УДК 004.03; 004.42
УКПП ХХХХХХ
№ держреєстрації
Інв. №

Міністерство освіти і науки України
Одеський державний екологічний університет (ОДЕКУ)
65016, м. Одеса, вул. Львівська, 15; тел/факс (0482) 32-67-40

ЗАТВЕРДЖУЮ
Проректор з наукової роботи ОДЕКУ
д-р.геогр.наук, проф.
_____Ю.С. Тучковенко
(підпис)

ЗВІТ
ПРО НАУКОВО-ДОСЛІДНУ РОБОТУ

АДАПТАЦІЯ ТА ВПРОВАДЖЕННЯ ЗАСОБІВ ПЕРЕВІРКИ НА ПЛАГІАТ ТВОРІВ ПРАЦІВНИКІВ ТА СТУДЕНТІВ УНІВЕРСИТЕТУ, З ВИКОРИСТАННЯМ БАЗ ДАНИХ УНІВЕРСИТЕТУ ТА ІНФОРМАЦІЙНИХ РЕСУРСІВ МЕРЕЖІ INTERNET

(остаточний)

Науковий керівник НДР
канд. геогр. наук, доц. _____ С.Д. Кузніченко
(підпис)

2017

Рукопис закінчено 25 листопада 2017 р.

Результати роботи розглянуто Науково-технічною радою ОДЕКУ, протокол від 28 грудня 2017 р №30

СПИСОК АВТОРІВ

Керівник НДР, в.о.завідувача кафедри канд. геогр. наук, доцент	С.Д. Кузніченко (вступ; розділи 1,2, 3; висновки)
Відповідальні виконавці:	
Канд. ф.-м. наук, доцент	В.П.Козловська (реферат; розділ 1)
Старш. викл.	Л.В. Ременяк (підрозділи 1.1, 1.2; підрозділи 2.1, 2.2; розділ 3)
Старш. викл.	Л.С. Кострицька (підрозділи 1.3, 1.4; підрозділи 3.1, 3.2)
Старш. викл.	Н.З.Штефан (підрозділи 2.3, 2.4; підрозділи 3.3, 3.4)
Виконавці:	
Аспірант	І.В. Бучинська (підрозділи 1.1, 1.2)
Магістр	О.О. Грибова (підрозділи 1.3, 1.4)
Магістр	М.С. Яковенко (підрозділи 2.1, 2.2)
Магістр	А.С. Мазур (підрозділи 2.3, 2.4; підрозділи 3.1)
Магістр	М.О. Щербина (підрозділи 3.2, 3.3)
Магістр	Л.А.Каніковська (підрозділи 2.3, 2.4)

РЕФЕРАТ

Звіт про НДР: 94 с., 2 табл., 66 рис., 35 джерело.

ПЛАГІАТ, ЗАСОБИ АВТОМАТИЧНОГО ПОШУКУ ПЛАГІАТУ, ПРОГРАМНЕ ЗАБЕСПЕЧЕННЯ, ОН-ЛАЙН РЕСУРС.

Об’єкт дослідження – програмні та інструментальні засоби автоматичної перевірки на плагіат творів працівників та студентів університету.

Мета роботи – дослідження, вдосконалення та впровадження наявних програмних та інструментальних методів та засобів перевірки на плагіат творів працівників та студентів університету з використанням баз даних університету та інформаційних ресурсів мережі Internet, а також розробка нових аналогічних програм та інструментальних методів та засобів.

Актуальність роботи полягає у необхідності створення в університеті єдиної системи перевірки на плагіат творів студентів та працівників університету, згідно вимогам закону України «Про вищу освіту».

Методи дослідження – методи системного тестування програмного забезпечення, методологія структурного проектування інформаційних систем, методи та засоби концептуального, логічного та фізичного проектування баз даних.

За результатами виконання роботи апробовані наявні програмні засоби для перевірки на плагіат, виконано обґрунтований вибір та доопрацьовування програмних засобів, які будуть впроваджені у навчально-наукову діяльність університету, розроблена структура бази даних творів студентів та працівників університету та процедура перевірки творів на плагіат.

Розроблені методичні рекомендації щодо практичного застосування програмних засобів структурними підрозділами університету.

Результати НДР є основою системи перевірки творів на плагіат в університеті. Програмні засоби впроваджені в навчально-виробничу діяльність університету: в підрозділи, які безпосередньо виконують перевірку творів, створених студентами та працівниками університету на плагіат.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМИНІВ

Скорочення

БД – база даних.

ІС – інформаційна система.

МП – мова програмування.

ОЗП – оперативний запам’ятовуючий пристрій

ОС – операційна система

ПЗ – програмне забезпечення.

СКБД – система керування базами даних.

CLR – Common Language Runtime – віртуальна машина, на якій виконуються всі мови платформи .NET Framework.

CRUD – Create, Read, Update, Delete – 4 базові функції управління даними «створення, зчитування, зміна і видалення».

GUI – Graphic User Interface – графічний інтерфейс користувача.

IDE – Integrated Development Environment – комплексне програмне рішення для розробки програмного забезпечення.

JVM – Java Virtual Machine – віртуальна машина Java

MVVM – Model-View-ViewModel – шаблон проектування.

SQL – structured query language – структурована мова запитів.

Терміни

Автор – фізична особа, творчою працею якої створено твір.

Десктопний додаток – додаток (комп’ютерна програма), яка виконується на настільному комп’ютері чи ноутбуці.

Плагіат – оприлюднення (опублікування), повністю або частково, чужого твору під іменем особи, яка не є автором цього твору (ст. 50 Закону України «Про авторське право і суміжні права»).

Плагіат академічний – навмисне відтворення докторантом, аспірантом або студентом у письмовій або електронній формі чужого твору,

опублікованого на паперовому або офіційно оприлюдненого на електронному носії, повністю або частково, під своїм іменем без посилання на автора.

Плагіат науковий – навмисне відтворення у наукових творах наукових, науково-педагогічних, педагогічних працівників у письмовій або електронній формі чужого твору, опублікованого на паперовому або офіційно оприлюдненого на електронному носії, повністю або частково, під своїм іменем без посилання на автора.

Твір – інформація, як результат наукової чи навчально-методичної діяльності конкретної особи (чи у співавторстві), представлена на паперових носіях або в електронному вигляді у мережі Інтернет чи оприлюднена на офіційному web-сайті університету (монографія, підручник, навчальний посібник, стаття, доповідь, препринт, автореферат і рукопис дисертації (дисертаційна робота), курсові та кваліфікаційні роботи і проекти).

Цитата – порівняно короткий уривок з літературного, наукового чи будь-якого іншого опублікованого (оприлюдненого на офіційному web-сайті) твору, який використовується, з обов'язковим посиланням на його автора і джерело цитування, іншою особою у своєму творі з метою зробити зрозумілішими свої твердження або для посилання на погляди іншого автора в автентичному формулюванні.

C# – мова програмування створена спеціально для роботи у середовищі Microsoft .NET Framework.

Java – мультиплатформенна мова програмування та платформа для виконання додатків, написаних на Java.

Linux – загальна назва UNIX-подібних операційних систем на основі однойменного ядра.

MacOS – перша комерційна графічна ОС, створена компанією Apple Computer. Зазвичай використовується на настільних комп’ютерах та ноутбуках від компанії Apple. Являється UNIX-подібною системою.

MongoDB – документо-орієнтована система керування базами даних з відкритим вихідним кодом (open source), яка не потребує опису колекцій (аналог таблиць в реляційних СКБД).

Open Source – «відкритий» вихідний код ПЗ, тобто такий, який можна використовувати в своїх роботах всім охочим.

PostgreSQL – вільна об’єктно-реляційна СКБД. Основана, як і всі реляційні СКБД на мові SQL.

Python – об’єктно-орієнтована, мультиплатформенна та скриптова мова програмування, зазвичай служить для написання наукових додатків, але може й використовуватись для написання повноцінних десктопних та веб-додатків.

Windows – узагальнююча назва ОС, розроблених компанією Microsoft, орієнтованих на застосування графічного інтерфейсу при управлінні. Найбільша, по кількості користувачів, операційна система.

XAML – декларативна мова розмітки. З точки зору моделі програмування .NET Framework мова XAML спрощує створення користувацького інтерфейсу для програми, написаної за допомогою .NET Framework.

ВСТУП

В останнє десятиліття актуальним завданням стає боротьба з неправомірним використанням чужої інтелектуальної власності, одним з проявів якого є плагіат. Найбільшого розмаху ця проблема досягла в освітній системі країни. З появою завдяки широкому розвитку Інтернет відкритого доступу до світового наукового досвіду, у вигляді безлічі науково-методичних публікацій в електронному форматі, збільшилася кількість робіт, що дублюють одна одну. Сама можливість вкрасти чи скопіювати чужу роботу чи її частину несе неприємні наслідки та погіршує рівень освіти. При цьому, цей процес охопив ввесь спектр наукових робіт – від дипломних робіт до статей і дисертацій.

Тому відповідно до Законів України «Про вищу освіту», «Про авторське право і суміжні права», Цивільного кодексу України актуальним є створення в університеті єдиної системи перевірки на плагіат творів студентів та працівників університету, з метою запобігання плагіату в наукових та навчальних працях працівників університету, докторантів, аспірантів, здобувачів наукового ступеня та студентів. Результатом впровадження системи перевірки на плагіат є підвищення якості навчання, розвиток навичок коректної роботи із джерелами інформації та формування звички до сумлінного дотримання вимог наукової етики, активізація самостійності та індивідуальності при створенні авторського твору та підвищення відповідальності з боку працівників, докторантів, аспірантів, здобувачів наукового ступеня, студентів усіх форм навчання за порушення загальноприйнятих правил.

Мета роботи є дослідження, вдосконалення та впровадження наявних програмних та інструментальних методів та засобів перевірки на плагіат творів працівників та студентів університету з використанням баз даних університету та інформаційних ресурсів мережі Internet, а також розробка нових аналогічних програм та інструментальних методів та засобів.

Завдання, що потребують вирішення відповідно до мети роботи:

- Дослідження та порівняльний аналіз існуючих програмних засобів пошуку плагіату;
- Апробування наявних програмних засобів для перевірки на плагіат, обґрунтування вибору та доопрацьовування відповідно вимогам університету програмних засобів, які будуть впроваджені у навчально-наукову діяльність університету;
- Аналіз існуючих алгоритмів пошуку текстових збігів та розробка нових алгоритмів;
- Розроблення структури бази даних, проектування та розробка програмного засобу перевірки на плагіат творів працівників і студентів університету з використанням внутрішньої бази даних університету;
- Розроблення методичних рекомендацій щодо практичного застосування програмних засобів структурними підрозділами університету.

Результати НДР стануть основою системи перевірки творів на плагіат в університеті. Програмні засоби будуть впроваджені в навчально-виробничу діяльність університету: в підрозділи, які безпосередньо будуть виконувати перевірку творів, створених студентами та працівниками університету на плагіат.

1 ДОСЛІДЖЕННЯ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ ІСНУЮЧИХ ОНЛАЙН-РЕСУРСІВ ТА ПРОГРАМНИХ ЗАСОБІВ ПОШУКУ ПЛАГІАТУ

У наукових публікаціях вітчизняних і зарубіжних авторів проводяться численні дослідження існуючих засобів автоматичного пошуку плагіату у текстових документах. Так в роботах [1-8] наданий порівняльний аналіз наявних програмних засобів. Деякі з цих програм є безкоштовними, а деякі з них комерційно доступні.

У розділі 1 представлено порівняння між найбільш відомими програмами виявлення плагіату, яке дасть можливість обрати програмні засоби, які найбільш відповідають цілям і задовольняють вимоги системи університету. Для дослідження були обрані інструменти, які найбільш популярні для виявлення плагіату у текстових документах, наданий короткий опис їх основних характеристик і результати апробування.

1.1 Класифікація інструментів автоматичного відстеження плагіату

У роботі [8] автором надана класифікація пропрієтарних і вільно поширюваних інструментів автоматичного відстеження плагіату, яка наведена у табл.1.1. Програмні засоби автоматичного відстеження плагіату, як видно з табл.1.1, крім того, що можуть бути пропрієтарними (комерційними) та вільно поширюваними, мають відмінності, пов’язані з типом бази даних, що підтримується, та з даними, якими вона оперує. Для цілей перевірки творів працівників та студентів на плагіат потрібний гібридний варіант бази даних, тому що перевірка передбачає, як доступ до внутрішньої бази даних, що наповнюється науково-методичними публікаціями університету, так і до матеріалів мережі Інтернет. Що стосується типів даних, то система університету призначена для відстеження

плагіату лише у текстових матеріалах, і не є обов’язковою умовою підтримка перевірки на плагіат програмних кодів і нетекстових (графічних) матеріалів.

Таблиця 1.1 – Класифікація інструментів автоматичного відстеження плагіату

Назва	Класифікація	Опис
Класифікація за даними	за типом бази даних	внутрішня (наповнюється власником і/або користувачем)
		зовнішня (електронний репозиторій, всі доступні в мережі Інтернет матеріали)
		гібридна (внутрішня і зовнішня)
	за типом даних, якими оперує	текстові матеріали
		програмні коди
		нетекстові матеріали (таблиці, схеми та організаційні діаграми, медіа, тощо)
Класифікація за характеристиками ресурсу	за типом	десктопне програмне забезпечення для локального встановлення на робочу станцію користувача
		он-лайн ресурс
	за формою власності	пропрієтарне
		вільнопоширюване
	за спектром доступних мов	для текстових матеріалів – наявність іноземних мов, наприклад, англійської, китайської, російської, тощо із врахуванням лінгвістичних особливостей кожної мови
		для програмних кодів - C, Java, php, Prolog, тощо
Класифікація за складністю метрик, що використовуються	поверхневі показники	кількість однакових фрагментів довжиною у п’ять слів у документах, що порівнюються
	структурні показники	міра подібності документів, що порівнюються, із врахуванням їх структури

Перевагу слід віддати інструментам, що є онлайн-ресурсами, тобто мають архітектуру веб-застосування. Це надасть можливість більш зручної роботи працівників університету з програмою. Для роботи з внутрішньою базою даних планується створення власного десктопного програмного забезпечення, яке буде встановлено на локальних комп'ютерах у відповідних підрозділах університету, які безпосередньо виконують перевірку творів, створених студентами та працівниками університету на плагіат. Етапи розроблення подібного програмного забезпечення наведені у розділі 2 НДР.

Актуальним для проведення порівняння є спектр доступних мов. Бажано, щоб інструмент перевірки на плагіат підтримував українську, російську та англійську мови.

З врахуванням вищесказаного сформулюємо у наступному пункті роботи основні критерії за якими буде проводитися порівняння інструментів автоматичного відстеження плагіату.

1.2 Критерії порівняння інструментів автоматичного відстеження плагіату

Аналіз будемо проводити лише для вільнопоширюваних ресурсів і пропроієтарного програмного забезпечення, доступ до якого університет має відповідного до заключних договорів про технічну апробацію. Дослідження проводилося за критеріями, наведеними у табл. 1.2.

В процесі порівняння було досліджено відповідність умов використання і функціональних та програмно-технічних вимог ресурсів потребам системи університету, чи задовольняють вони специфіки наукової діяльності університету.

Важливим параметром для аналізу є продуктивність ресурсу. Важливо, щоб звіт про перевірку текстового документа мав форму прийнятну для використання у звітності університету. Час використання ресурсу один з важливих факторів, поряд з кількістю користувачів та обмеженням щодо

об’єму введеного тексту. Важною характеристикою, що безпосередньо впливає на час впровадження ресурсу в підрозділах університету є особливості інтерфейсу (інтуїтивна зрозумілість, наявність інструктивних та довідкових матеріалів).

Таблиця 1.2 – Критерії порівняння інструментів автоматичного відстеження плагіату

№ з/п	Назва	Критерії
1.	Умови використання	тип ресурсу (пропрієтарний/ вільнопоширюваний)
		можливість безкоштовного тріал-доступу
		необхідність обов’язкової реєстрації на сайті та укладання договору з розробником
		кількість користувачів однієї наукової установи, що матимуть доступ до програми
2.	Функціональні та програмно-технічні вимоги	кількість мов, що підтримуються
		формати файлів, якими оперує ресурс
		база пошуку (мережа Інтернет, інституційний репозиторій, локальна системна БД) та інтеграція з пошуковими машинами
		типи елементів, що перевіряються (фрагменти введеного вручну тексту/ файли / веб-сторінки)
		обмеження щодо об’єму введеного тексту
		обмеження щодо кількості перевірок
		алгоритми, що використовуються системою для порівняння документів
3.	Продуктивність роботи	час виконання перевірки (від завантаження до звіту про результат)
		форма представлення кінцевого результату (статистичний звіт)
		дані про результати пошуку
		особливості використання / інсталювання
		особливості інтерфейсу (інтуїтивна зрозумілість, наявність інструктивних та довідкових матеріалів).

Апробація досліджувальних ресурсів буде здійснюватися шляхом введення однакової частини тексту статті в спеціальне поле на сайті, завантаження файлу статті та за URL-адресою статті в журналі. Нижче наведені результати апробації наявних ресурсів, характеристики, та висновки щодо можливості їх використання в системі перевірки на плагіат університету.

1.3 Порівняльний аналіз програмного забезпечення та онлайн-ресурсів

Нижче коротко приведемо опис та отримані результати тестування кожного з розглянутих ресурсів.

Антиплагиат. На головній сторінці сайту цього сервісу (рис. 1.1) зразу надається можливість перевірити певний текст чи файл на унікальність [9]. Проте, доступ до перевірки стає доступним тільки після реєстрації або входу через одну з соціальних мереж.

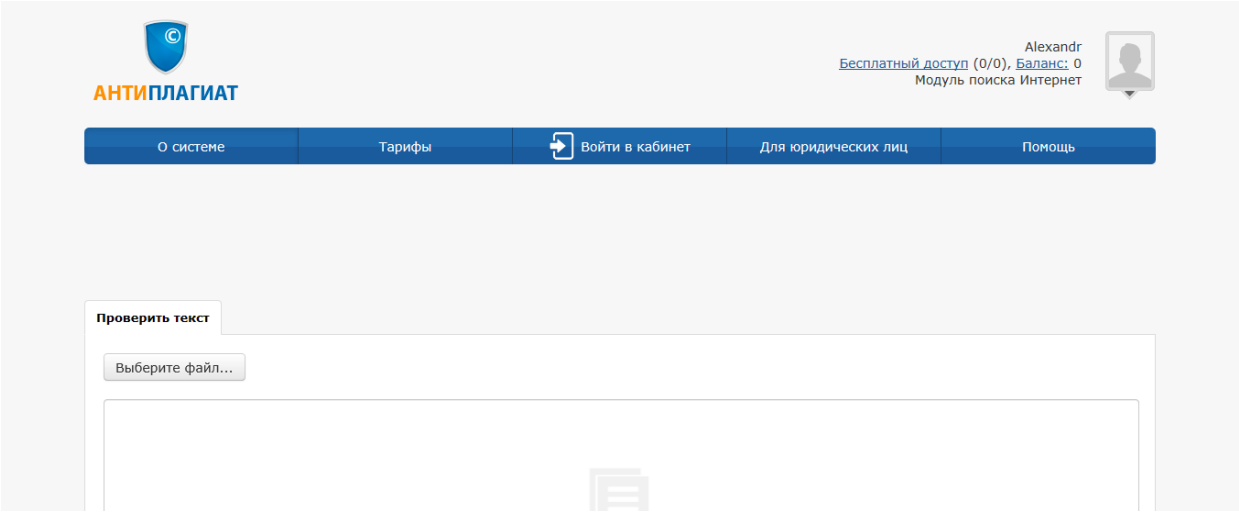

Рисунок 0.1 – Головна сторінка онлайн-ресурсу «АНТИПЛАГИАТ»

Після реєстрації та входу в систему стає доступною можливість пошуку плагіату в введеному тексті або вибраному файлі. Результат подається в вигляді звіту та відсотку унікальності документу. Звіт також

містить список доменів, на яких знайдені збіги, із визначеною часткою співпадання (%) по кожному домену та загальним показником унікальності всього тексту. За додаткові кошти можливий більш детальний звіт про результати пошуку.

Якщо вірити авторам сервісу, то для пошуку плагіату в них використовуються унікальні алгоритми власної розробки [9].

Пошук здійснюється по власній базі даних документів (текстів), базі даних електронної бібліотеки «Elibrary.ru» та за допомогою мережі Інтернет. Окрім цього, кожен ВУЗ чи компанія, що використовує цей сервіс має можливість наповнювати сервіс власною базою документів тим самим включаючи їх до перевірки.

Можливо здійснювати перевірку завантаженням файлів у форматах HTML, Plain Text (txt), DOC (MS Word), RTF, PDF, ZIP и RAR. Є обмеження: за раз можливо перевірити фрагмент тексту об'ємом до 5000 символів, що вимагає розбивки тексту на частини, підтримує перевірку файлів, необхідна реєстрація та передплата для зняття обмеженьта текстових фрагментів до 5000 символів.

Сайт сервісу «АНТИПЛАГИАТ» має наступні розділи:

1) «О системе» – надається інформація про систему, її роботу, можливості, авторів та володарів сервісу;

2) «Тарифы» – тарифи на послуги сервісу;

3) «Войти в кабинет» – перенаправляє до власного кабінету, в якому можна подивитись свою історію пошуку, звіти (результати) пошуку та баланс рахунку;

4) «Для юридических лиц» – розділ сайту, який розповідає про переваги сервісу для юридичних лиць;

5) «Помощь» – питання на відповіді на часто задаванні питання в роботі з системою «АНТИПЛАГИАТ».

Інтерфейс онлайн-ресурсу дещо незрозумілий одразу, але на сайті міститься інструкція щодо використання.

До недоліків системи слід віднести наступне:

– ліміт допустимого об’єму для текстових фрагментів (5 тис. символів);

– обмежена база даних (пошук запозичень здійснюється по російському сегменту мережі Інтернет);

– проблеми з підтримкою української мови;

– необхідність передплати.

Враховуючи всі зазначені недоліки, онлайн-ресурс «АНТИПЛАГИАТ» не рекомендується до використання у системі університету.

TEXT.RU. Ще один веб-сервіс для перевірки будь-яких текстів на плагіат, зображений на рис. 1.2. Так як і інші сервіси дає можливість перевірити тексти та сайти на унікальність [10].

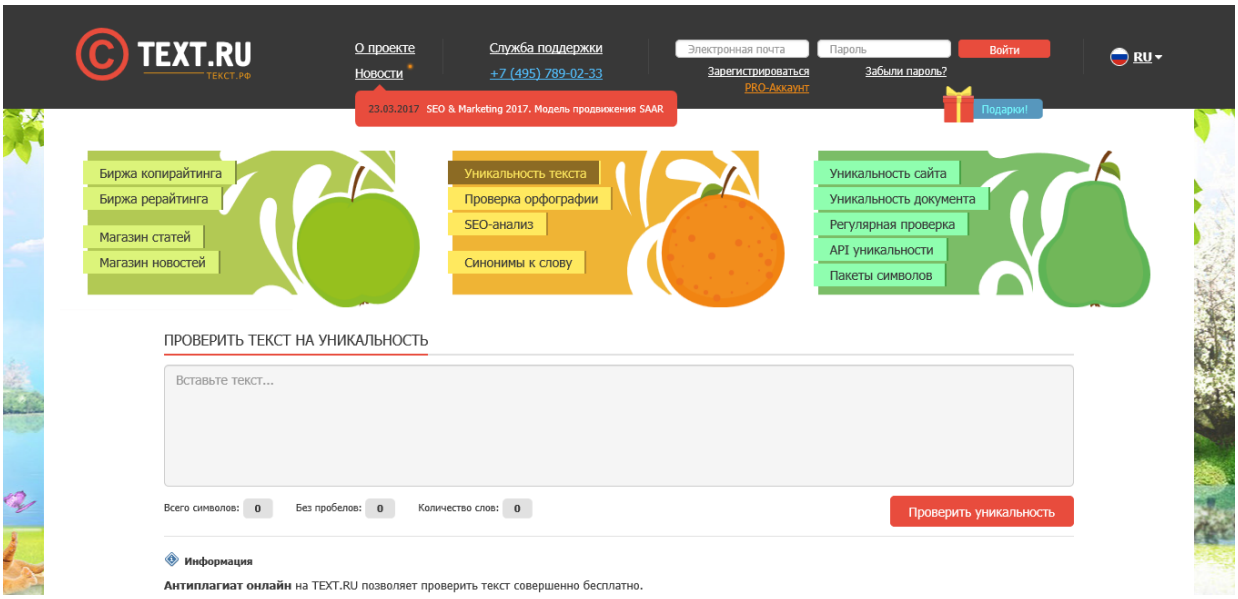

Рисунок 0.2 – Головна сторінка сайту сервісу «TEXT.RU»

TEXT.RU дозволяє перевіряти безкоштовно та без реєстрації, проте це займає деякий час, оскільки перевірка відбувається по черзі. Окрім перевірки на плагіат сервіс видає результати про перевірку на орфографічні помилки, SEO-аналіз тексту та кількість «води» в тексті. Яким чином знаходиться

«вода» в тексті – не зрозуміло. Підсумковий звіт містить список доменів, на яких знайдені збіги, із визначеною їх часткою (%) по кожному домену та загальним показником унікальності всього тексту.

Дозволяє розпізнавати плагіат, навіть якщо в тексті було здійснено перестановку слів і фраз, змінено відмінки, часи та інші граматичні категорії слова та додані нові слова. На сайті розміщено поле для вводу тексту, що перевіряється, допустимий обсяг якого від 100 до 15 000 символів. Є можливість запустити перевірку унікальності будь-якої кількості текстів.

Після реєстрації в системі стає доступною можливість перегляду історії пошуку, відкладеної перевірки (можна поставити текст на перевірку та закрити вкладку браузера), а також сама перевірка буде займати менше часу та стане доступною для перевірки більшої кількості текстів.

Сервіс також володіє власним API-інтерфейсом та дозволяє проводити регулярні перевірки текстів чи сайтів.

Сайт має наступні розділи:

1) «О проекте» – розповідається про «місію» проекту, відношення авторів до свого сервісу, його перспективи та інше;

2) «Новости» – різні новини сервісу, в основному такі, які стосуються самого сервісу «TEXT.RU» але іноді бувають і сторонні новини;

3) «Служба поддержки» – зворотній зв'язок з службою підтримки сервісу.

До недоліків системи можна віднести необхідність очікування в черзі та ліміт 15000 символів.

Враховуючи всі зазначені недоліки, онлайн-ресурс «TEXT.RU» не рекомендується до використання у системі університету.

Duplichecker. На головній сторінці даного ресурсу (рис.1.3) міститься поле для вводу текстового фрагменту з обмеженням до 1500 слів, а також поле для завантаження текстових файлів (максимум до 50 кБ) [11]. Підтримуються формати *.docx та *.txt. Кількість перевірок для незареєстрованих користувачів обмежена – 1 пошук/день. Реєстрація на сайті

дозволяє виконувати до 50 перевірок на день. Після реєстрації з'являється додаткове поле для вводу веб-посилання на необхідний файл, якщо він розміщений в мережі.

До недоліків системи можна віднести те, що ресурс не працює з текстовими файлами, ліміт на допустимий обсяг тексту складає 1500 слів.

Враховуючи всі зазначені недоліки, онлайн-ресурс «Duplichecker» не можна рекомендувати до використання у системі університету.

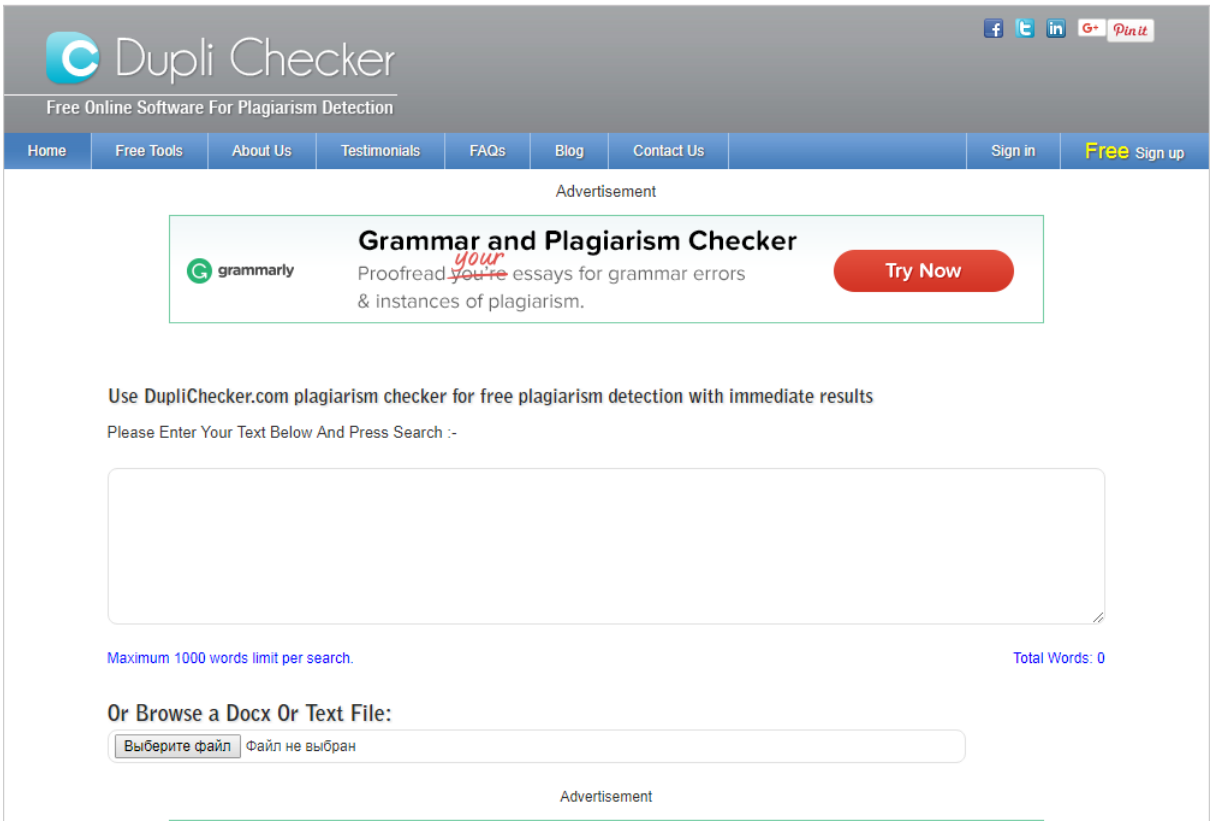

Рисунок 0.3 – Головна сторінка сайту сервісу «Duplichecker»

Unicheck. Онлайн-сервіс пошуку плагіату (рис.1.4), який перевіряє текстові документи на наявність запозичених частин тексту з відкритих джерел в Інтернеті чи внутрішньої бази документів користувача [12]. Є українською розробкою вітчизняної компанії «Антиплагіат». Система розкладає текст на окремі фрази та шукає збіг у режимі реального часу через Інтернет чи в документах з бібліотеки користувача, при цьому програма

розпізнає підміну символів в тексті (спосіб обману систем пошуку плагіату – заміна символів схожими символами з іншого алфавіту). Також Unplag визначає цитати та виноски, автоматично виключаючи їх зі звіту. Сервіс підтримує doc, docx,.rtf,.txt,.odt,.html та .pdf формати. Застосування інструменту для пошуку у внутрішній базі даних університету є безкоштовним. Плата вноситься лише за виявлення Інтернет-плагіату. Доступні мови: англійська, французька, українська, турецька, німецька, голландська.

Сервіс має зручний інтуїтивно зрозумілий інтерфейс і процедуру перевірки на плагіат. Розвинену технічну підтримку, представники компанії проводять безкоштовні семінари с працівниками університетів, де навчають роботі з сервісом.

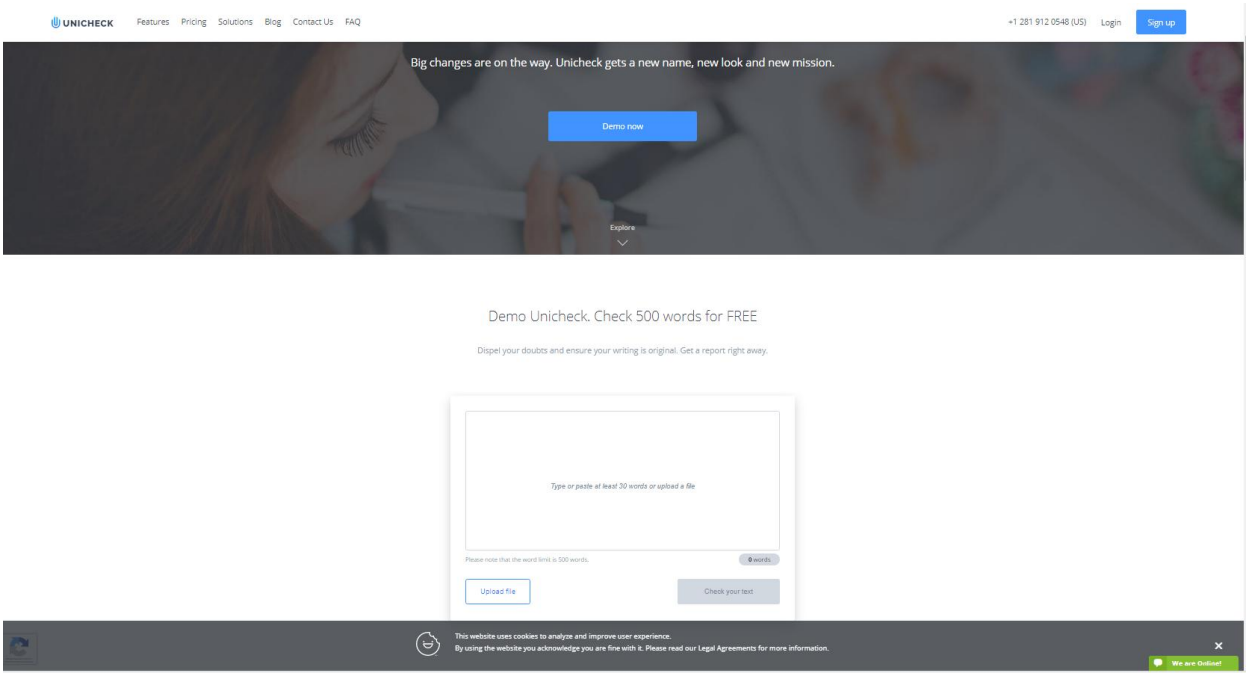

Рисунок 0.4 – Головна сторінка сайту сервісу «Unicheck»

Онлайн-сервіс активно впроваджується і успішно інтегрується у системи пошуку плагіату різних українських вищих закладів освіти. При

умові укладання договору між університетом і ТОВ «Антиплагіат», ресурс може бути інтегрований у систему університету.

StrikePlagiarism. Антіплагіатна інтернет-система польської компанії Plagiat.pl (рис.1.5), онлайн інструмент, створений для перевірки текстових документів [13]. Користувачі від вищого навчального закладу отримують індивідуальні облікові записи, захищені паролем.

StrikePlagiarism перевіряє документи у:

- внутрішній базі університету;
- базі даних документів інших вузів;
- всесвітніх веб-ресурсах.

Щоб користуватися базами даних інших вузів, необхідно підписати Декларацію про взаємний обмін базами даних. Система впроваджена більш ніж у 350 ВНЗ та видавництвах по всьому світу.

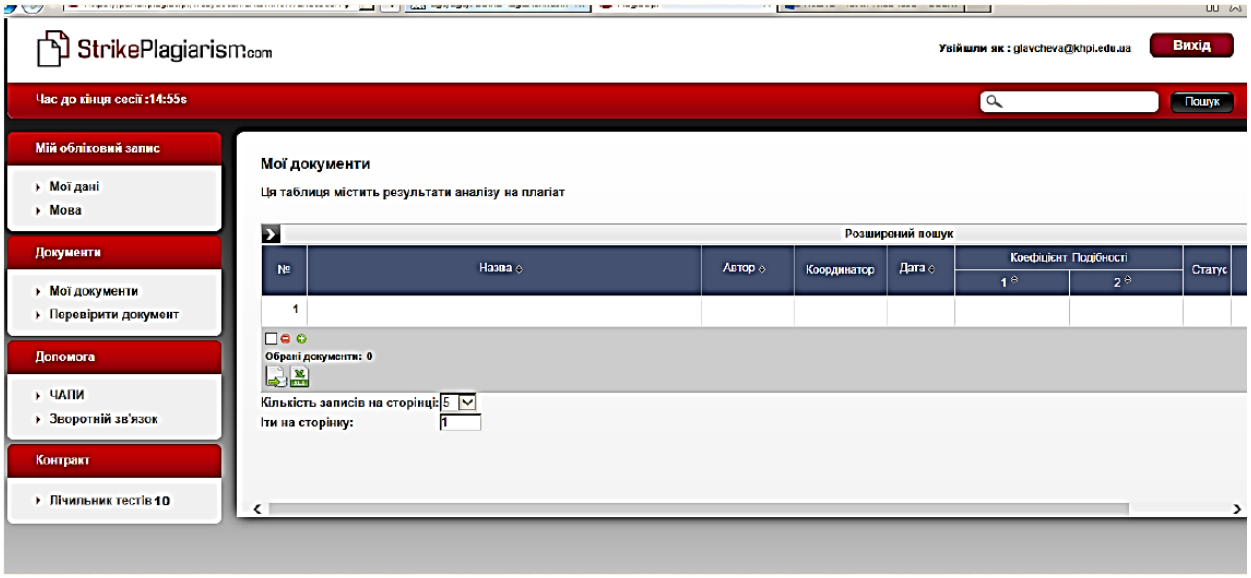

Рисунок 0.5 – Антіплагіатна інтернет-система «StrikePlagiarism»

При умові укладання договору між університетом і польською компанією «Plagiat.pl », ресурс може бути інтегрований у систему університету.

Advego Plagiatus. Програма розроблена компанією «Advego», яка займається поставкою різних послуг для веб-сайтів в інтернеті: копірайтинг, рерайтинг тощо. Проте, сама програма «Advego Plagiatus» (рис. 1.6), як стверджують автори, повністю безкоштовна [14].

Програма в своїй роботі використовує 3 пошукові системи: Yandex, Google, Rambler без використання власної бази документів (письмових робіт) [14].

Результатом роботи є посилання на сторінки, з яких, можливо, було скопійовано текст та відсотковий показник унікальності тексту, на основі якого робиться висновок про плагіат.

Програма має два режими роботи: «швидка перевірка», «глибока перевірка».

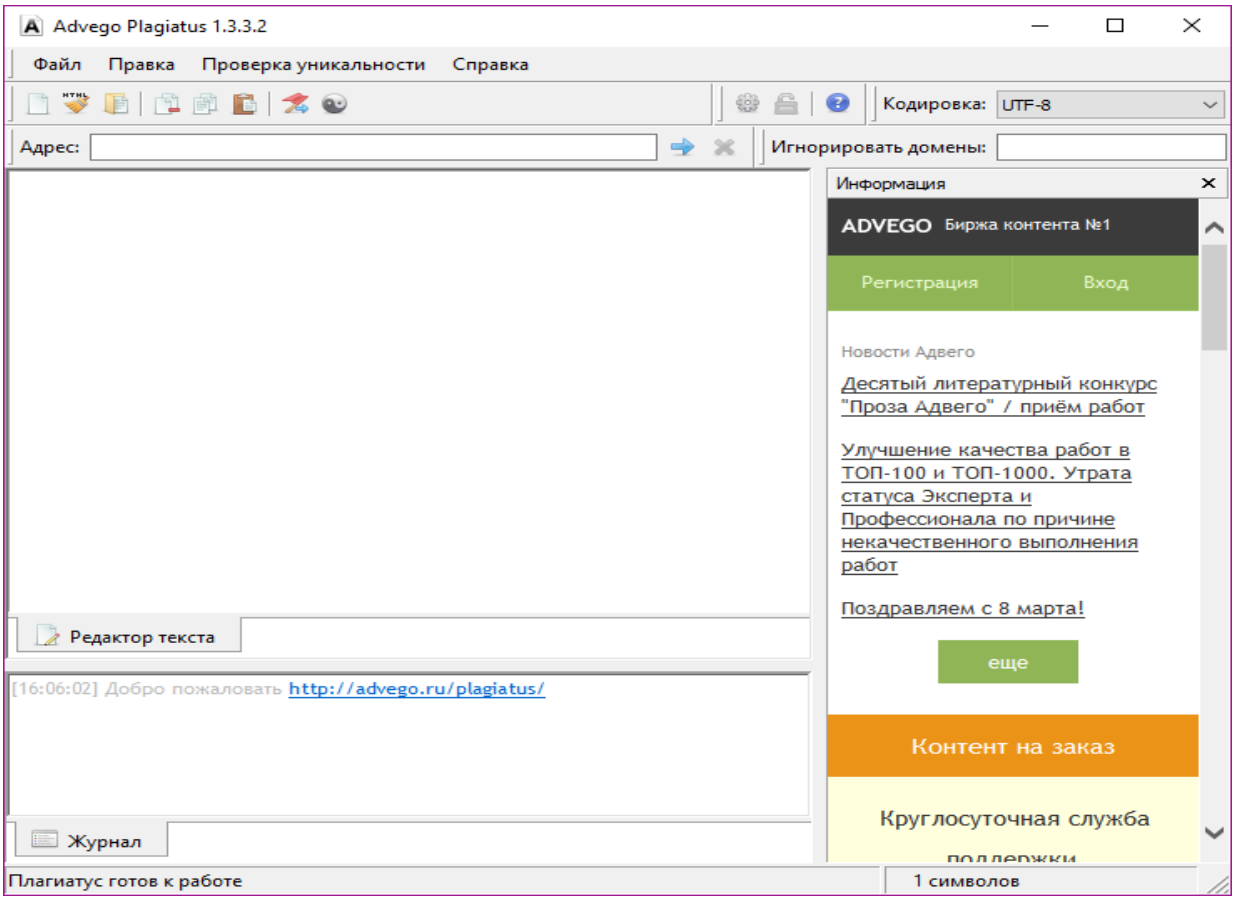

Рисунок 0.6 – Програма перевірки на плагіат «Advego Plagiatus»

При глибокій перевірці тексту результат є більш кращим, проте робота програми займає більше часу. В програмі немає можливості паралельної перевірки текстів та відсутня можливість завантаження для перевірки офісних файлів. При перевірці тексту є можливість ігнорування певних доменів.

До недоліків програми можна віднести те, що сканування, навіть у режимі швидкої перевірки, триває досить довго.

Враховуючи всі зазначені недоліки, програму «Advego Plagiatus» не можна рекомендувати для використання у підрозділах, що потребують перевірки великої кількості текстів.

Крім онлайн-сервісів, що описані вище, були досліджені ресурси: Plagiarisma.net, PlagScan, FindCopy (Miratools), Grammarly та інш. Всі ресурси мають певні недоліки та не можуть бути рекомендовані до використання в університетській системі перевірки на плагіат, або можуть бути використані частково. До основних недоліків даних ресурсів можна віднести обмеження на обсяг текст кількість перевірок.

1.4 Обґрунтування вибору програмних засобів для системи перевірки на плагіат творів працівників і студентів університету

Після проведеного аналізу існуючих аналогів можна виділити їх переваги та недоліки, і надати висновки щодо можливості інтегрування їх у систему університету. До недоліків даних сервісів можна віднести:

1) Більшою частиною наведені ресурси є комерційними продуктами, тому для детальної перевірки, повного спектру перевірок, більшої кількості джерел для перевірки, більшої кількості перевірок вони вимагають за це плату.

2) Більшість приведених аналогів не використовують власну локальну (внутрішню) базу даних для перевірки. Виняток складають ресурси

StrikePlagiarism і Unicheck (ТОВ «Антиплагіат»), при умові укладання договору.

3) Для перевірки робіт на плагіат всі ресурси потребують з'єднання з Інтернет.

4) Алгоритми, що використовують ресурси для перевірки на плагіат є закритими.

5) Деякі не є багатомовними, тобто підтримують лише одну мову текстового документу.

З урахуванням проведеного аналізу стає зрозумілим, що для впровадження в систему університету якісного, надійного програмного забезпечення з широкими можливостями підтримки як внутрішньої бази університету та і ресурсів Інтернет, є доцільним укладання договору на використання в підрозділах університету комерційного ресурсу. На наш погляд, перевагу слід віддати вітчизняному програмному забезпеченню Unicheck (ТОВ «Антиплагіат»). Прийнятним варіантом може бути програмне забезпечення StrikePlagiarism (Plagiat.pl). У розділі 3 НДР розроблені процедури перевірки на плагіат наукових публікацій працівників і студентів університету і дано методичні рекомендації щодо практичного застосування цих програмних засобів структурними підрозділами університету.

Відповідно до перелічених недоліків також постала необхідність в розробці власної локальної системи пошуку плагіату в студентських та наукових роботах, яка б була безкоштовною, мала можливість додавання необхідних документів до власної бази даних та була відкритою (Open Source). При умові її успішної апробації вона могла би бути резервним ресурсом перевірки на плагіат. В розділі 2 НДР наведені етапи її розроблення.

2 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АВТОМАТИЧНОГО ВИЯВЛЕННЯ ПЛАГІАТУ

2.1 Обґрунтування вибору програмних засобів для реалізації системи

2.1.1 Вибір мови програмування

Вибір мови програмування є досить важливим етапом, оскільки в залежності від мови програмування будуть змінюватись такі атрибути програми як:

− швидкість роботи програми;

− кросплатформеність (багатоплатформність) – можливість запуску програми під управлінням різних операційних систем;

− споживання програмою ресурсів, оскільки середовище запуску деяких програм, написаних на певних мовах програмування, по різному розподіляє і очищає ОЗП, а в деяких випадках задача контролю ОЗП лежить на програмістові.

Для розробки інформаційної системи оберемо одну з об’єктно-орієнтованих мов програмування високого рівня: java, php, c#.

Java – це об’єктно-орієнтована мова програмування з строгою типізацією, випущена в 1995 році компанією «Sun Microsystems». З 2009 року і по сьогоднішній день розробкою та підтримкою мовою займається компанія «Oracle», яка окрім неї займається ще й розробкою баз даних та інших програмних рішень [15]. Код, написаний на Java після компіляції перетворюється в так званий байт-код, який при виконанні інтерпретується віртуальною Java-машиною (JVM – Java Virtual Machine) для конкретної платформи [15]. Завдяки тому, що програми, написані на Java виконуються під управлінням JVM вони і є кросплатформеними, проте через це програми, написані на Java, виконуються повільніше. Проте, слід відмітити, що результати нових нещодавніх тестів швидкодії Java-програм не поступаються

в швидкості програмам, написаним на інших мовах програмування. Управлінням ОЗП в Java-програмах керує JVM, а тому можливе значне виділення ресурсів ОЗП при роботі Java-програм.

Переваги мови програмування Java: кросплатформеність; наявність великої кількості framework’s для рішення поставленої задачі; наявність «розумних» та зручних IDE (IntelliJ IDEA, NetBeans).

Недоліки мови програмування Java: недостатньо зручний та гнучкий framework для побудови GUI– Swing, а також майже відсутні аналоги (AWT – застарілий попередник бібліотеки Swing, а JavaFX майже немає документації); дуже «багатослівна» мова програмування в порівнянні з більшістю інших мов; відсутність багатьох синтаксичних особливостей, які наявні в інших мовах програмування.

Python – це інтерпретована об’єктно-орієнтована мова програмування з динамічною типізацією. Була розроблена ще в 1990 році Гвідо ван Россумом. Входить до всіх дистрибутивів Linux [16]. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. Це забезпечує мові Python кросплатформеність і швидкодію програм, написаних на ній [17]. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована [17]. Дизайн мови Python побудований навколо об'єктно-орієнтованої моделі програмування. Реалізація ООП в Python є елегантною, потужною та добре продуманою, але разом з тим, достатньо специфічною в порівнянні з іншими об'єктно-орієнтованими мовами.

Мова Python має деякі особливості:

− множинне успадкування;

− спеціальні методи, що керують життєвим циклом об’єкта: конструктори, деструктори, розподільники пам’яті (аналогічно як і в C++);

− властивості (імітація поля за допомогою функцій, аналогічно як і в C#);

− класи, вкладені у функції.

Інтерпретатор Python має інтерактивний режим роботи, при якому введені з клавіатури вирази відразу ж виконуються, а результат виводиться на екран. Для Python створено не так багато IDE, як для деяких інших мов програмування (хоча, для скриптових, інтерпретованих мов програмування зазвичай існує дуже невелика кількість IDE). Дуже часто для програмування на Python використовують розширений текстовий редактор (наприклад: Sublime Text, ATOM, Brackets).

Переваги мови програмування Python:

1) кросплатформеність;
2) швидкодія програм;
3) наявність інтерактивного інтерпретатору та «малослівності» самої мови програмування;
4) широка підтримку майже усіма операційними системами;
5) велика кількість різних бібліотек для рішення поставленої задачі.

Недоліки мови програмування Python:

1) відсутність «розумної» та зручної IDE;
2) важкість освоєння самої мови;
3) не сама вдала реалізація бібліотеки побудови GUI, а без RAD (Rapid Application Development – середовище швидкої розробки додатків) системи, яка є звичайним явищем в багатьох IDE, – написання графічних додатків є досить важким зайняттям.

C# (C Sharp, Сі шарп) – об'єктно-орієнтована мова програмування з безпечною (строгою) системою типізації даних для платформи .NET. Розроблена під егідою Microsoft Research [18]. Має близький синтаксис до Java та C++ мов програмування. Та на відміну від Java має багато зручних особливостей, які дозволяють краще, зрозуміліше та легше читати та писати код програми.

Як і в Java, програми написані на C#, транслюються в байт-код та виконуються під управлінням віртуальної машини. Аналогом JVM в мові C#

є CLR (Common Language Runtime – загальномовне виконуване середовище). Аналогічно до Java, виділенням та управлінням ОЗП займається віртуальна машина [18].

Зазвичай для розробки додатків на мові C# використовують IDE Visual Studio. Вона є дуже потужним засобом, є можливість встановити багато додатків до IDE (плагінів), в тому числі й комерційних (наприклад, ReSharper від компанії JetBrains). Наразі в Visual Studio за допомогою мови програмування C# можна розробляти будь-який вид додатків, ось деякі з них:

1) веб-сервіси (ASP .Net MVC, ASP .Net WebForms тощо);
2) хмарові сервіси (Microsoft Azure);
3) служби Windows;
4) кросплатформені мобільні додатки (Xamarin) та просто мобільні додатки на базі ОС Windows (WinRT, UWP);
5) десктопні додатки (WinForms, WPF, WinRT, UWP).

Переваги мови програмування C#:

− динамічне присвоєння типів (за допомогою зарезервованого слова var, або dynamic). Значно покращує читання коду і зменшує кількість написаного, особливо в тих місцях де відбувається великий ланцюжок викликів;

− методи розширення (extension methods) – розширення функціоналу існуючих класів без створення і успадкування чи композиції нових. Фактично це вбудований в мову програмування шаблон «Декоратор»;

− властивості – засіб для підтримки інкапсуляції без необхідності писати методи (гетери та сетери). Звернення до властивостей йде через методи доступу, хоча самі ці методи можна й не реалізовувати;

− підтримка асинхронного програмування самою бібліотекою .NET (зарезервовані слова async, await);

− індексатори – за допомогою них можна звертатись до об’єктів класу як до масиву, не приводячи їх до масиву, як в Java відбувається з типом String.

− делегати – спосіб звернення до методу, як до змінної класу чи об’єкту;

− LINQ – додавання синтаксису мови запитів (SQL) до мови програмування C#. Дозволяє значно скоротити деякі блоки коду та покращити читання цього самого коду;

− розумна та зручна IDE – Visual Studio;

− дуже зручна побудова GUI.

Недоліки мови програмування C#:

− відсутність кросплатформеності (частково, існує проект Mono, який дозволяє писати і виконувати код C# під управлінням сімейств операційних систем, відмінних від сімейства ОС Windows, проте і з деякими обмеженнями);

− більш повільніше виконання програм, особливо в порівнянні з програмами, написаними інтерпретованими мовами (Python, JavaScript, Lisp, та ін.).

Інформаційна система пошуку плагіату буде реалізована на мові програмування C#, оскільки вона є найбільш оптимальною для програмної розробки під управлінням ОС Windows. При цьому перенесення системи на будь який інший тип систем (веб-система, хмаровий сервіс тощо), при використанні мови програмування C# не є складним завданням.

2.1.2 Вибір системи керування базами даних

На даний момент існують два основних класи баз даних: реляційні та нереляційні. Перші використовують в роботі мову SQL (Structured Query Language – мова структурованих запитів) та роблять опір на відносини (relations, звідси і назва – реляційні), а інші використовують власну реалізацію CRUD (Create, Read, Update, Delete) функцій, та ще носять назву NoSQL бази даних.

Розглянемо по одній найвідомішій та найпоширенішій базі даних з різних класів СКБД:

- PostgreSQL (належить до реляційних СКБД);
- MongoDB (належить до нереляційних (NoSQL) СКБД).

PostgreSQL – вільна об’єктно-реляційна СКБД. Існує в реалізації для більшості UNIX-подібних систем, а також для сімейства ОС Windows. Базується на мові SQL та підтримує більшість можливостей стандарту SQL:2011[19]. Оскільки PostgreSQL базується на мові SQL, то вона і являється строго типізованою СКБД, тобто, при проектуванні бази даних повинна дотримуватись строга схема таблиць з певними типами даних, які зберігаються в полях таблиці. Серед реляційних баз даних славиться значною швидкістю, надійними механізмами транзакцій, реплікації та є легко розширюваною. Для роботи з цією СКБД потрібно розвернути власне сервер PostgreSQL, а для роботи з клієнтського додатку можна скористатись або консолью, або одним з графічних додатків, наприклад PG Lightning Admin.

Переваги СКБД PostgreSQL:

- для роботи з даними використовується відома багатьом мова SQL;
- витримує великі навантаження на сервер PostgreSQL;
- здатна вмістити в себе дуже велику кількість даних;
- використання відношень (relations) полегшує роботу з структурованими даними.

Недоліки СКБД PostgreSQL:

- необхідність дотримуватись строгої структури таблиць та даних;
- мала швидкість роботи з вставкою/читанням даних в порівнянні з NoSQL рішеннями СКБД;
- використання відношень (relations) погіршує роботу з безструктурними даними.

MongoDB – документо-орієнтована система керування базами даних з відкритим вихідним кодом (open source), яка не потребує опису колекцій

(аналог таблиць в реляційних СКБД) [20]. Код MongoDB написаний на C++[21].

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій. Розширення кластера або перетворення одного сервера в кластер проводиться без зупинки роботи БД простим додаванням нових машин [20].

При розробці автори виходили з необхідності спеціалізації баз даних, завдяки чому їм вдалося відійти від принципу «один розмір під усе». За рахунок мінімізації семантики для роботи з транзакціями з'являється можливість вирішення цілого ряду проблем, пов'язаних з нестачею продуктивності, причому горизонтальне масштабування стає простішим. Використовувана модель документів зберігання даних (JSON/BSON) простіше кодується, простіше управляється (у тому числі за рахунок застосування так званого «без схемного стилю» (англ. schemaless style), а внутрішнє угруповання релевантних даних забезпечує додатковий виграш в швидкодії [20].

Для роботи з MongoDB потрібно розвернути на деякій машині (локальній – localhost, чи видаленій – remote server) сам сервер MongoDB, а для роботи на стороні клієнта можна використати консоль Mongo, або певну клієнтську програму, наприклад Robomongo.

Основні можливості та переваги MongoDB:

− документо-орієнтоване сховище (проста та потужна JSON-подібна схема даних);

− досить гнучка мова для формування запитів;

− динамічні запити;

− повна підтримка індексів;

− профілювання запитів;

− швидкі оновлення «на місці»;

− ефективне зберігання двійкових даних великих обсягів, наприклад, фото та відео;

− журналювання операцій, що модифікують дані в БД;

− підтримка відмовостійкості і масштабованості: асинхронна реплікація, набір реплік і шардінг;

− може працювати відповідно до парадигми Map/Reduce.

Недоліки MongoDB:

− кожна NoSQL СКБД підтримує унікальну, специфічну мову запитів, яку потрібно буде вивчати кожен раз заново. В тому числі це відноситься і до MongoDB;

− при використанні NoSQL СКБД закривається один вид вразливостей (SQL-injection, наприклад) та відкриваються нові: ін’єкції в регулярних виразах, JSON ін’єкції, JavaScript ін’єкції і т. п.

Оскільки дані, які будуть використані в інформаційній системі не мають чіткою структури, а швидкість доступу до них та гнучкість роботи з ними грають важливу роль в роботі системи пошуку плагіату, то в якості СКБД для системи пошуку плагіату буде використана MongoDB.

2.2 Вибір алгоритму перевірки текстових документів на плагіат

Існує безліч алгоритмів для перевірки текстів на плагіат. Досить детальний огляд наведено в роботі [22]:

− метод зрівняння хеш-сумм – для документів (текстів) знаходиться хеш-сумма, а потім порівнюється. Працює тільки для 100% подібних текстів;

− TF*IDF – будується словник частотно вживаних слів, а потім на його основі будуються вектори текстів, які можна потім зрівняти різними векторними методами [23,24];

− Long Sent – документ розбивають на речення, які впорядковуються по убуванню довжини, а при рівності довжини – в алфавітному порядку.

Потім вибираються і сціплюються в строчку в алфавітному порядку 2 найбільших речення;

− Megashingles – заснований на представленні текстів у вигляді множини послідовностей фіксованої довжини, що складаються із сусідніх слів. При значному перетині таких множин документи будуть схожі один на одного;

− та інші.

На сьогоднішній день основними алгоритмами при роботі з пошуком довільного тексту (або знаходження схожостей в текстах) є алгоритми TF*IDF та шинглів, їх і розглянемо.

2.2.1 Алгоритм TF*IDF

TF-IDF (від англ. TF – term frequency, IDF – inverse document frequency) – статистична міра, яка використовується для оцінки важливості слова в контексті документа, що є частиною колекції документів або корпусу. Вага деякого слова пропорційний кількості вживання цього слова в документі, і обернено пропорційна частоті вживання слова в інших документах колекції [23].

Вагу широковживаних слів можна скоротити при обліку IDF, тобто міра TF-IDF складається з перемноження TF і IDF. В мірі TF-IDF більша вага буде у тих слів, які часто використовувалися в одному документі і рідше – в іншому [25,26].

Відповідно, при обчисленні ваги для кожного терма, алгоритм використовує формулу TF * IDF. Після цього в рядок в алфавітному порядку упорядковано 6 слів, які мають найбільше значення ваги. Контрольна сума CRC32 отриманого рядка обчислюється як сигнатури документа.

Можна TF * IDF алгоритм використовувати в купі з іншими методами. Наприклад, за допомогою цього алгоритму вибрати «важливі» слова з порівнюваних документів, а потім за допомогою «Відстані Левенштейна»

перевіряти їх на схожість. Чим меншим буде значення «Відстані Левенштейна» – тим більше це буде вказувати на «вкрадений» рядок, а в купі з Стеммером Портера [27] результат буде виглядати ще більш точним. Або, після знаходження значень TF*IDF документу побудувати рівні вектори, а потім знайти їхню косинусну подібність, яка буде вказувати на їх схожість [25,26].

Відстань Левенштейна (також редакційне відстань або дистанція редагування) між двома рядками в теорії інформації та комп'ютерної лінгвістики – це мінімальна кількість операцій вставки одного символу, видалення одного символу і заміни одного символу на інший, необхідних для перетворення одного рядка в іншу [27].

2.2.2 Алгоритм шинглів

Алгоритм шинглів (від англ. Shingles – лусочки) – алгоритм, розроблений для пошуку копій і дублікатів розглянутого тексту в документі. Інструмент (алгоритм) для виявлення плагіату [28].

Етапи, які проходить текст, що піддався порівнянню [29]:

1) канонізація тексту;
2) розбиття на шингли;
3) обчислення хеш шинглів;
4) випадкова вибірка 84 значень контрольних сум;
5) порівняння, визначення результату.

Канонізація тексту приводить оригінальний текст до єдиної нормальної форми. Текст очищається від прийменників, сполучників, знаків пунктуації, HTML-тегів та іншого непотрібного «сміття», яке не повинно брати участь в порівнянні. У більшості випадків також пропонується видаляти з тексту прикметники, так як вони не несуть смислового навантаження [30].

Також на етапі канонізації тексту можна приводити іменники до називному відмінку, єдиному числа, або залишати від них тільки корінь.

Шингл – це фрагмент тексту довжиною в кілька слів, з яким працюють різні програми перевірки унікальності. За допомогою шинглів можна практично безпомилково перевірити текст на унікальність, аж до синонімізованого тексту. Найбільш поширена довжина шинглу, використовувана в більшості сервісів пошуку плагіату знаходиться в діапазоні від 3 до 6 слів. Такі сервіси працюють приблизно в такий спосіб: вони розбивають текст на окремі елементи – фіксовані ділянки слів (задані шинглом – 3, 4 або 5), і потім знаходять ці фрагменти в інших текстах Інтернету або в локальній (внутрішній) базі [31,32].

Зрозуміло, навіщо потрібна така детальна перевірка унікальності. Неунікальні тексти мають меншу важливість в пошукових системах, а дві статті, що відрізняються одна від одної 3 реченнями – аж ніяк не унікальні. Для того щоб виявити такі статті-копіпаст якраз і використовується метод шинглів.

На другому етапі текст ділитися на шингли заданого розміру. Чим коротше шингл, тим менше буде унікальність тексту і точніше результат перевірки. А третій етап присвячений порівнюванню шинглів одного тексту з шинглами іншого.

Отже, унікальність тексту безпосередньо залежить від довжини шинглу. Якщо задати в налаштуваннях шингл рівним 1, то навряд чи в Інтернеті можна знайти текст, в якому не зустрічається хоч одне згадане слово даної статті. Унікальність в даному випадку буде дорівнювати нулю. А якщо поставити шингл 9, то унікальність вашої статті різко зросте, так як в інтернеті практично нереально знайти два однакових тексту, що містять ідентичні фрагменти з 9 слів, правда, якщо це не відвертий плагіат [29].

Розглянуті алгоритми пошуку плагіату дають практично однаковий результат, проте алгоритм TF*IDF легше реалізувати, тому був вибраний саме він.

2.3 Моделювання інформаційної системи

2.3.1 Загальні вимоги до інформаційної системи

Інформаційна система пошуку плагіату повинна виконувати ряд функцій, аби вона вважалась придатною до застосування користувачем, в не залежності від виду користувача (будь то учень, чи викладач, чи сторонній користувач). Це такі функції як:

1) додавання існуючих чи нових документів (студентських чи наукових робіт), які пройшли перевірку на плагіат до локальної (внутрішньої) бази даних програми, для забезпечення більш точної перевірки на плагіат нових документів (студентських чи наукових робіт);
2) введення нових, неперевірених на плагіат документів в чергу перевірки програмою їх на плагіат;
3) власне сама перевірка на плагіат;
4) видача результатів перевірки;
5) можливість зберегти отримані результати:
   - в печатній формі;
   - в файловій системі в вигляді текстового файлу;
   - в буфері обміну (наприклад, коли потрібно зберегти інформацію десь в іншому місці);
6) очистити результат перевірки робіт.

При розробці системи пошуку плагіату в студентських та/чи в наукових роботах було визначено чітке розмежування складових програмних засобів для правильного функціонування роботи:

- програмний засіб для додавання документів (наукових чи студентських робіт), які пройшли перевірку на плагіат до локальної (внутрішньої) бази даних – «DocumentAdder»;
- програмний засіб для перевірки певного документа на плагіат – «PlagiarismDetector».

Перед тим, як перейти до опису структури бази даних слід визначити, які дані потрібно зберігати в ній, а також зв’язки між колекціями (аналог таблиць в реляційних базах даних). В базі даних потрібно зберігати інформацію про документи, файли цих документів, логи (англ. *Logs*) програми та IDF-вектор, необхідний для перевірки на плагіат.

Інформація про документи буде тісно пов’язана з інформацією про файли, оскільки документи будуть (були) отримані з файлу.

2.3.2 Проектування бази даних системи

Архітектура системи пошуку плагіату в наукових та студентських роботах передбачає, що база даних може бути розміщена як на локальній машині так і на видаленому (remote) сервері. Програмні засоби, реалізовані в системі, формують і відсилають запити до віддаленого серверу БД, який забезпечує виконання запиту і повертає їм результат.

В нереляційних СКБД, на відміну від реляційних СКБД, структура самої СКБД відрізняється, проте, не кардинально. Структура даних в нереляційних СКБД не регламентована (іншими словами – слабко типована), тобто в окремій строчці чи документі можна добавити довільне поле без попереднього декларування змін структури всієї таблиці. Тому, коли з’являється необхідність змінити модель даних, то єдина достатня дія – відобразити зміни в коді додатку.

Визначивши функціональні вимоги до системи пошуку плагіату, описані в підрозділі 2.1 виконаємо проектування базм даних за допомогою СКБД MongoDB. Структура бази даних виглядає наступним чином:

1) Колекція документів:
   - ID документу;
   - хеш-сума документу;
   - автор документу;

- навчальна група (кафедра, факультет) автору документу;
- всі слова документу;
- дата додавання документу;
- TF-вектор документу;
- ID файлу документу (фактично, це посилання на відповідне поле в колекції файлів).

2) Колекція файлів:
   - ID файлу;
   - ім’я файлу;
   - шлях до файлу;
   - розширення файлу (наприклад, *.doc, *.docx, *.rtf, *.otd тощо).
3) Колекція логів:
   - ID логу;
   - повідомлення логу;
   - дата створення логу;
   - тип логу (повідомлення, інформація, помилка).
4) Колекція IDF значень слів:
   - ID IDF-значення;
   - саме слово;
   - IDF-значення слова.

А оскільки в якості СКБД була вибрана MongoDB, то при необхідності, в майбутньому можна буде з легкістю змінити структуру. Схема колекцій та полів бази даних зображена на рис. 2.1.

Розглянемо функціональні можливості користувачів системи пошуку плагіату.

Оскільки в системі пошуку плагіату немає функціональної частини, яка б могла стати перешкодою в користуванні іншим користувачам системи, то розмежувань прав доступу чи розмежування користувачів немає. Будь хто може користуватись системою в рівних умовах.

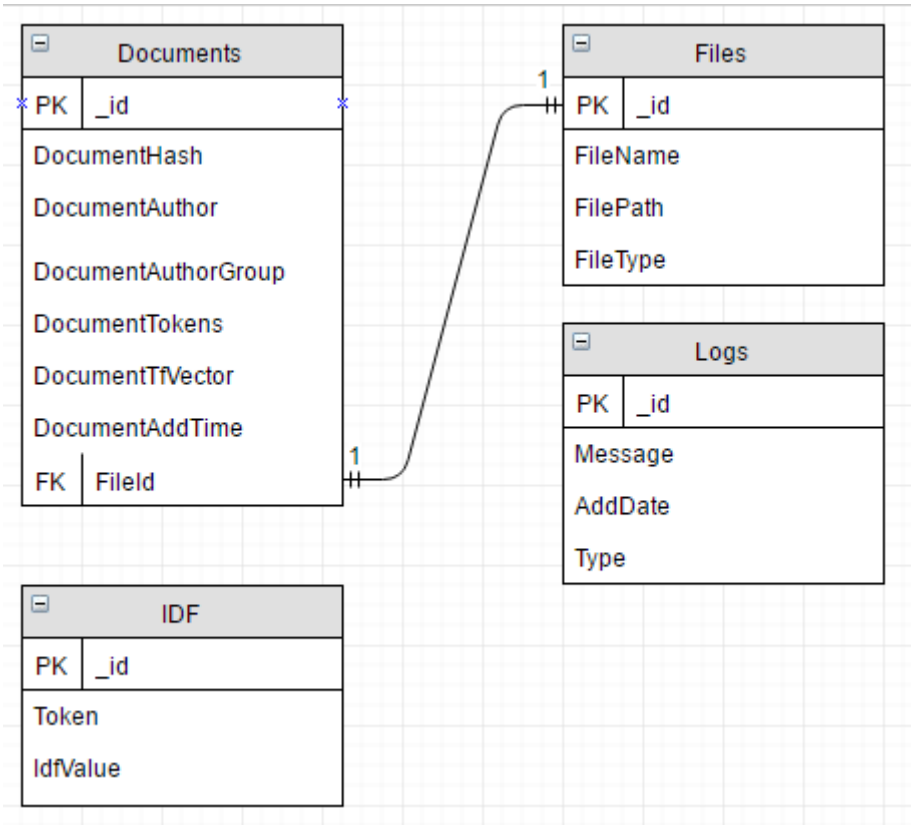

Рисунок 2.1 – Схема колекцій та їх полів в базі даних

Загалом, система може бути корисною для викладачів, студентів, аспірантів, тобто в освітній сфері. Проте, може бути використана й у науковій сфері, особливо, якщо існує певна база існуючих наукових робіт.

Власне для користувачів системи цікавим буде друга програма – «PlagiarismDetector», оскільки вона надає такі функції як:

1) введення нових, неперевірених на плагіат документів в чергу перевірки програмою їх на плагіат;

2) власне сама перевірка на плагіат;

3) видача результатів перевірки;

4) можливість детального перегляду результатів (з якими документами були знайдені збіги, та можливість відкрити їх в провіднику Windows);

5) можливість зберегти отримані результати:

а) в печатній формі;

б) в файловій системі в вигляді текстового файлу;

в) в буфері обміну (наприклад, коли потрібно зберегти інформацію десь в іншому місці);

6) очистити результат перевірки робіт.

Для обслуговування системи (поповнення її інформацією) можна виділити окремий тип користувачів – Адміністратор. Ними можуть бути як викладачі, так і довірені студенти – це вже може вирішити сам власник системи, кому довіряти додавання даних, які будуть в локальній (внутрішній) базі.

2.3.3 Проектування роботи програмних частин системи

Мінімальним набором для функціонування будь-якої системи, будь то веб-система, чи система пошуку плагіату, необхідні певні дані, сховище для цих даних та програмні засоби, які їх оброблюють та повертають результат користувачеві.

Перш ніж приступити до проектування ПЗ системи потрібно визначити, якими даними буде оперувати система та в якому типі сховища вони будуть зберігатись.

В загальному вигляді даними для системи пошуку плагіату в студентських та наукових будуть:

1) перевірені документи (студентські чи наукові роботи) – документи, які мають певні причини бути поміщеними в локальну (внутрішню) базу даних системи, з якими потім система буде порівнювати роботи, та на основі яких буде повертати результат (наприклад: реферати, дипломні роботи, курсові роботи, дисертації і т.п., які були здані та перевірені викладачами раніше);

2) вхідні дані – документи (студентські чи наукові роботи), які користувач хоче перевірити на плагіат (наприклад: реферати, курсові роботи, дипломні роботи, дисертації і т.п., які здаються користувачем вперше);

3) вихідні дані – результат роботи програми пошуку плагіату. Вихідні дані можуть мати різний вигляд, в залежності від потреб користувача.

Після визначення якими даними оперує система слід приступити до визначення СКБД, в якій вони будуть зберігатись. Вибір СКБД для системи пошуку плагіату в студентських та наукових роботах був визначений в підрозділі 2.1.2.

Після того, як були визначені дані, якими оперує система та СКБД, в якій вони будуть зберігатись, можна перейти до проектування програмних додатків, завдяки яким система буде функціонувати.

Схематична робота системи представлена на рис. 2.2.

2.3.4 Проектування підсистеми додавання документів в локальну (внутрішню) базу даних системи

Контекстна діаграма програми для додавання документів в базу даних – «DocumentAdder» – зображена на рис. 2.3. В ній описані лише загальні дані, які необхідні програмі для роботи, чинники, що мають вплив на роботу програми, та результат який має повертати програма.

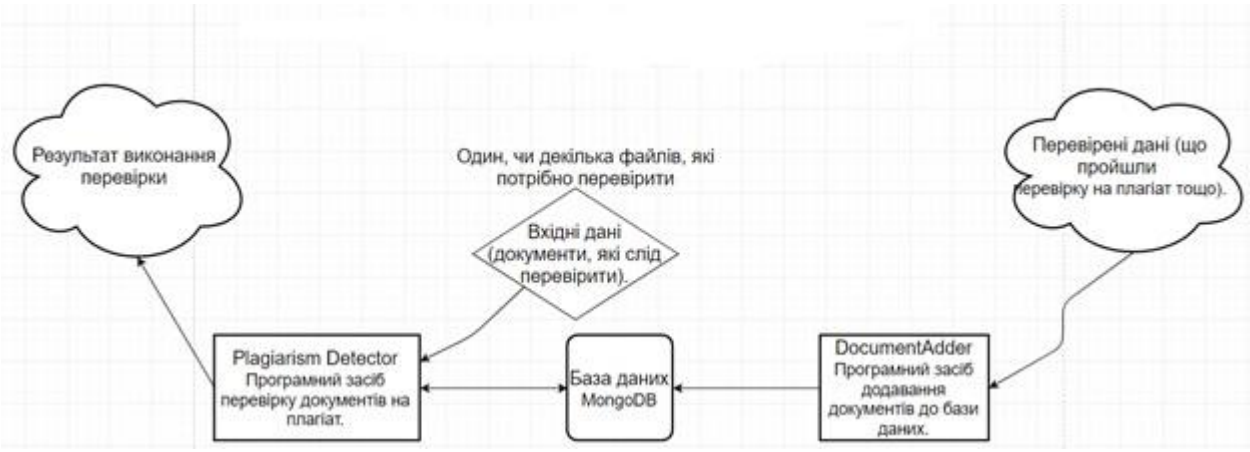

Рисунок 2.2 – Схематична робота системи

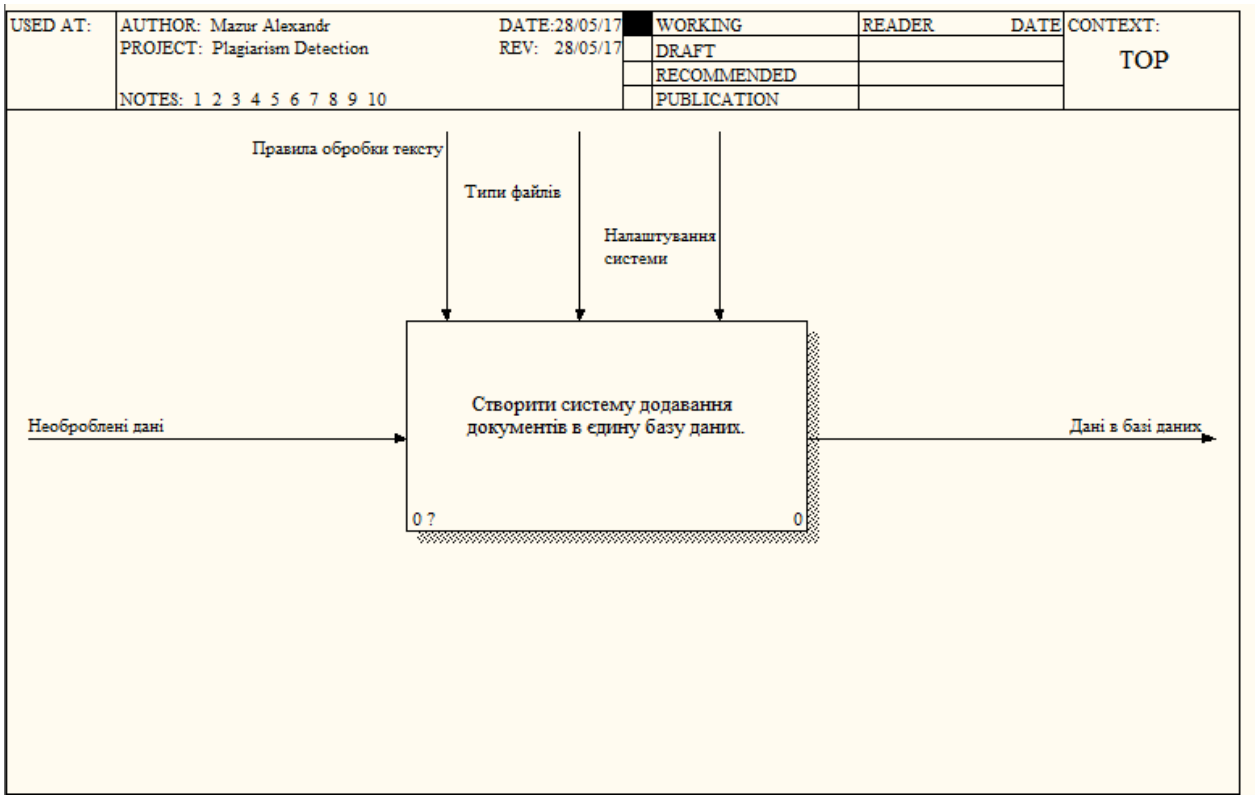

Рисунок 2.3 – Контекстна діаграма програми додавання даних в локальну (внутрішню) базу даних – «DocumentAdder»

Наведена контекстна діаграма на рис. 2.3 не несе інформації щодо того, як повинно програмне забезпечення «DocumentAdder» працювати всередині. Тому, для більш детального розгляду системи з середини потрібно зробити декомпозицію наведеної контекстної діаграми (рис. 2.3). На рис. 2.4 показана декомпозиція контекстної діаграми програми додавання даних в базу даних. Декомпозиція дає більш детальнішу інформацію про етапи роботи, які повинна пройти програма для виконання своєї функції:

1) обробка (нормалізація) тексту документу;

2) логування подій, які виникали під час роботи;

3) занесення нормалізованих (приведених до необхідного вигляду) даних до бази даних.

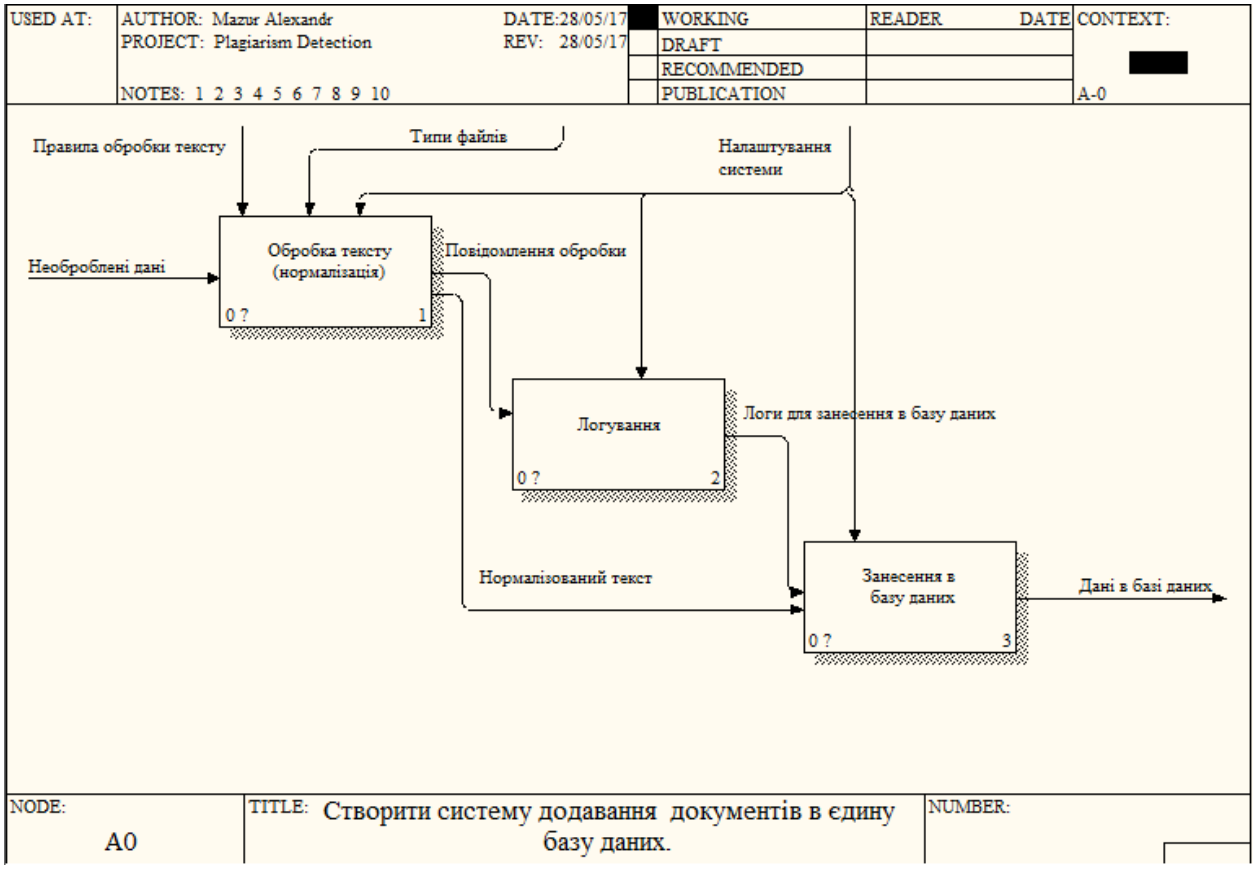

Рисунок 2.4 – Декомпозиція контекстної діаграми системи додавання документів в локальну (внутрішню) базу даних – «DocumentAdder»

Проте, для повного розуміння того, як має функціонувати дана програма цього рівня деталізації не достатньо, а тому, потрібно зробити декомпозицію кожного кроку.

Декомпозиція першого кроку роботи програми (Обробка (нормалізація) тексту) показано на рис. 2.5.

Завдяки цьому етапові в проектуванні роботи програми стає зрозумілим, що для досягнення цілі (в даному випадку це нормалізація документу), потрібно пройти 5 кроків:

1) перевірка підключення до серверу бази даних MongoDB;
2) перевірка, чи є вже такий документ в базі даних;
3) власне нормалізація даних;

4) формування документу (мається на увазі формування даних придатних для вставку їх в БД);

5) формування повідомлення для логування виконання роботи ПЗ.

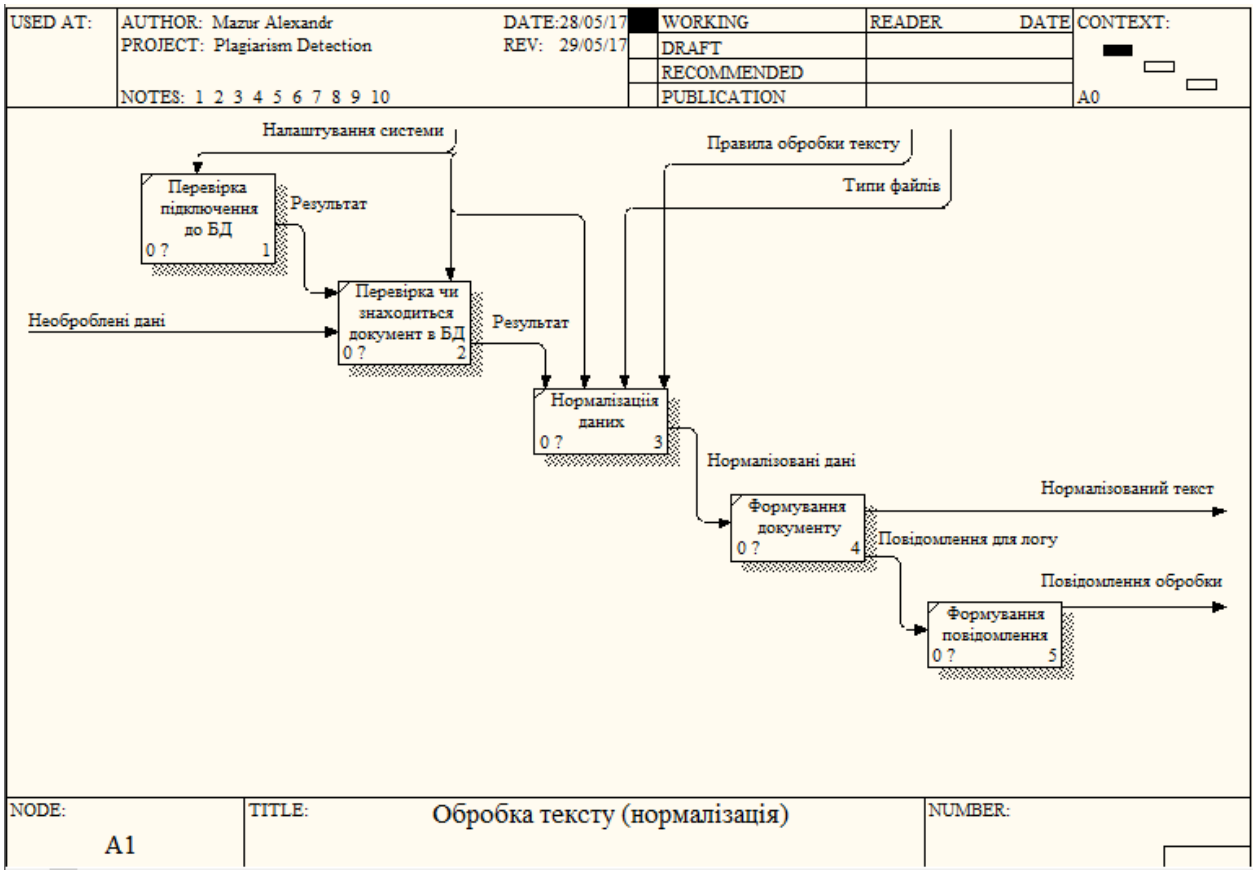

Рисунок 2.5 – Декомпозиція етапу обробки (нормалізації) тексту документу

Самим важким, з точки зору програмування, на даному кроці є нормалізація даних. Оскільки для нормалізації тексту програма повинна пройти певний шлях:

1) відкрити і отримати весь текст документу;

2) видалити з слів всі цифри і спеціальні символи;

3) видалити всі стоп-слова з колекції, стоп слова – заздалегідь підготовлена колекція слів, які не несуть корисності в перевірці;

4) перевести всі слова в lower case;

5) виділити основи слів.

Після обробки документу (наукової чи студентської роботи), програма повинна сповістити користувача про результат обробки. Для цих цілей в ПЗ повинна бути реалізована система логувань. Загальний вигляд реалізації системи логувань в програмі показаний на рис 2.6.

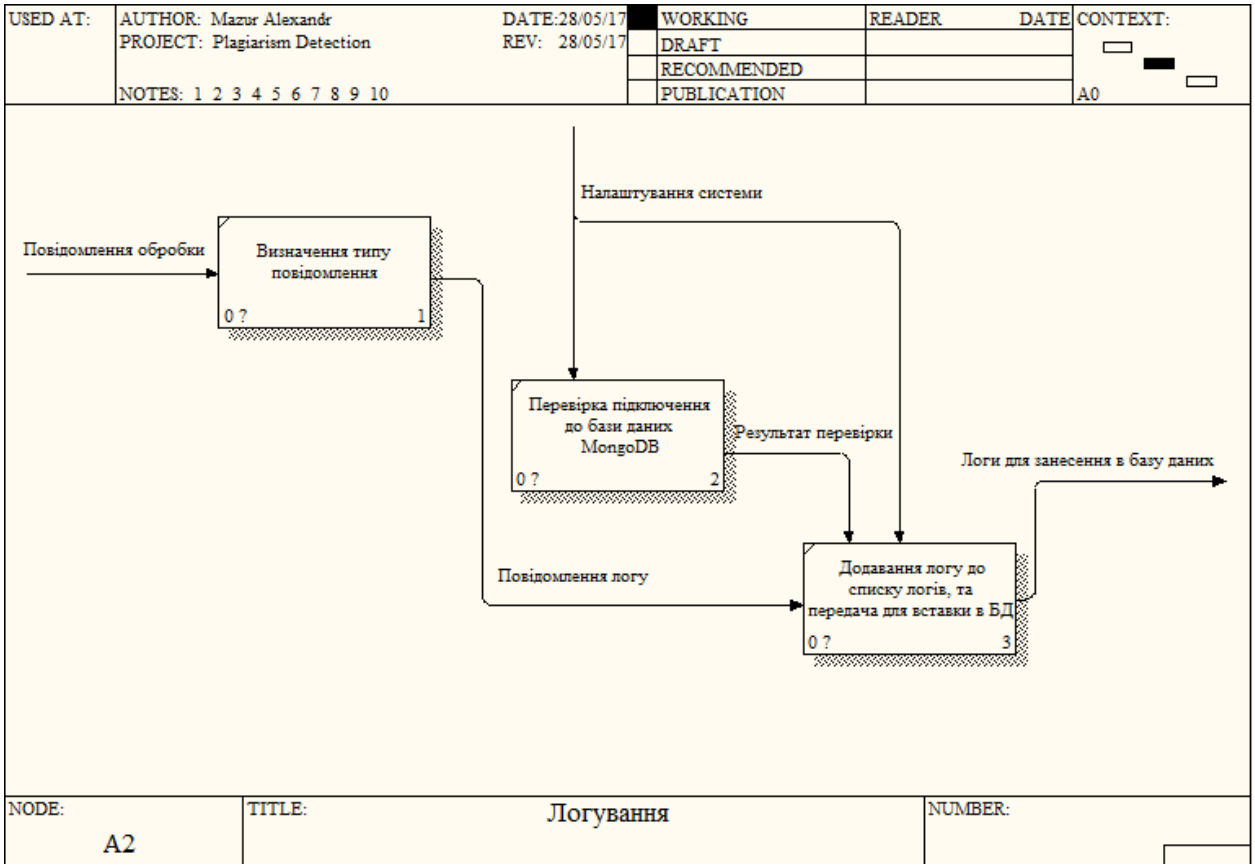

Рисунок 2.6 – Декомпозиція етапу логування

Етап логування в загальному випадку можна реалізувати в три кроки:

1) визначити тип повідомлення (просто повідомлення, інформаційне повідомлення, чи повідомлення про помилку);

2) перевірити, чи є можливість з’єднатись з сервером бази даних;

3) додати лог до списку логів та в базу даних.

Останнім етапом в розробці ПЗ додавання документів є саме додавання оброблених документів до бази даних. Декомпозиція етапу розробки

додавання документів зображена на рис. 2.7. Як видно з наведеної діаграми, для додавання документів до бази даних даний модуль програми повинен пройти три етапи:

1) перевірити підключення до бази даних;
2) підготувати дані до занесення в базу даних;
3) вставити дані до бази даних.

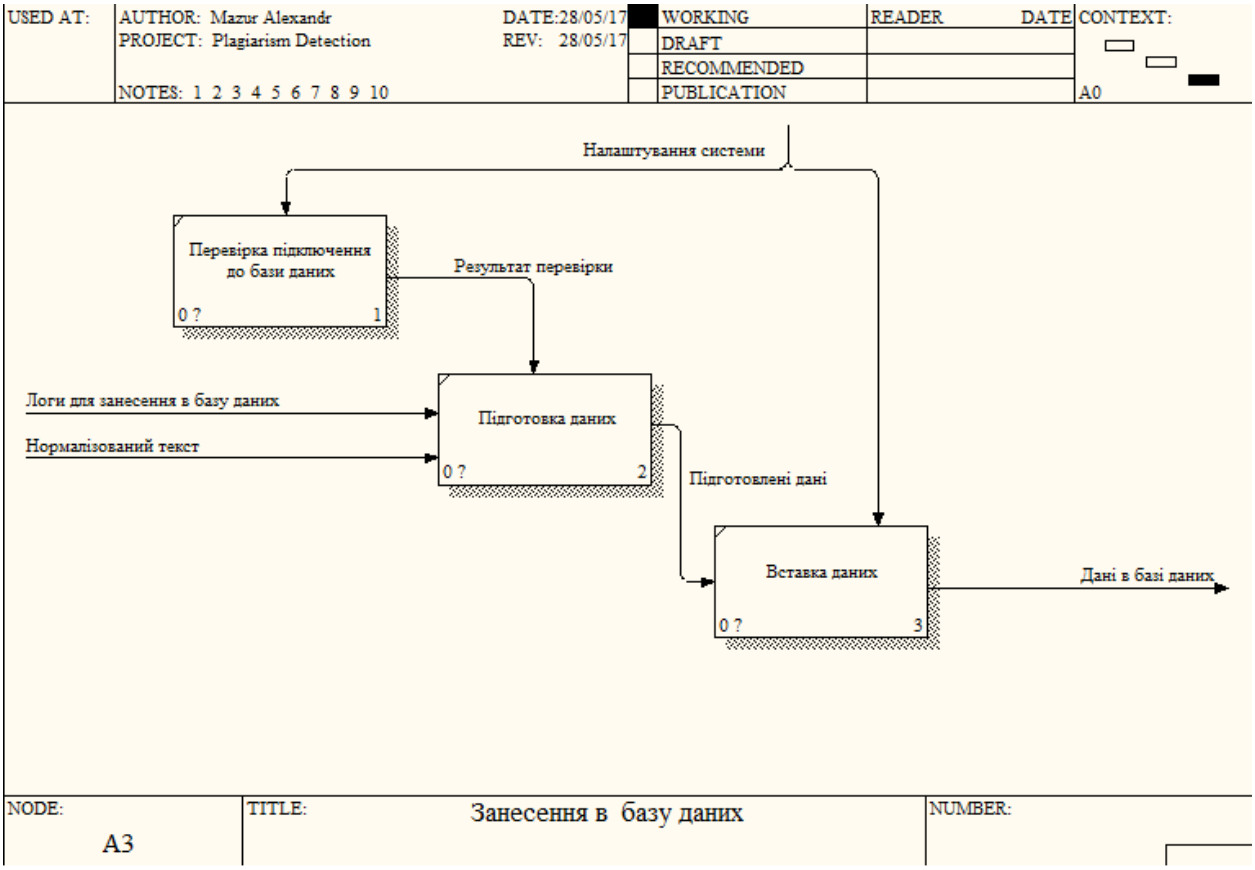

Рисунок 2.7 – Декомпозиція етапу додавання даних до бази даних

Отже, змоделювавши перший додаток системи – «DocumentAdder», виявилось, що найважчим етапом розробки даної програми являється перший етап – обробка (нормалізація) тексту документу та приведення тексту до необхідного вигляду.

2.3.5 Проектування підсистеми перевірки вхідних даних (документів) на плагіат

Згідно методології IDEF0 проектування будь-якої ІС необхідно спочатку побудувати її контекстну діаграму, а вже потім робити декомпозицію самої контекстної діаграми та бізнес-процесів, які присутні в системі, доки ІС не буде приведено до необхідного рівня деталізації.

Розпочнемо проектування програмного засобу перевірки документів на плагіат («PlagiarismDetector») з побудови його контекстної діаграми, зображеної на рис. 2.8.

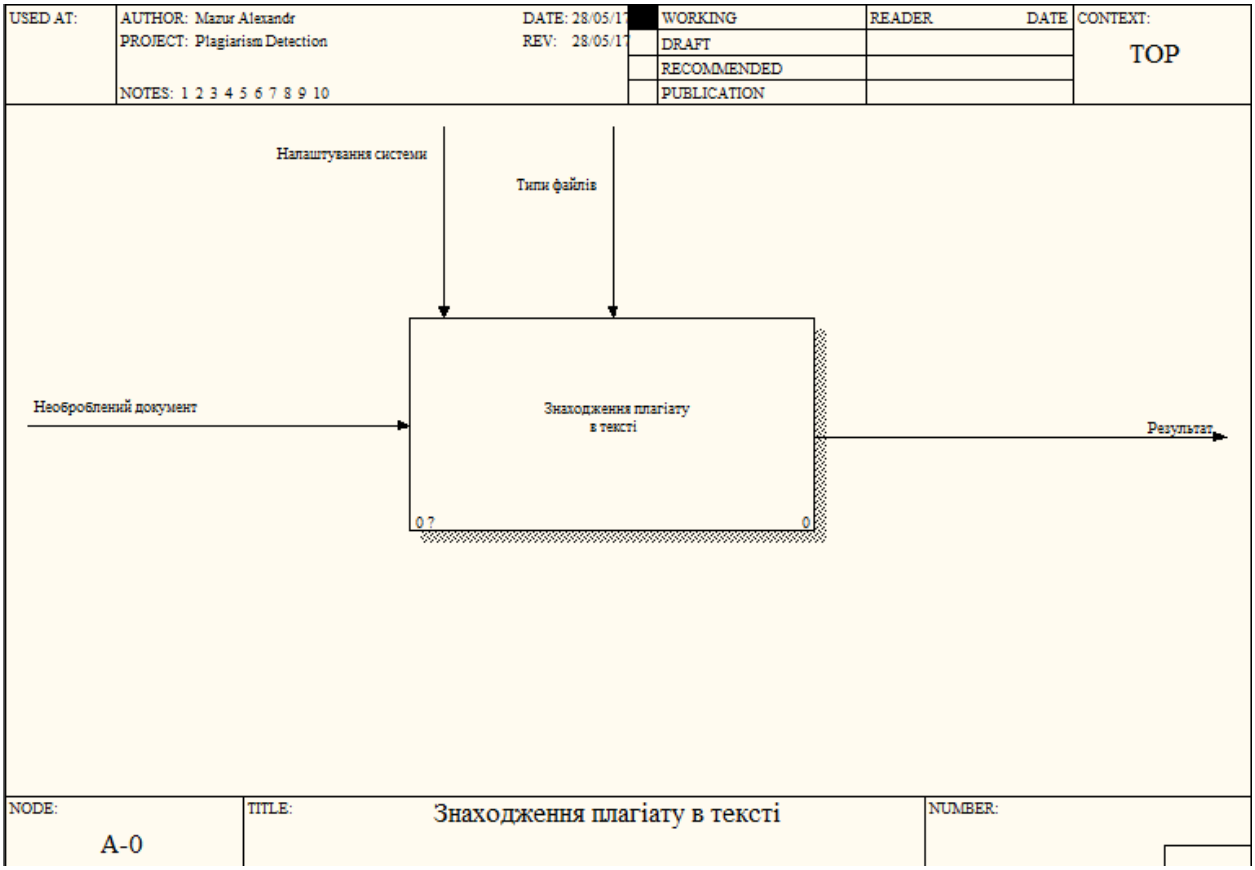

Рисунок 2.8 – Контекстна діаграма програмного засобу перевірки документів на плагіат – «PlagiarismDetector»

Наведена контекстна діаграма показує, що даний програмний засіб оперує невеликим набором чинників, які впливають на його роботу, а саме «Налаштування системи» та «Типи файлів». Отримуючи в вхідних даних необроблений, неперевірений документ (або групу документів – студентських чи наукових робіт) повертає результат (-и).

Оскільки контекстна діаграма будь-якої більш-менш серйозної ІС не дає достатньо інформації для початку розробки системи, то проведемо її декомпозицію, зображену на рис. 2.9.

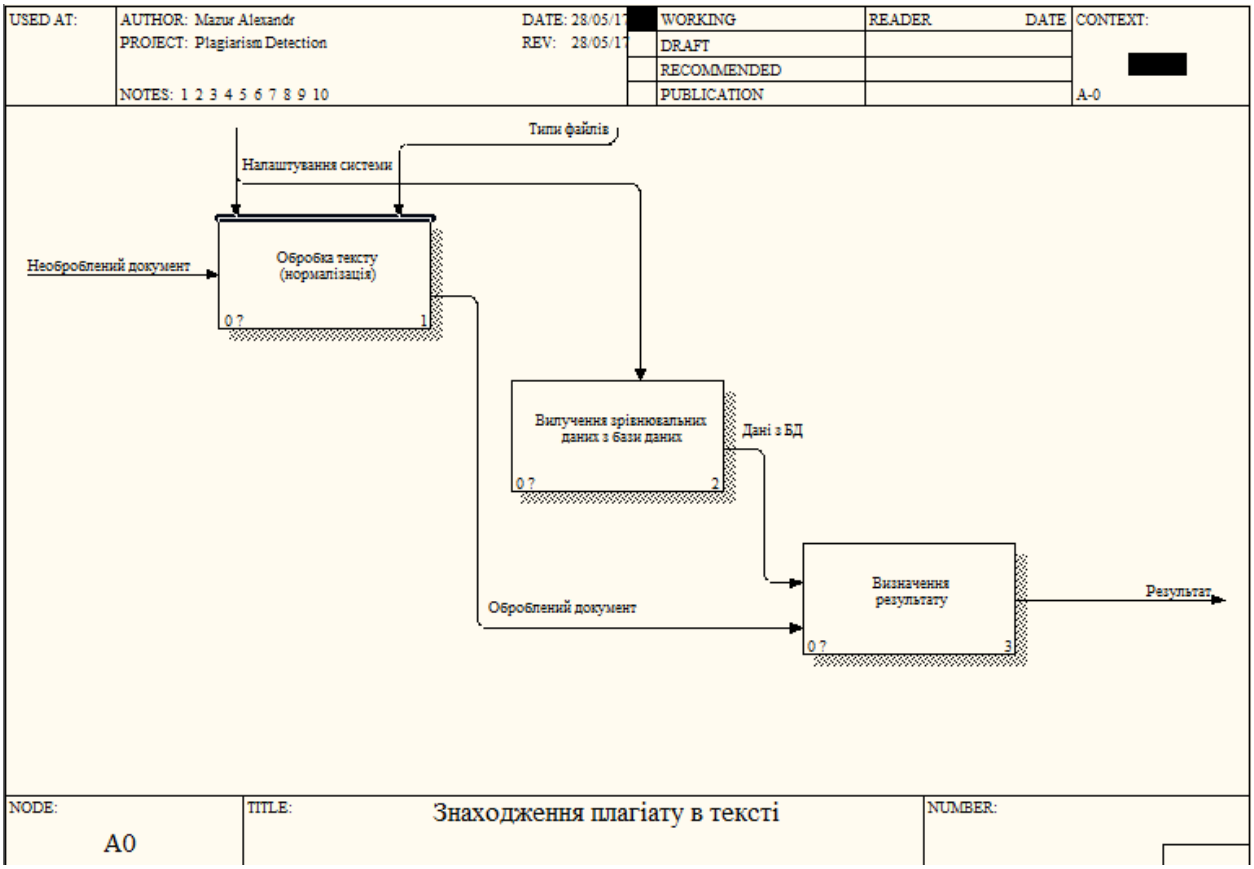

Рисунок 2.9 – Декомпозиція контекстної діаграми

Як видно з діаграми декомпозиції (рис. 2.9) для програмний засіб перевірки документу на плагіат повинен пройти три загальних етапи:

1) обробку документу (його нормалізацію);
2) вилучення зрівнювальних даних з БД;

3) визначення результату.

Проведемо декомпозицію кожного кроку для більш точного розуміння, як повинен виглядати кожен етап розробки програми перевірки на плагіат.

Перший етап – обробка документу зображено на рис. 2.10. Як видно з наведеної діаграми етап обробки документу можна виконати в 4 кроки:

1) перевірка підключення до БД;

2) перевірка, чи знаходиться документ в БД;

3) нормалізація даних документу;

4) формування документу (приведення його до вигляду, з яким можна буде працювати при перевірці на плагіат).

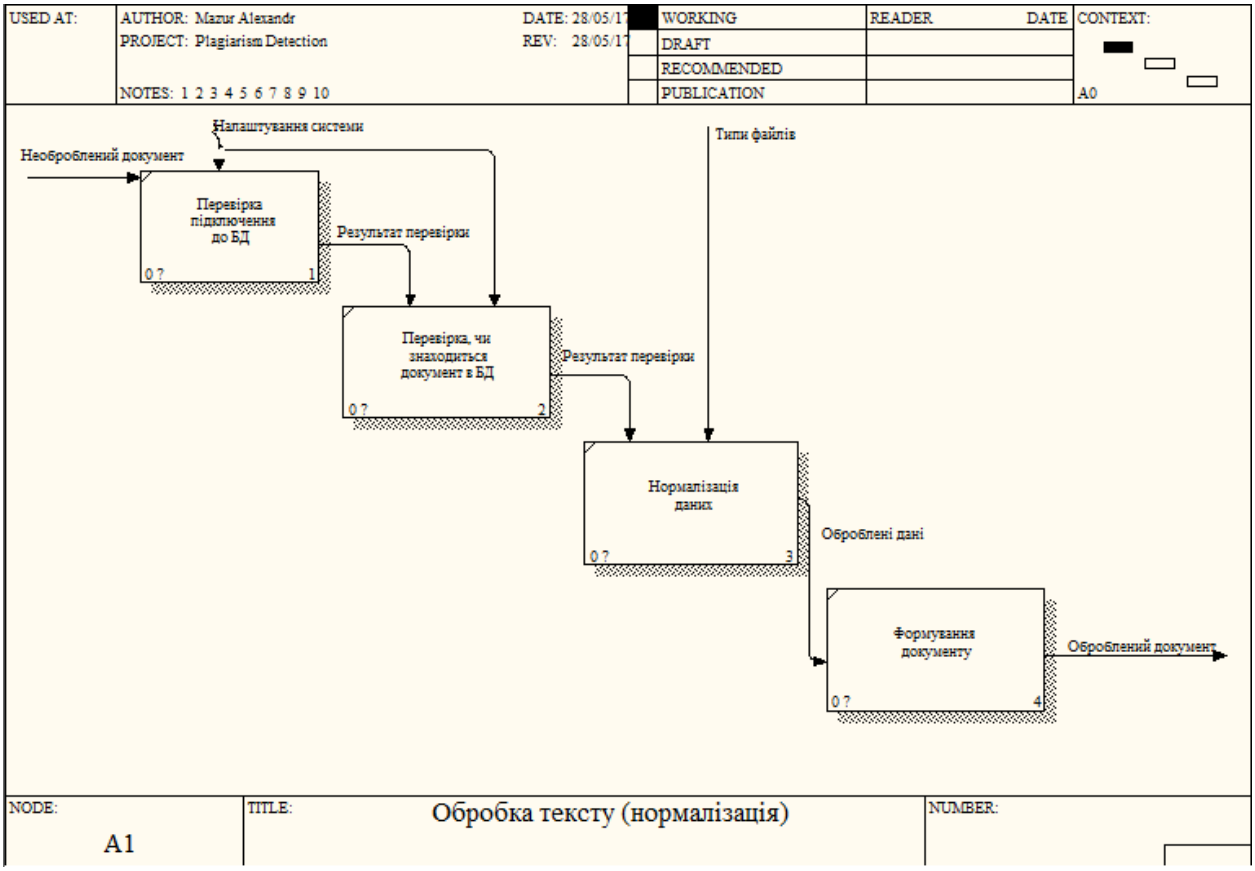

Рисунок 2.10 – Декомпозиція першого кроку проектування ПЗ – обробка документу (нормалізація)

Найбільшою проблемою на цьому етапі (як і в попередній програмі – «DocumentAdder» буде реалізація блоку «Нормалізація даних», дії цього блоку будуть аналогічними діям в попередньому ПЗ і описані в пункті 2.4.3.

Другий крок розробки програмного забезпечення для пошуку плагіату в студентських та наукових роботах полягає в вилученні всіх документів з бази даних (на базі цих робіт буде проводитись пошук плагіату в необхідному документі). Розглянемо етапи цього модулю на рис. 2.11.

1) перевірка підключення до серверу бази даних;

2) формування даних до потрібного вигляду.

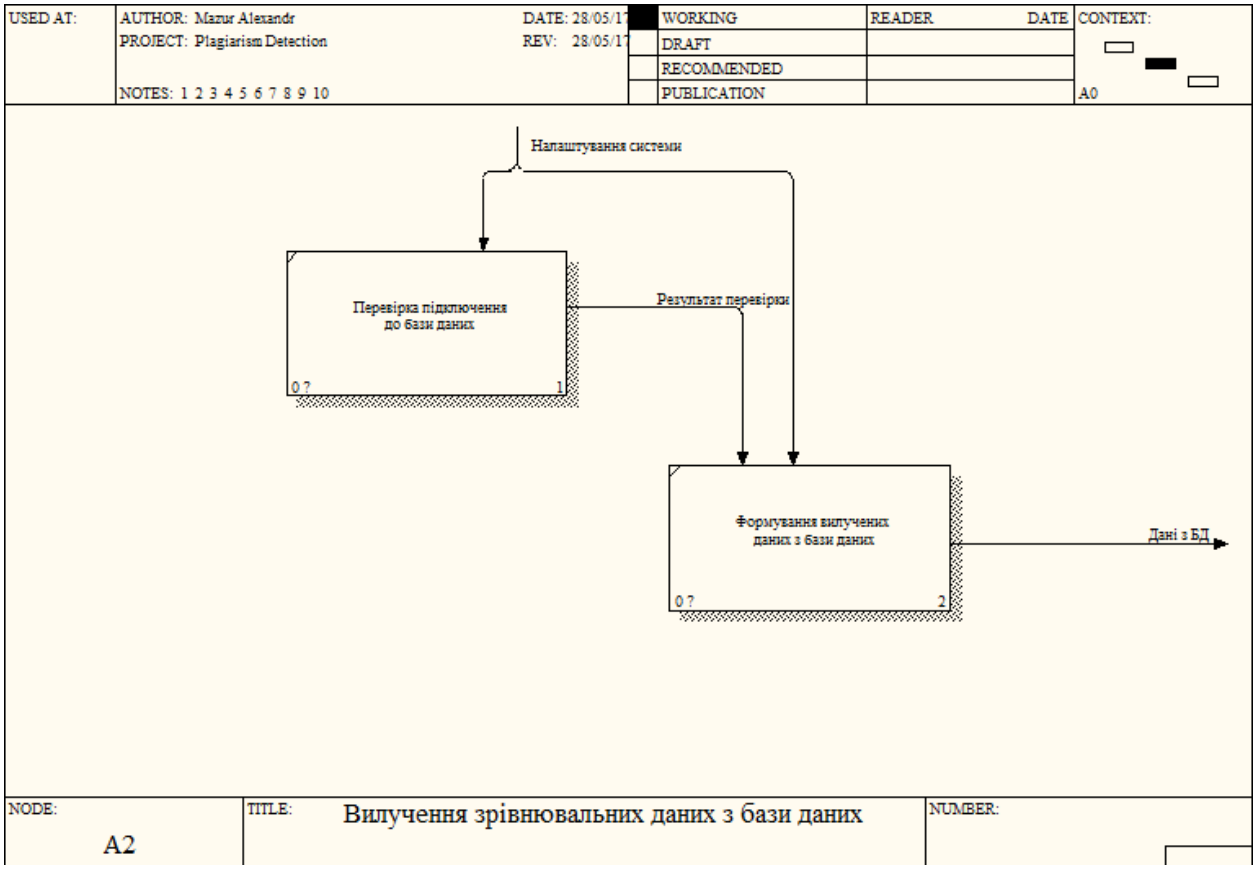

Рисунок 2.11 – Декомпозиція другого модулю ПЗ для знаходження плагіату – «PlagiarismDetector»

На другому етапі (формування даних) немає необхідності в нормалізації тексту документів із бази даних, оскільки перший ПЗ

(«DocumentAdder») зробить це заздалегідь, при додаванні довірених документів до бази даних.

Після даного етапу можна буде побудувати вектори документів, для того, щоб потім їх зрівняти за допомогою певного алгоритму.

Діаграма декомпозиції третього і останнього етапу в розробці програмного додатку для перевірки документів на плагіат наведена на рис. 2.12. Для отримання результату останній модуль програмного засобу повинен пройти три кроки:

1) формування рівних векторів;
2) знаходження косинусної подібності між векторами;
3) формування результату.

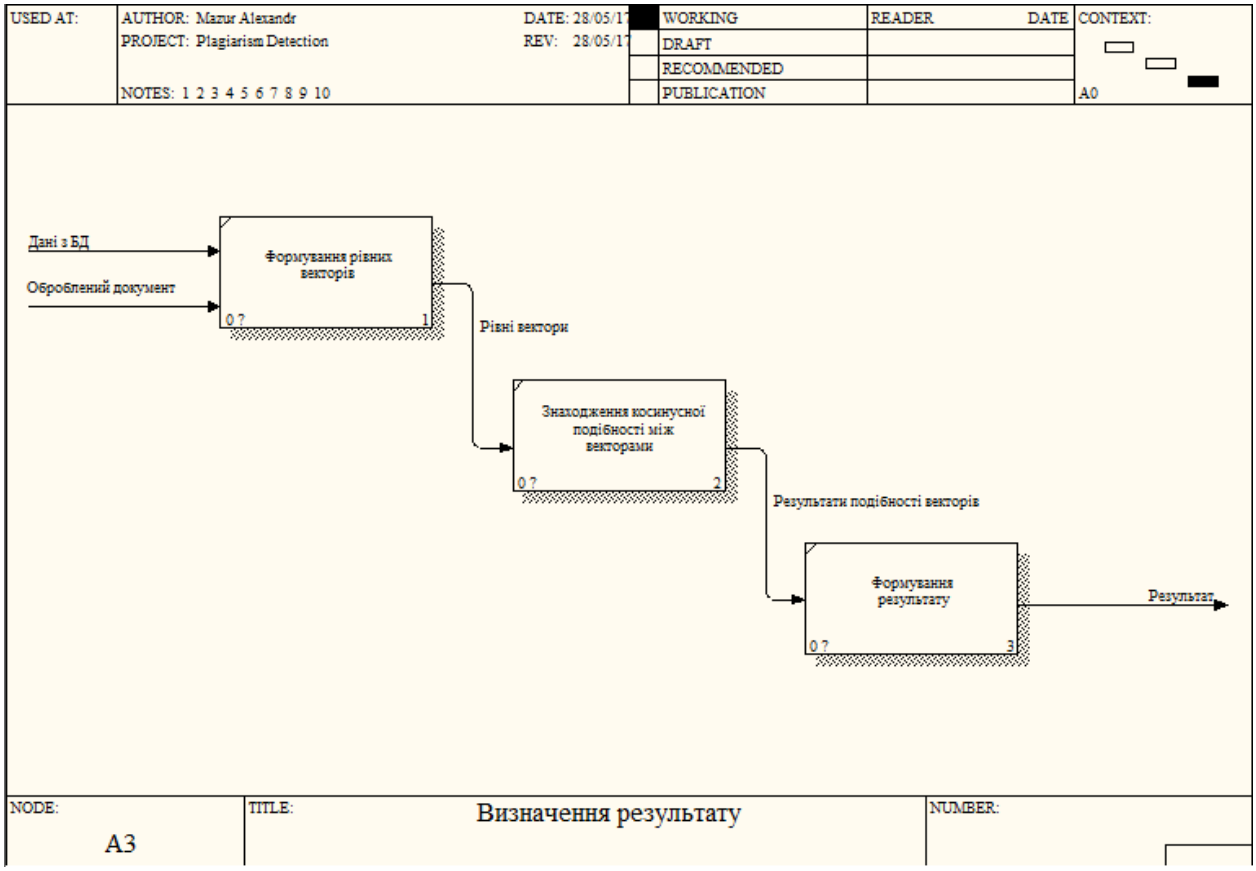

Рисунок 2.12 – Декомпозиція останнього кроку в розробці ПЗ перевірки на плагіат – «Визначення результату»

Для порівняння векторів алгоритмом косинусної подібності необхідно, аби вектори були однакових розмірів. Вектори містять в собі дрібні числа, які відповідають TF*IDF мірі слів документу. Якщо в одному документі немає певного слова із іншого документу, то в векторі цього документу це слово замінюється нулем [33].

Косинусна подібність – це міра подібності векторів, яка використовує для вимірювання косинус кута між векторами [34]. Використовуючи значення цієї функції можна знайти подібність між двома векторами, а оскільки вектори являють собою трансформовані документи, то ми знаходимо таким чином міру подібності двох документів.

2.4 Програмна реалізація інформаційної системи

2.4.1 Вимоги до системи пошуку плагіату

Користувачі даної системи для підтримки можливостей пошуку плагіату в студентських та наукових роботах мають можливість працювати з графічними додатками. Одним з найбільш значущих елементів будь-якої системи є GUI.

Головне завдання системи: надавати зручне, що має достатню кількість функціональних можливостей, графічне застосування користувачам для роботи з системою пошуку плагіату.

Основні фактори, за допомогою яких можна оцінити або навіть виміряти зручність використання системи це:

1) адекватність інтерфейсу;

2) ефективність запобігання та подолання помилок користувачів;

3) доступність.

Адекватність користувальницького інтерфейсу програми – це його відповідність тим завданням, які користувачі повинні і хотіли б вирішувати з її допомогою.

Ефективність запобігання та подолання помилок користувачами тим краще, чим рідше користувачі помиляються при роботі з даним інтерфейсом і чим менше часу і зусиль потрібно для подолання наслідків вже зроблених помилок. Велике значення має також ризик, пов'язаний з виникненням помилки.

Розроблена система повинна бути настільки зрозумілою, щоб користувач, який ніколи раніше не бачив її, але добре розбирається в предметній області, міг без всякого навчання почати її використовувати.

Для зручного користування системою пошуку плагіату розроблено два графічних програмних засоби:

1) для додавання довірених документів до локальної (внутрішньої) бази даних – «DocumentAdder»;

2) для виявлення плагіату у вказаних документах на основі цих самих документів – «PlagiarismDetector».

2.4.2 Етапи розробки системи пошуку плагіату

Програмні засоби, наявні в системі пошуку плагіату досить різні, хоча й пов’язані між собою певними функціональними частинами (робота з БД, документами, обмеженнями, перевірками тощо). Оскільки вони ще й різні по кількості часу, затраченому на їх розробку, то слід почати проектування і розробку з того програмного засобу, який йде першим в логічному ланцюзі роботи системи, тобто з ПЗ додавання довірених документів до бази даних.

Користувацький інтерфейс обох програмних засобів написаний за допомогою декларативної мови розмітки XAML, основаній на XML. Використання цієї мови значно пришвидшує розробку GUI, на відміну від написання GUI за допомогою коду мови програмування C#.

2.4.3 Етапи розробки програмного засобу для додавання довірених документів до бази даних

Перед тим як приступити безпосередньо до програмування додатку потрібно скласти певні етапи розробки:

1) Розробка графічного користувацького інтерфейсу:
   - розробка оповіщення користувача про хід роботи додатку;
   - розробка інтерфейсу налаштувань.
2) Розробка налаштувань програмного засобу:
   - застосування налаштувань;
   - збереження налаштувань при виході з програмного засобу;
   - завантаження налаштувань при старті додатку.
3) Розробка API-інтерфейсу для роботи з базою даних:
   - розробка CRUD-функцій;
   - розробка користувацьких класів (нових типів даних) для роботи з даним API;
   - розробка класів помилок (exception), які можуть виникати при роботі з базою даних через API.
4) Розробка методів для нормалізації тексту документів;
5) Розробка нових типів даних для функціонування програмного засобу;
6) Розробка методів побудови даних для алгоритму TF*IDF;
7) Розробка класу перевірок вхідних даних програмного засобу;
8) Розробка методів розширень (extension methods – особливість мови програмування C#, яка дозволяє розширити існуючі класи, не успадковуючи їх і не створюючи ціпку ієрархій) для колекцій фреймворку .NET;
9) Розробка класу для роботи з файлами вхідних документів;
10) Розробка безперервної роботи програми;

11) Розробка системи логувань для оповіщення користувачів про хід виконання роботи ПЗ;

12) Локалізація програми.

Після розробки даного ПЗ можна буде використати API-інтерфейс для роботи з БД і розроблені нові типи даних в інших програмних засобах системи.

2.4.4 Етапи розробки підсистеми виявлення плагіату

Побудова етапів розробки даного ПЗ дещо відрізняється від попереднього ПЗ. По перше – об’єм роботи при розробці цього програмного засобу значно менший, оскільки багато важких модулей ПЗ вже були розроблені в попередньому програмному додатку – робота з БД, нові типи даних, якими повинна оперувати програма. По друге, хід виконання роботи програми представлений в другому вигляді, а сповіщення про помилки наявні в вигляді спливаючих вікон, а отже система логування в даному програмному засобі не потрібна.

Отже, етапи розробки ПЗ:

1) Розробка графічного користувацького інтерфейсу:
   - розробка оповіщення користувача про хід роботи додатку;
   - розробка інтерфейсу налаштувань;
   - розробка інтерфейсу представлення даних (вхідних та результатів).
2) Розробка налаштувань програмного засобу:
   - застосування налаштувань;
   - збереження налаштувань при виході з програмного засобу;
   - завантаження налаштувань при старті додатку.
3) Створення нових типів даних для представлення результатів роботи програмного додатку.
4) Розробка методів для побудови рівних векторів для порівняння.

5) Розробка методів для порівняння векторів і виявлення долі схожості в відсотковому вигляді.

6) Розробка безперервної роботи програми, до тих пір, поки в черзі наявні документи, які потрібно перевірити на плагіат.

7) Локалізація ПЗ.

Як видно, після розробки першого додатку – роботи над другим залишилось не так багато.

Після визначення етапів розробки програмних засобів системи пошуку плагіату можна переходити безпосередньо до їхньої розробки.

2.4.5 Розробка користувацького інтерфейсу системи пошуку плагіату

Для швидшої розробки програмних засобів системи розробку слід починати з розробки графічного користувацького інтерфейсу, оскільки таким чином буде виконано дві роботи: розробка GUI і походу написання програмного засобу його можна буде тестувати «в живу».

Для написання ПЗ з використанням мови C# і технології WPF найбільш раціональним буде використання патерну (шаблон проектування) архітектури додатку MVVM. Для розробки користувацького інтерфейсу, був використаний додаток Microsoft Blend.

2.4.6 Розробка користувацького інтерфейсу підсистеми додавання документів до бази даних

Користувацький інтерфейс ПЗ «DocumentAdder» складається з меню в вигляді вкладок (табів), в залежності від активності яких змінюється вигляд ПЗ. Так, на вкладці «Логи» (рис. 2.13) (система сповіщення користувача про хід виконання роботи ПЗ) зображено:

1) ListView – графічний елемент управління спискового типу, в якому є можливість подати інформацію в різних колонках;

2) стек з кнопками, які дають змогу працювати з списком сповіщень програмного засобу;

3) кнопки «Старт» та «Стоп» які відповідно запускають ПЗ та зупиняють його роботу.

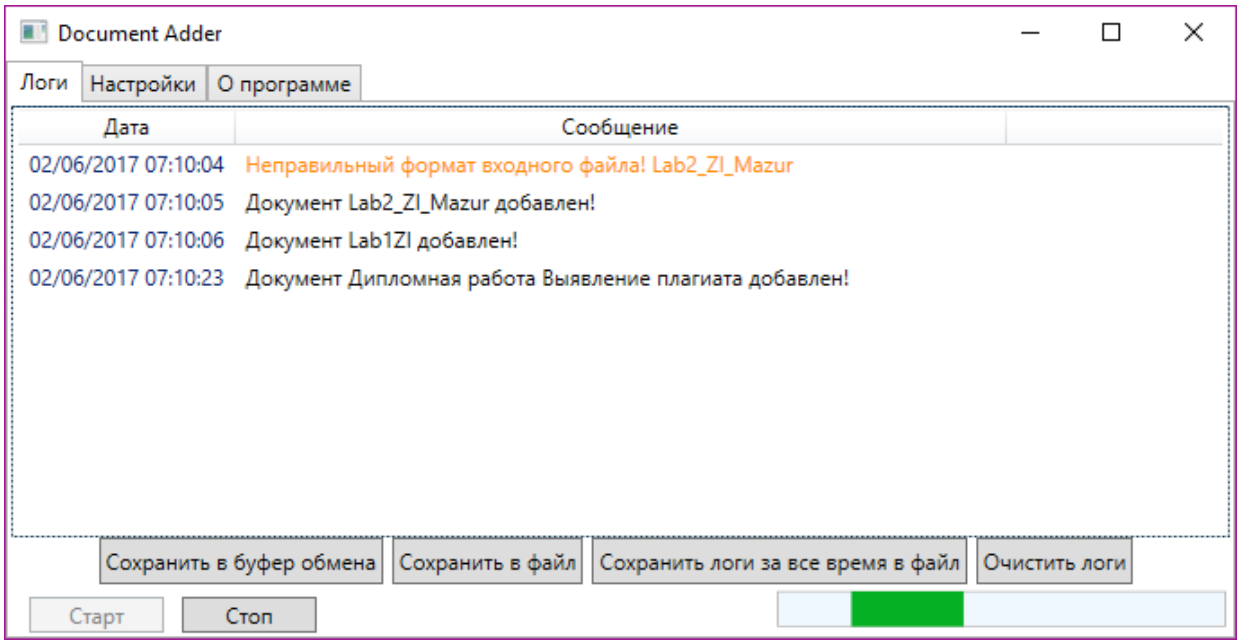

Рисунок 2.13 – Користувацький інтерфейс програми додавання довірених документів до БД – система сповіщення користувача

На вкладці «Настройки» (рис. 2.14) є можливість в лівому меню навігації вибрати клас налаштувань, які потрібно змінити:

1) «База даних» – налаштування для бази даних;

2) «Репозиторий» – налаштування джерел, звідки програма буде брати;

3) «Другие» – інші налаштування ПЗ.

Локалізація ПЗ працює наступним чином: CLR визначає мову системи і на основі цих даних виставляє мову ПЗ як в системі, перед включенням програми. Доступно 3 мови:

1) англійська;

2) українська;

3) російська.

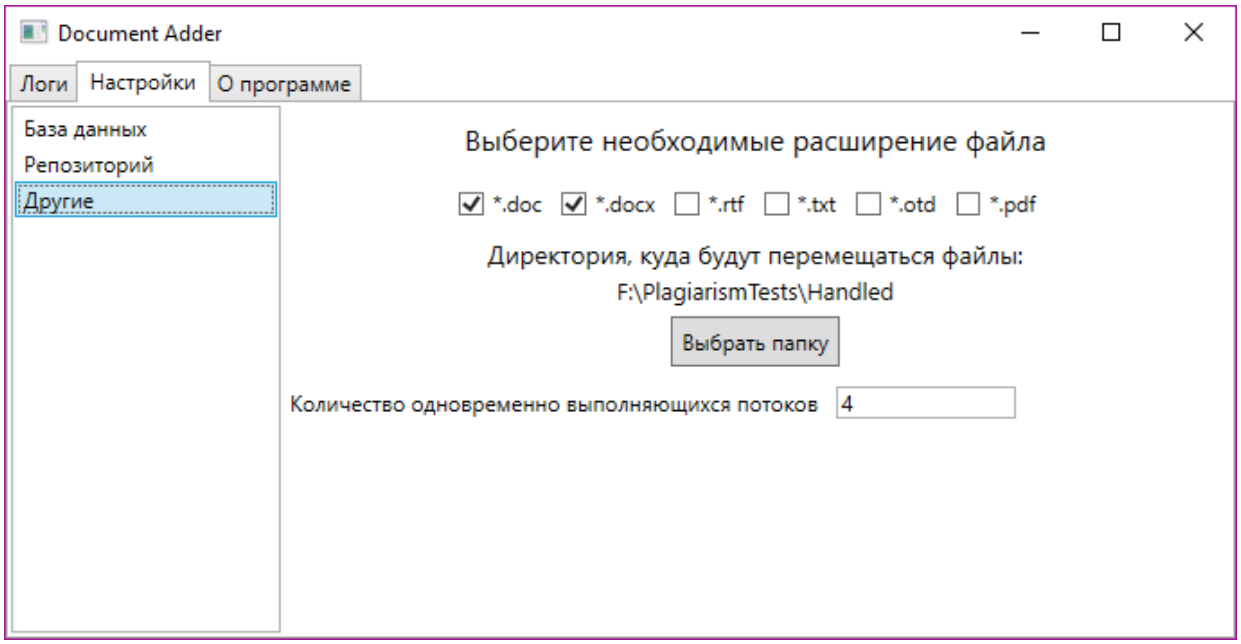

Рисунок 2.14 – Користувацький інтерфейс програми додавання довірених даних до БД – налаштування

2.4.7 Розробка користувацького інтерфейсу підсистеми перевірки документів на плагіат

Графічний користувацький інтерфейс програмного забезпечення «Plagiarism Detector» (рис. 2.15) дещо відрізняється від першого ПЗ. Для навігації по розділам не використовуються вкладки (таби), меню знаходиться в верхній частині ПЗ. За допомогою нього можна зробити певні програмні дії, або викликати пару вікон: «Налаштування», «Про програму».

Окрім меню там знаходяться:

1) два ListView, які представляють собою вхідні дані та результати;

2) кнопки: вибір документів які потрібно перевірити, старт та стоп ПЗ, а коли є результати роботи – то можна виконати певні дії з ними за допомогою кнопок в нижній частині ПЗ;

3) написи, які дають зрозуміти де знаходяться вхідні дані, а де результати роботи.

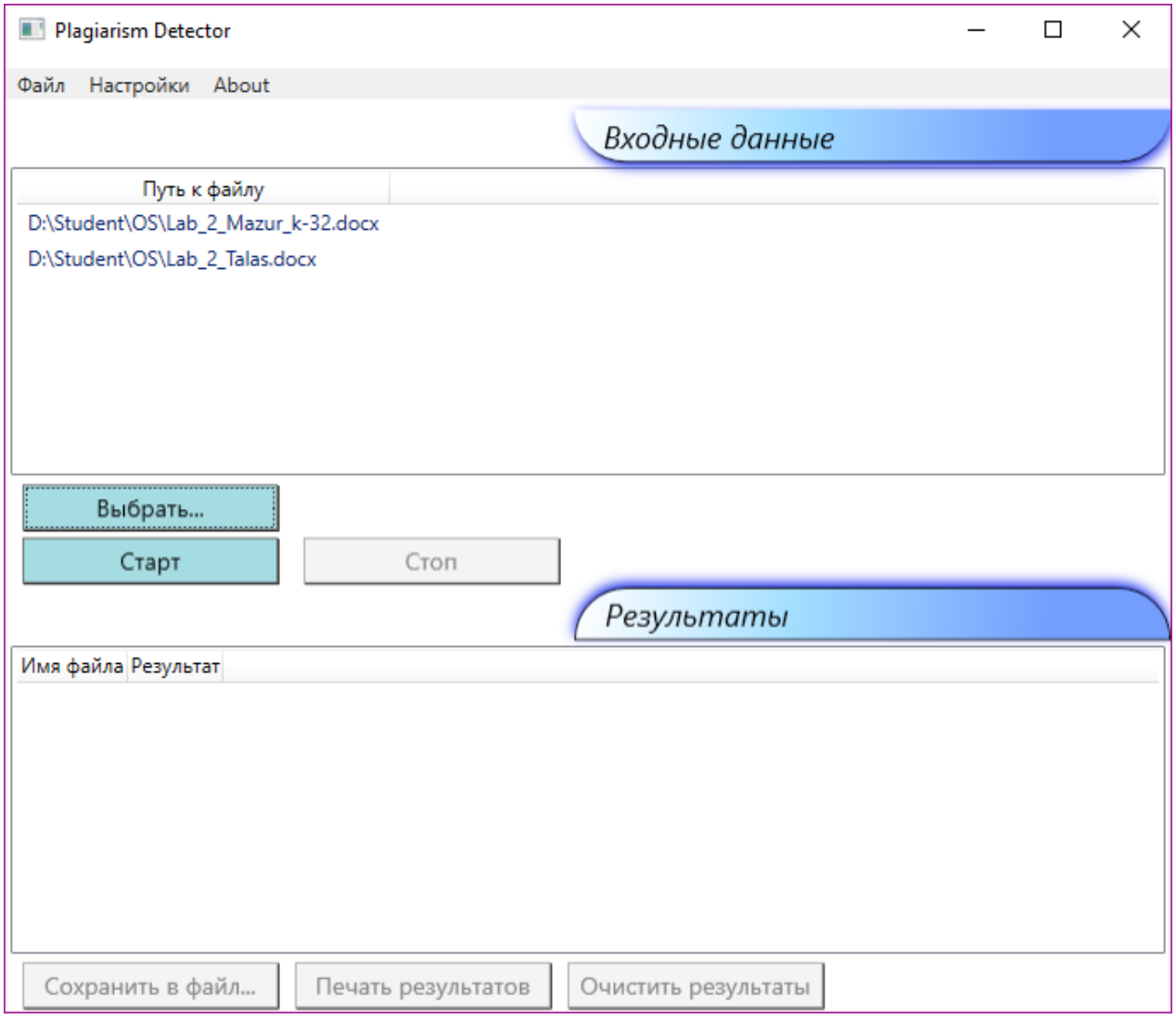

Рисунок 2.15 – Користувацький інтерфейс ПЗ «Plagiarism Detector»

Налаштування ПЗ знаходяться не в іншій вкладці, як в попередній програмі, а в окремому вікні (рис. 2.16), яке викликається по натисканню кнопки в меню «Настройки».

Для даного ПЗ налаштувань потрібно не так багато – лише для підключення до сервера бази даних MongoDB, вибір типів файлів, які потрібно опрацювати та кількість одночасно оброблюваних потоків.

Налаштування зберігаються в системній папці програмного засобу в вигляді JSON-строки, яка потім конвертується в JSON-об’єкт. Це досить зручно, оскільки бібліотека роботи з JSON-форматом працює швидко, та сама конвертує C# об’єкти в json-строку і десеріалізує їх назад в об’єкт.

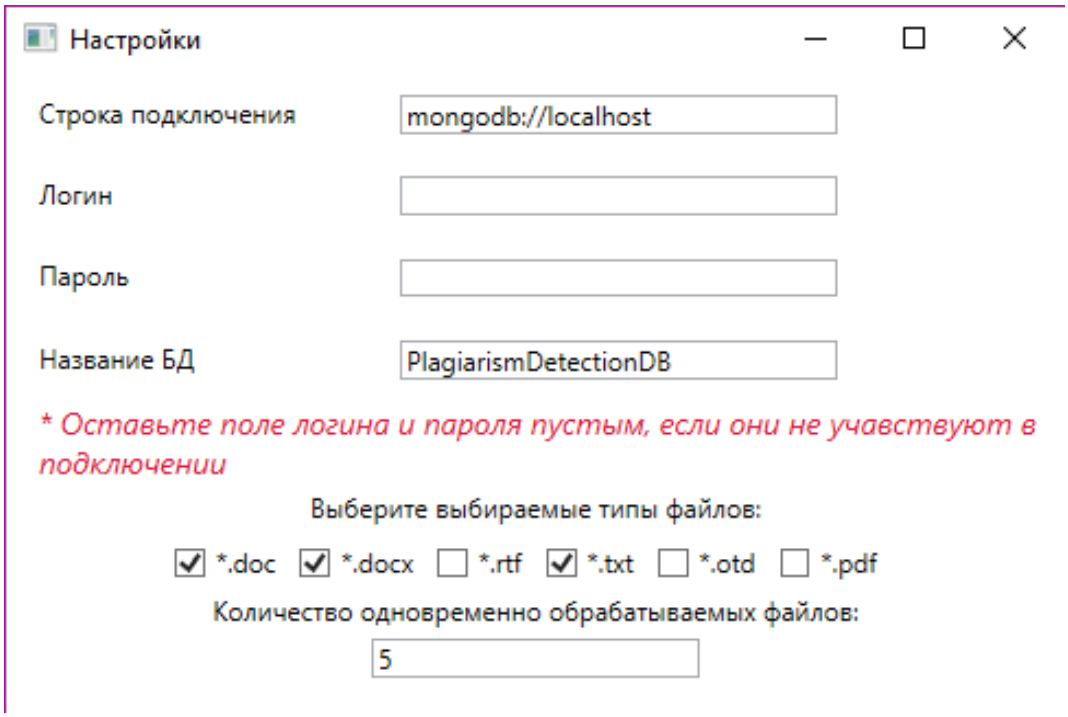

Рисунок 2.16 – Вікно налаштувань ПЗ «Plagiarism Detector»

Оскільки результати, отримані після роботи програмного засобу «Plagiarism Detector» недостатньо просто подивитись на екрані, потрібно десь їх зберігати. Реалізація цих можливостей надана в кнопках:

1) «Сохранить в файл» – зберігає отримані результати в текстовий файл (*.txt);

2) «Печать результатов» – викликає вікно принтеру для печаті результатів.

Приклад роботи збереження результатів в файл зображена на рис. 2.17 та рис. 2.18.

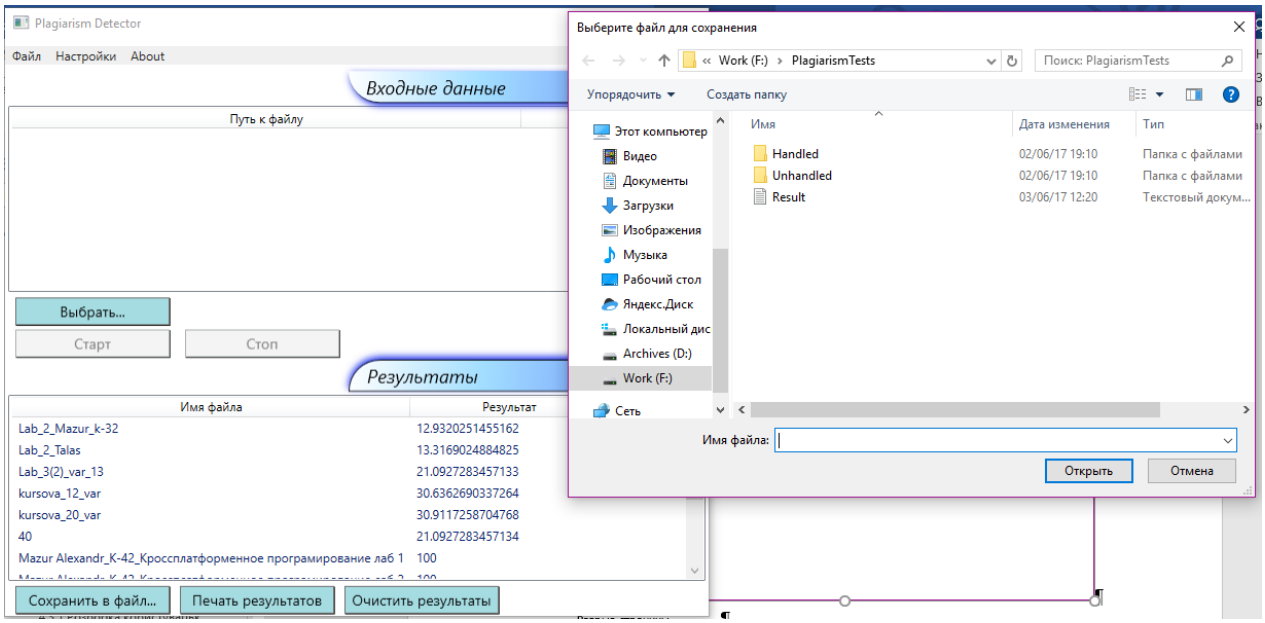

Рисунок 2.17 – Відкрите вікно збереження результатів ПЗ «Plagiarism Detector» в файл

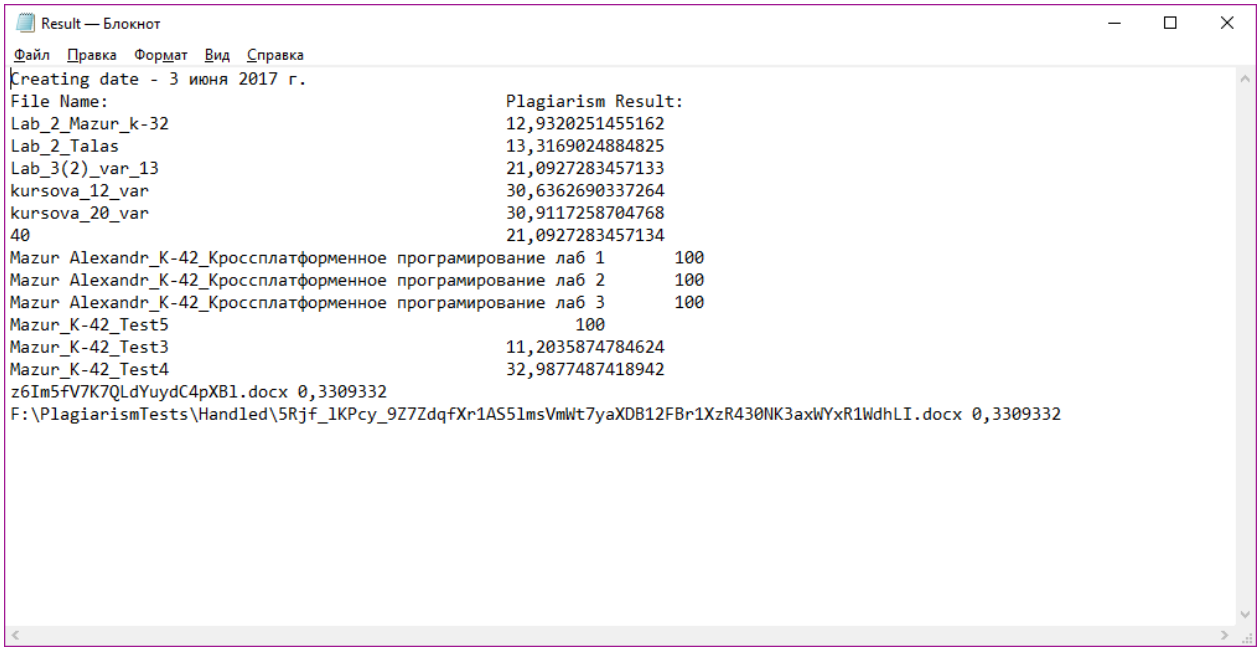

Рисунок 2.18 – Результат збереження результатів роботи ПЗ «Plagiarism Detector» в текстовому файлі

Для відкриття діалогового вікна використовується вбудований механізм в каркасі .NET Framework CommonOpenFileDialog. Він дозволяє:

- створювати файли;
- отримувати шлях виділеного файлу (або декількох файлів в вигляді колекції типу string з шляхами файлів).

Мова програмування дозволяє зручно працювати з форматуванням тексту використовуючи вбудований тип string. Запис файлу здійснюється як і в усіх інших МП:

- відкривається потік запису файлу;
- записуються дані в файл;
- закривається потік запису файлу.

Даний ПЗ, як і попередній (пункт 2.4.6), також локалізований під три мови, в залежності від мови, встановленій за замовчування в ОС:

1) англійська;
2) українська;
3) російська.

2.4.8 Розробка функціональної частини системи пошуку плагіату

Для реалізації функціональних частин (модулів) програмних засобів системи використовується мова програмування C# технології WPF відповідно до шаблону проектування MVVM.

Оскільки для написання та зв’язування (binding) графічних користувацьких інтерфейсів з функціональною частиною програмних засобів використовується шаблон проектування MVVM, то розпочати розробку ПЗ слід з створення моделей та вид-моделей ПЗ.

Відповідно до етапів розробки ПЗ, після розробки графічного користувацького інтерфейсу слід перейти до розробки налаштувань системи: їх налаштування, збереження та завантаження.

Мультипоточність системи досягається завдяки вбудованим в каркас .NET Framework механізмів асинхронного програмування (async, await, клас Task).

Безперервна робота програмних засобів системи функціонує завдяки об’єктам DispatcherTimer з каркасу .NET Framework. Коли натиснута кнопка «Старт», раз в певний період часу програмний засіб перевіряє необхідність подальшої роботи, якщо роботи немає, то в залежності від ПЗ – або виконується зупинка роботи, або просто відбувається подальше очікування роботи.

Для локалізації обох програмних засобів системи використовуються файли ресурсів, які потім приєднуються до написів додатку за допомогою механізму прив’язок (binding).

2.4.9 Розробка функціональної частини підсистеми додавання документів до бази даних «Document Adder»

Розробка функціональної частини даного ПЗ починається з написання API-інтерфейсу роботи з БД.

Для цього в каркасі програмного засобу представлені наступні класи:

1) DocumentAdder.Types.DataBase.DataBase– тут представлена вся робота з базою даних MongoDB (CRUD функції, перевірка підключення тощо). Інші класи даного API використовуються тільки для використання колекцій СКБД MongoDB в мові програмування C#;

2) DocumentAdder.Types.DocumentCollection – використовується для зручної роботи з колекцією документів (студентських чи наукових робіт) з СКБД MongoDB в мові програмування C#;

3) DocumentAdder.Types.FileCollection – використовується для зручної з колекцією файлів, в яких зберігаються документи (студентські чи наукові роботи) з СКБД MongoDB в мові програмування C#;

4) DocumentAdder.Types.IdfItem – використовується для зручної роботи з колекцією IDF-значень слів з СКБД MongoDB в мові програмування C#;

5) DocumentAdder.Types.Log – використовується для зручної роботи з логами програмного засобу з СКБД MongoDB в мові програмування C#;

Слід також зазначити, що для роботи з СКБД MongoDB в мові програмування C# можна використати загальний клас для роботи з колекціями – MongoDB.Bson.BsonDocument. Проте його використання є недостатньо гнучким і сильно обмежує розробника, особливо в тих випадках, коли потрібно використовувати дані з СКБД MongoDB в роботі ПЗ.

Після розробки API-інтерфейсу для роботи з СКБД MongoDB слід розробити методи нормалізації тексту та нові типи даних для функціонування ПЗ.

Методи роботи з документом, текстом документу та методи побудови даних для зручної роботи з базою даних MongoDB розроблені і представлені в класі DocumentAdder.Actions.DocumentAction.DocumentsActions. Слід також зазначити, що для покращеного функціонування методів використовується стеммінг [35]. Алгоритм стемінгу для кирилиці та латиниці представлений в Open Source проекті від компанії Iveonik та знаходиться в просторі імен Iveonik.Stemmers.

Методи розширень, перевірок, трансформацій, роботи з файлами представлені в класах:

- Helpers.Extensions – включає в себе методи для розширення роботи колекцій в ПЗ;
- Helpers.Verifications – включає в себе методи для перевірки деякий вхідних даних, наприклад правильність імені вхідного файлу;
- Helpers.Transformation – включає в себе методи для трансформації одних значень в інші;

− Actions.FileActions – включає в себе методи для роботи з файлами (наприклад дії з файлом – переміщення, видалення тощо) для функціонування деяких методів використовується тип (enum-клас) Types.FileActionType.

Для функціонування системи логувань (система сповіщень користувача про хід виконання роботи ПЗ) в ПЗ розроблені класи:

1) Log – представляє собою новий тип даних, який можна з легкістю використати для роботи з колекцією логів в базі даних;

2) LogType – enum-клас, показує тип логів (повідомлення, інформація, помилка);

3) LogViewModel – представляє функціональну частину системи логувань – тобто методи, команди, принципи роботи системи.

Після того, як програмний засіб для додавання довірених документів до локальної (внутрішньої) бази даних розроблений можна переходити до розробки другого ПЗ системи – ПЗ пошуку плагіату у вказаних документах, оскільки більшість функціоналу, який можна використати в наступному ПЗ, було розроблено в першому ПЗ.

2.4.10 Розробка функціональної частини підсистеми перевірки документів на плагіат «Plagiarism Detector»

На даному етапі залишилось реалізувати:

1) створення нових типів даних для представлення результатів роботи програмного додатку;

2) розробка методів для побудови рівних векторів для порівняння;

3) розробка методів для порівняння векторів і виявлення долі схожості в відсотковому вигляді.

Нові типи даних для представлення результатів розроблені і представлені в класах:

1) PlagiarismDetectResult – тип даних для подання результату роботи програми в головне вікно програми;

2) PlagiarismDetectExpandedResult – тип даних для розширеного подання результату роботи програми, використовується при натисканні кнопки контекстного меню «Показать подробности» (рис. 2.19 та рис. 2.20).

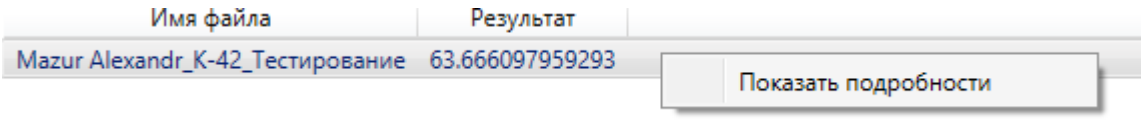

Рисунок 2.19 – Контекстне меню результатів програми «Plagiarism Detector»

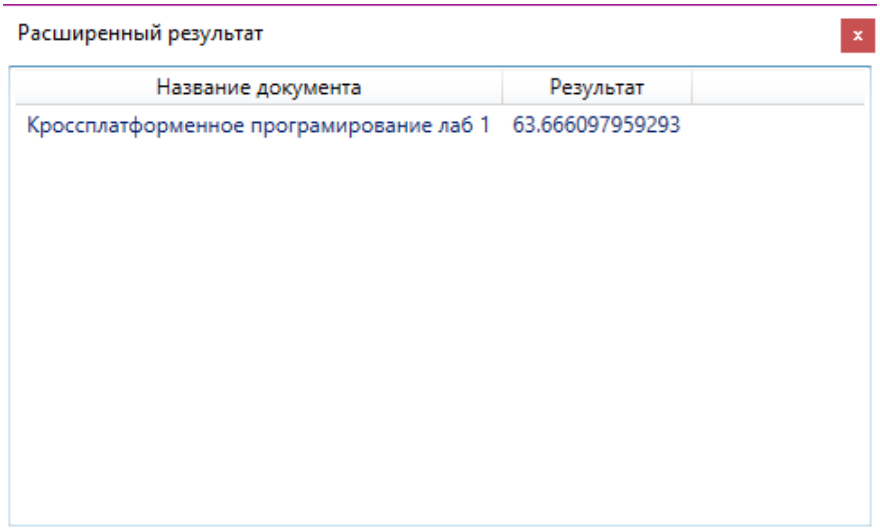

Рисунок 2.20 – Розширений результат представлення результатів програми

Методи порівняння (метод косиносної подібності) та побудови рівних векторів документів, а також інших частин програмного засобу (методів для роботи кнопок GUI, необхідних констант тощо) реалізовані в класі PlagiarismDetector.ViewModel.MainViewModel.

3 МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ЩОДО ЗАСТОСУВАННЯ РЕСУРСУ UNICHECK СТРУКТУРНИМИ ПІДРОЗДІЛАМИ УНІВЕРСИТЕТУ

Відповідно до наведеного у розділі 1 обґрунтованого аналізу наявних програм і ресурсів виявлення плагіату, було вирішено здійснити впровадження в систему виявлення плагіату в роботах студентів і працівників університету програму Unicheck вітчизняної компанії Антиплагіат, технічні і програмні характеристики якої відповідають вимогам системи університету.

В цьому розділі надамо методичні вказівки щодо використання системи Unicheck для адміністраторів, відповідальних осіб та інших користувачів системи.

Система перевірки текстів на унікальність Unicheck швидко порівнює завантажені файли з веб-індексом в режимі реального часу, ресурсами відкритого доступу та документами, які зберігаються у кабінеті користувача. Для здійснення перевірки документа необхідно виконати відповідні етапи, зміст яких наведений у наступних пунктах.

3.1 Етап реєстрації в системі

Запрошення з реєстраційними даними облікового запису (рис.3.1) користувач отримує на електронну пошту незалежно від його ролі в системі. Роботу можна розпочати натиснувши кнопку «Розпочати зараз», після чого стає можливим доступ до веб-сайту і вхід до системи Unicheck.

Якщо вхід до системи відбувається перший раз, то користувачу необхідно уважно прочитали і при умові згоди прийняти політику конфіденційності та умови використання системи шляхом натиснення кнопки «I AGREE» (рис.3.2).

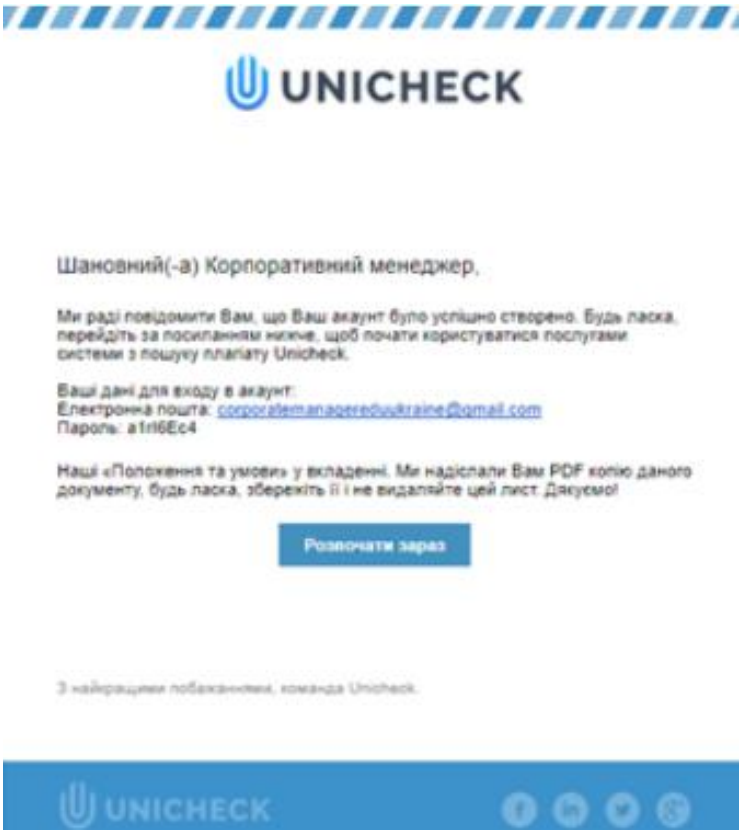

Рисунок 3.1 – Запрошення з реєстраційними даними облікового запису

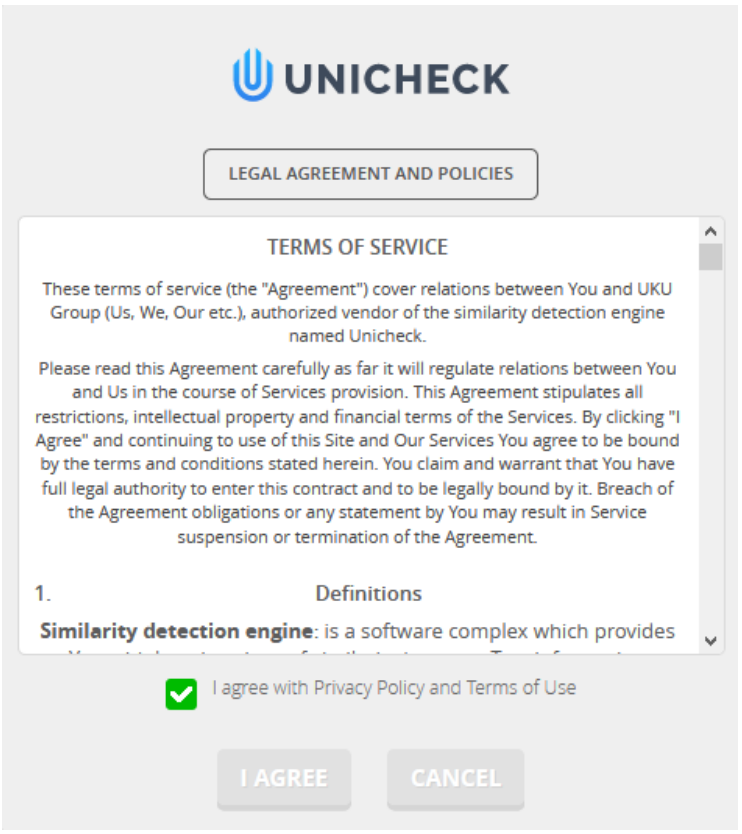

Рисунок 3.2 – Політика конфіденційності та умови використання системи

У випадку успішного створення акаунту користувач зможе увійти до нього ввівши електронну пошту та пароль, який надійшов в листі-запрошені. Пароль можна пізніше змінити в *налаштуваннях профілю*.

Для швидкого входу до акаунту можна використати вхід через кнопку «Увійти» з *Google+* або запам’ятати налаштування для входу в систему (рис.3.3). Такий вхід можливий якщо електронна пошта знаходиться на Gmail, або сервісі Google.

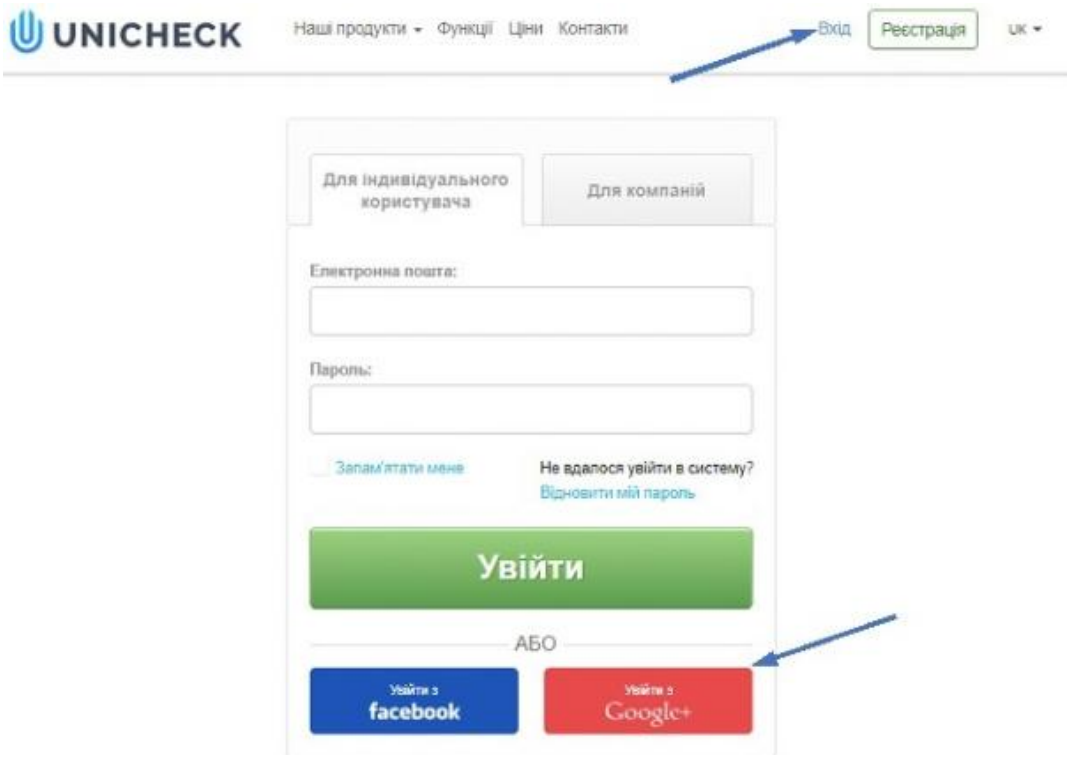

Рисунок 3.3 – Входу в акаунт через Google+

3.1.1 Налаштування профілю системи

Ввійшовши в свій обліковий запис користувач може змінити мову сторінки, а також пароль у пункті меню *Налаштування профілю*, який можна знайти у верхньому правому куті кожної сторінки (рис.3.4). Натиснувши значок *Профіль* можна перейти до пункту меню *Налаштування профілю*

(рис.3.5), а потім до розділу зміна паролю і дотримуючись інструкцій змініть пароль (рис.3.6).

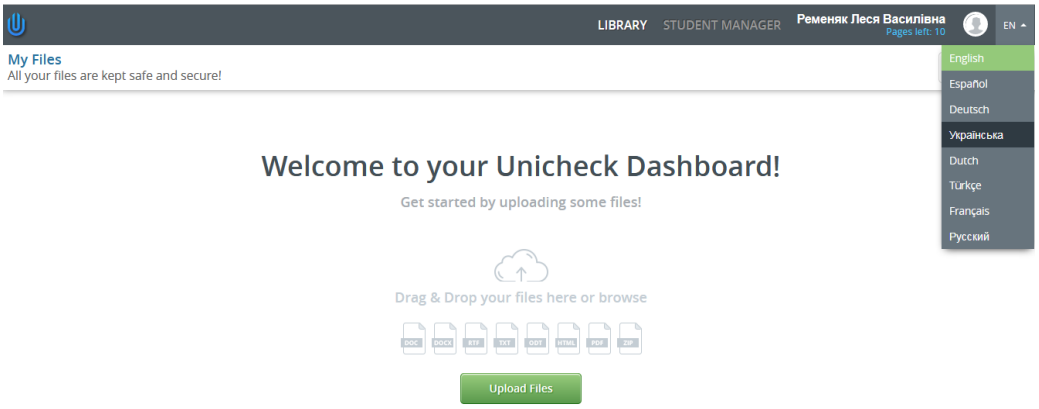

Рисунок 3.4 – Налаштування мови

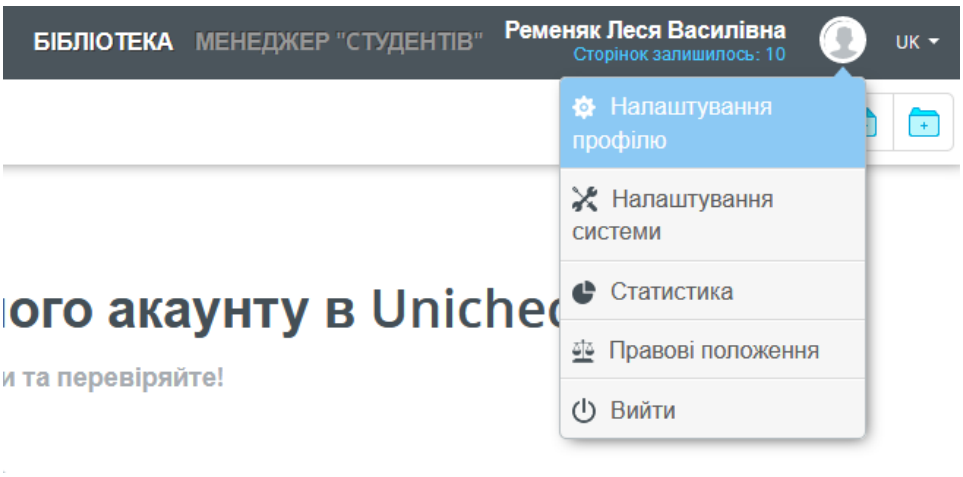

Рисунок 3.5 – пункту меню «Налаштування профілю»

3.1.2 Налаштування збереження оригіналів

Для збереження оригіналів документів, які будуть завантажуватися для перевірки Адміністратору необхідно включити опцію *Зберегти оригінали документів* в *Налаштуваннях системи* (рис.3.7). Після ввімкнення даної опції власники файлів можуть скачувати файли на локальний комп’ютер.

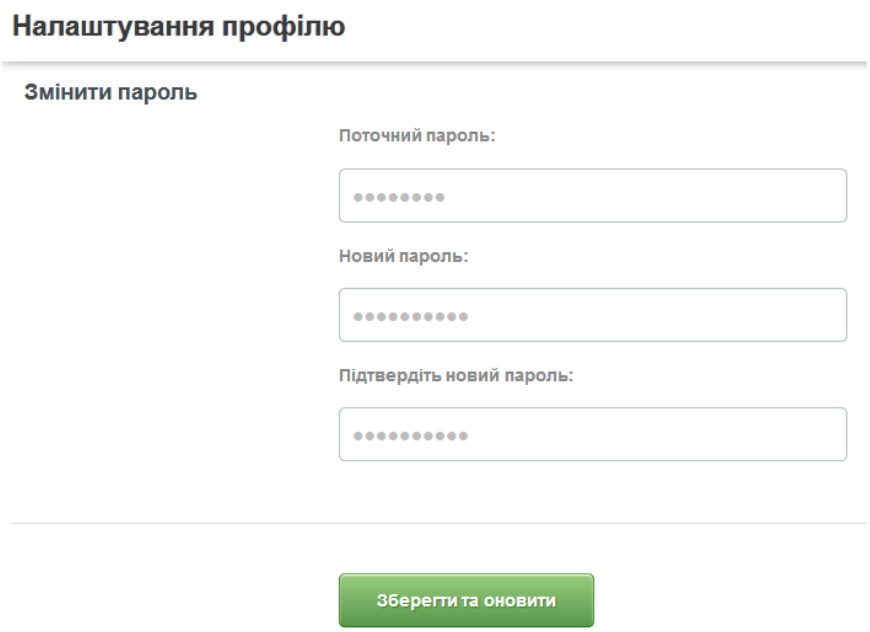

Рисунок 3.6 – Зміна паролю

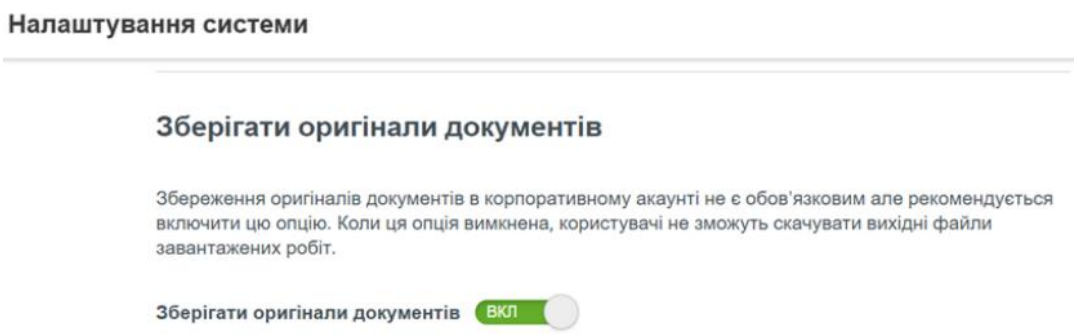

Рисунок 3.7 – Налаштування збереження оригіналів

3.2 Етап перевірки на ознаки плагіату

3.2.1 Завантаження файлів

Для того, щоб розпочати перевірку документів на плагіат, необхідно завантажити їх в бібліотеку аканту (рис.3.8) або створити новий файл. Щоб завантажити потрібний документ, створити новий файл або папку, користувачу потрібно натиснути на відповідну іконку в меню зображеному на рис.3.8.

При натисканні на іконку користувач може обрати відповідну опцію завантаження файлу: з локального комп'ютера, Google Drive, Dropbox, OneDrive (рис. 3.10). Є також можливість використовувати функцію Drag & Drop безпосередньо з бажаного джерела для додавання нових файлів в бібліотеку просто перетягнувши файл в область *Бібліотеки акаунту* (рис.3.11). Одночасно можна завантажувати до 10 файлів з розширеннями doc, docx, odt, rtf, html, txt, pdf, ppt, pptx, zip. Розмір одного файлу повинен бути не більше 70 MB.

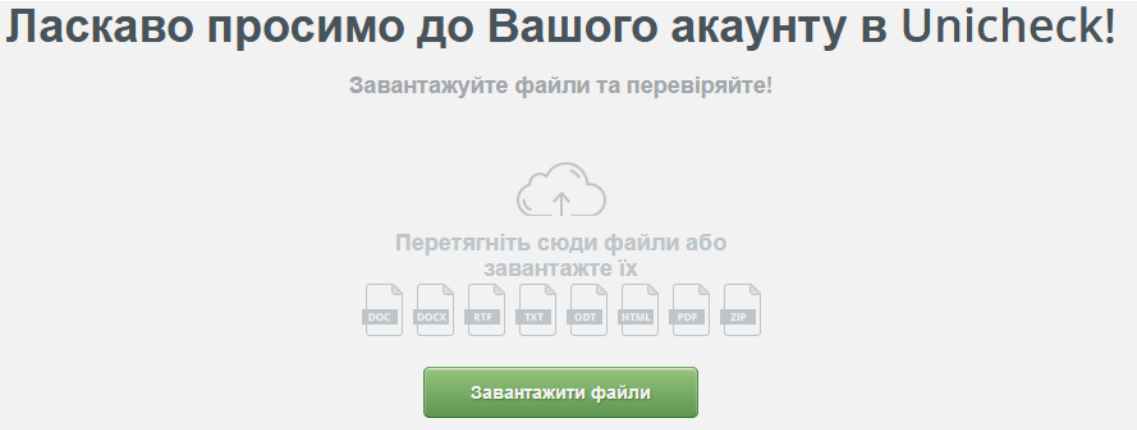

Рисунок 3.8 – Вікно завантаження файлу

Рисунок 3.9 – Меню завантаження файлу

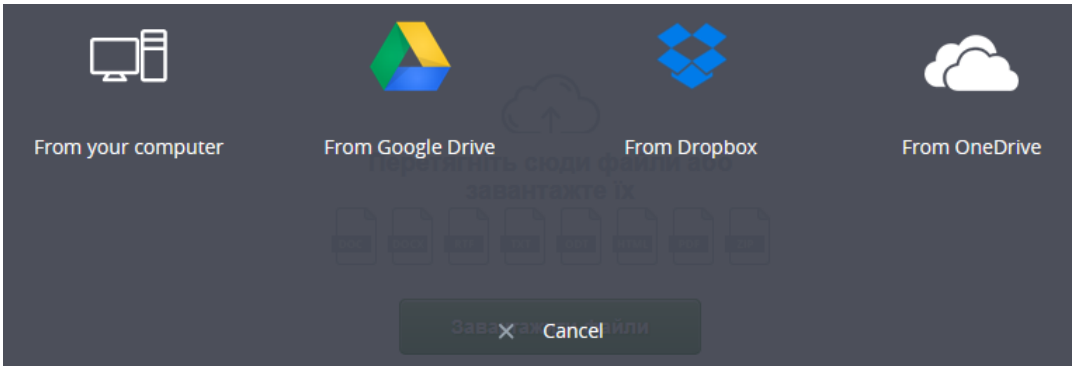

Рисунок 3.10 – Вікно вибору джерела завантаження файлу

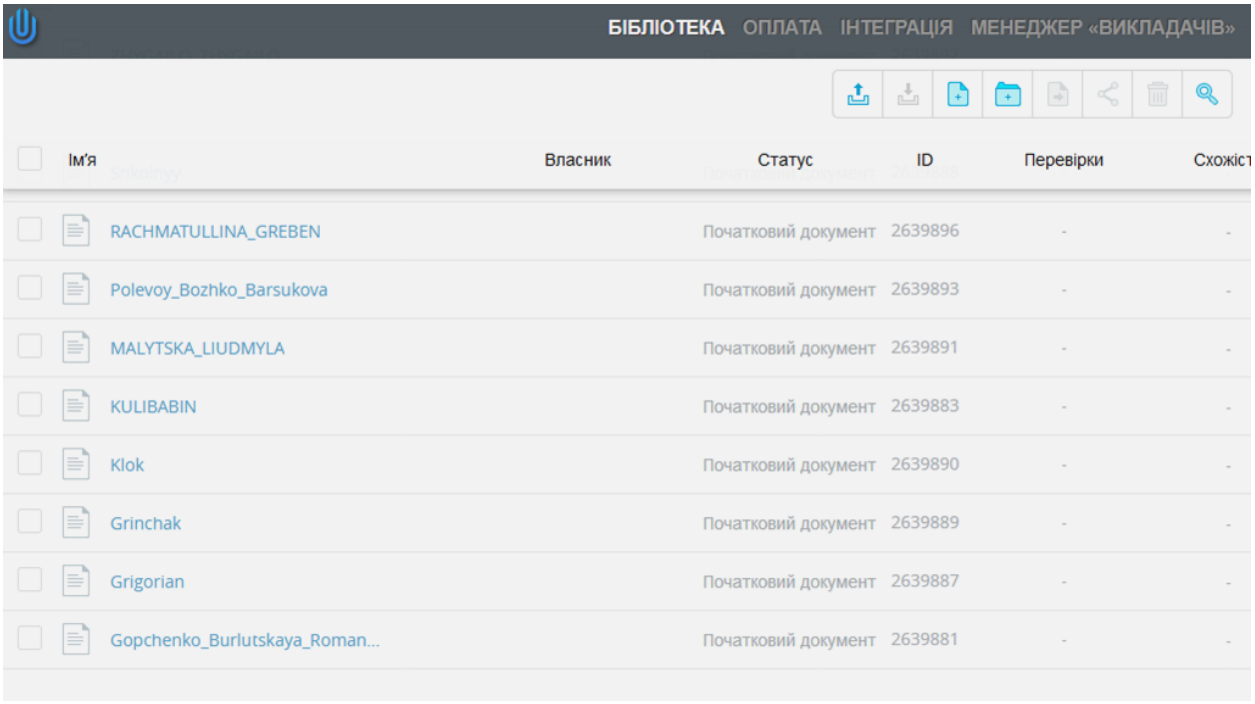

Рисунок 3.11 – Бібліотека акаунту

3.2.2 Типи перевірок

В системі є можливість вибрати декілька типів перевірок:

1) Інтернет – ця опція використовується для перевірки документу на плагіат з онлайн-джерел (одночасно можна перевіряти до 40 документів).

2) Бібліотека – ця опція використовується для перевірки документу на плагіат з одного або декількох документів або папок один з одного, щоб перевірити на плагіат з всіх файлів у власnsq бібліотеці або перевірити по всій бібліотеці університетського корпоративного акаунту (одночасно можна перевіряти до 35 файлів);

3) Інтернет+Бібліотека – ця опція використовується для перевірки документу на плагіат з онлайн-джерел і всіх файлів в університетському корпоративному аканті користувача.

Щоб почати перевірку необхідно: вибрати файл або декілька файлів в *Бібліотеці* і натиснути кнопку перевірити на плагіат, а потім вибрати один із доступних типів перевірок.

З одного акаунту одночасно можна перевіряти до 35 файлів по бібліотеці (в цьому випадку запущені файли до перевірки також перевіряються між собою).

3.2.3 Перевірка на плагіат через джерела Інтернет

Для початку перевірки через джерела Інтернет користувачу треба вибрати файли (папки), які треба перевірити з власної *Бібліотеки* та натиснути кнопку «Перевірити на схожість». У вікні що відкриється треба обрати пункт «Інтернет» (рис.3.12).

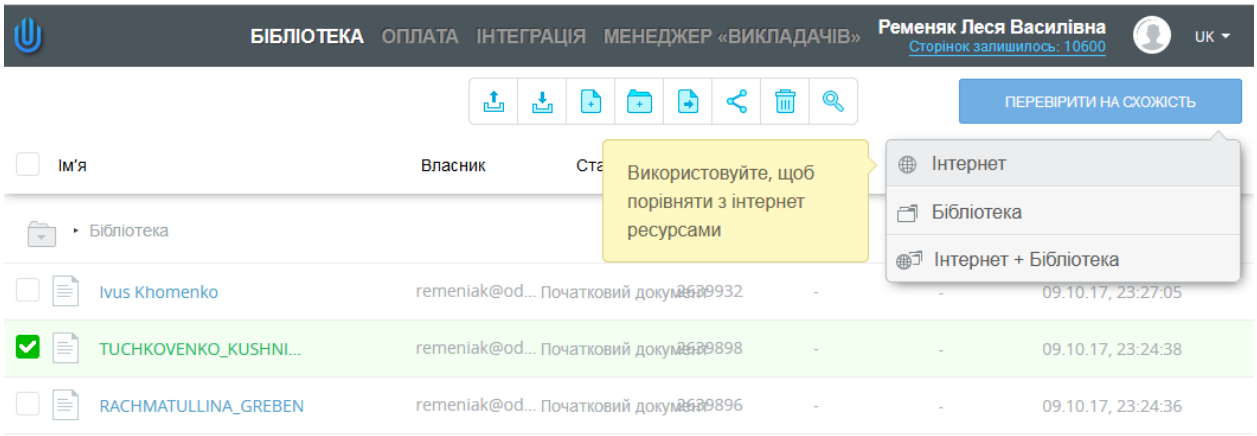

Рисунок 3.12 – Початок перевірки типу Інтернет

Відкривається вікно, в якому можна переглянути файли, які будуть перевірятись, додати або видалити файли до перевірки, та підтвердити перевірку (рис.3.13). Щоб швидко знайти документи у власній бібліотеці, можна використати рядок пошуку. Для фільтрації списку можна почати ввід імені документа. В цьому вікні також можна переглянути кількість сторінок і загальну вартість для поточної перевірки (рис.3.14). Коли все файли вибрані,

слід натиснути кнопку «Почати перевірку», щоб запустити процес сканування (3.15).

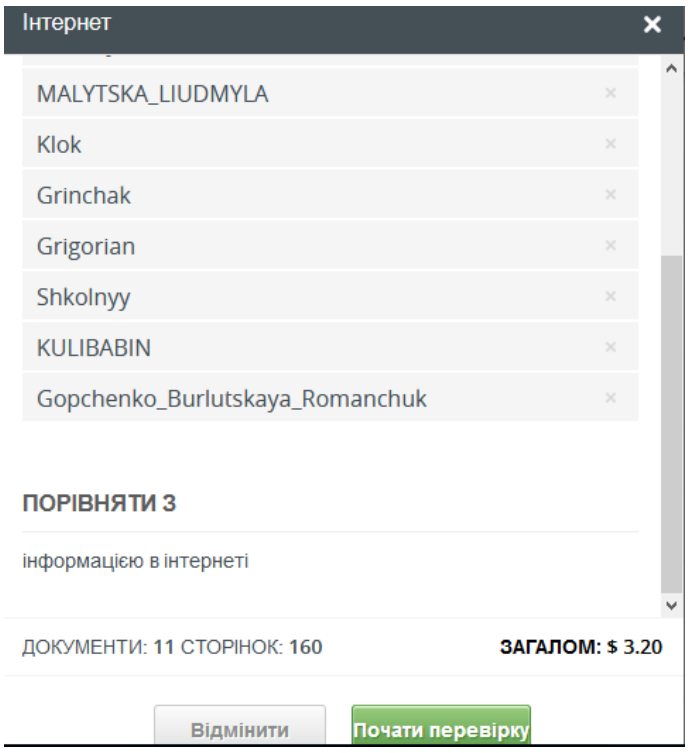

Рисунок 3.13 – Вікно файлів, що перевіряються

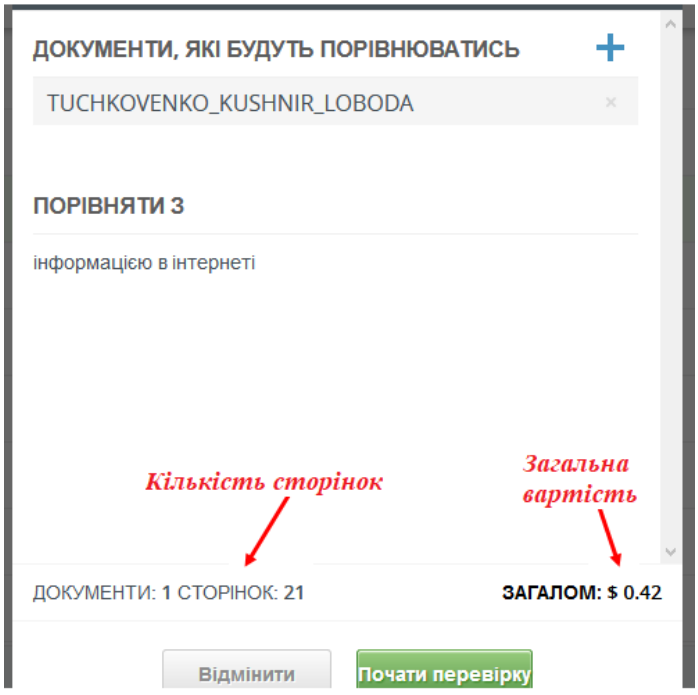

Рисунок 3.14 – Підрахунок кількості сторінок і загальної вартості

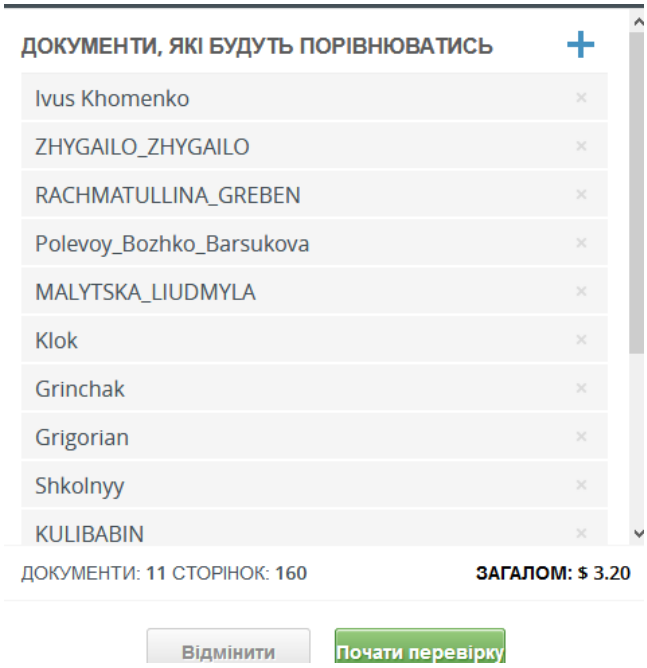

Рисунок 3.15 – Вікно документів для порівняння

3.2.4. Початок перевірки бібліотека

Для початку перевірки по *Бібліотеці* користувачу потрібно вибрати файли (папки), які треба перевірити з власної бібліотеки та натисніть кнопку «Перевірити на схожість». У вікні що відкриється обрати пункт «Бібліотека» (рис.3.16).

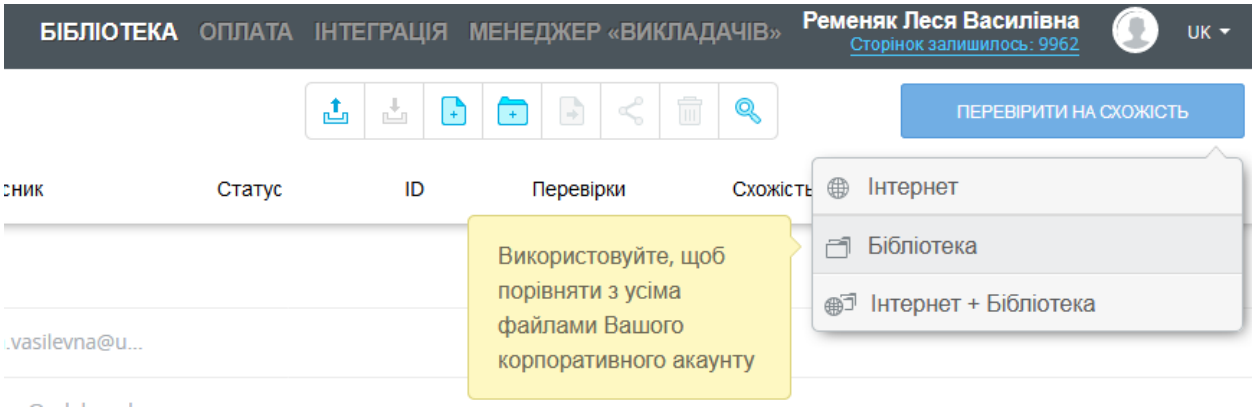

Рисунок 3.16 – Перевірка файлів по Бібліотеці

Далі буде запропоновано підтвердити перевірку на плагіат, додати чи видалити основні документи, які б користувач хотів б порівняти і документи, з якими будуть порівнюватися (*Бібліотека*). Можна порівняти з усіма файлами в Бібліотеці (рис. 3.17).

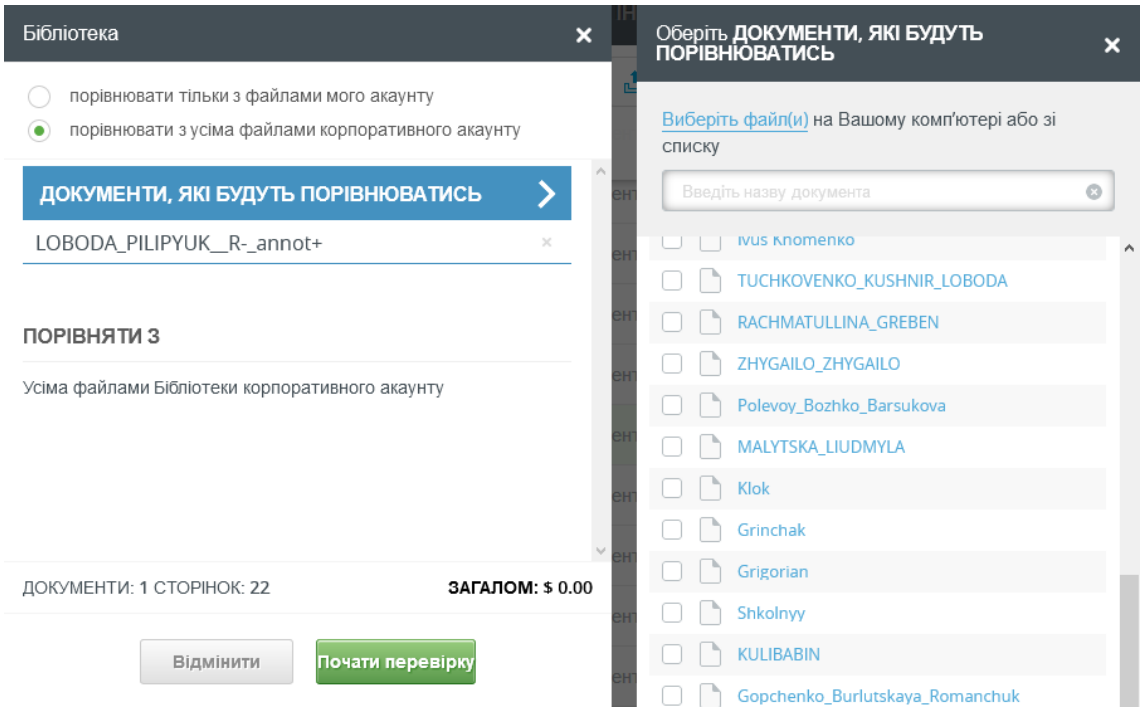

Рисунок 3.17 – Порівняння обраного файлу з файлами в Бібліотеці

3.3 Звіт подібності

Коли процес виявлення ознак плагіату запущено, користувач може відстежувати хід його виконання в колонці *Статус* завдяки індикатору, що показує який відсоток тексту опрацьовано (рис.3.18).

Після того, як перевірка завершиться, з'явиться галочка поряд із сформованим звітом подібності, відсоток подібності для кожного документу та дата, коли документ був завантажений чи перевірений (рис.3.19).

При натисненні на кнопку «Звіти», відкриється список звітів подібності (рис.3.20). Щоб відкрити конкретний звіт подібності, слід натиснути на його ім'я.

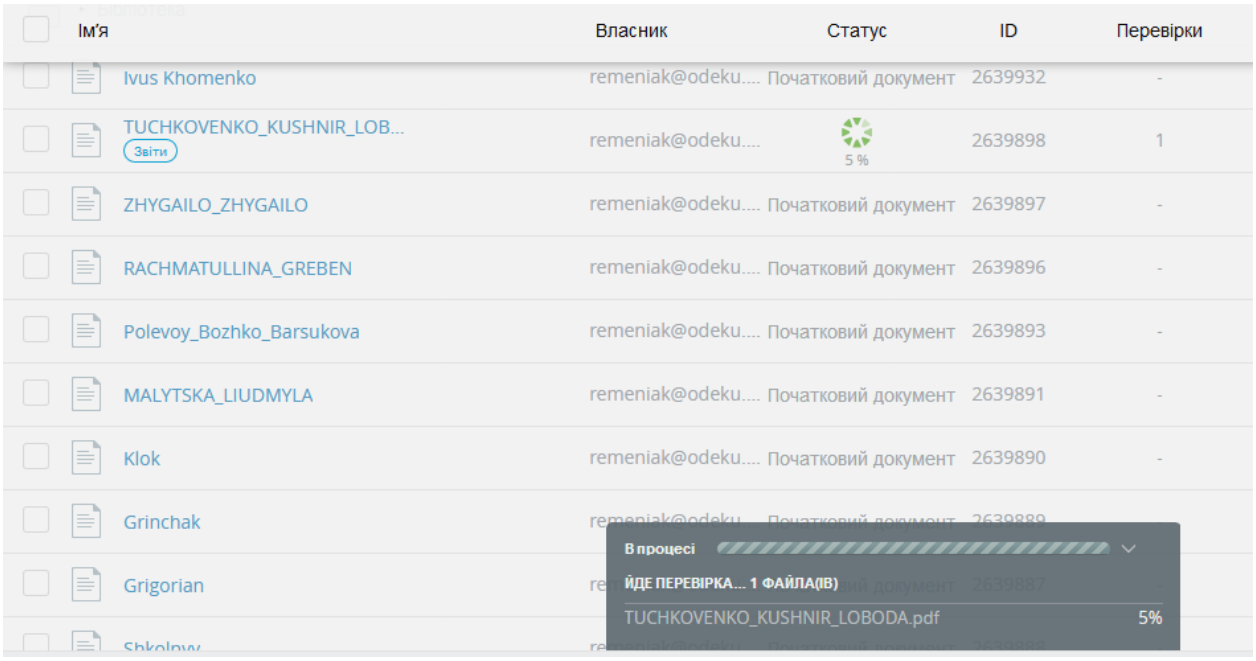

Рисунок 3.18 – Відстеження процесу перевірки

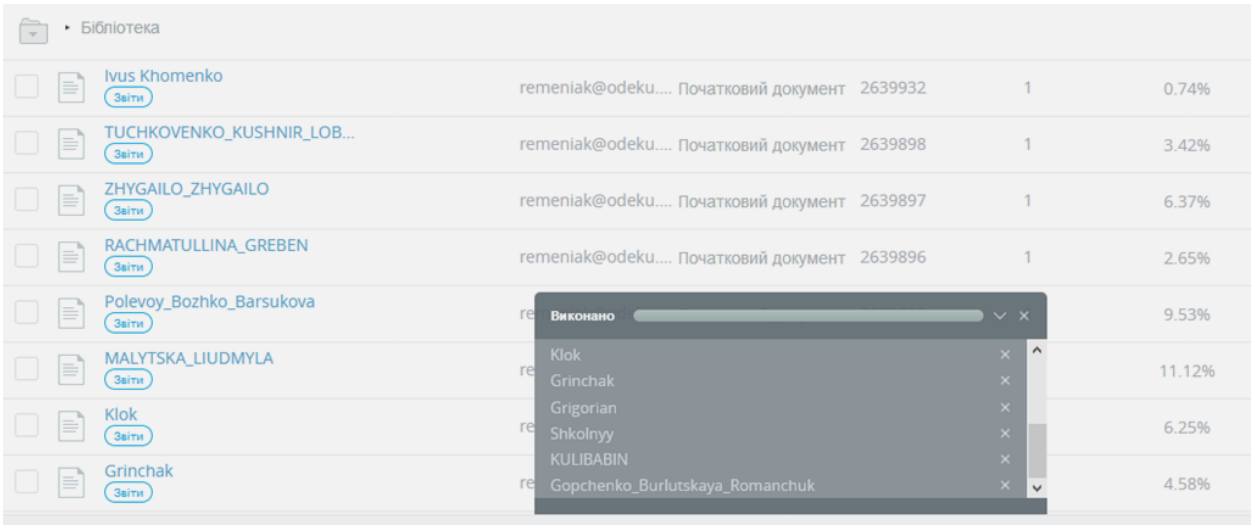

Рисунок 3.19 – Звіти перевірок

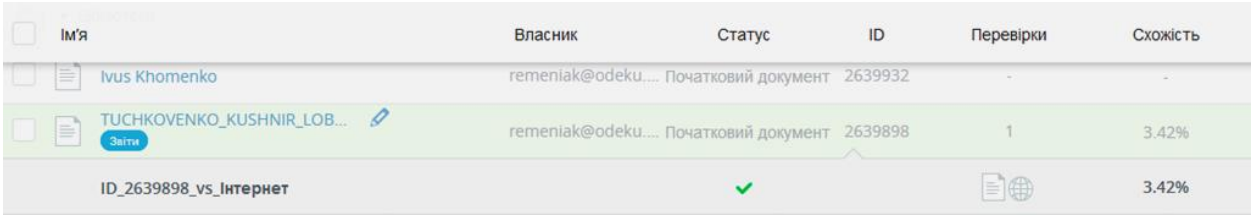

Рисунок 3.20 – Список звітів подібності

3.3.1 Знайдені текстові збіги

Перший екран звіту подібності містить текст з виділеними збігами, список з джерелами збігів, які можуть бути доданими чи прихованими і дані про виключенні джерела (рис.3.21). Користувач може натиснути на значок «Ланцюг» для перегляду збігів за певним джерелом зі збігом.

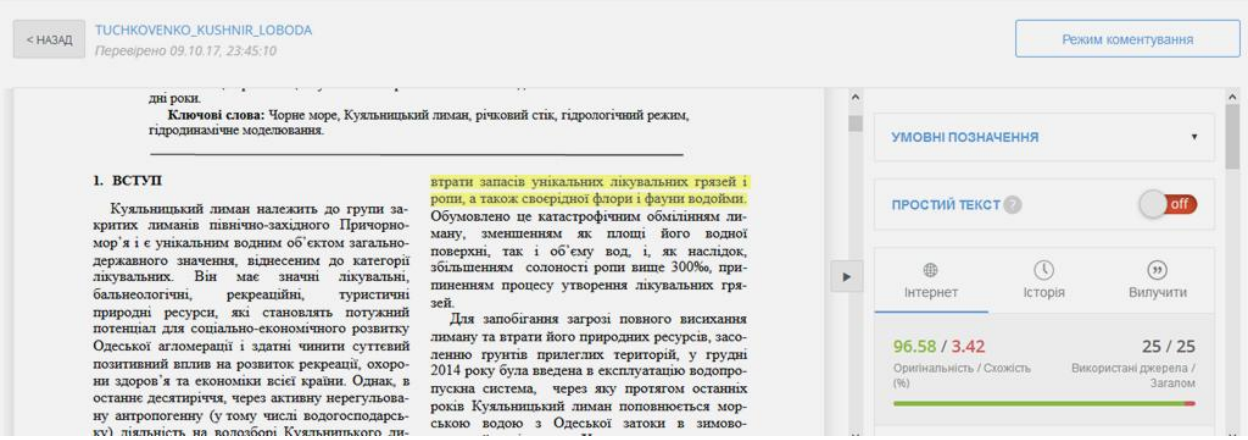

Рисунок 3.21 – Текстові збіги

3.3.2 Онлайн звіт

Кольорові позначення по тексту у онлайн звіті підсвічені кольорами згідно умовних позначень (рис.3.22).

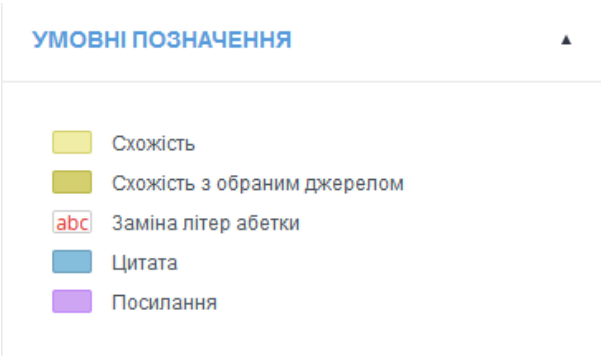

Рисунок 3.22 – Умовні позначення

Натиснувши на кожну частину тексту, яка підсвічена жовтим кольором буде показано джерело з яким було знайдено текстовий збіг (рис.3.23).

Також система групує всі частини тексту, які були знайдені з одного джерела в одному посиланні, просто натиснувши декілька раз на нього система буде переходити до кожного текстового збігу з цим джерелом у роботі (рис.3.24).

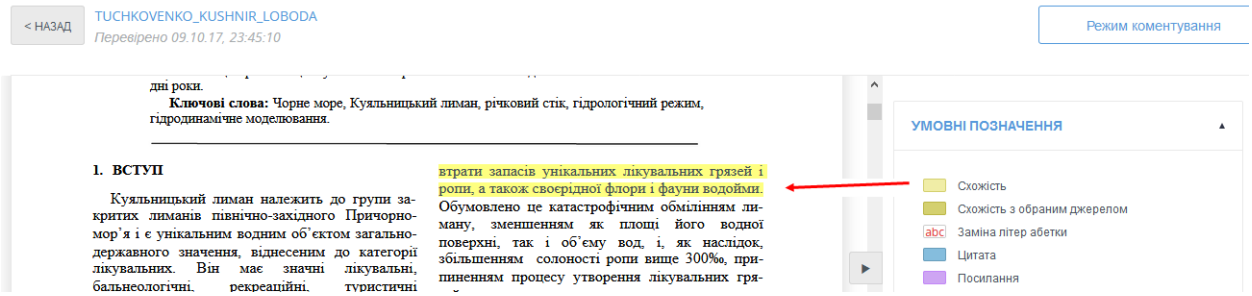

Рисунок 3.23 – Підсвідчення збігу згідно умовним позначенням

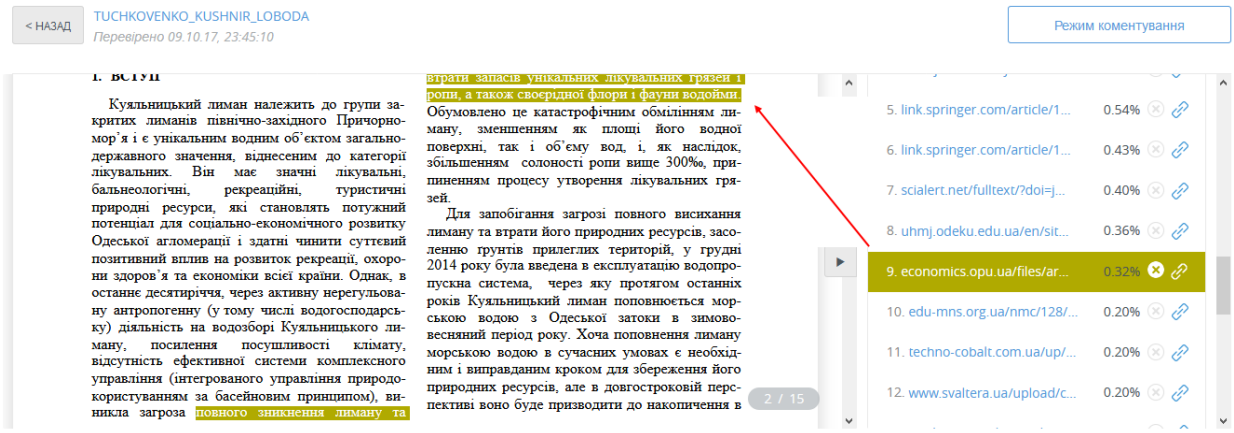

Рисунок 3.24 – Джерело з яким було знайдено текстовий збіг

3.3.3 Вилучення цитат

Щоб вилучити джерела, що уже цитувалися в звіті про плагіат, користувачу потрібно перейти на вкладку «Вилучити» і включити вилучення цитат і посилань в документі (рис.3.25). Натиснувши на будь-яку знайдену цитату, користувач може подивитись її розташування в тексті. И також може додати цитату назад до звіту, натиснувши на знак «Х» у списку цитат.

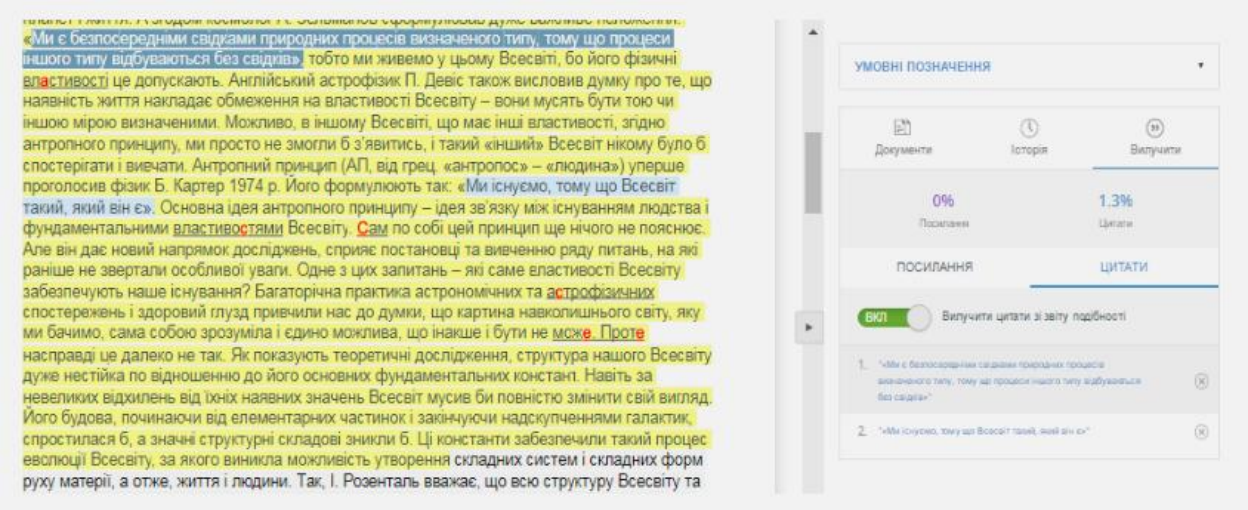

Рисунок 3.25 – Вилучення джерел, що уже цитувалися в звіті про плагіат

3.3.4 Вилучення посилань

Unicheck виявляє посилання відформатовані в стилях APA, MLA, Chicago/Turabian і Harvard. Треба перейти на вкладку «Вилучити/Посилання», щоб показати їх розташування в тексті (рис.3.26). Користувач може вручну відключити «Вилучення посилання зі звіту подібності» за допомогою перемикача.

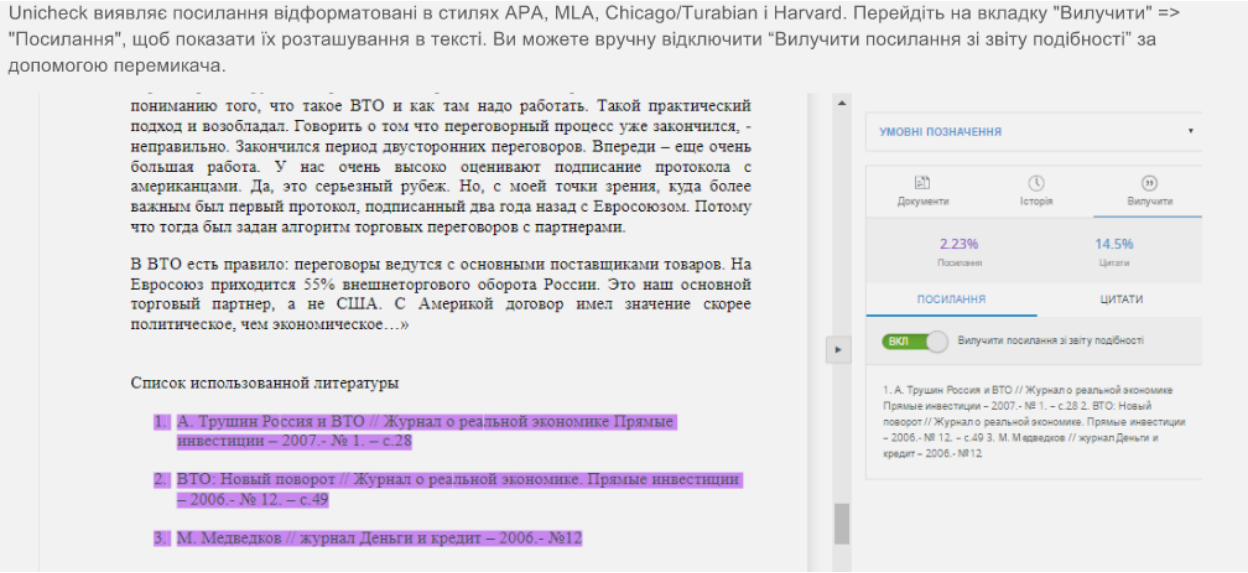

Рисунок 3.26 – Вилучення посилань

3.3.5 Режим коментування (Викладач)

Викладач має можливість залишити коментар в середині кожної студентської роботи. Для цього треба відкрити звіт про плагіат, натиснути на кнопку «Режим коментування» і вибрати область, яку треба прокоментувати. Потім з’явиться спливаюче вікно для введення коментаря (рис.3.27).

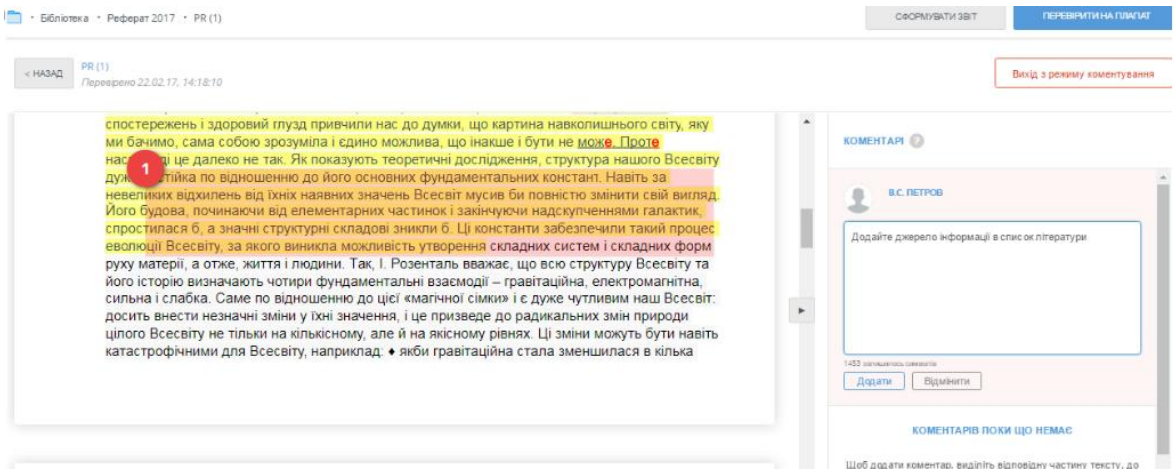

Рисунок 3.27 – Режим коментування викладачем

3.3.6 Режим коментування (Студент)

Нещодавно додані коментарі будуть з’являтися в звіті про плагіат студента разом зі значком, що показує кількість непрочитаних повідомлень. Для перегляду коментаря, треба натиснути кнопку «Режим коментування».

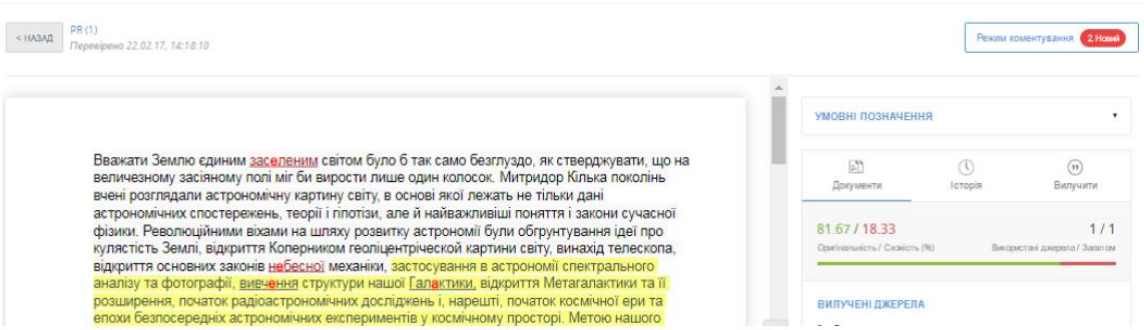

Рисунок 3.28 – Вигляд коментарів в звіті студента

3.3.7 Коментарі в PDF звіті подібності

Опублікований коментар також з’явиться у PDF версії звіту про плагіат. Для створення звіту в форматі PDF, треба натиснути на кнопку «Формування звіту» в правому верхньому кутку сторінки (рис.3.29), почекати, поки звіт створиться, і натиснути кнопку «Скачати звіт» (рис.3.30). Завантажити PDF звіт можуть як викладачі, так і студенти (якщо така функція включена в налаштуваннях).

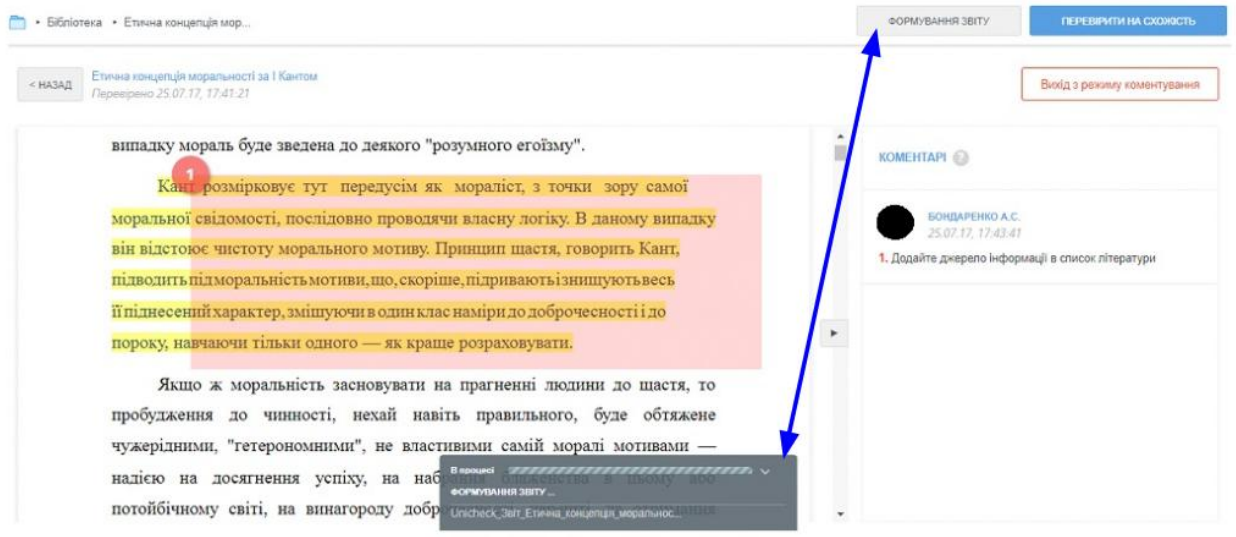

Рисунок 3.29 – Формування звіту

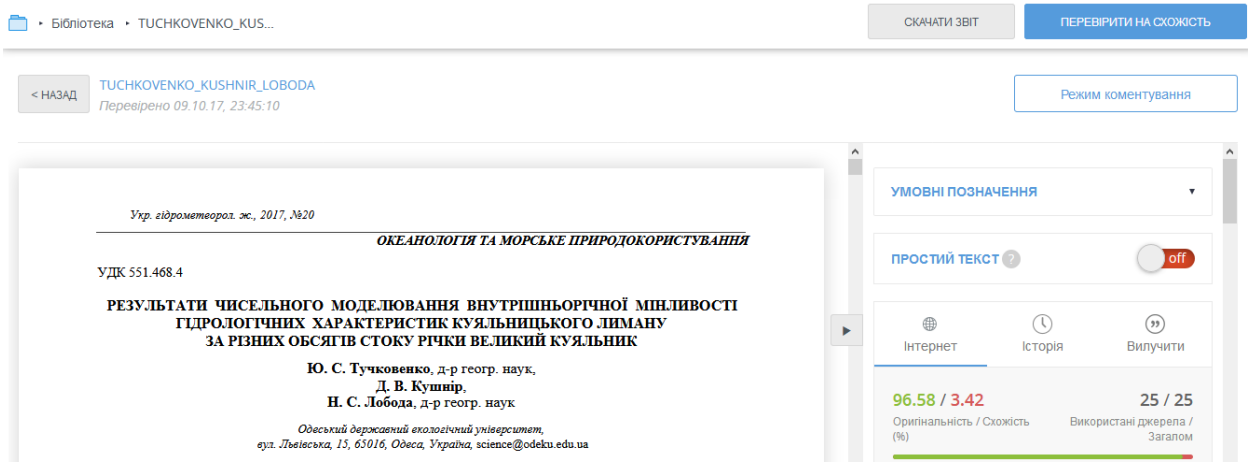

Рисунок 3.30 – Вигляд сформованого звіту

3.3.8 Звіт PDF

Звіт PDF складається з трьох розділів (рис.3.31):

1) Джерела подібності.

2) Перевіреного тексту з підсвіченими збігами, цитатами, посиланнями і коментарями до текстових частин (якщо такі є).

3) Розділу коментарів. Кожен коментар має свій номер, відповідно виділеній області в тексті.

Рисунок 3.31 – Розділи звіту PDF

3.3.9 Чутливість перевірки

Пошук збігів Unicheck можна налаштувати. Для того, щоб зробити його менш чутливим треба відкрити налаштування системи і дотримуватися інструкцій, вказаних на екрані. Користувач може вибрати необхідний відсоток, або бажану кількість слів, що необхідно виключити зі звіту.

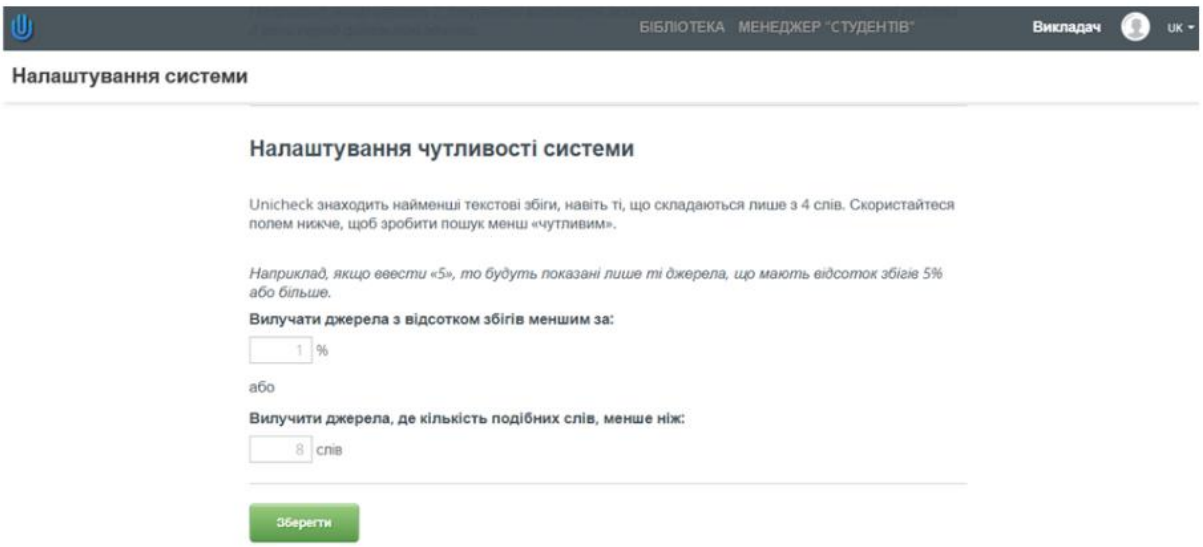

Рисунок 3.32 – Вікно налаштування чутливості системи

3.4 Менеджер викладачів

3.4.1 Додавання нових викладачів

Щоб побачити список всіх доступних викладачів у системі, а також зарезервовані за кожним викладачем та використані ним кошти, треба перейти в меню «Менеджер Викладачів» (доступний для адміністратора корпоративного акаунту). Для додавання нових викладачів в систему необхідно натиснути кнопку «Додати Нового Викладача» (рис.3.33).

Корпоративний адміністратор має дві функції для того, щоб додати нового викладача:

1) Додавання кожного користувача по одному, ввівши повне ім’я та адресу електронної пошти (рис.3.34).

2) Додавання багатьох користувачів одночасно, ввівши список адрес електронної пошти (рис.3.35). В цьому випадку адміністратор побачить ім’я викладача, як тільки він прийме запрошення і оновить особистий профіль (рис.3.36).

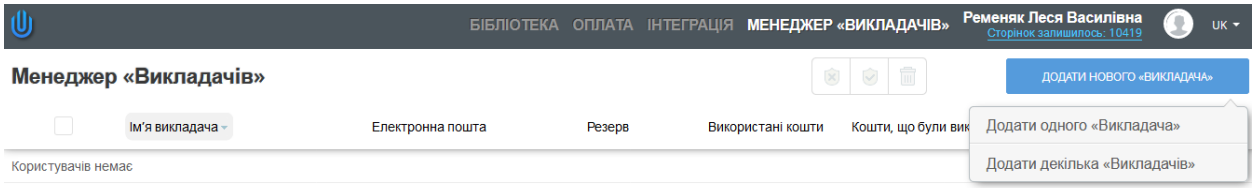

Рисунок 3.33 – Меню «Менеджер викладачів»

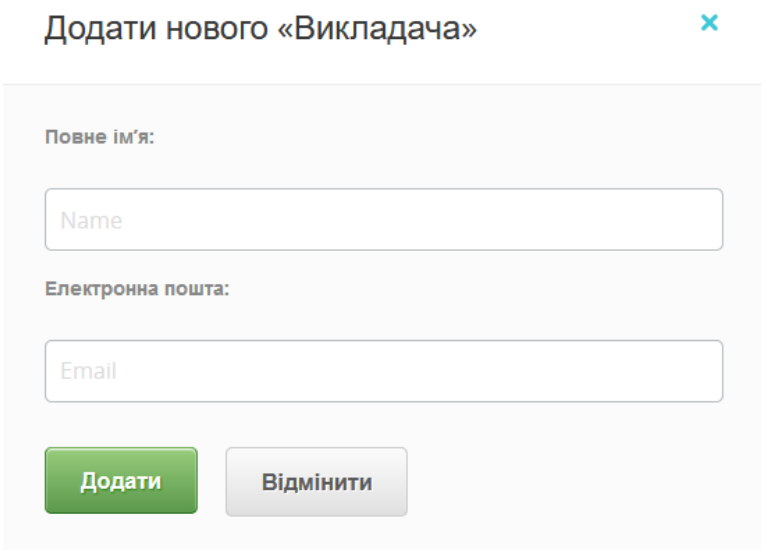

Рисунок 3.34 – Вікно додавання нового Викладача

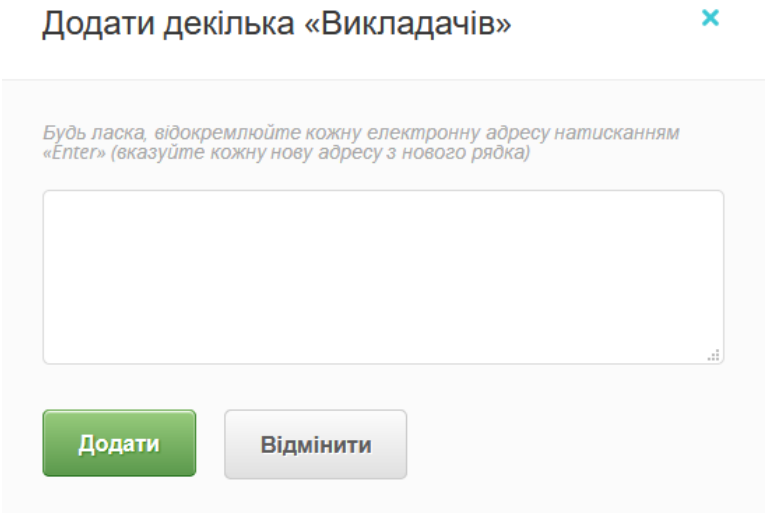

Рисунок 3.35 – Вікно додавання декількох Викладачів

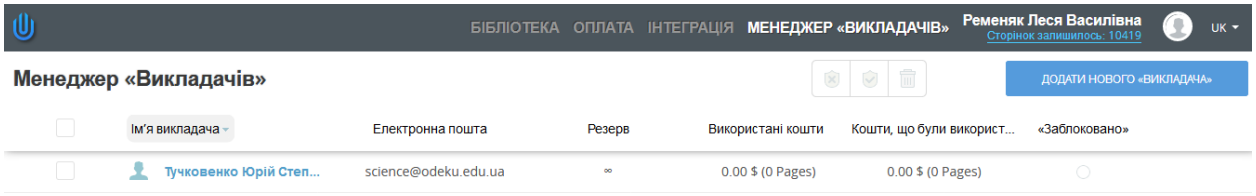

Рисунок 3.36 – Добавлений Викладач

3.4.2 Резервування для викладачів

Щоб визначити резерви для викладачів, у стовпчику «Резерв» треба натиснути на позначення ∞ (рис.3.37). Відкриється кількість сторінок резерву для викладачів (рис.3.38).

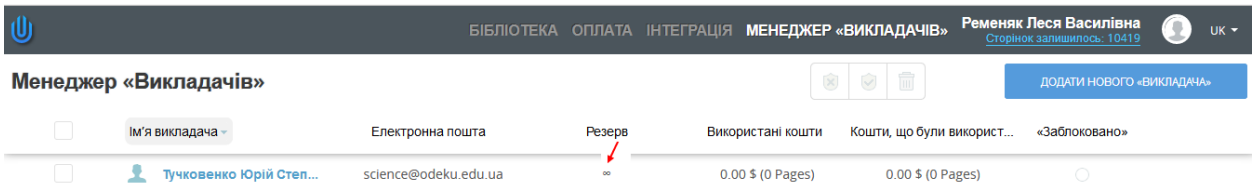

Рисунок 3.37 – Резерв для Викладача

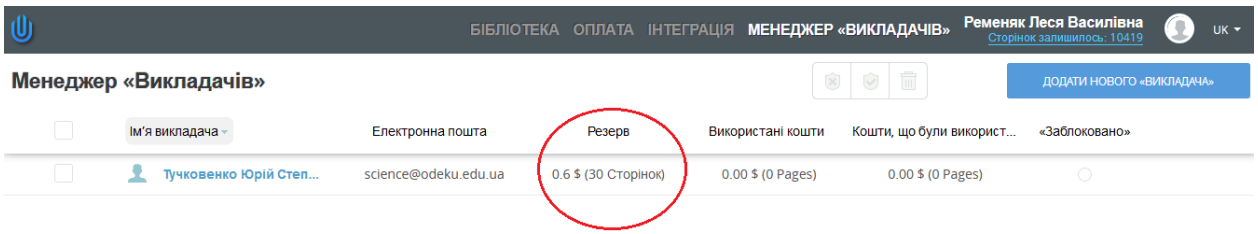

Рисунок 3.38 – Кількість наданих сторінок

3.4.3 Управління викладачами

Кожен раз, коли адміністратор додаєте нового користувача в систему, в бібліотеці створюється нова папка з ім'ям відповідним адресі електронної пошти користувача (рис.3.39). Адміністратор завжди можете заблокувати, розблокувати чи видалити користувачів з «Менеджера викладачів» (рис.3.40).

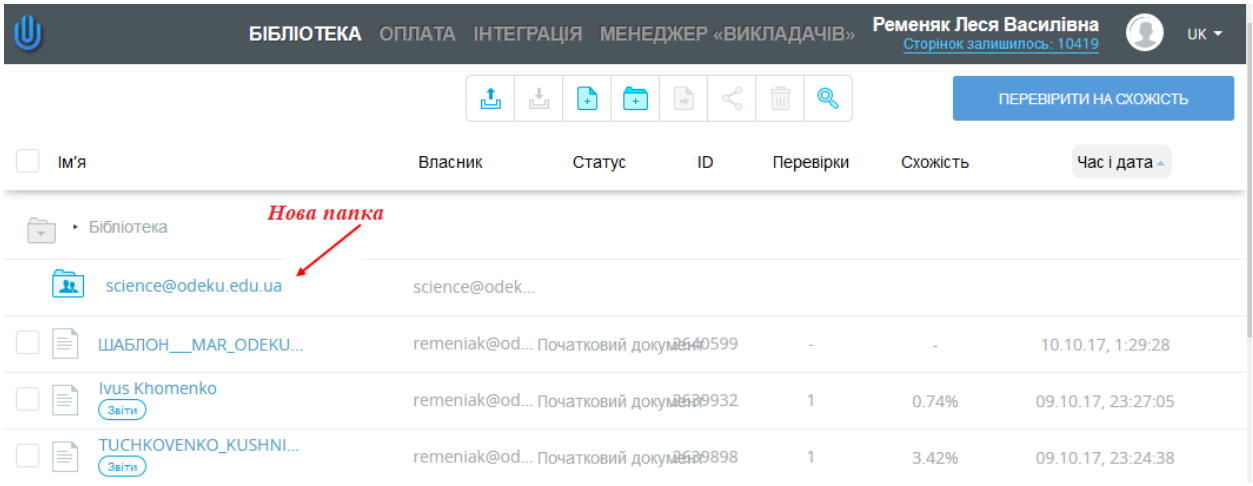

Рисунок 3.39 – Папка з ім'ям відповідним адресі електронної пошти користувача

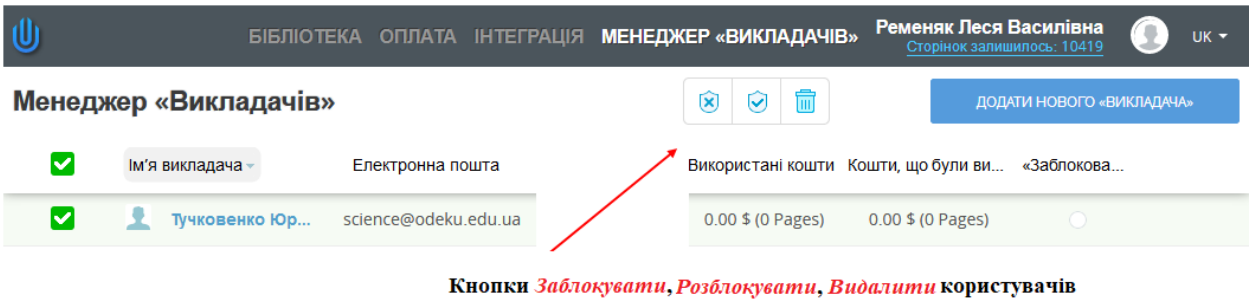

Рисунок 3.40 – Дії з новим користувачем системи

ВИСНОВКИ

У розділі 1 НДР на підставі наданої класифікації інструментів автоматичного відстеження плагіату у текстових документах і визначених програмно-технічних характеристик, проведено дослідження наявних у вільному доступі і пропрієтарних програмних засобів та онлайн ресурсів. Встановлено, що більшість ресурсів, що були проаналізовані, не відповідають вимогам, яким повинна відповідати система перевірки на плагіат творів працівників та студентів університету. До недоліків ресурсів слід віднести: відсутність можливості підключення внутрішньої бази, яка наповнюється безпосередньо науковими роботами працівників та студентами університету; відсутність підтримки мультимовності; обмежений набір форматів файлів, яким оперує ресурс; обмеження щодо об'єму введеного тексту і кількості перевірок тощо.

З огляду на потрібні технічні характеристики системи перевірки на плагіат творів працівників та студентів університету, перш за все, можливості інтегрування у єдиній базі пошуку як ресурсів мережі Інтернет так і внутрішньої (локальної) бази даних університету, в якості найбільш оптимального програмного ресурсу був запропонований сервіс вітчизняного розробника «Антиплагіат» Unichek.

Сервіс Unicheck може використовуватись як окремий онлайн ресурс, а також інтегруватись з навчальними системами (LMS-Learning Management Systems, наприклад, Moodle). Підтримує мультимовність пошуку. Генерує інформативний звіт про перевірку текстів у різних форматах (PDF, DOC, DOCX, ODT, RTF, TXT, HTML, ZIP). Надає можливість одночасної перевірки до 100 робіт через Інтернет та до 35 робіт через внутрішню базу із одного аканту. Надає широкі можливості щодо регулювання параметрів пошуку плагіату. Тому з огляду наведених програмно-технічних характеристик, сервіс відповідає вимогам системи університету, тому важливим етапом було тестування і адаптація ресурсу, а також розробка

регламенту перевірки і методичних рекомендацій щодо впровадження сервісу в окремих підрозділах університету.

В розділі 2 НДР були наведені етапи розробки власного десктопного програмного забезпечення, для встановлення на локальних комп'ютерах у відповідних підрозділах університету, які безпосередньо виконують перевірку творів, створених студентами та працівниками університету на плагіат. Програмне забезпечення виконано мовою програмування C# в купі з системою для побудови клієнтських додатків WPF. СКБД для функціонування системи вибрана MongoDB, це надає ряд переваг, таких як швидкість CRUD операцій тощо.

В програмі реалізовано алгоритм перевірки TF-IDF, заснований на розрахунку ваги слова, пропорційній кількості вживання цього слова в документі, і обернено пропорційній частоті вживання слова в інших документах колекції. Схожість слів перевірялась за допомогою відстані Левенштейна.

Завдяки моделюванню ІС були визначені: вимоги, які надаються до даної ІС; спроектована певна структура СКБД; описані функціональні можливості користувачів системи; спроектовані етапи роботи програмних засобів системи, які вони проходять перед виконанням своїх цілей.

Розроблена система має привабливий, функціональний, зручний і інтуїтивно-зрозумілий інтерфейс користувача з можливістю його швидкої модернізації у разі потреби. Підтримує перевірку на плагіат для англійської, української та російської мови. Дозволяє додавати у БД файли з розширеннями: doc, docx, rtf, txt, otd, pdf. Нажаль система не є кросплатформеною, оскільки C# WPF не дозволяє компілюватись і виконуватись на ОС, відмінних від Windows.

Система може бути використана дослідниками чи іншими програмістами для написання на її основі своєї подібної системи, оскільки всі інші реалізації закритих систем (тобто тих, що використовують в свої роботі власні локальні бази даних) недоступні для всіх.

В розділі 3 НДР розроблені методичні рекомендації щодо практичного застосування програмного засобу Unichek структурними підрозділами університету.

За результатами виконання роботи апробовані наявні програмні засоби для перевірки на плагіат, виконано обґрунтований вибір та доопрацьовування програмних засобів, які будуть впроваджені у навчально-наукову діяльність університету, розроблена структура бази даних творів студентів та працівників університету та процедура перевірки творів на плагіат.

Результати НДР є основою системи перевірки творів на плагіат в університеті. Програмні засоби впроваджені в навчально-виробничу діяльність університету: в підрозділи, які безпосередньо виконують перевірку творів, створених студентами та працівниками університету на плагіат.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шарапова Е.В., Шарапов Р.В. Универсальная система проверки текстов на плагиат «Автор.net» // Информатика и её применения. 2012. № 3 (6). С. 52–58.

2. Asim M. El Tahir Ali, Hussam M. Dahwa Abdulla Overview and Comparison of Plagiarism Detection Tools // CEUR Workshop Proceedings. 2011. Vol. 706. P. 161–172. URL: http://ceur-ws.org/Vol-706/poster22.pdf (дата звернення: 18.04.2017).

3. Nahas M.N. Survey and Comparison between Plagiarism Detection Tools // American Journal of Data Mining and Knowledge Discovery. 2017. № 2(2). P. 50–53. URL: http://www.sciencepublishinggroup.com/j/ajdmkd (дата звернення: 18.04.2017).

4. Болілий В.О., Копотій В.В. Перевірка унікальності тексту при оцінюванні студентських робіт творчого або дослідницького характеру // Наукові записки НДУ ім. Гоголя, Психолого-педогогічні науки // Збірник наукових праць. 2011. № 7. С. 134–145.

5. Шинкаренко В.І., Куропятник О.С. Система контролю плагіату в студенських роботах // Восточно-Европейский журнал передовых технологий. Харьков, 2012. № 4/2 (58). С.34–38.

6. Поповський О. І. Огляд програм порівняльного аналізу на збіг // Збірник тез доповідей 2-го Кіровоградського соціально-економічного форуму «Інформаційне суспільство і влада». Кіровоград. 2013. С. 99–100.

7. Мокін В. Б. Автоматизована система перевірки текстів на плагіат / В. Б. Мокін, В. В. Войтко, С. В. Бевз, О. В. Гавенко, І. А. Білоус // Вісник Вінницького політехнічного інституту. 2010. №5. С. 12–17.

8. Лупаренко Л.А. Інструментарій виявлення плагіату в наукових роботах: аналіз програмних рішень // Інформаційні технології і засоби навчання. 2014. Том 40. №2. С. 22–28.

9. Сервіс перевірки на плагіат «АНТИПЛАГИАТ». URL: https://www.antiplagiat.ru (дата звернення: 18.04.2017).

10. Сервіс перевірки на плагіат «TEXT.RU». URL: https://text.ru (дата звернення: 18.04.2017).

11. Сервіс перевірки на плагіат «Duplichecker». URL: http://www.duplichecker.com (дата звернення: 18.04.2017).

12. Сервіс перевірки на плагіат «Unicheck». URL: https://unicheck.com/ (дата звернення: 18.04.2017).

13. Сервіс перевірки на плагіат «StrikePlagiarism». URL: www.strikeplagiarism.com (дата звернення: 18.04.2017).

14. Сервіс перевірки на плагіат «Advego Plagiatus». URL: https://advego.ru/plagiatus/ (дата звернення: 18.04.2017).

15. Шилдт Герберт. Java 8. Полное руководство. М.: Издательский дом “Вильямс”, 2015. 1376 с.

16. Лав Р. Linux. Системное программирование. 2-е изд. СПб.: Питер, 2015. 448 с.: ил.

17. Doug Hellmann. The Python Standard Library by Example // Addison-Wesley Professional. 2011. 1344 с.

18. Офіційний сайт Microsoft Visual C# .NET. URL: https://www.visualstudio.com/ (дата звернення: 18.04.2017).

19. Офіційний сайт PostgreSQL. URL: http://www.postgresql.org/ (дата звернення: 18.04.2017).

20. Hawkins Tim, Plugge Eelco, Membrey Peter. The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing (1st ed.). Apress, 2002. pp. 350.

21. Бондарев В. М. Программирование на C++: Учеб. пособие. Харьков: СМИТ, 2004. 294 с.

22. Зеленков Ю.Г., Сегалович И.В. Сравнительный анализ методов определения нечетких дубликатов для WEB-документов // Труды 9-ой Всероссийской научной конференции «Электронные библиотеки:

перспективные методы и технологии, электронные коллекции» RCDL’2007. Переславль-Залесский, 2007. Т. 1. С. 166–174.

23. Соколов А. М. Исследование ускоренного поиска близких текстовых последовательностей с помощью векторных представлений // Кибернетика и системный анализ. 2008. №4. С. 26-42.

24. Соколов А. М. Векторные представления для эффективного сравнения и поиска похожих слов // Кибернетика и системный анализ. 2007. №4. С. 18–38.

25. Тихонов В. Архитектура метапоисковых систем. URL:http://www.cmsmagazine.ru/library/items/internet_info/metasearch (дата звернення: 18.04.2017).

26. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика : учеб. пособие / Большакова Е.И., Клышинский Э.С., Ландэ Д.В., Носков А.А., Пескова О.В., Ягунова Е.В. М.: МИЭМ, 2011. 272 с.

27. Шарапов Р.В., Шарапова Е.В. Пути расширения булевой модели поиска // Информационные системы и технологии. Известия Орел ГТУ – Орел: ОрелГТУ, 2009. №6(56). С. 74-78.

28. Broder A. On the resemblance and containment of documents // Compression and Complexity of Sequences (SEQUENCES'97). IEEE Computer Society. 1998. P. 21-29.

29. Диковицкий В. В., Шишаев М. Г. Обработка текстов естественного языка в моделях поисковых систем // Сборник научных трудов. 2010. №3. С. 54–60.

30. Лифцишин Ю. Модели информационного поиска. URL: http://yury.name/internet/03ianote.pdf (дата звернення: 18.04.2017).

31. Маннинг Д. К., Рагхаван. П., Шютце Х. Введение в информационный поиск. М.: Вильямс, 2011. 512 с.: ил.

32. Маннинг Д. К., Шютце Х. Foundations of statistical natural language processing. Массачусетс: The MIT Press Cambridge, 1999. 717 с.

33. А. В. Никитов, О. А. Орчаков, Ю. В. Чехович. Плагиат в работах студентов и аспирантов: проблема и методы противодействия // Университетское управление: практика и анализ. 2012. № 5. С. 3-68.

34. Sidorov, Grigori; Gelbukh, Alexander; Gómez-Adorno, Helena; Pinto, David. "Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model". Computación y Sistemas. 2004. 18 (3). P.491–504.

35. The Porter Stemming Algorithm. URL: https://tartarus.org/martin/PorterStemmer (дата звернення: 18.04.2017).