Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка автоматизованої системи оцінювання Web-ресурсів»

Студента 4 курсу, 9 групи,

спеціальності
122 «Комп'ютерні науки»

Аршулік Андрій Вікторович

_____
підпис студента

Науковий керівник
кандидат технічних наук, доцент

_____
підпис керівника

Козлов Валерій Володимирович

Гарант освітньої програми
кандидат технічних наук, доцент

_____
підпис керівника

Демідов Павло Георгійович

Київ 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О. І.
« » грудня 2020р.

Завдання
на випускну кваліфікаційну роботу (проект) студенту

Аршуліку Андрію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
«Розробка автоматизованої системи оцінювання Web-ресурсів»
Затверджена наказом ректора від *«15» грудня 2020 р. № 3780*
2. Строк здачі студентом закінченої роботи *28 травня 2021 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: обґрунтування та розробка автоматизованої системи оцінювання Web-ресурсів, з урахуванням сучасних тенденцій побудови
Об'єкт дослідження: процес проектування автоматизованої системи оцінювання Web-ресурсів.
Предмет дослідження: засоби створення автоматизованої системи оцінювання Web-ресурсів
4. Перелік графічного матеріалу_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Козлов В.В.	15.12.2020 р.	15.12.2020 р.
2	Козлов В.В.	15.12.2020 р.	15.12.2020 р.
3	Козлов В.В.	15.12.2020 р.	15.12.2020 р.

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ВСТУП
РОЗДІЛ 1. ЗАГАЛЬНА ХАРАКТЕРИСТИКА ПІДХОДІВ ДО АНАЛІЗУ WEB-РЕСУРСІВ
1.1. Види і параметри аналізу Web-ресурсів
1.2. Аналіз існуючих аналізаторів Web-ресурсів
1.3. Аналіз поведінки аналізатора Web-ресурсів
Висновки до розділу
РОЗДІЛ 2. АНАЛІЗ СУЧАСНИХ МЕТОДІВ ТА ТЕХНОЛОГІЙ РОЗРОБКИ АНАЛІЗАТОРІВ WEB-РЕСУРСІВ
2.1. Підходи до проектування архітектури Web-орієнтованого застосунка
2.2. Огляд засобів розробки клієнтської частини аналізатора Web-ресурсів
2.3. Обґрунтування вибору програмних та технічних засобів розробки аналізатора Web-ресурсів
Висновки до розділу
РОЗДІЛ 3. РОЗРОБКА АНАЛІЗАТОРА WEB-РЕСУРСІВ
3.1. Розробка архітектури аналізатора Web-ресурсів
3.2.Програмна реалізація проекту
3.3. Тестування та дослідна експлуатація
3.4. Інструкція користувача
Висновки до розділу
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ пор.	Назва етапів випускного кваліфікаційного проекту	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускного кваліфікаційного проекту*	*01.10.2020*	*01.10.2020*
2	*Розробка та затвердження завдання на випускний кваліфікаційний проект*	*15.12.2020*	*15.12.2020*
3	*Вступ*	*03.02.2021*	*03.02.2021*
4	*Розділ 1. Загальна характеристика підходів до аналізу WEB-ресурсів*	*26.02.2021*	*26.02.2021*
5	*Розділ 2. Аналіз сучасних методів та технологій розробки аналізаторів WEB-ресурсів*	*06.04.2021*	*06.04.2021*
6	*Розділ 3. Розробка аналізатора WEB-ресурсів*	*12.05.2021*	*12.05.2021*
7	*Висновки*	*15.05.2021*	*15.05.2021*
8	*Здача випускного кваліфікаційного проекту на кафедру науковому керівнику*	*20.05.2021*	*20.05.2021*
9	*Попередній захист випускного кваліфікаційного проекту*	*26.05.2021*	*02.06.2021*
10	*Виправлення зауважень, зовнішнє рецензування випускного кваліфікаційного проекту*	*27.05.2021*	*05.06.2021*
12	*Представлення готового зшитого випускного кваліфікаційного проекту на кафедру*	*28.05.2021*	*08.06.2021*
13	*Публічний захист випускного кваліфікаційного проекту*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2020 р.

9. Керівник випускної кваліфікаційної роботи (проекту)
Козлов В. В.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П. Г.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник
Аршулік А. В.
(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи (проекту)

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Керівник випускної кваліфікаційної роботи (проекту)
_____31.05.2021 р.
(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу (проект)

Випускна кваліфікаційна робота (проект) студента _____
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри _____Пурський О.І.
(підпис, прізвище, ініціали)
«_____»_____2021 р.

Анотація

У випускній кваліфікаційній роботі проведено дослідження сучасних технологій які дозволяють аналізувати WEB-ресурси, розглянуто засоби супроводження WEB-ресурсів, створено автоматизовану Web-систему оцінювання показників WEB-сайту.

Об’єктом дослідження є процес проектування системи оцінювання Web-ресурсів**.**

Предметом дослідження є створення програмної системи для проведення аналіз унікальності веб-сайтів.

Одержані результати полягають в розробці програмної системи для аналізу унікальності контенту веб-сайтів.

Ключові слова: веб-додаток, програмна система, аналіз унікальності контенту веб-сайтів, SEO, веб-сайт.

Annotation

In the final qualification work the research of modern technologies which allow to analyze WEB-resources is carried out, means of support of WEB-resources are considered, the automated Web-system of estimation of indicators of the WEB-site is created.

The object of research is the process of designing a system for evaluating Web-resources.

The subject of the study is to create a software system for analyzing the uniqueness of websites.

The results obtained are the development of a software system for analyzing the uniqueness of website content.

Keywords: we-application, software system, analysis of website content uniqueness, SEO, website.

ЗМІСТ

ВСТУП

Наразі неможливо уявити своє життя без мережі Інтернет, та веб-сайтів, які вона вміщує в собі. Веб-сайт – це сукупність веб сторінок та залежного вмісту, доступних у Всесвітній мережі, які об'єднані як за змістом, так і за навігацією під єдиним доменним ім'ям. Сайтом також називають вузол мережі Інтернет, комп'ютер, за яким закріплена унікальна IP-адреса, і взагалі будь-який об'єкт в Інтернеті, за яким закріплена адреса, що ідентифікує його в мережі (FTP-site, WWW-site тощо)[1].

Аналітика веб-сайтів є відносно молодою галуззю, яка хоча й існувала з моменту появи Інтернету, але активно розвиватися почала на етапі, коли широка доступність швидкісного Інтернету викликала активний інтерес до створення сайтів з боку підприємців і звичайних користувачів. Свою роль в цьому зіграли і пошукові системи, розвиток алгоритмів яких викликало посилення вимог до якості, коли хороший трафік з видачі, в умовах конкуренції між безліччю ресурсів, отримують тільки дійсно якісні веб-ресурси.

Створення усіх сайтів наявних в інтернеті має під собою комерційне підґрунтя. А для того щоб на них заробляти, потрібно грамотно налаштувати їх та провести велику роботу по розкрутці та оптимізації. В першу чергу, потрібно ознайомитись з таким поняттям, як SEO(оптимізація під пошукові сервери). Суть цієї речі полягає у процесі коригування HTML-коду, текстового наповнення (контенту), структури сайту, контроль зовнішніх чинників для відповідності вимогам алгоритму пошукових систем, з метою підняття позиції сайту в результатах пошуку в цих системах за певними запитами користувачів. Чим вище позиція сайту в результатах пошуку, тим більша ймовірність, що відвідувач перейде на нього з пошукових систем, оскільки люди зазвичай йдуть за першими посиланнями.

Мета та завдання дослідження. Метою даного дослідження є обґрунтування та розробка системи оцінювання Web-ресурсів, з урахуванням сучасних тенденцій побудови. Для досягнення поставленої мети необхідно

було вирішити наступні завдання:

- дослідити види і параметри аналізу Web –ресурсів;
- проаналізувати існуючі аналізатори Web -ресурсів та їх поведінку;
- розробити математичну модель системи для аналізу Web –сайтів;
- розробити програмну системи для аналізу унікальності контенту Web – сайтів;

Об'єкт дослідження: процес проектування системи оцінювання Web-ресурсів**.**

Предмет дослідження: створення програмної системи для проведення аналіз унікальності веб-сайтів.

Практичне значення цієї програмної системи полягає в тому, що вона може вирішити проблему аналізу унікальності вмісту веб-сайтів з точки зору оптимізатора SEO.

У процесі розробки програмної системи будуть використані такі інструменти та технології:

1. Visual Studio Code – засіб для створення, редагування та зневадження сучасних вебзастосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X.
2. StarUML – це проект з відкритим кодом для розробки швидких, гнучких, функціональних платформ UML / MDA для систем Windows.
3. Ranorex Studio 5.4.2 – середовище розробки, а також набір інструментів і бібліотек для написання тестів програмного забезпечення.
4. ASP.NET – технологія створення веб-застосунків і веб-сервісів від компанії Майкрософт.

РОЗДІЛ 1.
ЗАГАЛЬНА ХАРАКТЕРИСТИКА ПІДХОДІВ ДО АНАЛІЗУ WEB-РЕСУРСІВ

1.1. Види і параметри аналізу Web-ресурсів

Інтернет-комунікації представляють собою складне багатокомпонентне явище, яке є особливо актуальним для сучасної документально-комунікаційної системи суспільства, що охоплює такі її інституції, як бібліотеки, архіви, інформаційні та реферативні служби, книговидавництва тощо. Інтернет-комунікаційна система містить такі головні складові: ресурсну, сервісну й технологічну. Ресурсний (змістовний) компонент, який є джерелом розповсюдження знань та інформації, та сервісний компонент, що уособлює новітні засоби та канали функціонування інформації в суспільстві, мають особливе значення для сучасних інформаційних центрів, які спрямовані на вдосконалення виробництва і розповсюдження конкурентоспроможних інформаційних продуктів та послуг власної генерації у міжнародному інформаційному просторі[2-3].

Веб-сайт в Інтернеті, з інформаційно-змістовної точки зору, є одним з дієвих комунікаційних засобів, що спрямований на презентування та ефективне позиціювання будь-якої установи в інформаційно-комунікаційному просторі через надання повної достовірної інформації про себе; розповсюдження та надання вільного доступу до власних інформаційних ресурсів та послуг; забезпечення зворотного зв’язку з користувачами тощо. Уточнимо, що сукупність інформації, вміщуваної на веб-сайті, у міжнародному співтоваристві прийнято називати контентом або інформаційним наповненням. У зарубіжних країнах, зокрема США, контенту веб-сайтів приділяється особлива увага розробників та осіб, котрі займаються подальшою їхньою підтримкою, оскільки веб-сайт сприяє рекламуванню та просуванню самої установи, діючи як механізм некомерційної реклами. Наявність грамотно побудованого веб-сайту підвищує рівень оцінки установи

користувачем, який бажає ґрунтовно ознайомитися з її діяльністю та інформаційними ресурсами[4].

Порівняльний аналіз веб-сайтів здійснювався за двома головними критеріями: використання інтернет-сервісів (або послуг) та наявність позначок про останнє оновлення. До першого критерію відносяться інтернет-сервіси. Служби (або сервіси) Інтернету – це види послуг і водночас прикладні програми, розроблені для отримання доступу до інформації певного типу або обміну даними. Кожен сервіс характеризується властивостями, частина яких об'єднує його з однією групою сервісів, а інша частина – з другою. Згідно із класифікацією сервісів Інтернету прийнято їхній розподіл за великою кількістю ознак, серед яких: сервіси інтерактивні, прямі та відкладеного читання. Серед цих служб можна виділити ті, що призначені для комунікації, тобто для спілкування, передавання інформації (e-mail, ICQ), а також служби, завдання яких – зберігання інформації та забезпечення доступу до неї користувачів. В історії Інтернету існували різні види сервісів, одні з яких вже не використовуються, інші поступово втрачають свою популярність, у той час як треті переживають свій розквіт. Перерахуємо ті з сервісів, які й нині не втратили своєї актуальності: World Wide Web (www) – служба пошуку та перегляду гіпертекстових документів , що включають у себе графіку, звук і відео; e-mail – електронна пошта – служба передавання електронних повідомлень; Usenet, News (телеконференції, групи новин) – різновид мережної газети або дошки оголошень; ICQ – служба для спілкування в реальному часі за допомогою клавіатури та інші. Особливу групу сервісів Інтернету представляють сучасні програмні розроблення, що використовують мережу як середовище передавання інформації. Це, насамперед, програмні пакети для проведення відео та аудіо конференцій. Популярними стають соціальні сервіси, які дають користувачам можливість здійснювати спілкування в межах відповідної групи, серед них такі системи: створення веб-журналів, збереження мультимедійних веб-ресурсів та особливо – блогів (віртуальних щоденників), веб-блогів. Серед усього переліку сервісів можна

обрати придатні для просування результатів інформаційної діяльності ЦНІІ та для забезпечення зворотного зв'язку з користувачем-клієнтом, окрім умовно "типових": www та e-mail[4].

Другий критерій – останнє оновлення – є також важливим для оцінки веб-сайту, він свідчить про його "життєдіяльність", актуалізацію та функціонування в інформаційному просторі[4]. За міжнародними вимогами, веб-сайт потрібно оновлювати як мінімум один раз на тиждень, а нові повідомлення (новини) – щоденно.

Хоча в більшості випадків дослідження є всебічнім, можна виділити і кілька окремих видів веб-аналітики: аналіз відвідуваності, її показників в різних періодах та тенденцій, які проявляються; аналіз поведінкових факторів, тобто того, як відвідувачі поводяться на сторінках сайту; порівняння з конкурентами і загальними показниками по досліджуваної області в цілому.

Звичайно, це далеко не всі області, в яких може застосовуватися веб-аналітика, але саме перераховані вище є ключовими для підприємців, які ведуть бізнес в Інтернеті.

Також завданням веб аналітики є: визначення проблемних місць на сайті і виправлення помилок; моніторинг доступності та стабільності роботи ресурсу; аналіз і ведення статистики відвідуваності, визначення основних тенденцій; дослідження поведінки відвідувачів і факторів, які на неї впливають; аналіз ефективності проведених рекламних кампаній; поліпшення показників електронної комерції і установка цілей; дослідження результатів роботи за різними маркетинговими каналами; вироблення рекомендацій щодо поліпшення різних аспектів роботи сайту та взаємодії з відвідувачами[5].

На даний момент найбільш поширеними засобами для збору аналітичної інформації щодо веб-сайту є аналізатори лагів, лічильники відвідуваності, і системи Інтернет-статистики.

Лаг-аналізатори є найбільш складними інструментами такого роду, так як для роботи з ними потрібно досить високий рівень кваліфікації. Вони встановлюються безпосередньо на сервер, в кореневу папку сайту, і

дозволяють збирати статистику практично в усіх напрямках, дозволяючи вирішувати досить специфічні завдання вузького характеру[6].

Таким чином, можна зробити висновки, що аналітика веб-сайтів є необхідним засобом для визначення метрик ефективності роботи сайту, а також його поліпшення та розвитку, перетворення в зручний інструмент для ведення бізнесу або реалізації інших цілей власника ресурсу.

1.2. Аналіз існуючих аналізаторів Web-ресурсів

На сьогоднішній день є багато різних веб-сервісів для здійснення аналізу контенту. Для прикладу розглянемо деякі з них.

На рис. 1.1. зображено головну сторінку веб сайту seositecheckup.com. Він має багато факторів, серед яких поширені питання щодо SEO, оптимізація швидкості, сервер та безпека, мобільні можливості, соціальні медіа.

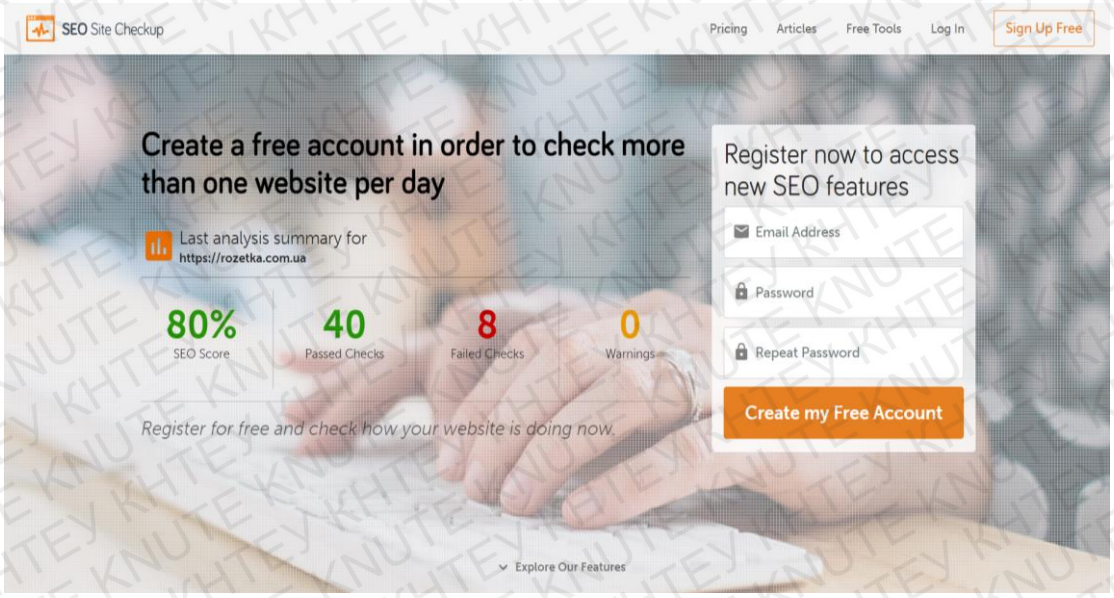

Рис. 1.1. Головна сторінка сайту «seositecheckup»

Це здоровий спосіб переглянути ці параметри та покращити свою присутність в Інтернеті.

Також розглянемо сервіс для аналізу контенту «Сематор». Він пропонує поглиблене технічне та поточне тестування SEO оптимізації для покращення продуктивності веб-сайту.

Більше, ніж 100 випусків аналізуються з різних істотних категорій,

включаючи наступне.

- Якість вмісту
- Соціальні засоби комунікації
- Теги HTML
- Зворотні посилання
- Звіт про сканування
- Швидкість сторінки
- Мобільність використання
- Структуровані дані

На рис. 1.2. зображено вигляд головної сторінки даного сайту.

Рис. 1.2. Головна сторінка сайту «seomator»

Для прикладу візьмемо ще один веб аналізатор Арефс. На рис. 1.3 зображено головну його сторінку. Який поставляється у багатоцільовому наборі інструментів, що забезпечує 360-градусний SEO аналіз.

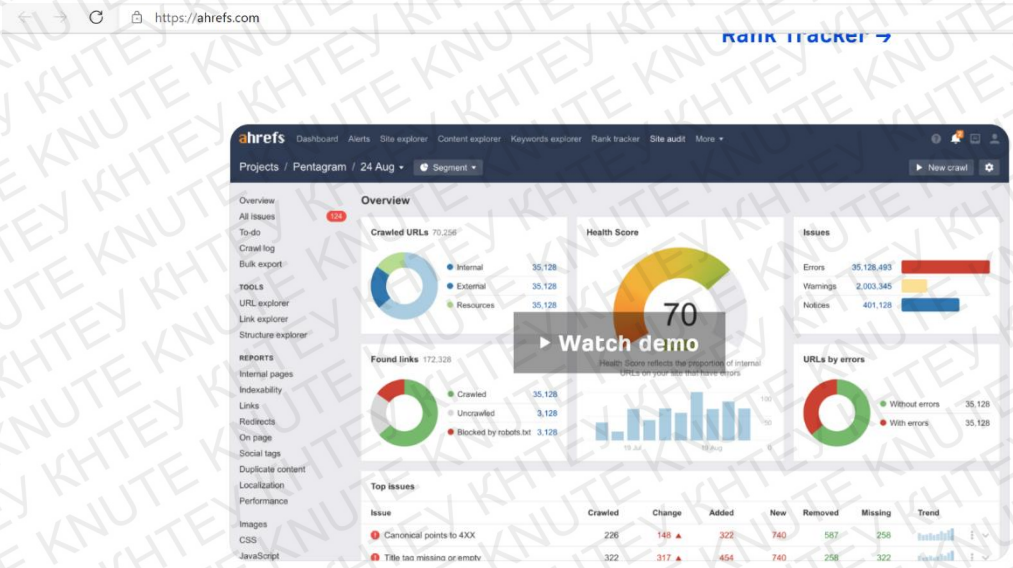

Рис. 1.3. Головна сторінка сайту «ahrefs»

За допомогою інструмента аудиту сайту Ahrefs ми можете переконатися, що немає нічого, що стримувало б ваш сайт у органічному рейтингу. Сканер може заглибитись у ваш сайт і перевірити його 110+ технічних питань.

- Внутрішні та зовнішні сторінки
- Продуктивність
- HTML та соціальні теги
- Якість вмісту
- Локалізація
- Вхідні та вихідні посилання
- JavaScript та CSS
- Зображення

Проаналізувавши існуючі аналоги можна зробити висновок, що вищезазначені сайти є якісними засобами для проведення аналізу контенту веб-сайтів. Кожен сайт функціонує досить добре (не виникало затримок при обробці запиту). Відвідавши представлені веб-сервіси, можна побачити що кожен з сайтів видавав схожу аналітику, представлену у вигляді питань щодо SEO, оптимізації швидкості, серверу та безпеки, мобільної можливості та соціальних медіа.

1.3. Аналіз поведінки аналізатора Web-ресурсів

Система аналізу поведінки включає множину прецедентів, які описують всі взаємодії, які користувачі мають з програмним забезпеченням. Вони відомі як функціональні вимоги. На додачу до прецедентів також включають нефункціональні вимоги, які є вимогами що накладають обмеження на проект. Специфікація вимог до системи включає: глосарій проекту, опис варіантів використання.

Табл. 1.1 є глосарієм основних термінів використаних при створенні системи аналізу веб-сайтів.

Таблиця 1.1

Глосарій основних термінів

Термін	Опис терміну
1. Основні поняття предметної області проекту	
веб-браузер	програмне забезпечення для навігації та перегляду веб сторінок.
веб-сайт	сукупність веб-сторінок доступних у мережі Інтернет.
веб-сервер	набір програм, який забезпечує обмін даними засобами протоколу передачі гіпертексту.
веб-сторінка	інформаційний ресурс доступний у мережі Інтернет, який можна переглянути за допомогою веб-браузера.
інтернет сервіс	послуги, що надаються в Інтернеті за допомогою спеціальних програм.
контент	вміст веб-сторінки.
пошукова	система онлайн-служба, що надає можливість пошуку інформації в Інтернеті.
пошуковий запит	слово, словосполучення чи просто набір символів, які вводить користувач пошукової системи для того, щоб знайти потрібну йому інформацію.

Продовження табл. 1.1.

хостинг	послуга, що надає дисковий простір для розміщення файлів сайту на сервері.
шардінг	процес збереження записів даних на декількох машинах.
URL-посилання	стандартизована адреса певного ресурсу
2. Користувачі системи	
Клієнт	особа, що використовує систему аналізу унікальності контенту веб-сайтів для того, щоб здійснити аналіз контенту певного веб-сайту на унікальність.
3. Документи (вхідні та вихідні)	
HTML 5	Специфікація

Після цього наведемо діаграми варіантів використання для акторів системи рис. 1.4.

Рис. 1.4. Діаграма варіантів використання

Далі представимо прототипи веб сторінки аналізатора. На рис. 1.5. зображено прототип «Аналіз веб-сайту».

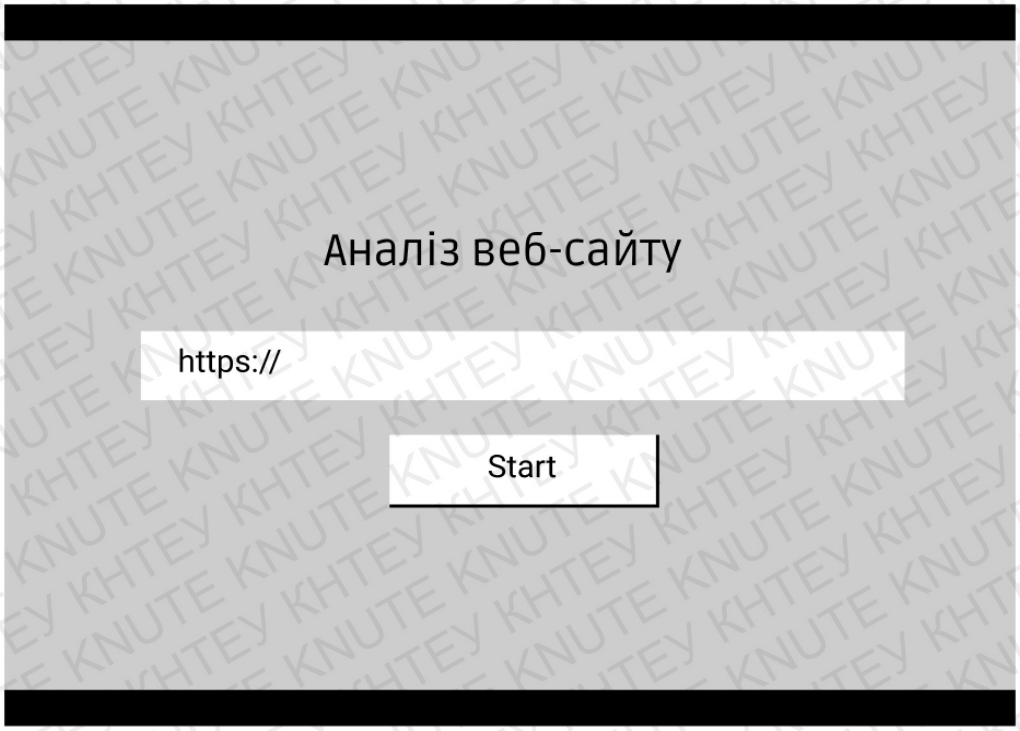

Рис. 1.5. Прототип «Аналіз веб -сайту»

Прототип «Результати аналізу» зображено на рис. 1.6.

Рис. 1.6. Прототип «Результати аналізу»

Варіант використання «Аналіз веб-сайту» наведено у табл. 1.2.

Таблиця 1.2

Варіант використання «Аналіз веб-сайту»

Контекст	Аналіз сайту
Дійові особи	Користувач
Передумова	Авторизація користувача
Тригер	Сайт працює, та відкрита сторінка аналізу
Сценарій	1. Вводимо URL-адрес 2. Кнопка Старт

Варіант використання «Результати перевірки» наведено у табл. 1.3.

Таблиця 1.3

Варіант використання «Результати перевірки»

Контекст	Перегляд результатів перевірки веб-сайту
Дійові особи	Користувач
Передумова	Авторизація користувача
Тригер	Проведено аналіз контенту веб-сайту
Сценарій	1. Відкриття сторінки з результатами 2. Перегляд результатів

Висновок до першого розділу

1. Визначено специфікацію вимог до розробленої системи.
2. Здійснено порівняння аналогів аналізу веб-сайтів, в наслідок чого було

прийнято рішення створення автоматизованої системи аналізу унікальності контенту веб-сайтів.

3. Подано варіанти використання та наведено прототипи функцій.
4. Представлено глосарій основних термінів використаних при створенні системи аналізу веб-сайтів.
5. Визначено всі функціональні та не функціональні вимоги до системи.

РОЗДІЛ 2.
АНАЛІЗ СУЧАСНИХ МЕТОДІВ ТА ТЕХНОЛОГІЙ РОЗРОБКИ АНАЛІЗАТОРІВ WEB-РЕСУРСІВ

2.1. Підходи до проектування архітектури Web-орієнтованого застосунка

Як правило, веб-програми створюються як додатки в архітектурі «клієнт-сервер», але серверна частина має різні архітектурні рішення.

Від самого початку World Wide Web (WWW) представлялася її творцям як “простір для обміну інформацією, в якому люди і комп'ютери можуть спілкуватися між собою”.[7] Тому перші Веб-додатки представляли собою примітивні файлові сервери, які повертали статичні HTML-сторінки що запросив їх клієнтам. Таким чином, Веб починалася як документо-орієнтована.

Наступним етапом розвитку Веб стала поява поняття додатків, які базувалися на таких інтерфейсах, як CGI (або FastCGI), а в подальшому - на ISAPI. Common Gateway Interface (CGI) – це стандартний інтерфейс роботи з серверами, що дозволяє виконувати серверні додатки, що викликаються через URL. Вхідний інформацією для таких додатків служило вміст HTTP-заголовка (і тіло запиту при використанні протоколу POST). CGI-додатки генерували HTML-код, який повертався браузеру. Основною проблемою CGI-додатків було те, що при кожному клієнтському запиті сервер виконував CGI-програму в реальному часі, завантажуючи її в окремий адресний простір[8].

Природно, що при використанні як CGI-, так і ISAPI-додатків розробники в основному вирішували одні й ті ж завдання, тому природним кроком стала поява нового, високорівневого інтерфейсу, який спростив завдання генерації HTML-коду, дозволив звертатися до компонентів і використовувати бази даних. Таким інтерфейсом стала об’єктна модель Active Server Pages (ASP), побудована на основі ISAPI-фільтра[8].

Основною ідеєю ASP з точки зору створення інтерфейсу додатку є те, що на Веб-сторінці присутні фрагменти коду, який інтерпретується Веб-сервером

і замість якого користувач отримує результат виконання цих фрагментів коду.

Незабаром після появи ASP були створені й інші технології, що реалізують ідею розміщення всередині Веб-сторінки коду, що виконується Веб-сервером. Найбільш відома з них на сьогоднішній день – технологія JSP (Java Server Pages), основною ідеєю якої є одноразова компіляція Java-коду (сервлету) при першому зверненні до нього, виконання методів цього сервлета і приміщення результатів виконання цих методів в набір даних, що відправляються в браузер[8].

Новітня версія технології Active Server Pages - ASP.NET, що є ключовою в архітектурі Microsoft NET Framework. За допомогою ASP.NET можна створювати Веб-додатки та Веб-сервіси, які не тільки дозволяють реалізувати динамічну генерацію HTML-сторінок, але і інтегруються з серверними компонентами і можуть використовуватися для вирішення широкого кола бізнес-задач, що виникають перед розробниками сучасних Веб-додатків[8].

У загальному випадку клієнтом Веб-сервера може бути не тільки персональний комп’ютер, оснащений звичайним Веб-браузером. Одночасно з широким розповсюдженням мобільних пристроїв з’явилася і проблема надання Веб-серверами даних, які можуть бути інтерпретовані цими пристроями. Оскільки мобільні пристрої мають характеристики, відмінними від характеристик персональних комп'ютерів (обмеженим розміром екрану, малим обсягом пам’яті, а нерідко і неможливістю відобразити що-небудь, крім кількох рядків чорно-білого тексту), для них існують і інші протоколи передачі даних і відповідні мови розмітки. При цьому виникає задача передачі даних на мобільний пристрій у відповідному форматі (і для цієї мети існують спеціальні сайти), або, що видається більш зручним, відбувається впізнання типу пристрою в момент його звернення до сервера і перетворення вихідного документа в формат, що потребується даному мобільного пристрою (Наприклад, за допомогою XSLT-перетворення)[8].

Іншим способом підтримки різних типів клієнтів є створення “розумних” серверних компонентів, які здатні генерувати різний код в залежності від типу

клієнта. Такий підхід, зокрема, реалізований в Microsoft ASP.NET.

Частина рішення порталу, як правило, є методом управління вмістом веб-сайтів, оскільки обсяг даних, який може бути наданий користувачам через великі компанії та портали, великий, тепер неможливо керувати цими даними “вручну”.

Підсумовуючи вищезазначений зміст, ми можемо зупинитися на основних функціях веб-архітектури:

- немає необхідності використовувати інше програмне забезпечення на клієнті – це дозволяє автоматично реалізовувати клієнтську частину на всіх платформах;
- можливість підключення до майже необмеженої кількості клієнтів;
- завдяки наявності єдиного місця зберігання даних та системи управління базами даних, вона забезпечує мінімальні вимоги для підтримки цілісності даних;
- наявність серверів та каналів зв’язку під час нормальної роботи;
- недоступний без сервера або каналу зв’язку;
- швидкість веб-сервера та каналу передачі даних досить низька;
- що стосується обсягу даних - очевидних обмежень щодо архітектури веб-системи немає.

При побудові архітектури веб-додатків необхідно мінімізувати залежності між структурними підрозділами[8]. Як правило, додаток складається з трьох структурних підрозділів навединих на рис. 2.1.

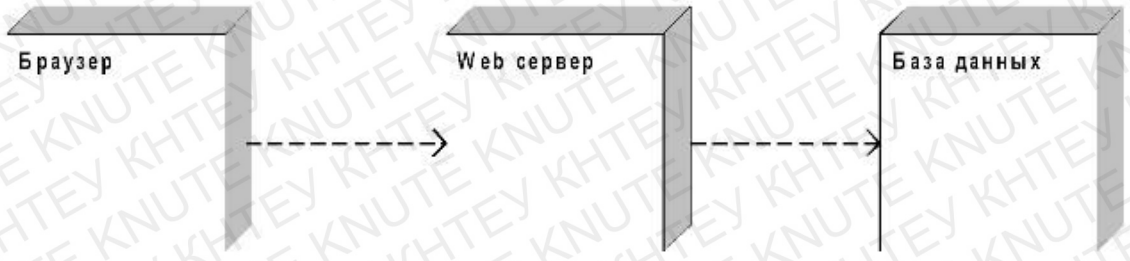

Рис. 2.1. Структурні підрозділи веб-додатку

1. Модуль, який працює під управлінням браузера.
2. Модуль, який працює під управлінням web-сервера.
3. База даних.

Ці структурні одиниці породжують два види зв’язків.

1. Зв’язок між браузером і серверної частиною.
2. Зв’язок між серверної частиною і базою даних.

Для досягнення мети максимальної незалежності між структурними підрозділами необхідно зробити так, щоб кожна структурна одиниця працювала лише на необхідних наборах даних. Розглянемо докладніше.

Браузер - це прикладна програма, яка використовується для перегляду веб-сторінок.

HTML є стандартною мовою розмітки документів. Більшість сучасних веб-браузерів можуть інтерпретувати мову HTML.

Веб-сервер – це програмне забезпечення, яке може отримувати HTTP-запити від клієнтів, обробляти їх та надсилати відповіді відповідно до стандартів протоколу.

База даних – це сукупність незалежних матеріалів, представлених у об’єктивній формі та систематизованих систематизовано, щоб ці матеріали могли знаходити та обробляти комп’ютери.

Взаємодія між базою даних та веб-сервером може бути організована на основі двох принципово різних схем:

1. Бізнес логіка знаходиться в базі даних.
2. Бізнес логіка знаходиться в коді web-сервера.

У першому випадку база даних зберігає дані та надає інтерфейси доступу до даних:

1. Вибірка даних вирішується шляхом подання.
2. Модифікація даних вирішена в процесі збереження.

Програма веб-сервера – це програма-драйвер, яка використовується для доступу до бізнес логіки. Іншими словами, він лише підключає браузер до ділової логіки, реалізованої в базі даних.

У другому випадку база даних зберігає дані та забезпечує прямий доступ до даних. Бізнес-логіка реалізована в коді веб-сервера. У цьому випадку база даних забезпечує транзакції для атомарних операцій[8].

Для того, щоб мінімізувати залежність між веб-сервером і базою даних, необхідно визначити ділову логіку лише в одному місці. Іншими словами, це може бути в коді веб-сервера або в базі даних. Це дуже важливо, оскільки це зменшує витрати на обробку запитів на зміну. Це пов’язано з тим, що зміна структурного підрозділу не перевищує його.

2.2. Огляд засобів розробки клієнтської частини аналізатора Web-ресурсів

Характеристика архітектури клієнт-сервер програми полягає в тому, що в системі є один або кілька серверів і один або більше клієнтів. Сервер і клієнт є незалежними частинами розподіленої інформаційної системи, включаючи програмні та апаратні компоненти[9].

Зазвичай клієнтська частина програми спілкується з користувачем через користувацький інтерфейс, формує параметри запиту користувача, а потім відправляє запит на сервер. Серверна частина приймає запит, виконує всі необхідні обчислення, а потім відправляє результати назад клієнту.

У випадку веб-програмування учасниками комунікації є сервер, клієнт і протокол для зв'язку між сервером і клієнтом. В загальному для веб-сайту це буде:

- протокол HTTP (HyperText Transfer Protocol). Прикладний мережний протокол на базі стеку TCP/IP, який призначено для передачі гіпертексту;
- веб-клієнт - Інтернет браузер;
- веб-сервер - додаток, який вміє обробляти HTTP запити[9].

Оскільки програма забезпечує зв’язок між клієнтом та сервером, контекст поділяється на клієнта та сервер. Нижче наведено методи збереження та відновлення контексту виконання веб-програми на клієнті.

Контекст виконання клієнта може зберігатися:

- В оперативній пам’яті клієнтської програми (браузера). Для цих цілей з випуском HTML5 браузери повинні підтримувати SessionStorage та LocalStorage. Це дуже розумно, оскільки в ньому використовується оперативна пам’ять клієнта замість оперативної пам’яті сервера, яка є “один на один”. З іншого боку, не всі дані можуть зберігатися на клієнті, оскільки не всі дані можуть бути перетворені в текстовий формат і передані на сервер.
- У невеликих фрагментах текстових даних, що зберігаються на клієнтських файлах cookie. Файли cookie зберігаються у текстових файлах, які операційна система виділяє для зберігання різної інформації про користувачів. Ці дані передаються на сервер щоразу в заголовку запиту HTTP. Особливим недоліком цього методу є те, що клієнт може заборонити використання файлів cookie.

Іншим досить надійним способом зберігання даних стану сеансу є використання елементів розмітки HTML. Клієнтська програма може за допомогою JavaScript і DOM створювати приховані елементи із прихованими полями, про які знає лише сервер. У свою чергу, сервер обробить ці дані та помістить у них відповіді (за необхідності).

Розробка дизайну та функцій графічного інтерфейсу користувача веб-додатків, створюється з використанням мови розмітки гіпертексту та суміжних технологій[10]. Можна навести наступні засоби розробки:

- Основи HTML / XHTML. Мова розмітки XML загального призначення.
- Шаблони та макети. Вони втілюють типові завдання, з якими веб-дизайнери часто мають справу: створення макетів або шаблонів для веб-сторінок, меню, елементів керування закладками, ієрархічної навігації по дереву тощо.
- HTML5. Усі сучасні браузери після цього підтримуватимуть інновації.
- CSS (каскадні таблиці стилів) – це спеціальна мова стилю сторінок, що

використовується для опису їхнього зовнішнього вигляду.

Програмування на стороні клієнта безпосередньо пов’язане з веб-дизайном, але винесено окремим пунктом. У розробці використовуються наступні засоби розробки:

- JavaScript. Популярна мова, яку підтримують усі сучасні браузери. На HTML сторінку додають JavaScript код і використовують об'єктно-орієнтованих підхід при написанні користувацьких сценаріїв.
- jQuery. Бібліотека, написана на JavaScript, яка включає велику кількість корисних функцій для обробки об’єктної моделі завантажених веб-сторінок: від редагування вмісту сторінки до створення різних візуальних ефектів.
- AJAX. Технологія, що використовується для створення динамічних веб-сторінок на основі асинхронного завантаження гіпертексту (результат асинхронного виконання HTTP-запитів) у певних областях поточної сторінки. Технологія AJAX використовує бібліотеки JavaScript та jQuery.

2.3. Обґрунтування вибору програмних та технічних засобів розробки аналізатора Web-ресурсів

Для реалізації даного завдання було обрано мову програмування C# та платформу ASP.NET. Це популярна платформа для розробки веб додатків, розроблена компанією Microsoft. ASP.NET включає в себе такий тип (фрейморк) веб-додатку як ASP.NET MVC, конкретно ця технологія реалізує шаблон Model-View-Controller. Також на мові C# існує безліч бібліотек для аналізу сайту. Тому було обрано мову C# та платформу ASP.NET

Структура архітектури MVC розділяє додаток на три основні групи компонентів:

- Модель (model): описує використовувані в додатку дані, а також логіку, яка пов'язана безпосередньо з даними, наприклад, логіку валідації даних. Як правило, об’єкти моделей зберігаються в базі даних[11].

- Уявлення (view): відповідають за візуальну частину або призначений для користувача інтерфейс, нерідко html-сторінка, через яку користувач взаємодіє з додатком. Також уявлення може містити логіку, пов'язану з відображенням даних. У той же час уявлення не повинно містити логіку обробки запиту користувача або управління даними[12].
- Контролер (controller): представляє центральний компонент MVC, який забезпечує зв'язок між користувачем та програмою, поданням і сховищем даних. Він містить логіку обробки запиту користувача. Контролер отримує введені користувачем дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді подання, наповненого даними моделей[13].

Вищесказане дозволяє реалізувати принципи поділу завдань. На рис. 2.2 показані три основних компоненти і існуючі між ними зв'язки.

Рис. 2.2. Структура архітектури MVC

Перевагами платформи ASP.NET MVC над іншими будуть наступні пункти:

- Архітектурний шаблон MVC. Інфраструктура ASP.NET MVC Framework реалізує шаблон MVC і при цьому забезпечує істотно поліпшений поділ відповідальності.

- Можливість розширення. Інфраструктура MVC Framework побудована у вигляді набору незалежних компонентів, які задовольняють інтерфейсу .NET або створені на основі абстрактного базового класу.
- Жорсткий контроль над HTML і HTTP. Інфраструктура ASP.NET MVC генерує ясний і відповідний стандартам код розмітки. Її вбудовані допоміжні методи HTML виробляють стандартизований висновок.
- Можливість тестування. Природний розподіл різних відповідальностей додатку з незалежних один від одного частин програмного забезпечення, яке підтримується архітектурою MVC, дозволяє спочатку будувати легко супроводжувані і тестовані програми. Проте проектувальники ASP.NET MVC на цьому не зупинилися. Для кожного фрагмента компонентно-орієнтованого проекту інфраструктури вони забезпечили структурованість, необхідну для задоволення вимог модульного тестування і засобів імітації.
- Потужна система маршрутизації. Удосконалення технології побудови веб-додатків супроводжувалося розвитком стилю URL-адрес.
- Побудова на основі кращих частин платформи ASP.NET.
- Сучасний API-інтерфейс. Платформа Microsoft .NET розвивалася з кожним великим випуском, підтримуючи багато передових аспектів сучасного програмування.
- Інфраструктура ASP.NET MVC має відкритий код[14].

Висновок до другого розділу

1. Наведено підходи до проектування архітектури Web-орієнтованого застосунка.
2. Визначено структурні підрозділи веб-додатку.
3. Подано засоби обробки клієнтської частини веб-додатку та вибрано найоптимальніші варіанти.
4. Вибрано програмні та технічні засоби розробки, а саме платформу

ASP.NET MVC.

5. Наведено переваги платформи ASP.NET MVC.

РОЗДІЛ 3.
РОЗРОБКА АНАЛІЗАТОРА WEB-РЕСУРСІВ

3.1. Розробка архітектури аналізатора Web-ресурсів

Для того, щоб краще зрозуміти процес роботи системи побудовано діаграми станів для кожного модуля.

Діаграма діяльності модуля формування результату показана на рис. 3.1.

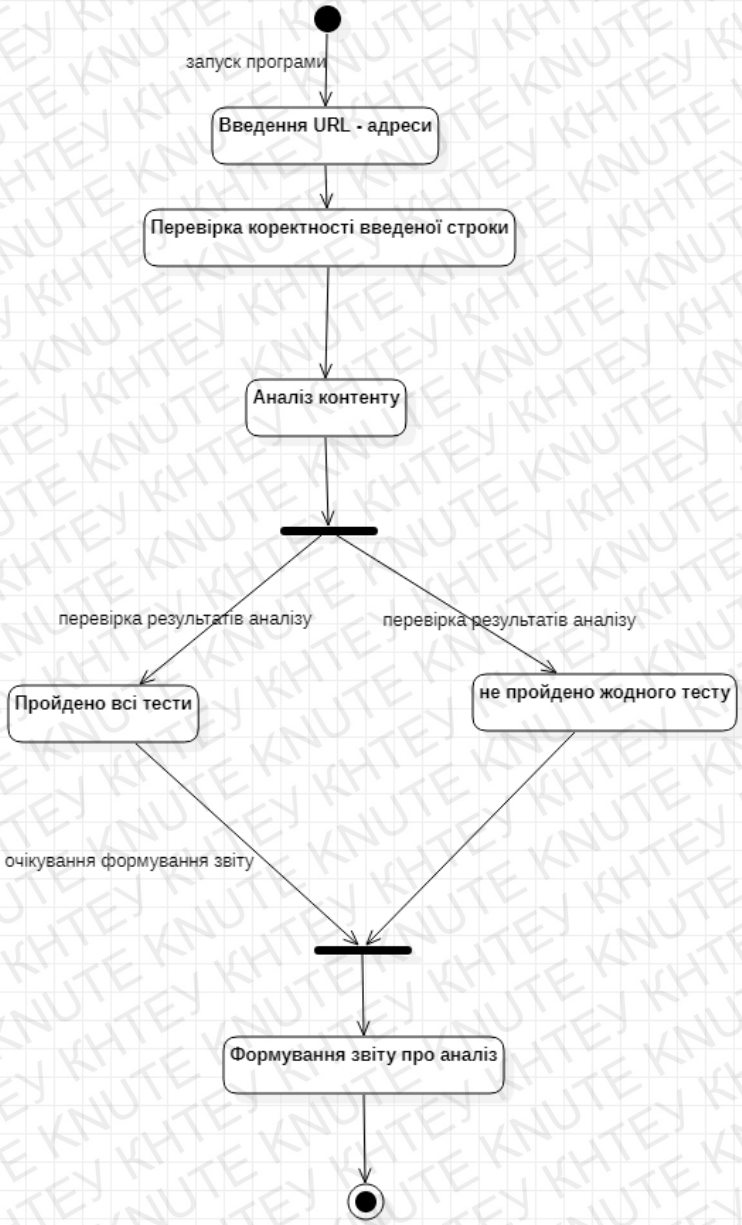

Рис. 3.1. Діаграма діяльності для модуля формування результату

Діаграма активності модуля виконання аналізу контенту зображена на рис. 3.2.

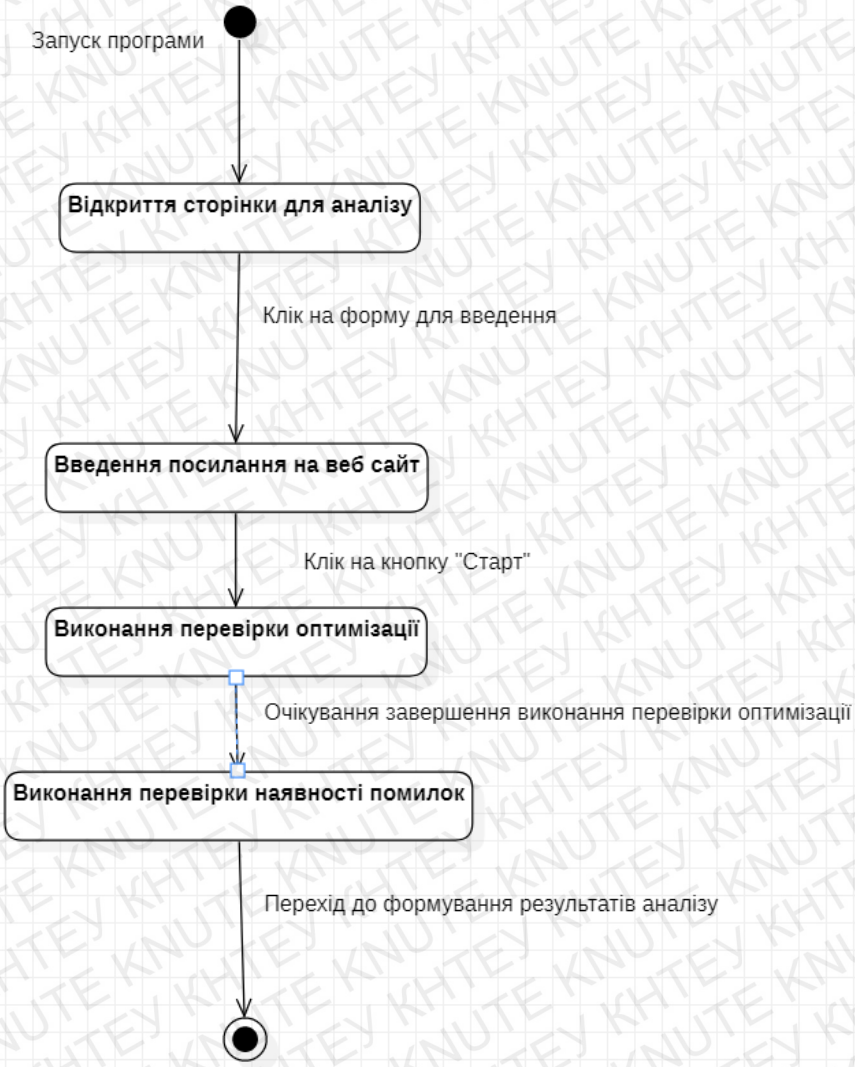

Рис. 3.2. Діаграма активності модуля виконання аналізу

Процес аналізу вмісту відбувається з допомогою користувача, який взаємодійте із програмною системою. Користувач вводить посилання в поле і натискає кнопку «Проаналізувати!». Після цього здійснюється аналіз вмісту веб-сайту.

Результатом аналізу є звіт, що включає:

- результатів перевірки оптимізованості веб-сайту;
- результатів перевірки наявності помилок;

Далі представлено ідентифікатори системи, які наведені в табл. 3.1.

Таблиця 3.1

Таблиця ідентифікаторів

Об’єкт	Властивість	Тип	Ідентифікатор
Користувач	Адреса сайту	Текст	URL
Аналіз контенту веб-сайту	Перевірка оптимізованості	Текст	Site
Перевірка оптимізованості веб-сайту	Адреса Заголовок Автор Мета теги Теги h1-h6 Ключові слова Наявність протоколу Open Graph Наявність іконки сайту Наявність кодування Наявність Viewport	Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст	Url Title Author Metas HeadingsCount Keywords ContainsOpenGraph ContainsFavicon ContainsCharset ContainsViewport
Звіт	Результат	Текст	result

Наступним етапом є конструювання діаграми класів UML, вона відображена на рис. 3.3.

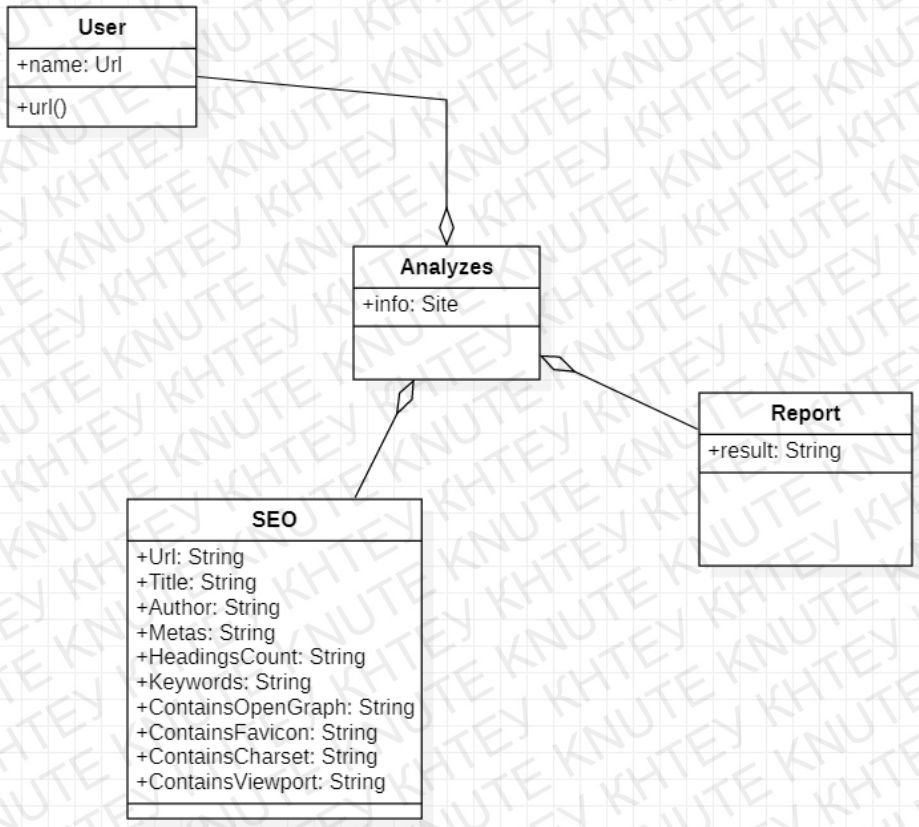

Рис. 3.3. Діаграма класів UML

Проектування структури бази даних дозволяє визначити та повністю описати всі відношення і поля та є остаточною ланкою перед етапом програмної реалізація бази даних.

3.2. Програмна реалізація проекту

Для реалізації даного завдання було обрано мову програмування C# та платформу ASP.NET.

ASP.NET – технологія створення веб-застосунків і веб-сервісів від компанії Майкрософт. Вона є складовою частиною платформи Microsoft.NET і розвитком старішої технології Microsoft ASP. На цей час останньою версією цієї технології є ASP.NET Core 6.0. ASP.NET зовні багато в чому зберігає схожість із старішою технологією ASP, що дає змогу розробникам відносно легко перейти на ASP.NET. У той же час внутрішній устрій ASP.NET істотно

відрізняється від ASP, оскільки вона заснована на платформі. NET і, отже, використовує всі нові можливості, що надаються цією платформою[15].

C# - об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Було розроблено декілька версій даної мови. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато від своїх попередників – мов C++, Object Pascal, Модула і Smalltalk - C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, мова C#, на відміну від C++, не передбачає множинне успадкування класів[16].

Ця популярна платформа для розробки веб додатків, розроблена компанією Microsoft. ASP.NET включає в себе такий тип (фрейморк) веб-додатку як ASP.NET MVC, конкретно ця технологія реалізує шаблон Model-View-Controller. Також на мові C# існує безліч бібліотек для аналізу сайту. Тому було обрано мову C# та платформу ASP.NET.

Щоб скомпонувати елементи інтерфейсу для веб-орієнтованої системи використовувалися такі мови як HTML та CSS.

HTML (Hyper Text Markup Language – мова розмітки гіпертексту) – це мова тегів, засобами якої здійснюється розмічання веб-сторінок для мережі Інтернет. Браузери отримують HTML-документи з веб-сервера або з локальної пам'яті й передають документи в мультимедійні веб-сторінки. HTML описує структуру веб-сторінки семантично і спочатку включені сигнали для зовнішнього вигляду документа[17].

CSS (Cascading Style Sheets) – це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних.

CSS є основною технологією всесвітньої павутини, поряд із HTML та JavaScript.

Основними функціями та моделями системи є:

- Title – перевірка наявності та вмісту тегу title;
- Author – перевірка наявності та вмісту тегу autor;
- Metas – перевірка наявності та вмісту тегу meta;
- HeadingsCount – перевірка наявності заголовків H1-H6;
- Keywords – перевірка наявності ключових слів;
- ContainsOpenGraph – перевірка наявності протоколу Open Graph;
- ContainsFavicon – перевірка наявності значка, який відображається поруч з назвою сайту у вікні браузера або у результатах пошукової системи;
- ContainsCharset – перевірка наявності кодування;
- ContainsViewport – виконання скріншоту головної сторінки сайту.

Для прикладу, наведемо код двох функцій:

1. ContainsFavicon

```
public static bool CheckFavicon()
    {
        List<string> metasList = new List<string>();
        var links = CQ.Find("link");
        foreach (var link in links)
        {
            var rel = link.GetAttribute("rel");
            if (rel != null)
                if (rel.Contains("icon"))
                    return true;
            var type = link.GetAttribute("type");
            if (type != null)
                if (type.Contains("image"))
                    return true;
        }
```

```
        return false;
    }
```

2. CheckTitle

```
public static string CheckTitle()
{
    var title = CQ.Find("title").FirstOrDefault();
    if (title != null)
        return title.InnerText;
    else
        return "";
}
```

Лістинг коду програми наведено в додатках А, Б, В, Г.

Для ефективної роботи веб-додатку визначено наступні вимоги до обладнання системи:

Вимоги до сервера:

1. центральний процесор Intel® Core™ i7-4810M Processor 3,0Ггц;
2. жорсткий диск з вільним 40 Гб пам’яті;

Вимоги до клієнта:

1. Windows 7-10 x32-x64; 47
2. Google Chrome 50.0.1750.154 чи новіший.

3.3. Тестування та дослідна експлуатація

Тестування є одним із завершальних етапів процесу розробки програмного продукту. Це дозволяє виявити помилки та недоліки з попередніх етапів розробки програмного забезпечення.

Було проведено системний вид тестування програми.

Коли проводиться функціональне тестування виявляються невідповідності між очікуваною та реальною поведінкою програмного продукту відповідно до функціональних вимог[18]. Результати проведення функціонального тестування програмної системи подано у табл. 3.2.

Таблиця 3.2

Результати функціонального тестування

Назва вимоги	Результат проходження тестування
Здійснення аналізу контенту веб-сайту	+
Перегляд результатів перевірки наявності помилок	+

3.4. Інструкція користувача

Програмна система розроблено на мові програмування C# з використанням платформи ASP.NET MVC. Для коректної роботи програмної системи необхідно встановити браузер Google Chrome чи альтернативний йому браузер останньої версії. Програмна система призначена для здійснення аналізу контенту веб-сайту, тобто перевірки оптимізованості. До програмної системи входять 2 основні модулі, інструкцію для роботи з якими наведено нижче.

Для того, щоб здійснити аналіз веб-сайту, користувачу необхідно заповнити поле коректними даними та натиснути на кнопку «Проаналізувати!» (рис. 3.4).

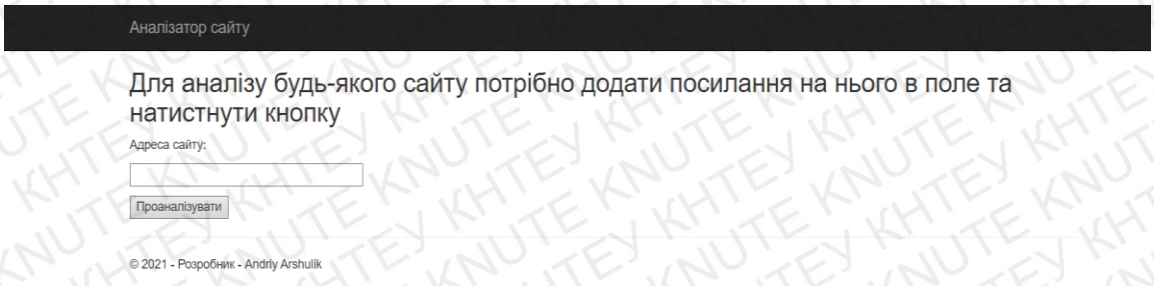

Рис. 3.4. Сторінка для аналізу контенту веб-сайту

Результати перевірки оптимізованості веб-сайту можна переглянути після здійснення аналізу контенту веб-сайту, дана інформація відображається під кнопкою «Проаналізувати» (рис. 3.5).

Рис. 3.5. Результати виконання перевірки оптимізованості веб-сайту

Висновки до третього розділу

1. Обґрунтовано вибір мови програмування.
2. Описано процес програмної реалізації програмної системи.
3. Використовуючи мову програмування C# та платформу ASP.NET MVC розроблено програмний продукт.

4. Описано процеси реалізації програмної системи.
5. Проведено функціональне тестування всіх варіантів використання та тестування безпеки програмної системи.
6. Детально описана інструкція користувача.

ВИСНОВКИ

В процесі виконання дипломної роботи розроблено програмну систему для аналізу унікальності контенту веб-сайтів, яка допомагає користувачу вирішити проблему аналізу унікальності вмісту веб-сайтів з точки зору оптимізатора SEO. Для розробки застосунку була обрана об'єктно-орієнтована мова програмування C#, що дозволяє забезпечити швидку та стабільну роботу.

Також за допомогою програмної системи будь-який звичайний користувач, незалежно від рівня знання SEO, зможе здійснити аналіз веб-сайту і дізнатися про його оптимізованість та побачити всі допущені неточності, або ж помилки при розробці його інтерфейсу.

Як наслідок, головною перевагою даного веб-додатку є його простота, де користувачу слід просто ввести URL адрес веб-сайту та отримати результати аналізу, після чого він зможе удосконалити свою площадку для більшої відвідуваності, швидкості та ергономічності.

Створення програмної системи складалося з таких етапів:

1. Загальна характеристика підходів до аналізу WEB-ресурсів. На даному етапі було здійснено опис предметної області, проведено аналіз існуючих аналогів, що реалізують функції предметної області та сформовано специфікацію функціональних та нефункціональних вимог до програмної системи.
2. Аналіз проектування та технологій. Проаналізовано підходи до проектування архітектури та оглянуто засоби розробки клієнтської частини аналізатора Web-ресурсів.
3. Програмна реалізація. На цьому етапі було розроблено архітектуру додатку та програмну систему за допомогою мови програмування C# з використанням платформи ASP.NET MVC. Також було проведено тестування та дослідну експлуатація.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Веб-сайт [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Вебсайт .
2. Оптимізація контенту веб-сторінки [Електронний ресурс]. – Режим доступу: http://igroup.com.ua/seo-articles/optymizatsiya-tekstu/ .
3. Унікальність контенту [Електронний ресурс]. – Режим доступу: http://igroup.com.ua/seo-articles/unikalnist/ .
4. Аналіз контенту веб-сайтів українських центрів науково-технічної інформації з точки зору використання інтернет-сервісів - Людмила Філіпова - ISSN 2076-9326. Вісник Книжкової палати. 2012. № 10.
5. Веб-аналітика як важлива складова успішного функціонування ЗМІ в Інтернеті. Ірина Мудра - УДК 070:659.3(477).
6. Способи унікалізації контенту на сайті [Електронний ресурс]. – Режим доступу: http://andrey.lviv.ua/blog/sposobi-unikalizatsii-kontentu .
7. Контент сайту та його складові [Електронний ресурс]. – Режим доступу: http://webstudio2u.net/ua/studio-web/686-kontent-saita.html .
8. Проектування і розробка корпоративних веб-додатків [Електронний ресурс]. – Режим доступу: https://subcase.ru/uk/proektirovanie-i-razrabotka-korporativnyh-web-prilozhenii-arhitektura.html .
9. Клієнт-серверна архітектура [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура .
10. Розробка WEB – додатків [Електронний ресурс]. – Режим доступу: https://tqm.com.ua/ua/rozrobka-veb-dodatkiv-servisiv-web-application-development/rozrobka-web-dodatkiv-dlia-bizniesu#klient-servernye%20https://metanit.com/sharp/aspnet5/3.1.php .
11. ASP.NET MVC Framework [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/ASP.NET_MVC_Framework.
12. Уявлення [Електронний ресурс]. – Режим доступу: https://stud.com.ua/97667/informatika/uyavlennya.

13. Відомості ASP.NET Core MVC, функції, структура та компоненти [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/ru-ru/aspnet/core/mvc/overview?view=aspnetcore-5.0.
14. Переваги ASP.NET Core MVC [Електронний ресурс]. – Режим доступу: https://professorweb.ru/my/ASP_NET/mvc/level1/1_2.php.
15. Характеристики ASP.NET [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/ASP.NET.
16. Мова програмування C# [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/C_Sharp .
17. HTML – мова розмітки гіпертексту [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/HTML .
18. Основні види тестування програмного забезпечення [Електронний ресурс]. – Режим доступу: https://qagroup.com.ua/publications/vydy-testuvannya-ta-vidminnosti-mizh-nymy/.

ДОДАТКИ

Додаток А

Програмний код контролера Web-додатку

```
using SiteAnalyzer.Models;
using SiteAnalyzer.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace SiteAnalyzer.Controllers
{
   public class HomeController : Controller
   {
      public ActionResult Index()
      {

         return View();
      }
      [HttpPost]
      public ActionResult Index(Site siteUrl)
      {
         //string url = "https://stackoverflow.com/";
         Site site = new Site();
         SiteAnalyzerService.Anylize(siteUrl.Url);
         site.Url = siteUrl.Url;
         site.Title = SiteAnalyzerService.CheckTitle();
         site.HeadingsCount = SiteAnalyzerService.CheckHeadings();
         site.Metas = SiteAnalyzerService.CheckMetas();
         site.ContainsOpenGraph = SiteAnalyzerService.CheckOpenGraph();
         site.ContainsFavicon = SiteAnalyzerService.CheckFavicon();
         site.Author = SiteAnalyzerService.CheckAuthor();
         site.ContainsCharset = SiteAnalyzerService.CheckCharset();
         site.Keywords = SiteAnalyzerService.CheckKeywords();
         site.ContainsViewport = SiteAnalyzerService.CheckViewport();
         return View(site);
      }
      public ActionResult About()
      {
         ViewBag.Message = "Your application description page.";

         return View();
```

```
    }

    public ActionResult Contact()
    {
        ViewBag.Message = "Your contact page.";

        return View();
    }
}
                                        }
```

Додаток Б

Програмний код моделей Web-додатку

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Web;

namespace SiteAnalyzer.Models
{
   public class Site
   {
      public string Url { get; set;}
      public string Title { get; set; }
      public string Author { get; set; }
      public List<int> HeadingsCount { get; set; }
      public List<string> Metas { get; set; }
      public List<string> Keywords { get; set; }
      public bool ContainsCharset { get; set; }
      public bool ContainsViewport { get; set; }
      public bool ContainsOpenGraph { get; set; }
      public bool ContainsFavicon { get; set; }

   }
}
```

Додаток В

Програмний код логічної структури Web-додатку

```
using OpenQA.Selenium;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Net;
using System.Web;
using OpenQA.Selenium.Remote;
using CsQuery;

namespace SiteAnalyzer.Services
{
   public class SiteAnalyzerService
   {
      public static CQ CQ { get; set; }
      public static bool Anylize(string url)
      {
         try
         {
            WebClient client = new WebClient();
            Stream data = client.OpenRead(new Uri(url));
            StreamReader reader = new StreamReader(data);
            string htmlContent = reader.ReadToEnd();
            data.Close();
            reader.Close();
            CQ = CQ.Create(htmlContent);
            return true;
         }
         catch
         {
            return false;
         }

      }
      public static string CheckTitle()
      {
         var title = CQ.Find("title").FirstOrDefault();
         if (title != null)
            return title.InnerText;
```

```
    else
        return "";
}
public static List<int> CheckHeadings()
{
    List<int> headingsCount = new List<int>();
    for (int i = 1; i<=6; i++)
    {
        var headings = CQ.Find("h" + i.ToString());
        headingsCount.Add(headings.Count());
    }
    return headingsCount;
}
public static List<string> CheckMetas()
{
    List<string> metasList= new List<string>();
    var metas = CQ.Find("meta");
    foreach(var meta in metas)
    {
        metasList.Add(meta.GetAttribute("content"));
    }
    return metasList;
}
public static bool CheckFavicon()
{
    List<string> metasList = new List<string>();
    var links = CQ.Find("link");
    foreach (var link in links)
    {
        var rel = link.GetAttribute("rel");
        if (rel != null)
            if (rel.Contains("icon"))
                return true;
        var type = link.GetAttribute("type");
        if (type != null)
            if (type.Contains("image"))
                return true;
    }
    return false;
}
public static bool CheckOpenGraph()
{
    var html = CQ.Text();
    var metas = CQ.Find("meta");
```

```
    if (html.Contains("og: http://ogp.me/ns#"))
        return true;
    foreach (var meta in metas)
    {
        var property = meta.GetAttribute("property");
        if (property != null)
            if (property.Contains("og:"))
                return true;
    }
    return false;
}
public static string CheckAuthor()
{
    var metas = CQ.Find("meta");
    foreach (var meta in metas)
    {
        var name = meta.GetAttribute("name");
        if (name != null)
            if (name.Contains("author"))
                return meta.GetAttribute("content");
    }
    return "";
}
public static bool CheckCharset()
{
    var metas = CQ.Find("meta");
    foreach (var meta in metas)
    {
        if (meta.GetAttribute("charset") != "")
            return true;
    }
    return false;
}
public static List<string> CheckKeywords()
{
    List<string> keywords = new List<string>();
    var metas = CQ.Find("meta");
    foreach (var meta in metas)
    {
        var name = meta.GetAttribute("name");
        if (name != null)
            if (name.Contains("keywords"))
            {
```

```
keywords.AddRange(meta.GetAttribute("content").Split(',').ToList());
                }
            }
            return keywords;
        }
        public static bool CheckViewport()
        {
            var metas = CQ.Find("meta");
            foreach (var meta in metas)
            {
                var name = meta.GetAttribute("name");
                if (name != null)
                    if (name.Contains("viewport"))
                        return true;
            }
            return false;
        }
    }
}
```

Додаток Г

Програмний код головної сторінки Web-додатку

```
@model SiteAnalyzer.Models.Site

@{
   ViewBag.Title = "Index";
}

<h2>Для аналізу будь-якого сайту потрібно додати посилання на нього в поле та натистнути кнопку</h2>

@using (Html.BeginForm("Index", "Home", FormMethod.Post))
{
   <p>Адреса сайту: </p>
   @Html.TextBox("Url", "", new { size = "300" })
   <p></p>
   <p><input type="submit" value="Проаналізувати" /></p>
}

@section Scripts {
   @Scripts.Render("~/bundles/jqueryval")
}

@if (Model != null)
{
   <br />
   <div class="container">
      <div class="row justify-content-center">
         <table class="table">
            <thead>
               <tr>
                  <th>Параметр</th>
                  <th>Значення</th>
               </tr>
            </thead>
            <tbody>
               <tr>
                  <td>Адреса: </td>
                  @if (Model.Url != null)
                  {
                     <td>@Model.Url</td>
                  }
                  else
```

```
    {
        <td>Адреси немає</td>
    }
</tr>
<tr>
    <td>Заголовок: </td>
    @if (Model.Title != null)
    {
        <td>@Model.Title</td>
    }
    else
    {
        <td>Заголовку немає</td>
    }
</tr>
<tr>
    <td>Автор: </td>
    @if (Model.Author != null)
    {
        <td>@Model.Author</td>
    }
    else
    {
        <td>Автора немає</td>
    }
</tr>
<tr>
    <td>Мета теги(@Model.Metas.Count()):</td>
    @if (Model.Metas != null)
    {

        <td>
            @foreach (var m in Model.Metas)
            {
                <p>@m</p>
            }
        </td>
    }
    else
    {
        <td>Мета тегів немає</td>
    }
</tr>
<tr>
```

```
    <td>Теги h1-h6:</td>
    @if (Model.HeadingsCount != null)
    {
        <td>
            <p>Кількість тегів h1: @Model.HeadingsCount[0]</p>
            <p>Кількість тегів h2: @Model.HeadingsCount[1]</p>
            <p>Кількість тегів h3: @Model.HeadingsCount[2]</p>
            <p>Кількість тегів h4: @Model.HeadingsCount[3]</p>
            <p>Кількість тегів h5: @Model.HeadingsCount[4]</p>
            <p>Кількість тегів h6: @Model.HeadingsCount[5]</p>
        </td>
    }
    else
    {
        <td>Тегів h1-h6 немає</td>
    }
</tr>
<tr>
    <td>Ключові слова(@Model.Keywords.Count()):</td>
    @if (Model.HeadingsCount != null)
    {
        <td>
            @foreach (var k in Model.Keywords)
            {
                <p>@k</p>
            }
        </td>
    }
    else
    {
        <td>Ключових слів немає</td>
    }
</tr>
<tr>
    <td>Наявність протоколу Open Graph:</td>
    @if (Model.ContainsOpenGraph)
    {
        <td>Протокол Open Graph є</td>
    }
    else
    {
        <td>Протокол Open Graph немає</td>
    }
</tr>
```

```
                <tr>
                    <td>Наявність іконки сайту:</td>
                    @if (Model.ContainsFavicon)
                    {
                        <td>Іконка сайту є</td>
                    }
                    else
                    {
                        <td>Іконки сайту немає</td>
                    }
                </tr>
                <tr>
                    <td>Наявність кодування:</td>
                    @if (Model.ContainsCharset)
                    {
                        <td>Кодування є</td>
                    }
                    else
                    {
                        <td>Кодування немає</td>
                    }
                </tr>
                <tr>
                    <td>Наявність Viewport:</td>
                    @if (Model.ContainsViewport)
                    {
                        <td>Viewport є</td>
                    }
                    else
                    {
                        <td>Viewport немає</td>
                    }
                </tr>
            </tbody>
        </table>
    </div>
</div>
}
```