КВАЛІФІКАЦІЙНА РОБОТА

Група ІПЗс-20
Ступар О.М.

2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Ступар Олександр Михайлович

УДК 004.378

Розробка веб-сайту з вбудованою CRM системою

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавра

Нормоконтроль

_____Стисло О.В.
(підпис, дата, розшифрування підпису)

Студент

_____Ступар О.М.
(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.
(підпис, дата, розшифрування підпису)

Керівник роботи

_____ к.т.н., доц. Демчина М. М.
(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

« ___ » _____ 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Ступар Олександр Михайлович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи

Розробка веб-сайту з вбудованою CRM системою

керівник роботи:

Демчина Микола Миколайович, кандидат технічних наук, доцент

затверджена наказом вищого навчального закладу від « 12» березня 2024 року № 19/1

2. Термін подання студентом роботи 05.06.2024

3. Вихідні дані роботи: мова програмування JavaScript, HTML, CSS, Python

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

1. Опис та порівняння існуючих аналогів

2. Вибір середовища, мови програмування та проектування архітектури веб-сайту

3. Програмна реалізація структури та функціоналу веб-сайту

5. Дата видачі завдання 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд існуючих аналогів	18.03.2024	Виконано
2.	Проектування та розробка мокапу	26.03.2024	Виконано
3.	Розробка дизайну веб-сайту	12.04.2024	Виконано
4.	Програмна реалізація веб-сайту	23.05.2024	Виконано
5.	Оформлення пояснювальної записки	29.05.2024	Виконано
6.	Оформлення графічного матеріалу та підготовка до захисту роботи	03.06.2024	Виконано

Студент _____ Ступар О.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Демчина М.М.
(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
16	Головна сторінка сайту Amazon	42	Дизайн сторінки із статтею
17	Сторінка категорій сайту	43	Головна сторінка особистого кабінету
18	Огляд товару	45	Приклад коду верстки основної сторінки
18	Головна сторінка сайту TD	46	HTML форми реєстрації
19	Форма зв'язку	47	Фрагмент використання Flask
36	Мокап основної сторінки	48	SQL – запити для створення таблиць
37	Мокап сторінки із статтею	49	Код SQLAlchemy
38	ERM діаграма	50	Код реєстрації користувачів
41	Дизайн основної сторінки	51	Код активності користувачів

АНОТАЦІЯ

У кваліфікаційній роботі розглядається процес розробки сайту-візитки, інтегрованого із CRM-системою, спеціально розробленого для афілійованої команди. Метою даної роботи є створення функціонального і зручного веб-ресурсу, який не тільки представляє діяльність команди, але й автоматизує управління взаємовідносинами з клієнтами.

У першому розділі дослідження наведено аналіз сучасних вимог до сайтів-візиток та CRM-систем, а також досвід їх інтеграції в різних галузях. Описано ключові функції, які необхідні для ефективної роботи афілійованої команди, такі як керування контактами, відстеження продажів та комунікацій, аналітика ефективності.

У другому розділі представлено процес розробки сайту, включаючи етапи планування та дизайну. Особливу увагу приділено вибору технологій та інструментів, що забезпечують безперебійну роботу і безпеку сайту та CRM-системи.

У заключній частині роботи підбито підсумки розробки та тестування сайту-візитки із CRM-системою, а також окреслено перспективи подальшого розвитку проекту. Відзначено досягнення поставлених цілей та потенційні напрямки для удосконалення функціоналу сайту.

Результатом виконаної роботи став готовий до використання веб-ресурс, який дозволяє афілійованій команді ефективно презентувати свою діяльність, автоматизувати процеси взаємодії з клієнтами та підвищити рівень обслуговування завдяки інтегрованій CRM-системі.

КЛЮЧОВІ СЛОВА: ВЕБ-САЙТ, HTML, CSS, JAVASCRIPT, FIGMA, КЛІЄНТ.

SUMMARY

The qualification work examines the process of developing a business card website integrated with a CRM system, specially designed for an affiliate team. The purpose of this work is to create a functional and convenient web resource that not only represents the team's activities but also automates the management of customer relationships.

The first section of the study analyzes the modern requirements for business card sites and CRM systems, as well as the experience of their integration in various industries. It describes the key functions that are necessary for the effective work of an affiliate team, such as contact management, sales and communication tracking, and performance analytics.

The second section presents the process of website development, including the planning and design stages. Special attention is paid to the choice of technologies and tools that ensure the smooth operation and security of the website and CRM system.

The final part of the paper summarizes the results of the development and testing of a business card website with a CRM system, as well as outlines the prospects for further development of the project. The achievement of the set goals and potential areas for improving the site's functionality were noted.

The result of the work done is a ready-to-use web resource that allows the affiliate team to effectively present their activities, automate customer interaction processes, and improve the level of service through an integrated CRM system.

KEY WORDS: WEBSITE, HTML, CSS, JAVASCRIPT, FIGMA, CLIENT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8
ВСТУП 9
РОЗДІЛ 1. ОПИС ТА ПОРІВНЯННЯ ІСНУЮЧИХ АНАЛОГІВ 12
1.1 Аналіз веб-сайтів за функціональними можливостями 12
1.2 Аналіз конкурентів 14
1.3 Постановка задачі 18
1.4 Загальний опис веб-сайту 20
Висновки до розділу 1 24
РОЗДІЛ 2. ВИБІР СЕРЕДОВИЩА, МОВИ ПРОГРАМУВАННЯ ТА ПРОЕКТУВАННЯ АРХІТЕКТУРИ ВЕБ-САЙТУ 25
2.1 Вибір середовища розробки 25
2.2 Обґрунтування вибору мови програмування 26
2.3 Проектування архітектури веб-сайту 28
2.4 Створення мокапу для веб-сайту 32
2.5 ERM діаграма 35
Висновки до розділу 2 36
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СТРУКТУРИ ТА ФУНКЦІОНАЛУ ВЕБ-САЙТУ 37
3.1 Розробка дизайну веб-сайту 37
3.2 Верстка веб-сайту 40
3.3 Програмування веб-сайту 42
3.4 Реалізація баз даних для особистого кабінету 44
3.5. Інтеграція API для обробки даних та взаємодії з іншими сервісами 49
3.6. Тестування веб-сайту 49
Висновки до розділу 3 51
ВИСНОВКИ 52
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 53

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTML – мова гіпертекстової розмітки;

UI – інтерфейс користувача;

CSS – каскадні таблиці стилів;

SCSS – препроцесор мови CSS;

JS – JavaScript;

БД – бази даних;

MVC – архітектурний шаблон;

Мокап – в контексті веб-розробки це детальний проект або модель, який демонструє функціональність системи або продукту.

ВСТУП

Актуальність теми. У сучасному цифровому світі наявність веб-сайту стала необхідністю для будь-якої команди чи бізнесу, незалежно від галузі діяльності. Це особливо актуально для сфери аффілейт маркетингу, де конкурентна перевага залежить від ефективної комунікації, організації та аналітики. Сайт-візитка з вбудованою CRM-системою для аффілейт команди є надзвичайно важливим інструментом, що дозволяє оптимізувати роботу, підвищити продуктивність та забезпечити високий рівень взаємодії з партнерами та клієнтами.

Аффілейт маркетинг – одна з най динамічніших галузей інтернет-маркетингу, яка постійно розвивається під впливом нових технологій та ринкових трендів. У цій сфері особливо важливо швидко адаптуватися до змін, ефективно керувати партнерськими програмами та слідкувати за результатами кампаній. Сайт-візитка з CRM-системою дозволяє централізувати всі ці процеси, забезпечуючи інтеграцію з різними платформами та інструментами.

Вбудована CRM-система надає можливість централізувати дані, зберігаючи всі контакти, угоди та комунікації в одному місці, що спрощує управління інформацією та забезпечує швидкий доступ до необхідних даних. Автоматизація рутинних завдань, таких як надсилання звітів, нагадувань та інших комунікацій, дозволяє зосередитися на стратегічних задачах. CRM-система також дозволяє відстежувати ключові метрики та показники ефективності, що допомагає приймати обґрунтовані рішення та коригувати стратегії в режимі реального часу. Крім того, спрощуються комунікації між членами команди та партнерами завдяки інтеграції різних каналів зв'язку.

Для команди, що працює у сфері аффілейт маркетингу, сайт-візитка з CRM-системою є потужним інструментом для підвищення ефективності роботи. Він дозволяє краще організувати діяльність, відстежувати результати та забезпечити високий рівень підтримки партнерів. В умовах постійного зростання

конкуренції та швидких змін у цифровому середовищі такі технологічні рішення є невід'ємною складовою успіху.

Отже, розробка сайту-візитки з CRM-системою не тільки відповідає сучасним тенденціям, але й є необхідним кроком для забезпечення конкурентоспроможності та ефективності роботи в довгостроковій перспективі.

Мета і завдання дослідження. Розробка та впровадження сайту-візитки з інтегрованою CRM-системою для команди, що займається аффілейт маркетингом, має на меті вивчення та врахування потреб користувачів у цій сфері. Основна мета полягає у створенні функціонального, привабливого та конкурентоспроможного веб-сайту, який задовольнятиме потреби сучасних учасників аффілейт маркетингу та сприятиме розвитку бізнесу в цій галузі.

Об’єкт дослідження. Об’єктом дослідження є процес розробки та впровадження сайту-візитки з інтегрованою CRM-системою. Особлива увага приділяється вивченню ефективності таких веб-рішень у покращенні комунікації, управлінні контактами, аналітиці та загальній організації роботи команди. Дослідження охоплює всі етапи створення сайту, від початкового аналізу вимог та проектування до програмування, та тестування.

Предмет дослідження. Включає в себе створення інтерфейсу користувача, розробку функціональності особистого кабінету, інтеграцію з системами передачі метрик та їх відслідковування, дослідження потреб та запитів користувачів та членів команд, які використовуватимуть даний продукт, забезпечення безпеки даних, аналіз використання доступний інструментів.

Методи дослідження. Для виконання завдання використовуватимусь такі інструменти як середовище Figma (для створення дизайну веб-сайту), а також будуть використані технології: мова гіпертекстової розмітки HTML, CSS – каскадна таблиця стилів, JavaScript для верстки клієнтської частини веб-сайту в середовищі Visual Studio, Python – мова програмування основної частини веб-сайту.

Практичне значення одержаних результатів. Результатом виконання кваліфікаційної роботи є готовий до реалізації сайт-візитка із CRM-системою.

Апробація результатів дослідження. Матеріали кваліфікаційної роботи були представлені на XI Міжнародній наукові конференції «СТУДЕНТСЬКІ НАУКОВІ ДИСКУСІЇ ПОЗА ФОРМАТОМ», яка відбулась 11 квітня 2024 року в Університеті Короля Данила.

Структура. Розділи – 3. Загальний обсяг основної частини – 48 сторінок. Список використаних джерел – 20.

РОЗДІЛ 1. ОПИС ТА ПОРІВНЯННЯ ІСНУЮЧИХ АНАЛОГІВ

1.1 Аналіз веб-сайтів за функціональними можливостями

У даному розділі проведено детальний аналіз функціональних можливостей різних веб-сайтів, враховуючи як CRM-системи, так і афілійні команди, що вже існують на ринку. Цей аналіз став основою для розробки кваліфікаційної роботи, яка вже успішно реалізована.

CRM-системи – (customer relationship management) є ключовим інструментом для будь-якої компанії, яка прагне підтримувати ефективні взаємовідносини з клієнтами. Ці системи не лише дозволяють збирати та аналізувати дані про клієнтів, а й автоматизують багато процесів управління продажами, маркетингом та обслуговуванням клієнтів.

Однією з переваг використання CRM-системи є можливість зберігати всю інформацію про клієнтів в одній централізованій базі даних. Це дозволяє співробітникам компанії легко отримувати доступ до важливої інформації, такої як контактні дані клієнтів, історія взаємодії, попередні покупки та пропозиції. У кваліфікаційній роботі було використав цю можливість CRM-систем для розробки модулю управління клієнтами, який дозволяє збирати та організовувати інформацію про користувачів веб-платформи.

Крім того, CRM-системи допомагають у плануванні та управлінні продажами. Вони надають інструменти для створення та відстеження угод, контролю над клієнтами та відстеження продажів. У кваліфікаційній роботі використовується функціонал CRM-систем для розробки модулю управління продажами, який допомагає веб-власникам відстежувати та керувати процесом продажів через їхню веб-платформу.

Також важливою характеристикою CRM-систем є їх здатність до аналізу даних. Вони надають звіти та аналітичні інструменти, які дозволяють компаніям отримувати інсайти щодо ефективності їхньої діяльності та виявляти можливості

для покращення. У кваліфікаційній роботі використано аналітичні функції CRM-систем для аналізу даних щодо взаємодії користувачів з веб-платформою та ефективності афілійного маркетингу.

Загалом, CRM-системи грають важливу роль у розробці та підтримці веб-платформ для афілійного маркетингу, допомагаючи підвищити ефективність взаємодії з клієнтами та оптимізувати управління продажами та маркетингом. У даній кваліфікаційній роботі було враховано ці можливості та успішно імплементовано їх у веб-платформу для досягнення мети.

Афілійні команди (affiliate programs) є важливим інструментом для веб-власників, які бажають заробляти кошти від продажу товарів та послуг через свої веб-сайти. Ці програми дозволяють веб-власникам отримувати комісійні за продажі, здійснені через їх реферальні посилання. Розглянемо деякі ключові функціональні можливості афілійних команд:

1. Створення реферальних посилань. Однією з основних можливостей афілійних програм є можливість створювати унікальні реферальні посилання для рекламних матеріалів. Ці посилання дозволяють веб-власникам відстежувати та отримувати винагороду за продажі, здійснені через їхні веб-сайти. У даній кваліфікаційній роботі враховано цю функціональність та розробив механізми для створення та керування реферальними посиланнями на веб-платформі для афілійного маркетингу.

2. Відстеження продажів та комісій. Афілійні програми надають інструменти для відстеження продажів та обчислення винагороди для афіліатів. Це дозволяє веб-власникам ефективно відстежувати виручку, генеровану через їх реферальні посилання, та відшкодовувати афіліатам відповідно до умов угоди. У кваліфікаційній роботі використано інструменти відстеження та аналізу продажів для реалізації системи комісійного винагородження для афіліатів.

3. Надання маркетингових матеріалів. Деякі афілійні програми надають своїм учасникам маркетингові матеріали, такі як банери, текстові оголошення та графічні зображення, щоб допомогти їм привернути увагу клієнтів та здійснити продажі. У кваліфікаційній роботі враховано можливість надання

маркетингових матеріалів для афіліатів та створив інтерфейс для їх використання та керування.

4. Аналітика та звітність. Деякі афілійні програми надають інструменти аналітики та звітності, які дозволяють веб-власникам аналізувати ефективність своїх рекламних кампаній. У кваліфікаційній роботі використано інструменти для реалізації звітності та аналітики, щоб веб-власники могли ефективно відстежувати та оптимізувати свої рекламні кампанії, відстежувати витрати та доходи рекламних кампаній.

Вищезгадані функціональні можливості афілійних команд грають важливу роль у підтримці та розвитку веб-платформ для афілійного маркетингу. Їхнє використання допомагає веб-власникам залучати та утримувати афіліатів, ефективно відстежувати продажі та оптимізувати свою діяльність для досягнення максимальних результатів. У кваліфікаційній роботі успішно враховано ці можливості та реалізувано їх для підтримки веб-платформи для афілійного маркетингу.

Даний аналіз був корисним у визначенні потреб та очікувань користувачів, а також у визначенні ключових функціональних можливостей моєї веб-платформи для успішного афілійного маркетингу.

1.2 Аналіз конкурентів

Аналіз конкурентів в сучасному бізнес-середовищі виявляється необхідним етапом стратегічного планування та розвитку будь-якого проекту або підприємства. В рамках даного розділу було здійснено глибокий аналіз ключових конкурентів на ринку афілійного маркетингу з метою зрозуміти їхні стратегії, функціональні можливості та переваги. Цей аналіз стане основою для формулювання стратегій власної веб-платформи для афілійного маркетингу, спрямованих на підвищення конкурентоспроможності та досягнення успіху на ринку.

Значення аналізу конкурентів, аналіз конкурентів має ключове значення для успіху будь-якої компанії чи проекту. Він допомагає виявити сильні та слабкі сторони конкурентів, їхні стратегії та тактики, а також прогнозувати можливі дії на ринку. На основі отриманих даних можна розробити ефективні стратегії розвитку, підвищення конкурентоспроможності та визначити свої унікальні переваги.

Аналіз веб-сайту: Amazon (URL: https://www.amazon.com).

Загальні відомості проєкту:

– дата заснування: 5 липня 1994 року;

– штаб-квартира: Сіетл, штат Вашингтон, США;

– кількість співробітників: 1,6 мільйона (станом на 2023 рік);

– галузь: електронна комерція, хмарні обчислення, штучний інтелект;

– продукти та послуги: продаж товарів, Amazon Prime, Amazon Web Services (AWS), Alexa.

Переваги проєкту включають:

– широкий вибір товарів: Amazon пропонує мільйони товарів у різних категоріях;

– конкурентні ціни: Amazon часто пропонує найкращі ціни на товари, завдяки своїй великій купівельній спроможності;

– зручність: Amazon пропонує зручний досвід покупок, з простим процесом замовлення та швидкою доставкою (рис. 1.1).

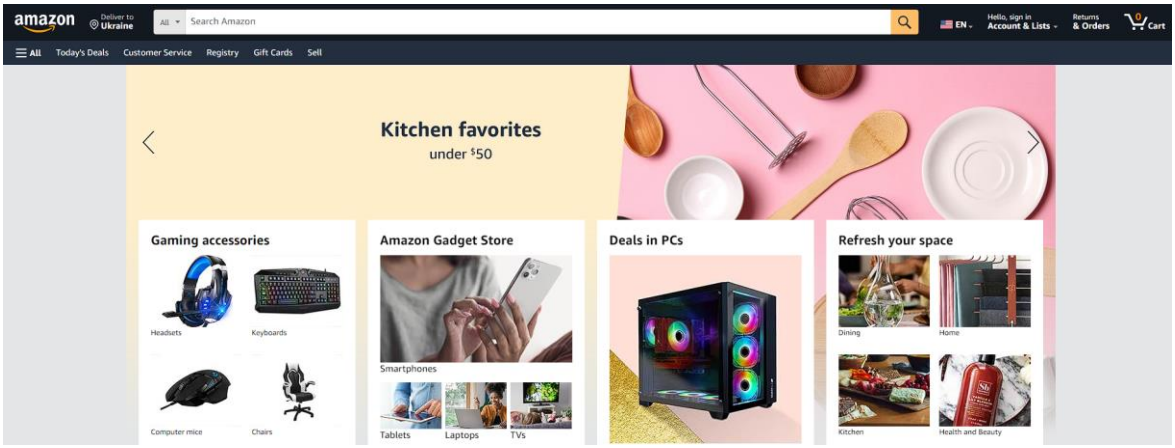


Рисунок 1.1 – Головна сторінка сайту Amazon

– надійність: Amazon має репутацію надійного продавця, з хорошою політикою повернення та обслуговування клієнтів.

Недоліки проєкту включають:

– складність навігації: на веб-сайті Amazon може бути складно знайти те, що ви шукаєте, через велику кількість товарів і категорій (рис. 1.2);

– підробки: на Amazon є ризик купити підроблені товари, особливо у сторонніх продавців;

– відгуки: не всі відгуки на Amazon є надійними, тому важливо бути обережним при їх читанні (рис. 1.3);.

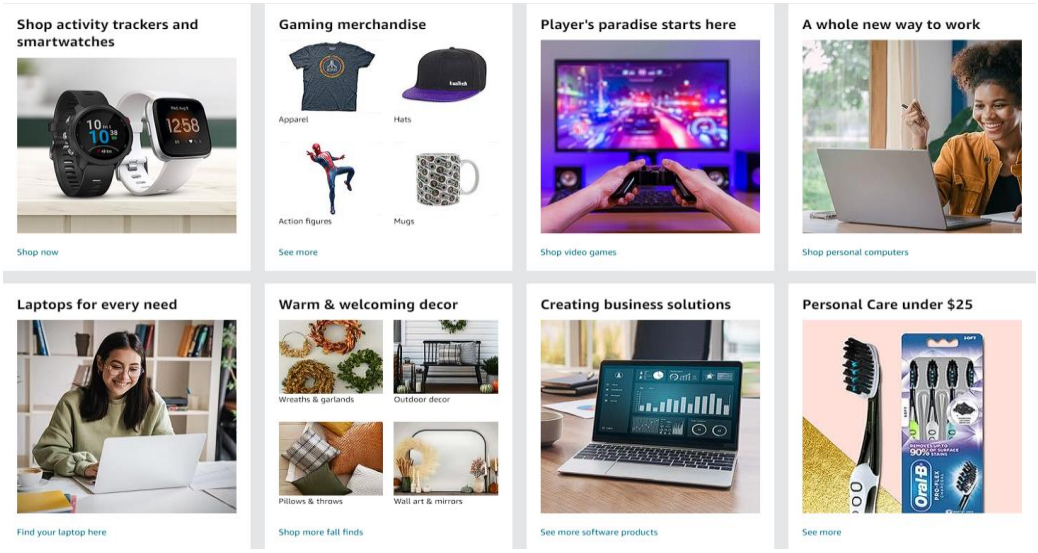


Рисунок 1.2 – Сторінка категорій сайту

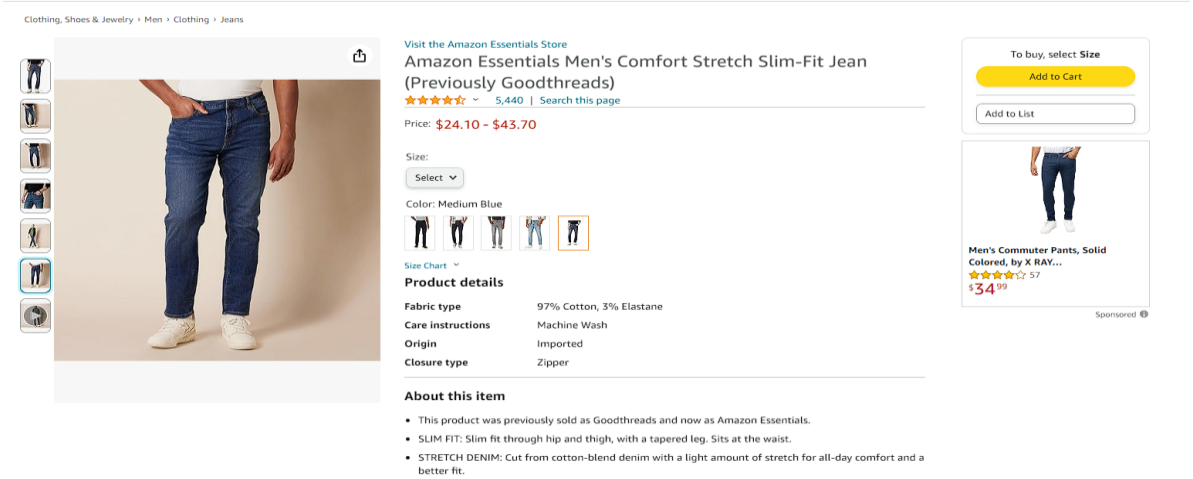


Рисунок 1.3 – Огляд товару

Аналіз веб-сайту: Trafic Devils (URL: https://devils.agency/) (рис. 1.5);

Загальні відомості проєкту:

– дата заснування: 2012 рік;

– штаб-квартира: Київ, Україна;

– кількість співробітників: більше п’ятдесяти;

– галузь: цифровий маркетинг, розробка веб-сайтів, SEO, PPC.

Продукти та послуги: розробка веб-сайтів, SEO, PPC, SMM, контент-маркетинг, email-маркетинг.

Рисунок 1.5 – Головна сторінка сайту TD

Переваги проєкту включають:

– досвід: devils.agency має більше 10 років досвіду роботи в сфері цифрового маркетингу;

– комплексний підхід: devils.agency пропонує широкий спектр послуг цифрового маркетингу, що може допомогти компаніям будь-якого розміру;

– результати: devils.agency має досвід досягнення успішних результатів для своїх клієнтів;

– відгуки: devils.agency має хороші відгуки від своїх клієнтів.

Недоліки проєкту включають:

– ціна: послуги devils.agency можуть бути дорогими для деяких компаній;

– портфоліо: портфоліо devils.agency не є публічним, тому важко оцінити якість їх роботи (рис. 1.6);

– відкритість: devils.agency не публікує багато інформації про свою команду та методи роботи.

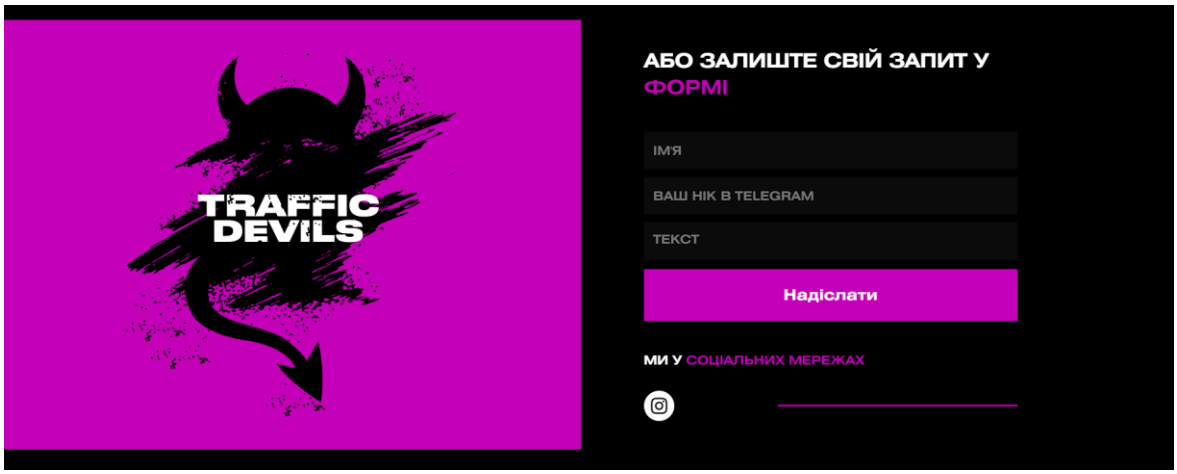


Рисунок 1.6 – Форма зв’язку

1.3 Постановка задачі

Постановка задачі є ключовим етапом у розробці веб-сайту з вбудованою CRM-системою та адаптацією під афілійний маркетинг. У цьому розділі робиться акцент на чіткому визначенні завдань, які потрібно вирішити під час розробки та реалізації проекту.

Головна мета проєкту:

– створення веб-сайту, який буде поєднувати в собі високий рівень функціональності, вбудовану CRM-систему та можливості афілійного маркетингу з метою ефективного управління клієнтською базою та збільшення обсягів продажів.

Ключові задачі проєкту:

– розробка функціональних можливостей веб-сайту: детальне вивчення потреб цільової аудиторії та визначення функціональних вимог до веб-сайту. Розробка інтерфейсу користувача, який буде зручним та інтуїтивно зрозумілим для відвідувачів. Реалізація основних функцій, таких як реєстрація користувачів, замовлення товарів або послуг, оплата та доставка;

– вбудована CRM-система: вибір та інтеграція CRM-системи, яка найкращим чином відповідає потребам бізнесу. Налаштування CRM-системи для збору та аналізу даних про клієнтів, керування продажами та взаємодії з афіліатами. Інтеграція CRM-системи з іншими компонентами веб-сайту для автоматизації процесів;

– адаптація під афілійний маркетинг: розробка системи відстеження та обліку продажів через афіліатів. Налаштування комісійних та бонусних програм для афіліатів з метою стимулювання їх активності. Розробка та надання маркетингових матеріалів для афіліатів для ефективного просування продуктів та послуг;

– тестування та впровадження: проведення комплексного тестування всіх функціональних можливостей веб-сайту та CRM-системи. Виправлення помилок та вдосконалення функціоналу на основі отриманих результатів тестування. Поступове впровадження рішення та навчання персоналу з користування новими системами;

– підтримка та подальший розвиток: забезпечення технічної підтримки та вирішення можливих проблем після впровадження. Постійний моніторинг роботи системи та здійснення необхідних оновлень та поліпшень. Розробка та впровадження нових функцій та можливостей відповідно до потреб бізнесу та змін на ринку.

1.4 Загальний опис веб-сайту

Даний сайт розроблений з метою забезпечити ефективне управління афіліатною програмою та сприяти залученню нових партнерів, що сприятиме зростанню продажів та розширенню бізнесу. Основні функціональні можливості веб-сайту включають керування афіліатами, відстеження продажів, налаштування комісійних, надання маркетингових матеріалів, а також звіти та аналітику.

Перш за все, важливо визначити процес керування афіліатами. Веб-сайт надає зручний інтерфейс для реєстрації та керування афіліатами, що дозволяє адміністраторам ефективно управляти партнерською мережею. Вони можуть створювати, редагувати та видаляти профілі афіліатів, встановлювати умови співпраці та відстежувати їхні успіхи.

Другий важливий аспект мого веб-сайту – це відстеження продажів через вбудовану CRM-систему. Ця функція виявляється надзвичайно корисною для ефективного управління афіліатною програмою та моніторингу ефективності кожного афіліата.

– по-перше, завдяки вбудованій CRM-системі, мій веб-сайт автоматично фіксує кожен продаж, який здійснюється через афіліатів. Кожен покупець, який переходить на мій сайт через унікальне посилання афіліата та робить покупку, автоматично реєструється в системі.

– далі, система аналізує ці дані та визначає, який саме афіліат був причетний до кожного продажу. Це дозволяє точно визначити обсяги продажів кожного афіліата та встановити відповідну комісію за їхню участь у генерації продажів або клієнтів.

– покроковий аналіз кожного продажу дозволяє зрозуміти, які афіліати найбільш успішні та ефективні у своїй діяльності. Це дозволяє вчасно виявляти та винагороджувати успішних афіліатів, а також розробляти стратегії для підтримки менш успішних партнерів.

– більше того, вбудована CRM-система надає можливість детального аналізу продажів та створення різноманітних звітів та аналітики. Можливість переглядати статистику продажів за різними періодами часу, визначати успішні та невдалі акції та стратегії, а також прогнозувати подальші тенденції розвитку бізнесу.

– крім того, CRM-система дозволяє ефективно взаємодіяти з афіліатами та надавати їм необхідну підтримку та інформацію. Завдяки цьому забезпечується висока якість обслуговування та сприяють побудові плідної співпраці з кожним афіліатом.

– наступний пункт, який варто розглянути більш детально – це налаштування комісійних. Адміністратори веб-сайту мають можливість встановлювати різні ставки комісійних для різних категорій товарів або послуг. Це дає їм гнучкість і можливість адаптувати стратегію компенсації до потреб свого бізнесу та специфіки продуктів чи послуг, які пропонуються.

– одним із важливих аспектів налаштування комісійних є можливість встановлення різних рівнів комісій за різні види діяльності або продукти. Наприклад, для продажу товарів вищої цінової категорії може бути встановлено вищий рівень комісій, щоб стимулювати афіліатів працювати з цими продуктами більш інтенсивно. У той же час, для просування нових товарів чи послуг адміністратор може встановити більш вигідні умови співпраці, щоб привернути увагу афіліатів і збільшити їхню активність.

– більше того, налаштування комісійних може включати в себе встановлення бонусних систем для стимулювання афіліатів до активної роботи. Наприклад, адміністратор може встановити бонус за досягнення певного обсягу продажів або за досягнення певного рівня активності. Це може стати ефективним інструментом мотивації для афіліатів та сприяти зростанню обсягів продажів.

– окрім того, система налаштування комісійних може забезпечити можливість автоматизації процесу встановлення та зміни комісійних. Наприклад, адміністратор може встановити правила, які автоматично змінюють рівень комісійних в залежності від різних факторів, таких як тип клієнта, тип

продукту або обсяги продажів. Це дозволяє ефективно управляти комісійними та швидко реагувати на зміни в ринкових умовах. Узагальнюючи, налаштування комісійних є важливою складовою афіліатної програми і надає адміністраторам веб-сайту можливість адаптувати стратегію компенсації до своїх потреб та досягти більшого успіху у приверненні та утриманні афіліатів.

Крім того, веб-сайт надає різноманітні маркетингові матеріали для афіліатів з метою підтримки їх у просуванні продуктів та послуг. Ця функціональність створює додаткові можливості для ефективного маркетингу та забезпечує афіліатам необхідні інструменти для успішної реклами.

– однією з ключових складових цієї функціональності є надання готових маркетингових матеріалів у різних форматах. Це включає в себе різноманітні банери різних розмірів та дизайнів, які можуть бути використані для розміщення на веб-сайтах або в інтернет-рекламних кампаніях. Також надаються текстові оголошення, які можна використовувати для написання рекламних повідомлень на соціальних медіа чи в електронних листах.

– крім того, на веб-сайті пропонуються готові лендінги – це спеціальні сторінки, які максимально ефективно презентують продукт чи послугу та стимулюють відвідувачів до здійснення покупки або дії. Ці лендінги мають вже готовий контент та дизайн, що значно полегшує завдання для афіліатів у просуванні продуктів.

– крім того, маркетингові матеріали можуть включати в себе готові статті, рекламні буклети, відео-презентації та інші рекламні матеріали, які можуть бути використані для залучення уваги цільової аудиторії та стимулювання її до покупки.

– важливо відзначити, що доступ до цих маркетингових матеріалів забезпечується через особистий кабінет афіліата на веб-сайті. Це робить процес отримання та використання маркетингових матеріалів максимально зручним та доступним для кожного афіліата.

– узагальнюючи, надання різноманітних маркетингових матеріалів для афіліатів є важливим елементом успішної афіліатної програми. Ця

функціональність допомагає афіліатам ефективно просувати продукти та послуги, забезпечуючи їм необхідні інструменти та ресурси для успіху.

Адміністратор має доступ до різноманітних звітів та аналітичних даних щодо продажів та активності афіліатів. Ця функціональність надає адміністратору веб-сайту важливий інструмент для ефективного управління програмою афіліатів та моніторингу її результативності.

– по-перше, адміністратор може отримати детальні звіти про продажі, які здійснюються через афіліатів. Це включає інформацію про кількість продажів, суму прибутку, обсяги продажів за певний період часу та інші ключові показники. Ці дані дозволяють адміністратору здійснювати аналіз ефективності афіліатської програми та визначати стратегії для подальшого розвитку.

– другий аспект – це аналітичні дані щодо активності афіліатів. Адміністратор може отримати інформацію про кількість переходів, реєстрацій та інших дій, здійснених афіліатами на веб-сайті. Ці дані дозволяють відстежувати активність кожного афіліата та визначати їх внесок у загальний обсяг продажів.

– крім того, адміністратор може отримати звіти про ефективність рекламних кампаній, які були запущені або підтримані афіліатами. Це включає інформацію про кількість переходів з кожної кампанії, конверсійні показники та інші метрики ефективності. Ці дані допомагають адміністратору оцінити ефективність рекламних матеріалів та визначити найбільш успішні стратегії маркетингу.

– загалом, доступ до різноманітних звітів та аналітичних даних є важливим інструментом для ефективного управління програмою афіліатів. Ця функціональність дозволяє адміністратору вчасно реагувати на зміни в ринкових умовах, оптимізувати стратегії маркетингу та максимізувати результативність афіліатської програми.

Отже, загальний опис веб-сайту надає вичерпну інформацію про його функціональність та можливості. Він покликаний стати ефективним інструментом для розвитку бізнесу та забезпечити його успіх на ринку афілійного маркетингу.

Висновки до розділу 1

У розділі 1 було проведено аналіз та порівняння існуючих аналогів веб-сайтів з вбудованою CRM-системою та адаптацією під афілійний маркетинг.

Під час аналізу функціональних можливостей існуючих веб-сайтів були виявлені їх переваги та недоліки. Особлива увага була приділена опису загальної концепції веб-сайту та постановці завдань, які потрібно вирішити під час розробки та реалізації проекту.

Далі буде розглянуто вибір середовища розробки, мови програмування та проектування архітектури веб-сайту. Вибір технічних рішень буде здійснений з урахуванням специфіки проекту та його вимог.

Для реалізації функціональності веб-сайту буде розроблено дизайн, виконана верстка та програмування, а також здійснено наповнення веб-сайту контентом. Після цього буде проведено тестування, щоб перевірити коректність роботи всіх функцій та виявити та виправити помилки.

Загальний опис веб-сайту дозволяє зрозуміти основні функції та можливості, які будуть доступні користувачам та адміністраторам. Подальший аналіз конкурентів надав можливість виявити сильні та слабкі сторони їхніх рішень та врахувати їх під час розробки власного веб-сайту.

В цілому, розділ 1 надає загальне уявлення про контекст, цілі та завдання дослідження, що буде детальніше розглянуто та реалізовано в наступних розділах кваліфікаційної роботи.

РОЗДІЛ 2. ВИБІР СЕРЕДОВИЩА, МОВИ ПРОГРАМУВАННЯ ТА ПРОЕКТУВАННЯ АРХІТЕКТУРИ ВЕБ-САЙТУ

2.1 Вибір середовища розробки

Під час створення веб-сайту, вибір оптимального середовища розробки грає вирішальну роль у забезпеченні продуктивності та ефективності процесу розробки. У даному випадку, було враховано різноманітні фактори та критерії, щоб забезпечити оптимальний вибір.

Першим критерієм були функціональні можливості середовища. Важливою була наявність інструментів для зручного написання, тестування та налагодження коду. Також врахована можливість інтеграції з додатковими розширеннями та плагінами, які дозволяють розширити функціональність та підвищити зручність роботи.

Другим критерієм була зручність використання. Інтерфейс та робочий процес у середовищі розробки повинні бути інтуїтивно зрозумілими та зручними для розробників різного рівня досвіду.

Підтримка спільної роботи також була важливою. Було враховано можливість використання спеціалізованих сервісів та інструментів для спільної роботи над кодом, щоб забезпечити ефективну розробнику.

Наступним критерієм була швидкість та продуктивність. Ці аспекти визначають швидкість розробки та час налагодження та тестування.

Після ретельного аналізу було прийняте рішення використовувати Visual Studio Code. Він ідеально відповідає всім вищезазначеним критеріям та має додаткові переваги, такі як широкий вибір розширень та плагінів, підтримка багатьох мов програмування, інтеграція з системами контролю версій та активна спільнота користувачів.

Visual Studio Code (VS Code) – це безкоштовний, легкий у використанні та потужний редактор коду, розроблений компанією Microsoft. Він є

одним з найпопулярніших інтегрованих середовищ розробки (IDE) серед програмістів усього світу.

Відмінною особливістю Visual Studio Code є його широкі можливості розширення. Він підтримує багато різних мов програмування, таких як JavaScript, Python, HTML/CSS, Java та інші, що робить його універсальним інструментом для розробки в різних областях програмування.

Однією з найбільших переваг VS Code є його вбудована інтеграція з системами контролю версій, такими як Git. Це дозволяє розробникам легко вести керування версіями свого коду прямо з інтерфейсу редактора.

Крім того, Visual Studio Code має багато корисних функцій, таких як автоматичне доповнення коду, інтегровані вікна терміналу, можливість встановлення різних розширень для роботи з різними технологіями та платформами.

Ще однією перевагою є широка підтримка спільноти. Visual Studio Code має активну спільноту користувачів, яка постійно розробляє нові розширення та плагіни, а також надає підтримку та допомогу одне одному через форуми, чати та інші спільнотні ресурси.

У підсумку, Visual Studio Code – це потужний, легкий у використанні та розширюваний редактор коду, який відмінно підходить для розробки різноманітних проектів, незалежно від мови програмування чи платформи.

2.2 Обґрунтування вибору мови програмування

При обґрунтуванні вибору мови програмування для розробки веб-сайту на пайтоні, враховувалось кілька ключових факторів, що визначають ефективність роботи та результативність проекту.

Перш за все, Python – це потужна та динамічна мова програмування, яка має широкий спектр застосувань. Вона відома своєю простотою та зрозумілістю синтаксису, що сприяє швидкому розвитку та підтримці коду.

Крім того, Python має велику кількість сторонніх бібліотек та модулів, які значно спрощують роботу з різними аспектами розробки веб-сайту, такими як обробка HTTP-запитів, робота з базами даних, рендерінг HTML сторінок тощо.

Другим важливим аспектом є широка підтримка Python веб-фреймворків. Наприклад, фреймворки, такі як Django та Flask, надають потужні інструменти для швидкої розробки веб-застосунків, що базуються на мові Python. Вони пропонують готові рішення для ряду типових задач, таких як маршрутизація URL-адрес, обробка форм, автентифікація користувачів, інтеграція з базами даних та інші.

Крім того, Python – це мова програмування з великою спільнотою розробників. Існує безліч джерел документації, підручників та онлайн-курсів, що полегшують вивчення та використання мови програмування.

З огляду на ці фактори, вибір Python для розробки веб-сайту є обґрунтованим рішенням. Його простота, потужність та розширюваність роблять його ідеальним вибором для реалізації різноманітних проектів веб-розробки.

Фреймворк Flask – це легкий, ефективний та розширюваний інструмент для розробки веб-додатків на мові програмування Python. Він є одним з найпопулярніших веб-фреймворків і здобув велику популярність серед розробників завдяки своїй простоті та гнучкості.

Однією з головних переваг Flask є його мінімалістичний дизайн. Він надає базовий набір функцій та компонентів, що дозволяє розробникам вільно структурувати свій код та впроваджувати лише ті функції, які необхідні для конкретного проекту. Це робить Flask ідеальним вибором для створення як простих, так і складних веб-додатків, залежно від потреб проекту.

Ще однією перевагою Flask є його простота в навчанні та використанні. Він має добре організовану документацію та активну спільноту розробників, яка завжди готова надати допомогу та поради. Це робить Flask чудовим вибором для початківців, а також для досвідчених розробників, які шукають ефективний інструмент для своїх проектів.

Незважаючи на свою простоту, Flask дозволяє створювати потужні веб-додатки з різноманітним функціоналом. Він має вбудовану підтримку шаблонів Jinja2, яка дозволяє легко рендерити HTML-сторінки та працювати зі змінними в шаблонах. Також Flask надає можливість роботи з базами даних, обробки HTTP-запитів, автентифікації користувачів та багато іншого, завдяки великій кількості розширень та плагінів.

Фреймворк Flask також відкритий для розширення та налаштування. Він дозволяє розробникам легко створювати власні розширення та плагіни, що дозволяє розширити його функціональність та використовувати його для різних типів проектів.

У підсумку, Flask – це потужний та ефективний веб-фреймворк, який надає розробникам гнучкість, простоту та можливості для створення різноманітних веб-додатків на мові програмування Python. Його мінімалістичний дизайн та простота використання роблять його відмінним вибором для будь-якого проекту.

2.3 Проектування архітектури веб-сайту

Під час проектування архітектури веб-сайту, враховано низку важливих аспектів, які визначатимуть структуру та функціональність мого проекту. Основною метою цього етапу є створення гнучкої, ефективної та масштабованої архітектури, яка відповідає вимогам проекту та забезпечує оптимальну роботу веб-додатку.

MVC. Першим і найважливішим кроком у процесі проектування архітектури веб-сайту було визначення архітектурного патерну, який відповідав би потребам мого проекту. Після ретельного аналізу різних можливостей я вирішив скористатися патерном Model-View-Controller (MVC). Цей патерн вже давно використовується веб-розробниками і визнаний як ефективний із забезпеченням гнучкості та легкості управління кодом.

MVC розділяє логіку програми на три основні компоненти: модель (model), представлення (view) та контролер (controller). Модель відповідає за обробку

даних, взаємодію з базою даних та виконання бізнес-логіки. Представлення відображає інформацію для користувача та призначене для представлення даних з моделі. Контролер відповідає за обробку вхідних даних, взаємодію з моделлю та оновлення представлення відповідно до дій користувача.

Однією з основних переваг використання патерна MVC є його здатність до розділення великих проектів на менші, більш керовані частини. Це сприяє полегшенню розробки, тестування та підтримки коду, оскільки кожен компонент виконує конкретну функцію і може бути змінений або модифікований незалежно від інших. Такий підхід дозволяє зберігати код чистим, організованим і легко зрозумілим.

Крім того, використання патерна MVC сприяє виділенню логіки бізнес-процесів від представлення даних, що робить код більш переносимим та масштабованим. Це дозволяє ефективно управляти розробкою проекту, забезпечуючи його гнучкість та готовність до змін.

Таким чином, використання патерна Model-View-Controller є важливим кроком у процесі проектування архітектури мого веб-сайту. Його гнучкість та легкість управління кодом дозволять побудувати масштабовану та ефективну систему, яка відповідатиме вимогам проекту та задовольнить потреби користувачів.

Після вибору патерна MVC та визначення основної архітектури веб-сайту переходимо до проектування бази даних, яка є ключовою складовою будь-якого веб-проекту. Для цього скористаємось реляційною моделлю даних, оскільки вона дозволяє ефективно зберігати та організовувати інформацію.

Першим кроком у розробці схеми бази даних було визначення необхідних таблиць та зв'язків між ними. Для цього проаналізовано основні сутності веб-сайту, такі як користувачі, продукти, замовлення та інші, і визначимо, як вони взаємодіють один з одним.

Далі створимо схему бази даних, відображаючи всі необхідні таблиці та зв'язки між ними. Кожна таблиця представляє собою окрему сутність або об'єкт, який містить певну інформацію, наприклад, дані користувачів, категорії товарів,

замовлення тощо. Зв'язки між таблицями відображають взаємозв'язки між сутностями, такі як один до одного, один до багатьох або багато до багатьох.

Особлива увага приділялася можливості масштабування бази даних. Це означає, що структура бази даних має бути спроектована таким чином, щоб забезпечити оптимальну продуктивність при зростанні обсягу даних. Для цього було використано нормалізовану структуру даних, уникав зайвої денормалізації та ретельно планував індексацію таблиць.

У підсумку, розробка схеми бази даних була важливим кроком у проектуванні архітектури веб-сайту. Її структура дозволить ефективно зберігати та організовувати дані, забезпечуючи оптимальну продуктивність та масштабованість проекту.

Основними компонентами архітектури є модулі, які відповідають за роботу з користувачами, продуктами та замовленнями. Кожен з цих модулів має власну чітко визначену функцію та фокусується на певній групі операцій.

Перший модуль – модуль користувачів, відповідає за управління реєстрацією, входом в систему, аутентифікацією та авторизацією користувачів. Цей модуль забезпечує можливість створення нових облікових записів, зміну персональної інформації та керування правами доступу.

Другий модуль – модуль продуктів, відповідає за управління асортиментом товарів або послуг, їх додавання, редагування та видалення. Цей модуль також може містити функції для категоризації продуктів, додавання зображень, описів та інших характеристик.

Третій модуль – модуль замовлень, відповідає за обробку замовлень користувачів, їх створення, перегляд, редагування та видалення. Цей модуль також може включати функції для відстеження статусу замовлень, розрахунку вартості та оплати, а також генерації звітності.

Кожен з цих модулів розроблений таким чином, щоб бути незалежним від інших частин системи. Це забезпечує гнучкість та масштабованість системи, оскільки зміни в одному модулі не впливають на інші, і дозволяє легко розширювати функціональність системи за потребою.

Також приділено особливу увагу питанням безпеки та захисту даних користувачів. У проекті враховано важливість цих аспектів і розроблено низку механізмів для забезпечення безпеки і конфіденційності даних.

Одним із ключових заходів є механізми аутентифікації та авторизації користувачів. Для цього було використано надійні протоколи, такі як OAuth або JWT (JSON Web Tokens), які дозволяють перевіряти ідентичність користувачів та керувати їх доступом до різних ресурсів системи.

Крім того, для захисту конфіденційної інформації від несанкціонованого доступу використано різні методи шифрування та хешування даних. Наприклад, паролі користувачів зберігалися у захешованому вигляді, що забезпечує їхню безпеку навіть у випадку проникнення до бази даних.

При проектуванні архітектури системи також врахована можливість внутрішніх та зовнішніх загроз та застосував відповідні заходи захисту. Наприклад, встановлено механізми контролю доступу до різних ресурсів системи, а також резервне копіювання даних для забезпечення їхньої безпеки та відновлення у разі необхідності.

У результаті, дані заходи забезпечують високий рівень безпеки та захисту даних користувачів у проекті, що є важливим аспектом для будь-якої веб-платформи.

У підсумку, проектування архітектури веб-сайту – це складний, але надзвичайно важливий етап розробки, який визначає майбутню стійкість, ефективність та функціональність проекту. Правильно спроектована архітектура визначає, як система буде взаємодіяти з користувачами, як вона буде масштабуватися з часом та як ефективно вона буде виконувати свої завдання.

Під час проектування архітектури важливо врахувати потреби користувачів, бізнес-вимоги, а також технічні обмеження. Ретельне планування та аналіз дозволяють створити оптимальну структуру системи, яка буде відповідати всім вимогам та очікуванням.

За допомогою правильної архітектури можна забезпечити оптимальну роботу веб-додатку, зменшити витрати на його підтримку та розвиток, а також забезпечити зручність у користуванні для кінцевих користувачів.

Розроблені модулі для роботи з користувачами, продуктами та замовленнями гарантують легкість управління та розширення системи у майбутньому. Кожен модуль має чітко визначену функцію та взаємодіє з іншими компонентами системи через визначені інтерфейси.

Особливу увагу приділено питанням безпеки та захисту даних. Механізми аутентифікації та авторизації користувачів, а також шифрування та хешування конфіденційної інформації забезпечують надійний захист системи від потенційних загроз та зловмисних атак.

Отже, ретельно продумана архітектура забезпечує оптимальну роботу веб-сайту та задовольняє потреби користувачів. Це є ключовим чинником успішного розвитку та функціонування будь-якого веб-сайту.

2.4 Створення мокапу для веб-сайту

Створення мокапу (макету) є одним із ключових етапів у процесі розробки веб-сайту. Мокап дозволяє візуалізувати дизайн та функціональність майбутнього веб-сайту, допомагає узгодити бачення між замовником і розробником, а також слугує орієнтиром для подальшої роботи. У цьому підрозділі розглядаються етапи створення мокапу для сайту-візитки з інтегрованою CRM-системою, спеціально розробленого для афілійованої команди.

Для створення мокапу було обрано Figma – сучасний та потужний інструмент для проектування інтерфейсів. Основними перевагами Figma є:

– хмарне зберігання даних, що дозволяє працювати над проектом з будь-якого пристрою та у будь-який час;

– можливість спільної роботи над проектом у режимі реального часу, що полегшує комунікацію та співпрацю між членами команди;

– великий набір інструментів для створення векторної графіки, інтерактивних прототипів та дизайну інтерфейсу;

– широка бібліотека готових компонентів, що дозволяє швидко створювати та змінювати макети.

Етапи створення мокапу:

1. Визначення цілей та вимог. Перед початком роботи над мокапом було визначено основні цілі та вимоги до сайту:

– відображення інформації про афілійовану команду та її діяльність;

– забезпечення зручного та інтуїтивного інтерфейсу для користувачів.

2. Створення інформаційної архітектури. На цьому етапі було розроблено інформаційну архітектуру сайту, яка включала визначення основних розділів та підрозділів. Це забезпечило логічну структуру та зручну навігацію для користувачів.

3. Створення високо деталізованих мокапів. Після визначення усіх вимог до візуальної частини були розроблені високо деталізовані мокапи (high-fidelity mockups). Ці мокапи включали детальний дизайн з урахуванням кольорової гами, шрифтів, іконок та інших графічних елементів. Високо деталізовані мокапи показують, як буде виглядати кінцевий продукт і забезпечують точніше уявлення про користувацький досвід.

Результати створення мокапу: мокапи (рис. 2.1), створені для сайту-візитки з інтегрованою CRM-системою, забезпечили:

– чітке бачення кінцевого продукту для всіх учасників проекту;

– зручність внесення змін та коригувань на ранніх етапах розробки;

– можливість тестування користувацького інтерфейсу та навігації перед початком кодування.

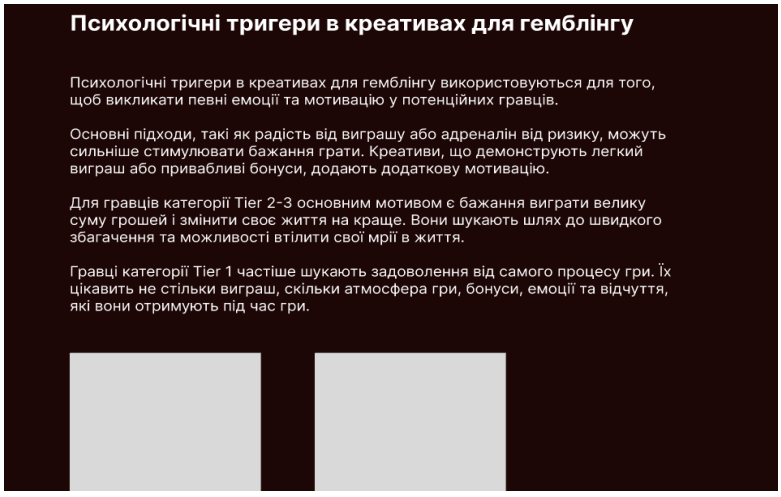
Результати створення мокапу: мокапи (рис. 2.2), створені для сайту-візитки з інтегрованою CRM-системою, забезпечили:

– чітке бачення кінцевого продукту для всіх учасників проекту;

– зручність внесення змін та коригувань на ранніх етапах розробки;

– можливість тестування користувацького інтерфейсу та навігації перед початком кодування.

2.1 – Мокап основної сторінки



2.2 – Мокап сторінки із статтею

Таким чином, створення мокапу в Figma стало важливим етапом у розробці сайту, що дозволило закласти міцний фундамент для подальшої роботи над проектом.

Створення мокапу для сайту-візитки з інтегрованою CRM-системою є ключовим етапом розробки, який дозволив визначити загальну структуру та дизайн веб-ресурсу. Використання інструменту Figma сприяло ефективній співпраці команди та забезпечило візуалізацію проекту на всіх етапах його розробки. Завдяки ретельному проектуванню та тестуванню макетів, ми змогли виявити та усунути потенційні проблеми на ранніх стадіях, що суттєво полегшило подальший процес розробки веб-сайту.

2.5 ERM діаграма

ERM діаграма (рис. 2.3) була створена для забезпечення чіткого розуміння структури бази даних і взаємодії між різними сутностями. Основні сутності, визначені в ERM діаграмі, включають:

– користувачі (users): зберігає інформацію про користувачів системи, включаючи ідентифікатор, ім'я, електронну пошту, пароль та роль;

– клієнти (clients): зберігає інформацію про клієнтів, з якими взаємодіє афілійована команда, включаючи ідентифікатор, ім'я, контактну інформацію та статус;

– транзакції (transactions): зберігає інформацію про фінансові операції, включаючи ідентифікатор транзакції, дату та пов'язану сутність клієнта;

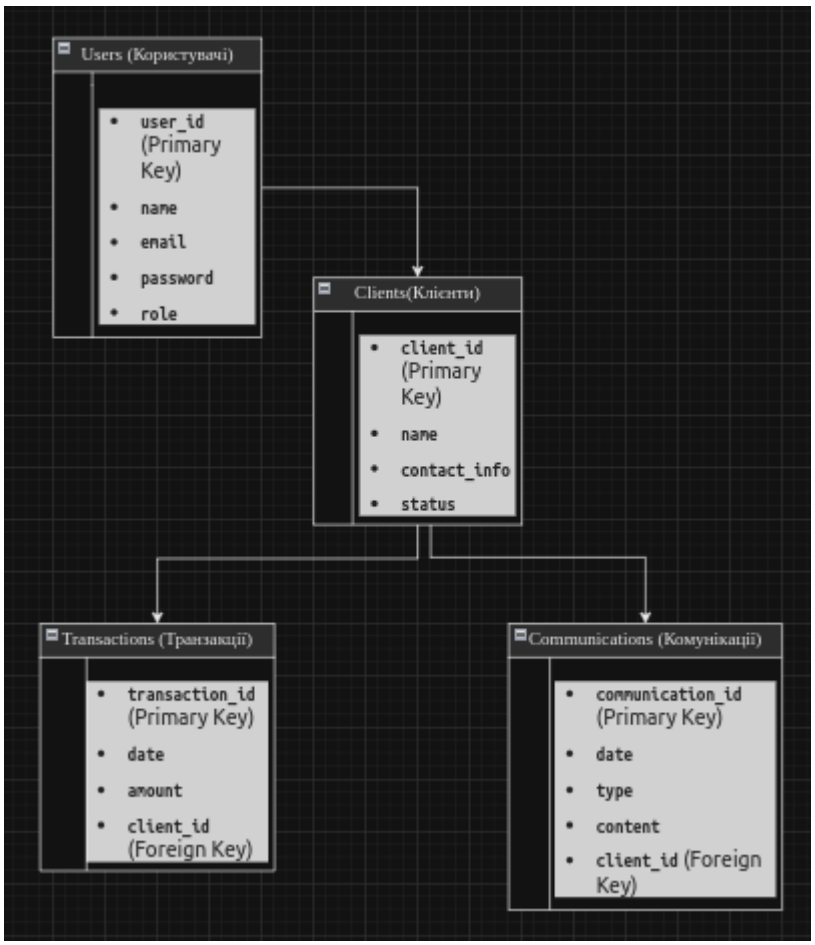
– комунікації (communications): відстежує всі комунікації з клієнтами, включаючи ідентифікатор комунікації, дату, тип та зміст.

ERM діаграма відображає зв'язки між сутностями та їх атрибути:

– users-clients: один користувач може взаємодіяти з багатьма клієнтами (зв'язок "один до багатьох").

– clients-transactions: один клієнт може мати багато транзакцій (зв'язок "один до багатьох").

– clients-communications: один клієнт може мати багато комунікацій (зв'язок "один до багатьох").



2.3 – ERM діаграма

Висновки до розділу 2

У розділі 2 було детально розглянуто ключові аспекти розробки веб-сайту, зосереджуючись на виборі середовища розробки, мові програмування та проектуванні архітектури. Вибір оптимального середовища та мови програмування є вирішальним етапом, оскільки від цього залежить ефективність та продуктивність розробки.

Архітектура веб-сайту визначає майбутню стійкість та функціональність проекту, тому її ретельне проектування є надзвичайно важливим. Розділ 2 надав важливі відомості та рекомендації для вибору оптимальних технологій та стратегій розробки, що сприятимуть успішному виконанню проекту та досягненню його цілей.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СТРУКТУРИ ТА ФУНКЦІОНАЛУ ВЕБ-САЙТУ

3.1 Розробка дизайну веб-сайту

Розробка дизайну веб-сайту є важливим етапом створення якісного та функціонального продукту. Дизайн впливає на перше враження користувачів, їхнє зручність використання веб-сайту, а також загальну ефективність взаємодії з ресурсом. У даному підрозділі розглядається процес створення дизайну веб-сайту на основі вже створених мокапів, описаних у розділі 2.

Створення концепції дизайну. На основі затверджених мокапів була розроблена загальна концепція дизайну, яка включає:

1) кольорова гама: використання кольорів, що асоціюються з брендом та створюють приємний користувацький досвід;

2) типографіка: вибір шрифтів, що забезпечують читабельність та гармонійність текстових елементів;

3) іконографіка: використання іконок для підвищення візуальної привабливості та інтуїтивності інтерфейсу;

4) візуальні елементи: розташування зображень, графіки та інших медіа-елементів для підтримки загального стилю веб-сайту.

Інструменти та технології, для реалізації дизайну веб-сайту були використані наступні інструменти та технології:

– Figma: основний інструмент для проектування та створення мокапів;

– Adobe Illustrator: для створення та обробки графічних елементів.

Основні елементи дизайну:

Головна сторінка (рис. 3.1) містить наступні елементи:

– шапка сайту (header): логотип, навігаційне меню, контактна інформація;

– головний банер: зображення або відео, що привертає увагу користувачів та містить ключове повідомлення;

– опис команди: коротка інформація про афілійовану команду, її місію та цілі;

– основні статті: блок з описом основних досягнень, та інформативними кейсами діяльності команди;

– вакансії: секція з вакансіями, яка перенаправляє на сайт із актуальною вакансією;

– футер (footer): додаткова навігація, контактна інформація, соціальні медіа.

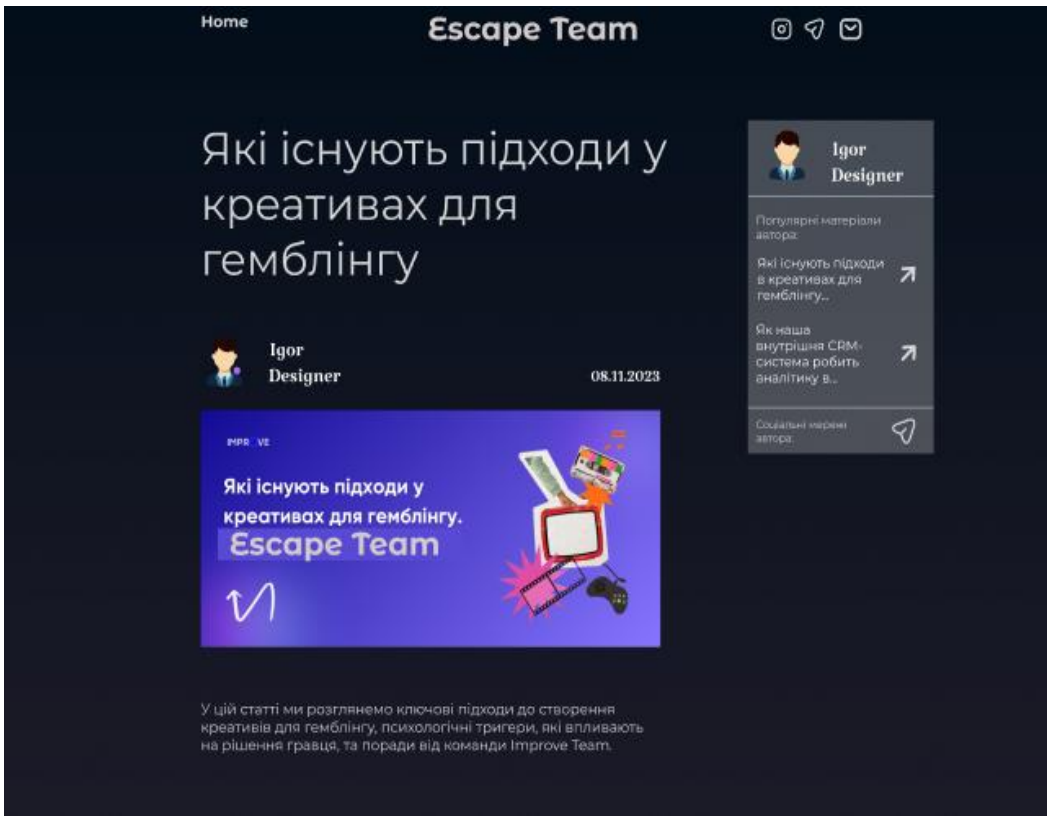
3.1 – Дизайн основної сторінки

Сторінка кейсів (рис 3.2) включає:

– контактна форма: блок із інформацією про автора даної статті;

– основна частина: відображення основної текстової та медіа інформації;

– відгуки: відгуки відвідувачів, та форма для залишення власного відгуку.



3.2 - Дизайн сторінки із статтею

Особистий кабінет (рис 3.3) містить:

– головна сторінка: основні метрики, важливі оповіщення, та інформація про користувача;

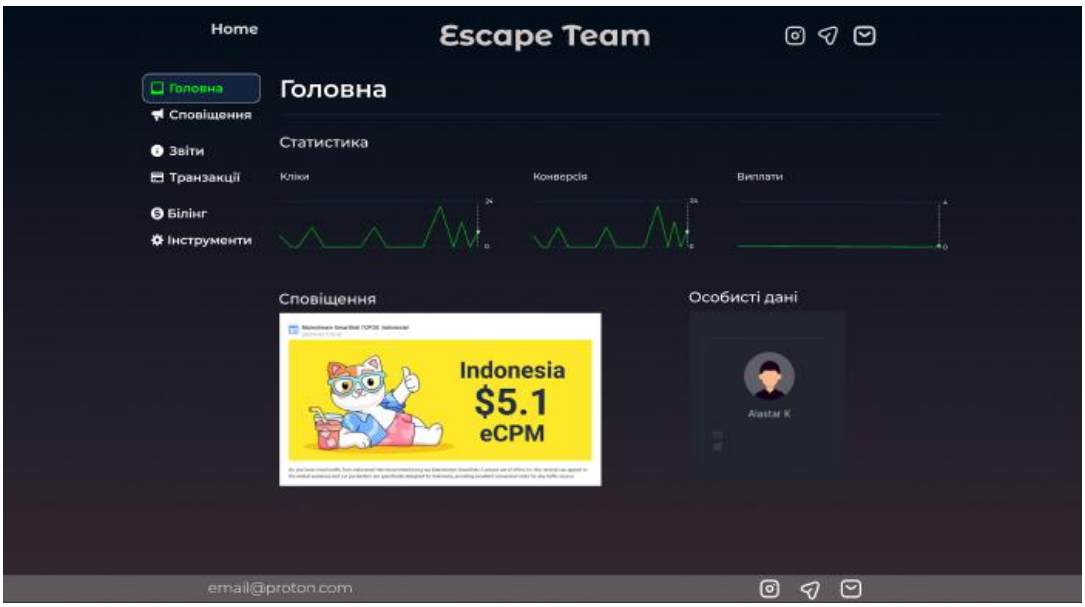
– сповіщення: відображення сповіщень різної пріоритетності;

– звіти: головний блок у особистому кабінеті, який надає можливість відслідковувати витрати та доходи працівника, а також метрики, для покращення роботи;

– транзакції: звіти про виплату, які уже виконані або в процесі;

– білінг: додавання та відстеження реквізитів для виплат;

– інструменти: можливість додавати власні метрики, для більш детального відслідковування трафіку.



3.3 – Головна сторінка особистого кабінету

Тестування дизайну. Після завершення розробки дизайну було проведено тестування з метою виявлення та усунення можливих недоліків. Тестування юзабіліті – оцінка зручності використання веб-сайту користувачами.

Процес розробки дизайну веб-сайту включав кілька етапів, починаючи від створення концепції та закінчуючи тестуванням. Використання сучасних інструментів та технологій дозволило створити привабливий та функціональний дизайн, який відповідає вимогам користувачів та замовника. Розроблений дизайн забезпечує зручну навігацію, естетичний вигляд та ефективну презентацію інформації, що сприяє досягненню цілей веб-сайту.

Наступним етапом є верстка веб-сайту, яка включає реалізацію розробленого дизайну у вигляді HTML та CSS коду, а також інтеграцію з системою керування контентом (CMS) та CRM-системою.

3.2 Верстка веб-сайту

У цьому підрозділі розглянемо процес верстки веб-сайту, який включає створення HTML-структури, застосування CSS-стилів та адаптивного дизайну. Верстка є важливою частиною розробки веб-сайту, оскільки від її якості

залежить зручність користування сайтом, його швидкість завантаження та коректне відображення на різних пристроях.

HTML (hypertext markup language) – структура є основою веб-сторінок. На цьому етапі ми створюємо базову структуру сайту, визначаючи основні секції та елементи:

– заголовок (header): включає логотип, навігаційне меню та інші елементи, що відображаються у верхній частині сторінки;

– основний контент (main): розділи з контентом, включаючи тексти, зображення, відео та інші мультимедійні елементи;

– футер (footer): нижня частина сторінки з контактною інформацією, посиланнями на політику конфіденційності, та іншими важливими елементами.

CSS (Cascading Style Sheets) використовується для стилізації HTML-елементів. Основні аспекти стилізації включають:

– типографіка: налаштування шрифтів, розмірів тексту, відступів та вирівнювання;

– колірна схема: вибір кольорів для тексту, фону, кнопок та інших елементів;

– макет (layout): визначення розташування елементів на сторінці за допомогою властивостей flexbox або grid.

– адаптивність: забезпечення коректного відображення сайту на різних пристроях за допомогою медіа-запитів.

Інструменти для верстки:

– HTML редактори: Visual Studio Code, Sublime Text, Atom;

– CSS фреймворки: Bootstrap, Foundation, Tailwind CSS, які спрощують процес верстки завдяки готовим стилям та компонентам;

– препроцесори CSS: SASS, LESS для написання більш організованого та підтримуваного CSS-коду.

Адаптивний дизайн (responsive design) дозволяє сайту коректно відображатися на різних пристроях, від настільних комп'ютерів до мобільних телефонів. Основні принципи включають:

– гнучкі сітки (fluid grids): використання відсоткових значень для ширини елементів.

– гнучкі зображення (flexible images): налаштування зображень так, щоб вони змінювали розмір разом з екраном;

– медіа-запити (media queries): CSS-правила, що застосовуються залежно від характеристик пристрою, наприклад, ширини екрану.

Нижче наведено приклади коду верстки основної сторінки (рис. 3.4).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Affiliate Team</title>
    <style>
        .div {
            display: flex;
            flex-direction: column;
            align-items: center;
            padding: 0 20px;
        }
        .div-2 {
            background-color: #5b4242;
            align-self: stretch;
            display: flex;
            width: 100%;
            flex-direction: column;
            justify-content: center;
        }
        @media (max-width: 991px) {
            .div-2 {
                max-width: 100%;
            }
        }
```

3.4 – Приклад коду верстки основної сторінки

3.3 Програмування веб-сайту

У цьому підрозділі розглянуто процес програмування веб-сайту з використанням мови програмування Python та фреймворка Flask. Flask є мікрофреймворком для створення веб-додатків, що забезпечує простоту та гнучкість у розробці, що робить його ідеальним для малих та середніх проектів.

Для програмування веб-сайту було обрано наступні технології:

– мова програмування: python;

– фреймворк: flask для back-end розробки;

– база даних: SQLite для зберігання даних;

– front-end: HTML, CSS, JavaScript для створення інтерфейсу користувача

Основні елементи включають форму для реєстрації та входу користувачів, а також інтерфейс особистого кабінету. HTML приклад форми реєстрації (рис. 3.4).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registration</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
  <form action="/register" method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>
    <button type="submit">Register</button>
  </form>
</body>
</html>
```

3.5 – HTML форми реєстрації

Back-end програмування: flask використовується для створення серверної частини додатку. Основні функції включають обробку HTTP-запитів, взаємодію з базою даних та аутентифікацію користувачів. Приклад коду (рис. 3.6) на Python з використанням Flask:

```
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), nullable=False, unique=True)
    password = db.Column(db.String(150), nullable=False)

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        hashed_password = generate_password_hash(password, method='sha256')
        new_user = User(username=username, password=hashed_password)
        db.session.add(new_user)
        db.session.commit()
        return redirect(url_for('login'))
    return render_template('register.html')
```

3.6 – Фрагмент використання Flask

Взаємодія між front-end і back-end здійснюється за допомогою HTTP-запитів. Flask обробляє ці запити та повертає відповідні відповіді, що дозволяє створювати динамічні та інтерактивні веб-сторінки. Тестування є важливим етапом у розробці веб-сайту, оскільки забезпечує стабільність та коректну роботу додатку. Для цього використовуються різні методи тестування. Одним із методів є юніт-тест на Python з використанням unittest.

3.4 Реалізація баз даних для особистого кабінету

У цьому підрозділі розглянемо процес розробки та реалізації бази даних для особистого кабінету користувачів. Особистий кабінет дозволяє користувачам зберігати та керувати своїми персональними даними, переглядати історію активності та взаємодіяти з іншими функціями веб-сайту. Основною

метою є створення ефективної та безпечної бази даних, яка підтримує всі необхідні функції особистого кабінету.

Вибір бази даних: для реалізації бази даних було обрано SQLite, оскільки це легка, вбудована система управління базами даних, яка ідеально підходить для невеликих та середніх проектів. SQLite не вимагає налаштування сервера та забезпечує простий та надійний спосіб зберігання даних.

Структура бази даних включає декілька таблиць, які зберігають різні типи даних, необхідні для функціонування особистого кабінету. Основними таблицями є:

– users: зберігає інформацію про користувачів, включаючи їх ідентифікатор, ім'я користувача, хешований пароль та інші персональні дані.

– user activity: зберігає історію активності користувачів, включаючи дату та час подій, тип події та додаткову інформацію.

SQL – запити (рис. 3.7) для створення таблиць:

```
CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT NOT NULL UNIQUE,
    password TEXT NOT NULL,
    email TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE user_activity (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    activity_type TEXT,
    activity_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users (id)
);
```

3.7 – SQL – запити для створення таблиць

Інтеграція бази даних з Flask: flask дозволяє легко інтегрувати базу даних за допомогою розширення Flask-SQLAlchemy, яке забезпечує ORM (Object-

Relational Mapping) для роботи з базою даних. Код для налаштування SQLAlchemy (рис. 3.8):

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), unique=True, nullable=False)
    password = db.Column(db.String(150), nullable=False)
    email = db.Column(db.String(150), nullable=False)
    created_at = db.Column(db.DateTime, default=db.func.current_timestamp())

class UserActivity(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    activity_type = db.Column(db.String(150), nullable=False)
    activity_date = db.Column(db.DateTime, default=db.func.current_timestamp())
```

3.8 – код SQLAlchemy

Реєстрація та авторизація користувачів: реєстрація та авторизація користувачів є критичними аспектами будь-якого веб-додатку, що потребує персоналізованого доступу. Вони забезпечують безпечний доступ користувачів до особистого кабінету та інших захищених ресурсів сайту.

Процес реєстрація користувачів:

1) користувач заповнює реєстраційну форму на веб-сайті, вказуючи ім'я користувача, пароль та інші необхідні дані;

2) сервер отримує ці дані та хешує пароль для безпечного зберігання;

3) сервер зберігає новий запис у таблиці users.

Елементи коду для реєстрації користувача (рис. 3.9):

```
from werkzeug.security import generate_password_hash
from flask import request, redirect, url_for, render_template

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        hashed_password = generate_password_hash(password, method='sha256')
        new_user = User(username=username, password=hashed_password, email=email)
        db.session.add(new_user)
        db.session.commit()
        return redirect(url_for('login'))
    return render_template('register.html')
```

3.9 – Код реєстрації користувачів

Процес авторизації користувачів:

1) користувач вводить своє ім'я користувача та пароль у форму входу на веб-сайті;

2) сервер отримує ці дані та шукає користувача у базі даних;

3) сервер перевіряє, чи збігається введений пароль з хешованим паролем у базі даних;

4) якщо перевірка пройшла успішно, створюється сесія користувача, яка дозволяє йому отримати доступ до захищених ресурсів.

Відслідковування активності користувачів – відслідковування активності користувачів дозволяє збирати дані про дії користувачів на сайті, що може бути використано для аналізу поведінки користувачів, покращення користувацького досвіду та виявлення потенційних проблем.

Процес відслідковування активності користувачів:

1) коли користувач виконує певну дію (наприклад, входить у систему, редагує профіль або додає новий запис), ця дія реєструється в системі;

2) створюється запис у таблиці user_activity, який містить ідентифікатор користувача, тип дії та дату і час виконання дії;

3) ці дані можуть бути використані для аналізу активності користувачів. Елементи коду для збереження активності користувача (рис. 3.10):

```
@app.route('/some_action', methods=['POST'])
def some_action():
    user_id = get_current_user_id()  # Функція для отримання поточного ідентифікатора користувача
    new_activity = UserActivity(user_id=user_id, activity_type='some_action')
    db.session.add(new_activity)
    db.session.commit()
    return 'Action performed!'
```

3.10 – Код активності користувачів

Деталізація процесу:

1) отримання поточного користувача: необхідно забезпечити механізм отримання ідентифікатора поточного користувача, який виконав дію. Це може бути реалізовано за допомогою сесій або токенів аутентифікації;

2) запис активності: коли користувач виконує дію, створюється новий запис у таблиці user_activity з інформацією про тип дії та час її виконання;

3) аналіз даних: збережені дані можуть бути використані для різних цілей, таких як відстеження частоти використання певних функцій, виявлення потенційних проблем у роботі системи або аналіз поведінки користувачів.

У цьому підрозділі було розглянуто процес розробки та реалізації бази даних для особистого кабінету, зокрема реєстрацію та авторизацію користувачів, а також відслідковування їх активності. Реєстрація та авторизація користувачів забезпечують безпечний доступ до особистого кабінету, тоді як відслідковування активності дозволяє аналізувати поведінку користувачів та покращувати функціональність веб-сайту. Створення ефективної та безпечної бази даних є ключовим етапом у забезпеченні функціональності особистого кабінету та загальної роботи веб-сайту.

3.5. Інтеграція API для обробки даних та взаємодії з іншими сервісами

Основна мета інтеграції зовнішніх API полягає у тому, щоб забезпечити веб-сайту доступ до даних та функцій, які надаються іншими сервісами. Це може бути критично важливою задачею для реалізації різноманітних функціональностей, що покращують користувацький досвід та роблять наш веб-сайт більш функціональним та корисним. Наприклад, можливість автоматичного оновлення інформації без потреби ручного втручання може значно зекономити час та зусилля для наших користувачів.

Додатково, інтеграція зовнішніх API дозволяє розширити можливості взаємодії з користувачами через соціальні мережі. Наприклад, ми можемо використовувати API соціальних мереж для авторизації користувачів на веб-сайті, що робить процес входу більш швидким та зручним для користувачів. Крім того, можливість імпорту профілів користувачів з соціальних мереж може полегшити їхню реєстрацію на нашому веб-сайті та підвищити залученість аудиторії.

Ще однією важливою функцією інтеграції зовнішніх API є обробка платежів. Використання API платіжних систем дозволяє нам забезпечити безпечні та надійні транзакції для наших користувачів, забезпечуючи при цьому зручний та ефективний спосіб оплати за наші товари чи послуги. Інтеграція зовнішніх API відкриває перед нами безліч нових можливостей для покращення функціональності та збільшення користувацького досвіду нашого веб-сайту.

3.6. Тестування веб-сайту

Тестування веб-сайту – це ключовий етап в розробці програмного забезпечення, спрямований на перевірку його функціональності, надійності та безпеки. Цей процес дозволяє виявити та усунути можливі помилки та недоліки, забезпечуючи високу якість продукту та задоволення користувачів. Також тестування веб-сайту відіграє важливу роль у забезпеченні правильності його

роботи та відповідності вимогам користувачів. Цей процес допомагає переконатися, що веб-сайт працює коректно на різних платформах та веб-браузерах, забезпечуючи при цьому швидку та зручну роботу для кожного користувача. Види тестування:

– функціональне тестування: перевірка правильності реалізації функцій веб-сайту та їх відповідність вимогам;

– навантажувальне тестування: визначення меж продуктивності веб-сайту та його реакції на велике навантаження користувачів;

– тестування безпеки: виявлення та усунення потенційних вразливостей та загроз безпеці веб-сайту;

– сумісність з браузерами та пристроями: перевірка відображення та функціональності веб-сайту на різних платформах та пристроях;

– тестування адаптивності: перевірка адаптивності веб-сайту на різних розмірах екранів та його відповідність різним пристроям.

Процес тестування веб-сайту: тестування веб-сайту – це складний процес, який включає в себе кілька етапів та методик, спрямованих на перевірку різних аспектів його функціонування. Розглянемо кожен етап процесу більш детально:

1) планування тестування, перший крок у процесі тестування – це планування. На цьому етапі визначається стратегія тестування, обсяг та види тестів, які будуть проводитися. Також складається план тестування, який включає в себе опис завдань, відповідальностей, ресурсів та графік виконання;

2) розробка тест-кейсів, тест-кейси – це документи, що описують кроки для виконання певного тесту, його очікуваний результат та критерії прийняття. Розробка детальних тест-кейсів дозволяє систематизувати тестувальні процеси та забезпечити повноту та ефективність тестування;

3) виконання тестів – на цьому етапі проводяться тести згідно з розробленими тест-кейсами. Тестувальний персонал виконує певні дії на веб-сайті та перевіряє його реакцію, результати та відповідність очікуваним вимогам.

4) виявлення та реєстрація дефектів – під час виконання тестів можуть бути виявлені помилки або недоліки в роботі веб-сайту. Кожна знайдена

проблема реєструється у спеціальній системі для подальшого аналізу та виправлення розробниками;

5) виправлення помилок – розробники виправляють виявлені під час тестування помилки та недоліки. Цей процес може включати в себе аналіз причин помилок, розробку та впровадження виправлень;

6) повторне тестування, після виправлення помилок проводиться повторне тестування для перевірки їхнього виправлення та впевненості у правильності функціонування веб-сайту.

7) останній етап – це прийомка веб-сайту. На цьому етапі здійснюється остаточна перевірка його готовності до випуску.

Висновки до розділу 3

У Розділі 3 виділені ключові аспекти програмної реалізації веб-сайту, які були детально розглянуті. Починаючи з розробки дизайну та верстки, переходячи до програмування та інтеграції зовнішніх сервісів, я доклав зусилля для забезпечення високої якості та ефективності нашого продукту.

Процес розробки веб-сайту виявився важливим і складним, вимагаючи ретельного планування та систематичного підходу до кожного етапу. Тестування веб-сайту виявилося незамінним для виявлення потенційних проблем та забезпечення його надійності та безпеки перед випуском в експлуатацію.

Інтеграція зовнішніх API додала веб-сайту новий функціонал та можливості, розширивши його можливості та забезпечивши зручну взаємодію з користувачами.

У цілому, процес програмної реалізації структури та функціоналу веб-сайту був захоплюючим і водночас викликав багато викликів. Проте завдяки цілеспрямованій роботі і великому зусиллю вдалось створити продукт, який відповідає вимогам сучасного інтернет-середовища і забезпечує задоволення потреб користувачів.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було проведено ґрунтовний аналіз існуючих аналогів та конкурентів, що дозволило ідентифікувати найефективніші підходи та стратегії для реалізації веб-сайту. Вибір оптимальних технологій став ключовим етапом, який сприяв створенню сучасного та функціонального продукту. У ході роботи були використані новітні методики та підходи до веб-розробки, що дозволило забезпечити високу якість кінцевого продукту, який відповідає всім встановленим вимогам та задовольняє потреби цільової аудиторії.

Особливу увагу було приділено розробці інтерфейсу користувача та його зручності, що включало тестування на різних етапах розробки для забезпечення максимальної користувацької задоволеності. Крім того, було впроваджено кілька новітніх технологій, таких як адаптивний дизайн, інтерактивні елементи та оптимізація продуктивності, що суттєво підвищило загальну привабливість та функціональність веб-сайту.

Цей проект став важливим етапом у для професійному розвитку. Завдяки цьому досвіду було отримано цінні знання у сфері веб-розробки, освоїв нові технології та інструменти, а також поглиблено навички у розробці комплексних веб-рішень. Процес роботи над проектом допомогло краще зрозуміти важливість ретельного планування, аналізу вимог та тестування, що є невід'ємною частиною успішної розробки програмного забезпечення. Отримані знання та навички стануть міцною основою для подальшого професійного зростання та розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інструкції та приклади для веб-розробки. URL: https://www.w3schools.com/ (Дата звертання: 29.03.2024).

2. Medium Article on Competitive Analysis. Ресурси для аналізу конкурентів у веб-розробці. URL: https://medium.com/@olex_world/competitor-analysis-c46505495606 (Дата звертання 30.03.2024).

3. Lucidchart. Інструменти та приклади для створення ERM діаграм. URL: https://www.lucidchart.com/pages/i18n/erd. (Дата звертання 02.04.2024).

4. Freeman, E., & Robson, E. (2014). Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages. O'Reilly Media. URL: https://www.amazon.com/Head-First-HTML-CSS-Standards-Based/dp/0596159900. (Дата звертання 05.04.2024).

5. Розробка веб застосунку з використанням фреймворку Flask та графічної бібліотеки Folium. URL: https://ekmair.ukma.edu.ua/server/api/core/bitstreams/8a2cc4c0-01e6-4052-856e-0376c0083966/content. (Дата звертання 06.04.2024).

6. Figma Resources. Інструменти та навчальні матеріали для створення мокапів. URL: https://www.figma.com/community/tag/mockup. (Дата звертання 09.04.2024).

7. JavaScript developer.mozilla.org: веб-сайти. URL: https://developer.mozilla.org/ru/docs/Web/JavaScript. (Дата звертання 10.04.2024).

8. Bootstrap Documentation. Інструменти та шаблони для розробки дизайну URL: https://bootstrap21.org/uk/docs/5.0/getting-started/introduction/ (дата звернення: 15.04.2024).

9. Nielsen Norman Group. Ресурси про юзабіліті та UX-дизайн. URL: https://www.nngroup.com/ (дата звернення: 19.04.2024).

10. Sweigart, Al. Automate the Boring Stuff with Python: Practical Programming for Total Beginners. URL: https://automatetheboringstuff.com/ (Дата

звернення: 25.04.2024).

11. Duckett, Jon. JavaScript and JQuery: Interactive Front-End Web Development. URL: https://eloquentjavascript.net/06_object.html (Дата звернення: 04.05.2024).

12. MySQL 5.7 Reference Manual. Офіційна документація по MySQL. URL: https://dev.mysql.com/doc/refman/5.7/en/. (Дата звернення: 10.05.2024).

13. Mozilla Developer Network (MDN): Web APIs. URL: https://developer.mozilla.org/en-US/docs/Web/API. (Дата звернення: 13.05.2024).

14. Keith, Jeremy. Resilient Web Design. URL: https://resilientwebdesign.com/. (Дата звернення: 15.05.2024).

15. Todd Zaki Warfel - "Prototyping: A Practitioner's Guide" 2009. 368 с. URL: https://www.amazon.com/Prototyping-Practitioners-Guide-Todd-Warfel/dp/0596514921. (Дата звернення: 18.05.2024).

16. Jennifer Niederst Robbins - "Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics" 2018. 808 с. URL:https://www.amazon.com/Learning-Web-Design-Beginners-JavaScript/dp/1449319270. (Дата звернення: 20.05.2024).

17. Douglas Crockford - "JavaScript: The Good Parts" 2008. 176 с. URL:https://www.amazon.com/JavaScript-Good-Parts-Douglas-Crockford/dp/0596517742. (Дата звернення: 23.05.2024).

18. David Flanagan - "JavaScript: The Definitive Guide: Activate Your Web Pages" 2020. 706 с. URL:https://www.amazon.com/JavaScript-Definitive-Guide-Activate-Guides/dp/0596805527. (Дата звернення: 15.05.2024).

19. Jon Duckett - "JavaScript and JQuery: Interactive Front-End Web Development" 2014. 640 с. URL: https://www.amazon.com/JavaScript-JQuery-Interactive-Front-End-Development/dp/1118531647. (Дата звернення: 15.05.2024).

20. Jon Duckett - "HTML and CSS: Design and Build Websites" 2011. 490 с. URL: https://www.amazon.com/HTML-CSS-Design-Build-Websites/dp/1118008189. (Дата звернення: 15.05.2024).

метадані

Заголовок
Розробка сайту-візитки з вбудованою CRM-системою

Автор
Ступар О. Науковий керівник / Експерт

підрозділ
King Danylo University

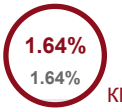
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

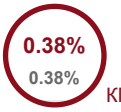
Заміна букв	ß	1
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	11

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КП 2

25
Довжина фрази для коефіцієнта подібності 2

8179
Кількість слів

63907
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	Розробка клієнтської частини веб-сайту для бюро перекладів «Парі-ІФ» [redacted] 6/12/2023 King Danylo University (King Danylo University)	31	0.38 %
2	Петрова.docx [redacted] 6/19/2023 Odessa National Polytechnic University (ІКС, кафедра інформац. технологій)	14	0.17 %
3	http://repository.ukd.edu.ua/bitstream/handle/123456789/397/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0_%D0%A2%D0%B0%D1%82%D0%B0%D1%80%D1%87%D1%83%D0%BA.pdf?sequence=1	14	0.17 %