Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Лисяка Дмитра Олександровича*
(ПІБ)

академічної групи *122-18-3*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка платформи онлайн-оголошень, що об'єднує людей для покупки, продажу товарів та послуг з використанням серверного JavaScript*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України

НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » __ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-3* (група) *Лисяка Дмитра Олександровича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка платформи онлайн-оголошень, що об'єднує людей для покупки, продажу товарами та послугами з використанням серверного JavaScript*

затверджена наказом ректора НТУ «ДП» від 18 травня 2022 р. № 268-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проектно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц. Спірінцев В.В*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Лисяк Д.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

ЗМІСТ

РЕФЕРАТ

Пояснювальна записка: 91с., 35 рис., 3 додатки, 23 джерела.

Об'єкт розробки: платформа онлайн-оголошень, що об'єднує людей для покупки, продажу товарами та послугами. Тематика: квіти.

Мета кваліфікаційної роботи: створення платформи для онлайн купівлі / продажу товарів і послуг формат: квіти. Користувачі платформи можуть розміщувати свої оголошення, попередньо зареєструвавшись за допомогою електронної пошти. За замовчуванням оголошення на платформі відображаються за датою їх розміщення.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточняється постановка завдання.

У першому розділі кваліфікаційної роботи проаналізовані загальні відомості з предметної галузі, визначено актуальність поставленої задачі, галузь застосування та проаналізовані задані вимоги до програмного виробу.

Другий розділ містить детальний опис інтерфейсу програми й опис використаних технологій та мов програмування, проаналізовані її функціональні можливості, здійснено проектування та розробка програми.

У третьому розділі визначено трудомісткість розробки програмного забезпечення, підраховані витрати на створення програмного забезпечення і приблизний час на його розробку.

Список ключових слів: КЛІЄНТ, ВЕБСАЙТ, ВЕБ-ДОДАТОК, БАЗА ДАННИХ, ШАБЛОНІЗАТОР, ФРЕЙМВОРК, ІНТЕРНЕТ, ОНЛАЙН ПРОДАЖІ.

ABSTRACT

Explanatory note: 91 pages, 35 pictures, 3 appendices, 23 sources.

Object of development: an online advertising platform that unites people to buy, sell goods and services.

The purpose of the qualification work: creating platforms for online buying / selling goods and services format: flowers. User platforms can publish their ads by pre-registering via email. By default, the ads on the platform are displayed for the given placement.

In the introductory analysis and the current state of the problem, the meta-qualification work and the field of its application are specified, the substantiation of the urgency of the topic is given and the statement of the task is specified.

In the first section of the qualification work the general information on the subject area is analyzed, the urgency of the task, the field of application is determined and the set requirements to the software product are analyzed.

Another section describes the interface of programs and described programs and programming, analyzes its functionality, detailed design and development of programs.

The third section identifies the complexity of software development, estimates the cost of creating software and the near time for its development.

Keywords: CLIENT, WEBSITE, WEB ADDITION, DATABASE, TEMPLATER, FRAMEWORK, INTERNET, ONLINE SALES.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ - Програмне забезпечення;

ІС - Інформаційна система;

API - Programming Interface;

БД - База даних;

UI – user interface;

СУБД - Система управління базами даних;

CSS – CascadingStyleSheets;

HTML -Hyper Text Markup Language.

ВСТУП

Метою кваліфікаційної роботи становить створення платформи онлайн оголошень – процес реалізації фізичних товарів, яка надає змогу дистанційно оформити замовлення, або ж розмістити оголошення. Тематика платформи онлайн-оголошень – квіти. Дана робота відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених виробничих функцій.

Розроблена платформа онлайн-оголошень являє собою не просто сторінку-візитку, а є повноцінним сервісом, що дозволяє реалізувати товари без наявності торгових точок будь-кому, абсолютно безкоштовно. Дошка оголошень дає можливість активно «розкручувати» продукти в інтернеті, що є гарантом успішного бізнесу, тож можна сказати, що дошка оголошень – трамплін для малого, середнього , іноді навіть великого бізнесу.

Приємний інтерфейс, простота у використанні, можливість необмежено робити оголошення або замовлення в любу годину дня – головні функції даної платформи. Навіть недосвідчені користувачі можуть з легкістю користуватися веб-застосунком. Усі перелічені можливості та переваги даного веб-застосунку не викликають жодного сумніву щодо його актуальності.

Для досягнення поставленої мети бути вирішені такі основні задачі:

- Проведений аналіз предметної галузі;
- Проведений аналіз організації роботи та основних виконуваних функцій;
- Проведений аналіз наявних аналогів;
- На основі результатів аналізів висунув основні вимоги до розробки;
- Створив алгоритм роботи веб сайту.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Всесвітня павутина - розподілена система, що надає доступ до пов'язаних між собою документів, розташованих на різних комп'ютерах, підключених до Інтернету. Всесвітнє павутиння утворюють сотні мільйонів веб-серверів. Більшість ресурсів Всесвітнього павутиння засновані на технології гіпертексту. Гіпертекстові документи, що розміщуються у Всесвітній мережі, називаються веб-сторінками. Декілька веб-сторінок, об'єднаних спільною темою, дизайном, а також пов'язаних між собою посиланнями і зазвичай знаходяться на тому самому веб-сервері, називаються веб-сайтом. Для завантаження та перегляду веб-сторінок використовуються спеціальні програми – браузери.

Всесвітня павутина викликала справжню революцію в інформаційних технологій та вибух у розвитку Інтернету. У повсякденному мовленні, говорячи про Інтернет, часто мають на увазі саме Всесвітнє павутиння. Однак важливо розуміти, що це не одне й те саме.

Всесвітнє павутиння утворюють мільйони веб-серверів мережі Інтернет, розташованих у всьому світі. Веб-сервер - це комп'ютерна програма, що запускається на підключеному до мережі комп'ютері та використовує протокол HTTP для передачі даних. У найпростішому вигляді така програма отримує по мережі HTTP-запит на певний ресурс, знаходить відповідний файл на локальному жорсткому диску і відправляє його по мережі комп'ютеру. Більш складні веб-сервери здатні у відповідь на HTTP-запит динамічно генерувати документи за допомогою шаблонів та сценаріїв.

Для перегляду інформації, отриманої від веб-сервера, клієнтському

комп'ютері застосовується спеціальна програма- Веб-браузер. Основна функція веб-браузера – відображення гіпертексту. Всесвітня павутина нерозривно пов'язана з поняттями гіпертексту та гіперпосилання. Більшість інформації в Інтернеті є саме гіпертекст.

Сайти – це одна із багатьох сходинок, яка рухає світ до «технологічного прогресу». У даний час кількість нових сайтів збільшується з неймовірною швидкістю, саму тому, основним завдання розробників при створенні сайту – зробити його максимально привабливим для потенційних користувачів і надати йому індивідуальність.

На початку розвитку інтернету можна було залучити досить велику кількість відвідувачів на сайт, не прикладаючи до його розробки великих зусиль і без особливих матеріальних витрат. Зараз же, в умовах постійно зростаючої конкуренції, функціональність сайту, його стиль і привабливість відіграють дуже велику роль. Адже незалежно від того створюється сайт для представлення якої-небудь фірми або просто для заробітку на рекламі, основною його функцією є залучення якомога більшої аудиторії. Тому для досягнення успіху важливий правильний підхід до вибору стратегії і виконавців.

Сьогодні обороти купівлі-продажу товарів і послуг в Інтернеті обчислюються в мільярдах доларів за рік, і згідно зі статистикою комерційний сегмент Інтернету показує стабільне зростання з року в рік . Все більше компаній-виробників відкривають свої сайти-представництва у мережі, які інформують потенційного покупця про товари і послуги та дають можливість безпосереднього замовлення і/або покупки бажаного товару. Завдяки зручності використання реалізація товарів в Інтернеті є однією з найбільш розповсюджених

За останніми даними, аудиторія в Інтернеті швидко зростає, продажі в Інтернеті у великих містах досягають 25%, а експерти підкреслюють тенденцію збільшення продажів через Інтернет. З кожним роком кількість

інтернет-магазинів збільшується, тому що це дійсно вигідно і зручно для покупців, не кажучи вже про бюджет і економію часу.

Існує велика кількість різних типів сайтів від «Односторінкового» (ваша візитка в інтернеті, яка розміщена на одній довгій сторінці) до «Форум» сайтів (сайти з самогенерованим контентом, тобто користувачі самі створюють контент), тож сайти бувають досить різноманітними. Останнім часом високої популярності набули електронні дошки оголошень – тип сайту, який надає користувачам можливість розміщувати оголошення про продаж товарів або послуги, що надаються. Перевага платформ онлайн оголошень полягає в тому, що переважна їх більшість – безкоштовні. Окрім того, що кожен може бачити всі оголошення та замовляти ті які йому потрібно, користувач може абсолютно безкоштовно виставити своє оголошення на продаж.

Безкоштовні дошки оголошень мають вигляд стандартного багатосторінкового веб-сайту з розбиванням на відповідні категорії товарів. Формат кожного конкретного ресурсу може бути вузькоспеціалізованим, тобто заточеним під окрему тематику (авто, дитяче, будівництво, тощо) або універсальним, коли об’єднані всі можливі тематичні категорії товарів та послуг

При розробці дошки оголошень, важливим стає питання внутрішньої SEO оптимізації проекту. При створенні проекту в обов'язковому порядку повинні бути дотримані SEO-вимоги:

Простота в дизайні:

- проект націлений на широке коло людей. Інтерфейс повинен бути простим і зрозумілим, щоб користувач не шукав приховану кнопку для додавання зображення або характеристик, а зміг інтуїтивно заповнити всю інформацію по оголошенню і розмістити на сайт.

Створення оголошення:

- слогани багатьох подібних проектів звучать приблизно так: "Розмісти оголошення за 30 секунд і товар буде продаватися". Даний факт має велике значення, тому як складна форма додавання або необхідність верифікації продавця може відлякати користувача і він піде до конкурентів.

Модуль пошуку:

- рядок пошуку є основним способом навігації по сайту. Вкрай важливо, щоб пошук працював швидко і максимально добре.

Управління перелінковкою:

- для рівномірного розподілу ваги між сторінками, необхідно реалізувати візуальні редактори для всіх текстів на сайті.

Окрім перелічених вимог також дуже важливо не обмежувати користувача, щоб він міг по максимуму використовувати дошку онлайн об’явлень задля своїх потреб: не обмежувати термін дії оголошення, не обмежувати кількість розміщення безкоштовних оголошень, не забороняти розміщувати оголошення з посиланням на зовнішні ресурси.

1.2. Призначення розробки та галузь застосування

Платформа онлайн оголошень була створена для того, щоб покупець міг з легкістю для себе обрати товар та замовити його не виходячи з дому. Покупцю для відгуку на оголошення, яке його зацікавило, достатньо просто набрати вказаний номер телефону або надіслати йому повідомлення на електронний поштовий адрес. Продавцю для розміщення інформації про продаж товарів та послуг достатньо зареєструватися та заповнити просту форму оголошення: обрати відповідну категорію, вказати назву та ціну, дати короткий опис та прикріпити кілька фотографій гарної якості.

Розроблений веб-застосунок може допомогти в розвитку бізнесу, бренду, збільшенню популярності. Так, наприклад, розміщення рекламних оголошень про товари та послуги на віртуальних дошках можна сміливо назвати одним з найоптимальніших підходів для малого, середнього та часто навіть великого бізнесу. У більшості випадків – це дуже бюджетно або взагалі безкоштовно.

Дошка онлайн об’явлень реалізована як веб-сайт, доступний в Інтернеті. Сайт складається з взаємозалежних частин, а функції кожної частини чітко розділені. Основна форма торгівлі веб-застосунку є C2C (customer-to-customer): у взаємодії кінцевих споживачів з кінцевими споживачами, сайт відіграє роль посередника між покупцем і продавцем.

1.3. Підстави для розробки

Підставами для розробки (виконання кваліфікаційної роботи) є:

- Освітня програма за спеціальністю 122 «Комп’ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 268-с від 18 травня 2022 р;
- завдання на кваліфікаційну роботу на тему «Розробка платформи онлайн оголошень, що об'єднує людей для покупки, продажу товарів та послуг з використанням серверного JavaScript».

1.4. Постановка завдання

Даний проект призначений для того, щоб користувач міг з легкістю вибрати для себе товар чи послугу та замовити його або з такою ж легкістю виставити свій товар на продаж.

Платформа онлайн-об’явлень повинна бути реалізована як доступний в інтернеті веб-сайт, який є посередником між покупцем та продавцем.

Даний додаток повинен мати наступні можливості:

- здійснення реєстрації користувачів;
- можливість швидкого створення безкоштовного оголошення з вказанням назви товару, категорії, ціни, короткого опису, та кількох фото;
- можливість редагування об’явлення та його видалення;
- відображення об’явлень (продавець бачить свої об’явлення на окремій сторінці);
- можливість добавляти товари у кошик та видаляти його з кошику
- відображення заказів та їх статус: «Новий», «прийнято», «відхилено»;
- сортування товарів за розділом, за категорією або за ціною
- пошук товарів по сайту.

Додаток повинен відповідати таким базовим умовам:

- простота в дизайні. Проект націлений на широке коло людей. Інтерфейс повинен бути простим і зрозумілим;
- створення оголошення повинно бути максимально зручним у використанні та швидким в реалізації;
- навігація по сайту повинна організовуватись максимально зручно та зрозуміло як для досвідчених та не дуже користувачів.

Виконавши всі перечисленні умови розробки веб-застосунку та реалізувавши функціонал, можна бути впевненим, що мета кваліфікаційної роботи досягнена.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Користувацький інтерфейс веб-сайту повинен бути зрозумілим, інтуїтивно представляти структуру розміщеної на ньому інформації та швидко та логічно переходити до розділів та сторінок. Усі форми для заповнення різною інформацією: форма реєстрації, форма для додавання оголошення, форма замовлення товару – повинні бути максимально простими та швидкими в реалізації. Пошук товару по сайту повинен виконуватися максимально просто, основним способом пошуку товарів є модуль пошуку (рядок пошуку), також на сайті повинні бути основні фільтри: фільтрація по ціні, фільтрація по категоріям.

Обмеження щодо кількості безкоштовних оголошень на одного користувача та розміщення посилань на інші зовнішні ресурси повинні бути відсутніми.

1.5.2. Вимоги до інформаційної безпеки

У процесі виконання даного веб-застосунку завантажувалися лише необхідні шаблони, модулі, пакети з перевірених джерел

Основні вимоги до інформаційної безпеки:

- забезпечення цілісності та захищеності даних користувача. Так, Для шифрування паролю користувача використовувалася адаптивна криптографічна хеш-функція формування ключа – bcrypt;
- конфіденційність інформації;
- доступність інформації для користувачів, які пройшли авторизацію.

1.5.3. Вимоги до складу та параметрів технічних засобів

Платформа онлайн оголошень для використання потребує лише підключення до мережі Інтернет та наявність браузера.

1.5.4. Вимоги до інформаційної та програмної сумісності

Платформу онлайн оголошень можна використовувати на будь-якій ОС за допомогою будь якого веб браузера, окрім Opera mini та Baidu Browser, встановленого на даному пристрої.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Платформа онлайн оголошень відіграє роль посередника між покупцем та продавцем.

У своєму онлайн-просторі сервіс дає можливість:

- зареєструватися або авторизуватися;
- взаємодію з поточними товарами: можливість детального перегляду товару і його замовлення;
- впроваджені необхідні функції для продавця: створювати привабливі оголошення, редагувати або видаляти створенні продукти, відслідковувати вхідні замовлення, приймати або відхиляти їх;
- впроваджені необхідні функції для покупців: можливість швидко знайти необхідну послугу або товар та замовити його, можливість відслідковувати історію своїх замовлень.

2.2. Опис застосованих математичних методів

Розробка платформи онлайн оголошень не потребує використання спеціальних математичних методів, тому вони не були використані для даної системи.

2.3. Опис використаних технологій та мов програмування

Visual Studio Code

Редактор коду - основний інструмент програміста. В редактори коду вбудовано багато корисних інструментів. Підсвічування коду, відладчики, автозаповнювач і компілятори. Visual Studio Code - безкоштовний редактор компанії Microsoft. Є одним з найбільш популярних редакторів на даний момент. У базовій версії є інтеграція з Git і режим відладки коду. Підтримує, в тому числі і через доповнення, велику кількість мов, в тому числі підходить і для розробки на мові Javascript. Його головними перевагами є легкість і зручність у використанні, особливо в веб розробці, а також вбудовані розширення.

Інтерфейс Visual Studio Code зображено на Рисунку 2.1.

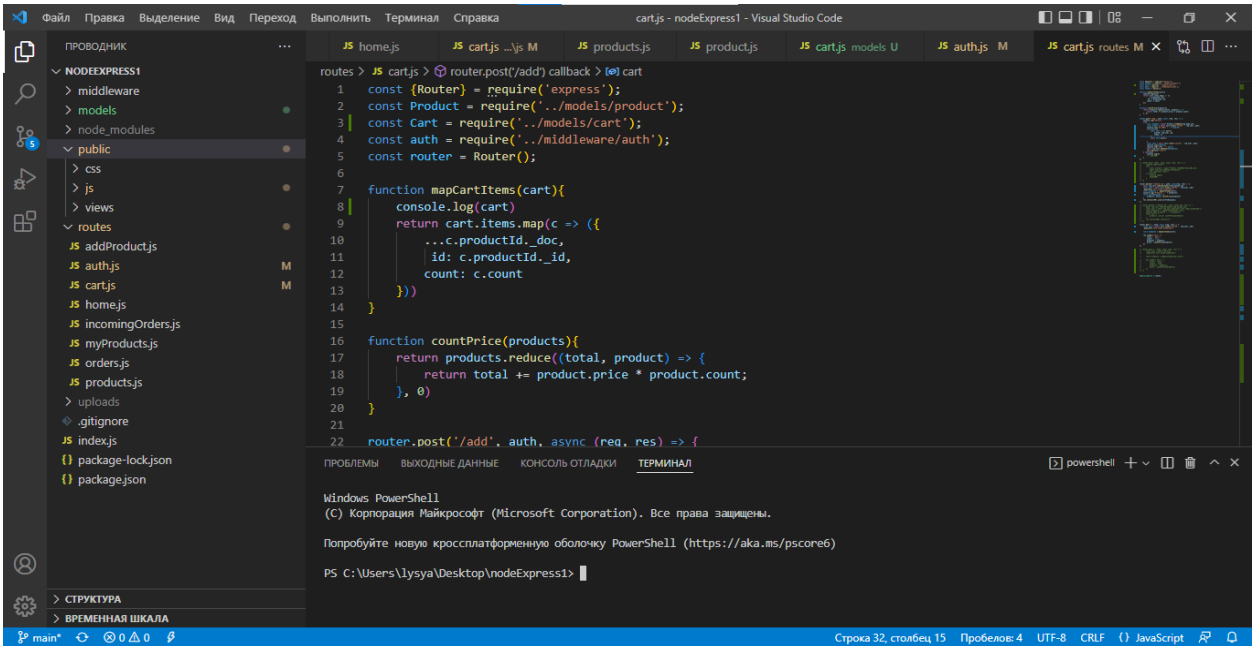

Рис. 2.1. Інтерфейс Visual Studio Code

Front-end

Для розробки front-end частини сайту онлайн-оголошень було використано наступні мови програмування та технології:

- HTML;

- CSS;
- JavaScript;
- Materialize;
- Handlebars.

HTML – (від англійського HyperText Markup Language — мова розмітки гіпертексту) - це мова розмітки, або ще один спосіб зберігання інформації. За допомогою HTML ти позначаєш текст, вказуючи своєму веб-переглядачу, як він має розуміти позначений текст. Веб-браузери отримують HTML документ від сервера за протоколами HTTP/HTTPS або відкривають з локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора.

CSS - каскадні таблиці стилів (від англійського Cascading Style Sheets) — спеціальна мова, що використовується для запису оформлення сторінок, написаних мовами розмітки даних. Концепція стилів працює так - текст спочатку виводиться, а потім форматується, за допомогою CSS стилів. Стилі дають змогу розмежувати вміст веб сторінки від її оформлення.

Materialize – це адаптивна бібліотека компонентів інтерфейсу, що призначена для створення вебсайтів та вебдодатків, які містять шаблони HTML та CSS для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу. Вона спрощує розробку динамічних вебсайтів та вебдодатків.

Handlebars - це шаблонний процесор, який динамічно генерує HTML-сторінку. Він зберігає HTML-сторінку чистою та відокремлює шаблони без логіки від бізнес-логіки у файлах JavaScript, покращуючи таким чином структуру програми, а також її ремонтопридатність та масштабованість. Це спрощує завдання оновлення даних на front-end вручну.

Back-end

Для розробки серверної частини веб-застосунку використовувалися:

- Express – фреймворк для створення веб-застосунків;
- MongoDB – документо-орієнтована система керування базами даних;
- Mongoose – бібліотека для роботи з MongoDB.

2.4. Опис структури системи та алгоритмів її функціонування

Розроблений веб-застосунок складається з взаємозалежних частин, а функції кожної частини чітко розділені.

Клієнтська частина

Структура папок та файлів клієнтської частини зображена на рис. 2.2.

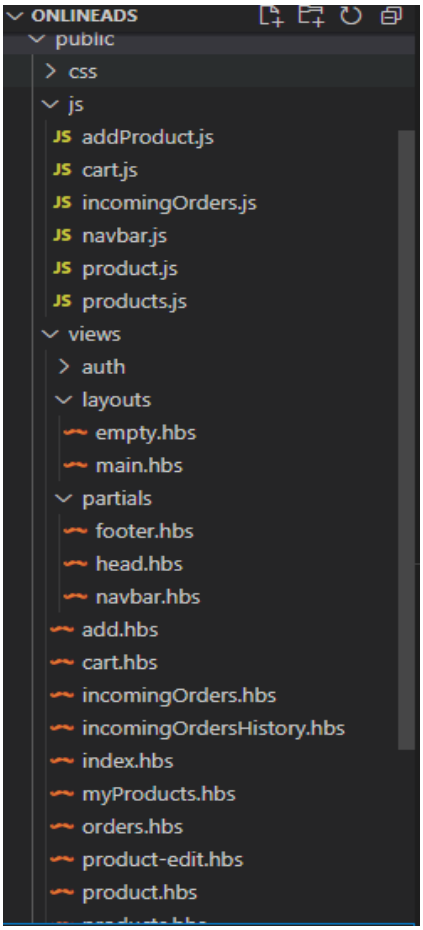

Рис. 2.2. Структура файлів клієнтської частини

У папці public знаходяться всі файли, які стосуються фронт-енду.

Для динамічного рендерингу HTML-сторінок використовується шаблонний процесор Handlebars.

У папці leyouts знаходиться головний шаблон main.hbs, який застосовується до кожної сторінки сайту.

Нерідко веб-сторінки в додатку використовують якісь спільні елементи. Однак тут виникає проблема: якщо потрібно змінити цей спільний елемент, то доведеться вносити зміни на всі веб-сторінки, які використовують цей елемент. І було б набагато простіше визначити цей елемент в одному місці, потім підключати на всі сторінки. Саме цю проблему вирішує папка partials, в якій зберігаються спільні елементи – шаблони: шапка профілю, хедер, футер, які, в свою чергу, підключаються до головного шаблону main.hbs.

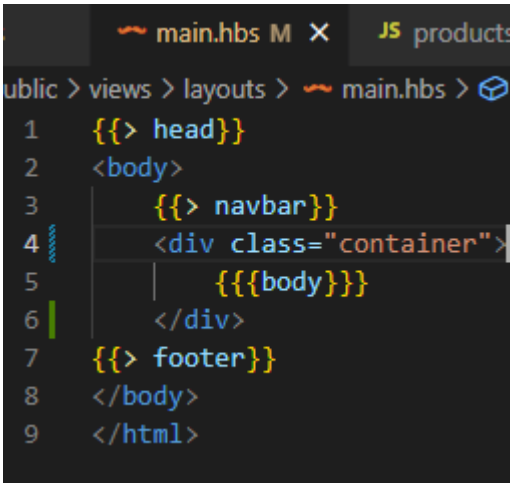

Рис. 2.3. Шаблон main.hbs

Скрипти були написані на JavaScript - мультипарадигменна мова, що використовується в ролі інтернет-інструменту, що вбудовується, що дає доступ до різноманітних складових додатків. Саме JS робить html-розмітку ресурсу та функціонал мережевих користувачів «живими». Завдяки цьому мовному типу сайт може активно відгукуватися будь-яку дію користувача. Видавати спливаючі блоки меню, здійснювати надсилання повідомлень, відкривати інші веб-сторінки за посиланням та багато іншого.

Папка js (див. рис. 2.2.) зберігає всі скрипти, які були прописані до даного веб-застосунку. Кожен скрипт підключений до hbs-документу відповідно до функцій, які він виконує.

- addProduct.js – виконує мережевий запит для загрузки інформації про продукт;
- cart.js – динамічно змінює корзину;
- incomingOrders.js – виконує мережеві запити для зміни статусу вхідних замовлень;
- navbar.js – ініціює методи фреймворку materialize для стилізації кнопок та спадного меню;
- product.js – ініціює метод фреймворку materialize для стилізації каруселі фотографій;
- products.js – виконує мережевий запит для сортування товарів.

Підключення бази даних MongoDB та бібліотеки Mongoose

База даних - це організована структура, яка призначена для зберігання та обробки взаємопов'язаної інформації. Її використання дозволяє зменшити навантаження на сервер, збільшується швидкість роботи з даними та контентом сторінок, а також збільшується безпека сайту, що немало важливо.

MongoDB - система управління базами даних, яка працює з документоорієнтованою моделлю даних. На відміну від реляційних СУБД, MongoDB не потребує таблиці, схеми або окремої мови запитів. Інформація зберігається як документів чи колекцій. Розробники позиціонують продукт як проміжну ланку між класичними СУБД та NoSQL. MongoDB не використовує схеми, як це роблять реляційні бази даних, що підвищує продуктивність усієї системи.

Mongoose - це бібліотека JavaScript, що дозволяє вам визначати схеми із строго-типізованими даними. Відразу після визначення схеми Mongoose дає можливість створити модель, засновану на певній схемі.

Потім модель синхронізується із документом MongoDB за допомогою визначення схеми моделі. Відразу після визначення схем та моделей ви можете користуватися різними функціями Mongoose для перевірки, збереження, видалення та запиту ваших даних, використовуючи звичайні функції MongoDB.

Створені моделі представлені на рис. 2.4.

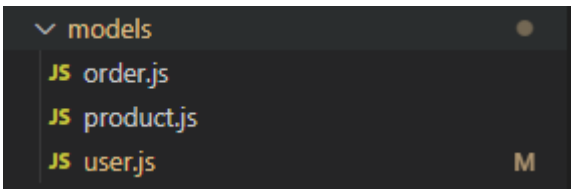

Рис. 2.4. Моделі даних

У моделях створені схеми, які описують всі властивості даної моделі. На даний момент Mongoose містить вісім типів даних схеми, які можуть мати властивість, що зберігається в MongoDB. Ці типи такі: String, Number, Date, Buffer, Boolean, mixed, ObjectId (* унікальний ідентифікатор об'єкта, первинний ключ, _id), Array.

Схеми моделей зображені на рис.2.5. - 2.7.

Рис. 2.5. Схема моделі користувача

Рис 2.6. Схема моделі оголошення (товару)

Рис. 2.7.Схема моделі замовлення

Колекції СУБД MongoDB відповідно до створених моделей, зображені на рис. 2.8.-2.10.

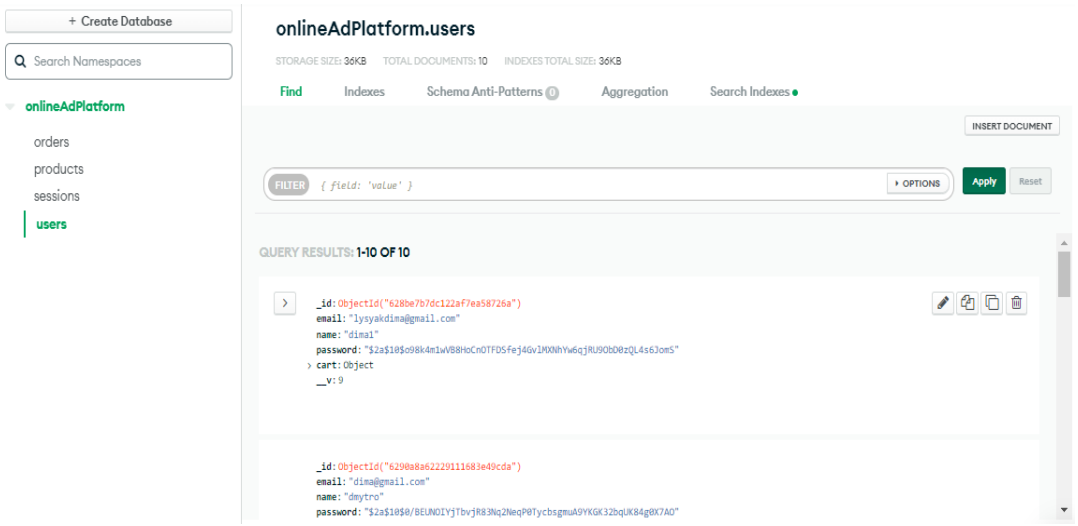

Рис. 2.8. Колекція моделей користувачів

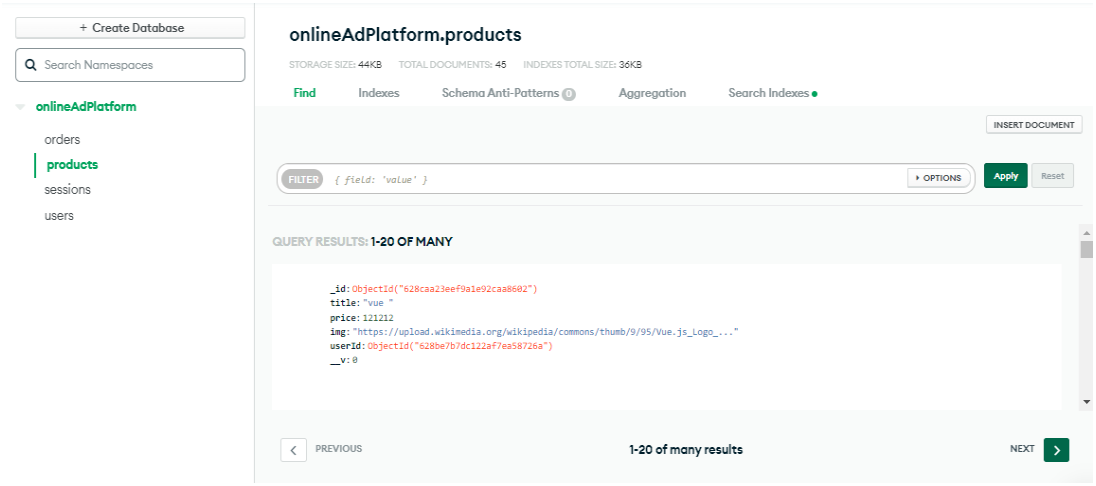

Рис. 2.9. Колекція моделей оголошень (товарів)

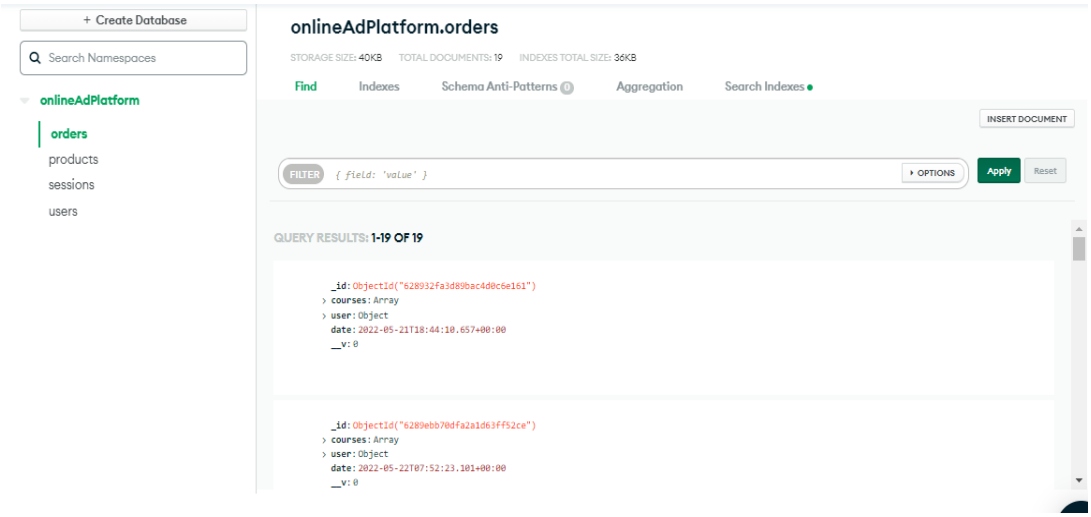

Рис. 2.10. Колекція моделей замовлень

Маршрутизація

У веб-розробці, процес, який відповідає за визначення обробника для конкретної сторінки, називається маршрутизація. Найчастіше кажуть "роутинг". Маршрутизація визначає, як програма відповідає на клієнтський запит до конкретної адреси (кінцевої точки), якою є URI (або шлях), та певного методу запиту HTTP (GET, POST, DELETE тощо).

- Метод GET вимагає представлення зазначеного ресурсу. Запити за допомогою GET повинні отримувати лише дані;
- Метод POST використовується для подання сутності до вказаного ресурсу, часто спричиняючи зміну стану або побічні ефекти на сервері;
- Метод DELETE видаляє вказаний ресурс.

Кожен маршрут може мати одну або кілька функцій обробки, що виконуються при зіставленні маршруту.

Шляхи маршрутів, у поєднанні з методом запиту, визначають конкретні адреси (кінцеві точки), де можуть бути створені запити. Шляхи маршрутів можуть бути рядками, шаблонами рядків або регулярними виразами. Для обробки запиту можна вказати кілька функцій зворотного

виклику, подібних до middleware (функції проміжної обробки). Єдиним винятком і те, що ці зворотні виклики можуть зініціюватись для обходу інших зворотних викликів маршруту. За допомогою цього механізму можна включити у маршрут попередні умови, а потім передати управління наступним маршрутам, якщо продовжувати роботу з поточним маршрутом не потрібно.

У розробленому веб-застосунку всі маршрути поділені на окремі модулі (роути), які знаходяться у папці «routes» рис. 2.11. Всі ці роути експортуються у головний файл додатку index.js рис. 2.12., де потім використовуються за допомогою методу use(), у якому вказаний URI – шлях, та сам роутер. рис. 2.13.

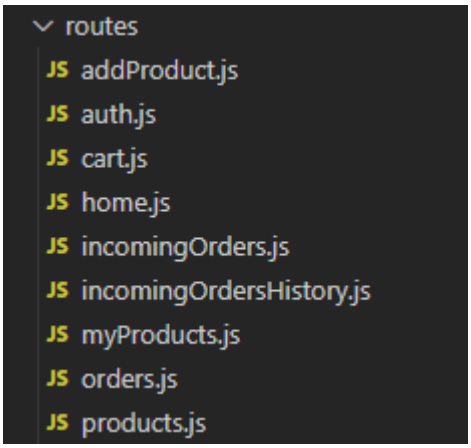

Рис. 2.11. Усі маршрути додатку

```
const homeRoutes = require('./routes/home');
const addRoutes = require('./routes/addProduct');
const cartRoutes = require('./routes/cart');
const productsRoutes = require('./routes/products');
const myProductsRoutes = require('./routes/myProducts');
const ordersRoutes = require('./routes/orders');
const incomingOrders = require('./routes/incomingOrders');
const incomingOrdersHistory = require('./routes/incomingOrdersHistory');
const authRoutes = require('./routes/auth');
```

Рис. 2.12.Імпрот маршрутів до головного файлу

```
app.use('/', homeRoutes);
app.use('/add', addRoutes);
app.use('/cart', cartRoutes);
app.use('/products', productsRoutes);
app.use('/myProducts', myProductsRoutes)
app.use('/orders', ordersRoutes);
app.use('/incomingOrders', incomingOrders);
app.use('/incomingOrdersHistory', incomingOrdersHistory);
app.use('/auth', authRoutes);
```

Рис.2.13. Реєстрація маршрутів

Підключення сесії

Сесія – це тимчасовий проміжок часу, протягом якого відбувається взаємодія користувача із сайтом. У сесії можна зберігати інформацію про данні користувача та його авторизацію. Сесії користувача зберігаються в базі даних, за для зменшення шансу злому додатку та оптимізації роботи сайту. За допомогою сесій можна обмежувати користувача у користуванні сайтом, наприклад обмежити доступ до таких сторінок, як «Додати оголошення», «Мої оголошення». Підключення сесії зображено на рис. 2.14. – 2.17.

```
const session = require('express-session');
const MongoStore = require('connect-mongodb-session')(session);
```

Рис .2.14. Підключення сесії, та пакету для синхронізації сесії з БД.

```
app.use(session({
    secret: 'some secret value',
    resave: false,
    saveUninitialized: false,
    store: store
}))
```

Рис. 2.15. Налаштування сесії

```
const store = new MongoStore({
    collection: 'sessions',
    uri: MONGODB_URI
})
```

Рис. 2.16. Налаштування колекції сесій та підключення її до БД

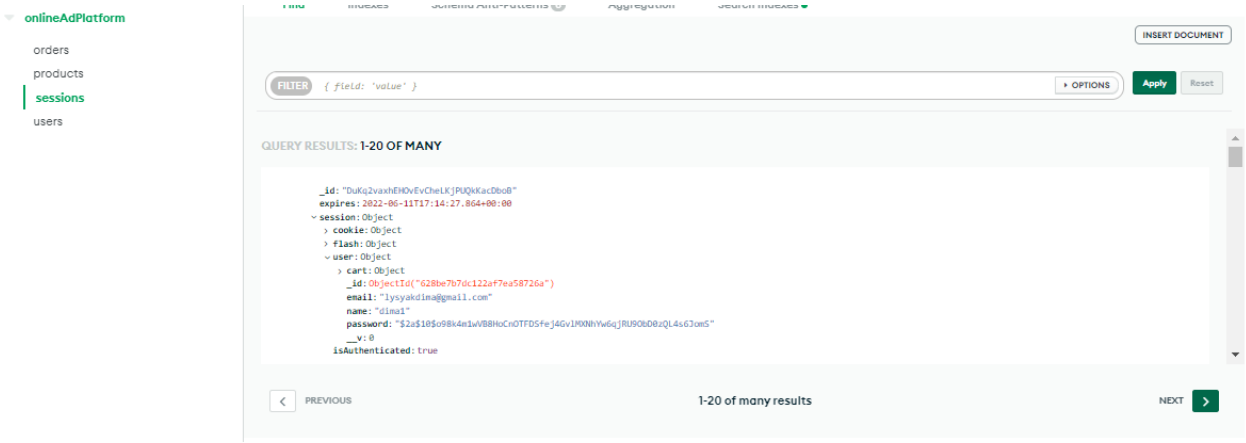

Рис. 2.17 Колекція сесій в БД

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Для введення вхідних даних використовуються форми, що приймають від користувача звичайний текст у кодировці utf-8, вибір у вигляді спадного списку та загрузка картинок у форматах png, gif, jpeg. Інформація вводиться вручну та відправляється на сервер. Дані зберігаються у JSON-подібних документах. Перевагою даного рішення є можливість легко контролювати вхідні та вихідні дані для обслуговування сайту.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

В процесі створення проекту був використаний ноутбук ASUS X540LJ з наведеними характеристиками

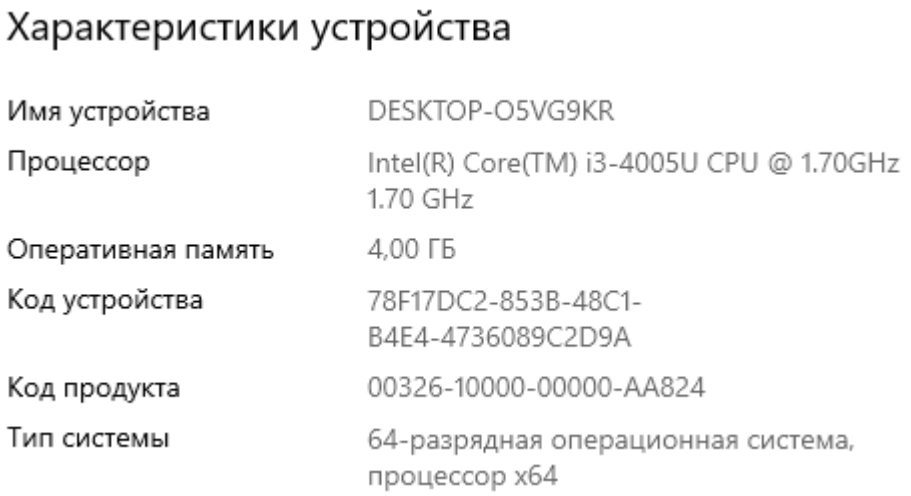

Рис. 2.18. Характеристики технічного засобу

2.6.2. Використані програмні засоби

Для розробки платформи онлайн-оголошень були використані такі програмні засоби :

- Редактор коду – Visual Studio Code;
- Веб-браузер - Google Chrome;
- Веб-інтерфейс для адміністрування СУБД MongoDB;
- GitHub.
-

2.6.3. Виклик та завантаження програми

З ціллю тестування або розширення функціоналу достатньо

використання локального серверу з файлами сайту.

Для запуску платформи онлайн оголошень потрібно завантажити файли сайту на хостинг та перейти на обрану інтернет-адресу.

2.6.4. Опис інтерфейсу користувача

Основний контент платформи онлайн оголошень – товар, це саме те, для чого користувач зайшов на сайт, тож головна сторінка сайту відразу відображає всі оголошення. Користувачі поділяються на два типи: авторизований або неавторизований. Авторизовані користувачі мають доступ до всього контенту сайту, всіх його сторінок та можливостей, на головній сторінці цю відмінність помітно лише у шапці профілю: рис. 2.19 для авторизованого користувача та рис. 2.20. для неавторизованого.

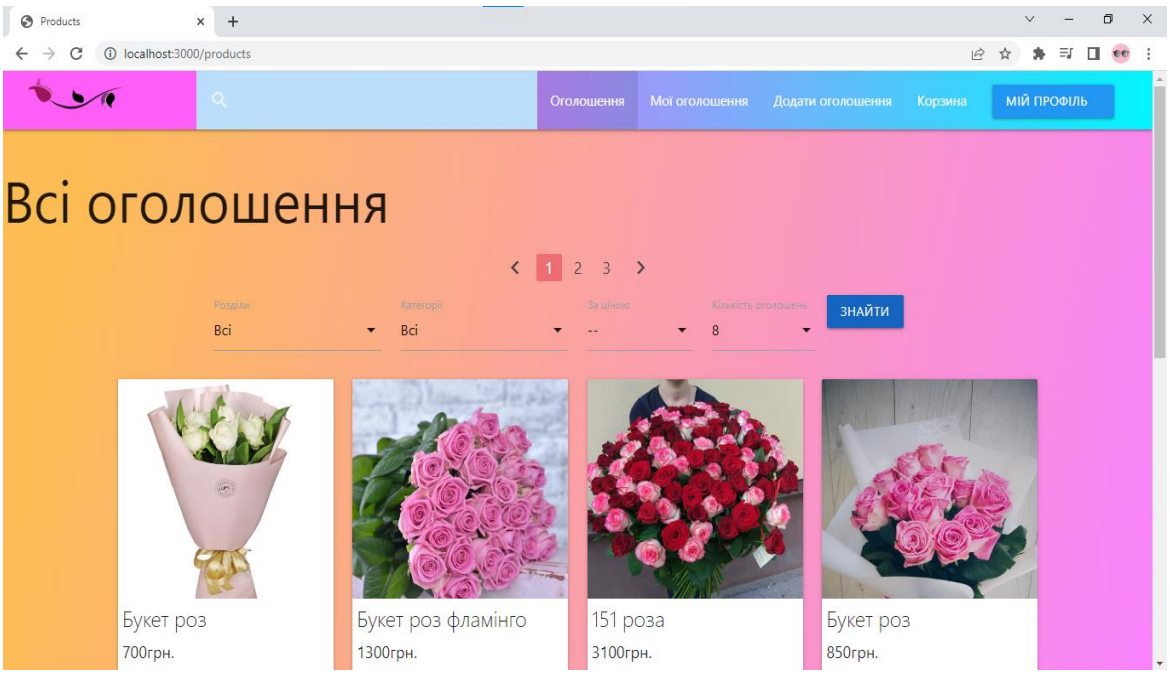

Рис. 2.19. Головна сторінка для авторизованого користувача

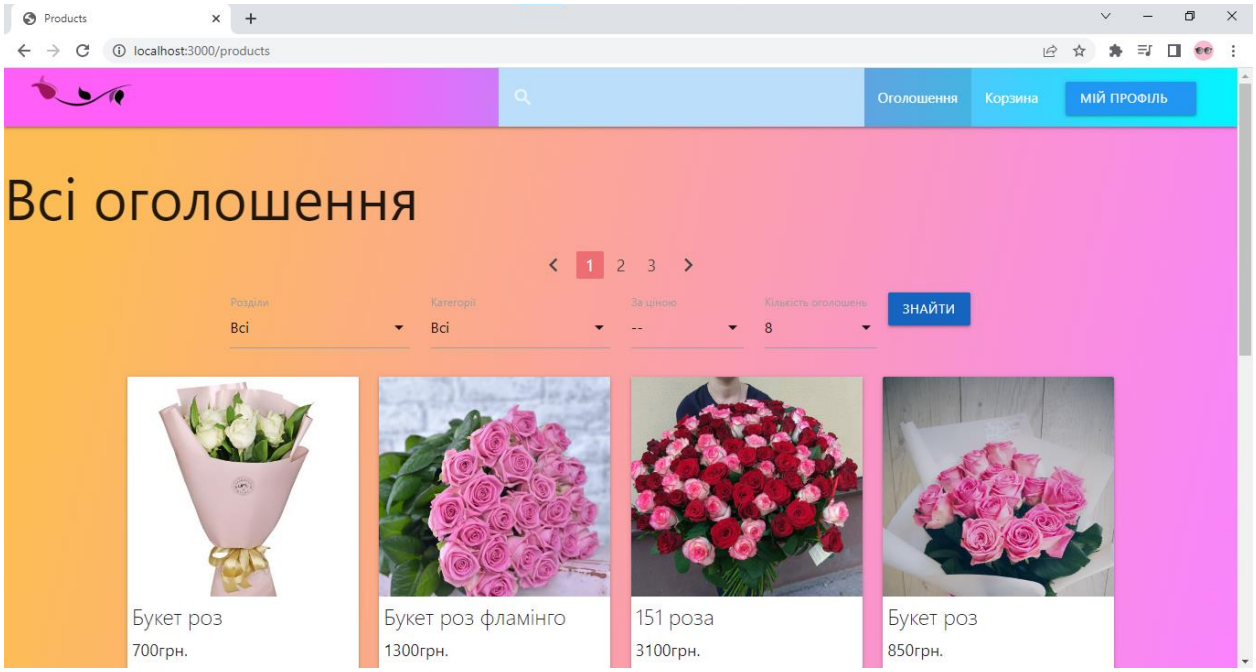

Рис. 2.20. Головна сторінка для неавторизованого користувача

Інтерфейс сайту на усіх сторінках ділиться на такі частини:

- Шапка сайту;
- Центральний блок;
- Підвал сайту.

У шапці розміщені посилання: на головну сторінку «Оголошення» та лого в лівому кутку шапки, «Мої оголошення», «Додати оголошення», «Корзина», спадне меню, в якому є лише посилання на сторінку «Увійти», якщо користувач неавторизований, та якщо користувач авторизований, то у спадному меню він бачить посилання на сторінки: «Мої замовлення», «Вхідні замовлення», «Історія вхідних замовлень».

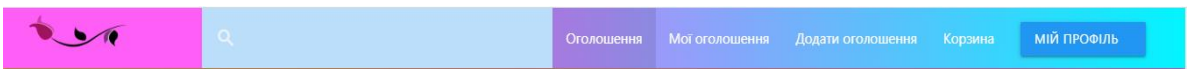

Рис. 2.21. «Шапка» сайту

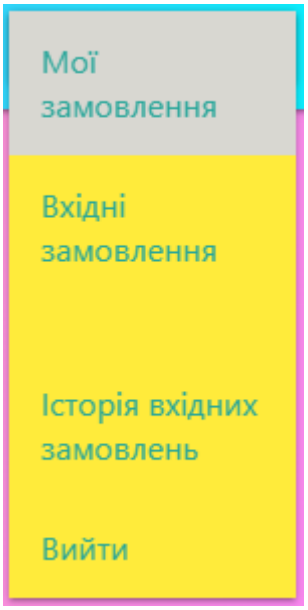

Рис. 2.21. Спадне меню для авторизованого користувача

Центральний блок інформації відмінний за контентом на кожній сторінці сайту. Так, на рисунках 2.23.-2.24. Зображено сторінку в якої є дві форми: форма для реєстрації та форма для авторизації.

Для реєстрації користувач повинен увести свій логін (електронну адресу), номер телефону, пароль та своє ім’я.

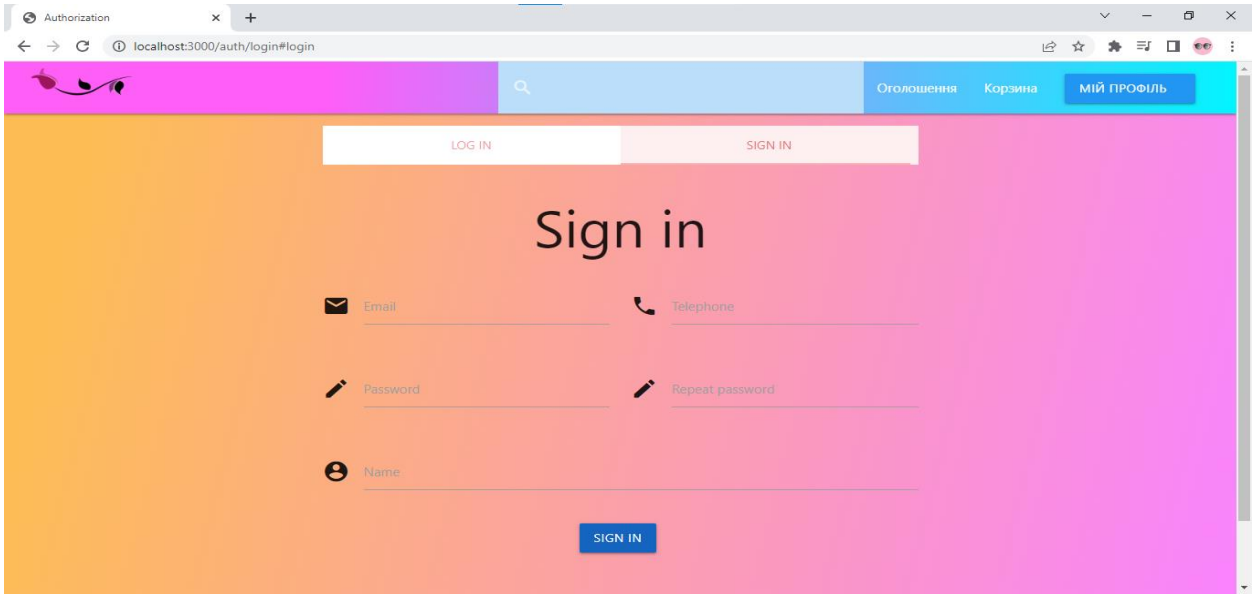

Рис.2.22. Форма для реєстрації

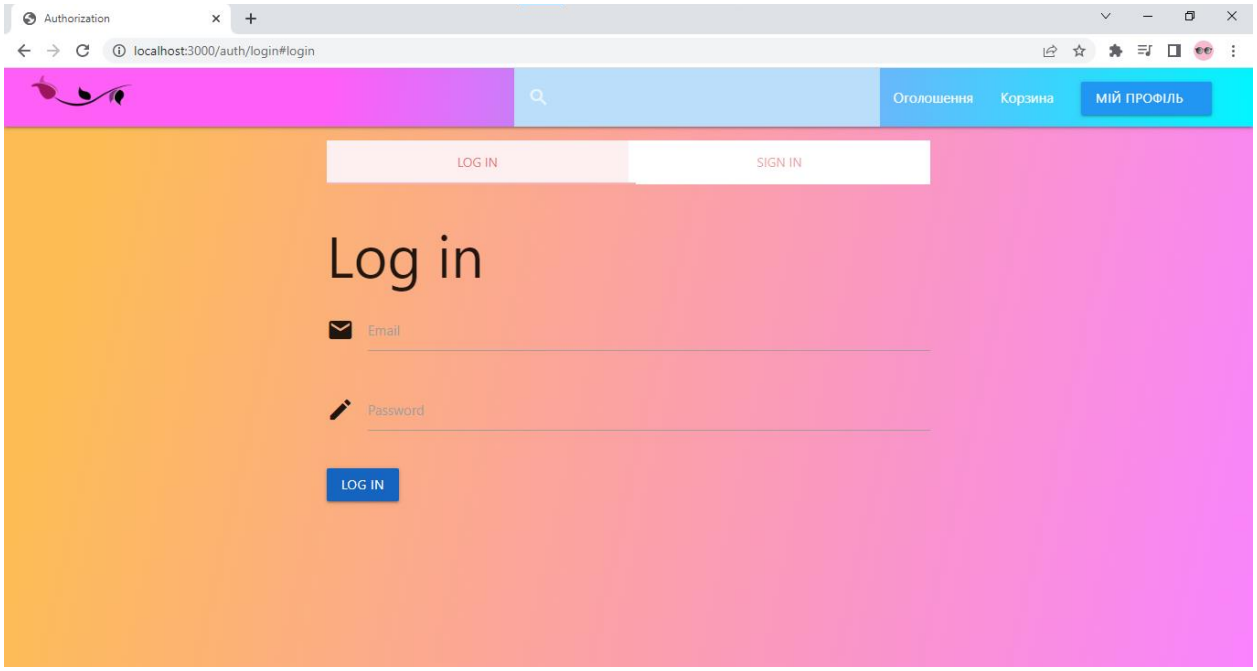

Рис. 2.23. Форма для авторизації

Валідація для реєстраційної форми.

Якщо при реєстрації користувач увів логін на який уже зареєстрований користувач, то на сторінці з'являється повідомлення з відповідним попередженням, а форма реєстрації очищується. Мінімальна кількість символів для паролю: 6.Номер можна увести лише український.

Авторизувавшись на сайт, у користувача з'являється можливість відстежувати за історією своїх замовлень у відділі «Мої замовлення», безкоштовно розміщати свої оголошення, бачити свої оголошення у відділі «Мої оголошення», відділ до якого приходять списки замовлень «Вхідні замовлення» і відділ «Історія вхідних замовлень».

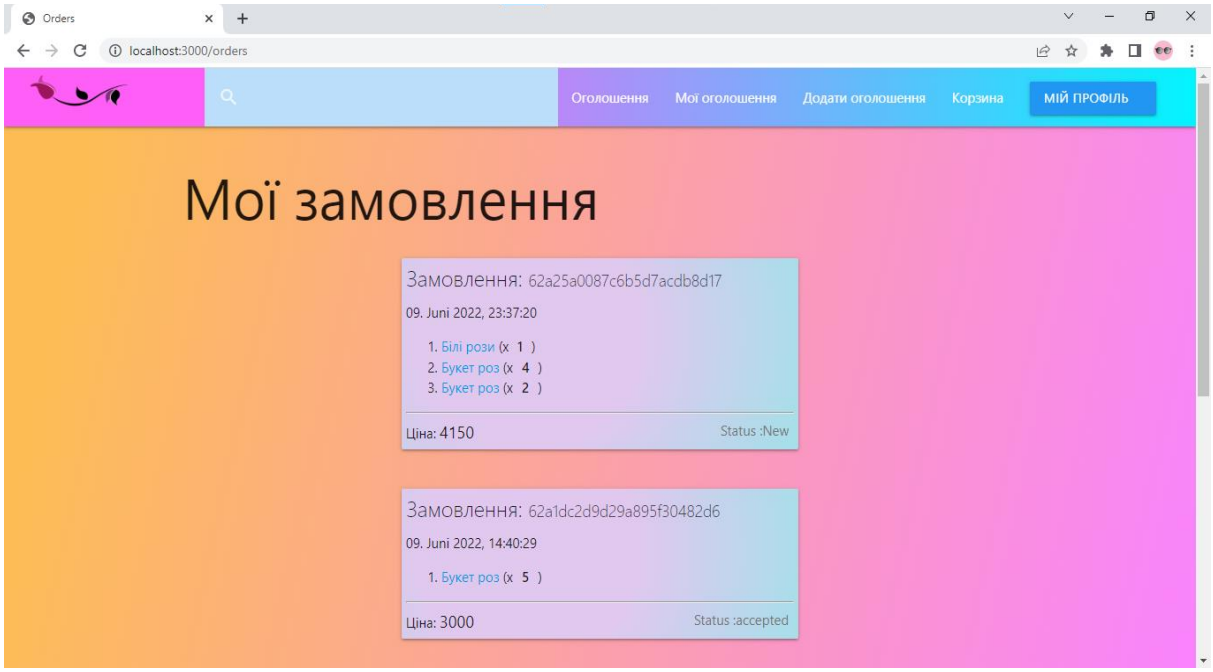

Рис. 2.24. Історія замовлень користувача

На кожній картці замовлення відображається її унікальний номер, дата покупки, посилання на замовлені товари та їх кількість, ціна всього замовлення та статус. Статус замовлення може бути : «New», «Accepted», «Rejected». «New» означає, що це нове замовлення і продавець ще ніяк не відреагував на нього, «Accepted» - продавець прийняв ваше замовлення, «Rejected» - продавець відхилив ваше замовлення.

Для того, щоб виставити своє оголошення на сайт, користувачеві потрібно заповнити просту, невеличку форму, в якій він вказує назву товару, ціну, категорію, розділ, описує товар, та додає декілька фотографій, перша з яких буде основною.

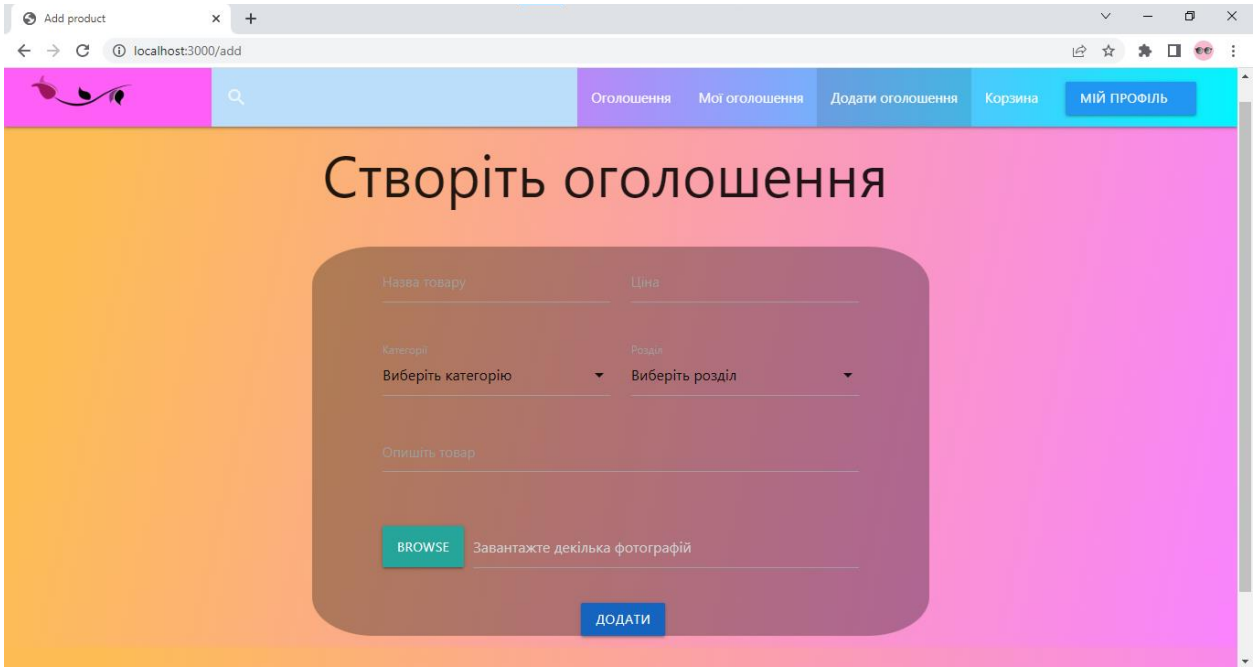

Рис. 2.25. Форма виставлення оголошення на сайт

Після відправки форми, об’явлення користувача з’являється на головній сторінці, але ,в свою чергу, власник товару бачить своє оголошення лише на сторінці «мої замовлення», де може відредагувати його, або видалити. рис.2.26 – рис2.27

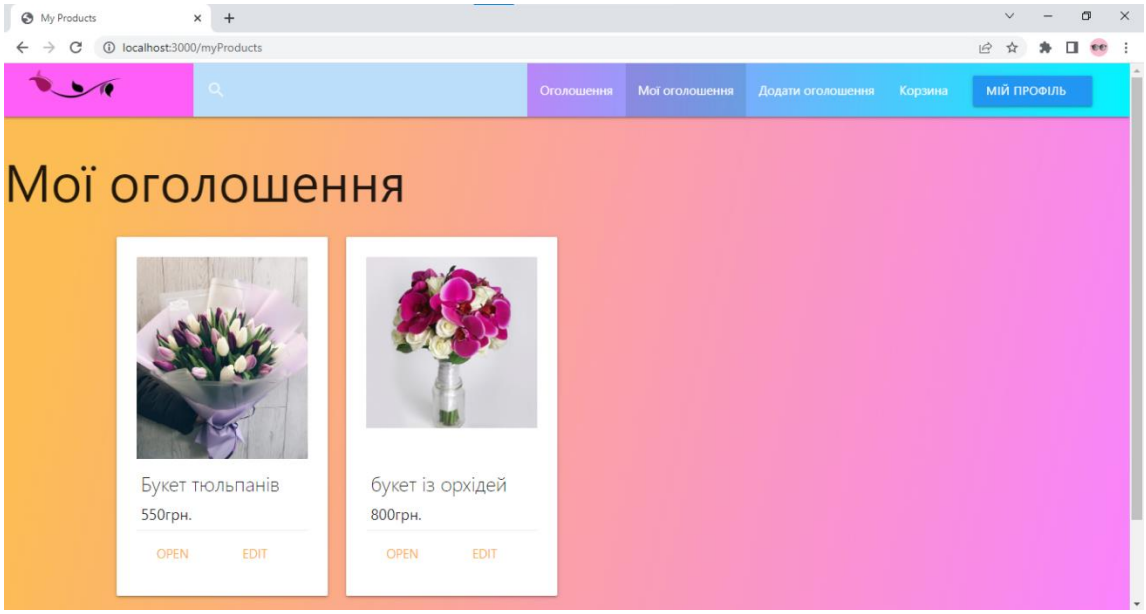

Рис 2.26. Сторінка «Мої оголошення»

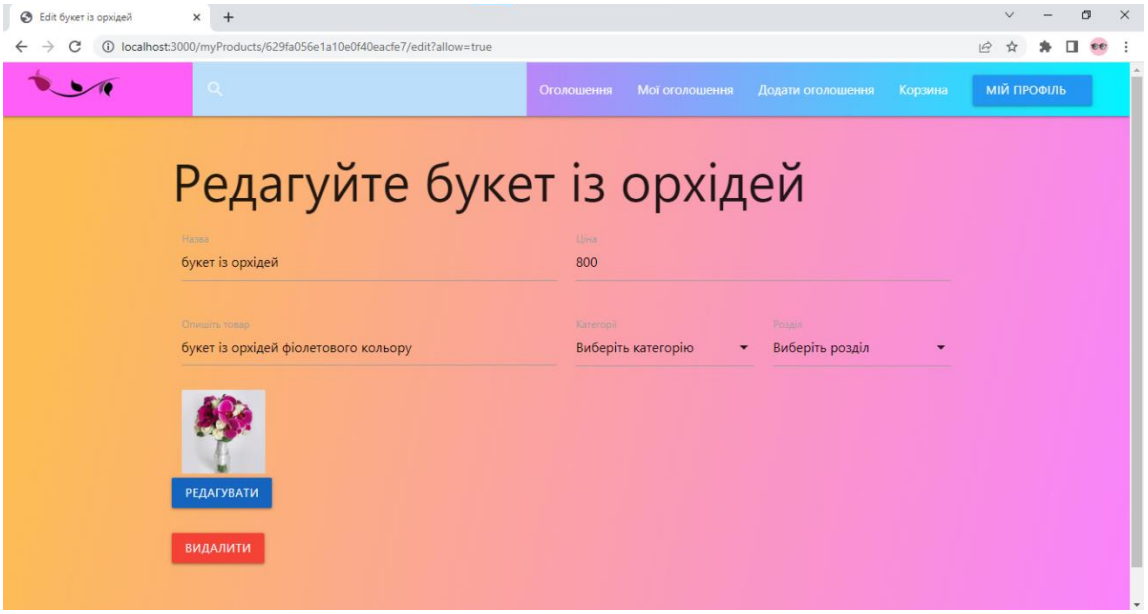

Рис. 2.27.Сторінка редагування оголошення

Якщо в користувача завоялють товар, на вкладці «Вхідні замовлення» з’являється позначка New з кількістю замовлень.

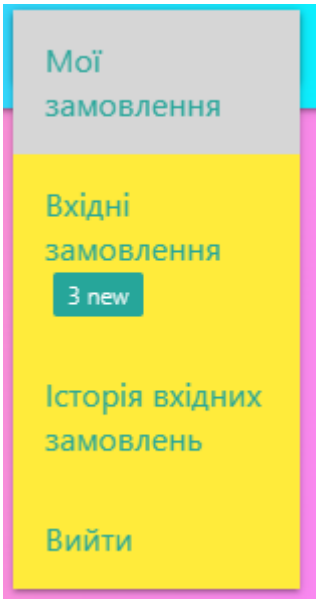

Рис. 2.28. Позначка нових замовлень

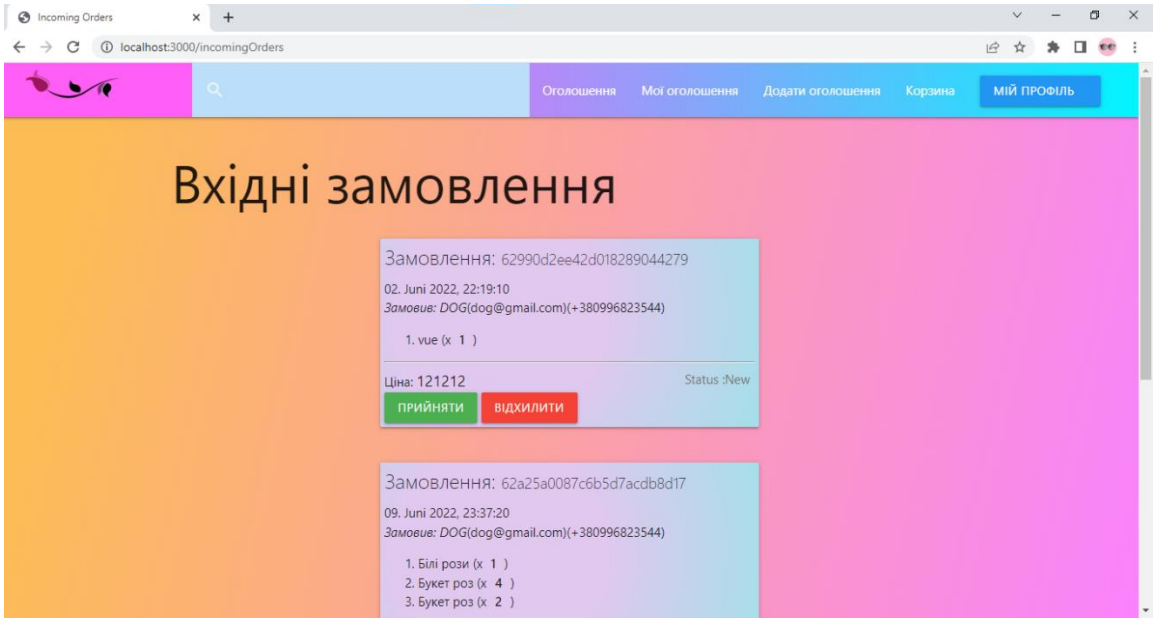

Рис. 2.29. Сторінка «Вхідні замовлення»

На картці вхідного замовлення, користувач бачить дату замовлення, дані покупця: ім’я, електронну пошту та номер, товари які в нього замовили, їхню кількість, загальну ціну замовлення та його статус. Продавець може прийняти або відхилити замовлення, нажавши на відповідні кнопки, після цього, оброблена картка замовлення потрапляє на сторінку «Історія вхідних замовлень», а у покупця у вкладці «Мої замовлення» змінюється статус замовлення.

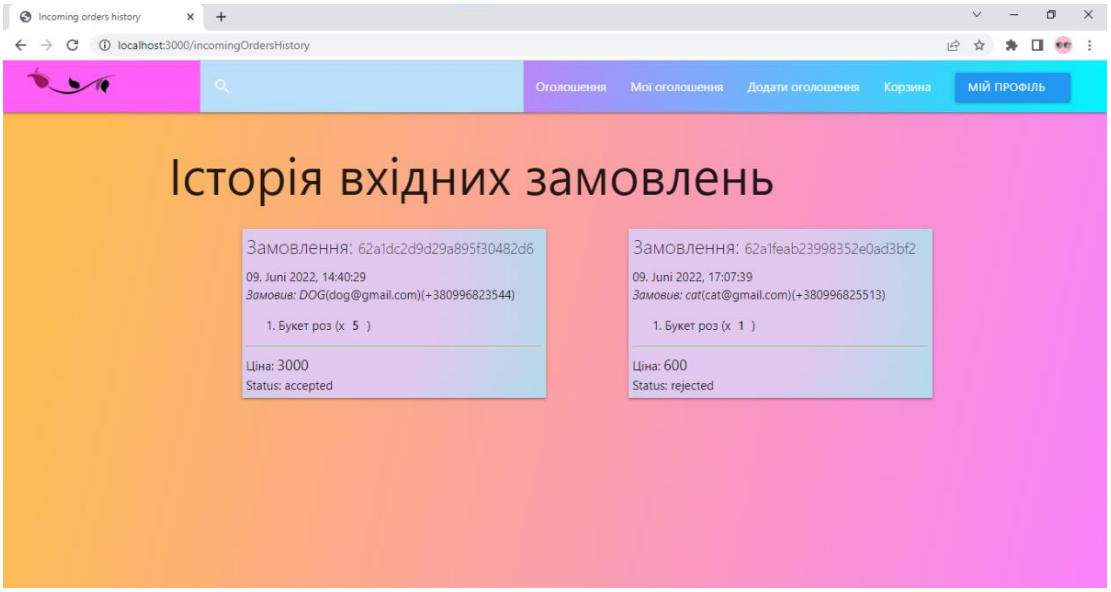

Рис. 2.30. Сторінка «Історія вхідних замовлень»

На головній сторінці «Оголошення», користувач може фільтрувати оголошення за: розділом, категорією, ціною, та обрати кількість товарів на сторінці. Натиснувши на картинку товару, «вспливає» його детальний опис.

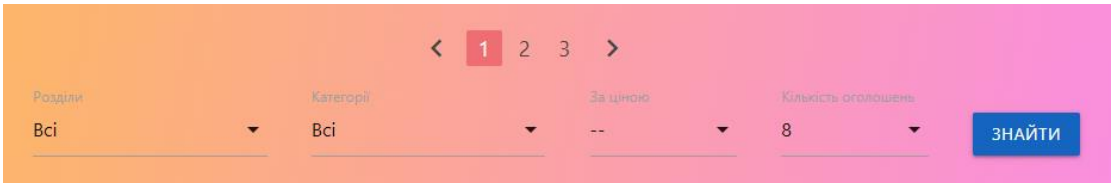

Рис. 2.31. Фільтри сортування товарів та пагінація

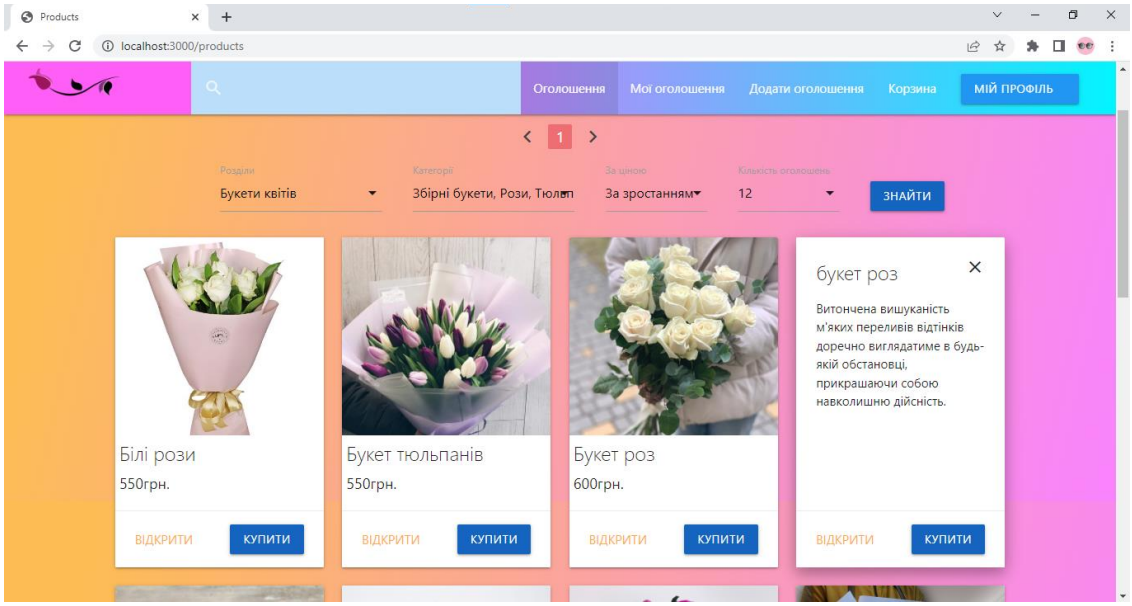

Рис. 2.32. Результат роботи фільтрів

Також товари можна шукати, використовуючи поле пошуку, що знаходиться у «шапці» сайту.

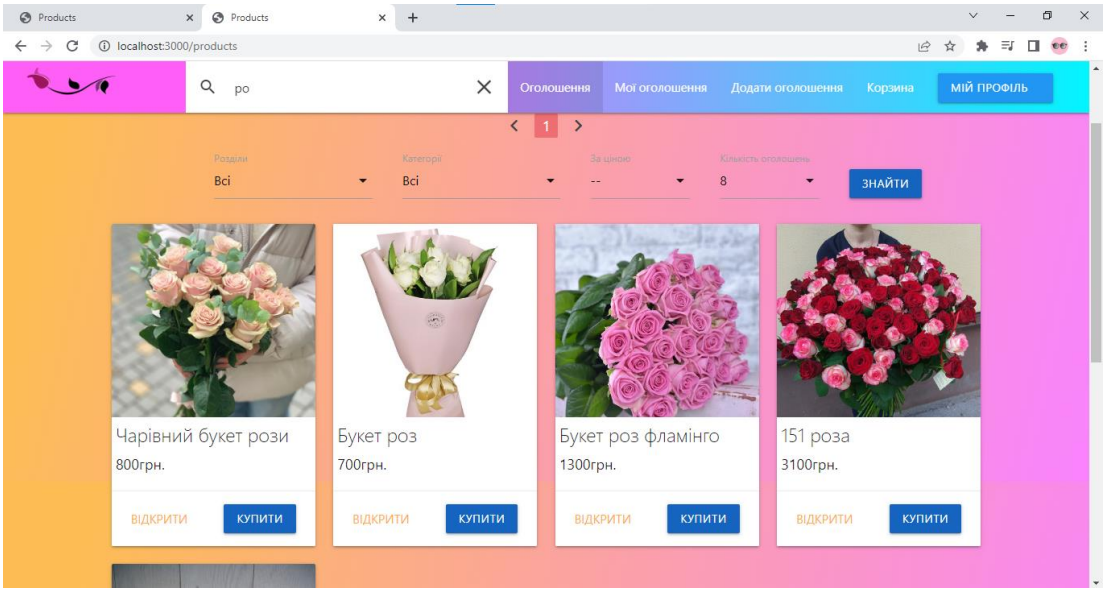

Рис. 2.33. Результат роботи рядку пошуку

Натиснувши на кнопку «відкрити», що на картці товару, можна перейти на сторінку товару, на якій знаходиться «карусель» фотографій, детальний опис товару та контактні дані автора оголошення.

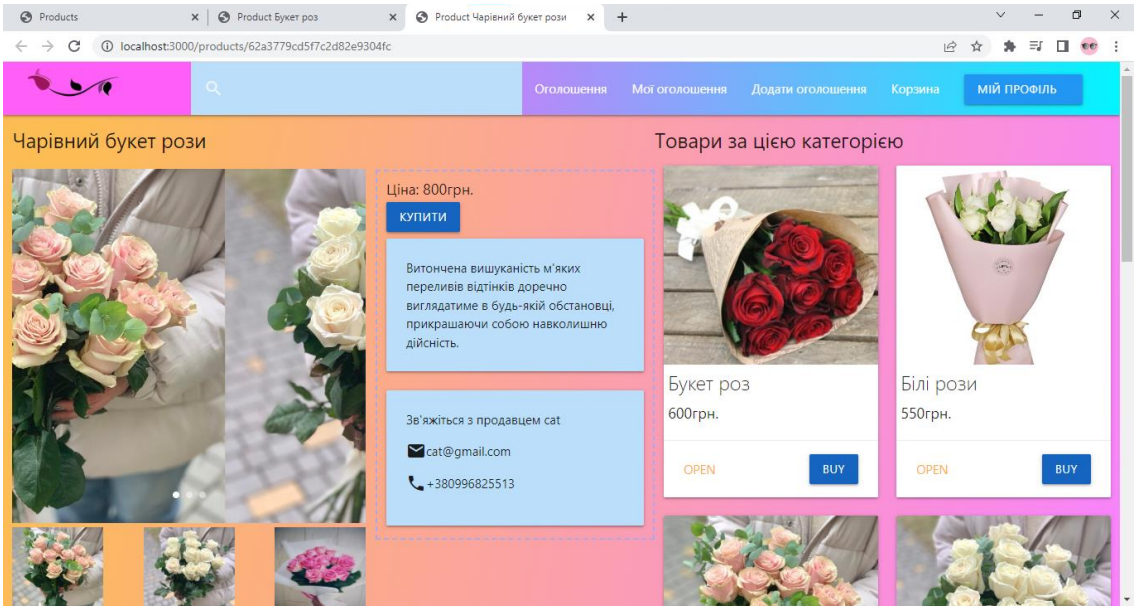

Рис.2.34. Сторінка оголошення

Натиснувши на кнопку «купити», товар додається до корзини. Корзину можна редагувати: змінювати кількість товарів або видаляти товар.

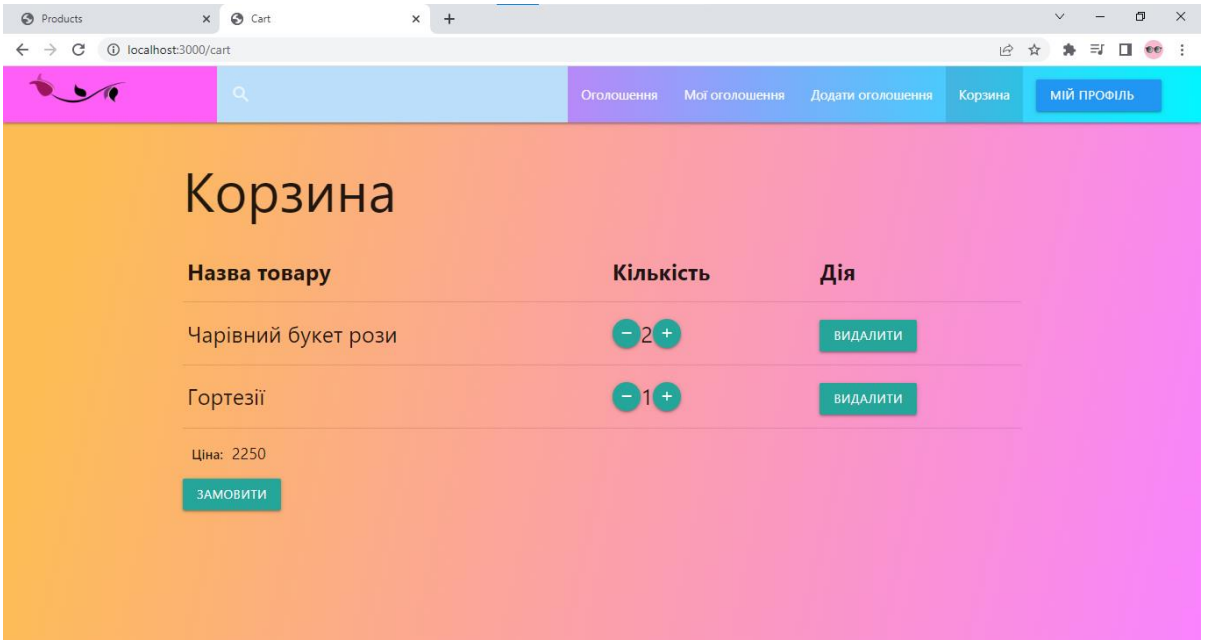

Рис. 2.35. Корзина

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. Передбачуване число операторів програми – 900;
2. Коефіцієнт складності програми (1,25…2,0) - 1,4;
3. Коефіцієнт корекції програми в ході розробки (0,05…0,1) – 0,06;
4. Годинна заробітна плата програміста – 80 грн/год;
5. Коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі (1,2…1,5) - 1,2;
6. Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності (від 3-х до 5 років) – 1,1;
7. Вартість машино-години ЕОМ – 9 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин.} \qquad (3.1)$$

Де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин)

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p)\,, \tag{3.2}$$

Де q – передбачуване число операторів програми (q = 1300);

C – коефіцієнт складності програми ($C = 1{,}4$);

p – коефіцієнт корекції програми в ході її розробки ($p = 0{,}06$)

Звідси умовне число операторів у програмі:

$$Q = 900 \cdot 1{,}4(1 + 0{,}06) = 1335{,}6$$

Витрати праці на вивчання опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75 \ldots 85) \cdot k}\,, \text{людино-годин} \tag{3.3}$$

де B- коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3-х до 5 років він складає 1,1;

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% ($B = 1.2$). З урахуванням коефіцієнта кваліфікації $k = 1{,}1$ отримуємо витрати праці на вивчення опису завдання:

$$t_u = (1335{,}6 \cdot 1{,}2)/(85 \cdot 1{,}1) = 17{,}14 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20 \ldots 25) \cdot k}\,, \text{людино-годин} \tag{3.4}$$

Де Q – умовне число операторів програми ;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення у формулу (3.4), отримаємо:

$$t_a = 1335{,}6/(25 \cdot 1{,}1) = 48{,}56, \text{ людино-години}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)\cdot k}, \text{ людино-годин} \quad (3.5)$$

$$t_n = \frac{1335{,}6}{24 \cdot 1{,}1} = 50{,}59, \text{ людино-годин.}$$

Витрати праці на налагодження програми ЕОМ:

- За умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)\cdot k}, \text{ людино-годин} \quad (3{,}6)$$

$$t_{атл} = 1335{,}6/(5 \cdot 1{,}1) = 242{,}83, \text{ людино-годин}$$

- За умовами комплексного налагодження завдання:

$$t_{отл}^{k} = 1{,}4 \cdot t_{отл}, \text{ людино-годин} \quad (3.7)$$

$$t_{отл}^{k} = 1{,}4 \cdot 242{,}83 = 339{,}96 \text{ людино-годин}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{др} + t_{до}, \text{ людино-годин} \quad (3.8)$$

Де $t_{др}$- трудомісткість підготовки матеріалів рукопису:

$$t_{др} = \frac{Q}{(15...20)\cdot k}, \text{ людино-годин} \quad (3.9)$$

$t_{до}$- трудомісткість редагування, печатки й оформлення документацій:

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин} \quad (3.10)$$

Підставляючи відповідні значення, отримуємо:

$$t_{др} = 1335{,}6/(20 \cdot 1{,}1) = 60{,}7, \text{ людино-годин}$$

$$t_{до} = 0{,}75 \cdot 60{,}7 = 45{,}52, \text{ людино-годин}$$

$$t_{\partial} = 60{,}7 + 45{,}52 = 106{,}22, \text{ людино-годин}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 35 + 17{,}14 + 48{,}56 + 50{,}59 + 242{,}83 + 106{,}22 = 500{,}34 \text{ людино-годин}$$

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ $K_{по}$ включають витрати на заробітну плату виконавця програми $З_{зп}$ і витрати машинного часу, необхідного на налагодження програми на ЕОМ:

$$K_{по} = З_{зп} + З_{мв}, \text{ грн.} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{зп} + t \cdot C_{пр}, \text{ грн.} \quad (3.12)$$

де: t – загальна трудомісткість, людино-годин;

$C_{пр}$ – середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 120грн/год, отримуємо

$$З_{зп} = 500{,}34 \cdot 80 = 40027{,}2 \text{ грн}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot C_{мч}, \text{ грн,} \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ – вартість машино-години ЕОМ, грн/год (10грн/год).

Підставивши в формулу (3.13) відповідні значення, визначимо

вартість необхідного для налагодження машинного часу:

$$З_{мв} = 339{,}96 \cdot 9 = 3059{,}64$$

Звідси витрати на створення програмного продукту:

$$K_{по} = 40027{,}2 + 3059{,}64 = 43086{,}84 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{міс,} \quad (3.14)$$

де B_k- число виконавців (дорівнює 1);

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні $F_p = 176$ годин)

Звідси витрати на створення програмного продукту:

$$T = 500{,}34/1 \cdot 176 = 2{,}82 \text{ міс.}$$

Висновок:

В процесі обчислення економічної частини, було виявлено, що для розробки платформи онлайн-оголошень необхідно майже 3 місяці роботи одного виконавця. Вартість проекту становить 43086 грн.

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є розроблена платформа онлайн оголошень формат якої – квіти.

У процесі виконання роботи були виконані такі основні завдання:

- Написання клієнтської частини додатку з використанням з використанням веб-технологій HTML, CSS, JavaScript, Materialize, Handlebars;
- Написання серверної частини додатку використовуючи фреймворк Express.js;
- Підключення бази даних MongoDB.

Усі використані технології гармонійно доповнюють один одного.

Інтерфейс розробленого веб-додатку в якому поєднуються теплі, приємні кольори, дуже простий та зрозумілий ,в ньому зможе розібратися навіть недосвідчений користувач.

Основні функції платформи онлайн-оголошень були реалізовані за планом: можливість зареєструватися, купити товар, виставити свої оголошення, редагувати або видаляти свої оголошення, отримувати сповіщення при покупці вашого товару, пошук по сайту, пагінація та фільтри.

Актуальність розробленого веб-додатку обумовлюється приваблення до себе бізнесу різного калібру, адже ,що може бути краще, ніж активна популяризація свого продукту та бренду в інтернеті задарма.

Отже, в ході виконання кваліфікаційної роботи було досягнуто всіх цілей та були виконані всі вимоги, що були висунуті до даного веб-додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. How To Become A Web Developer — Everything You Need To Know [Електронний ресурс] – Режим доступу до ресурсу: https://careerfoundry.com/en/blog/web-development/what-does-it-take-to-becomea-web-developer-everything-you-need-to-know-before-getting-started/. (дата звернення: 24.05.2022)
2. Кращі мови програмування 2022 року, які варто вивчати [Електронний ресурс] – Режим доступу до ресурсу: https://merehead.com/ru/blog/popular-programming-languages-2022/. (дата звернення: 27.04.2022).
3. Хавербеке Марейн. Виразний JavaScript. 2 видання, 2015. – 745 с.
4. Як створити дошку онлайн оголошень [Електронний ресурс] – Режим доступу до ресурсу: https://merehead.com/ru/blog/build-a-website-like-craigslist/ (Дата звернення: 25.05.2022).
5. Крокфорд Дуглас. JavaScript. Сильні сторони, 2016. — 176 с.
6. Топ 5 найпопулярніших дошок оголошень в Україні в 2022 році. [Електронний ресурс] – Режим доступу до ресурсу: https://oplata.com/uk/blog/top-5-samyh-populyarnyh-dosok-obyavlenii-v-ukraine-v (Дата звернення: 06.05.2022)
7. Документація MongoDB [Електронний ресурс] – Режим доступу до ресурсу https://www.mongodb.com/docs/ (Дата звернення: 01.06.2022)
8. Документація Handlebars [Електронний ресурс] – Режим доступу до ресурсу https://handlebarsjs.com/ (Дата звернення: 03.05.2022)
9. Документація Materialize [Електронний ресурс] – Режим доступу до ресурсу https://materializecss.com/about.html (Дата звернення: 02.05.2022)
10. Розробка структури сайту [Електронний ресурс] – Режим доступу до ресурсу: https://artjoker.ua/ru/uslugi/razrabotka-struktury-saita/ (Дата

звернення : 27.04.2022)

11. Express/ Node documentation [Електронний ресурс] – Режим доступу до ресурсу:https://developer.mozilla.org/ru/docs/Learn/Server-side/Express_Nodejs/Introduction (Дата звернення 06.05.2022)
12. Ніксон Р. Створюємо динамічні веб-сайти за допомогою MySQL, JavaScript, CSS і HTML5, 2016. - 510 с.
13. Ваш інтернет магазин від А до Я [Електронний ресурс] –Режим доступу до ресурсу: https://books.google.com.ua/books?id=Zt0qDwAAQBAJ&&hl=ru&source=gbs_navlinks_s (Дата звернення: 13.04.2022)
14. Walls C. Spring in Action / Craig Walls., 2018. – p.520
15. Duckett J. HTML and CSS : Design and Build Websites / Jon Duckett. Robson E. Head First HTML and CSS: A Learner's Guide to Creating StandardsBased Web Pages / E. Robson, E. Freeman., 2012. – p.768.
16. The 2020 Roadmap To Fullstack Web Development [Електронний ресурс] – Режим доступу до ресурсу: https://codingthesmartway.com/the-2020-roadmap-to-fullstack-web-development/ (дата звернення: 24.04.2022).
17. Документація Mongoose [Електронний ресурс] – Режим доступу до ресурсу: https://mongoosejs.com/docs/api.html (Дата звернення: 23.05.2022)
18. Введення в Mongoose для MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://code.tutsplus.com/ru/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527 (Дата звернення: 22.05.2022)
19. Маршрутизація в Express [Електронний ресурс] – Режим доступу до ресурсу: https://expressjs.com/ru/guide/routing.html (Дата звернення: 24.05.2022)
20. Fetch API [Електронний ресурс] – режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (Дата

звернення: 29.05.2022)

21. Асинхронний JavaScript [Електронний ресурс] – режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Learn/JavaScript/Asynchronous (Дата звернення: 17.05.2022)
22. Підручник Express частина 4 [Електронний ресурс] – режими доступу до ресурсу: https://developer.mozilla.org/ru/docs/Learn/Server-side/Express_Nodejs/routes (Дата звернення: 08.05.2022)
23. Імперія Amazon [Електронний ресурс] – режим доступу до ресурсу: https://elnews.com.ua/uk/imperiya-amazon-istoriya-ta-osoblyvosti-kompaniyi/ (Дата звернення: 19.04.2022)

ДОДАТОК А

КОД ПРОГРАМИ

Index.js

```
const express = require('express');
const path = require('path');
const flash = require('connect-flash');
const mongoose = require('mongoose');
const Handlebars = require('handlebars');
const exphbs = require('express-handlebars');
const session = require('express-session');
const MongoStore = require('connect-mongodb-session')(session);
const {allowInsecurePrototypeAccess} = require('@handlebars/allow-
prototype-access');
const homeRoutes = require('./routes/home');
const addRoutes = require('./routes/addProduct');
const cartRoutes = require('./routes/cart');
const productsRoutes = require('./routes/products');
const myProductsRoutes = require('./routes/myProducts');
const ordersRoutes = require('./routes/orders');
const incomingOrders = require('./routes/incomingOrders');
const incomingOrdersHistory = require('./routes/incomingOrdersHistory');
const authRoutes = require('./routes/auth');

const varMiddleware = require('./middleware/variables');
const userMiddleware = require('./middleware/user');

const MONGODB_URI =
`mongodb+srv://d1mpleo:inemev15@cluster0.fta6y.mongodb.net/onlineAd
Platform?retryWrites=true&w=majority`;
```

```
const app = express();
const hbs = exphbs.create({
   handlebars: allowInsecurePrototypeAccess(Handlebars),
   defaultLayout: 'main',
   extname: 'hbs'
})

app.engine('hbs', hbs.engine);
app.set('view engine', 'hbs');
app.set('views', 'public/views');

const store = new MongoStore({
   collection: 'sessions',
   uri: MONGODB_URI
})



app.use(express.static(path.join(__dirname, 'public')));
app.use('/media', express.static(path.join(__dirname, 'uploads')));

app.use(express.urlencoded({extended: true}));

app.use(session({
   secret: 'some secret value',
   resave: false,
   saveUninitialized: false,
   store: store
}))
```

```
app.use(flash());
app.use(varMiddleware);
app.use(userMiddleware);

app.use('/', homeRoutes);
app.use('/add', addRoutes);
app.use('/cart', cartRoutes);
app.use('/products', productsRoutes);
app.use('/myProducts', myProductsRoutes)
app.use('/orders', ordersRoutes);
app.use('/incomingOrders', incomingOrders);
app.use('/incomingOrdersHistory', incomingOrdersHistory);
app.use('/auth', authRoutes);

const PORT = process.env.PORT || 3000;

async function start(){

   try{
      await mongoose.connect(MONGODB_URI, {maxPoolSize:50,
wtimeoutMS:2500,});
      app.listen(PORT, () =>{
         console.log(`Server is running on port ${PORT}`);
      })
   }catch(e){
      console.log(e);
   }
}
```

```
start();
```

/routes/home.js

```
const {Router} = require('express');
const router = Router();

router.get('/', (req, res) => {
   res.redirect('/products');
})

module.exports = router;
```

/routes/addProduct.js

```
const {Router} = require('express');
const Product = require('../models/product');
const auth = require('../middleware/auth');
const router = Router();

const path = require("path");
const multer = require("multer");
const storage = multer.diskStorage({
   destination: function (req, file, cb) {
     cb(null, 'uploads/')
   },
   filename: function (req, file, cb) {
     cb(null, Date.now() + path.extname(file.originalname)) //Appending
extension
   }
 });
const upload = multer({ storage: storage });
```

```
router.get('/', auth, (req, res) => {
    res.render('add', {
        title : 'Add product',
        isAdd : true
    });
})

//creates new product
router.post('/', auth, upload.array("files"), async (req, res) => {

    const product = new Product({
        title: req.body.title,
        price: req.body.price,
        img: req.files,
        category: req.body.category,
        about: req.body.about,
        userId: req.user._id
    })

    try{
        await product.save();
        res.send(200);
    }catch(e){
        console.log(e);
        res.send(500);
    }
})
```

```
module.exports = router;
```

/router/auth.js

```
const {Router} = require('express');
const bcrypt = require('bcryptjs');
const User = require('../models/user');
const router = Router();

router.get('/login', async (req,res) => {
    res.render('auth/login', {
        title: 'Authorization',
        isLogin: true,
        loginError: req.flash('loginError'),
        registerError: req.flash('registerError')
    })
})
router.get('/logout', async (req,res) => {
    req.session.destroy(() => {
        res.redirect('/auth/login#login');
    })
})
router.post('/login', async(req, res) => {
    try{
        const {email, password} = req.body;
        const candidate = await User.findOne({email});

        if(candidate){
            const areSame = await bcrypt.compare(password,
candidate.password);
```

```
            if(areSame){
                req.session.user = candidate;
                req.session.isAuthenticated = true,
                req.session.save(err => {
                    if(err){
                        throw err;
                    }
                    res.redirect('/products');
                })
            }else{
                req.flash('loginError', 'Неправильний пароль');
                res.redirect('/auth/login#login');
            }
        }else{
            req.flash('loginError', 'Такого користувача не існує');
            res.redirect('/auth/login#login');
        }
    }catch(e){
        console.log(e);
    }
})
router.post('/register', async(req, res) => {
    try{
        const {email, password, repeatPassword, name, phone} = req.body;

        const candidate = await User.findOne({email});

        if(candidate){
            req.flash('registerError', 'Користувач із цією електронною адресою
```

```
вже існує');
            res.redirect('/auth/login#register');
        }else{
            if(!validateEmail(email)){
                req.flash('registerError', 'Невірний імейл');
                res.redirect('/auth/login#register');
            }else if(!validatePhone(phone)){
                console.log(phone);
                console.log(validatePhone(phone));
                req.flash('registerError', 'Невірний номер телефону');
                res.redirect('/auth/login#register');
            }else if(password !== repeatPassword){
                req.flash('registerError', 'Паролі не співпадають!');
                res.redirect('/auth/login#register');
            }else{
                const hashPassword = await bcrypt.hash(password, 10);
                const user = new User({
                    email, phone, name, password: hashPassword
                })
                await user.save();

                res.redirect('/auth/login#login');
            }
        }

    }catch(e){
        console.log(e);
    }
})
```

```
let validateEmail = function(email) {
    let re = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
    return re.test(email)
};
let validatePhone = function(phone) {
    let re = /^\+?3?8?(0\d{9})$/;
    return re.test(phone)
};

module.exports = router;
```

/router/cart.js

```
const {Router} = require('express');
const Product = require('../models/product');
const auth = require('../middleware/auth');
const router = Router();

function mapCartItems(cart){
    return cart.items.map(c => {
        console.log(c.productId);
        let productIdJson =  c.productId ? c.productId.toJSON() : {}
        let id = c.productId ? c.productId._id : ""
        return ({
        ...productIdJson,
         id,
        count: c.count
    })})
}
```

```
function countPrice(products){
    return products.reduce((total, product) => {
        return total += product.price * product.count;
    }, 0)
}
router.post('/add', auth, async (req, res) => {
    try{
        const product = await Product.findById(req.body.id);
        await req.user.addToCart(product);
        res.redirect("/cart")
    } catch(e){
        console.log(e)
        res.send(500)
    }
})

router.post('/add/:id', auth, async (req, res) => {
    try{
        console.log(req.params.id)
        await req.user.addToCart2(req.params.id);
        const user = await req.user.populate('cart.items.productId');
        const products =  mapCartItems(user.cart);
        const cart = {
            products, price: countPrice(products)
        }
        res.status(200).json(cart);
    } catch(e){
        console.log(e)
        res.send(500)
```

```
    }
})
router.delete('/remove/:id', auth, async(req, res) => {
    await req.user.removeFromCart(req.params.id);
    const user = await req.user.populate('cart.items.productId');
    const products =  mapCartItems(user.cart);
    const cart = {
        products, price: countPrice(products)
    }
    res.status(200).json(cart);
})

router.delete('/remove/all/:id', auth, async(req, res) => {
    await req.user.fullyRemoveProductFromCart(req.params.id);
    const user = await req.user.populate('cart.items.productId');
    const products =  mapCartItems(user.cart);
    const cart = {
        products, price: countPrice(products)
    }
    res.status(200).json(cart);
})
router.get('/', auth, async (req, res) => {
    const user = await req.user
    .populate('cart.items.productId');
    console.log(user.cart);
    const products = mapCartItems(user.cart);

    res.render('cart', {
        title : 'Cart',
```

```
        isCart : true,
        products : products,
        price : countPrice(products)
    })
})



module.exports = router;
```

/router/incomingOrders.js

```
const {Router} = require('express');
const Order = require('../models/order');
const Product = require ('../models/product.js');
const auth = require('../middleware/auth');
const { all } = require('express/lib/application');
const router = Router();

router.put('/:id/status/:status', auth, async (req, res) => {
    try{
        console.log(req.params);
        await Order.findByIdAndUpdate(req.params.id , {status :
req.params.status});
        res.send(200)
    } catch(e){
        console.log(e)
        res.send(500)
    }
})
```

```
router.get('/count', auth, async(req,res) => {
    try{

        const countIncOrders =  (await Order
        .find({products:{$elemMatch:{'product.userId' : req.user._id}},status :
'New'})
        .populate('user.userId')).length
        console.log(countIncOrders);
        res.setHeader('Content-Type', 'application/json');
        res.end(JSON.stringify({countIncOrders}));

    }catch(e){
         console.log(e);
         res.send(500);
    }
})



router.get('/', auth, async (req, res) => {
    try{
         const incOrders = await Order
         .find({products:{$elemMatch:{'product.userId' : req.user._id}},status :
'New'})
         .populate('user.userId');

         res.render('incomingOrders', {
             isIncomingOrders: true,
             title: 'Incoming Orders',
             incOrders: incOrders.map(ord => {
```

```
            return {
                ...ord._doc,
                price: ord.products.reduce((total, c) => {
                    return total += c.count * c.product.price
                }, 0)
            }
        })
    })

  }catch(e){
    console.log(e);
  }

})

module.exports = router;
```

/router/orders.js

```
const {Router} = require('express');
const Order = require('../models/order');
const auth = require('../middleware/auth');
const router = Router();

router.get('/', auth, async (req, res) => {
  try{
    const orders = await Order.find({'user.userId': req.user._id})
    .populate('user.userId')

    res.render('orders', {
      isOrder: true,
```

```
            title: 'Orders',
            orders: orders.map(o => {
                return {
                    ...o._doc,
                    price: o.products.reduce((total, c) => {
                        return total += c.count * c.product.price
                    }, 0)
                }
            }).reverse()
        })
    }catch(e){
        console.log(e);
    }

})

router.post('/', auth, async (req, res) => {
    try{
        const user = await req.user
        .populate('cart.items.productId');
        const products = user.cart.items.map(i => ({
            count: i.count,
            product: {...i.productId._doc}
        }))

        const order = new Order({
            user: {
                name: req.user.name,
                userId: req.user
```

```
            },
            products : products.reverse()
        })

        await order.save();
        await req.user.clearCart();

        res.redirect('/orders');
    }catch (e) {
        console.log(e)
    }
})
module.exports = router;
```

/router/products.js

```
const {Router} = require('express');
const Product = require ('../models/product.js');
const auth = require('../middleware/auth');
const router = Router();

router.get('/', async (req, res) => {
    let products = await Product.find()
    .populate('userId', 'email name')
    .select('price title img category about');

    if(req.user){
        products = products.filter(p => p.userId._id.toString() !=
req.user._id.toString());
    }
```

```
    res.render('products', {
        title : 'Products',
        isProducts : true,
        products : products.reverse()
    });
})

router.get('/filter', async (req, res) => {
    console.log(req.query);
    let {categories, order, offset, limit, title} = req.query;
    categories = isBlank(categories) ? undefined : categories.split(',');
    order = isBlank(order) ? undefined : order;
    console.log('ADIDHIDN : ' + categories);

    let products = await Product.find()
    .populate('userId', 'email name')
    .select('price title img category about');
    if(req.user){
        products = products.filter(p => p.userId._id.toString() !=
req.user._id.toString());
    }

    if(categories){
        products = products.filter(p => p.category !== undefined
        && categories.indexOf(p.category.toString()) !== -1)
    }
    console.log('ORDER : ' + order);
    if(order){
        if(order === 'asc'){
```

```
            products = products.sort((p1,p2) => p1.price - p2.price);
        }else{
            products = products.sort((p1,p2) => p2.price - p1.price);
        }
    }

    if(!isBlank(title)){
        products = products.filter(p =>
p.title.toLowerCase().includes(title.toLowerCase()));
    }

    let count = products.length;

    let startProductIndex = offset*limit;
    let endProductIndex = parseInt(startProductIndex) + parseInt(limit);
    products = products.reverse().slice(startProductIndex, endProductIndex);


    res.setHeader('Content-Type', 'application/json');
    res.end(JSON.stringify({products, count, isAuth :
req.session.isAuthenticated}));
})

router.get('/:id/edit',auth, async (req, res) => {
    if(!req.query.allow){
        return res.redirect('/');
    }

    const product = await Product.findById(req.params.id);
```

```
    res.render('product-edit', {
        title: `Edit ${product.title}`,
        product
    })
})

router.post('/edit', auth, async (req, res) => {
    const {id} = req.body;
    delete req.body.id;
    await Product.findByIdAndUpdate(id, req.body);
    res.redirect('/products');
})

router.post('/remove', auth, async(req, res) => {
    try{
        await Product.deleteOne({
            _id: req.body.id
        })
        res.redirect('/products');
    }catch(e){
        console.log(e);
    }

})

router.get('/:id', async (req, res) => {
    const product = await Product.findById(req.params.id)
    .populate('userId', 'email phone name');
```

```
    let similarProducts = await Product.find({category :
product.category}).limit(7);
    similarProducts = similarProducts.filter(p => p.id !== req.params.id);
    console.log(similarProducts.length);

    res.render('product', {
        layout: 'empty',
        title : `Product ${product.title}`,
        product,
        similarProducts
    });
})

function isBlank(str) {
    return (!str || /^\s*$/.test(str));
}

module.exports = router;
```

add.hbs

```
<div class="row">
    <div class="row col s6 offset-s3">
        <h1 >Створіть оголошення</h1>
    </div>

    <form id="uploadForm" action="/add" method="POST">
    <div class="form-style col s6 offset-s3">

            <div class="input-field col s5 offset-s1">
```

```
            <input id="title" name = "title" type="text" class="validate"
required>
            <label for="title">Назва товару</label>
            <span class="helper-text" data-error="Введіть назву"></span>
          </div>

          <div class="input-field col s5">
            <input id="price" name = "price" type="number" class="validate"
required min="10">
            <label for="price">Ціна</label>
            <span class="helper-text" data-error="Введіть ціну"></span>
          </div>

          <div class="input-field col s5 offset-s1">
            <select id = "category">
              <option value="" disabled selected>Виберіть
категорію</option>
              <option value="Збірні букети">Збірні букети</option>
              <option value="Рози">Рози</option>
              <option value="Тюльпани">Тюльпани</option>
              <option value="Орхідеї">Орхідеї</option>
              <option value="Піони">Піони</option>
              <option value="Лілії">Лілії</option>
              <option value="Ромашки">Ромашки</option>
              <option value="Гортезії">Гортезії</option>
              <option value="Хризантеми">Хризантеми</option>
              <option value="Іриси">Іриси</option>
            </select>
            <label>Категорії</label>
```

```
        </div>

        <div class="input-field col s5 offset-s1">
          <textarea id="about" class="materialize-textarea"></textarea>
          <label for="about">Опишіть товар</label>
        </div>

        <div class="input-field col s10 offset-s1">
          <div class = "file-field input-field">
            <div class = "btn">
              <span>Browse</span>
              <input id="imageFile" type="file" accept="image/png,
image/gif, image/jpeg" multiple>
            </div>

            <div class = "file-path-wrapper">
              <input class = "file-path validate" type = "text"
              placeholder = "Завантажте декілька фотографій" />
            </div>
            <div id="preview">

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

      <div class="col s6 offset-s5">
        <button class="btn btn-primary" type="submit">Додати</button>
      </div>
    </div>
```

```
    </form>
</div>
<script src="http://localhost:3000/js/addProduct.js"
type="application/javascript"></script>
```

products.hbs

```
    {{> navbar}}
    <div class="row">
        <div class = "col s4">
            <h5>{{product.title}}</h5>
            <div class="carousel carousel-slider">
                {{#each product.img}}
                <a class="carousel-item"><img src="/media/{{filename}}"></a>
                {{/each}}
            </div>
            <div class="box-images">
                {{#each product.img}}
                    <img src="/media/{{filename}}" class="materialboxed
product-images" width="650">
                {{/each}}
            </div>
        </div>
        <div class="col s3">
            <div class="row info-pannel">
                <div class="col s5">
                    <h6>Ціна: <span class="price
big">{{product.price}}грн.</span></h6>
                    <form action="/cart/add" method="POST">
                        <input type="hidden" name="id" value="{{product.id}}">
```

```
                <button type="submit" class="btn btn-primary pulse offset-
s4">Купити</button>
            </form>
        </div>
        <div class="col s12">
            <div class="card-panel blue lighten-4">
                <span>{{product.about}}</span>
            </div>
        </div>

        <div class="col s12 ">
            <div class="card-panel blue lighten-4">
                <span>Зв'яжіться з продавцем
{{product.userId.name}}</span>
                <p class="valign-wrapper"><i class="material-icons
prefix">email</i>{{product.userId.email}}</p>
                <p class="valign-wrapper"><i class="material-icons
prefix">phone</i>{{product.userId.phone}}</p>
            </div>
        </div>
    </div>
</div>
<div class="col s5 ">
    <div class="row">
        <h5>Товари за цією категорією</h5>
        {{#if similarProducts.length}}
        {{#each similarProducts}}
        <div class="col s6">
            <div class="card hoverable medium sticky-action">
```

```
                <div class="card-image waves-effect waves-block waves-
light">
                    <img class="activator" src="/media/{{img.0.filename}}"
alt="{{title}}">
                </div>
                <div class="card-content ">
                    <span class="card-title activator grey-text text-darken-
4">{{title}}</span>
                    <p class="price">{{price}}грн.</p>
                </div>
                <div class="card-reveal">
                    <span class="card-title grey-text text-darken-
4">{{title}}<i
                            class="material-icons right">close</i></span>
                    <p>{{about}}</p>
                </div>
                <div class="card-action actions">
                    <a href="/products/{{id}}" target="_blank">Open</a>
                    {{#if @root.isAuth}}
                    <form action="/cart/add" method="POST">
                        <input type="hidden" name="id" value="{{id}}">
                        <button type="submit" class="btn btn-
primary">Buy</button>
                    </form>
                    {{/if}}
                </div>
            </div>
        </div>
        {{/each}}
```

```
            {{/if}}
          </div>
        </div>

    </div>

<script src="http://localhost:3000/js/product.js"
type="application/javascript" ></script>
```

orders.hbs

```
<div class="container">
<h1>Мої замовлення</h1>


{{#if orders.length}}
    {{#each orders}}
        <div class="row">
            <div class="col s6 offset-s3">
                <div class="card order-card">
                    <div class="card-content">
                        <span class="card-title">
                            Замовлення: <small>{{_id}}</small>
                        </span>
                        <p class="date">{{date}}</p>


                        <ol>
                            {{#each products}}
                                <li>
                                    <a
```

```
href="http://localhost:3000/products/{{product._id}}">{{product.title}}</a
> (x<strong>{{count}}</strong>)

                    </li>
                {{/each}}
            </ol>

            <hr>
            <p>Ціна: <span class="price">{{price}}</span> <span
class="badge">Status :{{status}}</span></p>
          </div>
        </div>
      </div>
  {{/each}}
{{else}}
  <p>You do not have any orders</p>
{{/if}}
</div>
```

cart.hbs

```
<div class="container">
<h1>Корзина</h1>
<div id="cart">
{{#if products.length}}
  <table class="flow-text">
    <thead>
      <tr>
        <th>Назва товару</th>
        <th>Кількість</th>
```

```
            <th>Дія</th>
          </tr>
        </thead>
        <tbody>
          {{#each products}}
          <tr id="cart-position-{{id}}">
            <td >{{title}}</td>
              <td>
                <span class="valign-wrapper">
                  <a class="btn-floating btn-small waves-effect waves-
light"><i class="material-icons minus cart-icons" data-
id="{{id}}">remove</i></a>
                    <span id="count-{{id}}"> {{count}}</span>
                  <a class="btn-floating btn-small waves-effect waves-
light"><i class="material-icons plus cart-icons" data-
id="{{id}}">add</i></a>
                </span>
              </td>
            <td>
              <button class="btn btm-small js-remove" data-
id="{{id}}">Видалити</button>
            </td>
          </tr>
          {{/each}}
        </tbody>
    </table>
    <p><strong>Ціна:</strong><span class="price">{{price}}</span></p>

    <form action="/orders" method="POST">
```

```
        <button type="submit" class="btn" >Замовити</button>
    </form>
{{else}}
<p>Cart is empty</p>
{{/if}}
</div>
</div>

<script src="http://localhost:3000/js/cart.js" ></script>
```

js/products.js

```
let priceOrderOpt;
let categoriesOpt;
let itemsPerPageOpt;
let paginationButtons;
let search;
let currentOffset = 0;


document.addEventListener('DOMContentLoaded', function(){

    var elems = document.querySelectorAll('select');
    var instances = M.FormSelect.init(elems);
    categoriesOpt =
M.FormSelect.getInstance(document.getElementById('category'));
    priceOrderOpt = document.getElementById('order')
    itemsPerPageOpt = document.getElementById('limit')
    search = document.querySelectorAll('.searchBar');
    console.log(search);
```

```
    search.forEach(search => {
      search.addEventListener('keypress', function(e){
        if(e.key === 'Enter'){
          findProducts(currentOffset);
        }
      })
    });

    document.querySelector('#find').addEventListener('click', () => {
      findProducts(currentOffset);
    });

    findProducts(currentOffset);
});


function rerenderProducts(products, isAuth){

    let productsDiv = document.getElementById('products');
    productsDiv.innerHTML = "";
    for(let index in products){
        const product = products[index];
                let productHtml =
                `<div class="col s3">
                  <div class="card hoverable medium sticky-action" >
                    <div class="card-image waves-effect waves-block waves-
light">
                      <img class="activator responsive-img""
src="/media/${product.img[0].filename}" alt = "${product.title}">
```

```
                </div>
                <div class="card-content ">
                  <span class="card-title activator grey-text text-darken-4
truncate">${product.title}</span>
                  <p class="price">${product.price}грн.</p>
                </div>
                <div class="card-reveal">
                  <span class="card-title grey-text text-darken-
4">${product.title}<i class="material-icons right">close</i></span>
                  <p>${product.about}</p>
                </div>
                <div class="card-action actions">
                  <a href="/products/${product._id}"
target="_blank">Відкрити</a>
                  `


                  if(isAuth){
                  productHtml +=
                  `
                    <form action="/cart/add" method="POST">
                      <input type="hidden" name="id" value="${product._id}">
                      <button type="submit" class="btn btn-
primary">Купити</button>
                    </form>
                    `
                  }
                  productHtml +=
                  `
                </div>
```

```
            </div>
          </div>`
          console.log(typeof productHtml);
          productsDiv.innerHTML += productHtml;
        }
    }

function renderPagination(count, offset){
  currentOffset = offset;
  let paginationTop = document.getElementById('paginationTop');
  let paginationBottom = document.getElementById('paginationBottom');
  let paginationButtonsCount = count/limit.value;
  paginationTop.innerHTML = "";
  paginationBottom.innerHTML = "";
  for(let i = 0; i<paginationButtonsCount; i++){
    let page;
    if(i == offset){
      page = `
      <li class="waves-effect pagination-button active" ><a id="page-${i}"
class="pagination-button">${i+1}</a></li>
    `
    }else{
      page = `
      <li class="waves-effect pagination-button" ><a id="page-${i}"
class="pagination-button">${i+1}</a></li>
    `
    }
    paginationTop.innerHTML += page;
    paginationBottom.innerHTML +=page;
```

```
}
let liTopRightArrow = document.createElement('li');
let aTop = document.createElement('a');
let iTop = document.createElement('i');
liTopRightArrow.setAttribute('class', 'waves-effect arrowRight');
aTop.setAttribute('href','#!');
iTop.setAttribute('class', 'material-icons');
iTop.textContent = 'chevron_right';
aTop.appendChild(iTop);
liTopRightArrow.appendChild(aTop);

let liTopLeftArrow = document.createElement('li');
let aTopLeftArrow = document.createElement('a');
let iTopLeftArrow = document.createElement('i');
liTopLeftArrow.setAttribute('class', 'waves-effect arrowLeft');
aTopLeftArrow.setAttribute('href','#!');
iTopLeftArrow.setAttribute('class', 'material-icons');
iTopLeftArrow.textContent = 'chevron_left';
aTopLeftArrow.appendChild(iTopLeftArrow);
liTopLeftArrow.appendChild(aTopLeftArrow);

let liBotRightArrow = document.createElement('li');
let aBot = document.createElement('a');
let iBot = document.createElement('i');
liBotRightArrow.setAttribute('class', 'waves-effect arrowRight');
aBot.setAttribute('href','#!');
iBot.setAttribute('class', 'material-icons');
iBot.textContent = 'chevron_right';
aBot.appendChild(iBot);
```

```
liBotRightArrow.appendChild(aBot);

let liBotLeftArrow = document.createElement('li');
let aBotLeftArrow = document.createElement('a');
let iBotLeftArrow = document.createElement('i');
liBotLeftArrow.setAttribute('class', 'waves-effect arrowLeft');
aBotLeftArrow.setAttribute('href','#!');
iBotLeftArrow.setAttribute('class', 'material-icons');
iBotLeftArrow.textContent = 'chevron_left';
aBotLeftArrow.appendChild(iBotLeftArrow);
liBotLeftArrow.appendChild(aBotLeftArrow);

paginationTop.lastElementChild.after(liTopRightArrow);
paginationTop.firstElementChild.before(liTopLeftArrow);
paginationBottom.lastElementChild.after(liBotRightArrow);
paginationBottom.firstElementChild.before(liBotLeftArrow);

arrowsLeft = document.querySelectorAll('.arrowLeft');
arrowsRight = document.querySelectorAll('.arrowRight');
arrowsLeft.forEach(arrow => {
  arrow.addEventListener('click', () => {
    if(currentOffset != 0){
      findProducts(currentOffset-1)
    }
  })
})
arrowsRight.forEach(arrow => {
  arrow.addEventListener('click', () => {
    if(currentOffset < paginationButtonsCount-1){
```

```
        findProducts(parseInt(currentOffset)+1);
      }
    })
  })

  paginationButtons = document.querySelectorAll('.pagination-button');

  for (let i = 0; i < paginationButtons.length; i++) {
    paginationButtons[i].addEventListener('click', (event) => {
      let offset = event.target.id.slice(5);
      findProducts(offset);
    });
  }
}


function findProducts(offset){

  let categories = categoriesOpt.getSelectedValues();
  console.log(categories);
  let order = priceOrderOpt.options[priceOrderOpt.selectedIndex].value;
  let limit =
itemsPerPageOpt.options[itemsPerPageOpt.selectedIndex].value;
  let title = search[0].value;
  title = title ? title : "";


fetch(`/products/filter?categories=${categories}&&order=${order}&&offset
=${offset}&&limit=${limit}&&title=${title}`)
```

```
  .then((res) => res.json())
  .then((data) => {
      console.log(data);
      let {products, count, isAuth} = data;
      rerenderProducts(products, isAuth);
      renderPagination(count, offset);
  })
  .catch((err) => {
      console.log(err);
  })
}
```

ДОДАТОК Б

ВІДГУК

ЕКОНОМІЧНОГО РОЗДІЛУ

на кваліфікаційну роботу бакалавра
на тему:
" Розробка платформи он-лайн оголошень, що об'єднує людей для покупки, продажу товарів та послуг з використанням серверного JavaScript"
студента групи 122-18-3 Лисяка Дмитра Олександровича

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Диплом_Лисяк.docx	Пояснювальна записка до кваліфікаційної роботи . Документ Word.
Диплом_Лисяк.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откопмпільовану програму
Презентація	
Презентація_Лисяк.pptx	Презентація кваліфікаційної роботи