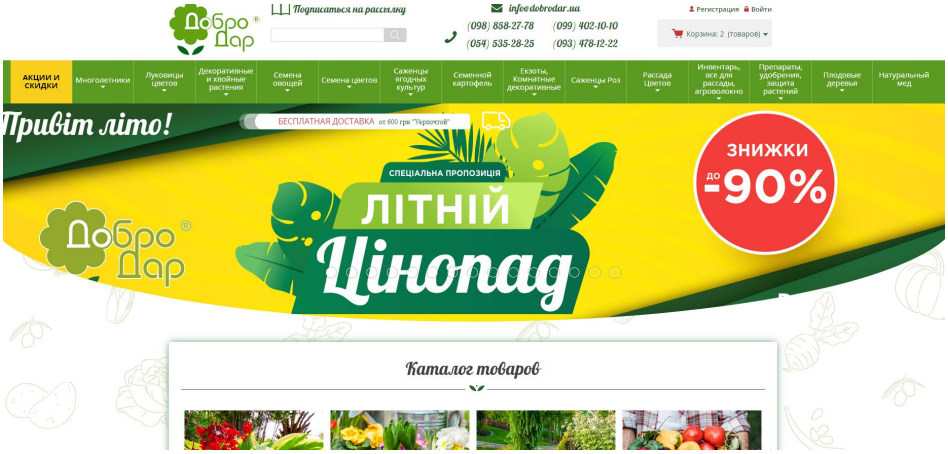

Рисунок 1.10 – Домашня сторінка магазину Добродар

Каталог з товарами, де можна побачити ціну, швидко додати до кошика вказавши бажану кількість, а при наведенні на товар відображається короткий опис (рис. 1.11).

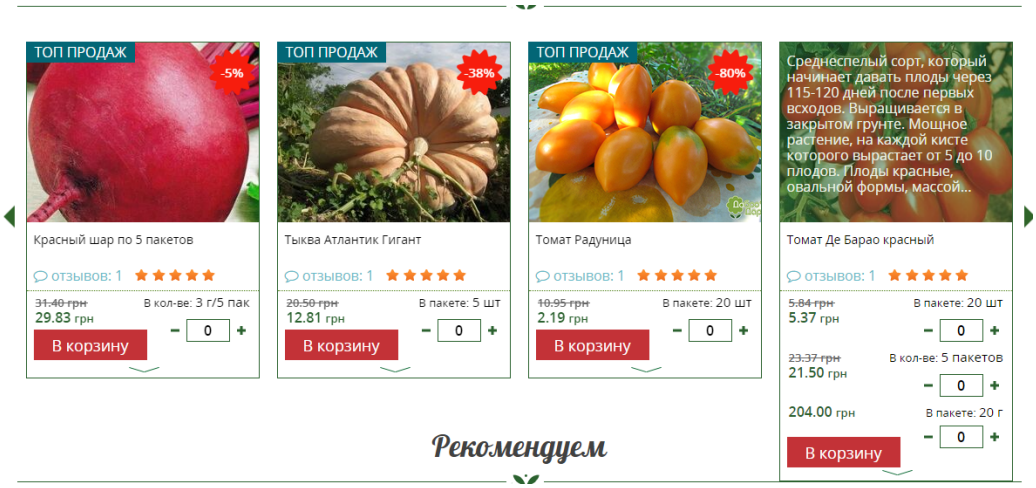

Рисунок 1.11 – Каталог товарів магазину Добродар

Кошик магазина має наступний вигляд та дає змогу обрати бажаний спосіб доставки, а також змінити кількість або й зовсім видалити певний товар (рис. 1.12).

Рисунок 1.12 – Кошик магазину Добродар

Далі маємо форму для вказання даних для отримання та вибору бажаного способу оплати, серед яких є можливість сплати онлайн за допомогою LiqPay (рис. 1.13).

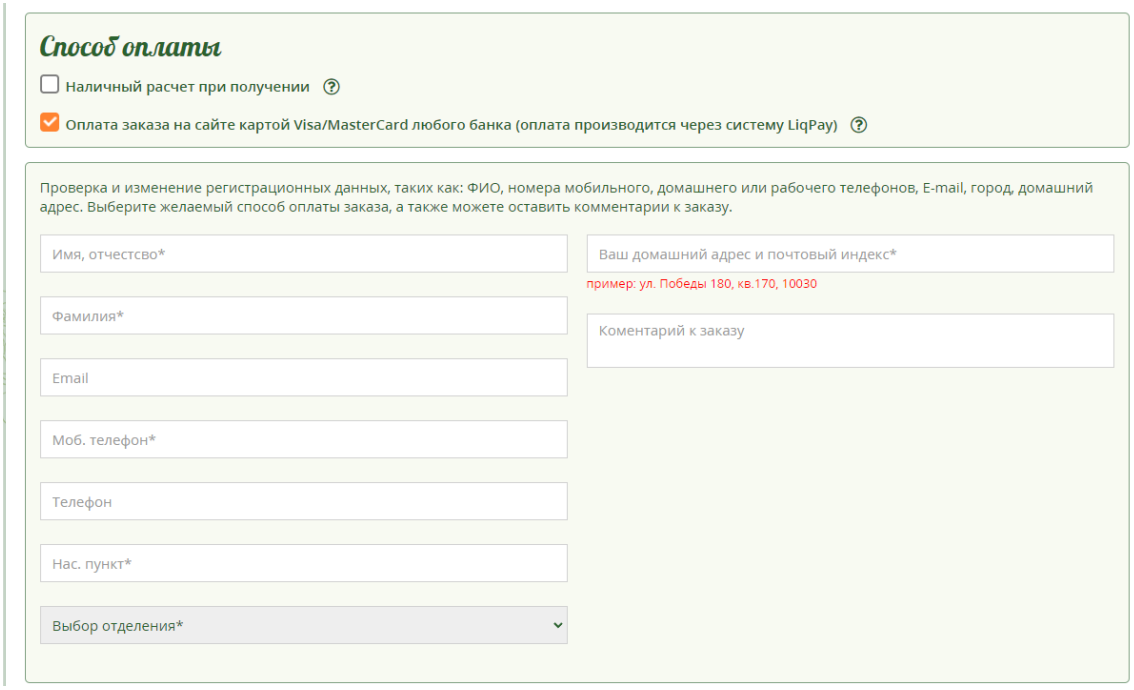

Рисунок 1.13 – Оформлення замовлення магазину Добродар

Недоліком даної форми є те, що при вказанні способу доставки «Новою поштою», все одно необхідно вводити домашню адресу, хоча це не має сенсу.

Даний магазин має більшість необхідних функцій для комфортного користування, проте має недоліки пов’язані з дизайном та логікою форми оформлення замовлення.

1.3. Постановка задачі

Проаналізувавши аналогічні відомі рішення, та визначивши необхідні вимоги для сайту, перейдемо до написання необхідних задач, які потребують реалізації.

Кінцевий результат онлайн магазину повинен містити у собі наступні сторінки та їхній функціонал:

- Головна сторінка. Повинна мати простий та зрозумілий інтерфейс з приємний дизайном, містити в собі головне меню із швидкою навігацією за категоріями та каталог продуктів;
- Сторінка з описом конкретного продукту повинна містити опис продукту, його ціну, зображення, а також можливість додати його до кошика, попередньо обравши бажану кількість товару;
- Кошик. Має відображати усі товари, додані до нього, їхню кількість з можливістю її редагування, видалення певного продукту, а також відображення кінцевої вартості замовлення;
- Сторінка оформлення замовлення має містити зрозумілу форму для вказання даних отримувача, вибору способу доставки та оплати;
- Сторінка реєстрації для відповідної реєстрації користувача у магазині;
- Сторінка особистого кабінету, з можливістю переглядати створені замовлення.

Слід не забути про те, що сайт повинен швидко та ефективно оброблювати усі дії користувачів, а також відповідати усім критеріям загальновідомих правил безпеки в інтернеті.

2. ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

2.1. Вибір мов програмування

Виконуючи поставлені задачі було обрані наступні мови програмування для візуальної частини : HTML, CSS, JavaScript, в той час серверна частина розроблювалась мовою Python.

HTML (Hyper Text Markup Language)

Мабуть, перше, що спадає на думку, коли ми чуємо “Web”, це звичайно мова розмітки HTML. З її назви можна зрозуміти, що це мова для побудови веб-сторінок, пов’язаних між собою за допомогою гіперпосилань, за допомогою розмітки.

Сам HTML документ складається з певних блоків, які мають наступні теги: <header>, <body> та <footer>, за допомогою яких веб-сторінка розбивається на три блоки, кожен з яких виводить необхідний у своєму блоці контент. Для прикладу <header> найчастіше використовується для відображення головного навігаційного меню, <footer> – для відображення якогось контенту внизу сторінки, а <body> в свою чергу, містить у собі основне наповнення сторінки. Header і footer, як правило, відображають однакову інформацію на усіх сторінках певного ресурсу, а body вже змінюється під конкретні вимоги тої чи іншої сторінки [3].

Існує дуже багато тегів, наприклад: <p>, <h1>, <img>, <table>, <video> тощо, кожен з яких використовується для певного відображення того чи іншого контенту. Так, наприклад, <p> використовується для виводу текстового абзацу, <img> – для виводу зображення і так далі.

CSS (Cascading Style Sheets)

Мови HTML достатньо лише для виведення певного контенту, а щоб весь цей контент обгорнути в привабливий дизайн на допомозі стає CSS.

CSS використовується для стилізації та компонування веб-сторінок – наприклад, для зміни шрифту, кольору, розміру та інтервалу вмісту, поділу його на кілька стовпців або додавання анімації та інших декоративних елементів [4].

За допомогою CSS ми можемо задати колір певного елементу, шрифт, задній фон, границі, відступи тощо.

Файл CSS складається з елементів, властивості яких ми хочемо змінити та переліку цих властивостей та їхніх значень. Наприклад, p{ color : red }. Даним кодом ми надаємо елементу p червоне забарвлення.

JavaScript

JavaScript це потужна мова, яка використовується для реалізації поведінки веб-сторінки, додавання інтерактивних елементів, відслідковування дій користувача, впровадження красивої анімації та багато іншого.

На базі цієї мови побудовано більшість фронтенд фреймворків, що робить його дуже популярним серед розробників візуальної частини веб-сторінок.

Послідовність інструкцій (що називається програмою, скриптом або сценарієм) виконується інтерпретатором, вбудованим у звичайний Web-браузер. Іншими словами, код програми вбудовується в HTML – документ і виконується на боці клієнта. Для виконання програми не потрібно навіть перезавантажувати Web-сторінку, всі програми виконуються у відповідь на будь-яку подію. Наприклад, перед відправленням даних форми можна перевірити їх на допустимі значення і, якщо значення не відповідають очікуваним, заборонити відправлення даних [5].

Python

Python є найпопулярнішою мовою серед розробників, яка використовується для розроблення алгоритмів, автоматизації процесів, аналізу та візуалізації даних та звичайно для створення веб-сайтів та веб-додатків. Через велику популярність мови, вона має дуже багато бібліотек та вже готових рішень для створення веб-сайтів.

Python проста у використанні та водночас має всі необхідні функції, вона має вбудовані типи даних високого рівня, такі як гнучкі масиви і словники. Python дозволяє розбивати програми на модулі, які можна буде в подальшому

використовувати в інших програмах. Також вона має автоматичне керування пам’яттю, типи зв’язані з об’єктами, а не змінними [6].

Більше того, Python має простий синтаксис де замість знаків пунктуації для позначення блоків коду, використовуються відступи. Також, вона є майже на всіх операційних системах, що в свою чергу робить мову більше універсальною.

2.2. Вибір фреймворків

В сучасному світі задля спрощення розробки веб-ресурсів створено багато фреймворків, які містять у собі вже певні готові рішення та шаблони. То ж використовувати чисту мову програмування без фреймворків є не найбільш ефективне рішення. Вони існують як для розробки фронт-енду так і бек-енду.

Bootstrap

Bootstrap це безкоштовний фреймворк, який використовується для розробки візуальної частини веб-сайту. Основною особливістю даного фреймворку є спрощена реалізація адаптивності та кросбраузерності.

В документації Bootstrap можна знайти дуже багато готових шаблонів з навігаційними меню, хедерами та футерами, формами реєстрації та багато іншого.

Для розташування елементів на сторінці в Bootstrap використовується сітка. Спочатку створюється контейнер, в який додається необхідна кількість рядків, а в рядках створюють колонки. Нижче наведемо приклад (рис. 2.1):

Рисунок 2.1 – Приклад сітки в Bootstrap [7]

Для створення такої сітки використовувався наступний код:

```
<div class="container">
  <div class="row">
```

```
    <div class="col-sm-8">col-sm-8</div>
    <div class="col-sm-4">col-sm-4</div>
  </div>
  <div class="row">
    <div class="col-sm">col-sm</div>
    <div class="col-sm">col-sm</div>
    <div class="col-sm">col-sm</div>
  </div>
</div>
```

Ще приємною особливістю є те, що для додавання стилів, немає необхідності створювати окремий CSS файл, в якому описувати бажані властивості елементів, все це можна зробити додавши необхідний клас в елемент та вказати бажані параметри, розміри тощо.

Django

Django це безкоштовний фреймворк з відкритим кодом для більш швидкої розробки серверної частини веб-сайтів та додатків. Він є дуже універсальним інструментом, за допомогою якого модна створювати не лише простий веб-сайт, а й соціальну мережу. Завдяки йому можна спростити собі життя, використовуючи вже готові шаблони та рішення для реєстрації користувачів, роботи а API, тощо. Також Django має вбудовану панель адміністрування, то ж розроблювати її самостійно не потрібно [8].

Django працює за MVT (Model View Template) архітектурою, тобто складається з трьох основних компонентів (рис. 2.2).

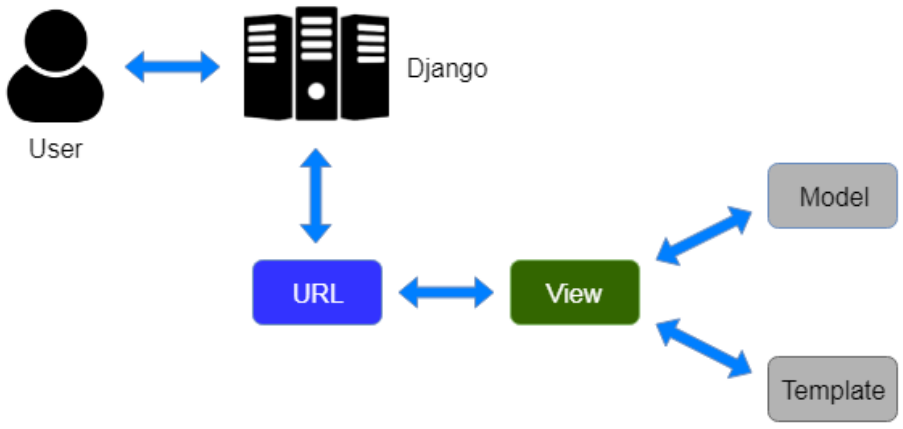

Рисунок 2.2 – Принцип роботи Django [8]

На рисунку вище бачимо, що користувач направляє світ запит до Django, Django оброблює даний запит та шукає необхідний шлях (URL). Якщо шлях знайдений то починає свою роботу Представлення, яке взаємодіє з базою даних та створює шаблон, який разом з даними буде надісланий користувачу, в якості відповіді. [8]

Даний фреймворк має дуже гарний захист від основних відомих видів кібератак, має безпечний спосіб управлінням паролями користувачів.

Django підтримує більшість відомих СУБД, таких як MySQL, SQLite, PostgreSQL тощо, з дуже простим встановленням та підключенням. Великим плюсом цього фреймворку є ще й те, що він використовує ORM (Object Relation Mapping). Це дає змогу взаємодіяти з базами даних використовуючи мову Python замість SQL.

Знаком якості даного фреймворку є те, що на ньому написані найпопулярніші веб-додатки, такі як YouTube, Instagram, Dropbox тощо.

2.3. Вибір СУБД (Система Управління Базами Даних)

СУБД це програмне забезпечення, за допомогою якого можна створювати бази даних та проводити операції над ними. Основними особливостями є робота з даними на зовнішніх носіях, обробка даних з операційної пам’яті, звітність щодо редагування даних тощо [9].

Найпопулярнішими СУБД є MySQL, Oracle, PostgreSQL, SQLite та інші. Список СУБД є доволі великим тому треба провести аналіз основних з них і визначитися з ідеальним рішенням для наших задач.

SQLite

Це СУБД на основі файлів, яка не потребує жодного встановлення чи налаштування, а отже вона не запускається, як окремий серверний процес, який потрібно налаштовувати, зупиняти та запускати. Через те, що вона використовує

безсерверну архітектуру, вона має гарну кросплатформеність. Вся база даних, доступна для читання та запису, міститься на одному дисковому файлі.

До переваг також можна віднести те, що SQLite є найкомпактнішою бібліотекою, серед перелічених вище. Хорошим варіантом її використання будуть веб-сайти з низьким трафіком або в якості освітніх цілей.

Одним з основних недоліків системи SQLite є відсутність багатокористувацьких можливостей, які можна знайти в повноцінних системах СУБД, таких як MySQL і PostgreSQL. Це означає відсутність детального контролю доступу, зручної системи керування користувачами та можливостей безпеки, крім шифрування самого файлу бази даних.Це є серйозним недоліком при розробці багатокористувацьких програм [10].

MySQL

MySQL це одна з найпопулярніших СУБД з відкритим кодом. Вона використовую архітектуру сервер-клієнт, на відміну від SQLite, що забезпечує більшу продуктивність [10].

Дана СУБД є гарним вибором з точки зору безпеки, оскільки вона використовує систему привілеїв доступу для аутентифікації користувачів, систему керування обліковими записами та шифрування з’єднання за допомогою SSL.

Також вона є простішою у встановленні ніж PostgreSQL, та має ширшу спільноту.

PostgreSQL

PostgreSQL є основною базою даних в проектах, написаних на Python. Вона, як і MySQL використовує клієнт-серверну технологію. PostgreSQL заслужив міцну репутацію завдяки своїй перевіреній архітектурі, надійності, цілісності даних та великому набору функцій [10].

За замовчуванням, обраний нами фреймворк Django, працює з базою даних SQLite, проте більшість розробників натомість використовують PostgreSQL через деякі її особливості, які підтримує Django:

- Django надає ряд типів даних, які працюватимуть лише з PostgreSQL.
- Django має django.contrib.postgres для виконання операцій з базою даних на PostgreSQL
- Якщо ви створюєте програму з картами або зберігаєте географічні дані, вам потрібно використовувати PostgreSQL, оскільки GeoDjango повністю сумісний лише з PostgreSQL
- Django підтримує більшість специфічних особливості PostgreSQL таких як: функції агрегації, поля та віджети, пошуки, операції міграції, валідатори та інше [11].

Завдяки можливості PostgreSQL виконувати паралельну обробку, він виходить на перше місце під час виконання великих SELECT запитів.

Свій вибір зупинимо на PostgreSQL, через наявні переваги у сумісному використанні з нашим фреймворком.

2.4. Вибір IDE

PyCharm

Розроблюючи наш проект, ми використовували таке середовище, як PyCharm . Це відомий продукт від компанії JetBrains створений спеціально для розробки коду на Python.

Він має швидкі автодоповнювачі, шаблони коду, швидкий рефакторінг та зрозумілу навігацію кодом. Також містить в собі редактори для HTML, CSS, JavaScript, що полегшують розробку веб-додатків. Має вбудовані консолі для Python, Django, SSH, повнофункціональний графічний налагоджувач, а також інтегрований з системою контролю версій. [12]

Ще в ньому наявні можливості підключення та відображення баз даних, а також створення UML та ERD діаграм.

Він самостійно створює та активую віртуальне оточення для наших проектів, що значно заощаджує наш час.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1. Інформаційна модель

Виходячи з основних поставлених задач та принципів побудови веб-сайтів, створимо діаграму для відображення структури сайту (рис. 3.1).

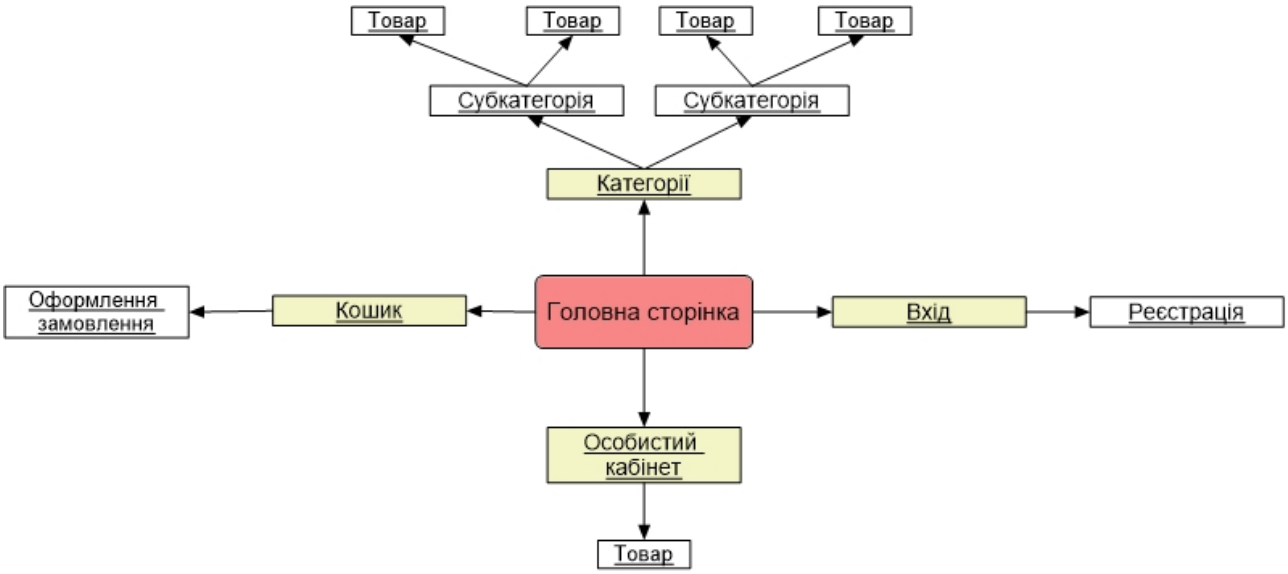

Рисунок 3.1 – Site Map діаграма

Наш сайт має два типи користувачів: адмін та користувач (рис. 3.2).

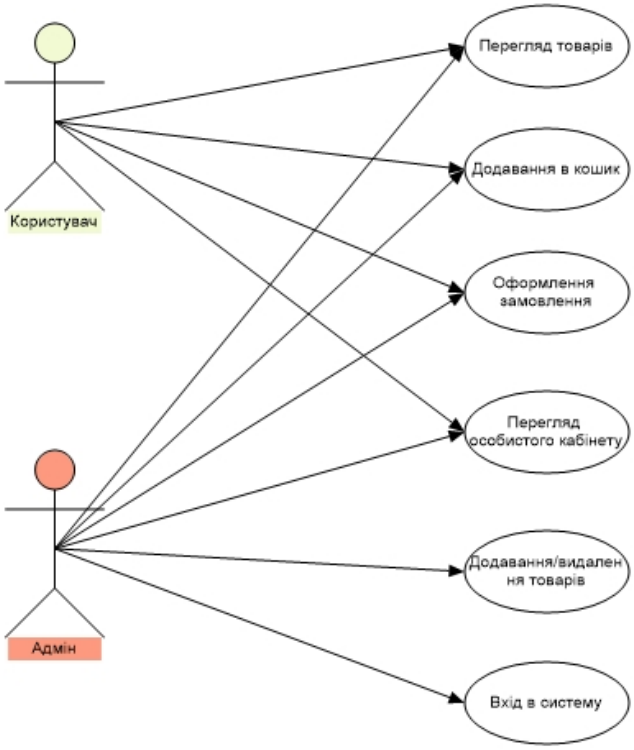

Рисунок 3.2 – Use Case діаграма

Відмінністю між ними є те, що адміністратор має права для редагування, додавання чи видалення товарів, та інших дій з БД.

3.2. Розроблення бази даних

Для повноцінної та ефективної роботи веб-сайту, потрібно мати базу даних, яка буде містити інформацію про користувача, товари, виконані замовлення тощо. Попередньо, ми вже вирішили, що будемо працювати з базою даних PostgreSQL.

В нашому випадку наша БД складається з таких таблиць:

- Customer, яка зберігає інформацію про користувачів;
- Category, містить інформацію про категорії;
- Product, містить інформацію про товар;
- Productimage, містить зображення для товару.
- Order, в якій міститься інформація про виконані замовлення;
- Orderitem, що є проміжною між таблицями Product та Order, для отримання інформації про куплений продукт;

Далі, зробимо детальний розбір кожної з таблиць.

Наведемо структуру таблиці Customer (рис. 3.3)

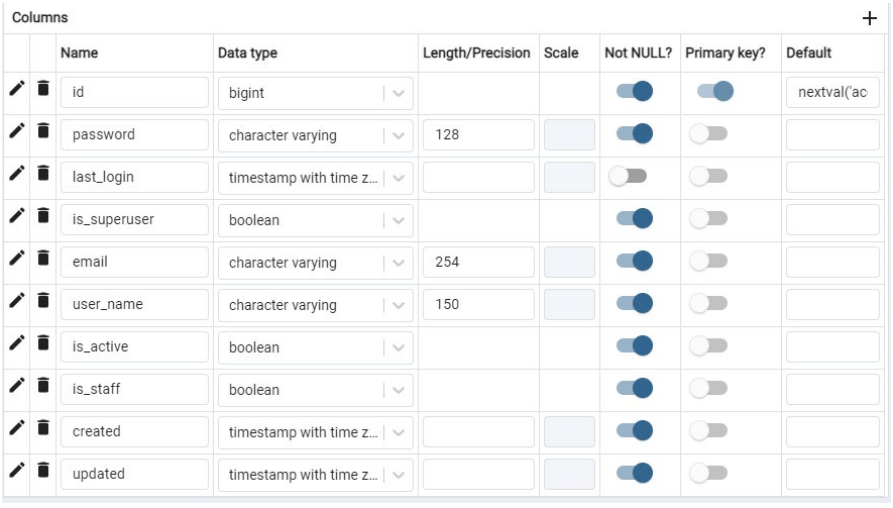

Рисунок 3.3 – Структура таблиці Customer

Дана таблиця складається з десяти колонок, які необхідні для зберігання інформації про користувача. Вона створена у наслідок розширення вже готової таблиці BaseUser, яка є шаблоном для створення користувачів, з певними змінами, такими як те, що при створенні користувача, замість поля username використовується email.

Далі наведена таблиця Category, для зберігання інформації про наявні категорії. (рис. 3.4)

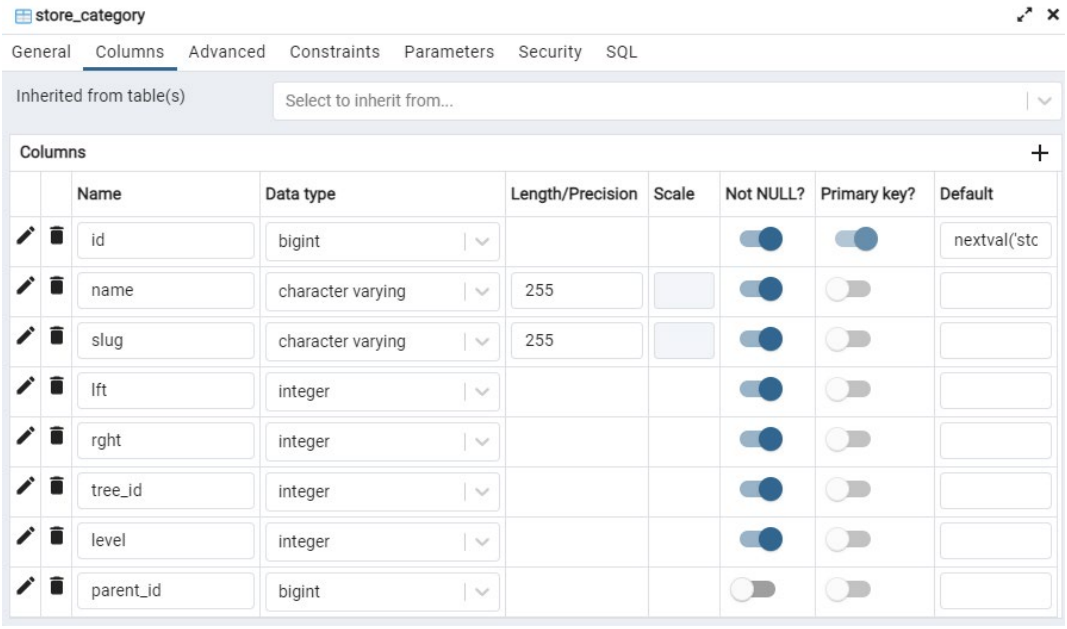

Рисунок 3.4 – Структура таблиці Category

Таблиця Category створена у вигляді дерева для представлення ієрархічних даних, за допомогою технології MPTT.

MPTT — це техніка для зберігання ієрархічних даних у базі даних. Мета полягає в тому, щоб зробити операції пошуку дуже ефективними [13].

В Django є спеціальний додаток Django-mptt для багаторазового використання, яка має на меті полегшити вам використання MPTT з вашими власними моделями Django.

Він піклується про деталі керування таблицею бази даних як деревоподібною структурою та надає інструменти для роботи з деревами екземплярів моделі [13].

Дана технологія використовувалась для реалізації під-категорій.

Наступною таблицею є Product (рис. 3.5)

store_product

General | Columns | Advanced | Constraints | Parameters | Security | SQL

Inherited from table(s) — Select to inherit from...

Columns

		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
✎	🗑	id	bigint			●	●	nextval('stc
✎	🗑	title	character varying	255		●	○	
✎	🗑	description	text			●	○	
✎	🗑	slug	character varying	255		●	○	
✎	🗑	regular_price	numeric	7	2	●	○	
✎	🗑	discount_price	numeric	7	2	●	○	
✎	🗑	is_active	boolean			●	○	
✎	🗑	created_at	timestamp with time zo...			●	○	
✎	🗑	updated_at	timestamp with time zo...			●	○	
✎	🗑	category_id	bigint			●	○	

Рисунок 3.5 – Структура таблиці Product

Таблиця Product є головною таблицею для зберігання інформації про товар, його назву, ціну, опис тощо.

Далі наведемо структуру таблиці Productimage (рис. 3.6)

Columns

		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
✎	🗑	id	bigint			●	●	nextval('s
✎	🗑	image	character varying	100		●	○	
✎	🗑	alt_text	character varying	255		○	○	
✎	🗑	is_feature	boolean			●	○	
✎	🗑	created_at	timestamp with tim...			●	○	
✎	🗑	updated_at	timestamp with tim...			●	○	
✎	🗑	product_id	bigint			●	○	

Рисунок 3.6 – Структура таблиці Productimage

Ця таблиця зберігає інформацію про зображення для певного продукту.

Створення окремої таблиці для зображень було необхідно для того, щоб товар міг мати декілька зображень, одне з яких буде головним та буде відображатися в каталозі, кошику тощо, а інші – в описі товару.

Наступною таблицею розглянемо Order (рис. 3.7)

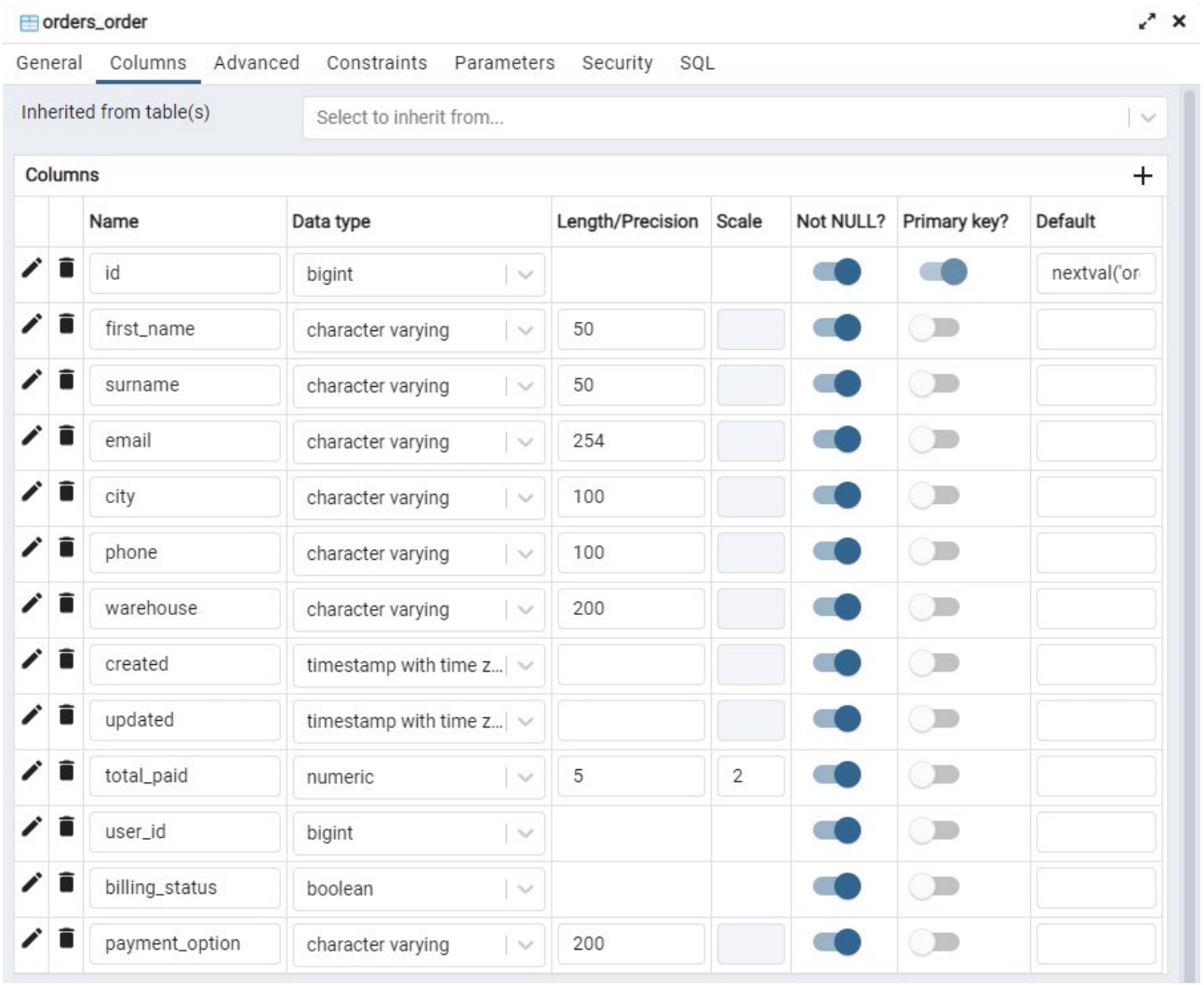

Рисунок 3.7 – Структура таблиці Order

При успішному виконанні замовлення, інформація про отримувача, дату та час замовлення, його статус та метод оплати буде додана до таблиці Order.

Вона пов’язана з таблицею Orderitem та Product та слугує для збереження інформації про товар, його кількість та ціну, який міститься в замовленні.

Структура таблиці наведена нижче. (рис. 3.8)

Columns

		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
✎	🗑	id	bigint			●	●	nextval('orc
✎	🗑	price	numeric	5	2	●	○	
✎	🗑	quantity	integer			●	○	
✎	🗑	order_id	bigint			●	○	
✎	🗑	product_id	bigint			●	○	

Рисунок 3.8 – Структура таблиці Orderitem

Створення даних таблиць наведено у “Додаток А”.

3.3. Програмна реалізація

На рисунку нижче зображена структура файлів проекту. (рис 3.9)

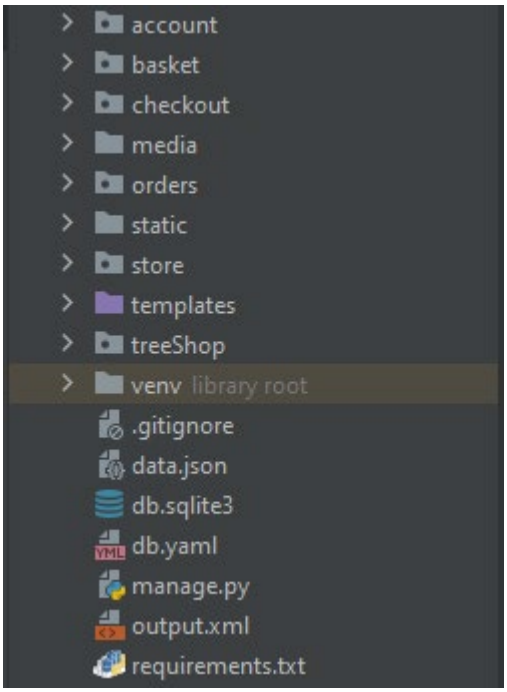

Рисунок 3.9 – Файлова структура проекту

Гарною практикою є написання кожного окремого функціонального модуля, в окремій папці, яка в Django називається «додаток».

Для реалізації виводу домашньої сторінки та інформації про наявні товари, деталі певного товару та список категорій використовуємо функції product_all, product_detail та category_list з файлу store/views.py відповідно. Повний код додатку Store наведено в “Додаток Б”.

Код даних функцій:

```
def product_all(request):
    products = Product.objects.all()
    return render(request, 'store/home.html', {'products': products})

def product_detail(request, slug):
    product = get_object_or_404(Product, slug=slug, is_active=True)
    return render(request, 'store/products/single.html', {'product': product})

def category_list(request, category_slug=None):
    category = get_object_or_404(Category, slug=category_slug)
    products = Product.objects.filter(category=category)
    return render(request, 'store/products/category.html', {'category': category, 'products': products})
```

Додаток Account відповідає за все, що пов’язане з користувачем: реєстрація, авторизація, активацію акаунта та відображення особистого кабінету.

Форма реєстрації має наступний вигляд: (рис 3.10)

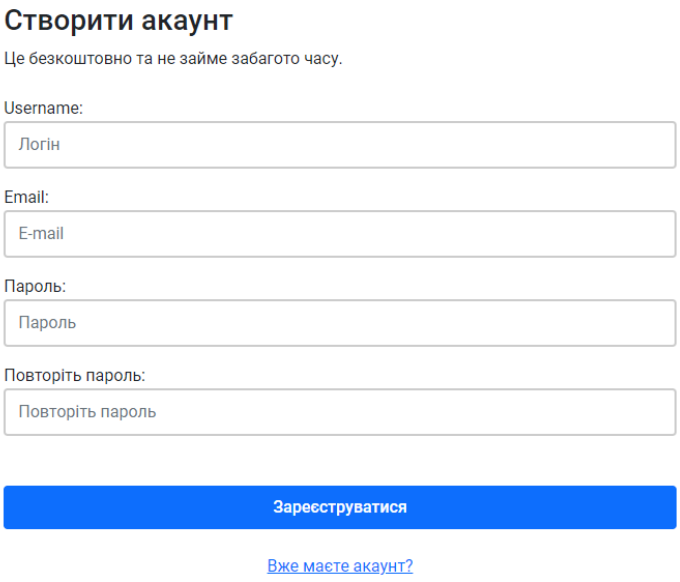

Рисунок 3.10 – Форма реєстрації

Вона складається з чотирьох полів, обов’язкових для введення. Дані в формі перевіряються, тобто електронна пошта потрібна бути правильного формату та не бути вже зайнятою для реєстрації, паролі потрібні співпадати між собою, а логін бути унікальним.

Для запобігання CSRF-атак використовуються csrf_token, який надає Django для боротьби з цим типом кібератаки. Якщо всі дані валідні вони записуються до БД, користувач отримує повідомлення, що реєстрація успішна, а на його пошту надходить лист з посиланням на активацію акаунта.

Розсилка відбувається за допомогою метода send_mail, в якому потрібно вказати тему листа, повідомлення, відправника, адресати та інші параметри, за необхідністю.

Код функції, яка відповідає за це все наведемо нижче:

```
def account_register(request):
    if request.user.is_authenticated:
        return redirect('account:dashboard')

    if request.method == 'POST':
        registerForm = RegistrationForm(request.POST)
        if registerForm.is_valid():
            user = registerForm.save(commit=False)
            user.email = registerForm.cleaned_data['email']
            user.set_password(registerForm.cleaned_data['password'])
            user.is_active = False
            user.save()
            current_site = get_current_site(request)
            subject = 'Activate your Account'
            message                                                    =
render_to_string('account/registration/account_activation_email.html', {
                'user': user,
                'domain': current_site.domain,
                'uid': urlsafe_base64_encode(force_bytes(user.pk)),
                'token': account_activation_token.make_token(user),
            })
            send_mail(subject, message, 'greenspace1@ukr.net',
                      [user.email], fail_silently=False)
```

```
                return HttpResponse('Реєстрація успішна! Перейдіть в вашу поштову скриню та активуйте акаунт')
        else:
            registerForm = RegistrationForm()
        return render(request, 'account/registration/register.html', {'form': registerForm})
```

Для генерування посилання для активації акаунту, використовуємо вже вбудоване в Django рішення. Посилання містить доменне ім’я, унікальний токен та унікальне значення uidb64.

Якщо активація успішна, то користувача перенаправляє в його особистий кабінет з повідомленням про успішну активацію, а також в БД записується, що даний користувач активний. Для більш детального коду даного додатку дивись “Додаток В”.

Наступним додатком є Basket. Він використовується для додавання, видалення та редагування товарів в кошику. Інформацію про товар доданий користувачем в кошик ми записуємо в сесії, а потім при необхідності беремо звідти дані.

Для цього маємо клас Basket, який містить функції для додавання інформації про продукт, його ціну та кількість, видалення її, оновлення, збір даних для подальшого їхнього використання в запитах до БД, а також функцію для підрахунку загальної вартості товарів.

Реалізація даного класу має вигляд:

```
class Basket:
    def __init__(self, request):
        self.session = request.session
        basket = self.session.get(settings.BASKET_SESSION_ID)
        if settings.BASKET_SESSION_ID not in request.session:
            basket = self.session[settings.BASKET_SESSION_ID] = {}
        self.basket = basket

    def add(self, product, qty):
        product_id = str(product.id)

        if product_id in self.basket:
```

```
                self.basket[product_id]["qty"] = qty
            else:
                self.basket[product_id] = {"price": str(product.regular_price),
"qty": qty}

            self.save()



        def __iter__(self):
            product_ids = self.basket.keys()
            products = Product.objects.filter(id__in=product_ids)
            basket = self.basket.copy()

            for product in products:
                basket[str(product.id)]["product"] = product

            for item in basket.values():
                item["price"] = Decimal(item["price"])
                item["total_price"] = item["price"] * item["qty"]
                yield item

        def __len__(self):
            return sum(item["qty"] for item in self.basket.values())

        def update(self, product, qty):
            product_id = str(product)
            if product_id in self.basket:
                self.basket[product_id]["qty"] = qty
            self.save()

        def get_subtotal_price(self):
            return  sum(Decimal(item["price"])  *  item["qty"]  for  item  in
self.basket.values())

        def get_total_price(self):
            return  sum(Decimal(item["price"])  *  item["qty"]  for  item  in
self.basket.values())

        def delete(self, product):
            product_id = str(product)
```

```
            if product_id in self.basket:
                del self.basket[product_id]
                self.save()

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

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

Повний код цього додатка дивись в “Додаток Г”.

Нижче наведемо те, як кошик виглядає візуально: (рис 3.11)

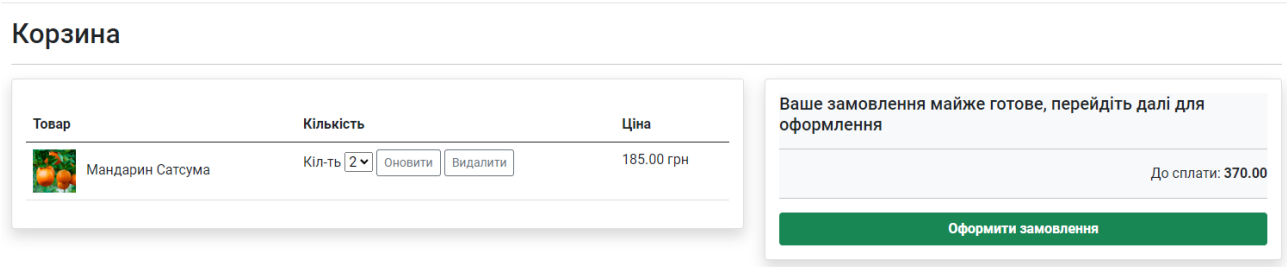

Рисунок 3.11 – Візуальний вигляд корзини

Як бачимо, виводиться вся необхідна інформація про товар, також інформація про загальну вартість всієї корзини та можливість оновити кількість товару, або видалити певний з них.

Наступним після функціоналу кошика, треба розглянути процес оформлення та обробки замовлення. За це відповідає додаток Checkout.

Нижче наведемо сторінку Checkout: (рис 3.12)

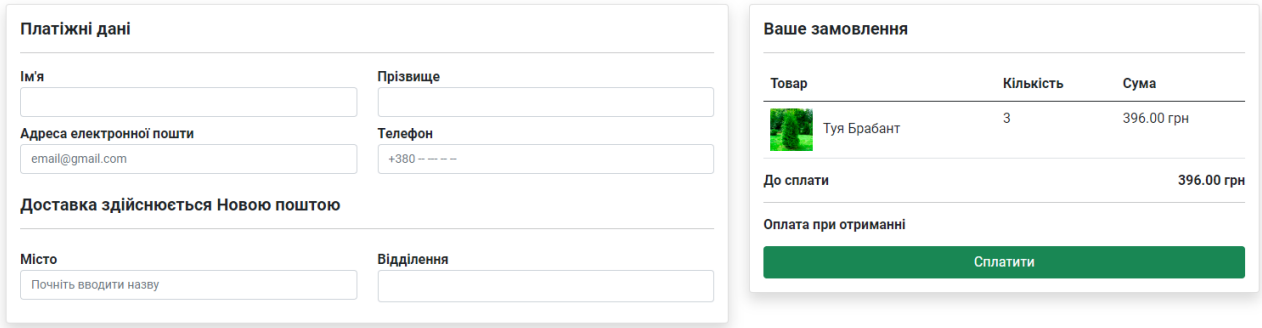

Рисунок 3.12 – Візуальний вигляд сторінки Checkout

Як бачимо, вона містить форму, яку потрібно обробити, та загальну інформацію з корзини. Реалізацію форми та всі інші шаблони можна знайти в “Додаток Д”.

Для обробки даної форми використовується функція placeorder. За допомогою POST запитів, до неї надходять дані, введені в форму. Функція, в свою чергу, оброблює ці дані і записує їх в БД, звертається до класу Basket за видаленням даних сесії, перенаправляє користувача на головну сторінку та виводить повідомлення про успішне замовлення.

Код даної функції наведений нижче:

```
@login_required
def placeorder(request):
    basket = Basket(request)
    if request.method == 'POST':
        neworder = Order()
        neworder.user = request.user
        neworder.first_name = request.POST.get('fname')
        neworder.surname = request.POST.get('lname')
        neworder.email = request.POST.get('email')
        neworder.phone = request.POST.get('phone')
        neworder.city = request.POST.get('city')
        neworder.warehouse = request.POST.get('warehouse')
        neworder.payment_option = request.POST.get('payment_option')
        total_paid = basket.get_subtotal_price()
        neworder.total_paid = total_paid
        neworder.billing_status = True
        neworder.save()
        order_id = neworder.pk
        for item in basket:
            OrderItem.objects.create(order_id=order_id,
product=item["product"], price=item["price"], quantity=item["qty"])
        basket.clear()
        messages.success(request, "Замовлення успішне, перейдіть в Кабінет,
щоб переглянути деталі")
    return redirect('/')
```

Також в формі було використане готове рішення для роботи з API сервісу Нова Пошта, для отримання інформації про існуючі відділення на містах.

Код даного скрипту наведений у “Додаток Е”.

А отже ввівши назву міста, можна отримати та обрати зі списку необхідне відділення: (рис 3.13)

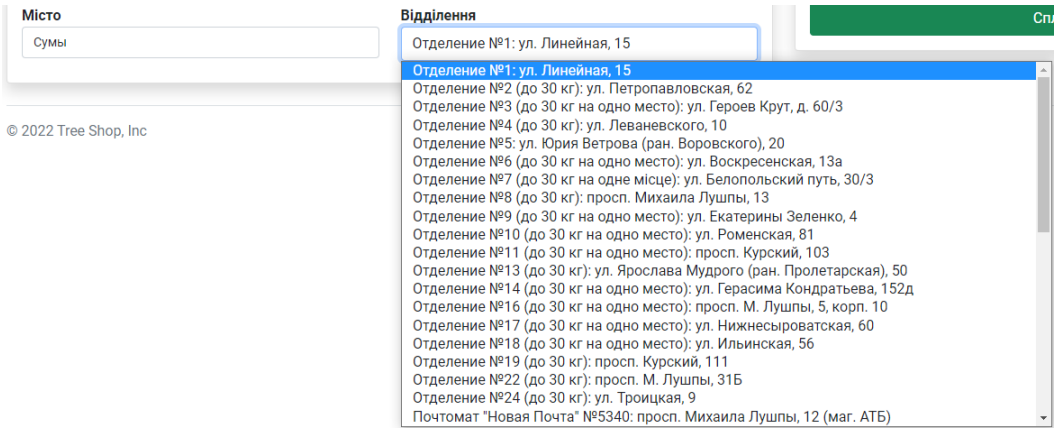

Рисунок 3.13 – Виведення списку відділень Нової Пошти

Додаток Order, в основному слугує для створення таблиць Order та OrderItem, та виведення інформацію про успішні замовлення в особистому кабінеті користувача.

Зміст функції user_orders для виведення виконаних замовлень клієнту:

```
def user_orders(request):
    user_id = request.user.id
    orders = Order.objects.filter(user_id=user_id).filter(billing_status=True)
    return orders
```

Дана функція повертає ті замовлення, які належать клієнту, та мають статус сплаченого.

3.4. Тестування

Будемо використовувати ручний метод тестування, щоб подивитися чи всі функції працюють вірно.

Розпочнемо зі сторінки продукту, де виберемо бажану його кількість та додаймо до кошика. (рис 3.14)

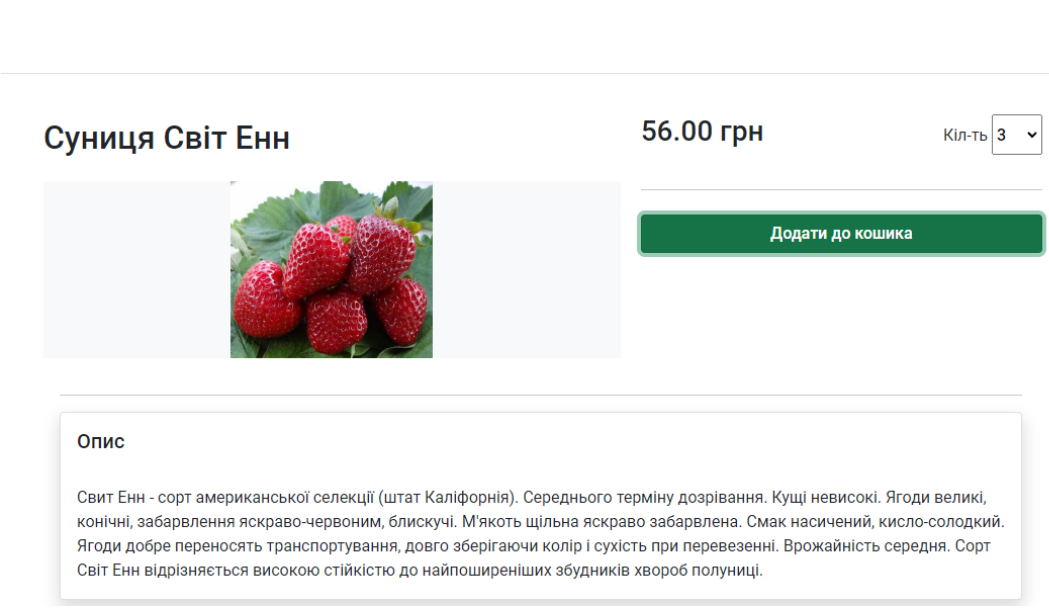

Рисунок 3.14 – Додавання до кошика

При натисканні кнопки «Додати до кошика», бачимо що кількість одиниць товару в кошику змінився з нуля на три, а в консолі знаходимо, що POST запит з додаванням товару в корзину виконаний успішно, про що інформує відповідь «200».(рис. 3.15)

Рисунок 3.15 – Успішне додавання товару в корзину

Додаймо ще кілька товарі, для наглядності та перейдемо до кошика.(рис. 3.16)

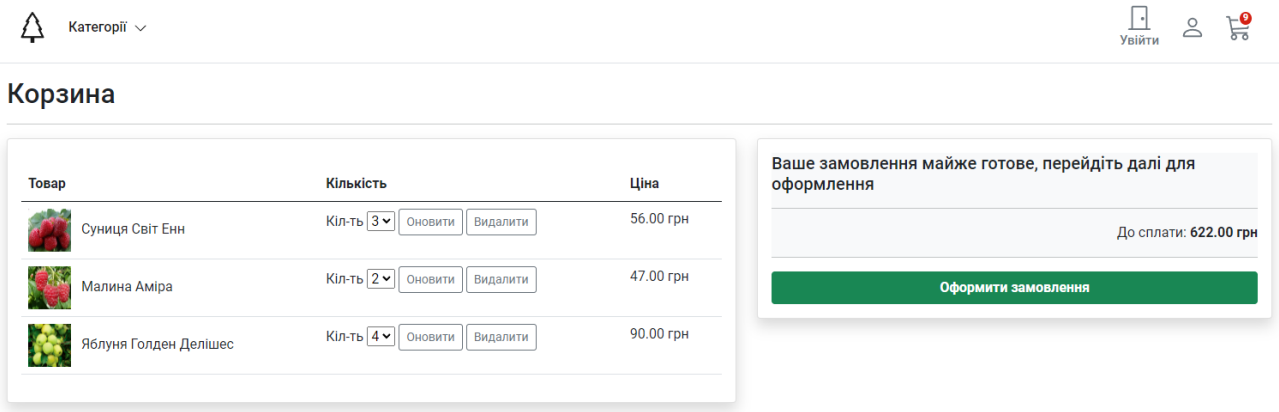

Рисунок 3.16 – Корзина нашого магазину

Як бачимо, що кількість товару та загальну вартість рахується вірно, то ж можемо перейти до тестування редагування кількості та видалення товару. До прикладу, видалимо перший товар з корзини. Кількість товару в кошику повинна оновитися до шести, а сума до сплати повинна скласти 454 грн (рис. 3.17)

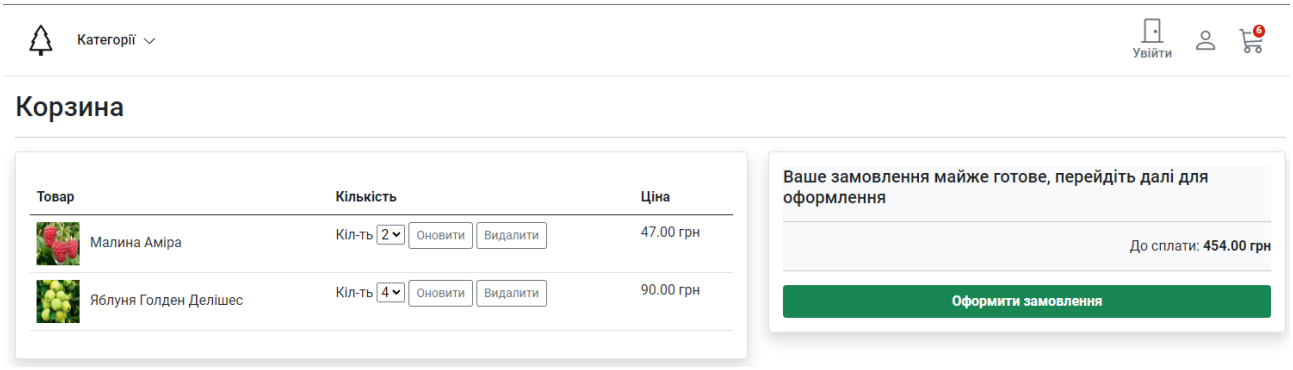

Рисунок 3.17 – Успішний результат видалення

Для оновлення кількості потрібно обрати бажану кількість товару та натиснути оновити.

Змінимо кількість останнього товару з 4 на 2. Як і в випадку з видаленням загальна кількість та вартість повинні оновитися. (рис. 3.18)

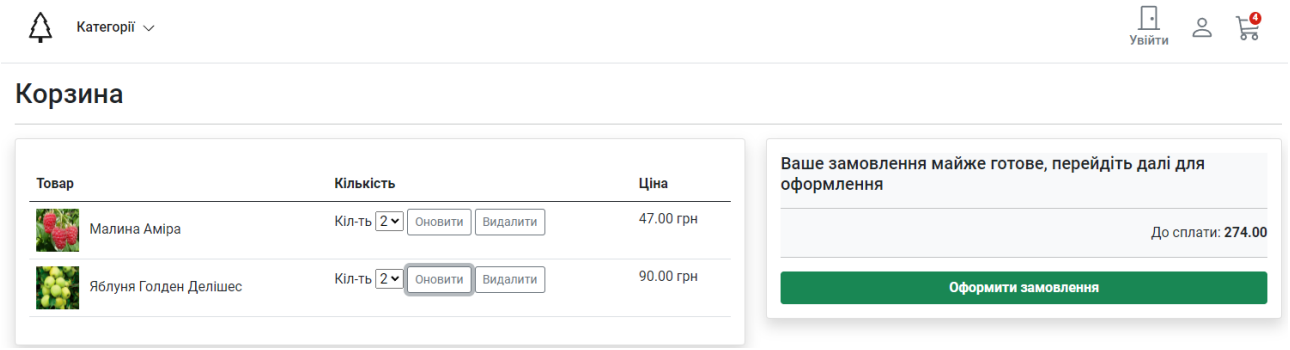

Рисунок 3.18 – Успішне оновлення кількості

Далі при натисканні «Оформити замовлення» ми перейдемо до реєстрації, оскільки нашого облікового запису ще немає.

На рисунках нижче, бачимо що валідація працює успішно, та не дає створити акаунт з некоректним email (рис. 3.19) або різними паролями (рис.3.20)

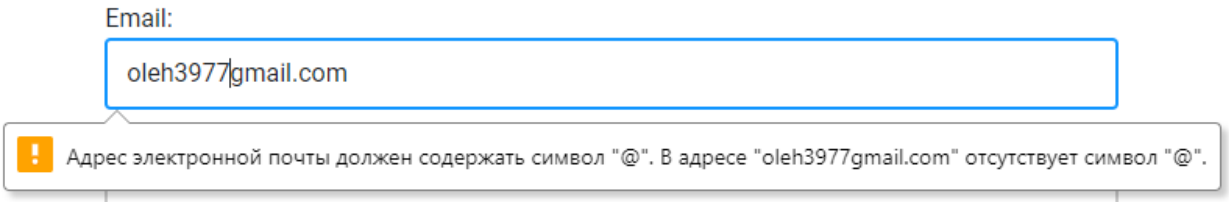

Рисунок 3.19 – Помилка при валідації email

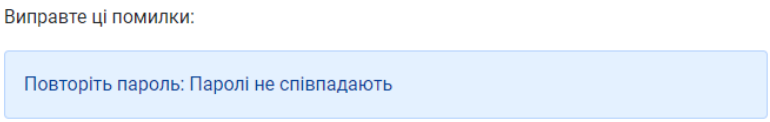

Рисунок 3.20 – Помилка при валідації паролів

Якщо ввести всі дані коректно та натиснути «Зареєструватися», ми отримаємо наступне повідомлення (рис. 3.21)

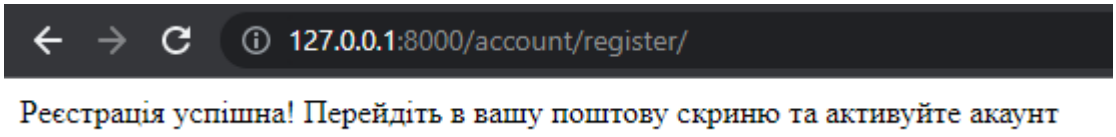

Рисунок 3.21 – Повідомлення при реєстрації

А на поштову скриню надійде лист з посиланням на активацію(рис 3.22)

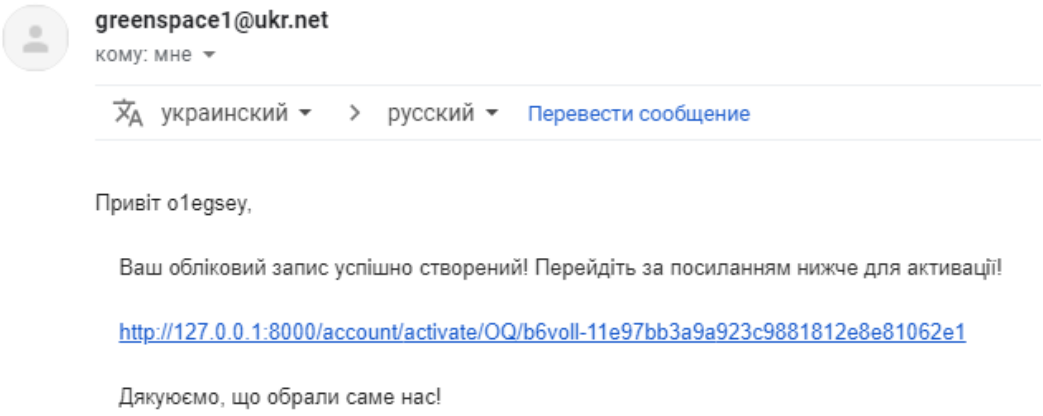

Рисунок 3.21 – Лист активації

При переході за посилання, акаунт активується, а користувача перенаправляємо до особистого кабінету, де він отримує повідомлення про успішну авторизацію. (рис. 3.22)

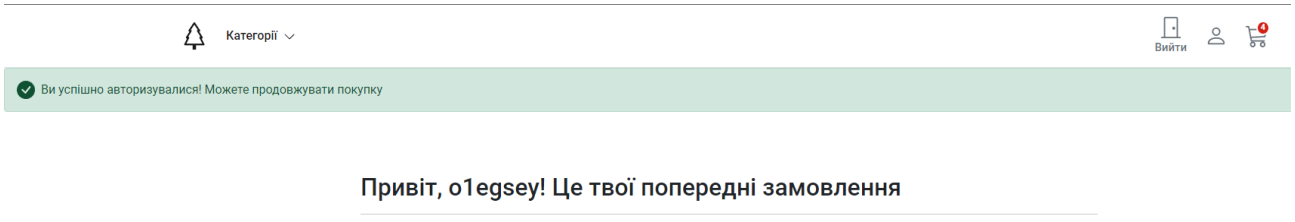

Рисунок 3.22 – Особистий кабінет після активації акаунту

Далі перейдемо до оформлення замовлення. Введемо інформацію отримувача, відділення для доставки та натиснемо «Сплатити» (рис. 3.23)

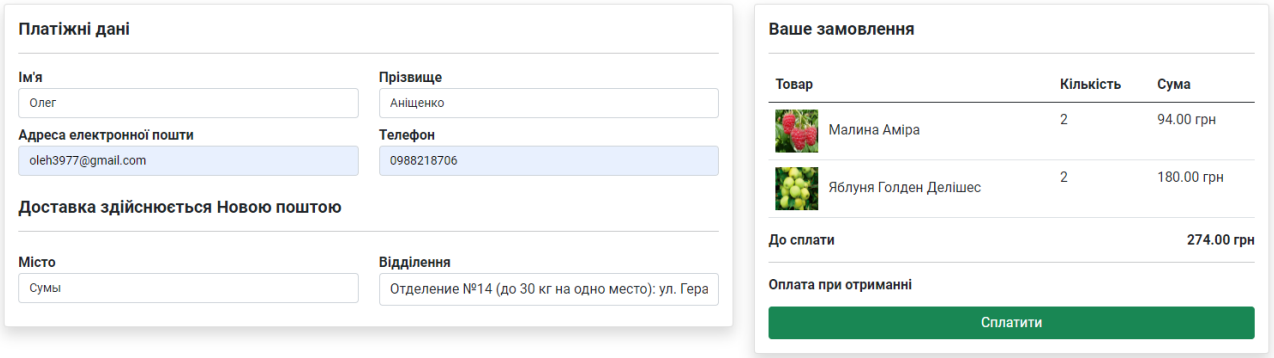

Рисунок 3.23 – Заповнена форма для оформлення замовлення

Роботу сайту можна буде назвати коректною, якщо інформація про користувача запишеться в БД, корзина анулюється, а користувача перенаправить до особистого кабінету, де він зможе побачити інформацію про дане замовлення.

Нижче наведемо результат. (рис. 3.24)

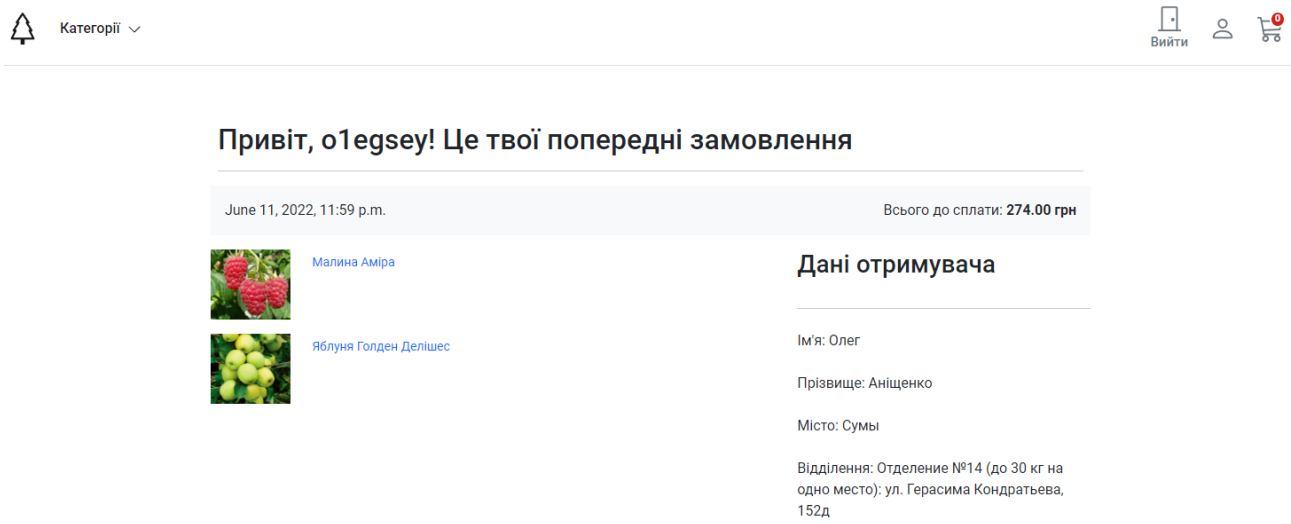

Рисунок 3.24 – Особистий кабінет користувача

Як бачимо, кошик анульований, дані про замовленні вірні, то ж можемо сказати, що даний елемент сайту працює коректно.

На цьому дане тестування можна завершити, грубих помилок не було знайдено.

ВИСНОВКИ

Виконуючи поставлену задачу, було створено веб-сайт, який відповідаю усім основним вимогам та критеріям, описаних в завданні. Було створено базу даних, яка зберігає в собі відомості про товари, користувачів тощо. Розроблено основні функції для навігації сторінкою, перегляду товару, додавання його в коши, оброблення замовлення. Була створена система для реєстрації та авторизації користувачів, з необхідністю підтвердження електронної пошти.

Був імплементований простий, зручний, а головне зрозумілий дизайн та інтерфейс для сайту.

Наприкінці було проведене тестування сайту, яке не виявило явних грубих помилок.

Зважаючи на всі ці факти, виконання поставлених завдань можна вважати успішним.

СПИСОК ЛІТЕРАТУРИ

1. Веб технології. Їх різновиди та функції - http://sites.znu.edu.ua/webprog/lect/1170.ukr.html
2. Введення в протоколи HTTP та HTTPS - https://docs.microsoft.com/ru-ru/azure/rtos/netx-duo/netx-duo-web-http/chapter1
3. Початок роботи з HTML - https://developer.mozilla.org/ru/docs/Learn/HTML/Introduction_to_HTML/Getting_started
4. Що таке CSS - https://developer.mozilla.org/ru/docs/Learn/CSS/First_steps/What_is_CSS
5. Мова JavaScript та її можливості - https://sites.google.com/site/webtehnologiietawebdizajn/mova-javascript-ta-ieie-mozlivosti
6. Що таке Python - http://www.plug.org.ua/documentation/about-python
7. Система сітки Bootstrap - https://getbootstrap.com/docs/5.2/layout/grid/
8. What is Django - https://www.ibm.com/cloud/learn/django-explained
9. СУБД (Система Управлінням Базами Даних) - https://itglobal.com/ru-ru/company/glossary/subd-sistema-upravleniya-bazami-dannyh/
10. A comparison of Relation Databases - https://logz.io/blog/relational-database-comparison/
11. Why Django is so impressive for developing with PostgreSQL and Python - https://www.enterprisedb.com/postgres-tutorials/why-django-so-impressive-developing-postgresql-and-python
12. JetBrains Pycharm - https://itpro.ua/product/jetbrains-pycharm/?tab=description
13. Django-mptt documentation overview - https://django-mptt.readthedocs.io/en/latest/overview.html

Додаток А

Код файлу store/models.py

```
from django.db import models
from django.urls import reverse
from mptt.models import MPTTModel, TreeForeignKey


class Category(MPTTModel):
    name = models.CharField(verbose_name='Category Name', help_text='Required and unique', max_length=255, unique=True)
    slug = models.SlugField(verbose_name='Category slug', max_length=255, unique=True)
    parent = TreeForeignKey('self', on_delete=models.CASCADE, null=True, blank=True, related_name='children')

    class MTTPMeta:
        order_insertion_by = ['name']

    class Meta:
        verbose_name = "Category"
        verbose_name_plural = "Categories"

    def get_absolute_url(self):
        return reverse("store:category_list", args=[self.slug])

    def __str__(self):
        return self.name


class Product(models.Model):
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    title = models.CharField(
        verbose_name="title",
        help_text="Required",
        max_length=255,
    )
    description = models.TextField(verbose_name="description", help_text="Not Required", blank=True)
    slug = models.SlugField(max_length=255)
    regular_price = models.DecimalField(verbose_name="Regular price", max_digits=7, decimal_places=2)
    discount_price = models.DecimalField(verbose_name="Discount price", max_digits=7, decimal_places=2)
    is_active = models.BooleanField(
```

```
        verbose_name="Product visibility",
        help_text="Change product visibility",
        default=True,
    )
    created_at = models.DateTimeField("Created at", auto_now_add=True,
editable=False)
    updated_at = models.DateTimeField("Updated at", auto_now=True)

    class Meta:
        ordering = ("-created_at",)
        verbose_name = "Product"
        verbose_name_plural = "Products"

    def get_absolute_url(self):
        return reverse("store:product_detail", args=[self.slug])

    def __str__(self):
        return self.title


class ProductImage(models.Model):
    product = models.ForeignKey(Product, on_delete=models.CASCADE,
related_name="product_image")
    image = models.ImageField(
        verbose_name="image",
        help_text="Upload a product image",
        upload_to="images/",
        default="images/default.png",
    )
    alt_text = models.CharField(
        verbose_name="Alternative text",
        help_text="Please add alternative text",
        max_length=255,
        null=True,
        blank=True,
    )
    is_feature = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True, editable=False)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        verbose_name = "Product Image"
        verbose_name_plural = "Product Images"
```

Код файлу orders/models.py

```
from django.conf import settings
from django.db import models
from store.models import Product


class Order(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE,
related_name="order_user")
    first_name = models.CharField(max_length=50)
    surname = models.CharField(max_length=50)
    email = models.EmailField(max_length=254, blank=True)
    city = models.CharField(max_length=100, blank=True)
    phone = models.CharField(max_length=100)
    warehouse = models.CharField(max_length=200, blank=True)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    total_paid = models.DecimalField(max_digits=5, decimal_places=2)
    payment_option = models.CharField(max_length=200, blank=True)
    billing_status = models.BooleanField(default=False)

    class Meta:
        ordering = ("-created",)

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


class OrderItem(models.Model):
    order = models.ForeignKey(Order, related_name="items",
on_delete=models.CASCADE)
    product = models.ForeignKey(Product, related_name="order_items",
on_delete=models.CASCADE)
    price = models.DecimalField(max_digits=5, decimal_places=2)
    quantity = models.PositiveIntegerField(default=1)

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

Код файлу account/models.py

```
from django.contrib.auth.models import (AbstractBaseUser, BaseUserManager,
PermissionsMixin)
from django.core.mail import send_mail
from django.db import models
from django.utils.translation import gettext_lazy as _
```

```
class CustomAccountManager(BaseUserManager):

    def create_superuser(self, email, user_name, password, **other_fields):

        other_fields.setdefault('is_staff', True)
        other_fields.setdefault('is_superuser', True)
        other_fields.setdefault('is_active', True)

        if other_fields.get('is_staff') is not True:
            raise ValueError(
                'Superuser must be assigned to is_staff=True.')
        if other_fields.get('is_superuser') is not True:
            raise ValueError(
                'Superuser must be assigned to is_superuser=True.')

        return self.create_user(email, user_name, password, **other_fields)

    def create_user(self, email, user_name, password, **other_fields):

        if not email:
            raise ValueError(_('You must provide an email address'))

        email = self.normalize_email(email)
        user = self.model(email=email, user_name=user_name,
                          **other_fields)
        user.set_password(password)
        user.save()
        return user


class Customer(AbstractBaseUser, PermissionsMixin):
    email = models.EmailField(_('email address'), unique=True)
    user_name = models.CharField(max_length=150)
    is_active = models.BooleanField(default=False)
    is_staff = models.BooleanField(default=False)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)

    objects = CustomAccountManager()

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['user_name']

    class Meta:
        verbose_name = "Accounts"
```

```
    verbose_name_plural = "Accounts"

def email_user(self, subject, message):
    send_mail(
        subject,
        message,
        'l@1.com',
        [self.email],
        fail_silently=False,
    )

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

Додаток Б

Код файлу views.py:

```
from django.shortcuts import render, get_object_or_404
from .models import Category, Product


def product_all(request):
    products = Product.objects.all()
    return render(request, 'store/home.html', {'products': products})


def product_detail(request, slug):
    product = get_object_or_404(Product, slug=slug, is_active=True)
    return render(request, 'store/products/single.html', {'product': product})


def category_list(request, category_slug=None):
    category = get_object_or_404(Category, slug=category_slug)
    products = Product.objects.filter(category=category)
    return render(request, 'store/products/category.html', {'category': category,
'products': products})
```

Код файлу urls.py:

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

app_name = 'store'

urlpatterns = [
    path('', views.product_all, name='store_home'),
    path('<slug:slug>', views.product_detail, name='product_detail'),
    path('shop/<slug:category_slug>/', views.category_list, name='category_list'),
]
```

Код файлу conext_processors.py:

```
from .models import Category


def categories(request):
    return {
        'categories': Category.objects.all()
    }
```

Додаток В

Код файлу forms.py:

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

from .models import Customer


class RegistrationForm(forms.ModelForm):
    user_name = forms.CharField(label='Username', min_length=5, max_length=50)
    email = forms.EmailField(max_length=100, error_messages={'required': 'Please
enter an email'})
    password = forms.CharField(label='Пароль', widget=forms.PasswordInput)
    password2 = forms.CharField(label='Повторіть пароль',
widget=forms.PasswordInput)

    class Meta:
        model = Customer
        fields = ('user_name', 'email')

    def clean_username(self):
        user_name = self.cleaned_data['user_name'].lower()
        r = Customer.objects.filter(user_name=user_name)
        if r.count():
            raise forms.ValidationError('Логін вже існує, спробуйте інший')
        return user_name

    def clean_password2(self):
        cd = self.cleaned_data
        if cd['password'] != cd['password2']:
            raise forms.ValidationError('Паролі не співпадають')
        return cd['password2']

    def clean_email(self):
        email = self.cleaned_data['email']
        if Customer.objects.filter(email=email).exists():
            raise forms.ValidationError('Email вже існує, спробуйте інший')
        return email

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields['user_name'].widget.attrs.update(
            {'class': 'form-control mb-3', 'placeholder': 'Логін'})
        self.fields['email'].widget.attrs.update(
```

```
                {'class': 'form-control mb-3', 'placeholder': 'E-mail', 'name': 'email',
'id': 'id_email'})
            self.fields['password'].widget.attrs.update(
                {'class': 'form-control mb-3', 'placeholder': 'Пароль'})
            self.fields['password2'].widget.attrs.update(
                {'class': 'form-control', 'placeholder': 'Повторіть пароль'})


    class UserLoginForm(AuthenticationForm):
        username = forms.CharField(widget=forms.TextInput(
            attrs={'class': 'form-control mb-3', 'placeholder': 'Логін', 'id': 'login-
username'}))
        password = forms.CharField(widget=forms.PasswordInput(
            attrs={'class': 'form-control', 'placeholder': 'Пароль', 'id': 'login-
pwd'}))
```

Код файлу token.py:

```
from django.contrib.auth.tokens import PasswordResetTokenGenerator
from six import text_type


class AccountActivationTokenGenerator(PasswordResetTokenGenerator):
    def _make_hash_value(self, user, timestamp):
        return text_type(user.pk) + text_type(timestamp) +
text_type(user.is_active)


account_activation_token = AccountActivationTokenGenerator()
```

Код файлу urls.py:

```
from django.contrib.auth import views as auth_views
from django.urls import path
from account import views
from .forms import UserLoginForm

app_name = 'account'

urlpatterns = [
    path('register/', views.account_register, name='register'),
    path('activate/<slug:uidb64>/<slug:token>', views.account_activate,
name='activate'),
    path('dashboard/', views.dashboard, name='dashboard'),
    path('login/',
auth_views.LoginView.as_view(template_name='account/registration/login.html',
form_class=UserLoginForm), name='login'),
```

```
    path('logout/',     auth_views.LogoutView.as_view(next_page='/account/login/'),
name='logout'),

]
```

Код файлу views.py:

```
from django.contrib import messages
from django.contrib.auth import login, logout
from django.contrib.auth.decorators import login_required
from django.contrib.sites.shortcuts import get_current_site
from django.http import HttpResponse
from django.shortcuts import redirect, render
from django.template.loader import render_to_string
from django.utils.encoding import force_bytes, force_str
from django.utils.http import urlsafe_base64_decode, urlsafe_base64_encode
from django.core.mail import send_mail

from orders.views import user_orders
from .forms import RegistrationForm
from .models import Customer
from .token import account_activation_token


@login_required
def dashboard(request):
    orders = user_orders(request)
    return render(request,
                  'account/user/dashboard.html',
                  {'section': 'profile', 'orders': orders})


def account_register(request):
    if request.user.is_authenticated:
        return redirect('account:dashboard')

    if request.method == 'POST':
        registerForm = RegistrationForm(request.POST)
        if registerForm.is_valid():
            user = registerForm.save(commit=False)
            user.email = registerForm.cleaned_data['email']
            user.set_password(registerForm.cleaned_data['password'])
            user.is_active = False
            user.save()
            current_site = get_current_site(request)
            subject = 'Activate your Account'
```

```
                message                                                    =
render_to_string('account/registration/account_activation_email.html', {
                    'user': user,
                    'domain': current_site.domain,
                    'uid': urlsafe_base64_encode(force_bytes(user.pk)),
                    'token': account_activation_token.make_token(user),
                })
                send_mail(subject, message, 'greenspace1@ukr.net',
                          [user.email], fail_silently=False)
                return HttpResponse('Реєстрація успішна! Перейдіть в вашу поштову скриню
та активуйте акаунт')
            else:
                registerForm = RegistrationForm()
            return    render(request,    'account/registration/register.html',    {'form':
registerForm})


        def account_activate(request, uidb64, token):
            try:
                uid = force_str(urlsafe_base64_decode(uidb64))
                user = Customer.objects.get(pk=uid)
            except(TypeError, ValueError, OverflowError, user.DoesNotExist):
                user = None
            if user is not None and account_activation_token.check_token(user, token):
                user.is_active = True
                user.save()
                login(request, user)
                messages.success(request, "Ви успішно авторизувалися! Можете продовжувати
покупку")
                return redirect('account:dashboard')


            else:
                return render(request, 'account/registration/activation_invalid.html')
```

Додаток Г

Код файлу urls.py:

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

app_name = 'basket'
urlpatterns = [
    path('', views.basket_summary, name='basket_summary'),
    path('add/', views.basket_add, name='basket_add'),
    path('delete/', views.basket_delete, name='basket_delete'),
    path('update/', views.basket_update, name='basket_update'),
]
```

Код файлу views.py:

```
from django.shortcuts import render, get_object_or_404
from .basket import Basket
from store.models import Product
from django.http import JsonResponse


def basket_summary(request):
    basket = Basket(request)
    return render(request, 'basket/summary.html', {'basket': basket})


def basket_add(request):
    basket = Basket(request)
    if request.POST.get('action') == 'post':
        product_id = int(request.POST.get('productid'))
        product_qty = int(request.POST.get('productqty'))
        product = get_object_or_404(Product, id=product_id)
        basket.add(product=product, qty=product_qty)

        basketqty = basket.__len__()
        response = JsonResponse({'qty': basketqty})
        return response


def basket_delete(request):
    basket = Basket(request)
    if request.POST.get('action') == 'post':
        product_id = int(request.POST.get('productid'))
        basket.delete(product=product_id)

        basketqty = basket.__len__()
```

```
        baskettotal = basket.get_total_price()
        response = JsonResponse({'qty': basketqty, 'subtotal': baskettotal})
        return response


def basket_update(request):
    basket = Basket(request)
    if request.POST.get('action') == 'post':
        product_id = int(request.POST.get('productid'))
        product_qty = int(request.POST.get('productqty'))
        basket.update(product=product_id, qty=product_qty)

        basketqty = basket.__len__()
        basketsubtotal = basket.get_subtotal_price()
        response = JsonResponse({'qty': basketqty, 'subtotal': basketsubtotal})
        return response
```

Додаток Д

Код файлу base.html:

```
{% load static %}
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <title>{% block title %}Tree Shop{% endblock %}</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link                        href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet"
          integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1"
crossorigin="anonymous">
    <link                                                          rel="canonical"
href="https://getbootstrap.com/docs/5.2/examples/footers/">
    {#HOva POstha api#}
    <link                                                        rel="stylesheet"
href="//code.jquery.com/ui/1.12.1/themes/smoothness/jquery-ui.css">
    <script src="https://code.jquery.com/jquery-3.5.1.min.js"
            integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
crossorigin="anonymous"></script>
    <script                      src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/js/bootstrap.bundle.min.js"
            integrity="sha384-
ygbV9kiqUc6oa4msXn9868pTtWMgiQaeYH7/t7LECLbyPA2x65Kgf80OJFdroafW"
crossorigin="anonymous">
    </script>

    <link rel="stylesheet" href="{% static 'core/css/base.css' %}">
    <link rel="stylesheet" href="{% static 'basket/css/basket.css' %}">
    <svg xmlns="http://www.w3.org/2000/svg" style="display: none;">
        <symbol id="check-circle-fill" fill="currentColor" viewBox="0 0 16 16">
            <path d="M16 8A8 8 0 1 1 0 8a8 8 0 0 1 16 0zm-3.97-3.03a.75.75 0 0 0-
1.08.022L7.477 9.417 5.384 7.323a.75.75 0 0 0-1.06 1.06L6.97 11.03a.75.75 0 0 0 1.079-
.02l3.992-4.99a.75.75 0 0 0-.01-1.05z"/>
        </symbol>
    </svg>
</head>

<style>
```

```
            @import
url(https://fonts.googleapis.com/css?family=Open+Sans:400,600&subset=latin,cyrillic);
            *{box-sizing: border-box;}
            nav ul {
                margin: 0;
                padding: 0;
                list-style: none;
            }
            nav ul:after {
                content: "";
                display: table;
                clear: both;
            }
            nav a {
                text-decoration: none;
                display: block;
                transition: .3s linear;
            }
            .topmenu > li {
                float: left;
                position: relative;
                border-left: 1px solid black;
            }
            .topmenu > li:first-child {border-left: 0;}
            .topmenu > li > a {
                padding: 20px 30px;
                font-size: 12px;
                text-transform: uppercase;
                color: #FEFDFD;
                letter-spacing: 2px;
            }
            .topmenu > li > a.active,
            .submenu a:hover {color: lightskyblue;}
            .topmenu .fa,
            .submenu .fa {
                margin-left: 5px;
                color: inherit;
            }
            .submenu {
                position: absolute;
                z-index: 5;
                min-width: 200px;
                background: white;
                border-top: 1px solid #CBCBCC;
                border-left: 1px solid #CBCBCC;
                border-right: 1px solid #CBCBCC;
```

```
                visibility: hidden;
                opacity: 0;
                transform-origin: 0% 0%;
                transform: rotateX(-90deg);
                transition: .3s linear;
            }
            .submenu li {position: relative;}
            .submenu li a {
                color: #282828;
                padding: 10px 20px;
                font-size: 13px;
                border-bottom: 1px solid #CBCBCC;
            }
            .submenu .submenu {
                position: absolute;
                left: 100%;
                top: -1px;
                transition: .3s linear;
            }
            nav li:hover > .submenu {
                transform: rotateX(0deg);
                visibility: visible;
                opacity: 1;
            }
        </style>

        <body>
        <header class="pb-3">
            <nav class="navbar navbar-expand-md navbar-light bg-white border-bottom">
                <div class="container px-md-4">
                    <div class="center-block d-flex w-100 navbar-collapse">
                        <a class="navbar-brand d-flex-inline" href="/">
                            <svg xmlns="http://www.w3.org/2000/svg" width="36" height="36"
fill="currentColor" class="bi bi-tree" viewBox="0 0 16 16">
                                <path d="M8.416.223a.5.5 0 0 0-.832 0l-3 4.5A.5.5 0 0 0 5
5.5h.098L3.076 8.735A.5.5 0 0 0 3.5 9.5h.191l-1.638 3.276a.5.5 0 0 0 .447.724H7V16h2v-
2.5h4.5a.5.5 0 0 0 .447-.724L12.31 9.5h.191a.5.5 0 0 0 .424-.765L10.902 5.5H11a.5.5 0 0 0
.416-.777l-3-4.5zM6.437 4.758A.5.5 0 0 0 6 4.5h-.066L8 1.401 10.066 4.5H10a.5.5 0 0 0-
.424.765L11.598 8.5H11.5a.5.5 0 0 0-.447.724L12.69 12.5H3.309l1.638-3.276A.5.5 0 0 0 4.5
8.5h-.098l2.022-3.235a.5.5 0 0 0 .013-.507z"/>
                            </svg>
                        </a>
                        <ul class="navbar-nav me-auto mb-2 mb-md-0">
                            <li class="nav-item dropdown">
                                <a class="nav-link dropdown-toggle d-none d-md-block fw500
active" href="#" id="navbarDropdown"
```

```
                                role="button"      data-bs-toggle="dropdown"      aria-
expanded="false">
                                 Категорії
                                 <i class="ps-1">
                                     <svg xmlns="http://www.w3.org/2000/svg" width="16"
height="16" fill="currentColor" class="bi bi-chevron-down" viewBox="0 0 16 16">
                                         <path fill-rule="evenodd" d="M1.646 4.646a.5.5
0 0 1 .708 0L8 10.293l5.646-5.647a.5.5 0 0 1 .708.708l-6 6a.5.5 0 0 1-.708 0l-6-6a.5.5 0
0 1 0-.708z" />
                                     </svg>
                                 </i>
                             </a>

                             {% load mptt_tags %}
                             <ul class="submenu">
                                 {% recursetree categories %}
                                     <li    {%    if    category.slug    ==    node.slug
%}class="active" {% endif %}>
                                         <a        class="dropdown-item"        href="{{
node.get_absolute_url }}">{{ node.name|title }}</a>
                                         {% if not node.is_leaf_node %}
                                             <ul class="submenu">
                                                 <a href="">{{ children }}</a>
                                             </ul>
                                         {% endif %}
                                     </li>
                                 {% endrecursetree %}
                             </ul>
                         </li>
                     </ul>

                     <button  class="navbar-toggler  border-0"  type="button"  data-bs-
toggle="collapse"
                             data-bs-target="#navbarCollapse"                      aria-
controls="navbarCollapse" aria-expanded="false"
                             aria-label="Toggle navigation">
                         <div>
                             <svg     xmlns="http://www.w3.org/2000/svg"     width="22"
height="22" fill="currentColor"
                                  class="bi bi-list" viewBox="0 0 16 16">
                                 <path  fill-rule="evenodd"  d="M2.5  11.5A.5.5  0  0  1  3
11h10a.5.5 0 0 1 0 1H3a.5.5 0 0 1-.5-.5zm0-4A.5.5 0 0 1 3 7h10a.5.5 0 0 1 0 1H3a.5.5 0 0
1-.5-.5zm0-4A.5.5 0 0 1 3 3h10a.5.5 0 0 1 0 1H3a.5.5 0 0 1-.5-.5z" />
                             </svg>
                         </div>
                         <span class="fs15 fw500">Shop</span>
```

```
                    </button>

                    {% if user.is_authenticated %}
                        <a type="button" role="button" href="{% url "account:logout" %}"
                           class="btn btn-outline-secondary border-0 basket-btn">
                            <div>
                                <svg   xmlns="http://www.w3.org/2000/svg"   width="30"
height="30" fill="currentColor"
                                     class="bi bi-door-closed" viewBox="0 0 16 16">
                                    <path  d="M3  2a1  1  0  0  1  1-1h8a1  1  0  0  1  1
1v13h1.5a.5.5 0 0 1 0 1h-13a.5.5 0 0 1 0-1H3V2zm1 13h8V2H4v13z" />
                                    <path d="M9 9a1 1 0 1 0 2 0 1 1 0 0 0-2 0z" />
                                </svg>
                            </div>
                            <span class="fs15 fw500">Вийти</span>
                        </a>
                    {% else %}
                        <a type="button" role="button" href="{% url "account:login" %}"
                           class="btn btn-outline-secondary border-0 basket-btn">
                            <div>
                                <svg   xmlns="http://www.w3.org/2000/svg"   width="30"
height="30" fill="currentColor"
                                     class="bi bi-door-closed" viewBox="0 0 16 16">
                                    <path  d="M3  2a1  1  0  0  1  1-1h8a1  1  0  0  1  1
1v13h1.5a.5.5 0 0 1 0 1h-13a.5.5 0 0 1 0-1H3V2zm1 13h8V2H4v13z" />
                                    <path d="M9 9a1 1 0 1 0 2 0 1 1 0 0 0-2 0z" />
                                </svg>
                            </div>
                            <span class="fs15 fw500">Увійти</span>
                        </a>
                    {% endif %}

                    <a type="button" role="button" href="{% url "account:dashboard" %}"
                       class="btn btn-outline-secondary border-0 basket-btn">
                        <div>
                            <svg     xmlns="http://www.w3.org/2000/svg"     width="30"
height="30" fill="currentColor"
                                 class="bi bi-person" viewBox="0 0 16 16">
                                <path d="M8 8a3 3 0 1 0 0-6 3 3 0 0 0 0 6zm2-3a2 2 0 1
1-4 0 2 2 0 0 1 4 0zm4 8c0 1-1 1-1 1H3s-1 0-1-1 1-4 6-4 6 3 6 4zm-1-.004c-.001-.246-.154-
.986-.832-1.664C11.516  10.68  10.289  10  8  10c-2.29  0-3.516.68-4.168  1.332-.678.678-.83
1.418-.832 1.664h10z" />
                            </svg>
                        </div>
                    </a>
```

```
                    <a    type="button"    id="cart"    role="button"    href="{%    url
"basket:basket_summary" %}"
                       class="btn btn-outline-secondary border-0 basket-btn">
                        {% with total_qty=basket|length %}
                            <div id="basket-qty" class="basket-qty">
                                {% if total_qty > 0 %}
                                    {{ total_qty }}
                                {% else %}
                                    0
                                {% endif %}
                            </div>
                        {% endwith %}
                        <div>
                            <svg     xmlns="http://www.w3.org/2000/svg"     width="30"
height="30" fill="currentColor"
                                 class="bi bi-cart3" viewBox="0 0 16 16">
                                <path  d="M0  1.5A.5.5  0  0  1  .5  1H2a.5.5  0  0  1
.485.379L2.89  3H14.5a.5.5  0  0  1  .49.598l-1  5a.5.5  0  0  1-.465.401l-9.397.472L4.415
11H13a.5.5 0 0 1 0 1H4a.5.5 0 0 1-.491-.408L2.01 3.607 1.61 2H.5a.5.5 0 0 1-.5-.5zM3.102
4l.84 4.479 9.144-.459L13.89 4H3.102zM5 12a2 2 0 1 0 0 4 2 2 0 0 0 0-4zm7 0a2 2 0 1 0 0 4
2 2 0 0 0 0-4zm-7 1a1 1 0 1 1 0 2 1 1 0 0 1 0-2zm7 0a1 1 0 1 1 0 2 1 1 0 0 1 0-2z" />
                            </svg>
                        </div>
                    </a>
                </div>
                <div class="d-md-none d-lg-none d-xl-none">
                    <div class="collapse navbar-collapse" id="navbarCollapse">
                        <ul class="navbar-nav me-auto mb-2 mb-md-0">
                            <li><a       class="dropdown-item"       href="{%       url
"store:store_home" %}">All</a></li>
                            {% for c in categories %}
                                <li {% if category.slug == c.slug %}class="selected" {%
endif %}>
                                    <a          class="dropdown-item"          href="{{
c.get_absolute_url }}">{{ c.name|title }}</a>
                                </li>
                            {% endfor %}
                        </ul>
                    </div>
                </div>
                <form class="d-flex w-100 d-md-none">
                    <input class="form-control me-2" type="search" placeholder="Search
products or FAQ"
                           aria-label="Search">
                    <button              class="btn              btn-outline-secondary"
type="submit">Search</button>
```

```
                </form>
            </div>
        </nav>

        {% if messages %}
            {% for message in messages %}
                <div    class="alert    alert-success    d-flex    align-items-center"
role="alert">
                    <svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img"
aria-label="Success:"><use xlink:href="#check-circle-fill"/></svg>
                    <div>
                        {{ message }}
                    </div>
                </div>
            {% endfor %}
        {% endif %}

    </header>
        <div id="content">{% block content %} {% endblock %}</div>
    </body>

    <svg xmlns="http://www.w3.org/2000/svg" style="display: none;">
        <symbol id="facebook" viewBox="0 0 16 16">
            <path d="M16 8.049c0-4.446-3.582-8.05-8-8.05C3.58 0-.002 3.603-.002 8.05c0
4.017  2.926  7.347  6.75  7.951v-5.625h-2.03V8.05H6.75V6.275c0-2.017  1.195-3.131  3.022-
3.131.876 0 1.791.157 1.791.157v1.98h-1.009c-.993 0-1.303.621-1.303 1.258v1.51h2.218l-.354
2.326H9.25V16c3.824-.604 6.75-3.934 6.75-7.951z"/>
        </symbol>
        <symbol id="instagram" viewBox="0 0 16 16">
            <path d="M8 0C5.829 0 5.556.01 4.703.048 3.85.088 3.269.222 2.76.42a3.917
3.917 0 0 0-1.417.923A3.927 3.927 0 0 0 .42 2.76C.222 3.268.087 3.85.048 4.7.01 5.555 0
5.827 0 8.001c0 2.172.01 2.444.048 3.297.04.852.174 1.433.372 1.942.205.526.478.972.923
1.417.444.445.89.719 1.416.923.51.198 1.09.333 1.942.372C5.555 15.99 5.827 16 8 16s2.444-
.01 3.298-.048c.851-.04 1.434-.174 1.943-.372a3.916 3.916 0 0 0 1.416-.923c.445-.445.718-
.891.923-1.417.197-.509.332-1.09.372-1.942C15.99 10.445 16 10.173 16 8s-.01-2.445-.048-
3.299c-.04-.851-.175-1.433-.372-1.941a3.926 3.926 0 0 0-.923-1.417A3.911 3.911 0 0 0
13.24.42c-.51-.198-1.092-.333-1.943-.372C10.443.01 10.172 0 7.998 0h.003zm-.717
1.442h.718c2.136 0 2.389.007 3.232.046.78.035 1.204.166
1.486.275.373.145.64.319.92.599.28.28.453.546.598.92.11.281.24.705.275 1.485.039.843.047
1.096.047 3.231s-.008 2.389-.047 3.232c-.035.78-.166 1.203-.275 1.485a2.47 2.47 0 0 1-
.599.919c-.28.28-.546.453-.92.598-.28.11-.704.24-1.485.276-.843.038-1.096.047-3.232.047s-
2.39-.009-3.233-.047c-.78-.036-1.203-.166-1.485-.276a2.478 2.478 0 0 1-.92-.598 2.48 2.48
0 0 1-.6-.92c-.109-.281-.24-.705-.275-1.485-.038-.843-.046-1.096-.046-3.233 0-2.136.008-
2.388.046-3.231.036-.78.166-1.204.276-1.486.145-.373.319-.64.599-.92.28-.28.546-.453.92-
.598.282-.11.705-.24 1.485-.276.738-.034 1.024-.044 2.515-.045v.002zm4.988 1.328a.96.96 0
```

```
1 0 0 1.92.96.96 0 0 0 0-1.92zm-4.27 1.122a4.109 4.109 0 1 0 0 8.217 4.109 4.109 0 0 0 0-8.217zm0 1.441a2.667 2.667 0 1 1 0 5.334 2.667 2.667 0 0 1 0-5.334z"/>
            </symbol>
            <symbol id="twitter" viewBox="0 0 16 16">
                <path d="M5.026 15c6.038 0 9.341-5.003 9.341-9.334 0-.14 0-.282-.006-.422A6.685 6.685 0 0 0 16 3.542a6.658 6.658 0 0 1-1.889.518 3.301 3.301 0 0 0 1.447-1.817 6.533 6.533 0 0 1-2.087.793A3.286 3.286 0 0 0 7.875 6.03a9.325 9.325 0 0 1-6.767-3.429 3.289 3.289 0 0 0 1.018 4.382A3.323 3.323 0 0 1 .64 6.575v.045a3.288 3.288 0 0 0 2.632 3.218 3.203 3.203 0 0 1-.865.115 3.23 3.23 0 0 1-.614-.057 3.283 3.283 0 0 0 3.067 2.277A6.588 6.588 0 0 1 .78 13.58a6.32 6.32 0 0 1-.78-.045A9.344 9.344 0 0 0 5.026 15z"/>
            </symbol>
        </svg>

        <div class="b-example-divider"></div>
        <div class="container">
            <footer class="d-flex flex-wrap justify-content-between align-items-center py-3 my-4 border-top">
                <div class="col-md-4 d-flex align-items-center">
                    <span class="mb-3 mb-md-0 text-muted">&copy; 2022 Tree Shop, Inc</span>
                </div>
                <ul class="nav col-md-4 justify-content-end list-unstyled d-flex">
                    <li class="ms-3"><a class="text-muted" href="#"><svg class="bi" width="24" height="24"><use xlink:href="#twitter"/></svg></a></li>
                    <li class="ms-3"><a class="text-muted" href="#"><svg class="bi" width="24" height="24"><use xlink:href="#instagram"/></svg></a></li>
                    <li class="ms-3"><a class="text-muted" href="#"><svg class="bi" width="24" height="24"><use xlink:href="#facebook"/></svg></a></li>
                </ul>
            </footer>
        </div>
        </html>
```

Код файлу home.html:

```
        {% extends "store/base.html" %}
        {% load static %}
        {% block title %}GreenStore{% endblock %}
        {% block content %}

            <section class="py-5 text-center container">
                <div class="row py-md-3.bg-image" style="background-image: url(https://images.unsplash.com/photo-1464036388609-747537735eab?ixlib=rb-1.2.1&raw_url=true&q=80&fm=jpg&crop=entropy&cs=tinysrgb&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=1470);">
                    <div class="col-lg-6 col-md-8 mx-auto">

                        <h1 class="h1 fw-bold">Plant, Care, Enjoy</h1>
```

```
                    <p class="lead text-muted">Trees never stop giving to us. That’s why
the Arbor Day Foundation has
                        never stopped working for them. Find out how we plan to answer
some of humanity’s greatest
                        challenges over the next 5 years.</p>
                </div>
            </div>
        </section>

        <div class="album py-5 bg-light">
            <div class="container">

                <div class="pb-3 h5">Популярні товари</div>
                <div class="row row-cols-1 row-cols-sm-2 row-cols-md-5 g-3">

                {% for product in products %}

                <div class="col">
                    <div class="card shadow-sm">
                            {% for image in product.product_image.all %}
                                {% if image.is_feature %}
                                    <img   class="img-fluid"   alt="Responsive   image"
src="{{ image.image.url }}">
                                {% endif %}
                            {% endfor %}
                        <div class="card-body">
                         <p class="card-text">
                            <a   class="text-dark   text-decoration-none   "   href="{{
product.get_absolute_url }}">{{ product.title }}</a>
                        </p>
                            <hr>
                            <p class="card-text"> {{ product.regular_price }} грн</p>
                        <div    class="d-flex    justify-content-between    align-items-
center"></div>
                        </div>
                    </div>
                </div>
                {% endfor %}
                </div>
            </div>
        </div>


    {% endblock %}
```

Код файлу single.html:

```
{% extends "../base.html" %}

{% block content %}
    <main class="pt-2">

        <div class="container" style="max-width: 1000px">

            <div class="row g-3">
                <div class="col-12 d-flex justify-content-between"></div>

                <div class="col-md-5 ps-3 col-lg-5 order-md-last p-0 order-1">
                    <div class="d-grid gap-2">

                        <div class="d-flex d-flex-inline justify-content-between">
                            <div><span   class="h3">{{   product.regular_price   }}
грн</span><p class="text-muted"></p></div>
                            <div class="">
                                <label for="select">Кіл-ть</label>
                                <select           id="select"           class="h6"
style="width:50px;height:40px;">
                                    <option value="1">1</option>
                                    <option value="2">2</option>
                                    <option value="3">3</option>
                                    <option value="4">4</option>
                                    <option value="5">5</option>
                                    <option value="6">6</option>
                                </select>
                            </div>
                        </div>
                        <hr>
                        <button            type="button"            id="add-button"
value="{{product.id}}" class="btn btn-success fw500">Додати до кошика</button>
                    </div>
                </div>

                <div class="col-md-7 col-lg-7 p-0">



                    <div class="card mb-3 border-0">
                        <div class="row g-0">
                            <div class="col-md-12">
                                <div class="card-body p-1">
                                    <h1 class="mb-0 h2 pe-4 pb-4">{{ product.title
}}</h1>
                                    <div class="bg-light">
```

```
                                                {% for image in product.product_image.all
%}
                                                    {% if image.is_feature %}
                                                        <img class="img-fluid mx-auto d-
block" width="200px" alt="Responsive image" src="{{ image.image.url }}">
                                                    {% else %}
                                                        <img class="img-fluid d-block-
inline pt-3" width="100px" alt="Responsive image" src="{{ image.image.url }}">
                                                    {% endif %}
                                                {% endfor %}
                                            </div>

                                        </div>

                                    </div>
                                </div>
                            </div>
                        </div>


                    </div>
                    <div class="col-12 d-flex justify-content-between">

                    </div>
                    <div class="container" style="max-width: 1000px">
                        <hr>
                        <div class="card shadow">
                                <div class="card-body">
                                    <h5>Опис</h5><br>
                                    {{ product.description }}
                                </div>
                        </div>
                    </div>
                </div>
            </main>



            <script>
                $(document).on('click', '#add-button', function(e) {
                    e.preventDefault();
                    $.ajax({
                        type: 'POST',
                        url: '{% url "basket:basket_add" %}',
                        data: {
                            productid: $('#add-button').val(),
```

```
                productqty: $('#select option:selected').text(),
                csrfmiddlewaretoken: "{{ csrf_token }}",
                action: 'post',
            },
            success: function(json){
                document.getElementById('basket-qty').innerHTML = json.qty
            },
            error: function (xhr, errmsg, err){}
        });
    })
</script>

{% endblock %}
```

Код файлу category.html:

```
{% extends "../base.html" %}
{% load static %}
{% block title %}
{% if category %}{{ category.name }}{% else %}Products{% endif %}
{% endblock %}
{% block content %}


    <div class="album py-5 bg-light">
        <div class="container">

            <div class="pb-3 h5">{{ category.name|title }}</div>
            <div class="row row-cols-1 row-cols-sm-2 row-cols-md-5 g-3">

            {% for product in products %}

            <div class="col">
                <div class="card shadow-sm">
                    {% for image in product.product_image.all %}
                        {% if image.is_feature %}
                            <img class="img-fluid" alt="Responsive image" src="{{
image.image.url }}">
                        {% endif %}
                    {% endfor %}
                    <div class="card-body">
                     <p class="card-text">
                        <a    class="text-dark    text-decoration-none"    href="{{
product.get_absolute_url }}">{{ product.title }}</a>
                    </p>
                        <hr>
```

```
                            <p class="card-text">{{ product.regular_price }} грн</p>
                        <div    class="d-flex    justify-content-between    align-items-
center"></div>
                        </div>
                    </div>
                </div>

            {% endfor %}

                </div>
            </div>
        </div>

    {% endblock %}
```

Код файлу checkout.html:

```
{% extends "store/base.html" %}
{% load static %}
{% block title %}Basket Summary{% endblock %}

{% block content %}
    <style>
            .checkoutform label {
                font-size:16px !important;
                font-weight:600;
            }
            .checkoutform input {
                font-size:14px !important;
            }

        </style>

    <main class="pt-3">
    <div class="container">
        <form action="{% url 'checkout:placeorder' %}" method="post">
            {% csrf_token %}
            <div class="row">
                <div class="col-md-7">
                    <div class="card shadow">
                        <div class="card-body">
                            <h5 class="fw-bold">Платіжні дані</h5>
                            <hr>
                            <div class="row checkoutform">
                                <div class="col-md-6">
                                    <label>Ім'я</label>
```

```
                                        <input   type="text"   required   class="form-
control" name="fname">
                                    </div>
                                    <div class="col-md-6">
                                        <label>Прізвище</label>
                                        <input   type="text"   required   class="form-
control" name="lname">
                                    </div>
                                    <div class="col-md-6 mt-2">
                                        <label>Адреса електронної пошти</label>
                                        <input   type="email"   required   class="form-
control" name="email" placeholder="email@gmail.com">
                                    </div>
                                    <div class="col-md-6 mt-2">
                                        <label>Телефон</label>
                                        <input   type="text"   required   class="form-
control" name="phone" placeholder="+380 -- --- -- --">
                                    </div>
                                    <div class="col-md-12">
                                        <br>
                                        <h5 class="fw-bold">Доставка здійснюється Новою
поштою</h5>
                                        <hr>
                                    </div>
                                    <div class="col-md-6 mt-2">
                                            <label>Місто</label>
                                            <input  id="cities"  type="text"  required
name="city" class="form-control" placeholder="Почніть вводити назву">
                                            <div id ="cityHint"></div>
                                            <div class="invalid-feedback">
                                                Оберіть місто
                                            </div>
                                        </div>
                                        <div class="col-md-6 mt-2">
                                            <label>Відділення</label>
                                            <select  id="warehouses"  name="warehouse"
class="form-control" required></select>
                                            <div class="invalid-feedback">
                                                Please select a valid country.
                                            </div>
                                        </div>


                                    <div class="col-md-12 mt-2">
                                        <input   type="hidden"   name="payment_option"
value="COD">
```

```
                                </div>
                            </div>
                        </div>
                    </div>

                    <div class="col-md-5">
                        <div class="card shadow">
                            <div class="card-body">
                                <h5 class="fw-bold">Ваше замовлення</h5>
                                <hr>
                                <table class="table table-stripeds table-bordereds">
                                    <thead>
                                        <tr>
                                            <th>Товар</th>
                                            <th>Кількість</th>
                                            <th>Сума</th>
                                        </tr>
                                    </thead>
                                    <tbody>
                                        {% for item in basket %}
                                             {% with product=item.product %}
                                        <tr>
                                            <td>
                                                {%        for        image        in
product.product_image.all %}
                                                    {% if image.is_feature %}
                                                        <img           class="me-2"
alt="Responsive image" src="{{ image.image.url }}" height="50px" width="50px">
                                                    {% endif %}
                                                {% endfor %}
                                                {{ product.title }}
                                            </td>
                                            <td>{{item.qty}}</td>
                                            <td>{{ item.total_price }} грн</td>
                                        </tr>
                                        {% endwith %}
                                        {% endfor %}
                                    </tbody>
                                </table>
                                <h6 class="fw-bold">До сплати
                                    <span                              class="float-end">
{{basket.get_subtotal_price}} грн </span>
                                </h6>
                                <hr>
                                <h6 class="fw-bold">Оплата при отриманні</h6>
```

```
                                    <div class="mt-3">
                                    <button  type="submit"  class="btn  btn-success  w-100">
Сплатити </button>
                                </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </form>
            </div>
            </main>

            <script src="/static/checkout/novaPoshta.js"></script>


        {% endblock %}
```

Код файлу summary.html:

```
        {% extends "store/base.html" %}
        {% load static %}
        {% block title %}Basket Summary{%endblock %}
        {% block content %}

            <div class="container">
                <div class="col-12">
                    <h1 class="h2">Корзина</h1>
                </div>
                <hr />
            </div>
            <div class="container">
                <div class="row">
                    {% if basket|length == 0 %}
                        <div   class="col-12">Ваша   корзина   порожня   <a   href="{%   url
'store:store_home' %}"> <br> Перейти в магазин</a></div>
                    {% else %}
                        <div class="col-md-7">
                            <div class="card shadow">
                                <div class="card-body">
                                    <table class="table table-stripeds table-bordereds">
                                        <div class="card mb-3 border-0 product-item" data-
index="{{product.id}}">
                                            <thead>
                                            <tr>
                                                <th>Товар</th>
                                                <th>Кількість</th>
                                                <th>Ціна</th>
```

```
                                        </tr>
                                        </thead>
                                        <tbody>
                                        {% for item in basket%}
                                            {% with product=item.product %}
                                                <tr>

                                                    <div>
                                                        <td>
                                                            {%     for     image     in
product.product_image.all %}
                                                                {% if image.is_feature
%}
                                                                    <img   class="me-2"
alt="Responsive image" src="{{ image.image.url }}" height="50px" width="50px">
                                                                {% endif %}
                                                            {% endfor %}
                                                            {{ product.title }}
                                                        </td>
                                                        <td>
                                                            <label    for="select">Кіл-
ть</label>
                                                            <select
id="select{{product.id}}">
                                                                <option selected>
                                                                    {{item.qty}}
                                                                </option>
                                                                <option
value="">1</option>
                                                                <option
value="">2</option>
                                                                <option
value="">3</option>
                                                                <option
value="">4</option>
                                                                <option
value="">5</option>
                                                                <option
value="">6</option>
                                                            </select>
                                                            <button       type="button"
id="update-button" data-index="{{product.id}}"
                                                                    class="btn     btn-
outline-secondary btn-sm update-button">
                                                                Оновити
                                                            </button>
```

```
                                                                    <button       type="button"
id="delete-button" data-index="{{product.id}}"
                                                                            class="btn     btn-
outline-secondary btn-sm delete-button">
                                                                        Видалити
                                                                    </button>
                                                                </td>
                                                                <td>{{ item.price }} грн</td>
                                                            </div>
                                                        </tr>
                                                    {% endwith %}
                                                {% endfor %}
                                                </tbody>
                                            </div>
                                        </table>
                                    </div>
                                </div>
                            </div>

                            <div class="col-md-5">
                                <div class="card shadow">
                                    <div class="card-body">
                                        <div class="col-12 bg-light">
                                            <h5>Ваше замовлення майже готове, перейдіть далі для
оформлення</h5>
                                            <hr>
                                            <div class="text-end">
                                                <div   class="">До   сплати:   <span   class="fw-
bold"></span>
                                                    <div id="subtotal" class="d-inline-flex fw-
bold">{{basket.get_subtotal_price}} грн</div>
                                                    <hr>
                                                </div>
                                            </div>
                                        </div>
                                        <div class="row">
                                            <div class="d-grid gap-2 ">
                                                <a       role="button"       href="{%       url
'checkout:checkout'   %}"   class="btn   btn-success   fw-bold"   type="button">Оформити
замовлення</a>
                                            </div>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        {% endif %}
```

```
    </div>
</div>


<script>
    // Delete Item
    $(document).on('click', '.delete-button', function (e) {
        e.preventDefault();
        var prodid = $(this).data('index');
        $.ajax({
            type: 'POST',
            url: '{% url "basket:basket_delete" %}',
            data: {
                productid: $(this).data('index'),
                csrfmiddlewaretoken: "{{csrf_token}}",
                action: 'post'
            },
            success: function (json) {
                $('.product-item[data-index="' + prodid + '"]').remove();
                document.getElementById("subtotal").innerHTML = json.subtotal;
                document.getElementById("basket-qty").innerHTML = json.qty
                document.location.reload()
            },
            error: function (xhr, errmsg, err) {}
        });
    })

    // Update Item
    $(document).on('click', '.update-button', function (e) {
        e.preventDefault();
        var prodid = $(this).data('index');
        $.ajax({
            type: 'POST',
            url: '{% url "basket:basket_update" %}',
            data: {
                productid: $(this).data('index'),
                productqty: $('#select' + prodid + ' option:selected').text(),
                csrfmiddlewaretoken: "{{csrf_token}}",
                action: 'post'
            },
            success: function (json) {
                document.getElementById("basket-qty").innerHTML = json.qty
                document.getElementById("subtotal").innerHTML = json.subtotal
            },
            error: function (xhr, errmsg, err) {}
        });
```

```
            })
        </script>

    {% endblock %}
```

Код файлу dashboard.html:

```
    {% extends "../../store/base.html" %} {% load static %} {% block title
%}Dashboard{%endblock %} {% block content %}

        <main class="pt-5">
            <div class="container" style="max-width: 1000px">
                <div class="col-12">
                    <h1 class="h2">Привіт, {{ user.user_name }}! Це твої попередні
замовлення</h1>
                </div>
                <hr>
            </div>

            <div class="container" style="max-width: 1000px">
                {% for order in orders %}
                    <div class="row g-3">
                        <div class="col-12 bg-light p-3 d-flex justify-content-
between">
                            <div class="d-flex d-flex-inline">
                                <div class="pe-3">{{ order.created }}</div>
                            </div>
                            <div class="text-end">
                                Всього до сплати: <span class="fw-bold">{{
order.total_paid }} грн</span>
                            </div>
                        </div>
                        <div class="col-md-5 col-lg-4 order-md-last p-0 order-3">
                            <div class="d-grid gap-2 ">
                                <h3>Дані отримувача</h3>
                                <hr>
                                <p>Ім'я: {{order.first_name}}</p>
                                <p>Прізвище: {{order.surname}}</p>
                                <p>Місто: {{order.city}}</p>
                                <p>Відділення: {{order.warehouse}}</p>
                            </div>
                        </div>

                        <div class="col-md-7 col-lg-8 p-0">
                            {% for item in order.items.all %}
                                <div class="card mb-3 border-0">
```

```
                                        <div class="row g-0">
                                            {% for image in item.product.product_image.all
%}
                                                {% if image.is_feature %}
                                                    <div   class="col-md-2   d-none   d-md-
block">
                                                        <img              class="img-fluid"
style="width:90px" src="{{ image.image.url }}"/>
                                                    </div>
                                                {% endif %}
                                            {% endfor %}
                                            <div class="col-md-10">
                                                <div class="card-body p-1">
                                                    <a         class="text-decoration-none"
href="{{ item.product.get_absolute_url }}">
                                                        <p                 class="card-text
small">{{item.product|title}}</p>
                                                    </a>
                                                </div>
                                            </div>
                                        </div>
                                    </div>
                                {% endfor %}
                            </div>
                        </div>
                    {% endfor %}
                </div>
            </main>
        {% endblock %}
```

Код файлу register.html:

```
{% extends "../../store/base.html" %}
{% block title %}Sign-up{% endblock %}
{% block content %}
    <style>
        .account-form input {
            border: 2px solid #ccc;
            height: calc(2em + .75rem + 2px);
        }

        .account-form input:focus {
            border-color: #1497ff;
            box-shadow: none;
        }
    </style>
```

```
        <div class="container-fluid">
            <div class="row no-gutter">
                <div class="col-md-12">
                    <div class="login d-flex align-items-center py-5">
                        <div class="container">
                            <div class="row">
                                <div class="col-12 col-lg-7  mx-auto">
                                    <form class="account-form p-4 rounded col-lg-10 mx-
auto" action="." method="post">
                                        {% csrf_token %}
                                        <h3   class="mb-2   font-weight-bold">Створити
акаунт</h3>
                                        <p  class="mb-4">Це  безкоштовно  та  не  займе
забагото часу.</p>

                                        {% if form.errors %}
                                            <p>Виправте ці помилки:</p>
                                            {% for field in form %}
                                                {% if field.errors %}
                                                    <div  class="alert  alert-primary"
role="alert">
                                                        {{    field.label    }}:    {{
field.errors|striptags }}
                                                    </div>
                                                {% endif %}
                                            {% endfor %}
                                        {% endif %}
                                        {{ form }}
                                        <button class="btn btn-primary btn-block py-2
mb-4 mt-5 fw500 w-100" type="submit">Зареєструватися</button>
                                        <p class="text-center">
                                            <a  href="{%  url  'account:login'   %}">Вже
маєте акаунт?</a>
                                        </p>
                                    </form>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        {% endblock %}
```

Код файлу login.html:

```
{% extends "../../store/base.html" %}
{% block title %}Log-in{% endblock %}
{% block content %}

    <div class="container-fluid">
        <div class="row no-gutter">
            <div class="col-md-12">
                <div class="login d-flex align-items-center py-5">
                    <div class="container">
                        <div class="row">
                            <div class="col-12 col-lg-6  mx-auto">
                                <form class="account-form p-4 rounded" action="{%
url 'account:login' %}" method="post">

                                    {% csrf_token %}
                                    <h3 class="mb-4">Вхід</h3>

                                    {% if form.errors %}
                                        <div      class="alert      alert-primary"
role="alert">
                                            Error:   Username   or   Password   not
correct!
                                        </div>
                                    {% endif %}

                                    <label                    class="form-label">{{
form.username.label}}</label>
                                    {{ form.username}}

                                    <label                    class="form-label">{{
form.password.label}}</label>
                                    {{ form.password}}
                                    <div class="d-grid gap-2">
                                        <input type="hidden" name="next" value="{{
next }}">
                                        <button  class="btn  btn-primary  py-2  mb-4
mt-5 fw-bold" type="submit" value="Log-in">Увійти
                                        </button>
                                    </div>
                                    <p class="text-center pb-3">
                                        <a    href="{%    url    "account:register"
%}">Створити акаунт</a>
                                    </p>
                                </form>
```

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

Код файлу account_activation_email.html:

```
        {% autoescape off %}
        Привіт {{ user.user_name }},

            Ваш обліковий запис успішно створений! Перейдіть за посиланням нижче для активації!

            http://{{ domain }}{% url 'account:activate' uidb64=uid token=token %}

            Дякуємо, що обрали саме нас!
        {% endautoescape %}
```

Додаток Е

Код файлу novaPoshta.js

```
const apiKey = ‘apikey’;
const url = 'https://api.novaposhta.ua/v2.0/json/';

const cities = JSON.stringify({"modelName": "Address","calledMethod": "getCities","apiKey": apiKey});

function getNPData(url,data) {
    let citiesArr = [];
    let xhr = new XMLHttpRequest();
    xhr.open("POST", url, true);
    xhr.setRequestHeader("Content-Type", "application/json");
    xhr.onreadystatechange = function () {
        if (xhr.readyState === 4 && xhr.status === 200) {
            var json = JSON.parse(xhr.responseText);

            for (cityId in json.data){
                citiesArr.push(json.data[cityId]['DescriptionRu']);
            }
        }
    };
    xhr.send(data);
    return citiesArr;
}

var citiesNamesArr = getNPData(url,cities);
console.log(citiesNamesArr);

function down(obj) {
    var reg = new RegExp('^' + obj.value, 'i'),
        t = document.getElementById('cityHint');
    t.innerHTML = '';
    t.style.visibility = "hidden";
    if (obj.value.length > 0)
        for (var i = 0; i < citiesNamesArr.length; i++) {
            if (reg.test(citiesNamesArr[i])) {
                t.innerHTML = citiesNamesArr[i];
                t.style.visibility = "visible";
                break;
            }
        }
}
```

```
    function copy() {
        var a = document.getElementById('cities').value = document.getElementById('cityHint').innerHTML;
        document.getElementById('cityHint').innerHTML = '';
        document.getElementById("cityHint").style.visibility = "hidden";

        let city = document.getElementById('cities').value;
        console.log(city);
        const warehouses = JSON.stringify({'modelName': 'AddressGeneral','calledMethod': 'getWarehouses',"methodProperties": {"CityName": city},'apiKey': apiKey});
        getNPWarehouses(url,warehouses);
    }

    let citiesElement = document.querySelector("#cities");
    citiesElement.addEventListener("input", showWarehouses, false);

    let warehousesElement = document.querySelector("#warehouses");

    let cityHint = document.querySelector("#cityHint");
    cityHint.addEventListener("click", copy, false);

    function showWarehouses(){
        down(this);
    }

    function getNPWarehouses(url,data) {
        warehousesElement.innerHTML = '';
        let citiesArr = [];
        let xhr = new XMLHttpRequest();
        xhr.open("POST", url, true);
        xhr.setRequestHeader("Content-Type", "application/json");
        xhr.onreadystatechange = function () {
            if (xhr.readyState === 4 && xhr.status === 200) {
                var json = JSON.parse(xhr.responseText);

                for (cityId in json.data){
                    warehousesElement.innerHTML += "<option value='"+ json.data[cityId]['DescriptionRu'] +"'>" + json.data[cityId]['DescriptionRu'] + "</option>";
                }

            }
        };
        xhr.send(data);
        console.log(citiesArr);
```

```
    return citiesArr;
}
```