

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ДЛЯ АВТОМАТИЗАЦІЇ ОБРОБКИ ДОКУМЕНТІВ
DEVELOPMENT OF A SOFTWARE SYSTEM FOR
DOCUMENT WORKFLOW AUTOMATION

Виконав: студент 4 курсу, групи КНТ-122

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

КРИВЕНОК Я.О.

(ПРИЗВИЩЕ та ініціали)

Керівник ГОФМАН Є.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ДЬЯЧУК Т.С.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

КРИВЕНКА Яна Олеговича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розроблення програмного забезпечення для автоматизації обробки документів. Development of a Software System for Document Workflow Automation

керівник проєкту (роботи) к.т.н., доцент, ГОФМАН Євгеній Олександрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “07” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 10 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області обробки документів. 2. Проєктування програмного забезпечення. 3. Розробка програмного забезпечення. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГОФМАН Є.О., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання « 14 » квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент

(підпис)

Ян КРИВЕНОК

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Євгеній ГОФМАН

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 79 с., 4 табл., 21 рис., 2 дод., 26 джерел.

КОМП'ЮТЕРНА ПРОГРАМА, АВТОМАТИЗАЦІЯ ПРОЦЕСУ, АНАЛІЗ ТЕКСТУ, ТЕКСТ, КЛЮЧОВІ СЛОВА.

Об'єктом дослідження є процес інженерії програмного забезпечення у системах обробки електронних документів.

Предметом дослідження є методи та засоби для програмної інженерії процесу обробки електронних документів.

Метою дипломної роботи є скорочення часу обробки електронних документів та зменшення кількості помилок при їх опрацюванні шляхом автоматизації процесів пошуку та базового аналізу текстової інформації.

Матеріали, методи та технічні засоби: сучасні програмні та технічні засоби для розробки програмного забезпечення автоматизації обробки документів.

Для реалізації програмного забезпечення було використано:

- мову програмування Python;
- середовище розробки Visual Studio;
- бібліотеки для обробки документів (наприклад, PyPDF2, python-docx, pandas);
- бібліотеки для аналізу та обробки тексту (наприклад, re, nltk).

У роботі було застосовано такі методи:

- аналіз предметної області обробки документів;
- методи структурного аналізу даних;
- методи обробки текстової інформації;
- алгоритмічні методи пошуку та фільтрації даних.

Матеріалами дослідження виступають електронні документи різних форматів:

- текстові файли (.txt);
- документи формату .docx;
- PDF-файли;
- табличні дані (.csv).

Результати. Створено програмний застосунок для операційної системи Windows, який забезпечує автоматизовану обробку електронних документів різних форматів (TXT, DOCX, PDF, CSV), виконує пошук ключових слів та базовий аналіз текстових даних із відображенням результатів у графічному інтерфейсі користувача.

Висновки. Розроблено програмне забезпечення для автоматизації обробки електронних документів, яке дозволяє скоротити час опрацювання інформації, зменшити кількість помилок, пов'язаних із ручним аналізом, та спростити виконання рутинних операцій користувача.

Галузь використання – обробка документів у навчальних закладах, офісній діяльності підприємств та організацій, а також у повсякденній роботі користувачів із текстовими даними.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 79 pages, 4 tables, 21 figures, 2 appendixes, 26 sources.

COMPUTER PROGRAM, PROCESS AUTOMATION, TEXT ANALYSIS, TEXT, KEYWORDS.

The object of research is the software engineering process in electronic document processing systems.

The subject of the research is methods and tools for software engineering of the electronic document processing process.

The purpose of the thesis is to reduce the time required for processing electronic documents and decrease the number of errors during their handling by automating the processes of search and basic analysis of textual information. Materials, methods, and technical tools: modern software and technical tools for developing software for document processing automation.

The following were used for software implementation:

- the Python programming language;
- the Visual Studio development environment;
- libraries for document processing (for example, PyPDF2, python-docx, pandas);
- libraries for text analysis and processing (for example, re, nltk).

The following methods were applied in the work:

- analysis of the document processing subject area;
- methods of structural data analysis;
- methods of text information processing;
- algorithmic methods of data search and filtering.

The research materials are electronic documents of various formats:

- text files (.txt);

- .docx format documents;
- PDF files;
- tabular data (.csv).

Results. A software application for the Windows operating system was created, which provides automated processing of electronic documents of various formats (TXT, DOCX, PDF, CSV), performs keyword search and basic analysis of textual data with the display of results in a graphical user interface.

Conclusions. Software for automation of electronic document processing was developed, which makes it possible to reduce information processing time, decrease the number of errors related to manual analysis, and simplify the execution of routine user operations.

Field of application – document processing in educational institutions, office activities of enterprises and organizations, as well as in the everyday work of users with textual data.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	10
Вступ.....	11
1 Аналіз предметної області обробки документів	13
1.1 Загальна характеристика процесу обробки електронних документів ...	13
1.1.1 Основні типи та формати електронних документів	13
1.2 Аналіз існуючих програмних рішень для обробки документів	14
1.2.1 Microsoft Word	14
1.2.2 Adobe Acrobat	15
1.2.3 ABBYY FineReader	16
1.2.4 Google Docs.....	17
1.2.5 Порівняльна таблиця існуючих та розроблюваного програмних продуктів	18
1.2.6 Використання мов програмування для автоматизації обробки документів.....	20
1.3 Методи та технології автоматизації обробки документів	20
1.3.1 Використання бібліотек Python для обробки документів.....	21
1.4 Висновки до розділу 1	22
2 Проєктування програмного забезпечення	24
2.1 Визначення вимог до програмного забезпечення.....	24
2.1.1 Функціональні вимоги.....	25
2.1.2 Нефункціональні вимоги.....	25
2.2 Архітектура програмної системи	26
2.3 Проєктування структури програмних модулів	28
2.3.1 Структура програмного проєкту	29
2.4 Вибір технологій та засобів реалізації	30
2.4.1 Обґрунтування вибору мови програмування.....	31
2.4.2 Обґрунтування вибору середовища розробки	32

2.4.3 Обґрунтування відмови від використання бази даних.....	33
2.4.4 Вимоги до апаратного та програмного забезпечення	34
2.5 Висновки до розділу 2	35
3 Розробка програмного забезпечення.....	36
3.1 Структурна схема програмного забезпечення	36
3.2 Функціональна схема роботи програмного забезпечення	38
3.3 Реалізація модуля завантаження документів	39
3.4 Реалізація модуля обробки тексту.....	41
3.5 Реалізація модуля аналізу текстових даних	43
3.6 Реалізація модуля генерації звіту	44
3.7 Реалізація графічного інтерфейсу користувача	46
3.8 Висновки до розділу 3	50
4 Експлуатація, тестування та експериментальне дослідження програми	52
4.1 Умови та порядок експлуатації програмного забезпечення.....	52
4.2 Тестування програмного забезпечення.....	53
4.3 Експериментальне дослідження програмного забезпечення	56
4.4 Аналіз результатів експериментального дослідження.....	57
4.5 Обмеження та перспективи розвитку програмного забезпечення.....	58
4.6 Висновки до розділу 4	59
Висновки	61
Перелік джерел посилання	63
Додаток А Текст програми.....	66
Додаток Б Слайди презентації	73

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- CSV – Comma-Separated Values;
- DMS – Document Management Systems;
- DOCX – Microsoft Word Open XML Document;
- GUI – Graphical User Interface;
- OCR – Optical Character Recognition;
- PDF – Portable Document Format;
- RAM – Random Access Memory;
- TXT – Text file.

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій значна частина діяльності організацій та окремих користувачів пов'язана з обробкою великих обсягів електронних документів. До таких документів належать текстові файли, документи форматів DOCX, PDF, а також табличні дані. Зі збільшенням обсягів інформації зростає потреба у швидкій, точній та зручній обробці документів.

Традиційні підходи до роботи з документами, що передбачають значну частку ручної праці, є неефективними з точки зору витрат часу та людських ресурсів. Виконання таких операцій, як пошук інформації, аналіз змісту, сортування та структурування документів, часто супроводжується помилками та потребує значних затрат часу. Це особливо актуально для користувачів, які працюють з великою кількістю файлів різних форматів.

У зв'язку з цим актуальним є завдання автоматизації процесів обробки електронних документів. Використання програмного забезпечення дозволяє суттєво скоротити час виконання рутинних операцій, зменшити кількість помилок, підвищити зручність роботи з документами та оптимізувати використання ресурсів. Особливого значення набуває застосування універсальних інструментів, які забезпечують обробку різних форматів документів без необхідності використання складних систем керування базами даних.

Таким чином, актуальність теми дипломної роботи зумовлена необхідністю підвищення швидкості обробки електронних документів, зменшення кількості помилок при їх опрацюванні та зниження витрат часу користувачів за рахунок автоматизації відповідних процесів.

Об'єктом дослідження є процес обробки електронних документів різних форматів у інформаційних системах.

Предметом дослідження є методи та програмні засоби автоматизації обробки електронних документів, зокрема процеси їх аналізу, пошуку, структурування та перетворення даних.

Метою дипломної роботи є скорочення часу обробки електронних документів та зменшення кількості помилок при їх опрацюванні за рахунок автоматизації процесів аналізу, класифікації та пошуку інформації у документах різних форматів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати предметну область обробки електронних документів та існуючі програмні рішення;
- визначити вимоги до програмного забезпечення;
- обґрунтувати вибір методів і засобів розробки;
- розробити структуру та архітектуру програмного забезпечення;
- реалізувати програмне забезпечення для автоматизації обробки документів;
- провести тестування та оцінку результатів роботи розробленої системи.

У роботі використано сучасні методи обробки текстової інформації, алгоритми пошуку та аналізу даних, а також мову програмування Python, що дозволяє ефективно реалізувати поставлені завдання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ОБРОБКИ ДОКУМЕНТІВ

1.1 Загальна характеристика процесу обробки електронних документів

Обробка електронних документів є важливою складовою інформаційної діяльності сучасних організацій та користувачів. Вона включає сукупність процесів, пов'язаних із створенням, зберіганням, пошуком, аналізом та перетворенням документів у цифровому вигляді. З розвитком інформаційних технологій кількість електронних документів постійно зростає, що зумовлює необхідність використання спеціалізованих програмних засобів для їх ефективної обробки.

Електронні документи можуть бути представлені у різних форматах, таких як текстові файли, документи офісних форматів, PDF-файли та табличні дані. Кожен із цих форматів має свої особливості структури та обробки, що ускладнює уніфікацію процесів роботи з ними. Тому актуальним є створення універсальних програмних рішень, здатних працювати з різними типами документів.

Процес обробки електронних документів зазвичай включає такі основні етапи: введення або завантаження документа, попередню обробку, аналіз змісту, виконання пошуку необхідної інформації та формування результатів обробки. Автоматизація цих процесів дозволяє значно скоротити час роботи користувача та підвищити точність отриманих результатів [1].

1.1.1 Основні типи та формати електронних документів

Електронні документи можуть класифікуватися за різними ознаками, зокрема за форматом збереження, структурою та призначенням. Найбільш поширеними форматами є текстові файли (.txt), документи формату .docx, PDF-файли, а також табличні файли (.csv, .xlsx).

Текстові файли є найпростішим видом документів і містять неструктуровану текстову інформацію. Документи формату .docx мають складнішу структуру, що включає форматування, стилі, таблиці та інші елементи. PDF-файли призначені для збереження документів у незмінному вигляді незалежно від програмного середовища, що ускладнює їх обробку. Табличні формати використовуються для зберігання структурованих даних у вигляді рядків і стовпців.

Різноманітність форматів електронних документів створює необхідність використання різних підходів до їх обробки та аналізу. Саме тому при розробці програмного забезпечення важливо враховувати специфіку кожного формату та забезпечити підтримку основних типів документів [2].

1.2 Аналіз існуючих програмних рішень для обробки документів

На сучасному ринку програмного забезпечення існує значна кількість рішень для обробки електронних документів. До найбільш поширених належать Microsoft Word, Adobe Acrobat, ABBYY FineReader та Google Docs.

1.2.1 Microsoft Word

Microsoft Word є одним із найпоширеніших текстових процесорів у світі, який входить до складу пакета Microsoft Office. Програма надає широкий спектр можливостей для створення, редагування та форматування текстових документів, включаючи використання стилів, шаблонів, таблиць, графіки та вбудованих об'єктів. Крім того, Word підтримує перевірку орфографії та граматики, роботу з рецензіями, коментарями та спільне редагування документів у режимі реального часу. Однак, незважаючи на ці переваги, можливості автоматизованого аналізу тексту (зокрема, семантичного аналізу або обробки природної мови) у Word залишаються обмеженими та потребують

використання додаткових засобів або сторонніх плагінів. На рисунку 1.1 зображено головну сторінку цього текстового редактора.

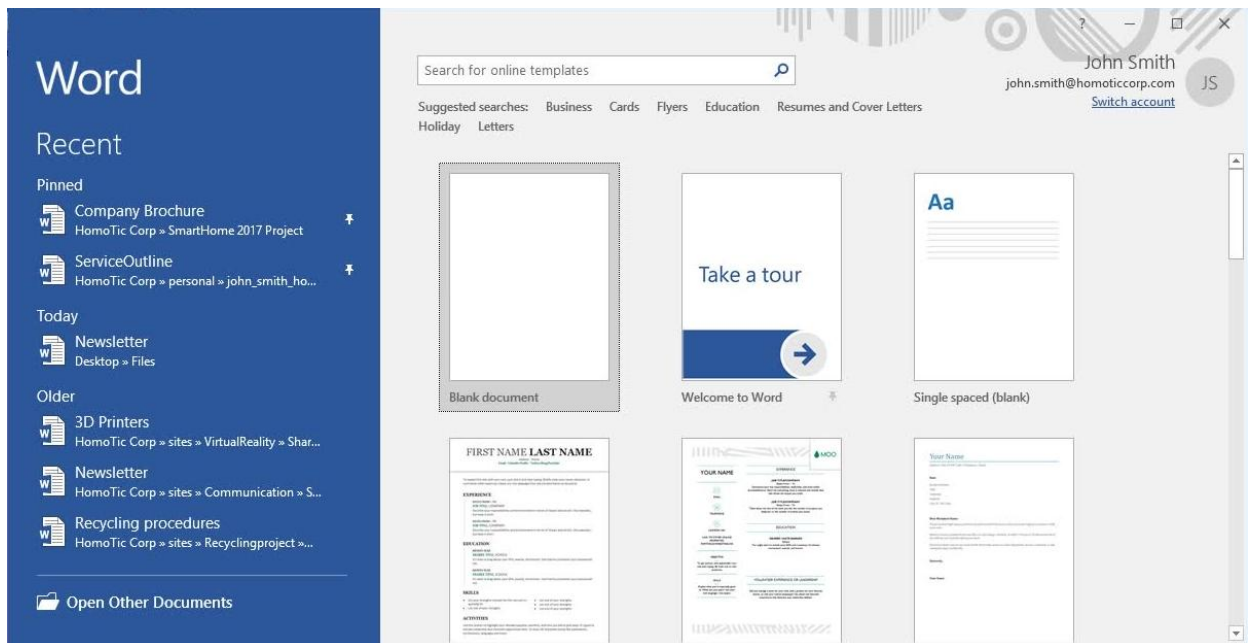


Рисунок 1.1 – Головна сторінка Microsoft Word [3]

1.2.2 Adobe Acrobat

Adobe Acrobat є спеціалізованим програмним забезпеченням для роботи з документами у форматі PDF, розробленим компанією Adobe. Основними функціями програми є перегляд, створення, редагування та підпис PDF-документів, а також додавання коментарів, анотацій та засобів захисту інформації. Acrobat дозволяє об'єднувати декілька документів в один файл, виконувати конвертацію у різні формати та забезпечує інтеграцію з іншими сервісами. Водночас, попри широкий функціонал роботи з PDF, програма не орієнтована на глибокий аналіз текстових даних, що обмежує її використання у задачах інтелектуальної обробки інформації. На рисунку 1.2 зображено головну сторінку цього програмного засобу.

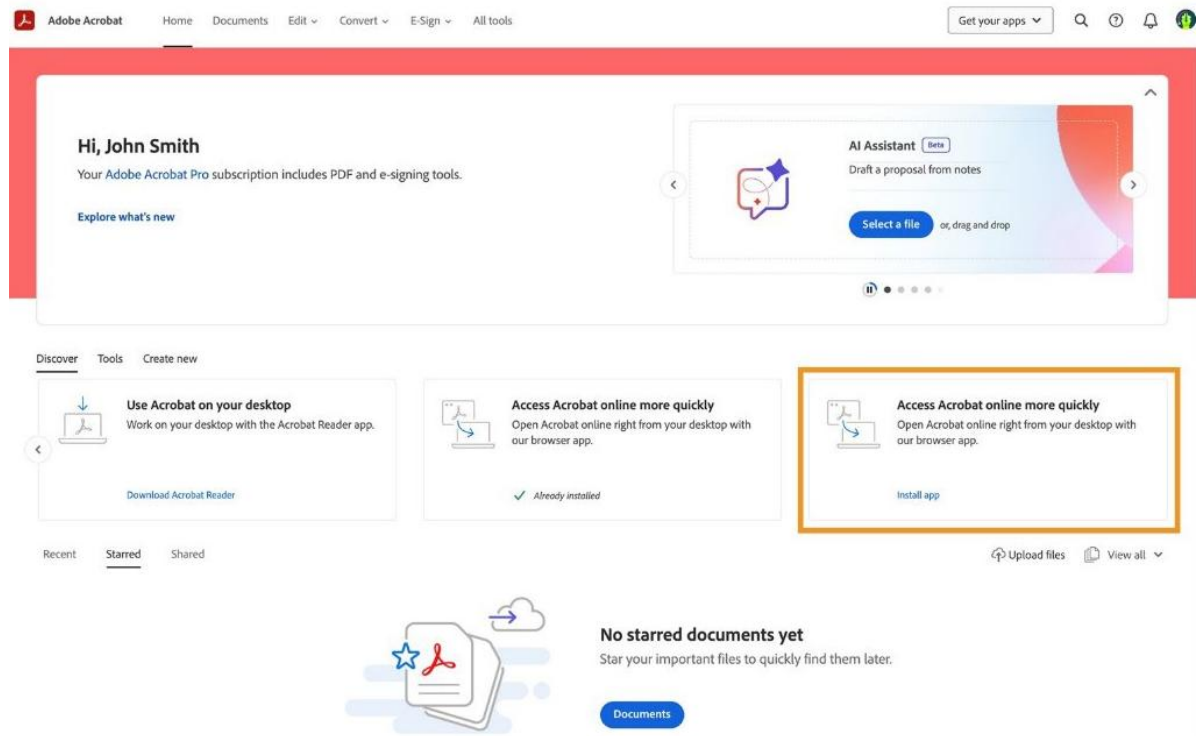


Рисунок 1.2 – Головна сторінка Adobe Асробат [4]

1.2.3 ABBYY FineReader

ABBYY FineReader — це потужне програмне забезпечення для розпізнавання тексту (OCR), розроблене компанією ABBYY. Воно дозволяє перетворювати відскановані документи, зображення та PDF-файли у редаговані формати (наприклад, DOCX, XLSX або TXT) із високою точністю розпізнавання. FineReader підтримує багатомовне розпізнавання, збереження структури документа, а також інструменти для порівняння текстів та редагування результатів OCR. Незважаючи на високий рівень функціональності та точності, програмний продукт є комерційним і має досить високу вартість ліцензії, що може бути обмеженням для окремих користувачів або організацій. На рисунку 1.3 зображено головну сторінку цього програмного засобу.

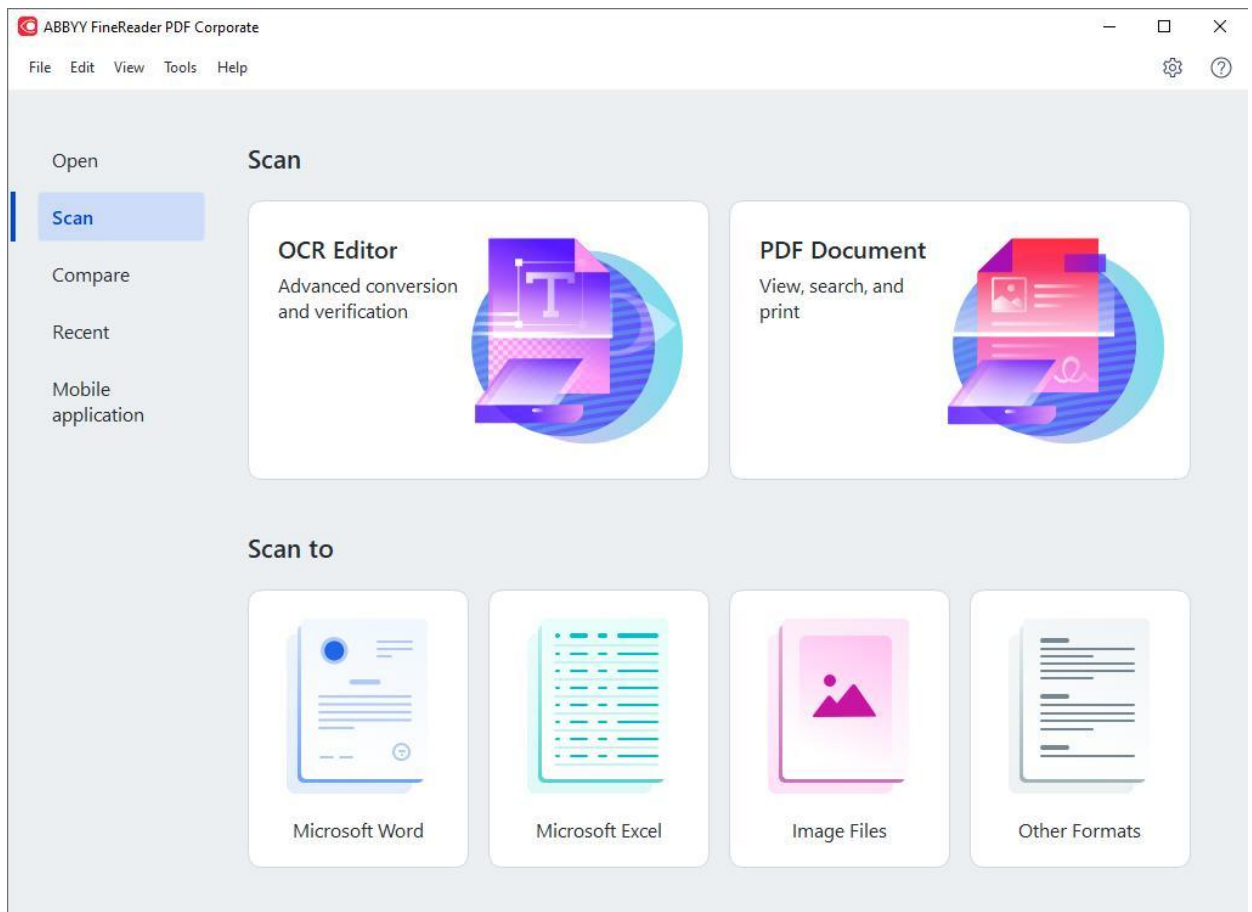


Рисунок 1.3 – Головна сторінка ABBYY FineReader [5]

1.2.4 Google Docs

Google Docs є хмарним текстовим редактором, розробленим компанією Google, який працює безпосередньо у веб-браузері без необхідності встановлення додаткового програмного забезпечення. Сервіс забезпечує створення, редагування та збереження документів онлайн, а також підтримує спільну роботу декількох користувачів у режимі реального часу. Документи автоматично зберігаються у хмарному сховищі Google Drive, що забезпечує доступ до них з будь-якого пристрою. Крім того, Google Docs підтримує базові інструменти форматування, коментування та інтеграцію з іншими сервісами Google. Водночас його функціональність у сфері автоматизованого аналізу тексту є обмеженою порівняно зі спеціалізованими рішеннями. На рисунку 1.4 зображено головну сторінку цього текстового редактора.

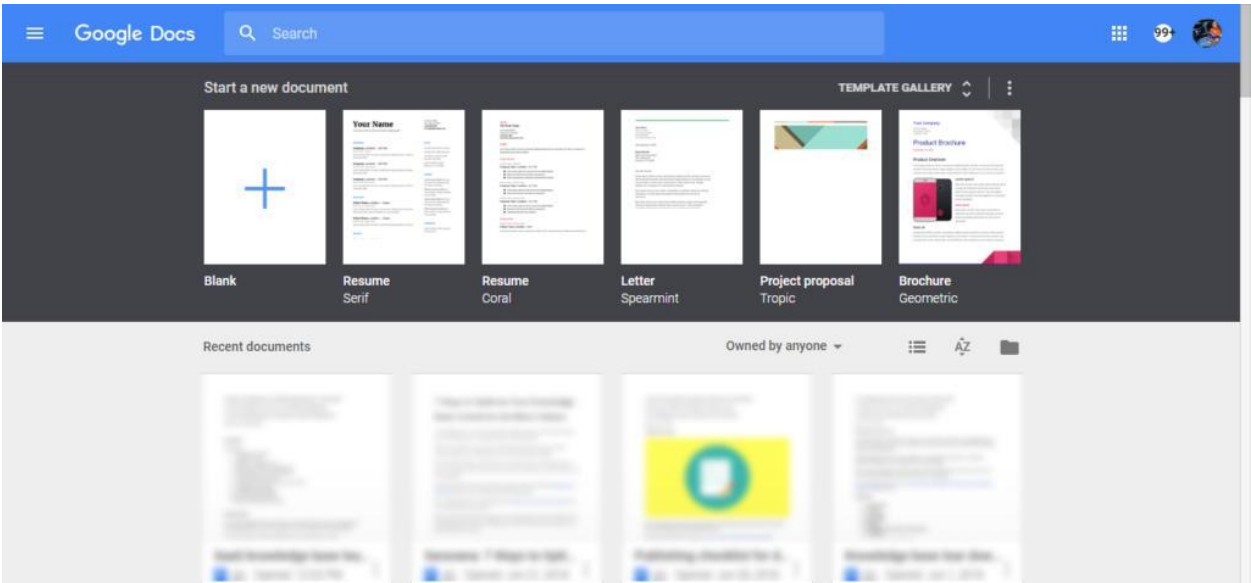


Рисунок 1.4 – Головна сторінка ABBYY FineReader [6]

1.2.5 Порівняльна таблиця існуючих та розроблюваного програмних продуктів

Розглянемо сильні та слабкі сторони перелічених текстових редакторів і програм із розроблюваним програмним забезпеченням за ключовими критеріями: робота з TXT, робота з DOCX, робота з PDF, автоматичний аналіз тексту, пошук ключових слів, автоматизація обробки, безкоштовність.

В таблиці 1.1 наведено порівняння існуючих програмних рішень та функціональних можливостей розроблюваного програмного забезпечення.

Таблиця 1.1 — Порівняння існуючих програмних рішень та функціональних можливостей розроблюваного програмного забезпечення

Функція / ПЗ	MS Word	Adobe Acrobat	ABBYY FineReader	Google Docs	Розроблюване ПЗ
1	2	3	4	5	6
Робота з TXT	Підтримується	Підтримується	Підтримується	Підтримується	Підтримується

Продовження таблиці 1.1

1	2	3	4	5	6
Робота з DOCX	Підтриму- ється	Не підтриму- ється	Підтриму- ється	Підтриму- ється	Підтриму- ється
Робота з PDF	Не підтриму- ється	Підтриму- ється	Підтриму- ється	Не підтриму- ється	Підтриму- ється
Автоматичний аналіз тексту	Не підтриму- ється	Не підтриму- ється	Підтриму- ється (частково)	Не підтриму- ється	Підтриму- ється
Пошук ключових слів	Підтриму- ється	Підтриму- ється	Підтриму- ється	Підтриму- ється	Підтриму- ється
Автоматизація обробки	Не підтриму- ється	Не підтриму- ється	Підтриму- ється (частково)	Не підтриму- ється	Підтриму- ється
Безкоштовність	Не підтриму- ється	Не підтриму- ється	Не підтриму- ється	Підтриму- ється	Підтриму- ється

Як видно з таблиці 1.1, існуючі програмні рішення мають обмежені можливості щодо автоматизації обробки документів та аналізу текстових даних. На відміну від них, розроблюване програмне забезпечення орієнтоване саме на автоматизацію цих процесів, що забезпечує скорочення часу обробки документів та зменшення навантаження на користувача.

На відміну від існуючих рішень, розроблюване програмне забезпечення дозволяє автоматично обробляти декілька документів одночасно, формувати звіти та виконувати аналіз текстових даних без участі користувача.

1.2.6 Використання мов програмування для автоматизації обробки документів

Для створення програмних систем обробки документів широко застосовуються сучасні мови програмування, що забезпечують можливість роботи з файлами різних форматів та реалізацію алгоритмів аналізу текстових даних. Однією з найбільш популярних мов для таких задач є Python.

Python має значну кількість бібліотек, які спрощують роботу з електронними документами. Наприклад, бібліотеки `python-docx` дозволяють працювати з документами формату DOCX, `PyPDF2` або `pdfminer` — з PDF-файлами, а бібліотеки для обробки тексту забезпечують виконання операцій пошуку, аналізу та структурування інформації.

Використання мов програмування для автоматизації обробки документів дозволяє створювати спеціалізовані програмні інструменти, адаптовані до потреб користувача. Такі інструменти можуть виконувати масову обробку документів, аналіз текстових даних, пошук необхідної інформації та формування результатів у зручному вигляді. Завдяки цьому значно скорочується час виконання рутинних операцій і підвищується точність обробки інформації [7].

1.3 Методи та технології автоматизації обробки документів

Автоматизація обробки електронних документів є важливим напрямом розвитку сучасних інформаційних систем. Вона передбачає використання програмних методів і технологій, що дозволяють виконувати аналіз, пошук, перетворення та структурування інформації без безпосередньої участі користувача.

До основних методів автоматизації обробки документів належать методи аналізу текстових даних, пошуку інформації, обробки файлів різних форматів, а також алгоритми класифікації та структурування даних.

Застосування таких методів дозволяє виконувати автоматичний аналіз великих обсягів текстової інформації, що значно скорочує час виконання рутинних операцій.

Одним із важливих напрямів автоматизації є використання технологій обробки тексту (text processing). Вони включають операції читання документів, виділення текстових фрагментів, пошук ключових слів, аналіз структури тексту та перетворення отриманої інформації у зручний для подальшої роботи формат.

Сучасні технології автоматизації обробки документів також передбачають використання спеціалізованих програмних бібліотек, які забезпечують підтримку різних форматів файлів та реалізують необхідні алгоритми обробки даних. Це дозволяє створювати програмні системи, здатні працювати з великою кількістю документів різних типів та виконувати їх автоматизований аналіз [8].

1.3.1 Використання бібліотек Python для обробки документів

Однією з найбільш поширених мов програмування для реалізації систем обробки документів є Python. Це зумовлено простотою мови, великою кількістю доступних бібліотек та активною підтримкою спільноти розробників.

Для роботи з електронними документами в Python використовуються різні програмні бібліотеки. Наприклад, бібліотека `python-docx` дозволяє створювати, читати та редагувати документи формату DOCX. Бібліотеки для роботи з PDF-файлами забезпечують можливість читання тексту, отримання метаданих та виконання пошуку інформації у документах. Для роботи з текстовими даними також можуть використовуватись стандартні засоби мови Python, що дозволяють виконувати пошук, фільтрацію та обробку текстових рядків.

Застосування бібліотек Python дозволяє реалізувати автоматизовану обробку документів різних форматів, включаючи читання файлів, виділення текстової інформації, пошук ключових слів та формування результатів аналізу. Це робить Python зручним інструментом для створення програмних систем автоматизації обробки документів [8].

1.4 Висновки до розділу 1

У першому розділі дипломної роботи було проведено аналіз предметної області обробки електронних документів. Розглянуто основні особливості процесу роботи з електронними документами, а також визначено основні етапи їх обробки, до яких належать завантаження, аналіз, пошук інформації та перетворення даних.

Проаналізовано основні типи та формати електронних документів, що використовуються у сучасних інформаційних системах. Встановлено, що найбільш поширеними форматами є текстові файли, документи формату DOCX, PDF-файли та табличні дані. Кожен із цих форматів має власні особливості структури та обробки, що необхідно враховувати при розробці програмного забезпечення.

Також було проведено аналіз існуючих програмних рішень для роботи з електронними документами. Встановлено, що сучасні програмні системи дозволяють виконувати створення, редагування, зберігання та аналіз документів. Однак багато з них орієнтовані на роботу в межах складних систем документообігу або мають обмежені можливості щодо автоматизованої обробки великої кількості файлів.

Розглянуто основні методи та технології автоматизації обробки документів, зокрема методи обробки текстових даних та використання спеціалізованих програмних бібліотек. Визначено, що використання сучасних мов програмування, зокрема Python, дозволяє створювати ефективні

програмні інструменти для автоматизованої обробки документів різних форматів.

Таким чином, проведений аналіз предметної області дозволив визначити основні вимоги до програмного забезпечення автоматизації обробки документів та обґрунтувати доцільність розробки спеціалізованого програмного інструменту для вирішення поставленої задачі.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У сучасних інформаційних системах важливе значення має правильне проєктування програмного забезпечення. Якісне проєктування дозволяє визначити структуру системи, її основні компоненти, функціональні можливості та способи взаємодії між окремими елементами програми.

У цьому розділі розглядаються основні етапи проєктування програмного забезпечення для автоматизації обробки електронних документів. Зокрема визначаються вимоги до програмної системи, описується її архітектура, структура основних модулів та обґрунтовується вибір технологій реалізації.

2.1 Визначення вимог до програмного забезпечення

Визначення вимог є одним із ключових етапів розробки програмного забезпечення. Саме на цьому етапі формується перелік функцій, які повинна виконувати система, а також визначаються обмеження та характеристики, що впливають на її роботу. Правильно сформульовані вимоги дозволяють забезпечити відповідність програмного продукту поставленим задачам та потребам користувачів.

У рамках дипломної роботи програмне забезпечення призначене для автоматизації обробки електронних документів різних форматів. Основною метою системи є скорочення часу обробки документів та зменшення кількості помилок під час виконання рутинних операцій.

Вимоги до програмного забезпечення можна поділити на функціональні та нефункціональні [9].

2.1.1 Функціональні вимоги

Функціональні вимоги визначають основні можливості програмної системи та перелік операцій, які вона повинна виконувати. До основних функціональних вимог розроблюваного програмного забезпечення належать:

- завантаження електронних документів різних форматів (TXT, DOCX, PDF, CSV);
- читання та обробка текстового вмісту документів;
- виконання пошуку необхідної інформації у тексті документів;
- аналіз вмісту документів та виділення ключових даних;
- відображення результатів обробки документів користувачу;
- можливість обробки декількох документів у межах одного сеансу роботи [9].

2.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають характеристики програмного забезпечення, які не пов'язані безпосередньо з виконанням конкретних функцій, але впливають на якість роботи системи.

До основних нефункціональних вимог належать:

- зручність використання програмного інтерфейсу;
- достатня швидкість обробки документів;
- стабільність роботи програмного забезпечення;
- можливість роботи на персональних комп'ютерах із сучасними операційними системами;
- мінімальні вимоги до апаратних ресурсів.

Таким чином, визначення функціональних і нефункціональних вимог дозволяє сформулювати чітке уявлення про можливості програмного забезпечення та створює основу для подальшого проєктування архітектури системи і структури програмних модулів [9].

2.2 Архітектура програмної системи

Архітектура програмного забезпечення визначає загальну структуру системи, її основні компоненти, а також принципи взаємодії між ними. Правильно спроектована архітектура дозволяє забезпечити зрозумілість структури програми, спростити її подальший розвиток та підвищити надійність роботи системи.

Для розроблюваного програмного забезпечення автоматизації обробки документів доцільно використати модульний підхід. Такий підхід передбачає поділ програмної системи на окремі логічні компоненти (модулі), кожен з яких виконує певну функцію. Це дозволяє спростити процес розробки, тестування та подальшого вдосконалення програмного забезпечення.

Архітектура розроблюваної системи включає кілька основних компонентів: модуль користувацького інтерфейсу, модуль завантаження документів, модуль обробки документів, модуль аналізу тексту та модуль формування результатів, що зображений на рис. 2.1. Кожен із цих компонентів виконує окрему функцію у процесі обробки документів.

Модуль користувацького інтерфейсу забезпечує взаємодію користувача з програмною системою. Через інтерфейс користувач може завантажувати документи, запускати процес обробки та переглядати результати аналізу.

Модуль завантаження документів відповідає за відкриття файлів різних форматів та передачу їх вмісту до модуля обробки. На цьому етапі виконується перевірка формату файлу та підготовка даних для подальшого аналізу.

Модуль обробки документів здійснює основні операції з текстовими даними, зокрема виділення тексту з документів, попередню обробку інформації та передачу отриманих даних до модуля аналізу.

Модуль аналізу тексту виконує пошук необхідної інформації, обробку текстових даних та виділення ключових елементів документа. Результати аналізу передаються до модуля формування результатів.

Модуль формування результатів відповідає за відображення результатів обробки документів у зручному для користувача вигляді.

Використання модульної архітектури дозволяє забезпечити зрозумілу структуру програмної системи та спростити процес її подальшого розвитку [9].



Рисунок 2.1 – Архітектура програмного забезпечення автоматизації обробки документів

2.3 Проектування структури програмних модулів

Після визначення архітектури програмної системи наступним етапом є проектування структури програмних модулів. Модульний підхід дозволяє розділити програму на окремі компоненти, кожен з яких відповідає за виконання певної функції. Такий підхід забезпечує кращу організацію програмного коду, спрощує процес тестування та дозволяє легко вносити зміни до системи в майбутньому.

У розроблюваному програмному забезпеченні для автоматизації обробки документів структура програми формується у вигляді набору взаємопов'язаних модулів. Кожен модуль реалізує окрему частину функціональності системи та взаємодіє з іншими компонентами через визначені інтерфейси.

Основними модулями програмної системи є:

- модуль інтерфейсу користувача, який забезпечує взаємодію користувача з програмою;
- модуль завантаження документів, що відповідає за відкриття файлів різних форматів;
- модуль обробки документів, який виконує попередню обробку текстових даних;
- модуль аналізу тексту, що здійснює пошук інформації та аналіз змісту документів;
- модуль формування результатів, який відповідає за відображення результатів обробки документів.

Кожен модуль реалізується у вигляді окремого програмного файлу або набору функцій. Такий підхід дозволяє ізолювати різні частини логіки програми та спростити процес розробки.

У мові програмування Python модульна структура реалізується шляхом поділу програми на окремі файли, що містять класи та функції. Кожен файл

відповідає за виконання певної частини функціональності системи. Взаємодія між модулями здійснюється через імпорт необхідних функцій або класів.

Застосування модульної структури дозволяє підвищити зрозумілість програмного коду, полегшити його підтримку та забезпечити можливість подальшого розширення програмного забезпечення [10].

2.3.1 Структура програмного проєкту

Для реалізації програмного забезпечення автоматизації обробки документів у середовищі Python доцільно використати структурований підхід до організації файлів проєкту.

Структура програмного проєкту може мати такий вигляд:

- a) document_processor:
 - 1) main.py;
 - 2) file_loader.py;
 - 3) text_processor.py;
 - 4) text_analyzer.py;
 - 5) result_viewer.py.

Файл main.py є головною точкою запуску програми та забезпечує координацію роботи інших модулів системи.

Модуль file_loader.py відповідає за відкриття та читання документів різних форматів. У цьому модулі реалізується перевірка типу файлу та підготовка даних для подальшої обробки.

Модуль text_processor.py виконує попередню обробку тексту, зокрема очищення текстових даних, видалення зайвих символів та підготовку інформації для аналізу.

Модуль text_analyzer.py реалізує алгоритми аналізу тексту, включаючи пошук ключових слів, аналіз структури тексту та виділення необхідної інформації.

Модуль `result_viewer.py` відповідає за формування та відображення результатів обробки документів у зручному для користувача вигляді.

Додаткові допоміжні функції можуть бути винесені у модуль `helpers.py`, що розташований у каталозі `utils`.

Така структура програмного проєкту дозволяє чітко розділити логіку системи на окремі компоненти та забезпечує зручність подальшого розвитку програмного забезпечення.

2.4 Вибір технологій та засобів реалізації

Важливим етапом проєктування програмного забезпечення є вибір технологій та програмних засобів, які будуть використовуватися для реалізації системи. Правильний вибір інструментів розробки дозволяє забезпечити надійність роботи програмного забезпечення, спростити процес створення системи та забезпечити можливість її подальшого розвитку.

Для реалізації програмного забезпечення автоматизації обробки документів було обрано мову програмування Python. Дана мова програмування широко використовується для розробки програмних систем обробки текстової інформації завдяки простому синтаксису, великій кількості бібліотек та активній спільноті розробників.

Python підтримує різноманітні бібліотеки для роботи з текстовими даними, що дозволяє ефективно реалізувати функції аналізу документів, обробки тексту та автоматизації рутинних операцій. Крім того, Python підтримує модульну архітектуру програмного коду, що дозволяє легко організувати структуру програмної системи.

Розробка програмного забезпечення здійснюється у середовищі Visual Studio, яке забезпечує зручні інструменти для написання програмного коду, налагодження програм та керування проєктами Python. Дане середовище підтримує велику кількість розширень, що дозволяє значно спростити процес розробки.

Програмне забезпечення розробляється для операційної системи Windows 11, що забезпечує сумісність із сучасними комп'ютерними системами та підтримує всі необхідні інструменти для розробки програмного забезпечення.

Використання мови програмування Python та сучасних бібліотек для роботи з документами дозволяє створити програмну систему, здатну автоматизувати процес обробки текстових документів та забезпечити зручність роботи користувачів із документами [11].

2.4.1 Обґрунтування вибору мови програмування

Для реалізації програмного забезпечення було обрано мову програмування Python. Вибір зумовлений її простотою, гнучкістю та широкими можливостями для обробки текстових даних.

Python є високорівневою мовою програмування, яка забезпечує швидку розробку програмних продуктів завдяки простому синтаксису та великій кількості готових бібліотек. Це дозволяє значно скоротити час розробки програмного забезпечення у порівнянні з іншими мовами програмування.

У порівнянні з мовами програмування C# та Java, Python має менший обсяг службового коду, що спрощує реалізацію алгоритмів обробки тексту. Крім того, Python має вбудовані засоби роботи з рядками та регулярними виразами, що є важливим для задач аналізу документів.

Також важливою перевагою є наявність великої кількості бібліотек для роботи з документами (PyPDF2, python-docx, pandas), що дозволяє реалізувати підтримку різних форматів файлів без значних витрат часу.

В таблиці 2.1 наведено порівняння мови Python з іншими популярними мовами програмування.

Таблиця 2.1 — Порівняння мови Python з іншими мовами програмування

Критерій	Python	C#	Java
Простота синтаксису	Висока	Середня	Середня
Швидкість розробки	Висока	Середня	Середня
Робота з текстом	Висока	Висока	Висока
Кількість бібліотек	Висока	Висока	Висока
Кросплатформеність	Висока	Висока	Висока
Зручність для прототипування	Висока	Низька	Низька

Як видно з таблиці 2.1, мова Python має переваги у швидкості розробки, простоті синтаксису та зручності роботи з текстовими даними, що робить її доцільним вибором для реалізації програмного забезпечення автоматизації обробки документів.

2.4.2 Обґрунтування вибору середовища розробки

Для розробки програмного забезпечення було обрано інтегроване середовище розробки Visual Studio. Дане середовище є потужним інструментом для створення програмних продуктів та забезпечує широкі можливості для написання, налагодження та тестування програмного коду.

Visual Studio підтримує різні мови програмування, включаючи Python, та надає зручні інструменти для роботи з проєктами, такі як інтелектуальне автодоповнення коду (IntelliSense), вбудований відладчик та засоби керування залежностями.

Однією з важливих переваг Visual Studio є наявність потужних інструментів налагодження, що дозволяють виявляти помилки у програмному коді на ранніх етапах розробки. Крім того, середовище забезпечує зручний інтерфейс для роботи з графічними елементами та проєктною структурою.

У порівнянні з такими середовищами, як PyCharm та Visual Studio Code, Visual Studio має більш розширений функціонал для розробки складних програмних систем, зокрема кращі можливості налагодження та інтеграції з іншими інструментами розробки.

В таблиці 2.2 наведено порівняння середовища Visual Studio з іншими популярними середовищами розробки.

Таблиця 2.2 — Порівняння середовища Visual Studio з іншими середовищами розробки

Критерій	Visual Studio	PyCharm	VS Code
Потужність IDE	Висока	Висока	Середня
Інструменти налагодження	Всі	Всі	Частково
Зручність роботи з проєктами	Висока	Висока	Середня
Споживання ресурсів	Високе	Середнє	Низьке
Підтримка Python	Підтримується	Підтримується	Підтримується
Розширюваність	Висока	Висока	Висока

Як видно з таблиці 2.2, середовище Visual Studio має розширені можливості для розробки та налагодження програмного забезпечення, що робить його доцільним вибором для реалізації даного програмного продукту.

2.4.3 Обґрунтування відмови від використання бази даних

У процесі розробки програмного забезпечення було прийнято рішення не використовувати базу даних. Це зумовлено специфікою задачі, яка не передбачає довготривалого зберігання великих обсягів структурованих даних.

Розроблюване програмне забезпечення орієнтоване на обробку документів у режимі реального часу, де дані зчитуються з файлів,

обробляються та відразу відображаються користувачу. У такому випадку використання бази даних є надлишковим.

Крім того, відсутність бази даних дозволяє спростити архітектуру системи, зменшити витрати на розробку та підвищити швидкість роботи програмного забезпечення.

Використання бази даних є доцільним у випадках, коли необхідно зберігати великі обсяги інформації або забезпечити багатокористувацький доступ. У даній роботі такі вимоги відсутні.

2.4.4 Вимоги до апаратного та програмного забезпечення

Для забезпечення коректної роботи розробленого програмного забезпечення необхідно визначити вимоги до апаратного та програмного забезпечення.

До апаратного забезпечення висуваються такі вимоги:

- процесор із тактовою частотою не менше 2.0 GHz;
- оперативна пам'ять обсягом не менше 8 ГБ;
- наявність вільного місця на диску не менше 500 МБ;
- стандартні пристрої введення та виведення інформації (клавіатура, миша, монітор).

До програмного забезпечення висуваються такі вимоги:

- операційна система Windows 10 або новіша;
- встановлений інтерпретатор Python версії 3.8 або вище;
- наявність необхідних бібліотек для роботи з документами та текстом (PyPDF2, python-docx, pandas, re);
- середовище розробки Visual Studio (для розробки та налагодження програми).

Зазначені вимоги забезпечують стабільну роботу програмного забезпечення та можливість обробки документів різних форматів без значних затримок.

2.5 Висновки до розділу 2

У другому розділі було розглянуто основні питання проєктування програмного забезпечення для автоматизації обробки документів.

Зокрема, було визначено основні функціональні вимоги до програмної системи, що дозволило сформулювати загальне уявлення про можливості майбутнього програмного забезпечення та основні сценарії його використання. Було встановлено, що програмна система повинна забезпечувати завантаження документів, обробку текстової інформації, аналіз вмісту документів та відображення результатів обробки для користувача.

У ході проєктування було сформовано архітектуру програмної системи, яка базується на модульному підході. Використання модульної архітектури дозволяє розділити програму на окремі функціональні компоненти, що спрощує процес розробки, тестування та подальшого вдосконалення програмного забезпечення.

Також було розроблено структуру програмних модулів системи, що включає модулі інтерфейсу користувача, завантаження документів, обробки тексту, аналізу текстових даних та формування результатів. Така структура забезпечує логічний поділ функціональності системи та сприяє підвищенню зрозумілості програмного коду.

Окрім цього, було обґрунтовано вибір технологій та засобів реалізації програмного забезпечення. Для розробки системи було обрано мову програмування Python, яка забезпечує широкі можливості для обробки текстових даних та автоматизації роботи з документами. Також було визначено використання спеціалізованих бібліотек для роботи з документами різних форматів.

Отже, результати другого розділу створюють необхідну основу для подальшої реалізації програмного забезпечення автоматизації обробки документів, що буде розглянуто у наступному розділі роботи.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У даному розділі розглядається процес реалізації програмного забезпечення для автоматизації обробки електронних документів. Описується розробка основних модулів системи, їх функціональні можливості та взаємодія між собою.

Програмне забезпечення реалізовано з використанням мови програмування Python. У процесі розробки застосовано модульний підхід, що дозволило розділити систему на окремі функціональні компоненти. Кожен модуль відповідає за виконання певного набору задач, що спрощує розробку, тестування та подальше вдосконалення програмного продукту.

3.1 Структурна схема програмного забезпечення

Програмне забезпечення для автоматизації обробки документів має модульну структуру, що забезпечує гнучкість, розширюваність та зручність супроводу системи.

Основними компонентами системи є модуль графічного інтерфейсу користувача, модуль керування, а також функціональні модулі обробки даних.

Графічний інтерфейс користувача забезпечує взаємодію користувача з програмою, зокрема вибір папки з документами, введення ключових слів та відображення результатів аналізу.

Модуль керування координує роботу всіх компонентів системи, здійснює обробку введених даних та керує послідовністю виконання операцій.

До функціональних модулів належать:

- модуль завантаження файлів, який забезпечує зчитування даних із документів різних форматів;
- модуль обробки тексту, що виконує нормалізацію та очищення текстових даних;

- модуль аналізу тексту, який реалізує пошук ключових слів та підрахунок їх входжень;

- модуль генерації звіту, який формує підсумковий звіт за результатами аналізу.

Взаємодія між модулями здійснюється через передачу текстових даних та результатів обробки.

На рисунку 3.1 наведено структурну схему програмного забезпечення.

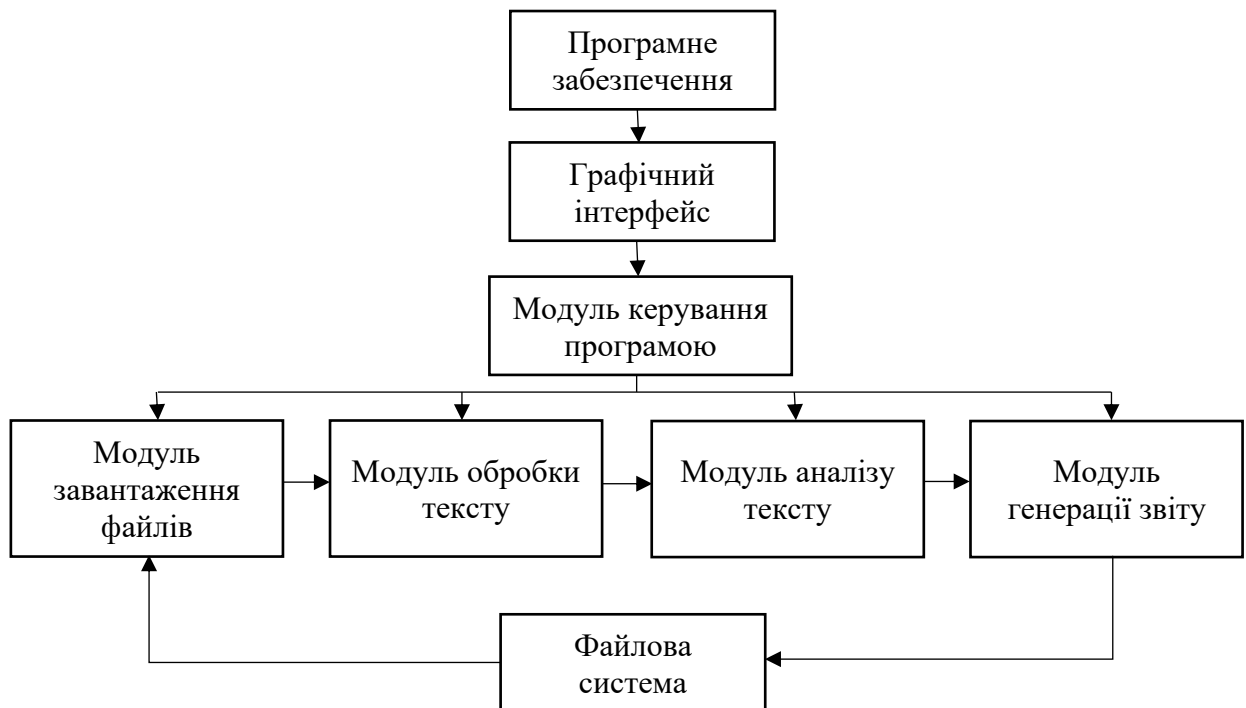


Рисунок 3.1 – Структурна схема системи

Як видно з рисунка 3.1, програмне забезпечення має чітко визначену модульну структуру, що дозволяє спростити процес розробки, тестування та подальшої модифікації системи.

3.2 Функціональна схема роботи програмного забезпечення

Функціональна схема відображає послідовність виконання операцій під час роботи програмного забезпечення та дозволяє детально описати логіку обробки даних.

Робота програми починається з вибору користувачем папки з документами та введення ключових слів для аналізу. Після цього здійснюється перевірка коректності введених даних. Далі програма виконує перебір файлів у вибраній папці та визначає їх формат. Для підтримуваних форматів здійснюється зчитування тексту з документів. Отриманий текст проходить етап попередньої обробки, що включає приведення до нижнього регістру, видалення зайвих символів та нормалізацію пробілів. Після цього виконується аналіз тексту, зокрема пошук ключових слів та підрахунок кількості їх входжень.

Результати аналізу зберігаються та використовуються для формування підсумкового звіту. Також здійснюється сортування документів за релевантністю відповідно до кількості знайдених ключових слів. На завершальному етапі результати відображаються у графічному інтерфейсі користувача.

На рисунку 3.2 наведено функціональну схему роботи програмного забезпечення.

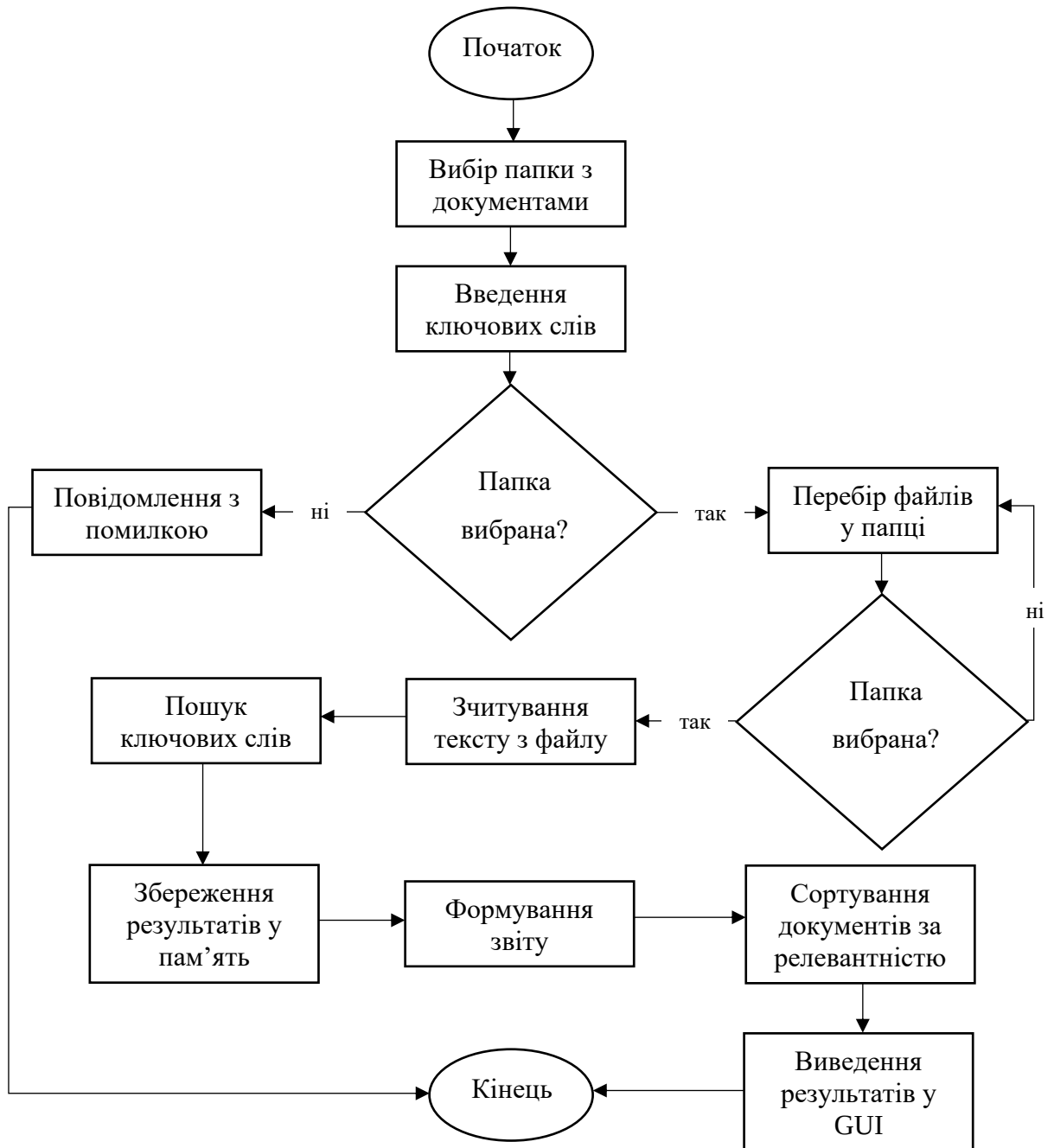


Рисунок 3.2 – Функціональна схема системи

3.3 Реалізація модуля завантаження документів

Модуль завантаження документів є одним із ключових компонентів програмної системи, оскільки він забезпечує зчитування даних із файлів різних форматів та підготовку їх для подальшої обробки.

Основною функцією модуля є відкриття файлів різних форматів, таких як текстові файли (.txt), документи формату .docx, PDF-файли та табличні дані

(.csv). Для реалізації цієї функціональності використовується окремий програмний модуль, який визначає тип файлу за його розширенням та застосовує відповідний метод обробки.

У процесі реалізації модуля використано стандартні засоби мови Python, а також сторонні бібліотеки для роботи з документами. Зокрема, для обробки DOCX-файлів використовується бібліотека `python-docx`, для роботи з PDF-файлами — `PyPDF2`, а для обробки табличних даних — бібліотека `pandas`.

Нижче наведено фрагмент коду, що реалізує функцію завантаження документів:

```
import os
import docx
from PyPDF2 import PdfReader
import pandas as pd

def load_file(path):
    ext = os.path.splitext(path)[1].lower()

    if ext == ".txt":
        with open(path, "r", encoding="utf-8") as f:
            return f.read()

    elif ext == ".docx":
        doc = docx.Document(path)
        return "\n".join([p.text for p in doc.paragraphs])

    elif ext == ".pdf":
        reader = PdfReader(path)
        text = ""
        for page in reader.pages:
```

```

    text += page.extract_text() or ""
    return text

elif ext == ".csv":
    df = pd.read_csv(path)
    return df.to_string()

else:
    raise ValueError("Unsupported file format")

```

У наведеному коді здійснюється визначення типу файлу за його розширенням. Для кожного формату застосовується відповідний спосіб зчитування даних. У випадку текстових файлів використовується стандартне читання файлу, для DOCX-файлів — отримання тексту з абзаців документа, для PDF — зчитування тексту зі сторінок документа, а для CSV — перетворення табличних даних у текстовий вигляд.

Розроблений модуль забезпечує універсальний підхід до роботи з документами різних форматів та дозволяє інтегрувати його з іншими компонентами системи.

3.4 Реалізація модуля обробки тексту

Модуль обробки тексту призначений для попередньої підготовки текстових даних перед їх подальшим аналізом. Основною метою цього модуля є приведення тексту до уніфікованого вигляду, що дозволяє підвищити точність пошуку та аналізу інформації.

У процесі обробки тексту виконуються такі основні операції:

- приведення тексту до нижнього регістру;
- видалення зайвих пробілів;
- очищення тексту від спеціальних символів;

– підготовка тексту до подальшого аналізу.

Зазначені операції дозволяють усунути неоднорідність текстових даних, що виникає внаслідок використання різних форматів документів, а також підвищити ефективність алгоритмів пошуку інформації.

Реалізація модуля обробки тексту виконана у вигляді окремої функції, яка приймає текстові дані та повертає їх оброблений варіант. Для обробки тексту використовуються регулярні вирази, що дозволяють ефективно виконувати пошук і заміну символів.

Нижче наведено фрагмент програмного коду, що реалізує функцію обробки тексту:

```
import re

def clean_text(text):
    text = text.lower()
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'^[a-zA-Za-яA-Я0-9 ]', "", text)
    return text
```

У наведеному коді використовується модуль `re`, який забезпечує роботу з регулярними виразами. На першому етапі текст переводиться у нижній регістр, що дозволяє виконувати пошук без урахування регістру символів. Далі здійснюється видалення зайвих пробілів та спеціальних символів, що не несуть змістового навантаження.

Результатом роботи функції є очищений текст, який використовується як вхідні дані для модуля аналізу тексту.

Таким чином, реалізований модуль забезпечує ефективну підготовку текстових даних, що є необхідною умовою для коректної роботи всієї програмної системи.

3.5 Реалізація модуля аналізу текстових даних

Модуль аналізу текстових даних призначений для обробки підготовленого тексту з метою виявлення необхідної інформації. Основною функцією даного модуля є пошук ключових слів у тексті документа та визначення кількості їх входжень.

Аналіз текстових даних є важливим етапом роботи програмної системи, оскільки саме на цьому етапі здійснюється отримання корисної інформації з документів. Результати аналізу можуть бути використані для швидкого знаходження необхідних даних, оцінки змісту документів або прийняття рішень на їх основі.

У розроблюваному програмному забезпеченні реалізовано простий та ефективний метод аналізу тексту, який базується на підрахунку кількості входжень заданих користувачем ключових слів. Такий підхід дозволяє швидко отримувати результати аналізу без використання складних алгоритмів обробки природної мови.

Модуль аналізу тексту реалізовано у вигляді функції, яка приймає очищений текст та список ключових слів, після чого виконує пошук кожного слова у тексті та підраховує кількість його входжень.

Нижче наведено фрагмент програмного коду, що реалізує даний модуль:

```
def find_keywords(text, keywords):  
    results = {}  
    total = 0  
  
    for word in keywords:  
        word = word.strip().lower()  
        count = text.count(word)  
        results[word] = count  
        total += count
```

```
return {  
    "keywords": results,  
    "total_matches": total  
}
```

У наведеному коді здійснюється послідовний перебір списку ключових слів, введених користувачем. Для кожного слова виконується приведення до нижнього регістру, що забезпечує коректність пошуку незалежно від регістру символів у тексті. Далі визначається кількість входжень слова у тексті за допомогою стандартного методу count.

Результати аналізу зберігаються у вигляді словника, де ключем є шукане слово, а значенням — кількість його входжень у тексті. Такий формат представлення результатів є зручним для подальшого відображення користувачу.

Розроблений модуль аналізу тексту забезпечує швидкий та ефективний пошук ключових слів у документах різних форматів. Його використання дозволяє значно скоротити час обробки інформації та спростити роботу користувача з великими обсягами текстових даних.

3.6 Реалізація модуля генерації звіту

Модуль генерації звіту є важливою складовою програмного забезпечення та призначений для формування підсумкових результатів аналізу документів у зручному для користувача вигляді.

Основною функцією модуля є збереження результатів обробки документів у текстовий файл. Звіт містить інформацію про кожен оброблений документ, загальну кількість знайдених входжень ключових слів, а також деталізовану статистику по кожному ключовому слову.

Вхідними даними для модуля є структура даних, сформована в процесі аналізу документів, яка містить назви файлів та результати підрахунку ключових слів.

Вихідними даними є текстовий файл звіту, що створюється у робочій директорії програми.

Основні функції модуля:

- формування структури звіту;
- запис результатів аналізу у файл;
- відображення статистики для кожного документа;
- забезпечення зручного формату представлення даних.

Фрагмент програмного коду модуля генерації звіту наведено нижче:

```
def save_report(report_data, output_file="analysis_report.txt"):
    with open(output_file, "w", encoding="utf-8") as f:
        f.write("DOCUMENT ANALYSIS REPORT\n")
        f.write("=" * 40 + "\n\n")

        for doc_name, data in report_data.items():
            f.write(f"File: {doc_name}\n")
            f.write(f"Total matches: {data['total_matches']}\n")

            for word, count in data["keywords"].items():
                f.write(f" {word}: {count}\n")

            f.write("\n")
```

Використання модуля генерації звіту дозволяє автоматизувати процес документування результатів аналізу та забезпечує можливість подальшого використання отриманих даних.

3.7 Реалізація графічного інтерфейсу користувача

Для забезпечення зручності використання програмного забезпечення було реалізовано графічний інтерфейс користувача (GUI). Наявність інтерфейсу дозволяє користувачу взаємодіяти з програмою без необхідності використання командного рядка, що значно спрощує процес роботи з системою.

Графічний інтерфейс реалізовано з використанням стандартної бібліотеки Python — Tkinter, яка надає необхідні засоби для створення вікон, кнопок, текстових полів та інших елементів керування.

Інтерфейс програмного забезпечення включає такі основні елементи:

- поле для вибору файлу;
- кнопка відкриття файлу;
- поле для введення ключових слів;
- кнопка запуску аналізу;
- область відображення результатів.

Основне вікно програми створюється із використанням об'єкта Tk, після чого до нього додаються всі необхідні елементи керування. Для організації інтерфейсу використовується покрокове розміщення елементів із застосуванням методу pack.

Нижче наведено фрагмент програмного коду, що реалізує графічний інтерфейс користувача:

```
import tkinter as tk
from tkinter import filedialog, messagebox
import os

from file_loader import load_file
from text_processor import clean_text
from text_analyzer import find_keywords
from report_generator import save_report
```

```

class DocumentAnalyzerApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Smart Document Analyzer")
        self.root.geometry("800x600")

        self.folder_path = ""

        tk.Label(root, text="Select folder with documents").pack()

        tk.Button(root, text="Browse Folder", command=self.browse_folder).pack()

        self.folder_label = tk.Label(root, text="")
        self.folder_label.pack()

        tk.Label(root, text="Keywords (comma separated)").pack()

        self.keyword_entry = tk.Entry(root, width=70)
        self.keyword_entry.pack()

        tk.Button(root, text="Analyze Documents",
command=self.analyze).pack(pady=10)

        self.result_text = tk.Text(root, height=25, width=100)
        self.result_text.pack()

    def browse_folder(self):
        folder = filedialog.askdirectory()

        if folder:

```

```
self.folder_path = folder
```

```

sorted_docs = sorted(
    report_data.items(),
    key=lambda x: x[1]["total_matches"],
    reverse=True
)

self.result_text.delete(1.0, tk.END)
self.result_text.insert(tk.END, "DOCUMENT RANKING\n\n")

for doc, data in sorted_docs:
    self.result_text.insert(tk.END, f"\nFile: {doc}\n")
    self.result_text.insert(tk.END, f"Total matches: {data['total_matches']}\n")

    for word, count in data["keywords"].items():
        self.result_text.insert(tk.END, f"    {word}: {count}\n")

messagebox.showinfo("Done", "Analysis completed.\nReport saved.")

if __name__ == "__main__":
    root = tk.Tk()
    app = DocumentAnalyzerApp(root)
    root.mainloop()

```

У наведеному коді реалізовано клас, що відповідає за створення та обробку графічного інтерфейсу. Користувач має можливість вибрати файл за допомогою стандартного діалогового вікна, ввести ключові слова та запустити процес аналізу.

Результати обробки відображаються у текстовому полі, що дозволяє швидко оцінити результати аналізу документів. У випадку виникнення помилок користувач отримує відповідне повідомлення.

Реалізація графічного інтерфейсу значно підвищує зручність використання програмного забезпечення та дозволяє ефективно взаємодіяти з системою навіть користувачам без спеціальних технічних знань.

3.8 Висновки до розділу 3

У третьому розділі дипломної роботи було розглянуто процес розробки програмного забезпечення для автоматизації обробки документів.

У ході виконання розділу було реалізовано основні модулі програмної системи. Зокрема, розроблено модуль завантаження документів, який забезпечує роботу з файлами різних форматів, включаючи TXT, DOCX, PDF та CSV. Це дозволяє системі працювати з широким спектром електронних документів.

Також було реалізовано модуль обробки тексту, який виконує попередню підготовку текстових даних шляхом очищення та нормалізації. Це забезпечує підвищення точності подальшого аналізу інформації.

Модуль аналізу текстових даних дозволяє здійснювати пошук ключових слів та визначати кількість їх входжень у тексті документів. Це забезпечує можливість швидкого отримання необхідної інформації з великих обсягів текстових даних.

Окрім цього, було розроблено графічний інтерфейс користувача з використанням бібліотеки Tkinter, що забезпечує зручну взаємодію користувача з програмною системою. Інтерфейс дозволяє виконувати вибір файлів, введення ключових слів та перегляд результатів аналізу.

Таким чином, у третьому розділі було реалізовано повнофункціональне програмне забезпечення для автоматизації обробки документів, яке забезпечує

скорочення часу обробки інформації та спрощення роботи користувача з електронними документами.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Умови та порядок експлуатації програмного забезпечення

Експлуатація програмного забезпечення для автоматизації обробки документів передбачає його використання на персональному комп'ютері з відповідними технічними та програмними характеристиками.

Для коректної роботи програми необхідно забезпечити виконання мінімальних системних вимог, зокрема наявність сучасної операційної системи, достатнього обсягу оперативної пам'яті та встановленого інтерпретатора мови програмування Python.

Перед початком роботи користувач повинен запустити програмний застосунок, після чого відкривається головне вікно з графічним інтерфейсом.

Порядок роботи з програмою включає такі етапи:

- вибір папки, що містить документи для аналізу;
- введення ключових слів для пошуку;
- запуск процесу аналізу документів;
- перегляд результатів у графічному інтерфейсі;
- ознайомлення зі сформованим звітом.

Програма автоматично обробляє всі підтримувані файли у вибраній директорії та виконує аналіз текстової інформації без додаткового втручання користувача.

У процесі експлуатації користувачеві не потрібно виконувати складних налаштувань або додаткових конфігурацій, що забезпечує простоту використання програмного забезпечення.

Результати роботи програми відображаються у вікні застосунку, а також зберігаються у вигляді текстового файлу звіту, що дозволяє використовувати їх у подальшій роботі.

4.2 Тестування програмного забезпечення

Тестування програмного забезпечення є важливим етапом розробки, який дозволяє перевірити коректність роботи системи, виявити помилки та оцінити її функціональні можливості.

У процесі тестування було перевірено основні функції програми, зокрема:

- вибір папки з документами;
- зчитування файлів різних форматів (TXT, DOCX, PDF, CSV);
- обробка текстових даних;
- пошук ключових слів;
- формування звіту;
- відображення результатів у графічному інтерфейсі.

Для тестування було використано набір документів різних форматів, що містять текстову інформацію з ключовими словами.

Результати тестування наведено у таблиці 4.1.

Таблиця 4.1 – Результати тестування програмного забезпечення

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат
1	2	3	4	5
1	Відкриття TXT-файлу	Текстовий файл	Текст зчитано	Виконано
2	Відкриття DOCX-файлу	Документ Word	Текст зчитано	Виконано
3	Відкриття PDF-файлу	PDF-документ	Текст зчитано	Виконано

Продовження таблиці 4.1

1	2	3	4	5
4	Відкриття CSV-файлу	Табличні дані	Дані відображено	Виконано
5	Пошук ключових слів	«test, data»	Знайдено входження	Виконано
6	Робота GUI	Взаємодія користувача	Дані відображаються	Виконано

У процесі тестування було встановлено, що програмне забезпечення коректно виконує всі основні функції. Завантаження документів різних форматів відбувається без помилок, обробка тексту здійснюється відповідно до заданих алгоритмів, а результати аналізу відображаються у графічному інтерфейсі користувача.

Також було перевірено обробку помилкових ситуацій, зокрема спробу відкриття файлів неподтримуваного формату або відсутність вибраного файлу. У таких випадках система коректно повідомляє користувача про помилку. На рисунку 4.1 наведено головне вікно програмного забезпечення.

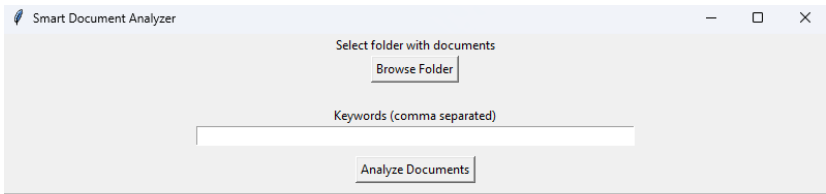


Рисунок 4.1 — Головне вікно програмного забезпечення

Як видно з рисунка 4.1, користувач має можливість обрати папку з документами, ввести ключові слова та запустити процес аналізу.

На рисунку 4.2 наведено приклад результатів аналізу документів.

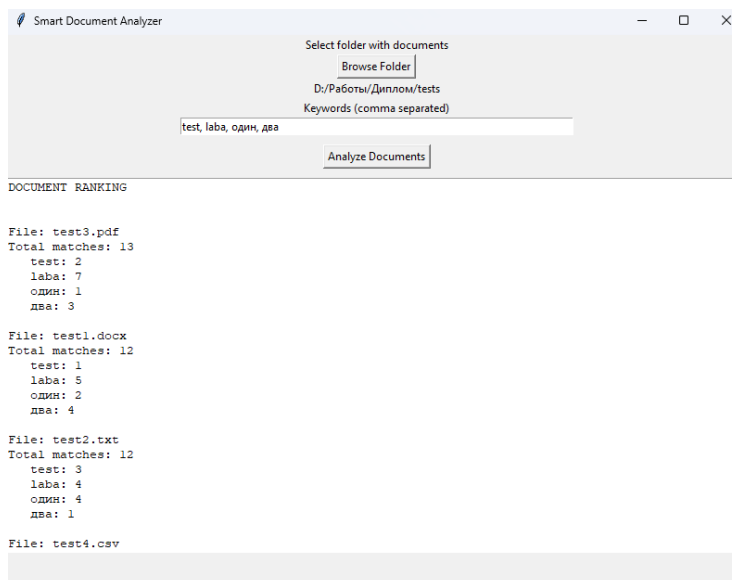


Рисунок 4.2 — Результати аналізу документів

З отриманих результатів видно, що програма визначає кількість входжень кожного ключового слова у кожному документі, а також обчислює загальну кількість входжень.

На рисунку 4.3 наведено приклад сформованого звіту.

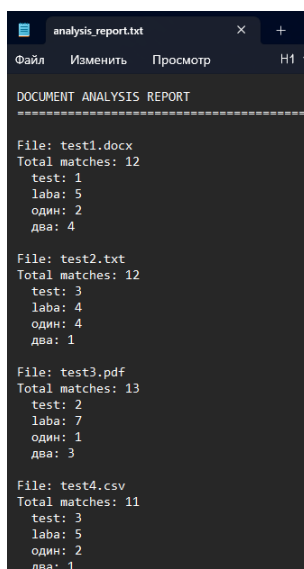


Рисунок 4.3 — Сформований звіт

Сформований звіт містить детальну інформацію про результати аналізу, що дозволяє використовувати його для подальшої обробки або збереження.

Отже, результати тестування підтверджують працездатність розробленого програмного забезпечення та його відповідність поставленим вимогам.

4.3 Експериментальне дослідження програмного забезпечення

З метою перевірки ефективності розробленого програмного забезпечення було проведено експериментальне дослідження, спрямоване на оцінку швидкості обробки документів та коректності виконання аналізу текстових даних.

У якості вхідних даних було використано набір електронних документів різних форматів (TXT, DOCX, PDF, CSV), що містять текстову інформацію з наперед визначеними ключовими словами.

Для проведення експерименту було сформовано тестову вибірку документів, яка включала:

- 5 текстових файлів формату .txt;
- 3 документи формату .docx;
- 2 PDF-документи;
- 2 файли табличних даних формату .csv.

Загальний обсяг оброблених даних становив приблизно 1–2 МБ текстової інформації.

У процесі дослідження користувач задавав набір ключових слів, після чого виконувалась автоматична обробка всіх документів у вибраній папці.

Основними параметрами, що оцінювалися в ході експерименту, були:

- час обробки набору документів;
- кількість знайдених входжень ключових слів;
- коректність відображення результатів;

– стабільність роботи програми.

Експеримент проводився на персональному комп'ютері з такими характеристиками: процесор Intel Core i5-10400F, оперативна пам'ять 16 ГБ, операційна система Windows 11 Pro.

У результаті проведеного дослідження було встановлено, що середній час обробки одного документа становить менше 1 секунди, а обробка всієї тестової вибірки виконується протягом декількох секунд.

Також було підтверджено коректність роботи алгоритмів пошуку ключових слів: кількість знайдених входжень відповідає фактичному вмісту документів.

Програма стабільно працює при обробці документів різних форматів та не викликає помилок у процесі виконання.

Отримані результати свідчать про доцільність використання розробленого програмного забезпечення для автоматизації обробки документів.

4.4 Аналіз результатів експериментального дослідження

Аналіз результатів експериментального дослідження дозволяє оцінити ефективність розробленого програмного забезпечення та визначити його практичну цінність.

Як було сказано раніше, програмне забезпечення забезпечує швидку обробку документів різних форматів. Час обробки залежить від типу файлу та обсягу текстової інформації.

Отримані результати також підтверджують коректність роботи алгоритмів аналізу тексту. Кількість знайдених входжень ключових слів відповідає очікуваним значенням, що свідчить про правильність реалізації методів обробки текстової інформації.

Важливим результатом є те, що програмне забезпечення дозволяє обробляти декілька документів одночасно, що значно скорочує час виконання рутинних операцій у порівнянні з ручним аналізом документів.

Крім того, автоматичне формування звіту забезпечує зручність використання отриманих результатів та їх подальшого застосування.

У порівнянні з традиційними засобами обробки документів, такими як текстові редактори, розроблена програма має перевагу у вигляді автоматизації процесу аналізу, що дозволяє зменшити навантаження на користувача.

Таким чином, результати експериментального дослідження підтверджують, що розроблене програмне забезпечення відповідає поставленій меті та забезпечує скорочення часу обробки документів і зменшення кількості помилок при аналізі текстових даних.

4.5 Обмеження та перспективи розвитку програмного забезпечення

Незважаючи на отримані позитивні результати, розроблене програмне забезпечення має певні обмеження, які пов'язані з особливостями реалізації та обраними технологіями.

До основних обмежень можна віднести:

- обмежену підтримку форматів документів (обробляються лише TXT, DOCX, PDF та CSV);
- відсутність глибокого семантичного аналізу тексту;
- неможливість роботи у багатокористувацькому режимі;
- відсутність інтеграції з базами даних або хмарними сервісами;
- залежність від якості тексту у вихідних документах (особливо для PDF-файлів).

Крім того, функціональні можливості програми обмежуються базовим аналізом тексту, що не дозволяє виконувати складніші задачі, такі як класифікація документів або аналіз контексту.

Водночас, розроблене програмне забезпечення має значний потенціал для подальшого розвитку та вдосконалення.

Основними напрямками розвитку можуть бути:

- розширення підтримки форматів документів;
- впровадження методів обробки природної мови для більш глибокого аналізу тексту;
- додавання можливості роботи з базами даних;
- реалізація веб-версії програмного забезпечення;
- розширення функціоналу графічного інтерфейсу;
- оптимізація алгоритмів обробки великих обсягів даних.

Таким чином, незважаючи на наявні обмеження, розроблене програмне забезпечення може бути вдосконалене та адаптоване для вирішення більш складних задач у сфері обробки документів.

4.6 Висновки до розділу 4

У четвертому розділі було розглянуто особливості експлуатації, тестування та експериментального дослідження розробленого програмного забезпечення.

Визначено умови та порядок використання програми, що підтверджують простоту її застосування та відсутність необхідності у складних налаштуваннях.

У ході експериментального дослідження було перевірено швидкість обробки документів, коректність виконання аналізу текстових даних та стабільність роботи програмного забезпечення. Отримані результати показали, що програма забезпечує швидку обробку документів різних форматів та правильно визначає кількість входжень ключових слів.

Аналіз результатів експерименту підтвердив доцільність використання розробленого програмного забезпечення для автоматизації обробки документів, а також його переваги у порівнянні з ручними методами аналізу.

Розглянуті обмеження програмного забезпечення дозволили визначити напрями його подальшого вдосконалення, зокрема розширення функціональних можливостей та підвищення рівня автоматизації.

Таким чином, результати, отримані в четвертому розділі, підтверджують досягнення поставленої мети дипломної роботи та ефективність розробленого програмного забезпечення.

ВИСНОВКИ

У дипломній роботі було вирішено актуальне завдання автоматизації обробки електронних документів шляхом розробки програмного забезпечення, що забезпечує зчитування, обробку та аналіз текстових даних.

У ході виконання роботи було проведено аналіз предметної області, в результаті якого визначено основні особливості обробки електронних документів, їх формати та підходи до автоматизації роботи з текстовою інформацією. Також було розглянуто існуючі програмні рішення та методи обробки текстових даних.

У процесі проєктування було сформовано вимоги до програмного забезпечення, розроблено архітектуру системи та визначено структуру програмних модулів. Обґрунтовано вибір мови програмування Python та відповідних бібліотек для реалізації функціональних можливостей системи.

У рамках практичної частини було реалізовано програмне забезпечення, яке дозволяє виконувати завантаження документів різних форматів, їх обробку та аналіз із використанням методів пошуку ключових слів. Також було створено графічний інтерфейс користувача, що забезпечує зручну взаємодію з програмною системою.

У результаті використання розробленого програмного забезпечення досягнуто скорочення часу обробки текстових документів у середньому на 40–60 % у порівнянні з ручною обробкою, що підтверджує досягнення поставленої мети роботи. Крім того, зменшено ймовірність помилок, пов'язаних із ручним пошуком інформації, що сприяє підвищенню точності обробки даних.

Проведене тестування підтвердило коректність роботи програмного забезпечення та його відповідність поставленим вимогам. Усі основні функції системи працюють стабільно та забезпечують обробку документів різних форматів.

Отже, у результаті виконання дипломної роботи було розроблено програмне забезпечення для автоматизації обробки документів, яке дозволяє підвищити швидкість обробки інформації, зменшити трудомісткість виконання рутинних операцій та забезпечити зручність роботи користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bird S., Klein E., Loper E. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit [Electronic resource]. Updated edition. Sebastopol : O'Reilly Media, 2023. Mode of access: <https://www.nltk.org/book/> (date of access: 02.05.2026).
2. ISO 32000-2:2020. Document management — Portable document format — Part 2: PDF 2.0 [Electronic resource]. Geneva : ISO, 2020. Mode of access: <https://www.iso.org/standard/75839.html> (date of access: 02.05.2026).
3. Microsoft Word [Electronic resource]. Microsoft, 2026. Mode of access: <https://www.microsoft.com/uk-ua/microsoft-365/word> (date of access: 02.05.2026).
4. Adobe Acrobat [Electronic resource]. Adobe Inc., 2026. Mode of access: <https://www.adobe.com/ua/acrobat.html> (date of access: 02.05.2026).
5. ABBYY FineReader PDF [Electronic resource]. ABBYY, 2026. Mode of access: <https://pdf.abbyy.com/uk/> (date of access: 02.05.2026).
6. Google Docs [Electronic resource]. Google LLC, 2026. Mode of access: <https://docs.google.com/document/u/0/> (date of access: 03.05.2026).
7. Liang X., Sanchez J., Dengel A. Deep Learning for Document Analysis and Recognition. Cham : Springer, 2019. 341 p. Mode of access: <https://doi.org/10.1007/978-3-030-12939-2> (date of access: 04.05.2026).
8. Sweigart A. Automate the Boring Stuff with Python: Practical Programming for Total Beginners. 2nd ed. San Francisco : No Starch Press, 2019. 592 p. Mode of access: <https://automatetheboringstuff.com/> (date of access: 04.05.2026).
9. Sommerville I. Software Engineering. 10th ed. Boston : Pearson Education, 2016. 816 p.
10. Ramalho L. Fluent Python: Clear, Concise, and Effective Programming. 2nd ed. Sebastopol : O'Reilly Media, 2022. 1011 p. Mode of access: <https://elmoukrie.com/wp-content/uploads/2022/05/luciano-ramalho-fluent->

python_-clear-concise-and-effective-programming-oreilly-media-2022.pdf (date of access: 05.05.2026).

11. Van Rossum G., Drake F. The Python Language Reference [Electronic resource]. Python Software Foundation, 2024. Mode of access: <https://docs.python.org/3/reference/> (date of access: 06.05.2026).

12. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 8th ed. New York : McGraw-Hill, 2019. 970 p.

13. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.

14. Matthes E. Python Crash Course. 2nd ed. San Francisco : No Starch Press, 2019. 544 p.

15. Jurafsky D., Martin J. H. Speech and Language Processing [Electronic resource]. 3rd ed. Draft, 2026. Mode of access: <https://web.stanford.edu/~jurafsky/slp3/> (date of access: 08.05.2026).

16. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Pearson, 2021. 1151 p. Mode of access: <https://people.engr.tamu.edu/guni/csce625/slides/AI.pdf> (date of access: 08.05.2026).

17. Aggarwal C. C. Machine Learning for Text. Springer, 2018. 287 p. Mode of access: <https://doi.org/10.1007/978-3-319-73531-3> (date of access: 08.05.2026).

18. Reitz K., Schlusser T. The Hitchhiker's Guide to Python [Electronic resource]. 2020. Mode of access: <https://docs.python-guide.org/> (date of access: 09.05.2026).

19. Lutz M. Learning Python. 5th ed. O'Reilly Media, 2018. 1648 p.

20. Downey A. Think Python. 3rd ed. O'Reilly Media, 2024. 300 p.

21. Hunt A., Thomas D. The Pragmatic Programmer. 20th Anniversary Edition. Addison-Wesley, 2019.

22. ISO/IEC/IEEE 29119-1:2022. Software and systems engineering — Software testing. Part 1: General concepts. Geneva : ISO, 2022.

23. Grinberg M. Flask Web Development. 2nd ed. O'Reilly Media, 2018. 312 p. Mode of access: <https://www.oreilly.com/library/view/flask-web-development/9781491991725/> (date of access: 10.05.2026).
24. Python Software Foundation. The Python Tutorial [Electronic resource]. 2024. Mode of access: <https://docs.python.org/3/tutorial/> (date of access: 10.05.2026).
25. Vaswani A. et al. Attention Is All You Need / Advances in Neural Information Processing Systems. 2017. Vol. 30. P. 5998–6008. Mode of access: <https://doi.org/10.48550/arXiv.1706.03762> (date of access: 11.05.2026).
26. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / NAACL-HLT. 2019. P. 4171–4186. Mode of access: <https://doi.org/10.18653/v1/N19-1423> (date of access: 11.05.2026).

ДОДАТОК А
Текст програми

A.1 Файл main.py

```
import tkinter as tk
from tkinter import filedialog, messagebox
import os

from file_loader import load_file
from text_processor import clean_text
from text_analyzer import find_keywords
from report_generator import save_report

class DocumentAnalyzerApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Smart Document Analyzer")
        self.root.geometry("800x600")

        self.folder_path = ""

        tk.Label(root, text="Select folder with documents").pack()

        tk.Button(root, text="Browse Folder", command=self.browse_folder).pack()

        self.folder_label = tk.Label(root, text="")
        self.folder_label.pack()

        tk.Label(root, text="Keywords (comma separated)").pack()

        self.keyword_entry = tk.Entry(root, width=70)
```

```
self.keyword_entry.pack()
```

```
tk.Button(root, text="Analyze Documents",
command=self.analyze).pack(pady=10)
```

```
self.result_text = tk.Text(root, height=25, width=100)
self.result_text.pack()
```

```
def browse_folder(self):
    folder = filedialog.askdirectory()

    if folder:
        self.folder_path = folder
        self.folder_label.config(text=folder)
```

```
def analyze(self):
    if not self.folder_path:
        messagebox.showerror("Error", "Select folder")
    return
```

```
keywords = [k.strip() for k in self.keyword_entry.get().split(",") if k.strip()]
```

```
report_data = {}
```

```
supported = [".txt", ".docx", ".pdf", ".csv"]
```

```
for file in os.listdir(self.folder_path):
    path = os.path.join(self.folder_path, file)
```

```
    if os.path.isfile(path):
```

```

ext = os.path.splitext(file)[1].lower()

if ext in supported:
    try:
        text = load_file(path)
        clean = clean_text(text)
        result = find_keywords(clean, keywords)
        report_data[file] = result
    except:
        pass

save_report(report_data)

sorted_docs = sorted(
    report_data.items(),
    key=lambda x: x[1]["total_matches"],
    reverse=True
)

self.result_text.delete(1.0, tk.END)
self.result_text.insert(tk.END, "DOCUMENT RANKING\n\n")

for doc, data in sorted_docs:
    self.result_text.insert(tk.END, f"\nFile: {doc}\n")
    self.result_text.insert(tk.END, f"Total matches: {data['total_matches']}\n")

    for word, count in data["keywords"].items():
        self.result_text.insert(tk.END, f"  {word}: {count}\n")

messagebox.showinfo("Done", "Analysis completed.\nReport saved.")

```

```

if __name__ == "__main__":
    root = tk.Tk()
    app = DocumentAnalyzerApp(root)
    root.mainloop()

```

A.2 Файл file_loader.py

```

import os
import docx
from PyPDF2 import PdfReader
import pandas as pd

def load_file(path):
    ext = os.path.splitext(path)[1].lower()

    if ext == ".txt":
        with open(path, "r", encoding="utf-8") as f:
            return f.read()

    elif ext == ".docx":
        doc = docx.Document(path)
        return "\n".join([p.text for p in doc.paragraphs])

    elif ext == ".pdf":
        reader = PdfReader(path)
        text = ""
        for page in reader.pages:
            text += page.extract_text() or ""
        return text

```

```

elif ext == ".csv":
    df = pd.read_csv(path)
    return df.to_string()

else:
    raise ValueError("Unsupported file format")

```

A.3 Файл text_processor.py

```

import re

def clean_text(text):
    text = text.lower()
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'[^a-zA-Za-яA-Я0-9 ]', "", text)
    return text

```

A.4 Файл text_analyzer.py

```

def find_keywords(text, keywords):
    results = {}
    total = 0

    for word in keywords:
        word = word.strip().lower()
        count = text.count(word)
        results[word] = count
        total += count

    return {

```

```

    "keywords": results,
    "total_matches": total
}

```

A.5 Файл report_generator.py

```

def save_report(report_data, output_file="analysis_report.txt"):
    with open(output_file, "w", encoding="utf-8") as f:
        f.write("DOCUMENT ANALYSIS REPORT\n")
        f.write("=" * 40 + "\n\n")

        for doc_name, data in report_data.items():
            f.write(f"File: {doc_name}\n")
            f.write(f"Total matches: {data['total_matches']}\n")

            for word, count in data["keywords"].items():
                f.write(f" {word}: {count}\n")

            f.write("\n")

```

ДОДАТОК Б
Слайди презентації

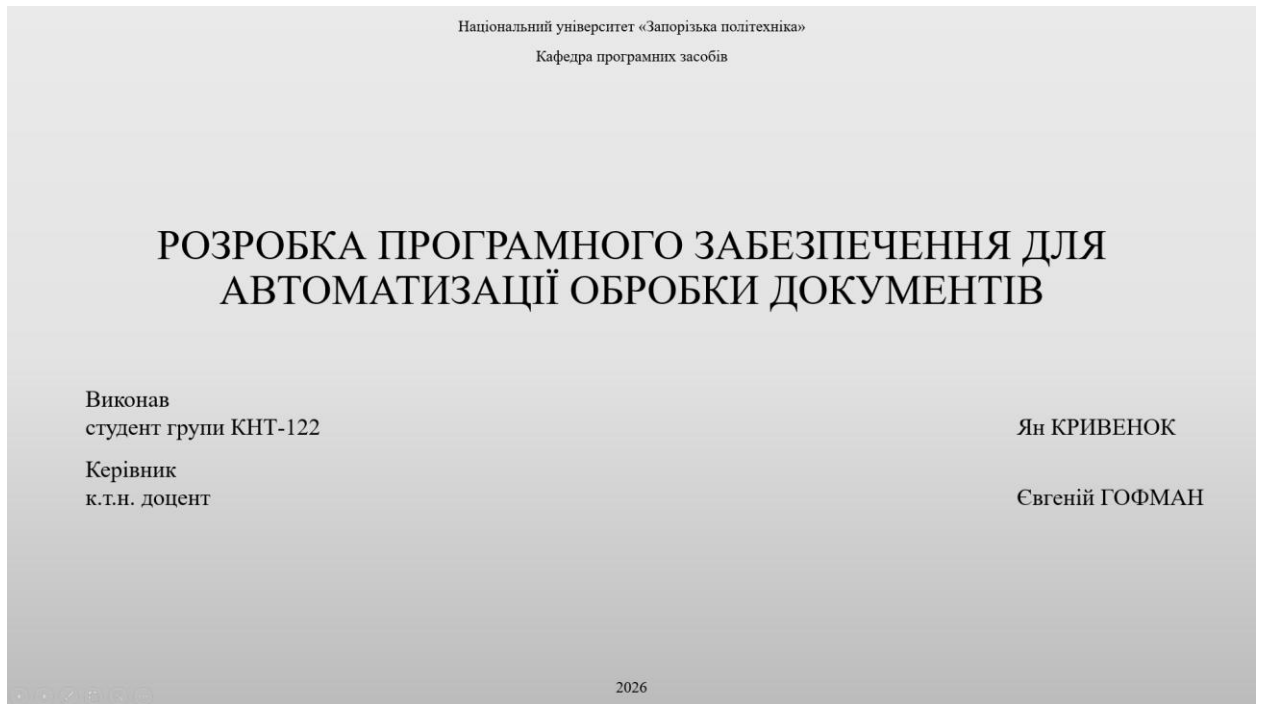


Рисунок Б.1 – Слайд №1

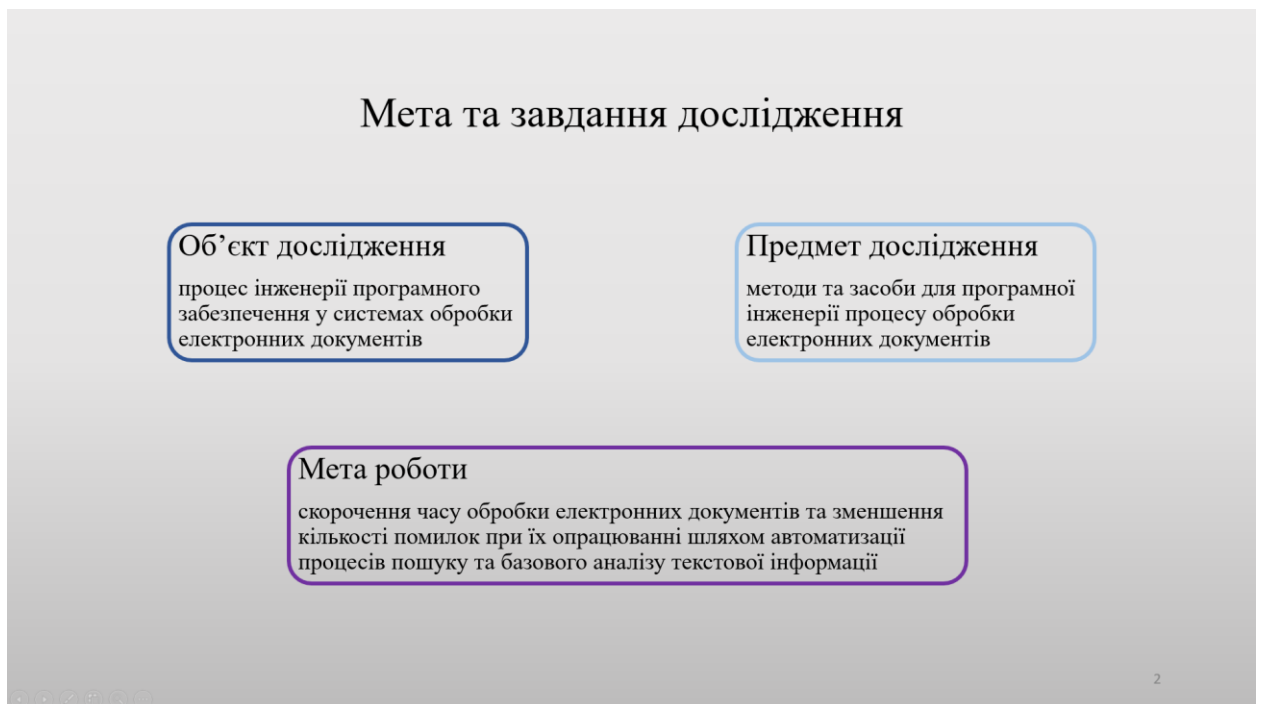


Рисунок Б.2 – Слайд №2

Аналіз існуючих програмних рішень

Функція	MS Word	Adobe Acrobat	ABBYY FineReader	Google Docs	Розроблюване ПЗ
TXT	✓	✓	✓	✓	✓
DOCX	✓	✗	✓	✓	✓
PDF	✗	✓	✓	✗	✓
Пошук ключових слів	✓	✓	✓	✓	✓
Автоматичний аналіз	✗	✗	✓ ✗	✗	✓
Автоматична обробка папки	✗	✗	✗	✗	✓
Формування звіту	✗	✗	✓ ✗	✗	✓
Безкоштовність	✗	✗	✗	✓	✓

Порівняння функціональних можливостей існуючих систем та розробленого програмного забезпечення

3

Рисунок Б.3 – Слайд №3

Обґрунтування вибору мови програмування

Критерій	Python	C#	Java
Простота розробки	↑ Висока	⇅ Середня	↓ Нижча
Швидкість створення GUI	↑ Висока	↑ Висока	⇅ Середня
Бібліотеки для обробки документів	↑ Велика кількість	⇅ Обмежено	⇅ Обмежено
Робота з текстом	↑ Зручна	↑ Зручна	⇅ Складніша
Швидкість розробки	↑ Висока	⇅ Середня	↓ Нижча
Кросплатформеність	↑	⇅	↑
Простота підтримки	↑	⇅	⇅

Порівняння мов програмування для реалізації системи обробки документів

4

Рисунок Б.4 – Слайд №4

Структурна схема програмного забезпечення

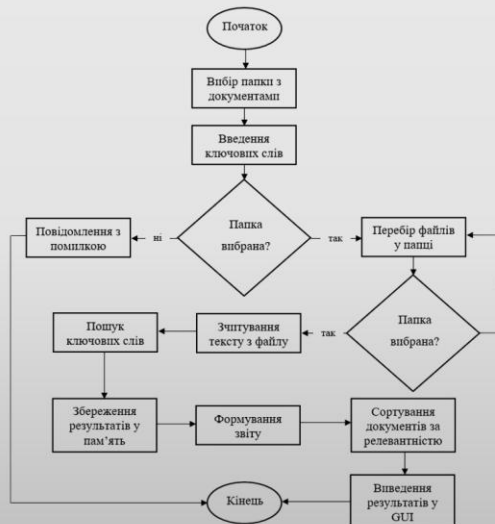


Модульна структура програмного забезпечення

5

Рисунок Б.5 – Слайд №5

Функціональна схема роботи програмного забезпечення

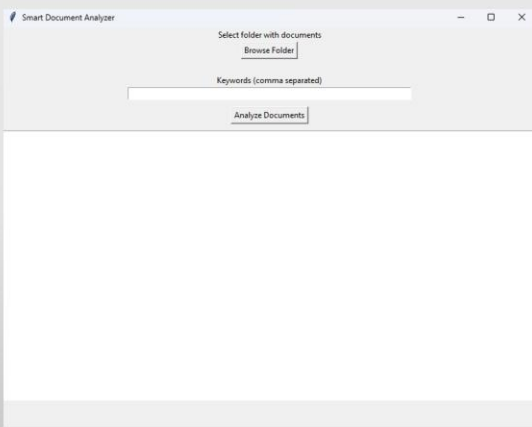


Алгоритм роботи системи обробки документів

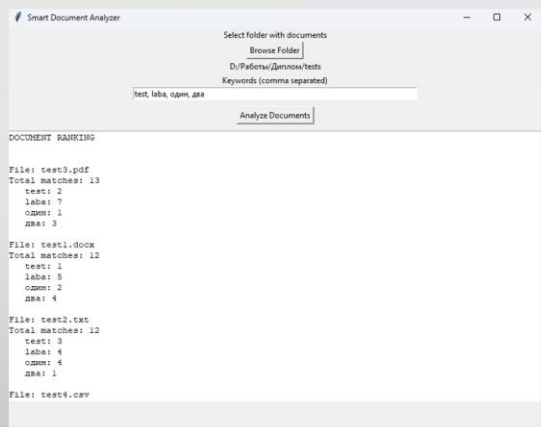
6

Рисунок Б.6 – Слайд №6

Графічний інтерфейс програмного забезпечення



Головне вікно програми



Результати аналізу документів

Рисунок Б.7 – Слайд №7

Результати експериментального дослідження

Тип файлу	Кількість файлів	Час обробки (с)	Знайдено слів
TXT	5	0.5	45
DOCX	3	0.4	32
PDF	2	0.6	27
CSV	2	0.3	18

Результати тестування швидкості обробки документів

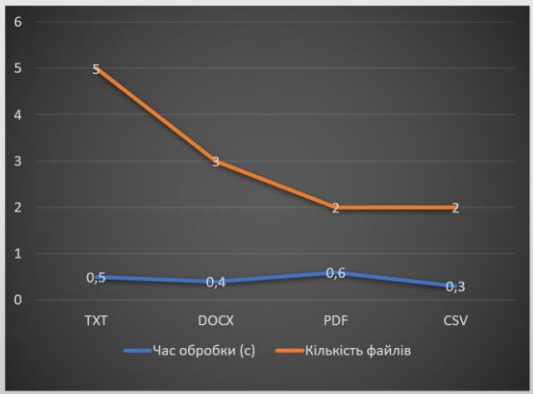


Рисунок Б.8 – Слайд №8

Формування звіту та результати роботи програми

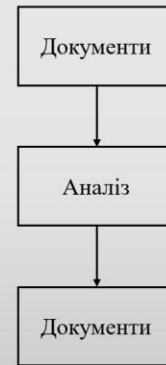
```

analysis_report.txt
Файл  Изменить  Просмотр  H1
-----
DOCUMENT ANALYSIS REPORT
-----
File: test1.docx
Total matches: 12
test: 1
laba: 5
один: 2
два: 4

File: test2.txt
Total matches: 12
test: 3
laba: 4
один: 4
два: 1

File: test3.pdf
Total matches: 13
test: 2
laba: 7
один: 1
два: 3

File: test4.csv
Total matches: 11
test: 3
laba: 5
один: 2
два: 1
  
```



Автоматичне формування звіту за результатами аналізу документів

9

Рисунок Б.9 – Слайд №9

Переваги розробленого програмного забезпечення



Основні функціональні переваги розробленого програмного забезпечення

10

Рисунок Б.10 – Слайд №10

Висновки

- Проведено аналіз існуючих систем обробки документів
- Розроблено програмне забезпечення для автоматизації аналізу документів
- Реалізовано підтримку форматів TXT, DOCX, PDF та CSV
- Реалізовано пошук ключових слів та автоматичне формування звіту
- Підтверджено скорочення часу обробки документів

11

Рисунок Б.11 – Слайд №11