

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 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАПОБІГАННЯ
НЕСАНКЦІОНОВАНОМУ ДОСТУПУ ДО ФАЙЛІВ
SOFTWARE FOR UNAUTHORIZED
FILE ACCESS PREVENTION

Виконав(ла): студент(ка) 4 курсу, групи КНТ-132

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ЖУРАВЛЬОВ Д.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ЗАЙКО Т.А.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

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(КИ)

ЖУРАВЛІОВА Данііла Владиславовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для запобігання несанкціонованому доступу до файлів. Software for Unauthorized File Access Prevention

керівник проєкту (роботи) к.т.н., доцент, ЗАЙКО Тетяна Анатоліївна,
(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. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЗАЙКО Т.А., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 20 ” 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 структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 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(ка)

_____ Данііл ЖУРАВЛЬОВ
(підпис) (Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Тетяна ЗАЙКО
(підпис) (Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
89 с., 3 табл., 20 рис., 3 дод., 17 джерел.

C#, VISUAL STUDIO, ЗАХИСТ ДАНИХ, МОВА ПРОГРАМУВАННЯ,
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ФАЙЛ.

Об'єкт дослідження – процес розробки програмного забезпечення запобігання несанкціонованому доступу до файлів.

Предмет дослідження – програмні засоби для запобігання несанкціонованому доступу до файлів.

Мета роботи – розробка програмного забезпечення для запобігання несанкціонованому доступу до файлів, що забезпечує збільшення рівня захищеності інформаційних ресурсів організації.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Visual Studio.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для запобігання несанкціонованому доступу до файлів. Створено програмне забезпечення для запобігання несанкціонованому доступу до файлів. Проведено тестування розробленої програмної системи для запобігання несанкціонованому доступу до файлів.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для запобігання несанкціонованому доступу до файлів за допомогою мови програмування C# та середовища розробки Visual Studio.

Галузь використання – захист даних.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 89 pages, 3 tables, 20 figures, 3 appendixes, 17 sources.

C#, VISUAL STUDIO, DATA PROTECTION, PROGRAMMING LANGUAGE, SOFTWARE, FILE.

The object of research is the process of developing software to prevent unauthorized access to files.

The subject of the research is software to prevent unauthorized access to files.

The purpose of this work is to develop software to prevent unauthorized access to files, which ensures an increase in the level of security of the organization's information resources.

Materials, methods and technical tools: C# programming language, Visual Studio development environment.

Results. Project solutions for the creation of software to prevent unauthorized access to files has been designed. The software to prevent unauthorized access to files has been developed. The developed software system to prevent unauthorized access to files has been tested.

Conclusions. The goal of the work has been achieved. The software tools to prevent unauthorized access to files using the C# programming language and the Visual Studio development environment have been developed.

Scope of use – data protection.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Програмне забезпечення для запобігання несанкціонованому доступу до файлів.....	11
1.2 Аналіз програмного забезпечення для запобігання несанкціонованому доступу до файлів.....	20
1.3 Висновки за розділом 1	31
2 Матеріали і методи	33
2.1 Вибір мови програмування	33
2.2 Вибір середовища розробки для створення програмного забезпечення для запобігання несанкціонованому доступу до файлів	37
2.3 Висновки за розділом 2	41
3 Опис програми	42
3.1 Структура програмного забезпечення для запобігання несанкціонованому доступу до файлів	42
3.2 Функціонування програмного забезпечення для запобігання несанкціонованому доступу до файлів	44
3.3 Опис особливостей реалізації програмного забезпечення для запобігання несанкціонованому доступу до файлів	47
3.4 Проєктування інтерфейсу програмного забезпечення для запобігання несанкціонованому доступу до файлів	50
3.5 Висновки за розділом 3	51
4 Експлуатація, тестування та експериментальне дослідження програми.....	53
4.1 Призначення й умови застосування програми	53
4.2 Характеристики програми для запобігання несанкціонованому доступу до файлів.....	55

4.3 Інструкція по експлуатації програми.....	56
4.3.1 Звернення до програми.....	56
4.3.2 Вхідні й вихідні дані.....	57
4.3.3 Повідомлення	57
4.4 Виконання програмного забезпечення для запобігання несанкціонованому доступу до файлів	57
4.5 Тестування програмного забезпечення для запобігання несанкціонованому доступу до файлів	61
4.6 Висновки за розділом 4	61
Висновки	62
Перелік джерел посилання	65
Додаток А Технічне завдання	67
Додаток Б Фрагмент тексту програми	72
Додаток В Слайди презентації	83

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

МП – мова програмування;

НД – несанкціонований доступ;

ПЗ – програмне забезпечення.

ВСТУП

У сучасному цифровому середовищі спостерігається стрімке зростання обсягів інформації, що зберігається та обробляється в електронній формі, у зв'язку з чим питання захисту файлів від несанкціонованого доступу набуває особливої значущості. Дані дедалі частіше становлять стратегічний ресурс, який має фінансову, наукову, комерційну або персональну цінність. В умовах глобальної цифровізації, розвитку хмарних сервісів і розширення можливостей віддаленої роботи підвищується ризик витоку інформації, її викрадення або модифікації сторонніми особами. Саме тому розроблення програмного забезпечення, орієнтованого на запобігання несанкціонованому доступу до файлів, розглядається як необхідний елемент сучасної інформаційної інфраструктури [1]-[3].

Зростання кількості кіберінцидентів, пов'язаних із компрометацією конфіденційних даних, підтверджує потребу у впровадженні ефективних механізмів автентифікації, авторизації, шифрування та моніторингу доступу. Поширення шкідливого програмного забезпечення, фішингових атак, соціальної інженерії та внутрішніх загроз актуалізує питання комплексного підходу до захисту інформаційних ресурсів. У такому контексті програмні рішення для запобігання несанкціонованому доступу до файлів мають не лише забезпечувати технічні бар'єри, а й інтегруватися в загальну систему управління інформаційною безпекою організації [4]-[6].

Актуальність таких програмних рішень також зумовлюється необхідністю збереження цілісності даних у корпоративних середовищах, навчальних закладах, державних установах та приватному секторі. Інформація, що міститься у файлах, може бути об'єктом промислового шпигунства, конкурентної боротьби або злочинної діяльності. У зв'язку з цим виникає потреба у створенні та вдосконаленні програмного забезпечення для запобігання несанкціонованому доступу до файлів, що відповідає сучасним викликам кібербезпеки та потребам цифрового суспільства [2]-[5].

Проте деякі програми для запобігання несанкціонованому доступу до файлів можуть бути досить складними, що може негативно впливати на продуктивність цифрової екосистеми організації. Реалізація постійного контролю доступу, журналювання подій, перевірки прав користувачів і шифрування даних потребує додаткових обчислювальних ресурсів. Проблемним аспектом виступає також складність налаштування та адміністрування деяких програм. Некоректна конфігурація політик доступу здатна або надмірно обмежити легітимних користувачів, або залишити критичні дані недостатньо захищеними. Необхідність регулярного оновлення програмного забезпечення з метою усунення виявлених вразливостей також може розглядатися як недолік. Крім того, впровадження та підтримка подібних систем часто потребують фінансових витрат, пов'язаних із придбанням ліцензій, навчанням персоналу та технічною підтримкою [3]-[7].

Тому актуальною є розробка програмного забезпечення для запобігання несанкціонованому доступу до файлів. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для запобігання несанкціонованому доступу до файлів, що забезпечує збільшення рівня захищеності інформаційних ресурсів організації.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для запобігання несанкціонованому доступу до файлів;
- здійснити проєктування програмного забезпечення для запобігання несанкціонованому доступу до файлів;
- створити програмне забезпечення для запобігання несанкціонованому доступу до файлів;
- виконати тестування розробленого програмного забезпечення для запобігання несанкціонованому доступу до файлів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для запобігання несанкціонованому доступу до файлів

Програмне забезпечення для запобігання несанкціонованому доступу до файлів являє собою комплекс засобів і технологій, призначених для захисту інформації від стороннього втручання та несанкціонованого перегляду, копіювання або зміни. Такі системи створюють додатковий рівень безпеки для даних, які зберігаються на комп'ютерах, серверах або знімних носіях, забезпечуючи контроль над тим, хто може отримувати доступ до файлів і які дії з ними можна виконувати. Основна мета такого програмного забезпечення полягає у гарантуванні конфіденційності, цілісності та доступності інформації, незалежно від того, чи вона передається мережею, зберігається локально, чи обробляється на віддалених серверах [1].

Особливістю таких програмних систем є поєднання кількох механізмів захисту. Вони можуть використовувати шифрування для перетворення даних у формат, який неможливо прочитати без відповідних ключів або паролів, а також забезпечувати контроль доступу через автентифікацію користувачів. Деякі рішення надають можливість створювати захищені сховища, де файли зберігаються із обмеженим доступом, або автоматично виконують резервне копіювання та відновлення даних для захисту від втрати інформації. Крім того, у таких програмах передбачено безпечне видалення файлів, що унеможлиблює їх відновлення після видалення [1].

Перевагами використання подібного програмного забезпечення є значне підвищення захищеності інформації, зменшення ризику витоку або несанкціонованого використання даних, а також забезпечення контролю над їхнім життєвим циклом. Воно дозволяє організаціям і приватним користувачам зберігати файли в безпечному середовищі, управляти доступом на рівні окремих документів або каталогів і запроваджувати політики безпеки відповідно до власних потреб. Такі системи також підвищують впевненість у

збереженні важливої інформації, зменшують залежність від фізичних носіїв і забезпечують можливість швидкого відновлення даних у разі технічних збоїв або небажаного втручання [1].

Забезпечення захищеності програмних продуктів орієнтоване на своєчасне виявлення вразливостей і мінімізацію потенційних небезпек, що можуть впливати на стабільність та надійність роботи систем. Джерелами ризиків виступають як зовнішні чинники, так і внутрішні недоліки, пов'язані з помилками розроблення, прорахунками в архітектурі або некоректною конфігурацією. За відсутності належного контролю подібні слабкі місця здатні призвести до компрометації даних, фінансових втрат і підриву репутації організації. У цьому контексті безпека програмного забезпечення розглядається як комплексний захисний механізм, що стримує вплив загроз та зменшує їхні наслідки [1]-[3].

Небезпеки, які впливають на програмні системи, умовно поділяються за джерелом виникнення. Зовнішні впливи пов'язані з діяльністю осіб або груп, що намагаються використати виявлені вразливості для отримання несанкціонованого доступу до інформації або порушення функціонування сервісів. До таких дій належать поширення шкідливих програм, спроби перевантаження інфраструктури запитами з метою блокування доступу, а також застосування методів соціальної інженерії для виманювання конфіденційних відомостей. Наслідком подібних інцидентів може стати витік персональних чи фінансових даних, що створює суттєві ризики для користувачів і компаній [1]-[3].

Внутрішні фактори небезпеки програмного забезпечення також мають значний вплив. Загрози можуть виникати через навмисні дії осіб, які вже мають доступ до інформаційних ресурсів, або внаслідок випадкових помилок персоналу. Неправильні налаштування, необережне поводження з конфіденційними даними чи недостатній контроль доступу здатні відкрити шлях до серйозних порушень безпеки. Навіть за відсутності злого наміру

людський фактор залишається одним із найвразливіших елементів будь-якої інформаційної системи [1]-[3].

Значення належного рівня захищеності програмних продуктів важко переоцінити, оскільки від цього залежить збереження конфіденційних відомостей, стабільність діяльності організацій та довіра користувачів. Інформаційні системи часто оперують персональними й фінансовими даними, тому їх компрометація може спричинити матеріальні збитки та юридичні наслідки. Крім того, інциденти безпеки здатні призвести до зупинки бізнес-процесів і негативно вплинути на репутацію. Дотримання встановлених нормативних вимог у сфері захисту інформації також є важливою складовою, адже недотримання правил може спричинити санкції та втрату довіри з боку партнерів і клієнтів [1]-[3].

Окремо варто зазначити, що програмні продукти часто становлять значну інтелектуальну цінність. Їхнє несанкціоноване використання або копіювання може завдати суттєвих збитків. Саме тому впровадження ефективних механізмів захисту, регулярне оновлення та аудит систем безпеки є невід'ємними елементами відповідального управління інформаційними ресурсами [1]-[3].

Забезпечення належного рівня захищеності програмних систем супроводжується значною кількістю труднощів, що виникають на різних етапах їх створення та експлуатації. Однією з ключових проблем залишається підтримання конфіденційності, цілісності та доступності інформації в умовах постійного розвитку технологій. Ускладнення архітектури сучасних рішень призводить до збільшення кількості потенційних вразливостей, пошук і усунення яких потребують значних зусиль. Чим масштабнішою та багатокомпонентнішою стає система, тим складніше гарантувати її повну стійкість до зловмисного впливу [1]-[3].

Додатковим викликом є динамічність кіберзагроз. Методи атак постійно вдосконалюються, з'являються нові способи обходу захисних механізмів, а вже відомі вразливості отримують неочікувані варіації

використання. У таких умовах заходи безпеки повинні безперервно оновлюватися, інакше ризик компрометації значно зростає. Водночас недостатній рівень обізнаності серед розробників або користувачів може сприяти впровадженню небезпечних практик, що відкривають шлях до інцидентів. Обмежені фінансові й кадрові ресурси, особливо в невеликих організаціях, часто ускладнюють впровадження комплексного підходу до захисту. Окрему складність становить підтримка застарілих систем, які не розраховані на сучасні механізми протидії загрозам, але продовжують використовуватися з виробничих причин [1]-[3].

Для мінімізації ризиків застосовуються різноманітні інструменти та методики. На етапі розроблення здійснюється аналіз вихідного коду з метою виявлення потенційних недоліків ще до запуску продукту в експлуатацію. Окрім перевірки коду, використовується тестування вже функціонуючих застосунків шляхом моделювання можливих сценаріїв атак, що дозволяє оцінити їхню стійкість у реальних умовах. Значну роль відіграють засоби фільтрації мережевого трафіку, які перешкоджають спробам впливу на вебзастосунки через поширені вектори атак [2]-[4].

Моніторинг подій безпеки та аналіз журналів активності допомагають своєчасно виявляти підозрілу поведінку в мережі й реагувати на інциденти. Використовуються також механізми сканування інфраструктури для пошуку відомих вразливостей, що дає змогу визначати пріоритетність усунення ризиків. Важливою складовою залишається впровадження надійних процедур автентифікації та контролю доступу, які обмежують можливість використання системи сторонніми особами. Не менш суттєвим є дотримання принципів безпечного програмування, що передбачає врахування потенційних загроз ще на стадії проєктування [2]-[4].

Підвищення рівня обізнаності персоналу розглядається як один із ключових елементів захисту, оскільки людський фактор часто стає причиною інцидентів. Регулярне оновлення програмного забезпечення та своєчасне встановлення виправлень допомагають усувати виявлені вразливості та

зменшувати ризик їх експлуатації. У сукупності такі заходи формують багаторівневу систему безпеки, яка спрямована на запобігання атакам, мінімізацію можливих збитків і підтримання стабільної роботи інформаційних ресурсів [2]-[4].

Забезпечення належного рівня захисту програмних систем розглядається як безперервний процес, що охоплює всі стадії їх створення та експлуатації. Досягнення стійкості до загроз передбачає застосування інженерного підходу, за якого вимоги безпеки інтегруються в архітектуру продукту ще на етапі проєктування. Упродовж життєвого циклу програмного забезпечення здійснюється систематичний контроль якості коду, аналіз потенційних ризиків і перевірка відповідності встановленим стандартам. Особлива увага приділяється формуванню безпечних принципів розроблення, що мінімізують ймовірність появи критичних помилок [4]-[5].

Важливою умовою підтримання захищеності є своєчасне оновлення програмних компонентів та усунення відомих вразливостей. У процесі експлуатації здійснюється регулярна перевірка застосунків шляхом моделювання можливих сценаріїв атак, що дає змогу оцінити ефективність упроваджених механізмів захисту. Контроль доступу до ресурсів реалізується через впровадження чітко визначених політик авторизації, багаторівневої автентифікації та обмеження прав відповідно до службової необхідності. Конфіденційні дані підлягають криптографічному захисту як під час зберігання, так і під час передавання каналами зв'язку, що знижує ризик їх перехоплення або несанкціонованого використання [4]-[5].

У сучасному цифровому середовищі захищеність програмних рішень виступає фундаментальною умовою стабільної діяльності організацій. Вона дозволяє запобігати витоку даних, фінансовим збиткам та порушенню безперервності сервісів. Нехтування вимогами безпеки здатне призвести до втрати довіри користувачів і значних репутаційних ризиків. Саме тому впровадження комплексного підходу до захисту, що поєднує технічні,

організаційні та процедурні заходи, розглядається як необхідна умова ефективного функціонування інформаційних систем [4]-[5].

Несанкціонований доступ до програмного забезпечення трактується як отримання можливості взаємодії з ресурсами без відповідних повноважень. Такий доступ може здійснюватися шляхом обходу механізмів захисту або використання технічних недоліків інфраструктури. Наслідки подібних порушень варіюються від компрометації конфіденційної інформації до повної втрати контролю над окремими сегментами мережі. У зв'язку з цим реалізація превентивних заходів і своєчасне виявлення підозрілої активності визнаються критично важливими для підтримання цілісності та надійності цифрового середовища [4]-[5].

Отримання доступу до інформаційних ресурсів та програмних засобів без належних повноважень може здійснюватися різними способами, які відрізняються складністю та рівнем підготовки злоумисників. Досить поширеним залишається автоматизований підбір облікових даних із використанням спеціалізованих інструментів, що перебирають можливі комбінації паролів і кодів. Уразливості програмного забезпечення також активно використовуються для проникнення в системи, оскільки помилки конфігурації або недоліки коду можуть відкривати шлях до внутрішніх ресурсів. Окрему групу загроз становлять методи психологічного впливу, за допомогою яких користувачів спонукають розкривати конфіденційну інформацію або виконувати небезпечні дії. Підроблені повідомлення, імітація офіційних запитів і маскування під довірені джерела часто стають інструментами для викрадення облікових даних [4]-[5].

Ризики підвищуються у випадках, коли до корпоративних програмних систем мають доступ сторонні організації або партнери. Недостатній рівень захищеності їхньої інфраструктури може стати непрямым каналом проникнення до критично важливих ресурсів. Додатковою проблемою визнається повторне використання однакових паролів у різних сервісах, що дозволяє поширювати компрометацію з одного середовища на інше. У разі

виявлення інциденту характер реагування визначається масштабом порушення, типом скомпрометованих активів і потенційними наслідками для діяльності організації. Універсального алгоритму протидії не існує, тому стратегія кіберзахисту повинна бути гнучкою та адаптованою до конкретних умов [4]-[5].

Запобігання несанкціонованому втручанню передбачає системне поєднання технічних і організаційних заходів. Ефективною практикою вважається обмеження прав доступу відповідно до службової необхідності, коли користувачам надається лише мінімальний обсяг повноважень для виконання їхніх функцій. Перегляд і коригування дозволів здійснюється регулярно з урахуванням контексту та рівня ризику. Значна увага приділяється формуванню політики управління паролями, що визначає вимоги до складності, унікальності та періодичності їх оновлення, а також розподіляє відповідальність за контроль облікових записів. Дотримання галузевих стандартів і рекомендацій сприяє підвищенню загального рівня захищеності [4]-[5].

Додатковий рівень безпеки забезпечується застосуванням багатофакторної автентифікації, за якої підтвердження особи відбувається не лише за допомогою пароля, а й через інші незалежні чинники перевірки. Такий підхід значно ускладнює використання викрадених облікових даних і зменшує ймовірність успішної компрометації облікових записів. Комплексне впровадження перелічених механізмів у поєднанні з постійним моніторингом подій безпеки створює передумови для своєчасного виявлення підозрілої активності та мінімізації можливих збитків [4]-[5].

Виявлення спроб несанкціонованого втручання значною мірою залежить від постійного спостереження за діями користувачів у межах програмного забезпечення. Фіксація операцій, що виконуються з обліковими записами, дозволяє своєчасно помічати нетипові сценарії поведінки та запобігати внутрішнім загрозам. Для цього впроваджуються спеціалізовані інструменти контролю, здатні реєструвати події в журналах, аналізувати

мережеві з'єднання, відстежувати зміни у файлах і реагувати на підозрілі дії. Аналітична обробка зібраних даних створює можливість оперативного виявлення аномалій, що можуть свідчити про спробу обходу встановлених обмежень або про компрометацію облікового запису [5]-[6].

Посилення стійкості до зовнішніх і внутрішніх атак досягається також завдяки комплексному захисту інфраструктури. Поєднання програмних засобів моніторингу з мережевими механізмами фільтрації трафіку формує додатковий бар'єр для несанкціонованих підключень до серверів, баз даних і вебзастосунків. Регулярна перевірка систем на наявність вразливостей і проведення контрольованого тестування допомагають виявляти слабкі місця до того, як ними скористаються зловмисники. Нехтування своєчасним оновленням програмних компонентів здатне призвести до масштабних інцидентів, оскільки відомі помилки можуть залишатися відкритими для експлуатації протягом тривалого часу [5]-[6].

Захист інформації під час її передавання мережею розглядається як критично важливий елемент кібербезпеки. Криптографічне перетворення трафіку унеможливорює його прочитання сторонніми особами навіть у разі перехоплення пакетів даних. Особливу увагу приділяють внутрішнім з'єднанням між серверами та компонентами інфраструктури, оскільки в межах центрів обробки даних передавання інформації інколи залишається незашифрованим. У разі проникнення до внутрішньої мережі це створює ризик прихованого спостереження за обміном даними між вузлами системи. Саме тому дедалі ширше впроваджується контроль і аналіз мережевої активності з довготривалим зберіганням журналів для подальшого розслідування інцидентів. Криптографічний захист застосовується як для віддалених підключень користувачів, так і для взаємодії між серверами, використовуючи сучасні протоколи захищеного з'єднання та механізми створення захищених тунелів [5]-[6].

Не менш важливим напрямом є забезпечення інформації під час її зберігання. Дані, що записуються на фізичні або віртуальні носії, підлягають

шифруванню з використанням секретних ключів, доступ до яких суворо регламентується. Такий підхід унеможливорює читання вмісту накопичувачів без відповідних повноважень навіть у разі фізичного або віддаленого доступу до обладнання. Управління ключами передбачає чітке розмежування прав і контроль використання, що мінімізує ризик несанкціонованого розкриття інформації. Додатково здійснюється аудит усіх середовищ, де потенційно можуть зберігатися копії даних, включаючи кеші та тимчасові сховища, оскільки навіть допоміжні компоненти можуть стати джерелом витоку [6]-[7].

Суттєву загрозу для інформаційних систем становлять шкідливі програми, здатні проникати до середовища під виглядом легітимних файлів або сервісів. Вони можуть маскуватися в електронних повідомленнях, вкладеннях або вебресурсах і після активації відкривати стороннім особам доступ до конфіденційної інформації. Протидія таким загрозам ґрунтується на використанні антивірусних рішень, регулярному скануванні систем та обмеженні запуску невідомих застосунків. Додатковий рівень контролю забезпечується через визначення переліку дозволених програм, що можуть виконуватися в корпоративному середовищі. У цьому випадку будь-яке програмне забезпечення, яке не входить до затвердженого списку, блокується, що значно знижує ймовірність запуску небезпечного коду навіть у разі компрометації облікового запису [6]-[7].

Визначено, що програмне забезпечення для запобігання несанкціонованому доступу до файлів являє собою комплекс засобів і технологій, призначених для захисту інформації від стороннього втручання та несанкціонованого перегляду, копіювання або зміни. Такі системи створюють додатковий рівень безпеки для даних, які зберігаються на комп'ютерах, серверах або знімних носіях, забезпечуючи контроль над тим, хто може отримувати доступ до файлів і які дії з ними можна виконувати. Основна мета такого програмного забезпечення полягає у гарантуванні конфіденційності, цілісності та доступності інформації, незалежно від того, чи вона передається мережею, зберігається локально, чи обробляється на віддалених серверах.

1.2 Аналіз програмного забезпечення для запобігання несанкціонованому доступу до файлів

Проаналізуємо програмне забезпечення для запобігання несанкціонованому доступу до файлів [8]-[13].

Програмне забезпечення Files.com (рис. 1.1) є хмарним рішенням, призначеним для безпечного зберігання та контрольованого обміну файлами. У його основі лежить підхід централізації даних, коли всі цифрові матеріали розміщуються в єдиному захищеному середовищі, що зменшує ризики несанкціонованого доступу та втрати інформації. Передавання та збереження файлів супроводжується сучасними криптографічними механізмами, завдяки яким дані залишаються захищеними як під час переміщення мережею, так і під час перебування у сховищі [8], [11].

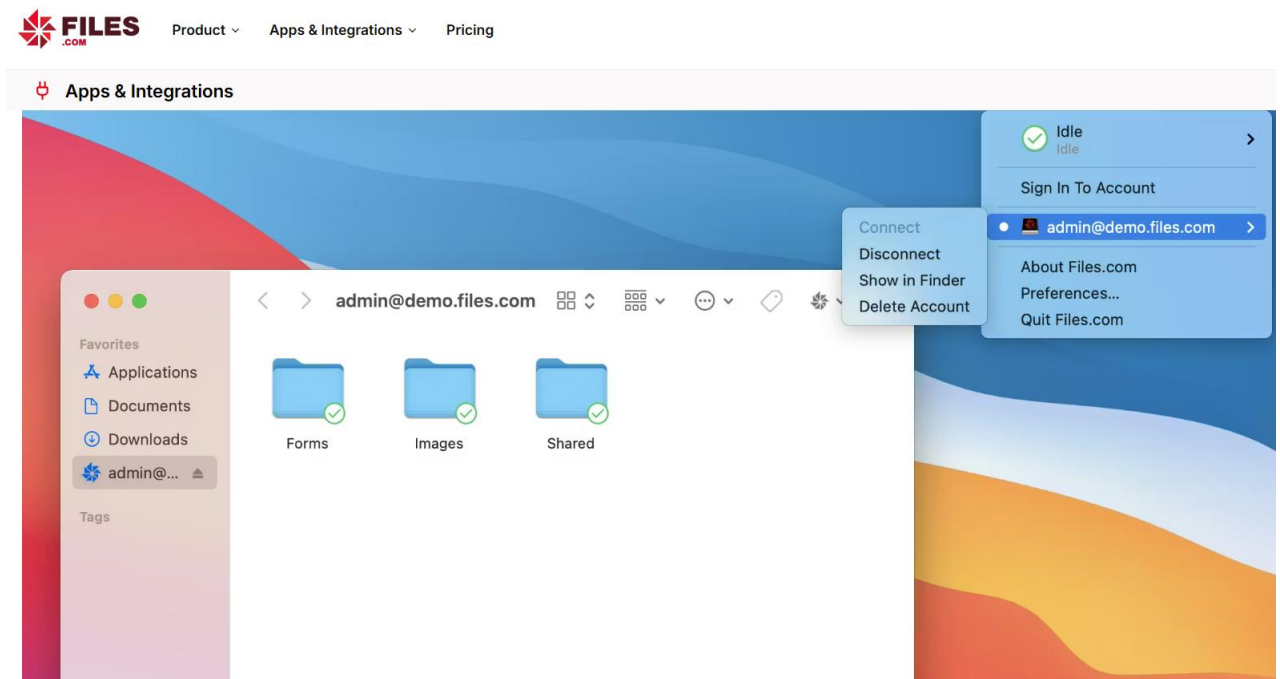


Рисунок 1.1 – Програмне забезпечення Files.com [11]

У програмному забезпеченні Files.com реалізовано гнучке адміністрування прав доступу. Кожному користувачеві надається окремий обліковий запис, що дозволяє визначати межі дозволених дій та відстежувати

всі операції з файлами. Передбачено можливість призначення різних рівнів повноважень залежно від ролі, що дає змогу регулювати, хто може переглядати, редагувати або поширювати матеріали. Доступ може надаватися через спеціальні запрошення-посилання, що забезпечує контрольований механізм взаємодії без необхідності пересилання вкладень електронною поштою [8], [11].

Платформа функціонує як централізоване хмарне сховище, яке поєднує можливості управління файлами та інструменти для командної роботи. Інтеграція з екосистемою Microsoft 365 сприяє зручній спільній роботі над документами без втрати контролю над місцем їх зберігання. За умови організації робочих процесів виключно через хмарне середовище досягається вищий рівень керованості даними, оскільки всі матеріали створюються та зберігаються в межах контрольованої інфраструктури. Адміністративні інструменти дають змогу здійснювати моніторинг активності, аналізувати використання ресурсів та забезпечувати прозорість доступу [8], [11].

Особлива увага приділяється захисту інформації за допомогою стійких алгоритмів шифрування, які застосовуються у сферах з підвищеними вимогами до безпеки. Це робить платформу придатною для зберігання конфіденційних даних і підтримання безперервності діяльності у разі непередбачених обставин. Розміщення файлів у хмарі також сприяє організації роботи розподілених команд, яким необхідний безпечний і постійний доступ до спільних ресурсів [8], [11].

Використання такого сервісу дозволяє відмовитися від локальних файлових серверів, що зменшує витрати на фізичну інфраструктуру та звільняє простір. Централізоване зберігання мінімізує дублювання документів, оскільки підтримується єдина актуальна версія кожного файлу. Передбачено можливість надання зовнішнім користувачам права перегляду без обов'язкового завантаження, що підвищує рівень контролю над розповсюдженням інформації. Обмін файлами здійснюється через безпечні

посилання, що знижує ризики, пов'язані з передаванням вкладень електронною поштою [8], [11].

Рішення підходить як для невеликих організацій, так і для великих компаній. Невеликий бізнес отримує можливість оптимізувати витрати на обладнання, тоді як масштабні структури можуть користуватися гнучким і розширюваним сховищем із високим рівнем захисту. Водночас слід враховувати, що ознайомчий доступ до функціоналу обмежений у часі, тому для повноцінного використання можливостей сервісу передбачається перехід на комерційну модель [8], [11].

Основними характеристиками програмного забезпечення Files.com є такі [8], [11]:

- хмарне сховище: централізоване розміщення файлів у захищеному онлайн середовищі з можливістю доступу з різних пристроїв без прив'язки до локальної інфраструктури [8], [11];

- захищена передача даних: шифрування інформації під час зберігання та передавання мережею для запобігання перехопленню й несанкціонованому доступу [8], [11];

- керування доступом: налаштування прав користувачів із визначенням дозволених дій щодо перегляду, редагування чи поширення файлів [8], [11];

- рольова модель: розподіл повноважень відповідно до функцій співробітників із можливістю гнучкого адміністрування [8], [11];

- обмін через посилання: надання доступу до файлів за допомогою захищених запрошень без необхідності надсилання вкладень електронною поштою [8], [11];

- інтеграція з бізнес-сервісами: підтримка взаємодії з корпоративними платформами, зокрема з екосистемою Microsoft 365, для організації спільної роботи [8], [11];

- аудит і моніторинг: ведення журналів активності користувачів для контролю операцій із файлами та підвищення прозорості процесів [8], [11];

- підтримка великих файлів: ефективне оброблення та передавання об'ємних даних без втрати продуктивності [8], [11];
- зменшення дублювання: збереження єдиної актуальної версії документа з метою уникнення надлишкових копій [8], [11];
- масштабованість: можливість розширення обсягу сховища відповідно до потреб організації без встановлення додаткового обладнання [8], [11].

Серед недоліків програмного забезпечення Files.com можна відзначити обмежений пробний доступ до функціоналу, що ускладнює повноцінне тестування можливостей сервісу перед ухваленням рішення щодо впровадження, а також залежність від стабільного Інтернет з'єднання, без якого робота з файлами стає неможливою [8], [11].

Програмне забезпечення Cove (рис. 1.2) позиціонується як хмарне рішення для захисту даних, орієнтоване на запобігання втраті інформації та несанкціонованому доступу до файлів [8], [12].

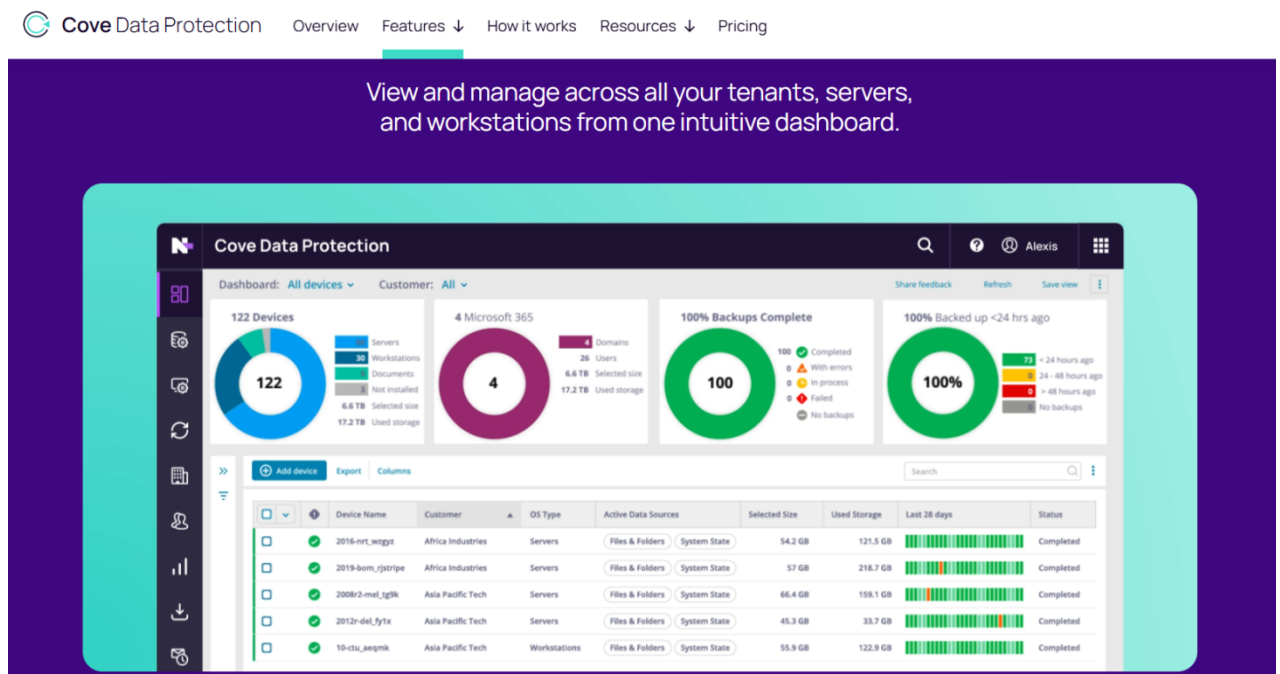


Рисунок 1.2 – Програмне забезпечення Cove [12]

Програмне забезпечення Cove функціонує у віддаленій інфраструктурі та забезпечує автоматизоване резервне копіювання ресурсів незалежно від їх

фізичного розташування. Такий підхід дозволяє організувати централізований контроль над даними для розподілених середовищ і підтримувати цілісність інформаційних активів навіть за умови роботи з кількома офісами або майданчиками [8], [12].

Архітектура програмного забезпечення Cove передбачає створення резервних копій із подальшим зберіганням у захищеному хмарному сховищі. Передавання інформації здійснюється із застосуванням стійких алгоритмів шифрування, що мінімізує ризик перехоплення під час транспортування мережею. Дані, які зберігаються у віддалених центрах обробки, також перебувають у зашифрованому вигляді, що унеможливорює їх використання сторонніми особами навіть у разі несанкціонованого доступу до інфраструктури. Для підвищення надійності копії додатково дублюються в межах хмарного середовища, що забезпечує збереження інформації у випадку технічних збоїв [8], [12].

Програмне забезпечення Cove створювалося з урахуванням потреб постачальників керованих послуг, яким необхідно одночасно адмініструвати інфраструктуру кількох клієнтів. Водночас функціональність платформи є придатною і для внутрішніх підрозділів великих організацій, які потребують централізованого інструменту резервування та відновлення. Інтеграція з іншими компонентами дає змогу поєднувати захист даних із засобами віддаленого моніторингу та управління, що спрощує адміністрування та підвищує прозорість процесів [8], [12].

Cove забезпечує резервне копіювання як окремих файлів, так і повних серверних середовищ, включно з операційними системами. Підтримується робота з фізичними та віртуальними машинами, а також із корпоративними службами обміну повідомленнями, платформами спільної роботи та базами даних. Передбачено механізми створення версій копій, що дозволяє повернутися до попереднього стану даних у разі виявлення несанкціонованих змін або шкідливого втручання. Це має особливе значення в ситуаціях, коли

порушення цілісності інформації стає очевидним лише через певний час після інциденту [8], [12].

Завдяки розміщенню резервних копій поза межами основної інфраструктури організації досягається додатковий рівень стійкості до фізичних пошкоджень або кібератак. У разі втрати працездатності основного майданчика відновлення може бути виконане на альтернативному обладнанні, що сприяє безперервності бізнес-процесів. Автоматизований характер резервування та виконання інкрементальних копій у фоновому режимі зменшує навантаження на персонал і дозволяє підтримувати актуальність збережених даних без постійного втручання [8], [12].

Таким чином, Cove виступає комплексним інструментом для захисту інформації, який поєднує хмарне зберігання, криптографічний захист, гнучке відновлення та підтримку різноманітних середовищ. Хоча рішення насамперед орієнтоване на компанії, що надають послуги з адміністрування, його можливості можуть бути ефективно використані й корпоративними структурами, які прагнуть підвищити рівень безпеки файлів та мінімізувати ризики втрати даних [8], [12].

Програмне забезпечення Cove має такі особливості [8], [12]:

- хмарна архітектура: програмне забезпечення Cove підтримує організацію резервного копіювання та відновлення даних через віддалену інфраструктуру без потреби у локальних сховищах [8], [12];

- автоматизоване резервування: програмне забезпечення Cove забезпечує виконання копіювання у фоновому режимі з мінімальним втручанням адміністратора [8], [12];

- інкрементальні копії: програмне забезпечення Cove підтримує збереження лише змінених даних для оптимізації трафіку та прискорення процесів відновлення [8], [12];

- наскрізне шифрування: захист інформації під час передавання та зберігання із застосуванням стійких криптографічних алгоритмів програмного забезпечення Cove [8], [12];

- багатосайтове управління: програмне забезпечення Cove надає централізований контроль даних для організацій із кількома офісами або клієнтськими середовищами [8], [12];

- підтримка серверних середовищ: резервне копіювання фізичних і віртуальних машин разом з операційними системами [8], [12];

- відновлення на альтернативне обладнання: програмне забезпечення Cove підтримує можливість розгортання даних на нових або чистих серверах у разі збою чи інциденту [8], [12];

- версійність копій: програмне забезпечення Cove підтримує збереження попередніх станів файлів для повернення до коректної версії після несанкціонованих змін [8], [12];

- інтеграція з іншими засобами: поєднання із засобами віддаленого моніторингу та адміністрування для комплексного управління [8], [12];

- включене хмарне сховище: програмне забезпечення Cove надає простір для розміщення резервних копій у межах сервісу без окремого придбання ресурсів [8], [12].

Серед недоліків програмного забезпечення Cove можна відзначити орієнтованість насамперед на постачальників керованих послуг, що може ускладнювати використання для невеликих організацій без спеціалізованої цифрової структури, а також залежність від стабільного Інтернет з'єднання для виконання резервного копіювання та відновлення даних [8], [12].

Програмне забезпечення Folder Lock (рис. 1.3) є інструментом для обмеження доступу до цифрових матеріалів і створення додаткового рівня безпеки для конфіденційної інформації. Програмний продукт Folder Lock орієнтований на запобігання перегляду, копіюванню або зміні файлів сторонніми особами шляхом застосування механізмів блокування та шифрування. Захист може поширюватися не лише на окремі документи, а й на цілі каталоги, знімні носії та інші сховища даних, що підключаються до пристрою [9], [13].

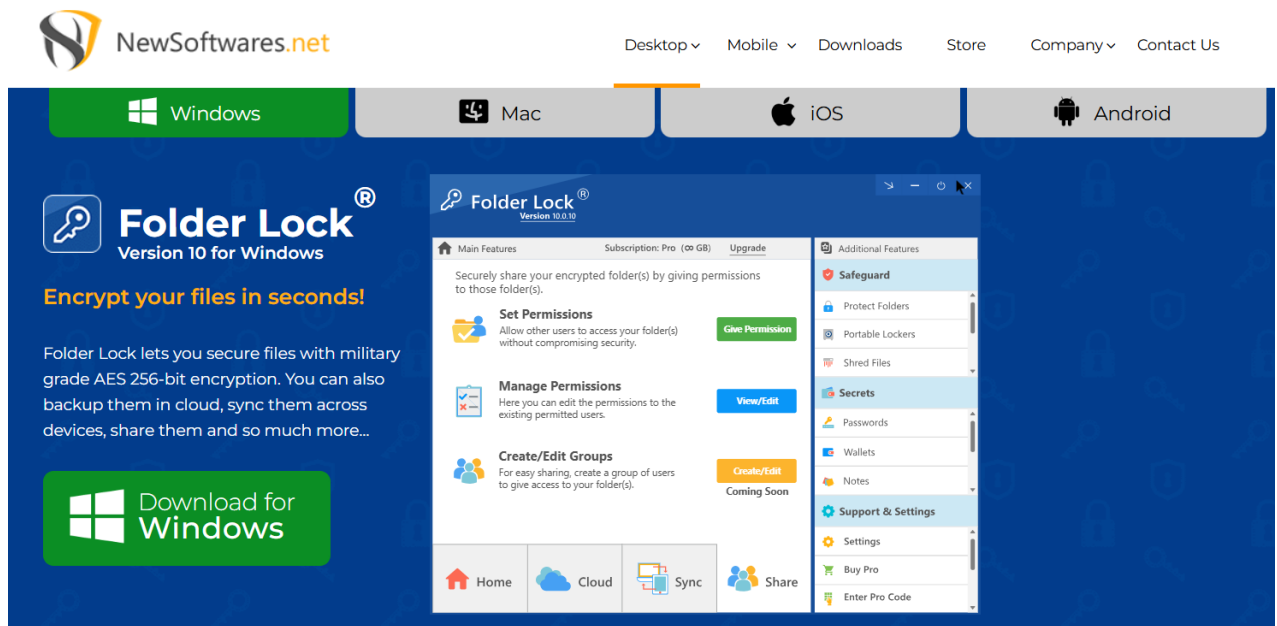


Рисунок 1.3 – Програмне забезпечення Folder Lock [13]

У межах програмного середовища Folder Lock передбачено можливість створення захищеного простору, куди переміщуються файли для подальшого зберігання в зашифрованому вигляді. Доступ до такого сховища контролюється за допомогою єдиного пароля, який виконує функцію основного ключа. Без його введення перегляд або відкриття вмісту стає неможливим, що суттєво знижує ризик несанкціонованого ознайомлення з даними навіть у разі фізичного доступу до комп'ютера чи носія [9], [13].

Особливістю рішення Folder Lock є не лише приховування інформації, а й застосування криптографічного перетворення, яке унеможливорює читання файлів поза програмою. Таким чином забезпечується подвійний захист: обмеження доступу через автентифікацію та перетворення вмісту у формат, непридатний для використання без відповідних повноважень. Такий підхід дозволяє зберігати важливі документи, персональні матеріали або фінансові дані з підвищеним рівнем конфіденційності [9], [13].

Окрему увагу у Folder Lock приділено безпечному видаленню інформації. Звичайне переміщення файлів до кошика або очищення накопичувача не гарантує повного зникнення даних, оскільки вони можуть бути відновлені спеціалізованими засобами. У програмі Folder Lock

передбачено механізм безповоротного знищення, який перезаписує інформаційні області носія, роблячи відновлення практично неможливим. За потреби може виконуватися очищення вільного простору диска, що усуває залишкові сліди раніше видалених матеріалів [9], [13].

Завдяки поєднанню блокування, шифрування та безпечного стирання Folder Lock розглядається як комплексний засіб для контролю доступу до файлів. Використання такого програмного забезпечення сприяє захисту персональної та службової інформації від несанкціонованого втручання, витоку або відновлення після видалення [9], [13].

Програмне забезпечення Folder Lock має такі особливості [9], [13]:

- блокування папок: програмна система Folder Lock забезпечує обмеження доступу до каталогів шляхом встановлення пароля та приховування їх вмісту від сторонніх користувачів [9], [13];

- шифрування файлів: програмна система Folder Lock забезпечує перетворення даних у захищений формат, який неможливо прочитати без авторизації в програмі [9], [13];

- захист зовнішніх носіїв: програмна система Folder Lock надає можливість обмеження доступу до USB-накопичувачів та інших підключених пристроїв зберігання [9], [13];

- створення захищених сховищ: програмна система Folder Lock забезпечує формування спеціальних контейнерів для безпечного зберігання конфіденційної інформації [9], [13];

- головний пароль: використання єдиного ключа доступу для відкриття зашифрованого простору [9], [13];

- безпечне видалення даних: програмна система Folder Lock забезпечує остаточне знищення файлів без можливості їх подальшого відновлення спеціалізованими засобами [9], [13];

- очищення вільного простору: програмна система Folder Lock забезпечує перезапис раніше видалених даних для усунення залишкових слідів на диску [9], [13];

– захист персональної інформації: забезпечення конфіденційності документів, фінансових даних та інших важливих матеріалів [9], [13];

– інтуїтивний інтерфейс: програмна система Folder Lock забезпечує зручне керування функціями без потреби у складних технічних налаштуваннях [9], [13];

– локальне використання: робота без обов’язкового підключення до хмарних сервісів, що підвищує автономність [9], [13].

Серед недоліків програмного забезпечення Folder Lock можна відзначити залежність від збереження головного пароля, оскільки його втрата може призвести до неможливості доступу до зашифрованих даних, а також обмежену функціональність у безкоштовній версії, що спонукає до придбання повної ліцензії для використання всіх можливостей програми [9], [13].

Порівняльну характеристику програмного забезпечення для запобігання несанкціонованому доступу до файлів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для запобігання несанкціонованому доступу до файлів

Критерій порівняння	Files.com [11]	Cove [12]	Folder Lock [13]
Хмарне зберігання даних	+	+	—
Резервне копіювання та відновлення	+—	+	—
Інтеграція із зовнішніми сервісами	+	+—	—
Локальний захист файлів	+—	—	+
Безпечне видалення та очищення даних	—	+—	+
Контролювання прав користувачів	+	+	+—

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для запобігання несанкціонованому доступу до файлів. Проте деякі програми для запобігання несанкціонованому доступу до файлів можуть бути досить складними, що може негативно впливати на продуктивність цифрової екосистеми організації. Реалізація постійного контролю доступу, журналювання подій, перевірки прав користувачів і шифрування даних потребує додаткових обчислювальних ресурсів. Проблемним аспектом виступає також складність налаштування та адміністрування деяких програм. Некоректна конфігурація політик доступу здатна або надмірно обмежити легітимних користувачів, або залишити критичні дані недостатньо захищеними. Необхідність регулярного оновлення програмного забезпечення з метою усунення виявлених вразливостей також може розглядатися як недолік. Крім того, впровадження та підтримка подібних систем часто потребують фінансових витрат, пов'язаних із придбанням ліцензій, навчанням персоналу та технічною підтримкою. Тому актуальною є розробка програмного забезпечення для запобігання несанкціонованому доступу до файлів [3]-[7].

При розробці програмного забезпечення для запобігання несанкціонованому доступу до файлів необхідно забезпечити такі функціональні вимоги:

- підтримка можливості встановлення параметрів захисту для текстових та табличних документів;
- підтримка можливості зняття раніше встановленого захисту за умови введення коректного пароля;
- можливість формування та редагування переліку файлів, що підлягають обробці, із додаванням окремих документів або пошуком у вибраній теці;
- підтримка вибору типу документа для забезпечення коректного застосування відповідних режимів захисту;

- можливість встановлення пароля на відкриття або редагування документа з обов'язковим підтвердженням введених даних;
- підтримка вибору режиму обмеження доступу до документа, зокрема режим перегляду без редагування, режиму внесення змін із контролем прав або повної відсутності захисту;
- забезпечення перевірки коректності введених користувачем параметрів перед початком виконання операцій;
- підтримка обробки виняткових ситуацій, пов'язаних із відсутністю файлу, недостатніми правами доступу або помилками під час взаємодії з файловою системою;
- підтримка інформування користувача про результати виконання операцій та можливі помилки в процесі роботи програми.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для запобігання несанкціонованому доступу до файлів являє собою комплекс засобів і технологій, призначених для захисту інформації від стороннього втручання та несанкціонованого перегляду, копіювання або зміни. Такі системи створюють додатковий рівень безпеки для даних, які зберігаються на комп'ютерах, серверах або знімних носіях, забезпечуючи контроль над тим, хто може отримувати доступ до файлів і які дії з ними можна виконувати. Основна мета такого програмного забезпечення полягає у гарантуванні конфіденційності, цілісності та доступності інформації, незалежно від того, чи вона передається мережею, зберігається локально, чи обробляється на віддалених серверах.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для запобігання несанкціонованому доступу до файлів. Проте деякі програми для запобігання несанкціонованому доступу до файлів можуть бути досить складними, що може негативно впливати на продуктивність цифрової екосистеми організації. Реалізація

постійного контролю доступу, журналювання подій, перевірки прав користувачів і шифрування даних потребує додаткових обчислювальних ресурсів. Проблемним аспектом виступає також складність налаштування та адміністрування деяких програм. Некоректна конфігурація політик доступу здатна або надмірно обмежити легітимних користувачів, або залишити критичні дані недостатньо захищеними. Необхідність регулярного оновлення програмного забезпечення з метою усунення виявлених вразливостей також може розглядатися як недолік. Крім того, впровадження та підтримка подібних систем часто потребують фінансових витрат, пов'язаних із придбанням ліцензій, навчанням персоналу та технічною підтримкою. Тому актуальною є розробка програмного забезпечення для запобігання несанкціонованому доступу до файлів.

Сформульовано функціональні вимоги до програмного забезпечення для запобігання несанкціонованому доступу до файлів.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для запобігання несанкціонованому доступу до файлів було використано мову програмування C# [14]-[15].

Мова програмування C# використовується для побудови масштабованих, безпечних і продуктивних програмних рішень. Вона була розроблена компанією Microsoft як складова платформи .NET і з моменту появи постійно еволюціонує, поєднуючи строгість статичної типізації з сучасними підходами до розробки [14]-[15].

Важливо відзначити, що C# підтримує мультипарадигмальність. Поряд із класичним об'єктно-орієнтованим підходом активно використовуються елементи функціонального програмування, зокрема лямбда-вирази, LINQ-запити, імутабельні структури даних. Це дозволяє реалізовувати як традиційні корпоративні системи, так і високонавантажені сервіси з асинхронною обробкою подій. Підтримка `async/await` значно спрощує створення неблокуючих операцій, що є важливим для мережевих застосунків і серверних рішень [14]-[15].

C# характеризується високим рівнем читабельності та передбачуваності синтаксису. Строга система типів, механізми інкапсуляції, спадкування й поліморфізму формують чітку структуру програмного коду. Завдяки цьому полегшується підтримка великих проєктів, що розробляються командами, та забезпечується кращий контроль якості [14]-[15].

Окремої уваги заслуговує кросплатформеність сучасної реалізації .NET. Якщо раніше C# асоціювалася переважно з операційною системою Windows, то нині застосунки можуть розгортатися на Linux і macOS без суттєвих змін у коді. Це розширює сферу застосування мови від настільних програм до хмарних сервісів і контейнеризованих рішень [14]-[15].

Мова активно використовується для створення вебзастосунків за допомогою ASP.NET, мобільних програм, хмарних сервісів, мікросервісної архітектури, а також ігор. У сфері розробки відеоігор C# стала однією з ключових мов завдяки інтеграції з рушієм Unity, що дозволяє створювати як інди-проекти, так і комерційні продукти різного масштабу [14]-[15].

Суттєвою перевагою C# є розвинена система управління пакетами та бібліотеками, що дає змогу швидко інтегрувати сторонні компоненти. Інструменти автоматизованого тестування, статичного аналізу коду та безперервної інтеграції сприяють впровадженню сучасних методологій розробки [14]-[15].

Крім технічних аспектів, слід враховувати й фактор спільноти. C# має широку міжнародну аудиторію розробників, велику кількість навчальних матеріалів, відкритих проєктів і документації. Це спрощує входження в екосистему та сприяє швидкому розв'язанню практичних задач [14]-[15].

Доцільність використання мови програмування C# для розробки програмного забезпечення, спрямованого на запобігання несанкціонованому доступу до файлів, зумовлюється поєднанням технічних, архітектурних та безпекових переваг цієї платформи. Сучасне середовище розробки на базі .NET забезпечує високий рівень інтеграції з операційною системою, що є особливо важливим у задачах контролю доступу, роботи з файловою системою, управління правами користувачів і взаємодії з механізмами безпеки на рівні ОС [14]-[15].

Однією з ключових переваг є наявність розвинених стандартних бібліотек, які надають засоби для реалізації криптографічних алгоритмів, хешування, цифрового підпису, симетричного та асиметричного шифрування. Використання вбудованих класів простору імен System.Security дозволяє реалізувати механізми захисту без необхідності залучення сторонніх компонентів, що знижує ризики сумісності та спрощує підтримку програмного продукту. Крім того, платформа .NET регулярно оновлюється, що забезпечує

актуальність криптографічних інструментів і відповідність сучасним стандартам безпеки [14]-[15].

Важливою характеристикою C# є типобезпечність і кероване середовище виконання, що зменшує ймовірність виникнення помилок, пов'язаних із некоректною роботою з пам'яттю. Автоматичне керування пам'яттю та перевірка типів під час компіляції підвищують надійність програмного забезпечення, що особливо актуально для систем безпеки, де помилки можуть призвести до критичних наслідків [14]-[15].

Суттєву роль відіграє також зручність реалізації механізмів автентифікації та авторизації. Платформа .NET підтримує інтеграцію з різними моделями безпеки, включаючи рольовий доступ, роботу з доменними обліковими записами та взаємодію з сервісами каталогів. Це спрощує створення програмних рішень, які можуть функціонувати як у локальному середовищі, так і в корпоративній інфраструктурі [14]-[15].

Окрім цього, мова програмування C# підтримує об'єктно-орієнтований підхід, що сприяє структурованості коду, модульності та повторному використанню компонентів. У системах запобігання несанкціонованому доступу це дозволяє чітко розмежувати логіку шифрування, перевірки прав доступу, журналювання подій та інтерфейс взаємодії з користувачем. Така архітектурна організація полегшує тестування, аудит коду та подальше масштабування [14]-[15].

Не менш важливою є підтримка сучасних інструментів розробки, зокрема інтегрованих середовищ, що забезпечують налагодження, аналіз коду та автоматизоване тестування. Це сприяє підвищенню якості кінцевого продукту та зменшенню кількості вразливостей [14]-[15].

Таким чином, використання мови програмування C# для створення програмного забезпечення, призначеного для захисту файлів від несанкціонованого доступу, є обґрунтованим вибором завдяки поєднанню безпечної моделі виконання, розвинених криптографічних засобів, інтеграції з

операційною системою та зручності підтримки складних механізмів контролю доступу [14]-[15].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для запобігання несанкціонованому доступу до файлів наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для запобігання несанкціонованому доступу до файлів

Критерій порівняння мов програмування	Мова програмування		
	C#	Java	Go
Інтеграція з механізмами безпеки операційної системи	+	+—	+—
Наявність вбудованих криптографічних бібліотек	+	+	+—
Зручність реалізації інтерфейсу користувача для настільних систем	+	+—	—
Швидкість розробки	+	+—	+
Збалансованість безпеки та продуктивності	+	+—	+

Отже, для реалізації програмного забезпечення для запобігання несанкціонованому доступу до файлів обрано мову програмування C#, яку можна охарактеризувати як сучасну, гнучку й технологічно зрілу мову програмування, що поєднує стабільність корпоративного інструменту з можливостями інноваційного розвитку, що робить її актуальною для широкого спектра прикладних і дослідницьких завдань, зокрема, для розробки програми для запобігання несанкціонованому доступу до файлів. Крім того, важливою перевагою є наявність розвинених стандартних бібліотек, які

надають засоби для реалізації криптографічних алгоритмів, хешування, цифрового підпису, симетричного та асиметричного шифрування.

2.2 Вибір середовища розробки для створення програмного забезпечення для запобігання несанкціонованому доступу до файлів

В якості середовища розробки для створення програмного забезпечення для запобігання несанкціонованому доступу до файлів було обрано середовище Visual Studio [16]-[17].

Середовище розробки Visual Studio надає комплексну платформу для повного життєвого циклу програмного забезпечення. Середовище поєднує редактор коду, систему збірки, засоби тестування, налагодження, профілювання та аналізу продуктивності, що дозволяє організувати розробку як цілісний, керований процес [16]-[17].

Однією з важливих характеристик є інтелектуальна система підказок IntelliSense, яка забезпечує автоматичне доповнення коду, відображення сигнатур методів і перевірку синтаксису в режимі реального часу. Це підвищує швидкість написання програм і зменшує кількість технічних помилок. Підтримується рефакторинг коду, що дозволяє безпечно змінювати структуру проекту без порушення його логіки [16]-[17].

Visual Studio має потужні інструменти діагностики. Профілювальники продуктивності, аналіз використання пам'яті та моніторинг потоків виконання допомагають оптимізувати застосунки та усувати «вузькі місця». Для складних систем це є суттєвою перевагою, оскільки дозволяє забезпечити стабільну роботу навіть за високого навантаження [16]-[17].

Важливою особливістю є підтримка багатьох мов програмування в межах одного середовища, зокрема C#, C++, Python та інших. Це створює можливість розробки багатокомпонентних рішень із різними технологічними стеком без переходу до інших інструментів. Інтеграція з платформою .NET забезпечує зручне створення настільних, веб і хмарних застосунків [16]-[17].

Середовище підтримує автоматизоване тестування, включаючи модульні, інтеграційні та навантажувальні тести. Наявність вбудованих інструментів для керування тестами сприяє впровадженню принципів test-driven development і підвищує якість кінцевого продукту. Додатково реалізовано засоби аналізу покриття коду тестами, що дозволяє оцінити рівень перевірки функціональності [16]-[17].

Visual Studio інтегрується із системами контролю версій, зокрема Git, що забезпечує ефективну командну роботу. Можливість перегляду історії змін, порівняння версій і керування гілками безпосередньо в інтерфейсі середовища спрощує організацію колективної розробки [16]-[17].

Окремої уваги заслуговує підтримка розробки хмарних рішень і контейнеризованих застосунків. Інструменти взаємодії з сервісами хмарної інфраструктури дозволяють розгортати та тестувати застосунки безпосередньо з середовища розробки. Це сприяє скороченню часу між написанням коду та його впровадженням [16]-[17].

Таким чином, Visual Studio характеризується високим рівнем функціональної насиченості, гнучкістю налаштування та орієнтацією на професійну розробку складних програмних систем. Поєднання інтелектуальних засобів редагування, розвинених інструментів налагодження та підтримки сучасних технологій робить це середовище ефективним рішенням для створення програм різного масштабу та призначення [16]-[17].

Доцільність використання Visual Studio для розробки програмного забезпечення, спрямованого на запобігання несанкціонованому доступу до файлів, зумовлюється поєднанням функціональних можливостей, інструментів контролю якості та глибокої інтеграції з платформою .NET. У межах створення систем інформаційної безпеки особливого значення набувають надійність коду, можливість детального налагодження та контроль потенційних вразливостей, що повною мірою підтримується середовищем Visual Studio [16]-[17].

Visual Studio забезпечує розширені засоби статичного аналізу коду, автоматичної перевірки типів і попередження потенційно небезпечних конструкцій ще на етапі компіляції. Це дозволяє зменшити кількість логічних помилок і підвищити стійкість програмного продукту до збоїв. Для систем, що реалізують механізми автентифікації, шифрування та контролю доступу, можливість раннього виявлення помилок є критично важливою [16]-[17].

Суттєвою перевагою Visual Studio виступає потужний інструментарій налагодження. Наявність покрокового виконання, аналізу стану змінних, моніторингу потоків виконання та винятків дозволяє детально перевіряти реалізацію механізмів безпеки. У програмному забезпеченні, пов'язаному із захистом файлів, необхідно ретельно контролювати логіку перевірки прав доступу та обробки помилкових ситуацій, і Visual Studio створює для цього сприятливі умови [16]-[17].

Важливим чинником є інтеграція з системами контролю версій, засобами модульного тестування та інструментами безперервної інтеграції. Це сприяє впровадженню сучасних практик розробки, що забезпечують стабільність і передбачуваність оновлень. Регулярне тестування компонентів шифрування та авторизації дозволяє своєчасно виявляти потенційні вразливості [16]-[17].

Середовище також підтримує зручне проєктування графічних інтерфейсів для настільних застосунків. Для систем контролю доступу до файлів важливо забезпечити інтуїтивно зрозумілий інтерфейс керування правами, журналами подій і параметрами безпеки. Інструменти візуального конструювання значно пришвидшують цей процес та зменшують кількість помилок у взаємодії користувача з програмою [16]-[17].

Крім того, Visual Studio має розвинену екосистему розширень, що дозволяє інтегрувати додаткові засоби аналізу безпеки, профілювання продуктивності та перевірки відповідності стандартам кодування. Це особливо актуально для програм, які повинні відповідати підвищеним вимогам до надійності та захищеності [16]-[17].

Таким чином, використання Visual Studio при створенні програмного забезпечення для запобігання несанкціонованому доступу до файлів є обґрунтованим завдяки поєднанню потужних засобів налагодження, контролю якості, інтеграції з платформою .NET та підтримки сучасних методологій розробки, що забезпечує високий рівень надійності та безпеки кінцевого продукту [16]-[17].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для запобігання несанкціонованому доступу до файлів наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для запобігання несанкціонованому доступу до файлів

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	MonoDevelop	SharpDevelop
Повноцінна підтримка .NET	+	+—	+—
Розширені засоби налагодження	+	+—	—
Підтримка тестування та аналізу покриття	+	+—	+—
Зручність створення інтерфейсу користувача	+	+—	+—
Стабільність та зрілість платформи	+	+—	—

Таким чином, для створення програмного забезпечення для запобігання несанкціонованому доступу до файлів обрано середовище розробки Visual Studio, що поєднує редактор коду, систему збірки, засоби тестування, налагодження, профілювання та аналізу продуктивності, що дозволяє

організувати розробку як цілісний, керований процес. Крім того, Visual Studio має розвинену екосистему розширень, що дозволяє інтегрувати додаткові засоби аналізу безпеки, профілювання продуктивності та перевірки відповідності стандартам кодування.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для запобігання несанкціонованому доступу до файлів обрано мову програмування C#, яку можна охарактеризувати як сучасну, гнучку й технологічно зрілу мову програмування, що поєднує стабільність корпоративного інструменту з можливостями інноваційного розвитку, що робить її актуальною для широкого спектра прикладних і дослідницьких завдань, зокрема, для розробки програми для запобігання несанкціонованому доступу до файлів. Крім того, важливою перевагою є наявність розвинених стандартних бібліотек, які надають засоби для реалізації криптографічних алгоритмів, хешування, цифрового підпису, симетричного та асиметричного шифрування.

Для створення програмного забезпечення для запобігання несанкціонованому доступу до файлів обрано середовище розробки Visual Studio, що поєднує редактор коду, систему збірки, засоби тестування, налагодження, профілювання та аналізу продуктивності, що дозволяє організувати розробку як цілісний, керований процес. Крім того, Visual Studio має розвинену екосистему розширень, що дозволяє інтегрувати додаткові засоби аналізу безпеки, профілювання продуктивності та перевірки відповідності стандартам кодування.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для запобігання несанкціонованому доступу до файлів

Структуру програмного забезпечення для запобігання несанкціонованому доступу до файлів наведено на рис. 3.1.



Рисунок 3.1 – Структура програмного забезпечення для запобігання несанкціонованому доступу до файлів

Структура програмного забезпечення для запобігання несанкціонованому доступу до файлів сформована за модульним принципом, що забезпечує розмежування відповідальності компонентів, підвищує надійність та спрощує супровід системи. Архітектурно така система

складається з взаємопов'язаних модулів, кожен з яких реалізує окремий функціональний рівень захисту.

Першим модулем виступає модуль ініціалізації та керування життєвим циклом застосунку. Його призначення полягає у запуску програмного середовища, завантаженні конфігураційних параметрів, перевірці цілісності системних компонентів та ініціалізації служб безпеки. У межах цього модуля реалізується створення екземплярів форм, завантаження початкових параметрів та організація взаємодії між іншими складовими. Також здійснюється налаштування середовища виконання, перевірка доступності необхідних бібліотек для роботи з офісними документами, а також обробка глобальних винятків. Він забезпечує коректний старт програми та контроль її завершення.

Другим компонентом є модуль головної форми керування захистом. Саме в межах цієї форми виконуються всі операції із встановлення параметрів захисту файлів, вибору документів, введення паролів та ініціювання процесів шифрування або зняття обмежень. Функціонально реалізується обробка подій інтерфейсу, перевірка коректності введених даних, відображення статусного рядка та формування повідомлень для користувача. Через цей модуль здійснюється виклик службових процедур, пов'язаних із безпосередньою обробкою документів.

Третім модулем є модуль довідкової форми. Його призначення полягає у наданні користувачу інформації щодо принципів роботи програми, рекомендацій із налаштування параметрів захисту та пояснення можливих повідомлень системи. У функціональному аспекті він забезпечує відображення текстових матеріалів. Наявність окремої форми дозволяє відокремити інформаційний компонент від функціонального, що підвищує зручність використання.

Четвертим складником структури є модуль обробки офісних документів. Його призначення полягає у взаємодії з файлами форматів Word та Excel, аналізі їх властивостей і застосуванні або знятті механізмів захисту.

У межах цього модуля реалізується відкриття документів, встановлення паролів на редагування чи відкриття, збереження змін та контроль помилок, пов'язаних із несумісністю форматів або пошкодженням файлів. Він виконує ключову функцію безпосереднього впливу на об'єкти захисту. Також виконується відображення поточного стану виконання операцій у статусному рядку головної форми, генерації інформаційних і попереджувальних повідомлень, а також фіксації результатів виконаних дій. Завдяки цьому забезпечується прозорість процесу встановлення або зняття захисту та можливість аналізу виконаних операцій.

П'ятим модулем є модуль перевірки та валідації параметрів безпеки. Його призначення полягає у контролі складності паролів, перевірці коректності введених даних та забезпеченні відповідності встановлених параметрів визначеним правилам. У функціональному аспекті здійснюється аналіз довжини та структури пароля, попередження про слабкі комбінації та формування повідомлень про помилки. Це сприяє підвищенню загального рівня захищеності документів.

Таким чином, структура настільного програмного продукту для запобігання несанкціонованому доступу до файлів базується на чіткому розподілі відповідальності між модулем ініціалізації, головною формою керування, довідковою формою, компонентом обробки документів, підсистемою перевірки параметрів та механізмом інформування. Така організація забезпечує логічність побудови, зручність використання та надійність реалізації функцій захисту файлів офісних застосувань.

3.2 Функціонування програмного забезпечення для запобігання несанкціонованому доступу до файлів

Функціонування програмного забезпечення для запобігання несанкціонованому доступу до файлів подамо за допомогою схеми, зображеної на рис. 3.2.

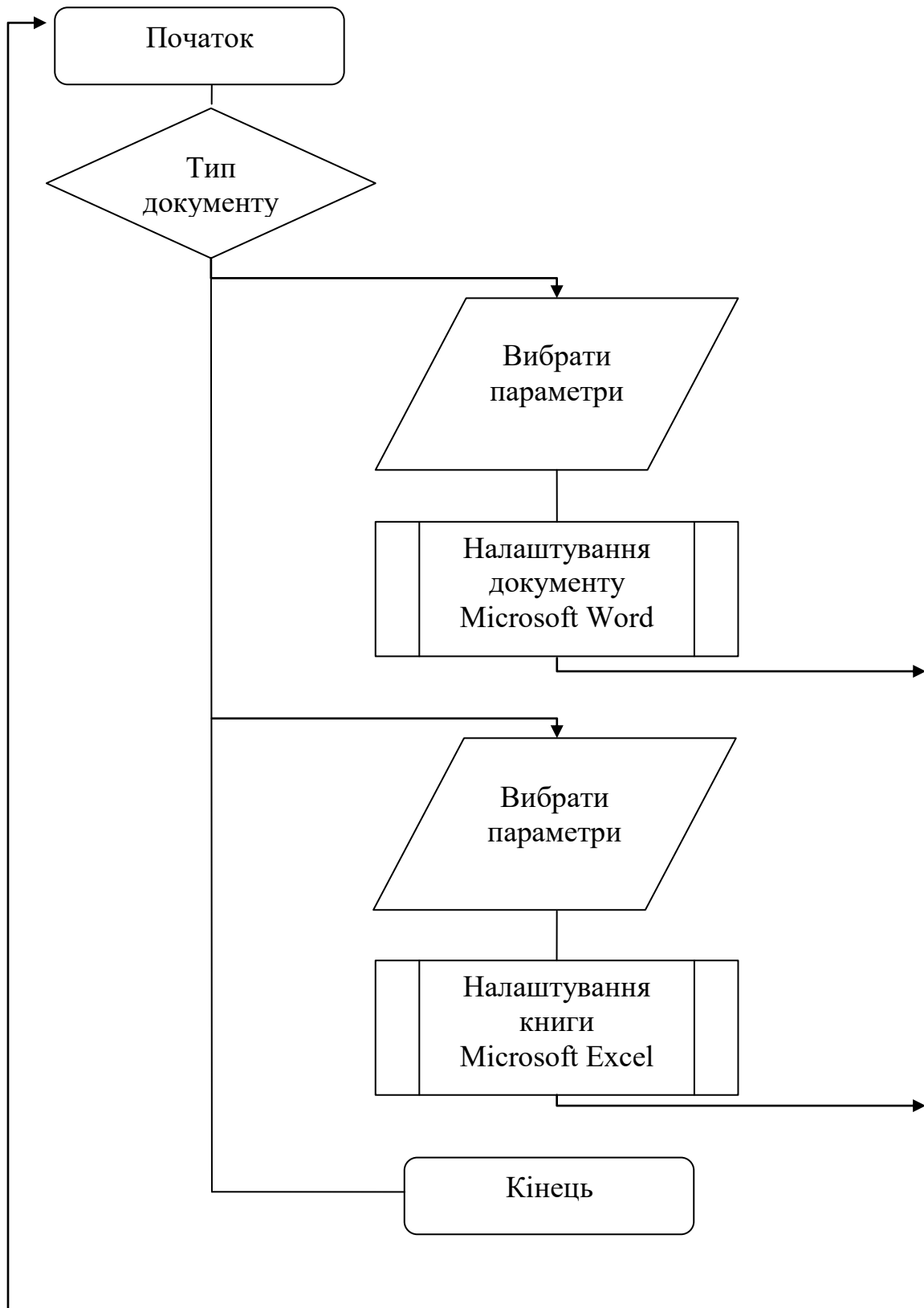


Рисунок 3.2 – Схема функціонування програмного забезпечення для запобігання несанкціонованому доступу до файлів

Функціонування настільного програмного забезпечення для запобігання несанкціонованому доступу до файлів офісних застосувань організовано за принципом вибору цільового середовища обробки документів із подальшою адаптацією внутрішніх механізмів до відповідного формату. Після запуску програми користувач визначає, з яким типом файлів буде здійснюватися робота – текстовими документами або електронними таблицями, створеними у Microsoft Word чи Microsoft Excel. Обраний варіант встановлює логіку подальших дій, оскільки для кожного формату передбачено специфічні параметри захисту та особливості застосування обмежень.

Після визначення типу документа активується відповідний режим функціонування, у межах якого стають доступними налаштування, притаманні конкретному застосуванню. Для текстових файлів можуть бути реалізовані обмеження на редагування, форматування або відкриття без пароля, тоді як для електронних таблиць передбачено захист структури книги, окремих аркушів чи комірок. Хоча користувач має можливість змінити обраний режим у процесі роботи, одночасна обробка різних типів документів не передбачена, що забезпечує цілісність виконання операцій та зменшує ризик некоректного застосування параметрів.

Внутрішня реалізація механізмів захисту базується на використанні можливостей платформи .NET Framework, що забезпечує взаємодію з компонентами офісного середовища. Для встановлення та зняття обмежень застосовуються бібліотеки Microsoft.Office.Interop.Word та Microsoft.Office.Interop.Excel, які надають програмний доступ до функціоналу відповідних застосувань. Через ці бібліотеки здійснюється відкриття документа, зміна його властивостей безпеки та збереження результату з урахуванням обраних параметрів.

Процедура встановлення захисту передбачає виклик вбудованих методів, що реалізують обмеження на рівні самого документа. Застосування стандартних функцій захисту гарантує сумісність із внутрішніми механізмами офісних програм і дозволяє використовувати перевірені алгоритми обмеження

доступу. Під час виконання операцій програма відображає інформацію про поточний стан у статусному рядку головної форми та генерує повідомлення щодо результатів виконання дій або виникнення помилок. Це забезпечує прозорість процесу та контроль з боку користувача.

У випадку зняття захисту відбувається перевірка коректності введених даних, зокрема пароля, після чого ініціюється відповідна процедура розблокування. Усі зміни фіксуються шляхом повторного збереження документа з оновленими параметрами доступу. Такий підхід дозволяє працювати без безпосереднього втручання в структуру файлу, покладаючись на стандартні механізми офісних застосувань, що підвищує надійність і стабільність функціонування.

Загалом робота програми характеризується послідовністю дій: вибір формату документа, визначення параметрів захисту, виконання операції та інформування користувача про результат. Реалізація через засоби .NET та інтеграцію з офісними бібліотеками забезпечує коректність обробки файлів і відповідність установлених обмежень внутрішнім правилам відповідних застосувань.

Отже, функціонування програмного забезпечення ґрунтується на чіткій логіці вибору режиму роботи, використанні стандартних механізмів захисту офісних документів і забезпеченні постійного зворотного зв'язку з користувачем, що дозволяє ефективно встановлювати та знімати обмеження доступу без порушення структури файлів.

3.3 Опис особливостей реалізації програмного забезпечення для запобігання несанкціонованому доступу до файлів

Реалізація програмного забезпечення для встановлення та зняття захисту з файлів офісних застосувань ґрунтується на використанні можливостей платформи .NET Framework та механізмів автоматизації компонентів Microsoft Word і Microsoft Excel. Архітектурно система

побудована як настільний застосунок із чітким розмежуванням логіки інтерфейсу та функціональних процедур обробки документів, що забезпечує керованість, стабільність і передбачуваність виконання операцій.

Однією з ключових особливостей реалізації є використання стандартних засобів міжпроцесної взаємодії з офісними застосуваннями через бібліотеки автоматизації. Відкриття документа здійснюється шляхом створення екземпляра відповідного застосування та завантаження вибраного файлу в пам'ять із перевіркою його доступності. Перед виконанням будь-яких змін виконується контроль існування файлу, перевірка прав доступу до каталогу та аналіз можливих блокувань іншими процесами. Такий підхід мінімізує ризик пошкодження документа або виникнення конфліктів під час обробки.

Алгоритм встановлення параметрів захисту передбачає послідовне виконання операцій перевірки вхідних даних, ініціалізації об'єкта документа, застосування відповідного методу обмеження доступу та збереження змін. Для текстових документів і електронних таблиць використовуються передбачені середовищем функції захисту, що дозволяють встановлювати пароль на відкриття, редагування або зміну структури. Після успішного виконання операції документ зберігається з оновленими властивостями, а програмний модуль звільняє ресурси, коректно завершуючи взаємодію з офісним процесом.

Важливою складовою реалізації є система перевірки коректності введених користувачем параметрів. Перед застосуванням захисту здійснюється контроль заповнення обов'язкових полів, аналіз довжини та допустимості символів у паролі, а також перевірка відповідності підтвердження введених значень. У разі виявлення некоректних даних виконання операції блокується, а користувач отримує відповідне повідомлення через механізм інформування головної форми. Це запобігає виникненню помилок, пов'язаних із неправильним введенням параметрів.

Окрему увагу приділено механізмам обробки винятків. Усі критичні ділянки коду, пов'язані з відкриттям файлів, застосуванням захисту та збереженням результатів, реалізовані з використанням конструкцій перехоплення помилок. Передбачено обробку ситуацій відсутності файлу, недостатності прав доступу, неправильного пароля під час зняття захисту та внутрішніх помилок офісних бібліотек. У разі виникнення виняткової ситуації програма не припиняє роботу аварійно, а формує повідомлення користувачу та повертається до стабільного стану. Такий підхід забезпечує підвищення надійності та стійкості програмного продукту.

Механізми забезпечення безпеки також охоплюють захист службових параметрів програми. Конфігураційні значення зберігаються з урахуванням обмеження доступу до них, а чутливі дані не записуються у відкритому вигляді. У процесі роботи тимчасові об'єкти та змінні, що містять конфіденційну інформацію, звільняються після завершення операції, що зменшує ризик несанкціонованого використання даних у межах системи.

Забезпечення цілісності виконання операцій реалізовано шляхом логічної послідовності дій та перевірки кожного етапу перед переходом до наступного. Якщо одна з операцій завершується з помилкою, подальші кроки не виконуються. Після кожної завершеної процедури оновлюється статусний рядок головної форми, що відображає поточний стан процесу та інформує користувача про результат.

Робота з файловою системою здійснюється через стандартні засоби .NET, що дозволяють перевіряти шляхи до файлів, визначати їх розширення, аналізувати атрибути та виконувати контроль доступності. Перед збереженням зміненого документа перевіряється можливість перезапису файлу, а в разі потреби реалізується механізм підтвердження заміни або збереження копії. Це забезпечує додатковий рівень безпеки та зменшує ймовірність втрати даних.

Таким чином, особливості реалізації програмного забезпечення полягають у поєднанні стандартних механізмів платформи .NET, використанні вбудованих засобів захисту офісних документів, впровадженні системи

перевірки даних та комплексній обробці виняткових ситуацій. Реалізовані підходи забезпечують стабільність роботи, коректність застосування параметрів захисту та підвищений рівень надійності під час обробки файлів.

3.4 Проєктування інтерфейсу програмного забезпечення для запобігання несанкціонованому доступу до файлів

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для запобігання несанкціонованому доступу до файлів виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для запобігання несанкціонованому доступу до файлів враховані функціональні вимоги до програми:

- підтримка можливості встановлення параметрів захисту для текстових та табличних документів;
- підтримка можливості зняття раніше встановленого захисту за умови введення коректного пароля;
- можливість формування та редагування переліку файлів, що підлягають обробці, із додаванням окремих документів або пошуком у вибраній теці;
- підтримка вибору типу документа для забезпечення коректного застосування відповідних режимів захисту;
- можливість встановлення пароля на відкриття або редагування документа з обов'язковим підтвердженням введених даних;
- підтримка вибору режиму обмеження доступу до документа, зокрема режиму перегляду без редагування, режиму внесення змін із контролем прав або повної відсутності захисту;
- забезпечення перевірки коректності введених користувачем параметрів перед початком виконання операцій;

– підтримка обробки виняткових ситуацій, пов'язаних із відсутністю файлу, недостатніми правами доступу або помилками під час взаємодії з файловою системою;

– підтримка інформування користувача про результати виконання операцій та можливі помилки в процесі роботи програми.

Інтерфейс головної форми програмного забезпечення для запобігання несанкціонованому доступу до файлів наведено на рис. 3.3.

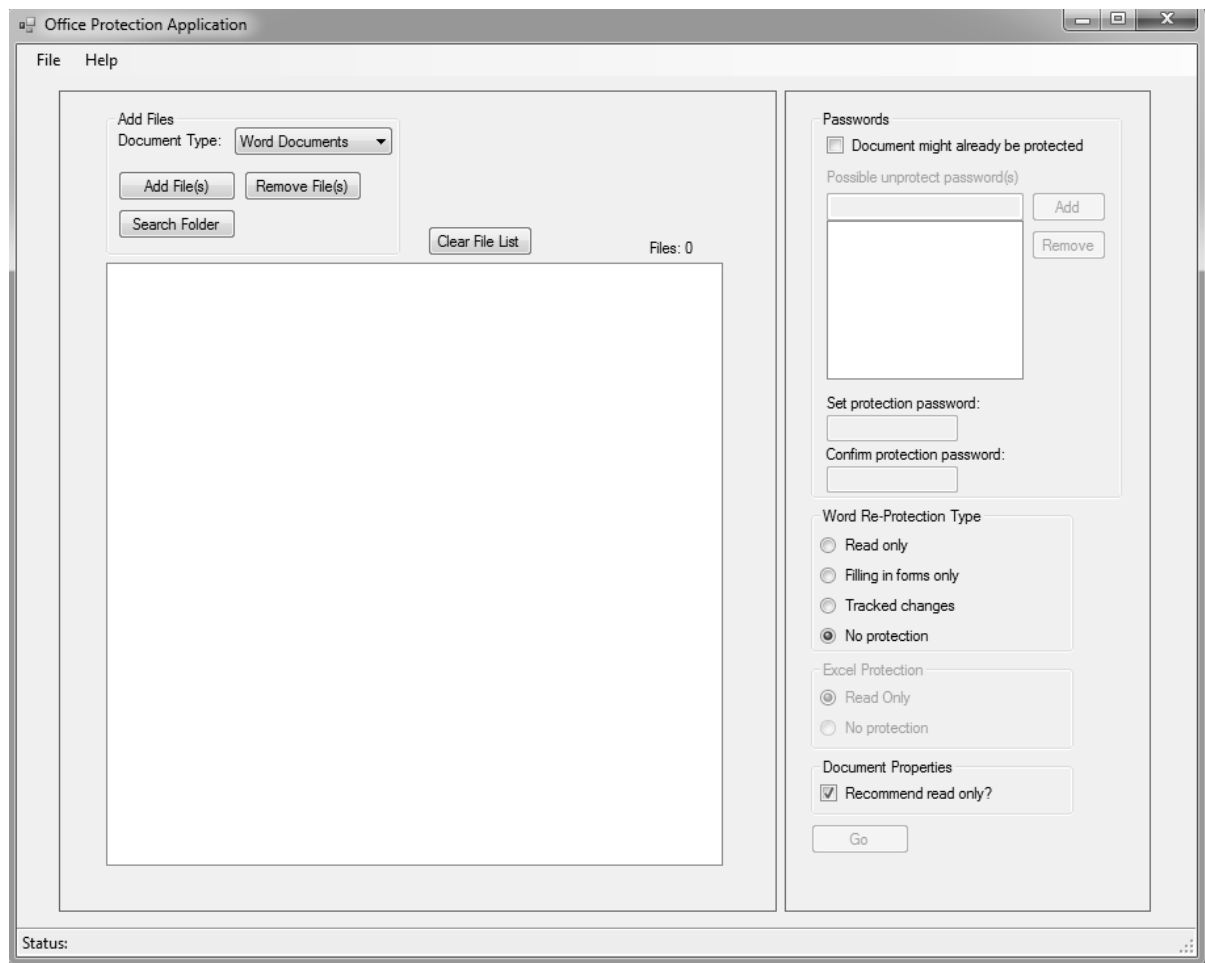


Рисунок 3.3 – Інтерфейс головної форми програмного забезпечення для запобігання несанкціонованому доступу до файлів

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для запобігання несанкціонованому доступу до файлів.

Запропоновано структуру програмного забезпечення для запобігання несанкціонованому доступу до файлів, яка сформована за модульним принципом, що забезпечує розмежування відповідальності між модулем ініціалізації, головною формою керування, довідковою формою, компонентом обробки документів, підсистемою перевірки параметрів та механізмом інформування. Така організація забезпечує логічність побудови, зручність використання та надійність реалізації функцій захисту файлів офісних застосунків, що підвищує надійність та спрощує супровід програмної системи.

Описано функціонування програмного забезпечення для запобігання несанкціонованому доступу до файлів, що ґрунтується на чіткій логіці вибору режиму роботи, використанні стандартних механізмів захисту офісних документів і забезпеченні постійного зворотного зв'язку з користувачем, що дозволяє ефективно встановлювати та знімати обмеження доступу без порушення структури файлів.

Описано особливості реалізації програмного забезпечення для запобігання несанкціонованому доступу до файлів, що полягають у поєднанні стандартних механізмів платформи .NET, використанні вбудованих засобів захисту офісних документів, впровадженні системи перевірки даних та комплексній обробці виняткових ситуацій. Реалізовані підходи забезпечують стабільність роботи, коректність застосування параметрів захисту та підвищений рівень надійності під час обробки файлів

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для запобігання несанкціонованому доступу до файлів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для запобігання несанкціонованому доступу до файлів. Програма написана на мові С#, що поєднує стабільність корпоративного інструменту з можливостями інноваційного розвитку, що робить її актуальною для широкого спектра прикладних і дослідницьких завдань. В якості середовища розробки для створення програмного забезпечення для запобігання несанкціонованому доступу до файлів обрано Visual Studio, що поєднує редактор коду, систему збірки, засоби тестування, налагодження, профілювання та аналізу продуктивності, що дозволяє організувати розробку як цілісний, керований процес.

Програмне забезпечення для запобігання несанкціонованому доступу до файлів забезпечує виконання таких функцій:

- підтримка можливості встановлення параметрів захисту для текстових та табличних документів;
- підтримка можливості зняття раніше встановленого захисту за умови введення коректного пароля;
- можливість формування та редагування переліку файлів, що підлягають обробці, із додаванням окремих документів або пошуком у вибраній теці;
- підтримка вибору типу документа для забезпечення коректного застосування відповідних режимів захисту;
- можливість встановлення пароля на відкриття або редагування документа з обов'язковим підтвердженням введених даних;
- підтримка вибору режиму обмеження доступу до документа, зокрема режиму перегляду без редагування, режиму внесення змін із контролем прав або повної відсутності захисту;

- забезпечення перевірки коректності введених користувачем параметрів перед початком виконання операцій;
- підтримка обробки виняткових ситуацій, пов'язаних із відсутністю файлу, недостатніми правами доступу або помилками під час взаємодії з файловою системою;
- підтримка інформування користувача про результати виконання операцій та можливі помилки в процесі роботи програми.

Для роботи програмного забезпечення для запобігання несанкціонованому доступу до файлів потрібні такі технічні та програмні ресурси процесор із тактовою частотою не нижче 1,8–2,0 ГГц. Для комфортної роботи бажано застосовувати багатоядерний процесор (2 ядра і більше), що забезпечить швидке виконання операцій обробки документів. Обсяг оперативної пам'яті повинен становити не менше 4 ГБ, при цьому рекомендованим є 8 ГБ для забезпечення стабільної роботи під час одночасного запуску офісних програм. Вільний простір на жорсткому диску або SSD має бути не менше 500 МБ для встановлення програми та збереження службових файлів, без урахування розміру оброблюваних документів.

Програмне забезпечення повинно включати операційну систему сімейства Microsoft Windows версії не нижче Windows 10, оскільки застосунок орієнтований на використання стандартних механізмів платформи Windows для роботи з файловою системою та правами доступу. Для функціонування програми необхідна встановлена платформа .NET Framework версії 4.7.2 або новішої (або відповідне середовище .NET Runtime, якщо застосовано сучасну реалізацію).

Оскільки програмний продукт взаємодіє з офісними документами, на комп'ютері повинні бути встановлені офісні застосування, зокрема Microsoft Word та Microsoft Excel версії не нижче 2016 року випуску. Їх наявність забезпечує коректну роботу механізмів автоматизації та встановлення параметрів захисту документів.

Функціональні характеристики розробленого програмного забезпечення для запобігання несанкціонованому доступу до файлів наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для запобігання несанкціонованому доступу до файлів наведено у додатку Б.

4.2 Характеристики програми для запобігання несанкціонованому доступу до файлів

Програмне забезпечення для запобігання несанкціонованому доступу до файлів характеризується функціональною спрямованістю на забезпечення конфіденційності документів та обмеження доступу до їх вмісту. Його основним призначенням є реалізація механізмів встановлення та зняття захисту з файлів офісного формату із використанням стандартних засобів операційної системи та інтегрованих можливостей прикладних програм. У процесі роботи забезпечується контроль доступу до обраних документів шляхом застосування параметрів обмеження редагування або відкриття.

Архітектура програмного продукту побудована з урахуванням розмежування інтерфейсної та функціональної складових. Логіка обробки документів відокремлюється від компонентів взаємодії з користувачем, що підвищує зрозумілість структури та спрощує супровід. Передбачено використання механізмів платформи .NET Framework для організації доступу до файлової системи, керування ресурсами та забезпечення стабільності виконання операцій.

Характерною ознакою програмного забезпечення для запобігання несанкціонованому доступу до файлів є наявність вбудованої системи перевірки коректності введених даних. Перед виконанням операцій здійснюється аналіз параметрів, що вводяться користувачем, та перевіряється можливість їх застосування до конкретного документа. Це дозволяє запобігти

виникненню помилок, пов'язаних із некоректними значеннями або невідповідністю формату файлу.

Особлива увага приділяється обробці виняткових ситуацій. Передбачено перехоплення помилок, що можуть виникати під час відкриття, модифікації або збереження документів, а також у процесі взаємодії з зовнішніми програмами. У разі виникнення збоїв забезпечується коректне завершення поточної операції без порушення цілісності даних та без аварійного припинення роботи застосунку.

Програмне забезпечення для запобігання несанкціонованому доступу до файлів характеризується сумісністю з поширеними офісними застосуваннями, зокрема з Microsoft Word та Microsoft Excel, що дозволяє використовувати вбудовані механізми захисту документів без зміни їх внутрішньої структури. Взаємодія з цими програмами здійснюється через стандартні засоби автоматизації, що забезпечує коректність обробки файлів та збереження їх функціональних властивостей.

Додатковою характеристикою є орієнтація на безпечну роботу з файловою системою. Під час виконання операцій здійснюється перевірка доступності файлів, контролюється можливість їх зміни та забезпечується правильне звільнення ресурсів після завершення обробки. Це сприяє підвищенню надійності та зменшує ризик втрати або пошкодження інформації. Отже, ПЗ відзначається функціональною цілісністю, орієнтацією на забезпечення конфіденційності даних, стабільністю виконання операцій та інтеграцією з поширеними офісними середовищами, що у сукупності формує комплексний підхід до захисту файлів від несанкціонованого доступу.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для запобігання несанкціонованому доступу до файлів необхідно запустити виконуваний файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для запобігання несанкціонованому доступу до файлів є шлях до обраного документа, параметри встановлення або зняття захисту, а також значення пароля чи іншого механізму обмеження доступу, що вводиться користувачем. До вхідної інформації також належать команди, ініційовані через інтерфейс користувача, зокрема вибір режиму роботи, підтвердження виконання операції та зазначення способу збереження зміненого файлу.

Вихідними даними програмного забезпечення для запобігання несанкціонованому доступу до файлів є змінений документ із застосованими параметрами захисту або файл, з якого обмеження доступу було успішно знято. Результатом роботи також є інформація про стан виконаної операції, що відображається в інтерфейсі користувача.

4.3.3 Повідомлення

Користувач програмного забезпечення для запобігання несанкціонованому доступу до файлів може отримати такі повідомлення: повідомлення про успішне встановлення або зняття захисту; попередження про некоректність введених параметрів; сповіщення про відсутність доступу до файлу або його використання іншою програмою; повідомлення про помилку під час відкриття чи збереження документа; інформування про необхідність перевірки правильності введеного пароля.

4.4 Виконання програмного забезпечення для запобігання несанкціонованому доступу до файлів

Після запуску програмного забезпечення для запобігання несанкціонованому доступу до файлів користувачу відображається головне

вікно (рис. 4.1), у якому зосереджено всі основні елементи керування. У верхній частині інтерфейсу розташовано меню з двома розділами, один із яких призначено для завершення роботи або повторного запуску застосунку, а інший забезпечує отримання довідкової інформації про програмний продукт.

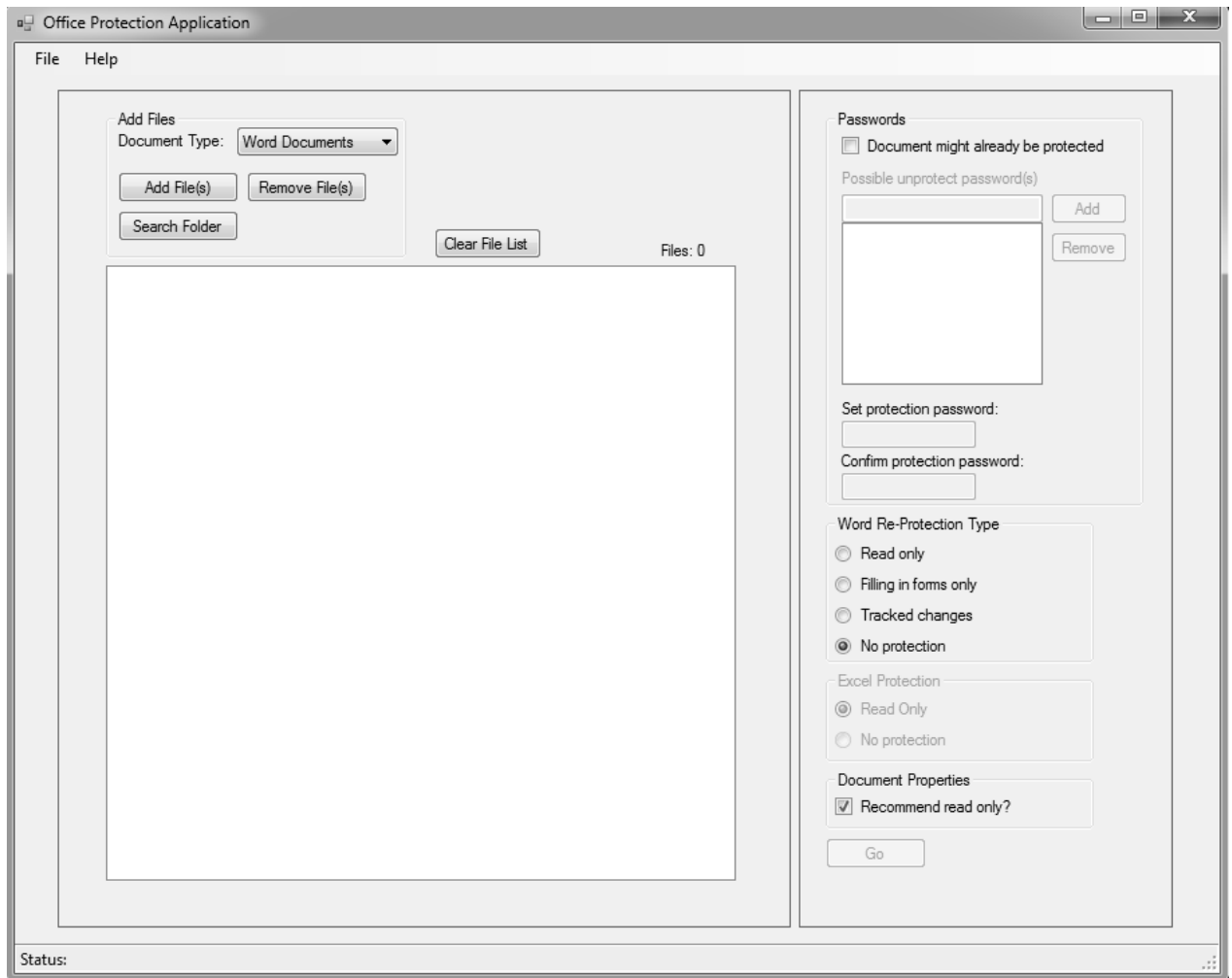


Рисунок 4.1 – Інтерфейс головної форми програмного забезпечення для запобігання несанкціонованому доступу до файлів

Інтерфейс програми умовно поділено на дві функціональні області. В одній із них реалізовано механізм формування переліку документів, що підлягають обробці. У центральній частині цієї області відображається список вибраних файлів офісних форматів. Додавання нових позицій здійснюється або шляхом безпосереднього вибору окремих файлів, або через автоматизований пошук документів відповідного типу в зазначеному каталозі.

Передбачено можливість видалення окремих елементів зі сформованого переліку після їх попереднього виділення, а також повне очищення списку. Тип документів визначається через спеціальний елемент вибору, що дозволяє встановити формат оброблюваних файлів.

Інша функціональна область призначена для задання параметрів захисту. У ній реалізовано засоби керування паролями, вибору режимів обмеження доступу та встановлення додаткових властивостей документа. Якщо передбачається, що файл уже має встановлений пароль, активується відповідний режим, після чого стає доступним введення можливих варіантів для зняття обмеження. Додані значення відображаються у спеціальному полі та можуть бути видалені за потреби. Окремо передбачено введення нового пароля з обов'язковим підтвердженням для запобігання помилкам під час встановлення захисту.

Для текстових документів доступні режими обмеження, які визначають характер дозволених дій із вмістом, зокрема перегляд без редагування, внесення змін із фіксацією виправлень або роботу лише з визначеними формами. Для електронних таблиць передбачено спрощений вибір між режимом перегляду та відсутністю обмежень. Активація відповідних параметрів відбувається залежно від попередньо обраного типу файлів, що забезпечує логічну узгодженість інтерфейсу та запобігає помилковому застосуванню несумісних налаштувань.

У разі встановлення пароля доступ до документа надалі може бути отриманий лише після введення коректного значення, що гарантує обмеження несанкціонованих дій.

За допомогою довідкової форми (рис. 4.2) користувач може отримати основну інформацію про програму.

Після визначення всіх необхідних параметрів ініціюється виконання процедури обробки (рис. 4.3), у результаті чого відбувається застосування вибраних налаштувань до зазначених файлів.

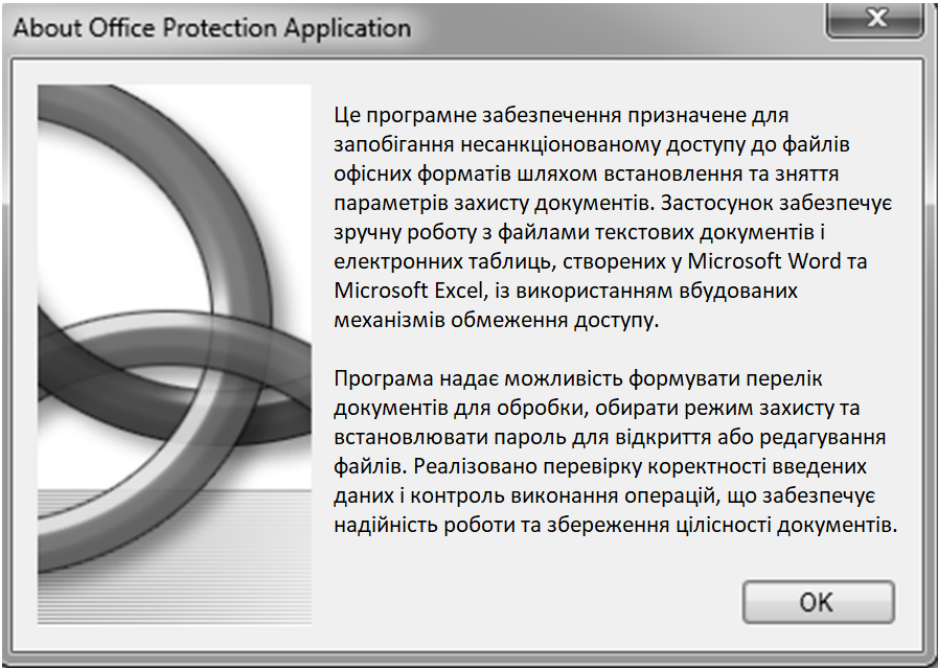


Рисунок 4.2 – Довідкова форма

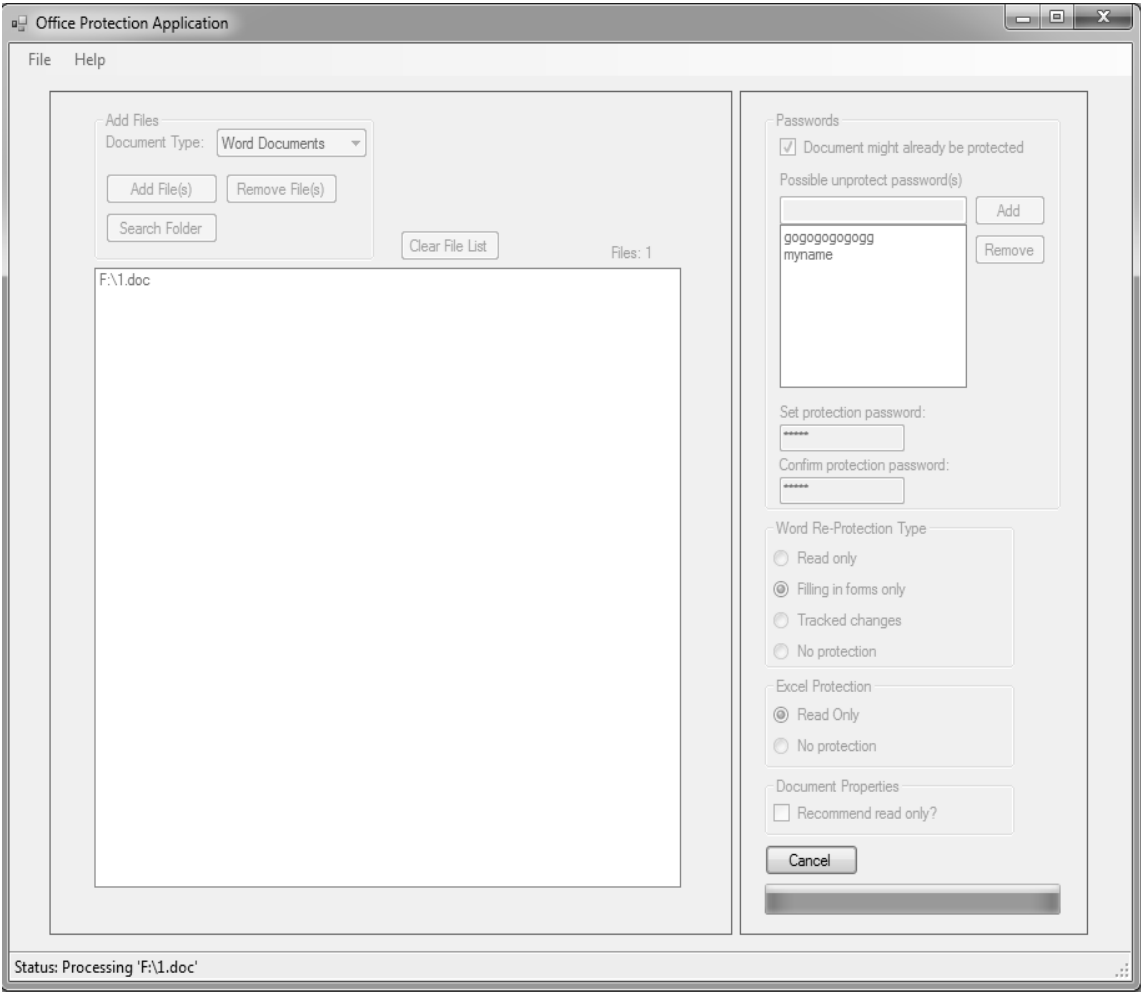


Рисунок 4.3 – Встановлення параметрів захисту файлу

4.5 Тестування програмного забезпечення для запобігання несанкціонованому доступу до файлів

Виконано тестування програмного забезпечення для запобігання несанкціонованому доступу до файлів.

Виявлено, що програма для запобігання несанкціонованому доступу до файлів функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу запобігання несанкціонованому доступу до файлів.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою процесу запобігання несанкціонованому доступу до файлів.

4.6 Висновки за розділом 4

Описано програмне забезпечення для запобігання несанкціонованому доступу до файлів. Виконано тестування розробленого програмного забезпечення для запобігання несанкціонованому доступу до файлів. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу запобігання несанкціонованому доступу до файлів.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для запобігання несанкціонованому доступу до файлів.

Визначено, що програмне забезпечення для запобігання несанкціонованому доступу до файлів являє собою комплекс засобів і технологій, призначених для захисту інформації від стороннього втручання та несанкціонованого перегляду, копіювання або зміни. Такі системи створюють додатковий рівень безпеки для даних, які зберігаються на комп'ютерах, серверах або знімних носіях, забезпечуючи контроль над тим, хто може отримувати доступ до файлів і які дії з ними можна виконувати. Основна мета такого програмного забезпечення полягає у гарантуванні конфіденційності, цілісності та доступності інформації, незалежно від того, чи вона передається мережею, зберігається локально, чи обробляється на віддалених серверах.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для запобігання несанкціонованому доступу до файлів. Проте деякі програми для запобігання несанкціонованому доступу до файлів можуть бути досить складними, що може негативно впливати на продуктивність цифрової екосистеми організації. Реалізація постійного контролю доступу, журналювання подій, перевірки прав користувачів і шифрування даних потребує додаткових обчислювальних ресурсів. Проблемним аспектом виступає також складність налаштування та адміністрування деяких програм. Некоректна конфігурація політик доступу здатна або надмірно обмежити легітимних користувачів, або залишити критичні дані недостатньо захищеними. Необхідність регулярного оновлення програмного забезпечення з метою усунення виявлених вразливостей також може розглядатися як недолік. Крім того, впровадження та підтримка подібних систем часто потребують фінансових витрат, пов'язаних із придбанням ліцензій, навчанням персоналу та технічною підтримкою. Тому

актуальною є розробка програмного забезпечення для запобігання несанкціонованому доступу до файлів.

Сформульовано функціональні вимоги до програмного забезпечення для запобігання несанкціонованому доступу до файлів.

Для реалізації програмного забезпечення для запобігання несанкціонованому доступу до файлів обрано мову програмування C#, яку можна охарактеризувати як сучасну, гнучку й технологічно зрілу мову програмування, що поєднує стабільність корпоративного інструменту з можливостями інноваційного розвитку, що робить її актуальною для широкого спектра прикладних і дослідницьких завдань, зокрема, для розробки програми для запобігання несанкціонованому доступу до файлів. Крім того, важливою перевагою є наявність розвинених стандартних бібліотек, які надають засоби для реалізації криптографічних алгоритмів, хешування, цифрового підпису, симетричного та асиметричного шифрування.

Для створення програмного забезпечення для запобігання несанкціонованому доступу до файлів обрано середовище розробки Visual Studio, що поєднує редактор коду, систему збірки, засоби тестування, налагодження, профілювання та аналізу продуктивності, що дозволяє організувати розробку як цілісний, керований процес. Крім того, Visual Studio має розвинену екосистему розширень, що дозволяє інтегрувати додаткові засоби аналізу безпеки, профілювання продуктивності та перевірки відповідності стандартам кодування.

Розроблено програмне забезпечення для запобігання несанкціонованому доступу до файлів.

Запропоновано структуру програмного забезпечення для запобігання несанкціонованому доступу до файлів, яка сформована за модульним принципом, що забезпечує розмежування відповідальності між модулем ініціалізації, головною формою керування, довідковою формою, компонентом обробки документів, підсистемою перевірки параметрів та механізмом інформування. Така організація забезпечує логічність побудови, зручність

використання та надійність реалізації функцій захисту файлів офісних застосувань, що підвищує надійність та спрощує супровід програмної системи.

Описано функціонування програмного забезпечення для запобігання несанкціонованому доступу до файлів, що ґрунтується на чіткій логіці вибору режиму роботи, використанні стандартних механізмів захисту офісних документів і забезпеченні постійного зворотного зв'язку з користувачем, що дозволяє ефективно встановлювати та знімати обмеження доступу без порушення структури файлів

Описано особливості реалізації програмного забезпечення для запобігання несанкціонованому доступу до файлів, що полягають у поєднанні стандартних механізмів платформи .NET, використанні вбудованих засобів захисту офісних документів, впровадженні системи перевірки даних та комплексній обробці виняткових ситуацій. Реалізовані підходи забезпечують стабільність роботи, коректність застосування параметрів захисту та підвищений рівень надійності під час обробки файлів

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для запобігання несанкціонованому доступу до файлів.

Виконано тестування розробленого програмного забезпечення для запобігання несанкціонованому доступу до файлів. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу запобігання несанкціонованому доступу до файлів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is Software Security Definition and Best Practice. URL: <https://www.geeksforgeeks.org/ethical-hacking/what-is-software-security-definition-and-best-practice/> (date of access: 01.05.2026).
2. What Is Unauthorized Access? Top 8 Practices for Detecting and Responding to It. URL: <https://www.syteca.com/en/blog/detecting-and-responding-to-unauthorized-access> (date of access: 01.05.2026).
3. Software Security: Definitions and Guidance. URL: <https://www.crowdstrike.com/en-us/cybersecurity-101/cybersecurity/software-security/> (date of access: 01.05.2026).
4. How to prevent unauthorized computer access. URL: <https://www.computerhope.com/issues/ch000464.htm> (date of access: 01.05.2026).
5. How to Secure Your Computer from Unauthorized Access. URL: <https://www.airdroid.com/uem/how-to-secure-your-computer-from-unauthorized-access/> (date of access: 01.05.2026).
6. Best Practices to Prevent Unauthorized Access and Safeguard Your Data. URL: <https://fidelissecurity.com/cybersecurity-101/data-protection/prevent-unauthorized-access/> (date of access: 01.05.2026).
7. How to Protect Your Data from Unauthorized Access. URL: <https://cypressdatadefense.com/blog/unauthorized-data-access/> (date of access: 01.05.2026).
8. The Best Data Protection Software. URL: <https://www.comparitech.com/data-privacy-management/best-data-protection-software/> (date of access: 01.05.2026).
9. Best Data Protection Software to Keep Your Files Safe. URL: <https://windowsreport.com/data-protection-software/> (date of access: 01.05.2026).
10. The Best Personal Computer Security Software: The Ultimate Guide to Protecting Your Device. URL: <https://www.acronis.com/en/blog/posts/best-personal-computer-security-software/> (date of access: 01.05.2026).

11. Files.com. URL: <https://www.files.com/apps/desktop-app> (date of access: 01.05.2026).
12. N-able Cove Data Protection. URL: <https://www.n-able.com/products/cove-data-protection/backup> (date of access: 01.05.2026).
13. Folder Lock. URL: <https://www.newsoftwares.net/folderlock/> (date of access: 01.05.2026).
14. C# Tutorial (C Sharp). URL: <https://www.tpointtech.com/c-sharp-tutorial> (date of access: 01.05.2026).
15. Learn .NET. URL: <https://dotnet.microsoft.com/en-us/learn> (date of access: 01.05.2026).
16. Navigating the Depths of Visual Studio Documentation: Your Guide to Development Resources. URL: <https://www.oreateai.com/blog/navigating-the-depths-of-visual-studio-documentation-your-guide-to-development-resources/49afbc783eaab679872b0d6294165342> (date of access: 01.05.2026).
17. Visual Studio documentation. URL: <https://learn.microsoft.com/en-us/visualstudio/windows/?view=visualstudio> (date of access: 01.05.2026).

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для запобігання несанкціонованому доступу до файлів.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для запобігання несанкціонованому доступу до файлів», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 7 квітня 2026 р.

A.2 Призначення розробки

Програмний продукт призначений для захисту файлів.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для запобігання несанкціонованому доступу до файлів забезпечує виконання таких функцій:

- підтримка можливості встановлення параметрів захисту для текстових та табличних документів;
- підтримка можливості зняття раніше встановленого захисту за умови введення коректного пароля;
- можливість формування та редагування переліку файлів, що підлягають обробці, із додаванням окремих документів або пошуком у вибраній теці;
- підтримка вибору типу документа для забезпечення коректного застосування відповідних режимів захисту;

- можливість встановлення пароля на відкриття або редагування документа з обов'язковим підтвердженням введених даних;
- підтримка вибору режиму обмеження доступу до документа, зокрема режиму перегляду без редагування, режиму внесення змін із контролем прав або повної відсутності захисту;
- забезпечення перевірки коректності введених користувачем параметрів перед початком виконання операцій;
- підтримка обробки виняткових ситуацій, пов'язаних із відсутністю файлу, недостатніми правами доступу або помилками під час взаємодії з файловою системою;
- підтримка інформування користувача про результати виконання операцій та можливі помилки в процесі роботи програми.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для запобігання несанкціонованому доступу до файлів повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

A.3.3 Вимоги до надійності

Програмне забезпечення для запобігання несанкціонованому доступу до файлів повинно забезпечити надійне функціонування.

A.3.4 Умови експлуатації

Для експлуатації програми для запобігання несанкціонованому доступу до файлів необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення для запобігання несанкціонованому доступу до файлів потрібні такі технічні та програмні ресурси процесор із тактовою частотою не нижче 1,8-2,0 ГГц. Для комфортної роботи бажано застосовувати багатоядерний процесор (2 ядра і більше), що забезпечить швидке виконання операцій обробки документів. Обсяг оперативної пам'яті повинен становити не менше 4 ГБ, при цьому рекомендованим є 8 ГБ для забезпечення стабільної роботи під час одночасного запуску офісних програм. Вільний простір на жорсткому диску або SSD має бути не менше 500 МБ для встановлення програми та збереження службових файлів, без урахування розміру оброблюваних документів.

Програмне забезпечення повинно включати операційну систему сімейства Microsoft Windows версії не нижче Windows 10, оскільки застосунок орієнтований на використання стандартних механізмів платформи Windows для роботи з файловою системою та правами доступу. Для функціонування програми необхідна встановлена платформа .NET Framework версії 4.7.2 або новішої (або відповідне середовище .NET Runtime, якщо застосовано сучасну реалізацію).

Оскільки програмний продукт взаємодіє з офісними документами, на комп'ютері повинні бути встановлені офісні застосування, зокрема Microsoft Word та Microsoft Excel версії не нижче 2016 року випуску.

А.3.6 Вимоги до маркування і пакування

Програма для запобігання несанкціонованому доступу до файлів може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для запобігання несанкціонованому доступу до файлів».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для запобігання несанкціонованому доступу до файлів є шлях до обраного документа, параметри встановлення або зняття захисту, а також значення пароля чи іншого механізму обмеження доступу, що вводиться користувачем. До вхідної інформації також належать команди, ініційовані через інтерфейс користувача, зокрема вибір режиму роботи, підтвердження виконання операції та зазначення способу збереження зміненого файлу.

Вихідними даними програмного забезпечення для запобігання несанкціонованому доступу до файлів є змінений документ із застосованими параметрами захисту або файл, з якого обмеження доступу було успішно знято. Результатом роботи також є інформація про стан виконаної операції, що відображається в інтерфейсі користувача.

ДОДАТОК Б
Фрагмент тексту програми

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Windows.Forms;
using Microsoft.Office.Interop.Word;
using Microsoft.Office.Interop.Excel;

namespace PrgrmPrtct
{
    private void DdFl()
    {
        OpenFileDialog dlg = new OpenFileDialog();
        dlg.Multiselect = true;
        if (dlg.ShowDialog() == DialogResult.OK)
        {
            foreach (var fl in dlg.FileNames)
            {
                lstfls.Items.Add(fl);
            }
        }
    }

    private void SrchFldr()
    {
        FolderBrowserDialog dlg = new FolderBrowserDialog();
        if (dlg.ShowDialog() == DialogResult.OK)
        {
            string[] fls =
Directory.GetFiles(dlg.SelectedPath);
            foreach (var fl in fls)
            {
                lstfls.Items.Add(fl);
            }
        }
    }

    private void RmvFl()
    {
        if (lstfls.SelectedItem != null)
        {
            lstfls.Items.Remove(lstfls.SelectedItem);
        }
    }

    private void ClrFls()
    {
        lstfls.Items.Clear();
    }
}

```

```

    }

    private void StrtPrcc()
    {
        List<string> flslst = new List<string>();
        foreach (var itm in lstfls.Items)
        {
            flslst.Add(itm.ToString());
        }

        string pss = txtpsswrld.Text;
        string cnf = txtcnfrm.Text;
        string npr = txtnprtc.Text;

        bool xst = chkxst.Checked;

        string typ = cmbtyp.SelectedItem.ToString();
        string md = "npr";
        if (rdbwrld.Checked) md = "rd";
        if (rdbxcl.Checked) md = "md";

        lgcmngr.PrccFls(flslst, pss, cnf, npr, xst, typ, md);

        MessageBox.Show("dne");
    }
}

public class LgclMngr
{
    public void PrccFls(List<string> flslst, string pss,
string cnf, string npr, bool xst, string typ, string md)
    {
        if (flslst == null) return;
        if (flslst.Count == 0) return;

        foreach (var fl in flslst)
        {
            if (!File.Exists(fl)) continue;

            if (typ == "Wrd")
            {
                WrkWrd(fl, pss, cnf, npr, xst, md);
            }
            else
            {
                WrkXcl(fl, pss, cnf, npr, xst, md);
            }
        }
    }
}

```

```

    }

    private void WrkWrd(string fl, string pss, string cnf,
string npr, bool xst, string md)
    {
        Application wrdpp = new Application();
        Document dcm = null;

        try
        {
            dcm = wrdpp.Documents.Open(fl);

            if (xst && !string.IsNullOrEmpty(npr))
            {
                try
                {
                    dcm.Unprotect(npr);
                }
                catch
                {
                }
            }

            if (!string.IsNullOrEmpty(pss) && pss == cnf)
            {
                if (md == "rd")
                {
                    dcm.Protect(WdProtectionType.wdAllowOnlyReading, false, pss);
                }
                else if (md == "md")
                {
                    dcm.Protect(WdProtectionType.wdAllowOnlyFormFields, false, pss);
                }
            }

            dcm.Save();
        }
        catch
        {
        }
        finally
        {
            if (dcm != null)
            {
                dcm.Close();
            }
        }
    }

```

```

        wrdpp.Quit();
    }
}

private void WrkXcl(string fl, string pss, string cnf,
string npr, bool xst, string md)
{
    Microsoft.Office.Interop.Excel.Application xclpp =
new Microsoft.Office.Interop.Excel.Application();
    Workbook wrkbk = null;

    try
    {
        wrkbk = xclpp.Workbooks.Open(fl);

        if (xst && !string.IsNullOrEmpty(npr))
        {
            try
            {
                wrkbk.Unprotect(npr);
            }
            catch
            {
            }
        }

        if (!string.IsNullOrEmpty(pss) && pss == cnf)
        {
            if (md == "rd")
            {
                wrkbk.Protect(pss, true, true);
            }
        }

        wrkbk.Save();
    }
    catch
    {
    }
    finally
    {
        if (wrkbk != null)
        {
            wrkbk.Close();
        }
        xclpp.Quit();
    }
}

```

```

    }
}

public void VldPrmtrs(List<string> flslst, string pss, string
cnf, string typ)
{
    if (flslst == null)
        throw new Exception("flslst null");

    if (flslst.Count == 0)
        throw new Exception("n fls");

    foreach (var fl in flslst)
    {
        if (string.IsNullOrEmpty(fl))
            throw new Exception("nvl fl");

        if (!File.Exists(fl))
            throw new Exception("fl nt fnd");

        FileInfo fnf = new FileInfo(fl);
        if (fnf.Length == 0)
            throw new Exception("mpty fl");
    }

    if (!string.IsNullOrEmpty(pss) || !string.IsNullOrEmpty(cnf))
    {
        if (pss != cnf)
            throw new Exception("pss nt mtch");

        if (pss.Length < 4)
            throw new Exception("pss sht");
    }

    if (typ != "Wrd" && typ != "Xcl")
        throw new Exception("typ nvl");
}

public List<string> FltrFlsByTyp(List<string> flslst, string typ)
{
    List<string> rslt = new List<string>();

    if (flslst == null)
        return rslt;

    foreach (var fl in flslst)
    {
        if (string.IsNullOrEmpty(fl))
            continue;
    }
}

```

```

        string xt = Path.GetExtension(fl).ToLower();

        if (typ == "Wrd")
        {
            if (xt == ".doc" || xt == ".docx")
                rslt.Add(fl);
        }

        if (typ == "Xcl")
        {
            if (xt == ".xls" || xt == ".xlsx")
                rslt.Add(fl);
        }
    }

    return rslt;
}

public string GnrBckp(string fl)
{
    if (string.IsNullOrEmpty(fl))
        return null;

    if (!File.Exists(fl))
        return null;

    string dr = Path.GetDirectoryName(fl);
    string nm = Path.GetFileNameWithoutExtension(fl);
    string xt = Path.GetExtension(fl);

    string bckpnm = nm + "_bckp_" + DateTime.Now.Ticks.ToString()
+ xt;
    string bckppth = Path.Combine(dr, bckpnm);

    File.Copy(fl, bckppth, true);

    return bckppth;
}

public void RstrBckp(string frmpth, string tpth)
{
    if (string.IsNullOrEmpty(frmpth))
        return;

    if (string.IsNullOrEmpty(tpth))
        return;

    if (!File.Exists(frmpth))

```

```

        return;

        File.Copy(frmpth, tpth, true);
    }

    public void WrtLg(string lgpth, string txt)
    {
        if (string.IsNullOrEmpty(lgpth))
            return;

        string dr = Path.GetDirectoryName(lgpth);

        if (!Directory.Exists(dr))
            Directory.CreateDirectory(dr);

        using (FileStream fs = new FileStream(lgpth, FileMode.Append,
            FileAccess.Write, FileShare.Read))
        {
            using (StreamWriter sw = new StreamWriter(fs))
            {
                sw.WriteLine(DateTime.Now.ToString() + " : " + txt);
            }
        }
    }

    public string RdLg(string lgpth)
    {
        if (string.IsNullOrEmpty(lgpth))
            return string.Empty;

        if (!File.Exists(lgpth))
            return string.Empty;

        using (FileStream fs = new FileStream(lgpth, FileMode.Open,
            FileAccess.Read, FileShare.ReadWrite))
        {
            using (StreamReader sr = new StreamReader(fs))
            {
                string cnt = sr.ReadToEnd();
                return cnt;
            }
        }
    }

    public void ClrTmpFls(string drpth)
    {
        if (string.IsNullOrEmpty(drpth))
            return;
    }

```

```

    if (!Directory.Exists(drpth))
        return;

    string[] fls = Directory.GetFiles(drpth);

    foreach (var fl in fls)
    {
        try
        {
            FileInfo fnf = new FileInfo(fl);
            if (fnf.Attributes.HasFlag(FileAttributes.Temporary))
            {
                File.Delete(fl);
            }
        }
        catch
        {
        }
    }
}

public bool ChkFlLck(string fl)
{
    if (string.IsNullOrEmpty(fl))
        return false;

    if (!File.Exists(fl))
        return false;

    FileStream fs = null;

    try
    {
        fs = File.Open(fl, FileMode.Open, FileAccess.ReadWrite,
FileShare.None);
    }
    catch
    {
        return true;
    }
    finally
    {
        if (fs != null)
            fs.Close();
    }

    return false;
}

```

```

public void StFlRdl(string fl)
{
    if (string.IsNullOrEmpty(fl))
        return;

    if (!File.Exists(fl))
        return;

    FileInfo fnf = new FileInfo(fl);
    fnf.IsReadOnly = true;
}

public void RmvFlRdl(string fl)
{
    if (string.IsNullOrEmpty(fl))
        return;

    if (!File.Exists(fl))
        return;

    FileInfo fnf = new FileInfo(fl);
    fnf.IsReadOnly = false;
}

public void PrcssWthBckp(List<string> flslst, string pss, string
cnf, string npr, bool xst, string typ, string md, string lgpth)
{
    if (flslst == null)
        return;

    foreach (var fl in flslst)
    {
        try
        {
            string bckp = GnrBckp(fl);

            WrtLg(lgpth, "bckp crtd: " + bckp);

            if (ChkFlLck(fl))
            {
                WrtLg(lgpth, "fl lckd: " + fl);
                continue;
            }

            if (typ == "Wrd")
                WrkWrd(fl, pss, cnf, npr, xst, md);
            else
                WrkXcl(fl, pss, cnf, npr, xst, md);
        }
    }
}

```

```
        WrtLg(lgpth, "prcss dne: " + fl);
    }
    catch (Exception xcp)
    {
        WrtLg(lgpth, "rr: " + xcp.Message);
    }
}
}
```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Програмне забезпечення для запобігання несанкціонованому доступу до файлів

Виконав: Данііл Журавльов

ст. групи КНТ-132

Керівник: Тетяна ЗАЙКО

к.т.н., доцент НУ «Запорізька політехніка»

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес розробки програмного забезпечення запобігання несанкціонованому доступу до файлів.

Предмет дослідження – програмні засоби для запобігання несанкціонованому доступу до файлів.

Метою роботи є розробка програмного забезпечення для запобігання несанкціонованому доступу до файлів, що забезпечує збільшення рівня захищеності інформаційних ресурсів організації.

2

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення мети роботи потрібно розв'язати такі завдання:

- виконати аналіз предметної області та програмних засобів для запобігання несанкціонованому доступу до файлів;
- здійснити проектування програмного забезпечення для запобігання несанкціонованому доступу до файлів;
- створити програмне забезпечення для запобігання несанкціонованому доступу до файлів;
- виконати тестування розробленого програмного забезпечення для запобігання несанкціонованому доступу до файлів.

3

Рисунок В.3 – Слайд 3

Порівняння аналогічного програмного забезпечення

Критерій порівняння	Files.com	Cove	FolderLock
хмарне зберігання даних	+	+	–
резервне копіювання та відновлення	+–	+	–
інтеграція із зовнішніми сервісами	+	+–	–
локальний захист файлів	+–	–	+
безпечне видалення та очищення даних	–	+–	+
контролювання прав користувачів	+	+	+–

4

Рисунок В.4 – Слайд 4

Вимоги до програми

При розробці програмного забезпечення для запобігання несанкціонованому доступу до файлів необхідно забезпечити такі функціональні вимоги:

- підтримка можливості встановлення параметрів захисту для текстових та табличних документів;
- підтримка можливості зняття раніше встановленого захисту за умови введення коректного пароля;
- можливість формування та редагування переліку файлів, що підлягають обробці, із додаванням окремих документів або пошуком у вибраній теці;
- підтримка вибору типу документа для забезпечення коректного застосування відповідних режимів захисту;
- можливість встановлення пароля на відкриття або редагування документа з обов'язковим підтвердженням введених даних;
- підтримка вибору режиму обмеження доступу до документа, зокрема режиму перегляду без редагування, режиму внесення змін із контролем прав або повної відсутності захисту;
- забезпечення перевірки коректності введених користувачем параметрів перед початком виконання операцій;
- підтримка обробки виняткових ситуацій, пов'язаних із відсутністю файлу, недостатніми правами доступу або помилками під час взаємодії з файловою системою;
- підтримка інформування користувача про результати виконання операцій та можливі помилки в процесі роботи програми.

5

Рисунок В.5 – Слайд 5

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	C#	Java	Go
Інтеграція з механізмами безпеки операційної системи	+	+-	+-
Наявність вбудованих криптографічних бібліотек	+	+	+-
Зручність реалізації інтерфейсу користувача для настільних систем	+	+-	-
Швидкість розробки	+	+-	+
Збалансованість безпеки та продуктивності	+	+-	+

6

Рисунок В.6 – Слайд 6

Порівняння середовищ розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	MonoDevelop	SharpDevelop
Повноцінна підтримка .NET	+	+—	+—
Розширені засоби налагодження	+	+—	—
Підтримка тестування та аналізу покриття	+	+—	+—
Зручність створення інтерфейсу користувача	+	+—	+—
Стабільність та зрілість платформи	+	+—	—

7

Рисунок В.7 – Слайд 7

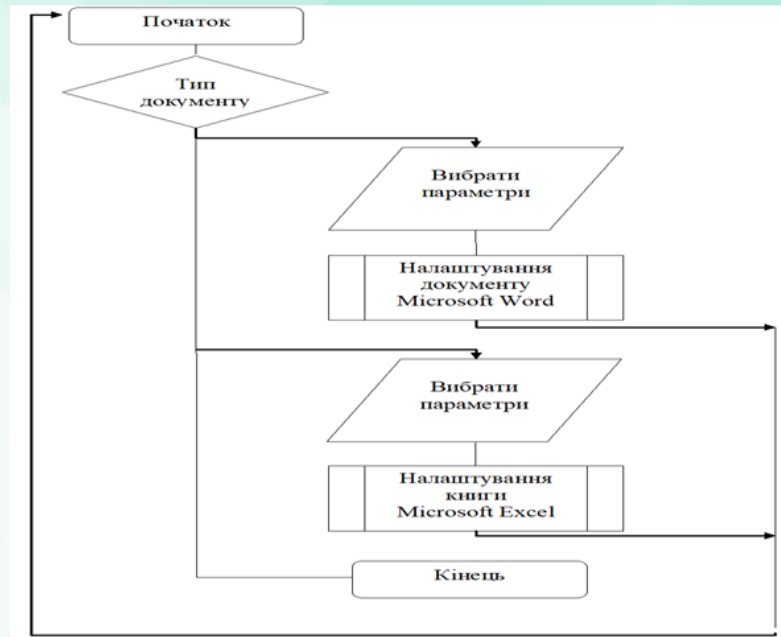
Структура програми



8

Рисунок В.8 – Слайд 8

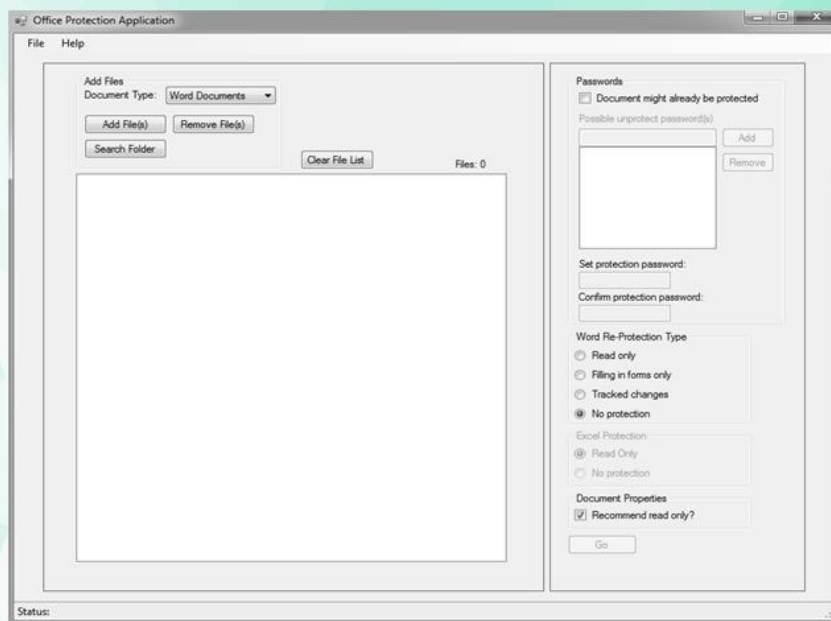
Схема функціонування програми



9

Рисунок В.9 – Слайд 9

Робота з програмою



10

Рисунок В.10 – Слайд 10

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для запобігання несанкціонованому доступу до файлів.

Сформульовано функціональні вимоги до програмного забезпечення для запобігання несанкціонованому доступу до файлів. Для реалізації програмного забезпечення для запобігання несанкціонованому доступу до файлів обрано мову програмування C# та середовище розробки Visual Studio.

Розроблено програмне забезпечення для запобігання несанкціонованому доступу до файлів. Запропоновано структуру програмного забезпечення для запобігання несанкціонованому доступу до файлів, яка сформована за модульним принципом, що забезпечує розмежування відповідальності між модулем ініціалізації, головною формою керування, довідковою формою, компонентом обробки документів, підсистемою перевірки параметрів та механізмом інформування.

Виконано тестування розробленого програмного забезпечення для запобігання несанкціонованому доступу до файлів. За результатами тестування встановлено, що програмне забезпечення в цілому відповідає технічному завданню.

11

Рисунок В.11 – Слайд 11