Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Веб-застосунок для продажу велотоварів
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.190148.01.08.ПЗ

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

Керівник д-р фіз.-мат. наук, проф. _____ Бедратюк Л.П.
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. техн. наук, доцент _____ Гурман І.В
Науковий ступінь, звання Підпис Ініціали, прізвище

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

5 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
02 01 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЕКТ)

Зволянському Олександру Олександровичу
Прізвище, ім'я, по батькові студента

1. Тема роботи (проекту) Веб-застосунок для продажу велотоварів

Керівник роботи (проекту) Бедратюк Леонід Петрович д.фіз.-мат.наук., проф
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)
1.Дослідження предметної області та постановка задачі
2.Проектування програмного забезпечення
3.Програмна реалізація

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Три креслення у форматі А3 (UML-діаграма варіантів використання, ER-діаграма бази даних, UML-діаграма послідовностей)

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Гурман І.В., канд.тех.наук, доцент	3.06	3.06
Антиплагіат	Гурман І.В., канд.тех.наук, доцент	2.06	3.06.

7. Дата видачі завдання « 02 » січня 2023р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи (проекту)	Строк виконання етапів проекту (роботи)	Примітка
1 Ознайомлення з тематикою кваліфікаційної роботи(КвР), визначення та узгодження індивідуальних тем КвР	01.12– 31.12.2022	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (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 з використанням відповідних засобів розробки	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, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки	26.05 – 30.05.2023	
9 Здача КвР на кафедру; підготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР	з 01.06.2023	

Студент _____ О.О. Зволянський
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Л.П. Бедратюк
Підпис Ініціали, прізвище

АНОТАЦІЯ

- Кваліфікаційна робота: «Веб-застосунок для продажу велотоварів».
- Автор роботи: Зволянський О.О.
- Керівник роботи: Бедратюк Л.П.
- Обсяг – 65 с., 24 рис., 3 таблиці, 3 додатки, 42 джерел.

Мета кваліфікаційної роботи: розробка веб-застосунку для продажу велотоварів.

Моя кваліфікаційна робота спрямована на розробку та реалізацію веб-застосунку для продажу велотоварів. Метою цього проекту є створення концепції та розробка веб-застосунку, який забезпечує продаж високоякісних велотоварів широкому колу споживачів.

У межах роботи буде проведений аналіз потреб та вимог споживачів, вивчено ринок та конкурентів, визначено оптимальну технологічну платформу та розроблено привабливий дизайн інтерфейсу магазину.

Головним результатом моєї роботи буде створення та запуск функціонуючого веб-застосунку для продажу велотоварів. Цей застосунок буде забезпечувати зручний та інтуїтивно зрозумілий інтерфейс для замовлення та оплати товарів, що дозволить клієнтам здійснювати покупки швидко та з комфортом.

1.06.2023
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.190148.19.08.ПЗ	Пояснювальна записка	65		
2	А4		Завдання на	1		
			кваліфікаційну роботу			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	12		
5	А3	КвРІПЗ.190148.19.08.Е8	UML - діаграма	1		
			використання			
6	А3	КвРІПЗ.190148.19.08.Е8	ER - діаграма бази даних	1		
7	А3	КвРІПЗ.190148.19.08.Е8	UML - діаграма	1		
			послідовностей			

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.190148.01.08.ВД		
Розроб.		Зволянський О.О		5.06	Веб-застосунок для продажу велотоварів.	Літ.	Арк. 1 / Акрушів 1
Перевір.		Бедратюк Л.П.		5.06			
Н. Контр.		Гурман І.В.		5.06	Відомість документів	ХНУ, ІПЗ-19-1	
Затверд.		Бедратюк Л.П.		5.06			

ЗМІСТ

					КвРІПЗ.190148.01.08.ПЗ	
Змн.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Зволянський О.О.		5.06	Веб-застосунок для продажу велотоварів	Літ. / Арк. 4 / Акрушів 65
Перевір.		Бедратюк Л.П.		5.06		
Н. Контр.		Гурман І.В.		3.06		ХНУ, ІПЗ-19-1
Затверд.		Бедратюк Л.П.		5.06		

ВСТУП

Велоспорт – це не просто хобі, а стиль життя для тисяч людей по всьому світу. Велосипеди та аксесуари до них стають все більш популярними, і з цього приводу з'являється більше інтернет-магазинів, які пропонують різноманітні товари для любителів цього виду спорту.

У рамках кваліфікаційної роботи було створено веб-застосунок для продажу велотоварів, що має на меті забезпечити користувачам можливість знайти і придбати необхідні товари зручно та швидко, не виходячи з дому. Веб-застосунок, який розроблено, має декілька головних функцій, що дозволяють користувачам здійснювати покупки велотоварів.

До основних можливостей застосунку належать пошук товарів за різними параметрами, перегляд товарів та опису їх характеристик, можливість здійснити замовлення та сплату онлайн, а також відслідковування статусу замовлення.

При розробці застосунку була надана особлива увага налагодженню взаємодії з користувачами та забезпеченню максимальної простоти та зручності використання. На сайті розміщено велику кількість відгуків клієнтів, які мають можливість залишити свій коментар та оцінку товару. Це дозволяє іншим користувачам зробити розсудливий вибір, опираючись на думки реальних покупців. Також, веб-застосунок має просту та зрозумілу інтерфейс, що робить процес вибору товару швидким та простим.

Результатом розробки застосунку стало створення місця, де любителі велосипедів зможуть легко та зручно знайти необхідні товари, порівняти їх характеристики та придбати за доступними цінами. Веб-застосунок, що розроблено, відповідає всім сучасним вимогам до інтернет-магазинів та дозволяє забезпечити високий рівень сервісу для користувачів.

У звіті з кваліфікаційної роботи буде розглянуто процес розробки застосунку, від постановки задачі та аналізу вимог до підсистем та модулів, до розробки архітектури та інтерфейсу користувача. Також буде детально описано

використані технології та інструменти, використовувані для розробки засосунку.

Звіт з кваліфікаційної роботи також міститиме аналіз результатів роботи веб-застосунку та його ефективності. Буде проведено порівняння з іншими інтернет-магазинами велотоварів та наведено статистику популярності застосунку серед користувачів.

Веб-застосунок, розроблений в рамках практики, забезпечує користувачам можливість швидко та зручно знайти та придбати необхідні товари, забезпечуючи максимальний рівень безпеки під час онлайн-платежів та надаючи можливість відслідковування статусу замовлення.

Основними функціями застосунку є пошук товарів за різними параметрами, перегляд товарів та їх характеристик, замовлення та онлайн-оплата, а також відгуки клієнтів та можливість їх залишення. Інтерфейс веб-застосунку дуже зручний та простий у використанні, що робить процес вибору товару швидким та легким.

Отже, розроблений веб-застосунок має потенціал стати лідером серед інших інтернет-магазинів велотоварів завдяки своїм функціональним можливостям. Велика кількість відгуків клієнтів та їхній позитивний відгук про продукцію та сервіс роблять застосунок більш привабливим для потенційних покупців. Загалом, розробка цього веб-застосунку є важливим кроком в напрямку розвитку електронної комерції, оскільки він дозволяє ефективно використовувати Інтернет як майданчик для продажу товарів. Він також може допомогти залучити нових клієнтів та збільшити обсяг продажів, що, в свою чергу, сприятиме зростанню бізнесу в галузі велотоварів.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Веломагазин - це спеціалізований роздрібний магазин, який пропонує широкий асортимент велосипедів, велозапчастин, аксесуарів та іншого велосипедного обладнання. Веломагазини зазвичай націлені на задоволення потреб любителів велосипедного спорту, рекреаційних велосипедистів та професіоналів.

У веломагазині можна знайти різні типи велосипедів, такі як шосейні, гірські, міські, BMX, електровелосипеди та інші. Крім того, веломагазини пропонують широкий вибір велозапчастин, які можуть бути потрібні для ремонту та покращення велосипеда, наприклад, гальма, покришки, передачі, педалі та інше. Також в магазинах можна знайти різноманітні аксесуари, включаючи шоломи, велосипедні сумки, світло, замки, комп'ютери та інші предмети, що покращують комфорт та безпеку велосипедиста.

Веломагазини зазвичай мають кваліфікований персонал, який може надати консультації та рекомендації щодо вибору велосипеда або велозапчастини, а також здійснити професійний сервіс та ремонт велосипедів. Деякі веломагазини також пропонують послуги прокату велосипедів, що дозволяє клієнтам випробувати різні моделі перед покупкою або скористатись велосипедом на час подорожі чи відпочинку.

Веломагазини можуть мати фізичні магазини, де клієнти можуть особисто переглянути та вибрати товар, а також онлайн-платформи, де можна зробити замовлення через Інтернет. Онлайн-веломагазини зазвичай пропонують зручність покупок з будь-якого місця та часу, а також широкий вибір товарів та зручні способи оплати та доставки.

Предметна область веб-застосунку для продажу велотоварів включає в себе широкий спектр продуктів, пов'язаних з велосипедами і їх обладнанням. В цю область можна включити такі товари, як велосипеди різних типів та розмірів, комплектуючі, захист та аксесуари для велосипедів, спортивний одяг та взуття для катання на велосипеді, а також спорядження для безпечного зберігання та транспортування велосипедів.

Структура веб-застосунку для продажу велотоварів повинна бути зручною для користувачів і включати такі розділи, як "Каталог товарів", "Кошик", "Кошик замовлень", "Оплата та доставка", "Особистий кабінет", "Контакти". Розділ "Каталог товарів" має бути підібраний і організований таким чином, щоб користувачам було легко знайти необхідний товар за типом велосипеда, брендом, ціновим діапазоном та іншими характеристиками. Розділ "Кошик" має дозволяти користувачам додавати товари до списку покупок та переглядати їх перед оформленням замовлення. Розділ "Кошик замовлень" містить деталі замовлення, які користувач може змінити до оформлення замовлення. Розділ "Оплата та доставка" повинен містити інформацію про способи оплати та доставки товару.

Особистий кабінет користувача містить інформацію про замовлення, історію покупок та можливість редагування профілю. Розділ "Контакти" містить контактну інформацію, таку як адресу та телефони, для зв'язку з підтримкою клієнтів.

Основні функціональні особливості веб-застосунку для продажу велотоварів повинні бути спрямовані на забезпечення зручного та ефективного процесу покупки для клієнтів. Основні функції веб-застосунку можуть включати:

– Пошук товарів: клієнти повинні мати можливість швидко та зручно знайти необхідний товар, використовуючи пошук за категоріями, фільтрами або ключовими словами.

– Каталог товарів: веб-застосунок повинен мати повний та оновлюваний каталог товарів з детальними описами, характеристиками та фотографіями, щоб

клієнти могли зробити обґрунтований вибір.

– Кошик покупок: клієнти повинні мати можливість збирати товари, які вони бажають купити, у віртуальному кошику покупок, змінювати кількість товарів або видаляти їх з кошика перед оформленням замовлення.

– Оформлення замовлення: клієнти повинні мати можливість легко та швидко оформити замовлення, заповнивши форму з необхідною інформацією про адресу доставки, спосіб оплати тощо.

– Оплата: веб-застосунок повинен підтримувати різні способи оплати, такі як оплата кредитною карткою, електронним переказом, готівковий платіж під час отримання товару та інші.

– Доставка: клієнти повинні мати можливість вибрати зручний спосіб доставки, а також вказати адресу доставки. Веб-застосунок повинен надавати інформацію про терміни та вартість доставки.

– Обробка замовлення: веб-застосунок повинен мати систему для обробки замовлень, яка дозволяє відстежувати статус замовлення.

Аналіз предметної області є важливим етапом при розробці інтернет-магазину велотоварів, оскільки він дозволяє зрозуміти потреби та очікування цільової аудиторії і належним чином спроектувати функціонал та інтерфейс системи. Крім того, аналіз допомагає підвищити ефективність роботи магазину та забезпечити задоволення користувачів від покупок у ньому.

При аналізі важливо враховувати тенденції та інновації в галузі велосипедного спорту та велотоварів. Наприклад, електричні велосипеди та розумні велоаксесуари досить популярні в даний час, тому доцільно включити ці категорії товарів до асортименту магазину.

Змістовний аналіз також допомагає визначити особливості ринку велотоварів, зокрема, цінову політику та конкурентну ситуацію. Це дозволяє планувати цінову політику магазину, визначати стратегії просування та реклами, а також уникати помилок, які допускають конкуренти.

Крім того, аналіз потреб користувачів та їхнього поведінки при покупках в інтернет-магазинах дозволяє врахувати їхні очікування щодо зручності та швидкості роботи магазину, процесу замовлення та доставки товарів, можливості оплати та повернення товару. Це допомагає створити зручний та привабливий для клієнтів інтернет-магазин велотоварів.

Для успішного розроблення програмного забезпечення необхідно визначити та деталізувати всі необхідні функції, які задовольнять потреби цього бізнесу.

Один з перших кроків - це розуміння та аналіз цільової аудиторії. Я повинен визначити, кому буде призначений мій веломагазин і які є їхні потреби та вподобання щодо велотоварів. Це допоможе налагодити ефективну комунікацію та розробити стратегії маркетингу, спрямовані на цю цільову аудиторію.

Також, необхідно провести дослідження ринку велосипедів та велотоварів, щоб визначити популярні моделі, бренди та категорії товарів. Це допоможе мені визначити, які товари будуть пропонуватися в моєму веломагазині та які будуть їхні основні характеристики.

Додатково, варто провести дослідження конкурентів, які вже працюють у сфері продажу велотоварів. Я повинен вивчити їхні стратегії, сильні та слабкі сторони, цінову політику та маркетингові стратегії. Це допоможе мені розробити унікальність мого бізнесу та знайти конкурентні переваги.

Для успішного функціонування веломагазину, важливо врахувати різні типи пристроїв, на яких буде доступний сайт. Це включає телефони, планшети, ноутбуки та комп'ютери. Забезпечення зручного відображення та навігації на всіх цих пристроях є критично важливим. Крім того, мій веб-сайт повинен бути сумісним з різними веб-браузерами, щоб користувачі мали можливість відвідувати його з будь-якого браузера за їхнім вибором.

Також є основні проблеми реалізації інтернет-магазину які лежать в наступних аспектах:

– Технічні аспекти: розробка функціональності та налагодження

системи, забезпечення стабільності та швидкості роботи, вибір технологій та інфраструктури, захист від кібератак та зломів безпеки.

– Дизайн та використання: створення зручного та привабливого інтерфейсу для користувачів, адаптація сайту до різних пристроїв та браузерів, забезпечення зручного пошуку та навігації, інтеграція з платіжними системами та забезпечення безпеки платежів.

– Управління контентом та продуктами: ефективне управління каталогом товарів, оновлення та відображення інформації про товари, управління запасами та відстеження замовлень, забезпечення зворотного зв'язку з клієнтами.

– Маркетинг та просування: реклама та просування інтернет-магазину, залучення цільової аудиторії, аналіз конкурентів та розвиток конкурентоспроможності, встановлення ефективних маркетингових стратегій та акцій.

– Клієнтське обслуговування: підтримка клієнтів, відповіді на запитання, обробка скарг і повернень, забезпечення задоволеності та лояльності клієнтів.

– Законодавчі та правові аспекти: дотримання вимог щодо захисту даних та приватності, дотримання законодавства щодо електронної комерції та споживчих прав.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

В ході аналізу різних існуючих розробок для магазинів з продажу велотоварів, я звернув увагу на декілька важливих недоліків, які варто врахувати при розробці власного застосунку. Нижче наведено детальний огляд цих недоліків:

– Обмежена гнучкість: Деякі існуючі розробки мають обмежені можливості налаштування та розширення. Це може ускладнити пристосування магазину до унікальних потреб та вимог бізнесу.

– Складність використання: Деякі розробки можуть бути складними у використанні, особливо для неосвічених користувачів. Це може вимагати додаткового навчання або підтримки, що займає час і кошти.

– Високі вимоги до ресурсів: Деякі розробки можуть бути важкими для сервера або мати високі вимоги до обладнання. Це може призвести до повільної роботи сайту або потребувати додаткових інвестицій в інфраструктуру.

– Обмежені можливості адаптації до змін: Деякі розробки можуть мати обмежені можливості адаптації до швидких змін в електронній комерції або велотоварній галузі. Це може обмежити можливості розвитку та конкурентоспроможності магазину.

Після перегляду декількох вільнодоступних сайтів, що спеціалізуються на продажу велосипедів, я помітив, що більшість з них не має зручного каталогу для перегляду товарів та їх замовлення. Після аналізування деяких з провідних сайтів, я

визначив певні недоліки, які слід виправити на власному сайті. Однак, варто зазначити, що на ринку відсутні монополісти, оскільки у кожного користувача є свої вимоги та переваги. Це призводить до зростання конкуренції та наявності різноманітних пропозицій зі своїми плюсами та мінусами.

Перший сайт який я розглянув був "Вело планета". Зовнішній вид головної сторінки показано на рисунку 1.1.

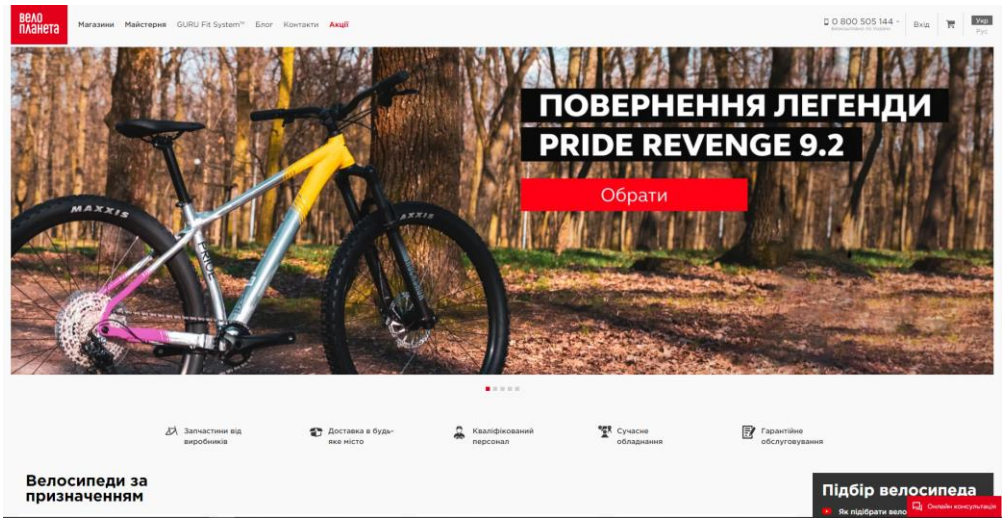


Рисунок 1.1 – Головна сторінка інтернет-магазину "Вело планета" https://veloplaneta.ua/

Недоліком сайту Вело-планета є відсутність зручної розкладки товарів у каталозі. Це може ускладнити навігацію для покупців, особливо якщо магазин має велику кількість товарів. Користувачі можуть мати проблеми з пошуком конкретного товару або виявленням альтернативних варіантів.

Недостатня функціональність фільтрації: Вело-планета також має недолік у функціональності фільтрації товарів. Це зображено на рисунку 1.2. Користувачі можуть бути обмежені в можливості швидкого та точного вибору товарів, використовуючи фільтри за різними параметрами, такими як ціна, розмір, бренд тощо. Це може призвести до затрат часу на пошук та негативного враження від користування сайтом.

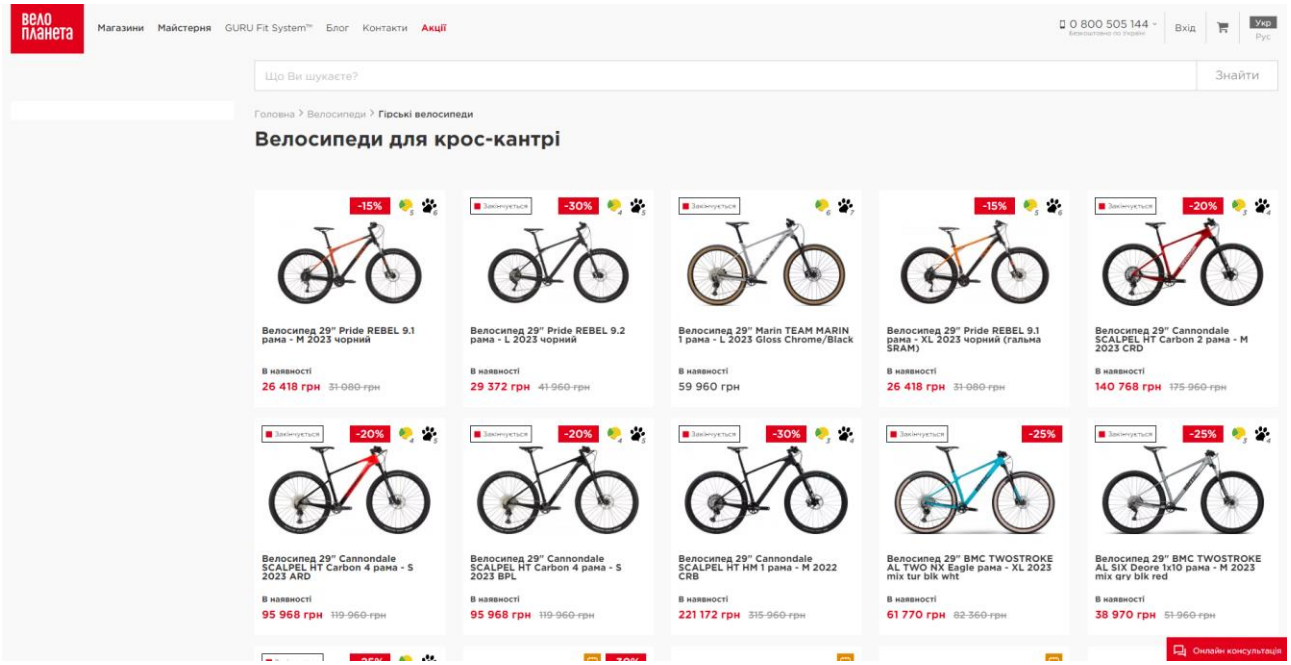


Рисунок 1.2 – Каталог товарів на сайті - https://veloplaneta.ua/

Відсутність інтегрованої системи оплати: Ще одним недоліком Вело-планета є відсутність інтегрованої системи оплати. Це може ускладнити процес замовлення для покупців, оскільки вони повинні використовувати зовнішні платіжні системи або здійснювати оплату поштою чи банківським переказом. Відсутність зручної системи оплати може знизити конверсію замовлень та вплинути на загальний досвід покупців.

Неоптимальна швидкість завантаження: Іншим недоліком Вело-планета є неоптимальна швидкість завантаження сторінок. Довгий час завантаження може

призвести до втрати зацікавленості покупців та зниження їхньої терпимості. Швидкість завантаження є критично важливою для користувачів, і відсутність оптимізації сайту може вплинути на його популярність і відвідуваність.

Відсутність мобільної адаптації: Одним з головних недоліків Вело-планета є відсутність адаптації сайту для мобільних пристроїв. З урахуванням зростаючої кількості користувачів, які використовують мобільні пристрої для покупок, це може обмежити досяжність та зручність використання сайту для багатьох потенційних клієнтів.

Наступний сайт, на який я звернув увагу, є інтернет-магазином велотоварів під назвою "Velosiped" (рисунок 1.3).

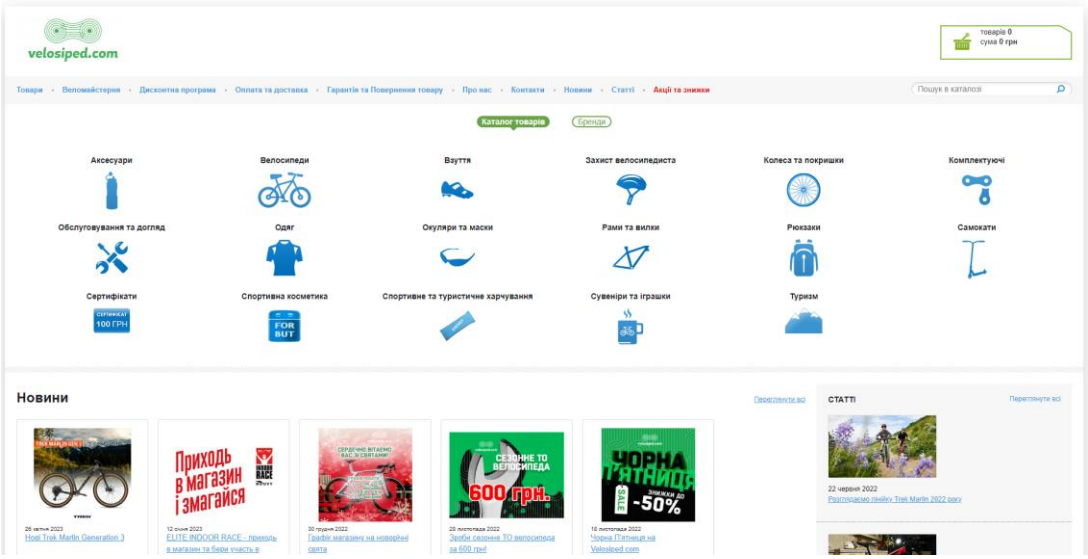


Рисунок 1.3 – Головна сторінка інтернет-магазину " Velosiped "
https://velosiped.com/

Цей сайт пропонує широкий вибір велосипедів, аксесуарів, запчастин та екіпірування для велосипедистів. Однак, слід зазначити, що дизайн цього сайту можна вважати застарілим. Зовнішній вигляд та оформлення "Velosiped" не відповідають сучасним тенденціям у дизайні веб-сайтів. Застосовані стилі, кольорова палітра та композиція елементів веб-сторінок можуть здатися застарілими та нецікавими для сучасних користувачів. Можливо, використання застарілих шрифтів та незручна навігація роблять взаємодію з сайтом менш зручною та привабливою для відвідувачів. Його дизайн і розміщення елементів неадаптовані до різних розмірів екранів мобільних пристроїв, таких як

смартфони та планшети. Це призводить до незручностей під час перегляду сайту на маленьких екранах. Текст може бути занадто малим, кнопки та посилання

можуть бути недоступними для натискання пальцями, а загальна композиція сторінок може виглядати зіпсованою. Користувачам може бути складно знайти та отримати потрібну інформацію, що негативно впливає на їх взаємодію з сайтом та загальний користувальницький досвід. Основні недоліки, які я помітив на цьому сайті, включають:

– Неінтуїтивний дизайн і навігація: Веб-сайт може мати складну структуру і неясну навігацію, що робить його важким у використанні для відвідувачів. Недостатня візуальна організація сторінок та нечітке позиціонування важливих елементів можуть збивати з пантелику та викликати плутанину серед користувачів.

– Погана оптимізація для мобільних пристроїв: Якщо сайт не є адаптивним або не має мобільної версії, це може призвести до проблем з відображенням та навігацією на мобільних пристроях. Користувачі, які відкривають сайт на смартфонах або планшетах, можуть зіткнутися зі складнощами при перегляді та взаємодії з контентом.

– Повільна швидкість завантаження: Якщо сайт має довгий час завантаження сторінок, це може негативно позначитися на користувацькому досвіді. Відвідувачі можуть втратити терпіння та залишити сайт, якщо вони не можуть швидко отримати доступ до інформації або переглянути товари.

– Обмежений вміст та інформація: Якщо сайт має недостатньо докладну або актуальну інформацію про товари, це може ускладнити процес прийняття рішення покупки для відвідувачів. Відсутність детальних описів, технічних характеристик або зображень товарів може призвести до незадоволення і втрати потенційних клієнтів.

– Відсутність інтегрованої підтримки чату або зворотного зв'язку: Якщо сайт не надає можливості зв'язатися з представниками компанії через онлайн-чат або зворотний зв'язок, це може ускладнити спілкування з клієнтами та вирішення їхніх питань або проблем. Відсутність можливості отримати оперативну

допомогу може зменшити задоволення та довіру користувачів.

– Оглядаючи ці недоліки, я вирішив розробити свій власний застосунок для продажу велотоварів, який врахує ці недоліки і надасть клієнтам більш зручний та задовільний досвід покупок.

1.3 Визначення функціональних вимог до програмного забезпечення

Визначення функціональних вимог до програмного забезпечення має на меті описати основні функції та можливості програмного забезпечення, яке буде розроблено для сайту з продажу велотоварів. Визначення функціональних вимог допоможе зрозуміти, які функції повинні бути.

Нижче наведено приклади функціональних вимог до програмного забезпечення для сайту з продажу велотоварів:

– Реєстрація користувачів: Система повинна забезпечувати можливість реєстрації нових користувачів. Користувачі повинні мати можливість створити обліковий запис, вказавши необхідну інформацію, таку як ім'я, електронну пошту та пароль.

– Авторизація користувачів: Після реєстрації користувачі повинні мати можливість виконати вхід на сайт, використовуючи свої облікові дані. Система повинна перевіряти достовірність введених даних та надавати доступ до особистого кабінету користувача.

– Перегляд каталогу товарів: Користувачі повинні мати можливість переглядати каталог велотоварів, де товари розподілені за категоріями та підкатегоріями. Каталог повинен включати зображення, опис, ціни та доступність товарів.

– Пошук товарів: Користувачам повинна бути надана можливість шукати конкретні товари за назвою, категорією, брендом або іншими параметрами. Система повинна виконувати пошук і повертати відповідні результати згідно з введеними критеріями.

– Додавання товарів до кошика: Користувачі повинні мати можливість

додавати товари до свого кошика покупок. Кількість товарів повинна бути нарахована, а також враховані ціни та інші деталі, такі як розмір або колір.

– Оформлення замовлення: Користувачі повинні мати можливість оформити замовлення після додавання товарів до кошика. Система повинна зберігати інформацію про замовлення, включаючи деталі доставки та спосіб оплати.

– Оплата і підтвердження замовлення: Система повинна надати користувачам можливість здійснити оплату за замовлення в онлайн-режимі. Після успішної оплати, користувачам повинно бути надіслано підтвердження замовлення разом з деталями та підсумком.

– Керування адміністратором: Адміністратор повинен мати можливість управляти каталогом товарів, додавати нові товари, видаляти чи редагувати наявні. Також адміністратор повинен мати доступ до замовлень користувачів і можливість керувати їх статусами.

Ці приклади функціональних вимог надані з метою ілюстрації і можуть бути розширені або змінені відповідно до конкретних потреб і вимог проекту. У процесі розробки програмного забезпечення важливо провести детальний аналіз і конкретизувати всі необхідні функції, що допоможе забезпечити успішне виконання проекту. Для досягнення цієї мети, потрібно виявити всі потреби та очікування користувачів, а також врахувати вимоги бізнесу.

Потім необхідно проаналізувати ці вимоги і перевести їх у конкретні функціональні вимоги, які описують поведінку системи і взаємодію з користувачем. Крім того, необхідно врахувати можливі майбутні розширення і функціональні вимоги, щоб забезпечити гнучкість та масштабованість системи.

Усі ці функціональні вимоги повинні бути детально описані, включаючи вхідні та вихідні дані, дії, які система повинна виконувати, та очікувані результати.

Детальний перелік дій, які користувачі можуть здійснювати на ресурсі, наведено на рисунку 1.4.

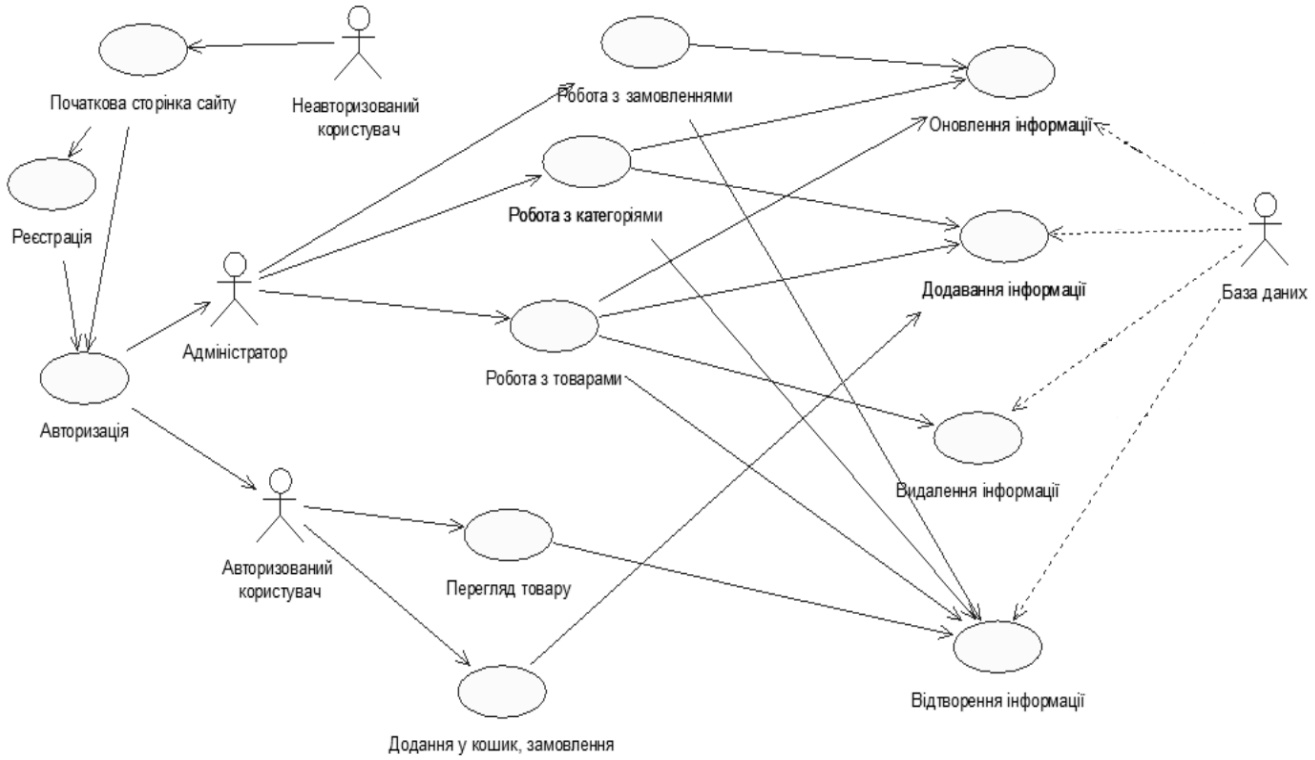


Рисунок 1.4 – Діаграма варіантів використання

Програмне забезпечення надає різноманітні можливості в залежності від ролі користувачів. Таблиця 1.1 відображає варіанти використання для кожної ролі користувача.

Таблиця 1.1 – варіанти використання для кожної ролі

Адміністратор	- Додавання нових товарів - Видалення або редагування товарів - Керування каталогом товарів - Керування замовленями користувачів
Клієнт	- Реєстрація на сайті - Вхід на сайт - Перегляд каталогу товарів - Пошук товарів - Додавання товарів до кошика покупок - Оформлення замовлення - Оплата за замовлення

Продовження таблиці 1.1

Гість	- Перегляд каталогу товарів - Пошук товарів - Реєстрація на сайті - Додавання товарів до кошика покупок

Ця таблиця відображає основні ролі користувачів і їх функціональні можливості. Звичайні користувачі можуть переглядати та шукати товари, додавати їх до кошика та оформляти замовлення. Адміністратори мають додаткові можливості для керування каталогом товарів та замовленнями. Гості мають обмежений доступ і можуть переглядати товари та реєструватися на сайті.

У кінцевому результаті проекту передбачається наявність декількох основних сторінок, включаючи головну сторінку, де буде представлена інформація про магазин, а також каталог товарів, з якого користувачі зможуть переходити на сторінку з деталізованою інформацією про товари. Додатково, передбачається наявність сторінки авторизації/реєстрації, сторінок для редагування облікового запису користувача і сторінки-корзини, де користувачі зможуть переглянути зібрані товари.

Сайт інтернет-магазину велотоварів повинен бути доступний на різних типах пристроїв, включаючи телефони, планшети, ноутбуки та комп'ютери, забезпечуючи зручне відображення та навігацію. Користувачі мають мати можливість легко переглядати та взаємодіяти з вмістом сайту незалежно від пристрою, який вони використовують. Додатково, важливо забезпечити сумісність з різними веб-браузерами, щоб користувачі мали можливість відвідувати сайт з будь-якого браузера за їхнім вибором.

Під час визначення функціональних вимог для програмного забезпечення інтернет-магазину велотоварів, були ідентифіковані основні функції, які повинні бути реалізовані. Одна з них - реєстрація та авторизація користувачів, що дозволяє їм створювати облікові записи та входити до системи для здійснення покупок. Крім того, користувачі повинні мати можливість переглядати каталог

товарів, шукати конкретні товари за різними критеріями, додавати їх до кошика покупок, оформляти та оплачувати замовлення.

Окрім функцій, спрямованих на користувачів, також необхідно передбачити функції для керування адміністратором. Це може включати можливість додавання нових товарів, редагування існуючих, керування запасами, стеження за замовленнями та взаємодію з користувачами.

Ці функціональні вимоги становлять основу для подальшої розробки програмного забезпечення.

1.4 Висновки до першого розділу.

Було проведено детальний опис предметної області, яка включала призначення веб-застосунку, ролі користувачів і можливі операції. Також був проведений аналіз існуючих подібних розробок з урахуванням сучасних потреб, особливо в бізнесі та економіці.

Результатами аналізу були виявлені особливості та обмеження для розроблюваного веб-застосунку. Це включало такі аспекти, як функціональні вимоги, технічні обмеження, безпеку, продуктивність та інші фактори, що впливають на розробку та використання застосунку. Обґрунтування доцільності розробки веб-застосунку було надано на основі виявлених потреб і недоліків існуючих рішень, а також можливостей покращення процесів та результатів в контексті цієї області.

У результаті цього розділу було розроблене технічне завдання (ТЗ), яке включало всі вимоги, обмеження, функціонал та призначення веб-застосунку. ТЗ стало основою для подальшого процесу розробки, визначення архітектури,

вибору технологій та реалізації функціоналу. Описані в цьому розділі деталі допомогли уточнити цілі та напрямки розробки, а також забезпечили належну постановку задачі для подальшого виконання проекту.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз та вибір архітектури веб-застосунку

Термін "клієнт-серверна архітектура" використовується для опису структури або моделі взаємодії між різними комп'ютерними системами або пристроями. У цій архітектурі відбувається обмін даними між двома основними компонентами - клієнтом і сервером (рисунок 2.1).

Рисунок 2.1 – Рисунок архітектури клієнт-сервер

Клієнт - це кінцевий користувач або пристрій, який виконує запити до сервера для отримання певних послуг або ресурсів. Клієнтська частина може бути програмою, веб-браузером або мобільним застосункуом, залежно від конкретного контексту.

Сервер - це централізована система, яка надає послуги або ресурси клієнтам. Сервер приймає запити від клієнтів, обробляє їх і повертає необхідну інформацію або результати.

Така архітектура дозволяє розподіляти завдання між клієнтськими і серверними компонентами, що забезпечує ефективне виконання різних функцій. Клієнт-серверна архітектура широко застосовується в інтернет-технологіях, де клієнти взаємодіють з серверами для доступу до веб-сторінок, електронної пошти, баз даних та інших сервісів.

Проведемо аналіз використання клієнт-серверного типу архітектури в сучасних веб-застосунках. Клієнт-серверна архітектура є однією з найпоширеніших та ефективних архітектур для розробки веб-застосунку.

Клієнт-серверна архітектура передбачає розділення функцій та обов'язків між клієнтською (користувачем) та серверною (системою) частинами. Клієнтська частина забезпечує інтерфейс для взаємодії з користувачем, тоді як серверна частина виконує обробку запитів, зберігає та керує даними, та повертає відповіді клієнту.

Переваги використання клієнт-серверної архітектури включають:

– Розділення відповідальності: Клієнт та сервер виконують свої відповідні завдання, що дозволяє розподілити навантаження та полегшує розвиток та підтримку системи.

– Масштабованість: Клієнт-серверна архітектура дозволяє горизонтальне масштабування, де можна додавати нові сервери для обробки більшого обсягу запитів, що поліпшує продуктивність системи.

– Більша безпека: Серверна частина контролює доступ до ресурсів та забезпечує безпеку даних, що дозволяє зменшити ризик витоку інформації.

– Зручність розробки: Розділення на клієнтську та серверну частину спрощує розробку та тестування, оскільки можна фокусуватись на конкретних аспектах кожної частини.

Однак, використання клієнт-серверної архітектури також має свої недоліки:

– Залежність від мережі: Для взаємодії між клієнтом та сервером потрібне наявність мережі, що може створити проблеми при відсутності зв'язку або недостатній швидкості.

– Збільшений обсяг комунікації: Комунікація між клієнтом та сервером вимагає передачі даних через мережу, що може призвести до збільшення обсягу даних та затримок.

– Більша складність розробки: Розробка та підтримка клієнт-серверної архітектури може бути більш складною, особливо при великому обсязі функцій та вимог до системи.

В моїй кваліфікаційній роботі я використовую C# ASP.NET MVC (Model-View-Controller) (Рисунок 2.1) для розробки веб-застосунку. Цей фреймворк

надає множину інструментів і можливостей для швидкої та ефективної розробки веб-застосунків з використанням мови програмування C#.

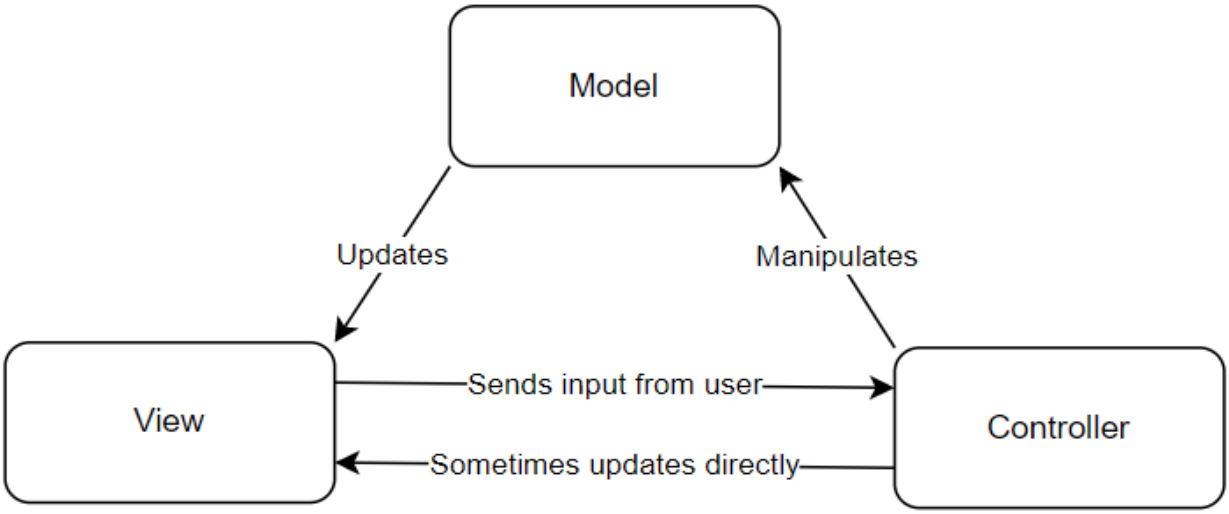


Рисунок 2.1 — Схема MVC архітектури.

ASP.NET MVC базується на архітектурному шаблоні Model-View-Controller, що дозволяє розділити логіку застосунку на три основні компоненти:

– Модель (Model): Це представлення даних та бізнес-логіки в застосунку. Модель відповідає за доступ до даних, валідацію та обробку логіки.

– Представлення (View): Відображає дані користувачу та відповідає за візуальне представлення застосунку. Використовуючи HTML, CSS та розмітку Razor, представлення створює веб-сторінки з відображенням даних.

– Контролер (Controller): Обробляє запити від користувача, взаємодіє з моделлю та відображенням. Контролер приймає запити, обробляє їх, виконує необхідні операції з даними та повертає відповідь відповідно до запиту.

ASP.NET MVC надає розширені можливості для роботи з маршрутизацією, аутентифікацією, авторизацією, кешуванням, управлінням станом, тестуванням та багатьма іншими аспектами розробки веб-застосунків.

Цей фреймворк дозволяє розробникам швидко створювати масштабовані та добро змодельовані веб-застосунки з використанням розподіленої архітектури, що полегшує підтримку та розширення застосунків у майбутньому.

Використання ASP.NET MVC в моїй кваліфікаційній роботі дозволяє мені забезпечити розділення відповідальностей, за допомогою архітектурного шаблону MVC можна чітко визначити ролі та обов'язки кожної частини застосунків. Модель відповідає за логіку обробки даних, представлення - за відображення інформації, а контролер - за керування потоком даних між моделлю та представленням.

ASP.NET MVC надає гнучкість у виборі технологій, бібліотек та інструментів, що дозволяє легко розширювати функціональність застосунку. Наявність різноманітних розширень та пакетів дозволяє швидко додавати нові функції та забезпечувати бажану функціональність.

Фреймворк ASP.NET MVC підтримує тестування коду, що спрощує процес розробки і допомагає забезпечити якість програмного забезпечення. Можна легко створювати автоматизовані тести для перевірки правильності роботи окремих частин застосунку. Надає широкі можливості для реалізації механізмів автентифікації, авторизації та контролю доступу. Це дозволяє забезпечити безпеку застосунку та конфіденційність даних користувачів.

Клієнт-серверна архітектура передбачає розподілення функцій та відповідальностей між клієнтською та серверною частинами застосунку. Клієнтська частина - це інтерфейс, з яким взаємодіє користувач, і вона відповідає за представлення інформації та взаємодію з користувачем. Серверна частина - це зберігання та обробка даних, виконання бізнес-логіки та надання відповідей клієнту.

Вибір клієнт-серверної архітектури для розробки онлайн-застосунку обумовлений декількома факторами:

Інтерактивність: Клієнт-серверна архітектура дозволяє забезпечити інтерактивність та швидку відповідь на дії користувача. Користувачі можуть взаємодіяти з інтерфейсом застосунку, вносити зміни та отримувати оновлені дані без необхідності перезавантаження сторінок.

Розподіл завдань: Клієнт-серверна архітектура дозволяє чітко розподілити завдання між клієнтом та сервером. Це сприяє кращій організації роботи, підтримці паралельної обробки та масштабуванню.

Безпека: Клієнт-серверна архітектура дозволяє реалізувати механізми безпеки, такі як аутентифікація та авторизація, збереження конфіденційності даних та захист від несанкціонованого доступу.

Масштабованість: Клієнт-серверна архітектура дозволяє розподіляти навантаження між клієнтами та сервером, що сприяє масштабованості системи. Можливість додавання додаткових серверів для обробки зростаючої кількості користувачів дозволяє забезпечити стабільну роботу застосунку.

У розробці серверної архітектури на основі ASP.NET MVC можуть бути використані RESTful API та серверний рендерінг

У випадку, коли потрібно забезпечити взаємодію з іншими системами або

Дстворити веб-служби, можна використовувати RESTful API. Цей підхід дозволяє експонувати ресурси застосунку через стандартні HTTP методи, такі як GET, POST, PUT та DELETE, що забезпечує простоту та доступність інтерфейсу.

ASP.NET MVC також підтримує серверний рендерінг, де сторінки генеруються на сервері і відправляються до клієнта вже у вигляді HTML. Цей підхід дозволяє забезпечити кращу швидкодію та SEO-оптимізацію, але може бути менш гнучким у відображенні динамічного контенту.

В проекті можуть бути використані сервіси, які забезпечують окремі функціональні можливості, такі як робота з базою даних, обробка платежів або інтеграція зі зовнішніми сервісами. Це дозволяє розподілити функціональність застосунку на окремі компоненти та спростити його розширення та підтримку.

Ці підходи можуть бути використані в розробці серверної архітектури на базі ASP.NET MVC, залежно від потреб та вимог проекту. Комбінація цих підходів дозволяє створити ефективну та масштабовану архітектуру для веб-застосунку. Враховуючи технічні особливості платформи ASP.NET MVC та вибравши архітектуру MVC для розробки веб-застосунку, можна забезпечити

ефективну роботу застосунку, зручний розширювальний потенціал та високу продуктивність.

Отже, аналіз та вибір архітектури веб-застосунку для продажу велотоварів на платформі ASP.NET MVC є критичним етапом, що враховує бізнес-логіку, масштабованість, безпеку та доступність. Правильно спроектована архітектура дозволяє розробити функціональності, які задовольняють потреби користувачів, забезпечують високу продуктивність та безпеку застосунку, а також дозволяють ефективно розширювати його функціональні можливості у майбутньому.

2.2 Детальне проектування

Метою цього розділу є проведеня детального проектування системи "Інтернет-магазин з продажу велотоварів" з використанням мови програмування C# та фреймворку ASP.NET MVC. Метою цього розділу є визначення основних архітектурних складових, моделей даних, функціональних модулів та інших ключових елементів проекту. В рамках розділу будуть детально розглянуті такі аспекти:

Архітектурні складові:

Визначення структури системи, включаючи розподіл функціональності між клієнтською та серверною частинами, а також взаємодію між ними. Розглянуться можливі варіанти розподілу компонентів та вибір найбільш оптимальної архітектурної моделі.

Моделі даних:

Розробка структури бази даних для зберігання інформації про велотовари, клієнтів, замовлення та інші відповідні сутності. Здійснюється аналіз та проектування зв'язків між таблицями, вибір типів полів та інших характеристик.

Функціональні модулі:

Визначення основних функцій та модулів, які будуть доступні користувачам системи, такі як додавання товарів до кошика, оформлення замовлення, реєстрація та авторизація користувачів, керування обліковим

записом тощо. Для кожного модуля буде проведено детальний аналіз вимог та проектування функціональності.

Інші ключові елементи:

Будуть розглянуті додаткові аспекти проекту системи "Інтернет-магазин з продажу велотоварів". Зокрема, будуть проведені детальні дослідження та проектування таких ключових елементів:

– Визначення методів оплати, їх інтеграція з платіжними шлюзами та розробка відповідних функціональних модулів для обробки платежів та підтвердження операцій.

– Розгляд можливостей інтеграції з зовнішніми сервісами, наприклад, з системами доставки товарів, системами розсилки повідомлень або соціальними мережами. Визначення способів обміну даними та розробка необхідних модулів для взаємодії з цими сервісами.

– Аналіз потенційних загроз безпеці, визначення методів автентифікації та авторизації користувачів, розробка механізмів шифрування та захисту конфіденційної інформації.

– Розгляд адаптивного дизайну та розробка інтерфейсу, що коректно відображається на різних пристроях (наприклад, комп'ютерах, планшетах, мобільних телефонах) і працює на різних браузерах (наприклад, Google Chrome, Mozilla Firefox, Safari).

У результаті вивчення цього розділу буде сформовано детальне проектування системи "Інтернет-магазин з продажу велотоварів". Це проектування буде використовуватись як основа для подальшої реалізації та розробки веб-застосунку.

Детальне проектування охоплюватиме всі аспекти системи, включаючи архітектурні складові, моделі даних, функціональні модулі, систему оплати, інтеграцію зі сторонніми сервісами, захист даних та безпеку, підтримку різних пристроїв та браузерів. Воно забезпечить ясне розуміння вимог проекту та визначить кроки, необхідні для успішної реалізації та розробки системи "Інтернет-магазин з продажу велотоварів"

2.3 Вибір системи керування базами даних

У цьому розділі проводиться детальний аналіз схеми бази даних проекту, включаючи таблиці, поля, типи даних та відношення між ними. Описання бази даних зазвичай представляється у нормалізованій формі та може бути візуалізовано за допомогою ER-діаграми або діаграми класів.

Кожне поле має зазначений тип даних, який відповідає його характеристикам (наприклад, рядок, число, дата тощо). Також надаються пояснення до кожного поля, які розкривають його призначення та значення.

Важливо також вказати відношення між таблицями, наприклад, зв'язки один до одного, один до багатьох або багато до багатьох. Це допомагає встановити зв'язки між сутностями та визначити, як дані взаємодіють між собою.

Для більш зрозумілого представлення структури даних та моделі бази даних рекомендується використовувати ER-діаграму або діаграму класів. Ці діаграми графічно відображають таблиці, поля та їх взаємозв'язки, що дозволяє легко сприймати структуру бази даних та її модель.

Загальна мета цього розділу полягає в тому, щоб уявити структуру бази даних проекту та визначити необхідні таблиці, поля та їх відношення для забезпечення ефективного зберігання та обробки даних.

Розглянемо типи баз данних які можна використати з використанням фреймворку ASP.NET MVC:

– Реляційна база даних (RDBMS): Реляційні бази даних є одними з найпоширеніших типів баз даних. Вони забезпечують організацію даних у вигляді таблиць зі структурованими зв'язками між ними. Для роботи з реляційними базами даних у C# зазвичай використовується мова SQL (Structured Query Language).

– Об'єктно-реляційна база даних (ORDBMS): Об'єктно-реляційні бази даних поєднують переваги реляційних та об'єктно-орієнтованих підходів до зберігання та обробки даних. Вони дозволяють працювати з об'єктами замість таблиць, що полегшує моделювання складних структур даних.

– NoSQL база даних: NoSQL бази даних відрізняються від реляційних баз тим, що вони не використовують SQL та не вимагають фіксованої схеми даних. Це дозволяє більшу гнучкість та швидкодію при роботі з великими обсягами неструктурованих даних, таких як документи, графи, ключ-значення тощо.

У виборі конкретного типу бази даних варто враховувати потреби проекту, обсяг та структуру даних, а також вимоги до продуктивності та масштабованості системи. ASP.NET MVC забезпечує гнучкість у виборі бази даних, дозволяючи підключатися до різних типів баз даних за допомогою ORM-фреймворків, таких як Entity Framework.

Вибір Microsoft SQL Server для проекту на C# з використанням фреймворку ASP.NET MVC має кілька переваг. По-перше, ця база даних оптимально працює з іншими продуктами Microsoft, такими як Visual Studio та .NET Framework, що спрощує розробку, тестування та впровадження програмного забезпечення. По-друге, вона ідеально поєднується з Entity Framework, що дозволяє зручно взаємодіяти з базою даних, використовуючи об'єктно-орієнтований підхід, та забезпечує високу продуктивність та швидкий доступ до даних. Крім того, Microsoft SQL Server гарантує надійність, стабільність та можливості масштабування для обробки зростаючого обсягу даних та навантаження. І нарешті, вона забезпечує різноманітні механізми безпеки, що дозволяють зберігати дані конфіденційними та цілими.

Вибір реляційної бази даних Microsoft SQL Server є обґрунтованим рішенням, яке забезпечить надійну та ефективну роботу з даними у розроблюваному веб-застосунку.

2.4 Проектування серверної частини веб-застосунку

У цьому розділі я здійснюю налаштування серверу для реалізації веб-застосунку. Основні кроки цього процесу включають створення форм для відправки даних на сервер та реалізацію сценаріїв, які обробляють отримані дані.

Перш за все, я розробляю форми, за допомогою яких користувачі зможуть взаємодіяти з веб-застосунком. Ці форми дозволяють збирати дані, які клієнт вводить, і передавати їх на сервер для подальшої обробки. Для цього використовуються вбудовані засоби фреймворку ASP.NET MVC, які дозволяють легко створювати форми з необхідними полями та кнопками.

Далі, я реалізовую сценарії, що обробляють дані, надіслані з клієнта. Ці сценарії виконуються на сервері і включають в себе перевірку та валідацію введених даних, збереження їх у базі даних, виконання необхідних операцій та генерацію відповідей для клієнта. Для цього використовуються контролери та дії (actions) фреймворку ASP.NET MVC, які дозволяють організувати логіку обробки запитів та відправку результатів назад до клієнта.

В процесі проектування серверної частини веб-застосунку, я здійснюю налаштування сервера для ефективної реалізації функціоналу застосунку. Це включає кроки, такі як створення форм для передачі даних на сервер і реалізацію сценаріїв для обробки цих даних.

Одним з перших кроків є розробка форм, які дозволяють користувачам взаємодіяти з застосунком. Ці форми забезпечують можливість збору даних, які вводить користувач, і їх передачу на сервер для подальшої обробки. Для цього використовуються інструменти, що надаються фреймворком ASP.NET MVC, які спрощують процес створення форм з необхідними полями і кнопками.

Після цього, я реалізовую сценарії, які обробляють дані, що надходять з клієнта. Ці сценарії виконуються на сервері і включають перевірку та валідацію введених даних, їх збереження у базі даних, виконання необхідних операцій та генерацію відповідей для клієнта. Для цього використовуються контролери та дії (actions) фреймворку ASP.NET MVC, які дозволяють організувати логіку обробки запитів та відправку результатів назад до клієнта.

Основна мета серверної частини веб-застосунку полягає у прийманні та повній обробці даних, які надходять від клієнтів. Цей процес вимагає належного налаштування сервера та розробки відповідних сценаріїв для ефективної обробки цих даних. По завершенні цього розділу, серверна частина веб-

застосунку буде належним чином налаштована і готова приймати та обробляти дані від клієнтів.

2.5 Аналіз та вибір технологій для розробки клієнтської частини веб-застосунку

У цьому підрозділі проводиться огляд і аналіз різних технологій та засобів, необхідних для розробки веб-застосунку. Вивчається мова розмітки HTML, складові DHTML, каскадні таблиці стилів (CSS), об’єктна модель документа (DOM), методи створення JavaScript-сценаріїв, керування елементами сторінок на основі DOM, використання бібліотеки jQuery, мова програмування PHP та відповідні фреймворки, методи взаємодії веб-сценаріїв та СКБД, мова запитів до баз даних MySQL, підхід до побудови інтерактивних інтерфейсів користувача з використанням технології Ajax, а також протоколи та формати обміну даними в мережі Інтернет. В результаті аналізу цих технологій і засобів буде обрано найбільш відповідні для реалізації поставлених завдань веб-застосунку.

Під час аналізу технологій і методів реалізації веб-застосунку, була висвітлена значущість HTML5. HTML5 є однією з ключових технологій, що була розглянута. Ця технологія надає потужні інструменти для створення розмітки та структури веб-застосунків, включаючи нові теги, покращені форми, мультимедійні можливості, графічні ефекти та багато іншого. Використання HTML5 дозволяє розробникам створювати сучасні, інтерактивні та мобільно-дружні веб-застосунки, які працюють ефективно на різних пристроях і в різних браузерах.

Ще однією важливою технологією, яка була розглянута, є JavaScript. JavaScript - це мова програмування, яка дозволяє створювати динамічні та інтерактивні елементи на веб-сторінках. Вона дозволяє розробникам контролювати поведінку елементів сторінки, обробляти події, взаємодіяти з користувачем та виконувати різноманітні дії на сторінці.

Ще одним важливим аспектом веб-розробки є технологія Ajax (Asynchronous JavaScript and XML). Ajax дозволяє веб-сторінкам асинхронно обмінюватися даними з сервером без необхідності перезавантаження сторінки. Це дозволяє створювати більш швидкі та плавні веб-застосунки, які реагують на дії користувача миттєво. Використовуючи Ajax, розробники можуть оновлювати окремі частини сторінки, відправляти та отримувати дані в фоновому режимі і динамічно змінювати вміст сторінки без перезавантаження.

JavaScript і Ajax використовуються в комбінації з HTML та CSS для створення багатофункціональних та інтерактивних веб-застосунків. Вони дозволяють розробникам створювати багатошарові інтерфейси, реалізовувати асинхронну взаємодію з сервером, валідувати дані та багато іншого. Завдяки JavaScript і Ajax веб-застосунки стають більш динамічними, зручними та користувацьки орієнтованими.

Окрім базових можливостей JavaScript і Ajax, також було розглянуто кілька популярних бібліотек, які допомагають спростити розробку веб-застосунків. Одна з них - це бібліотека jQuery. jQuery є широко використовуваною інструментальною бібліотекою JavaScript, яка спрощує маніпулювання HTML-документами, обробку подій, анімацію, взаємодію з сервером і багато іншого. Вона забезпечує простий та елегантний синтаксис, що дозволяє розробникам швидко та ефективно виконувати різноманітні завдання веб-розробки.

Іншою популярною бібліотекою є React. React - це бібліотека JavaScript для розробки інтерфейсів користувача. Вона дозволяє розробникам створювати компонентну структуру веб-застосунків, яка полегшує управління станом, перевикористання коду та розширення функціональності. React базується на концепції віртуального DOM, що дозволяє ефективно оновлювати лише необхідні частини сторінки при зміні стану.

Додатково, було розглянуто інші бібліотеки та фреймворки, такі як Angular, Vue.js, Ember.js та інші, які також надають зручні інструменти для

розробки веб-застосунків. Вибір конкретної бібліотеки або фреймворка залежить від вимог проекту, досвіду розробника та персональних уподобань.

Вибір технологій та бібліотек залежить від потреб проекту, вимог до функціональності, відповідальності, складності, масштабованості та інших факторів. Важливо врахувати як поточні потреби, так і майбутні перспективи розвитку проекту. Правильний вибір технологій і методів реалізації веб-застосунків є важливим етапом, оскільки він впливає на ефективність, швидкість розробки, підтримку та масштабованість проекту. Ретельний аналіз і обґрунтування вибору технологій допомагає забезпечити успішну реалізацію веб-застосунку та задоволення потреб користувачів.

2.6 Проектування інтерфейсу користувача

У цьому розділі я зосереджуюся на проектуванні структури веб-застосунків, що включає організацію даних та ієрархію матеріалів, яка потрібна для їх представлення кінцевому користувачеві. Процес створення структури застосунку можна розділити на два основні етапи: структуризацію інформації та візуальне представлення структури.

Першим етапом є збір вимог і аналіз, де я вивчаю потреби користувачів, їхні очікування та завдання, які вони повинні виконувати за допомогою інтерфейсу. Цей етап допомагає зрозуміти контекст використання застосунку та визначити його основні функції.

Другий етап - це розробка інформаційної архітектури, де я організовую структуру застосунку, його ієрархію та навігацію. Це включає класифікацію та групування розділів та функціональних блоків, визначення зв'язків між ними та створення схеми навігації.

Наступним етапом є створення макетів інтерфейсу, де я візуалізую структуру та розміщення елементів на сторінках застосунку. Це може включати створення провідного макету, який показує загальний вигляд сторінок, а також детальніші макети окремих блоків чи компонентів. Макет подано на рисунка 2.2.

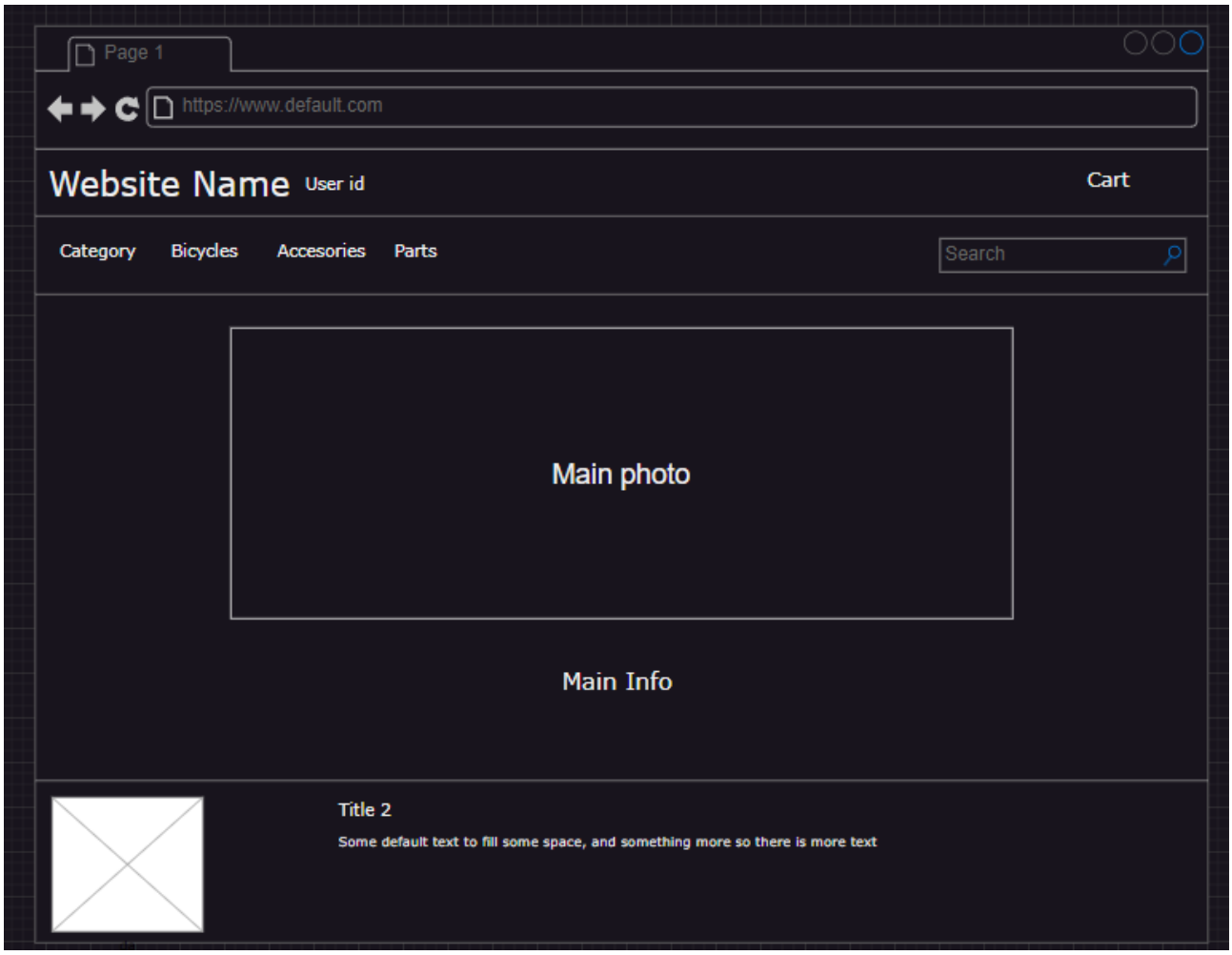


Рисунок 2.2 – Макет головної сторінки

Головна сторінка має наступну будову:

– Шапка (Header): Включає логотип або назву сайту, основне меню навігації, пошукову стрічку та інші важливі посилання або функціональні елементи.

– Банер або зображення: Це велике зображення або банер, яке привертає увагу користувачів та може містити ключові повідомлення, акції або важливу інформацію.

– Вміст: На головній сторінці можуть бути розміщені короткі огляди або віджети з найбільш важливими або популярними матеріалами, новинами, акціями або продуктами.

– Секції або блоки: Головна сторінка може містити різні секції або блоки, які відображають різноманітний контент, такий як вибір товарів, рекомендації, відгуки користувачів, огляди, блоги тощо. Кожен блок може мати свою

заголовок, текст, зображення та посилання.

– Статичні або навігаційні елементи: Головна сторінка може також містити навігаційні елементи, які допомагають користувачам швидко перейти до певних сторінок або розділів сайту, наприклад, посилання на категорії товарів, послуги, контакти або додаткові розділи.

– Підвал (Footer): Включає додаткову навігацію, посилання на соціальні мережі, контактну інформацію, правові заяви та іншу додаткову інформацію.

Це загальна структура, і конкретний вигляд головної сторінки може варіюватися в залежності від конкретних потреб та дизайну веб-ресурсу.

Структура та дизайн інших сторінок ресурсу подібні до макету головної сторінки. Кожна сторінка має включати шапку з логотипом або назвою сайту, основне меню навігації та пошукову стрічку для зручної навігації та пошуку інформації.

Далі натиснувши на іконку товар користувача перекине на сторінку цього товару. Макет сторінки товару подано а рисунку 2.3 Сторінка товару мого веб-застосунку має наступну структуру, яка сприяє зручному та інформативному представленню товару користувачам:

Зображення товару: На початку сторінки знаходиться основне зображення товару, яке найчастіше є його фотографією. Зображення може бути достатньо великим, щоб користувачі мали можливість детально роздивитися товар.

Інформація про товар: Під зображенням товару розташована інформація про нього, яка може включати назву товару, його опис, характеристики (наприклад, розмір, матеріал, колір), ціну, наявність на складі та інші важливі деталі.

Кнопки дій: На сторінці товару можуть бути розміщені кнопки або посилання для виконання певних дій, таких як додавання товару до кошика, оформлення замовлення або додаткові опції взаємодії з товаром, наприклад, додати до списку бажань чи порівняти з іншими товарами.

Додаткова інформація: Нижче можуть бути розміщені додаткові вкладки або секції з більш детальною інформацією про товар, як-от технічні

характеристики, відгуки та рейтинги, рекомендації щодо використання або догляду за товаром, інструкції та інші додаткові матеріали.

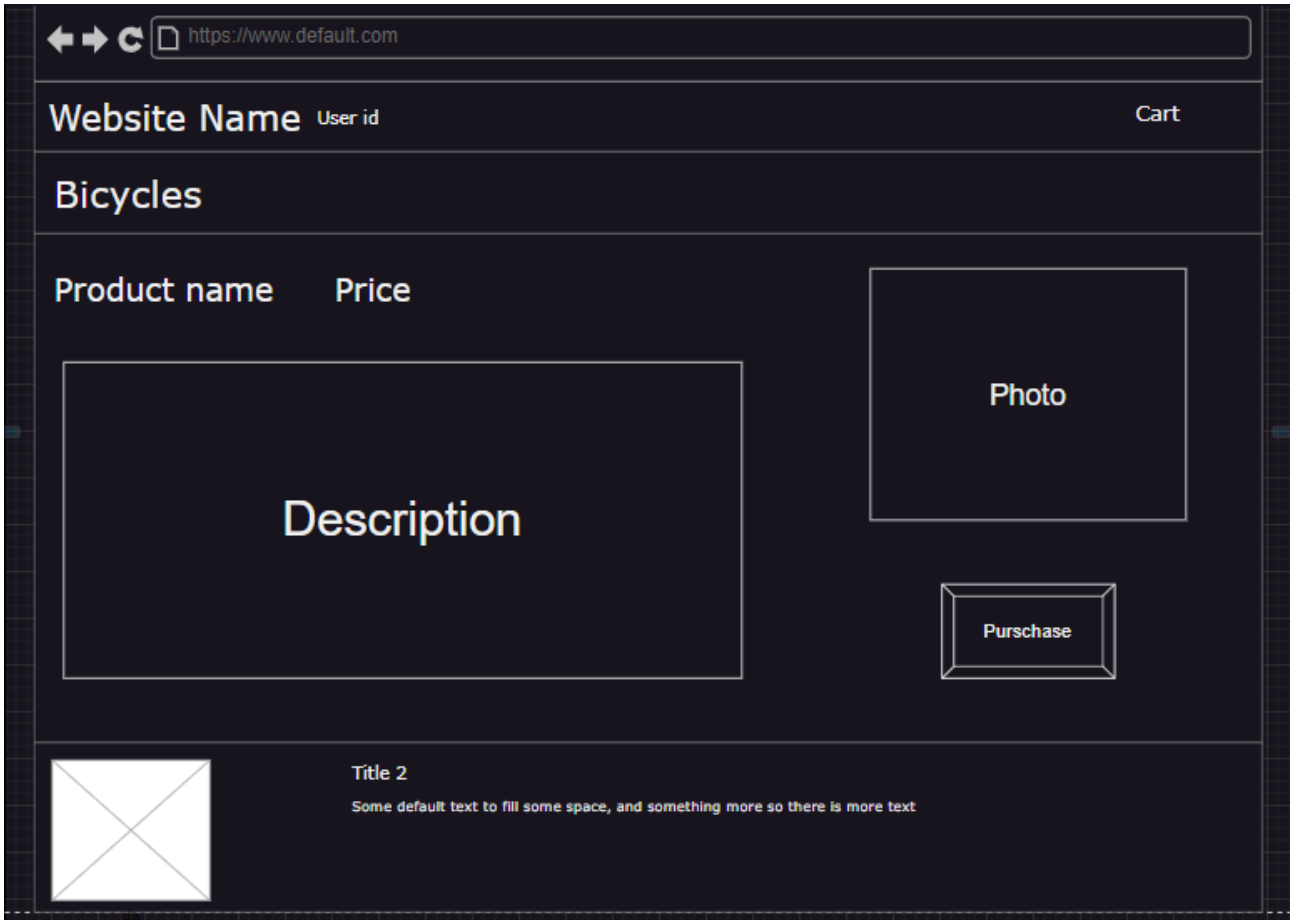


Рисунок 2.3 – Макет сторінки перегляду товару

Ця структура сторінки товару допомагає користувачам отримати повну та зрозумілу інформацію про товар, а також здійснити необхідні дії для придбання або подальшої взаємодії з ним.

Також на інших сторінках можуть бути розміщені банери або зображення, які акцентують увагу користувачів та містять важливу інформацію або повідомлення. Вміст сторінок може варіюватися в залежності від їх призначення, але може включати огляди товарів, додаткові послуги, блогові пости, контактну інформацію та інші корисні матеріали для користувачів. Загалом, дизайн інших сторінок ресурсу використовує схожі елементи та структуру, що допомагає забезпечити єдність та зручність використання для користувачів. Проте, кожна сторінка може мати унікальний контент та елементи, що відповідають її конкретній меті та функціональності.

Після створення макетів переходимо до етапу дизайну інтерфейсу, де вирішується питання кольорової гами, використання шрифтів, графіки та інших візуальних елементів Один з ключових аспектів дизайну інтерфейсу - це врахування брендового стилю компанії або проекту. Наша мета полягає в тому, щоб створити спільність та цілісність у дизайні, що відображатиме ідентичність та цінності бренду. Це може включати в себе використання певних кольорів, шрифтів або графічних елементів, що асоціюються з брендом.

Окрім цього, важливим аспектом є зручність сприйняття дизайну користувачами. Я прагнему створити інтерфейс, який буде привабливим та легким у використанні. Одним з варіантів, який я обрав, є використання чорно-жовтої кольорової гами. Це поєднання кольорів, яке широко використовується на багатьох популярних веб-сайтах, і має свою психологічну впливовість. Чорний колір вказує на стильність та сучасність, а жовтий - на енергію та акцентує увагу. Використання чорно-жовтого інтерфейсу має свої переваги. По-перше, він створює контраст та виділяє ключові елементи інтерфейсу, допомагаючи користувачам швидко зорієнтуватись та зосередитись на важливій інформації. Крім того, такий колірний підхід викликає асоціацію з вже знайомими користувачам сайтами, які використовують подібну кольорову гаму, що сприяє відчуттю знайомства та комфорту.

Велика увагу приділяється візуальному представленню структури. Це охоплює колірні, графічні та стильові рішення. Коректно підібраний колірна схема та графічні елементи допомагають створити привабливий та зручний для використання веб-застосунок. Крім того, стильове оформлення додає веб-застосунку професійного вигляду та виражає його унікальний характер.

Під час структуризації інформації я класифікую та групую розрізнені матеріали, щоб об'єднати їх в категорії або рубрики. Це допомагає користувачеві легше зорієнтуватися в веб-застосунку та швидше знайти потрібну інформацію. Особлива увага приділяється присвоєнню зрозумілих назв категоріям або рубрикам, щоб користувачі змогли швидко розуміти їх зміст.

Крім того, при проектуванні макета веб- застосунку і дизайну, я розглядаю такі аспекти, як навігація та взаємодія користувача з інтерфейсом. Навігація повинна бути логічною та інтуїтивно зрозумілою, щоб користувачі з легкістю могли переміщатися по різних сторінках та розділах застосунку. Елементи інтерфейсу, такі

як кнопки, посилання та меню, мають бути розміщені зручно та зрозуміло, забезпечуючи зручну навігацію.

Дизайн веб- застосунку - це багатогранний процес, який включає в себе низку важливих аспектів, що сприяють створенню естетично привабливого та зручного для користування інтерфейсу. Одним з ключових елементів дизайну є вибір підходящих шрифтів, які забезпечують читабельність тексту та створюють відповідну атмосферу. Правильно підібраний шрифт може підкреслити стиль та тон застосунку.

У дизайні також велике значення має розміщення зображень. Вони можуть використовуватись для створення настрою, передачі інформації або привертання уваги користувачів. Важливо розміщувати зображення таким чином, щоб вони гармонійно вписувалися у загальний дизайн і не заважали сприйняттю контенту.

Простір на сторінці також має велике значення. Використання відповідного простору дозволяє структурувати інформацію, покращує сприйняття контенту та робить інтерфейс більш зрозумілим. Важливо збалансувати кількість тексту, графічних елементів та порожнього простору для створення приємного вигляду сторінки.

Після завершення цього етапу розробки веб- застосунку, ми матимо створений макет з визначеною структурою та дизайном. Цей макет буде слугувати основою для подальшої реалізації та розробки функціоналу веб-застосунку. Він дозволить нам візуалізувати кінцевий результат та перевірити відповідність дизайну задуму та очікуванням.

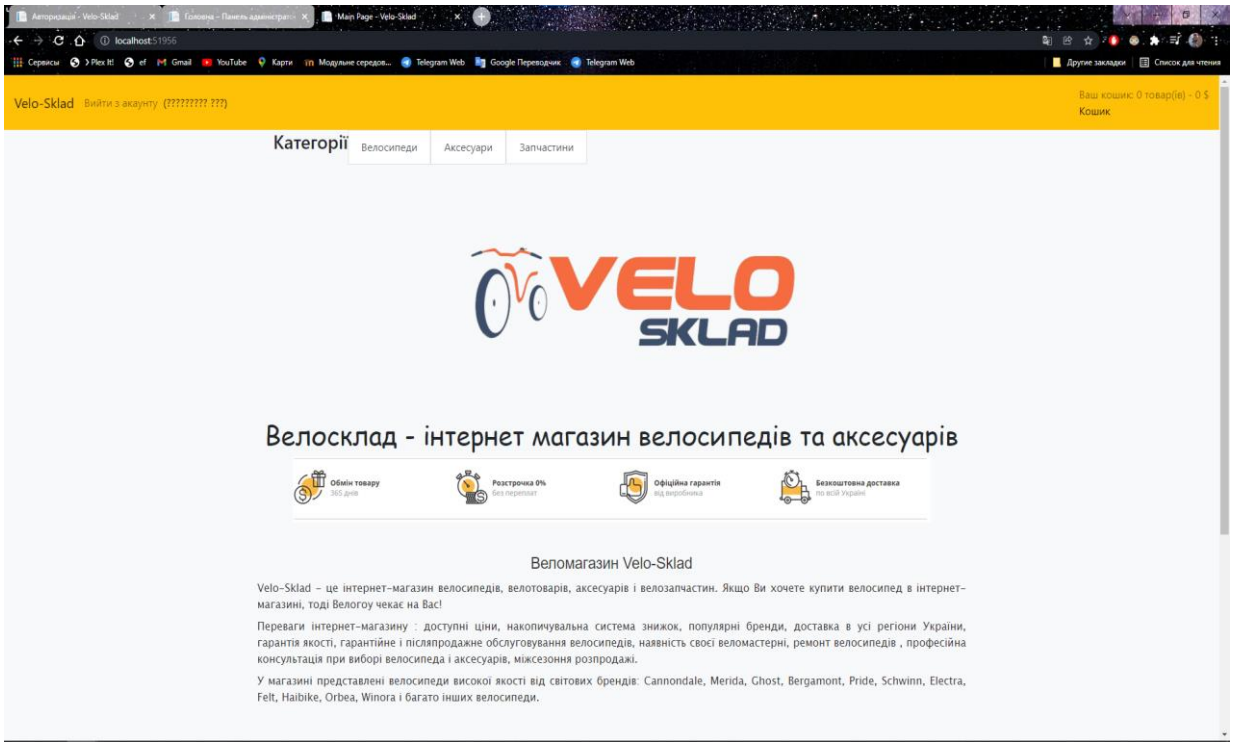


Рисунок 2.4 – Головна сторінка сайту

Це буде важливим кроком для подальшої реалізації програмного забезпечення, оскільки дозволить це визначити, як інформація буде представлена користувачам і як вони будуть взаємодіяти з застосунку.

2.7 Висновки до другого розділу

В результаті проведеного розділу "Проектування веб- застосунку " було виконано аналіз та здійснено вибір оптимальної архітектури для розроблюваного застосунку. Було проведено проектування структури бази даних, включаючи вибір та налаштування необхідних таблиць та зв'язків між ними. Крім того, було розроблено серверну та/або клієнтську частини застосунку з урахуванням вимог та функціональності.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Розробка бази даних

У цьому розділі здійснюється детальний опис процесу створення, налаштування та інтеграції бази даних у веб-застосунок, з використанням технологій ASP.NET та Microsoft SQL Server. На цьому етапі розробки важливо створити структуру бази даних, включаючи таблиці, зв'язки між ними та поля для зберігання потрібної інформації.

Під час розробки бази даних на основі ASP.NET і Microsoft SQL Server використовуються інструменти і функціональні можливості, які надають ці технології. ASP.NET забезпечує зручні інструменти для роботи з базою даних, такі як Entity Framework, який дозволяє взаємодіяти з базою даних за допомогою об'єктно-орієнтованого підходу. Microsoft SQL Server, у свою чергу, є потужною системою керування базами даних, яка надає широкі можливості для створення, модифікації та оптимізації бази даних.

Крім створення самої бази даних, також важливо врахувати керування версіями бази даних та здійснювати оновлення при необхідності. Це допомагає забезпечити стабільну та сумісну роботу бази даних з веб-застосунком під час змін і вдосконалення.

Загалом, розробка бази даних є важливим етапом в процесі створення веб-застосунку. Використання технологій ASP.NET і Microsoft SQL Server дозволяє ефективно керувати даними, забезпечувати їх безпеку та забезпечувати потрібну функціональність для веб-застосунку.

Основна функція бази даних цього застосунку включає в себе збереження інформації про користувачів, товар,категорії, замовлення, все це подано на рисунках 3.1.1.-.3.1.3.

Першою розглянемо таблицю "tblCategories" яка містить деталі про кожну категорію на веб-сайті. Ця таблиця забезпечує зберігання інформації, пов'язаної з категоріями, та надає необхідну структуру для організації та відображення цих категорій на веб-сайті.

У цій таблиці можна знайти різні поля, які містять деталі про категорії. Наприклад, такі поля можуть включати унікальний ідентифікатор категорії, назву категорії, опис, дату створення, а також інші атрибути, які допомагають ідентифікувати та характеризувати кожну категорію.

Поле	Тип	Опис
Id	int	Унікальний ідентифікатор в базі даних
Name	Nvarchar(50)	Назва категорії
Slug	Nvarchar(50)	Унікальна назва категорії
Sorting	int	Унікальний ідентифікатор за яким вони сортируються по списку категорій

Рисунок 3.1.1. - Таблиця "tblCategories"

Наступна таблиця "tblOrderDetails" містить детальну інформацію про замовлення на веб-сайті. Ця таблиця забезпечує зберігання інформації, пов'язаної з кожним окремим замовленням, та надає необхідну структуру для відстеження та обробки замовлень. Ця таблиця включає в себе номер замовлення, код користуача та кількість продукції замовленої ним.

Поле	Тип	Опис
Id	int	Унікальний ідентифікатор в базі даних
OrderId	int	Номер замовлення
UserId	int	Унікальний ідентифікатор користувача
ProductId	int	Унікальний ідентифікатор продукту
Quantity	int	Кількість продукції

Рисунок 3.1.2. – Таблиця tblOrderDetails

І наступна таблиця на яку потрібно звернути увагу є tblProducts. Таблиця "tblProducts" містить інформацію про кожен товар, що присутній на веб-сайті. Ця таблиця дозволяє зберігати та управляти детальною інформацією про кожен товар, що продається, і забезпечує необхідну структуру для ефективного відображення та управління товарами.

У таблиці "tblProducts" зберігаються різні атрибути товарів, включаючи назву товару, унікальний код, деталі товару, ціну, назву категорії, а також назву картинки, яка ілюструє товар. Кожен товар має унікальний код, який використовується для ідентифікації товару в системі та його взаємодії з іншими модулями веб-застосунку. Назва товару дозволяє ідентифікувати та відрізняти товари один від одного, а деталі товару надають додаткову інформацію про характеристики та особливості товару. Ціна товару вказує на його вартість, що дозволяє користувачам зрозуміти його вартість перед здійсненням покупки. Категорія товару вказує на приналежність товару до певної групи або рубрики, що спрощує навігацію та пошук товарів за певними критеріями. Назва картинки вказує на зображення товару, яке може бути відображено на веб-сторінці для привертання уваги користувачів.

Поле	Тип	Опис
Id	int	Унікальний ідентифікатор в базі даних
Name	Varchar(50)	Назва товару
Slug	Varchar(50)	Унікальна назва товару
Description	Varchar(MAX)	Деталі товару
Price	Numeric(18,2)	Ціна товару
CategoryName	Varchar(50)	Назва категорії
CategoryId	Int	Унікальний ідентифікатор категорії
ImageName	Varchar(100)	Назва картинки

Рисунок 3.1.3. – Таблиця tblProducts

Створення таблиці "tblProducts" забезпечує зручний та організований спосіб зберігання та управління інформацією про кожен товар на веб-сайті.

Щоб розглянути зв’язки між таблиця було створенно діаграму БД (рисунок 3.1.4).

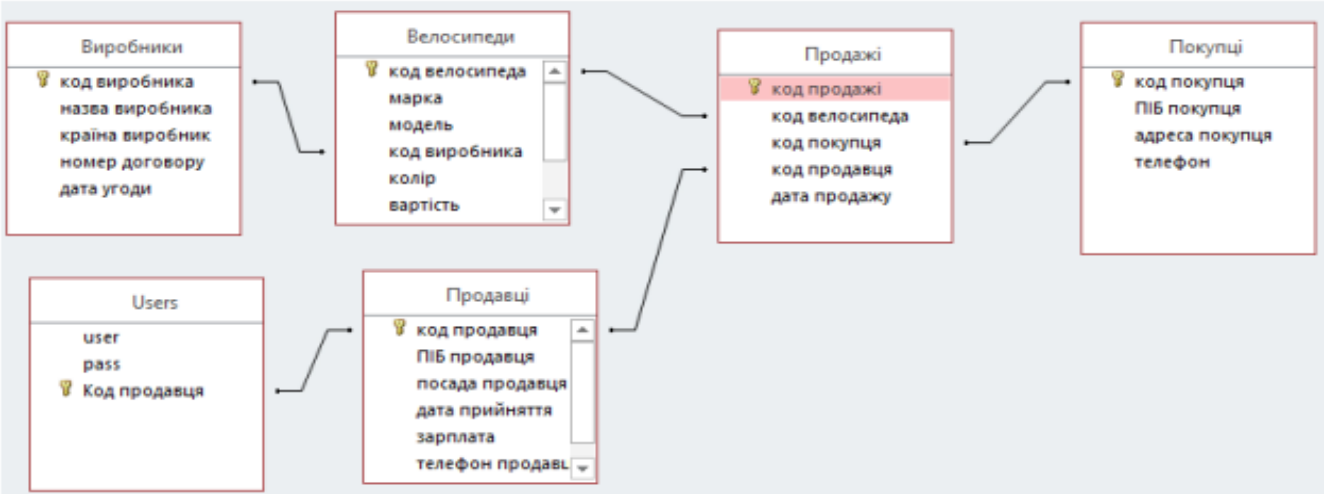


Рисунок 3.1.4 – Діаграма таблиць та зв’язків в базі даних.

3.2 Розробка програмних модулів

У цьому розділі проводиться детальний опис реалізації всіх запланованих модулів веб-застосунку з використанням технології ASP.NET MVC.

Перший модуль, що потрібно розробити, - це модуль авторизації. У цьому модулі буде забезпечена можливість вхіду в систему для користувачів зареєстрованих веб-застосунку. Авторизація передбачає перевірку введених користувачем облікових даних (логіну та паролю) і, в разі їх правильності, надання доступу до захищених ресурсів. При розробці модуля авторизації на основі ASP.NET MVC можна використовувати вбудований механізм аутентифікації та авторизації, який забезпечує зручну роботу з ролевим доступом та керуванням сеансами користувачів. Реалізовано код авторизаці було в наступному виді (рисунок 3.2.1).

```
public ActionResult Login(LoginViewModel model)
{
    // Перевірка валідності введених даних
    if (ModelState.IsValid)
    {
        // Перевірка відповідності логіна та пароля в базі даних
        if (ValidateUser(model.Username, model.Password))
        {
            // Авторизація користувача
            FormsAuthentication.SetAuthCookie(model.Username, false);
            return RedirectToAction("Index", "Home");
        }
        else
        {
            // Відображення повідомлення про невірні дані авторизації
            ModelState.AddModelError("", "Невірний логін або пароль.");
        }
    }

    // Відображення форми авторизації з помилками
    return View(model);
}

// Метод перевірки логіна та пароля користувача в базі даних
private bool ValidateUser(string username, string password)
```

Рисунок 3.2.1 – Код авторизаці.

Наступний модуль є не менш важливим, без нього взаємодія з модулем авторизації буде не доступна, це модуль реєстрації. Цей модуль відповідає за обробку запитів користувачів, які бажають створити новий обліковий запис на веб-сайті. Реалізація модулю реєстрації включає перевірку введеної інформації, створення нового користувача в базі даних та надання доступу до функціональності після успішної реєстрації. Реалізація модуля реєстрації зображено на рисунках 3.2.2-3.2.4.

Створення моделі RegistrationViewModel для представлення даних реєстрації користувача:

```
public class RegistrationViewModel
{
    public string Email { get; set; }
    public string Password { get; set; }
    public string ConfirmPassword { get; set; }
    // Додаткові поля для інформації про користувача
    // ...
}
```

Рисунок 3.2.2 – Модель RegistrationViewModel

Реалізація контролера AccountController, який відповідає за обробку дій, пов'язаних з реєстрацією:

```
public class AccountController : Controller
{
    private readonly UserManager<ApplicationUser> _userManager;
    private readonly SignInManager<ApplicationUser> _signInManager;

    public AccountController(UserManager<ApplicationUser> userManager, SignInManager<ApplicationUser> signInManager)
    {
        _userManager = userManager;
        _signInManager = signInManager;
    }

    [HttpGet]
    public IActionResult Register()
    {
        return View();
    }

    [HttpPost]
    public async Task<IActionResult> Register(RegistrationViewModel model)
    {
        if (ModelState.IsValid)
        {
            var user = new ApplicationUser { UserName = model.Email, Email = model.Email };

            var result = await _userManager.CreateAsync(user, model.Password);
            if (result.Succeeded)
            {
                await _signInManager.SignInAsync(user, isPersistent: false);
                return RedirectToAction("Index", "Home");
            }

            foreach (var error in result.Errors)
            {
                ModelState.AddModelError(string.Empty, error.Description);
            }
        }

        return View(model);
    }
```

Рисунок 3.2.3 – Реалізація контролера AccountController

Створення відповідного представлення Register.cshtml, яке відображає форму реєстрації:

```
< h2 > Реєстрація </ h2 >

< form asp - controller = "Account" asp - action = "Register" method = "post" >
    < div >
        < label > Email:</ label >
        < input type = "email" asp -for= "Email" required />
    </ div >
    < div >
        < label > Пароль:</ label >
        < input type = "password" asp -for= "Password" required />
    </ div >
    < div >
        < label > Підтвердження паролю:</ label >
        < input type = "password" asp -for= "ConfirmPassword" required />
    </ div >
    < button type = "submit" > Зареєструватися </ button >
</ form >
```

Рисунок 3.2.4 – Представлення Register.cshtml

Цей механізм реєстрації користувача пропонує простий та ефективний спосіб створення облікового запису в системі. Після успішної реєстрації, інформація про користувача зберігається в базі даних, що дозволяє йому автоматично увійти до системи. Крім того, важливо забезпечити валідацію введених даних та відповідну обробку можливих помилок.

Для успішної реєстрації користувача, система перевіряє унікальність обраного користувачем імені та адреси електронної пошти. Якщо ім'я користувача або адреса електронної пошти вже використовуються іншим користувачем, система повідомляє про це та генерує відповідне повідомлення про помилку. Це забезпечує унікальність облікових записів користувачів у системі. Це забезпечить коректне збереження даних про користувача та його обліковий запис.

Ще одним значущим модулем у розробці даного веб-застосунку є модуль управління товарами, який відіграє ключову роль у динамічному оновленні та контролі асортименту товарів. Модуль управління товарами є важливою складовою частиною розробки даного веб-застосунку і відіграє ключову роль у динамічному оновленні та контролі асортименту товарів. Цей модуль забезпечує адміністратору зручний та ефективний інтерфейс для проведення різноманітних дій, пов'язаних з товаром, таких як додавання нових товарів до системи, оновлення існуючих записів та налаштування цін.

Для успішної реалізації модуля управління товарами на платформі ASP.NET MVC, можна використати наступні компоненти:

Модель Product для представлення даних про товар (рисунок 3.2.5).

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Description { get; set; }
    public decimal Price { get; set; }
    // Додаткові властивості товару
    // ...
}
```

Рисунок 3.2.5 – Модель Product.

Контролер ProductController, який відповідає за обробку дій, пов'язаних з управлінням товарами (рисунки 3.2.6- 3.2.7)

```
public class ProductController : Controller
{
    private readonly ProductService _productService;

    public ProductController(ProductService productService)
    {
        _productService = productService;
    }

    // Дія для відображення списку товарів
    public IActionResult Index()
    {
        var products = _productService.GetAllProducts();
        return View(products);
    }

    // Дія для створення нового товару
    [HttpGet]
    public IActionResult Create()
    {
        return View();
    }

    [HttpPost]
    public IActionResult Create(Product product)
    {
        if (ModelState.IsValid)
        {
            _productService.CreateProduct(product);
            return RedirectToAction("Index");
        }

        return View(product);
    }
```

Рисунок 3.2.6 – Реалізація модуля створення нового товару

Контролер ProductController є ключовим елементом у модулі управління товарами і відповідає за обробку різних дій, пов'язаних з цими товарами. Він містить методи, які дозволяють виконувати такі дії, як перегляд списку товарів, додавання нового товару, редагування існуючих товарів та видалення товарів з системи.

Наприклад, метод Index() в контролері ProductController відповідає за відображення списку всіх товарів. Він може отримувати дані про товари з бази даних і передавати їх до відповідного представлення, де вони можуть бути відображені у зручному форматі для користувача.

Метод Create() дозволяє адміністратору додавати новий товар до системи. Цей метод приймає дані, введені користувачем в формі додавання товару, і зберігає їх у базі даних.

Аналогічно, методи Edit() та Delete() в контролері ProductController відповідають за редагування та видалення існуючих товарів відповідно. Вони отримують ідентифікатор товару, який потрібно змінити або видалити, і виконують відповідні дії з базою даних.

```
// Дія для редагування існуючого товару
[HttpGet]
public IActionResult Edit(int id)
{
    var product = _productService.GetProductById(id);
    if (product == null)
    {
        return NotFound();
    }

    return View(product);
}

[HttpPost]
public IActionResult Edit(Product product)
{
    if (ModelState.IsValid)
    {
        _productService.UpdateProduct(product);
        return RedirectToAction("Index");
    }

    return View(product);
}

// Дія для видалення товару
[HttpPost]
public IActionResult Delete(int id)
{
    _productService.DeleteProduct(id);
    return RedirectToAction("Index");
}
```

Рисунок 3.2.7– Реалізація модуля редагування та видалення існуючого товару.

Цей код демонструє базову реалізацію модуля управління товарами, де адміністратор може переглядати, створювати, редагувати та видаляти товари. Звичайно, його можна доповнити додатковою функціональністю, такою як завантаження зображень, фільтрація товарів тощо.

Наступним є модуль кошика який надає користувачам можливість збирати товари для покупки і здійснювати оформлення замовлення. Цей модуль містить такі функції як:

– Додавання товару до кошика: Користувач може вибрати бажаний товар зі сторінки і додати його до свого кошика. Це може бути зроблено за допомогою кнопки "Додати до кошика" або іншим способом. При додаванні товару, його інформація, така як назва, ціна та інші деталі, будуть збережені у кошику користувача.

– Перегляд вмісту кошика: Користувач може переглядати список товарів, які він додав до свого кошика. Це може бути представлено у вигляді таблиці або списку, де відображається назва товару, ціна, кількість та загальна сума для кожного товару.

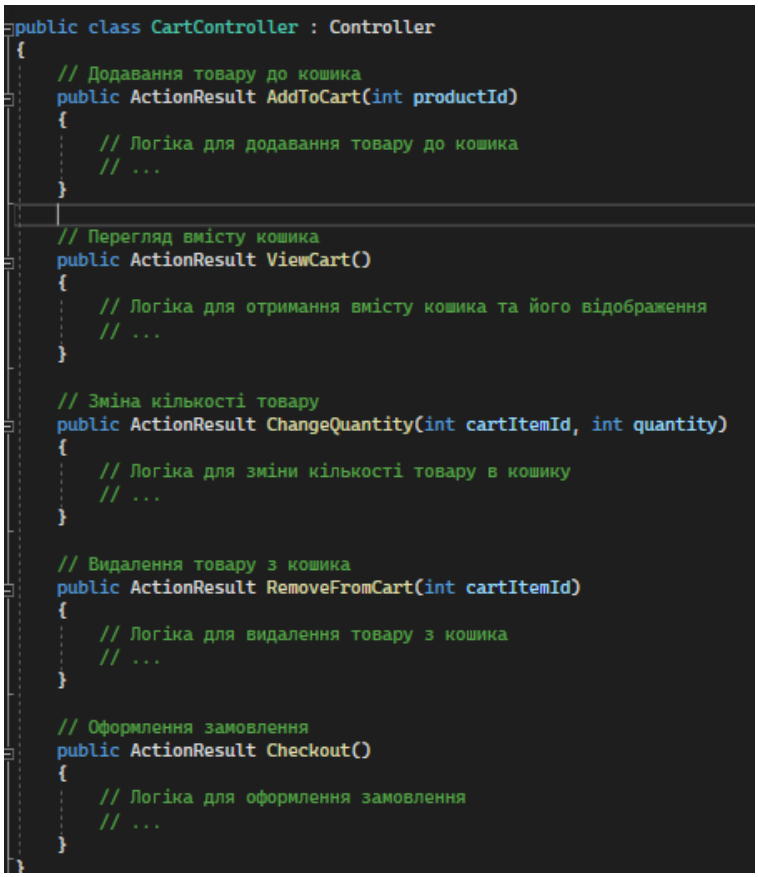
– Зміна кількості товарів: Користувач може змінювати кількість товарів, які він хоче придбати. Наприклад, він може збільшити або зменшити кількість одного товару безпосередньо в кошику.

– Видалення товарів: Користувач може видаляти товари зі свого кошика, якщо він більше не бажає їх придбати. Це може бути зроблено за допомогою кнопки "Видалити" або іншого механізму.

– Оформлення замовлення: Користувач може перейти до процесу оформлення замовлення, коли він закінчив вибір товарів у кошику. Цей процес може включати вибір способу доставки, оплати та заповнення необхідних даних для здійснення покупки.

– Розрахунок загальної суми замовлення: Модуль кошика може виконувати розрахунок загальної суми замовлення, яка включає в себе вартість усіх товарів, враховуючи їх кількість та ціни. Ця інформація може бути відображена користувачу перед оформленням замовлення.

Нижче на рисунку 3.2.8. наведений код, який може слугувати прикладом реалізації модуля кошика покупок.



```
public class CartController : Controller
{
    // Додавання товару до кошика
    public ActionResult AddToCart(int productId)
    {
        // Логіка для додавання товару до кошика
        // ...
    }

    // Перегляд вмісту кошика
    public ActionResult ViewCart()
    {
        // Логіка для отримання вмісту кошика та його відображення
        // ...
    }

    // Зміна кількості товару
    public ActionResult ChangeQuantity(int cartItemId, int quantity)
    {
        // Логіка для зміни кількості товару в кошику
        // ...
    }

    // Видалення товару з кошика
    public ActionResult RemoveFromCart(int cartItemId)
    {
        // Логіка для видалення товару з кошика
        // ...
    }

    // Оформлення замовлення
    public ActionResult Checkout()
    {
        // Логіка для оформлення замовлення
        // ...
    }
}
```

Рисунок 3.2.8 – Реалізація модуля корзини.

В цьому прикладі CartController містить методи, які відповідають за різні дії, пов'язані з кошиком покупок. Кожен метод виконує свою конкретну функцію, як описано вище. В реальному проекті код методів буде додатково розширений логікою обробки даних, роботою з базою даних та взаємодією з відповідними представленнями.

3.3 Інструкція користувача

У данному розділі наведено послідовні кроки, які потрібно виконати для користування веб-застосунку в залежності від категорії користувача: адміністратора, зареєстрованого користувача або гостя. Першим кроком після відкриття сайту є перехід на головну сторінку, яка має сучасний дизайн і надає різні можливості. На головній сторінці користувач може обрати категорію товарів, яку він бажає переглянути та придбати.

Також, на цій сторінці користувач може авторизуватись та переглянути вміст свого кошика, який відображає обрані товари. Вигляд головної сторінки показаний на рисунку 3.3.1.

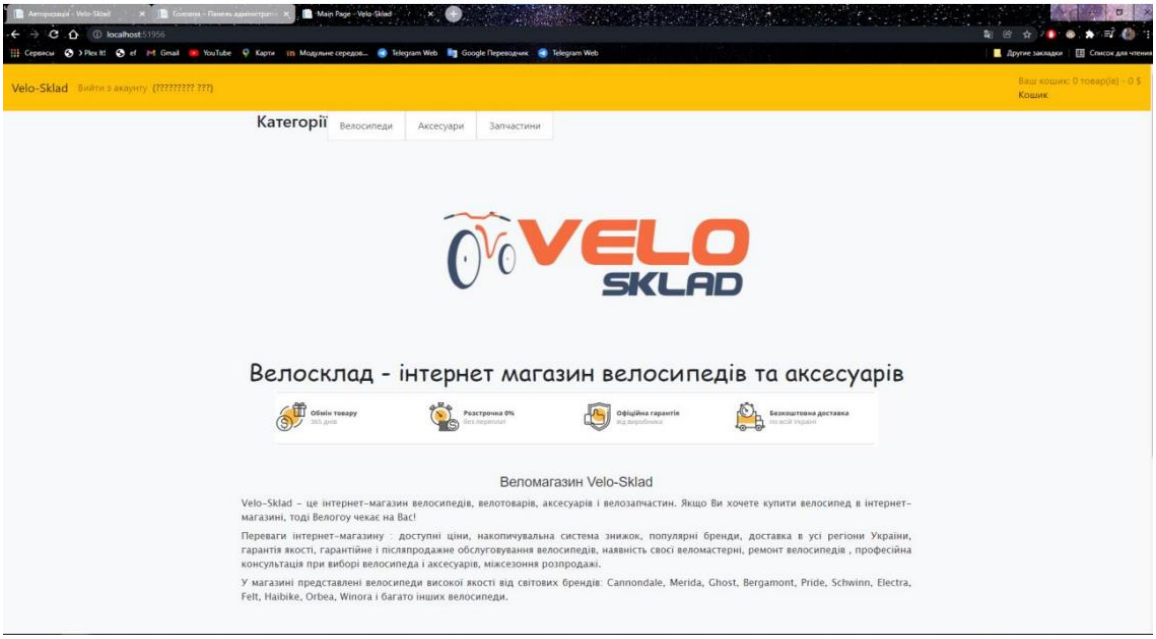


Рисунок 3.3.1. – Головна сторінка веб-застосунку

Якщо користувач не авторизований, він має можливість авторизуватись, після чого відкриється сторінка авторизації, де користувачеві потрібно ввести свій логін та пароль. Вигляд цієї сторінки показаний на рисунку 3.3.2.

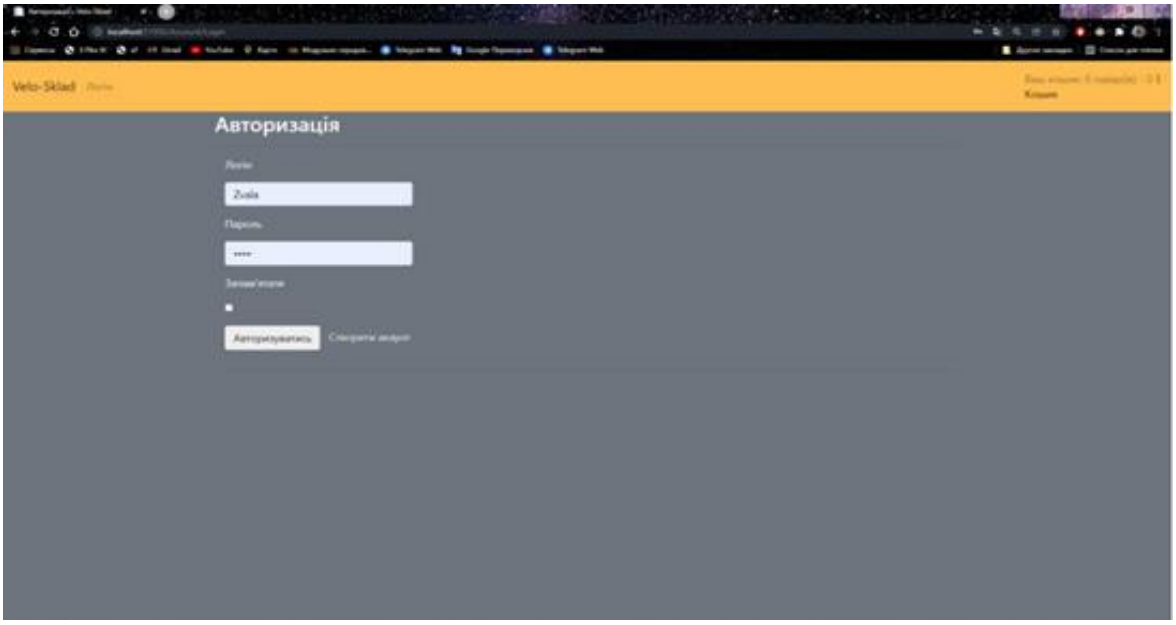


Рисунок 3.3.2. – Сторінка авторизації

Щоб оформити замовлення, користувач повинен бути зареєстрованим на сайті. Для цього він має натиснути кнопку "Створити акаунт", після чого буде перенаправлений на сторінку реєстрації, де потрібно ввести необхідну інформацію, таку як ім'я, фамілію, електронну адресу, логін та пароль. Після успішної реєстрації користувача перенаправлять на сторінку авторизації, а також виведуть повідомлення про успішне створення облікового запису. Вигляд сторінки реєстрації показаний на рисунку 3.3.3.

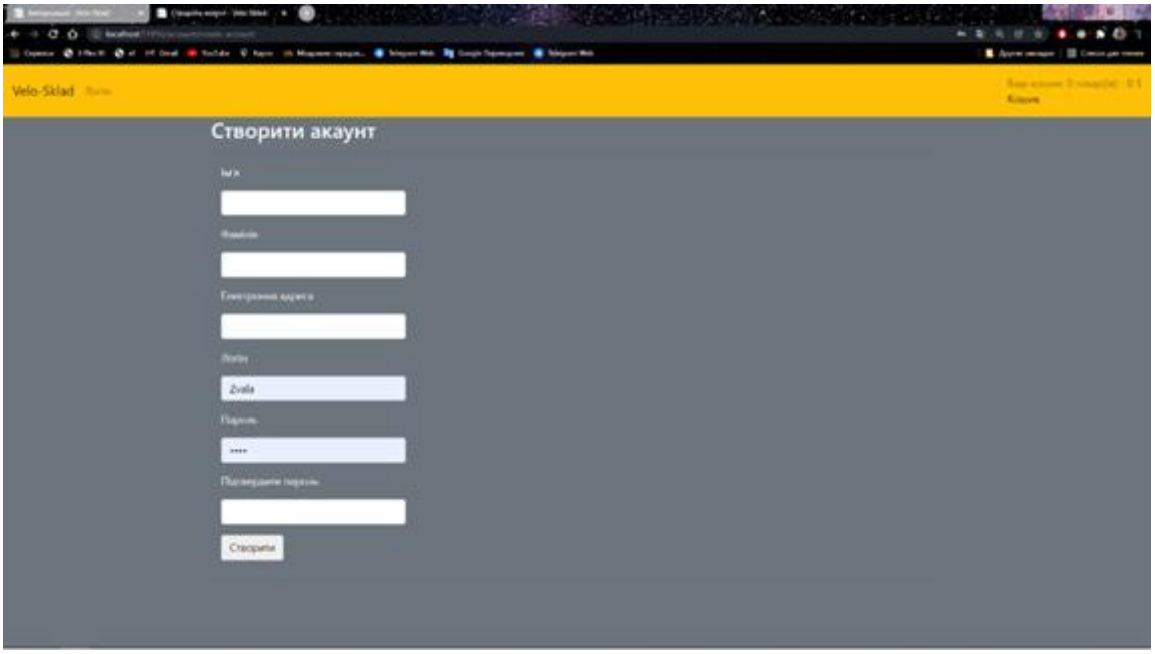


Рисунок 3.3.3. – Сторінка реєстрації користувача

Незважаючи на те, авторизований користувач чи ні , він має можливість перегляду товарів. Для цього потрібно перейти на головну сторінку і вибрати одну з трьох доступних категорій товарів (Велосипеди, Аксесуари, Запчастини). Вигляд сторінки категорії "Велосипеди" показаний на рисунку 3.3.4. Даний функціонал дозволяє користувачеві зручно ознайомитись з асортиментом товарів та вибрати потрібний товар для придбання.

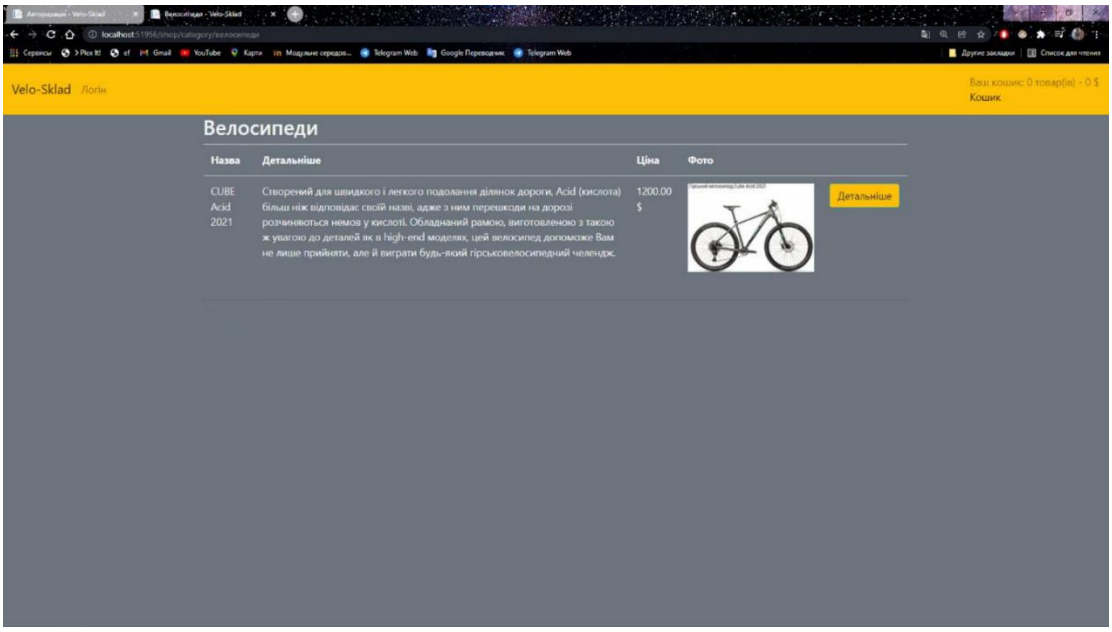


Рисунок 3.3.4. – Сторінка категорії "Велосипеди"

3.4 Технічні характеристики веб-застосунку

Зазначені вимоги є мінімальними для забезпечення належного функціонування розробленого веб-ресурсу. У розділі про технічні характеристики веб-застосунку були надані деталі щодо необхідного програмного та апаратного забезпечення для серверної та клієнтської частини.

Технічні вимоги до сервера включають:

– Операційна система: Windows або Linux.
– Веб-сервер: Apache, налаштований та встановлений.
– Мова програмування: PHP.
– База даних: MySQL версії 8.0 або вище.

Технічні вимоги до апаратного забезпечення сервера включають:

– Підключення до Інтернету.
– Мінімум 1 ГБ оперативної пам'яті.
– Потужний процесор.
– Вільне місце на жорсткому диску для зберігання веб-застосунку (приблизно 100 МБ) та бази даних (приблизно 40 МБ).

Технічні вимоги для портативних комп’ютерів:

– Операційна система: Windows, Linux, MacOS.
– Веб-браузер.
– Підключення до Інтернету.
– Мінімум 2 ГБ оперативної пам'яті.
– Процесор Athlon x4 або вище.

Технічні вимоги до мобільних пристроїв для користування веб-ресурсом:

Операційна система:

– Android версії 6.0 або вище.
– iOS версії 11 або вище.

Апаратне забезпечення:

– Мінімум 2 ГБ оперативної пам'яті.
– Процесор з достатньою продуктивністю для плавної роботи веб-застосунку.
– Достатньо вільного простору на внутрішньому сховищі для зберігання застосунку та кешу (залежить від розміру застосунку та збереження даних офлайн).
– Підключення до Wi-Fi або мобільна мережа.
– Роздільна здатність дисплею мінімум 720p (1280x720 пікселів).

Зазначені технічні вимоги забезпечують належне та комфортне використання веб-ресурсу. У результаті розробки було створено веб-ресурс для торгівлі вело товарами, описано реалізацію основних модулів та інтерфейсу, а також підготовлено інструкцію для користувачів.

3.5. Аналіз методів тестування веб- застосунку та розробка тестів

У данному розділі здійснюється верифікація і валідація веб-застосунку шляхом аналізу методів тестування та розробки відповідних тестів. Передбачається обґрунтування вибору конкретних методів тестування, розробка тестових наборів даних, а також аналіз результатів проведеного тестування.

Почнемо з обґрунтування вибору методів тестування. Під час розгляду різних методів, враховувалися специфікації проекту та його вимоги до якості. Наприклад, було визначено, що для забезпечення безперебійного функціонування веб-застосунку необхідно провести функціональне тестування, щоб переконатися, що всі основні функції працюють коректно. Також, для забезпечення стабільності і безпеки, було вирішено провести навантажувальне тестування та тестування на захист від вторгнень. Деякі з найпоширеніших типів тестування веб-застосунків включають:

– Функціональне тестування: цей тип тестування перевіряє правильність роботи функціоналу програми. Виконуються вхідні та вихідні тести, щоб перевірити коректність відповідей та обробки помилок.

– Тестування інтерфейсу користувача (UI): цей тип тестування перевіряє, чи відповідає інтерфейс користувача веб-застосунку визначеним вимогам. Перевіряється зовнішній вигляд, розташування та поведінка елементів інтерфейсу.

– Тестування продуктивності: ці тести спрямовані на перевірку продуктивності веб-застосунку, таких як час завантаження сторінки, швидкість відгуку сервера, оптимізація використання ресурсів та обробка навантаження.

– Тестування сумісності: цей тип тестування перевіряє, чи працює веб-застосунок на різних платформах, операційних системах і браузерах. Перевіряється правильність відображення та робота застосунку на різних пристроях.

Тестування навантаження: це тестування виконується для виявлення обмежень продуктивності веб-застосунку шляхом навантаження сервера і спостереження за його реакцією на це навантаження.

Після вибору методів тестування, розроблялися відповідні тестові набори даних. Для кожного методу тестування визначалися вхідні дані та очікувані результати. Наприклад, для функціонального тестування визначалися сценарії взаємодії з різними елементами веб-застосунку, а для навантажувального тестування встановлювалися максимальні навантаження та спостерігалося за

реакцією системи. Для перевірки роботи сайту було проведено функціональне тестування з 2 сценаріїв, яке спрямоване на перевірку правильності роботи програми у таблиці 3.5.1.

Тестовий сценарій 1: Зареєструватися на сайті

– Відкрити веб-браузер і перейти на головну сторінку сайту.

– Знайти посилання на реєстрацію і натиснути на нього.

– Заповнити обов'язкові поля форми реєстрації, такі як ім'я, електронна пошта та пароль.

– Натиснути кнопку "Зареєструватися".

– Перевірити, що на екрані з'являється повідомлення про успішну реєстрацію.

– Спробувати увійти до облікового запису з використанням введених при реєстрації облікових даних.

– Переконатися, що вхід виконується успішно.

Таблиця 3.5.1 – сценарій №1

Дія	Очікуваний результат	Отриманий результат
1. Відкрити веб-браузер і перейти на головну сторінку сайту.	Головна сторінка сайту завантажується.	Головна сторінка сайту завантажена.
2. Знайти посилання на реєстрацію і натиснути на нього.	Перехід на сторінку реєстрації.	Сторінка реєстрації відкрита.
3. Заповнити обов'язкові поля форми реєстрації, такі як ім'я, електронна пошта та пароль.	Поля форми заповнені коректно.	Поля форми успішно заповнені.
4. Натиснути кнопку "Зареєструватися".	Зареєстрація користувача успішна.	Користувача успішно зареєстровано.

Продовження таблиці 3.5.1

5. Перевірити, що на екрані з'являється повідомлення про успішну реєстрацію.	Повідомлення про успішну реєстрацію відображається.	Повідомлення про успішну реєстрацію видно.
6. Спробувати увійти до облікового запису з використанням введених при реєстрації облікових даних.	Успішний вхід до облікового запису.	Успішний вхід до облікового запису.
7. Переконатися, що вхід виконується успішно.	Доступ до особистого кабінету користувача.	Доступ до особистого кабінету користувача забезпечений.

Тестовий сценарій 2: Додати товар до кошика

– Увійти до облікового запису на веб-сайті.

– Перейти на сторінку каталогу товарів.

– Вибрати бажаний товар, натиснувши на його назву або зображення.

– На сторінці товару перевірити, що відображається інформація про товар, його ціна та опис.

– Знайти кнопку "Додати до кошика" і натиснути на неї.

–

– Переконатися, що на екрані з'являється повідомлення про успішне додавання товару до кошика.

– Перейти до сторінки кошика та перевірити, що вибраний товар з'явився в списку товарів кошика.

– Перевірити, що сума товарів у кошику відображається правильно.

Таблиця 3.5.2 – сценарій №2

Дія	Очікуваний результат	Отриманий результат
1. Увійти до облікового запису на веб-сайті.	Успішний вхід до облікового запису.	Успішний вхід до облікового запису.
2. Перейти на сторінку каталогу товарів.	Каталог товарів відображається	Каталог товарів відкритий.
3. Вибрати бажаний товар, натиснувши на його назву або зображення.	Відкрита сторінка товару з його інформацією.	Сторінка товару відкрита.
4. На сторінці товару перевірити, що відображається інформація про товар, його ціна та опис.	Інформація про товар, його ціна та опис відображаються коректно.	Інформація про товар, його ціна та опис відображаються.
5. Знайти кнопку "Додати до кошика" і натиснути на неї.	Товар успішно доданий до кошика.	Товар доданий до кошика.
6. Переконатися, що на екрані з'являється повідомлення про успішне додавання товару до кошика.	Повідомлення про успішне додавання товару до кошика відображається.	Повідомлення про успішне додавання товару до кошика видно.
7. Перейти до сторінки кошика та перевірити, що вибраний товар з'явився в списку товарів кошика.	Товар присутній у списку товарів кошика.	Товар присутній у списку товарів кошика.

Продовження таблиці 3.5.2

8. Перевірити, що сума товарів у кошику. відображається правильно.	Сума товарів в кошику відображається коректно.	Сума товарів в кошику відображається правильно.

Після проведення тестування, було здійснено аналіз отриманих результатів. Результати тестування були порівняні з очікуваними результатами та виявлені відхилення. Наприклад, можливі помилки або недоліки в роботі веб-застосунку.

3.6 Висновки до третього розділу

Під час роботи над розділом 3 було виконано кілька етапів розробки веб-застосунку. Спочатку була створена фізична база даних і її успішно підключено до модулів застосунку. Потім були реалізовані різні модулі застосунку, а їх роботу детально описано. Після завершення розробки веб-застосунку було підготовлено керівництво користувача для всіх категорій користувачів: неавторизованих, авторизованих і адміністраторів. Керівництво включає пояснення роботи всіх модулів веб-застосунку та ілюстрації для кращого розуміння. Останнім кроком було проведення тестів для перевірки роботи веб-застосунку, і всі тестові випадки були успішно пройдені з відмінним результатом. Результати розробки веб-застосунку є задовільними і відповідають очікуванням.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було проведено детальний аналіз предметної області, щоб визначити актуальність розробки веб-застосунку для продажу вело товарів. Використовуючи методи, такі як побудова IDEF0-діаграм декомпозиції та аналіз існуючих рішень, була створена таблиця порівняння, в якій визначені основні переваги, які потрібно реалізувати у веб-застосунку. Також були визначені функціональні та нефункціональні вимоги до розроблюваного веб-застосунку та основні актори програмного забезпечення. Для наочності була побудована UML діаграма використання.

Під час проектування веб-застосунку було розглянуто різні типи архітектури програмного забезпечення, і після аналізу було обрано клієнт-серверну архітектуру. Також була створена логічна модель бази даних з визначенням сутностей, атрибутів та зв'язків.

У процесі проектування серверної частини веб-застосунку був використаний шаблон MVC, і було створено UML діаграму послідовностей для демонстрації роботи одного з модулів. Були розроблені макети для клієнтської частини застосунку і визначений дизайн веб-застосунку.

Для реалізації веб-застосунку було використано широкий спектр засобів та технологій, що дозволило створити функціональний і ефективний продукт. Платформа ASP.NET була використана для розробки серверної частини, забезпечуючи надійність, масштабованість та безпеку застосунку. Мова програмування C# дозволила створювати потужні інструменти та функціональні рішення для веб-застосунку.

Одним із ключових елементів розробки були фреймворки ASP.NET MVC та Entity Framework. ASP.NET MVC надав потужний інструментарій для розробки модульної архітектури, зручного управління станами та ефективної обробки запитів. Entity Framework забезпечив простоту та ефективність роботи з базою даних, спрощуючи розробку та забезпечуючи швидкий доступ до інформації.

Для зберігання даних була використана СКБД Microsoft SQL Server, яка володіє високою надійністю та швидкодією, а також надає розширені можливості для керування та маніпулювання даними. Використання SQL Server дозволило забезпечити надійну та безпекову роботу з базою даних.

У процесі розробки була використана мова розмітки HTML і каскадні таблиці стилів CSS для створення привабливого та користувач-орієнтованого інтерфейсу користувача. Це дозволило створити зручне та інтуїтивно зрозуміле середовище для користувачів веб-застосунку.

Після завершення проектування фізична база даних була реалізована згідно з логічною моделлю. Це включало створення таблиць, визначення відношень та налагодження індексів для оптимізації доступу до даних. Це дозволило забезпечити ефективне зберігання та обробку інформації у веб-застосунку.

Основні модулі серверної частини веб-застосунку були розроблені з урахуванням функціональних та нефункціональних вимог. Це включало реалізацію бізнес-логіки, обробку запитів, зберігання та маніпулювання даними. Клієнтська частина інтерфейсу користувача була розроблена з використанням сучасних технологій і методів, що забезпечили зручну та привабливу взаємодію з веб-застосунком.

Останнім етапом було проведення тестування системи, яке включало створення кейс-тестів для перевірки роботи різних модулів. Це дозволило виявити та виправити всі недоліки та помилки у веб-застосунку, забезпечивши його бездоганну роботу.

В результаті успішного виконання кваліфікаційної роботи був створений веб-застосунок для продажу вело товарів. Даний застосунок повністю задовольняє всі вимоги, успішно пройшов тестування і працює надійно та ефективно. Виконання даної роботи також сприяло отриманню нових навичок у проектуванні та розробці веб-застосунків та поглибленню наявних знань у цій галузі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Халап, І.В. "Основи програмування в середовищі .NET" / І.В. Халап. - Київ: Навчальна книга – Богдан, 2019. - 320 с.

2. Дубінін, Є.В., Жукова, В.М. "ASP.NET MVC: Професійний підхід" / Є.В. Дубінін, В.М. Жукова. - Київ: Диалектика, 2015. - 384 с.

3. Харченко, О.В. "Основи програмування на мові C#" / О.В. Харченко. - Київ: Видавничий дім "Ін Юре", 2016. - 352 с.

4. Гаврилюк, В.М., Довгий, В.І. "ASP.NET 4.5 і чистий код" / В.М. Гаврилюк, В.І. Довгий. - Київ: Київський університет, 2014. - 288 с.

5. Шевченко, А.В., Клочко, І.О. "ASP.NET 4.5 з використанням C#" / А.В. Шевченко, І.О. Клочко. - Київ: Центр учбової літератури, 2013. - 432 с.

6. Бондаренко, Є.А., Лещинський, І.М. "ASP.NET 4.5: Створення веб-додатків" / Є.А. Бондаренко, І.М. Лещинський. - Київ: Видавництво "Слово", 2014. - 520 с.

7. Васильєв, І.С., Черевко, А.А. "ASP.NET. Технології AJAX і DHTML" / І.С. Васильєв, А.А. Черевко. - Київ: Видавничо-поліграфічний центр "Київський університет", 2016. - 344 с.

8. Синявський, В.В., Кучеренко, А.А. "ASP.NET 3.5" / В.В. Синявський, А.А. Кучеренко. - Київ: Диалектика, 2009. - 416 с.

9. Котляр, І.В. "Основи програмування в середовищі Visual Studio 2019" / І.В. Котляр. - Київ: Видавничий дім "Ін Юре", 2019. - 400 с.

10. Бобер, В.В. "ASP.NET 4.0 для професіоналів" / В.В. Бобер. - Київ: Диалектика, 2010. - 960 с.

11. Лабунець, О.П., Мальцев, А.В. "Основи програмування на мові C#. Навчальний посібник" / О.П. Лабунець, А.В. Мальцев. - Київ: НУХТ, 2013. - 220 с.

12. Іванчук, О.Є., Боднарчук, Ю.С. "ASP.NET: Технології створення веб-додатків" / О.Є. Іванчук, Ю.С. Боднарчук. - Київ: Київський університет, 2009. - 424 с.

13. Молчанов, А.В. "ASP.NET 3.5. Профессиональное программирование" / А.В. Молчанов. - Київ: Диалектика, 2008. - 1088 с.

14. Біловодський, В.І., Поліщук, О.О. "ASP.NET. Практичний курс" / В.І. Біловодський, О.О. Поліщук. - Київ: Диалектика, 2010. - 736 с.

15. Л. П. Бедратюк. Дипломний проект: методичні вказівки щодо його виконання для студентів спеціальності 121 «Інженерія програмного забезпечення» / Л. П. Бедратюк, Г. І. Радельчук, Ю. В. Форкун, О. М. Яшина. Хмельницький: ХНУ, 2020. – 77 с.

16. Халап, І.В. "Основи програмування в середовищі .NET" / І.В. Халап. - Київ: Навчальна книга – Богдан, 2019. - 320 с.

17. Іванова, О.М. "Введення в бази даних: навчальний посібник" / О.М. Іванова. - Львів: Світ, 2020. - 256 с.

18. Петров, В.С. "Алгоритми та структури даних" / В.С. Петров. - Київ: Видавництво НТУУ "КПІ", 2018. - 400 с.

19. Сидоренко, Н.О. "Веб-дизайн та розробка сайтів" / Н.О. Сидоренко. - Одеса: Фенікс, 2017. - 192 с.

20. Мельник, О.Р. "Основи мережних технологій" / О.Р. Мельник. - Київ: КНЕУ, 2021. - 240 с.

21. Литвин, А.П. "Математичний аналіз" / А.П. Литвин. - Львів: Видавництво ЛНУ імені Івана Франка, 2019. - 480 с.

22. Кравець, В.М. "Електронні таблиці: навчальний посібник" / В.М. Кравець. - Львів: Карп'юк, 2018. - 224 с.

23. Соловйов, Д.О. "Бази даних: навчальний посібник" / Д.О. Соловйов. - Київ: Наш формат, 2021. - 320 с.

24. Іванчук, О.Є., Боднарчук, Ю.С. - "ASP.NET: Технології створення веб-додатків" / О.Є. Іванчук, Ю.С. Боднарчук. - Київ: Київський університет, 2009. - 424 с.

25. Молчанов, А.В. - "ASP.NET 3.5. Профессиональное программирование" / А.В. Молчанов. - Київ: Диалектика, 2008. - 1088 с.

26. Біловодський, В.І., Поліщук, О.О. - "ASP.NET. Практичний курс" / В.І. Біловодський, О.О. Поліщук. - Київ: Диалектика, 2010. - 736 с.

27. Харрінгтон, Маріо - "ASP.NET MVC 5 з використанням C# та .NET Framework 4.5.1" / Маріо Харрінгтон. - Київ: Наш формат, 2016. - 688 с.

28. Лебідь, Максим - "ASP.NET Core 3.1: Разработка кросс-платформенных веб-приложений" / Максим Лебідь. - Київ: Лебідь, 2020. - 688 с.

29. Фрейман, Адам - "ASP.NET Core 2.0: Создание масштабируемых и производительных веб-приложений" / Фрейман Адам. - Київ: ДМК Пресс, 2018. - 608 с.

30. Коткін, Володимир - "ASP.NET Core MVC з Entity Framework Core" / Володимир Коткін. - Київ: Дія, 2018. - 400 с.

31. Шаров, Андрій - "ASP.NET Core 3.1: Разработка веб-приложений" / Андрій Шаров. - Київ: Контакт, 2020. - 432 с.

32. Фрейман, Адам - "ASP.NET Core 2.1: Создание современных веб-приложений с помощью ASP.NET Core MVC и Angular" / Адам Фрейман. - Київ: ДМК Пресс, 2019. - 640 с.

33. Фрейман, Адам - "ASP.NET Core 2.2: Разработка полнофункциональных веб-приложений" / Адам Фрейман. - Київ: ДМК Пресс, 2020. - 768 с.

34. Прайс, Марк Дж. - "ASP.NET Core 3.0: Разработка веб-приложений на C#" / Марк Дж. Прайс. - Київ: ДМК Пресс, 2020. - 896 с.

35. Прайс, Марк Ж. - "ASP.NET Core 2.0. Кросплатформенное программирование на C#" / Марк Ж. Прайс. - Київ: ДМК Пресс, 2019. - 928 с.

36. Піскопо, Джесеппе - "ASP.NET Core: Разработка приложений любого типа и масштаба" / Джесеппе Піскопо. - Київ: ДМК Пресс, 2020. - 1024 с.

37. Макаренко, Валерія - "ASP.NET Core и Angular 2. Разработка современных веб-приложений" / Валерія Макаренко. - Київ: ДМК Пресс, 2017. - 800 с.

38. Гащенко, Роман - "ASP.NET Core 2.1. MVC, SignalR, EF Core" / Роман Гащенко. - Київ: ДМК Пресс, 2019. - 512 с.

39. Соркин, Михаил - "ASP.NET Core и кросс-платформенная разработка. Веб-приложения, работающие на любых устройствах" / Михаил Соркин. - Київ: ДМК Пресс, 2018. - 704 с.

40. Петриченко, Іван - "ASP.NET Core 2.1: Профессиональное программирование" / Іван Петриченко. - Київ: ДМК Пресс, 2019. - 1024 с.

41. Хомси, Алекс - "ASP.NET Core. Разработка приложений на C# для профессионалов" / Алекс Хомси. - Київ: ДМК Пресс, 2020. - 928 с.

42. Розников В. А., Захарова О. В., Захарова Е. Г. Інформаційні системи та сервіси // Проблеми програмування. – 2017. – № 4 – С. 60-72

ДОДАТОК А

(Обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Основне завдання полягає в розробці веб-застосунку для торгівлі будівельними інструментами. Цей проект включає створення веб-сайту, який надає можливість здійснювати покупки онлайн.

1.Підстава для розробки

Підставою для розробки даного веб-застосунку є «Завдання на кваліфікаційну роботу», затверджене завідувачем кафедри програмного забезпечення. Найменування розробки: «Веб-застосунок для продажу велотоварів».

2 Призначення розробки

2.1 Функціональне призначення

Функціональне призначення веб-застосунку полягає в розробці веб-застосунку для продажу велотоварів, який надасть клієнтам можливість переглядати та купувати товари через свій комп'ютер або мобільний пристрій. Додаток також забезпечує функціонал електронного платежу та виступає посередником між клієнтами та працівниками магазину.

2.2 Експлуатаційні призначення

Мій веб-додаток призначений для широкого кола користувачів без спеціальної підготовки і доступний через веб-браузери на різних пристроях, таких як персональні комп'ютери, ноутбуки, планшети та смартфони. Додаток підтримує операційні системи, такі як Windows, Mac, Android та iOS.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

- додавання товарів до каталогу;
- реєстрація та авторизація у системі;
- налаштування особистого профілю;

- відображення товарів та інформації про них;
- фільтрація для пошуку товарів;
- додавання товарів до кошику;
- оформлення замовлення.

3.2 Вимоги до надійності

Мій веб-додаток має гарантувати безпеку особистих даних користувачів та передавати інформацію про їх спосіб оплати тільки у зашифрованому форматі. Система повинна надати користувачам повідомлення про помилки та автоматично перенаправити їх на функціонуючу сторінку при виникненні помилок.

3.3 Умови експлуатації

Експлуатаційні умови повинні відповідати нормам щодо санітарних і технічних параметрів для персональних комп'ютерів, таких як температура та вологість навколишнього середовища, згідно з вимогами ГОСТ 15150-69 [7]. Обслуговування веб-застосунку можуть здійснювати лише спеціалісті з відповідним навчанням, такі як адміністратори або розробники. Використання веб-застосунку дозволяється користувачам, які мають доступ до Інтернету.

3.4 Вимоги до складу та параметрів технічних засобів

Технічні вимоги до апаратного забезпечення сервера включають:

- Підключення до Інтернету.
- Мінімум 1 ГБ оперативної пам'яті.
- Потужний процесор.
- Вільне місце на жорсткому диску для зберігання веб-додатку (приблизно 100 МБ) та бази даних (приблизно 40 МБ).

Технічні вимоги для портативних комп’ютерів:

– Операційна система: Windows, Linux, MacOS.
– Веб-браузер.
– Підключення до Інтернету.
– Мінімум 2 ГБ оперативної пам'яті.
– Процесор Athlon x4 або вище.

Технічні вимоги до мобільних пристроїв для користування веб-ресурсом:
Операційна система:
– Android версії 6.0 або iOS версії 11 вище.
Апаратне забезпечення:
– Мінімум 2 ГБ оперативної пам'яті.
– Процесор з достатньою продуктивністю для плавної роботи веб-застосунку.
– Достатньо вільного простору на внутрішньому сховищі для зберігання додатку та кешу (залежить від розміру додатку та збереження даних офлайн).

3.5 Вимоги до інформаційної та програмної сумісності

Для створення веб-застосунку використовувалися технології платформи ASP.NET, мова програмування C# та фреймворки ASP.NET MVC і Entity Framework

3.6 Спеціальні вимоги

Для коректної функціональності ПЗ спеціальні вимоги відсутні.

4 Вимоги до програмної документації

Набір документації, що надається у момент здачі проекту:

– Опис модулів програми з демонстрацією їх роботи
– Текст програми з коментарями
– Керівництво користувача для звичайних користувачів та адміністратора

– Довідкова інформація
– Керівництво для розробника

5 Стадії та етапи розробки

Стадії та етапи розробки веб-застосунку продемонстровано у таблиці 1

Таблиця 1 - стадії та етапи розробки

Стадія розробки	Етапи робіт	Зміст робіт
Технічне завдання 02.01 – 31.01.2023	Обґрунтування необхідності розробки програми	Коротка характеристика програмного забезпечення; підстава і призначення розробки; вимоги до програмної системи і документація; стадії і етапи розробки програми; порядок контролю і приймання
Ескізний проект 01.02 – 14.02 2023	Розробка ескізного проекту	Попередня розробка структури вхідних і вихідних даних; уточнення середовища програмування; розробка і опис загальної алгоритмічної структури системи, що буде розроблюватися
Технічний проект 15.02 – 28.02 2023	Розробка технічного проекту	Уточнення структури вхідних і вихідних даних; розробка докладного алгоритму; розробка структури програми; остаточне визначення конфігурації технічних засобів
Робочий проект 01.03 – 10.04.2023	Розробка програмного забезпечення	Реалізація програмного забезпечення; відладка; проведення попереднього тестування
Розробка програмної документації 11.04 – 20.04.2023	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням
Тестування системи 21.04 – 30.04.2023	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Впровадження	Підготовка і передача програми	Підготовка і передача програмного забезпечення; навчання персоналу використовуванню програмного забезпечення; внесення коректувань в програмне забезпечення і документацію

6 Порядок контролю та приймання

Контроль здійснюється користувачами застосунку підключеними на етапі тестування. Прийом програми здійснюється після повного її розгортання, тестування і налаштування для справного функціонування.

ДОДАТОК Б

(Обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

Код View/ User profile

```
@model MVC_Store.Models.ViewModels.Account.UserProfileVM

@{
    ViewBag.Title = "Профіль";
}

<h2>Ваш профіль</h2>


@if (TempData["SM"] != null)
{
    <div class="alert alert-success">
        @TempData["SM"]
    </div>
}


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">

        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.Id)

        <div class="form-group">
            <p class="control-label col-md-2">Ім'я</p>
            <div class="col-md-10">
                @Html.EditorFor(model => model.FirstName, new { htmlAttributes = new {
@class = "form-control" } })
```

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

        <div class="form-group">
            <p class="control-label col-md-2">Прізвище</p>
            <div class="col-md-10">
                @Html.EditorFor(model => model.LastName, new { htmlAttributes = new { @class
= "form-control" } })
                @Html.ValidationMessageFor(model => model.LastName, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <p class="control-label" style="padding-left:15px">Електронна адреса</p>
            <div class="col-md-10">
                @Html.EditorFor(model => model.EmailAdress, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.EmailAdress, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <p class="control-label col-md-2">Логін</p>
            <div class="col-md-10">
                @Html.EditorFor(model => model.Username, new { htmlAttributes = new { @class
= "form-control" } })
                @Html.ValidationMessageFor(model => model.Username, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <p class="control-label col-md-2">Пароль</p>
            <div class="col-md-10">
                @Html.PasswordFor(model => model.Password,new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.Password, "", new { @class =
"text-danger" })
            </div>
```

```
        </div>


        <div class="form-group">
            <p class="control-label" style="padding-left:15px">Підтвердити пароль</p>
            <div class="col-md-10">
                @Html.PasswordFor(model => model.ConfirmPassword,new { @class = "form-
control" })
                @Html.ValidationMessageFor(model => model.ConfirmPassword, "", new { @class
= "text-danger" })
            </div>
        </div>


        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Змінити данні" class="btn btn-default" />
            </div>
        </div>
    </div>
}
```

```
}
```

КОД CARTCONTROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Mail;
using System.Web;
using System.Web.Mvc;
using MVC_Store.Models.Data;
using MVC_Store.Models.ViewModels.Cart;

namespace MVC_Store.Controllers
{
    public class CartController : Controller
    {
```

```
public ActionResult Index()
{
    var cart = Session["cart"] as List<CartVM> ?? new List<CartVM>();

    if (cart.Count == 0 || Session["cart"] == null)
    {
        ViewBag.Message = "Your cart is empty.";
        return View();

    }
    decimal total = 0m;

    foreach (var item in cart)

    {
        total += item.Total;
    }

    ViewBag.GrandTotal = total;
    return View(cart);
}

public ActionResult CartPartial()

{
    CartVM model = new CartVM();
    int qty = 0;
    decimal price = 0m;
    if (Session["cart"] != null)

    {
        var list = (List<CartVM>) Session["cart"];

        foreach (var item in list)

        {
            qty += item.Quantity;
            price += item.Quantity * item.Price;
        }
```

```
            model.Quantity = qty;
            model.Price = price;

        }
        Else

        {
            model.Quantity = 0;
            model.Price = 0m;

        }

        return PartialView("_CartPartial", model);
    }

    public ActionResult AddToCartPartial(int id)

    {
        List<CartVM> cart = Session["cart"] as List<CartVM> ?? new List<CartVM>();
        CartVM model = new CartVM();

        using (Db db = new Db())

        {
            ProductDTO product = db.Products.Find(id);

            var productInCart = cart.FirstOrDefault(x => x.ProductId == id);

            if (productInCart == null)

            {
                cart.Add(new CartVM()

                {
                    ProductId = product.Id,
                    ProductName = product.Name,
                    Quantity = 1,
                    Price = product.Price,
```

```
                Image = product.ImageName
            });
        }
        else
        {
            productInCart.Quantity++;
        }
    }

    int qty = 0;
    decimal price = 0m;

    foreach (var item in cart)

    {
        qty += item.Quantity;
        price += item.Quantity * item.Price;
    }

    model.Quantity = qty;
    model.Price = price;

    Session["cart"] = cart;

    return PartialView("_AddToCartPartial", model);
}

public JsonResult IncrementProduct(int productId)

{
    List<CartVM> cart = Session["cart"] as List<CartVM>;

    using (Db db = new Db())

    {
        CartVM model = cart.FirstOrDefault(x => x.ProductId == productId);

        model.Quantity++;
```

```
            var result = new {qty = model.Quantity, price = model.Price};

            return Json(result, JsonRequestBehavior.AllowGet);
        }
    }


    public ActionResult DecrementProduct(int productId)

    {
        List<CartVM> cart = Session["cart"] as List<CartVM>;

        using (Db db = new Db())

        {

            CartVM model = cart.FirstOrDefault(x => x.ProductId == productId);

            if (model.Quantity > 1)
                model.Quantity--;
            else

            {
                model.Quantity = 0;
                cart.Remove(model);

            }

            var result = new {qty = model.Quantity, price = model.Price};

            return Json(result, JsonRequestBehavior.AllowGet);
        }

    }

    public void RemoveProduct(int productId)

    {
        List<CartVM> cart = Session["cart"] as List<CartVM>;
```

```
    using (Db db = new Db())

    {
        CartVM model = cart.FirstOrDefault(x => x.ProductId == productId);

        cart.Remove(model);
    }
}

public ActionResult PaypalPartial()

{
    List<CartVM> cart = Session["cart"] as List<CartVM>;

    return PartialView(cart);

}

[HttpPost]
public void PlaceOrder()

{
    List<CartVM> cart = Session["cart"] as List<CartVM>;

    string userName = User.Identity.Name;

    int orderId = 0;

    using (Db db = new Db())

    {
        OrderDTO orderDto = new OrderDTO();

        var q = db.Users.FirstOrDefault(x => x.Username == userName);
        int userId = q.Id;

        orderDto.UserId = userId;
        orderDto.CreatedAt = DateTime.Now;
```

```
                db.Orders.Add(orderDto);
                db.SaveChanges();

                orderId = orderDto.OrderId;

                OrderDetailsDTO orderDetailsDto = new OrderDetailsDTO();

                foreach (var item in cart)
                {
                    orderDetailsDto.OrderId = orderId;
                    orderDetailsDto.UserId = userId;
                    orderDetailsDto.ProductId = item.ProductId;
                    orderDetailsDto.Quantity = item.Quantity;

                    db.OrderDetails.Add(orderDetailsDto);
                    db.SaveChanges();
                }
            }

            var client = new SmtpClient("smtp.mailtrap.io", 2525)
            {

                Credentials = new NetworkCredential("ee053ae36467a9", "8bf25933605f02"),
                EnableSsl = true

            };

            client.Send("shop@example.com", "admin@example.com", "Нове замовлення", $"Ви
отримали замовлення. Номер замовлення: {orderId}");

            Session["cart"] = null;
        }
    }
}
```

КОД ACCOUNTCONTROLLER.CS

```
using System;
```

```
using System.Linq;
using System.Web.Mvc;
using System.Web.Security;
using MVC_Store.Models.Data;
using MVC_Store.Models.ViewModels.Account;
using Microsoft.AspNet.Identity;

namespace MVC_Store.Controllers
{
    public class AccountController : Controller
    {
        public ActionResult Index()
        {
            return RedirectToAction("Login");
        }

        [HttpGet]
        public ActionResult Register()
        {
            return View("Register");
        }

        [HttpPost]
        public ActionResult Register(RegisterViewModel model)
        {
            if (!ModelState.IsValid)
                return View("Register", model);

            if (!model.Password.Equals(model.ConfirmPassword))
            {
                ModelState.AddModelError("", "Паролі не співпадають!");
                return View("Register", model);
            }

            using (Db db = new Db())
            {
                if (db.Users.Any(x => x.Username.Equals(model.Username)))
                {
                    ModelState.AddModelError("", $"Ім'я користувача {model.Username}
вже використовується.");
                    model.Username = "";
                    return View("Register", model);
                }

                UserDTO userDTO = new UserDTO()
                {
                    FirstName = model.FirstName,
                    LastName = model.LastName,
                    Email = model.Email,
                    Username = model.Username,
                    Password = model.Password
                };

                db.Users.Add(userDTO);
                db.SaveChanges();

                int id = userDTO.Id;

                UserRoleDTO userRoleDTO = new UserRoleDTO()
                {
                    UserId = id,
                    RoleId = 2
                };

                db.UserRoles.Add(userRoleDTO);
                db.SaveChanges();
            }
```

```
            TempData["SuccessMessage"] = "Ви успішно зареєстровані. Увійдіть, щоб
продовжити.";

            return RedirectToAction("Login");
        }

        [HttpGet]
        public ActionResult Login()
        {
            string userName = User.Identity.Name;

            if (!string.IsNullOrEmpty(userName))
                return RedirectToAction("UserProfile");

            return View();
        }

        [HttpPost]
        public ActionResult Login(LoginViewModel model)
        {
            if (!ModelState.IsValid)
                return View(model);

            bool isValid = false;

            using (Db db = new Db())
            {
                if (db.Users.Any(x => x.Username.Equals(model.Username) &&
x.Password.Equals(model.Password)))
                    isValid = true;

                if (!isValid)
                {
                    ModelState.AddModelError("", "Неправильне ім'я користувача або
пароль.");
                    return View(model);
                }
                else
                {
                    FormsAuthentication.SetAuthCookie(model.Username,
model.RememberMe);
                    return Redirect(FormsAuthentication.GetRedirectUrl(model.Username,
model.RememberMe));
                }
            }
        }

        public ActionResult Logout()
        {
            FormsAuthentication.SignOut();
            return RedirectToAction("Login");
        }

        public ActionResult UserNavPartial()
        {
            string userName = User.Identity.Name;

            UserNavPartialViewModel model;

            using (Db db = new Db())
            {
                UserDTO dto = db.Users.FirstOrDefault(x => x.Username == userName);

                model = new UserNavPartialViewModel()
                {
                    FirstName = dto.FirstName,
```

```
                    LastName = dto.LastName
                };
            }

            return PartialView(model);
        }

        [HttpGet]
        public ActionResult UserProfile()
        {
            string userName = User.Identity.Name;

            UserProfileViewModel model;

            using (Db db = new Db())
            {
                UserDTO dto = db.Users.FirstOrDefault(x => x.Username == userName);

                model = new UserProfileViewModel(dto);
            }

            return View("UserProfile", model);
        }

        [HttpPost]
        public ActionResult UserProfile(UserProfileViewModel model)
        {
            bool isUserNameChanged = false;

            if (!ModelState.IsValid)
            {
                return View("UserProfile", model);
            }

            if (!string.IsNullOrWhiteSpace(model.Password))
            {
                if (!model.Password.Equals(model.ConfirmPassword))
                {
                    ModelState.AddModelError("", "Паролі не співпадають.");
                    return View("UserProfile", model);
                }
            }

            using (Db db = new Db())
            {
                string userName = User.Identity.Name;

                if (userName != model.Username)
                {
                    userName = model.Username;
                    isUserNameChanged = true;
                }

                if (db.Users.Where(x => x.Id != model.Id).Any(x => x.Username ==
userName))
                {
                    ModelState.AddModelError("", $"Ім'я користувача {model.Username}
вже використовується. Виберіть інше ім'я.");
                    model.Username = "";
                    return View("UserProfile", model);
                }

                UserDTO dto = db.Users.Find(model.Id);

                dto.FirstName = model.FirstName;
                dto.LastName = model.LastName;
                dto.Email = model.Email;
```

```
                dto.Username = model.Username;

                if (!string.IsNullOrWhiteSpace(model.Password))
                {
                    dto.Password = model.Password;
                }

                db.SaveChanges();
            }

            TempData["SuccessMessage"] = "Ваші дані було успішно оновлено!";

            if (!isUserNameChanged)
                return View("UserProfile", model);
            else
                return RedirectToAction("Logout");
        }
    }
                                            }
```

Код Login.cshtml

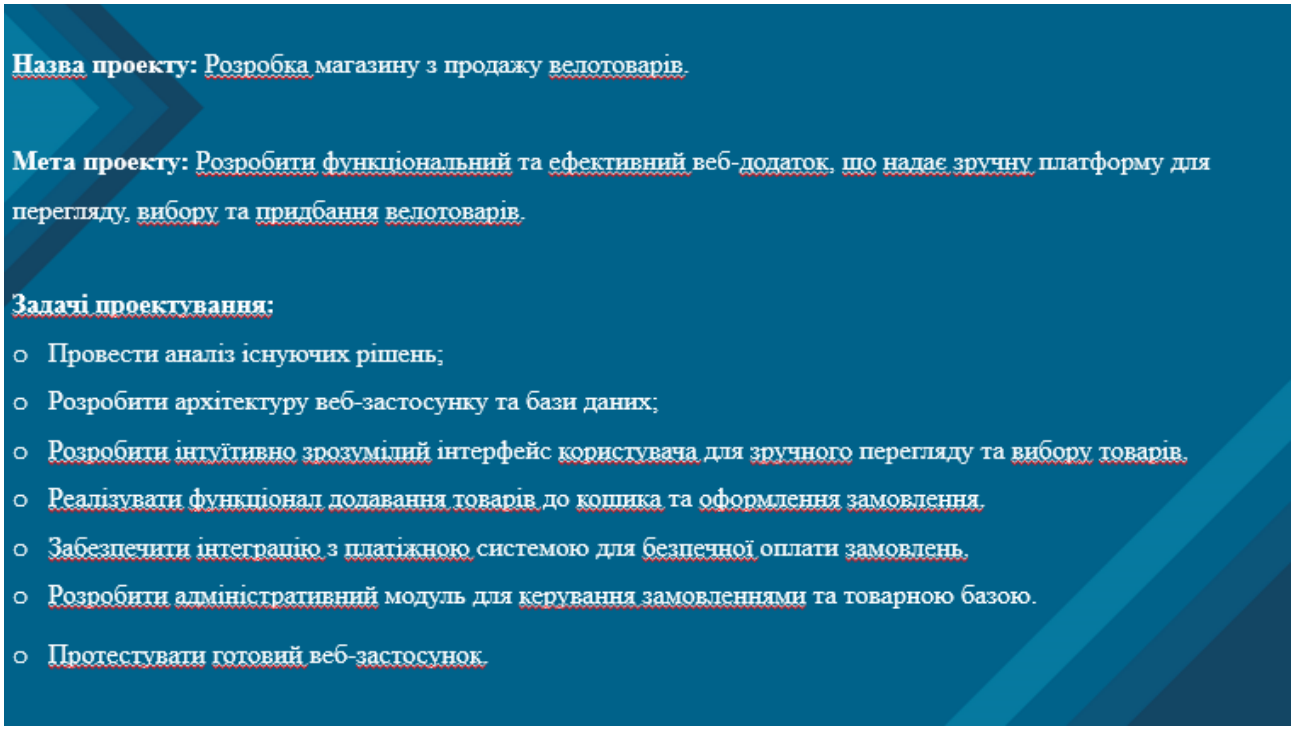
```
@model MVC_Store.Models.ViewModels.Account.LoginUserVM

@{
    ViewBag.Title = "Авторизація";
}

<head>
    <link type="text/css" rel="stylesheet" href="@Url.Content("~/Content/Style.css")"
/>
</head>

<h2>Авторизація</h2>

@if (TempData["SM"] != null)
{
    <div class="alert alert-success">
        @TempData["SM"]
    </div>
}

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            <p class="control-label col-md-2">Логін</p>
            <div class="col-md-10">
                @Html.EditorFor(model => model.Username, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Username, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <p class="control-label col-md-2">Пароль</p>
            <div class="col-md-10">
                @Html.PasswordFor(model => model.Password, new { @class = "form-
control" })
```

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

        <div class="form-group">
            <p class="control-label col-md-1">Запам'ятати</p>
            <div class="col-md-10">
                <div class="checkbox">
                    @Html.EditorFor(model => model.RememberMe)
                    @Html.ValidationMessageFor(model => model.RememberMe, "", new {
@class = "text-danger" })
                </div>
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Авторизуватись" class="btn btn-default" />
                  <a href="@Url.Action("CreateAccount", "Account")">Створити
акаунт</a>
            </div>
        </div>
    </div>
                                                }
```

ДОДАТОК В

(Обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ



Актуальність курсового проекту

Актуальність поставленої задачі для веломагазину полягає в тому, що зростає популярність онлайн-покупок та електронної комерції, в тому числі й у сфері велотоварів. Застосування веб-застосунку для продажу велотоварів надає багато переваг як для бізнесу, так і для клієнтів.

Перш за все, веб-застосунок дозволить веломагазину розширити свої ринкові можливості і привернути більше клієнтів, оскільки буде доступний для користувачів з будь-якого місця та в будь-який час. Клієнти зможуть швидко та зручно знайти потрібний товар, порівняти ціни та характеристики, що сприятиме підвищенню продажів.

Додатково, веб-застосунок дозволить зберігати та обробляти дані про покупців, їхні вподобання та історію покупок. Це дасть змогу веломагазину створити більш персоналізовані пропозиції та рекомендації для клієнтів, покращуючи їхнє задоволення від покупок і збільшуючи лояльність до бренду.

Крім того, веб-застосунок забезпечить зручність та швидкість процесу замовлення та оплати товарів. Клієнти зможуть легко вибрати бажаний товар, обрати зручний спосіб доставки та здійснити безпечну оплату в онлайн-режимі. Це дозволить зменшити час та зусилля, які затрачаються на здійснення покупок, і створить позитивний досвід для клієнтів.

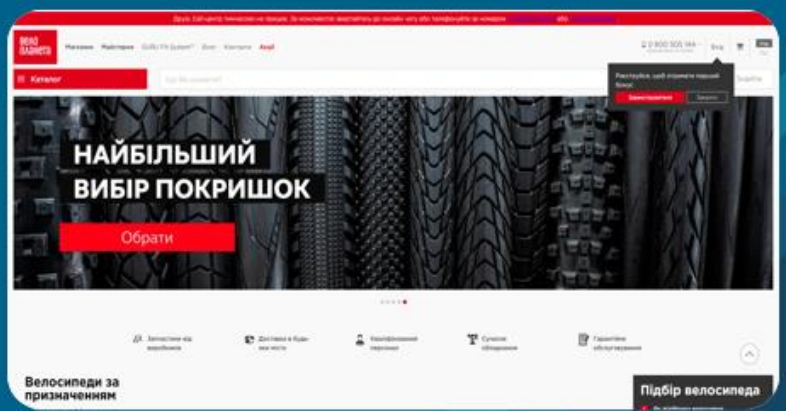
Аналіз існуючих рішень

Аналог 1: "Велопланета"

Недоліки:

Недостатня швидкість завантаження сторінок, що впливає на користувацький досвід. Відсутність системи керування запасами, що може призводити до недостовірної інформації про наявність товарів.

Обмеження: Відсутність можливості зберігання платіжних даних користувачів, що може порушувати вимоги безпеки та конфіденційності.



Аналог 2: "Велосипед"

Недоліки:

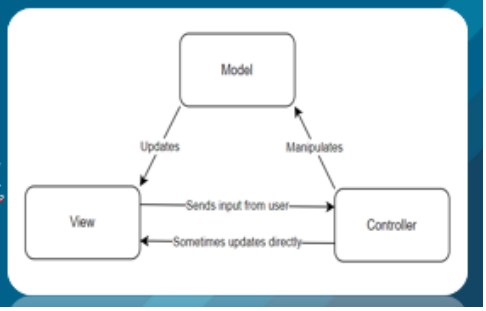
- Неінтуїтивний і складний інтерфейс, що ускладнює навігацію для користувачів. Обмежена функціональність в системі каталогування товарів.
- Обмеження: Відсутність інтеграції з платіжними системами, що призводить до неефективного процесу оплати замовлень.
- Недостатня адаптивність для різних пристроїв, що ускладнює доступ до магазину для користувачів з мобільних пристроїв. Обмежена можливість налаштування фільтрів та сортування в системі пошуку товарів.

Архітектура

Архітектура ASP.NET MVC (Model-View-Controller) є однією з популярних архітектурних підходів для розробки веб-застосунків. Вона надає розподілення обов'язків між компонентами застосунку, що сприяє покращенню модульності, розширюваності та обслуговуваності коду.

Основна ідея ASP.NET MVC полягає в розділенні застосунку на три основні компоненти:

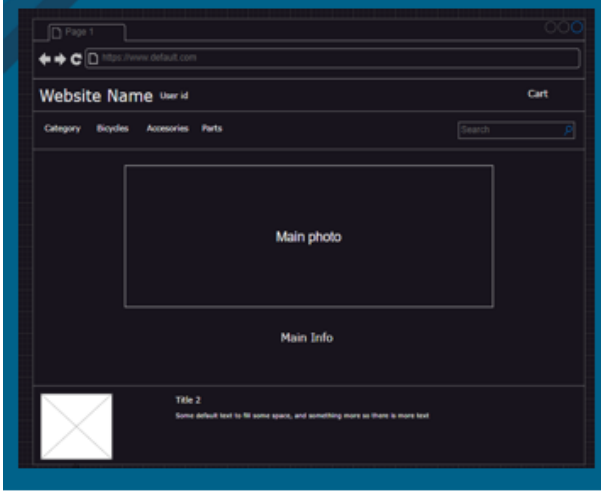
1. Модель (Model): Вона відповідає за представлення даних та бізнес-логіки застосунку. Модель може охоплювати доступ до бази даних, обробку даних та взаємодію з іншими джерелами даних.
2. Представлення (View): Це компонент, який відповідає за відображення інформації користувачу. Воно має візуальну форму, яка може бути HTML-сторінкою, шаблоном або іншими видами веб-інтерфейсу.
3. Контролер (Controller): Він обробляє вхідні запити від користувача, взаємодіє з моделлю для отримання необхідних даних та визначає, яке представлення повинно бути показано користувачу. Контролер також може виконувати логіку обробки полій.



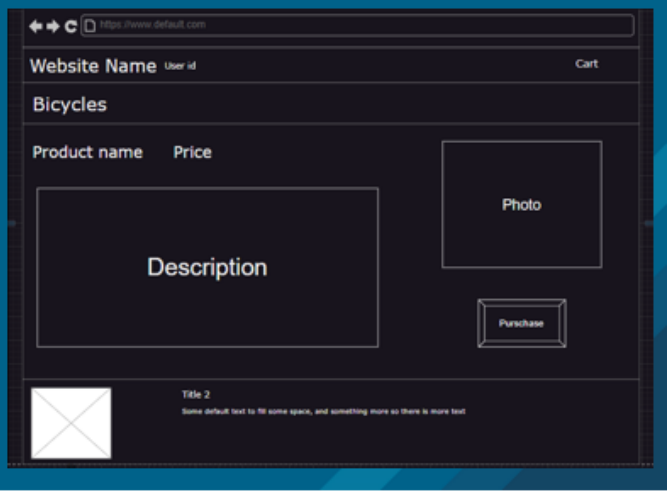
Макет інтерфейсу

Розроблено макети інтерфейсу, в яких я візуалізую структуру та розміщення елементів на сторінках застосунку на який показано загальний вигляд сторінок.

Макет головної сторінки



Макет сторінки перегляду товару

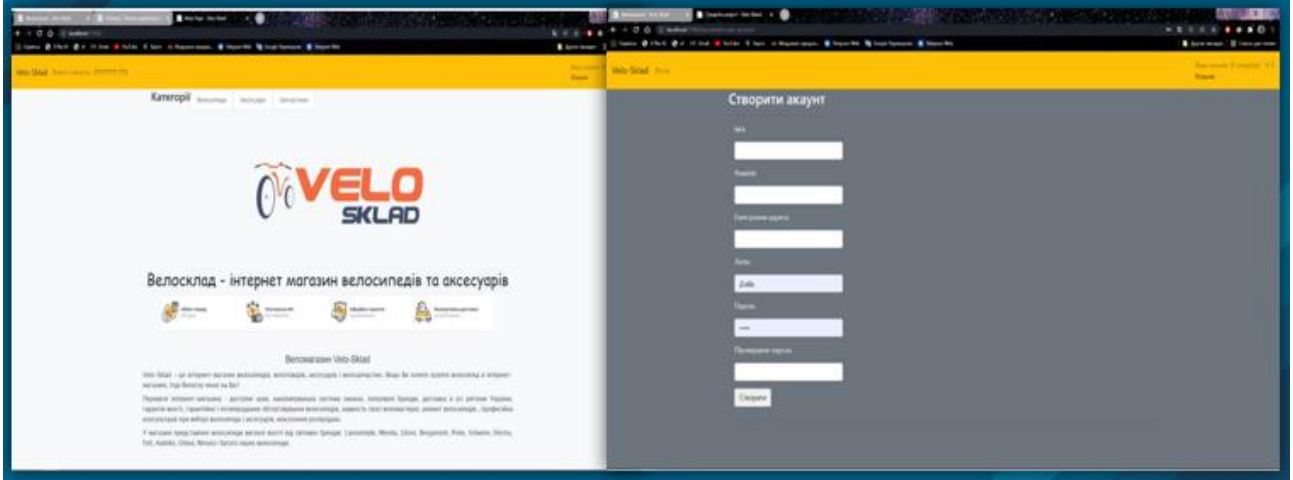


Користувацький інтерфейс	• Сторінка "Головна": відображає список категорій товарів та пропозиції. • Сторінка "Категорія товарів": відображає товари у вибраній категорії. • Сторінка "Деталі товару": показує повну інформацію про вибраний товар. • Сторінка "Кошик": дозволяє переглядати та керувати замовленнями користувача. • Сторінка "Оформлення замовлення": дозволяє користувачам оформити своє замовлення. • Сторінка "Управління замовленнями" (адміністративний доступ): дозволяє адміністраторам переглядати, редагувати та видаляти замовлення.
Компонент "Управління замовленнями"	• Перегляд замовлень: дозволяє адміністраторам переглядати список замовлень, включаючи інформацію про клієнтів, дату замовлення та загальну вартість. • Редагування замовлень: дозволяє адміністраторам редагувати існуючі замовлення, включаючи додавання, видалення або зміну кількості товарів. • Видалення замовлень: дозволяє адміністраторам видаляти замовлення, які більше не потрібні.
Адміністратор сайту	• Додавання нового замовлення: дозволяє клієнтам оформити нове замовлення, вибравши товари такількість, яку вони бажають придбати. • Перегляд замовлень: дозволяє адміністраторам переглядати список замовлень, включаючи інформацію про клієнтів, дату замовлення та загальну вартість. • Редагування замовлень: дозволяє адміністраторам редагувати існуючі замовлення, включаючи додавання, видалення або зміну кількості товарів. • Видалення замовлень: дозволяє адміністраторам видаляти замовлення, які більше не потрібні.

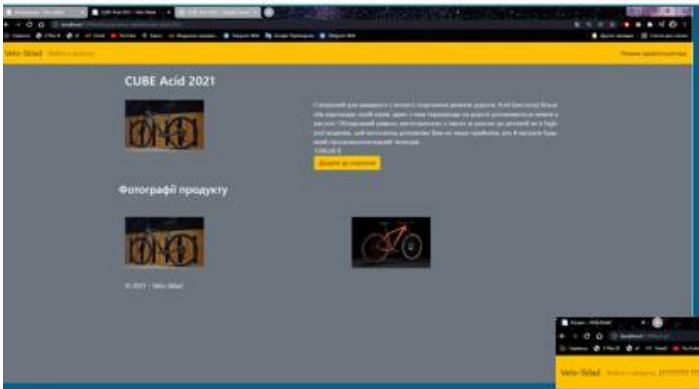
Результати роботи

Головна сторінка веб-застосунку

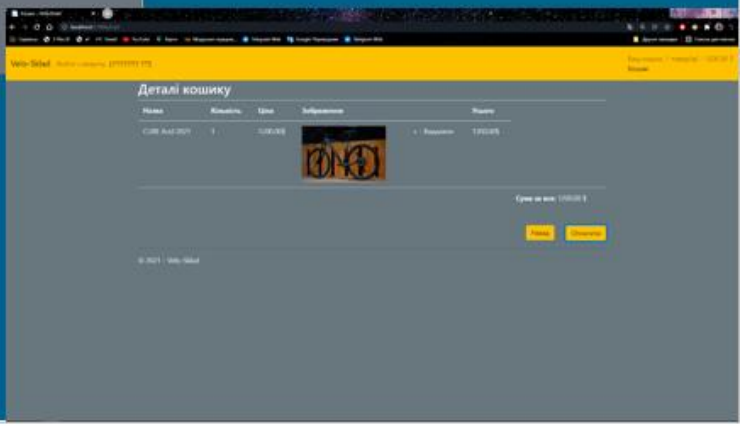
Сторінка реєстрації користувача



Вигляд сторінки товару для авторизованого користувача



Вигляд сторінки «Кошик»

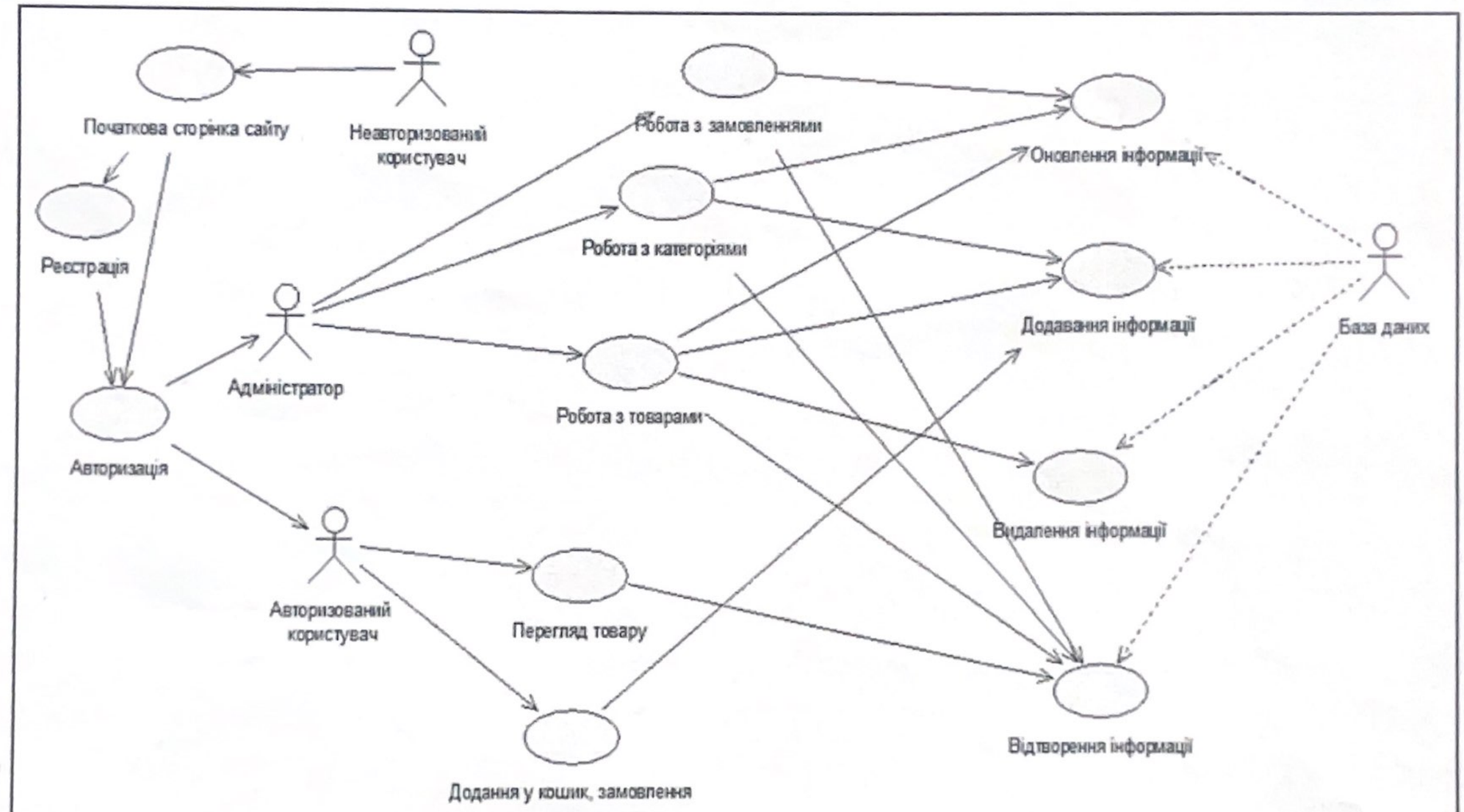


Висновки

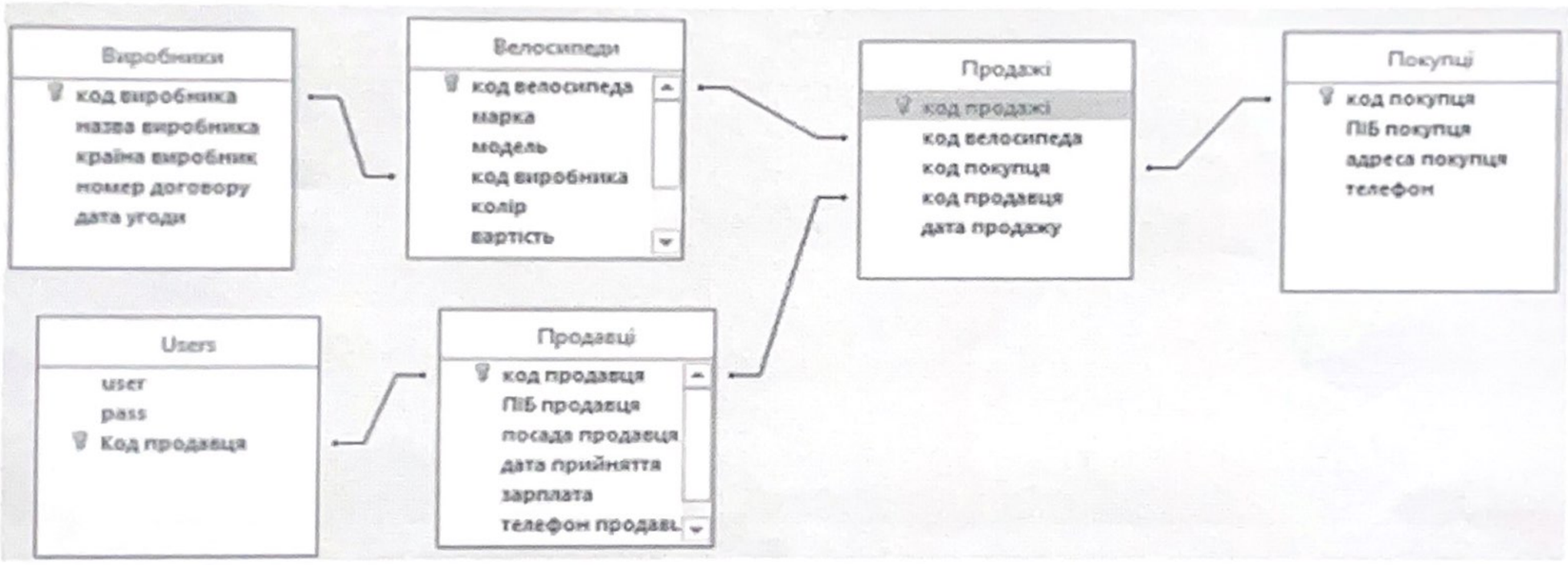
В ході дипломного проекту "Розробка магазину з продажу велотоварів" була успішно розроблена програмна система, яка забезпечує функціональність онлайн-магазину для продажу велотоварів.

- Результати проектування включають розроблені моделі, алгоритми, структурну схему, формати і алгоритми передачі та обробки даних, а також схему інтерфейсу, що відповідають потребам користувачів та найкращим практикам розробки програмного забезпечення.
- Розроблене ПЗ має такі переваги порівняно з існуючими рішеннями: зручний інтерфейс для користувачів, можливість розширення та налаштування функціоналу, ефективна обробка та передача даних, інтеграція з платіжною системою для забезпечення зручного способу оплати замовлень.
- Практична цінність розробленого ПЗ полягає у забезпеченні зручного та ефективного інтернет-магазину для продажу велотоварів, що дозволяє клієнтам зручно оглядати продукти, робити замовлення та здійснювати оплату онлайн. Крім того, система дозволяє власникам магазину ефективно керувати товарами, замовленнями та платежами, сприяючи покращенню їхньої роботи та задоволенню потреб клієнтів.

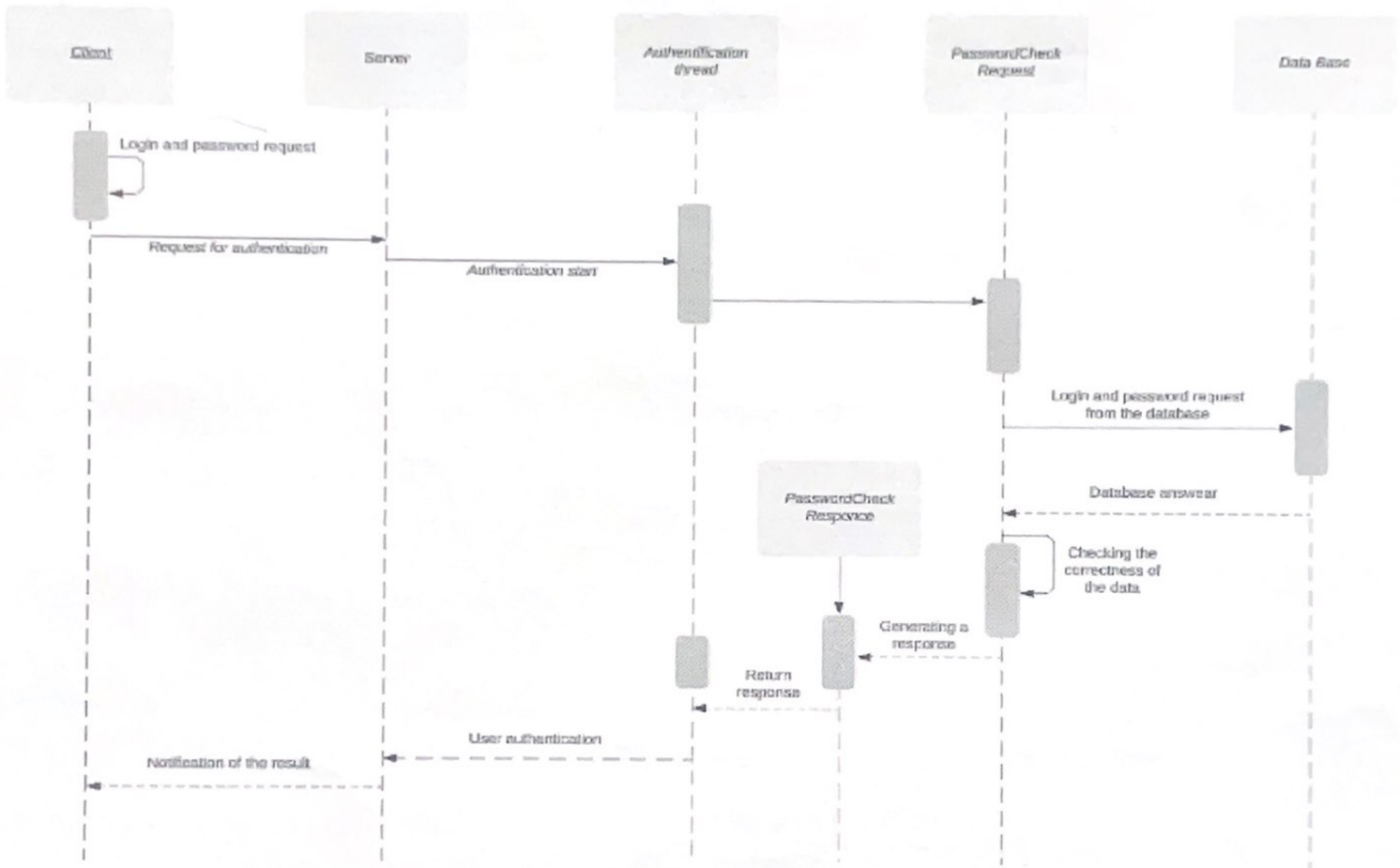
Дякую за увагу



					КвРІПЗ.190148.01.08.Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	UML — діаграма варіантів використання	Літера / Маса / Масштаб
Розробив		Авраменко О.О.				
Керівник		Бедратюк Л.П.				Аркуш 1 / Аркушів 3
Н. Контр.		Гурман І.В.				ХНУ, ІПЗ-19-1
Зав. Каф.		Бедратюк Л.П.				



					КвРІПЗ.190148.01.08.Е8		
Зм.	Арк.	№ докум.	Підпис	Дата	Діаграма зв'язків бази даних	Літера	Маса / Масштаб
Розробив		Янгінський О.О.		5.06			
Керівник		Бедратюк Л.П.		5.06		Аркуш 2	Аркушів 3
Н. Контр.		Гурман І.Б.		5.06		ХНУ, ІПЗ-19-1	
Зав. Каф.		Бедратюк Л.П.		5.06			



					КвРІПЗ.190148.01.08.Е8	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Зелінський С.О.		3.06	UML - Діаграма послідовностей	Літера / Маса / Масштаб
Керівник		Бадратюк Л.П.		3.06		Аркуш 3 / Аркушів 3
Н. Контр.		Гурман І.Б.		3.06		ХНУ, ІПЗ-19-1
Зав. Каф.		Бадратюк Л.П.		3.06		

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Зволянський Олександр Олександрович

Тема Веб-застосунок для продажу велотоварів

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 5 ; кількість сторінок записки 65

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі було досліджено і проаналізовано предметну область, визначено усі функціональні та нефункціональні вимоги. Був проведений аналіз існуючих програм на ринку, розглянуто їх переваги і недоліки, та доведено актуальність розробки нового програмного забезпечення. Розглянуто інструменти для реалізації спроектованих рішень, в результаті чого створено програмне забезпечення. Також було проведено тестування програми, за результатами якого доведено, що розроблене програмне забезпечення працює коректно та готове до експуатації.

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

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі 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 проведено аналіз сучасних архітектур, розглянуто їх переваги і недоліки та визначено, що система буде відповідати монолітній архітектурі та моделі клієнт-сервер. У третьому розділі підготовлено всі залежності для написання коду та виконано практичну розробку програмних модулів і описано їх особливості, в результаті чого створено програмний продукт. Було виконано тестування системи.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною, оскільки на сьогодні в Україні вже існують аналоги схожих сайтів, які успішно користуються попитом, вони полегшують процес покупки і водночас допомагають продавцю збувати продукцію набагато швидше.

5. Негативні сторони роботи Веб-застосунок з продажу велотоварів має деякі негативні аспекти, які потребують удосконалення. Одна з таких проблем полягає в обмеженому функціоналі фільтрів. Для поліпшення користувацького досвіду варто розширити можливості фільтрації, щоб клієнти могли зручно знаходити необхідні товари.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Матеріал пояснювальної записки структурований, послідовний, чіткий та простий, що дозволяє чітко зрозуміти викладений матеріал у рамках тематики проектування. Графічний матеріал дає можливість наочно побачити деталі проектування системи.

8. Інші зауваження

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

РЕЦЕНЗЕНТ Говорущенко Тетяна Олександрівна, доктор технічних наук, професор, зав.кафедри комп'ютерної інженерії та інформаційних систем (КПС) ХНУ

"5" червня 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: «Веб-застосунок для продажу велотоварів»
Автор: Зволянський Олександр Олександрович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Бедратюк Леонід Петрович, д. фіз.-мат. наук, проф
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) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 9,32% і адресується до 244 джерела з Інтернет і 63 джерела з Бібліотеки, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 2.06.2023

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Леонід БЕДРАТЮК

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

здобувача вищої освіти

Зволянський О.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.

01.06.2023
дата

підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 6%**

ID: 114587 Назва: БКР Веб-застосунок для продажу велотоварів Додано в БД: 2023-06-02 Автора: Зволянський О.О. Керівники: Бедратюк Л.П. д.ф-м.н. проф. Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	79842	680	2524 (3%)	34 (5%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1015397735

Дата перевірки:
02.06.2023 16:11:21 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
02.06.2023 16:20:27 EEST

ID користувача:
100005589

Назва документа: КвР Зволянський Антиплагіат

Кількість сторінок: 68 **Кількість слів: 12389** **Кількість символів: 97087** **Розмір файлу: 5.34 MB** **ID файлу: 1015061585**

9.32%
Схожість

Найбільша схожість: 4.82% з джерелом з Бібліотеки (ID файлу: 1015045648)

2.53% Джерела з Інтернету	244	Сторінка 70
8.35% Джерела з Бібліотеки	63	Сторінка 71

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел