МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-06

Дипломний проект

здобувача освіти денної форми навчання
РП.06.23.000.ДП

ЦВЯТКОВ
СЕРГІЙ
ДМИТРОВИЧ

м. Одеса
2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

Спеціальність: **121 «Інженерія програмного забезпечення»**

Освітня програма: «**Розробка програмного забезпечення»**

Група: **4РП-06**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) на тему:

Модернізація програми перевірки унікальності текстів на веб-сайті

Проектний матеріал складається з пояснювальної записки на 65 сторінках та графічного (презентаційного) матеріалу на 10 аркушах (слайдах).

Дипломник _____ (Цвятков С.Д.)

Керівник _____ (Шувалова І.О.)

Консультанти:

з економічної частини _____ (Копайгородська Т.Г.)

з охорони праці _____ (Чорновол Н.І.)

з дотримання вимог ЄСКД _____ (Петрашова В.І.)

старший консультант _____ (Кунуп Т.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист « 26 » 06 2023 р. Протокол ДКК № 3

Оцінка ДКК 4 (добре)

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНАХТ»

Відділення **комп'ютерних систем** Комісія **КТ та ПІ**

Спеціальність **121 «Інженерія програмного забезпечення»**

Освітня програма **«Розробка програмного забезпечення»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР _____

"_____" _____ 2023 р.

ЗАВДАННЯ

на дипломний проект (роботу)

Здобувачеві (здобувачці) освіти Цвяткова Сергія Дмитровича

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Модернізація програми перевірки унікальності текстів на веб-сайті

затверджена наказом по коледжу від "17" жовтня 2022 р. № 235-А2-ОД

2. Термін здачі закінченого проекту (роботи) _____

3. Вихідні данні до проекту (роботи) Microsoft Visual Studio, .Net, C#, MS SQL Server, JavaScript LINQ, API, HTTP, Angular, CSS, Microsoft Edge, Postman, MySQL, СУБД

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

1. Технологічний розділ1.1. Аналітичний огляд технологій побудови веб-орієнтованих систем. 1.2 Об'єкт та предмет дослідження. Методологія дослідження. 1.1.3 Принципи роботи існуючих програм перевірки унікальності. 1.1.4 Аналіз проблем та обмежень існуючих систем. 1.1.5. Визначення ключових вимог до системи. 1.2. Розробка системи для перевірки тексту на унікальність 2. Економічний розрахунок. 3. Охорона праці.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Презентація (10 слайдів): 1. Вступний слайд. 2. Мета проекту. 3. Об'єкт та предмет дослідження. 4. Що таке унікальність тексту. 5. Структура веб-додатку. 6. Серверна частина. 7. Клієнтська частина. 8-10. Веб-додаток

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний	Шувалова І.О.		
Економічний	Копайгородська Т.Г.		
Охорона праці	Чорновол Н.І.		
Нормоконтроль	Петрашова В.І.		
Старший консультант	Кунуп Т.В.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту (роботи)	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	Розділ 1. Технологічний розділ	22.05.23	
2	Розділ 2. Економічний розрахунок	22.05.23	
3	Розділ 3. Охорона праці	22.05.23	
4	Розробка презентації до дипломної роботи	24.05.23	
5	Чистове оформлення пояснювальної записки	24.0523	
6	Підготовка доповіді до захисту	27.05.23	
7	Отримання рецензії, відповіді на зауваження	05.06.23	
	Рецензента		
8	Захист роботи	09.06.23	

Дипломник _____
(підпис)

Керівник _____
(підпис

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 06.23.000 ДП ПЗ	*Дипломний проект*		
А4			*РП 06.23.000 ДП ПЗ*	*Пояснювальна*	*1*	
				записка		

Ізм.	Арк.	№ докум.	Підпис	Дата	*РП 06.23.000 ДП*		
Розробив		Цвятков С.Д.			Модернізація програми перевірки унікальності текстів на веб-сайті	Літ.	Аркуш 3 / Аркушів 65
Перевірив		Шувалова І.О..				Н Д П	
Н. Контр.		Петрашова В.І				ВСП «ОТФК ОНТУ» гр.РП-06	
Затверд.		Кунуп Т.В.					

ЗМІСТ

ВСТУП

У сучасному інформаційному суспільстві, де дані стають одним з найважливіших ресурсів, значення унікальності тексту набуває особливої ваги. Це не лише питання етики і культури спілкування, але й фундаментальний аспект, що визначає цінність інформації. Дипломна робота присвячена аналізу причин, чому перевірка унікальності тексту так важлива в глобальному світі та на прикладі України.

У наш час інформація є легкодоступною, а різноманітність джерел інформації ще більше підсилює цей ефект. Однак така доступність може сприяти поширенню плагіату, який підриває процес створення нового і унікального контенту. Перевірка унікальності тексту є одним з ефективних засобів боротьби з цим явищем, забезпечуючи визнання інтелектуальної праці, захист авторських прав та розкриття потенціалу оригінальних ідей.

В Україні, як і в багатьох інших країнах світу, актуальність цього питання висока. Перш за все, це стосується сфери освіти та наукових досліджень, де унікальність та оригінальність текстів є ключовими. Освітні заклади, вчені, науковці, студенти - всі вони зіштовхуються з необхідністю створювати та представляти унікальний контент.

Окрім цього, унікальність тексту має велике значення в сфері цифрового маркетингу. Завдяки алгоритмам пошукових систем, унікальний контент має більш високий рейтинг, що сприяє видимості брендів та продуктів в інтернеті. Таким чином, унікальність тексту стає ключовим фактором успіху в сфері інтернет-маркетингу.

Метою даної дипломної роботи є розробка та впровадження модернізації програми перевірки унікальності текстів на веб-сайті з метою покращення її ефективності, точності та швидкості роботи.

Для досягнення мети потрібно зробити наступні кроки:

1. Визначити проблему, та знайти існуючи рішення
2. Зрозуміти поняття та значення унікальності тексту

3. Мати визначення ключових вимог до системи перевірки унікальності текстів

4. Побудувати основних функцій та компонентів системи

5. Розробити архітектуру та дизайн системи

6. Зробити тестування функціональності та продуктивності системи

7. Впровадження модернізованої системи на веб-сайті помилок.

1 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

1.1 Аналітичний огляд побудування системи аналізу текста

1.1.1 Об'єкт та предмет дослідження. Методологія дослідження

Об'єктом нашого дослідження є процес перевірки унікальності текстів в цифровому середовищі, а точніше - програми для перевірки унікальності текстів на веб-сайтах. Ці програми використовуються в різних сферах - від освіти до бізнесу - для гарантування авторства, виявлення плагіату та підтримки якості контенту.

Предметом нашого дослідження є модернізація програми перевірки унікальності текстів. Ми зосередимося на вивченні методів і технік модернізації, що можуть підвищити ефективність, точність та швидкість роботи цих програм. Наша мета - зробити ці системи більш надійними, доступними та корисними для користувачів.

Для досягнення цієї мети ми використаємо різні методи дослідження. Наша методологія базується на комбінації кількох підходів, що включають теоретичний аналіз, емпіричне дослідження та практичну розробку.

Ми почнемо з вивчення теоретичних аспектів унікальності тексту та систем перевірки унікальності. Ми дослідимо літературу з цієї області, включаючи наукові статті, книги та онлайн-ресурси, щоб зрозуміти основні концепції, принципи та методи перевірки унікальності текстів. Також ми проаналізуємо існуючі програми та їх характеристики, а також визначимо проблеми та обмеження, які вони можуть мати.

Ми проведемо аналіз потреб користувачів, щоб зрозуміти, які вимоги вони пред'являють до програм перевірки унікальності текстів. Для цього ми використаємо методи, такі як опитування, інтерв'ю та аналіз відгуків користувачів. Ці дані допоможуть нам краще зрозуміти потреби користувачів та врахувати їх при плануванні модернізації.

На основі нашого теоретичного аналізу та емпіричного дослідження ми розробимо концепцію модернізованої системи перевірки унікальності текстів.

Ми визначимо ключові функції та компоненти системи, а також розробимо її архітектуру та дизайн. Після цього ми перейдемо до програмування та тестування системи, використовуючи сучасні технології та методики розробки програмного забезпечення.

Після розробки системи ми проведемо її тестування, щоб переконатися в її ефективності, надійності та відповідності вимогам користувачів. Ми внедремо систему на веб-сайт і зібраємо обратний зв'язок від користувачів, щоб зрозуміти, як ми можемо ще покращити її.

Методологія є ітераційною та гнучкою, що дозволяє нам адаптуватися до нових виявлених потреб та проблем, а також використовувати нові знання та технології, що з'являються в ході нашого дослідження.

1.1.2 Поняття та значення унікальності тексту

Унікальність тексту — його оригінальність, винятковість, унікальність. Такий текст немає аналогів, не копіює інші твори і є авторським твором. Унікальність тексту важлива у багатьох сферах, від науки та освіти до бізнесу та журналістики.

У науковій та освітній сферах унікальність тексту гарантує авторство ідей, висновків та результатів дослідження. Наукові журнали та конференції вимагають унікальних статей, щоб гарантувати, що кожна публікація є новим вкладом у знання. В освіті унікальність тексту підтверджує, що студенти самостійно освоїли матеріал та можуть застосовувати знання на практиці.

У бізнесі та журналістики унікальність тексту також має велике значення. Унікальні тексти привертають увагу аудиторії, покращують репутацію компанії чи видання та сприяють лояльності клієнтів чи читачів. Крім того, в контексті цифрового маркетингу унікальність тексту впливає на ранжування веб-сайтів у пошукових системах, що може впливати на видимість бренду в інтернеті.

Унікальність тексту залежить від кількох факторів. По-перше, це оригінальність ідей та думок, викладених у тексті. По-друге, це унікальність формулювань та стилю письма. По-третє, це унікальність структури та

організації тексту. Всі ці аспекти разом визначають унікальність тексту та роблять його цінним для читачів та корисним для авторів та видавців.

Однак, унікальність тексту може бути під загрозою через таке явище, як плагіат. Плагіат — використання чужих ідей чи текстів без належного визнання авторства. Це серйозна проблема в науці та освіті, і вона може підірвати довіру до наукових публікацій та освітніх закладів. У бізнесі та журналістики плагіат може пошкодити репутацію та викликати юридичні проблеми.

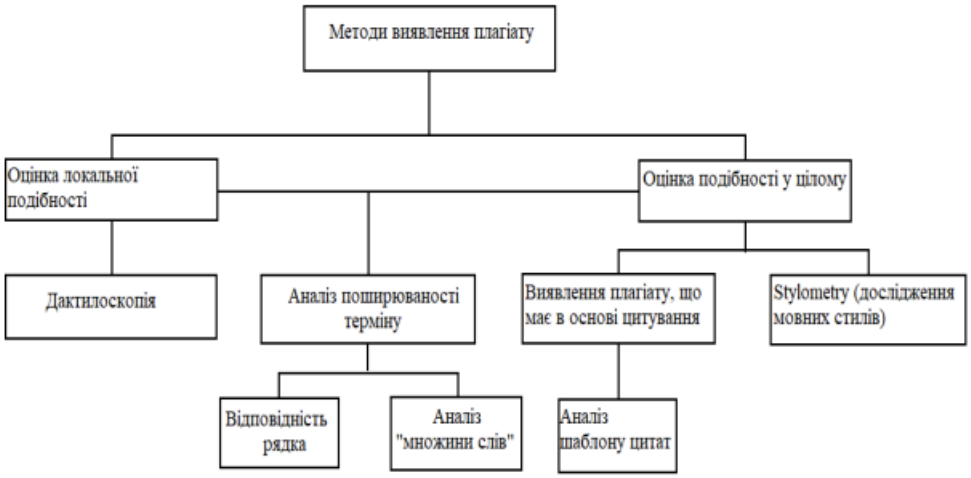


Рисунок 1.1 - Схема класифікації методів комп’ютерного виявлення плагіату[1]

Тому перевірка унікальності тексту стає дедалі важливішою. Сучасні технології та алгоритми дозволяють автоматично порівнювати тексти з великими базами даних та визначати рівень їх унікальності. Це допомагає запобігти плагіату, підтвердити авторство та покращити якість текстів. Насамкінець, унікальність тексту є ключовим чинником у сучасному інформаційному суспільстві. Вона підтверджує авторство, гарантує оригінальність ідей та підтримує довіру до джерел інформації. Перевірка унікальності тексту є важливим інструментом підтримки цих цінностей і забезпечення якості текстів у різних сферах.

Ніжче преведена таблиця прівнянь переваг та недоліків перевірки унікальності тексту:

Таблиця 1.1 Перівняння переваг та недоліків перевірки унікальності тексту

Методи	Переваги	Недоліки
Пошук прямих збігів	Простий у використанні та розумінні; ефективний проти явного плагіату.	Може не виявляти складніші форми плагіату, такі як перефразування або переклад.
Статистична аналітика	Може виявляти перефразування, ґрунтуючись на аналізі частоти слів та структури речень.	Може бути складнішим у використанні та розумінні; може давати помилкові спрацьовування.
Семантичний аналіз	Може виявляти складні форми плагіату, включаючи переклад, і навіть перефразування, виходячи з аналізі сенсу тексту.	Вимагає складних алгоритмів та великих обчислювальних ресурсів; може бути складним розуміння.

1.1.3 Принципи роботи існуючих програм перевірки унікальності

Для автоматичного виявлення плагіату та забезпечення оригінальності контенту використовуються системи тестування на унікальність текстів. Їх основна функція полягає в аналізі вхідного тексту та порівнянні його з великим обсягом іншого тексту для виявлення збігів.

Однак принцип роботи цих систем може значно відрізнятися залежно від використовуваного алгоритму, бази даних для порівняння та інших факторів:

1. Основою більшості програм перевірки унікальності тексту є алгоритми, які здійснюють пошук і порівняння фрагментів тексту. Існує кілька типів алгоритмів, які можна використовувати для цієї мети.

2. Алгоритми перетворюють фрагменти тексту в унікальні числові значення або хеші. За допомогою хешування програма може швидко порівняти введений текст із великою кількістю іншого тексту, що зберігається в базі даних.

3. Алгоритми перетворюють текст на вектори в багатовимірному просторі, де кожне слово чи фраза представляє один вимір. Таким чином програма може порівнювати вхідний текст з іншими текстами, визначаючи ступінь подібності між векторами.

4. Програми перевірки унікальності тексту повинні мати доступ до великої бази даних текстів, щоб виконувати порівняння. Ці бази даних можуть включати наукові статті, книги, веб-сторінки, студентські роботи та інші види текстів.

5. Після порівняння введеного тексту з базою даних програма формує звіт із зазначенням ступеня унікальності тексту. Цей звіт може містити загальний показник унікальності, а також деталі конкретних збігів з іншими текстами.

Важливо відзначити, що існуючі програми перевірки унікальності можуть мати певні обмеження. Вони можуть не виявити деякі типи плагіату,

наприклад перефразування або переклад тексту з іншої мови. Крім того, вони можуть давати неправильні результати через алгоритмічні помилки або недоліки бази даних. Тому однією з головних цілей модернізації цих програм може бути підвищення їх точності та надійності.

1.1.4 Аналіз проблем та обмежень існуючих систем

Незважаючи на те, що технологія перевірки унікальності просунулася далеко вперед за останні роки, все ще існує ряд проблем і обмежень, пов’язаних з існуючими системами, які можуть вплинути на їхню ефективність і надійність.

Можливість виявлення складного плагіату. Деякі програми можуть мати труднощі з виявленням складних форм плагіату, таких як перефразування, синтаксичний плагіат (зміна порядку слів без зміни значення) або плагіат перекладу (переклад тексту з іноземної мови на іншу). Це пояснюється тим, що більшість сучасних систем покладаються на визначення точних або майже точних збігів, тоді як ці форми плагіату передбачають глибшу трансформацію тексту.

Обмежений доступ до баз даних. Деякі програми перевірки унікальності тексту можуть мати доступ лише до обмеженої кількості джерел тексту. Це може обмежити їх здатність виявляти плагіат, особливо якщо оригінальний текст походить із джерела, яке не включено до бази даних програми.

Помилкові спрацьовування. Іноді програми перевірки унікальності тексту можуть генерувати хибні спрацьовування, неправильно ідентифікуючи фрагменти тексту як плагіат, хоча насправді це не так. Це може статися, наприклад, коли текст містить законні цитати або загальні фрази, які присутні в багатьох інших джерелах.

Складність і зручність використання. Деякі програми перевірки унікальності тексту можуть бути складними у використанні, з неінтуїтивно зрозумілими інтерфейсами користувача та процесами перевірки, які потребують часу та технічних навичок. Це може перешкодити деяким людям

використовувати їх, таким чином обмежуючи їхню ефективність у виявленні та запобіганні плагіату.

Повага до конфіденційності. Ще одна проблема – дотримання конфіденційності користувачів. Деякі програми зберігають тексти, подані на перевірку, у своїх базах даних, які потім можна використовувати для подальших порівнянь. Це викликає проблеми з конфіденційністю та захистом даних, особливо якщо тексти містять чутливу або конфіденційну інформацію.

Рисунок 1.2. Види плагіату[2]

Ці проблеми та обмеження підкреслюють необхідність розробки нових інструментів і підходів для перевірки унікальності текстів, які є більш ефективними, надійними, простими у використанні та поважають конфіденційність користувачів.

1.1.5 Визначення ключових вимог до системи перевірки унікальності текстів

Розробка ефективної системи перевірки унікальності вимагає чіткого розуміння ключових вимог, яким має відповідати така система. Ці вимоги обумовлені як потребами користувачів, так і проблемами, пов'язаними з виявленням плагіату. Ось деякі основні вимоги до системи перевірки

унікальності тексту. По-перше, система перевірки унікальності тексту має бути ефективною для виявлення плагіату. Він повинен мати можливість не тільки виявляти прямий плагіат, тобто дослівне копіювання, але й виявляти складніші форми плагіату, такі як парафразний плагіат і плагіат перекладу. Щоб досягти такого рівня ефективності, системі знадобиться використовувати передові алгоритми та мати велику базу даних текстів для порівняння.

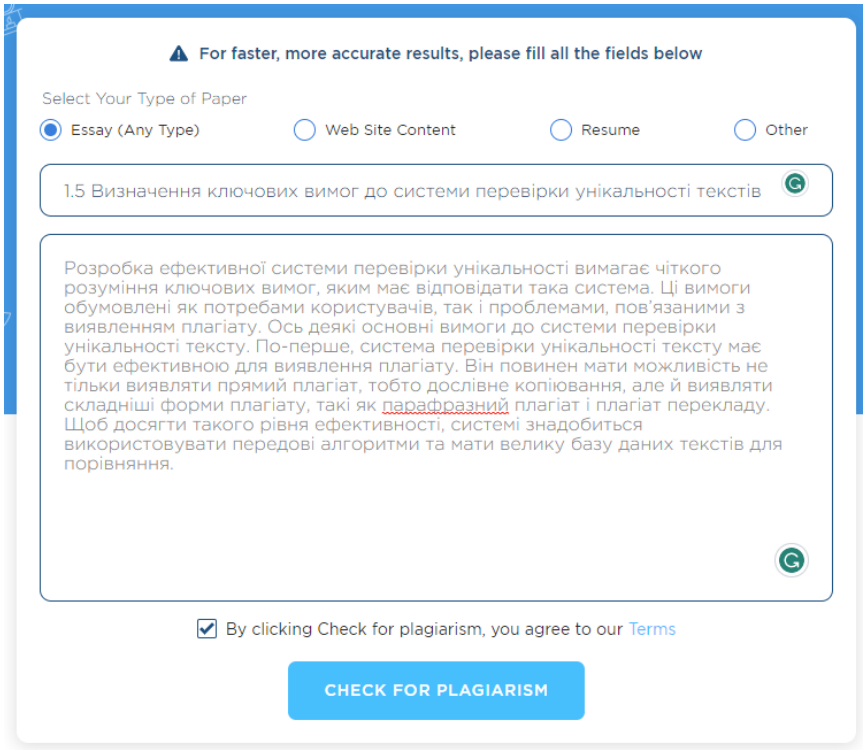


Рисунок 1.3. Приклад платформи для перевірки тексту на унікальність

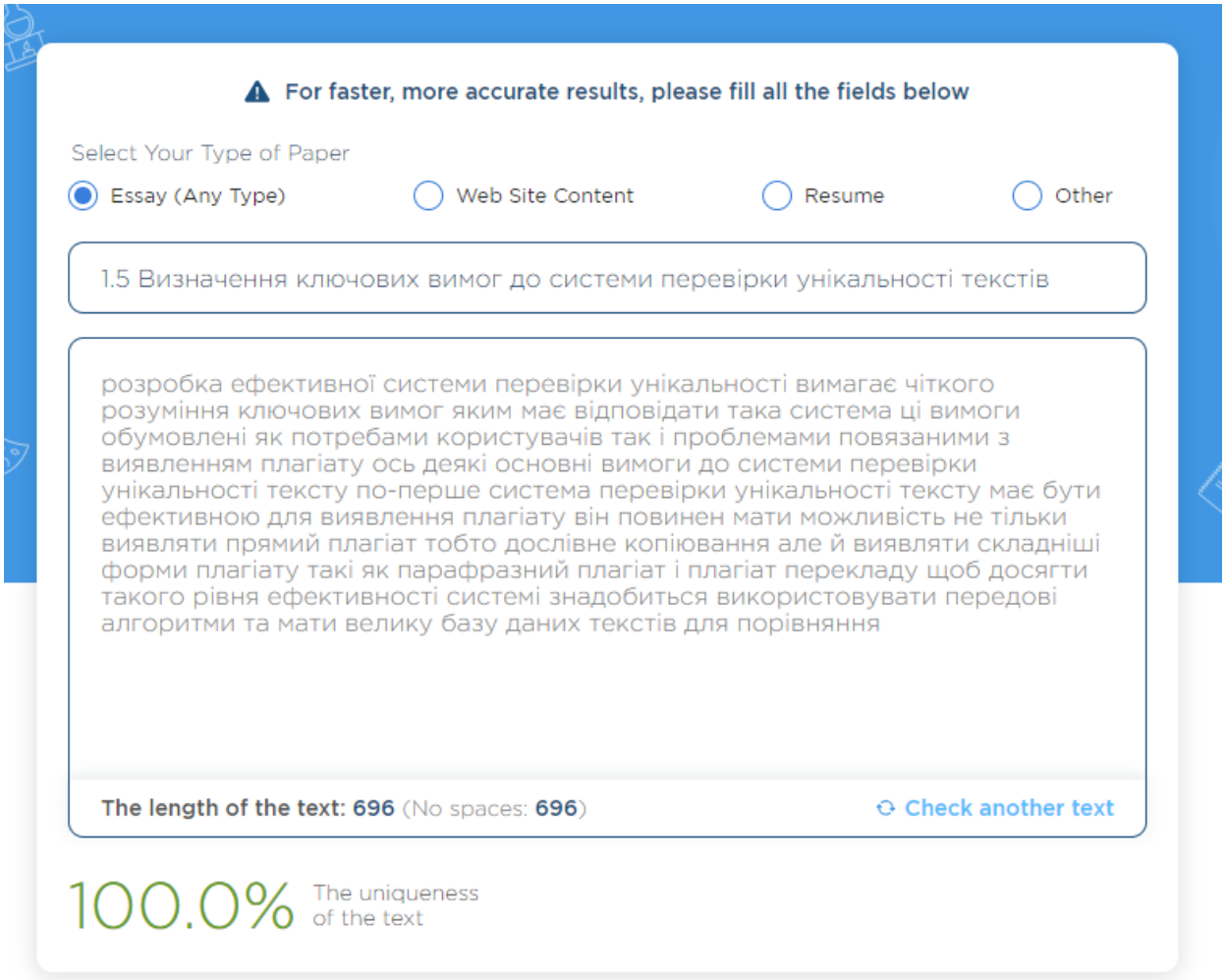


Рисунок 1.4. Приклад результату перевірки

Ще одним важливим аспектом є швидкість і масштабованість. Вкрай важливо, щоб система могла обробляти й аналізувати великі обсяги тексту за розумний проміжок часу. Це особливо важливо для організацій, яким необхідно перевірити велику кількість документів, наприклад університетів чи видавництв. Точність і надійність однаково важливі. Система перевірки унікальності тексту повинна забезпечувати точні та надійні результати, зводячи до мінімуму як помилкові спрацьовування, тобто помилкову ідентифікацію оригінальних текстів як плагіату, так і помилкові негативи, тобто нездатність ідентифікувати плагіат. З точки зору зручності використання, система повинна бути розроблена таким чином, щоб її легко могли використовувати користувачі з різним рівнем технічної підготовки. Інтуїтивно зрозумілий інтерфейс користувача та простий і зрозумілий процес оформлення замовлення можуть значно покращити взаємодію з користувачем.

Повага до конфіденційності користувачів є ще однією фундаментальною вимогою. Система не повинна зберігати надіслані тексти для перевірки без згоди користувача та повинна бути оснащена відповідними засобами захисту даних. Крім того, налаштування має бути ключовою особливістю системи. Користувачам може знадобитися налаштувати параметри відповідно до їхніх конкретних потреб, наприклад вибрати джерела тексту для порівняння або встановити рівень подібності, який визначає плагіат. Нарешті, система повинна бути розроблена з урахуванням сумісності, що дозволяє її інтеграцію з іншими програмами та платформами, якими користуються користувачі, такими як системи управління навчанням або програми обробки текстів.

Ці вимоги надають загальний огляд характеристик, якими повинна володіти система перевірки унікальності текстів, щоб бути ефективною. Однак важливо пам’ятати, що вони можуть відрізнятися залежно від конкретного контексту використання та потреб користувача.

Теоретичний розділ цього дослідження надав детальне уявлення про різні аспекти, пов’язані з перевіркою унікальності текстів. Він досліджував важливість унікальності тексту в різних галузях, зокрема в академічних колах, видавничій справі та цифровому маркетингу, наголошуючи на тому, що наявність унікального та оригінального контенту має вирішальне значення для збереження інтелектуальної цілісності, сприяння творчості та покращення видимості в Інтернеті.

Аналіз проблем і обмежень існуючих систем перевірки унікальності текстів висвітлив низку проблем, які необхідно вирішити для підвищення ефективності та надійності таких систем. До них належать труднощі з виявленням складних форм плагіату, обмежений доступ до текстових баз даних, генерація помилкових спрацьовувань, складність зручності використання та проблеми конфіденційності користувачів. На основі цих міркувань визначено основні вимоги до системи перевірки унікальності текстів. Ідеальна система повинна мати можливість ефективно ідентифікувати

різні форми плагіату, обробляти великі обсяги текстів за розумний час, забезпечувати точні та надійні результати, бути простою у використанні, поважати конфіденційність користувачів, пропонувати можливості налаштування та інтегруватися з іншими програмами та платформами.

Розуміння цих вимог має вирішальне значення для проектування та розробки системи перевірки унікальності тексту, яка є дійсно ефективною та відповідає потребам користувача. Інформація, зібрана в цьому теоретичному розділі, може бути використана як основа для розробки нової системи, яка буде досліджена в практичному розділі цього дослідження.

Підсумовуючи, перевірка унікальності текстів є вирішальним процесом у багатьох сферах, і його наслідки виходять далеко за межі простого запобігання плагіату. Для того, щоб повністю використати його потенціал і подолати обмеження поточних систем, важливо мати правильні інструменти та методи. Це теоретичне дослідження забезпечило міцну основу для розуміння існуючих проблем і визначення необхідних вимог до вдосконаленої та ефективної системи перевірки унікальності.

1.2 Розробка системи для перевірки тексту на унікальність

1.2.1 Формулювання вимог до нової системи

Перед тим, як почати будь-який процес розробки, важливо визначити і сформулювати вимоги до майбутньої системи. Вимоги поділяються на функціональні та нефункціональні. Функціональні вимоги визначають, що система повинна робити, в той час як нефункціональні вимоги визначають, як система повинна працювати (табл. 2.1)

Таблиця 2.1

Вимоги до системи

Тип вимоги	Назва вимоги	Опис
Функціональна	Перевірка унікальності тексту	Система повинна перевіряти текст на унікальність, порівнюючи його з базою даних.
Функціональна	Повідомлення про результати	Після перевірки система надає детальний звіт про унікальність тексту.
Функціональна	Інтеграція із веб-сайтом	Система повинна бути інтегрована на веб-сайт, зручна для користувача.
Нефункціональна	Продуктивність	Система повинна швидко обробляти запити, незалежно від їх об'єму.
Нефункціональна	Надійність	Система повинна забезпечувати точні результати перевірки.
Нефункціональна	Безпека	Система повинна забезпечувати безпеку перевіряємих даних.

Тип вимоги	Назва вимоги	Опис
Нефункціональна	Сумісність	Система повинна бути сумісною з різними браузерами та пристроями.
Нефункціональна	Масштабованість	Система повинна бути готова до масштабування і витримувати збільшення кількості користувачів та запитів.

Функціональні вимоги.

– Система перевірки унікальності текстів повинна включати такі функціональні вимоги:

– Перевірка унікальності тексту: Система повинна здати перевіряти введений текст на унікальність. Система повинна порівнювати введений текст із базою даних текстів для визначення унікальності.

– Повідомлення про результати: Після перевірки система повинна надавати детальний звіт про унікальність тексту, включаючи відсоток унікальності та джерела співпадінь.

– Інтеграція із веб-сайтом: Система повинна бути інтегрована на веб-сайт таким чином, щоб користувачі могли легко вставити текст для перевірки унікальності прямо на сайті.

Нефункціональні вимоги.

Важливо також врахувати наступні нефункціональні вимоги:

– Продуктивність: Система повинна швидко обробляти запити на перевірку унікальності тексту, незалежно від їх об'єму.

– Надійність: Система повинна бути надійною і забезпечувати точні результати. Помилкові повідомлення про співпадіння можуть призвести до неправильних результатів і втрати довіри користувачів.

– Безпека: Система повинна забезпечувати безпеку перевіряємих даних. Тексти, які користувачі вводять для перевірки, можуть містити конфіденційну інформацію, тому важливо забезпечити їх захист від несанкціонованого доступу.

– Сумісність: Система повинна бути сумісною з різними браузерами та пристроями, щоб користувачі могли використовувати її незалежно від свого обраного браузера або пристрою.

– Масштабованість: Система повинна бути готова до масштабування. Вона повинна бути спроектована так, щоб витримувати зростання кількості користувачів та запитів, не втрачаючи при цьому продуктивності.

– Обмеження. Також потрібно врахувати обмеження для системи. Це можуть бути технологічні обмеження, бюджет, час на розробку та впровадження, і так далі.

Ці вимоги мають бути сформульовані у відповідності до потреб користувачів та бізнес-вимог веб-сайту. Важливо, щоб вони були ясними, конкретними і вимірюваними, щоб у процесі розробки могли служити чітким керівництвом.

Отже, виходячи з цих вимог, ми можемо перейти до наступного кроку - проектування архітектури системи.

1.2.2 Проектування архітектури системи

При проектуванні архітектури системи, важливо враховувати не тільки її функціональні вимоги, але й нефункціональні вимоги, такі як продуктивність, безпека, сумісність і масштабованість.

Вибір технологій. Під час вибору технологій для розробки системи варто розглянути використання ASP.NET в поєднанні з Angular. ASP.NET - це

високопродуктивний фреймворк для створення веб-додатків, що розроблено Microsoft. Він включає в себе набір бібліотек, що дозволяють легко створювати надійні та безпечні веб-додатки. Щодо бази даних, можна використати SQL Server, також випущений Microsoft, який відомий своєю надійністю та продуктивністю.

Angular – це фреймворк для створення клієнтської частини веб-додатків. Він включає в себе набір інструментів, які дозволяють розробникам легко створювати інтерактивні, високопродуктивні та масштабовані веб-інтерфейси.

Алгоритми перевірки унікальності. Основою системи перевірки унікальності текстів є алгоритми, які виконують фактичну перевірку. Один з підходів - використання алгоритмів гешування, таких як MD5 або SHA, для створення унікальних гешів текстових блоків. Ці геші потім порівнюються з базою даних гешів для виявлення співпадінь.

Також можна розглянути використання більш складних алгоритмів, що базуються на машинному навчанні. Ці алгоритми можуть виявляти плагіат, навіть якщо текст був перефразований, використовуючи семантичний аналіз та інші техніки машинного навчання.

Серверна та клієнтська частина. На серверній частині, що базується на ASP.NET, буде виконуватися основна робота по обробці запитів від користувачів, перевірки унікальності текстів і зберігання результатів. Серверна частина повинна бути високопродуктивною і надійною, щоб витримувати велику кількість запитів і забезпечувати безпеку даних.

Клієнтська частина, що будується на Angular, відповідатиме за інтерфейс користувача. Вона повинна бути зручною для користувачів, забезпечувати швидкий доступ до функцій перевірки унікальності і відображати результати в зрозумілому форматі.

Таким чином, використання комбінації ASP.NET і Angular дозволить створити потужну, продуктивну і зручну в користуванні систему перевірки унікальності текстів.

Структура бази даних. Для зберігання результатів перевірки унікальності текстів нам потрібна база даних. У нашому випадку ми використовуємо SQL Server. Потрібно створити структуру бази даних, яка максимально відповідає потребам системи.

Можна запропонувати таку схему бази даних (рис. 2.1) :

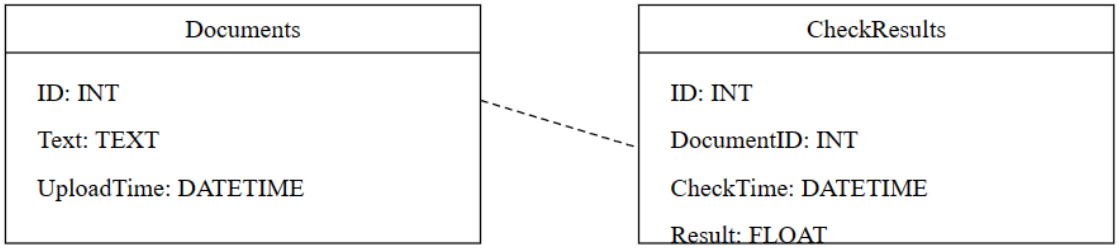


Рисунок 1.5. Схема БД для перевірки текстів

– Таблиця "Documents" - зберігає інформацію про документи, які були завантажені для перевірки. Поля: ID, Text (текст документа), UploadTime (час завантаження).

– Таблиця "CheckResults" - зберігає результати перевірки унікальності. Поля: ID, DocumentID (ID документа, що відповідає полю ID в таблиці Documents), CheckTime (час перевірки), Result (результат перевірки).

Серверні компоненти

На основі ASP.NET створюється серверний бекенд. Він відповідатиме за обробку запитів від клієнтської частини, виконання перевірки унікальності текстів і зберігання результатів в базі даних. Важливими компонентами серверної частини будуть контролери, що відповідають за обробку запитів, та сервіси, що виконують логіку перевірки унікальності.

Клієнтські компоненти

Клієнтська частина, створена на основі Angular, включатиме ряд компонентів. Головні з них:

– Компонент завантаження документів - дозволяє користувачам завантажувати документи для перевірки.

– Компонент перегляду результатів - відображає результати перевірки унікальності.

– Сервіси для взаємодії з сервером - відповідають за відправку запитів на сервер і отримання відповідей.

Таким чином, ми проектуємо систему, що складається з бази даних, серверних та клієнтських компонентів. Всі ці елементи взаємодіють між собою, щоб забезпечити ефективну та надійну роботу системи перевірки унікальності текстів.

Кожен компонент системи розробляється з урахуванням вимог до продуктивності, масштабованості та безпеки. База даних проектується таким чином, щоб оптимізувати зберігання та пошук даних. Серверна частина розробляється з урахуванням принципів SOLID для забезпечення чистого коду та легкої підтримки. Клієнтська частина розробляється з орієнтацією на зручність та інтуїтивність для кінцевого користувача.

У цьому контексті, використання ASP.NET для серверної частини та Angular для клієнтської частини дозволяє реалізувати всі вищезгадані вимоги, а також забезпечити високу продуктивність, безпеку та надійність системи перевірки унікальності текстів.

1.2.3 Написання коду

Після визначення вимог та проектування архітектури системи і структури бази даних, наступним етапом є розробка системи. Наша система буде розроблена на основі ASP.NET для бекенду і Angular для фронтенду.

Для бекенду на ASP.NET, ми створюємо контролери для обробки запитів від клієнтської частини. Контролери мають методи для обробки HTTP запитів GET та POST, які відповідають за завантаження документів і отримання результатів перевірки відповідно. Сервіс перевірки унікальності реалізує алгоритм перевірки, який аналізує текст документа і визначає ступінь його унікальності.

База даних, розроблена на основі SQL Server, включає таблиці Documents і CheckResults, які зберігають інформацію про завантажені

документи і результати їх перевірки відповідно. Розробка бази даних включає створення цих таблиць з визначеними полями, а також налаштування відповідних зв'язків між ними.

Клієнтська частина на Angular розроблена для забезпечення зручного інтерфейсу для користувачів. Основні компоненти фронтенду включають компонент завантаження документів для відправки документів на сервер та компонент перегляду результатів для відображення результатів перевірки унікальності.

Усі ці компоненти разом складають нашу систему перевірки унікальності текстів. Розробка системи включає не тільки створення цих компонентів, але і їх інтеграцію, а також тестування для забезпечення їх коректної роботи разом. Результатом цього етапу є робоча версія системи, яка готова до детального тестування і впровадження.

Під час розробки системи ми також підтримуємо принципи чистого коду та програмування, орієнтованого на об'єкти. Це забезпечує зрозумілість коду, легкість його підтримки та модифікації. Наприклад, ми стараємося дотримуватися принципу єдиного обов'язку (Single Responsibility Principle) і принципу відкритості / закритості (Open/Closed Principle), створюючи класи та методи, що виконують лише одне конкретне завдання, але водночас є готовими до розширення через наслідування та поліморфізм.

Ми також використовуємо систему контролю версій Git для ведення історії змін коду і співпраці між учасниками команди. Це дозволяє нам ефективно управляти змінами і вирішувати будь-які конфлікти, що виникають під час одночасного редагування коду різними учасниками команди.

Також важливим аспектом розробки є оптимізація продуктивності. Ми використовуємо різні методи та інструменти для забезпечення високої продуктивності, включаючи алгоритми і структури даних для оптимальної обробки та зберігання даних, а також технології кешування та асинхронної обробки для забезпечення швидкого відгуку системи.

Всі ці етапи та техніки разом допомагають нам розробити ефективну, надійну і легко підтримувану систему перевірки унікальності текстів.

Для більш конкретного уявлення про структуру коду, давайте розглянемо деякі приклади. Нагадаю, що наш бекенд розроблений на ASP.NET Core.

По-перше, розглянемо моделі, які представляють наші таблиці в базі даних:

```
public class Document
{
    public int ID { get; set; }
    public string Text { get; set; }
    public DateTime UploadTime { get; set; }
    public virtual ICollection<CheckResult> CheckResults { get; set; }
}

public class CheckResult
{
    public int ID { get; set; }
    public int DocumentID { get; set; }
    public Document Document { get; set; }
    public DateTime CheckTime { get; set; }
    public float Result { get; set; }
}
```

Тут Document і CheckResult представляють таблиці Documents і CheckResults в базі даних відповідно. Вони мають властивості, які відповідають полям цих таблиць. Крім того, Document має колекцію CheckResults, яка представляє зв'язок один-до-багатьох між документами та результатами перевірки.

Далі, розглянемо контролер для обробки запитів від клієнтської частини:

```
    [Route("api/[controller]")]
    [ApiController]
    public class DocumentsController : ControllerBase
    {
        private readonly DocumentsContext _context;

        public DocumentsController(DocumentsContext context)
        {
            _context = context;
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<Document>> GetDocument(int id)
        {
            // Get the document with the given ID
        }

        [HttpPost]
        public async Task<ActionResult<Document>> PostDocument(Document
document)
        {
            // Save the new document
        }
    }
```

Цей контролер має два методи: GetDocument, який обробляє GET запити для отримання документа за його ID, і PostDocument, який обробляє POST запити для збереження нового документа.

Це дуже простий приклад і в реальному проекті код буде більш складним і включатиме більше деталей. Але це демонструє основну структуру коду на бекенді в ASP.NET Core.

Після розгляду базового прикладу структури коду на бекенді, давайте перейдемо до важливої частини розробки – тестування. Використовуючи вбудовані засоби ASP.NET Core та фреймворк xUnit, ми можемо створити модульні тести для перевірки правильності нашого коду.

Ось приклад модульного тесту для нашого DocumentsController:

```
    public class DocumentsControllerTests
    {
        private readonly DocumentsController _controller;
```

```
        private readonly Mock<DocumentsContext> _mockContext;

        public DocumentsControllerTests()
        {
            _mockContext = new Mock<DocumentsContext>();
            _controller = new DocumentsController(_mockContext.Object);
        }

        [Fact]
        public async Task GetDocument_ReturnsCorrectDocument()
        {
            // Arrange
            var testDocument = new Document { ID = 1, Text = "Test text" };
            _mockContext.Setup(x =>
x.Documents.Find(1)).Returns(testDocument);

            // Act
            var result = await _controller.GetDocument(1);

            // Assert
            Assert.Equal(testDocument, result.Value);
        }
    }
```

Після розгляду базового прикладу структури коду на бекенді, давайте перейдемо до важливої частини розробки – тестування. Використовуючи вбудовані засоби ASP.NET Core та фреймворк xUnit, ми можемо створити модульні тести для перевірки правильності нашого коду.

Ось приклад модульного тесту для нашого DocumentsController:

```
    public class DocumentsControllerTests
    {
```

```
        private readonly DocumentsController _controller;
        private readonly Mock<DocumentsContext> _mockContext;

        public DocumentsControllerTests()
        {
            _mockContext = new Mock<DocumentsContext>();
            _controller = new DocumentsController(_mockContext.Object);
        }

        [Fact]
        public async Task GetDocument_ReturnsCorrectDocument()
        {
            // Arrange
            var testDocument = new Document { ID = 1, Text = "Test text" };
            _mockContext.Setup(x                                                    =>
x.Documents.Find(1)).Returns(testDocument);

            // Act
            var result = await _controller.GetDocument(1);

            // Assert
            Assert.Equal(testDocument, result.Value);
        }
    }
```

Цей тест перевіряє, чи метод GetDocument контролера повертає вірний документ. Ми використовуємо "макет" (mock) контексту даних, щоб імітувати взаємодію з базою даних.

Важливо мати достатню кількість тестів для переконання в надійності програмного коду. Тести повинні покривати різні сценарії використання і перевіряти коректність реалізації бізнес-логіки.

На останньому етапі розробки ми проводимо інтеграційне тестування, для переконання, що всі частини системи правильно взаємодіють між собою. Наприклад, ми можемо перевірити, що при завантаженні нового документу він коректно зберігається в базі даних, і результати його перевірки на унікальність доступні для перегляду.

Таким чином, за допомогою методів і прийомів розробки, перерахованих вище, ми розробляємо надійну, ефективну і зручну для користувача систему перевірки унікальності текстів.

Структура коду на клієнтській частині в основному залежить від компонентної архітектури, що використовується в Angular. Наприклад, ми можемо мати компонент DocumentUpload, який дозволяє користувачам завантажувати документи, та компонент CheckResult, який відображає результати перевірки.

Ось приклад коду компонента DocumentUpload:

```
import { Component } from '@angular/core';
import { DocumentService } from '../services/document.service';

@Component({
  selector: 'app-document-upload',
  templateUrl: './document-upload.component.html'
})
export class DocumentUploadComponent {
  constructor(private documentService: DocumentService) { }

  onFileSelected(event) {
    const file: File = event.target.files[0];
    if (file) {
      this.documentService.uploadDocument(file).subscribe(
        response => console.log(response),
        error => console.error(error)
      );
    }
  }
}
```

Цей компонент має метод onFileSelected, який викликається, коли користувач вибирає файл для завантаження. Метод використовує сервіс DocumentService для надсилання файлу на сервер.

Наступним кроком є створення компонента CheckResult, який відповідає за відображення результатів перевірки документа на унікальність. Ви можете представити, як ми можемо використовувати HTTP сервіси для отримання цих даних від сервера та відображення їх у шаблоні HTML компонента.

Всі ці компоненти, сервіси та інші елементи взаємодіють між собою для створення цілісного користувацького досвіду, дозволяючи користувачам використовувати нашу систему перевірки унікальності текстів ефективно і зручно.

Важливою є взаємодія із бекендом. У нашому Angular-додатку ми створюємо сервіси, які використовують HttpClient для виконання HTTP-запитів до сервера. Наприклад, DocumentService може виглядати так:

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class DocumentService {
  private readonly apiUrl = 'http://localhost:5000/api/documents';

  constructor(private http: HttpClient) { }

  uploadDocument(file: File): Observable<any> {
    const formData = new FormData();
    formData.append('file', file);

    return this.http.post(this.apiUrl, formData);
  }

  getDocument(id: number): Observable<any> {
    return this.http.get(`${this.apiUrl}/${id}`);
  }

  getCheckResult(id: number): Observable<any> {
    return this.http.get(`${this.apiUrl}/${id}/check-result`);
  }
}
```

Цей сервіс має методи для завантаження документа, отримання документа та отримання результату перевірки документа. Всі вони використовують HttpClient для виконання HTTP-запитів до нашого API.

Разом з Angular Router ми можемо створити маршрутизацію на нашому сайті. Наприклад, ми можемо перейти до сторінки завантаження документа, коли користувач відкриває наш веб-сайт, та перенаправити користувача до сторінки з результатами перевірки після завантаження документа.

Останній аспект, який ми обговоримо, це стилізація. Angular дозволяє використовувати CSS на рівні компонентів, що робить стилізацію більш гнучкою і організованою. Ми можемо створити окремі CSS-файли для кожного компонента і включити їх у декоратор @Component відповідного компонента.

Величезною перевагою Angular є те, що він має вбудовану підтримку двосторонньої прив'язки даних (two-way data binding), що спрощує синхронізацію між моделлю та представленням.

Також важливо враховувати перформанс та безпеку. Щодо перформансу, ми можемо використовувати техніки оптимізації, такі як Lazy Loading для модулів Angular, що дозволяє зменшити час завантаження нашого додатку, завантажуючи лише ті модулі, які потрібні в даному конкретному сценарії.

```
const routes: Routes = [
  { path: 'document-upload', loadChildren: () => import('./document-upload/document-upload.module').then(m => m.DocumentUploadModule) },
  { path: 'check-result', loadChildren: () => import('./check-result/check-result.module').then(m => m.CheckResultModule) },
];
```

З цим налаштуванням маршрутів, модулі DocumentUploadModule і CheckResultModule будуть завантажуватися лише тоді, коли користувач переходить до відповідної сторінки.

Щодо безпеки, Angular має вбудовані механізми для запобігання таким загрозам, як cross-site scripting (XSS). Наприклад, Angular автоматично санітаризує значення, які використовуються в шаблонах компонентів, перед вставкою їх в DOM.

Величезна важливість має також обробка помилок. Ми можемо використовувати глобальний обробник помилок Angular, щоб відловлювати та обробляти неочікувані помилки в нашому коді. Наприклад, ми можемо вивести повідомлення про помилку для користувача або надіслати звіт про помилку на наш сервер для аналізу.

Отже, після розробки клієнтської та серверної частин нашої системи, ми отримуємо повноцінний веб-додаток для перевірки унікальності текстів.

1.2.4 Тестування системи

Тестування є невід'ємною частиною процесу розробки програмного забезпечення. Воно гарантує, що наша система виконує свої функції коректно, ефективно і зручно для кінцевих користувачів. Існує кілька видів тестування, які ми будемо використовувати.

Модульне тестування: Модульне тестування, відоме також як юніт-тестування, фокусується на перевірці окремих частин програмного коду, таких як функції або класи. У нашому випадку, ми можемо створити модульні тести для наших сервісів та контролерів в ASP.NET Core і Angular. Для цього ми можемо використовувати бібліотеки, такі як xUnit для ASP.NET Core та Jasmine з Karma для Angular.

Інтеграційне тестування: Після тестування окремих частин коду, ми переходимо до інтеграційного тестування, яке перевіряє, як різні частини системи взаємодіють між собою. Наприклад, ми можемо створити інтеграційні тести, які викликають наші API кінцеві точки та перевіряють правильність відповідей.

Навантажувальне тестування: Цей вид тестування використовується для визначення, як наша система справляється з великим навантаженням,

наприклад, великою кількістю запитів або великим обсягом даних. Для цього ми можемо використовувати інструменти, такі як JMeter або Gatling.

Юзабиліті-тестування: Кінцева мета будь-якої системи - забезпечити хороший досвід користувача. Юзабиліті-тестування дозволяє нам визначити, наскільки зручно і зрозуміло наш веб-сайт для користувачів. Ми можемо проводити юзабиліті-тестування шляхом залучення реальних користувачів або тестувальних груп для оцінки зручності і навігації нашої системи, а також виявлення можливих проблем у взаємодії з користувачами.

Кожен з цих видів тестування має свої особливості і методики, але вони всі необхідні для забезпечення якості та надійності нашої системи перевірки унікальності текстів. Проведення всіх цих видів тестування допоможе виявити можливі проблеми, помилки та вразливості, що дозволить нам вчасно виправити їх і забезпечити високу якість нашої системи.

Загальний підхід до тестування полягає у створенні тестових сценаріїв, виконанні тестів з використанням різних наборів даних та оцінці результатів тестування. За допомогою автоматичного тестування, ми можемо автоматизувати виконання тестів і перевірку результатів, що дозволить нам ефективно проводити тестування великого обсягу і забезпечити швидке виявлення проблем.

Тестування важливо проводити на різних етапах розробки, починаючи з модульного тестування на ранніх етапах, і досягаючи навантажувального і юзабиліті-тестування на пізніших етапах. Це дозволить нам виявити проблеми на ранніх етапах та покращити якість та надійність нашої системи.

Зважаючи на різні види тестування, давайте розглянемо приклади для кожного з них:

Модульне тестування. Наприклад, для нашого сервісу DocumentService, ми можемо написати модульний тест для методу uploadDocument, щоб перевірити, чи відбувається коректна передача файлу на сервер:

```
describe('DocumentService', () => {
  let documentService: DocumentService;
```

```
  let httpClient: HttpClient;
  let httpTestingController: HttpTestingController;

  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [HttpClientTestingModule],
      providers: [DocumentService]
    });

    documentService = TestBed.inject(DocumentService);
    httpClient = TestBed.inject(HttpClient);
    httpTestingController = TestBed.inject(HttpTestingController);
  });

  it('should upload document successfully', () => {
    const file = new File(['Test file'], 'test.txt', { type: 'text/plain' });

    documentService.uploadDocument(file).subscribe(response => {
      expect(response).toBeTruthy();
    });

    const req =
httpTestingController.expectOne('http://localhost:5000/api/documents');
    expect(req.request.method).toEqual('POST');
    req.flush({ success: true });

    httpTestingController.verify();
  });
});
```

У цьому прикладі ми створюємо фейковий файл і викликаємо метод uploadDocument сервісу DocumentService. Після цього ми перевіряємо, чи повертається успішна відповідь від сервера.

Інтеграційне тестування. Для перевірки взаємодії між клієнтською та серверною частинами, ми можемо створити інтеграційні тести. Наприклад, ми можемо перевірити, чи правильно отримується документ з сервера після його завантаження:

```
describe('DocumentsController', () => {
  let controller: DocumentsController;
  let httpTestingController: HttpTestingController;

  beforeEach(() => {
```

```
      TestBed.configureTestingModule({
        imports: [HttpClientTestingModule],
        providers: [DocumentsController]
      });

      controller = TestBed.inject(DocumentsController);
      httpTestingController = TestBed.inject(HttpTestingController);
    });

    it('should get document by ID', () => {
      const testDocument = { id: 1, text: 'Test document' };

      controller.GetDocument(1).subscribe(response => {
        expect(response).toEqual(testDocument);
      });

      const req = httpTestingController.expectOne('http://localhost:5000/api/documents/1');
      expect(req.request.method).toEqual('GET');
      req.flush(testDocument);

      httpTestingController.verify();
    });
  });
```

В цьому прикладі ми створюємо інстанс контролера «DocumentsController» і перевіряємо, чи правильно повертається документ з сервера після виклику методу «GetDocument» з використанням HTTP запиту.

Навантажувальне тестування:

Під час навантажувального тестування ми перевіряємо, як система справляється з великим обсягом запитів або навантаженням. Наприклад, ми можемо використовувати JMeter для створення навантажувального тесту, який симулює багато одночасних запитів до нашого API. Ми можемо перевірити, чи відповідає наша система з прийнятною швидкістю та чи зберігає вона стабільність під час навантаження.

Юзабиліті-тестування:

Юзабиліті-тестування спрямоване на оцінку зручності та ефективності використання системи користувачами. Ми можемо проводити юзабиліті-тестування з реальними користувачами або тестувальними групами, які

виконують різні завдання та оцінюють користувацький досвід. Наприклад, ми можемо надати користувачам завдання завантажити документ, переглянути результати перевірки та оцінити легкість використання та інтуїтивність інтерфейсу.

Ці приклади демонструють, як реалізувати різні види тестування в контексті нашої системи перевірки унікальності текстів. Завдяки цим тестам ми можемо впевнитися, що наша система працює належним чином, забезпечує високу якість та надійність, а також забезпечує зручний досвід користувача.

1.2.5 Документація по користуванню системою

В даному розділі буде наведено інструкції щодо використання системи перевірки унікальності текстів. Детальні кроки, необхідні для виконання, наведено нижче:

Заходження на веб-сайт:

– Відкрийте веб-браузер та введіть адресу нашого веб-сайту.

– Здійсніть вхід до системи, використовуючи ваші облікові дані або виконайте реєстрацію, якщо ви новий користувач.

Завантаження тексту:

– Після входу у систему, перейдіть до розділу "Завантажити текст".

– Оберіть текстовий файл, який ви бажаєте перевірити на унікальність.

– Натисніть кнопку "Завантажити" для початку процесу завантаження.

Аналіз тексту:

– Після завантаження тексту, система розпочне процес аналізу та перевірки його унікальності.

– Зачекайте, доки система завершить аналіз та генерування результатів.

Перегляд результатів:

– Після завершення аналізу, ви отримаєте результати перевірки унікальності вашого тексту.

– Ретельно ознайомтеся з виявленими схожими фрагментами тексту та їх джерелами.

Подальші кроки:

– Врахуйте, що виявлення схожих фрагментів не є автоматичним підтвердженням плагіату, але вимагає додаткового дослідження та перевірки.

– Збережіть результати перевірки для подальшого аналізу та посилань на джерела.

Ці інструкції допоможуть вам користуватися нашою системою перевірки унікальності текстів ефективно.

Після успішного входу у систему, користувач опиняється на головній сторінці, де доступний основний функціонал системи. Для початку процесу перевірки унікальності тексту, необхідно перейти до відповідного розділу (рис 2.2).

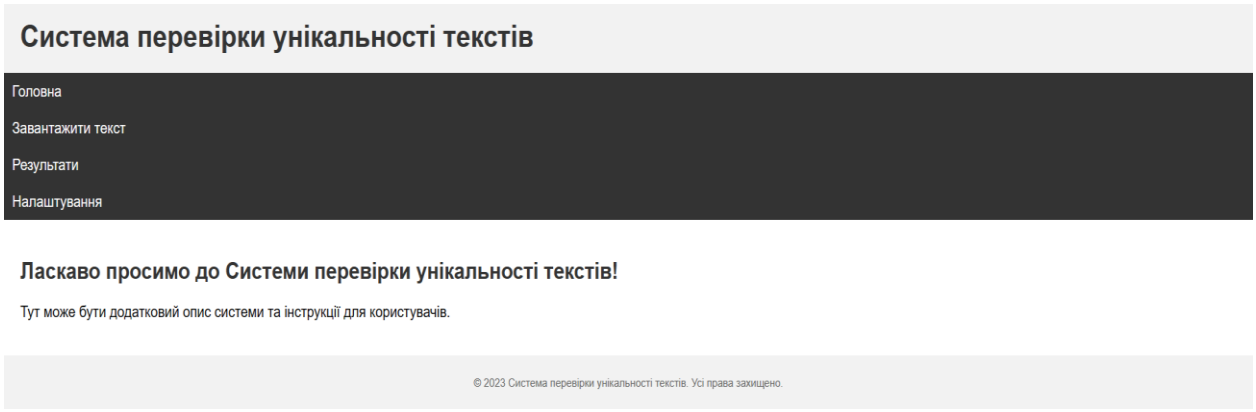


Рисунок 1.6. Головна сторінка системи

В цьому розділі ви знайдете можливість завантажити свій текстовий файл, який потрібно перевірити на унікальність. Для цього, оберіть файл з вашого комп'ютера за допомогою кнопки "Обрати файл" або аналогічного елемента інтерфейсу. Після вибору файлу, натисніть кнопку "Завантажити" для початку процесу завантаження.

Після завантаження файлу, система розпочне аналіз та перевірку унікальності тексту. Цей процес може зайняти певний час, залежно від розміру та складності тексту.

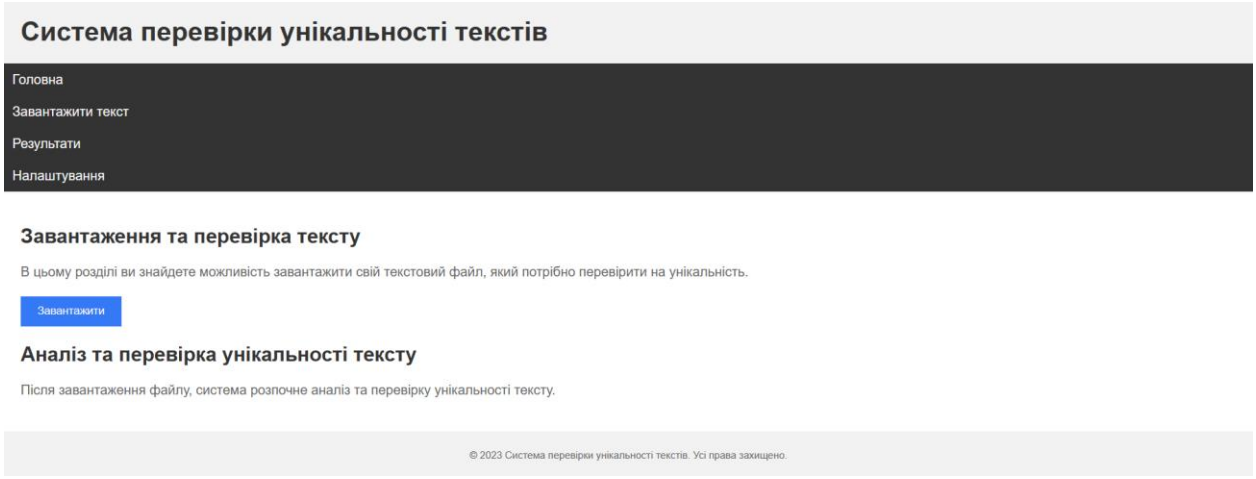


Рисунок 1.7. Сторінка для завантаження тексту

Після завершення аналізу, ви отримаєте результати перевірки. Вони будуть включати інформацію про виявлені схожі фрагменти тексту, їхнє розташування та посилання на можливі джерела.

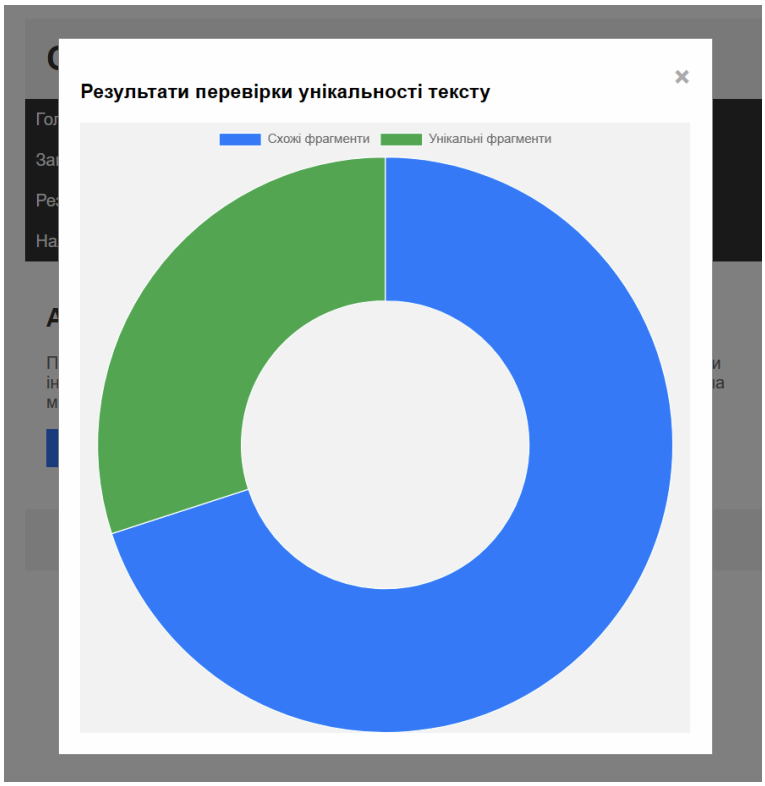


Рисунок 1.8. Результати перевірки

Важливо зазначити, що виявленнясхожих фрагментів не завжди означає наявність плагіату. Вони можуть бути результатом співпадінь чи цитувань.

Тому, після отримання результатів, рекомендується ретельно проаналізувати виявлені фрагменти та додатково перевірити їх, використовуючи інші джерела або інструменти.

Пам'ятайте, що система перевірки унікальності текстів слугує лише інструментом для виявлення можливого плагіату. Оцінка та інтерпретація результатів залишаються відповідальністю користувача.

2 ЕКОНОМІЧНА ЧАСТИНА

2.1 Резюме

Метою даної дипломної роботи є розробка та впровадження модернізації програми перевірки унікальності текстів на веб-сайті з метою покращення її ефективності, точності та швидкості роботи. Для розробки веб-інтерфейсу були використані технології HTML, CSS та JavaScript. Це дозволило створити зручний і інтуїтивно зрозумілий інтерфейс для користувачів. База даних була спроектована для зберігання інформації про завантажені тексти, результати перевірки та інші відомості. Використання бази даних дозволяє зберігати дані ефективно і забезпечувати швидкий доступ до них. Система пройшла різноманітні види тестування, такі як модульне, інтеграційне, нагрузочне та юзабіліті-тестування.

Це дозволило перевірити працездатність, стабільність та коректність роботи системи у різних умовах та навантаженнях. Результати розробки системи свідчать про те, що вона є ефективним інструментом для перевірки унікальності текстів. Користувачі можуть легко завантажити свої тексти та отримати результати перевірки у зручному форматі.

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення. Оцінка якості програмного продукту включає визначення трудомісткості та вартості його створення.

2.2. Визначення трудомісткості розробки програмного забезпечення.

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога.

Таблиця 2.1. Каталог аналогів

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП імітаційного моделювання	7800 – 8800

У таблиці 3.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт.Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар = 229 х 0,7 = 160,3 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5.).

Таблиця 2.3.Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП.

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПО	1,75 – 1,2
Б	ПО – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПО маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПО типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Та*L_1*К_н=160,3 *0,12*0,7= 15,38 \text{ (люд/годин)} \quad (3.1)$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Та*L_2*К_н=160,3 *0,11*0,7 = 14,11 \text{ (люд/годин)} \quad (3.2)$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Та*L_3*К_н*К_т=160,3 *0,61*0,7*0,8= 62,58 \text{ (люд/годин)} \quad (3.3)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 3 (стр), розробка ТП $N_{тп}$= 9 (стр), розробка робочого проекту $N_{рп}$=14 (стр), пояснювальна записка відповідно $N_{пз}$=36 (стр)

Розрахунок зведений у таблицю 3.6.

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1.ТЗ	$T_{Ртз}$=15,38	$T_{кк}$=0,7*$N_{тз}$= 0,7*3=2,1	$T_{нк}$=0,15*$N_{тз}$=0,15*3=0,45
2.Розробка ТП	$T_{Ртп}$=14,11	$T_{кк}$=0,7*$N_{тп}$=0,7*9=6,3	$T_{нк}$=0,15*$N_{тп}$=0,15*9=1.4
3.Розробка РП	$T_{Ррп}$= 62,58	$T_{кк}$=0,7*$N_{рп}$=0,7*14=9,8	$T_{нк}$=0,15*$N_{рп}$=0,15*14=2,1
4.Розробка ПЗ	$T_{пз}$=1,5**$N_{пз}$=1,5*36 =54	$T_{кк}$=0,7*$N_{тз}$=0,7*36=25,2	$T_{нк}$=0,15*$N_{пз}$=0,15*36 =5,4
Усього, в т.ч.:	198,9		
- на розробку	$\Sigma T_р$=146.1		
- контроль керівника		$\Sigma T_{кк}$=43,4	
- нормоконтроль			$\Sigma T_{нк}$=9,4

2.3. Розрахунок ціни програмного продукту.

Для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПП. Розрахунок основної заробітної плати виконавців приведений у таблиці 3.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2023» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2023 року - 6700 гривень; мінімальну погодинну тарифну ставку – 40.46 грн.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців.

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	146,1	40,46	5911,21
2.Контроль керівника	43,4	60,10	2608,34
3.Нормоконтроль	9,4	60,10	564,94
Усього	-	-	ΣЗо= 9084,49

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.8.

Таблиця 2.8. Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	60	3.0	180,0
Разом	-	-	-	$В_{м\,і}$ =180,0
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$ = 0,1*180=18,00
Усього				$В_м = В_{мі} + В_{тр\_з=}$ 198.00

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 23.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	198.00	$В_м$ (див. табл. 3.8)
2. Основна заробітна плата	9084,49	$З_о$ (див. табл. 3.7.)
3.Додаткова заробітна плата	908,49	$Зд = 0{,}1 \times Зо$ = 9084,89*0,1
4.Відрахування до єдиного фонду соціального внеску	2198,46	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22* (9084,49+908,49)
5. Накладні витрати	3633,79	$Внак. = 0{,}4 \times Зо$ =0,4 * 9084,49
6. Повна собівартість	16023,24	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 198.00+9084,49+908,49+2198,46+ 3633,79

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$П=(С_п*Р)/100=(16023,24*10)/100=1602,32$грн (2.4)

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о = С_п + П$ = 16023,24+1602,32= 17625,56 грн; (2.5)

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$Ц_р = Ц_о + ПДВ$ = 17625,56+17625,56 *0.2 = 21150,67 грн; (2.6)

3 ОХОРОНА ПРАЦІ

Охорона праці має вирішальне значення в практичній сфері діяльності, що спрямована на створення безпечних і нешкідливих умов праці. Забезпечення безпеки і запобігання небезпечним умовам праці потребує значних фінансових витрат та впровадження наукових розробок в галузі охорони праці.

При праці програмістів важливо дотримуватися норм безпеки праці, враховуючи законодавчі акти, зокрема "Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями" (НПАОП 0.00-7.15-18) та "Правила охорони праці під час експлуатації електронно-обчислювальних машин" (НПАОП 0.00-1.28-10), що були затверджені Державним комітетом України з промислової безпеки, охорони праці та гірничого нагляду 26 березня 2010 року наказом № 65. Ці нормативні акти містять вимоги щодо створення безпечних умов праці для програмістів та визначають заходи для запобігання можливим ризикам та пошкодженням здоров'я, пов'язаним з роботою з екранними пристроями та електронно-обчислювальними машинами.

1 Вимоги гігієни до робочого середовища

1.1 Вимоги що до умов приміщення

– Площа на одне робоче місце становить не менше ніж 6 квадратних метрів, а об'єм - не менше ніж 20 кубічних метрів.

– Щодо підлоги, покриття повинно бути матовим, а поверхня – рівною, неслизькою і мати антистатичні властивості, щоб запобігти виникненню статичної електрики, яке може спричинити пошкодження обладнання.

– Стосовно кліматичних умов, приміщення має бути обладнане системою опалення для забезпечення комфортної температури, кондиціонуванням повітря для контролю вологості та системою припливно-витяжної вентиляції для забезпечення свіжого повітря.

Вимоги умов приміщення не тільки включає в себе належну організацію приміщення, а ще і робочого місця з метою забезпечення оптимальної робочої позиції та комфорту працівника.

1.2 Вимоги організації робочого місця

– Робоче місце повинно бути організоване таким чином, щоб працівник мав достатньо місця для розміщення обладнання, документів та інших робочих матеріалів. Проведення кабелів та дротів повинно бути організоване таким чином, щоб уникнути перешкод та потенційних небезпек. Також робоче місце повинні відповідати сучасним вимогам ергономіки і забезпечувати належне розташування обладнання (монітора, клавіатури, принтера) і документів на робочій поверхні. Важливо, щоб робочий стіл мав достатньо місця для ніг, з висотою не менше 600 мм, шириною не менше 500 мм, глибиною на рівні колін не менше 450 мм і на рівні простягнутої ноги не менше 650 мм.

– Стілець, на якому сидить працівник, повинен мати належну підтримку для спини та забезпечувати правильне положення тіла. Він повинен мати регульовану висоту, щоб працівник міг встановити оптимальну позицію. Крім того, шаг регулювання елементів стільця повинен бути в межах 15-20 мм для лінійних розмірів й 2-5 градусів для кутових розмірів.

– Комп'ютерний монітор повинен бути розташований належним чином, на відстані та висоті, які забезпечують комфортне зорове сприйняття. Клавіатура та миша повинні бути розташовані так, щоб забезпечити природну позицію рук та уникнути надмірного напруження.

– Робоче місце повинно мати належне освітлення, що забезпечує достатню яскравість та мінімізує відблиски на екрані. Також важливо уникати перебування працівника в темряві або надмірно яскравому освітленні. Також Робочі місця з візуальним екраном (ВДТ) слід розташовувати таким чином, щоб природне світло падало з боку, переважно зліва, що допомагає уникнути надмірного навантаження на очі працівника.

– Робочі столи повинні відповідати сучасним вимогам ергономіки і забезпечувати належне розташування обладнання (монітора, клавіатури, принтера) і документів на робочій поверхні. Важливо, щоб робочий стіл мав достатньо місця для ніг, з висотою не менше 600 мм, шириною не менше 500 мм, глибиною на рівні колін не менше 450 мм і на рівні простягнутої ноги не менше 650 мм.

Дотримання цих організаційних вимог щодо робочого місця для користувачів комп'ютерів та моніторів забезпечить їм комфортні умови праці. Це сприятиме уникненню непотрібного фізичного напруження та покращенню їх продуктивності й здоров'я.

1.3 Освітлення робочого місця

Освітлення робочого місця програміста є важливим аспектом охорони праці згідно зі стандартами, встановленими державними органами. Згідно з Нормативними Правилами "Вимоги щодо освітлення приміщень" (НПАОП 0.00-1.29-10), затвердженими Державним комітетом України з промислової безпеки, охорони праці та гірничого нагляду, робоче місце програміста повинно відповідати таким вимогам:

– Рекомендована яскравість освітлення на робочому місці програміста повинна бути в межах 300-500 люксів (вимірювання освітленості). Це допомагає забезпечити достатню видимість без зайвого напруження очей.

– Світильники повинні бути розташовані таким чином, щоб забезпечити рівномірне розподілення світла по всій робочій поверхні. За необхідності можуть використовуватися спеціальні абажури або дифузори для розсіювання світла.

– Рекомендується використовувати світло з колоровою температурою близькою до природного світла, наприклад, біле або нейтральне світло. Завдяки цьому досягається краща чіткість зорового сприйняття та зменшується втома очей.

– Для уникнення відблисків на екрані монітора рекомендується розташовувати світильники таким чином, щоб уникнути прямого падіння світла на екран. Також можна встановити спеціальні абажури або використовувати екрани, що розсіюють світло.

1.4 Мікроклімат

Оптимальний мікроклімат у робочому приміщенні для програміста є важливим аспектом охорони праці, яким необхідно керуватися відповідно до нормативно-правових актів та рекомендацій державних органів. Зокрема, Державним комітетом України з промислової безпеки, охорони праці та гірничого нагляду встановлені такі вимоги до мікроклімату робочого приміщення:

– Рекомендована температура в робочому приміщенні для програміста знаходиться в діапазоні від 20 до 24 градусів Цельсія .

– Вологість повітря: Рекомендована вологість повітря в робочому приміщенні знаходиться в діапазоні від 40% до 60%.

– Швидкість руху повітря: Швидкість руху повітря в робочому приміщенні не повинна перевищувати 0,25 метра на секунду.

– Чистота повітря: Робоче приміщення повинно мати ефективну систему припливно-витяжної вентиляції для забезпечення свіжого та чистого повітря.

1.5 Електробезпека

– Електропроводка: Рекомендується встановлювати електропроводку відповідно до вимог нормативних документів, таких як "Правила улаштування електроустановок" (ПУЕ). З'єднання проводів повинні бути надійними, а ізоляція має бути відповідати нормам безпеки.

– Електроустаткування: Важливо використовувати електрообладнання, яке має сертифікат відповідності та відповідає вимогам безпеки. Розташовуйте обладнання з дотриманням правил відстаней до легкозаймистих матеріалів.

2. Пожежна безпека

Пожежна безпека визначається як стан, коли додержуючись встановленої ймовірності, виключається можливість виникнення пожежі та її подальшого поширення, а також забезпечується захист людей від небезпечних пожежних факторів і збереження матеріальних цінностей. У приміщенні можуть бути різноманітні потенційні причини пожежі, такі як коротке замикання електропроводки, неправильне використання побутових електроприладів та порушення протипожежних заходів. Згідно з Правилами улаштування електроустановок (ПУЕ), для забезпечення пожежної безпеки необхідно вжити такі заходи:

– Приміщення з ЕОМ повинні бути оснащені системою автоматичної пожежної сигналізації відповідно до вимог Переліку однотипних за призначенням об'єктів, які підлягають обладнанню автоматичними установками пожежогасіння та пожежної сигналізації, з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками з розрахунку 2 шт. на кожні 20 кв. м площі приміщення з урахуванням граничнодопустимих концентрацій вогнегасної рідини відповідно до вимог Правил пожежної безпеки в Україні. В інших приміщеннях допускається встановлювати теплові, пожежні відповідно ГОСТ 12.4.009-75 та ISO3941-77.

– Перевірка електрообладнання: Регулярно проводьте огляд та перевірку електрообладнання на наявність пошкоджень, стан ізоляції та відповідність нормативним вимогам. Запобігайте перевантаженню електричних мереж та використанню несумісних електроприладів.

– Використання пожежогасних засобів: Мають бути доступні та легкодоступні пожежогасники, які відповідають вимогам протипожежної безпеки. Періодично перевіряйте їх на наявність заряду та цілісність.

– Експлуатаційні вимоги: Дотримуйтеся інструкцій виробників щодо експлуатації та обслуговування електронно-обчислювальних приладів. Не залишайте ввімкнені пристрої без нагляду та не перевантажуйте їх.

Дослідження та вирішення проблем, пов'язаних із забезпеченням здорового та безпечного робочого середовища, має велике значення у виробничих системах. Це одне з найважливіших завдань, спрямованих на забезпечення безпеки та добробуту працівників. Проведення досліджень, виявлення можливих причин виробничого травматизму, професійних захворювань, аварій і пожеж та розробка ефективних заходів щодо їх усунення є одним із способів створення безпечних умов праці для людей. Це допоможе зберегти здоров'я працівників, підвищити продуктивність праці та покращити загальне функціонування організації. Забезпечення комфортного та безпечного робочого середовища також відіграє важливу роль у забезпеченні добробуту та продуктивності працівників. Це включає в себе сприятливі параметри мікроклімату, належне освітлення, безпечне обладнання та відповідні заходи безпеки. Ці умови допомагають підтримувати фізичний і психологічний комфорт працівників, знижують ризик нещасних випадків і покращують загальну робочу атмосферу в організації.

ВИСНОВКИ

У процесі виконання дипломної роботи було розроблено програму для перевірки текстів на унікальність. У результаті роботи можна зробити наступні висновки:

Була розроблена система перевірки унікальності текстів на веб-сайті. Ця система дозволяє користувачам завантажувати свої тексти для аналізу та отримувати результати перевірки.

В системі була реалізована модернізація програми перевірки унікальності текстів, що включає в себе розробку веб-інтерфейсу, бази даних, серверних та клієнтських компонентів.

Для розробки веб-інтерфейсу були використані технології HTML, CSS та JavaScript. Це дозволило створити зручний і інтуїтивно зрозумілий інтерфейс для користувачів.

База даних була спроектована для зберігання інформації про завантажені тексти, результати перевірки та інші відомості. Використання бази даних дозволяє зберігати дані ефективно і забезпечувати швидкий доступ до них.

Система пройшла різноманітні види тестування, такі як модульне, інтеграційне, нагрузочне та юзабіліті-тестування. Це дозволило перевірити працездатність, стабільність та коректність роботи системи у різних умовах та навантаженнях.

Результати розробки системи свідчать про те, що вона є ефективним інструментом для перевірки унікальності текстів. Користувачі можуть легко завантажити свої тексти та отримати результати перевірки у зручному форматі.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Angular University [Електронний ресурс] https://angular-university.io/ (15.05.2023)

2. ASP.NET Core Documentation [Електронний ресурс] https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1 (16.05.2023)

3. Duckett, J. (2014). JavaScript and JQuery: Interactive Front-End Web Development. Wiley.

4. Fowler, M. (2012). Patterns of Enterprise Application Architecture. Addison-Wesley Professional.

5. Mardan, A. (2018). Full-Stack React, TypeScript, and Node. Apress.

6. Negrino, T., & Smith, D. (2018). JavaScript and jQuery: Visual QuickStart Guide. Peachpit Press.

7. Richardson, C., & Smith, R. (2019). Microservices Patterns: With examples in Java. Manning Publications.

8. ASP.NET Core документація [Електронний ресурс] https://docs.microsoft.com/uk-ua/aspnet/core/?view=aspnetcore-5.0 (14.05.2023)

9. Веллі, К. Angular: Повне керівництво для розробників / К. Веллі, А. Балес. - Київ: Видавництво "БХВ-Петербург", 2020. - 564 с.

10. Крокфорд, Д. JavaScript: Хороші частини / Д. Крокфорд. - Львів: Видавництво Старого Лева, 2013. - 176 с.

11. Макдональд, М. ASP.NET Core 2.0 MVC и Razor Pages для начинающих: как самостоятельно изучить основы MVC для работы над реальными проектами / М. Макдональд. - Київ: Комп'ютерна література, 2018. - 610 с.

12. Основи Angular [Електронний ресурс] https://angular.io/start (12.05.2023)

13. Фрімен, Е. HTML і CSS. Навчаємося з головою / Е. Фрімен, Р. Робсон. – Київ: Комп'ютерна література, 2015. - 624 с.

Додаток А.

Лістинг візуалізації даних

```
public class Document
{
    public int ID { get; set; }
    public string Text { get; set; }
    public DateTime UploadTime { get; set; }
    public virtual ICollection<CheckResult> CheckResults { get; set; }
}

public class CheckResult
{
    public int ID { get; set; }
    public int DocumentID { get; set; }
    public Document Document { get; set; }
    public DateTime CheckTime { get; set; }
    public float Result { get; set; }
}

[Route("api/[controller]")]
[ApiController]
public class DocumentsController : ControllerBase
{
    private readonly DocumentsContext _context;

    public DocumentsController(DocumentsContext context)
    {
        _context = context;
    }
```

```
    [HttpGet("{id}")]
    public async Task<ActionResult<Document>> GetDocument(int id)
    {
        // Get the document with the given ID
    }

    [HttpPost]
    public async Task<ActionResult<Document>> PostDocument(Document document)
    {
        // Save the new document
    }
}

public class DocumentsControllerTests
{
    private readonly DocumentsController _controller;
    private readonly Mock<DocumentsContext> _mockContext;

    public DocumentsControllerTests()
    {
        _mockContext = new Mock<DocumentsContext>();
        _controller = new DocumentsController(_mockContext.Object);
    }

    [Fact]
    public async Task GetDocument_ReturnsCorrectDocument()
    {
        // Arrange
```

```
        var testDocument = new Document { ID = 1, Text = "Test text" };
        _mockContext.Setup(x => x.Documents.Find(1)).Returns(testDocument);

        // Act
        var result = await _controller.GetDocument(1);

        // Assert
        Assert.Equal(testDocument, result.Value);
    }
}
public class DocumentsControllerTests
{
    private readonly DocumentsController _controller;
    private readonly Mock<DocumentsContext> _mockContext;

    public DocumentsControllerTests()
    {
        _mockContext = new Mock<DocumentsContext>();
        _controller = new DocumentsController(_mockContext.Object);
    }

    [Fact]
    public async Task GetDocument_ReturnsCorrectDocument()
    {
        // Arrange
        var testDocument = new Document { ID = 1, Text = "Test text" };
        _mockContext.Setup(x => x.Documents.Find(1)).Returns(testDocument);

        // Act
        var result = await _controller.GetDocument(1);
```

```
        // Assert
        Assert.Equal(testDocument, result.Value);
    }
}
```

```
import { Component } from '@angular/core';
import { DocumentService } from '../services/document.service';

@Component({
  selector: 'app-document-upload',
  templateUrl: './document-upload.component.html'
})
export class DocumentUploadComponent {
  constructor(private documentService: DocumentService) { }

  onFileSelected(event) {
    const file: File = event.target.files[0];
    if (file) {
      this.documentService.uploadDocument(file).subscribe(
        response => console.log(response),
        error => console.error(error)
      );
    }
  }
}
```

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
```

```
@Injectable({
  providedIn: 'root'
})
export class DocumentService {
  private readonly apiUrl = 'http://localhost:5000/api/documents';

  constructor(private http: HttpClient) { }

  uploadDocument(file: File): Observable<any> {
    const formData = new FormData();
    formData.append('file', file);

    return this.http.post(this.apiUrl, formData);
  }

  getDocument(id: number): Observable<any> {
    return this.http.get(`${this.apiUrl}/${id}`);
  }

  getCheckResult(id: number): Observable<any> {
    return this.http.get(`${this.apiUrl}/${id}/check-result`);
  }
}
```

ДОДАТОК Б

Слайди мультимедійної презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

Модернізація програми перевірки унікальності текстів на веб-сайті

Студента групи 4РП-06
Цвяткова Сергія Дмитровича

Мета проекту

Метою даної дипломної роботи є розробка та впровадження модернізації програми перевірки унікальності текстів на веб-сайті з метою покращення її ефективності, точності та швидкості роботи.

Для досягнення мети потрібно зробити наступні кроки:

1. Визначити проблему, та знайти існуючи рішення
2. Зрозуміти поняття та значення унікальності тексту
3. Мати визначення ключових вимог до системи перевірки унікальності текстів
4. Побудувати основних функцій та компонентів системи
5. Розробити архітектуру та дизайн системи
6. Зробити тестування функціональності та продуктивності системи
7. Впровадження модернізованої системи на веб-сайті помилок.

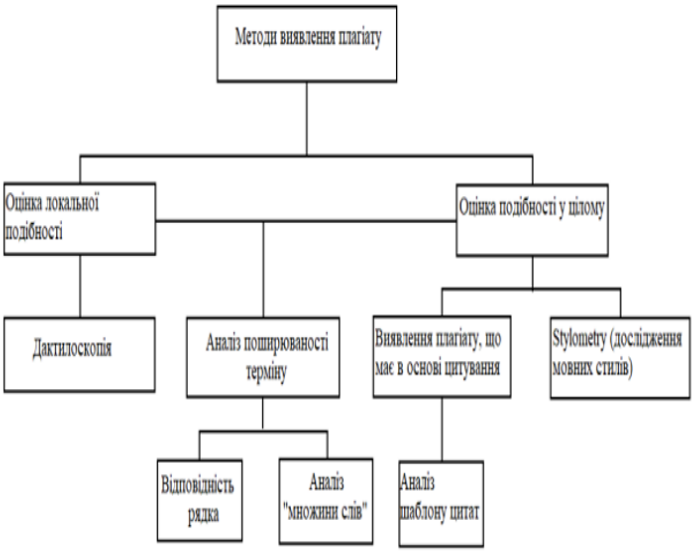
Об’єкт та предмет дослідження

Об'єктом нашого дослідження є процес перевірки унікальності текстів в цифровому середовищі, а точніше - програми для перевірки унікальності текстів на веб-сайтах. Ці програми використовуються в різних сферах - від освіти до бізнесу - для гарантування авторства, виявлення плагіату та підтримки якості контенту.

Що таке унікальність тексту

Унікальність тексту — його оригінальність, винятковість, унікальність. Такий текст немає аналогів, не копіює інші твори і є авторським твором. Унікальність тексту важлива у багатьох сферах, від науки та освіти до бізнесу та журналістики.

У науковій та освітній сферах унікальність тексту гарантує авторство ідей, висновків та результатів дослідження. Наукові журнали та конференції вимагають унікальних статей, щоб гарантувати, що кожна публікація є новим вкладом у знання. В освіті унікальність тексту підтверджує, що студенти самостійно освоїли матеріал та можуть застосовувати знання на практиці.



Структура веб-додатку

1. Головна сторінка
2. Авторизація
3. Форма перевірки контенту
4. Видача результатів

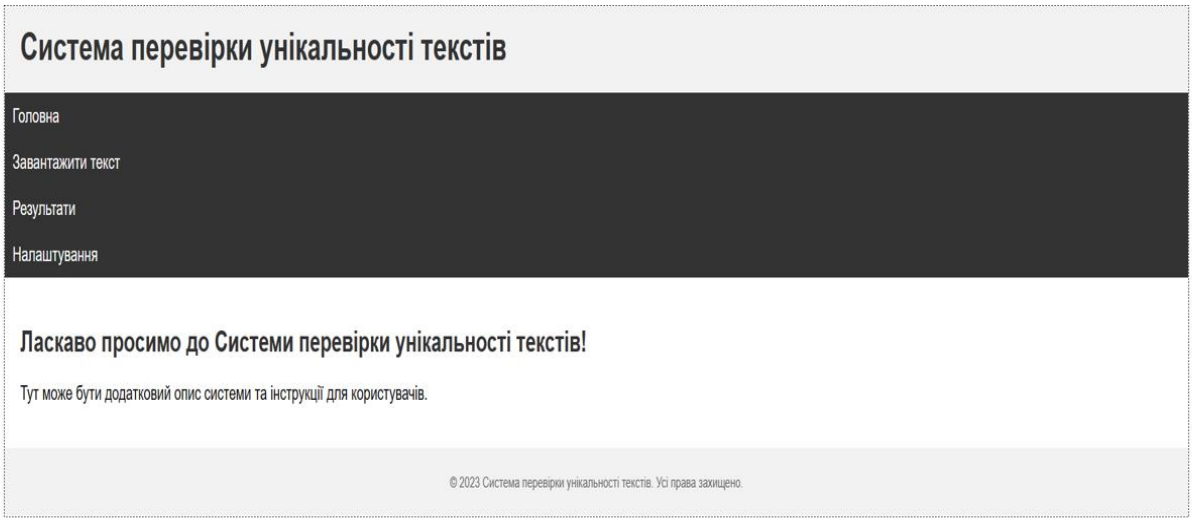
Серверна частина

- Для бекенду на ASP.NET, ми створюємо контролери для обробки запитів від клієнтської частини. Контролери мають методи для обробки HTTP запитів GET та POST, які відповідають за завантаження документів і отримання результатів перевірки відповідно. Сервіс перевірки унікальності реалізує алгоритм перевірки, який аналізує текст документа і визначає ступінь його унікальності.
- База даних, розроблена на основі SQL Server, включає таблиці Documents і CheckResults, які зберігають інформацію про завантажені документи і результати їх перевірки відповідно. Розробка бази даних включає створення цих таблиць з визначеними полями, а також налаштування відповідних зв'язків між ними.

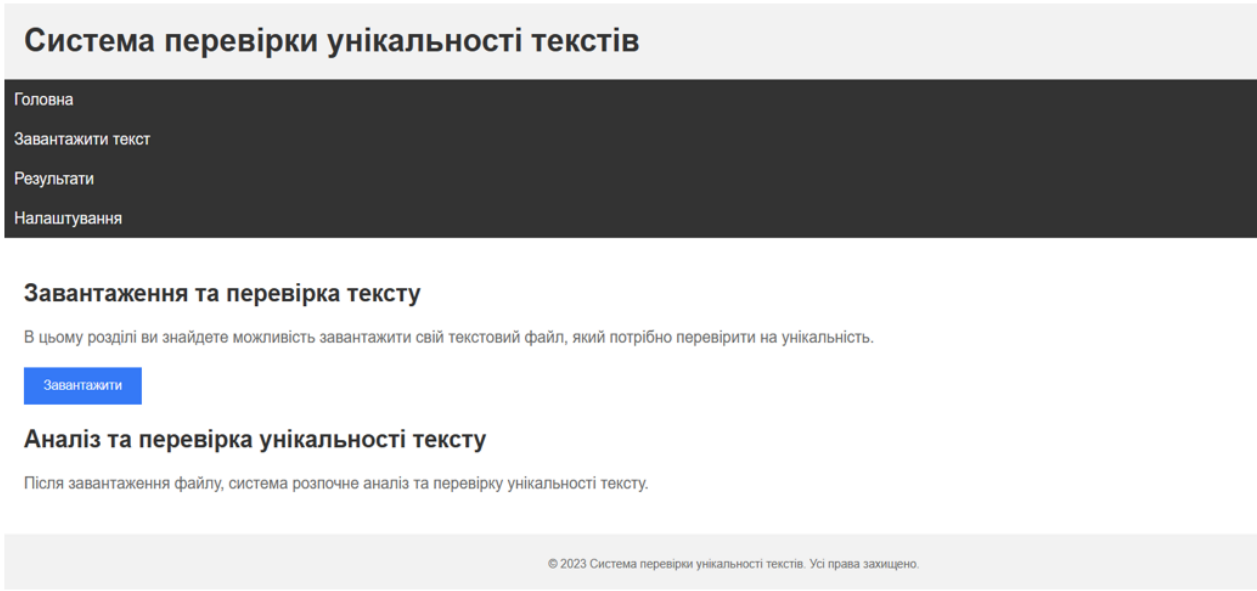
Клієнтська частина

Angular – це фреймворк для створення клієнтської частини веб-додатків. Він включає в себе набір інструментів, які дозволяють розробникам легко створювати інтерактивні, високопродуктивні та масштабовані веб-інтерфейси.

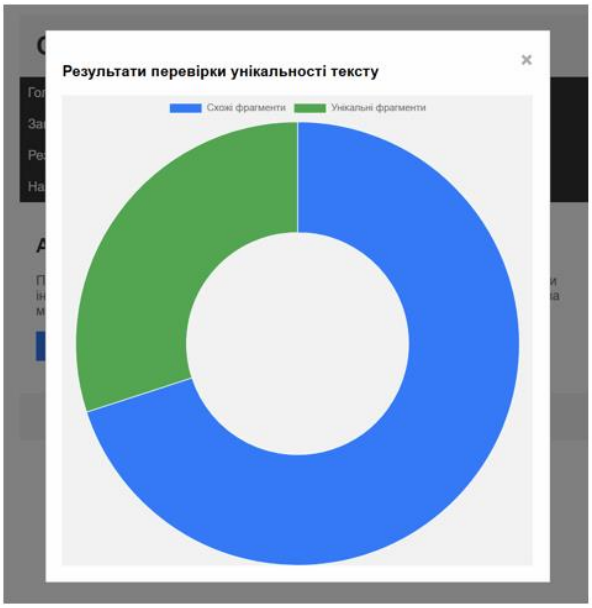
Веб-додаток



Веб-додаток



Веб-додаток



Висновки

У результаті даної роботи була розроблена система перевірки унікальності текстів на веб-сайті. Ця система дозволяє користувачам завантажувати свої тексти для аналізу та отримувати результати перевірки.

В системі була реалізована модернізація програми перевірки унікальності текстів, що включає в себе розробку веб-інтерфейсу, бази даних, серверних та клієнтських компонентів.

Для розробки веб-інтерфейсу були використані технології HTML, CSS та JavaScript. Це дозволило створити зручний і інтуїтивно зрозумілий інтерфейс для користувачів.

База даних була спроектована для зберігання інформації про завантажені тексти, результати перевірки та інші відомості. Використання бази даних дозволяє зберігати дані ефективно і забезпечувати швидкий доступ до них.

Система пройшла різноманітні види тестування, такі як модульне, інтеграційне, нагрузочне та юзабіліті-тестування. Це дозволило перевірити працездатність, стабільність та коректність роботи системи у різних умовах та навантаженнях.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Цвятков Сергій Дмитрович

(прізвище, ім`я та по батькові)

Спеціальність 121 " Інженерія програмного забезпечення"

Освітня програма «Розробка програмного забезпечення»

Керівник дипломного проекту (роботи) Шувалова Ірина Олегівна

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) Модернізація програми перевірки унікальності текстів на веб-сайті

Обсяг розрахунково-пояснювальної записки 65 сторінок

Обсяг графічної (презентаційної) частини 10 аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню
Дипломний проект відзначається глибокою проробкою теми, розробкою ефективних алгоритмів перевірки унікальності текстів та їх інтеграцією у веб-сайт.

б) характеристика виконання кожного розділу дипломного проекту (роботи)
Проект містить відповідні теоретичні розділи, де розглядаються сучасні методи перевірки текстів на унікальність, та практичний розділ, де презентується модернізована програма із розширеним функціоналом.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) Розрахунково-пояснювальна записка детально розкриває технічні аспекти модернізації, а графічна частина ефективно допомагає у візуалізації архітектурних рішень.

г) перелік позитивних якостей дипломного проекту (роботи)

Позитивними аспектами роботи є зосередженість на актуальній проблемі перевірки унікальності, глибокий аналіз та розробка ефективних рішень для підвищення продуктивності та точності перевірки.

д) основні недоліки дипломного проекту (роботи)

Недоліком є недостатня увага до безпеки даних та можливих способів зловживання системою для плагіату.

Оцінка розрахункової частини добре

Оцінка графічної частини добре

Загальна оцінка добре

Прізвище, ім'я, по батькові рецензента Кривченко Юрій Вікторович

Місце роботи і посада рецензента ВСП "Одеський технічний фаховий коледж ОНТУ", голова циклової комісії комп'ютерних технологій та програмної інженерії

Підпис:

«16» червня 2023 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНАХТ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Цвятков Сергій Дмитрович

(прізвище, ім`я та по батькові)

Спеціальність: 121 - Інженерія програмного забезпечення

Освітня програма: Розробка програмного забезпечення

Тема дипломного проекту: Модернізація програми перевірки унікальності текстів на веб-сайті

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки)

Пояснювальна записка складена відповідно до стандартів ДСТУ. Обсяг та зміст роботи відповідають поставленим цілям дипломного проектування. Розрахунково-пояснювальний розділ проекту характеризується чіткою структурою та високою якістю виконання.

б) самостійність роботи над проектом:

Цвятков С.Д. ефективно зайнявся опрацюванням наукового матеріалу, провів глибокий аналіз методик та стратегій розробки веб-орієнтованих систем, продемонстрував уміння критично аналізувати літературні джерела та вдало використовував новітні інформаційні технології під час досліджень.

в) теоретична підготовка випускника (випускниці):

Студент Цвятков С.Д. має теоретичну та практичну базу на високому рівні. Він проявляє високу концентрацію, пильність та відповідальність у своїй роботі.

г) вміння розв'язувати виробничі та конструкторські питання _____

Студент Цвятков С.Д. за період роботи показал Високий рівень теоретичної та практичної підготовки

Оцінка розрахункової частини «Добре»

Оцінка графічної частини «Добре»

Загальна оцінка «Добре»

Прізвище, ім'я, по батькові керівника дипломного проекту _____

Шувалова Ірина Олегівна

Місце роботи і посада керівника дипломного проекту Викладач, ВСП „ОТФК ОНТУ" Шувалова І.О

Підпис _____

«09» 06 2023 р.

UNICHECK

Ім'я користувача:
Наталія Вікторівна Копусь

ID перевірки:
1015519310

Дата перевірки:
09.06.2023 08:22:28 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
09.06.2023 08:26:26 EEST

ID користувача:
100011688

Назва документа: 4РП-06 Цвятков С.Д

Кількість сторінок: 59 Кількість слів: 10058 Кількість символів: 83042 Розмір файлу: 1.09 MB ID файлу: 1015173815

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.24%
Схожість

Найбільша схожість: 0.59% з Інтернет-джерелом (http://elartu.tntu.edu.ua/bitstream/lib/31148/1/dyplom_Yuzvak_V_202..

5.24% Джерела з Інтернету | 1000 ..... Сторінка 61

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи | 21

Підозріле форматування | 10 сторінок

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Цвятков Сергій Дмитрович,
здобувач освіти гр. 4РП-06, та

Шувалова Ірина Олегівна,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної кваліфікаційної роботи молодшого спеціаліста на тему:

«Модернізація програми перевірки унікальності текстів на веб-сайті» (автор роботи – Цвятков С.Д., керівник роботи – Шувалова І.О.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2023 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Цвятков С.Д. /

Керівник _____ / Шувалова І.О. /

«09» 06 2023 р.