Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Мантура Владислава Андрійовича
Прізвище, ім'я, по батькові студента

на здобуття ступеня вищої освіти Бакалавра

Вебресурс для продажу комп'ютерних товарів
Назва теми

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ. 200167.01.14.ПЗ

Виконав студент IV курсу група ІПЗ-20-1 _____ Владислав МАНТУР
Підпис Ім'я, ПРІЗВИЩЕ

Керівник старший викладач _____ Ганна БЕДРАТЮК
Науковий ступінь, звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер канд. пед. наук., доцент _____ Наталія ПРАВОРСЬКА
Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ім'я, ПРІЗВИЩЕ

6 червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Л. П. Бедратюк
2.01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Мантуру Владиславу Андрійовичу

Прізвище, ім'я, по батькові студента

1. Тема кваліфікаційної роботи Вебресурс для продажу комп'ютерних товарів

Керівник проєкту (роботи) Бедратюк Ганна Іванівна, старший викладач

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 02.01.2024 р. № 6
2. Строк подання студентом роботи на кафедру 01.06.2024
3. Вихідні дані до роботи Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задачі, проєктування програмного забезпечення, програмна реалізація, тестування програмного забезпечення
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Презентаційні матеріали (слайди, 20 шт.) Три креслення (блок-схема алгоритму Левенштейна для неточного пошуку, загальна діаграма потоків даних, схема компоновки елементів архітектури)

6. Консультанти розділів дипломного проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Праворська Н.І., доцент	30.05.24	06.06.24
Антиплагіат	Форкун Ю. В., доцент	31.05.24	06.06.24

7. Дата видачі завдання « 02 » січня 2024р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Ознайомлення з тематикою дипломного проєктування (ДП), визначення та узгодження індивідуальних тем кваліфікаційних робіт (КвР)	01.12 – 31.12.2023	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розроблення технічного завдання	01.01 – 20.02.2024	
3 Проєктування програмного забезпечення	21.02 – 20.03.2024	
4 Програмна реалізація з використанням відповідних засобів розробки. Тестування ПЗ	21.03 – 30.04.2024	
5 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог стандартів	01.05 – 25.05.2024	
6 Попередній захист КвР	Травень 2024	Згідно графіка
7 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 – 30.05.2024	
8 Здача КвР на кафедру; підготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР	з 01.06.2024	

Студент _____ Владислав МАНТУР
Підпис Ініціали, ПРІЗВИЩЕ

Керівник роботи _____ Ганна БЕДРАТЮК
Підпис Ініціали, ПРІЗВИЩЕ

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ. 200167.01.14.ПЗ	Пояснювальна записка	78		
2	А4		Завдання на кваліфікаційну роботу	2		
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	20		

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ. 200167.01.14.ПЗ		
Виконав		Мантур В.А.		06.06	Вебресурс для продажу комп'ютерних товарів	Літ.	Арк. 4 / Аркушів 78
Керівник		Бедратюк Г.І.		06.06			
Н. Контр.					Відомість документів	ХНУ, ІПЗ-20-1	
Зав. Каф.		Бедратюк Л.П.		06.06			

ЗМІСТ

ВСТУП

Історію людства можна розглядати як процес зростання потреб у зберіганні та обробці інформації. Спочатку люди зображували події на стінах печер, пізніше винайшли мову для вираження ідей і покращення умов життя. Зі збільшенням обсягу інформації виникла потреба у її зберіганні в надійних місцях, як-от книги. У добу Відродження люди почали масово поширювати знання, винаходячи пристрої для обробки даних, такі, як друкарські машини. Сьогодні ми живемо в епоху, коли знання всього світу доступні через інтернет.

Часи інтернету дають можливість оперувати інформацією настільки ефективно, що географічні кордони та часові обмеження майже зникають. В сучасному цифровому світі вебзастосунки стали невід'ємною частиною життя, пропонуючи послуги від соціальних мереж і розваг до електронної комерції.

Однією з найпопулярніших категорій інтернет-застосунків є системи електронної комерції, які дозволяють користувачам здійснювати покупки онлайн без необхідності відвідування фізичних магазинів.

Популярність онлайн-торгівлі продовжує зростати через зручність, доступність і широкий вибір продуктів, які вона пропонує. За останні роки глобальні події, такі як пандемія COVID-19, значно прискорили перехід споживачів до онлайн-покупок, що спонукало бізнеси активніше інтегрувати цифрові технології у свої комерційні операції.

Цей перехід до цифрової комерції створив попит на високоефективні, надійні та безпечні вебзастосунки, здатні задовольнити зростаючі вимоги споживачів і підтримувати високий рівень конкурентоспроможності. Основні технологічні рішення в цьому напрямку включають розробку інтуїтивно зрозумілих інтерфейсів користувача, забезпечення надійності транзакцій та захисту даних, а також інтеграцію із сучасними платіжними системами та логістичними сервісами.

У світі, де діджиталізація охоплює всі аспекти життя, електронна комерція стає не тільки зручним, але й необхідним інструментом для розвитку бізнесу. Особливо це стосується ринку комп'ютерної техніки, де швидкість, доступність інформації про товари та зручність в їх придбанні відіграють ключову роль для залучення та утримання клієнтів.

Актуальність розробки вебзастосунків для електронної комерції обумовлена низкою постійних змін глобальних трендів у споживацьких вподобаннях, що змушує дану сферу постійно розвиватись та адаптуватись Це спонукає бізнеси адаптувати свої стратегії до цифрового формату, щоб задовольнити потреби різних клієнтів і залишатися конкурентоспроможними на ринку, за для збільшення продажів.

Зростання мобільних технологій та широкий доступ до інтернету суттєво впливають на поведінку споживачів, зокрема на їхні звички здійснювати покупки. Мобільні пристрої, такі як смартфони та планшети, стають все більш популярними інструментами для онлайн-шопінгу завдяки їхній зручності, портативності та постійному доступу до інтернету. Ця тенденція сприяє розвитку та поширенню онлайн-платформ, які адаптуються до мобільних пристроїв і забезпечують споживачам зручний та ефективний досвід покупок.

Також, зростання вимог до персоналізації та індивідуалізації покупок стимулює подальший розвиток технологій у сфері веброзробки. Споживачі хочуть не просто купувати товари, а отримувати персоналізований досвід, що відповідає їхнім унікальним потребам і вподобанням, і сучасні вебзастосунки можуть надати це, інтегруючи штучний інтелект та машинне навчання для аналізу даних користувачів.

Вебресурси даної направленості забезпечують користувачам зручний доступ до широкого асортименту комп'ютерних товарів, що сприяє підвищенню рівня обслуговування та задоволеності клієнтів. Комерційні вигоди, в свою чергу, включають збільшення продажів та оптимізацію процесів онлайн-торгівлі.

Об’єктом дослідження даної КРБ є процес електронної комерції у сфері продажу комп’ютерної техніки. Це включає в себе взаємодію між продавцем та покупцем, процеси вибору, покупки та доставки товарів через інтернет.

В свою чергу, предметом дослідження є вебзастосунок для продажу комп’ютерних товарів, його архітектура, функціональність, інтерфейс користувача та технічне забезпечення. Окрім цього, до предмету дослідження відносяться основні компоненти сайту, такі як система управління контентом, інтеграція з базою даних, інструменти для адміністрування, методи реалізації різного функціоналу.

Метою ж кваліфікаційної роботи є розроблення функціонального та ефективного вебресурсу для продажу різних комп’ютерних товарів, забезпечивши в ньому функціонал, який відповідає сучасним тенденціям та базовим потребам користувачів.

Таким чином, поставлені задачі для даної кваліфікаційної роботи бакалавра можна визначити наступним чином:

– проаналізувати поточні тенденції та особливості ринку електронної комерції комп’ютерної техніки;

– визначити вимоги та очікування цільової аудиторії щодо функціональності та зручності вебресурсу;

– розробити архітектуру та дизайн вебсайту, забезпечивши інтуїтивно зрозумілий інтерфейс і легку навігацію;

– реалізувати вебресурс з використанням вибраних технологій;

– протестувати вебсайт для забезпечення його надійності і швидкодії;

– провести аналіз ефективності нового вебресурсу та визначити потенційні напрямки для подальшого розвитку.

1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Вебресурс – це ресурс, доступ до якого здійснюється через Інтернет. Він може бути сайтом, порталом, застосунком, чи інтернет-магазином. Такі ресурси включають різний контент, наприклад текст, зображення, відео, аудіо, та різного роду анімацію [1].

Вебресурсом, або як їх прийнято називати сайтом, є будь-яким цифровий актив, до якого можна отримати доступ через інтернет [2]. Вебсайти можуть бути статичними, пропонуючи фіксований набір вмісту, або динамічними, що дозволяє даним оновлюватися в реальному часі [3]. Вебпортали зазвичай слугують точками доступу до широкого спектру ресурсів та служб, інтегрованих на єдиній платформі [4]. На сьогодення, сайти відіграють неймовірно важливу роль у сучасному світі, надаючи різноманітні можливості для спілкування, навчання, розваг та комерційної діяльності.

Основними класифікаціями вебсайтів є: блог (вебжурнал), стрімінгові сервіси, бізнес-сайт та корпоративний вебсайт, вебсайт спільноти, змістовий вебсайт та інформаційний вебсайт, вебсайт електронної комерції, ігровий вебсайт, урядовий вебсайт, вебсайт новин, персональний вебсайт, навчальний вебсайт, ресурси навчальних закладів [5].

Серед вебресурсів, комерційні платформи користуються значною популярністю. Наприклад, згідно зі звітом eCommerce Foundation за 2023 рік [6], близько 63% споживачів в Європі віддають перевагу онлайн-покупкам замість відвідування фізичних магазинів. Це число продовжує зростати зі збільшенням доступності інтернету і розвитком електронної комерції. Вебмагазини, онлайн-маркетплейси, та електронні комерційні платформи стали не тільки зручними, але й економічно ефективними способами ведення бізнесу [7].

Одним із ключових аспектів успіху комерційних вебресурсів є здатність до швидкої адаптації до змін у споживацьких потребах та використання даних для особистого підходу до кожного клієнта [8]. Ці особливості дозволяють не тільки збільшувати продажі, але й зміцнювати лояльність клієнтів.

Зазвичай, вебресурси застосовуються в бізнесі для продажу товарів. Онлайн-платформи класифікують так [9]: онлайн-магазини, електронні комерційні платформи, маркетплейси, сервіси цифрового контенту, платформи для продажу послуг, вебаукціони, статичні сторінки (лендінги).

Комерційні вебресурси мають кілька спільних ключових функціональних зон: каталог продукції, систему управління контентом, корзину для покупок, систему обробки замовлень, та інструменти аналітики та звітності [10]. Кожна з цих зон вимагає інтеграції з внутрішніми та зовнішніми системами, зокрема з системами управління базами даних, платіжними шлюзами [11]. Зазвичай, даний спектр технологій використовується, як реалізація мінімальних потреб для можливих покупців.

Дійсно, зазвичай попит народжує пропозицію. Якщо користувачам комфортно використовувати лише функціонал, який надається, зникає потреба в розробці нових технологій та методик, витрачаючи на це час та гроші [12].

Однією з головних проблем даного типу вебзастосунків є те, що головні тенденції зазвичай встановлюються лише компаніями-гігантами, що свідчить про недостатній аналіз користувацьких потреб [13]. Введення кращого функціоналу, може дозволити збільшувати користувацький досвід, тим самим покращуючи продажі та зацікавлюючи нових клієнтів [14].

Іноді, нехтується навіть автоматизацією процесів, таких як створення замовлень. Такі ресурси реалізуються, як лендінги [15] – вебсторінки, які зазвичай використовується для маркетингових або рекламних цілей, але при цьому фокусуються на вузькій кількості інформації і слугують лише для привернення уваги, або продажу товарів. Зазвичай комерційні діячі, які звертаються до цього типу вебсайтів, мають на меті з меншими витратами, продавати власний товар, здійснюючи це через дзвінки, або, в кращому випадку,

через анкетування. Даний підхід до користувачів, зазвичай оцінюється ними в поганому світлі [16].

Деякі продавці користуються послугами маркетплейсів. за для продажу з меншим використанням зусиль. Маркетплейс [17] – це вид онлайн-платформи, яка з'єднує продавців та покупців для здійснення комерційних угод. Це місце, де багато продавців можуть пропонувати свої товари або послуги різним клієнтам. Маркетплейси можуть бути загальними, пропонуючи широкий спектр товарів і послуг, або спеціалізованими, фокусуючись на конкретній категорії товарів або ринковому сегменті. Серед користувачів, маркетплейси також доволі популярні [18]. В даному випадку, з’являються переваги для однієї сторони, але це спричиняє проблеми для іншої.

Найбільш оптимальним типом комерційних вебресурсів є онлайн-магазини, які окрім того, що зазвичай спеціалізуються на певних категоріях товарів, надаючи користувачам переглядати вміст в сфері, яка їх цікавить, також функціонують як цілісні системи, в яких просто розширювати список можливостей та автоматизовувати процеси [19].

Хоча створення середньостатистичних онлайн-магазинів вимагає певних витрат на їх розробку та утримання, але розвиток технологій дозволяє покращувати будь-які типи застосунків, а особливо різного типу магазини, з мінімальними потребами до розробників [20]. Наприклад, на сьогодення, в західній частині Європи та в Північній Америці, активно використовують штучний інтелект для впровадження функціоналу, який покращує користувацький досвід [21]. Але, якщо переглядати вітчизняні програмні продукти, подібні нововведення це велика рідкість [22].

Важливо розуміти, що при розробці онлайн-магазинів, потрібно приділяти увагу не тільки покупцям, так як вони не єдина сторона, яка бере участь комерційних-відносинах при роботі з такого роду застосунками [23]. Варто забезпечувати комфортні умови для роботи з даними адміністрації, яка займається обробкою замовлень [24]. Як було сказано раніше, функціонування вебресурсів такого характеру не можливе без процесів додавання, редагування

даних про товари і покупців. Отже, для забезпечення нормальних процесів, потрібно враховувати потреби обох видів користувачів: покупців і адміністрації.

Враховуючи, що онлайн-покупки обов'язково передбачають передачу особистих даних, такі як імена, адреси, дані платіжних систем, забезпечення безпеки – це критично важливий компонент [25]. Автентифікація та авторизація є частинами безпеки для будь-яких онлайн-платформ даного класу, особливо коли йдеться про електронну комерцію і вебдодатки, що займаються продажем товарів [26]. Один з найвідоміших випадків витоку даних користувачів стався з компанією Target у 2013 році, коли хакери отримали доступ до даних понад 40 мільйонів банківських карток клієнтів. Хакери використали вразливості в мережевій безпеці, щоб встановити шкідливе програмне забезпечення на пунктах продажу. Цей випадок підкреслює важливість ретельного захисту інформаційних систем від несанкціонованого доступу та важливість регулярного оновлення безпеки для захисту даних.

Отже, можна зробити висновок, що для електронної комерції найкращий вибір для середньостатистичних продавців – це створення онлайн-магазинів Головною проблематикою сучасних комерційних вебресурсів є те, що вони не використовують достатню кількість сучасних клієнт-орієнтованих технологій, таких як наприклад штучний інтелект. Також, нехтування безпекою користувачів та їх даних, прямопропорційно впливає на зацікавленість можливих клієнтів у подібного роду ресурсів.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Функціонування сучасних онлайн-магазинів майже не можливе без використання автоматизації процесів, таких як обробка запитів від покупців, формування списку товарів, рекомендацій та інших. Більшість існуючих застосунків намагають лаконічно та ненав'язливо вводити власні інструменти у

структуру ресурсу, таким чином формуючи більшу привабливість та цікавість користувачів. Але, як уже зазначалось, більшість таких тенденцій сформовані великими компаніями, що має вплив і на онлайн-магазини. Саме тому важливим етапом перед проєктуванням застосунку є аналіз конкурентів, для визначення ідей, які можна реалізувати у власному вебресурсі.

В Україні наразі існує лише невелика кількість магазинів, що зосереджені виключно на продажі комп'ютерної техніки та її комплектуючих. Більшість таких товарів продаються через великі маркетплейси, де, як було зазначено раніше, можуть виникнути певні труднощі, коли мова йде про різну продукцію специфічного направлення.

Одним з найкращих і найпопулярніших прикладів онлайн-магазинів – це Citrus [27]. Його головну сторінку зображено на рисунку 1.1.

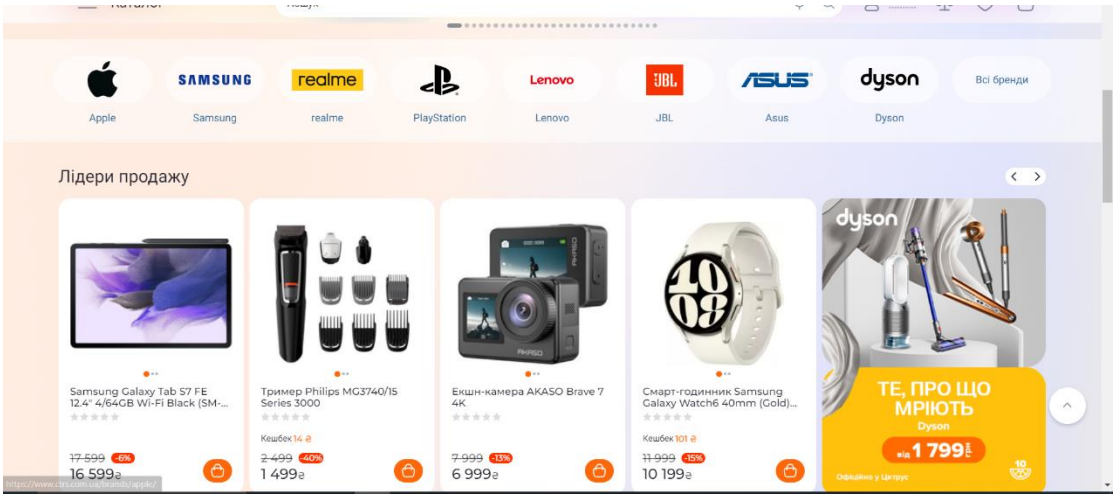


Рисунок 1.1 – Головна сторінка магазину Citrus [27]

Citrus – це мережа магазинів, що надає широкий вибір товарів, включаючи комп'ютери, ноутбуки, смартфони та аксесуари. Основними перевагами є зручність доставки та опція самовивозу з філій. Додатково, існує програма лояльності для сталого клієнтського контингенту, яка включає знижки та спеціальні пропозиції. Також, як і в більшості інших магазинів, даний вебзастосунок має систему скидок.

Ефективним підходом є групування товарів на головній сторінці за категоріями, які особливо цікавлять споживачів, що демонструє Citrus шляхом виділення брендів. Це допомагає користувачам, зацікавленим у брендових та статусних продуктах, легко вибирати необхідні категорії.

Іншим прикладом можна назвати маркетплейс Rozetka [28]. Хоч це і не вузьконаправленний магазин, але при цьому даний магазин користується високою популярністю у користувачів інтернету, через його зручний функціонал. Головна сторінка даного маркетплейсу зображена на рисунку 1.2.

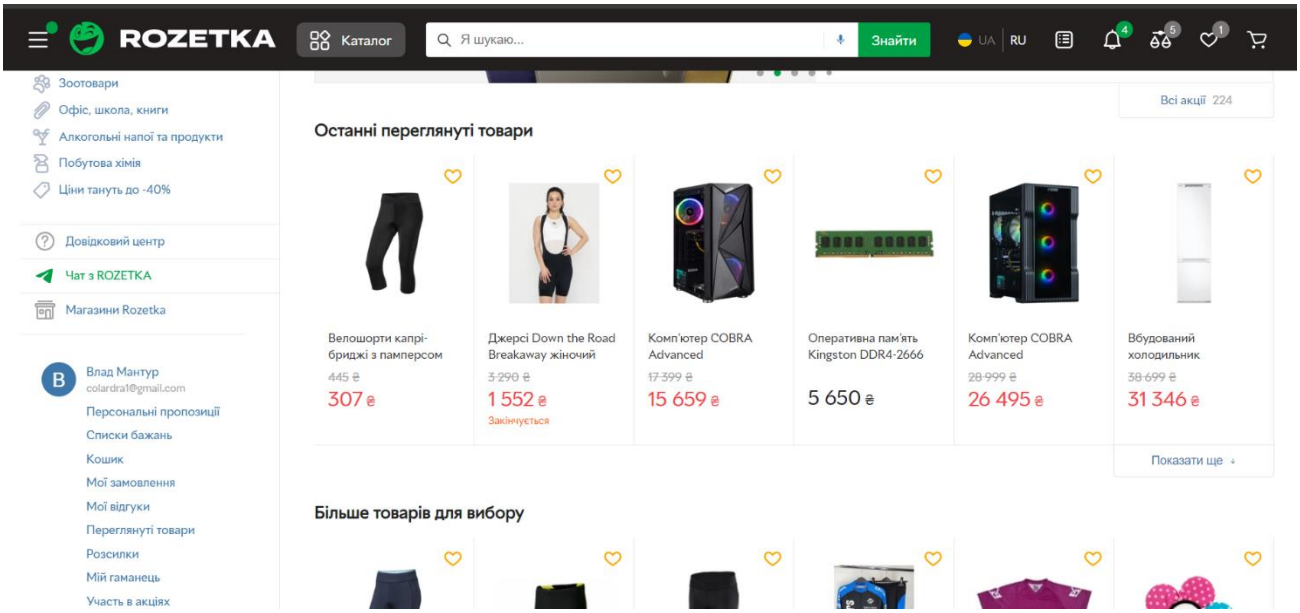


Рисунок 1.2 – Головна сторінка маркетплейсу Rozetka [28]

Rozetka є одним з провідних онлайн-магазинів в Україні, який пропонує широкий асортимент комп'ютерної техніки, електроніки та інших товарів. Відома своїм великим вибором продукції, Rozetka також забезпечує легкий доступ до покупок через свою інтернет-платформу. Особливо зручним є функціонал замовлень, де дані користувача та адреси автоматично заповнюються для спрощення процесу покупки

Rozetka здобула визнання серед споживачів не тільки завдяки широкому асортименту, але й через зручність своєї платформи, яка дозволяє користувачам легко використовувати навігацію та здійснювати покупки. Крім того, Rozetka

активно використовує інноваційні технології для підвищення ефективності свого сервісу, включаючи штучний інтелект для аналізу поведінки покупців та підбору індивідуальних рекомендацій.

Однак деякі клієнти висловлюють невдоволення щодо доставки в віддалені регіони та проблеми з обміном та поверненням товарів, які часто супроводжуються некомпетентною підтримкою асистентів.

Інша система, яка найбільше підходить до теми дипломної роботи – це система магазинів «ЧІП» [29] (Рисунок 1.3).

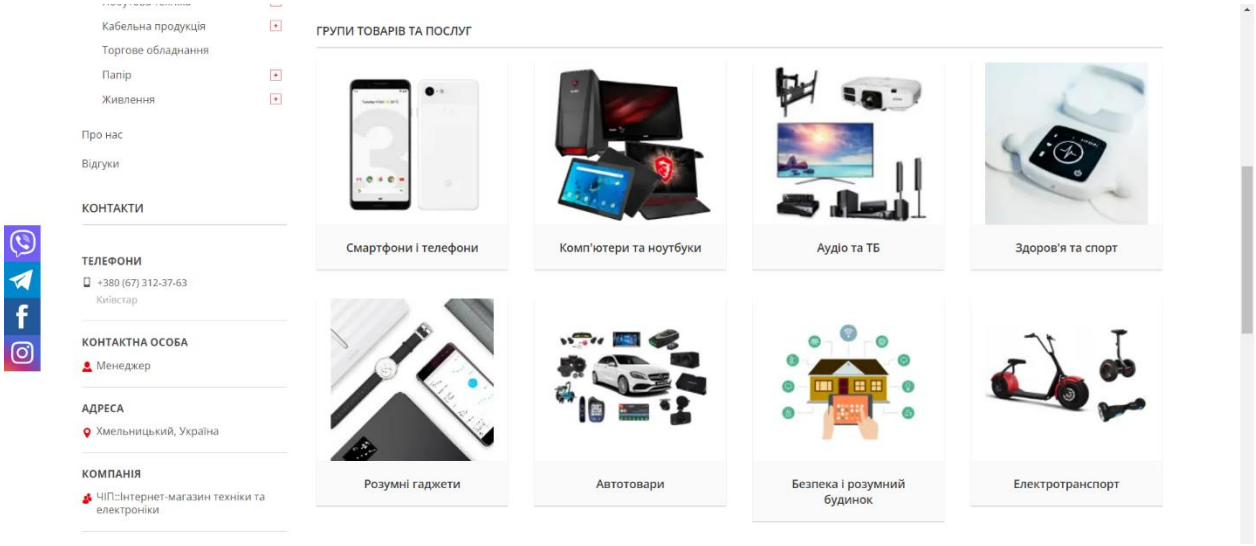


Рисунок 1.3 – Головна сторінка магазину «ЧІП» [29]

ЧІП — це мережа магазинів, яка зосереджена на продажу комп'ютерів, ноутбуків та комплектуючих. Вони пропонують професійні консультації фахівців для допомоги у виборі продукції та залучають клієнтів за допомогою знижок. ЧІП дозволяє здійснювати покупки як у фізичних магазинах, так і через онлайн-платформу.

Проте, слід врахувати, що асортимент у ЧІП може бути меншим у порівнянні з більшими роздрібними мережами, а ціни на деякі товари можуть бути вищими. При аналізі вебсторінок ЧІП можна виявити схожість з такими ресурсами як Rozetka, хоча функціонал їхнього сайту є більш обмеженим, але він задовольняє основні потреби покупців.

На основі обробленої інформації можна провести абстрагування інформації і винести основні пункти у таблицю, зображену нижче (Таблиця 1.1).

Таблиця 3.1 – Порівняльна характеристика описаних ресурсів

Назва вебресурсу	**Переваги**	**Недоліки**
Citrus	Широкий асортимент комп'ютерних товарів. Часті акції та знижки для клієнтів. Програма лояльності для покупців. Поділ на інтуїтивно зрозумілі категорії.	Можуть бути вищі ціни порівняно з іншими магазинами. Обмежений вибір товарів для покупців в деяких регіонах.
Rozetka	Великий вибір товарів усіх категорій. Зручна інтернет-платформа для покупок з швидкою доставкою. Використання ШІ для клєінтів. Зручна система автоматичного заповнення замовлень.	Можуть бути проблеми з доставкою у віддалені регіони. Можуть виникати проблеми з обміном та поверненням товарів. Погана підтримка клієнтів.
ЧІП	Спеціалізація на комп'ютерах, ноутбуках та комплектуючих. Асистенти для вибору товару. Часті акції та знижки на товари. Можливість купувати як в магазинах, так і онлайн.	Обмежений асортимент товарів порівняно з більшими магазинами. Можливість вищих цін на деякі товари.

Отже, проаналізувавши різні ресурси, можна зробити висновок про те, що ключові переваги, які цінують користувачі при виборі онлайн-магазину, охоплюють наявність широкого асортименту товарів, спеціалізацію на певних товарах і зручність онлайн платформ для покупок. Також, потрібно окремо виділити, що деякі платформи для зручності використовують штучний інтелект, як інструмент вза'ємодії з користувачами.

1.3 Визначення функціональних та нефункціональних вимог до програмного забезпечення

Одним з головних етапів при створенні програмного продукту є формування списку функцій, які мають бути реалізовані в подальшому. Визначивши переваги і недоліки уже існуючих застосунків, можна визначити основні функції застосунку.

Важливо почати з того, що продуктивність вебресурсу повинна забезпечувати швидке завантаження сторінок та плавну інтерактивність, щоб користувачі не відчували затримок під час перегляду товарів або оформлення покупок. Одним з рішень, має бути реалізована система динамічних вікон, які з’являтимуться для виконання простих дій користувачів.

Безпека є вирішальною, особливо коли справа доходить до обробки особистих даних клієнтів, що включає захист від несанкціонованого доступу та витоку даних. Даней елемент має забезпечуватись системами авторизації та автентифікації, а дані користувачів мають зберігатись на віддалених носіях.

Надійність системи повинна гарантувати, що вебресурс залишається працездатним і доступним у різних умовах, з мінімальним часом простою. Масштабованість важлива для підтримки зростаючого числа користувачів та об'ємів даних без втрати продуктивності.

Так як купівля комп’ютерних товарів може відбуватись людьми будь-якого віку, інтерфейс має забезпечувати просте використання базових функцій, потрібних для здійснення покупок. Інші функції можуть бути неявними і розраховані на досвідчених користувачів.

Передбачається, що в застосунку буде можливість взаємодіяти з товарами, замовленнями, системою реєстрації та авторизації. Відповідно до цього, можна поділити додаток на наступні групи користувачів: гості, зареєстровані користувачі та адміністратори. Гості – це користувачі, які не мають власного профілю в системі, тому мають доступ лише до функцій реєстрації та перегляду

товарів, їх сортування та фільтрування. Зареєстровані користувачі мають такий самий функціонал, як гості, але окрім цього можуть додавати товари в корзину, оформляти замовлення, мають доступ до параметрів профілю, можуть оцінювати товари, створювати списки власних адрес. Адміністрація – це особливий тип зареєстрованих користувачів, які мають доступ до взаємодії з функціями редагування, додавання та видалення товарів, моніторингу замовлень.

Таким чином, доступні функції користувачів можна розділити на категорії та створити діаграму варіантів використання для вебресурсу по продажу комп'ютерних товарів (Рисунок 1.4).

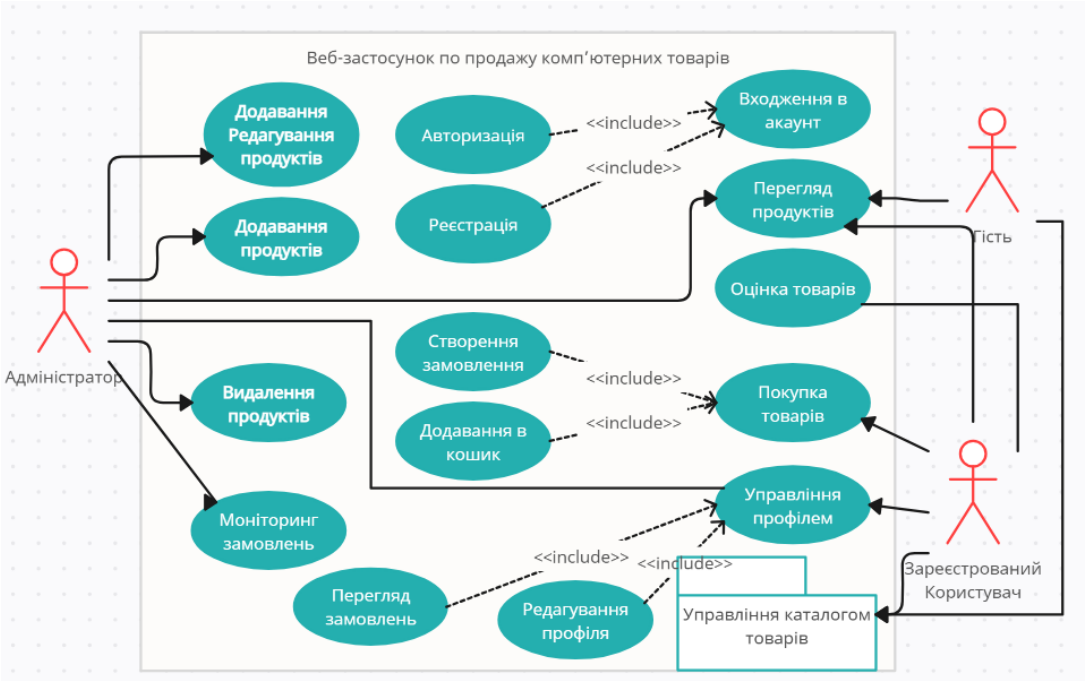


Рисунок 1.4 – Діаграма використання вебресурсу по продажу комп'ютерних товарів

Як видно на рисунку, найпростіший тип користувачів – це «Гості». Покупці, які не створили власного акаунту, мають можливість лише авторизуватись, зареєструватись, або переглядати товари. Це зумовлено тим, що якщо надати незареєстрованим користувачам робити замовлення, набагато важче буде відслідковувати зміни в його даних. Але, якщо покупець зацікавився даним

сайтом, йому має бути доступна можливість переглядати товари перед покупкою, щоб оцінити якість перед можливою покупкою.

Зареєстрованим користувачам не надана можливість авторизації та реєстрації по тій причині, що в застосунку не потрібна система мультиакаунтів, що дозволяє знехтувати комфортом переключення між профілями. Але при цьому користувач може в будь-який момент вийти з акаунту і авторизуватись, як інша особистість.

Коли користувач взаємодіє з товаром, йому має надаватись можливість додавання його у кошик і вибору кількості. Уже у кошику користувач має мати доступ до створення нового замовлення. Також у кошику будуть реалізовані системи редагування даних обраних позицій, включаючи видалення та зміну обраної кількості.

При створенні замовлення, дані користувача мають заповнюватись автоматично, але при цьому має надаватись можливість їх зміни, що визначити нового отримувача. Для можливих адрес, по прикладу деяких великих маркетплейсів, передбачається створення хмари допустимих значень. Іншими словами, коли користувач додає нову адресу, вона має зберігатись, як одна з тих, яку він може обрати знову в майбутньому.

Якщо користувач хоче змінити дані свого профілю, йому має надаватись відповідний функціонал на його особистій сторінці. Також на даній сторінці має бути забезпечена функція додавання нових адрес і їх видалення. Для покупців має передбачатись і можливість перегляду раніше зроблених замовлень.

Після покупки товару, користувачу надається право надавати йому оцінку. Це дозволяє покращити якість оцінювання, так як користувач надаватиме її лише отримавши фізичний об’єкт. Оцінка має надаватись по п’ятибальній системі і відображатись у зручному та звичному для користувачів форматі. Якщо багато користувачів оцінили один і той самий товар, мусить відображатись середньо арифметичний показник.

Для ефективної взаємодії з каталогом товарів, мусить бути розроблена навігаційна панель, для обробки списків товарів. Функціонал, який має

надаватись був відокремлена від загальної діаграми використання завдяки декомпозиції і зображена далі (Рисунок 1.5).

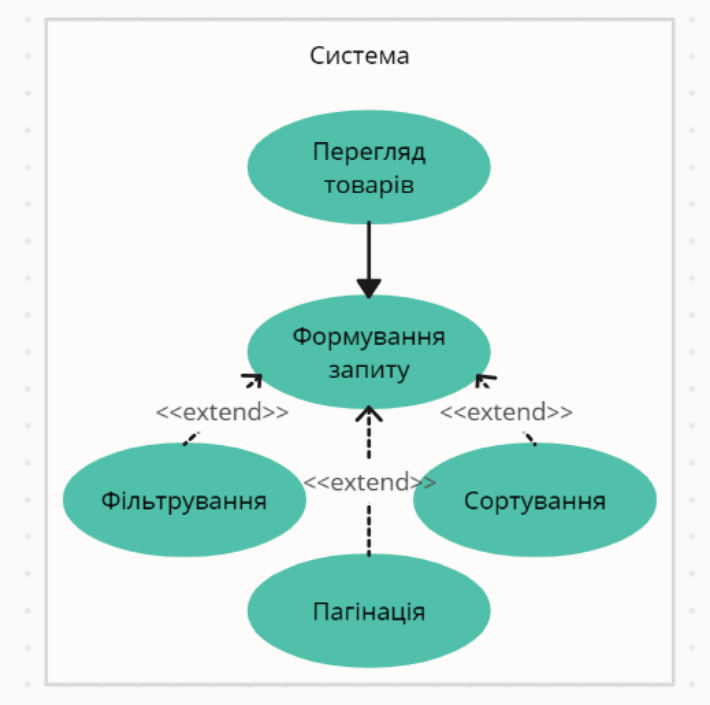


Рисунок 1.5 – Декомпозиція управління каталогом

Як видно з рисунку вище, користувачу мають надаватись функції пагінації, для розділення товару на частини, за для швидшого завантаження відображень, сортування за конкретними параметрами, а також, фільтрування. Фільтрування має відбуватись за категорією, ціною а також назвою. Обробка введених користувачем назв має відбуватись з допомогою алгоритму пошуку Левенштейна, за для підвищення користувацького досвіду, за рахунок ігнорування синтаксичних і орфографічних помилок.

При детальному перегляді інформації про товар, користувачу має надаватись можливість його обговорення з асистентом. Асистент має бути реалізований, як штучний інтелект, який прив’язаний до однієї з уже існуючих мовних моделей. Реалізація обговорення товару має відбуватись за допомогою використання Promt Engineering. Тематика розмови з асистентом має завжди стосуватись лише конкретних товарів. Також, штучний інтелект має знати

інформацію про стан товару в магазині. Для економного використання ресурсів застосунку, на користувача мають діяти обмеження на кількість повідомлень і їх розмір в контексті однієї розмови.

Для розподілу доступності функцій, має бути реалізована система ролей, їх перевірки та прив'язування до користувачів. Ролі мають відповідати статусу користувачів. Таким чином мають бути забезпечені наступні варіанти: адміністратор, користувач. Для гостя створення окремої ролі не обов'язкове, має лише перевірятись чи покупець має власний акаунт.

Для адміністрації сайту, мають бути забезпечені окремі сторінки, які надають функції управління товарами, такі як створення, редагування та видалення. При роботі з інформацією з товаром, має бути забезпечена функція встановлення галереї фотографій.

Адміністрації мають надаватись функції роботи з замовлення, такі як перегляд існуючих та їх редагування. Також в інструментах адміністратора мають надаватись функції перегляду статистики замовлень для визначення прибутку протягом певного проміжку часу.

Адміністратор також має мати можливість використання певних функцій користувачів, за для перегляду якості внесених даних у каталогах товарів. Відповідно, для швидшого пошуку, для панелі адміністрування передбачається використання фільтрування, сортування і системи пагінації, як і у випадку функцій користувача.

Отже, для успішної реалізації проєкту, потрібно реалізувати ряд функцій, які будуть розподілені по ролях користувачів, таких як: адміністратор, зареєстрований користувач та гість. Функції гостя обмежені лише переглядом товарів та авторизацію. Користувачі зможуть взаємодіяти з створенням замовлень, оцінкою товарів, віртуальним асистентом, пошуком по каталогам товарів. Адміністратори мають мати можливість опрацьовувати замовлення та працювати з даними існуючих товарів, або створювати нові. Окрім цього, вебресурс має дотримуватись ряду вимог, таких як безпека, висока швидкодія, адаптивність та інші.

1.4 Постановка завданнь на кваліфікаційну роботу бакалавра

Для успішної реалізації онлайн-магазину комп’ютерних товарів потрібно впровадити ряд функцій, розподілених за ролями користувачів: адміністратор, зареєстрований користувач та гість. Загальний функціонал має включати можливості створення замовлень, оцінки товарів, віртуальної асистенції, адміністрування та пошуку за каталогами. Таким чином завдання для кваліфікаційної роботи бакалавра можна поділити на три частини.

По-перше, потрібно спроєктувати функціональні частини застосунку, які направлені на покращення користувацького досвіду. Потрібно приділити велику увагу розробці пошукової системи та розробці асистента на основі штучного інтелекту. Наступною задачею буде проєктування архітектури застосунку, яка враховує усі функціональні вимоги.

По-друге, на основі розробленого проекту, потрібно реалізувати модулі застосунку. Компоненти мають відповідати спроектованим елементам та відповідати тематиці проекту.

Останнім етапом має бути проведене тестування розроблених компонентів застосунку, на основі яких буде описане документація програмного забезпечення. Тестування має охоплювати усі головні функціональні частини.

Досконале виконання кожного пункту дозволить реалізувати повноцінний інтернет-магазин комп’ютерних товарів, який міститиме сучасний функціонал, необхідний для покращення користувацького досвіду.

1.5 Висновки до розділу 1 і постановка задач для проєктування

Аналізуючи різні аспекти електронної комерції, можна зробити висновок, що створення онлайн-магазинів є найкращим вибором для середньостатистичних продавців, незважаючи на необхідність певних інвестицій. Такі платформи дають можливість створити конкурентоспроможний ресурс, якщо звернути увагу на

впровадження сучасних технологій, таких як, наприклад, штучний інтелект, і забезпечення належної безпеки даних користувачів. На додаток, ефективне управління онлайн-магазинами, з урахуванням потреб як користувачів, так і адміністрації, може значно підвищити рентабельність вкладень та популярність платформи за допомогою різних сучасних методів.

З іншого боку, ключові переваги, які цінують користувачі, включають широкий асортимент товарів, спеціалізацію на певних товарах і зручність онлайн-платформ. Використання штучного інтелекту, в даних цілях, як інструменту взаємодії з користувачами є додатковим бонусом, що підвищує задоволеність клієнтів.

Враховуючи усе сказане у даному розділі, можна скласти список завдань, які потрібно виконати під час проєктування системи.

По-перше, потрібно розробити методики для реалізації описаних у розділі функцій. Це має включати алгоритми та підходи, які будуть використані у майбутньому при реалізації проєкту. Потрібно проаналізувати їх ефективність та обдумати змінні, які будуть використані, як системні параметри для налаштування роботоспроможності.

По-друге, потрібно визначити потоки даних у системі, побудувавши діаграми для позначення оброки інформації та її результатів. Відповідно, потрібно визначити системні компоненти вебресурсу, поділивши його на первинні функціональні блоки.

Далі, на основі отриманих результатів, потрібно обрати архітектурні патерни, які будуть ефективними при виконані поставлених задач. Має бути описана їх взаємодія та розподіл функціоналу між модулями системи.

Для функціонування майбутнього вебресурсу, має бути розроблена інформаційна модель даних, яка буде використана при реалізації бази даних.

Також, під час проєктування, перед реалізацією, потрібно буде визначити початкові дані та звідки вони будуть отримані, що в подальшому вплине на якість тестування розробленого продукту.

2. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Визначення процесів у вебресурсі для продажу комп’ютерних товарів

При проєктуванні програмного продукту, потрібно визначити, які дані та процеси будуть використовуватись для забезпечення його функціональності. Як було зазначено під час дослідження предметної області, вебзастосунок має надавати функціонал для наступних груп користувачів: адміністратор, зареєстрований користувач та гість.

Загальна діаграма потоків даних проєкту для всіх користувачів зображена на рисунку нижче (Рисунок 2.1)

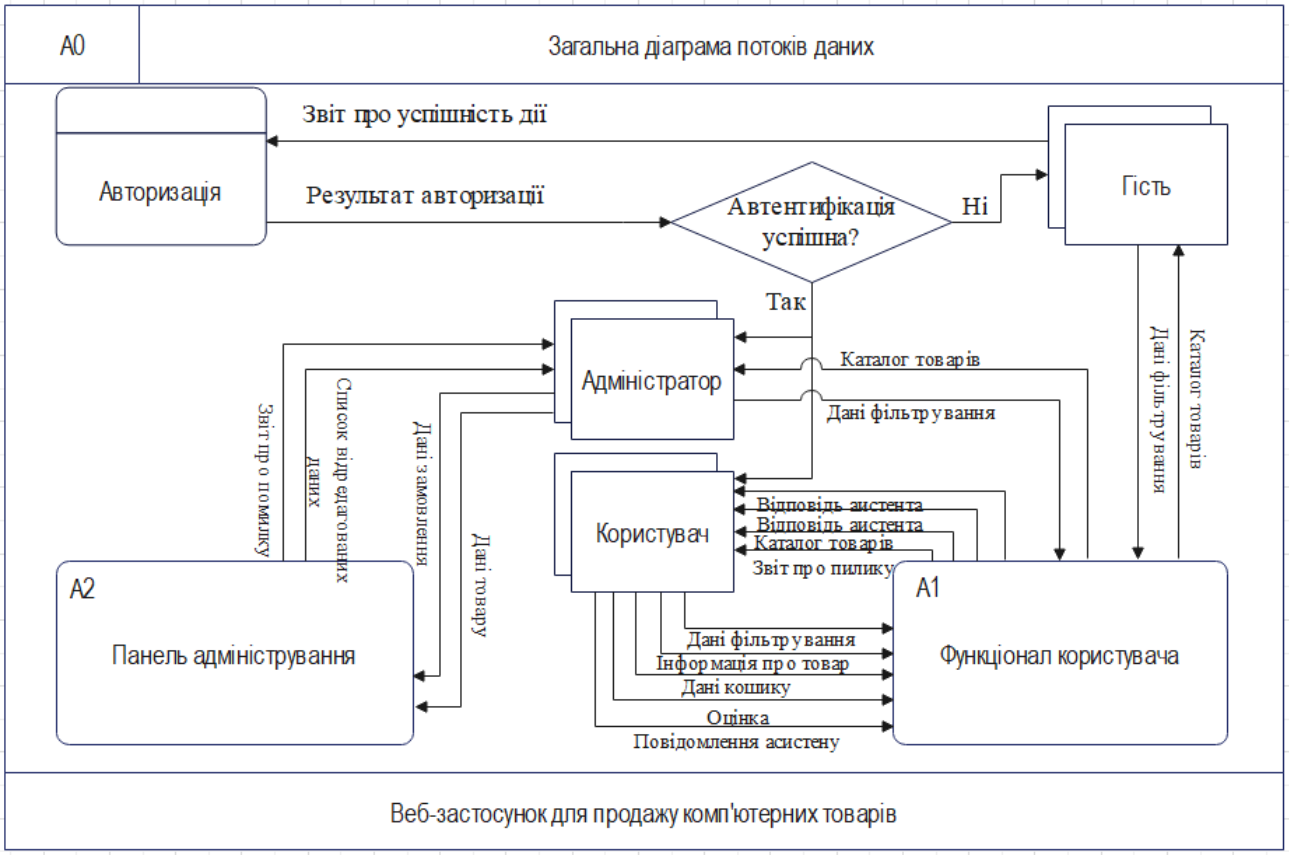


Рисунок 2.1 – Загальна діаграма потоків даних

Загальна діаграма складається з кількох декомпонованих елементів, які розділяють потоки даних на функціонал користувача і адміністратора. На

рисунку вище, можна побачити основні дані, які надаються усіма ролями користувачів, та яку інформацію вони отримують.

Увесь функціонал гостя, можна побачити на даній діаграмі. Гість, як особливий статус, користувача, має можливість виконати авторизацію, в незалежності від того, чи це процес авторизації чи реєстрації. В результатів, він може отримати доступ до прав зареєстрованих користувачів, або прав адміністратора. При цьому, якщо персональні данні ведені не вірно, до гостя повернеться інформацію про помилку авторизації.

Також, гості мають можливість переглядати товари. Тому при виконанні дій, які передбачаються перегляд інформації, які завжди включають передачу параметрів фільтрування, покупець отримуватиме список товарів для перегляду. Якщо ж параметри якимось чином не дійсні, в результаті їх модифікування не передбаченим способом, то повертатиметься пустий каталог.

Список процесів, які відбуваються в функціях зареєстрованих користувачів, можна побачити на наступному рисунку (рисунок 2.2).

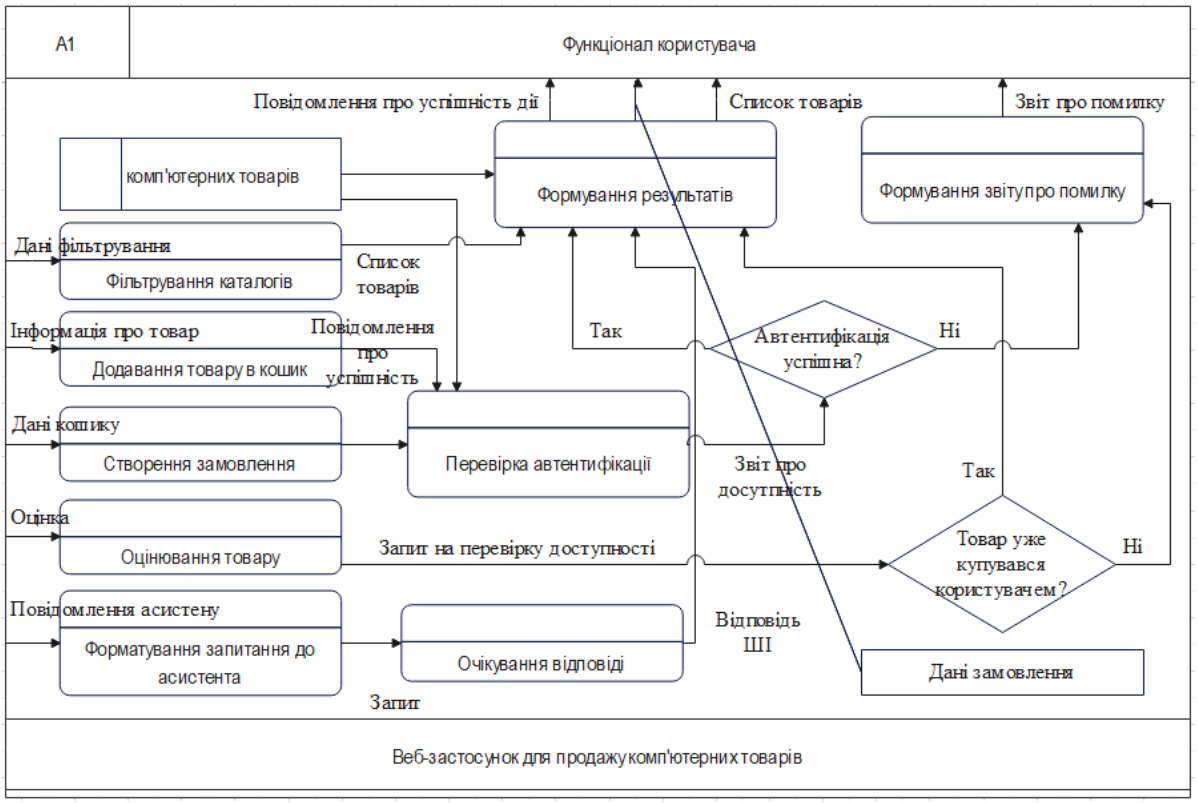


Рисунок 2.2 – Діаграма потоків даних в функціях корисутвача

Як і гість, користувач може переглядати товари. Як видно з рисунку, одразу після процесу фільтрування даних, для користувача формується список, який буде відправлений назад. При цьому обробка помилок не відбувається під час цього потоку даних, так як при формуванні некоректних запитів до БД, буде відправлятись пустий каталог.

Після закінчення процесів додавання товару в кошик і створення замовлення, завжди має проходити процес перевірки автентифікації. Якщо, наприклад гість спробує виконані ці дії, вони будуть відмінені, після чого сформується повідомлення про помилку, яка перенаправить користувача до вікна авторизації та реєстрації.

Як зазначалось в описі функціональних особливостей застосунок має мати можливість взаємодіяти з асистентом на основі штучного інтелекту. Тому, коли користувач передає власне повідомлення, після обробки воно буде відправлятись до мовної моделі, яка надаватиме відповідь. Відповідь має бути завжди. Якщо наприклад, сталася помилка при передачі інформації до ШІ, то відповідь має сформуватись у будь-якому випадку, але при цьому міститиме текст, який повідомить про несправності.

Оцінка товару може бути доступною лише після його покупки, тому після надання оцінки користувачем, цей параметр перевіряється у базі даних. При виконанні цих процесів, автентифікація може не грати ролі, тому що незареєстрований користувач в будь-якому разі не матиме персональних даних для перевірки наявності куплених товарів. У разі, якщо користувач не купляв позицію з каталогу, йому буде відправлятись звіт, який повідомлятиме про необхідність виконання деяких дій перед виставленням оцінки.

Якщо користувач отримує роль адміністратора, він може мати доступ до відповідних процесів, що зосередженні на взаємодією з продукцією та замовленнями. Система даних, які передаються для функціонування цього елементу, зображена на наступному рисунку (Рисунок 2.3).

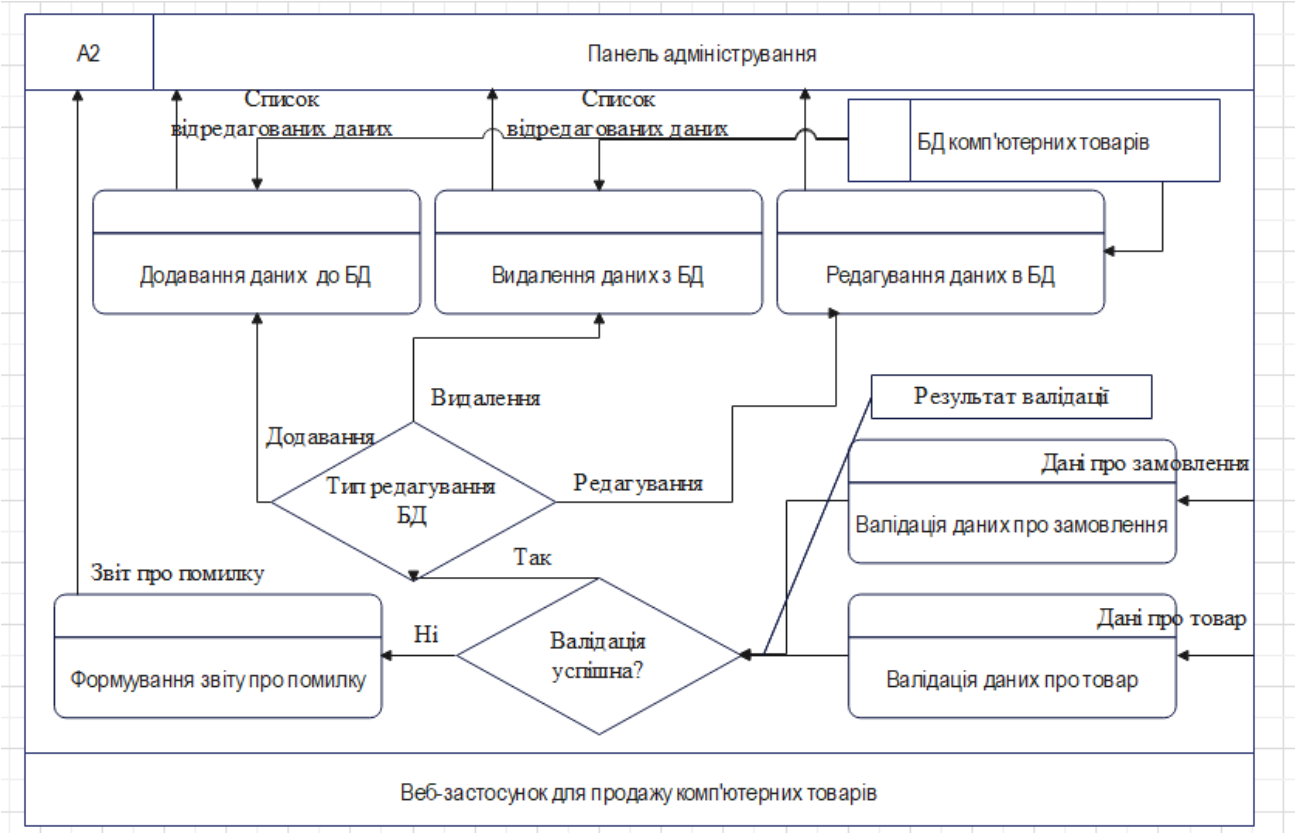


Рисунок 2.3 – Діаграма потоків даних в панелі адміністратора

Адміністратор, як і інші типи користувачів, може отримувати дані каталогу товарів. Але окрім цього, він може вносити зміни в уже існуючі елементи. При передачі даних про товари, перевіряється їх якість та відповідність параметрам БД. При помилках, адміністратору повертається інформація про те, де саме виникла помилка. Дані можуть набувати різного характеру в залежності від виду змін до бази даних, тому на їх основі можна роботи висновки, який процес має відбуватись. Якщо передаються дані без ідентифікатора – це створення, з ідентифікатором і зміненою структурою даних – редагування, а якщо є лише ідентифікатор – видалення.

Отже, створення майбутнього вебзастосунку для продажу комп’ютерних товарів, вимагало передбачення процесів, які відбуватимуться в системі. Це дозволило згрупувати функції, які мають бути реалізовані та визначено послідовність дій, які мають виконуватись, для отримання запланованих результатів та їх взаємозв’язки.

2.2 Проєктування функціоналу користувацького направлення

Для створення конкурентоздатного вебзастосунку, потрібно реалізовувати функціонал, який збільшить користувацький досвід, шляхом додавання інструментів, які можуть спростити життя користувачам. Для створення таких особливостей у даному вебресурсі, передбачається що буде використано кілька сучасних підходів для подібного роду вебресурсів.

Спочатку, має бути реалізована зручна пошукова система, на основі алгоритму визначення відстані Левенштайна між словами. Перевірка має відбуватись для кожного елементу з каталогу, а також знаходити відповідності навіть в частинах слова. Щоб перевірити, чи два слова можуть бути схожими при пошуку, буде порівнюватись результат алгоритму Левештейна з числом допустимих операцій з словом.

Даний алгоритм має бути реалізований за принципом динамічного програмування, який полягає у розбитті задачі на менші підзадачі та збереженні результатів для подальшого використання у вирішенні більш великої задачі. Основні кроки алгоритму які мають бути реалізовані:

– створення матриці, де рядки відповідають символам першого рядка, а стовпці – символам другого;

– ініціалізація першого рядка та першого стовпця матриці значеннями від 0 до довжин відповідних рядків;

– обчислення відстані Левенштейна для кожної пари символів патерну та рядка, з яким зрівнюється;

– заповнення усієї матриці відповідним чином;

В результаті, значення в матриці, що займає нижню праву позицію, позначатиме мінімальну кількість кроків, яку потрібно зробити для перетворення одного слова в інше. Іншими словами, результат показуватиме кількість неточностей, які допустив користувач. Порівняння похибок з певним визначеним параметром, можна зрозуміти, чи саме цей об’єкт підпадає під потреби.

Для того, щоб визначити, які неточності допускаються при введені запитів, потрібно визначити умовну формулу, яка від довжини слів буде вираховувати допустимі похибки. Це зумовлено тим, що чим більший розмір запиту, тим більше разів користувач може помилитись. Але при цьому, чим більше символів використовує користувач, тим більш впевненим у своїх запитах він може бути.

Базуючись на вище сказаному, функція залежності має мати вигляд кривої, швидкість якої з часом спадає, що притаманно логарифмам. Для збільшення швидкості росту функції на початку, можна піднести змінну до степеню, або використовувати коефіцієнти. Припускаючи, що користувач буде вводити в середньому до 20 слів, найкраще використовувати для помірного початкового зростання формулу:

$$y = \ln(x^2), \tag{2.1}$$

де y – кількість допустимих помилок, x – довжина введеного тексту.

Дана формула (2.1) може мати графічне представлення. При її використанні, можна намалювати графік швидкості росту функції, зображений на наступному рисунку (Рисунок 2.4)

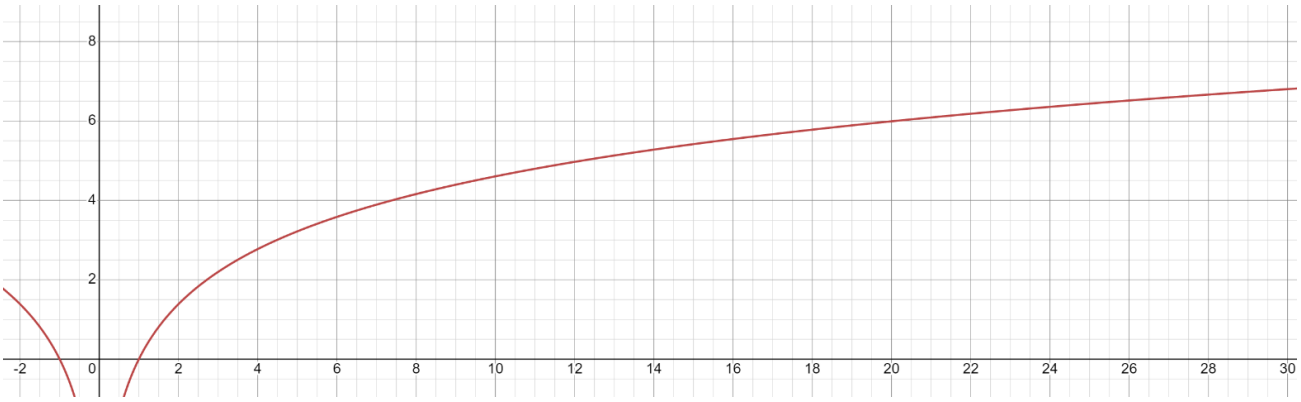


Рисунок 2.4 – Графік залежності похибки від кількості символів

З рисунка видно, що якщо користувач не введе жодного символу, то пошукова система не виведе результатів, тому що кількість похибок не може бути

від’ємною. Починаючи від двох до 10 символів, функція швидко зростає, дозволяючи швидко знаходити збіги в схожих словах. Якщо користувач буде вводити велику кількість тексту, то пошукова система буде розраховувати, що він зробить малу кількість помилок, відповідно до розміру.

На основі визначених функцій, було створено діаграму потоків даних для даного алгоритму (Додаток А). Діаграма складається з трьох вкладених циклів. Основний цикл потрібний для розбиття слова на частинки, які рівні розміру шаблону, для якого проводиться пошук. Два рівні вкладених циклів, потрібні для заповнення матриці. Для спрощення алгоритму, було використано систему циклів, замість рекурентної формули. Відповідно до цього, складність алгоритму за BigO-нотацією, можна визначити, як $O(n^2m)$. В цьому випадку, залежність швидкості виконання залежить від розміру шаблону, а також розміру рядка, в якому проходить пошук.

Інший важливий компонент застосунку – це можливість користуватись чатом для обговорення товару з штучний інтелектом. Найкращий спосіб реалізувати таку вза’ємодію можна за допомогою уже існуючих мовних моделей. Сучасні чат-боти надають доступ до власного API, що дозволяє створювати чати з власних пристроїв.

Щоб перетворити чат з моделлю, яка може обговорювати будь-яку тему і направити її у правильне русло, потрібно використовувати promt engineering. Більшість API, таких як OpenAI API, Nexus API і інші, використовують систему ролей у спілкуванні. Ролі діляться на «user», «system», «assistant». Коли надсилається запит, потрібно вказати роль того, хто це пише. «System» вказує асистенту налаштування і вказівки, до того, що має відбуватись і з ким ведеться розмова. «User» – це повідомлення відправлені користувачем. Відповідно, для асистента вказується своя роль.

Для забезпечення тематики розмови, при відправлені першого запиту в чат, потрібно вказати системні рекомендації. В даному випадку, буде відправлятись повідомлення: «Дана розмова відбувається з клієнтом інтернет-магазину.».

Також, до повідомлення мають прикріплюватись дані про товар, до яких мовна модель буде звертатись.

До повідомлень, які відправляє користувач, має прикріплюватись додатковий текст. У системах асистентів є деякі тригери, які дозволяють виділяти важливу інформацію. Наприклад, текст написаний у верхньому регістрі, буде помічатись, як важливий, при цьому якщо речення починається з слів «ЗВЕРНИ УВАГУ», то це буде надавати більший пріорітет додатковим умовам. Такий підхід спілкування часто використовується різними сучасними моделями штучного інтелекту.

Коли користувач відправлятиме повідомлення, до нього прикріплюватиметься текст, який повідомлятиме, що якщо тема не стосується конкретного пристрою потрібно замість відповіді, просто написати про це.

Враховуючи, що при відправлені системного повідомлення, надаються про статус товару в магазині, модель може надавати інформацію, яка пов’язана як з даними магазину, так і з даними, які відомі моделі. Тому, користувачі можуть запитувати модель про те, чи варто купляти даний прилад, чи його ціна краща, в порівнянні з ціною в інших магазин, порівнювати з іншими товарами подібного характеру та інше.

По тій причині, що спілкування з моделями вимагає певних грошових вкладень, то для одного сеансу спілкування буде надаватись обмеження в розмірі 2000 токенів, які може відправляти користувач. Токени в мовних моделях представляють собою найменші одиниці тексту, на які вони розбивають вхідний текст для подальшої обробки. Розбивка тексту на токени дозволяє моделі аналізувати текст на рівні окремих компонентів, що допомагає в здійсненні більш точних прогнозів та генерації тексту. Таким чином, 2000 токенів можна оцінити на теперішній курс у 0.011 доларів. Також, має бути встановлено обмеження на відправлення токенів за одне повідомлення. Найкраще встановити розмір в 150 токенів, що приблизно буде дорівнювати 75-ти словам.

Остання особливість застосунку, яка буде реалізована у застосунку – це транспортування даних у форматі BASE64 прямо у HTML-сторінки ресурсу.

Дане питання було висвітлено у праці Мантура В.А [30]. Принцип роботи алгоритму висвітлений нижче (Рисунок 2.5)

Рисунок 2.5 – Підзавантаження інформації у форматі Base64

Даний підхід дозволяє зберігати медіафайли у зручному бітовому форматі, який можна зберігати у реляційних базах даних та просто створювати резервні копії. Недоліком такого підходу є те, що при завантажені одразу великої кількості даних, вебресурс може довго завантажуватись, що призводить до погіршення користувацького досвіду. Дану проблему можна вирішити за допомогою послідовного підзавантаження фото, коли перша частина передається одразу для відображення головних компонентів, а інша частина уже після повного завантаження сторінки.

Отже, в даному вебзастосунку, для покращення користувацького досвіду, потрібно реалізувати окрім базових інструментів систему пошукової системи з неточним порівнянням рядків за допомогою алгоритму знаходження відстані Левенштейна, асистента на основі штучного інтелекту з використанням уже існуючих великих мовних моделей, а також реалізувати функцію передачі і зберігання даних у форматі BASE64.

2.3 Архітектура вебресурсу для продажу комп’ютерних товарів

Відповідно до функцій описаних у попередніх розділах, можна розробити систему компонентів, які будуть використані і у проєкті. Потрібно врахувати усі особливості системи та вид застосунку. Також, потрібно обрати шаблони проєктування, які будуть використані під час розробки інформаційної системи.

Спочатку потрібно визначити які зв’язки будуть між компонентами ПЗ, щоб розподілити функції на блоки. В даному проєкті пропонується використати схему зображену на наступному рисунку (Рисунок 2.6).

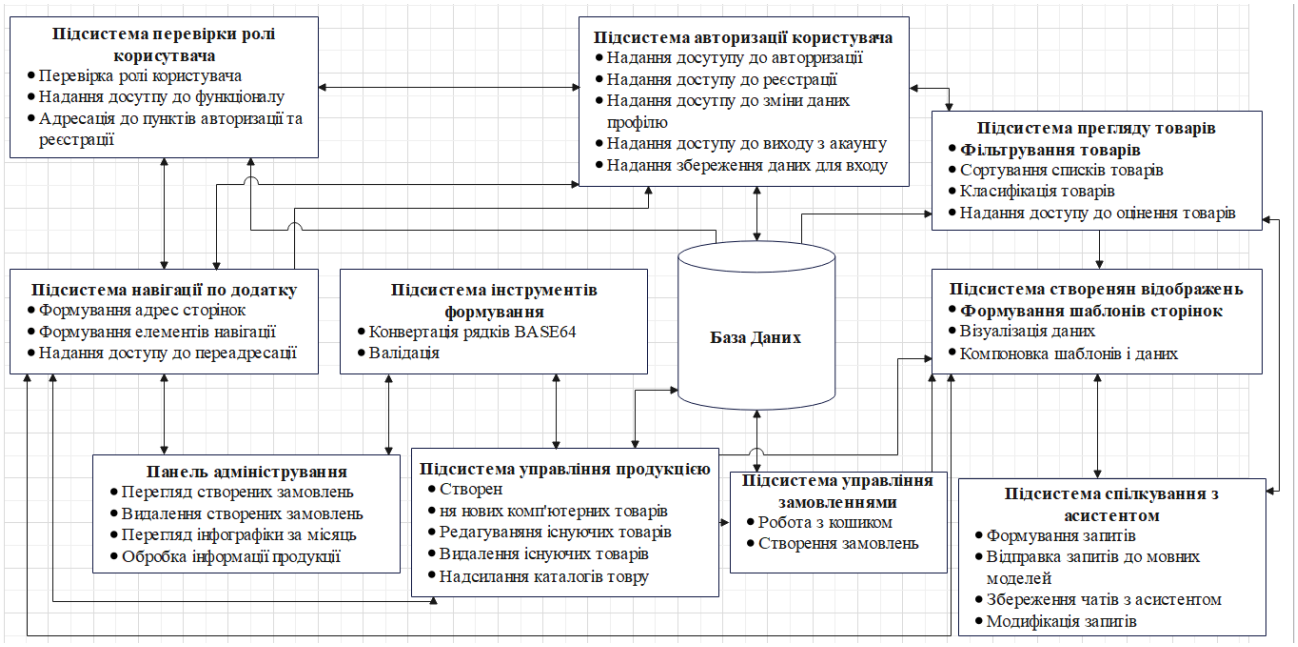


Рисунок 2.6 – Схема вебресурсу для продажу комп’ютерних товарів

На діаграмі представлено розгалужену структуру системи, яка включає кілька основних блоків, що взаємодіють між собою через централізовану базу даних. Система організована навколо ключових процесів, таких як перевірка ролей користувачів, надання доступу до різних функціональних можливостей, як-от авторизація, реєстрація та управління профілями. Також існує система перегляду товарів, яка дозволяє фільтрувати, сортувати, класифікувати товари та

надавати доступ до оцінки товарів. Інша частина діаграми відноситься до системи створення зображень, включаючи формування малюнків, візуалізацію планів і компонування малюнків і даних. Додаткові елементи включають панель адміністрування для вирішення замовлень та обробки інформації продукції, а також системи управління продукцією з можливостями зв'язку з асистентом.

Основною підсистемою для вебресурсу для продажу комп’ютерних товарів – це управління продукцією. Вона, як видно по назві, відповідає за дії, які потрібні при доступі до даних товарів. З даною системою може вза’ємодіяти панель адміністрування, з якої викликаються функції взаємодії з елемнтами БД.

Для доступу до кожної з сторінок, використовується підсистема навігації, яка визначає шляхи до кожного компоненту. Для того, щоб обмежити користувачів по ролях, використовується компонент перевірки ролей, яка визначає правила для користувачів.

Підсистема перегляду товарів надає засоби для фільтрування, сортування, розподілення їх по класах і оцінювання товарів. Функціонал системи також проходить процеси валідації.

Більшість підсистем, після обробки даних, звертаються до блоку створення відображень, яка функціонує як середовище для візуалізації контенту. Дана система передбачає створення компонованих елементів інтерфейсу, які є спільними для всіх сторінок. Також дана система створює деякі динамічні компоненти, які доступні користувачам.

Спираючись на визначені компоненти, для реалізації вебресурсу по продажу комп’ютерних товарів добре підходить архітектурний шаблон MVC. Даний патерн, призначений для розробки програмного забезпечення, який розділяє дані програми, інтерфейс користувача та логіку управління на три взаємопов’язані компоненти. Це розділення допомагає управляти складними програмами, спрощуючи модифікацію і масштабування застосунка, оскільки розробники можуть працювати незалежно над різними аспектами програми.

Для даного вебзастосунку, MVC є відмінним вибором. У цьому випадку, модель управляє всіма даними, логікою та правилами застосунка, що охоплює такі аспекти, як дані користувачів, інформація про товари та замовлення.

Для таких підсистем, як перевірка ролей користувачів і інструменти формування, можна виділити додаткові шари логіки, які будуть взаємодіяти з контролерами, але при цьому не будуть їх частино. Як уже зазначалось, це дозволить створити більш масштабований проєкт, функціонал якого можна постійно розширювати.

Що головне, при використані даного патерну, можна оперувати великою кількістю інформації, що важливо у онлайн-магазинах. Виносячи фільтрування і сортування на рівень контролер, користувачу буде надходити лише інформація, яка потрібна для перегляду в конкретний момент часу. Також, це дозволить використовувати широкий спектр продукції, що також є одним з основних елементів подібного роду застосунків.

Дана архітектура передбачає розділення функціоналу на декілька частин, таким чино розподіляючи обов’язки на два елементи – клієнт і сервер. Сервером, в даному випадку, буде віддалений пристрій, на якому зберігатиметься інформація з бази даних. Окрім цього велика частина функціоналу виконуватиметься саме у цьому компоненті.

Формування клієнту, дозволить відправляти частину функціоналу разом з відображенням. Процеси будуть оброблятись браузером, при цьому застосування деяких дій оброблятиметься саме на пристрої користувача. Це дозволить зменшити навантаження на сервер, при цьому надавши можливості динамічно змінювати деякі елементи інтерфейсу.

Дана архітектура добре взаємодіє з патерном MVC. Модель і контролери, а також деякі шари логіки, можна помістити на сервер, таким чином створивши безпечне середовище для функціонування елементів, так як користувачі не матимуть доступ до внутрішніх процесів. Відображення – це частина клієнтської частини, яка міститиме окрім елементів інтерфейсу код, який не буде впливати на роботу бази даних. Таким чином, система спілкування з асистентом також

може розміщуватись, як цілісний інструмент в середині відображень, так як для його роботи потрібно лише надати дані серверу, але отримані відповіді ніяким чином не впливають на процеси, що відбуваються у вебресурсі, а потрібні лише для інформування самого користувача.

Розподіляючи структуру вебресурсу таким чином, можна розділити і елементи раніше створеної схеми компонентів. До серверної частини відносяться підсистеми бази даних, перевірки ролі користувача, панелі адміністратора, інструментів формування, авторизації користувача, перегляду товарів і управління замовленнями. При цьому, підсистеми інструментів формування та перегляду товарів і перевірки авторизації можна віднести за межі патерну MVC, перетворивши їх на функціональні класи, які надаватимуть список можливостей для використання в контролерах. До контролерів будуть віднесені усі інші названі системи. База даних буде представлятись у вигляді моделей.

Клієнтська частина буде складатись з підсистеми навігації, створення відображень та спілкування з асистентом. При цьому спілкування з асистентом буде прив’язано до відображень, як частина загального функціоналу, розширяються їх.

На основі поєднання патерну, архітектури та компонентів, можна скласти фінальну діаграму, яка відображатиме розподіл компонентів (Рисунок 2.7).

Рисунок 2.7 – Загальна схема компоновки елементів архітектури

Як видно з рисунку, схема стала простішою, а компоненти стали поділені на підгрупи, які в свою чергу поділені на менші системні компоненти – клієнта та сервера. Даний підхід дозволив централізувати використання функцій, що дозволяє робити систему стійкою до змін.

Інкапсульована логіка по функціональних блоках дозволяє вносити зміни у структуру логіки, без зміни функціоналу інших компонентів. Також, це дозволяє в майбутньому розширювати функціонал, без необхідності вносити правки в уже існуючі компоненти. Навіть при видалені елементів, буде прибиратись їх функціонал, але при цьому більша частина модулів продовжать функціонувати без змін.

Варто зазначити, що для взаємодії з даним архітектурним підходом, зв’язок клієнта і сервера забезпечується за рахунок API, які визначає до якого контролеру звертається користувач. Для ефективної роботи даного процесу, варто використовувати принцип RESTful API, який передбачає структуровану систему запитів. Таким чином, будь-яке посилання матиме чіткі дані для обробки сервером. Це дозволить використовувати як і динамічні запити, зі сторони клієнта, так і навігацію через посилання.

Отже, для реалізації вебресурсу для продажу комп’ютерних товарів, потрібно створити десять основних функціональні блоки, які розподілятимуть між собою задачі описані у попередніх розділах. Для більш чіткого і структурованого розподілу задач, буде використано архітектурний шаблон MVC з додатковими логічними шарами.

Для створення інформаційної системи, яка надаватиме віддалений доступ до ресурсів, буде застосовано клієнт-серверна архітектура з використанням RESTful API. Це поділить застосунок на ще менші компоненти, розділяючи обов’язки програмного продукту як на пристрої користувачів, так і на сервер, який забезпечить можливість підключення багатьох людей та одночасне використання ними вебзастосунку.

2.4 Опис структури даних та моделі бази даних

Майже не можливо створити сучасний вебресурс без використання баз даних, особливо якщо це стосується комерційної діяльності. Саме тому їх проєктування є важливим етапом у розробці програмного продукту. На основі функцій та даних, з якими буде працювати інформаційна система, можна сформувати список усіх сутностей, які мають зберігатись в спроєктованому централізованому сховищі.

На рисунку 2.8 зображена схема ERD для вебресурсу, який спеціалізується на продажі комп'ютерних товарів. Перш за все потрібно звернути увагу на елементи каталогів, які продаватимуться на ресурсі. Даний об'єкт можна назвати, як «Продукт». Кожний продукт може мати декілька фотографій, які показують його з різних сторін. Також, дана сутність тісно пов'язана з сутністю категорій, так як багато продуктів можуть мати одну і ту ж категорію, що буде використано для процесів фільтрування.

Користувачі, це інший вид сутності, який має вплив на всі ніші. По-перше, даний елемент може взаємодіяти з кошиком, поміщаючи туди елементи продукції. Дані з кошика можуть відправлятися до сутності замовлень, користуючись інформацією користувача. По-друге, кожний користувач може надати оцінку товарам. Для того, щоб визначати, чи товар був придбаний і чи не створювалась оцінка до того, дані цих сутностей нерозривно пов'язані.

Як зазначалось раніше, один користувач може мати багато адрес, тому для забезпечення цього функціоналу, було виділено окрему сутність. Також, користувач може мати багато замовлень, які міститимуть існуючі товари. Але товари можуть замовлятись на інших осіб, окрім власника акаунту, тому деякі властивості повторюють властивості користувача.

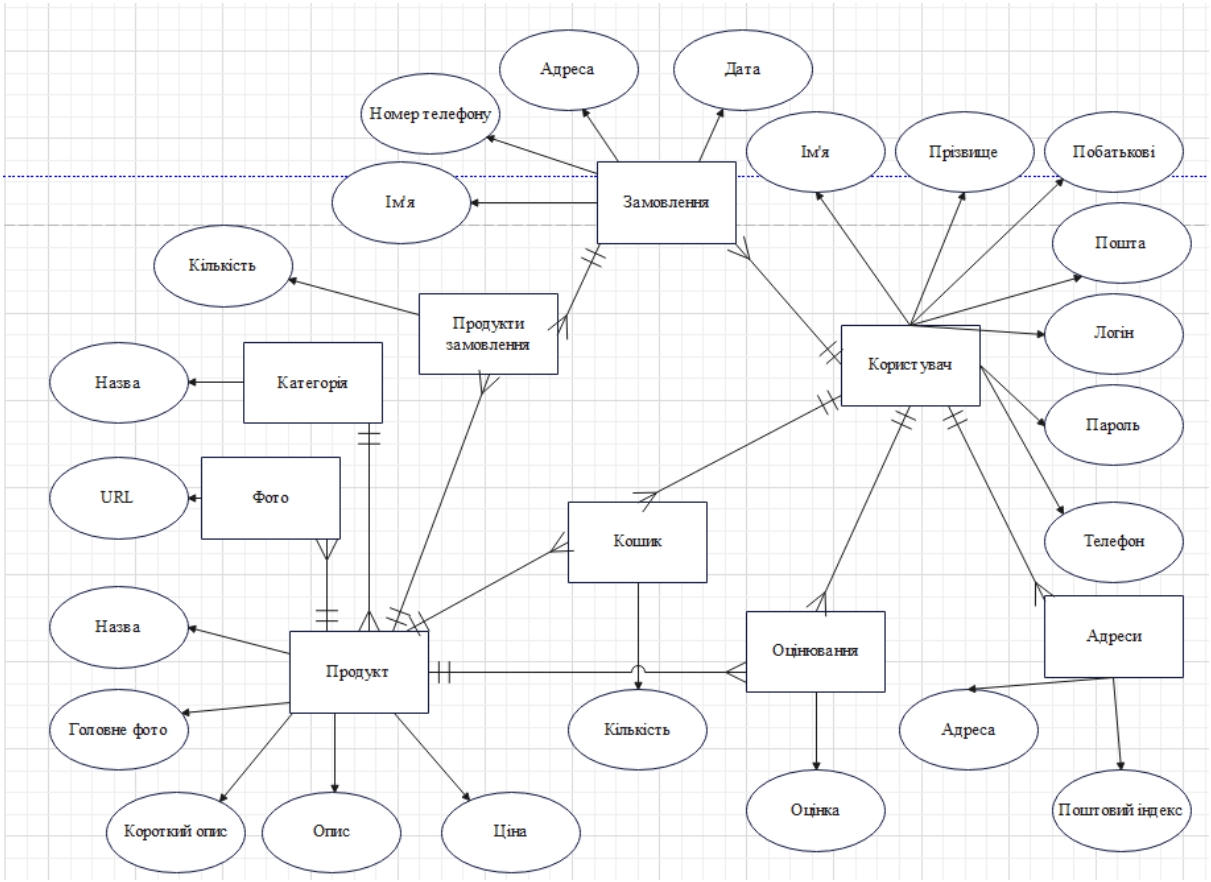


Рисунок 2.8 – ERD представлення даних в вебресурсі

Для того, щоб підготувати дані до роботи з різними СКБД, потрібно дотриматись кількох умов. Спроєктована структура має бути спільною для реалізації з використанням різних реляційних структур.

По-перше, потрібно визначити усі назви атрибутів сутностей латиницею, для більшої сумісності з різними програмними елементами.

По-друге, потрібно визначити властивості сутностей, які будуть створювати взаємозв'язки між таблицями. Але варто зауважити, що зв'язки мають забезпечувати процеси нормалізації, мінімум до третьої нормальної форми, таким чином забезпечивши стійкість системи.

На рисунку 2.9 зображено даталогічну модель бази даних вебзастосунку для продажу комп'ютерних товарів. Вона включає в себе: ProductImage, Cart, Product, User, UserAddress, Order, OrderProduct, Category, Rating, UserRole, Role.

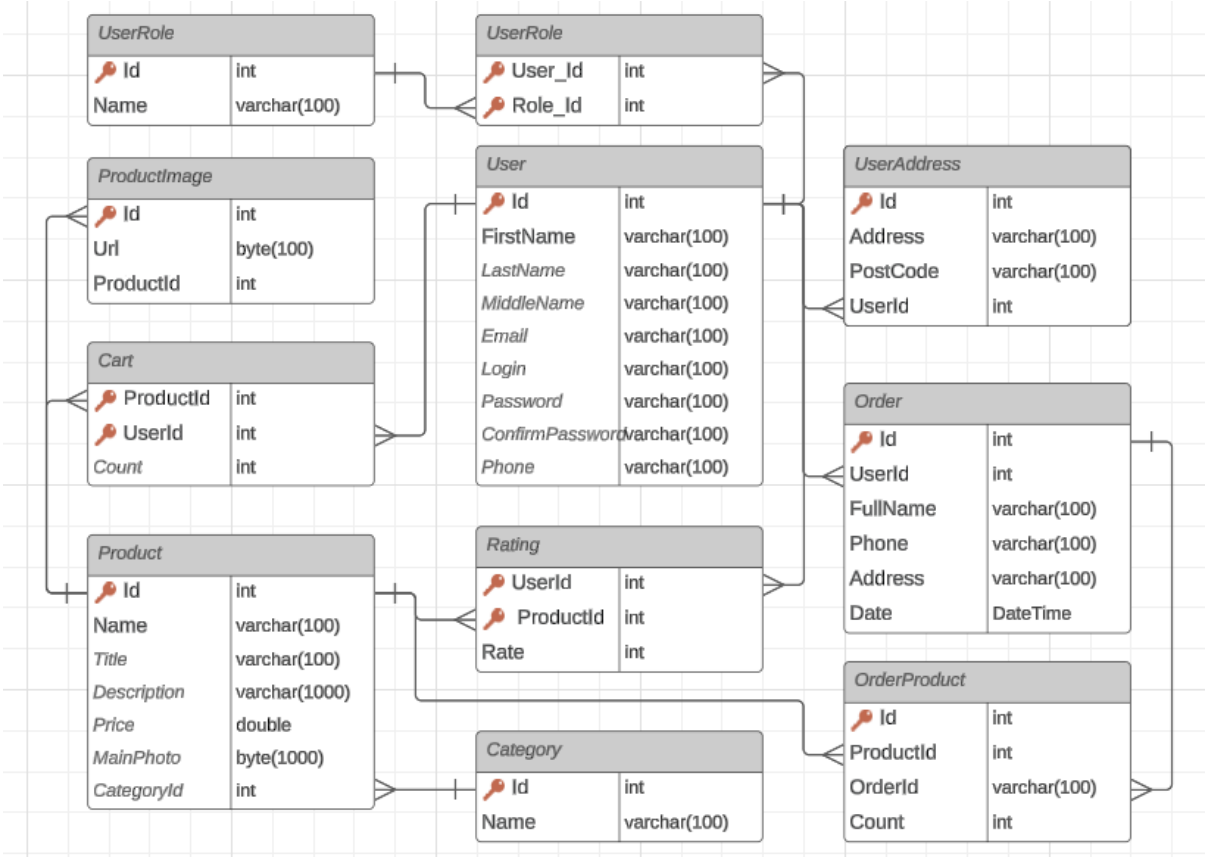


Рисунок 2.9 – Даталогічна модель бази даних для вебресурсу

Таблиця Product призначена для зберігання інформації про товари. Вона містить атрибути: Id, Name, Title, Description, Price, CategoryId, MainPhoto. Повний опис атрибутів зображено в додатках (Додаток Б.1)

Продукти мають зв’яок оди до багатьох з сутністю PhotoImage, яка відповідає списку усіх фотографій продукції. Атрибути даної таблиці включають: Id та Name (Додаток Б.2).

Також один продукт може відноситись до однієї категорії товарів. Атрибути даної таблиці включають: Id та Name (Додаток Б.3).

Користувач, у вигляді сутності User має атрибути: Id, FirstName, LastName, MiddleName, Email, Login, Password, ConfirmPassword, Phone (Додаток Б.4). Дана таблиця має зв’язки багато до одного з багатьма іншими таблицями, але при цьому дана сутність не залежна від інших.

Користувачі з продукцією мають зв’язок один до багатьох з сутністю «Cart», яка представляє з себе кошик. Атрибути даної сутності: ProductId, UserId,

Count (Додаток Б.5). дана таблиця не має власного ідентифікатора, натомість він складається з 2 зовнішніх ключів. Це дозволяє перевіряти чи користувач уже поклав даний продукт у кошик.

Для сутності Raiting, для оцінювання товарів, використовується така ж схема для створення комбінованого первинного ключа з зовнішніх. Атрибути даної таблиці: UserId, ProductId, Count (Додаток Б.6).

Для створення ролей, які є в користувачів, виділена діаграма «Role», яка складається з атрибутів Id, Name (Додаток Б.7).

Один користувач може мати багато ролей, але при цьому багато ролей можуть відноситись до одного користувача. Це створює зв'язок «багато до багатьох». Для того, щоб його розірвати, створена сутність UserRole яка складається лише з UserId та RoleId (Додаток Б.8). Дана таблиця дозволить забезпечити зв'язок «один до багатьох», що потрібно для нормалізації.

Один користувач може мати багато адрес. Тому окрема сутність UserAddres має атрибути Id, Address, PostCode, UserId (Додаток Б.9).

Користувачі також можуть робити замовлення, які записуватимуться у сутність Order. Дана таблиця має атрибути: Id, UserId, FullName, Phone, Address, Date (Додаток Б.10). Дана таблиця не містить інформацію про продукцію, але надає власний ідентифікатор для створення списків.

Щоб містити інформацію про продукти в замовлені, створена наступна таблиця «OrderProducts» з такими визначеними атрибутами, як Id, ProductId, OrderId, Count (Додаток Б.11).

Для забезпечення третьої форми нормалізації, потрібно щоб між всіма таблицями був зв'язок «один до багатьох». Враховуючи структуру описаних таблиць, можна сказати, що вони уже відповідають даному стандартну і не потребують покращень.

Отже, було спроєктовано модель для бази даних, яка може працювати з різними СКБД. Для вебресурсу для продажу комп'ютерних товарів, необхідно розробити одинадцять сутностей, та реалізувати їх у вигляді моделей. Також, під

час реалізації важливо дотриматись описаних зв’язків, які дозволять позбутись аномалій, які можуть виникати під час оперування даними.

2.5 Проєктування модулів інформаційної системи

Визначивши основні функціональні блоки, архітектуру та дані, якими буде оперуватись у інформаційній системі, потрібно визначити, які програмні модулі будуть знаходить у проєкті.

При використані архітектурного патерну MVC, розробку зазвичай починають з моделей, тому вони мають бути спроєктовані в першу чергу. Враховуючи, що моделі мають точно описувати структуру бази даних, то було спроєктовано діаграму класів, що демонструє копію даталогічної моделі, яка зображена у додатку В.1.

Для забезпечення централізованого оперування усіма даними і доступу до них, має бути реалізований клас «GameStoreDBContext». Він міститиме в собі колекції з типами даних кожного класу моделей. Також, для того щоб наповнювати колекції даними з бази даних, потрібно надати метод підключення, через який створюватиметься зв’язок.

Наступним кроком є проєктування модулів контролерів. Контролери забезпечать обробку даних моделей та створюють відображення для користувачів. Відповідно до визначених підсистем у попередніх розділах, потрібно реалізувати наступні контролери: ProductsController; AccountController, ShopController, CartController, AdminController, OrderController, HomeContrroller.

Таким чином, зв’язок між контролерами буде виглядати, як зображено на рисунку нижче (додаток В.2)

Частина методів, які зображена на рисунку, будуть використовуватись для відправки відображень, або надавання доступу для виконання дій без змінити відображень. Для точності, усі методи які пишуться з великої літери означають методи, до яких користувачі можуть виконувати запити, використовують

RESTful-підхід. Тому, якщо користувач спробує відправити запит до Profile з контролера «Account», він отримає сторінку перегляду профіля, в той час як звертання до CreateAdress, лише виконає операцію про видаленні адреси користувача і поверне відповідь про те, чи дія виконана успішна.

Методи, які написані з малої літери, виконують допоміжні функції. Наприклад, getUserCart з контролера замовлень, надаватиме список усієї продукції в кошику користувача, перед створенням нового замовлення.

Функції управління моделями виконуються через, уже раніше названий, контекст даних. Майже всі контролери при виконанні операцій або отримують доступ до БД для перегляду, або внесення змін. Дана централізація дозволяє робити проєкт більш розширюваним і стійким до змін.

При роботі з моделями, важливо дотримуватись принципів валідації. Наприклад, коли користувач буде вводити кількість продукції, яку він хоче придбати, його потрібно обмежити в кількості від 0 до певного числа. Для цього потрібно реалізувати класи, які оброблятимуть вхідні дані. На зображені нижче можна побачити приклад валідації кількості елементів (додаток В.3).

Для того, щоб дотримуватись принципів ООП, і правила DRY, була створена ієрархія класів, де батьківський елемент приймає і опрацьовує інформацію про дані, а дочірні елементи містять правила обробки.

Для того, щоб звільнити контролери від зайвої логіки обробки даних про користувачів та створення системи запитів, на окремий шар логіки, було винесено функції спілкування з акаунтами користувачів, які швидко обмінюються даними між компонентами (Додаток В.4).

CustomMembership і CustomPrincipial забезпечують функції управління авторизацією, збереження інформації про користувача на його комп’ютер, а також перевіркою ролей. Інші фрагменти класів, потрібні для серіалізації даних та відправки повідомлень з серверу до клієнта і навпаки, а також для створення копій серіалізованих користувачів для збереження в файлах cookie браузерів.

Для відокремлення від контролерів процесів фільтрування та конвертації рядків у формати BASE64, були створені відповідні класи (Додаток В.5).

Клас ProductFilter забезпечує функціонал для пошуку на основі алгоритму Левенштейна, фільтрування по категоріях, а також сортування, яке використовує клас перечислень – OrderTypes. Для роботи з рядками BASE64 і їх формуванням для перегляду у відображеннях, використовується JSImageByteConvertor, який забезпечує зберігання даних медіафайлів у форматі бітів, а при створені відображень, конвертує їх у рядки з зашифрованими даними.

Для забезпечення роботи чат-асистенту, для нього виділиться окремий модуль у відображені, який буде реалізований в залежності від технологій розробки. Його функціонал полягатиме в записі повідомлень користувачів, відправки запитів до API і відображені результатів. Також, саме в цьому модулі будуть записані обмеження для спілкування користувачів з чат-ботом.

Відображення не передбачаються, як окремі модулі і мають бути прив’язані до роботи контролерів. Але потрібно звернути увагу, що в застосунку має бути сформована система компонування різних відображень, для того, щоб забезпечити спільні елементи для різних сторінок.

Повну діаграму класів застосунку, в розкритому вигляді продемонстровано в додатку В.6, де можна побачити і всю структуру зв’язків між модулями.

Отже, для успішної реалізації програмного продукту, потрібно дотриматись описаної у даному розділі структури класів. Потрібно розділити класи за функціоналом та розмістити у відповідних каталогах, які передбачатимуть чітке розуміння їх призначення.

2.6 Аналіз та вибір технологій і методів реалізації

Враховуючи описані компоненти у застосунку, можна сформувати чітке бачення технологій, які будуть використані для реалізації поставлених задач. Вибір складається з мов програмування, бібліотек функцій та компонентів.

Почати варто з реалізації клієнт-серверної архітектури з патерном MVC. Є декілька мов програмування, фреймворки яких підтримують реалізацію даного

підходу. Перша – PHP з бібліотекою Laravel, яка дозволяє створити серверне представлення з усіма компонентами [31]. Laravel – це високорівневий фреймворк, який пропонує зручні інструменти для швидкої і ефективної розробки, включаючи систему маршрутизації, вбудовану аутентифікацію, міграції баз даних та багато іншого [32]. Але мова PHP в даний час все менше користується популярністю, через що в майбутньому важче буде найти людей, які будуть готові підтримувати проєкт на даній мові.

Інший варіант – використовувати популярний фреймворк ASP.NET MVC, який працює на мові C# [33]. Дана технологія постійно підтримується і оновлюється компанією Microsoft, через що він користується високою популярністю [34].

Технологія ASP.NET MVC через свою популярність має велику кількість бібліотек, які дозволяють реалізовувати різні компоненти. Також, велика частина технологій одразу використовується разом з проєктами на його основі. Наприклад, система Razor, дозволяє писати код на C# при формування відображень, що дозволяє формувати перегляд інформації динамічно, ще до того, як воно потрапить до користувача [35]. Це відкриває доступ до компонування елементів інтерфейсу, що важливо для реалізації даного вебресурсу.

Так як в застосунку передбачається система пагінації, для неї можна використати клієнтську бібліотеку AspNetCore.Pagination [36], яка створить класи, для швидкого встановлення елементів управління сторінками. Також, дана утиліта дозволяє розбивати дані з БД на частини, для швидшого оперування.

Для створення бази даних, можна використовувати СКБД MS SQL Server [37], яка також розробляється компанією Microsoft і рекомендується для використання з «.NET» технологіями. Для даної СКБД пропонуються також зручні сервера Azure, які мають функціонал для відслідковування трафіка, швидкого створення резервних копій і міграцій.

Для тестування уже обраних компонентів, обрано бібліотеку NUnit, яка надає можливості, як для інтеграційного, так і юніт-тестування.

Разом з MS SQL Server і ASP.NET, зручно використовувати фреймворк Entity Framework, який забезпечує функціонування принципу «Code First» [38]. Даний принцип, реалізує можливість створити моделі, без створення бази даних, після чого на основі класів, фреймворк створить структуру таблиць SQL Server. Також, дана технологія дозволяє використовувати ще одну особливість «.NET», а саме мову запитів LINQ, що спрощує звернення до БД [39]. Використання цього фреймворку також створює віртуальну копію бази даних, що збільшує навантаження саме на оперативну пам’ять, але при цьому прискорює виконання процесів з усією інформацією.

Для клієнтської частини найкращим вибором є використанням мови JavaScript. Вона підтримується усіма сучасними браузерами, а також має добру підтримку версій та функціоналу [40]. Відповідно, дана мова використовується у клієнтській частині з HTML 5 та CSS 3.

Так, як є все ж існує проблема з підтримкою версій деякими старими браузерами, потрібно використати babel [41], бібліотеку яка покращує підтримку коду, шляхом формування поліфілів. Поліфіл – це шматок коду, який надає можливість використовувати функціональність, яка може бути відсутня в старіших версіях браузерів, забезпечуючи сумісність з новими стандартами або функціоналом мов програмування [41].

Для підтримки babel, потрібно встановити у проєкт webpack – систему збірки проєктів з використанням різних бібліотек. Дана технологія користується популярністю і часто використовується навіть в великих проєктах.

Передбачається, що система матиме динамічні компоненти, як наприклад, спілкування з чат-ботом. Для цього найкраще використати фреймворк React.JS, який надаватиме можливість, вводити JS-код прямо у сторінки вебсайту. Дана технологія займає найбільшу частку на ринку, порівняно з подібними [42]. Також, спілкування з чат-ботом потребує використання постійного надсилання запитів від клієнта до серверів мовних моделей. Для спрощення цього процесу, буде використано частину бібліотеки JQuery, а саме технологію передачі Ajax, яка не

лише дозволяє структурувати асинхронний код, а й може взаємодіяти з модулями ASP.NET, надаючи можливість опрацьовувати помилки запитів [43].

Також, для спрощення створення деяких компонентів, буде використана бібліотека стилів Bootstrap. Для реалізації системи оцінювання буде використана бібліотека rate-react.js, що надає можливість не лише створити візуальну частину, а й надає список подій для реалізації запитів при діях користувачів. Те саме і стосується компоненту chat-react.js, який дозволить створити візуально-приємний інтерфейс для спілкування з чат-ботом.

Отже, враховуючи описані компоненти у застосунку, можна сформувати список технологій для реалізації поставлених задач. Для реалізації серверної частини буде використаний ASP.NET MVC на базі C#. Для створення моделей буде використано Entity Framework для зручної роботи з базами даних за принципом "Code First". MS SQL Server буде відповідним вибором для БД. Для клієнтської частини основним вибором стане JavaScript разом з HTML5 і CSS3. Допоміжні бібліотеки як babel і webpack забезпечать оптимізацію та сумісність коду. React.JS дозволить вводити JS-код прямо у сторінки, що є ідеальним для реалізації динамічних компонентів. Для запитів буде використана технологія Ajax. Компоненти стилізації та інтерактивності, такі як Bootstrap, chat-react.js і rate-react.js, допоможуть створити зручний інтерфейс.

2.7 Висновки до розділу 2 і постановка задач для реалізації

Як висновок, можна сказати що, створення майбутнього вебзастосунку для продажу комп’ютерних товарів включало передбачення процесів, які відбуватимуться в системі. Це дозволило згрупувати та реалізувати функції, спрямовані на отримання запланованих результатів. Для покращення користувацького досвіду, заплановано інтегрувати систему пошуку з алгоритмом Левенштейна для неточного порівняння рядків, асистента на базі штучного інтелекту та функції передачі та збереження даних у форматі BASE64.

Буде створено десять основних функціональних блоків, які використовують архітектурний шаблон MVC з додатковими логічними шарами. Застосування клієнт-серверної архітектури з використанням RESTful API забезпечить віддалений доступ до ресурсів, що покращить модульність та зручність використання вебзастосунку.

Також, було спроєктовано модель бази даних, що може працювати з різними системами керування базами даних, та розроблено одинадцять сутностей для уникнення аномалій при операціях з даними.

На основі визначених особливостей, було визначено необхідні інструменти для реалізації. На серверній стороні буде використано ASP.NET MVC на базі C#. В якості основи для клієнтської частини обрано JavaScript з HTML5 і CSS3. Бібліотеки babel і webpack забезпечать оптимізацію коду та його сумісність. React.JS разом із Ajax дозволить реалізувати динамічні компоненти та асинхронні запити. Інтерфейс буде оживлено за допомогою Bootstrap, chat-react.js та rate-react.js, створюючи залучаючий і функціонально багатий користувацький досвід.

Під час реалізації, потрібно буде дотриматись усіх визначених пунктів у даному розділі. Кожний компонент має бути реалізовано відповідно його опису і представленому графічному матеріалу.

Окрім цього, під час реалізації, потрібно буде визначити елементи інтерфейсу, який відповідатиме спроєктованим модулям. Інтерфейс має використовувати визначені і розділі технології та бути зручним у використанні.

Далі, потрібно буде провести тестування модулів на виявлення помилок, після чого виправити їх. Дану процедуру потрібно буде проводити до того моменту, доки можливість похибок у робі системи не буде мінімальним.

Кінцеві результати тестування мають бути записані та обгрунтовані. При цьому мають проводитись різні види перевірки стійкості системи до умов, які можуть виникнути під час користування.

Фінальним етапом розробки має бути створення документації по керівництву користувача, де будуть описані особливості використання застосунку. Мають бути і технічні вимоги до пристроїв для взаємодії з системою.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Програмна реалізація модулів системи

3.1.1 Програмна реалізація серверних модулів застосунку

Порядок реалізації компонентів MVC має стійку послідовність. Найпершим етапом виступає створення опису бази даних, за допомогою моделей. Для цього було сформовано набір файлів з форматом .cs, які відповідають за створення описових класів.

Атрибути бази даних складаються з властивостей, які записані в класах. Тому для визначення типів даних, підбирались відповідні типи і для властивостей. Для того, щоб реалізувати зв’язки «один-до-багатьох», у класі, який може мати лише один екземпляр іншої таблиці, за допомогою компонування, було створено властивість, яка посилається на пов’язаний клас. Для відповідного класу, використовувалось створення списку класів, який реалізовується інтерфейсом ICollection.

Для опису особливих властивостей, застосовано систему атрибутів, яка надається EntityFramework. До використаних атрибутів відносяться особливості опису відображення інформації та визначення ключових і зовнішніх полів.

Також, на даному етапі було реалізовано і процеси валідації, забезпечені також атрибутами. Вони передбачають перевірку необхідності постійної передачі конкретних властивостей в клас та опис доступності введених даних, таких як пошта і номера телефонів. Окрім цього, для забезпечення вимог інформаційної системи, таких, як визначення доступних максимальних і мінімальних значень, було розроблено власні атрибути: «NumberMaxAttribute» і «NumberMinAttribute», які, як визначено на етапі проєктування, походять від спільного абстрактного класу «AbstractNumberComparison», який визначив основну логіку валідації обмеження. «NumberMaxAttribute» має похідний клас «RangeLengthAttribute», для визначення максимальної довжини переданих рядків, за рахунок розширення функціоналу. Найвищий батьківський клас у

ієрархії валідації «AbstractNumberComparison», представляється наступним розробленим кодом:

```
abstract public class AbstractNumberComparison : ValidationAttribute
{
    private double _limitNumber;

    public AbstractNumberComparison(double limitNumber)
    {
        _limitNumber = limitNumber;
    }

    public override bool IsValid(object value)
    {
        try
        {
            double number = Convert.ToDouble(value);
            return checkNumbers(number, _limitNumber);
        }
        catch
        {
            return false;
        }
    }

    protected abstract bool checkNumbers(double val1, double val2);
}
```

Усі похідні класи реалізовують структуру методу checkNumbers, надаючи можливість змінювати стиль порівняння.

Описавши структуру моделей, було реалізовано файл «GameStoreDBContext.cs», який містить в собі усі колекції можливих моделей. Для доступу до даних колекцій, використано структуру даних «список», реалізована базовим класом «List<T>«.

Для реалізації бази даних, було запущено застосунок, що дозволило EntityFramework за описаною структурою відтворити БД. Тип СКБД, версію та шлях до неї, вказано в конфігурації.

Наступним етапом було реалізовано контролери. Першим, важливим контролером, створено керування користувачами. «AccountController.cs» надає функції для реєстрації, авторизації, переадресації, відповідно до прав і обробку даних cookie. Даний компонент реалізовано, як надбудову для управління класами «CustomMembership», «CustomPrincipal» та «CustomRole». Доступ до функцій забезпечується атрибутом CustomAuthorize.

«CustomMembership» забезпечує валідацію користувачів та управління їх абстрактним представленням. Для забезпечення перевірки ролей користувачів і надання різних прав, використовується клас «CustomRole». Щоб забезпечити поєднання даних про користувачів, пошук їх ролей та перевірку прав, використовується клас «CustomPrincipal».

Окрім вищеназваного, «AccountController.cs» надає доступ до персональної сторінки користувача. Метод формування даної сторінки передбачає отримання з бази даних інформації про користувача та його замовлення, для подальшої реалізації відображення історії замовлень.

Для обміну інформацією про користувача між контролерами, застосовано моделі «LoginView» та «UserSerializeModel». Якщо коритсувач проводить процес авторизації та ідентифікації, інформація серіалізується та передається до контролерів для подальшої їх обробки. Таким чином формуються і файли cookie, що містять інформацію про користувача.

Наступним етапом було реалізовано «AdminController.cs» та «PrdouctController.cs». Дані елементи надають інструменти панелі адміністрування. Перший файл містить методи для роботи з замовленням, включаючи їх редагування і видалення. Враховуючи, що товари у вебзастосунку матимуть кілька зображень для відтворення, а кожне зображення представляється, як масив байтів, утворюється двовимірній матриці, де кожний рядок – це окрема фотографія.

Для обробки даних фотографій, використано підхід перетворення інформації в формат BASE64 і навпаки. Коли інформація про фото надходить на сервер, вона зберігається у базу даних, у вигляді масиву бітів. Але при спробі отримати зображення для відправки його у відображення, застосовується конвертація у рядок, за названим методом.

Враховуючи, що кілька контролерів мають вміти взаємодіяти з фото, функції конвертування винесено в окремий клас, під назвою «JSImageByteConvertor». Додаткові методи даного класу передбачають роботу з перевіркою якості інформації, яка надходить. Наприклад, якщо з HTML сторінки

надходить інформація про картинку, перевіряється чи її формат відповідає дозволеним. Для того, щоб перевірити рядок у форматі, який може бути переглянутим у браузері використовується наступний метод:

```
public static bool CheckImgStringToBase64
    (string input, out byte[] output)
{
    Regex exp =
    new Regex(@"^|^");
    if (checkFileFormat(input, exp))
    {
        string base64String = exp.Replace(input, String.Empty);
        return TryGetFromBase64String(base64String, out output);
    }

    output = null;
    return false;
 }
```

Код наведений вище, демонструє використання регулярних виразів, для пошуку текстового компоненту, який свідчить про кодування фото. Далі, весь інший текст, перевіряється на можливість перетворитись у бітовий рядок. У випадку невдачі, функція поверне false. Таким чином, даний метод надає додатковий шар валідації, для перевірки медіаматеріалів, які надходять.

Для початкової сторінки користувачів реалізовано «HomeController». Він слугує для відображення головної сторінки застосунку. З нього, можна виконати перехід до сторінок «ShopController», який надає доступ до перегляду товарів. Сторінки перегляду товарів, передбачають реалізацію функцій фільтрування і сортування. Дані процеси були винесені в окремий клас «ProductFilter», методи якого отримують списки продукції і повертають оновлені варіації.

Сортування відбуваються за категоріями, які записані у клас-перечислення. Контролер передає у метод ідентифікатор типу сортування, на основі якого відбувається вибір підходящого процесу. Фільтрування модифікує SQL-запит, створений за допомогою LINQ, в результаті чого з БД дістаються лише дані категорій вказаного ідентифікатору. Пошук продукції за словами, відбувається з допомогою алгоритму на основі знаходження відстані Левенштейна.

Система оцінюванню товару також реалізована у «ShopController». Методи виставлення оцінок передбачають перевірку у базі даних, чи зареєстрований користувач і чи створював він покупку товару, який він переглядає. Право

виставлення користувачем оцінки, записується у динамічний об’єкт ViewBag для передачі у представлення.

Для забезпечення часткового перегляду інформації з БД, використано фреймворк AspNetCore.Pagination. Для створення стилю кнопок, створено клас «PaginationCustomOption».

Для керування замовленнями і покупками, використовуються класи «CartController» та «OrderController». Управління корзиною дозволяє додавати товари у кошик, а також переходити до сторінки перегляду товару. Для спілкування з користувачем у динамічному режимі, використовується клас «JSONMessage». Даний клас містить поле для повідомлення, типу запиту, та статусу результату. Для того, щоб створити систему динамічних адрес у профілі користувача та на сторінці оформлення замовлень, створений похідний клас «JSONAddressMessage», який містить додаткове поле моделі адреси.

Отже, в процесі реалізації серверної частини застосунку, було реалізовано усі спроєктовані модулі. Також, встановлено взаємозв’язки між компонентами і моделями, в результат чого сформовано базу даних з використанням EntityFramework. Деякі компоненти, були додані в результаті використання інструментів розробки.

3.1.2 Програмна реалізація клієнтських модулів застосунку

Після реалізації серверної частини застосунку, було реалізовано сторінки інтерфейсу користувачів. Сторінки створювались під вимоги кожного з реалізованих контролерів.

Для початку було створено файл «_Layout.cshtml», який містить розмітку Razor. Даний файл містить спільну інформацію для всі сторінок застосунку. Він підключає основні стилі, скрипти та метатеги. Окрім цього, даний шаблон реалізовує навігаційну панель – header, та інформаційну панель - footer. Для створення деяких елементів інтерфейсу, використовувались «хелпери», які

автоматично генерували розмітку. Також, за допомогою компонування HTML з C#, було визначено динамічне формування доступу до навігаційних елементів за роллю користувача.

Наступним кроком було створено головну сторінку застосунку (додаток Г.1). На цьому етапі підібрано кольорову гаму та вигляд і анімації основних компонентів даного застосунку.

Далі, реалізовано сторінку з каталогами товарів (додаток Г.2). Дана сторінка містить інформацію про наявні та відсутні товари, коротку інформацію про них та рейтинг від покупців. Натискання кнопки «В корзину» викликає подію виклику вікна покупки.

Варто виділити реалізовану систему пошуку. Даний елемент створено з використанням випадаючого вікна, яке при вході знаходиться в закритому стані. Критерії операцій сортування, фільтрування і пошуку повністю відповідають функціям контролера «ShopController». Елемент представлений у вигляді форми, за допомогою якої надсилається запит до цього ж контролеру, але з параметрами пошуку. Розкритий варіант системи пошуку зображено у додатку Г.3.

Кожний елемент в каталозі представляє собою посилання, які відправляє користувача на сторінку перегляду детальної інформації (додаток Г.4). Особливістю даної сторінки – це можливість обговорювати товар з штучним інтелектом. При відкритті сторінки, Razor передає інформацію з БД про назву товару і дані про нього до JS, який в свою чергу передає інформацію до React.JS компоненту. Чат реалізований за допомогою компоненту бібліотеки react-chat.js. При завантаженні сторінки, створюється системне повідомлення до користувача, яке повідомляє, що бот хоче обговорити даний товар.

Системні повідомлення, які надсилаються до мовних моделей не відображаються користувачу і місять інформацію з налаштуваннями розмови, такі як ціна товару, його кількість на складі і повідомлення про потребу в поведінці, як в асистента. Коли користувач надсилає повідомлення, до його тексту додається прохання відповідати чітко лише по темі товару, а також

створюється прохання, що якщо запит не відповідає тематиці, то відповісти, що повідомлення не стосується тематики.

Відповідь надсилається не одразу, при цьому створюється значок очікування. В залежності від розміру повідомлення і швидкості інтернет-з'єднання, швидкість отримання відповіді може змінюватись. Це створює ефект спілкування з справжнім асистентом. В будь-який момент, користувач може закрити чат, при цьому, якщо був надісланий запит, який в процесі опрацювання, після формування відповіді, з'явиться значок нового повідомлення.

Для створення стабільності у спілкуванні з ботом, уся історія спілкування записується у масив об'єктів, де вказуються повідомлення на їх автор. Якщо штучна модель відмовляється оброблювати запит, він надсилається до наступної у списку. Доступ до моделей відбувається через проксі-сервера, що використовуються для тестування.

Приклад розмови з чат-ботом і очікування відповіді, зображено у додатках (додаток Г.5) При натисканні кнопки «В корзину», користувач динамічно, з використанням React.js, отримає повідомлення, в якому написано прохання підтвердити додавання у кошик (додаток Г.6). Також, реалізовано можливість вказувати кількість товарів, при цьому обмеження на некоректні значення вказані не лише на сервері, а й у клієнтській частині, за допомогою HTML-атрибутів. Тому, якщо користувач вводить некоректне число, він отримає повідомлення про помилку до того, як відбудеться відправлення запиту на сервер.

Якщо користувач уже положив товар до корзини, при повторній спробі покласти до корзини, користувачу надаватиметься повідомлення про помилку інформації, з проханням змінити кількість товару у корзині.

Також, сторінка детальної інформації надає можливість виставляти оцінки за придбані товари. Для цього використовується компонент rating-react.js. Якщо сервер повідомляє, що користувач уже купував товар, то компонент отримує статус «writable» і додається прослуховувач події виставлення оцінки. Якщо оцінка користувачем уже виставлена, при повторній спробі виставити оцінку,

буде відправлено новий запит на заміну старих даних. Виставлення оцінки ініціює перерахунок середніх даних.

Корзина (додаток Г.7) реалізована з використання функцій JavaScript. Таким чином, коли відбувається видалення товару з корзини, інформація про загальну суму оновлюється автоматично, а запит до серверу на зміну даних корзини користувача, відбувається у фоновому режимі.

Після підтвердження товарів в корзині, користувача переадресовує до сторінки оформлення замовлень (додаток Г.8). Поля для створення замовлення заповнюються автоматично, на основі даних акаунту, при цьому їх можна змінити, якщо замовник бажає змінити отримувача. Система додавання адрес (додаток Г.9), як і вікно покупки, реалізовано в динамічному стилі. Коли користувач спробує додати адресу, на сервер відправляється повідомлення з перевіркою авторизації користувача.

Користувач у будь-який момент може перейти на сторінку власного акаунту (додаток Г.10). У цьому вікні, користувач може переглянути усі раніше додані адреси у вигляді хмари тегів. Також, на цій сторінці є можливість додавати і нові теги у динамічному режимі. Якщо користувач хоче змінити дані для автозаповнення замовлення, йому надається можливість змінити дані профілю. Також, на цій сторінці користувач може отримати доступ до перегляду історії власних замовлень (додаток Г.11).

Адміністратор має доступ до кількох додаткових сторінок – перегляд інформації про товарів та виконання дій з ними, а також до перегляду замовлень.

Головні інструменти адміністратора знаходяться на сторінці перегляду інформації про товар (додаток Г.12). Тут у компактному вигляді знаходиться інформація про всі товари та надається доступ до їх редагування, додавання та видалення. Також, для зручності, адміністрації також надаються вікна сортування та фільтрування даних. Адміністратор у будь-який момент може перейти до вікна додавання або редагування товарів (додаток Г.13).

Додавання фото реалізовано за допомогою системи «Drag-and-Drop». Коли користувач перетягує зображення, або їх список, до вікна, на їх основі

створюється масив елементів input з атрибутом типу «файл». Також, адміністратор мусить вказати головне фото для товару, інакше відправка форми приведе до помилки. Вибір головного фото, також створює додатковий input для зберігання його копії.

Подібні функції, як і при роботі з товарами, надаються для роботи з замовленнями. Єдина відмінність даного набору інструментів – це використання фреймворку Plotly.js для створення графіку кількості продажів за поточний місяць (додаток Г.14).

Для забезпечення можливості відкриття застосунку на різних пристроях, з різними розширеннями екрану, було реалізовано адаптацію стилів за допомогою CSS. Частина стилів автоматично адаптувались за допомогою класів, які надаються Bootstrap. Для зменшення об’єму коду, інтерфейс описувався, як частина гнучкої методології.

Отже, інтерфейс користувача побудований з використанням різних інструментів та підходів до розробки. Інтерфейс також орієнтований на зручність його використання та швидкість переходу між компонентами, для підвищення користувацького досвіду. Усі сторінки описуються структуру контролерів і постійно вза’ємодіють з ними, обмінюючись даними через посилання, AJAX-запити та HTML-форми.

3.2 Тестування вебзастосунку

3.2.1 Вибір та обґрунтування методів тестування застосунку

У рамках кваліфікаційної роботи бакалавра, яка зосереджена на розробці вебзастосунку для продажу комп’ютерних товарів, одним з основних етапів є тестування, що включає забезпечення високої якості та стабільності програмного продукту. Було розглянуто комплексний підхід до тестування, об'єднуючи різні методики та інструментарії для всебічної перевірки компонентів системи.

Під час створення програмного продукту, проводились тести, що складались з юніт-тестів, що дозволяли перевіряти окремі модулі програми незалежно один від одного. Це сприяло швидкому і точному виявленню помилок на рівні базових компонентів, таких як методи отримання даних з контролерів і обробку даних в середині них. Для цього застосовувався фреймворк UnitTesting.

Далі було вирішено провести, інтеграційні тести, що об'єднуватимуть кілька модулів, перевіряючи їх здатність коректно взаємодіяти між собою. Використання фреймворку NUnit дозволяє структурувати тести, які моделюють реальні сценарії використання системи, забезпечуючи високий рівень довіри до інтеграції компонентів.

Для забезпечення глибокого аналізу якості користувацького інтерфейсу, клієнтської частини та функціональності контролерів вебзастосунку було вирішено використовувати ручне тестування. Цей підхід дозволяє залучити високу увагу до деталей інтерфейсу та взаємодії користувача з системою, особливо в умовах комплексних користувацьких сценаріїв.

Застосування системи документування через Test Case забезпечує структурований підхід до ручного тестування. Це не лише сприяє організації тестових процесів, але й надає засоби для точної фіксації результатів тестів, аналізу проблем та їх відстеження до моменту вирішення. Застосування детально опрацьованих тест-кейсів дозволяє концентруватися на специфічних аспектах програми, які важливі для кінцевого користувача, забезпечуючи охоплення широкого спектру можливих сценаріїв використання.

Віртуальна база даних, яка здатна до самовідновлення є складовою тестування, оскільки вона забезпечує рівність даних між тестами, мінімізує потребу вручну відновлювати стан бази до певного тесту та дозволяє виконувати більшу кількість тестів у коротші терміни. Це, у свою чергу, сприяє більш швидкому виявленню помилок і вдосконаленню програмного продукту

Цей комплексний підхід до тестування не тільки підвищує вірогідність виявлення помилок перед розгортанням застосунку, але й сприяє зростанню довіри до стабільності та продуктивності системи в цілому. Ретельне тестування

є ключовим фактором успіху програмних проєктів, зокрема у випадку, коли програмний продукт є складним і має велику кількість взаємозалежностей між власними компонентами.

3.2.2 Результати виконання тестів

Тестування відбувалось під час усього періоду розробки програмного продукту. Під час реалізації кожного компоненту, для перевірки його дієздатності, створювались Unit-тести. Фреймворк Visual Studio UnitTesting, дозволяє тестувати кожний окремий метод, без взаємодії одне з одним.

Більшість тестів стосувались лише перевірки здатності контролерів відправляти відображення, або відповідати на динамічні запити. Для реалізації даних процесів, створювався екземпляр контролеру, викликався метод і перевірялося, чи отримується не порожній результат. Це дозволяло побачити, чи є помилки в логікі компонентів.

В результаті, таким чином, було протестовано більшість методів у контролерах, як зображено на рисунку 3.1.

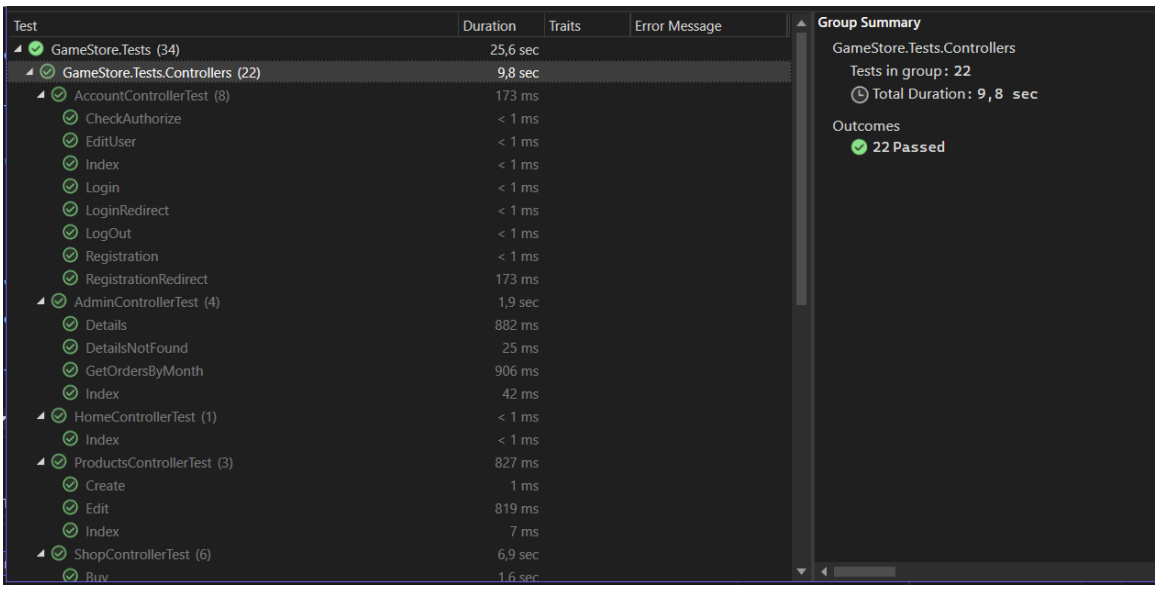


Рисунок 3.1 – Результати Unit-тестування

Для збереження цілісності даних та сталості оцінювання, було реалізовано класи моку бази даних. Даний механізм реалізовано з використанням абстрактного класу InMemoryDBProvider, який дозволяє створювати тимчасові бази даних за допомогою контексту EntityFramework, що імітують реальні умови використання без ризику засмічення чи модифікації основної бази даних. Реалізований провайдер надає можливість створювати власний контекст бази даних у будь-який момент, а також очищати уже раніше створену, створюючи подібну систему до архітектурного патерну Singleton. Для створення абстракції, даний клас містить віртуальний метод «createMoq()», який використовується іншими методами, але не має власної реалізації.

Для прив'язки контексту до того, який використовується у основному застосунку, реалізовано клас GameStoreMoq, застосовується для генерації початкових даних перед кожним тестом, забезпечуючи однакові умови для повторюваних тестів.

Для перевірки взаємодії компонентів панелі адміністрації з базою даних, реалізовано інтеграційні тести. Тести поділяються на дві частини. Перша – перевіряє можливості фільтрування і сортування контролерів, а також реакцію компонентів на неіснуючі дані. Друга частина, відповідає за послідовність процесів додавання, редагування та видалення елементів.

В основі процесів сортування та фільтрування, лежить перевірка елементів ViewBag кожного представлення. Коли в контролерах відбувається обробка параметрів опрацювання каталогів, вони записуються у ViewBag. Таким чином, від результату, який знаходиться в даному компоненті, визначалось, чи процеси проходили успішно. Обробка методів, таких як Edit, при отримані неіснуючих індексів, мусить повертати помилку «Not Found», з статусом коду 404, що і використано під час тестування.

Послідовне тестування різних контролерів, відбувається за допомогою ініціалізації спільного моку для всіх тестів. Тести проходять послідовно, вносячи зміни до БД. Після того, як метод контролеру спрацьовує, з моку дістається

контекст бази даних, що дозволяє провести пошук змін і порівняти результати з очікуваними.

Як можна побачити з прикладу, створення продукції, відповідало стандартам валідації, описаним при розробці моделей. Спроби вводити неточні дані, приводило до спричинення помилок рушієм EntityFramework. Також, частина даних, таких як зашифровані рядки у форматі Base64, були отримані шляхом використання сторонніх онлайн-інструментів.

Спочатку було протестовано інструменти опрацьовування даних про товари, після чого написані інтеграційні тести для опрацювання замовлень. Для зручності сприймання інформації, було створено систему підкласів, для розподілу перевірок функцій на блоки. Результати інтеграційного тестування зображено на рисунку 3.2.

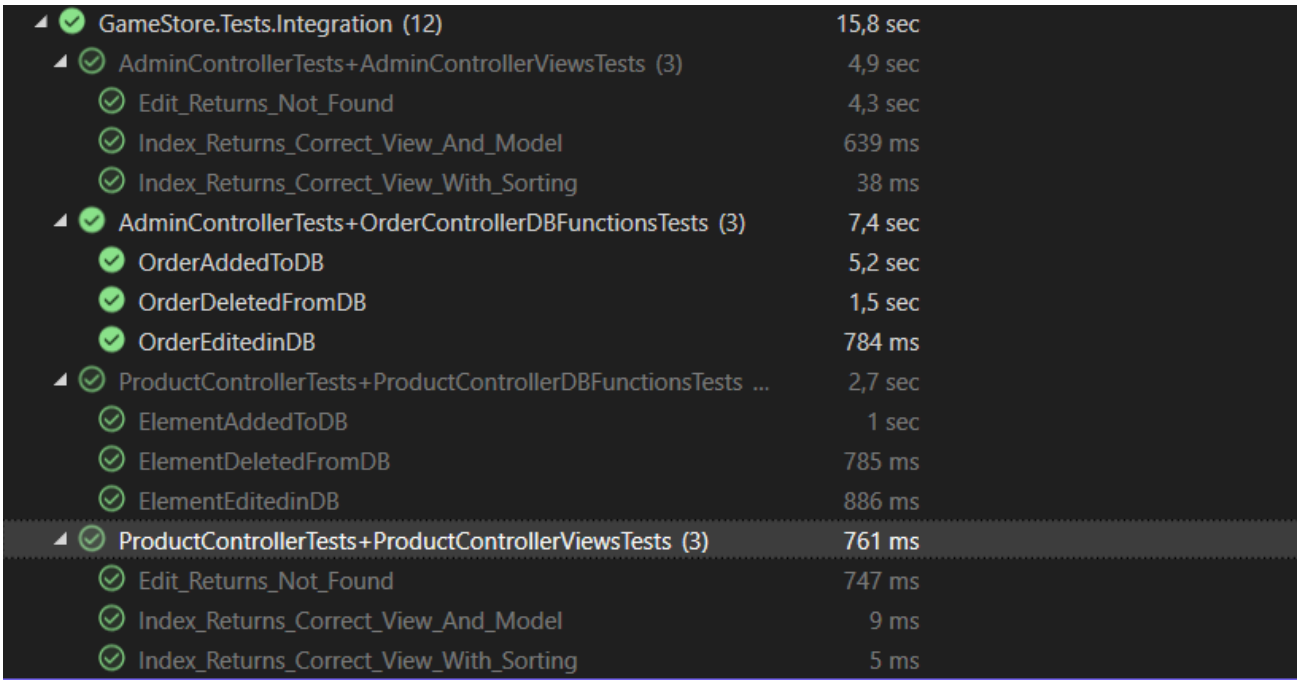


Рисунок 3.2 – Результати інтеграційних тестів

Останнім етапом у тестуванні було створення Test Case та їх виконання у ручному режимі. Перевірки відбувались для усіх основних функцій застосунку, враховуючи кроки, які можуть негативно вплинути на виконання алгоритмів, для

перевірки стресостійкості системи. Відповіді системи записувались за стандартами Test Case Management.

Для початку, було проведено тестування системи авторизації. Результати тестування надані в таблиці 3.1.

Таблиця 3.1 – Тестування процесу додавання товарів в корзину

Дія	Очікуваний результат	Успішність
ПЕРЕДУМОВИ		
Вхід на головну сторінку через посилання	З’явиться головна сторінка	Passed
ОПИС ТЕСТОВГО ВИПАДКУ		
Натиснути кнопку «Розпочати покупки»	Відкриється сторінка з каталогом товарів	Passed
Натиснути кнопку «В корзину» біля будь-якого товару	Відкриється вікно з підтвердежнням дії. Інший інтерфейс заблокується.	Passed
Закрити вікно	Інтерфейс розблокується.	Passed
Натиснути на будь-яку картку товару	Відкриється сторінка детального опису товару.	Passed
Натиснути на кнопку «В корзину»	Відкриється вікно з підтвердежнням дії. Інший інтерфейс заблокується.	Passed
Натиснути «Підтвердити»	Вікно покупки закриється.	Passed

Окрім цього, було протестовано і інші випадки додавання товару у корзину. Наприклад, операція успішно здійснюється і при спробі купити товар з каталогу товару. Спроби змінювати кількість товару успішні, при умові, що зміна кількості відбувається у межах від 1 до 20, інакше система повідомляє про помилку. Перехід між різними сторінками каталогу, ніяк не впливає на процес додавання товару в корзину. Авторизація, після відкриття вікна додавання в кошик переадресує на сторінку, з якої відкривалось вікно, але не відкриє його.

Наступним кроком складено тести, для перевірки доступності дій користувача, при відсутності авторизації або ролей. Стійкість даної системи

визначалась спочатку для критично важливих компонентів програмного застосунку, таких як безпека (Таблиця 3.2).

Таблиця 3.2 – Тестування прав користувачів різних ролей

Дія	Очікуваний результат	Успішність
ПЕРЕДУМОВИ		
Користувач має бути неавторизованим	У навігаційній панелі відображатимуться пропозиції авторизації.	Passed
ОПИС ТЕСТОВГО ВИПАДКУ		
Натиснути «В кошик».	Переадресує до сторінки авторизації.	Passed
Написати біля доменного імені слово «admin»	Переадресує до сторінки авторизації.	Passed
Перейти на сторінку кошику	Переадресує до сторінки авторизації.	Passed
Зареєструватись, як користувач	Переадресує у кошик. Користувач стане авторизованим.	Passed
Натиснути «В кошик».	Відкриється вікно з підтвердежнням дії.	Passed
Написати біля доменного імені слово «admin»	Переадресує до сторінки авторизації.	Passed
Вийти з акаунту	Переадресує на головну сторінку. Ідентифікація користувача зникне.	Passed
Увійти в акаунт адміністратора	Переадресує на головну сторінку. З’являться нові навігаційні поля.	Passed

Додатково було перевірено і інші функції, пов’язані з допуском користувачів. Спроби увійти до сторінки профілю без авторизації, приведе до переадресації до сторінки авторизації. Успішний вхід у систему переадресує до сторінки, до якої користувач намагався отримати доступ. З панелі адміністратора, користувач може успішно отримувати доступ через навігаційні елементи, так і з допомогою посилань. Спроби увійти до оформлення замовлень у неавторизованому стані, відправляють до вікна авторизації.

Наступні тести проводились для системи оцінювання товарів. Даний функціонал також відноситься до прав системи ролей, але має дещо складнішу структуру допуску (Таблиця 3.3).

Таблиця 3.3 – Тестування оцінювання товару

Дія	Очікуваний результат	Успішність
ПЕРЕДУМОВИ		
Користувач має бути неавторизований	У навігаційній панелі відображатимуться пропозиції авторизації.	Passed
ОПИС ТЕСТОВГО ВИПАДКУ		
Натиснути на карту товару	Відкриється інформація про товар.	Passed
Поставити оцінку товару	Нічого не відбудеться.	Passed
Авторизуватись	Навігаційна панель зміниться.	Passed
Поставити оцінку товару, який не був придбаний	Нічого не відбудеться.	Passed
Поставити оцінку товару, який був придбаний	Оцінка зміниться. Середня оцінка перерахується.	Passed
Перейти на сторінку товару і кілька раз змінити оцінку	Кожного разу оцінка оновлюватиметься.	Passed

Для системи важливо перевірити процес оформлення замовлення, тому було проведене детальне тестування усього процесу, починаючи від опрацювання даних корзини до формування історії на особистій сторінці користувача (Таблиця 3.4).

Таблиця 3.4 – Тестування оформлення замовлення

Дія	Очікуваний результат	Успішність
ПЕРЕДУМОВИ		
Користувач має бути авторизований	Користувач зможе знаходитись в корзині	Passed
ОПИС ТЕСТОВГО ВИПАДКУ		

Продовження таблиці 3.4

Покласти кілька товарів в корзину і перейти у неї	Корзина міститиме лише обрані товари, в записаній кількості. Сума замовлення підрахована коректно.	Passed
Видалити кілька товарів з корзини	Товари зникнуть. Сума замовлення зменшиться.	Passed
Перезавантажити сторінку	Вміст корзини не зміниться.	Passed
Натиснути «Підтвердити»	Відкриється сторінка оформлення замовлення. Дані заповняться автоматично відповідно акаунту.	Passed
Оформити замовлення	З’явиться повідомлення про покупку	Passed
Закрити повідомлення	Перенесе на сторінку користувача. В історії з’явиться пункт з замовленням.	Passed

Додатково, перевірились варіанти з внесенням до замовлення іншої особистої інформації. При цьому, змінювалась адреса, яка формувалась у новому вікні. Результати були успішні. Також, додавання адреси було протестовано на сторінці користувача, що також мало позитивні результати.

Важливим компонентом у застосунку є пошукова система. Результати її тестування надані у таблиці 3.5

Таблиця 3.5 – Тестування пошукової системи

Дія	Очікуваний результат	Успішність
ПЕРЕДУМОВИ		
Користувач знаходиться на сторінці каталогу товарів	Відображається список функцій для фільтрування.	Passed
ОПИС ТЕСТОВГО ВИПАДКУ		
Змінити категорію на ПК	В каталозі відображатимуться лише ПК.	Passed
Сортувати за спаданням	Ціни будуть в порядку спадання	Passed
Написати ключове слово для пошуку «Нтбк»	Будуть відображені лише товари, з назвами схожими на «Ноутбук».	Passed

Елементи пошукової системи тестувались за різними параметрами. Окрім цього, перевірялись комбінації різних параметрів. У всіх випадках, результати були успішні. Пошук за словом видавав результати, в яких точність залежала від розміру запиту. Таким чином, запити за великими реченнями знаходились, лише якщо вони доволі близько повторювали структуру елементів бази даних. При введені малих запитів, виводиться більшість товарів, в назві яких є шаблони схожі на введений.

У кінці тестування інших модулів, було проведено тести про процеси спілкування з асистентом на основі штучного (Таблиця 3.6).

Таблиця 3.6 – Тестування штучного асистента

Дія	Очікуваний результат	Успішність
ПЕРЕДУМОВИ		
Користувач знаходиться на сторінці будь-якого товару	У кутку екрана знаходиться символ нового повідомлення.	Passed
ОПИС ТЕСТОВГО ВИПАДКУ		
Натиснути на іконку чату	Відкриється чат з повідомленням.	Passed
Написати прохання розповісти про товар	Зі сторони бота з’явиться значок друкування.	Passed
Спробувати написати ще щось	Система не дозволить написати.	Passed
Закрити вікно чату	Коли асистент напише, з’явиться значок нового повідомлення-відповідь.	Passed
Написати питання не про товар і зачекати відповіді	Асистент повідомить, що це не стосується теми розмови.	Passed
Запитати щось з інформації, з сторінки товару і зачекати	Асистент повідомить інформацію з сторінки товару.	Passed
Перезавантажити сторінку інформації про товар	Чат буде очищено. Асистент знову привітається.	Passed

Додатково, якщо запити до бота були надто великі, то асистент ввічливо відмовлявся відповідати на них. При відключенні доступу до інтернету під час спілкування з ботом, він повідомляв про неполадки. Якщо розмова була надто

довгою, він повідомляв, що на жаль, зараз не може відповісти на усі запитання. Спроби відправити пусті повідомлення не викликали ніякої реакції.

Отже, було протестовано усі головні компоненти застосунку, використовуючи різні інструменти та підходи до перевірки якості. Частину тестів було автоматизовано, при цьому процеси, які вимагають наявності візуальних ефектів або змін, було провірено у ручному режимі і задокументовано.

3.2.3 Аналіз результатів тестування

Після завершення виконання тестів, було зроблено ряд висновків. По-перше, усі розроблені компоненти функціонують вірно. Кожний контролер завжди відправляє відображення, запуск жодного контролеру не порушує виконання процесів. Контролери здатні працювати асинхронно, що було важливо для клієнт-серверної архітектури.

Також, результати тестування показали, що система стійка до виключних ситуацій. Наприклад, система адекватно реагує на отримання недійсного елементу з бази даних, або спроби додати сутності, які не підпадають під критерії валідації. Усі виключення перехоплюються та оброблюються.

Інтеграційне тестування показало, що контролери здатні працювати коректно при взаємодії з базою даних та одне з одним. Послідовна передача інформації проходить успішно.

Ручне тестування інтерфейсу показало, що кожний компонент інтерфейсу здатний функціонувати нормально. Система ролей дотримується усіх вимог. Обмеження для різних користувачів не викликають збоїв у роботі. Функціональні компоненти клієнтської частини здатні відправляти запити до контролерів, а при тестуванні стресових ситуацій, таких, як зникнення доступу до інтернету, показало, що функціонування компонентів, які побудовані на запитах припиняється, при цьому система здатна виконувати анімаційні процеси.

Пошукова система та асистент на основі ШІ працюють відповідно до поставлених функціональних вимог. Неточний пошук адаптується під запити користувачів. Штучний інтелект здатний оперувати інформацією магазину, а також ввічливо і коректно відповідати на питання не пов’язані з покупкою.

Разом з Unit і інтеграційним тестуваннями відбувалось автоматичне тестування швидкодії, яке показувало високі результати, що свідчить про ефективність системи. Компоненти, які створювали моки бази даних працювали значно довше, але це пов’язано з постійним створенням і видаленням штучних баз даних, що сильно сповільнює процес.

Отже, в результаті усіх тестів, можна сказати, що система працює коректно, а усі її складові мають високий рівень ефективності та швидкодії. Усі поставлені функціональні задачі виконано та працюють належним чином. Компоненти забезпечують швидкий відгук при їх використанні, що добре впливає на користувацький досвід.

3.4 Керівництво користувача

Робота застосунку починається з його запуску на сервері та встановлення доменного імені. Для продуктів .NET, зазвичай використовуються сервера Azure, або будь-які, які місять або здатні встановити програмне забезпечення для функціонування .NET 8.0 і вище.

Після запуску серверу, до сторінок застосунку можна отримати доступ, якщо використати URL-адресу у пошуковому рядку будь-якого браузера, користувача перенесе на головну сторінку.

З головної сторінки, користувач може перейти до каталогу товарів, кошику і персональної сторінки акаунту. Також передбачається кнопка виходу з акаунту. Якщо користувач не зареєстрований, йому надається можливість переходу до сторінки авторизації та реєстрації. Для акаунту адміністратора передбачається можливість переходу на сторінки роботи з товарами та замовленнями.

Каталог товарів містить функції сортування, фільтрування, переходу між сторінками та вибору товарів. Фільтрування може відбуватись за ключовим словом та за категорією товарів. Категорія товарів обмежена доступним списком. Пошук за словом дозволяє вводити неточні слова або словосполучення. Сортування відбувається за датою і ціною.

Покупка товару може відбуватись як з сторінки каталогу товарів, так і з сторінки детального опису про товар. Для того, щоб перейти на сторінку детального опису, потрібно натиснути на картку товару у списку.

На сторінці детального опису, користувач може натиснути на значок чату у правій нижній частині екрану. Дана дія відкриває доступ до часу з асистентом. Усі повідомлення, які надходять асистенту мусять стосуватись теми спілкування, в іншому випадку, система відмовить відповідати на них. Повідомлення мають бути короткі і чіткі. Краще розділяти запитання на декілька, якщо вони складні. Відповідь може надходити не одразу, все залежить від швидкості інтернету. Кожний чат обмежений в кількості можливих повідомлень, тому після втрати функції написання, потрібно перезавантажити сторінку.

Оцінювання товару доступне, лише якщо користувач уже придбав даний товар. Оцінювання відбувається лише на сторінці детальної інформації про товар по п’ятибальній шкалі.

Будь-яка дія, яка потребує авторизації, в неавторизованому стані, приведе до переадресації до сторінки входу у систему.

Під час додавання товару у кошик, користувач має бути авторизований. Коли користувач обрав товар, з’являється вікно з підтвердженням дії, при цьому надається можливість обрати кількість товару, в діапазоні від 1 до 20. Закриття даного вікна, призведе до відміни дії. Після додавання товару у кошик, його не можна додати знову, до моменту доки товар не буде видалений з кошику, або не буде створене замовлення. Товари, які відсутні на складі, не можуть бути додані до кошика.

Сторінка кошика передбачає передперегляд доданих товарів перед замовленням. На даній сторінці можна видаляти уже обрані товари, а також переглядати суму, яка передбачається до сплати.

Оформлення замовлення може відбуватись лише з сторінки кошику. Для цього, користувач має бути авторизований. При оформленні замовлень, дані отримувача заповнюють автоматично, на основі інформації акаунту, з якого відбувався вхід. Дані можна змінити в відповідних полях. Перед встановленням адреси доставки, потрібно її створити. Після створення адреси, її можна обрати з списку доступних у майбутньому.

Якщо замовлення підтверджено, воно буде автоматично опрацьовано, а інформація про нього буде доступна у профілі користувача, у пункті «Історія замовлень». Замовлень може бути створено необмежену кількість разів.

На сторінці профіля, користувач може змінити дані власного акаунту, а також переглянути список раніше створених адрес. Адреси можна видаляти та додавати, якщо це не обхідно. Також, на даній сторінці можна переглянути історію усіх покупок.

3.5 Вимоги до технічних та програмних засобів

Враховуючи, що застосунок складається з клієнтської та серверної частини, вимоги до технічних описаний для кожного компоненту окремо.

Для серверної частини потрібно використовувати хмарні середовища, на яких встановлено бібліотеку .NET версії 8.0, або вище. Можливе використання власних серверів, для функціонування яких мінімально потрібно: операційна система Windows 7 або вище, SDD на 512 ГБ, оперативна пам’ять на 4 ГБ, будь-який процесор, на 4 ядра.

Для запуску клієнтської частини потрібний будь-який браузер, який здатний підтримувати JavaScript. Використання застосунку можливе на будь-якому пристрої, який може містити браузер.

3.6 Висновки до розділу 3

В результаті розробки вебзастсоунку для продажу комп’ютерних товарів, було розроблено усі компоненти застосунку. Спочатку було реалізовано серверні компоненти, які складатись з моделей та контролерів. Також, були створені шари програмного застосунку, які складались з класів фільтрування, сортування, систем передачі повідомлень та конвертування даних.

Для створення сервера, використовувалась мова програмування C# з фреймворком ASP.NET MVC та бібліотекою EntityFramework. Компонування даних елементів дозволило створити стійку до змін систему обробки запитів користувачів, доступ до якої надається за RESTful-архітектурою.

Другий тип реалізованих компонентів відповідав клієнтській частині застосунку. Були розроблені елементи інтерфейсу за допомогою HTML, CSS та Razor. Для стилізації були описані власні стилі, а також бібліотека стилів Bootstrap. Для динамічності застосунку використовувались бібліотеки JQuery та React.js з додатковими бібліотеками компонентів. Найважливіші частини даної системи стали реалізація ШІ-асистента та динамічних елементі інтерфейсу, з можливістю обміну даними між власним сервером та проксі-серверами OpenAI.

Останнім етапом у розробці вебзастосунку, стало тестування та документування результатів. Тестування показало, що розроблене програмне забезпечення функціонує, як було задумано. Система працює ефективно та стійка до виключних ситуаціях. Усі контролери здатні передавати відображення та коректно формувати елементи для відправки користувачу. Моделі здатні проводити валідацію даних, а деяка інформація, що проходить крізь системи застосунку, валідується по кілька разів.

Компоненти застосунку здатні вза’ємодіяти одне з одним у паралельному та послідовному застосуванні. База даних здатна швидко обробляти великі об’єми інформації. Відповідно, було зроблено висновки, що для роботи даного вебзастосунку не потрібні потужні пристрої.

ВИСНОВКИ

В даній кваліфікаційні роботі було пройдено та описано усі етапи розробки вебзастосунку для продажу комп'ютерних товарів з використанням різних сучасних підходів та інструментів для покращення користувацького досвіду.

Для початку, було проаналізовано предметну область. В результаті було виявлено ключові аспекти застосунків для проведення комерційної діяльності та визначено, що проблематика даної теми полягає у малій орієнтації на зручність клієнтів та малу кількість нововведень, у застосунки такого роду, зі сторони програмної реалізації.

Також, було проаналізовано інші комерційні вебзастсоунки на території України і в результаті було визначено, що користувачі надають перевагу широкому вибору товарів, зручним системам перегляду та вибору товарів, а також застосункам з якісними інструментами користувацького направлення.

На основі потреб користувачів, були визначені функціональні та нефункціональні вимоги для застосунку, які включали орієнтованість на клієнтів та адміністрацію. Також, на цьому етапі визначено, що особливими інструментами, які були реалізовані – це просунута система пошуку товарів та використання штучного інтелекту для спілкування з клієнтами.

Далі, було спроектовано програмний застосунок. Для початку, було виділено та розроблено інструменти користувацького направлення. Для покращення застосунку було також вирішено розробити систему динамічних вікона і функціонал для зберігання та оброки даних у реляційних БД у форматі Base64, основні моменти якого висвітлені на XV Всеукраїнській науковій конференції «Актуальні проблеми комп'ютерних наук АПКН-2023».

На основі визначених та спроєткованих інструментів, було визначено компоненти вебзастосунку. Для їх вза'ємодії було обрано шаблон проектування MVC, а для його коректного розбиття на функціональні категорії, була обрана клієнт-серверна архітектура. На основі обраних архітектур, була розроблена їх

взаємодія та розподіл обов'язків. Також, було розроблено структуру бази даних, яка описувала усі необхідні сутності для реалізації програмного застосунку.

Розроблення інтернет-магазину комп'ютерних товарів розпочалась з створення серверної частини застосунку. Вона включала створення моделей, які описували та реалізовували структуру бази даних, та контролерів, які дозволили реалізувати RESTful-архітектуру доступу до функціоналу застосунку. Контролери взаємодіють з логічною частиною магазину, які забезпечують функціональні особливості користувацького направлення.

Для створення клієнтської частини, використано принципи компонування інтерфейсу, а також деякі бібліотеки елементів інтерфейсу. Наприклад, таким чином створено систему спілкування з штучним інтелектом.

Тестування застосунку показало, що розроблені компоненти та функціонал працюють коректно. Логічні елементи здатні взаємодіяти одне з одним у паралельному та послідовному порядку. Ручне тестування показало, що елементи інтерфейсу адаптовані до різних розмірів дисплею, а також швидко виконують операції вза'ємодії з сервером та зміни інформації на екрані.

Розроблений застосунок може використовуватись, як справжній інтернет-магазин комп'ютерних товарів, надаючи функціонал, як для покупців, так і для адміністрації. При цьому, даний вебзастосунок може покращуватись в подальшому за рахунок розширення функціоналу та покращення систем безпеки. У застосунку можна також збільшити кількість ролей, для залучення більшої кількості працівників магазину, таких як бухгалтери, технічну підтримку та інші. Ще одним шляхом покращення може бути введення більшої кількості функціоналу на основі ШІ, таких як генерація системи рекомендацій.

На основі аналізу розроблених компонентів застосунку, можна зробити висновки, що усі поставлені задачі на кваліфікаційну роботу виконані. Проектування, реалізація та тестування були виконані в повній мірі та дозволили створити функціонуючий вебзастосунок для продажу комп'ютерних товарів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Carrier B. Website Design: How to Build Websites That Work on Any Browser (The Essential Guide to Typography for Print and Web Design). New York Zoe Lawson, 2022. 192 p.
2. Wood K. Confident Web Design: How to Design and Create Websites and Futureproof Your Career. London: Kogan Page, 2020. 312 p.
3. Jain S. Web Designing and Publishing. New Delhi: BPB Publications, 2020. 374 p.
4. Chamba-Eras L., Jacome R., Guaman-Quinche R., Coronel-Romero E. Analysis of usability of universities Web portals using the Prometheus tool // 4th International Conference on eDemocracy & eGovernment, 2019, 5 p.
5. Types of Websites. Hubspot : вебсайт. URL: https://blog.hubspot.com/website/types-of-websites (дата звернення: 02.03.2024).
6. European E-Commerce Report 2023. EuroCommerce : вебсайт. URL: https://www.eurocommerce.eu/app/uploads/2023/09/2023-european-e-commerce-report-light-version-final-19-sep.pdf (дата звернення: 02.03.2024).
7. Qin Z., Shuai Q., Wang G., Zhang P., Cao M. E-Commerce: Concepts, Principles, and Application. Singapore: Springer Nature Singapore, 2022. 424 p.
8. Delphiki J. eCommerce 360: Start your Online Business, Create your eCommerce and Sell on Marketplaces. e-Commerce book about how to make money selling online and the essentials selling in China. China : Julian Delphiki, 2020, 24p.
9. Ngwana N.D. Ecommerce Excellence: A Beginner's Guide to Online Retail. New York: Billion Ideas Company, 2023. 46 p.
10. 12 Features Every E-Commerce Website Should Care About. Agilie : вебсайт. URL: https://agilie.com/blog/12-features-every-e-commerce-website-should-care-about (дата звернення: 02.03.2024).
11. The Importance of an E-commerce Payment System for Your Business. LinkedIn : вебсайт. URL: https://www.linkedin.com/pulse/importance-ecommerce-payment-system-your-business-utpgroup-uk-rx14e/ (дата звернення: 02.03.2024).

12. Heinemann G. B2B ECommerce: Basics, Business Models and Best Practices in Business-to-Business Online Trade. Wiesbaden: Springer Fachmedien Wiesbaden, 2022. 205 p.

13. Saini S.K. The Ultimate Guide to E-commerce and Dropshipping From Zero to Hero: Proven Strategies and Insider Tips for Launching and Scaling Your E-commerce or Dropshipping Business. New Delhi: Sunil Kumar Saini, 2023. 42 p.

14. How to Improve Your Business Website. Pixolabo : вебсайт. URL: https://pixolabo.com/how-to-improve-your-business-website/ (дата звернення: 02.03.2024).

15. Calicchio S. Landing Pages: What They Are and How They Work. The Handbook that Explains All the Basics of Landing Page Marketing, from Creation to Optimisation. Milan: Stefano Calicchio, 2021. 39 p.

16. The Importance of Landing Pages. WSI World : вебсайт. URL: https://www.wsiworld.com/blog/the-importance-of-landing-pages (дата звернення: 02.03.2024)

17. Corrot P., Nussenbaum A. The Online Marketplace Advantage: Sell More, Scale Faster, and Create a World-Class Digital Customer Experience. Hoboken: Wiley, 2023. 224 p.

18. The Pros and Cons of Selling on Online Marketplaces. MarginMedia : вебсайт. URL: https://marginmedia.com.au/our-blog/the-pros-and-cons-of-selling-on-online-marketplaces (дата звернення: 02.03.2024).

19. Palasia A. Service Quality of Online Stores: A Case Study of Impact of Service Quality on Customers Satisfaction and Perceptions. Seattle: Independent author, 2023. 198 p.

20. Charlesworth A. Digital Marketing: A Practical Approach. New York: Taylor & Francis, 2023. 370 p.

21. AI in E-commerce. Firmbee : вебсайт. URL: https://firmbee.com/ai-in-e-commerce (дата звернення: 02.03.2024).

22. Як українські ритейлери використовують штучний інтелект. Deloitte : вебсайт. URL: https://www2.deloitte.com/ua/uk/pages/press-room/deloitte-press/2023/11-16.html (дата звернення: 02.03.2024).

23. Fota A. Online Shopping Intentions: Antecedents and Moderators of Shopping Intention Formation in New Fields of E-Commerce. Wiesbaden: Springer Fachmedien, Imprint: Springer Gabler, 2022. 281 p.

24. E-Shop – Administration. Intros.gr : вебсайт. URL: https://www.intros.gr/blog/item/805-e-shop-%E2%80%93-administration (дата звернення: 02.03.2024).

25. Hoffman A. Web Application Security: Exploitation and Countermeasures for Modern Web Applications. Sebastopol: O'Reilly Media, 2020. 330 p.

26. We Still Haven’t Learned the Major Lesson of the 2013 Target Hack. Slate : вебсайт. URL: https://slate.com/technology/2022/04/breached-excerpt-hartzog-solove-target.html (дата звернення: 202.03.2024).

27. Цитрус : вебсайт. URL: https://www.ctrs.com.ua/ (дата звернення: 07.03.2024).

28. Rozetka : вебсайт. URL: https://rozetka.com.ua/ (дата звернення: 07.03.2024).

29. ЧІП : вебсайт. URL: https://chip-mag.com/ua/ (дата звернення: 07.03.2024).

30. Мантур В.А. Використання кодування зображень у форматі base64 для вебзастосунків з використанням реляційних баз даних // Актуальні проблеми комп’ютерних наук АПКН-2023: Збірник наукових праць за матеріалами XV Всеукраїнської наук.-практ. конф., м. Хмельницький, 17-18 листопада 2023 р. Хмельницький, 2023. С. 184–186.

31. Stauffer M. Laravel: Up & Running. Sebastopol: O'Reilly Media, 2023. 562 p.

32. Minasian J. Master Laravel For Beginners: Get From Zero To Proficiency In The Laravel Framework: Laravel Book. Independently Published, 2023. 110 p.

33. Smith L. Software Architecture with ASP.NET 8 MVC. Fourth Edition. New York: Ls Independent Publishing, 2023. 238 p.

34. Smith L. Software Architecture with ASP.NET 6 MVC and Visual Studio 2022. Third Edition. Seattle: Amazon Digital Services LLC - Kdp, 2021. 312 p.

35. Cohen E. ASP.NET MVC Web API Razor Pages Websites Development. Seattle: Amazon Digital Services LLC - Kdp, 2023. 162 p.

36. Pagination in .NET API. Dev.to : вебсайт. URL: https://dev.to/drsimplegraffiti/pagination-in-net-api-4opp (дата звернення: 03.04.2024).

37. Hughes S. Hands-On SQL Server 2019 Analysis Services: Design and Query Tabular and Multi-dimensional Models Using Microsoft's SQL Server Analysis Services. Birmingham: Packt Publishing, 2020. 474 p.

38. Smith J.P. Entity Framework Core in Action. Second Edition. Shelter Island: Manning Publications Company LLC, 2021. 624 p.

39. Ritscher W. NET Essentials: LINQ for Databases. LinkedIn, 2021. 16 p.

40. Ventimiglia T.A. Learn JavaScript Programming: 3 Books in 1 - The Best Beginner's Guide to Learn JavaScript and Master Front End & Back End Programming. Independently Published, 2023. 586 p.

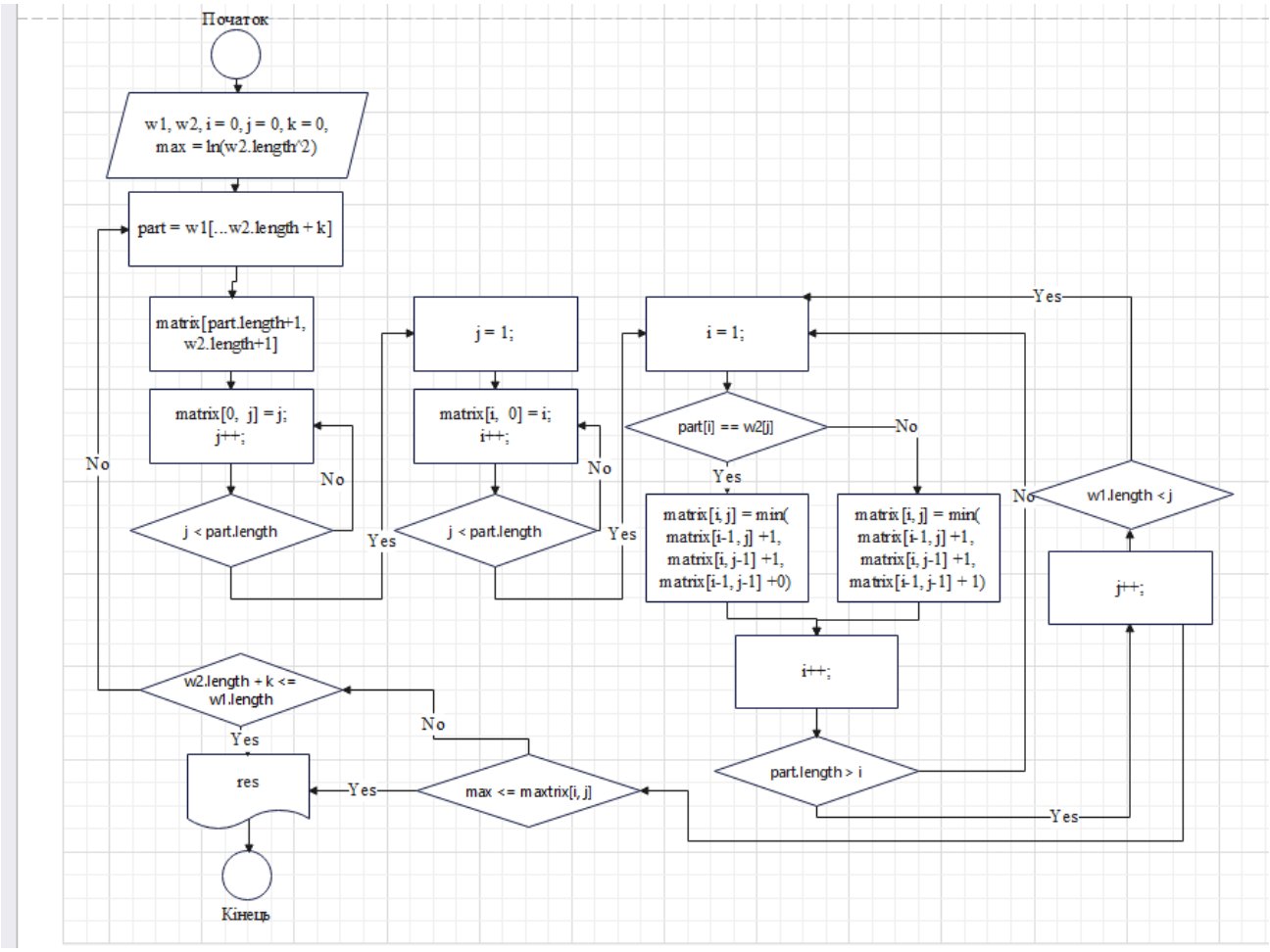
41. Babel.js : вебсайт. URL: https://babeljs.io/ (дата звернення: 28.05.2024).

42. Frontend Frameworks Popularity. GitHub Gist : вебсайт. URL: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190 (дата звернення: 03.04.2024).

43. jQuery : вебсайт. URL: https://jquery.com/ (дата звернення: 28.05.2024).

ДОДАТОК А

БЛОК-СХЕМА НЕТОЧНОГО ПОШУКУ НА ОСНОВІ ПАТЕРНУ З ВИКОРИСТАННЯМ АЛГОРИТМУ ЛЕВЕШТЕЙНА



Додаток А.1 – Блок-схема алгоритму Левенштейна для пошуку

ДОДАТОК Б

ІНФОРМАЦІЙНА СТРУКТУРА БАЗИ ДАНИХ

Таблиця Б.1 – Список властивостей сутності «Продукт»

Product		
Атрибут	Тип даних	Опис
Id	int	Ідентифікатор кожного продукту, ключова властивість.
Title	varchar(100)	Короткий опис товару.
Name	varchar(100)	Назва товару.
Description	varchar(1000)	Повний опис товару.
Price	float	Ціна на одиницю продукції.
CategoryId	int	Зовнішній ключ, який містить в собі ідентифікатор категорії. Посилається на таблицю Categories.
MainPhoto	byte(1000)	Властивість для зберігання обкладинки.

Таблиця Б.2 – Список властивостей сутності «Фото»

ProductPhoto		
Атрибут	Тип даних	Опис
Id	int	Ідентифікатор фотографій, ключова властивість.
URL	byte(1000)	Бітове представлення картинки.
ProductId	int	Зовінішній ключ, який прив’язує фото до «Product».

Таблиця Б.3 – Список властивостей сутності «Фото»

Categories		
Атрибут	Тип даних	Опис
Id	int	Ідентифікатор категорій, ключова властивість.
Name	varchar(100)	Назва категорії, до якої відносяться.

Таблиця Б.4 – Список властивостей сутності «Користувач»

User		
Атрибут	Тип даних	Опис
Id	int	Ідентифікатор користувача, ключова властивість.

Продовження таблиці Б.4

FirstName	varchar(100)	Ім’я власника акаунту.
LastName	varchar(100)	Прізвище власника акаунту.
MiddleName	varchar(100)	По-батькові власника акаунту.
Email	varchar(100)	Пошта власника акаунту.
Login	varchar(100)	Користувацький псевдонім для входження в систему та відображення на панелі навігації.
Password	varchar(100)	Пароль користувача.
ConfirmPassword	varchar(100)	Пароль для підтвердження.
Phone	varchar(100)	Телефон власника акаунту.

Таблиця Б.5 – Список властивостей сутності «Кошик»

Cart		
Атрибут	Тип даних	Опис
UserId	int	Частина первинного ключа, зовнішній ключ на «User» .
ProductId	int	Частина первинного ключа, зовнішній ключ на «Product» .
Count	int	Кількість доданого товару.

Таблиця Б.6 – Список властивостей сутності «Оцінювання»

Rating		
Атрибут	Тип даних	Опис
UserId	int	Частина первинного ключа, зовнішній ключ до «User» .
ProductId	int	Частина первинного ключа, зовнішній ключ до «Product».
Rate	int	Оцінка поставлена користувачем за товар.

Таблиця Б.7 – Список властивостей сутності «Роль»

Role		
Атрибут	Тип даних	Опис
Id	int	Ідентифікатор ролі.
Name	varchar(100)	Текстове визначення ролі.

Таблиця Б.8 – Список властивостей сутності «Роль користувача»

Rating		
Атрибут	Тип даних	Опис
UserId	int	Частина первинного ключа, зовнішній ключ до «User» .
RoleId	int	Частина первинного ключа, зовнішній ключ до «Role».

Таблиця Б.9 – Список властивостей сутності «Адреси користувача»

UserAddress		
Атрибут	Тип даних	Опис
Id	int	Ідентифікатор адреси, первинний ключ.
Address	varchar(100)	Адреса користувача.
PostCode	varchar(100)	Поштовий код адреси користувача.
UserId	int	Зовнішній ключ з ідентифікатором користувача для таблиці «User»

Таблиця Б.10 – Список властивостей сутності «Замовлення»

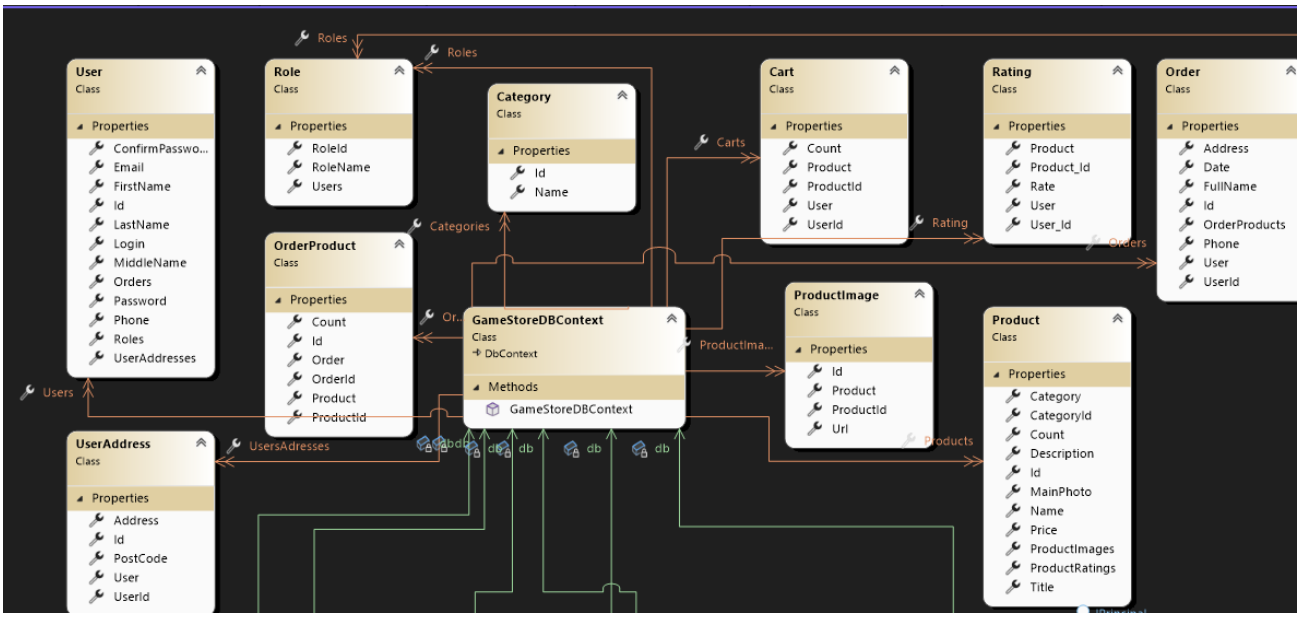
Order		
Атрибут	Тип даних	Опис
Id	int	Ідентифікатор замовлення, первинний ключ.
UserId	int	Зовнішній ключ з ідентифікатором користувача для таблиці «User»
FullName	varchar(100)	Повне ім’я замовника.
Phone	varchar(100)	Телефон замовника.
Address	varchar(100)	Адреса для замовлення.
Date	DateTime	Дата замовлення.

Таблиця Б.11 – Список властивостей сутності «Продукти замолвення»

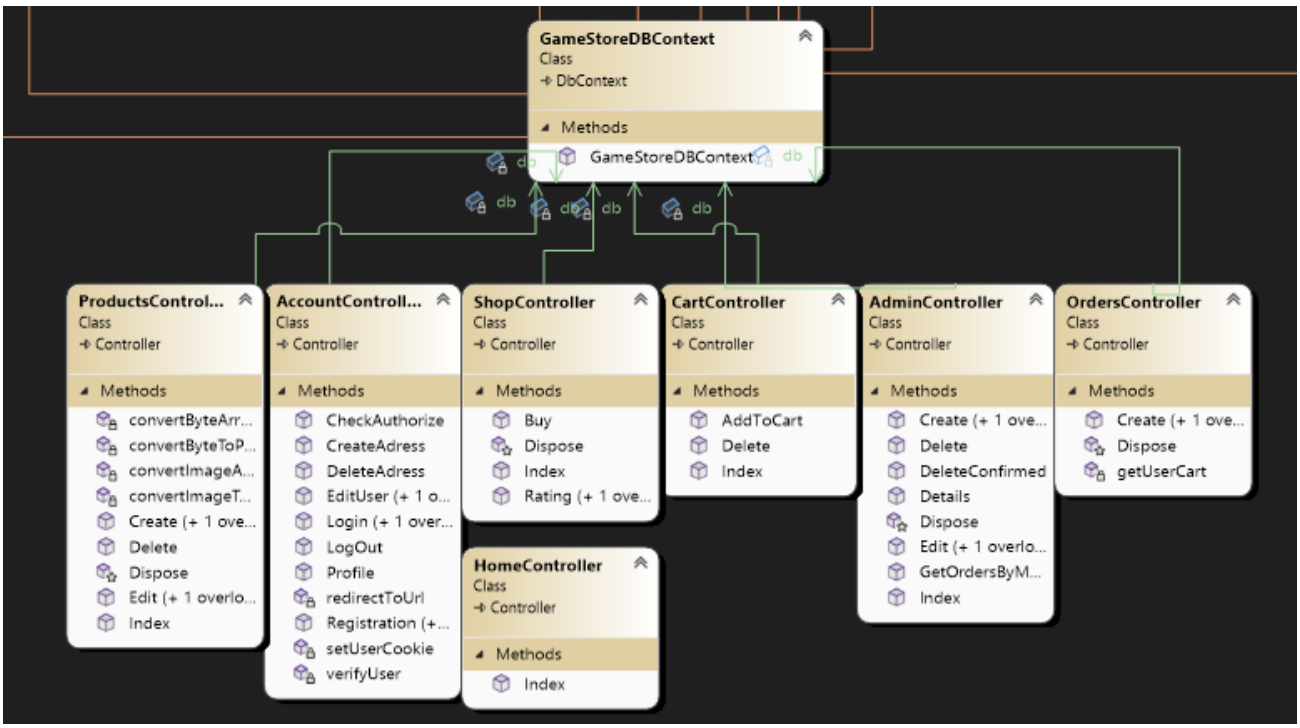
Order		
Атрибут	Тип даних	Опис
Id	int	Ідентифікатор продукту в списку, первинний ключ.
ProductId	int	Зовнішній ключ з ідентифікатором на таблицю «Product»
OrderId	int	Зовнішній ключ з ідентифікатором на таблицю «Order»
Count	int	Кількість продукції в замовленні.

ДОДАТОК В

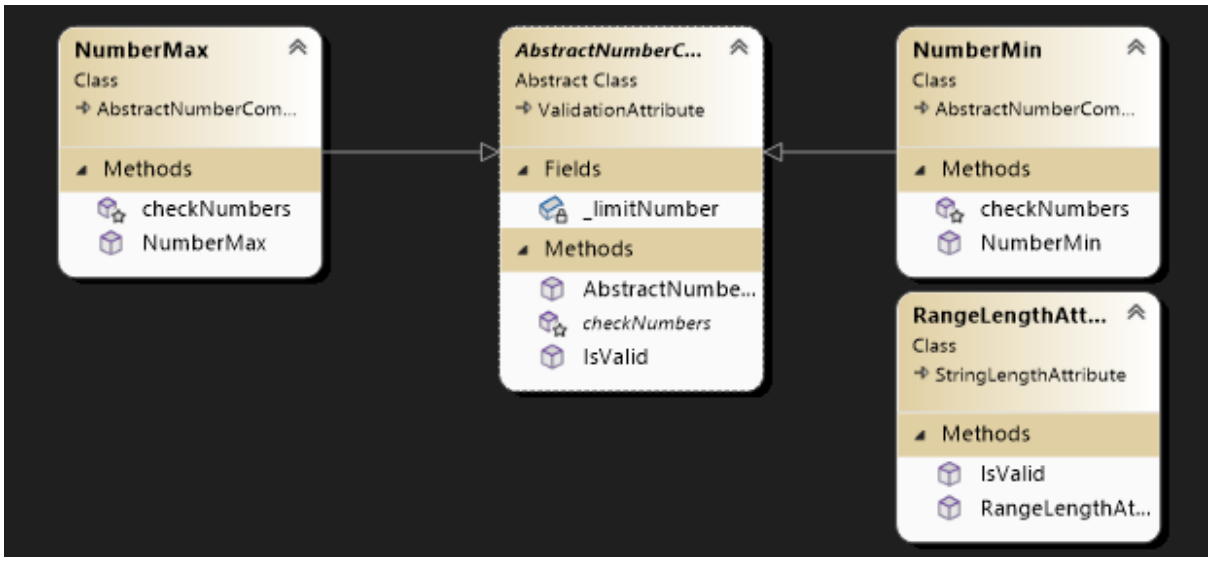
ПОВНА СТРУКТУРА КЛАСІВ ВЕБЗАСТОСУНКУ ПО ПРОДАЖУ КОМП'ЮТЕРНИХ ТОВАРІВ



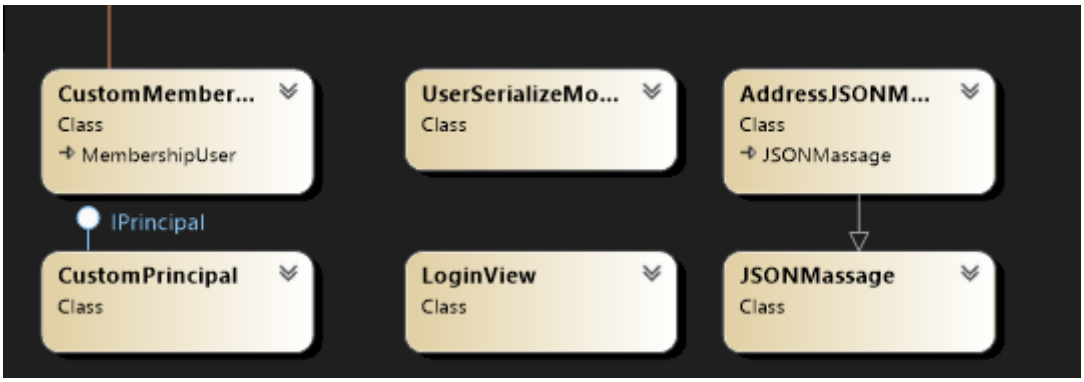
Додаток В.1 – Діаграма класів моделей



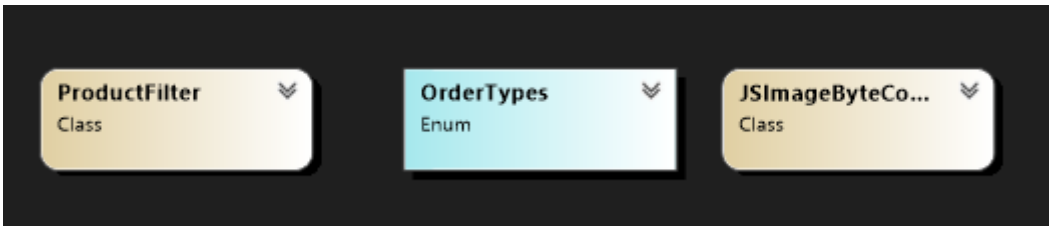
Додаток В.2 – Діаграма класів контроллерів



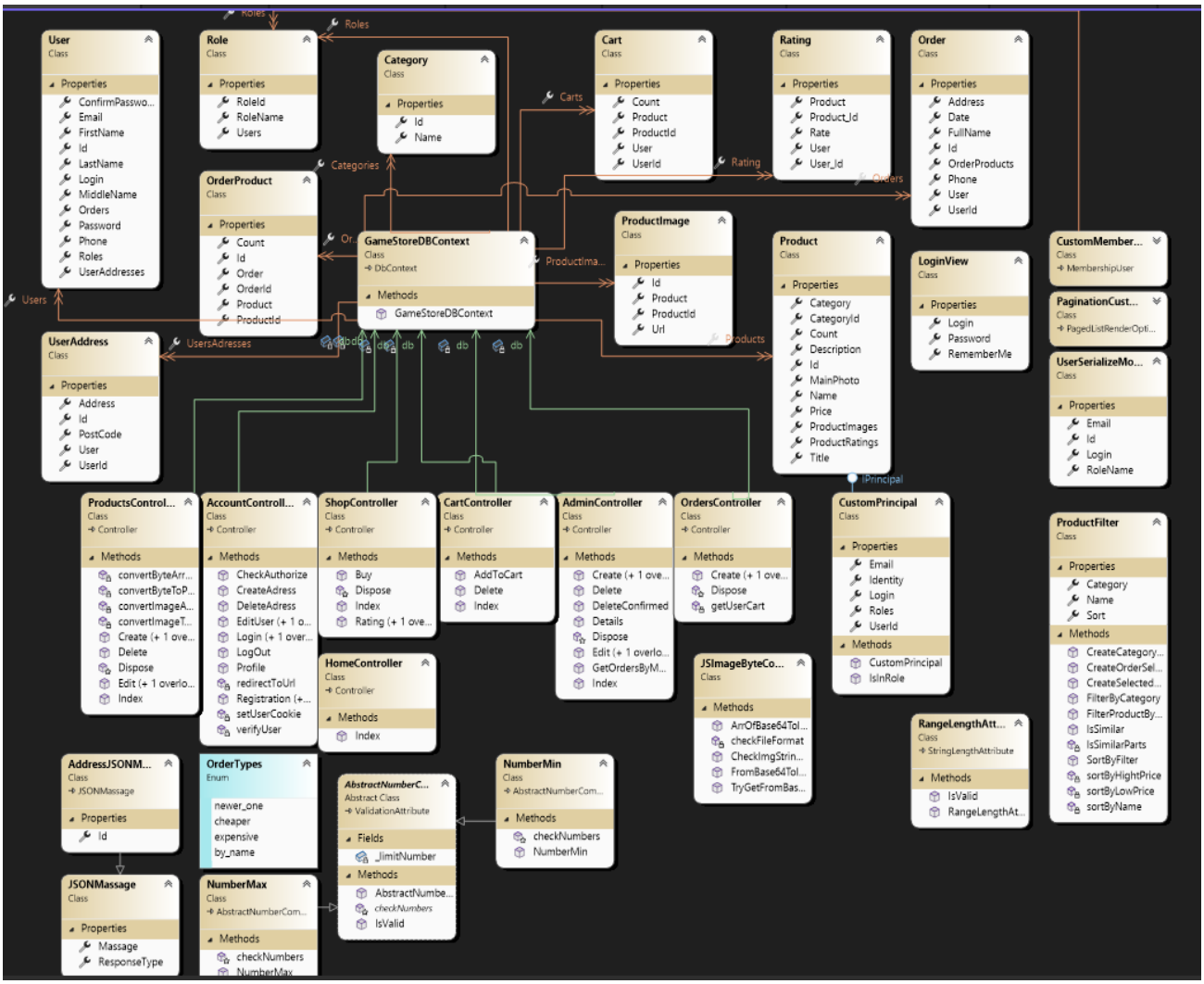
Додаток В.3 – Діаграма класів для валідації моделей



Додаток В.4 – Діаграма взаємодії з користувачем



Додаток В.5 – Діаграма класів для логічних компонентів застосунку



Додаток В.6 – Розкрита діаграма класів для усіх компонентів застосунку

ДОДАТОК Г

СТОРІНКИ ІНТЕРФЕЙСУ КОРИСТУВАЧА

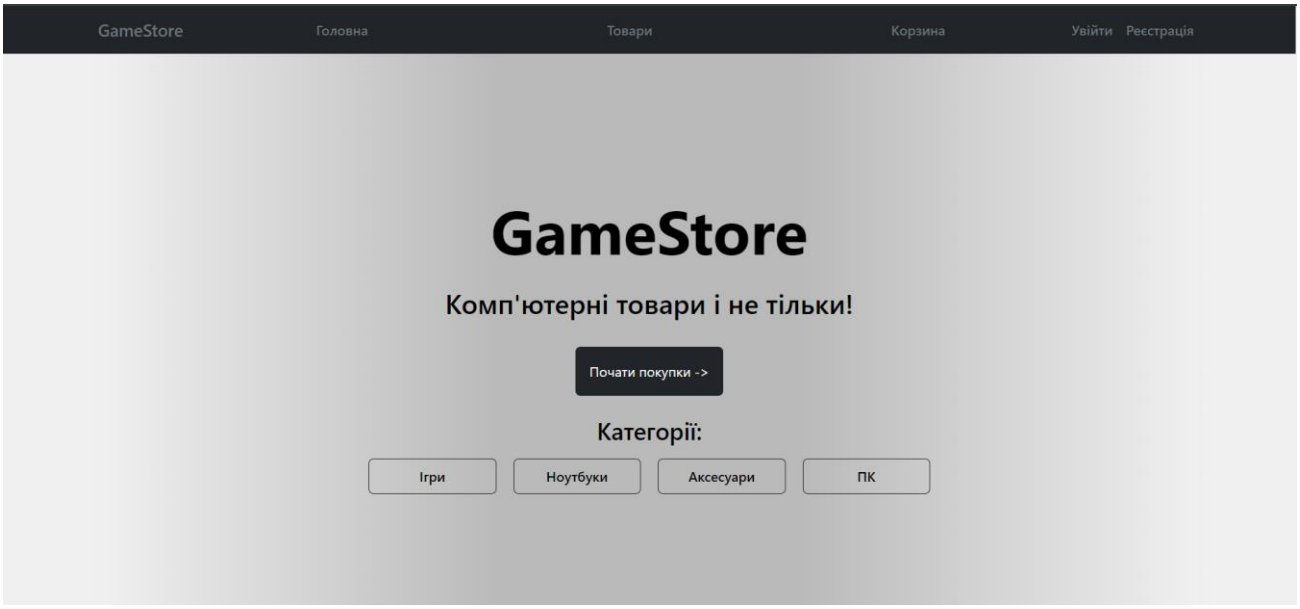


Рисунок Г.1 - Головна сторінка застосунку

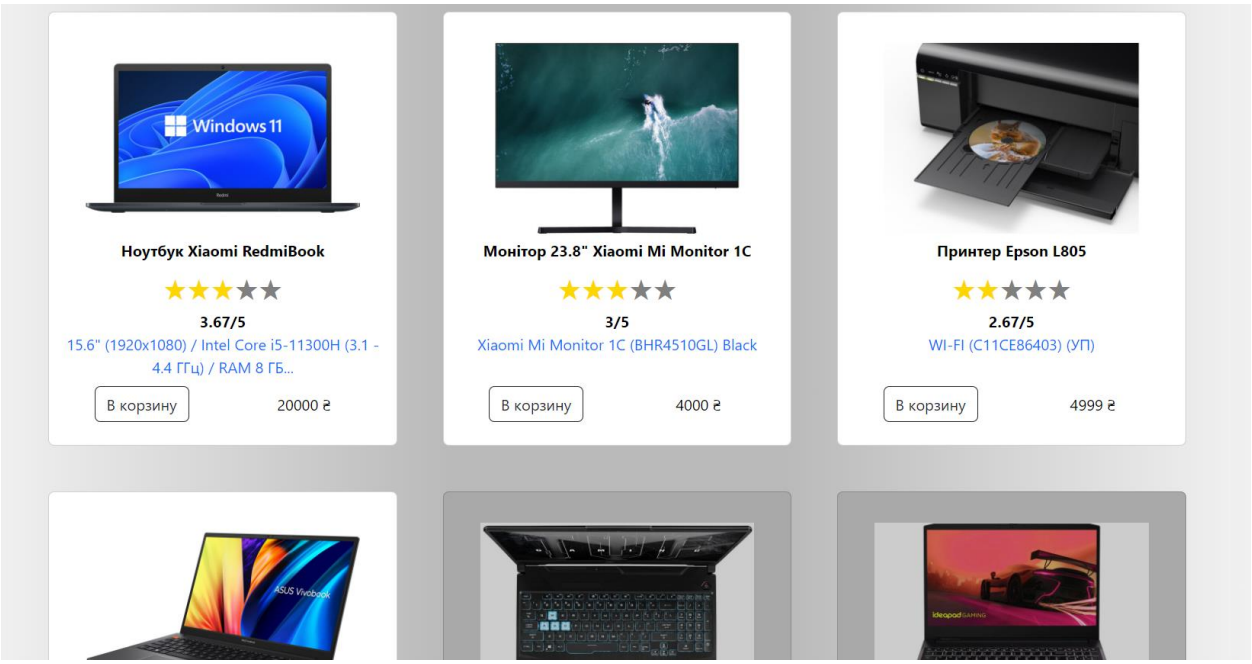


Рисунок Г.2 - Сторінка з каталогом товарів

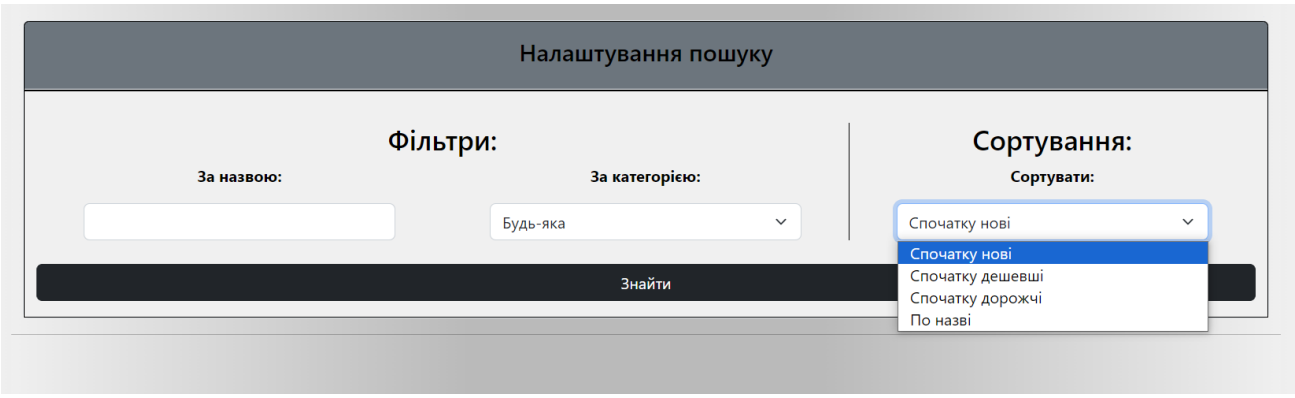


Рисунок Г.3 – Розкрите вікно пошуку

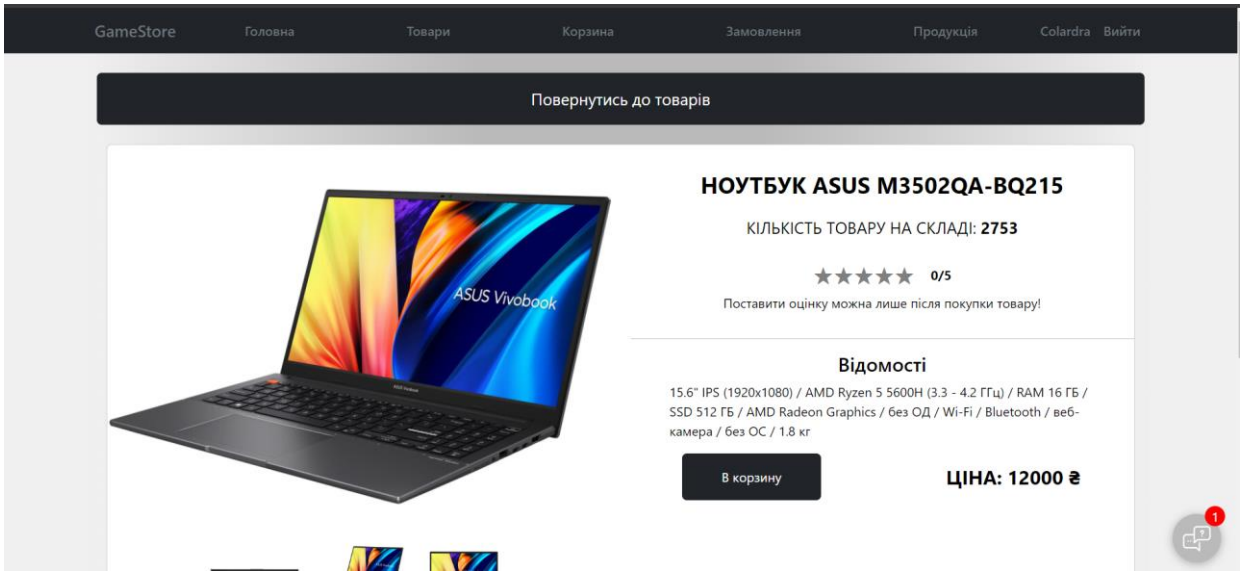


Рисунок Г.4 – Детальна інформація про товар

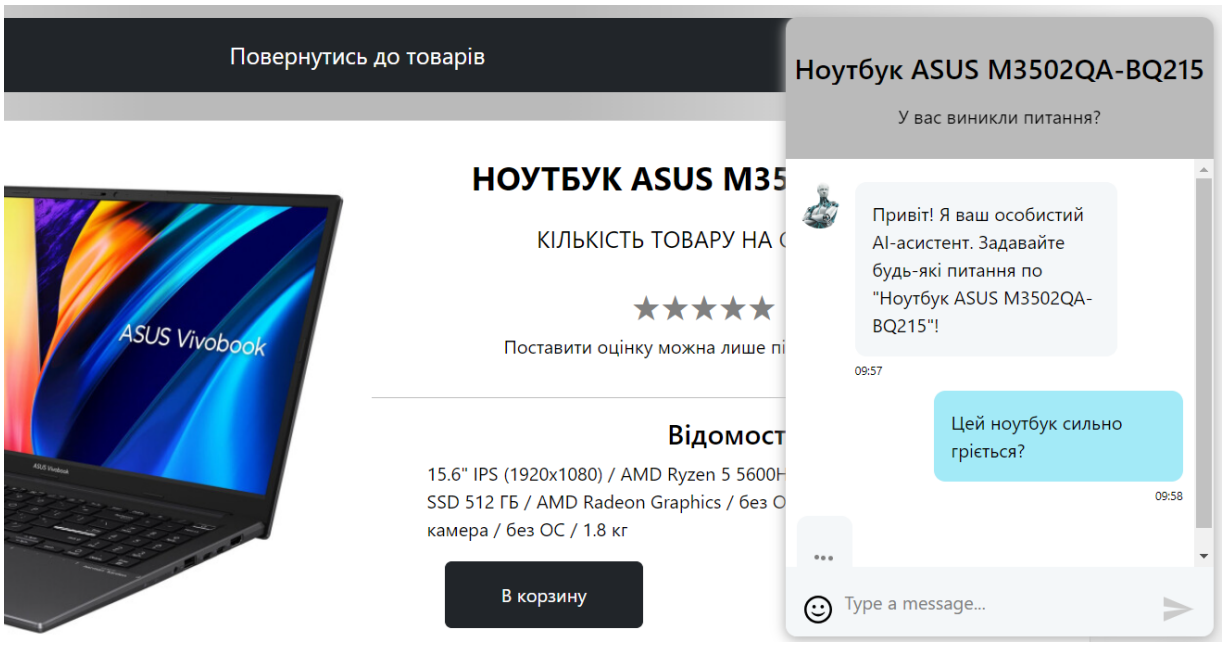


Рисунок Г.5 – Спілкування з штучним інтелектом

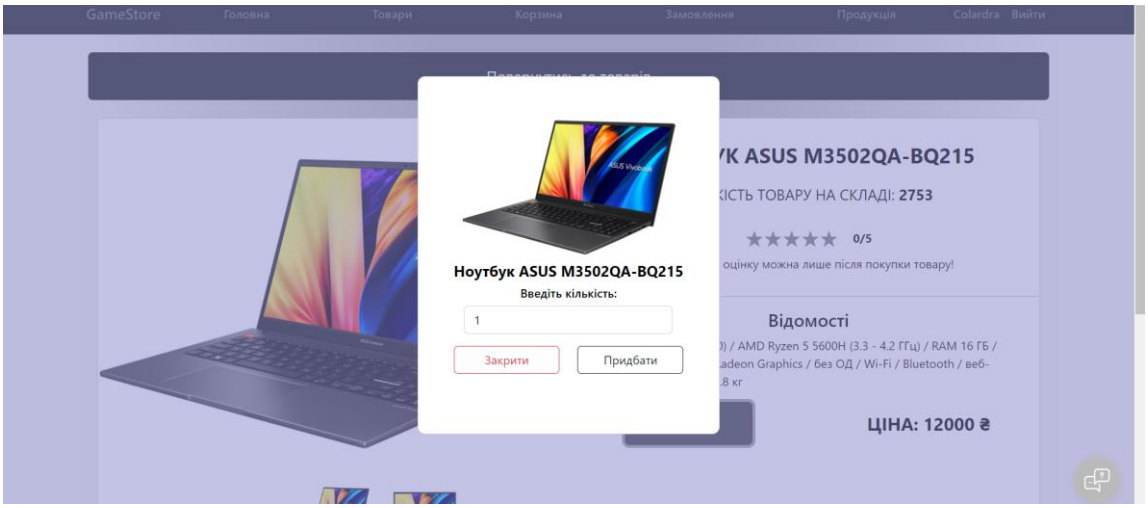


Рисунок Г.6 – Вікно покупки товару

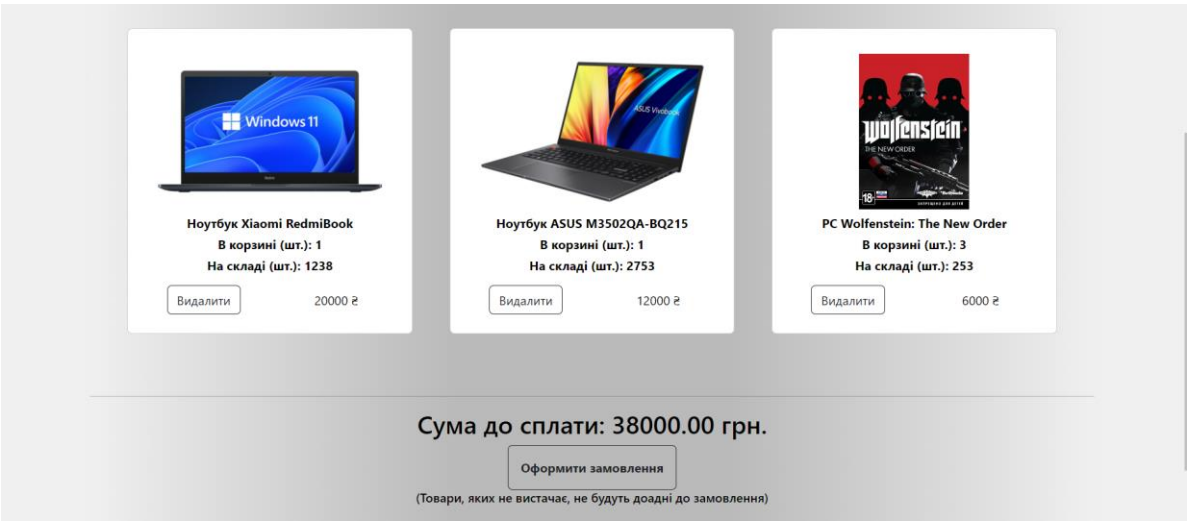


Рисунок Г.7 – Сторінка кошику

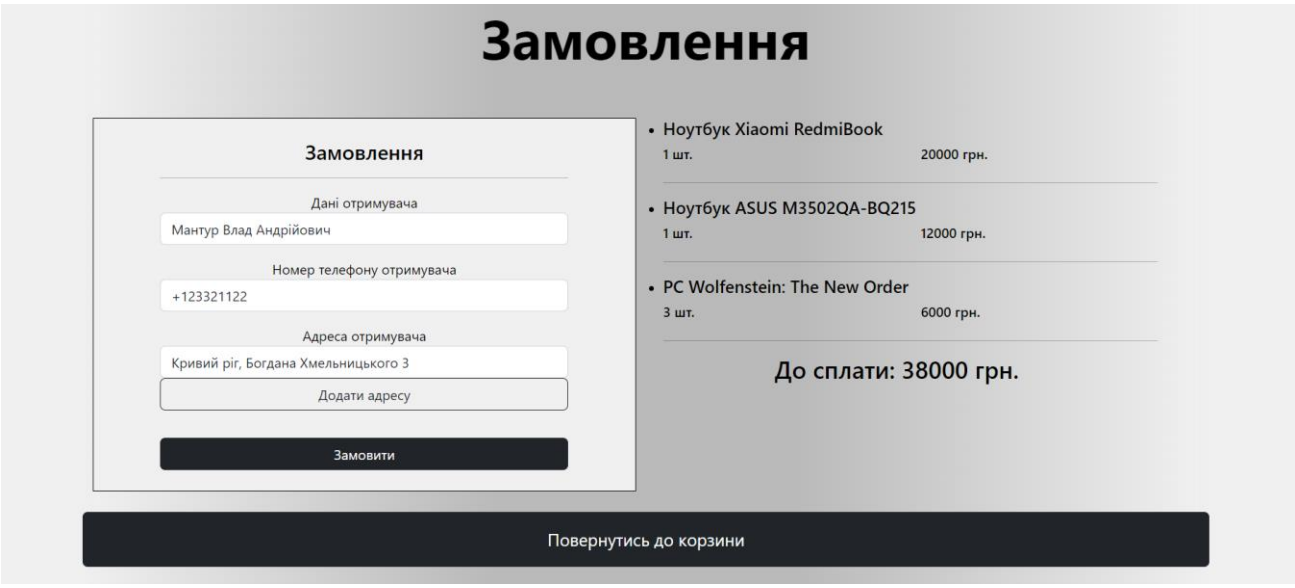


Рисунок Г.8 – Сторінка оформлення замовлень

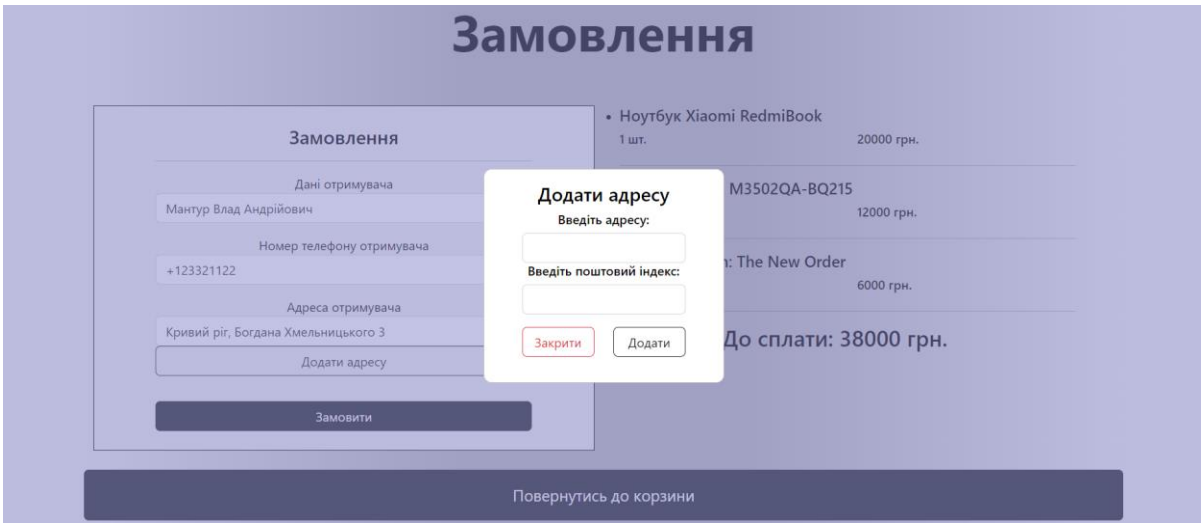


Рисунок Г.9 – Вікно додавання адрес

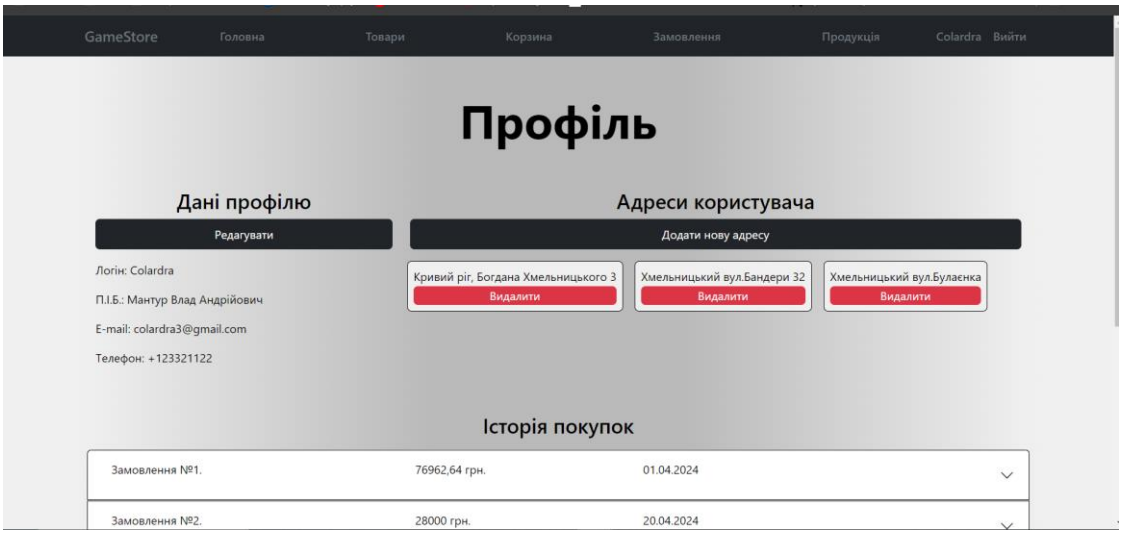


Рисунок Г.10 – Сторінка користувача

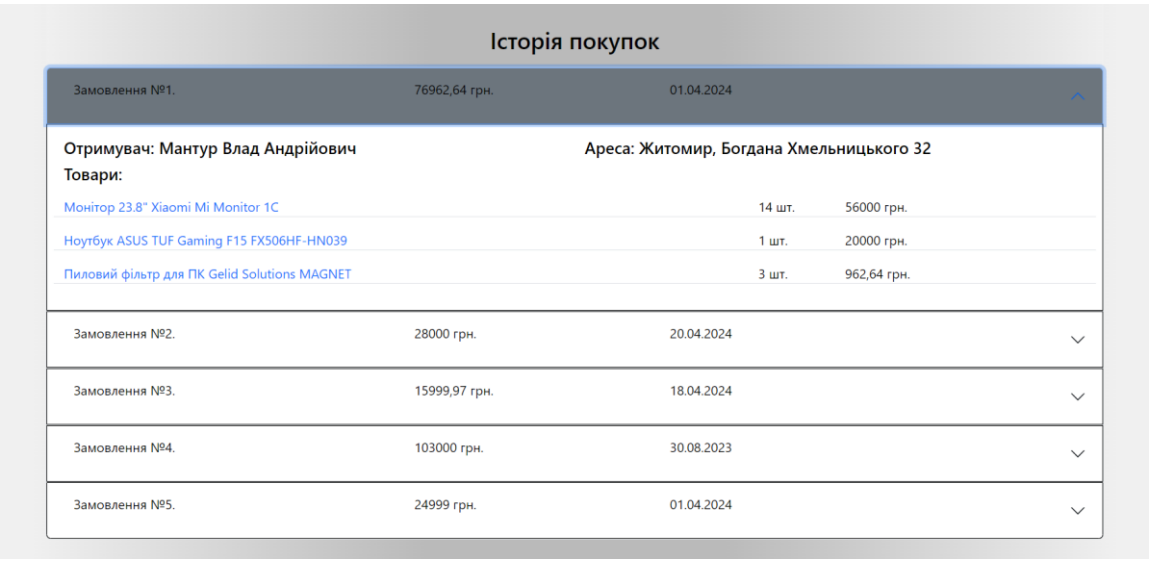


Рисунок Г.11 – Історія замовлень

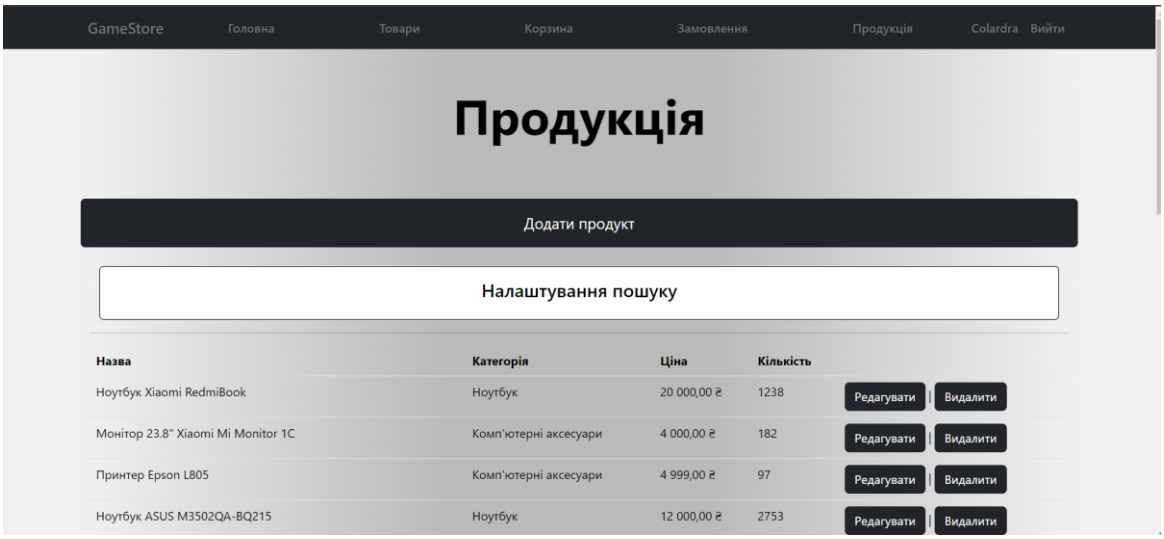


Рисунок Г.12 – Панель адміністратора

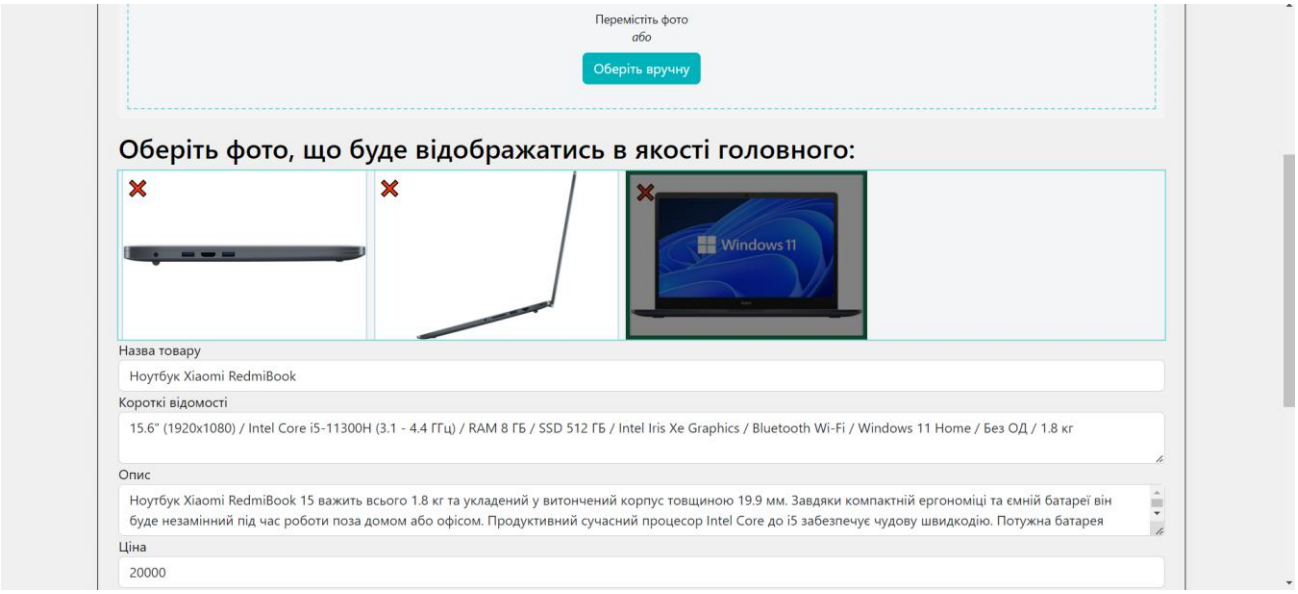


Рисунок Г.13 – Сторінка додавання товарів

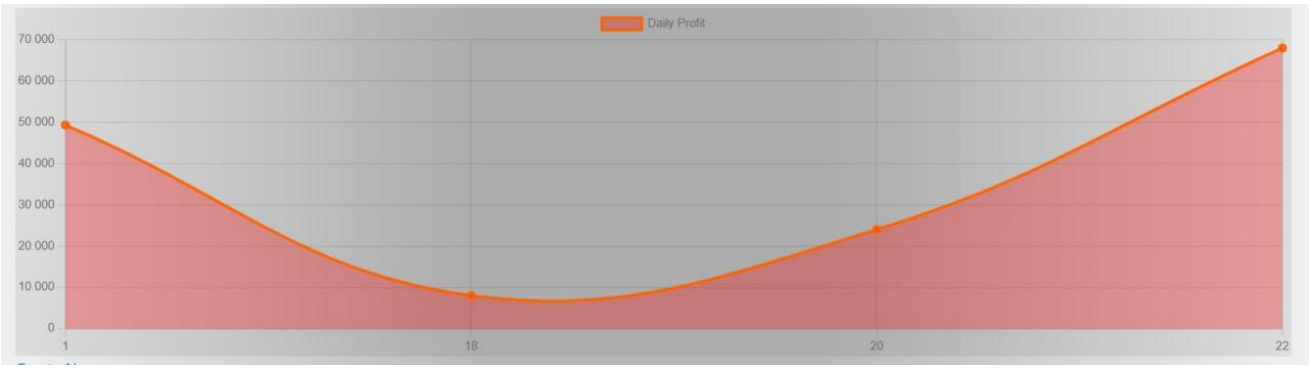


Рисунок Г.14 – Графік отримання прибутку за місяць

ДОДАТОК Д

ЛІСТИНГ КОДУ

Г.1 Програмний код модуля Product.cs

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using GameStore.Validations;
using System.Collections.Generic;
using GameStore.Validation;

namespace GameStore.Models
{
    public class Product
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }

        [Required(ErrorMessage = "Поле обов'язкове для заповнення!")]
        [Display(Name = "Назва товару")]
        public string Name { get; set; }

        [Required(ErrorMessage = "Поле обов'язкове для заповнення!")]
        [Display(Name = "Короткі відомості")]
        [DataType(DataType.MultilineText)]
        [RangeLength(200, MinimumLength = 6)]
        public string Title { get; set; }

        [Required(ErrorMessage = "Поле обов'язкове для заповнення!")]
        [Display(Name = "Опис")]
        [DataType(DataType.MultilineText)]
        public string Description { get; set; }

        [Required(ErrorMessage = "Поле обов'язкове для заповнення!")]
        [Display(Name = "Ціна")]
        [DataType(DataType.Currency)]
        public double Price { get; set; }

        [Required(ErrorMessage = "Поле обов'язкове для заповнення!")]
        [Display(Name = "Кількість товару на складі")]
        [NumberMin(0, ErrorMessage = "Мінімальна кількість товару - 0")]
        public int Count { get; set; }

        [Required(ErrorMessage = "Поле обов'язкове для заповнення!")]
        [Display(Name = "Категорія товару")]
        public int CategoryId { get; set; }
        [ForeignKey("CategoryId")]
        public Category Category { get; set; }

        public byte[] MainPhoto { get; set; }

        public virtual ICollection<ProductImage> ProductImages { get; set; }

        public virtual ICollection<Rating> ProductRatings { get; set; }
    }
}
```

Г.2 Програмний код модуля ProductController.cs

```
using GameStore;
using GameStore.Controllers;
using GameStore.Controllers.FiltrationModels;
using GameStore.Models;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web.Mvc;

namespace GameStore.Tests.Controllers
{
    [TestClass]
    public class ProductsControllerTest
    {
        [TestMethod]
        public void Index()
        {
            // Arrange
            ProductsController controller = new ProductsController();

            // Act
            ViewResult result = (ViewResult)controller.Index(1, new
ProductFilter(), 0);

            // Assert
            Assert.IsNotNull(result);
        }

        [TestMethod]
        public void Create()
        {
            // Arrange
            ProductsController controller = new ProductsController();

            // Act
            ViewResult result = (ViewResult)controller.Create();

            // Assert
            Assert.IsNotNull(result);
        }

        [TestMethod]
        public void Edit()
        {
            // Arrange
            GameStoreMoq provider = new GameStoreMoq(new GameStoreDBContext());
            ProductsController controller = new ProductsController();

            // Act
            ViewResult result = (ViewResult)controller.Edit(1);

            // Assert
            Assert.IsNotNull(result);
            provider.deleteDB();
        }

    }
}
```

Г.3 Програмний код модуля Product/Index.cshtml

```
@model PagedList.IPagedList<GameStore.Models.Product>
@using PagedList.Mvc;
@using GameStore.Controllers.Functions.HtmlSettings;

@{
 ViewBag.Title = "Продукція";
 Layout = "~/Views/Shared/_Layout.cshtml";
}

<link href="~/Content/PagedList.css" rel="stylesheet" type="text/css" />
<h2 class="text-center py-5 display-3 fw-bold">Продукція</h2>
<div class="row my-4">
 <a class="btn btn-dark p-3 return-url-text" href="/Products/Create">Додати
продукт</a>
</div>

<div class="container my-3">
 <div class="accordion" id="accordionFlushExample">
  <div class="accordion-item">
   <h2 class="accordion-header" id="flush-headingOne">
    <button class="d-block text-center accordion-button collapsed" type="button"
data-bs-toggle="collapse" data-bs-target="#flush-collapseOne" aria-
expanded="false" aria-controls="flush-collapseOne">
     <h4 class="text-center">Налаштування пошуку</h4>
    </button>
   </h2>
   <div id="flush-collapseOne" class="accordion-collapse collapse" aria-
labelledby="flush-headingOne" data-bs-parent="#accordionFlushExample">
    <section class="py-3 black-border light-gray-background">
     @using (Html.BeginForm("Index", "Products"))
     {
      <div class="container">
       <div class="container-fluid wraped flex-column">
        <div class="wrapper row">
         <div class="col-md-8 my-3 filter-right-border">
          <h3 class="text-center">Фільтри:</h3>
          <div class="wraped row justify-content-sm-center">
           <div class="text-center col-md-6 flex-column">
            <p class="fw-bold mb-0">За назвою:</p>
            <div class="row px-5 pt-3">@Html.TextBox("Name", ViewBag.NameSearch
as string, new { @class = "form-control" })</div>
           </div>

           <div class="text-center col-md-6 flex-column">
            <p class="fw-bold mb-0">За категорією:</p>
            <div class="row px-5 pt-3">@Html.DropDownList("Category",
ViewBag.CategoriesFilter as IEnumerable<SelectListItem>, new { @class = "form-
select" })</div>
           </div>
          </div>
         </div>
         <div class="col-md-4 my-3">
          <h3 class="text-center">Сортування:</h3>
          <div class="wraped row justify-content-sm-center">
           <div class="text-center flex-column justify-content-center col-12">
            <p class="fw-bold mb-0"> Сортувати:</p>
            <div class="row px-5 pt-3">@Html.DropDownList("Sort",
ViewBag.SortFilter as IEnumerable<SelectListItem>, new { @class = "form-select"
})</div>
           </div>
          </div>
         </div>
        </div>
        <div class="wraped text-center row">
```

```
          <input class="btn btn-dark mt-2" type="submit" value="Знайти" />
         </div>
        </div>
       </div>
       <input type="checkbox" name="filtering" class="unvisible_checkbox"
value="1" checked />
      }
     </section>
    </div>
   </div>
 </div>
</div>

<hr />

<table class="table">
 <tr>
  <th>
   Назва
  </th>
  <th>
   Категорія
  </th>
  <th>
   Ціна
  </th>
  <th>
   Кількість
  </th>
  <th></th>
 </tr>

 @foreach (var item in Model)
 {
  <tr>
   <td>
    @Html.DisplayFor(modelItem => item.Name)
   </td>
   <td>
    @Html.DisplayFor(modelItem => item.Category.Name)
   </td>
   <td>
    @Html.DisplayFor(modelItem => item.Price)
   </td>
   <td>
    @Html.DisplayFor(modelItem => item.Count)
   </td>
   <td>
    @Html.ActionLink("Редагувати", "Edit", new { id = item.Id }, new { @class =
"btn btn-dark" }) |
    @Html.ActionLink("Видалити", "Delete", new { id = item.Id }, new { @class =
"btn btn-dark" })
   </td>
  </tr>
 }

</table>

@Html.PagedListPager(Model,
 page => Url.Action("Index",
 new { page, name = ViewBag.NameSearch, category = ViewBag.CoosedCategory, Sort
= ViewBag.Sort }),
 PaginationCustomOption.ShowOnlySomePages(5, "Початок", "Кінець", "Попередня",
"Наступна"))
```

Г.4 Програмний код модуля Product/Edit.cshtml

```
@model GameStore.Models.Product

@{
 ViewBag.Title = "Редагувати продукт";
 Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2 class="text-center py-5 display-3 fw-bold">Додати продукт</h2>

<div class="row align-items-center m-0">
 <div class="mx-auto col-sm-12 col-md-6 col-lg-4 black-border light-gray-
background p-4 full-form">
  @using (Html.BeginForm("Edit", "Products", FormMethod.Post, new { enctype =
"multipart/form-data" }))
  {
   @Html.AntiForgeryToken()

   <div class="form-horizontal">
    <h4>Product</h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })

    <fieldset class="upload_dropZone text-center mb-3 p-4">

     <legend class="visually-hidden">Image uploader</legend>

     <svg class="upload_svg" width="60" height="60" aria-hidden="true">
      <use href="#icon-imageUpload"></use>
     </svg>

     <p class="small my-2">Перемістіть фото<br><i>або</i></p>

     <input id="upload_image_background" data-post-name="image_background"
         data-post-url="https://someplace.com/image/uploads/backgrounds/"
         class="position-absolute invisible"
         type="file"
         multiple accept="image/jpeg, image/png" />

     <label class="btn btn-upload mb-3" for="upload_image_background">Оберіть
вручну</label>

    </fieldset>

    <h2>Оберіть фото, що буде відображатись в якості головного: </h2>
    <fieldset>
     <div calss="gallery_scroll container">
      <div class="upload_gallery d-flex justify-content-left mb-0">
       @{
        if (ViewBag.selectedImage == null)
        {
         ViewBag.selectedImage = "";
        }

        <input class='main_img_container unvisible_checkbox'
name='selectedImage' value="@ViewBag.selectedImage">

        if (ViewBag.ImageFile != null)
        {
         List<string> arrOfImageFiles = (List<string>)ViewBag.ImageFile;
         foreach (string str in arrOfImageFiles)
         {
          string selectedImgClassString = "";
          if (ViewBag.selectedImage != null && str ==
(string)ViewBag.selectedImage)
```

```
          {
           selectedImgClassString = "selected_img";
          }
          <div class="gallery_cheked_img mx-1 @selectedImgClassString">
           <img class="upload_img" alt="ViewError" src="@str">
           <p class='close_span '
onclick="$(this).parent().remove();">&#10060</p>
           <input class="unvisible_checkbox" name="imageFile" value="@str">
          </div>
         }
        }
       }
      </div>
     </div>
    </fieldset>

    <div class="form-group">
     @Html.LabelFor(model => model.Name, htmlAttributes: new { @class =
"control-label" })
     <div class="col-md-12 row m-0">
      @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class =
"form-control" } })
      @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-
danger" })
     </div>
    </div>

    <div class="form-group">
     @Html.LabelFor(model => model.Title, htmlAttributes: new { @class =
"control-label" })
     <div class="col-md-12 row m-0">
      @Html.EditorFor(model => model.Title, new { htmlAttributes = new { @class
= "form-control" } })
      @Html.ValidationMessageFor(model => model.Title, "", new { @class = "text-
danger" })
     </div>
    </div>

    <div class="form-group">
     @Html.LabelFor(model => model.Description, htmlAttributes: new { @class =
"control-label" })
     <div class="col-md-12 row m-0">
      @Html.EditorFor(model => model.Description, new { htmlAttributes = new {
@class = "form-control" } })
      @Html.ValidationMessageFor(model => model.Description, "", new { @class =
"text-danger" })
     </div>
    </div>

    <div class="form-group">
     @Html.LabelFor(model => model.Price, htmlAttributes: new { @class =
"control-label" })
     <div class="col-md-12 row m-0">
      @Html.EditorFor(model => model.Price, new { htmlAttributes = new { @class
= "form-control" } })
      @Html.ValidationMessageFor(model => model.Price, "", new { @class = "text-
danger" })
     </div>
    </div>

    <div class="form-group">
     @Html.LabelFor(model => model.Count, htmlAttributes: new { @class =
"control-label" })
     <div class="col-md-12 row m-0">
      @Html.EditorFor(model => model.Count, new { htmlAttributes = new { @class
= "form-control" } })
```

```
      @Html.ValidationMessageFor(model => model.Count, "", new { @class = "text-
danger" })
     </div>
    </div>

    <div class="form-group">
     @Html.LabelFor(model => model.CategoryId, htmlAttributes: new { @class =
"control-label" })
     <div class="col-md-12 row m-0">
      @Html.DropDownListFor(model => model.CategoryId, ViewBag.Categories as
SelectList,  new { @class = "form-control" })
      @Html.ValidationMessageFor(model => model.CategoryId, "", new { @class =
"text-danger" })
     </div>
    </div>

    <div class="form-group">
     <div class="form-group row d-flex justify-content-center m-0 mt-3">
      <input type="submit" value="Зберегти" class="btn btn-dark" />
     </div>
    </div>
   </div>
  }
 </div>
</div>
<div class="row my-4">
 <a class="btn btn-dark p-3 return-url-text"
href="/Products/Index">Повернутись</a>
</div>

@section scripts {
 <script src="~/Scripts/page_scripts/DragAndDrop.js"></script>
}
```

Г.5 Програмний код модуля Product/Create.cshtml

```
@model GameStore.Models.Product

@{
 ViewBag.Title = "Додати продукт";
 Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2 class="text-center py-5 display-3 fw-bold">Додати продукт</h2>
<div class="row align-items-center m-0">
 <div class="mx-auto col-sm-12 col-md-6 col-lg-4 black-border light-gray-
background p-4 full-form">
  @using (Html.BeginForm("Create", "Products", FormMethod.Post, new { enctype =
"multipart/form-data" }))
  {
   @Html.AntiForgeryToken()

   <div class="form-horizontal">
    <h4>Product</h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })

    <fieldset class="upload_dropZone text-center mb-3 p-4">

     <legend class="visually-hidden">Image uploader</legend>

     <svg class="upload_svg" width="60" height="60" aria-hidden="true">
      <use href="#icon-imageUpload"></use>
     </svg>

     <p class="small my-2">Перемістіть фото<br><i>або</i></p>
```

```
     <input id="upload_image_background" data-post-name="image_background"
         data-post-url="https://someplace.com/image/uploads/backgrounds/"
         class="position-absolute invisible"
         type="file"
         multiple accept="image/jpeg, image/png" />

     <label class="btn btn-upload mb-3" for="upload_image_background">Оберіть
вручну</label>

    </fieldset>

    <h2>Оберіть фото, що буде відображатись в якості головного: </h2>
    <fieldset>
     <div calss="gallery_scroll container">
      <div class="upload_gallery d-flex justify-content-left mb-0">
       @{
        if (ViewBag.selectedImage == null)
        {
         ViewBag.selectedImage = "";
        }

        <input class='main_img_container unvisible_checkbox'
name='selectedImage' value="@ViewBag.selectedImage">

        if (ViewBag.ImageFile != null)
        {
         string[] arrOfImageFiles = (string[])ViewBag.ImageFile;
         foreach (string str in arrOfImageFiles)
         {
          string selectedImgClassString = "";
          if (ViewBag.selectedImage != null && str ==
(string)ViewBag.selectedImage)
          {
           selectedImgClassString = "selected_img";
          }
          <div class="gallery_cheked_img mx-1 @selectedImgClassString">
           <img class="upload_img" alt="ViewError" src="@str">
           <p class='close_span '
onclick="$(this).parent().remove();">&#10060</p>
           <input class="unvisible_checkbox" name="imageFile" value="@str">
          </div>
         }
        }
       }
      </div>
     </div>
    </fieldset>

    <div class="form-group">
     @Html.LabelFor(model => model.Name, htmlAttributes: new { @class =
"control-label" })
     <div class="col-md-12 row m-0">
      @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class =
"form-control" } })
      @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-
danger" })
     </div>
    </div>

    <div class="form-group">
     @Html.LabelFor(model => model.Title, htmlAttributes: new { @class =
"control-label" })
     <div class="col-md-12 row m-0">
      @Html.EditorFor(model => model.Title, new { htmlAttributes = new { @class
= "form-control" } })
```

```
      @Html.ValidationMessageFor(model => model.Title, "", new { @class = "text-
danger" })
     </div>
    </div>

    <div class="form-group">
     @Html.LabelFor(model => model.Description, htmlAttributes: new { @class =
"control-label" })
     <div class="col-md-12 row m-0">
      @Html.EditorFor(model => model.Description, new { htmlAttributes = new {
@class = "form-control" } })
      @Html.ValidationMessageFor(model => model.Description, "", new { @class =
"text-danger" })
     </div>
    </div>

    <div class="form-group">
     @Html.LabelFor(model => model.Price, htmlAttributes: new { @class =
"control-label" })
     <div class="col-md-12 row m-0">
      @Html.EditorFor(model => model.Price, new { htmlAttributes = new { @class
= "form-control" } })
      @Html.ValidationMessageFor(model => model.Price, "", new { @class = "text-
danger" })
     </div>
    </div>

    <div class="form-group">
     @Html.LabelFor(model => model.Count, htmlAttributes: new { @class =
"control-label" })
     <div class="col-md-12 row m-0">
      @Html.EditorFor(model => model.Count, new { htmlAttributes = new { @class
= "form-control" } })
      @Html.ValidationMessageFor(model => model.Count, "", new { @class = "text-
danger" })
     </div>
    </div>

    <div class="form-group">
     @Html.LabelFor(model => model.CategoryId, htmlAttributes: new { @class =
"control-label" })
     <div class="col-md-12 row m-0">
      @Html.DropDownListFor(model => model.CategoryId, ViewBag.Categories as
SelectList,  new { @class = "form-control"  })
      @Html.ValidationMessageFor(model => model.CategoryId, "", new { @class =
"text-danger" })
     </div>
    </div>

    <div class="form-group">
     <div class="form-group row d-flex justify-content-center m-0 mt-3">
      <input type="submit" value="Створити" class="btn btn-dark" />
     </div>
    </div>
   </div>
  }
 </div>
</div>
<div class="row my-4">
 <a class="btn btn-dark p-3 return-url-text"
href="/Products/Index">Повернутись</a>
</div>

@section scripts {
 <script src="~/Scripts/page_scripts/DragAndDrop.js"></script>
}
```

Г.6 Програмний код модуля ProductFilter.cs

```
using GameStore.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Diagnostics;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Xml.Linq;

namespace GameStore.Controllers.FiltrationModels
{
 public class ProductFilter
 {
  public string Name { get; set; }
  public int? Category { get; set; }
  public int? Sort { get; set; }

  public IQueryable<Product> FilterProductByText(IQueryable<Product> products)
  {
   if (String.IsNullOrEmpty(Name)) return products;
   var filteredProducts = products
    .ToList()
    .Where(
    s => IsSimilarParts(
     s.Name,
     Name,
     (int)Math.Round(Math.Log(Math.Pow(Math.Min(s.Name.Length, Name.Length),
1.5)), 0)
     )
    );
   return filteredProducts.AsQueryable();
  }

  private bool IsSimilarParts(string text, string pattern, int maxDistance)
  {
   if(text.Length < pattern.Length)
   {
    string temp = text;
    text = pattern;
    pattern = temp;
   }

   pattern = pattern.ToLower();
   text = text.ToLower();

   int slen = text.Length;
   int plen = pattern.Length;

   if (plen > slen) return false;

   for (int i = 0; i <= slen - plen; i++)
   {
    bool res = IsSimilar(text.Substring(i, plen), pattern, maxDistance);
    if (res == true) return true;
   }

   return false;
  }

  public static bool IsSimilar(string s, string t, int maxDistance)
  {
   int n = s.Length;
```

```
   int m = t.Length;

   if (Math.Abs(n - m) > maxDistance)
   {
    return false;
   }

   int[,] dp = new int[n + 1, m + 1];

   for (int i = 0; i <= n; i++)
   {
    for (int j = 0; j <= m; j++)
    {
     if (i == 0)
     {
      dp[i, j] = j;
     }
     else if (j == 0)
     {
      dp[i, j] = i;
     }
     else
     {
      int cost = s[i - 1] == t[j - 1] ? 0 : 1;
      dp[i, j] = Math.Min(Math.Min(dp[i - 1, j] + 1, dp[i, j - 1] + 1), dp[i -
1, j - 1] + cost);
     }
    }
   }

   return dp[n, m] <= maxDistance;
  }

  public IQueryable<Product> FilterByCategory(IQueryable<Product> products)
  {
   if(Category == null || Category == 0) return products;
   return products.Where(s => s.Category.Id == Category);
  }

  public IQueryable<Product> SortByFilter(IQueryable<Product> products)
  {
   if (!Enum.IsDefined(typeof(OrderTypes), Sort)) return products;

   switch ((OrderTypes)Sort)
   {
    case OrderTypes.newer_one: break;
    case OrderTypes.cheaper: products = sortByLowPrice(products); break;
    case OrderTypes.expensive: products = sortByHightPrice(products); break;
    case OrderTypes.by_name: products = sortByName(products); break;
   }

   return products;
  }

  public List<SelectListItem>
CreateSelectedCategoryFilterList(IQueryable<Category> categories)
  {
   List<SelectListItem> list =  CreateCategoryFilterList(categories);

   if (Category != null)
   {
    SelectListItem sli = list.FirstOrDefault(el => el.Value ==
Category.ToString());
    if (sli != null)
    {
     sli.Selected = true;
```

```
    }
   }

   return list;
  }

  public static List<SelectListItem>
CreateCategoryFilterList(IQueryable<Category> categories)
  {
   List<SelectListItem> list = categories.Select(category => new SelectListItem
   {
    Text = category.Name,
    Value = category.Id.ToString(),
   }).ToList();

   list.Insert(0, new SelectListItem { Text = "Будь-яка", Value = "0" });

   return list;
  }

  public static List<SelectListItem> CreateOrderSelectList()
  {
   return new List<SelectListItem>
   {
    new SelectListItem { Text = "Спочатку нові", Value =
((int)OrderTypes.newer_one).ToString() },
    new SelectListItem { Text = "Спочатку дешевші", Value =
((int)OrderTypes.cheaper).ToString() },
    new SelectListItem { Text = "Спочатку дорожчі", Value =
((int)OrderTypes.expensive).ToString() },
    new SelectListItem { Text = "По назві", Value =
((int)OrderTypes.by_name).ToString() }
   };
  }

  private IQueryable<Product> sortByLowPrice(IQueryable<Product> product)
  {
   return product.OrderBy(el=> el.Price);
  }

  private IQueryable<Product> sortByHightPrice(IQueryable<Product> product)
  {
   return product.OrderByDescending(el => el.Price);
  }

  private IQueryable<Product> sortByName(IQueryable<Product> product)
  {
   return product.OrderBy(el => el.Name);
  }
 }
}
```

Г.8 Програмний код модуля JSImageByteConvertor.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Web;

namespace GameStore.Controllers.Functions.Data
{
 public class JSImageByteConverter
 {
  //prepare img to html view
```

```
  public static string FromBase64ToImageString(byte[] input)
  {
   return "" + Convert.ToBase64String(input, 0,
input.Length);
  }

  //prepare img array to html view
  public static IEnumerable<string> ArrOfBase64ToImageString(IEnumerable<byte[]>
input)
  {
   List<string> output = new List<string>();
   foreach (byte[] byteArr in input)
   {
    output.Add(FromBase64ToImageString(byteArr));
   }

   return output;
  }

  //check img format
  public static bool CheckImgStringToBase64(string input, out byte[] output)
  {
   Regex exp = new Regex(@"^|^");
   if (checkFileFormat(input, exp))
   {
    string base64String = exp.Replace(input, String.Empty);
    return TryGetFromBase64String(base64String, out output);
   }

   output = null;
   return false;
  }

  //check img code format
  public static bool TryGetFromBase64String(string input, out byte[] output)
  {
   output = null;
   try
   {
    output = Convert.FromBase64String(input);
    return true;
   }
   catch (FormatException)
   {
    return false;
   }
  }

  private static bool checkFileFormat(string file, Regex exp)
  {
   return exp.IsMatch(file);
  }
 }
}
```

Г.9 Програмний код модуля Pagination.cs

```
using PagedList.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace GameStore.Controllers.Functions.HtmlSettings
{
```

```
 public class PaginationCustomOption : PagedListRenderOptions
 {
  public static PagedListRenderOptions ShowOnlySomePages(int countOfPages,
string first,  string last,  string prev, string next)
  {
   return new PagedListRenderOptions
   {
    DisplayLinkToFirstPage = PagedListDisplayMode.Always,
    DisplayLinkToLastPage = PagedListDisplayMode.Always,
    DisplayLinkToPreviousPage = PagedListDisplayMode.Always,
    DisplayLinkToNextPage = PagedListDisplayMode.Always,
    MaximumPageNumbersToDisplay = countOfPages,
    LinkToFirstPageFormat = first,
    LinkToPreviousPageFormat = prev,
    LinkToNextPageFormat = next,
    LinkToLastPageFormat = last,
   };

  }
 }
}
```

Г.10 Програмний код модуля AbstractNumberComparsion.cs

```
using System;
using System.ComponentModel.DataAnnotations;

namespace GameStore.Validation
{
 [AttributeUsage(AttributeTargets.Property | AttributeTargets.Field,
AllowMultiple = false)]
 abstract public class AbstractNumberComparison : ValidationAttribute
 {
  private double _limitNumber;

  public AbstractNumberComparison(double limitNumber)
  {
   _limitNumber = limitNumber;
  }

  public override bool IsValid(object value)
  {
   try
   {
    double number = Convert.ToDouble(value);
    return checkNumbers(number, _limitNumber);
   }
   catch
   {
    return false;
   }
  }

  protected abstract bool checkNumbers(double val1, double val2);
 }
}
```

Г.11 Програмний код модуля CustomWidget.js

```
import React, {useState } from 'react';
import { Widget, addResponseMessage, toggleInputDisabled, toggleMsgLoader,
handleSubmit } from 'react-chat-widget';
import ReactDOM from 'react-dom';
import 'react-chat-widget/lib/styles.css';
import './CustomWidget.css';
```

```
const API_KEY = "ng-cWvuN6GLw2IGxflg188fur2XaoAxb"

function CustomWidget({ good, count, price }) {

 const [messages, setMessages] = useState([
  {
   message: `Привіт! Я ваш особистий AI-асистент. Задавайте будь-які питання по
"${good.trim()}"!
   ДОДАТКОВА ІНФОРМАЦІЯ ЯКУ Я ЗНАЮ ПРО ЦЕЙ ТОВАР НА ЦЬОМУ САЙТІ: ${count},
${price}`, sentTime: "just now", sender: "ChatGPT", }, ]);
 const [isTyping, setIsTyping] = useState(false);
 const handleSendRequest = async (message) => {

  if (message.length > 100) {
   addResponseMessage("Важко читати довгі повідомлення...");
   return;
  }

  message = message + `. ВІДПОВІДАЙ УКРАЇНСЬКОЮ І ЯКЩО ТЕКСТ ДО КРАПКИ АБО
ПИТАННЯ НЕ МОЖЕ СТОСУВАТИСЬ "${good}" ТО НАПИШИ ПРО ТЕ, ЩО ЗАПИТАННЯ НЕ
КОРЕКТНЕ, АБО ПРО ТЕ ЩО ТИ ЙОГО НЕ ЗРОЗУМІВ І НЕ ВІДПОВІДАЙ НА НЬОГО`;

  const newMessage = {
   message,
   direction: 'outgoing',
   sender: "user",
  };

  setMessages((prevMessages) => [...prevMessages, newMessage]);
  setIsTyping(true);

  try {
   const response = await processMessageToChatGPT([...messages, newMessage]);
   const content = response.choices[0]?.message?.content;
   if (content) {
    const chatGPTResponse = {
     message: content,
     sender: "ChatGPT",
    };
    addResponseMessage(content);
    setMessages((prevMessages) => [...prevMessages, chatGPTResponse]);
   }
  } catch (error) {
   addResponseMessage("Упс... У мене якісь неполадки...");
   setMessages((prevMessages) => [...prevMessages.slice(0, prevMessages.length -
1)]);
  } finally {
   setIsTyping(false);
  }
 };

 async function processMessageToChatGPT(chatMessages) {
  const apiMessages = chatMessages.map((messageObject) => {
   const role = messageObject.sender === "ChatGPT" ? "assistant" : "user";
   return { role, content: messageObject.message };
  });
  let models = ["mistral-large-2402", "mistral-next", "gpt-3.5-turbo-0125",
"gpt-3.5-turbo-1106", "gpt-3.5-turbo-0613", "gemini-pro", "gemini-pro-vision"]
  for (let i = 0; i < models.length;) {
   try {
    const apiRequestBody = {
     "model": models[i],
     "messages": [
      { role: "system", content: "Покупець магазина техніки" },
      ...apiMessages,
```

```
      ],
     };
     const response = await fetch("https://api.naga.ac/v1/chat/completions", {
      method: "POST",
      headers: {
       "Authorization": "Bearer " + API_KEY,
       "Content-Type": "application/json",
      },
      body: JSON.stringify(apiRequestBody),
     });
     console.log(response.json());
     return response.json();
    } catch (error) {
     if (i == models.length) throw new Error("All Model Deny");
     i++;
    }
   }
  }
  const handleNewUserMessage = (newMessage) => {
   if (!isTyping) {
    handleSendRequest(newMessage);
   }
  };
  React.useEffect(() => {
   toggleInputDisabled(isTyping);
   toggleMsgLoader(isTyping);
  }, [isTyping]);

  React.useEffect(() => {
   addResponseMessage(`Привіт! Я ваш особистий AI-асистент. Задавайте будь-які
питання по "${good.trim()}"!`);
   toggleInputDisabled(isTyping);
   toggleMsgLoader(isTyping);
  }, []);
  return (
   <div className="widget">
    <Widget
     resizable={true}
     handleNewUserMessage={handleNewUserMessage}
     handleSubmit={handleSubmit}
     showBadge = {true}
     title={good}
     subtitle="У вас виникли питання?"
     emojis={true}
     profileAvatar={'../../Content/Img/robot.png'}
    />
   </div>
  );
 }
 (function () {
  const domNode = document.getElementById('chat-widget');
  if (domNode) {
   const name = document.getElementsByClassName('good-name')[0].innerHTML;
   const count = document.getElementsByClassName('count')[0].innerHTML;
   const price = document.getElementsByClassName('good-price')[0].innerHTML;
   ReactDOM.render(
    <CustomWidget
     good={name}
     count={count}
     price={price}
    />,
    domNode
   );
  }
 })()
```

ДОДАТОК Е

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Вебресурс для продажу комп'ютерних товарів

Студента IV курсу
Групи ІПЗ-20-1
Мантура Владислава
Андрійовича

Керівник: ст. викладач Бедратюк Г.І.
ХНУ, 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Кафедра інженерії програмного забезпечення

Актуальність

Сучасні споживачі віддають перевагу швидкості, зручності та можливості вибору, які надає онлайн-шопінг. Це спонукає бізнеси адаптувати свої стратегії до цифрового формату, щоб задовольнити потреби своїх клієнтів і залишатися конкурентоспроможними на ринку.

Актуальність розробки веб-застосунків для електронної комерції обумовлена низкою постійних змін глобальних трендів у споживацьких вподобаннях, що змушує дану сферу постійно розвиватись та адаптуватись.

.

Мета та задачі проектування

Поставленою задачею було створення інтернет-магазину для продажу комп'ютерних товарів, враховуючи сучасні тенденції.

Метою роботи є розробка вебресурсу комерційного характеру, який міститиме сучасні технології та практики для кращого користувацького досвіду.

Для досягнення поставленої мети, поставлено наступні завдання:

- Аналіз предметної області;
- Аналіз особливостей сучасних методик розробки;
- Проектування архітектури вебзастосунку;
- Вибір інструментів і технологій для реалізації;

Особливості

— Для реалізації системи пошуку, щоб дозволити опрацьовувати неточності, використаний алгоритм пошуку відстані Левенштейна з динамічним визначенням допустимої похибки;

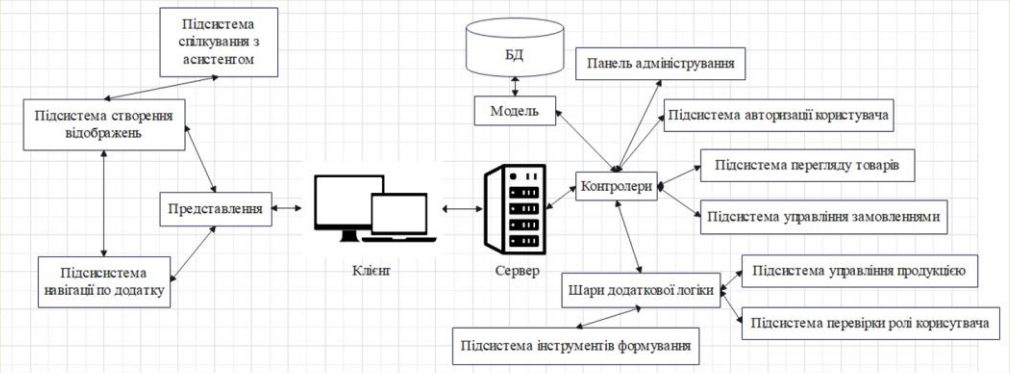
— Надання користувачам можливості обговорення товарів за допомогою вбудованого чату з ШІ, який використовує prompt engineering з підходом до особливостей сучасних мовних моделей;

— Використання динамічних елементів на сторінках веб-ресурсу для кращого UX ;

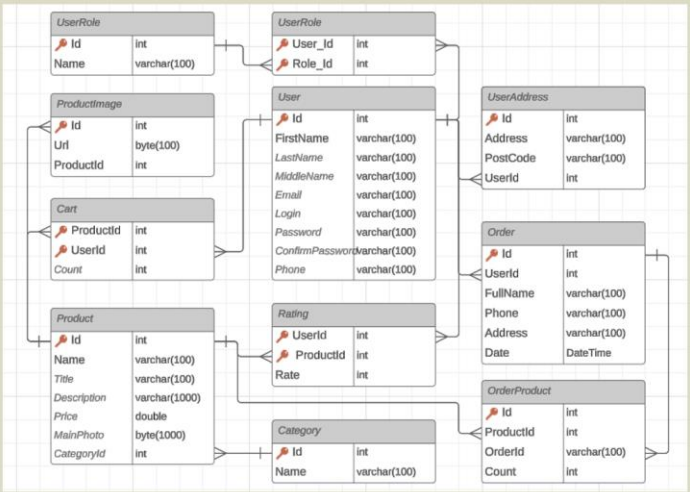
— Зберігання медіафайлів у форматі Base64 у реляційній базі даних.

Компоненти архітектури ресурсу

Для реалізації додатку було використано патерн проектування MVC та клієнт-серверну архітектуру.



Даталогічна модель інформаційної системи



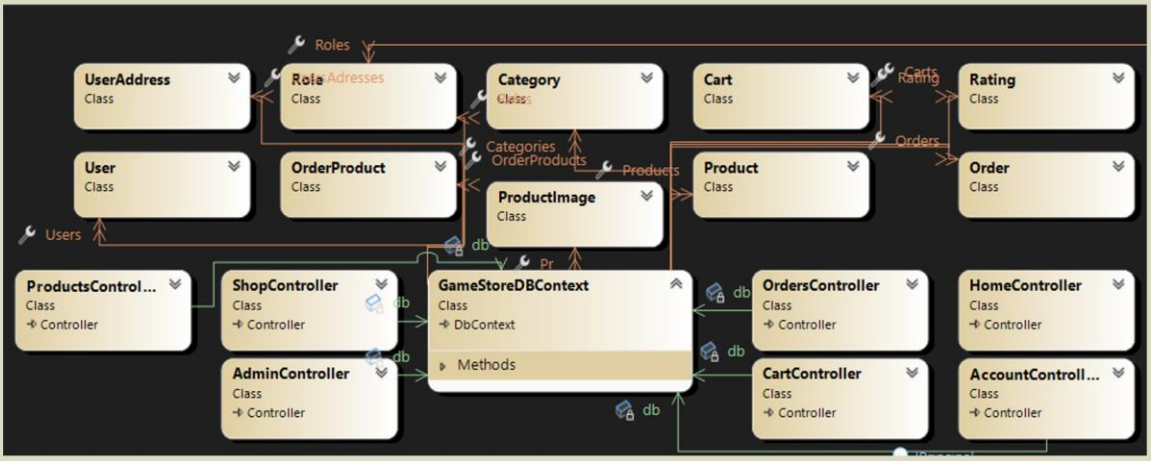
Вибір засобів розробки

Серверна частина:
ASP.NET MVC
Entity Framework
NUnit

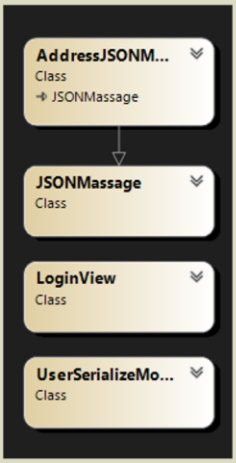
Клієнтська частина:
JQuery
React
Babel
Webpack

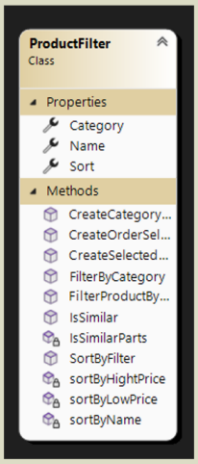
Набори компонентів:
MvcPagedList.Core
Bootstrap
react-chat-elements
react-rating

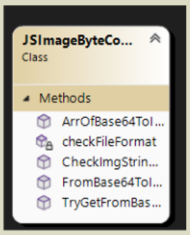
Діаграма класів ресурсу



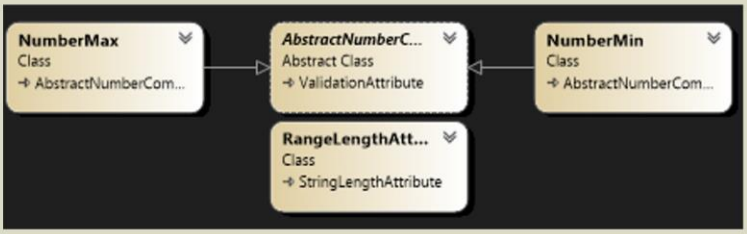
Діаграма класів ресурсу

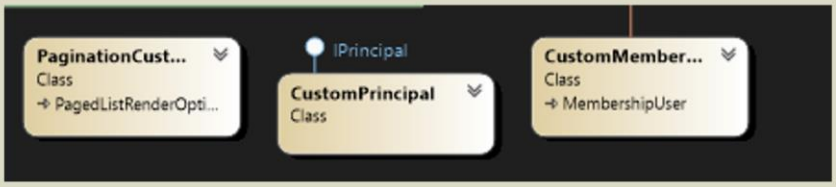






Діаграма класів ресурсу





Діаграма класів ресурсу

Принципи роботи деяких елементів

Алгоритм взаємодії з чат-ботом:

1. Ініціалізується чат, в який передається інформація про стан товару в магазині, з позначкою “system”.
2. Створюється тема, яка описує’ться як спілкування клієнта з асистентом.
3. Коли користувач вводить повідомлення, проходить пошук вільних мовних моделей.
4. При надсиланні повідомлення, до нього долається інформація з інструкціями, написаними у верхньому реєстрі, які повідомляють, як потрібно реагувати на повідомлення.
5. Проходить перевірка на обмеження користувача в кількості токенів, яка рівна 150 за повідомлення.
6. Після отриманні відповіді з серверу, вона надається користувачу.
7. Уся історія чату зберігається, для надання контексту різним мовним системам.

Користувач обмежується 2000 токенами за один сеанс спілкування, що по теперішньому курсі приблизно дорівнює 0.009$

Принципи роботи деяких елементів

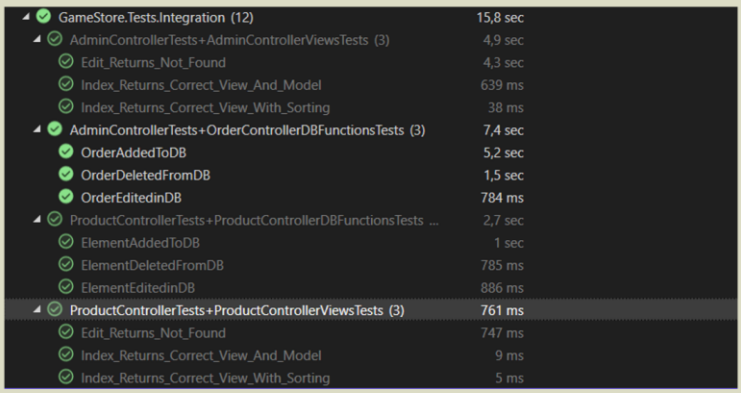
Алгоритм пошуку відстані Левенштейна для пошукової системи:

1. Назва товару, для якого виконується аналіз, розподіляється на рівні частини, які задовільняють розмір патерну.
2. Визначається кількість слів у патерні, який шукається та кожної частини назви товару, який перевіряється.
3. Обирається слово з найбільшою кількістю символів.
4. Визначається кількість допустимих помилок за формулою $Log_e(maxLength^2)$.
5. Складається матриця змін за формулою Левенштейна.
6. Обирається праве нижнє значення матриці.
7. Порівнюється результат з доступною кількістю помилок.
8. Операція виконується для кожного товару, який пройшов попередню фільтрацію.

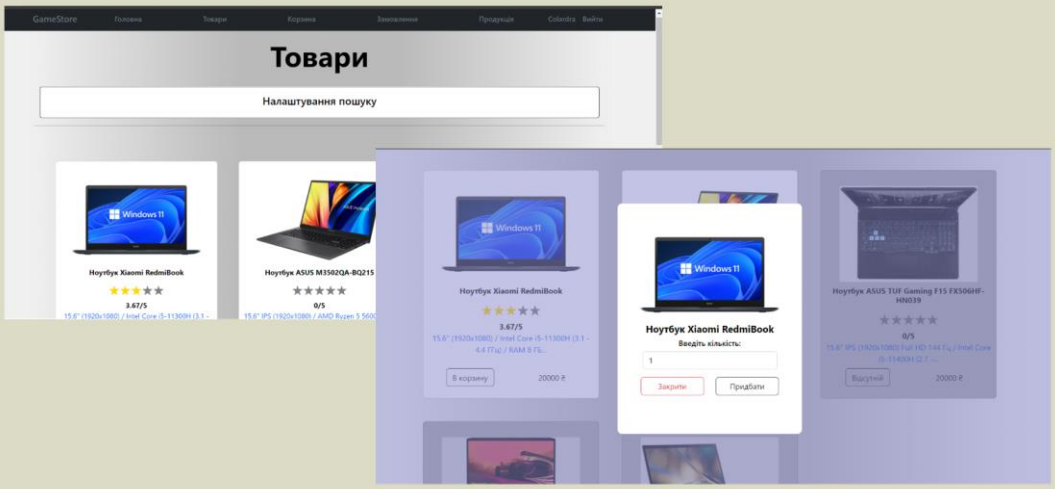
Загальна складність розробленого алгоритму $O(n^2m)$.

Тестування

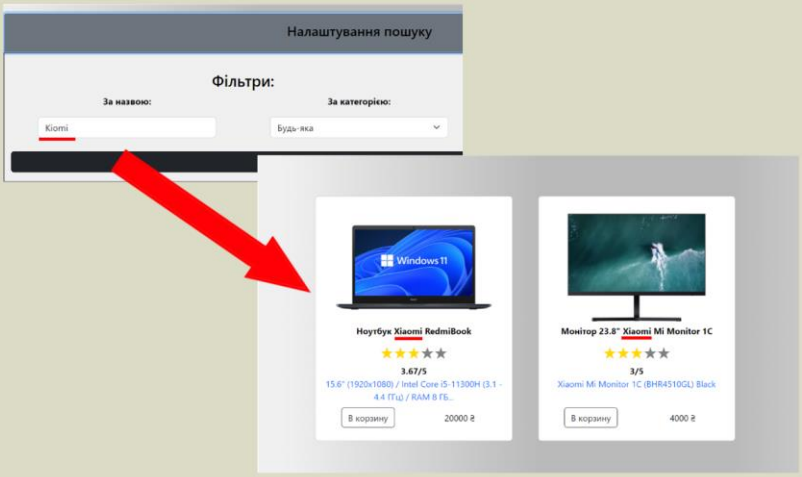
Для тестування використовувалось: Unit-тести, інтеграційне тестування та ручне тестування з Test-Case нотацією.



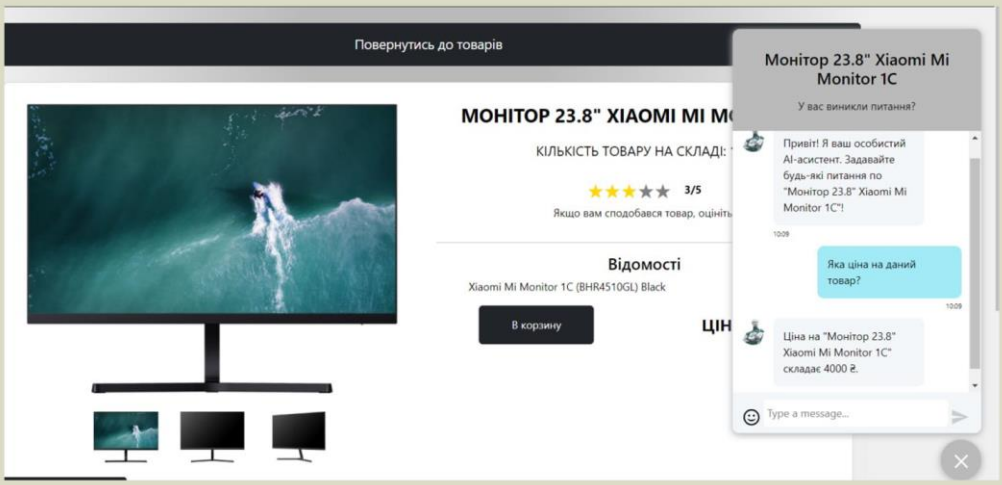
Скриншоти додатку



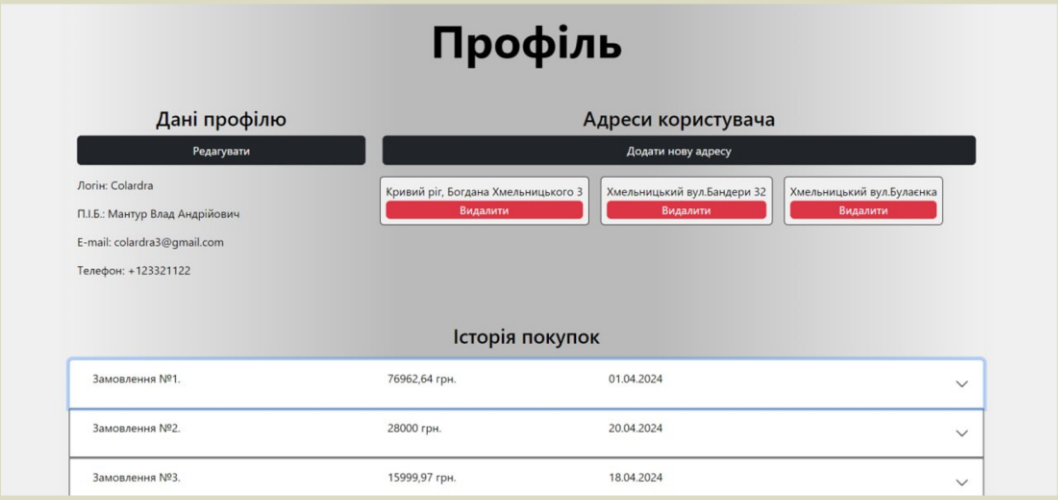
Скриншоти додатку



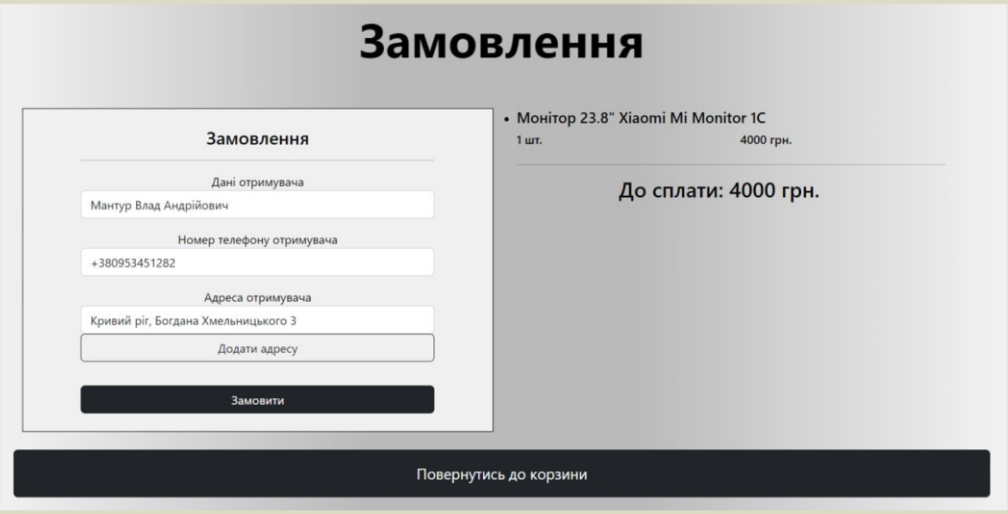
Скриншоти додатку



Скриншоти додатку



Скриншоти додатку



Публікації

Мантур В.А. Використання кодування зображень у форматі base64 для вебзастосунків з використанням реляційних баз даних // Актуальні проблеми комп'ютерних наук АПКН-2023: Збірник наукових праць за матеріалами XV Всеукраїнської наук.-практ. конф., м. Хмельницький, 17-18 листопада 2023 р. Хмельницький, 2023. С. 184–186.

Висновки

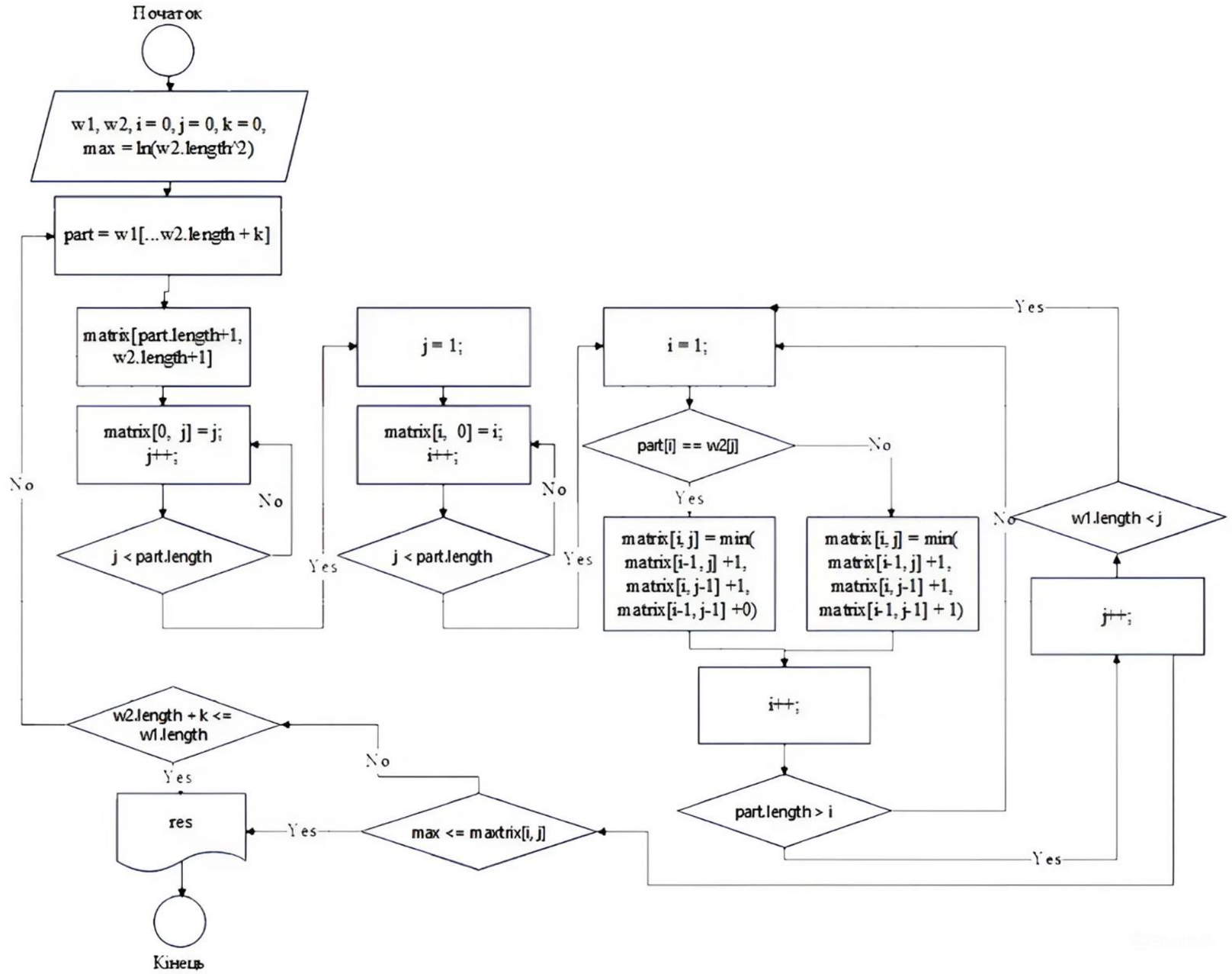
В результаті було отримано веб-застосунок для продажу комп'ютерних товарів, який розроблений з використанням сучасних технологій.

Поставлені задачі, для досягнення мети, було виконано. Для початку було проаналізовану предметну область. Визначені особливості, які застосовуються в сучасній розробці онлайн-магазинів. Було спроектовано архітектуру застосунку на основі клієнт-серверної з використанням патерну MVC. Обрано ряд інструментів для реалізації. Результат розробки протестовано та проаналізовано, що дозволило знайти та виправити помилки.

Даний проект має багато перспектив для подальшого вдосконалення. Наприклад, в майбутньому планується додати систему коментарів, розширення категорій товарів, додавання функцій онлайн-оплати.

Дякую за увагу!

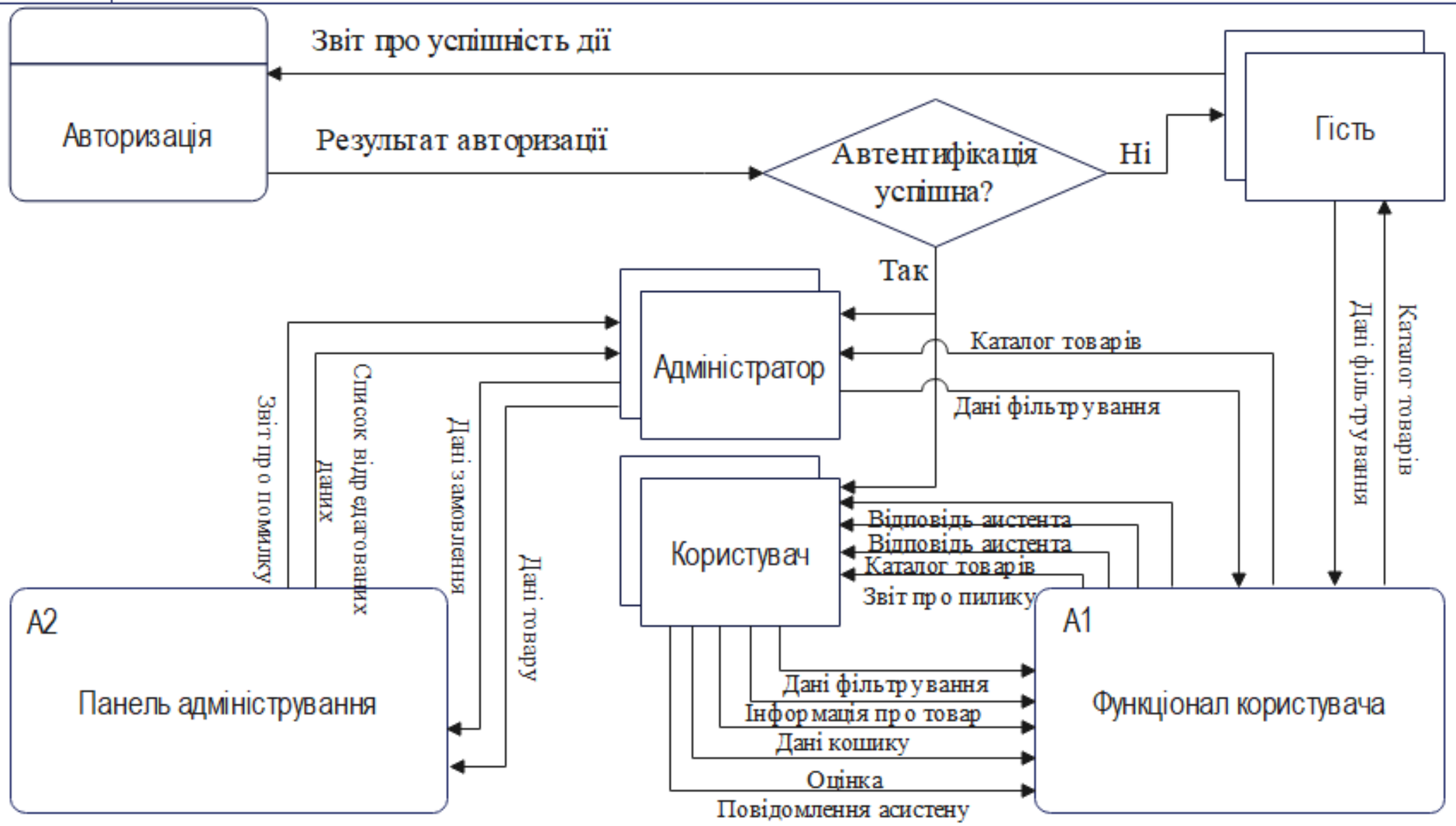
ГРАФІЧНА ЧАСТИНА



					КвРІПЗ.200167.01.14.ПЗ		
						Літера	Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата	Вебресурс для продажу комп'ютерних товарів Відомість документів			
Розробив		Мантур В.А.				Аркуш	Аркушів	
Керівник		Бедратюк Г.І.						
Консульт.					Блок-схема алгоритму Левенштейна для неточного пошуку			
Н. Контр.								
Зав. каф.		Бедратюк Л.П.						

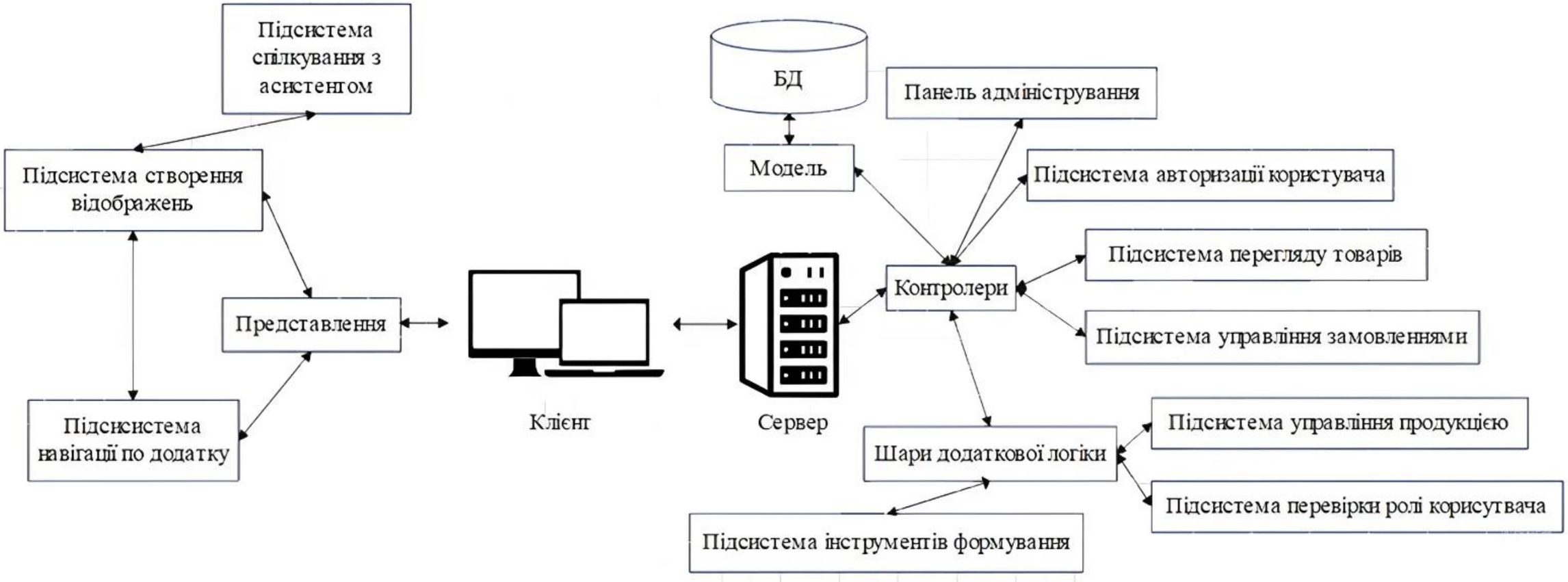
A0

Загальна діаграма потоків даних



Веб-застосунок для продажу комп'ютерних товарів

					КвРІПЗ.200167.01.14.ПЗ	
Зм.	Арк.	№ докум.	Підпис	Дата	Вебресурс для продажу комп’ютерних товарів Відомість документів	Літера / Маса / Масштаб
Розробив		Мантур В.А.				
Керівник		Бедратюк Г.І.				Аркуш / Аркушів
Консульт.						
Н. Контр.					Загальна діаграма потоків даних	
Зав. каф.		Бедратюк Л.П.				



					КвРІПЗ.200167.01.14.ПЗ			
					Вебресурс для продажу комп'ютерних товарів Відомість документів	Літера	Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата				
Розробив		Мантур В.А.						
Керівник		Бедратюк Г.І.				Аркуш	Аркушів	
Консульт.								
					Схема компоновки елементів архітектури			
Н. Контр.								
Зав. каф.		Бедратюк Л.П.						

СУПРОВІДНІ ДОКУМЕНТИ

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти
Мантур В.А.
Прізвище, ініціали

факультет ІТ, 4 курс, група ІПЗ-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності в Хмельницькому національному університеті», згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та/або Anti-Plagiarism) і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

06.06.2024
дата

підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

	Документ		Сумарний збіг по Базі Даних	
ID: 128062 Назва: БКР Вебресурс для продажу комп'ютерних товарів Додано в БД: 2024-06-03 Автора: МАНТУР Владислав Керівники: БЕДРАТЮК Г.І. ст.викладач Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	115569	1022	1794 (2%)	22 (2%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK
by Turnitin

Ім'я користувача:
ІПЗ

Дата перевірки:
03.06.2024 10:34:07 EEST

Дата звіту:
03.06.2024 11:08:46 EEST

ID перевірки:
1016313819

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100012953

Назва документа: БКР_Вебресурс для продажу комп'ютерних товарів_Мантур В.А._Бедратюк Г.І

Кількість сторінок: 2 **Кількість слів:** 267 **Кількість символів:** 4205 **Розмір файлу:** 2.30 MB **ID файлу:** 1016111005

37.8%
Схожість

Найбільша схожість: 29.6% з джерелом з Бібліотеки (ID файлу: 1015083739)

36.7% Джерела з Інтернету 136 ..... Сторінка 4

35.6% Джерела з Бібліотеки 131 ..... Сторінка 4

0% Цитат

Не знайдено жодних цитат

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Мантур Владислав Андрійович

Тема Вебресурс для продажу комп'ютерних товарів

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 78

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі здійснено аналіз предметної області, визначено всі функціональні та нефункціональні вимоги. Проведено порівняльний огляд існуючих програмних продуктів на ринку, проаналізовано їхні переваги та недоліки, що підкреслило необхідність створення нового програмного забезпечення. Обрано інструменти для реалізації розроблених рішень, на основі яких було створено програмне забезпечення. Після завершення розробки, програму протестовано, що і підтвердило її коректну роботу та готовність до впровадження.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі обґрунтовано актуальність теми, визначено мету та завдання дипломного проектування. Перший розділ присвячено аналізу предметної області, розгляду існуючих рішень та визначенню функціональних і нефункціональних вимог до розроблюваного програмного забезпечення. У другому розділі проаналізовано сучасні підходи до покращення користувацького досвіду, архітектури, їхні переваги та недоліки, після чого обрано патерн MVC та модель клієнт-сервер для системи. Також, у даному розділі спроектовані усі компоненти застосунку та зв'язки між ними. Третій розділ включає практичну розробку програмних модулів та опис їхніх особливостей, що призвело до створення програмного продукту. У цьому розділі також проведено три види тестування системи відповідно до функціональних вимог, яке підтвердило коректну роботу вебресурсу.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною, оскільки в Україні вебресурси для продажу ще не повністю відповідають потребам ринку та користувачів. Було застосовано новітні технології для створення програмного продукту, а також використано сучасні архітектурні рішення, що забезпечують високу функціональність і зручність використання.

5. Негативні сторони роботи У роботі реалізовано пошук та додавання товарів лише за обмеженими категоріями, що дуже обмежує кількість можливих товарів. Окрім цього, спілкування з асистентом реалізується лише через клієнтську частину, що може призвести до вразливостей у системі. Тому варто надсилати повідомлення через сервер.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та представлено у вигляді діаграм і рисунків. Пояснювальна записка підготовлена згідно з вимогами чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує на високу оцінку. Матеріал пояснювальної записки структурований, логічний, зрозумілий та лаконічний, що сприяє чіткому розумінню викладеного матеріалу в контексті теми проектування. Графічний матеріал наочно демонструє деталі проектування системи.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре»

РЕЦЕНЗЕНТ (прізвище, ім'я, по-батькові, посада, місце роботи) Висоцька Наталія Олександрівна, доц. кафедри КІС, ХНУ

"6" 06 2024 р. _____
(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів:

Назва кваліфікаційної роботи: «Вебресурс для продажу комп'ютерних товарів»

Автор: Мантур Владислав Андрійович
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Бедратюк Ганна Іванівна
Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	
3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені у роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системою перевірки на плагіат Unicheck виявлено схожість з деякими документами у частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів, у рамках основних написів, у назвах публікацій переліку джерел посилання;

2) запозичення, виявлені в тексті роботи, є фрагментарними.

Максимальний обсяг запозичень, визначений системою Anti-Plagiarism, складає 1.0%. Обсяг запозичень, визначений системою Unicheck виявлення збігів ідентичності/схожості, складає 37.8% і адресується до 136 джерел з Інтернету і 131 джерела з бібліотеки, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 4. 06 2024 р.

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Ганна БЕДРАТЮК