Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Електронний ресурс для продажу ветеринарних товарів
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.2101174.19.18ПЗ

Виконав студент IV курсу група ІПЗ-19-1 _____ М. О. Скоробогатий
Підпис Ініціали, прізвище

Керівник канд. техн. наук, доцент _____ О. М.Яшина
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. техн. наук, доцент _____ Ю. В. Форкун
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

6 червня 2023 р.

Хмельницький 2023

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
01 03 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Скоробогатому Максиму Олександровичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Електронний ресурс для продажу ветеринарних товарів

Керівник проекту (роботи) Яшина Оксана Миколаївна, канд. техн. наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2023 р. № 5
2. Строк подання студентом проекту (роботи) на кафедру 01.06.2023 р.
3. Вихідні дані до проекту (роботи) Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація, тестування програмного забезпечення
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Три креслення у форматі А3:
1.Діаграма варіантів використання
2.Функціональна схема
3.Інфологічна модель

6. Консультанти розділів дипломного проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Форкун Ю. В., доцент кафедри ІПЗ	5.06.2023	5.06.2023
Антиплагіат	Гурман І. В., доцент кафедри ІПЗ	5.06.2023	5.06.2023

7. Дата видачі завдання « 01 » березня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1 Ознайомлення з тематикою кваліфікаційної роботи (КвР), визначення та узгодження індивідуальних тем КвР	01.12 – 30.12.2022	
2 Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2023	
3 Проектування програмного забезпечення	01.02 – 28.02.2023	
4 Програмна реалізація	01.03 – 10.04.2023	
5 Тестування програмного забезпечення	11.04 – 30.04.2023	
6 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог стандартів	01.05 – 25.05.2023	
7 Попередній захист КвР	Травень 2023 (згідно графіка)	
8 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки	26.05 – 1.06.2023	
9 Підготовка до захисту та захист КвР	з 01.06.2023	

Студент _____ М. О. Скоробогатий
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ О.М. Яшина
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Електронний ресурс для продажу ветеринарних товарів».

Автор роботи: Скоробогатий М.О.

Керівник роботи: к.т.н., доцент Яшина Оксана Миколаївна.

Пояснювальна записка: 99 с., 29 рис., 5 табл., 2 дод., 40 джерел.

Графічна частина: Три креслення у форматі А3.

ЕЛЕКТРОННИЙ РЕСУРС, ВЕБ-ДОДАТОК, ВЕБ-РЕСУРС, ІНТЕРНЕТ-МАГАЗИН, ЕЛЕКТРОННА КОМЕРЦІЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, АНАЛІЗ ВИМОГ, ПРЕДМЕТНА ОБЛАСТЬ, ЕЛЕКТРОННИЙ МАГАЗИН, ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

Метою роботи є розробка електронного ресурсу для ведення торгівлі ветеринарними товарами та товарами для домашніх тварин. Дана реалізація дасть можливість його користувачам в автоматичному та ручному режимі обирати, замовляти, а також оплачувати за товари для своїх домашніх улюбленців.

У кваліфікаційній роботі здійснено детальний аналіз предметної області, розроблено дизайн та відповідно до нього інтерфейс, що дружній до користувача та відповідає вимогам проекту. За результатами аналізу вимог та предметної області здійснено програмну реалізацію програмного продукту.

Для розробки електронного ресурсу використано сучасні фреймворки та середовища програмування.

В результаті було здійснено розробку електронного ресурсу, на якому можна купити та продати ветеринарні товари.

01.06.23
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.2101174.01.18.ПЗ	Пояснювальна записка	99		
2	А4		Завдання на КвР	1		
3	А4		Анотація	1		
			Графічні документи	3		
4	А3	КвРІПЗ.2101174.01.18.Е8	Діаграма варіантів	1		
			використання			
5	А3	КвРІПЗ.2101174.01.18.Е8	Функціональна схема	1		
6	А3	КвРІПЗ.2101174.01.18.Е8	Інфологічна модель	1		

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.2101174.01.18.ВД		
Виконав		Скоробогатий		5.06	Електронний ресурс для продажу ветеринарних товарів	Літ.	Арк. 1 / Акрушів 99
Керівник		Яшина О.М.					
Н. Контр.		Форкун Ю.В.		5.06	Відомість документів	ХНУ, ІПЗ-19-1	
Зав. Каф.		Бедратюк Л.П.		7.06			

ЗМІСТ

ВСТУП

В умовах постійної глобалізації відбувається впровадження інтернет-технологій у всі сфери життя, що у свою чергу формує потужну та сучасну інформаційну економіку.

Важливим завданням розвитку країни стає перехід до інформаційної економіки, в якій велика увага приділяється науково-технічному прогресу, масовому поширенню інноваційних технологій та електронного управління [2].

У цих умовах інтернет-технології є потужним поштовхом для розвитку та формування економіки загалом, і зокрема цифрової. У результаті на ринку більшість підприємств намагаються вийти на електронний ринок. Інтернет-торгівлі властива поява власних бізнес-процесів і моделей, без яких неможливе її існування, завдяки інтерактивній взаємодії продавців та покупців стає можливим мати доступ до ринку в будь-якій точці світу та у будь-який час. Це дає можливість задовольнити запити та вимоги покупця на високому рівні, також швидко реагувати на зміни ринку [3].

Світова економіка та її головні тенденції безпосередньо пов'язані з сучасним формуванням інтернет-торгівлі. Ця тенденція обумовлена низкою факторів: забезпечується розширене відтворення на новій матеріальній основі, надається доступ до інформації та знанням, які трансформуються у товарні форми, в умовах розвитку електронної комерції відбувається формування фінансової економіки. Відмінна риса, якою полягає в тому, що максимізація прибутку відбувається без будь-якого виробництва, тобто підвищується важливість фінансового капіталу. Все вищезгадане характеризує той факт, що відбувається перетворення економічних відносин у новітньому постіндустріальному рівні. В наш час інтернет-торгівля набирає величезної популярності.

Насамперед визначимо, що таке інформаційні ресурси. Перш за все інтернет-торгівля - це сфера економіки, яка включає в собі фінансові та торгові

транзакції, що здійснюються у свою чергу за допомогою комп'ютерних мереж, та бізнес-процеси, пов'язані з проведення таких транзакцій. Інтернет став важливим компонентом для ведення успішного бізнесу, у будь-яких організацій чи компаній.

Інтернет – це потужний інструмент для дослідження, розвитку торгівлі та ведення бізнесу, впливаючи на аудиторію дистанційно, при цьому не має жодних територіальних обмежень. На сьогоднішній день понад 95% користувачів використовують Інтернет для пошуку інформації, покупки товарів та послуг, створення власних бізнес-проектів, для навчання, роботи, розваг і т.д. Інтернет-комерція все більше і більше входить у життя великої кількості людей. Вона відкриває нові можливості для ведення бізнесу, і більшість компаній вже перейшли на ведення та розвиток своєї справи в Інтернет-середовищі.

Отже, можна зробити висновок про те, що тема розробки електронного ресурс для продажу ветеринарних товарів є досить актуальною, особливо в сучасних умовах ведення війни, в ході якої страждає велика кількість тварин і потребує різноманітного роду догляду та відповідно якісних ветеринарних товарів.

Мета роботи – розробка електронного ресурсу, за допомогою якого користувачі зможуть у зручному режимі онлайн обирати, купувати, оплачувати та замовляти доставку ветеринарних товарів.

Об’єкт дослідження – процес створення замовлень, що здійснюють користувачі, подальший аналіз та обробка.

Предмет дослідження – відображення даних про вибір ветеринарних товарів користувачами з можливістю здійснення ними замовлень із подальшим їх обліком.

Виходячи із об’єкту, предмету та мети дослідження, можна окреслити такі завдання, що вирішуються для її досягнення:

– здійснення аналізу предметної області із визначенням її головних особливостей;

– проведення аналізу існуючого програмного забезпечення для здійснення торгівлі ветеринарними товарами із детальним аналізом негативних та позитивних особливостей;

– формування технічного завдання (специфікації);

– проектування електронного ресурсу для торгівлі ветеринарними товарами;

– виконання програмної реалізації електронного ресурсу згідно із визначеними вимогами;

– здійснення тестування електронного ресурсу.

В глобальному плані можливий розвиток даного електронного ресурсу, шляхом об’єднання із іншими інтернет-платформами, а також інтеграцію у соціальні мережі, що користуються найбільшим попитом серед користувачів даного сегменту.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Інтернет-магазин - це засіб для подання або реалізації товару, роботи або послуги шляхом вчинення електронного правочину. У свою чергу, електронний правочин - це дія особи, спрямована на придбання, зміну або припинення цивільних прав та обов'язків, вчинена з використанням інформаційно-телекомунікаційних систем (ІТС). За допомогою інтернет-магазину продавець реалізує можливість укладання угод про продаж товарів/робіт/послуг дистанційно через мережу Інтернет.

Зовні схема роботи у всіх інтернет-магазинів приблизно однакова: покупець заходить на сайт, вибирає необхідний товар та оформляє замовлення, після підтвердження замовлення відправляється в службу доставки, клієнт отримує товар і віддає за нього гроші (якщо вони не були перераховані раніше). Після чого починається пост-продажна робота з покупцем (email-маркетинг, реклама та ін.). Але функціонування цієї схеми може бути забезпечене по-різному.

Відвантаження товару. Тут можливо два варіанти: класична схема (коли інтернет-магазин здійснює відвантаження зі свого складу) та дроп-шипінг. Крім того, інтернет-магазини часто перепрофілюються на маркетплейси (основний КВЕД для них - 63.12 «Веб-портали»). Такий сайт – це не зовсім магазин. Це торговий майданчик, на який приходять продавці та продають свій товар. Втім, часто сайт поєднує обидві функції - є інтернет-магазином, і маркет-плейсом одночасно (як, наприклад, Rozetka).

Оплата. Відповідно до ч. 1 ст. 13 Закону № 675 розрахунки у сфері електронної комерції можуть здійснюватись:

1) з використанням платіжних інструментів (інтернет-еквайринг);

2) із використанням електронних грошей;

3) шляхом переказу коштів;

4) готівкою (працівнику магазину або перевізнику післяплатою);

5) в інший спосіб, передбачений законодавством України (наприклад, у кредит).

Доставка покупцю. Інтернет-магазин може доставляти товар як власною, і сторонньою кур'єрською службою. При цьому доставка може здійснюватися адресно покупцю, у точку видачі інтернет-магазину або у відділення стороннього перевізника. Незалежно від того, ким і куди здійснюється доставка, її вартість може бути включена в ціну реалізації товару (тоді інтернет-магазин не виділяє окремо послугу доставки в документах), або виділена окремо.

Якщо інтернет-магазин у продажних документах окремо виділяє вартість доставки (а отже, надає послугу доставки покупцю), він здійснює окремий вид діяльності

Не зважаючи на те, що робота оф-лайн простого магазину схоже до роботи інтернет-магазину, останні сьогодні виходять на ринок та стають досить перспективним та прибутковим способом ведення різного роду бізнесу від дрібного до мега-корпоративного. Оскільки при такому підході здійснюється збільшення прибутку при одночасному зниженні витрат. Це є закономірністю, оскільки будь-який електронний ресурс, в тому числі інтернет-магазин має ряд суттєвих переваг порівняно із звичайним магазином:

– створити інтернет-магазин значно швидше та дешевше у порівнянні із найпростішим маленьким магазинчиком;

– достатньо великий асортимент із одночасним оновленням товарів та послуг (із можливістю залуження чат-ботів, соціальних мереж та платіжних систем);

– вільний доступ до асортименту товару при наявності мережі інтернет із будь-якого місця на планеті;

– конкурентоспроможні ціни за рахунок безпосереднього співробітництва із постачальниками та виробниками без посередників.

Однак, електронні ресурси, зокрема й інтернет-магазини мають свої недоліки. Основними серед яких, можна виділити:

– невизначеність реального існування товару;

– відповідність основних параметрів якості;

– ризик різного роду шахрайств при проведенні грошових транзакцій;

– проблеми із доставкою.

Можна виокремити такі основні вимоги, що висуваються користувачами до будь-якого інтернет- магазину:

– зрозумілий інтерфейс, а також зручна система навігації по магазину;

– зручна система посилань, яка дозволяє оптимальним способом одержати всю повноту необхідної користувачеві інформації;

– зведення до мінімуму дій користувача для вибору, оплати, доставки та здійснення покупки.

Першим та основним етапом під час розробки інтернет-магазину є підготовчий. На цьому етапі здійснюється ретельний та якісний аналіз особливостей ринку, на який планує вихід той чи інший електронний ресурс (інтернет-магазину), також здійснюється вивчення потреб цільової аудиторії (ЦА), проводиться аналіз діяльності конкурентів, а також конкурентоздатності продукції, асортимент товару тощо. На основі результатів та даних, що отримані під час проведення досліджень розробляється концепція електронного ресурсу (інтернет-магазину).

В рамках розроблюваної та спроектованої концепції інтернет-магазину даються відповіді на запитання, що стосуються безпосередньо до механізму роботи майбутнього інтернет-магазину, а саме:

– яким чином буде здійснюватись закупівля товару де він буде зберігатися;

– особливості формування цін на товар;

– оплата замовлень;

– етапи та умови доставки.

Наступним етапом після розробки концепції є проектування програмного продукту. На даному етапі важливо врахувати всі нюанси концепції задля грамотної реалізації ідеї.

Проектування інтернет-магазину – обов'язковий етап розробки успішного проекту. Його називають наукою, оскільки у кожному проекті свої особливості та складності. Жоден серйозний проект неможливо реалізувати грамотно без ретельного проектування, оскільки навіть крута ідея буде реалізована погано, якщо попередньо не підготуватися.

Визначальним та ключовим фактором етапу проектування програмного забезпечення, зокрема й електронного ресурсу (інтернет-магазину) є створення технічного завдання (специфікації) на розробку, в якому детально прописуються підстава розробки, мета, визначається постановка задачі, прописуються функціональні та нефункціональні вимоги, визначається цільова аудиторія, вимоги до візуалу та інтерфейсу користувача, подається загальна структура розроблюваного інтернет-магазину, із врахуванням загальної концепції.

Загалом етап проектування інтернет-магазину має такі етапи: визначення цілей, збір та обробка даних, маркетингове дослідження, проектування інформаційної системи, сценарії поведінки, прототипування, юзабіліті-тестування, оформлення технічного завдання.

Ефективність інтернет-магазину найчастіше визначається вже після запуску проекту, коли підприємець отримує перший прибуток. У такому разі можна здійснити оцінку загальної відвідуваності, конверсію в покупки, конверсію з покупок, окупність, відмови, покинуті кошики, середній чек і т.д.

Перед запуском проекту також можна цінити його прибутковість та ефективність. Щоб оцінити, чи продаватиме інтернет-магазин, необхідно проаналізувати компоненти сайту.

Спершу потрібно перевірити торгову частину: кошик, оплату, доставку. У каталозі повинні бути швидкі кнопки, кошик повинен розташовуватись у правому верхньому кутку, а реєстрація має бути простою та зручною. Необхідно перевірити наявність фільтрації (допомагають швидко знайти потрібний товар і швидше його купити), блоку «схожі товари» або «переглянуте» (впливає на просування ресурсу та орієнтованість клієнта в асортименті), інформацію про знижки та акції. Також необхідно перевірити зручність способів оплати та доставки: що більше варіацій, то краще. Наявність можливості скласти список бажань та оформлення підписки на новини формує лояльність та підвищує ймовірність покупки.

Друга частина – зв'язок із відвідувачами. Комунікація з клієнтам найважливіша умова як для маркетингу, так і для комфортної покупки. Варіантів реалізації зворотного зв'язку безліч, але найдієвіші – онлайн-консультант, FAQ, відгуки, зворотній дзвінок. Також важливо помістити контакти та посилання на соціальні мережі. Ще один спосіб утримання клієнта – оповіщення після покупки, наприклад, пропозиція надіслати квитанцію на електронну пошту.

Неможливо уявити хороший інтернет-магазин без налаштування безпеки та антиспаму, які потрібні як власнику, так і клієнту. Тому необхідно перевіряти налаштування захисних плагінів та фільтрацію трафіку від DDoS-атак. Для цього створюються бекапи, якщо планується регулярне оновлення платформи. Також слід звернути увагу на антиспам.

Тестування дизайну, контенту, функціональності також можна здійснювати ще на етапі проектування. Прототипи сайту допоможуть визначити, наскільки зручно розміщено логотип, контактну інформацію, пошуковий рядок. Зручна головна сторінка та інформаційна картка товару - ключові компоненти інтернет-магазину, що продає. Це досить зручно роботи, якщо працювати за методикою Scrum, коли контент створюється одночасно з розробкою, і є можливість перевірки текстів та фотографій.

Перед запуском сайту також необхідно провести перевірку SEO та аналітики. Технічна оптимізація сайту, така як метатеги, robot.txt, мапа сайту, уникнення дублікатних сторінок та оптимізація зображень, є важливою для підвищення видимості сайту в пошукових системах. Також потрібно підключити інструменти аналітики, щоб мати можливість оцінити ефективність інтернет-магазину на основі статистичних даних після запуску.

Важливо відзначити, що проектування сайту неможливе без попереднього тестування. Перевірка повинна здійснюватись після кожного етапу проектування інформаційної системи магазину. Це включає контроль функціональності, зручності використання та ефективності сайту. Постійний контроль за цими аспектами є важливим для забезпечення якісної роботи сайту.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Для визначення вимог до розроблюваного програмного забезпечення необхідно здійснити аналіз наявних розробок у визначеній предметній області. Далі подано аналіз існуючого програмно-технічного забезпечення в області продажу товарів для домашніх улюбленців (рис.1.1-1.3).

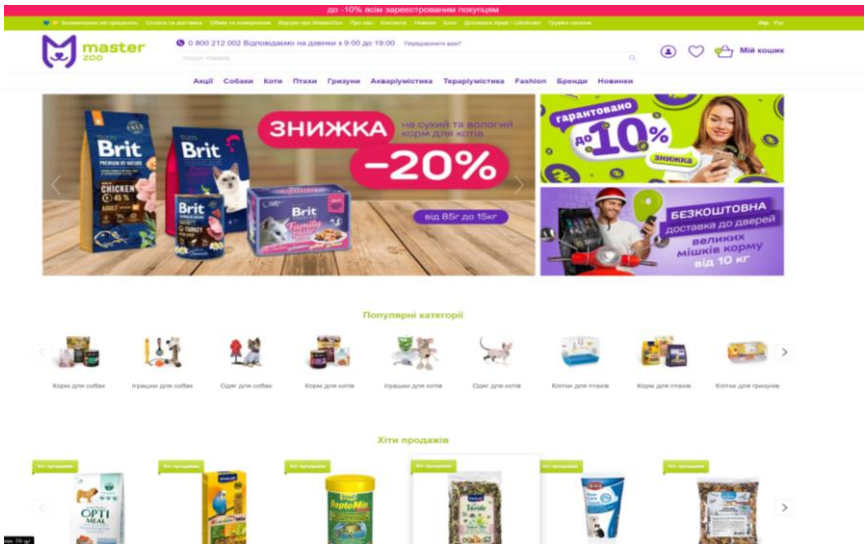


Рисунок 1.1 - Головне вікно інтернет-магазину «Masterzoo» (https://masterzoo.ua/ua/)

Головною перевагою усіх представлених ресурсів є наявність каталогів товарів, де зібрано весь асортимент, пропонованих товарів, представлений в каталозі. У ньому міститься повна, достовірна і доступна інформація, що характеризує пропонований товар, а саме: найменування, фірмове найменування, зображення, опис товару, які викладені в скороченому вигляді для ознайомлення зі змістом товару, складники та інгредієнти, ціна товару та інше.

Також на представлених прикладах можливе оформлення замовлення. Якщо покупця зацікавив якийсь товар, що розміщений на сайті, то є можливість миттєвого оформлення замовлення на придбання товару. Загалом, у переважній більшості процедура замовлення та отримання товару виглядає наступним чином:

– необхідно обрати товар та оформити замовлення, при цьому вказавши свої контактні дані (для незареєстрованих користувачів), зареєстровані користувачі можуть здійснити замовлення через особистий кабінет;

– далі здійснюється вибір необхідної адреси, місця та способу доставки (зазвичай це випадаючий список меню);

– після цього здійснюється підтвердження всієї раніше обраної інформації та відбувається підтвердження замовлення. Інколи ще буває опція зворотного зв’язку із менеджером компанії для підтвердження замовлення.

Також для спрощення процесу оформлення замовлення та його доставки є можливість реєстрації особистого кабінету, що значно спрощує цю процедуру.

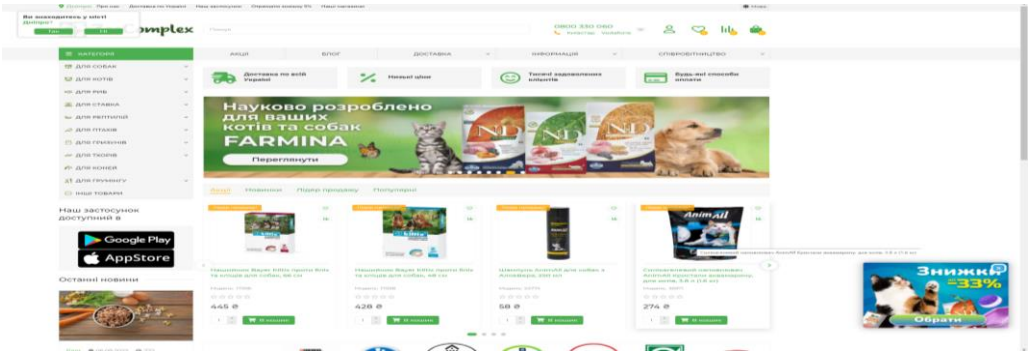


Рисунок 1.2 – Головне вікно інтернет-магазину «zoocomplex.ua» (http://zoocomplex.ua/)

Конфіденційність інформації користувача. Будь-яка інформація, отримана від клієнта, використовується виключно для внутрішніх цілей Інтернет-магазинів Petslike.net, zoocomplex.ua, Master з кінцевою метою поліпшення обслуговування відвідувачів сайту.

Консультація менеджерів. У випадках коли є необхідність у повернені товару чи здійснити консультацію для якогось товару, то є можливість зателефонувати чи написати менеджеру.

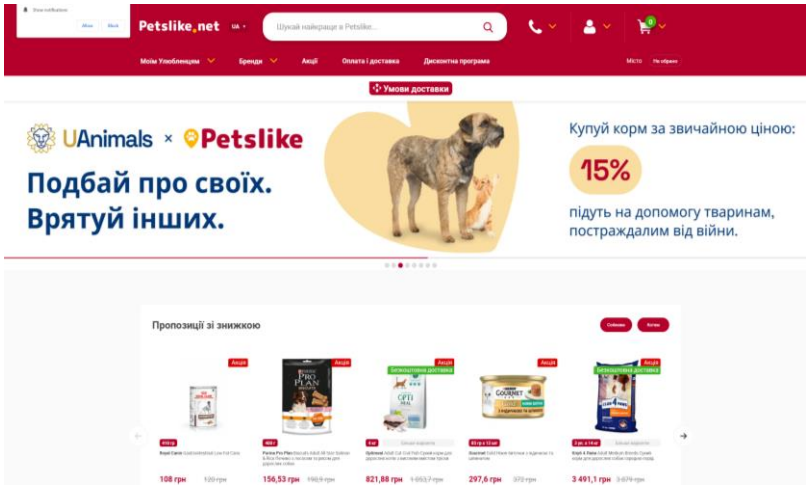


Рисунок 1.3 - Головне вікно інтернет-магазину «petslike.net» (https://petslike.net/)

Отже, після проведеного аналізу можна зробити висновок про те, що обрана предметна область є досить популярною. У даному підрозділі наведено декілька прикладів існуючих інтернет-магазинів, описано їх позитивні сторони та недоліки.

1.3 Визначення вимог до електронного-ресурсу

Для вдалої розробки будь-якого програмного продукту необхідно пройти всі стадії проектування, а також аналізу вимог. На основі здійснення даного аналізу формується технічне завдання, згідно із яким розробляється програмне забезпечення.

Отже, виділяють функціональні та нефункціональні вимоги до розробки програмного забезпечення. Для розроблюваного електронного ресурсу можна виділити такі загальні вимоги:

– інтернет-магазин повинен представляти товари замовника в Інтернеті, знайомити відвідувача з асортиментом, організовувати взаємодія відвідувача сайту із замовником, надавати довідкову інформацію;

– інтернет-магазин має бути доступний в мережі інтернет під доменним ім'ям;

– інтернет-магазин має бути українською мовою або англомовною версією;

– адміністративна частина сайту має бути так само українською мовою;

– інтерфейс електронного ресурсу (інтернет-магазину) повинен мати інтуїтивно зрозумілий вид та навігацію. За основу можна взяти вже звичний вигляд інтернет-магазину з навігацією зліва, пошуком посередині та кошиком праворуч. Але кожен відвідувач незалежно від віку має легко знаходити необхідний йому товар.

– інтернет-магазин не повинен бути завантаженим сайтом, швидко завантажуватись, не навантажувати техніку користувачів та мати швидкий відгук;

– повинен мати зрозумілий та логічний каталог з можливістю сортування товарів за різними параметрами, а також різними способами відображення.

– у каталозі інтернет-магазину у користувача має бути можливість фільтрації товарів за різними критеріями.

– наявність форми зворотного зв'язку;

– наявність опису характеристик товару, що продається, немає чіткого розуміння наскільки деталізована має бути інформація, але її має вистачати для загального образу товару покупцем;

– всі товари необхідно супроводжувати фотографіями з різних ракурсів;

– має бути система оплати.

Графічні елементи навігації мають бути забезпечені альтернативним підписом. Система має забезпечувати навігацію по всіх доступним користувачеві ресурсів та відображати відповідну інформацію.

У інтернет-магазину має бути журнал обліку та статистики.

У має бути передбачений механізм резервного копіювання структури та вмісту бази даних.

В інтернет-магазині має бути система реєстрації та авторизації користувачів, користувачі сайту, залежно від статусу авторизації матимуть різні права доступу. Неавторизовані користувачі будуть мати доступ лише до публічної частини сайту. А зареєстровані та авторизовані користувачі матимуть доступ як до загальнодоступної частини сайту, так і до інформації свого особистого кабінету.

Доступ до особистого кабінету зареєстрованого користувача повинен здійснюватися при використанні логіна та пароля (логін та пароль користувач отримує після реєстрації на сайті).

Зручність навігації. Меню та пункти каталогу мають відображатися таким чином, щоб у користувача не виникало труднощів, як перейти з однієї точки сайту до іншої. Навігація по сайту має бути інтуїтивно зрозумілою та зручною для користувача. Каталог товарів є також частиною навігації сайту. Він також має забезпечувати зрозумілість відображення. Якщо каталог великий, то потрібно ввести кілька можливих шляхів навігації, наприклад, за фірмою чи за моделлю товару. Дуже зручно використовувати пошук за каталогом.

Корзина покупок. Важливо використовувати віртуальний кошик покупок при навігації по каталогу, це дасть можливість відвідувачу відкласти у неї товар у міру перегляду каталогу товарів. Також важливо мати можливість доступу до кошику на етапі перегляду її перед оформленням замовлення для перерахунку або видалення товарів.

Оформлення замовлення. Дуже важливо, щоб кожне замовлення обов'язково супроводжувався видачею ідентифікатора замовлення.

Детальний опис тих дій, які користувачі розроблюваного електронного ресурсу можуть виконувати представлено на рисунку 1.7 та у таблиці 1.

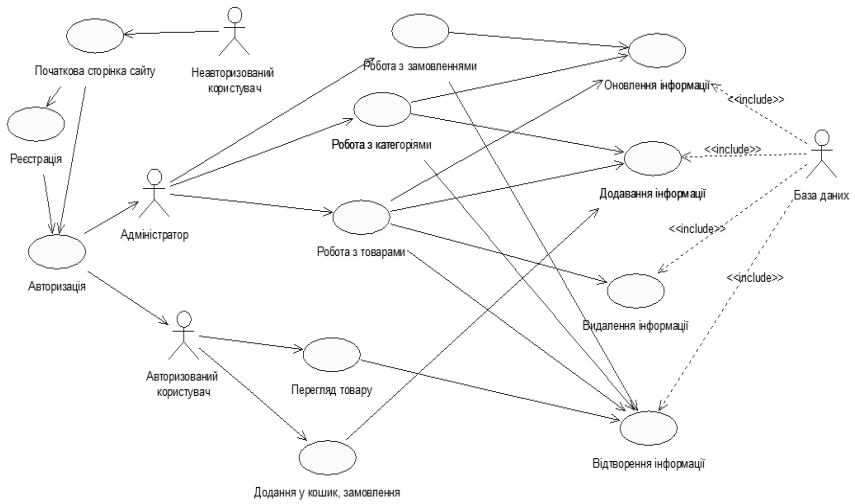


Рисунок 1.7 – Діаграма варіантів використання

Таблиця 1 – Опис дій користувачів електронного ресурсу

Користувачі ресурсу	Опис
Користувач, що не авторизувався	Такий користувач має право переглядати асортимент товару, робити вибір, читати опис продукції, оплачувати та замовляти той чи інший товар, додаючи його у корзину. Однак, для того, щоб оформити замовлення він повинен кожного разу поновлювати свої реквізити та особисті дані. Також він може в подальшому зареєструватись та авторизуватись.
Користувач, що авторизувався	Даний користувач здійснює всі такі самі дії, що й неавторизований, але може зберігати цю інформацію у своєму профілі, налаштовувати особистий кабінет під свої потреби та зберігати персональну інформацію про себе. Це зручно тим, що не потрібно додатково вводити інформацію про себе щоразу, як здійснюється покупка.

Продовження Таблиці 1

Адміністратор (модератор, контент-менеджер)	Адміністратор керує роботою інтернет-магазину, виправляє помилки, якщо такі є, дає відповіді на відгуки за потреби, а також добавляє, зберігає та видаляє їх. Також займається наповненням ресурсу (завантажує описи до товарів та послуг, фото та відео і т.д.).

Отже, вимоги до електронного ресурсу передбачають цілий спектр різного роду роботи для забезпечення адекватної роботи та задоволення потреб кінцевого споживача. Як бачимо на етапі аналізу вимог передбачено, що має бути неавторизований користувач, який має право переглядати асортимент товару, робити вибір, читати опис продукції, оплачувати та замовляти той чи інший товар, додаючи його у корзину; авторизований користувач – може здійснювати ті ж дії, що й неавторизований, але зі збереженням вподобань та інформації про себе; адміністратор, що виконує роль контент-менеджера та модератора.

На основі аналізу предметної області та виділення вимог до програмного продукту здійснюється розробка технічного завдання (підрозділ 1.4.)

1.4 Технічне завдання для розроблюваного електронного сервісу

Вступ

Дане технічне завдання розробляється у відповідності до існуючих стандартів щодо розробки програмного забезпечення для реалізації дипломної роботи на тему: «Електронний ресурс для торгівлі ветеринарних товарів».

1. Підстава для розробки

Підставою для розробки є «Завдання на дипломний проект», що затверджено завідувачем кафедри інженерії програмного забезпечення. Найменування розробки програмного забезпечення: «Електронний ресурс для торгівлі ветеринарними товарами».

2. Перелік термінів, що використовуються

Шаблон - підготовленим макетом графічного представлення сайту.

Сервісний модуль (модуль) - функціонально завершений програмний блок, який призначений для виконання певного практичного завдання. Сервісні модулі (на відміну від сторінок) не мають власного інтерфейсу користувача (на сайті), зате приймають від сторінок введені відвідувачем дані (і/або зчитують дані з бази) і, на основі заданого алгоритму, роблять їх обробку. Результатом роботи модуля може бути як виведення результатів пошуку на сторінку, так і модифікація бази даних або відправка електронної кореспонденції.

Інформаційна панель – візуально об'єднані елементи контенту, навігації або індикації.

Розділ сайту (розділ) – найбільший структурний елемент, що поєднує сторінки сайту за принципом формування логічно закінченої групи інформації.

Підрозділ сайту – структурне об'єднання сторінок сайту всередині одного з його розділів.

3. Загальна інформація про проект

Електронний ресурс для торгівлі ветеринарними товарами розробляється в рамках дипломного проектування бакалаврів за освітньою програмою «Інженерія програмного забезпечення». Метою є реалізація інтернет-магазину, що буде надавати можливість швидкого пошуку, перегляду, вибору та покупки товарів для тварин із можливістю подальшої оплати он-лайн, а також замовлення різного роду доставки (сервіси, кур’єрська тощо). Цільова аудиторія – це власники ветеринарних аптек, власники домашніх тварин, зоокуточки та зоопарки. Тобто фізичні особи, що досягли віку 18 років, а також юридичні особи.

Строк розробки – 3 місяці.

4.Технології та сценарій реалізації системи

Електронний ресурс передбачає наявність мережі Інтернет, повинен підтримувати різнопланову орієнтацію екрану для різноманітних пристроїв та мати лише українську мову інтерфейсу або як варіант ще англомовну версію.

5.Функціональні та нефункціональні вимоги

– Крім штатної аутентифікації (login-password), використовується зовнішня через соціальну мережу Facebook.

– Значення властивостей та характеристик за їх відсутності у сутності не виводяться;

– У всіх областях, де використовується таблично-стрічкове подання товару, застосовується – стандартне каталожне подання товару якщо не вказано інше.

– «Замовити в 1 клік» - функція доступна для формування замовлення без введення даних, що потрібні при оформленні через повну процедуру. Потрібно ввести лише телефон, у системі створюється віртуальний користувач, поточний товар або вміст кошика (залежить від місця розміщення функції) розміщується на замовлення. Якщо користувач авторизований, телефон не запитується, відразу відбувається оформлення замовлення та перехід на Крок №3 оформлення, замовлення зіставляється з поточним користувачем.

– Міста та пункти продажу (видачі товарів) - у системі визначаються два довідники: «міста для вивезення та доставки товарів», «довідник пунктів продажу/видачі товарів» із прив'язкою до міста.

– Форми авторизації/реєстрації/відновлення пароля. Усі спливаючі форми введення виконуються в єдиному стилі, для відновлення пароля використовується EMAIL або PHONE користувача.

До нефункціональних вимог відноситься:

– відправлення квитанції та копії пропозиції на пошту чи месенджер не більше, ніж через 5 хвилин після підтвердження платежу користувачем;

– тривалість відповіді системи на запит.

4. Опис інтерфейсу (екранів додатку)

Дизайн додатку в стилі мінімалізму та не більше 4 кольорів.

Область №1 - в області розміщується:

Верхній банер, якщо банерів даного типу в системі немає, вся область банеру не відображається.

Перемикання мов здійснюється шляхом перекидання користувача на головну сторінку сайту з обраною мовою.

Механізм вибору поточного магазину - область містить вікно, що випадає, що дає можливість вибору магазину: двоетапний вибір міста, потім магазину. Структуризація магазинів по містах здійснюється в адміністратором сайту та забезпечується механізмом обміну із зовнішньою обліковою системою. Можна переглядати розміщення магазину на інтерактивній карті.

Вхід до розділу користувача (форма авторизації, що спливає). Якщо користувач авторизований, замість напису «авторизація» відображається його ПІБ.

Область №2 - в області розміщується:

Верхнє меню містить посилання на загальну інформацію (оплата, доставка, гарантія, контакти). Меню можна змінити за допомогою стандартних системних механізмів.

Логотип компанії. На всіх сторінках, крім головної, під час натискання на кліку переходить на головну сторінку поточної мови.

Система пошуку - у поле введення можна ввести повну або часткову назву товару, після введення 4-го більше символів, система починає пропонувати у вигляді dropbox товари по збігу підрядків поточного введення. З запропонованого списку може здійснюватися перехід на картку товару. Можливий варіант пошуку, при якому в полі введення друкується назва товару + Enter, після чого на окремій сторінці система видає список товарів, що відповідають умові, використовується посторінкова вистава. Опис сторінки «Результати пошуку». Для регулювання області пошуку використовується список розділів першого рівня каталогу. Під рядком пошуку розміщується

підказка з найчастіших пошукових запитів, через кому, при натисканні на пошуковий запит він переноситься в рядок пошуку. Підказки для пошуку змінюються через область, що вмикається.

Область №3 - головне товарне меню

Перший рівень формується за основними розділами каталогу. При виборі розділу, здійснюється розгортання інших підрозділів у меню.

Меню, що випадає, містить розгорнуту структуру обраного розділу, «маркетингову пропозицію» або фоновий малюнок (висновок у такій послідовності). Маркетингових пропозицій одного розділу може бути кілька. При виборі маркетингового пропозиції здійснюється перехід у картку товару чи сторінку з акцією.

Область №4 - область головного банера

Формується із типів банерів. Гортання та перегляд банерів здійснюється на таймері та вручну.

Над банерами розміщується класифікація банерів, що показуються: топ акцій, новинки, спеціальні комплекти. Якщо банерів якогось класу немає, у класифікаторі відповідний тип не показується.

Під головним банером розміщується рядок із маркетинговими відзнаками: умови гарантії, умови знижки тощо. Елементи складаються з картинки, назви маркетингової відмінності та пояснювального тексту. Блок може змінюватися через інструменти «включена область».

Область №5 - «топ продажів»

Стрічка складається з товарів з маркетинговою ознакою «топ-продажів», у стрічку потрапляють товари, які мають пріоритет по полю SORT, а також залишки не нульові. Всі товари відображаються відповідно до «Стандартне подання товару в каталозі» + для кожного товару відображається його розділ. Перегляд товарів у блоці не передбачено. Є можливість переглянути всі товари з ознакою «топ продаж» (посилання «Всі») на окремій сторінці з посторінковою навігацією. Для представлення всіх товарів з ознакою “топ продажів” використовується макет «Стандартна інформаційна сторінка».

Область №6 - «пропозиція тижня»

Блок складається з товарів з маркетинговою ознакою «акція», до блоку потрапляють товари, які мають пріоритет по полю сортування, а також ненульові залишки. Усі товари відображаються відповідно до «Стандартне подання товару в каталозі». Гортання товарів відсутнє. У верхній частині блоку показується лічильник зворотного відліку часу акції товару з найближчою датою завершення.

Інтеграції

Здійснення оновлення та актуалізації інформації за асортиментом товарів потрібно здійснити забезпечення інтегрування із базою даних. Для того, щоб здійснювати відправлення всіх замовлень потрібне забезпечення підтвердження замовлень (електронна пошта, месенджери, чат-боти). Задля цього необхідна інтеграція із електронним сервісом та службами месенджерів. Також для того, щоб клієнти могли оплачувати здійснені покупки, потрібна інтеграція платіжної системи (наприклад, PayPal).

Підводячи підсумки першого розділу, можна зробити висновки про те, що було детально проаналізовано предметну область, виділено межі її використання, здійснено аналіз вимог та розроблено технічне завдання.

1.5 Висновки до 1-го розділу. Постановка задачі

Отже, в результаті написання 1-го розділу було здійснено аналіз предметної області, виявлено функціональні та нефункціональні вимоги.

Для досягнення мети необхідно розв’язати ряд таких задач:

– здійснення аналізу предметної області із визначенням її головних особливостей;

– проведення аналізу існуючого програмного забезпечення для здійснення торгівлі ветеринарними товарами із детальним аналізом негативних та позитивних особливостей;

– формування технічного завдання (специфікації);

– проектування електронного ресурсу для торгівлі ветеринарними товарами;

– виконання програмної реалізації електронного ресурсу згідно із визначеними вимогами;

– здійснення тестування електронного ресурсу.

2 ПРОЕКТУВАННЯ ВЕБ-РЕСУРСУ

2.1 Проектування архітектури та функціональної структури електронного ресурсу

Під час розробки будь-якого програмного забезпечення проектування є важливим етапом, на якому здійснюється визначення архітектури розроблюваного продукту, його функціональна структура, а також макет сторінок.

Для того, щоб працювати із додатками в мережі інтернет в переважній більшості використовується клієнт-серверну архітектуру, на основі якої в сучасних реаліях працює переважна більшість інтернет-ресурсів.

Термін «клієнт-серверна архітектура» це збірне поняття, тому що воно оскільки має в своїй основі компоненти, які доповнюють один одного, а саме клієнта і сервера (рисунок 2.1).

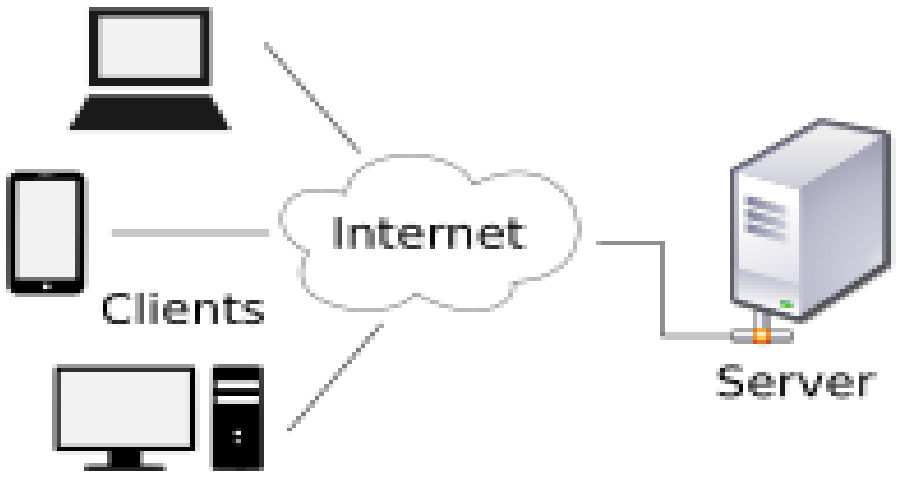


Рисунок 2.1 – Схематичний рисунок архітектури клієнт-сервер

Інтернет забезпечує одночасне користування великої кількості людей додатками та сайтами. Всі ці користувачі можуть надсилати запити до одного комп'ютера, який повинен вміти обробляти ці запити, а також надсилати відповіді на них. Даний підхід називається клієнт-серверною архітектурою, яка

описує, як здійснюється робота із користувачами, а також де здійснюється збереження дані та як забезпечується їх захист.

У клієнт-серверній архітектурі використовуються три компоненти:

Клієнт – програма, яку ми використовуємо в інтернеті. Найчастіше це браузер, але може бути інша окрема програма.

Сервер – це комп'ютер, на якому зберігається сайт або програма. Коли ми заходимо на сайт магазину, звертаємося до сервера, на якому знаходиться сайт.

База даних – це програма, в якій зберігаються всі дані програми. Для магазину це буде база клієнтів, товарів та замовлень.

Особливості клієнта. Клієнт – це завжди програма. Її призначення – дати користувачеві зручний спосіб взаємодії із сервером.

З точки зору сервера, вибрати товар – це надіслати запит спеціальною мовою запитів, наприклад SQL. Але для простого користувача це складно. Тому клієнт пропонує зручний спосіб взаємодії, щоб не писати код своїми руками. Потрібно розуміти, що зайти на сайт – це не просто ввести адресу в браузер. За цією дією ховаються спеціальні запити, відповіді та їхнє розшифрування. Для користувача це зайве, і клієнт у вигляді браузера робить цю роботу за нас.

Особливості сервера. Сервер – потужний комп'ютер, основним завданням якого – безперебійна робота та можливість обробляти мільйони запитів від користувачів.

Сервер дозволяє не дублювати програми. Без них для замовлення продуктів довелося б завантажити весь сайт до себе на комп'ютер, вибрати товари, записати їх та відправити на комп'ютер магазину. Оскільки сайт знаходиться на сервері, тисячі людей можуть звертатися до одного сервера і отримувати від нього потрібну інформацію.

Якщо сервер виконує функції програми та бази даних, то така архітектура називається дворівневою. Такий підхід використовують для невеликих програм, де немає великої кількості клієнтів. Хоч такий спосіб і простіший, але його надійність невелика. Якщо сервер зламають, то зловмисники отримають усі дані.

Для вирішення проблеми безпеки в клієнт-серверній архітектурі використовують базу даних. Вона зберігається окремо від сервера. Сервер у разі виконує роль логічної машини, яка обробляє дані, але з зберігає їх.

Особливості бази даних. У клієнт-серверній архітектурі сервер - це не тільки комп'ютер, на якому знаходиться програма або сайт. Ще це база, де зберігаються всі дані програми. У клієнтів немає прямого доступу до бази даних, оскільки це порушило б їх приватність. Наприклад, зокрема особисту інформацію інших користувачів у соціальних мережах.

Клієнти запитують інформацію на сервері. Якщо сервер вважає, що клієнт має права на отримання інформації, то він її надає. Завдяки цьому ми не можемо користуватися обліковими записами своїх друзів у соціальних мережах або отримувати інформацію про банківські перекази незнайомих нам людей.

За такої схеми роботи архітектура називається трирівневою, оскільки складається з трьох компонентів.

Як і в будь-якого рішення, клієнт-сервер має плюси і мінуси. Далі подано найчастіші випадки. Найчастіше мінуси клієнт-серверної архітектури пов'язані з працездатністю сервера чи бази даних. Розробники можуть вирішувати більшість мінусів, але не все так просто. Рішення одних мінусів призводить до інших, найчастіше зростання вартості розробки та підтримки.

До плюсів даного виду архітектури відноситься:

– Відсутність дублювання. Весь сайт або програма зберігається на одному комп'ютері-сервері. Це дозволяє використовувати його з різних пристроїв, будь то комп'ютер або мобільний телефон.

– Мінімальні вимоги до користувача. Від нього потрібна лише наявність програми-клієнта. Для роботи із сайтами достатньо мати браузер.

– Безпека. Дані зберігаються в базі даних, і користувачі не можуть їх переглядати. Це забезпечує безпеку для персональних даних.

– Продуктивність. Сервери зазвичай продуктивніші, ніж комп'ютери користувачів. Це дозволяє обробляти тисячі запитів від сотні різних користувачів одночасно.

Мінуси:

– Перевантаження сервера. Популярні портали можуть отримувати багато запитів одночасно. Наприклад, при десяти мільйонах запитів на секунду сервер може не витримати та вимкнутись. Цим користуються хакери під час використання DDoS-атаки.

– Вихід з ладу сервера чи бази даних, оскільки це зробить сервіс недоступним для всіх користувачів.

– Висока вартість обладнання. Сервер схожий на простий комп'ютер, але його комплектуючі мають бути розраховані на безперебійну роботу 24/7. Така надійність забезпечується компонентами зі спеціальними функціями, через що вартість обладнання підвищується.

– Витрати на підтримку. Зазвичай недостатньо просто отримати сервер і забути про його існування. Має бути спеціаліст, який обслуговуватиме сервер і швидко реагуватиме у разі поломок.

Щоб позбавитися більшості перерахованих мінусів, розробники використовують кластери серверів.

Далі подано опис розгорнутого сценарію за стандартом RUP функціональної схеми електронного ресурсу для продажу ветеринарних товарів.

1. Зацікавлені особи варіантів використання та їх вимоги: в цьому розділі описується, що має робити програмне забезпечення для усіх наявних та можливих учасників даного процесу.

1) Адміністратор: даний варіант використання здійснює заповнення контенту, а саме на основі наявних даних про товар добавляти та редагувати інформацію про ветеринарні товари, тобто їх опис та фото. Також адміністратор здійснює керування замовленнями, здійснюючи підготовку до відправлення та корегування на етапі проплати та замовлення.

2) Клієнт: даний вид користувача має змогу використовувати без перешкод весь функціонал сайту.

2. Користувач розроблюваного програмного продукту, а саме електронного ресурсу для торгівлі ветеринарними товарами, тобто основний актор цього прецеденту. Даний вид прецеденту можна назвати менеджером, що приймає рішення на основі розроблюваного програмного забезпечення.

Отже, загальна модульна структура розроблюваного програмного забезпечення (електронного ресурсу) подано на рисунку 2.1.

Рисунок 2.1 – Модульна структура електронного ресурсу.

У таблиці 2.1 подано опис основних модулів, з яких складається розроблюваний програмний продукт. До цих модулів відноситься: головний каталог із товарами, а також їх опис; сторінка аутентифікації; інструменти для роботи адміністратора та їх призначення; кошик; модуль замовлення.

Таблиця 2.1 – Призначення програмних модулів

№ з/п	Позначення	Призначення
1	Головний модуль	В даному каталозі розміщується головний модуль, що має головну сторінку. В даній сторінці є каталог товарів
2	Сторінка аунтентифікації	На сторінці аутентифікації міститься модуль який відображає аутентифікацію користувачів. Згідно із технічним завданням це зареєстровані користувачі, а також адміністратора.
3	Інструменти для роботи адміністратора	Даний модуль містить інструментарій для роботи із базами даних
4	Інформація про товар (наявний асортимент, та той, що продався чи очікується його поставка)	Міститься фото з інформацією про товар (фото, опис і т.д.)
5	Кошик для здійснення покупок в електронному ресурсі	Міститься інформація про товари, що вже додані до корзини чи в обробці
6	Модуль замовлення	Здійснюється оформлення замовлення

2.2 Визначення основних модулів

Для визначення ключових модулів, та головних сторінок необхідно здійснити визначення передумов прецеденту (preconditions). Згідно із визначенням передумовами прецеденту є перелік подій, які завжди повинні виконуватися до початку сценарію поточного прецеденту. Разом з тим у виконуваному сценарії прецеденту передумови не здійснюють перевірку, оскільки вони вважаються результатом успішного виконанням деякого іншого прецеденту.

Для цього мають виконуватись певні умови:

— програмне забезпечення повинно бути в активному стані;

— вибрана клієнтом веб-сторінка повинна знаходитись у вільному доступі та відкриватись при виборі клієнта;

— повинна бути можливість добавити обраний товар у кошик, сформувавши відповідним чином замовлення;

— формування замовлення клієнтом: вибір способу оплати та доставки.

До основних модулів також відноситься основний успішний сценарій: даний модуль слугує для опису здійснення успішної операції. Це означає, що дії, які призводять до успішного завершення подій в основному процесі мають свою логіку та завершення, а саме:

— користувач може заходити на головну сторінку сайту;

— після ознайомлення із головною сторінкою є можливість пройти процес аунтентифікації;

— у користувача є можливість ознайомлення із товаром, шляхом вибору відповідних сторінок товару;

— користувач може обирати товар, добавляючи той, що його цікавить у кошик;

— здійснення оформлення замовлення необхідного товару.

Сценарій роботи адміністратора електронного ресурсу:

— у адміністратора є можливість заходити на головну сторінку сайту, а також сторінку адміністрування і здійснювати там редагування та наповнення інформаційного контенту;

— адміністратор має можливість проходити аунтентифікацію та авторизацію;

— може здійснювати використання сторінки інструментів.

Також для успішної роботи електронного ресурсу необхідним аспектом є виконання пост-умов (postconditions). До пост-умов відноситься перелік умов, що постіно повинні виконуватися у випадку успішного виконання основного сценарію, а саме в той момент коли здійснюється задоволення інтересів усіх зацікавлених осіб.

Виконання пост-умов:

— усі здійсненні замовлення були внесені та в подальшому оброблені в базі даних;

— здійснена транзакція була успішно зафіксована в базі даних розроблюваного програмного продукту;

— здійснюється зміна статусу товарів. Дана умова виконується тільки в тому випадку, якщо клієнт купив останню одиницю товару. Тоді відповідним чином це також має відображатись у головному каталозі та на сторінці опису товару.

Також необхідно забезпечити модуль для спеціальних системних вимог. Даний пункт опису має мати перелік нефункціональних вимог, атрибутів, а також якості або певних обмежень в функціонуванні програмного забезпечення, що є пов’язаними із даним прецедентом. Наприклад, це може бути:

— необхідність забезпечення стовідсоткової або повної надійності обробки усіх існуючих транзакцій;

— необхідність забезпечення стовідсоткової або повної надійності особистих даних всіх користувачів;

— достатньої потужності електронного ресурсу для одночасного обслуговування великої кількості користувачів;

— необхідність забезпечення можливості локалізації інтерфейсу користувача програмного забезпечення на інші мови, зокрема англійську (українська мова забезпечується по замовчуванню та має бути в даному ресурсі однозначно).

Також може бути добавлено до потрібних модулів перелік необхідних технологій та додаткових пристроїв. Типовим прикладом можуть бути технічні вимоги, які висувають стейкголдери для технологій введення та виведення даних тощо.

— програмне забезпечення розробляється у вигляді веб-орієнтованої програмної системи;

— здійснено вибір розробки програмного продукту на користь концепції MVC;

— розроблюваний програмний продукт повинен адекватно та якісно відображатись у більшості браузерах мобільних та планшетних пристроїв.

2.3 Проектування логічної моделі бази даних

Мета етапу логічного проектування – перетворення концептуальної моделі на основі обраної моделі даних на логічну модель, не залежну від особливостей використовуваної надалі СУБД для фізичної реалізації бази даних.

Для її досягнення виконуються такі процедури.

1. Вибір моделі даних. Найчастіше вибирається реляційна модель даних у зв'язку з наочністю табличного подання даних та зручності роботи з ними.

2. Визначення набору таблиць виходячи з ER-моделі та їх документування. Для кожної сутності моделі ER створюється таблиця. Ім'я сутності – ім'я таблиці. Здійснюється формування структури таблиць виходячи з

викладених у параграфі 1.4 правил. Встановлюються зв'язки між таблицями за допомогою механізму первинних та зовнішніх ключів. Структури таблиць та встановлені зв'язки між ними документуються.

3. Нормалізація таблиць. Для правильного виконання нормалізації проектувальник повинен глибоко вивчити семантику та особливості використання даних.

4. Перевірка логічної моделі даних щодо можливості виконання всіх транзакцій, передбачених користувачами.

5. Визначення вимог підтримки цілісності даних та їх документування. Ці вимоги є обмеження, які вводяться з метою запобігти поміщенню в базу даних суперечливих даних. На цьому етапі питання цілісності даних висвітлюються безвідносно до конкретних аспектів її реалізації. Повинні бути розглянуті такі типи обмежень:

6. Створення остаточного варіанта логічної моделі даних та обговорення його з користувачами. На цьому етапі готується остаточний варіант ER-моделі, що представляє логічну модель даних. Сама модель та оновлена документація, включаючи словник даних та реляційну схему зв'язку таблиць, представляється для перегляду та аналізу користувачам, які повинні переконатися, що вона точно відображає предметну область.

Щоб вирішити поставлені завдання, необхідно передбачити відповідні таблиці у базі даних.

Перші дві таблиці застосовуються для опису товару який продається в розроблюваному електронному ресурсі.

Таблиця «Товар»:

— код товару (первинний ключ);

— найменування товару;

— постачальник (компанія яка поставляє товар);

— ціна за одиницю товару;

— категорія (зовнішній ключ).

Таблиця «Категорії»:

— код категорії (первинний ключ);

— найменування категорії;

— товари, які належать певній категорії (зовнішній ключ).

Далі подано таблиці, що використовуються для зберігання замовлень користувачів, а також їх подальшої обробки. А саме для можливості оформляти, корегувати, видаляти замовленнями, змінювати інформацію про товари в ролі адміністратора, а також для виведення чеку чи квитанції про замовлення за допомогою даного електронного ресурсу.

Таблиця «Замовлення»:

— Код замовлення;

— Зовнішній ключ;

— Дата та час замовлення;

— Номер телефону;

— Прізвище, ім’я по батькові замовника;

— Поштова адреса замовника.

У випадку, коли замовник захоче змінити якісь свої дії чи внести зміни у замовлення, то використовується таблиця Деталі замовлення, що містить три поля.

Таблиця «Деталі замовлення»:

— Номер первинного ключа;

— Код замовлення (зовнішній ключ);

— Код товару (зовнішній ключ).

Всі інші таблиці використовуються для того, щоб зберігати дані про користувача, а також розмежовувати роль клієнта і роль адміністратора.

Таблиця «Ролі»:

— Код ролі (первинний ключ);

— Найменування ролі.

Таблиця «Замовники»:

— Код замовника (первинний ключ);

— Найменування замовника;

— Поштова адреса;

— Пароль;

— Номер телефону;

— Перелік товарів;

— Поштова адреса (поле булевого типу);

— Номеру телефону (поле булевого типу);

— Булеве поле для двохфакторної авторизації.

Булеві поля використовуються для того, щоб визначити чи підтвердив покупець свої дані: поштову адресу, номер телефону, чи має двохфакторну авторизацію. На рисунку 2.2 зображено інфологічну модель.

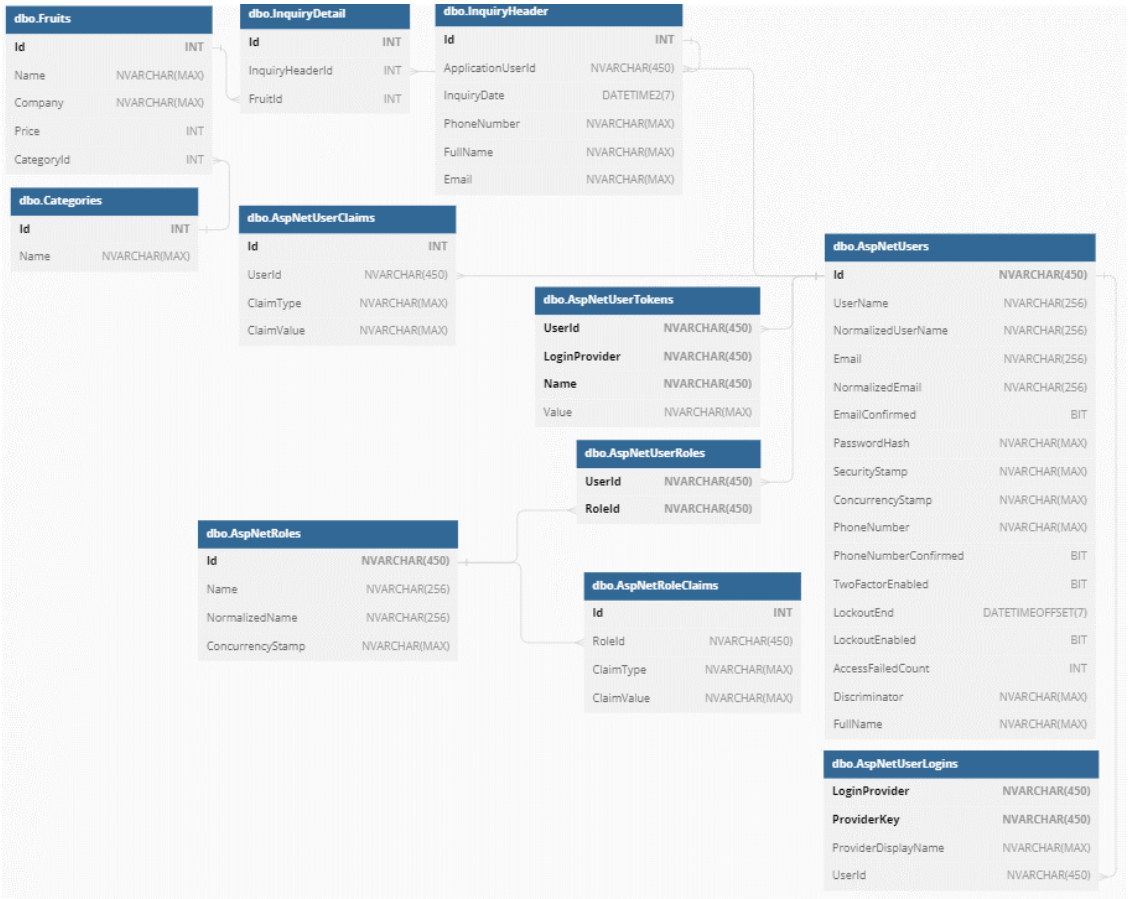


Рисунок 2.2 – Інфологічна модель

2.4 Проектування інтерфейсу користувача

Досить ефективним при розробці проекту електронного ресурсу, зокрема під час його проектування доцільно використовувати прототипуванням (створення макету). Прототипування - це процес, в рамках якого дизайнери створюють, експериментують та втілюють у життя концепцію, починаючи від нотаток на папері та закінчуючи цифровим проектуванням. По суті, прототип - це попередній макет дизайну, який дозволяє користувачам чітко уявити його або взаємодіяти з ним, допоки не буде розроблений кінцевий продукт. Це четвертий етап процесу дизайн-мислення, що супроводжується тестуванням юзабіліті.

Ключовою особливістю прототипів є те, що їх проектують без жодного рядка коду. Існує безліч інструментів прототипування, які допомагають дизайнерам пов'язувати артборди, створювати анімації та інтерактивний інтерфейс без участі розробника, що значно знижує вартість проекту. Розумно тестувати прототип з користувачами та зацікавленими сторонами, щоб усунути всі помилки перш, ніж вкладати гроші, час та трудові ресурси у розробку дизайнерського рішення.

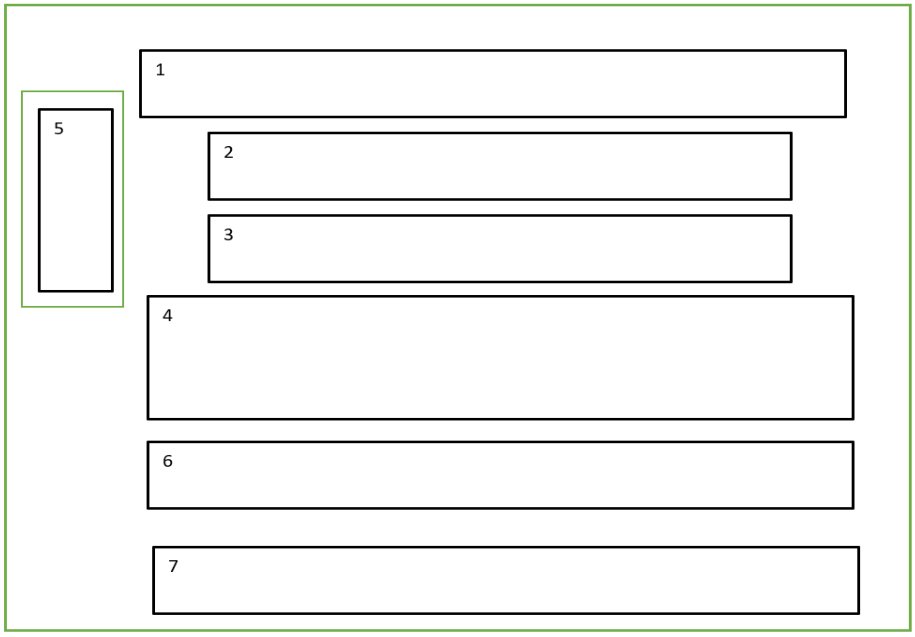


Рисунок 2.4 – Макет головної сторінки

Головний модуль електронного ресурсу має таку будову:

– область шапки або header, на якій розміщується: назва торгової марки та відповідно брендів логотип, телефонні номери для контактів, реєстраційна форма з відповідним посиланням, кнопка із закликом до дії, меню, кошик;

– область навігації navi, у якій знаходиться лінк на головну сторінку разом із посиланням на інші сторінки ресурсу;

– область прокручування;

– функціональна область, де здійснено розміщення функціональних кнопок, для того, щоб можна було фільтрувати асортимент товару, виділяти його та відсортовувати у потрібному порядку. Також в дану область можна також дозволяти чи забороняти виведення інформації відносно того товару, який проданий; виведення інформації відносно загальної кількості продукції на складі, можливість працювати із навігацією на усіх сторінках з вмістимим на електронному ресурсі;

– навігаційна область із розміщеним контентом по сторінках та бічна панель зліва чи справа, що надає можливість обирати розширений вибір параметрів виведення продукції магазину;

– область контенту – надається виведення інформації про ті товари, що є в наявності;

– область футеру – в даній області розміщена інформація про всі необхідні контакти із посиланням на потрібні соціальні мережі, а також довідкові сторінки інтернет-магазину.

Дизайнерські рішення для решти сторінок електронного ресурсу подібні до вигляду головної сторінки та містять відповідну інформацію для довідки. Разом з тим існує певна відмінність, яка полягає в тому, що області під номерами три, чотири, п’ять, шість не розміщені на цих сторінках. Оскільки даний ресурс оформлений та розроблений у стилі мінімалізму, що в даний час знаходиться у трендах, то відповідне значення має й зручність сприймання інформації користувачем, що буде відвідувати даний сайт. Тому, для того, аби користувачу було цікаво та зручно здійснювати навігацію по сайту та

переглядати інформацію було організовано виведення контенту додаткових сторінок із орієнтуванням по центру і з меншою шириною, аніж під час виведення інформації про продукцію на головній сторінці. В особистому кабінеті для користувача, що авторизувався міститься інформація про обраний товар, здійснені та оформлені, а також спосіб доставки та оплати. Дана сторінка із здійсненими замовленнями відповідає структурі головної сторінки.

Як видно, сторінка адміністратора має будову, як показано на макеті рисунок 2.5.

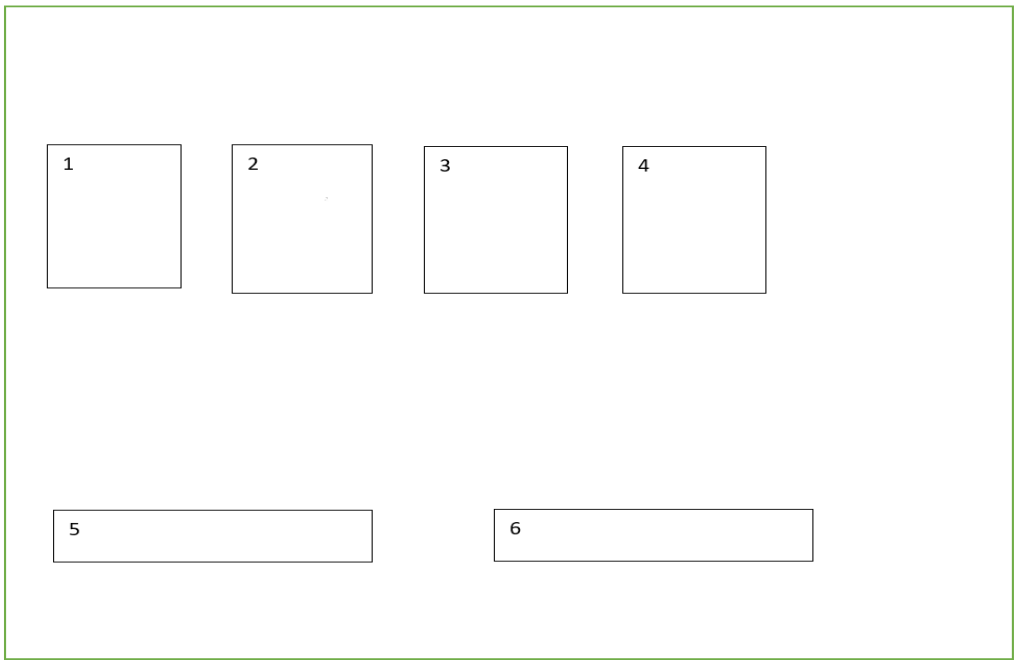


Рисунок 2.5 – Макет сторінки адміністратора

Для зручності користувачів у даному веб-ресурсі навігаційна структура передбачена у частині навігації та дублюється у частині футера. Реалізація навігаційної складової здійснюється на основі вказівникового та графічного типу (за функціональним та візуальним критерієм).

Загалом навігаційна складова досить проста та зрозуміла. Перехід між сторінками здійснюється простим натисканням миші або свайпу у випадку мобільного пристрою.

Службові сторінки (замовлення товару, вибір способу оплати, авторизація і т. д.) організовані за таким же принципом.

Користувач, що має роль адміністратора може працювати за аналогічним алгоритмом, здійснюючи перехід у формах, що призначення для модерування інформації. Також є можливість постійного повернення на головну сторінку електронного ресурсу.

2.5 Аналіз та вибір технологій і методів реалізації веб-ресурсу

Мовою програмування, що була вибрана для реалізації, є C#. Це сучасна об'єктно-орієнтована і типобезпечна мова, яка дозволяє розробникам створювати безпечні та надійні програми, що працюють у середовищі .NET. C# належить до відомої сімейства мов C.

Програми, написані на C#, виконуються у середовищі .NET, яке включає в себе Common Language Runtime (CLR) та набір класових бібліотек. CLR є реалізацією загальномовної інфраструктури мови (CLI), яка є міжнародним стандартом, розробленим компанією Microsoft. CLI є основою для створення середовищ виконання та розробки, в яких різні мови та бібліотеки можуть взаємодіяти безперешкодно.

Крім середовища виконання, .NET також постачається з розширеними класовими бібліотеками. Ці бібліотеки надають широкий набір функціональності для різних робочих завдань. Вони організовані у простори імен, що забезпечують різноманітні корисні можливості, від роботи з файлами і рядками до обробки XML та розробки веб-додатків. Зазвичай програми на C# активно використовують бібліотеки класів .NET для вирішення типових завдань.

Microsoft SQL Server використовує версію SQL як мову запитів, відому як Transact-SQL або T-SQL. T-SQL є реалізацією стандарту SQL-92 з розширеннями, які надає додатковий синтаксис для збережених процедур та підтримує транзакції для взаємодії з базою даних та керуючими додатками.

Microsoft SQL Server та Sybase ASE використовують протокол Tabular Data Stream (TDS) для взаємодії з мережею. Цей протокол також був реалізований в проекті FreeTDS з метою забезпечити можливість взаємодії різних програм з базами даних Microsoft SQL Server та Sybase.

SQL Server також підтримує Open Database Connectivity (ODBC) - інтерфейс взаємодії між додатками та СУБД. Зокрема, версія SQL Server 2014 надає можливість підключатися до бази даних через веб-сервіси, використовуючи протокол SOAP. Це дозволяє програмам, призначеним для Windows, з'єднуватися з SQL Server. Крім того, Microsoft випустила сертифікований драйвер JDBC, який дозволяє програмам на Java, таким як BEA і IBM WebSphere, підключатися до Microsoft SQL Server з версії 2014.

SQL Server має можливості віддзеркалення та кластеризації баз даних. Кластер серверів SQL представляє собою групу однаково налаштованих серверів, яка допомагає розподілити навантаження між ними. Усі сервери в кластері мають спільне віртуальне ім'я, і дані автоматично розподіляються між IP-адресами серверів у кластері під час роботи. У разі відмови або збою на одному з серверів, навантаження автоматично переноситься на інший сервер.

SQL Server 2014 також має вбудовану підтримку .NET Framework. Це означає, що процедури бази даних можуть бути написані мовою .NET, використовуючи повний набір бібліотек, доступних для .NET Framework, включаючи Common Type System. Проте, на відміну від інших процесів, .NET Framework, як базова система для SQL Server 2014, використовує додаткову пам'ять та вбудовані засоби керування SQL Server, замість загальних алгоритмів Windows. Це покращує продуктивність, оскільки алгоритми розподілу ресурсів спеціально налаштовані для використання в SQL Server.

Використання DHTML для динамічної зміни змісту сторінки.

Використання XMLHttpRequest для звернення до сервера «на льоту».

Використання цих двох підходів дозволяє створювати набагато більше зручні WEB-інтерфейси користувача на тих сторінках сайтів, де необхідно активну взаємодію з користувачем. Використання Ajax стало найбільш популярно після того, як компанія Google почала активно використовувати його при створення своїх сайтів, таких як Gmail, Google maps та Google suggest. Створення цих сайтів підтвердило ефективність використання цього підходу.

Для роботи з даними під час розробки використовували технологію Entity Framework, яка працює з поняттям сутності та дозволяє працювати з об’єктами, а не таблицями. Для використання Entity Framework з MySQL ми повинні встановити бібліотеку MySQL.Data.EntityFramework.

Інфраструктура Entity Framework являє собою обгортку, яка зв’язує базу даних з об’єктно-орієнтованою парадигмою, створюючи віртуальну об’єктну базу даних. Вона була створена корпорацією Microsoft для розробників на платформі Net, щоб дозволити їм зберігати дані у застосунках в реляційних базах даних [23].

Основним завданням Entity Framework Core є зберігання об’єктів у базі даних, доступу до них та забезпечення ніби зв’язку між базою даних та застосунком ASP.NET.

Entity Framework використовує реляційну базу даних, у якій дані представляють у вигляді таблиць з рядками. Запити до реляційної бази даних пишуться на мові SQL. Проте, сервери баз даних використовують діалекти мови SQL, які відрізняються одна від одної при звертанні до даних [23].

Таким чином, Entity Framework дозволяє нам перетворити об’єкти у прийнятну форму для зберігання у таблицях баз даних та реалізовувати виконання запитів до цих даних.

Щоб дозволити серверам обробляти різні дані, Entity Framework покладається на версію сервера, яка реалізує запити до бази даних. Щоб отримати об’єкт, .NET Entity Framework Core реалізує оператор SQL, який надсилає запит серверу бази даних для отримання даних, що представляють

об’єкт, поля якого заповнюються полями об’єкта .Net.Для роботи з колекціями Entity Framework Core використовує вбудовану у платформу .Net мову LINQ, яка реалізовує маніпуляції з колекціями об’єктів .Net.

Перевагами використання Entity Framework Core є [23]:

- відкритий sourse-code;
- можливість безпосереднього створення бази даних без прямого написання коду, тобто з наявних сутностей класів Entity Framework самостійно створює базу даних за їх кодом;
- Entity Framework не має прив’язки та не залежить від релізів фреймворку .NET;
- можливість працювати з будь-якою реляційною базою даних та чинним провайдером Entity Framework;
- можливість генерації команд мови SQL із мови LINQ в сутності;
- безпосередня підтримка параметризованих запитів;
- можливість виконувати маніпуляції з командами: вставка, видалення, оновлення;
- підтримка роботи не лише із вбудованими даними, а також із програмованими класами;
- можливість безпечної компіляції завдяки перевірці синтаксичних помилок і безпеки типів;
- підтримка роботу з процедурами [23].

Razor Pages. Сторінки Razor у шаблоні MVC — це перегляд або, іншими словами, типова візуалізація інформації бази даних користувача. Сторінки Razor реалізують і відображають інформацію з моделей як стандартну HTML-розмітку за допомогою стилів (CSS, початкового завантаження). Відображення інформації на сторінці завдяки розмітці та backend-класу з’явилося відносно нещода. Отже, це нововведення схоже на використання WebForms, але без підтримки збереження стану.

Очевидно, що підхід такого рішення (відображення змісту завдяки розмітці та backend-класу), має перевагу в тому, що зникає необхідність у прошарку моделі сторінки (модель даних). Як результат, внутрішня частина реалізації механізму зображення сторінки являє собою як контролер, так і модель. Тобто, підтримує класичну парадигму об’єктно-орієнтованого програмування, а саме: інкапсуляцію даних і методів роботи з ними в одному об’єкті.

Врешті-решт, модель, що відображає сторінку — це просто клас. Тому відсутні будь-які причини, чому цим класом не може бути частина шаблону MVC (model-view-controller), а саме — контролер.

У результаті Razor Pages являє собою краще рішення для написання клієнт-серверних застосувань, які реалізовуються за допомогою паттерна MVC. Тому що технологія Razor Pages створює та використовує традиційне та цілком інтуїтивно зрозуміле поняття “сторінки”, а не прийняті в паттерні MVC (model-view-controller) моделі та контролери, які можуть бути розкидані по всьому проєкту, що ускладнює відладку, тестування та підтримку візуальної частини проєкту.

Однією з характерних особливостей Razor Pages є те, що сторінка представляє собою блочну систему, тобто будь-які дані, що візуалізуються користувачу, реалізовані в окремому файлі з розширенням .cshtml. Таким чином, навігація відтворюваних даних стає зрозумілішою і краще піддається підтримці [23].

Також характерною рисою Razor Pages є те, що код, написаний мовою C#, можна реалізувати безпосередньо у файлі розмітки Razor Pages, тобто у файлі з розширенням .cshtml. Завдяки цій особливості полегшується візуалізація даних, які представлені у вигляді колекцій об'єктів, що обробляються моделями та зберігаються в базі даних.

Для розробки веб-застосунку була обрана платформа ASP.NET з використанням патерну MVC.

Платформа ASP.NET є основною технологією для створення серверних елементів веб-додатків від Microsoft. Вона залишила далеко позаду свою попередницю ASP [21], яка створювалася як мінімальний набір засобів для вставки динамічного вмісту у звичайні веб-сторінки. На відміну від неї, ASP.NET є повнофункціональною платформою, що дозволяє створювати складні та надзвичайно швидкі веб-додатки. Для розробки інтернет-магазину було обрано ASP.NET за такі основні переваги:

– Може створювати веб-додатки, що реалізують шаблон Model-View-Controller (ASP.NET MVC framework);

– Розширюваний набір елементів керування та бібліотек класів прискорюють розробку;

– Краща інтеграція зі службами Microsoft;

– Програми, розроблені за допомогою ASP.NET, можна відкрити в будь-якому браузері на всіх пристроях;

– Можливість підключення та використання великої кількості готових сервісів;

– Код компілюється швидше, і більшість помилок виявляється під час розробки.

ASP.NET MVC базується на вбудованому шаблоні MVC (Model-View-Controller). Це дозволяє контролювати форму вашої веб-програми та взаємодіяти з компонентами, які містяться в програмі. Шаблон MVC ділить веб-додаток на три частини: Модель є основним компонентом шаблону MVC, який представляє поведінку програми незалежно від інтерфейсу користувача. Ця модель забезпечує пряме керування даними, логікою та правилами програми. Подання може бути будь-яким представленням вихідної інформації, наприклад графіком або діаграмою. Кілька представлень однієї інформації можуть співіснувати одночасно. Контролер приймає вхідні дані та перетворює їх на команди для моделі або подання. Розділення сприяє тому, що окремий компонент виконує виключно свої операції. Таким чином, тестування та

відлагодження проекту відбувається почастинно, а не цілком. Як середовище програмування обрано Visual Studio.

2.6 Висновки до 2-го розділу

Отже, у другому розділі здійснено вибір архітектури електронного ресурсу. Даний тип є клієнт-серверною архітектурою. Здійснено проектування логічної моделі бази даних та користувацький інтерфейс. Також здійснено вибір засобів реалізації даного ресурсу. Як середовище програмування обрано Visual Studio із використанням шаблону ASP.NET MVC framework.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ЕЛЕКТРОННОГО РЕСУРСУ

3.1 Реалізація модулів електронного ресурсу

Розроблюваний електронний ресурс містить головну сторінку, на якій відображаються інші модулі за допомогою відповідного меню. Дане меню дозволяє здійснювати перехід по інших сторінках сайту, зокрема для перегляду інформації про усі доступні та наявні на складі товари, а саме інформацію про категорію, ціну, можливість замовлення та доставки. Дана функція реалізована за допомогою циклу обробки колекції, зокрема foreach, що зображено на рисунку 3.1.

```
@model PetsListViewModel

@{
    ViewBag.Title = "Товари";
}

@foreach (var p in @Model.Pets)
{
    @Html.Partial("PetSummary", p)
}

<div class="btn-group pull-right">
    @Html.PageLinks(Model.PagingInfo, x => Url.Action("List",
    new { page = x, category = Model.CurrentCategory }))
</div>
```

Рисунок 3.1 – Приклад циклу обробки колекції foreach

Для здійснення обробки категорій товарів, а також їхнього виведення використовуються контролери. Зокрема, сторінка виведення категорій товарів обробляється контролером PetController, який повертає зовнішній вигляд (view).

Для того, щоб здійснювати вибір товару та реалізувати можливість його покупки, а також бронювання розроблено елемент Корзина, який надає таку

можливість. Даний інструмент вибору реалізовано із використанням контролерів, а саме контролеру CartController, що показано на рисунку 3.2.

```
{
    Ссылок: 13
    public class CartController : Controller
    {
        private IPetRepository repository;
        private IOrderProcessor orderProcessor;

        Ссылок: 6
        public CartController(IPetRepository repo, IOrderProcessor processor)
        {
            repository = repo;
            orderProcessor = processor;
        }
        ссылка: 1
        public ViewResult Index(Cart cart, string returnUrl)
        {
            return View(new CartIndexViewModel
            {
                Cart = cart,
                ReturnUrl = returnUrl
            });
        }
```

Рисунок 3.2 – Реалізація корзини

Під час виконання маніпуляцій із вибором товару необхідно натиснути на кнопку «В корзину». Після цього здійснюється виклик методу AddToCart, реалізація якого подана на рисунку 3.3.

```
public RedirectToRouteResult AddToCart(Cart cart, int petId, string returnUrl)
{
    Pet pet = repository.Pets
        .FirstOrDefault(g => g.PetId == petId);

    if (pet != null)
    {
        cart.AddItem(pet, 1);
    }
    return RedirectToAction("Index", new { returnUrl });
}
```

Рисунок 3.3 – Реалізація вибору товару у корзину

При натисканні кнопки «замовити» відбувається виклик методу ShippingDetails, який у свою чергу викликає постметод CheckOut. Цей метод перенаправляє користувача на форму для введення даних про замовлення товару (рисунок 3.4). Потрібно зазначити, що всі поля є обов’язковими. Це

означає, що користувач обов’язково повинен ввести всі дані, які є необхідними. Якщо цього не відбудеться, то система не зможе здійснити всі дії, що необхідні для здійснення замовлення і відповідно не дозволить користувачеві здійснити цей вибір та оформити замовлення без введення цих даних.

```
@using (Html.BeginForm())
{
    @Html.ValidationSummary();
    <h3>Дані</h3>
    <div class="form-group">
        <label>Ваше ім'я:</label>
        @Html.TextBoxFor(x => x.Name, new { @class = "form-control" })
    </div>

    <h3>Адреса доставки</h3>
    foreach (var property in ViewData.ModelMetadata.Properties)
    {
        if (property.PropertyName != "Name" && property.PropertyName != "GiftWrap")
        {
            <div class="form-group">
                <label>@(property.DisplayName ?? property.PropertyName)</label>
                @Html.TextBox(property.PropertyName, null, new { @class = "form-control" })
            </div>
        }
    }
```

Рисунок 3.4 – Реалізація форми для введення даних про замовлення товару

Під час входу до адміністративної панелі контролер AccountController (рисунок 3.5), викликається метод Login. Цей метод відкриває форму, в якій модератор повинен ввести логін та пароль.

```
<div class="panel">
    <div class="panel-heading">
        <h3>Вхід в систему</h3>
    </div>
    <div class="panel-body">
        <p class="lead">Будь ласка, увійдіть до системи, щоб отримати доступ до адмін. панелі:</p>
        @using (Html.BeginForm())
        {
            @Html.ValidationSummary()
            <div class="form-group">
                <label>Ім'я користувача:</label>
                @Html.TextBoxFor(m => m.UserName, new { @class = "form-control" })
            </div>
            <div class="form-group">
                <label>Пароль:</label>
                @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
            </div>
            <input type="submit" value="Увійти" class="btn btn-primary" />
        }
    </div>
</div>
```

Рисунок 3.5 – Форма для введення паролю

3.2 Реалізація бази даних

Для успішної реалізації електронного ресурсу необхідно також здійснити розробку бази даних. Оскільки було розроблено ER-діаграму, то відповідно до цієї розробки та за допомогою відповідних сучасних засобів та технологій можна описати та здійснити реалізацію бази даних.

Базу даних реалізовано в проекті за допомогою моделі даних які представляють собою клас на мові C# . Проект налічує 6 класів: *ApplicationUser, Category, OrderDetail, OrderHeader, Product, ShoppingCart.* Кожен з цих класів надає дані та реагує на команди контролера, змінюючи свій стан.

Створено клас ApplicationDbContext, який наслідує *DbContext*. В ньому знаходяться таблиці до яких можна робити запити. Посилаючись на цей клас створено міграції, використовуючи команду Add-Migration Initial в Package Manager Console, які додають таблиці в базу даних. Екземпляр DbContext представляє комбінацію шаблонів Unit Of Work і Repository, щоб його можна було використовувати для запитів із бази даних і групування змін, які потім будуть записані назад у сховище як одиниця. *DbContext* концептуально схожий на *ObjectContext*.

DbContext зазвичай використовується з похідним типом, який містить властивості *DbSet<TEntity>* для кореневих сутностей моделі. Ці набори автоматично ініціалізуються під час створення екземпляра похідного класу. Цю поведінку можна змінити, застосувавши атрибут *SuppressDbSetInitializationAttribute* або до всього похідного класу контексту, або до окремих властивостей класу. Модель даних сутності, що підтримує контекст, можна вказати кількома способами. Під час використання підходу Code First властивості *DbSet<TEntity>* у похідному контексті використовуються для побудови моделі за домовленістю. Захищений метод OnModelCreating можна замінити, щоб налаштувати цю модель. Більше контролю над моделлю, яка використовується для підходу Model First, можна отримати, створивши *DbCompiledModel* явно з *DbModelBuilder* і передавши цю модель одному з

конструкторів *DbContext*. При використанні підходу Database First або Model First Model Data Entity можна створити за допомогою Entity Designer (або вручну шляхом створення файлу EDMX), а потім цю модель можна вказати за допомогою рядка з'єднання сутності або об'єкта EntityConnection. Сформовані таблиці проекту продемонстровано на рисунку 3.6.

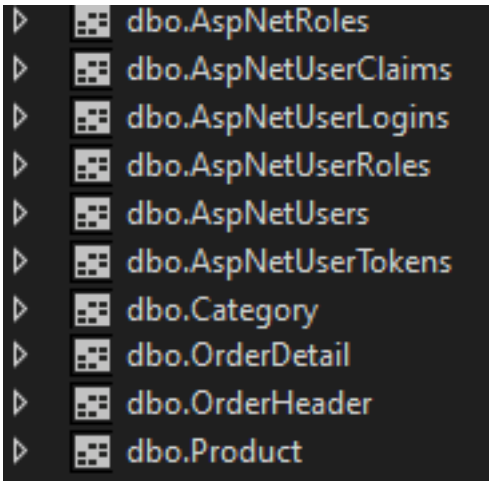


Рисунок 3.6 – Сформовані таблиці проекту.

Після проведених маніпуляцій можна використовувати таблиці в контролері. EntityFrameworkCore автоматично розуміє які таблиці пов'язані і сам створює додаткові таблиці. Наприклад, вище наведені таблиці AspNetUser і AspNetRoles, допоміжною таблицею для них є AspNetUserRoles в якій містяться ключові поля, таблиця створена автоматично.

3.3 Реалізація інтерфейсу користувача

Інтерфейс користувача є комплексом програмних та апаратних засобів, які забезпечують взаємодію між користувачем та комп'ютером або іншим пристроєм. Основою цієї взаємодії є діалоги. У цьому контексті, діалог означає структурований обмін інформацією між людиною та комп'ютером, що відбувається в режимі реального часу з метою досягнення спільного рішення конкретної задачі. Кожен діалог складається з окремих процесів введення та

виведення, які фізично забезпечують зв'язок між користувачем та пристроєм. Обмін інформацією здійснюється шляхом передачі повідомлень, на які, в окремих випадках, потрібно дати відповідь. Далі наведено опис реалізації основних компонентів користувацького інтерфейсу даного ресурсу.

Рисунок 3.7 – Головна сторінка

Для реєстрації на цьому ресурсі було створено спеціальну форму, яка відкривається над усіма вікнами. Форма реалізована у мінімалістичному стилі з використанням чорно-білих відтінків, що не негативно впливає на зір. Детальний вигляд цієї форми можна побачити на рисунку 3.8.

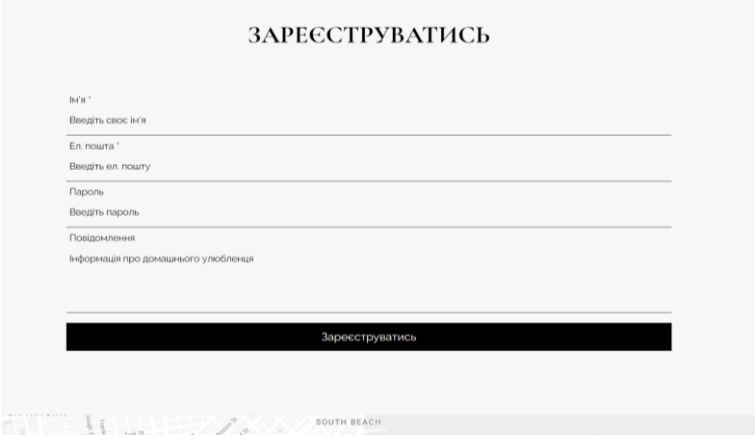


Рисунок 3.8 – Реєстраційна форма

Елемент «Контейнер товару» є важливою складовою цього веб-ресурсу, оскільки він дозволяє користувачам переходити до процесу покупки. Рисунок відображає зображення цього елементу.

Динамічна сторінку перегляду списку товарів з пагінацією. Тут відображаються товари згідно обраної категорії.

Рисунок 3.17 – Контейнер товару

Користування електронним ресурсом дуже просте і не вимагає складних дій або спеціальних пояснень. При відвідуванні сайту користувач автоматично потрапляє на головну сторінку, на якій доступний весь інструментарій для користувача. На головній сторінці ви можете переглянути першу сторінку каталогу продукції, скористатися пошуком та додати товари до кошика.

Перше поле містить назву продукту, що дозволяє вам ознайомитися з коротким описом товару.

У другому полі є зменшене зображення товару з фронтального ракурсу. Біля зображення розташована інформація про ціну та кнопка «Додати до кошика».

Всі ці функції доступні на головній сторінці, яка відображається при відвідуванні сайту.

Якщо кількість товарів не вміщається на першій сторінці каталогу товарів, користувач може переміщатися сторінками каталогу за допомогою покажчиків у вигляді цифр з номерами сторінок.

Кошик користувача є невід'ємною частиною будь-якого інтернет-магазину. Користувач може додавати в кошик товар, що його цікавить і видаляти. Видалення товару відбувається за натисканням на кнопку з відповідним надписом. Кошик користувача зображено рисунку 3.9.

Рисунок 3.9 – Вигляд вікна «Кошик»

Будь-які дії користувача над кошиком відбуваються як у фоновому режимі, тобто сторінка не оновлюється повністю. Оновлюється лише вміст кошика. Також користувач може переглянути загальну вартість всіх товарів та перейти на сторінку оформлення замовлення, що зображена на рисунку 3.10.

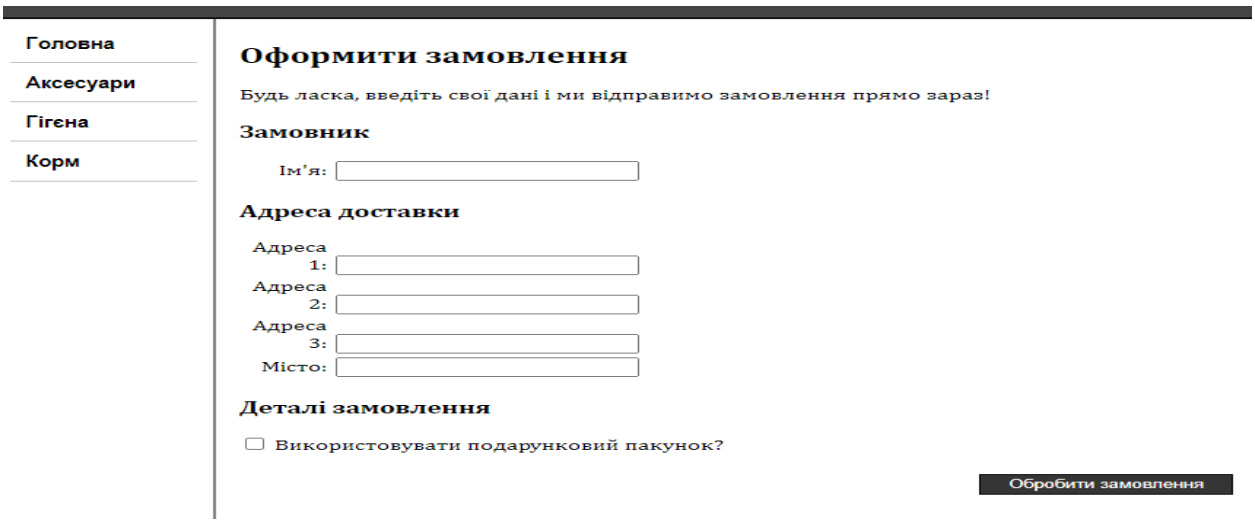


Рисунок 3.10 – Вигляд форми для здійснення замовлення товару

Для адміністрування даного електронного ресурсу існує відповідний розділ, який доступний лише адміністраторам сайту. У цьому розділі адміністратор може додавати, видаляти, а також здійснювати зміни щодо інформації про товар, що показано на рисунку 3.11.

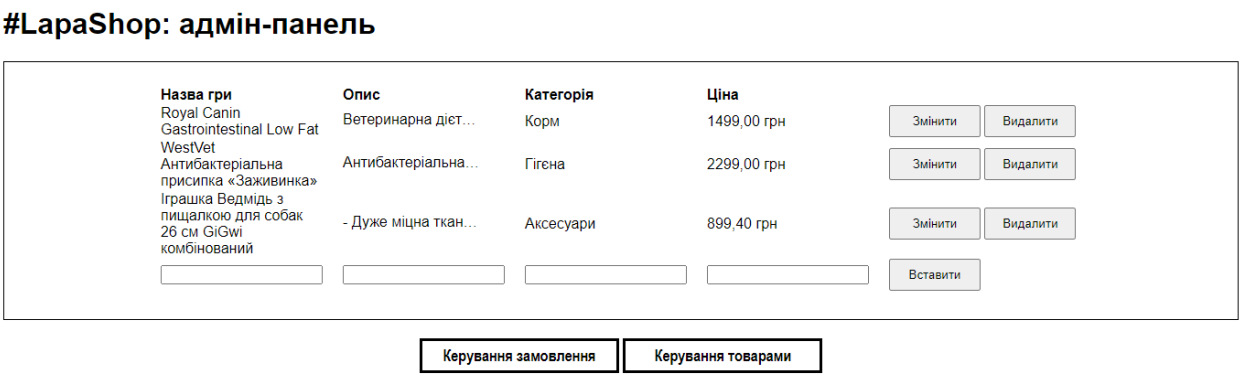


Рисунок 3.11 – Вигляд сторінки адміністрування

Отже, як видно даний електронний ресурс досить простий та зручний у використанні. Містить всю необхідну інформацію про асортимент та наявність товару, має можливість здійснення змовлення та офрмлення покупки із вибором місця доставки та способу оплати. Також слід відмітити інтерфейс, який є дружнім до користувача, не переобтяжений зайвими елементами та виконаний у стилі мінімалізму.

3.4 Аналіз методів тестування електронного ресурсу

Для того, щоб випустити дійсно якісний та адекватно працюючий програмний продукт його необхідно протестувати. Тестування відіграє життєво важливу роль в процесі розробки і створення якісного програмного забезпечення. Необхідно серйозно ставитися до аналізу і проєктування структурованого процесу, який забезпечить своєчасний і успішний випуск проєкту [20, 23, 24].

Тестування програмного забезпечення – це:

– процес дослідження з метою отримання інформації про якість продукту;

– процес перевірки відповідності заявлених до продукту вимог та реально реалізованої функціональності, що здійснюється шляхом спостереження за його роботою у штучно створених ситуаціях та на обмеженому наборі тестів, обраних певним чином;

– оцінка системи для того, щоб знайти відмінності між тим, якою система має бути і якою вона є.

У широкому сенсі, тестування - це одна з технік контролю якості (Quality Control), яка включає планування, складання тестів, безпосередньо виконання тестування та аналіз отриманих результатів.

Важливо розуміти, що тестування ПЗ включає не тільки проведення тестів, але й багато інших дій, пов'язаних з процесом забезпечення якості [20]:

– аналіз та планування;

– розробку тестових сценаріїв;

– оцінку критеріїв закінчення тестування;

– написання звітів;

– рецензування документації (у тому числі й вихідного коду);

– проведення статичного аналізу.

Більшість тестів працює на верхніх рівнях і не дозволяє відловлювати усі баги. Ті ж інтеграційні випробування перевіряють, як взаємодіють між собою різні системи.

На відміну від них, юніт-тести потрібні у таких випадках:

– якщо код незрозумілий - виникли питання щодо його роботи;

– якщо код часто змінюється – щоб не перевіряти його вручну;

якщо оновлення однієї частини коду можуть зламати щось в іншій частині [20, 23].

Модульне тестування [4, 18, 19] (Unit Testing) – це тип тестування програмного забезпечення, у якому тестуються окремі модулі чи компоненти

програмного забезпечення. Його ціль полягає в тому, щоб перевірити, що кожна одиниця програмного коду працює належним чином. Цей вид тестування виконується розробниками на етапі кодування програми. Модульні тести ізолюють частину коду та перевіряють його працездатність. Одиницею для вимірювання може бути окрема функція, метод, процедура, модуль чи об'єкт.

У моделях розробки SDLC, STLC, V Model модульне тестування – це перший рівень тестування, який виконується перед інтеграційним тестуванням. Модульне тестування – це метод тестування WhiteBox [20, 23], який зазвичай виконується розробником.

Модульне тестування засобу розбиття на сторінки можна провести, створивши імітоване сховище, впровадивши його в конструктор класу PetController та викликавши метод List(), щоб запросити конкретну сторінку.

Етапи тестування веб-проєктів [20, 21, 22, 23, 24]:

Підготовчий етап та вивчення документації.

В даний етап входить аналіз технічного завдання; вивчення кінцевих макетів; тест кейсів; матриці відповідності (для валідації покриття вимог щодо продукту тестами) і складання плану тестування.

Тестування верстки.

Візуальна частина:

– невірне відображення блоків, складових інтерфейсу, нестиковки колірної гами;

– тестування локалізованих версій (переклад сайту);

– відповідність макету (шари у PhotoShop);

– у разі зменшення/збільшення масштабів (75-150%) без візуальних недоліків;

– підсвічування полів з помилками;

– перевірка у дозволах (+ прокрутка);

Перевірити можна так: FirefoxMenu -> Інструменти -> Веб-розробник -> Адаптивний дизайн або Resolution Test Plugin у Chrome.

Доступність і відсутність JS помилок:

– чи натискаються клікабельні елементи (внутрішні/зовнішні посилання, посилання на електронну пошту, кнопки, іконки);

– під час наведення на клікабельні елементи курсор змінюється, в іншому випадку – ні;

– підказки на незрозумілих клікабельних елементах;

– під час відключення зображень повинні бути підписи невеликим сірим кольором (у Web Developer -> Images -> Replace Image With Alt Attributes);

– працездатність при вимкненому JS. Критичні функції повинні бути доступні без JS (в Web Developer -> Disable -> Disable JS -> All JS)

Коректна робота, надійна верстка:

– перевірка роботи з даними (введення великої та малої кількості тексту у форму; блоки з контентом міняються місцями (Firebug (HTML -> Edit)));

– перевірка роботи стилів (введення тексту з заголовками, з абзацом і без, з картинками).

404-і запити:

чи є 404-і помилки (Firefox -> Tools -> Validate links)

Функціональне тестування.

Вид тестування, у якому виявляється некоректна/неправильна робота функціоналу програми.

Необхідними перевірками є:

– коректність роботи головних функцій сайту;

– перехід за посиланнями;

– перевірка користувальницьких форм (валідація полів, обов’язкові/необов’язкові поля, повідомлення про помилки під час неправильного введення, додавання коментарів у блог, зворотний зв’язок);

– пошук і покупка товару, оформлення замовлення;

– звірка переданого замовником контенту з наявним на сайті;

– перевірка можливої авторизації/реєстрації;

– додавання, видалення та редагування даних користувачів, товарів і замовлень.

Ad-hock тестування – імпровізаційне тестування без підготовки.

Допомагає зрозуміти:

– чи зрозумілим є призначення форм;

– чи відзначені обов'язкові поля та чи всі обов'язкові поля відмічені;

– чи вбудована обов'язкова перевірка заповнених форм;

– чи відбувається перевірка правильності введення контактних даних.

– З переваг даного тестування можна виділити:

– достатньо швидке знайомство з системою;

– специфічні несправності;

– масу питань і пропозицій;

– економію часу.

Негативне тестування, яке зазвичай називають тестуванням шляху помилок або тестуванням на збій, це процес застосування якомога більшої кількості креативних підходів та перевірки програми на предмет наявності невірних даних. Його призначення полягає в тому, щоб перевірити, чи показуються помилки користувачеві, де вони можуть бути, або більш витончено обробляти неправильні значення. Проводиться для забезпечення стабільності додатків [20, 21, 22, 23, 24].

Еквівалентні тести – це тести, які призводять до одного і того ж результату. Група тестів є класом еквівалентності, за умови дотримання наступних умов:

– всі тести написані для виявлення однієї помилки;

– якщо один з тестів виявить помилку, то інші теж її виявлять;

– зворотне теж вірно.

Еквівалентна область – частина області вхідних або вихідних даних, для яких поведінка компонентів або систем, ґрунтуючись на специфікації, вважаються однаковими.

Exploratory testing, також носить назву інтуїтивного тестування, має на увазі одночасне проєктування, виконання тестів і навчання продукту.

Usability тестування (User Experience).

Дозволяє перевірити комфортне використання сайту для користувача, наскільки легко знайти необхідну інформацію або виконати бажані дії.

Навігаційне тестування сайту. Чи всі сторінки, кнопки та поля на них, зрозумілі під час використання, доступ до головної сторінки та меню з усіх інших сторінок можливий, навігація проста та інтуїтивно зрозуміла [20, 21, 22, 23, 24].

Тестування контенту. Відсутність граматичних/орфографічних помилок, контент інформативний та структурований, зображення та заголовки мають відповідні розміри і розміщені вірно.

Зручність використання. Чи зрозуміла структура веб-додатку, яке враження справляє і чи наявні зайві компоненти на сторінках.

Тестування UI (User Interface). Відповідність стандартам графічних інтерфейсів й елементів дизайну, правильність локалізованих версій, тестування з різними дозволами, на смартфонах і планшетах.

Тестування сумісності (конфігураційне тестування).

Тип нефункціонального тестування програмного забезпечення, що дозволяє перевірити, чи може ПЗ працювати на іншому обладнанні, операційних системах, додатках, мережевих середовищах або мобільних пристроях [20, 21, 22, 23, 24].

Кросплатформенне (багатоплатформне) тестування сайту. Окремі функції можуть мати проблеми з певними операційними системами, тому необхідно перевіряти роботу програми у різних версіях Windows, Unix, Mac, Linux, Solaris і ін.

Кросбраузерні тестування сайту. Також коректна робота залежить від типу браузера. Верстка повинна бути кросбраузерною, щоб забезпечити однакову візуальну частину, доступність, функціональність і дизайн у всіх браузерах. Необхідно перевіряти масштабованість, розширюваність, рамки для елементів у фокусі, відсутність JS помилок (лівий нижній кут сторінки). Перевіряти роботу необхідно у таких браузерах, як: Internet Explorer, Firefox, Chrome, Safari, Opera, Edge різних версій.

Перегляд на мобільних пристроях. Незважаючи на перевірку роботи веб-додатків при різній роздійльній на комп'ютері, найчастіше помилки на мобільних пристроях залишаються непоміченими. Отже, наполегливо рекомендується перевіряти коректне відображення та роботу вашого веб-додатку на мобільних пристроях різних операційних пристроях, а також на планшетах.

Тестування БД. Необхідно перевірити правильність встановлення зв'язку з сервером, перевірити сумісність сервера з ПО, апаратними засобами, базою даних і мережею. Також слід перевірити, що відбувається під час переривання будь-якої дії, під час наступного з'єднання з сервером під час виконання операцій.

Тестування продуктивності.

Методика нефункціонального тестування, для вимірювання таких параметрів системи як чутливість та стабільність, за різних навантажень. Дозволяє досліджувати швидкодію сайту та можливості масштабованості додатку, наприклад, під час додавання нових користувачів. Проводиться з метою з'ясувати, яке навантаження сайт здатний витримати. Тестування продуктивності вимірює атрибути якості системи, такі як масштабованість, надійність і використання ресурсів.

Навантажувальне тестування – це метод тестування продуктивності, у якому реакція системи вимірюється в різних умовах навантаження. Відповідає за реакцію веб-додатка у разі збільшення робочого навантаження. Навантажувальні випробування проводяться для нормальних і пікових навантажень (одночасна купівля товару або авторизація на сайті великої кількості користувачів).

Підхід навантажувального тестування:

- Оцінити критерії прийнятності продуктивності
- Визначити критичні сценарії
- Модель робочого навантаження
- Визначити цільові рівні навантаження

– Дизайн тестів

– Виконати тести

– Проаналізуйте результати

– Завдання навантажувального тестування: час відгуку, пропускна здатність, утилізація ресурсів, максимально призначене для користувача навантаження, бізнес-метрики.

3.5 Тестування електронного ресурсу

Для того аби розуміти наскільки розроблений програмний продукт відповідає поставленим вимогам та виявити недоліки необхідно здійснити його тестування та аналіз.

Було проведено тестування контролерів, що здійснювалось за допомогою модульного тестування. Однак, слід зауважити, що даний вид тестування не забезпечує таких моментів як фільтрація, маршрутизація, прив'язка моделі.

– Перед тим, як розпочати тестування, необхідно обговорити всі важливі деталі з командою (BA, PM, розробники).

– Використовувати широкий підхід з використанням технік тест-аналізу та набору методик тест-дизайну.

– Визначити види тестування, які необхідно провести.

– Визначити цілі та ключових користувачів веб-додатків.

– Списки пристроїв, ОС та браузерів, на яких необхідно провести тестування.

– Доступ для різних ролей відвідувачів.

Необхідність складання та передачі документації

Тести, які враховують взаємодію компонентів, що беруть участь у відповідях на запит, включені до інтеграційних тестів. Тестування проводиться за допомогою модульних тестів. Це легко зробити, просто клацніть правою кнопкою миші на методі, до якого ви хочете додати тест, і виберіть Project Add

Unit Test. Відразу після цього створіть окремий клас для тестування з методом test.

```
[TestMethod]
Ссылок: 0
public void Can_Paginate()
{
    //arrange
    Mock<IPetRepository> mock = new Mock<IPetRepository>();
    mock.Setup(m => m.Pets).Returns(new List<Pet>
{
new Pet { PetId = 1, Name = "Товар1"},
new Pet { PetId = 2, Name = "Товар2"},
new Pet { PetId = 3, Name = "Товар3"},
new Pet { PetId = 4, Name = "Товар4"},
new Pet { PetId = 5, Name = "Товар5"}
});
    PetController controller = new PetController(mock.Object);
    controller.pageSize = 3;

    // act
    PetsListViewModel result = (PetsListViewModel)controller.List(null, 2).Model;

    List<Pet> pets = result.Pets.ToList();
    Assert.IsTrue(pets.Count == 2);
    Assert.AreEqual(pets[0].Name, "Товар4");
    Assert.AreEqual(pets[1].Name, "Товар5");
}
```

Рисунок 4.2 - Модульне тестування: розбиття на сторінки

Модульний тест для перевірки функціональності фільтрації за категорією буде мати такий вигляд (рисунок 4.3).

```
[TestMethod]
Ссылок: 0
public void Can_Filter_Pets()
{
    //(arrange)
    Mock<IPetRepository> mock = new Mock<IPetRepository>();
    mock.Setup(m => m.Pets).Returns(new List<Pet>

new Pet { PetId = 1, Name = "Товар1", Category="Cat1"},
new Pet { PetId = 2, Name = "Товар2", Category="Cat2"},
new Pet { PetId = 3, Name = "Товар3", Category="Cat1"},
new Pet { PetId = 4, Name = "Товар4", Category="Cat2"},
new Pet { PetId = 5, Name = "Товар5", Category="Cat3"}

    PetController controller = new PetController(mock.Object);
    controller.pageSize = 3;

    // Action
    List<Pet> result = ((PetsListViewModel)controller.List("Cat2", 1).Model)
        .Pets.ToList();

    // Assert
    Assert.AreEqual(result.Count(), 2);
    Assert.IsTrue(result[0].Name == "Товар2" && result[0].Category == "Cat2");
    Assert.IsTrue(result[1].Name == "Товар4" && result[1].Category == "Cat2");
}
```

Рисунок 4.3 - Модульне тестування: фільтрація за категорією

Для перевірки того, що метод дії Menu() коректно додав деталі про обрану категорію, у модульному тесті можна прочитати значення властивості ViewBag, яке доступне через клас ViewResult. Нижче показаний тестовий метод (рисунок 4.4).

```
[TestMethod]
Ссылок: 0
public void Indicates_Selected_Category()
{
    // Організація - створення імітованого сховища
    Mock<IPetRepository> mock = new Mock<IPetRepository>();
    mock.Setup(m => m.Pets).Returns(new Pet[] {
        new Pet { PetId = 1, Name = "Товар1", Category="Живлення"},
        new Pet { PetId = 2, Name = "Товар2", Category="Гігієна"}
    });

    // Організація - створення контролера
    NavController target = new NavController(mock.Object);

    // Організація - визначення обраної категорії
    string categoryToSelect = "Гігієна";

    // Дія
    string result = target.Menu(categoryToSelect).ViewBag.SelectedCategory;

    // Твердження
    Assert.AreEqual(categoryToSelect, result);
}
```

Рисунок 4.4 - Модульне тестування: повідомлення про обрану категорію

Клас Cart відносно простий, але в ньому є кілька важливих аспектів поведінки, у коректній роботі яких необхідно впевнитись. Кошик, що неправильно функціонує, порушить роботу всієї програми. Ми маємо протестувати кожен засіб окремо. Для розміщення цих тестів ми створимо у проекті PetStore.UnitTests новий файл модульного тестування на ім'я CartTests.cs.

Перша поведінка відноситься до додавання елемента до корзини. При першому додаванні в кошик об'єкта Pet повинен бути доданий новий екземпляр CartLine. Нижче показаний тестовий метод, включаючи визначення класу модульного тестування.

```
[TestMethod]
Ссылок: 0
public void Can_Add_New_Lines()
{
    // Організація - створення кількох тестових товарів
    Pet pet1 = new Pet { PetId = 1, Name = "Товар1" };
    Pet pet2 = new Pet { PetId = 2, Name = "Товар2" };

    // Організація - створення кошика
    Cart cart = new Cart();

    // Дія
    cart.AddItem(pet1, 1);
    cart.AddItem(pet2, 1);
    List<CartLine> results = cart.Lines.ToList();

    // Твердження
    Assert.AreEqual(results.Count(), 2);
    Assert.AreEqual(results[0].Pet, pet1);
    Assert.AreEqual(results[1].Pet, pet2);
}
```

Рисунок 4.5 - Модульне тестування: перевірка кошика (CartTests)

Однак, якщо користувач вже додав об'єкт Pet в кошик, необхідно збільшити кількість у відповідному екземплярі CartLine, а не створювати новий. Модульний тест виглядає так:

```
[TestMethod]
Ссылок: 0
public void Can_Add_Quantity_For_Existing_Lines()
{
    // Організація - створення кількох тестових товарів
    Pet pet1 = new Pet { PetId = 1, Name = "Товар1" };
    Pet pet2 = new Pet { PetId = 2, Name = "Товар2" };

    // Організація - створення кошика
    Cart cart = new Cart();

    // Дія
    cart.AddItem(pet1, 1);
    cart.AddItem(pet2, 1);
    cart.AddItem(pet1, 5);
    List<CartLine> results = cart.Lines.OrderBy(c => c.Pet.PetId).ToList();

    // Твердження
    Assert.AreEqual(results.Count(), 2);
    Assert.AreEqual(results[0].Quantity, 6);    // 6 екземплярів додано до кошика
    Assert.AreEqual(results[1].Quantity, 1);
}
```

Рисунок 4.6 - Модульне тестування: перевірка кошика (Can_Add_Quantity_For_Existing_Lines

```
[TestMethod]
Ссылок: 0
public void Calculate_Cart_Total()
{
    // Організація - створення кількох тестових товарів
    Pet pet1 = new Pet { PetId = 1, Name = "Товар1", Price = 100 };
    Pet pet2 = new Pet { PetId = 2, Name = "Товар2", Price = 55 };

    // Організація - створення кошика
    Cart cart = new Cart();

    // Дія
    cart.AddItem(pet1, 1);
    cart.AddItem(pet2, 1);
    cart.AddItem(pet1, 5);
    decimal result = cart.ComputeTotalValue();

    // Твердження
    Assert.AreEqual(result, 655);
}
```

Рисунок 4.8 - Модульне тестування: перевірка кошика

Останній тест дуже простий, адже потрібно переконатися, що в результаті очищення корзини її вміст коректно видаляється. Нижче показано потрібний модульний тест:

```
[TestMethod]
Ссылок: 0
public void Can_Clear_Contents()
{
    // Організація - створення кількох тестових товарів
    Pet pet1 = new Pet { PetId = 1, Name = "Товар1", Price = 100 };
    Pet pet2 = new Pet { PetId = 2, Name = "Товар2", Price = 55 };

    // Організація - створення кошика
    Cart cart = new Cart();

    // Дія
    cart.AddItem(pet1, 1);
    cart.AddItem(pet2, 1);
    cart.AddItem(pet1, 5);
    cart.Clear();

    // Твердження
    Assert.AreEqual(cart.Lines.Count(), 0);
}
```

Рисунок 4.9 - Модульне тестування: перевірка кошика

Після написання модульних тестів ми отримали такі результати (рисунок 4.10).

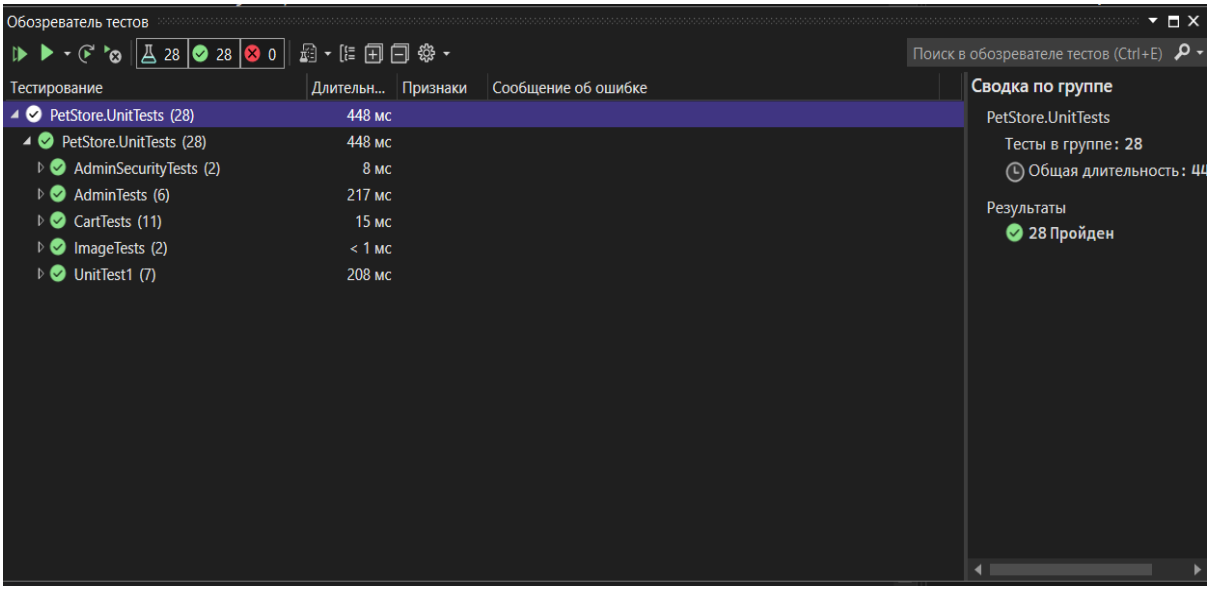


Рисунок 4.10 – Приклад модульного тесту

Отже, підводячи підсумки стосовно проведеного тестування можна зробити висновки про те, що даний програмний продукт працює адекватно та перебоїв. Всі модулі працюють злагоджено та не виникає небажаних реакцій на дії користувача. Також виявлено, що виокремлений модуль, що відповідає за сортування даних статистики відповідає вимогам, працює правильно та адекватно, не виявляє якихось відхилень у своїй роботі.

Висновки до 3-го розділу

Узагальнюючи всю пророблену роботу можна відмітити, що розроблений електронний ресурс має простий функціонал, досить легкий у використанні та містить інтерфейс, дружній до користувача та виконаний у стилі мінімалізму.

Необхідно також зазначити, що розроблений програмний продукт може в подальшому розвиватись, налагоджувати свою подальшу роботу та використовуватись у якості платформи для продажу товарів різного призначення для тварин.

ВИСНОВКИ

Отже, після проведеного дослідження та розробки програмного забезпечення можна зробити наступні висновки.

В першому розділі здійснено аналіз предметної області із визначенням основних вимог, зокрема функціональних та нефункціональних.

Також проаналізовано існуючі на цьому етапі програмні рішення, виділено переваги та недоліки таких мережевих ресурсів. На основі визначених бізнес-вимог, логічних вимог, функціональних та нефункціональних вимог розроблялось технічне завдання. Крім того, за результатами аналізу існуючого програмного забезпечення зроблено висновок, що важливими характеристиками подібних мережевих ресурсів є:

- безпека використання;
- можливість швидкого реагування на запити;
- простота та зрозумілість користувацького інтерфейсу.
- стабільність та надійність роботи;

В другому розділі продемонстровано вибір архітектури розроблюваного електронного ресурсу та визначено, що таким буде клієнт-серверна архітектура на основі шаблону MVC. Також в даному розділі здійснено визначення та опис основних модулів, здійснено аналіз та вибір технологій, а також методів реалізації даного електронного ресурсу.

В третьому розділі подано безпосередню реалізацію електронного ресурсу для торгівлі товарів для тварин, включаючи основні модулі та користувацький інтерфейс. Також здійснювався аналіз основних технологій та методик проведення тестування існуючого програмного забезпечення загалом, та даного електронного ресурсу зокрема.

Було встановлено, що тестування інтернет-магазинів має свої особливості. Зокрема, необхідно провести перевірку функціональності, починаючи з найважливіших етапів: здійснення пошуку товару, додавання його до кошика,

оформлення та оплати замовлення. Також важливо оцінити зручність використання, переконатися, що покупець може швидко і комфортно зробити покупку, чи наявна логічна поведінка сторінок, чи не має додаткових дій, які ускладнюють процес вибору та заважають покупцю. Крім того, важливим аспектом є визначення технічної здатності інтернет-магазину витримувати навантаження від відвідувачів, а також проведення перевірки на працездатність у різних браузерах та на різних пристроях.

Таким чином, результатом даної кваліфікаційної роботи є створений електронний ресурс, який надає можливість здійснювати зручні продажі, розміщення, вибір та покупку товарів для тварин. Тому, всі поставлені завдання у процесі виконання даної роботи були успішно виконані, а мета даної кваліфікаційної роботи була досягнута.

Отримані знання, уміння та навички стануть у нагоді в майбутньому, оскільки в процесі розв’язування завдань використовуються сучасні інформаційні технології. Слід зазначити, що даний електронний ресурс може розвиватися і в майбутньому стати зручною інтернет-платформою для інших учасників бізнесу, що займаються реалізацією товарів для домашніх тварин.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Волохін, О. М. Каталогізація цифрових ресурсів Інтернет: Дублінське ядро метаданих: посібник / О. М. Волохін. – Кіровоград, 2020. – 70 с.
2. Art-lemon [Електронний ресурс]. – Режим доступу: https://art-lemon.com/site-test. – Дата звернення: 15.01.2023.
3. Інтернет-магазин petslike.net [Електронний ресурс]. – Режим доступу: https://petslike.net//. – Дата звернення: 25.01.2023.
4. Як тестувати веб-сайт: основні етапи і поради / Brainlab [Електронний ресурс]. – Режим доступу: https://brainlab.com.ua/uk/blog-uk/yak-testuvati-veb-sayt-osnovn-etapi-poradi. – Дата звернення: 25.03.2023.
5. Розников, В. А., Захарова, О. В., Захарова, Е. Г. Інформаційні системи та сервіси // Проблеми програмування. – 2019. – № 4 – С. 60-72.
6. Web Application Testing: 8 Step Guide to Website Testing [Електронний ресурс] / Guru99. – Режим доступу: https://www.guru99.com/web-application-testing.html. – Дата звернення: 15.04.2022.
7. Інтернет-магазин masterzoo.ua [Електронний ресурс]. – Режим доступу: https://masterzoo.ua/. – Дата звернення: 15.01.2023.\
8. Уніфікована мова програмування UML [Електронний ресурс] / Портал знань. – Режим доступу: http://www.znannya.org/?view=uml. – Дата звернення: 15.04.
9. Григоровська, О., Панасововович, С. Сучасні інформаційні технології і популярні системи // Зап. Чернівці. наук. б-ки ім. В.Стефаника. – 2019. – Вип. 13. – С. 519–522.
10. Винокуров, В. Д., Макарчук, О. М., Патланжоглу, М. О. Практичний курс інформаційних систем та технологій: Створення баз даних / за редакцією академіка АПН України Олійника А.М. – Київ, 2020. – 348 с.
11. Кренке, Д. Теорія та практика побудови баз даних: 8-е вид. – Суми, 2017. – 800 с.

12. How to test a website [Електронний ресурс] / Geteasyqa. – Режим доступу: https://geteasyqa.com/qa/test-website/. – Дата звернення: 17.01.2023.
13. Adobe [Електронний ресурс]. – Режим доступу: https://helpx.adobe.com/ua/illustrator/using/design-website-layout.html. – Дата звернення: 27.01.
14. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. – Уведено вперше; чинний від 2016–07–01. – Київ : ДП «УкрНДНЦ», 2016. – 17 с.
15. Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE [Електронний ресурс]. URL: https://www.guru99.com/unit-testing-guide.html (Дата звернення: 15.05.2023).
16. Інтернет-магазин zoocomplex [Електронний ресурс]. URL: https://zoocomplex.com.ua (Дата звернення: 26.01.2023).
17. NOSQL – переваги та недоліки нереляційних баз даних [Електронний ресурс]. QAinfo. URL: https://www.quality-assurance-group.com/nosql-perevagy-ta-nedoliky-nerelyatsijnyh-baz-danyh/ (Дата звернення: 15.01.2023).
18. Модульне тестування [Електронний ресурс]. QaLight. URL: https://qalight.ua/baza-znaniy/modulne-testuvannya/ (Дата звернення: 15.04.2023).
19. Тестування проекту [Електронний ресурс]. URL: https://qalearning.com.ua/theory/lectures/material/testing-intro/ (Дата звернення: 15.05.2023).
20. Метод тестування whitebox [Електронний ресурс]. QaLight. URL: https://qalight.ua/baza-znaniy/white-black-grey-box-testuvannya/ (Дата звернення: 15.04.2023).
21. SWB [Електронний ресурс]. URL: https://superbwebsitebuilders.com/ru/kak-sozdat-svoj-prostoj-html-sajt-v-bloknote/ (Дата звернення: 24.02.2023).

22. Модульне тестування [Електронний ресурс]. QaLight. URL: https://qalight.ua/baza-znaniy/modulne-testuvannya/ (Дата звернення: 15.05.2023).
23. Інтернет-магазин з продажу товарів для домашніх тварин. Курсовий проект з дисципліни Програмування Інтернет, М. О. Скоробагатий, керівник: О. М. Яшина.
24. Тестування веб-проектів [Електронний ресурс]. URL: https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etapi-ta-poradi/. (Дата звернення: 15.04.2023).
25. Албаши М. Веб-програмування на PHP і MySQL: практичний курс [Текст] / М. Албаши. – К. : ДМК Прес, 2018. – 352 с. ISBN 978-617-09-3899-3.
26. Веб-програмування: основи HTML, CSS, JavaScript [Електронний ресурс]. –URL: https://developer.mozilla.org/uk/docs/Learn/Getting_started_with_the_web/ (дата звернення: 01.06.2023).
27. Джейсон Г. Практичне веб-програмування на PHP: збірник задач та прикладів [Текст] / Г. Джейсон, Д. Сулліван. – К. : Видавництво "Діалектика", 2019. – 256 с. ISBN 978-617-7555-48-7.
28. Кулик А. Сучасний веб-дизайн. Розробка і розміщення сайтів: навчальний посібник [Текст] / А. Кулик. – К. : Ліра-К, 2017. – 264 с. ISBN 978-966-485-250-5.
29. Лях І. Програмування для Інтернету: HTML, CSS, JavaScript, PHP, MySQL [Текст] / І. Лях. – К. : ТОВ «Видавничий дім «Ін Юре», 2019. – 376 с. ISBN 978-617-7851-37-6.
30. Мозіла Девелопер Нетворк. Веб-технології [Електронний ресурс]. – URL: https://developer.mozilla.org/uk/docs/Web (дата звернення: 01.06.2023).
31. Ніколс Д. Веб-програмування. HTML, CSS, JavaScript, PHP, MySQL [Текст] / Д. Ніколс. – К. : Видавництво “Діалектика”, 2019. – 400 с. ISBN 978-617-7555-23-4.

32. Ореллі К. PHP. Підручник з програмування [Текст] / К. Ореллі. – К. : Видавнича група BHV, 2019. – 416 с. ISBN 978-617-7555-28-9.
33. Перроне К. Веб-програмування і бази даних для кожного [Текст] / К. Перроне. – К. : ДМК Прес, 2018. – 352 с. ISBN 978-617-09-3897-9.
34. Програмування веб-сторінок: HTML, CSS, JavaScript [Електронний ресурс]. – URL: https://webref.ru/course/html-css-js (дата звернення: 01.06.2023).
35. Саммерфілд М. PHP та MySQL для абсолютних початківців [Текст] / М. Саммерфілд. – К. : Видавнича група BHV, 2019. – 384 с. ISBN 978-617-7555-50-0.
36. Сафронов Д. Сучасні веб-технології [Текст] / Д. Сафронов. – К. : ДМК Прес, 2020. – 432 с. ISBN 978-617-09-5094-1.
37. Томсон Л. Практичне програмування на JavaScript [Текст] / Л. Томсон. – К. : Видавничий дім «Пітер», 2018. – 608 с. ISBN 978-617-7091-38-0.
38. Христич В. Сучасні веб-студії та технології [Текст] / В. Христич. – К. : ДМК Прес, 2019. – 336 с. ISBN 978-617-09-4420-2.
39. Чайкін Д. Програмування для Інтернету: навчальний посібник [Текст] / Д. Чайкін. – К. : Видавничий дім «Ін Юре», 2020. – 392 с. ISBN 978-617-7851-60-4.
40. Черепанов В. Програмування на мові PHP [Текст] / В. Черепанов. – К. : ДМК Прес, 2019. – 224 с. ISBN 978-617-09-4840-8.

ДОДАТОК А
(обов’язковий)

КОД (ЛІСТИНГ)

Контролер PetController

```
namespace PetStore.WebUI.Controllers
{
    public class PetController : Controller
    {
        private IPetRepository repository;
        // кількість товару на 1 сторінці
        public int pageSize = 4;
        public PetController(IPetRepository repo)
        {
            repository = repo;
        }
        public ViewResult List(string category, int page = 1)
        {
            PetsListViewModel model = new PetsListViewModel
            {
                Pets = repository.Pets
                    .Where(p => category == null || p.Category == category)
                    .OrderBy(pet => pet.PetId)
                    .Skip((page - 1) * pageSize)
                    .Take(pageSize),
                PagingInfo = new PagingInfo
                {
                    CurrentPage = page,
                    ItemsPerPage = pageSize,
                    TotalItems = category == null ?
                        repository.Pets.Count() :
                        repository.Pets.Where(pet => pet.Category == category).Count()
                },
                CurrentCategory = category
            };
            return View(model);
        }
        public FileContentResult GetImage(int petId)
        {
            Pet pet = repository.Pets
                .FirstOrDefault(g => g.PetId == petId);

            if (pet != null)
            {
                return File(pet.ImageData, pet.ImageMimeType);
            }
            else
            {
                return null;
            }
        }
    }
}
```

Контролер NavController

```
namespace PetStore.WebUI.Controllers
{
    public class NavController : Controller
    {
        private IPetRepository repository;

        public NavController(IPetRepository repo)
        {
            repository = repo;
        }
```

```
        public PartialViewResult Menu(string category = null)
        {
            ViewBag.SelectedCategory = category;

            IEnumerable<string> categories = repository.Pets
                .Select(pet => pet.Category)
                .Distinct()
                .OrderBy(x => x);

            return PartialView("FlexMenu", categories);
        }
    }
}
```

Контролер CartController

```
namespace PetStore.WebUI.Controllers
{
    public class CartController : Controller
    {
        private IPetRepository repository;
        private IOrderProcessor orderProcessor;

        public CartController(IPetRepository repo, IOrderProcessor processor)
        {
            repository = repo;
            orderProcessor = processor;
        }
        public ViewResult Index(Cart cart, string returnUrl)
        {
            return View(new CartIndexViewModel
            {
                Cart = cart,
                ReturnUrl = returnUrl
            });
        }

        public RedirectToRouteResult AddToCart(Cart cart, int petId, string returnUrl)
        {
            Pet pet = repository.Pets
                .FirstOrDefault(g => g.PetId == petId);

            if (pet != null)
            {
                cart.AddItem(pet, 1);
            }
            return RedirectToAction("Index", new { returnUrl });
        }

        public RedirectToRouteResult RemoveFromCart(Cart cart, int petId, string
returnUrl)
        {
            Pet pet = repository.Pets
                .FirstOrDefault(g => g.PetId == petId);

            if (pet != null)
            {
                cart.RemoveLine(pet);
            }
            return RedirectToAction("Index", new { returnUrl });
        }
        public PartialViewResult Summary(Cart cart)
        {
```

```
            return PartialView(cart);
        }
        public ViewResult Checkout()
        {
            return View(new ShippingDetails());
        }

        [HttpPost]
        public ViewResult Checkout(Cart cart, ShippingDetails shippingDetails)
        {
            if (cart.Lines.Count() == 0)
            {
                ModelState.AddModelError("", "Извините, ваша корзина пуста!");
            }

            if (ModelState.IsValid)
            {
                orderProcessor.ProcessOrder(cart, shippingDetails);
                cart.Clear();
                return View("Completed");
            }
            else
            {
                return View(shippingDetails);
            }
        }


    }
}
```

Контролер AdminController

```
namespace PetStore.WebUI.Controllers
{
    [Authorize]
    public class AdminController : Controller
    {
        public ViewResult Create()
        {
            return View("Edit", new Pet());
        }
        IPetRepository repository;


        public AdminController(IPetRepository repo)
        {
            repository = repo;
        }

        public ViewResult Index()
        {
            return View(repository.Pets);
        }
        public ViewResult Edit(int petId)
        {
            Pet pet = repository.Pets
                .FirstOrDefault(g => g.PetId == petId);
            return View(pet);
        }


        [HttpPost]
        public ActionResult Edit(Pet pet, HttpPostedFileBase image = null)
```

```
        {
            if (ModelState.IsValid)
            {
                if (image != null)
                {
                    pet.ImageMimeType = image.ContentType;
                    pet.ImageData = new byte[image.ContentLength];
                    image.InputStream.Read(pet.ImageData, 0, image.ContentLength);
                }
                repository.SavePet(pet);
                TempData["message"] = string.Format("Изменения в товаре \"{0}\" были
сохранены", pet.Name);
                return RedirectToAction("Index");
            }
            else
            {

                return View(pet);
            }
        }


        [HttpPost]
        public ActionResult Delete(int petId)
        {
            Pet deletedPet = repository.DeletePet(petId);
            if (deletedPet != null)
            {
                TempData["message"] = string.Format("Товар \"{0}\" был удален",
                    deletedPet.Name);
            }
            return RedirectToAction("Index");
        }
    }
}
```

Контролер AccountController

```
namespace PetStore.WebUI.Controllers
{
    [Authorize]
    public class AdminController : Controller
    {
        public ViewResult Create()
        {
            return View("Edit", new Pet());
        }
        IPetRepository repository;


        public AdminController(IPetRepository repo)
        {
            repository = repo;
        }

        public ViewResult Index()
        {
            return View(repository.Pets);
        }
        public ViewResult Edit(int petId)
        {
            Pet pet = repository.Pets
                .FirstOrDefault(g => g.PetId == petId);
            return View(pet);
```

```
        }

        // Перегрузка Edit
        [HttpPost]
        public ActionResult Edit(Pet pet, HttpPostedFileBase image = null)
        {
            if (ModelState.IsValid)
            {
                if (image != null)
                {
                    pet.ImageMimeType = image.ContentType;
                    pet.ImageData = new byte[image.ContentLength];
                    image.InputStream.Read(pet.ImageData, 0, image.ContentLength);
                }
                repository.SavePet(pet);
                TempData["message"] = string.Format("Изменения в товаре \"{0}\" были
сохранены", pet.Name);
                return RedirectToAction("Index");
            }
            else
            {
                // Что-то не так со значениями данных
                return View(pet);
            }
        }


        [HttpPost]
        public ActionResult Delete(int petId)
        {
            Pet deletedPet = repository.DeletePet(petId);
            if (deletedPet != null)
            {
                TempData["message"] = string.Format("Товар \"{0}\" был удален",
                    deletedPet.Name);
            }
            return RedirectToAction("Index");
        }
    }
}
```

Модульні тести UnitTest1

```
namespace PetStore.UnitTests
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void Can_Paginate()
        {

            Mock<IPetRepository> mock = new Mock<IPetRepository>();
            mock.Setup(m => m.Pets).Returns(new List<Pet>
    {
        new Pet { PetId = 1, Name = "Товар1"},
        new Pet { PetId = 2, Name = "Товар2"},
        new Pet { PetId = 3, Name = "Товар3"},
        new Pet { PetId = 4, Name = "Товар4"},
        new Pet { PetId = 5, Name = "Товар5"}
    });
            PetController controller = new PetController(mock.Object);
            controller.pageSize = 3;
```

```
            PetsListViewModel result = (PetsListViewModel)controller.List(null,
2).Model;


            List<Pet> pets = result.Pets.ToList();
            Assert.IsTrue(pets.Count == 2);
            Assert.AreEqual(pets[0].Name, "Товар4");
            Assert.AreEqual(pets[1].Name, "Товар5");
        }
        [TestMethod]
        public void Can_Generate_Page_Links()
        {


            HtmlHelper myHelper = null;


            PagingInfo pagingInfo = new PagingInfo
            {
                CurrentPage = 2,
                TotalItems = 28,
                ItemsPerPage = 10
            };


            Func<int, string> pageUrlDelegate = i => "Page" + i;


            MvcHtmlString result = myHelper.PageLinks(pagingInfo, pageUrlDelegate);

            Assert.AreEqual(@"<a class=""btn btn-default"" href=""Page1"">1</a>"
                + @"<a class=""btn btn-default btn-primary selected""
href=""Page2"">2</a>"
                + @"<a class=""btn btn-default"" href=""Page3"">3</a>",
                result.ToString());
        }
        [TestMethod]
        public void Can_Send_Pagination_View_Model()
        {

            Mock<IPetRepository> mock = new Mock<IPetRepository>();
            mock.Setup(m => m.Pets).Returns(new List<Pet>
    {
        new Pet { PetId = 1, Name = "Товар1"},
        new Pet { PetId = 2, Name = "Товар2"},
        new Pet { PetId = 3, Name = "Товар3"},
        new Pet { PetId = 4, Name = "Товар4"},
        new Pet { PetId = 5, Name = "Товар5"}
    });
            PetController controller = new PetController(mock.Object);
            controller.pageSize = 3;

            PetsListViewModel result
                = (PetsListViewModel)controller.List(null, 2).Model;


            PagingInfo pageInfo = result.PagingInfo;
            Assert.AreEqual(pageInfo.CurrentPage, 2);
            Assert.AreEqual(pageInfo.ItemsPerPage, 3);
            Assert.AreEqual(pageInfo.TotalItems, 5);
            Assert.AreEqual(pageInfo.TotalPages, 2);
        }
        [TestMethod]
        public void Can_Filter_Pets()
```

```
    {

        Mock<IPetRepository> mock = new Mock<IPetRepository>();
        mock.Setup(m => m.Pets).Returns(new List<Pet>
{
    new Pet { PetId = 1, Name = "Товар1", Category="Cat1"},
    new Pet { PetId = 2, Name = "Товар2", Category="Cat2"},
    new Pet { PetId = 3, Name = "Товар3", Category="Cat1"},
    new Pet { PetId = 4, Name = "Товар4", Category="Cat2"},
    new Pet { PetId = 5, Name = "Товар5", Category="Cat3"}
});
        PetController controller = new PetController(mock.Object);
        controller.pageSize = 3;

        List<Pet> result = ((PetsListViewModel)controller.List("Cat2", 1).Model)
            .Pets.ToList();


        Assert.AreEqual(result.Count(), 2);
        Assert.IsTrue(result[0].Name == "Товар2" && result[0].Category == "Cat2");
        Assert.IsTrue(result[1].Name == "Товар4" && result[1].Category == "Cat2");
    }
    [TestMethod]
    public void Can_Create_Categories()
    {

        Mock<IPetRepository> mock = new Mock<IPetRepository>();
        mock.Setup(m => m.Pets).Returns(new List<Pet> {
            new Pet { PetId = 1, Name = "Товар1", Category="Живлення"},
            new Pet { PetId = 2, Name = "Товар2", Category="Живлення"},
            new Pet { PetId = 3, Name = "Товар3", Category="Здоров'я"},
            new Pet { PetId = 4, Name = "Товар4", Category="Живлення"},
        });


        NavController target = new NavController(mock.Object);


        List<string> results = ((IEnumerable<string>)target.Menu().Model).ToList();


        Assert.AreEqual(results.Count(), 2);
        Assert.AreEqual(results[0], "Живлення");
        Assert.AreEqual(results[1], "Здоров'я");

    }
    [TestMethod]
    public void Indicates_Selected_Category()
    {

        Mock<IPetRepository> mock = new Mock<IPetRepository>();
        mock.Setup(m => m.Pets).Returns(new Pet[] {
            new Pet { PetId = 1, Name = "Товар1", Category="Живлення"},
            new Pet { PetId = 2, Name = "Товар2", Category="Гігієна"}
         });


        NavController target = new NavController(mock.Object);


        string categoryToSelect = "Гігієна";

        string result = target.Menu(categoryToSelect).ViewBag.SelectedCategory;
```

```
            Assert.AreEqual(categoryToSelect, result);
        }
        [TestMethod]
        public void Generate_Category_Specific_Game_Count()
        {

            Mock<IPetRepository> mock = new Mock<IPetRepository>();
            mock.Setup(m => m.Pets).Returns(new List<Pet>
            {
                new Pet { PetId = 1, Name = "Товар1", Category="Cat1"},
                new Pet { PetId = 2, Name = "Товар2", Category="Cat2"},
                new Pet { PetId = 3, Name = "Товар3", Category="Cat1"},
                new Pet { PetId = 4, Name = "Товар4", Category="Cat2"},
                new Pet { PetId = 5, Name = "Товар5", Category="Cat3"}
            });
            PetController controller = new PetController(mock.Object);
            controller.pageSize = 3;


            int res1 =
((PetsListViewModel)controller.List("Cat1").Model).PagingInfo.TotalItems;
            int res2 =
((PetsListViewModel)controller.List("Cat2").Model).PagingInfo.TotalItems;
            int res3 =
((PetsListViewModel)controller.List("Cat3").Model).PagingInfo.TotalItems;
            int resAll =
((PetsListViewModel)controller.List(null).Model).PagingInfo.TotalItems;


            Assert.AreEqual(res1, 2);
            Assert.AreEqual(res2, 2);
            Assert.AreEqual(res3, 1);
            Assert.AreEqual(resAll, 5);
        }

    }
}
```

Модульні тести CartTests

```
namespace PetStore.UnitTests
{
    [TestClass]
    public class CartTests
    {
        [TestMethod]
        public void Can_Add_New_Lines()
        {

            Pet pet1 = new Pet { PetId = 1, Name = "Товар1" };
            Pet pet2 = new Pet { PetId = 2, Name = "Товар2" };


            Cart cart = new Cart();


            cart.AddItem(pet1, 1);
            cart.AddItem(pet2, 1);
            List<CartLine> results = cart.Lines.ToList();


            Assert.AreEqual(results.Count(), 2);
```

```
    Assert.AreEqual(results[0].Pet, pet1);
    Assert.AreEqual(results[1].Pet, pet2);
}
[TestMethod]
public void Can_Add_Quantity_For_Existing_Lines()
{

    Pet pet1 = new Pet { PetId = 1, Name = "Товар1" };
    Pet pet2 = new Pet { PetId = 2, Name = "Товар2" };


    Cart cart = new Cart();


    cart.AddItem(pet1, 1);
    cart.AddItem(pet2, 1);
    cart.AddItem(pet1, 5);
    List<CartLine> results = cart.Lines.OrderBy(c => c.Pet.PetId).ToList();


    Assert.AreEqual(results.Count(), 2);
    Assert.AreEqual(results[0].Quantity, 6);
    Assert.AreEqual(results[1].Quantity, 1);
}
[TestMethod]
public void Can_Remove_Line()
{

    Pet pet1 = new Pet { PetId = 1, Name = "Товар1" };
    Pet pet2 = new Pet { PetId = 2, Name = "Товар2" };
    Pet pet3 = new Pet { PetId = 3, Name = "Товар3" };


    Cart cart = new Cart();


    cart.AddItem(pet1, 1);
    cart.AddItem(pet2, 4);
    cart.AddItem(pet3, 2);
    cart.AddItem(pet2, 1);


    cart.RemoveLine(pet2);


    Assert.AreEqual(cart.Lines.Where(c => c.Pet == pet2).Count(), 0);
    Assert.AreEqual(cart.Lines.Count(), 2);
}
[TestMethod]
public void Calculate_Cart_Total()
{

    Pet pet1 = new Pet { PetId = 1, Name = "Товар1", Price = 100 };
    Pet pet2 = new Pet { PetId = 2, Name = "Товар2", Price = 55 };


    Cart cart = new Cart();


    cart.AddItem(pet1, 1);
    cart.AddItem(pet2, 1);
    cart.AddItem(pet1, 5);
    decimal result = cart.ComputeTotalValue();
```

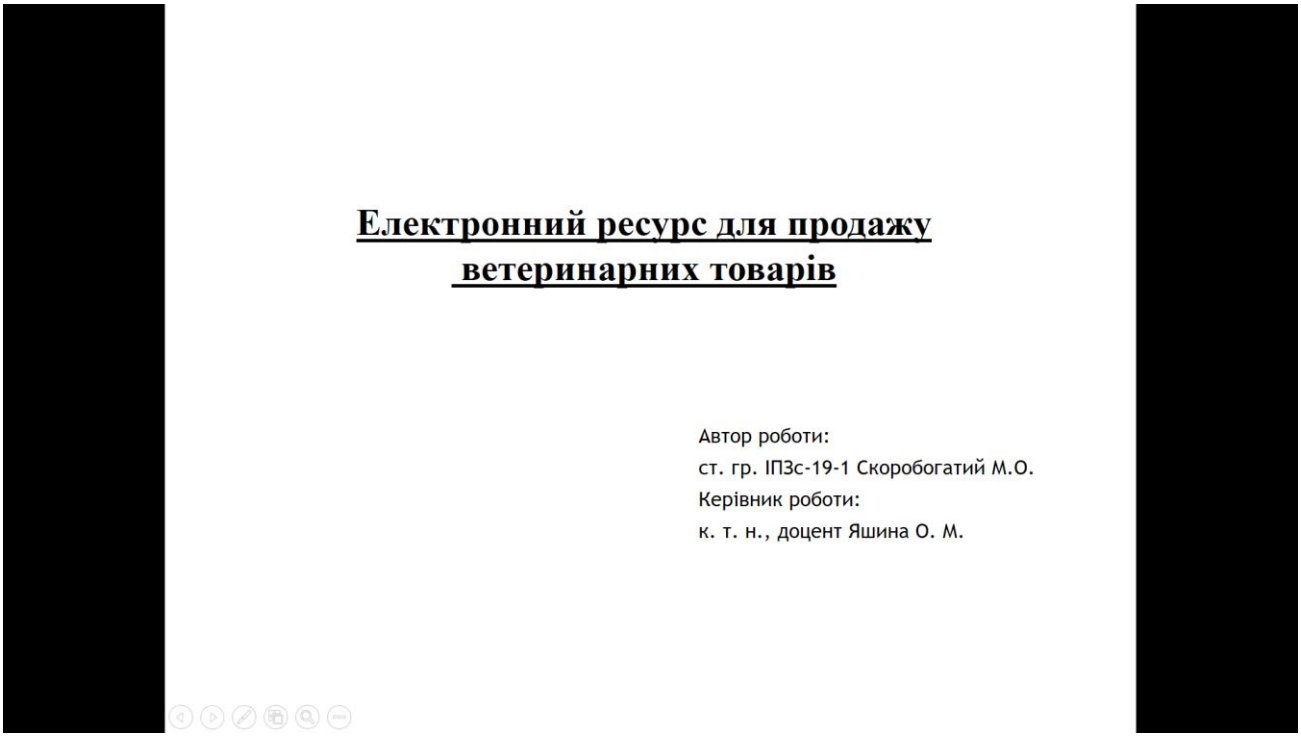
```
        Assert.AreEqual(result, 655);
    }
    [TestMethod]
    public void Can_Clear_Contents()
    {

        Pet pet1 = new Pet { PetId = 1, Name = "Товар1", Price = 100 };
        Pet pet2 = new Pet { PetId = 2, Name = "Товар2", Price = 55 };


        Cart cart = new Cart();


        cart.AddItem(pet1, 1);
        cart.AddItem(pet2, 1);
        cart.AddItem(pet1, 5);
        cart.Clear();


        Assert.AreEqual(cart.Lines.Count(), 0);
    }

    [TestMethod]
    public void Can_Add_To_Cart()
    {

        Mock<IPetRepository> mock = new Mock<IPetRepository>();
        mock.Setup(m => m.Pets).Returns(new List<Pet> {
    new Pet {PetId = 1, Name = "Товар1", Category = "Кат1"},
}.AsQueryable());


        Cart cart = new Cart();


        CartController controller = new CartController(mock.Object, null);


        controller.AddToCart(cart, 1, null);


        Assert.AreEqual(cart.Lines.Count(), 1);
        Assert.AreEqual(cart.Lines.ToList()[0].Pet.PetId, 1);
    }

    [TestMethod]
    public void Adding_Pet_To_Cart_Goes_To_Cart_Screen()
    {

        Mock<IPetRepository> mock = new Mock<IPetRepository>();
        mock.Setup(m => m.Pets).Returns(new List<Pet> {
    new Pet {PetId = 1, Name = "Товар1", Category = "Кат1"},
        }.AsQueryable());


        Cart cart = new Cart();


        CartController controller = new CartController(mock.Object, null);


        RedirectToRouteResult result = controller.AddToCart(cart, 2, "myUrl");
```

```
            Assert.AreEqual(result.RouteValues["action"], "Index");
            Assert.AreEqual(result.RouteValues["returnUrl"], "myUrl");
        }


        [TestMethod]
        public void Can_View_Cart_Contents()
        {

            Cart cart = new Cart();


            CartController target = new CartController(null, null);


            CartIndexViewModel result
                = (CartIndexViewModel)target.Index(cart, "myUrl").ViewData.Model;


            Assert.AreSame(result.Cart, cart);
            Assert.AreEqual(result.ReturnUrl, "myUrl");
        }
        [TestMethod]
        public void Cannot_Checkout_Empty_Cart()
        {

            Mock<IOrderProcessor> mock = new Mock<IOrderProcessor>();


            Cart cart = new Cart();


            ShippingDetails shippingDetails = new ShippingDetails();


            CartController controller = new CartController(null, mock.Object);


            ViewResult result = controller.Checkout(cart, shippingDetails);


            mock.Verify(m => m.ProcessOrder(It.IsAny<Cart>(),
It.IsAny<ShippingDetails>()),
                Times.Never());


            Assert.AreEqual("", result.ViewName);


            Assert.AreEqual(false, result.ViewData.ModelState.IsValid);
        }
        [TestMethod]
        public void Cannot_Checkout_Invalid_ShippingDetails()
        {

            Mock<IOrderProcessor> mock = new Mock<IOrderProcessor>();


            Cart cart = new Cart();
            cart.AddItem(new Pet(), 1);


            CartController controller = new CartController(null, mock.Object);
```

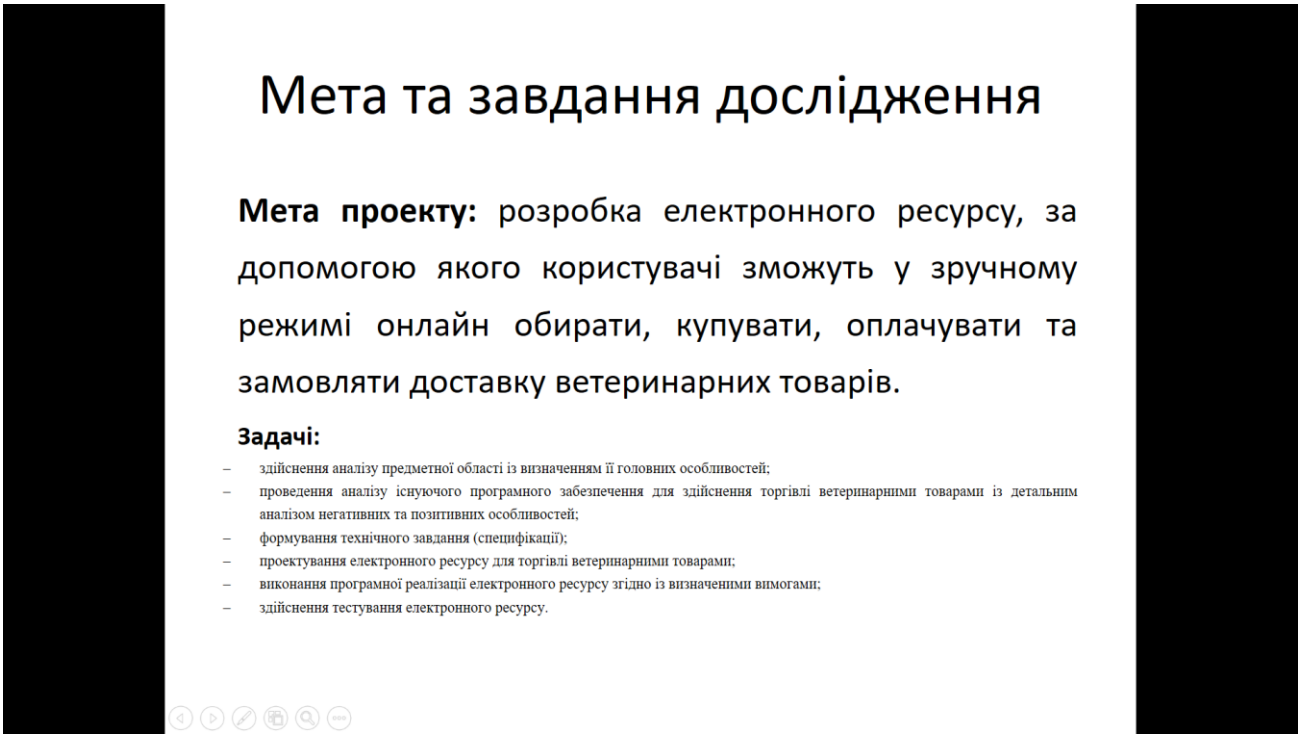
```
            controller.ModelState.AddModelError("error", "error");


            ViewResult result = controller.Checkout(cart, new ShippingDetails());


            mock.Verify(m => m.ProcessOrder(It.IsAny<Cart>(),
It.IsAny<ShippingDetails>()),
                Times.Never());


            Assert.AreEqual("", result.ViewName);


            Assert.AreEqual(false, result.ViewData.ModelState.IsValid);
        }
        [TestMethod]
        public void Can_Checkout_And_Submit_Order()
        {

            Mock<IOrderProcessor> mock = new Mock<IOrderProcessor>();

            Cart cart = new Cart();
            cart.AddItem(new Pet(), 1);


            CartController controller = new CartController(null, mock.Object);


            ViewResult result = controller.Checkout(cart, new ShippingDetails());


            mock.Verify(m => m.ProcessOrder(It.IsAny<Cart>(),
It.IsAny<ShippingDetails>()),
                Times.Once());


            Assert.AreEqual("Completed", result.ViewName);


            Assert.AreEqual(true, result.ViewData.ModelState.IsValid);
        }
    }

}
```

ДОДАТОК Б
(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ



Слайд презентації 1



Слайд презентації 2

Актуальність

- На даний момент в умовах (спочатку пандемії ковід-19, зараз в умовах військового стану) тема розробки електронного ресурсу для здійснення покупок товарів для домашніх тварин за допомогою мережі інтернет є досить актуальною. Крім цього важливим завданням розвитку країни в умовах військового стану стає перехід до інформаційної економіки, в якій велика увага приділяється науково-технічному прогресу, масовому поширенню інноваційних технологій та електронного управління.
- Інтернет-комерція все більше і більше входить у життя великої кількості людей. Вона відкриває нові можливості для ведення бізнесу, і більшість компаній вже перейшли на ведення та розвиток своєї справи в Інтернет-середовищі.
- Отже, можна зробити висновок про те, що тема розробки електронного ресурс для продажу ветеринарних товарів є досить актуальною, особливо в сучасних умовах ведення війни, в ході якої страждає велика кількість тварин і потребує різноманітного роду догляду та відповідно якісних ветеринарних товарів.

Слайд презентації 3

Предметна область

Розроблюване програмне забезпечення створюється для здійснення автоматизації у режимі ручного вибору товарів для тварин, в тому числі ветеринарних, а також замовлення із можливістю оплати у режимі онлайн.

Предметною областю даного дослідження є торгівля товарами для тварин за допомогою електронного ресурсу в мережі Інтернет.

Слайд презентації 4

Аналіз стану проблеми та інших рішень

Основними перевагами розглянутих рішень є:

- наявність великого каталогу наявного товару;
- вибір товару, що мають знижку ;
- можливість здійснити пошук товару без авторизації.

Недоліки:

- відсутність можливості вибору мови (багато ресурсів до цих пір використовують російську мову як основну);
- немає відгуків реальних покупців та можливості залишити такий відгук;
- досить складний пошук та перегляд товару;
- Відсутність кнопки повернення зі сторінки перегляду товару на головну сторінку та можливості продовжити покупки.

Слайд презентації 5

Визначено вимоги до ресурсу

- ресурс повинен представляти товари замовника в Інтернеті, знайомити відвідувача з асортиментом, організовувати взаємодія відвідувача сайту із замовником, надавати довідкову інформацію;
- ресурс має бути українською мовою або англомовною версією;
- адміністративна частина сайту має бути так само українською мовою;
- інтерфейс електронного ресурсу (інтернет-магазину) повинен мати інтуїтивно зрозумілий вид та навігацію. За основу можна взяти вже звичний вигляд інтернет-магазину з навігацією зліва, пошуком посередині та кошиком праворуч. Але кожен відвідувач незалежно від віку має легко знаходити необхідний йому товар.

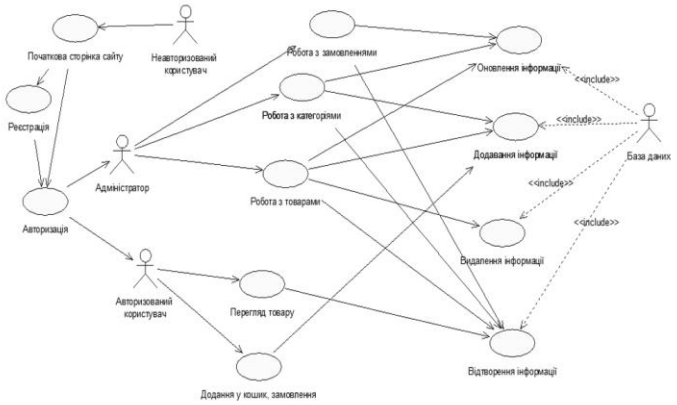
Слайд презентації 6

Визначено вимоги до ресурсу

- ресурс не повинен бути завантаженим сайтом, швидко завантажуватись, не навантажувати техніку користувачів та мати швидкий відгук;
- повинен мати зрозумілий та логічний каталог з можливістю сортування товарів за різними параметрами, а також різними способами відображення.
- у каталозі ресурсу в користувача має бути можливість фільтрації товарів за різними критеріями.
- наявність форми зворотного зв'язку;
- наявність опису характеристик товару, що продається, немає чіткого розуміння наскільки деталізована має бути інформація, але її має вистачати для загального образу товару покупцем;
- всі товари необхідно супроводжувати фотографіями з різних ракурсів.

Слайд презентації 7

Діаграма варіантів використання



Слайд презентації 8

Слайд презентації 9

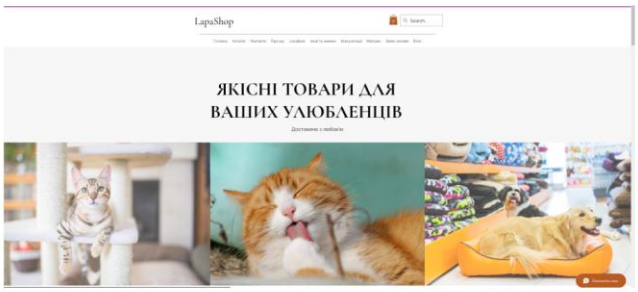
Слайд презентації 10

Модель бази даних

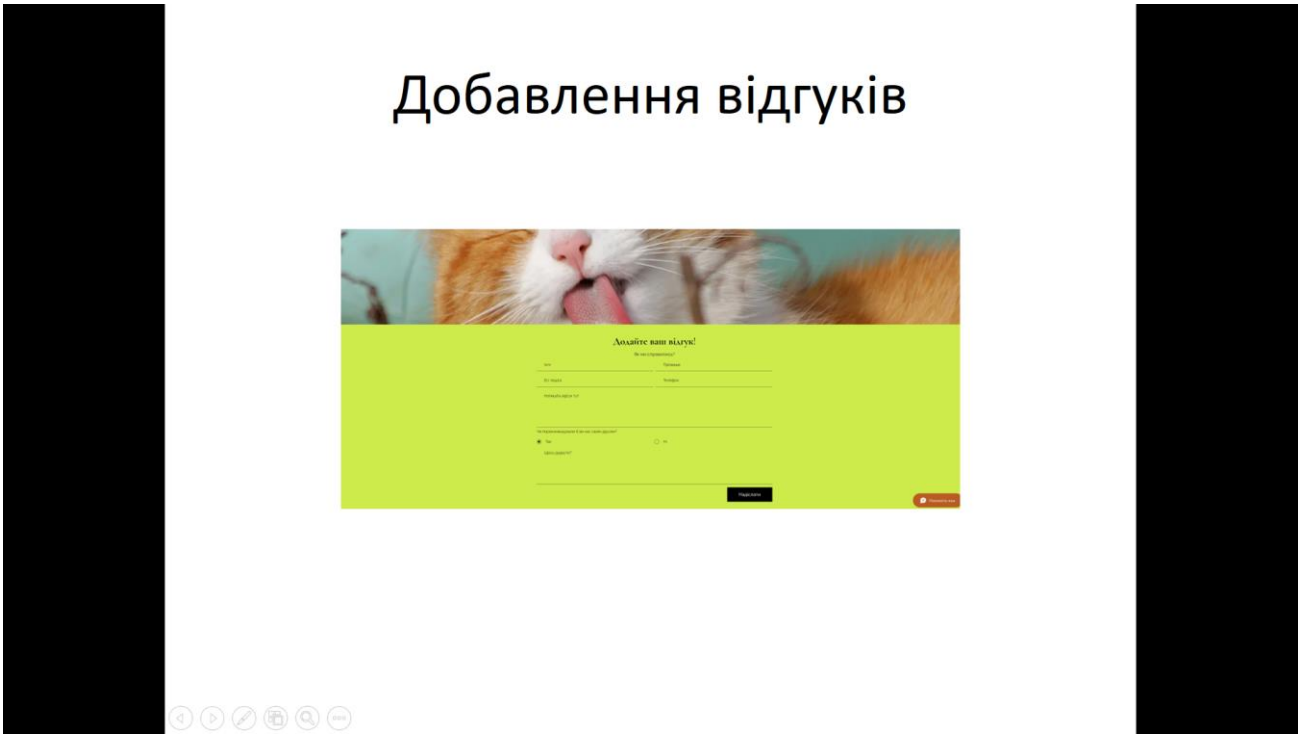
Слайд презентації 11

Результати роботи

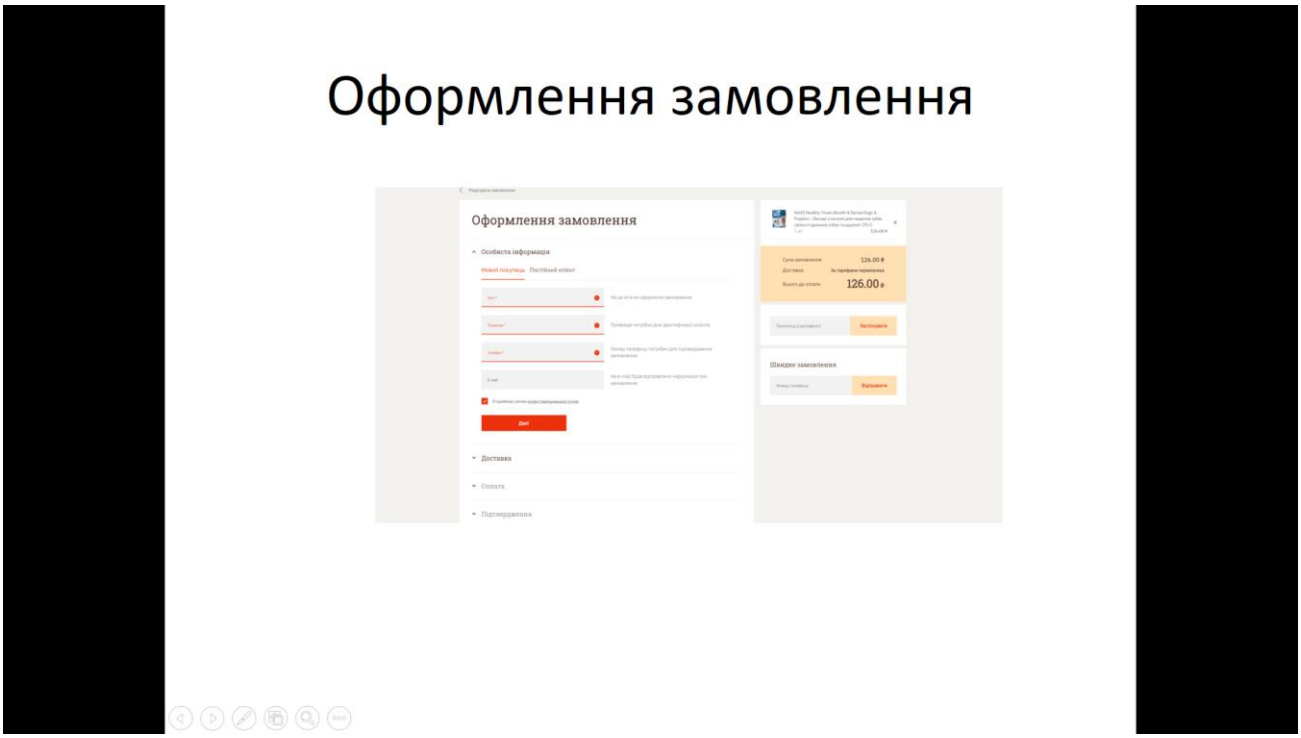
головне вікно



Слайд презентації 12

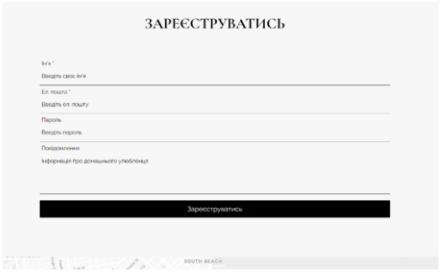


Слайд презентації 13



Слайд презентації 14

Реєстрація користувача



Слайд презентації 15

Сторінка адміністратора

Слайд презентації 16

Отримані результати

В результаті виконання даної дипломної роботи отримано програмне забезпечення з можливістю автоматичного вибору товарів для тварин у ручному режимі за допомогою мережі Інтернет в режимі онлайн.

Слайд презентації 17

Перспективи розвитку

- В подальшому даний проект може розвиватись і розширюватись та стати платформою для здійснення покупок та розміщення асортименту товару для великої кількості продавців із своїми інтернет-магазинами. Тобто розширення до так званого маркет-плейсу.

Слайд презентації 18

Висновки

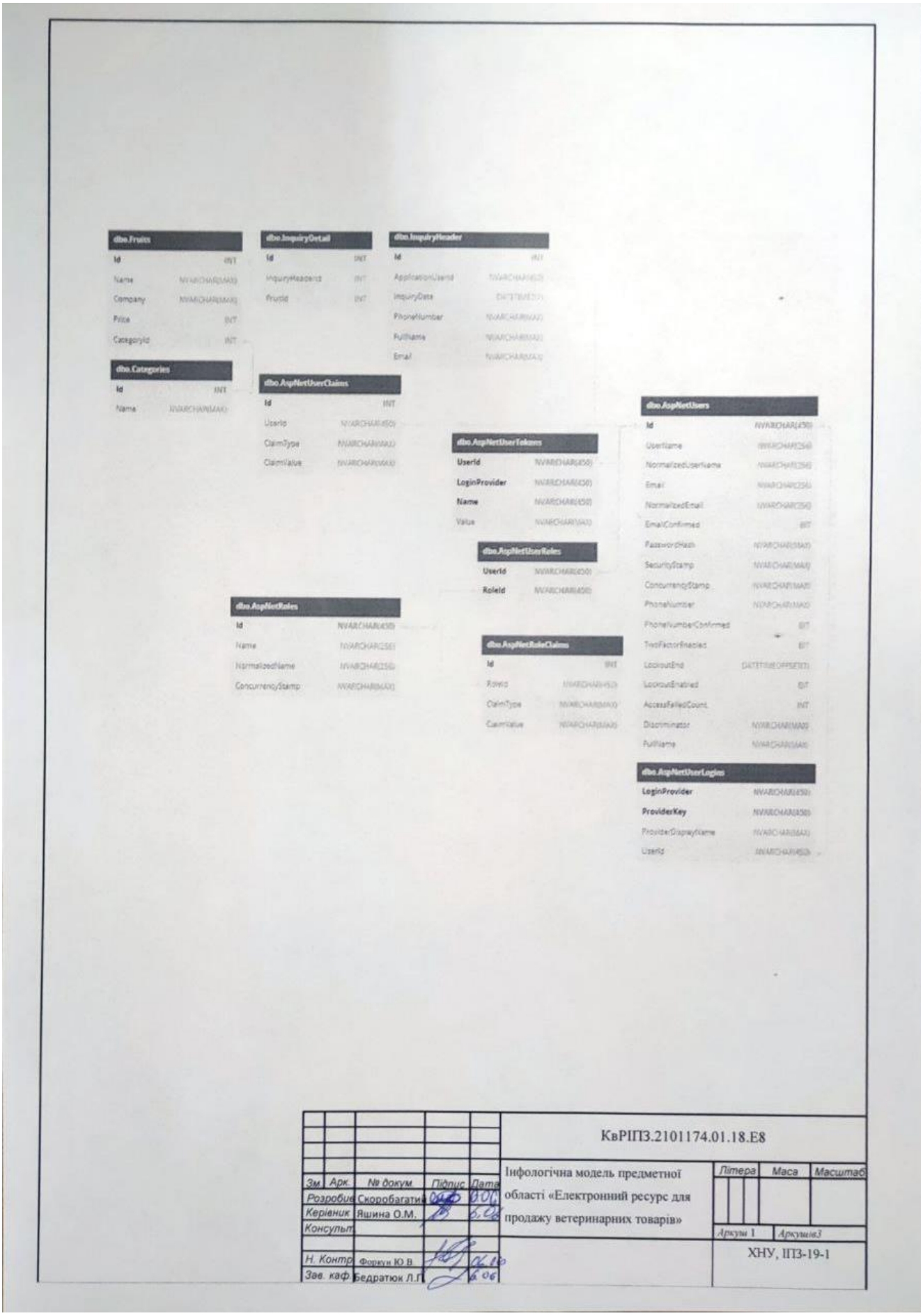
- здійснено аналіз предметної області із визначенням її головних особливостей;
- проведено аналіз існуючого програмного забезпечення;
- розроблено технічне завдання (ТЗ);
- зроблено проектування програмного продукту;
- виконано програмну реалізацію проекту;
- здійснено тестування веб-ресурсу.

Слайд презентації 19

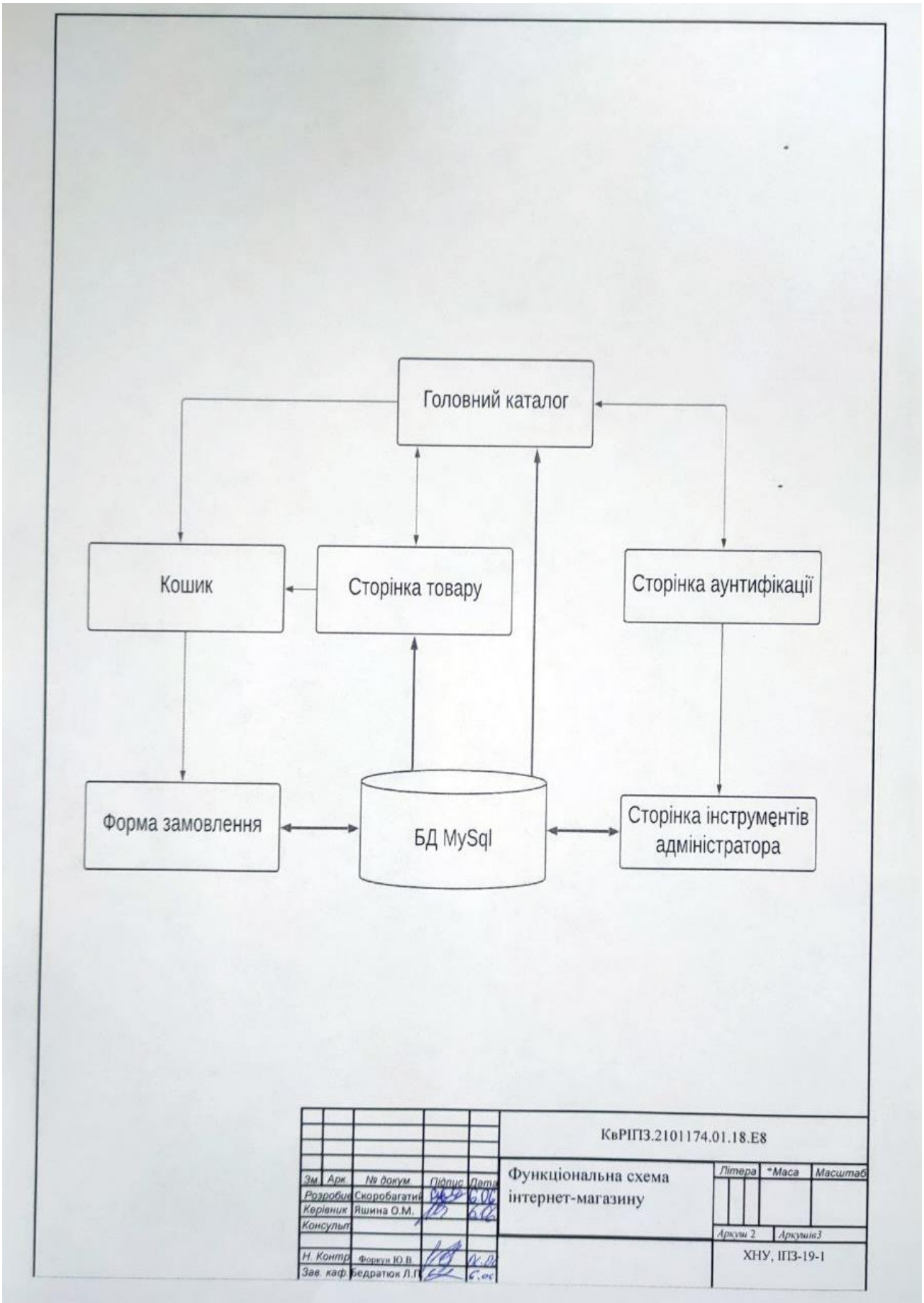
Дякую за увагу!

Слайд презентації 20

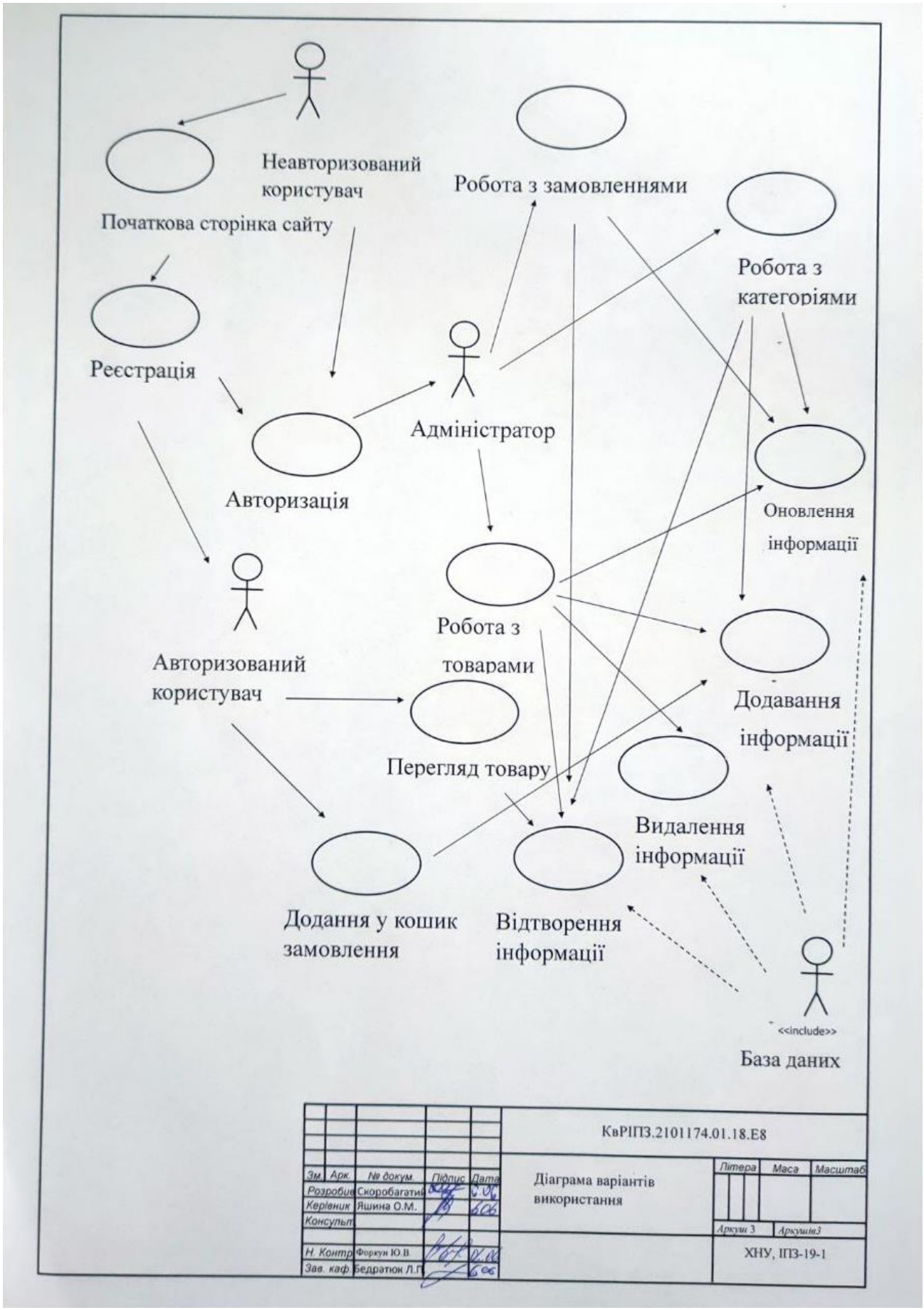
ГРАФІЧНА ЧАСТИНА



					КвРІПЗ.2101174.01.18.Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	Інфологічна модель предметної області «Електронний ресурс для продажу ветеринарних товарів»	Літера / Маса / Масштаб
Розробив		Скоробагатий		30.0		
Керівник		Яшина О.М.		6.06		Аркуш 1 / Аркушів 3
Консульт.						
Н. Контр.		Форкун Ю.В.		6.06	ХНУ, ІПЗ-19-1	
Зав. каф.		Бедратюк Л.П.		6.06		



					КвРІПЗ.2101174.01.18.Е8			
Зм.	Арк.	№ докум.	Підпис	Дата	Функціональна схема інтернет-магазину	Літера	*Маса	Масштаб
Розробив		Скоробагатий		6.06				
Керівник		Яшина О.М.		6.06		Аркуш 2	Аркушів 3	
Консульт.						ХНУ, ІПЗ-19-1		
Н. Контр.		Форкун Ю.В.		6.06				
Зав. каф.		Бедратюк Л.П.		6.06				



					КвРІПЗ.2101174.01.18.Е8		
Зм.	Арк.	№ докум.	Підпис	Дата	Діаграма варіантів використання	Літера	Маса	Масштаб
Розробив		Скоробагатий						
Керівник		Яшина О.М.				Аркуш 3	Аркушів3	
Консульт.								
Н. Контр.		Форкун Ю.В.				ХНУ, ІПЗ-19-1		
Зав. каф.		Бедратюк Л.П.						

СУПРОВІДНІ ДОКУМЕНТИ

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

здобувача вищої освіти
Скоробагатий М.О.
Прізвище, ініціали

факультет ІТ, 4 курс, група ІПЗ-19-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності в Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення 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 плагіат оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії 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 плагіату в текстах робіт.

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

1.06.2023
дата

підпис

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

Дата перевірки:
05.06.2023 08:40:31 EEST

Дата звіту:
05.06.2023 08:41:13 EEST

ID перевірки:
1015421215

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100005589

Назва документа: **КвР_Скоробогатий_плагіат**

Кількість сторінок: **83** Кількість слів: **12543** Кількість символів: **98760** Розмір файлу: **4.01 MB** ID файлу: **1015083739**

18.8%
Схожість

Найбільша схожість: **7.85%** з Інтернет-джерелом (https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etap.

16.6% Джерела з Інтернету | 447 ..... Сторінка 85

6.61% Джерела з Бібліотеки | 95 ..... Сторінка 87

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 1

Mon Jun 05 07:40:41 EEST 2023, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 8%**

ID: 114665 Назва: БКР Електронний ресурс для продажу ветеринарних товарів Додано в БД: 2023-06-05 Автора: Скоробогатий М.О. Керівники: Яшина О.М. к.т.н. доц. Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	73031	1156	5307 (7%)	89 (8%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Скоробагатий Максим Олександрович

Тема Електронний ресурс для продажу ветеринарних товарів

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 99

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі було проведено дослідження та аналіз предметної області, визначено всі функціональні та нефункціональні вимоги. Також був проведений аналіз наявних електронних ресурсів на ринку, розглянуті їх переваги і недоліки, що підкреслило актуальність розробки нового електронного ресурсу. Були вивчені інструменти для втілення задуманих рішень, що призвело до створення електронного ресурсу. Крім того, було проведено тестування, під час якого підтвердилося, що розроблений електронний ресурс працює належним чином і готовий до використання.

2. Висновок про відповідність роботи поставленому завданню Виконано кваліфікаційну роботу з повним дотриманням поставленого завдання та всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі доведено актуальність теми, визначено мету та завдання дипломного проектування. У першому розділі проведено аналіз предметної області, розглянуто існуючі рішення та визначені функціональні і нефункціональні вимоги до розроблюваного програмного забезпечення. У другому розділі підготовлено всі залежності для написання коду та виконано практичну розробку програмних модулів і описано їх особливості. У третьому розділі виконано модульне тестування системи та проведено його у відповідності до функціональних вимог, в результаті було підтверджено коректну роботу
програми.

4. Позитивні сторони роботи Необхідність електронного ресурсу для продажу ветеринарних товарів є актуальною, оскільки сьогоднішнім споживачам потрібно зручне та доступне місце для придбання необхідних продуктів для своїх домашніх тварин. Такий електронний ресурс може надати власникам тварин можливість з легкістю знайти та придбати всі необхідні товари, включаючи корми, ліки, аксесуари та інші ветеринарні продукти. Застосування електронних ресурсів для продажу ветеринарних товарів також сприяє зростанню електронної комерції в цій галузі та полегшує доступ до широкого асортименту товарів для власників тварин. З урахуванням зростаючої популярності онлайн-покупок та зручності використання електронних ресурсів, такий проект є важливим та актуальним.

5. Негативні сторони роботи Електронний ресурс обмежує можливість особистого контакту з продавцем, ускладнюючи спілкування. Клієнти не можуть випробувати товар перед покупкою, що може викликати незадоволення. Також існують ризики пов'язані з доставкою, включаючи затримки, втрати або пошкодження товарів.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення кваліфікаційної роботи відповідає її темі і включає в себе діаграми та рисунки. При цьому пояснювальна записка оформлена з урахуванням вимог, встановлених діючими стандартами.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки, оскільки матеріал пояснювальної записки є структурованим, послідовним, чітким та зрозумілим, що дозволяє чітко уявити вміст проектування. Графічний матеріал надає можливість наочно ознайомитись з деталями системи, що проектується.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота була виконана у повному обсязі, відповідає поставленій задачі і заслуговує на оцінку "задовільно".

РЕЦЕНЗЕНТ Лисенко Сергій Миколайович доктор технічних наук, професор кафедри комп'ютерної інженерії та інформаційних систем

"05" 06 2023 р. (підпис)

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

Підтверджуємо ознайомлення з результатами звіту/звітів подібності щодо роботи, продукованими програмно-технічним засобом (ами) перевірки текстів на плагіат:

Назва: «Електронний ресурс для продажу ветеринарних товарів»

Автор: Скоробагатий Максим Олександрович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Яшина Оксана Миколаївна, канд. техн. наук, доцент
Після аналізу звіту подібності 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 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 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 досягнені. Робота може бути допущена до захисту (наступного року) після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

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

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

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посиланання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 18,8% і адресується до 542 джерел, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 7.06.23

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Оксана ЯШИНА