ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «СИСТЕМА РЕЗЕРВНОГО КОПІЮВАННЯ ДАНИХ ДЛЯ ЛОКАЛЬНОЇ МЕРЕЖІ»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Михайло ЛИННИК*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КІД-42
Михайло ЛИННИК
Ім'я, ПРІЗВИЩЕ
Керівник: Артем АНТОНЕНКО
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії
Ступінь вищої освіти Бакалавр
Спеціальність 123 Комп'ютерної інженерії
Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
Ім'я, ПРІЗВИЩЕ
«_____» _____ 2026р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Линник Михайло Сергійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Система резервного копіювання даних для локальної мережі
керівник кваліфікаційної роботи Артем АНТОНЕНКО,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» 02 2026р. № 51
2. Строк подання кваліфікаційної роботи «08» 05 2026р.
3. Вихідні дані до кваліфікаційної роботи: Науково-технічна література, методи системного аналізу, методи математичного моделювання для розрахунку алгоритмів маршрутизації, математичний апарат, емпіричні методи
Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Дослідження проблематики резервного збереження даних у локальних мережах
2. Розробка архітектурних рішень та алгоритмів функціонування програмного комплексу автоматизованого резервного збереження даних
3. Технічна реалізація та експериментальна апробація прототипу системи автоматизованого резервного збереження даних
4. Перелік ілюстративного матеріалу: *презентація*
5. Дата видачі завдання «09» 02 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	25.02-24.03.2026	Виконано
2	Обробка матеріалу	23.03-05.04.2026	Виконано
3	Розробка теоретичної частини	06.04-10.04.2026	Виконано
4	Розробка практичної частини	11.04-20.04.2026	Виконано
5	Висновки за результатами аналізу	21.04-06.05.2026	Виконано
6	Оформлення пояснювальної записки	07.05-08.05.2026	Виконано
7	Розробка презентації	09.05-11.05.2026	Виконано
8	Передзахист	27.05-29.05.2026	Виконано
9	Здача в деканат	02.06-05.06.2026	Виконано

Здобувач вищої освіти _____ Михайло ЛИННИК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Артем АНТОНЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 60 стор., 11 рис., 27 джерел.

Мета дослідження – Розробка програмного комплексу автоматизованого резервного збереження даних для локальних мереж.

Об'єкт дослідження – Процес резервного збереження та захисту інформаційних масивів у локальних мережах.

Предмет дослідження – Програмно-технічні комплекси та алгоритми інтелектуального резервного копіювання даних з використанням хмарних технологій.

Короткий зміст роботи: Сучасні малі підприємства функціонують у надзвичайно динамічному та конкурентному середовищі, де раптова втрата цифрової інформації може призвести до катастрофічних фінансових збитків та зупинки бізнесу. Загрозу становлять як апаратні збої та людський фактор, так і цілеспрямовані хакерські атаки чи віруси-вимагачі. Існуючі рішення для резервного копіювання, такі як ручне перенесення на диски, мережеві сховища або спеціалізовані корпоративні продукти, часто є ненадійними, економічно невиправданими або надто складними для компаній без виділеного штату IT-фахівців.

На основі проведеного дослідження було доведено актуальність створення унікального рішення, яке б поєднало фінансову доступність, надійність та простоту використання публічних хмарних сховищ. У роботі запропоновано та розроблено архітектуру системи «Reserve», яка використовує Google Drive API для віддаленого збереження даних. Програмний комплекс забезпечує глибоку компресію об'єктів та надійне симетричне криптографічне шифрування за стандартом AES-256. Крім того, теоретично обґрунтовано інтеграцію алгоритмів машинного навчання, зокрема ансамблевого методу випадкового лісу (Random Forest), для динамічної пріоритезації черги завантаження з метою оптимізації мережевого трафіку. Експериментальна апробація підтвердила високий рівень відмовоустійливості та ефективності розробленої системи.

КЛЮЧОВІ СЛОВА: РЕЗЕРВНЕ КОПІЮВАННЯ, ЛОКАЛЬНА МЕРЕЖА,

GOOGLE DRIVE API, ШИФРУВАННЯ ДАНИХ, МАШИННЕ НАВЧАННЯ, ХМАРНЕ СХОВИЩЕ.

ABSTRACT

Textual part of the qualification work for the bachelor's degree: 60 pages, 11 figures, 27 sources.

The goal of the work – Development of an automated data backup software complex for small business local networks.

The object of research – The process of data backup and protection of information arrays in local networks.

The subject of research – Software and hardware complexes and algorithms for intelligent data backup using cloud technologies.

Summary of the work: Modern small enterprises operate in a highly dynamic and competitive environment where the sudden loss of digital information can lead to catastrophic financial losses and business shutdown. Threats include hardware failures, human errors, targeted hacker attacks, and ransomware. Existing backup solutions, such as manual transfer to disks, Network Attached Storage, or specialized corporate products, are often unreliable, economically unjustified, or overly complex for companies without dedicated IT staff.

Based on the research conducted, the relevance of creating a unique solution that combines the financial accessibility, reliability, and ease of use of public cloud storage was proven. The thesis proposes and develops the architecture of the "Reserve" system, which uses the Google Drive API for remote data storage. The software complex provides deep object compression and reliable symmetric cryptographic encryption according to the AES-256 standard. Furthermore, the integration of machine learning algorithms, particularly the Random Forest ensemble method, is theoretically justified for dynamic prioritization of the upload queue to optimize network traffic. Experimental testing confirmed the high level of fault tolerance and efficiency of the developed system.

KEYWORDS: DATA BACKUP, LOCAL NETWORK, GOOGLE DRIVE API, DATA ENCRYPTION, MACHINE LEARNING, CLOUD STORAGE.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

ПОДАННЯ
ГОЛОВІ ДЕРЖАВНОЇ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня бакалавра

Направляється здобувач Линник М. С. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 123 Комп'ютерна інженерія
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)
на тему: «Система резервного копіювання даних для локальної мережі».
Кваліфікаційна робота і рецензія додаються.

Директор ННІ _____ Катерина НЕСТЕРЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувач Линник М. С. показав відмінну теоретичну підготовку, уміння володіти та розуміння новими комп'ютерними технологіями, користуватися навчальною і науково-технічною літературою. Працюючи над завданнями, які були поставлені керівником, проявив ініціативність, сумлінність та хист до інженерної роботи.
Все це дозволяє оцінити кваліфікаційну роботу студента Линника Михайло Сергійовича на оцінку «відмінно», та присвоїти йому кваліфікацію «бакалавр з комп'ютерної інженерії».

Керівник кваліфікаційної роботи _____ Артем АНТОНЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

«_» _____ 2026 року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач *Линник М. С.* допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедрою Комп'ютерної інженерії _____ Наталія ЛАЩЕВСЬКА
(назва) *(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну бакалаврську роботу

здобувача вищої освіти Линник Михайло Сергійович
(прізвище, ім'я, по батькові)
на тему: «Система резервного копіювання даних для локальної мережі»

Актуальність:

Бакалаврська кваліфікаційна робота присвячена одній із найбільш актуальних проблем сучасного корпоративного сектору — забезпеченню надійного резервного збереження та захисту цифрової інформації у локальних мережах малого бізнесу. В умовах стрімкого зростання кількості кіберзагроз (зокрема вірусів-вимагачів) та регулярних апаратних збоїв, запропонована розробка спеціалізованого та фінансово доступного програмного комплексу має високу практичну значущість. Автор пропонує сучасний підхід, що поєднує можливості публічних хмарних платформ із методами криптографічного захисту та інтелектуальної аналітики.

Позитивні сторони:

1. У роботі наведено якісний порівняльний аналіз провідних хмарних сервісів. Автор вдало аргументував вибір платформи Google Drive API як найбільш оптимального фундаменту для інтеграції, враховуючи специфіку потреб малого бізнесу.
2. Автор продемонстрував високу компетентність у проєктуванні модульної архітектури програмного забезпечення. Розроблено комплексний технологічний ланцюжок, який включає надійне симетричне шифрування за стандартом AES-256 та попередню компресію даних перед їх відправкою у хмару.
3. Значною перевагою є наукова новизна роботи, яка полягає в обґрунтуванні використання ансамблевих алгоритмів машинного навчання для динамічного оцінювання та пріоритезації черги завантаження файлів.

Недоліки:

1. Незважаючи на успішну експериментальну апробацію, у поточній реалізації прототипу системи процес формування черги файлів здебільшого покладається на ручний вибір користувача, а повноцінний алгоритм інкрементного копіювання та фонового моніторингу метаданих відзначено лише як перспективу подальшого розвитку.
2. У розділі експериментальних досліджень було б доцільно додати графіки або таблиці з розрахунками точності для класифікації файлів різної важливості.

Висновок: *кваліфікаційна бакалаврська робота заслуговує оцінку " відмінно", а здобувач Линник М. С.. заслуговує присвоєння кваліфікації: бакалавра з комп'ютерної інженерії.*

Рецензент: _____ _____
науковий ступінь, вчене звання *підпис* *Ім'я, ПРІЗВИЩЕ*

ЗМІСТ

ВСТУП 11
1 ДОСЛІДЖЕННЯ ПРОБЛЕМАТИКИ РЕЗЕРВНОГО ЗБЕРЕЖЕННЯ ДАНИХ У ЛОКАЛЬНИХ МЕРЕЖАХ 13
1.1 Дослідження наявних програмно-техничних комплексів для організації резервного збереження даних 13
1.2 Порівняльний аналіз сучасних хмарних платформ із підтримкою програмних інтерфейсів 17
1.3 Аналіз структурних типів даних та джерел інформації в межах локальної мережі 22
2 РОЗРОБКА АРХІТЕКТУРНИХ РІШЕНЬ ТА АЛГОРИТМІВ ФУНКЦІОНУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ АВТОМАТИЗОВАНОГО РЕЗЕРВНОГО ЗБЕРЕЖЕННЯ ДАНИХ 25
2.1 Структурна організація програмного забезпечення та взаємозв'язки між його модулями 25
2.2 Теоретичні засади та обґрунтування вибору алгоритмів машинного навчання для визначення пріоритетності інформаційних масивів 28
2.3 Функціональна логіка та алгоритмічна модель процесу інтелектуального резервного копіювання. 31
2.4 Компонентна структура та ергономіка графічного інтерфейсу користувача 34
3 ТЕХНІЧНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА АПРОБАЦІЯ ПРОТОТИПУ СИСТЕМИ АВТОМАТИЗОВАНОГО РЕЗЕРВНОГО ЗБЕРЕЖЕННЯ ДАНИХ 39
3.1 Програмна реалізація компонента взаємодії з програмним інтерфейсом Google Drive API 39
3.2 Особливості програмного моделювання та практичного впровадження алгоритмів розумного збереження даних 43

3.3 Програмна розробка та практичне втілення компонентів графічного інтерфейсу 45
3.4 Комплексне дослідження показників швидкодії, відмовоустійливості та ефективності програмного комплексу..... 50
ВИСНОВКИ..... 55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 57
Додаток А 60

ВСТУП

У сучасному світі цифрова інформація перетворюється на один із найцінніших активів будь-якої компанії. Для малих підприємств, які функціонують у надзвичайно динамічному та конкурентному середовищі, раптова втрата даних у межах робочої локальної мережі може призвести до катастрофічних фінансових збитків, підриву репутації та навіть до примусового припинення операційної діяльності. Чинники руйнування інформаційних масивів є різноманітними: від типових апаратних збоїв та випадкового видалення файлів співробітниками до деструктивних впливів шкідливого програмного забезпечення, зокрема вірусів-вимагачів та цілеспрямованих кібератак.

Існуючі підходи до організації резервного копіювання мають суттєві обмеження. Найпростіше ручне копіювання на зовнішні носії демонструє вкрай низький рівень надійності через залежність від дій персоналу. Використання локальних мережевих сховищвимагає відчутних стартових капіталовкладень на купівлю обладнання та залишається вразливим до масштабних локальних інцидентів. Водночас спеціалізоване комерційне програмне забезпечення від лідерів індустрії відрізняється високою вартістю ліцензій та надмірною складністю для невеликих фірм, де немає штату кваліфікованих системних інженерів.

Зважаючи на це, виникає об'єктивна необхідність у розробці спеціалізованого програмного забезпечення, яке б органічно поєднало фінансову доступність, надійність та простоту публічних хмарних сховищ із розвиненим функціоналом автоматизації та безпеки. Побудова такої системи на базі програмних інтерфейсів глобальних платформ, таких як Google Drive API, є найбільш раціональним рішенням. Додаткова інтеграція алгоритмів машинного навчання дозволить автоматично аналізувати поведінку користувачів і виділяти найбільш критичні об'єкти для першочергового

збереження, мінімізуючи навантаження на мережу. Саме дослідженню та розробці такої комплексної інтелектуальної системи резервного збереження даних присвячена дана кваліфікаційна робота.

1 ДОСЛІДЖЕННЯ ПРОБЛЕМАТИКИ РЕЗЕРВНОГО ЗБЕРЕЖЕННЯ ДАНИХ У ЛОКАЛЬНИХ МЕРЕЖАХ

1.1 Дослідження наявних програмно-техничних комплексів для організації резервного збереження даних

Сучасні малі підприємства функціонують у надзвичайно динамічному та конкурентному середовищі, де ключовими факторами виживання та успіху стають швидкість прийняття рішень, організаційна гнучкість та жорстка оптимізація операційних витрат. У таких умовах цифрова інформація перетворюється на один із найцінніших активів компанії, а її раптова втрата в межах робочої локальної мережі може призвести до катастрофічних фінансових збитків, серйозного підриву репутації перед клієнтами, юридичних санкцій та навіть до повного примусового припинення операційної діяльності фірми [2].

Комплексний аналіз показує, що типові чинники руйнування інформаційних масивів у локальних мережах є досить різноманітними. Серед них часто виділяються суто апаратні збої, такі як каскадні відмови жорстких дисків, вихід з ладу комутаційного обладнання або центральних серверів, критичні програмні помилки в системному забезпеченні, а також випадкове видалення чи некоректний перезапис важливих файлів самими співробітниками через неуважність. Крім того, не меншу загрозу становлять зовнішні деструктивні впливи, зокрема шкідливе програмне забезпечення. Сюди відносяться сучасні віруси-вимагачі та програми-здирники типу ransomware, які здатні за лічені хвилини зашифрувати всі доступні мережеві ресурси та робочі станції. Велику небезпеку несуть і цілеспрямовані хакерські атаки на периметр мережі, фізична крадіжка комп'ютерного обладнання, а

також непередбачувані форс-мажорні обставини типу пожеж, аварій системи опалення чи затоплень офісних приміщень [1, 9].

Глибоке дослідження потреб невеликих компаній у сфері побудови систем резервного копіювання та комплексного захисту мережевих даних дозволяє сформувати перелік специфічних вимог до таких інструментів [3]:

Простіть використання та конфігурації. Оскільки малий бізнес найчастіше має обмежений штат IT-фахівців або взагалі обходиться без постійного системного адміністратора, програмне рішення повинно мати інтуїтивно зрозумілий інтерфейс. Воно не має вимагати глибоких вузькоспеціалізованих знань для розгортання на вузлах локальної мережі, первинного налаштування політик та щоденного моніторингу поточного стану копій [20].

Економічна доступність рішень. Фінансові бюджети малих підприємств на автоматизацію та безпеку завжди жорстко лімітовані, тому розроблювана архітектура має бути максимально вигідною як на стадії первинного впровадження, так і під час тривалої експлуатації, що включає витрати на ліцензування, технічну підтримку та оплату дискового простору.

Високий рівень автоматизації. Створення резервних копій у межах локальної мережі має відбуватися повністю в автоматичному режимі за чітко визначеним розкладом або у відповідь на конкретні події (наприклад, при модифікації файлів на робочих станціях), що повністю усуває людський фактор та ризик забування.

Надійність та гарантія збереження інформації. Кінцевий користувач повинен мати абсолютну впевненість у тому, що дані успішно скопійовані, а інтегровані механізми регулярної перевірки цілісності створених архівів дозволяють гарантувати їхню повну придатність до зчитування у будь-який момент часу [5].

Комплексна безпека даних. Конфіденційна інформація має бути надійно захищена від несанкціонованого доступу чи перехоплення як на етапі її транспортування всередині локальної мережі за допомогою захищених

мережевих протоколів, так і під час безпосереднього зберігання у кінцевому сховищі шляхом стійкого криптографічного шифрування [10].

Гнучкість керування. Система зобов'язана надавати користувачеві можливість вибіркового резервування конкретних директорій чи типів файлів на різних вузлах мережі, гнучкого налаштування індивідуальних політик версійності, термінів зберігання даних та чіткого визначення винятків для оптимізації простору.

Ефективне та швидке відновлення. Процес ліквідації наслідків збоїв, який включає як повне аварійне відновлення всієї системи, так і покрокове вибіркове повернення окремих файлів чи документів, має виконуватися максимально просто, прозоро та за мінімальний проміжок часу [6].

Сучасні підходи та готові практичні рішення для організації резервного копіювання, які сьогодні застосовуються на практиці, можна детально класифікувати за їхніми технологічними та експлуатаційними особливостями.

Найпростішим з точки зору первинних витрат є ручне копіювання інформації на зовнішні фізичні носії, такі як портативні жорсткі диски, швидкісні твердотільні накопичувачі або звичайні USB-флешки. Проте цей підхід демонструє вкрай низький рівень надійності через тотальну залежність від дисципліни та дій персоналу. Повна відсутність системної автоматизації, регулярні збої у графіках копіювання, неможливість адекватно вести облік версій файлів та зберігання копій у тому ж самому фізичному приміщенні поруч із оригіналами роблять цей метод абсолютно беззахисним перед більшістю сучасних загроз безпеці.

Більш просунутим варіантом є використання локальних мережевих сховищ, відомих як системи NAS. Вони забезпечують чудову швидкість обміну даними та оперативне відновлення в межах високошвидкісної локальної мережі, залишаючи компанії повний контроль над фізичним розміщенням її серверів та накопичувачів. З іншого боку, побудова повноцінного та відмовостійкого NAS вимагає відчутних стартових капіталовкладень на купівлю обладнання, регулярного технічного

обслуговування, а також залишається вразливою до масштабних локальних інцидентів, наприклад, пожеж, крадіжок або скоординованих кібератак на локальну інфраструктуру, якщо сховище не дублюється за межами офісу.

Зовсім інший клас рішень представляє спеціалізоване комерційне програмне забезпечення для резервного копіювання від лідерів індустрії, яскравими прикладами якого є продукти брендів Acronis або Veeam Backup & Replication. Цей софт володіє колосальним інструментарієм, підтримує інкрементне та диференційне копіювання, централізоване керування всіма вузлами мережі та складні індивідуальні політики зберігання [3, 4]. Вони демонструють високу надійність, проте висока вартість комерційних ліцензій та надмірна складність інтерфейсів конфігурації роблять їх малопридатними та економічно невиправданими для невеликих фірм, у яких немає виділеного штату кваліфікованих системних інженерів.

Паралельно розвиваються спеціалізовані хмарні сервіси резервного збереження типу Backblaze, Carbonite або IDrive. Вони беруть на себе завдання повної автоматизації процесу та гарантують безпечне віддалене зберігання даних на ізольованих географічних майданчиках, пропонуючи вбудований контроль версій та зручні інструменти аварійного відновлення [13]. Вони є надійним вибором, але вартість їхніх спеціалізованих тарифних планів для бізнесу може суттєво перевищувати ціну звичайних дисків, а жорстко обмежений функціонал не завжди дозволяє тонко адаптувати систему під унікальні внутрішні потреби конкретної локальної мережі.

На цьому тлі вкрай привабливо виглядають хмарні файлові сховища загального призначення, такі як Google Drive, Microsoft OneDrive або Dropbox, які часто розглядаються компаніями як базова інфраструктура для резервування. Вони надзвичайно поширені, доступні за ціною і часто вже інтегровані в повсякденну роботу співробітників для синхронізації та спільного доступу до документів [11, 18]. Вони забезпечують надійне зберігання даних за межами локального офісу та зручний доступ через веб-інтерфейс з будь-якої точки світу.

Проте, якщо оцінювати їх як готові системи резервного копіювання, стають помітними суттєві обмеження: у них зазвичай відсутні вбудовані алгоритми інкрементного копіювання, немає механізмів глибокої дедуплікації даних на рівні локальної мережі та бракує інструментів централізованого адміністрування клієнтських вузлів. Така технологічна обмеженість створює чітку необхідність у проєктуванні та розробці додаткового спеціалізованого програмного забезпечення, яке б узяло на себе завдання реалізації необхідного рівня автоматизації, інтелектуального шифрування перед відправкою та загального підвищення ефективності використання дискового простору.

Таким чином, детальний аналіз наявних методологічних підходів та специфічних вимог невеликих організацій наочно доводить актуальність створення унікального рішень для локальних мереж. Таке рішення має органічно поєднати в собі фінансову доступність, надійність та простоту експлуатації публічних хмарних сховищ із розвиненим функціоналом автоматизації, наскрізної безпеки та інтелектуальної обробки трафіку. Побудова такої системи на базі використання відкритих програмних інтерфейсів (API) наявних глобальних хмарних платформ є найбільш раціональним та перспективним вектором для досягнення поставленої мети [16].

1.2 Порівняльний аналіз сучасних хмарних платформ із підтримкою програмних інтерфейсів

Визначення оптимальної хмарної платформи є одним із найбільш фундаментальних та відповідальних етапів під час проєктування та розробки сучасної системи резервного копіювання, яка покладається на гібридне або повністю віддалене збереження інформаційних масивів локальної мережі [14, 20]. При цьому інженерний аналіз не повинен обмежуватися лише

порівнянням вартості тарифних планів чи доступного обсягу дискового простору. Набагато важливішим чинником виступає наявність гнучкого, стабільного та багатофункціонального програмного інтерфейсу (API), який дає розробнику можливість реалізувати складну внутрішню логіку життєвого циклу резервних копій, що кардинально відрізняється від алгоритмів простої двосторонньої синхронізації файлів. Серед усього різноманіття сучасних хмарних сховищ, які відкривають свої програмні інтерфейси для сторонніх інтеграцій, можна чітко виділити кілька ключових технологічних рішень.

Платформа Google Drive, рис.1.1, яка є невіддільною частиною глобальної екосистеми Google Workspace, на сьогодні має величезну популярність як серед приватних користувачів, так і в корпоративному сегменті. Вона надає у розпорядження розробників потужний та високопродуктивний RESTful API третьої версії (Google Drive API v3) [11], що супроводжується детально документованою та офіційно підтримуваною бібліотекою для мови програмування Python [13]. Цей інструментарій дозволяє організувати повноцінну та безпечну роботу з файловою структурою віддаленого сховища безпосередньо з вузлів локальної мережі, забезпечуючи операції безшовного завантаження нових архівів, оновлення існуючих об'єктів, видалення застарілих версій, проведення швидкого контекстного пошуку, гнучкого зчитування метаданих, а також динамічного керування правами доступу. Крім того, сервіс пропонує дуже привабливі та гнучкі тарифи, включаючи вагомий безкоштовний стартовий обсяг сховища, що мінімізує витрати для невеликих підприємств. Механізм автентифікації та авторизації клієнтських додатків базується на базі загальноприйнятого стандарту OAuth 2.0 [12], який гарантує високий рівень безпеки та є зрозумілим для адміністрування.



Рисунок 1.1 – Хмарне файлове сховище Google Drive

Іншим вагомим гравцем є Microsoft OneDrive, рис.1.2, який має максимально тісну інтеграцію з операційною системою Windows, що домінує на робочих станціях більшості локальних мереж, та пакетом офісних програм Microsoft 365. Взаємодія з цією платформою на програмному рівні реалізується через універсальний інтерфейс Microsoft Graph API [17]. Цей сервіс також має у своєму арсеналі відповідний Python SDK та повністю підтримує базовий набір операцій над файловими об'єктами. OneDrive є звичним середовищем для компаній, які вже інвестували в інфраструктуру від Microsoft, і так само надає базовий безкоштовний ліміт та різноманітні варіанти комерційних підписок. Безпека тут теж тримається на протоколі OAuth 2.0, проте інтеграція через масштабний Graph API часто вимагає від розробника надлишкового занурення в архітектуру суміжних хмарних сервісів Microsoft, що може суттєво ускладнити та уповільнити процес створення цільової утиліти для бекапу локальної мережі.



Рисунок 1.2 – Хмарне файлове сховище Microsoft OneDrive

У свою чергу, сервіс Dropbox, рис.1.3, вважається одним із першопрохідців у сфері хмарного збереження та хмарної синхронізації, заслуживши репутацію швидкого та безвідмовного інструменту. Для автоматизації процесів він пропонує власний Dropbox API [18] з офіційним пакетом розробника під Python, який повністю закриває базові потреби щодо маніпулювання файловими структурами. Разом із цим, цінова політика Dropbox зазвичай є менш вигідною для малого бізнесу порівняно з головними конкурентами за аналогічні обсяги пам'яті, хоча сервіс і зберігає стабільне ядро лояльних користувачів, використовуючи той самий перевірений протокол OAuth 2.0 для захисту з'єднань.



Рисунок 1.3 – Хмарне файлове сховище Dropbox

Кардинально інший підхід уособлює сервіс об'єктного зберігання Amazon S3 (Simple Storage Service), рис.1.4, який входить до складу хмари AWS та є фактичним індустріальним стандартом для великих корпоративних систем резервного копіювання і довгострокового утримання величезних масивів інформації [19]. Він володіє неймовірно масштабованим, відмовостійким та потужним програмним інтерфейсом, який чудово підтримується в Python через спеціалізовану бібліотеку boto3. Незважаючи на всі архітектурні переваги Amazon S3, його модель тарифікації, яка динамічно залежить від обсягів збереження, кількості окремих API-запитів та напрямків мережевого трафіку, а також специфічний і складний процес авторизації за допомогою механізму AWS Signature, створюють високий поріг входження. Це робить його надто важким та нераціональним рішенням для невеликих локальних мереж малого бізнесу, де немає досвідчених хмарних архітекторів.



Рисунок 1.3 – Хмарне файлове сховище Amazon S3

Зважаючи на специфіку та реальні пріоритети малого бізнесу, серед яких на першому місці стоять гранична простота щоденної експлуатації, жорсткий контроль фінансових витрат та легкість інтеграції в наявну IT-інфраструктуру, як найбільш оптимальний та збалансований фундамент для реалізації системи резервного копіювання в локальній мережі було обрано саме Google Drive API. Основним аргументом на користь такого вибору стала повсюдна поширеність сервісів Google серед корпоративних користувачів, що дозволяє суттєво

спростити та прискорити процедуру первинної авторизації нових робочих станцій у системі [12]. Крім того, відносна простота, логічність та вичерпність офіційної технічної документації Google Drive API [11], у поєднанні з наявністю зручної, стабільної клієнтської бібліотеки для мови Python [13], дозволяють зробити процес безпосередньої розробки та подальшої підтримки програмного коду максимально швидким та ефективним. Важливу роль відіграють також комерційні чинники, адже гнучка цінова політика разом із безкоштовним дисковим простором гарантують пряму економічну вигоду для підприємства, тоді як сам програмний інтерфейс у повному обсязі надає всі необхідні низькорівневі та високорівневі функції для побудови надійного локального рішення автоматизованого бекапу. Таким чином, інтеграція обраного Google Drive API дозволяє створити збалансоване, доступне та інтуїтивно зрозуміле програмне забезпечення, здатне ефективно закрити всі потреби малого бізнесу в автоматизованому захисті мережевих даних без залучення зайвих фінансових та людських ресурсів.

1.3 Аналіз структурних типів даних та джерел інформації в межах локальної мережі

Проєктування ефективної архітектури системи резервного збереження вимагає детальної класифікації та глибокого аналізу всіх джерел, які генерують та накопичують критично важливу інформацію в межах локальної інфраструктури малого підприємства [5, 22]. Основними вузлами-джерелами в таких мережах зазвичай виступають персональні робочі станції співробітників, локальні файлові сервери або мережеві накопичувачі, на яких зосереджено щоденний документообіг, фінансова звітність, юридична та інженерна документація. Окремим важливим та технологічно складним сегментом є локальні сервери баз даних, які забезпечують функціонування

систем автоматизації торгівлі, складського обліку чи управління взаємовідносинами з клієнтами. Кожен із цих об'єктів має свої унікальні особливості доступу, специфічні режими блокування файлів під час активної роботи персоналу та різну інтенсивність оновлення інформації, що робить задачу їх централізованого збору та каталогізації першочерговою для запобігання втрати даних [14].

Сама структура інформаційних масивів, що підлягають регулярному резервуванню в локальній мережі, є неоднорідною і ділиться на кілька базових технологічних категорій. Найбільший обсяг зазвичай становить неструктурована інформація, представлена звичайними користувацькими файлами, такими як текстові документи, таблиці, презентації, графічні матеріали та цифрові архіви. Для оптимізації роботи з цим типом даних програмний комплекс має підтримувати логіку пофайлового або поблокового аналізу для точного виявлення модифікацій. З іншого боку, структуровані дані, до яких належать файли реляційних баз даних та журнали транзакцій, вимагають принципово інших підходів, оскільки вони перебувають у стані постійної зміни в процесі роботи підприємства. Логіка взаємодії з такими джерелами передбачає інтеграцію механізмів створення тимчасових знімків файлової системи, що дозволяє безпечно скопіювати стан бази на конкретний момент часу без зупинки операційної діяльності компанії [20]. Окреме місце посідають напівструктуровані дані, наприклад, конфігураційні файли мережевого обладнання та налаштування сервісів у форматах XML або JSON, збереження яких є критично важливим для швидкого відновлення працездатності всієї локальної мережі у разі масштабного збою.

Для забезпечення коректного функціонування системи та її стабільної взаємодії з програмним інтерфейсом Google Drive виникає необхідність формування внутрішньої структури метаданих самого додатка резервного копіювання. Цей внутрішній опис є інформаційним містком і містить відомості про унікальні ідентифікатори об'єктів у хмарному сховищі, контрольні суми файлів для перевірки їхньої цілісності, точні часові мітки останньої

модифікації, а також оригінальні відносні та абсолютні шляхи в локальній файловій системі. Алгоритм порівняння динамічно аналізує ці параметри під час кожного робочого циклу, щоб безпомилково визначити, які саме елементи зазнали змін і потребують відправки у хмару. Логіка системи самостійно розраховує сумарний обсяг нових даних шляхом послідовного додавання розмірів усіх модифікованих об'єктів, що дозволяє програмі оцінити навантаження на мережевий канал та зіставити його з поточною пропускною здатністю локальної мережі. Таке чітке структурування джерел та внутрішніх метаданих гарантує високу швидкість обробки інформації, мінімізує надлишковий дублюючий трафік та забезпечує стабільність усього процесу захисту даних [11, 13].

2 РОЗРОБКА АРХІТЕКТУРНИХ РІШЕНЬ ТА АЛГОРИТМІВ ФУНКЦІОНУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ АВТОМАТИЗОВАНОГО РЕЗЕРВНОГО ЗБЕРЕЖЕННЯ ДАНИХ

2.1 Структурна організація програмного забезпечення та взаємозв'язки між його модулями

Ефективність функціонування розроблюваної системи резервного копіювання в межах локальної мережі безпосередньо залежить від раціонального вибору її архітектурного шаблону. Для досягнення високої гнучності, відмовостійкості та простоти подальшого супроводу було обрано модульну клієнт-орієнтовану архітектуру. Такий підхід дозволяє розділити загальну логіку програми на кілька слабопов'язаних компонентів, кожен з яких відповідає за суворо визначене коло завдань — від моніторингу локальних директорій на робочих станціях до низькорівневої взаємодії з віддаленими хмарними серверами. Центральним елементом системи виступає диспетчер завдань, або керуюче ядро автоматизації, яке здійснює загальну координацію, керує життєвим циклом усіх процесів бекапу, відстежує стан системних ресурсів та забезпечує точне виконання встановленого користувачем часового розкладу. Загальна концептуальна схема взаємодії цих архітектурних компонентів під час виконання типового робочого циклу наочно відображена на рисунку 2.1, де детально показано напрямки руху інформаційних потоків і керуючих сигналів між внутрішніми модулями програми та зовнішнім середовищем Google Drive.

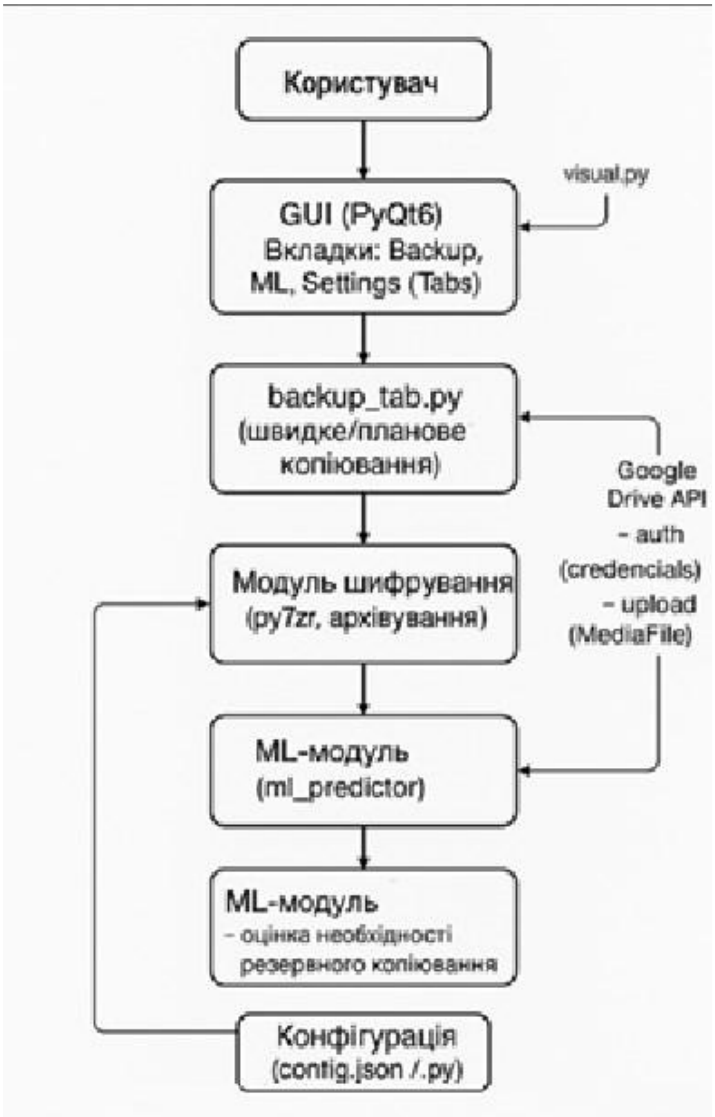


Рисунок 2.1 — Схема структурно-функціональної організації та інформаційних зв’язків елементів системи бекапу

Першим критично важливим компонентом у цій структурі є модуль сканування файлової системи та обліку метаданих. Його головне призначення полягає у регулярному обході цільових локальних папок на вузлах мережі, виявленні нових чи модифікованих файлів та веденні локального реєстру станів. Замість тривалого та ресурсомісткого повного копіювання, яке б створювало надмірне навантаження на канали зв'язку, цей модуль реалізує інтелектуальну логіку інкрементного аналізу. Він зчитує базові атрибути кожного об'єкта, обчислює унікальні хеш-значення для перевірки цілісності та миттєво порівнює їх із записами у локальній базі даних метаданих. Якщо

часова мітка або обчислена контрольна сума файлу не збігаються з попередніми параметрами, модуль безпомилково маркує цей об'єкт як такий, що потребує резервування, і передає інформацію про нього далі по технологічному ланцюжку.

Отримавши перелік змінених елементів, ядро системи активує модуль підготовки, компресії та безпеки даних. Оскільки конфіденційність комерційної інформації малого бізнесу є першочерговим пріоритетом, цей компонент виконує повну попередню обробку файлів перед їх виходом за межі периметра безпеки локальної мережі [10]. Логіка роботи цього модуля побудована так, що вихідні файли спочатку групуються та пакуються у компактні архівні структури, що дозволяє суттєво зекономити мережевий трафік та оптимізувати використання дискового простору в хмарі. Одразу після цього архіви обробляються стійкими алгоритмами симетричного криптографічного шифрування з використанням унікальних ключів, які генеруються та зберігаються виключно на стороні локального сервера компанії. Це гарантує, що навіть у разі перехоплення трафіку зловмисниками або потенційного витоку даних на публічній платформі, вміст резервних копій залишиться повністю захищеним і непридатним для стороннього читання.

За безпосереднє транспортування підготовлених зашифрованих пакетів відповідає модуль мережевої взаємодії та інтеграції з API. Цей компонент є програмним шлюзом, який безпосередньо взаємодіє з хмарними сервісами за допомогою інструментів Google Drive API v3 [11]. Логіка його функціонування включає кілька послідовних етапів. Спочатку модуль ініціює процедуру безпечної автентифікації через локально збережені токени доступу протоколу OAuth 2.0, а в разі закінчення терміну їхньої дії — в автоматичному режимі оновлює їх без залучення користувача. Далі, використовуючи можливості офіційних клієнтських бібліотек, модуль організує безпосередню відправку даних. Для великих файлів, які часто трапляються в мережевих сховищах, розробники передбачили логіку поблокового завантаження, що дозволяє передавати інформацію окремими послідовними частинами фіксованого

розміру. Це забезпечує високу стійкість процесу до випадкових обривів зв'язку, оскільки у разі тимчасової втрати мережевого з'єднання система здатна відновити завантаження саме з того місця, де воно було перерване, не починаючи передачу всього гігантського масиву спочатку.

Замикає архітектурну структуру системи модуль моніторингу, логування та сповіщень. Він функціонує паралельно з усіма іншими процесами, збираючи детальні текстові звіти про роботу кожного підрозділу програми, фіксуючи успішні операції завантаження, а також детально реєструючи будь-які помилки чи позаштатні ситуації. До таких ситуацій можна віднести відсутність доступу до певного локального файлу через його поточне блокування іншим офісним додатком або вихід за межі доступних квот вільного місця в хмарному профілі. Вся оброблена інформація ретельно заноситься в локальний журнал подій і одночасно може дублюватися через інтегровані канали зовнішніх сповіщень на електронну пошту або в корпоративний месенджер відповідальної особи. Такий всеохопний зворотний зв'язок дозволяє з мінімальними зусиллями контролювати працездатність системи, забезпечуючи її високу автономність, що є ідеальним рішенням для умов відсутності постійного IT-персоналу [27].

2.2 Теоретичні засади та обґрунтування вибору алгоритмів машинного навчання для визначення пріоритетності інформаційних масивів

Організація безперебійного захисту даних у межах локальної мережі невеликого підприємства часто стикається із жорсткими інфраструктурними обмеженнями, серед яких найгострішими є лімітована пропускна здатність інтернет-каналу та обмежене часове вікно, виділене для резервного копіювання. За умов, коли обсяги комерційної інформації постійно зростають,

спроба одночасного відправлення всіх модифікованих файлів у хмарне сховище Google Drive може повністю паралізувати роботу мережі у робочі години або призвести до зриву графіка оновлень. Для вирішення цієї проблеми виникає об’єктивна необхідність інтеграції в систему інтелектуального механізму динамічної пріоритезації трафіку. Замість застарілих підходів, що базуються на статичних правилах, сучасна архітектура може ефективно спиратися на методи машинного навчання, які здатні самостійно аналізувати патерни поведінки користувачів і автоматично виділяти найбільш критичні об'єкти для першочергового збереження [21, 24].

Теоретичний фундамент побудови такої моделі покладається на задачу багатокласової класифікації або регресійного оцінювання, де кожному файлу в локальній мережі присвоюється певний ранг або числовий індекс значущості. Математичний апарат, що описує цей процес, зазвичай оперує багатовимірним простором ознак, які витягуються з метаданих файлової системи та журналів аудиту мережевої взаємодії. Модель розраховує інтегральний показник пріоритетності як складну функцію, яка зважує часові інтервали між останніми редагуваннями, глибину вкладеності директорії, тип розширення файлу, а також частоту звернень до нього з боку різних робочих станцій. Замість фіксованого підсумовування цих параметрів, алгоритм машинного навчання динамічно адаптує вагові коефіцієнти для кожної ознаки, враховуючи специфіку бізнес-процесів конкретної компанії, що дозволяє мінімізувати ймовірність помилкового заниження пріоритету важливих документів.

Під час аналізу потенційних аналітичних методів для реалізації цього функціоналу було розглянуто кілька популярних родин алгоритмів. Зокрема, лінійні моделі та логістична регресія приваблюють своєю низькою обчислювальною складністю, проте вони демонструють незадовільну точність через нездатність виявляти складні нелінійні залежності між ознаками, наприклад, коли комбінація певного розширення та специфічного часу модифікації різко підвищує цінність файлу. З іншого боку, нейромережеві

структури, попри свою високу гнучкість, вимагають надмірних обчислювальних ресурсів для навчання та тривалого часу збору репрезентативної вибірки, що є економічно та технічно неприйнятним для скромних апаратних потужностей серверів малого бізнесу [25].

Найбільш раціональним та обґрунтованим вибором для архітектури локальної системи бекапу є використання ансамблевих методів на основі дерев рішень, зокрема алгоритму випадкового лісу (Random Forest) або градієнтного бустингу. Логіка побудови дерева рішень чудово підходить для аналізу різнорідних метаданих, оскільки вона дозволяє одночасно працювати як із числовими параметрами, наприклад, розміром файлу у байтах чи кількістю днів від моменту створення, так і з категоріальними ознаками, такими як ім’я користувача-власника або мережева назва вузла. Під час формування структури кожного дерева алгоритм спирається на концепцію інформаційної ентропії, яка математично вимірює ступінь хаосу чи невизначеності у вибірці даних. Концептуальна логіка цього процесу передбачає, що система послідовно розщеплює весь масив локальних файлів на підгрупи за такими критеріями, які забезпечують максимальне очищення вибірки та найвищий приріст інформації, відокремлюючи застарілі тимчасові файли від активних баз даних чи фінансових звітів [3, 26].

Для того щоб уникнути проблеми перенавчання, коли поодиноке дерево занадто сильно підлаштовується під поточний стан файлової системи й втрачає здатність до узагальнення, ансамблевий підхід об’єднує десятки або сотні таких незалежних дерев у єдину систему. Підсумкове рішення про присвоєння файлу високого, середнього чи низького пріоритету приймається шляхом колективного голосування або усереднення ймовірностей усіх побудованих дерев. Це забезпечує колосальну стійкість моделі до шумів вхідних даних, що часто трапляються в локальних мережах через хаотичні дії персоналу. Завдяки такій математичній структурі, розроблюваний модуль пріоритезації не лише гарантує раціональне використання дискового простору на Google Drive, але й працює надзвичайно швидко, споживаючи мінімальну кількість оперативної

пам’яті сервера, що повністю задовольняє жорсткі експлуатаційні вимоги досліджуваного сегмента ринку.

2.3 Функціональна логіка та алгоритмічна модель процесу інтелектуального резервного копіювання.

Функціональна логіка спеціалізованого програмного комплексу «Reserve» спроєктована таким чином, щоб забезпечити повністю автономний, надійний та захищений процес резервного збереження інформації з інтеграцією елементів інтелектуального прийняття рішень, що максимально відповідає технічним та економічним реаліям функціонування локальних мереж у сегменті малого бізнесу. Весь технологічний життєвий цикл захисту даних можна розділити на кілька послідовних взаємопов'язаних етапів.

Першочерговим етапом виступає гнучке конфігурування параметрів роботи системи користувачем або мережевим адміністратором через інтуїтивно зрозумілий графічний інтерфейс додатка. На цьому кроці чітко визначаються конкретні цільові директорії та папки на локальних робочих станціях чи мережевих вузлах, які потребують постійного моніторингу та регулярного дублювання. Паралельно з цим налаштовується часовий розклад виконання регламентних операцій, що дозволяє запускати процес резервування автоматично у визначені години (наприклад, у періоди мінімального завантаження локальної мережі під час нічних змін) або з заданою циклічною періодичністю. Критично важливим аспектом безпеки на даному етапі є введення користувачем унікального текстового пароля для шифрування, який виступає основою для забезпечення абсолютної конфіденційності комерційної та фінансової інформації перед її відправкою у зовнішнє середовище. Програма автоматично записує всі ці конфігураційні параметри у стійке локальне сховище для подальшого використання під час

робочих сесій. Безпечна авторизація та встановлення довірчих відносин із віддаленою хмарною платформою Google Drive реалізується на базі застосування стандартного безпечного протоколу OAuth 2.0 [12], що повністю виключає ризик компрометації або перехоплення облікових даних користувача зловмисниками.

Після успішного завершення етапу первинної конфігурації та запуску відповідних фонових служб система переходить до фази безперервного моніторингу файлової системи та ретельного відбору об'єктів. Відповідно до встановлених часових правил планувальника або за безпосереднім ручним запитом користувача, спеціалізований модуль резервного копіювання, який у поточній версії програми взаємодіє з логікою графічної вкладки керування бекапом, формує точний перелік файлів, що підлягають поточній обробці. У повноцінній автоматизованій моделі функціонування в межах локальної мережі цей процес базується на фоновому скануванні виділених директорій, під час якого система здійснює глибоке порівняння динамічних метаданих файлів, включаючи точний час останньої модифікації, поточний розмір об'єкта в байтах та обчислені контрольні суми, із записами в історичній базі даних попередніх сесій. Такий підхід дозволяє безпомилково ідентифікувати лише нові або змінені документи, відсікаючи дублікати та оптимізуючи трафік.

Наступним кроком, який передує безпосередньому початку передачі даних, є інтелектуальний аналіз та динамічна пріоритезація сформованої черги, що стає можливим за умови повної інтеграції та належного налаштування аналітичного компонента машинного навчання. Спеціалізований модуль прогнозування та оцінки черги приймає на вхід сформований вектор ознак, який відображає поточні характеристики об'єктів та стан сховища, на основі чого розраховує пріоритетність кожного завдання. У поточній архітектурній реалізації, якщо навчена математична модель не була завантажена в пам'ять сервера, система автоматично переходить на використання детермінованого логічного правила, згідно з яким за наявності хоча б одного модифікованого об'єкта в черзі процес резервування визнається

необхідним до виконання. У перспективній розгорнутій версії комплексу результати роботи аналітичної моделі здатні безпосередньо впливати на ранжування файлів у черзі завантаження, дозволяючи першочергово відправляти критично важливі для бізнесу бази даних, відкладаючи другорядну інформацію на періоди низької активності мережі.

Усі відібрані та впорядковані за пріоритетом файли передаються на етап низькорівневої обробки, яка включає послідовне стиснення та стійке криптографічне кодування інформації. Ці ресурсомісткі обчислювальні операції виконуються в ізольованому фоновому потоці робочого модуля системи. Кожен окремий файл або сформований пакет даних піддається високоефективній компресії за допомогою алгоритмів, що підтримуються спеціалізованою бібліотекою роботи з архівами, де найчастіше застосовуються модифікації методів Лемпеля-Зіва-Маркова для максимального зменшення фізичного обсягу даних. Одразу після створення компактного архіву бінарний потік обробляється симетричним алгоритмом шифрування AES із довжиною ключа 256 біт [10]. Криптографічні перетворення спираються на наданий користувачем пароль, що гарантує надійний захист інформаційних масивів від несанкціонованого доступу чи аналізу навіть у разі повної компрометації облікового запису на публічному хмарному сховищі.

Фінальним етапом функціонального циклу є безпосереднє транспортування підготовлених зашифрованих пакетів у віддалене хмарне середовище. Робочий модуль системи ініціює захищене мережеве з’єднання з програмним інтерфейсом Google Drive API [11], використовуючи збережені та автоматично оновлювані токени авторизації, після чого здійснює завантаження архівів у строго визначені каталоги на віддаленому диску. Програма здійснює безперервний покроковий моніторинг статусу передачі кожного інформаційного блоку. У разі успішного завершення операції система фіксує транзакцію у внутрішній базі даних, записуючи точний час завершення та унікальний ідентифікатор файлу, присвоєний хмарою. Якщо ж під час передачі виникають випадкові мережеві збої, втрати пакетів або тимчасові обриви

зв'язку, що є поширеною проблемою для локальних мереж малого бізнесу, архітектура передбачає активацію відмовоустійливого механізму повторних спроб, який намагається відновити завантаження кілька разів із прогресивною затримкою перед тим, як зафіксувати критичну помилку.

Протягом усього описаного життєвого циклу програмний комплекс підтримує безперервний та прозорий зворотний зв’язок із користувачем за допомогою елементів графічного інтерфейсу, де в реальному часі відображаються індикатори прогресу виконання операцій, текстові повідомлення про успішне збереження даних або деталізовані сповіщення про виникнення позаштатних ситуацій. Паралельно з цим у фоновому режимі реалізовано всеохопне логування роботи всіх підсистем за допомогою стандартного системного модуля реєстрації подій [1]. Накопичення цих діагностичних даних у текстових журналах дозволяє оперативно виявляти приховані помилки конфігурації, аналізувати історію проведених операцій та підтримувати високий рівень автономності системи. Таким чином, архітектурна логіка «Reserve» успішно об'єднує гнучкі методи моніторингу локальної мережі, глибоку попередню оптимізацію і захист даних та надійні алгоритми хмарної інтеграції з можливістю подальшого масштабування в бік інтелектуального управління процесами.

2.4 Компонентна структура та ергономіка графічного інтерфейсу користувача

Графічний інтерфейс користувача (ГІК) програмного комплексу «Reserve» спроєктовано та реалізовано на базі передової бібліотеки інструментів PyQt6. Головним пріоритетом при розробці візуального шару додатку стало надання кінцевому користувачу максимально простої, прозорої та інтуїтивно зрозумілої оболонки для взаємодії з системою. Це критично

важливо для представників малого бізнесу, персонал яких найчастіше не володіє глибокими спеціалізованими технічними знаннями або досвідом у сфері системного адміністрування локальних мереж [27]. Вся логіка побудови візуальних форм фокусується на полегшенні процесів первинного конфігурування ключових параметрів роботи додатка та забезпечення безперервного, наочного моніторингу поточного стану резервного збереження файлів.

Головне вікно програми має чітко структуровану двокомпонентну архітектуру, що включає лінійну бічну панель навігації та велику центральну область контенту, призначену для динамічного відображення вмісту обраної вкладки. Бічна панель навігації виконує роль головного координаційного вузла керування, де розташовані інтерактивні елементи для миттєвого перемикання між основними функціональними розділами системи. Для підвищення рівня ергономічності та зручності сприйняття всі навігаційні кнопки забезпечені механізмами динамічного візуального відгуку, що змінюють свій графічний стиль при наведенні курсору миші або примусово виділяються кольором, коли відповідна вкладка стає активною. Додатково на цій панелі розміщено фірмову графічну іконку додатка, яка робить інтерфейс більш впізнаваним.

Серед основних навігаційних напрямків бічної панелі виділяється інфо-панель, яка слугує стартовим екраном додатка і транслює загальну інформацію про стан системи, а також містить велику інтерактивну кнопку швидкого переходу, що миттєво перенаправляє користувача до інструментів створення резервної копії. Наступним важливим елементом панелі є розділ деталей машинного навчання, який відповідає за виведення аналітичної інформації про роботу інтелектуального модуля прогнозування. У поточному стані системи ця область відображає текстовий статус працездатності предиктора, інформуючи оператора про те, що модель є порожньою, якщо відповідні математичні ваги ще не були завантажені в пам'ять додатка. Замикає навігаційний блок кнопка переходу до глобальних налаштувань програми, яка відкриває доступ до системних параметків конфігурації комплексу.

Центральна область контенту динамічно змінюється залежно від обраного розділу на бічній панелі. Основним робочим простором для виконання операцій захисту даних є спеціалізована вкладка резервного копіювання. Вона об’єднує комплекс візуальних інструментів, розроблених для спрощення вибору та підготовки файлів. Для навігації локальною файловою системою та мережевими каталогами у цю вкладку інтегровано зручний провідник, реалізований у вигляді деревоподібної структури елементів, яка дозволяє наочно переглядати папки та покроково вибирати необхідні об'єкти для дублювання. Всі відібрані користувачем елементи миттєво додаються до суміжного графічного списку вибраних файлів, який підтримує динамічне візуальне відображення черги і в перспективі орієнтований на інтеграцію зручної технології перетягування об'єктів мишею безпосередньо з вікон операційної системи.

Для гарантування повної конфіденційності бізнес-інформації під час її перебування у хмарі, безпосередньо під списком файлів розташоване захищене текстове поле введення пароля шифрування, де відображувані символи автоматично маскуються для захисту від сторонніх очей. Тут же розміщено розвинений блок налаштування часового розкладу автоматизації, який пропонує гнучкий вибір між використанням інтерактивного календаря для точного визначення дати виконання бекапу та створенням унікального користувацького графіка із деталізацією періодичності, конкретних днів тижня та точного часу запуску фонового процесу. Нижню частину робочої області займає графічний індикатор прогресу, який у реальному часі відображає відсотковий стан виконання поточної операції, а керування всім циклом здійснюється за допомогою двох функціональних кнопок, призначених для миттєвого старту копіювання або екстреного скасування поточного процесу.

У разі активації розділу глобальних налаштувань, центральна область інтерфейсу відображає спеціалізовану вкладку конфігурації системних параметрів програми. Головним елементом тут виступає динамічний індикатор статусу авторизаційного токена Google Drive, який дозволяє оператору з

першого погляду визначити, чи є активним поточне з'єднання з хмарною платформою. Спеціально для підвищення комфорту роботи користувачів у різних умовах освітлення, у вкладку інтегровано блок перемикання візуальних тем оформлення, що дозволяє за допомогою графічних кнопок із тематичними іконками миттєво активувати світлу, темну або системну кольорову палітру додатка.

Рівень автоматизації та додаткові параметри функціонування регулюються за допомогою набору інтерактивних прапорців-чекбоксів, які керують параметрами автоматичного запуска програми при старті операційної системи, можливістю тимчасового повного вимкнення аналітичного модуля машинного навчання для заощадження ресурсів процесора, а також деактивацією спливаючих системних сповіщень. Вибір мови відображення всіх текстових елементів реалізовано через зручне випадаюче меню, яке наразі підтримує перемикання між українською та англійською локалізаціями. Для вирішення потенційних проблем із мережевою автентифікацією у нижній частині вкладки передбачено спеціальні кнопки безпосереднього управління токеном доступу, які дозволяють в ручному режимі примусово оновити сесію зв'язку або повністю скинути поточні авторизаційні дані для зміни облікового запису хмари.

Окрім вкладкових компонентів, графічна оболонка системи містить низку наскрізних загальних елементів, які підвищують рівень інформативності та автономності додатка. У самій верхній частині головного вікна розташований яскравий інформаційний банер статусу токена, який здійснює безперервний фоновий моніторинг зв'язку з Google Drive та динамічно змінює свій фоновий колір залежно від стану підключення, миттєво сигналізуючи про виникнення помилок авторизації. Для мінімізації візуального навантаження на робочий стіл користувача додаток підтримує можливість повного згортання в системний трей операційної системи, де відображається компактна інтерактивна іконка програми, що відкриває швидкий доступ до контекстного меню з командами розгортання вікна або повного виходу з додатка.

Будь-які інформаційні повідомлення, попередження про критичне заповнення хмарного диска або сповіщення про успішне завершення тривалого сеансу резервного копіювання виводяться за допомогою стандартних системних діалогових вікон, які тимчасово фокусують на собі увагу користувача до моменту ознайомлення з ними. Завдяки повній підтримці мовної локалізації та динамічній адаптації кольорів під обрану тему, візуальна система програми забезпечує максимальний комфорт при щоденному використанні, а її внутрішня архітектура спроєктована за модульним принципом, що гарантує легке розширення інтерфейсу та додавання нових керуючих елементів у майбутніх версіях програмного комплексу.

3 ТЕХНІЧНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА АПРОБАЦІЯ ПРОТОТИПУ СИСТЕМИ АВТОМАТИЗОВАНОГО РЕЗЕРВНОГО ЗБЕРЕЖЕННЯ ДАНИХ

3.1 Програмна реалізація компонента взаємодії з програмним інтерфейсом Google Drive API

Побіжний аналіз архітектури системи показує, що побудова надійного, стабільного та повністю захищеного каналу інформаційного обміну з хмарним сервісом Google Drive є одним із фундаментальних стовпів, на яких тримається вся життєдіяльність програмного комплексу «Reserve». Технічна реалізація цього інтеграційного шлюзу повністю покладається на офіційну клієнтську бібліотеку Google API для мови Python. Вибір такого інструментарію дозволяє розробнику абстрагуватися від низькорівневих маніпуляцій з мережевими сокетами та використовувати перевірені часом, оптимізовані методи взаємодії з віддаленою хмарною інфраструктурою, адаптовані під умови локальних мереж невеликих підприємств.

Початок будь-якої сесії взаємодії та верифікація повноважень додатка в межах мережі починається з процедури автентифікації користувача, яка суворо відповідає сучасному криптографічному протоколу OAuth 2.0. Для ініціалізації цього процесу програмна система зчитує конфігураційні параметри та унікальні ідентифікатори клієнта, які заздалегідь експортуються з консолі розробника Google і зберігаються у локальному захищеному файлі облікових даних у форматі JSON. Коли утиліта запускається вперше, або у випадку, коли діючі повноваження з якихось причин анулюються чи відкликаються, запускається тригер перенаправлення. Програма викликає системний веб-браузер і відкриває офіційну сторінку авторизації Google, де

власник облікового запису наочно бачить перелік запитуваних прав доступу до свого дискового простору та підтверджує легітимність дій програми.

Після успішного проходження верифікації віддалені сервери автентифікації повертають набір криптографічних маркерів, що складається з короткострокового токена доступу (access token) та довгострокового токена оновлення (refresh token). Щоб забезпечити збереження стану між перезапусками програми та уникнути повторного виклику вікна браузера при кожній спробі копіювання файлів із локальної мережі, ці об'єкти піддаються процедурі бінарної серіалізації. Програмний код використовує можливості стандартного модуля серіалізації Python для трансформації складних об'єктів у послідовність байтів, яка записується у локальний захищений файл токена.

Наявність у цій структурі токена оновлення є критично важливою умовою для забезпечення повної автономності системи в межах локальної мережі малого підприємства. Завдяки йому керуюче ядро додатка здатне самостійно, без залучення людського фактора чи будь-яких додаткових сповіщень, виконувати фонові запити на оновлення поточної робочої сесії, що гарантує безперебійне виконання нічних або регламентних завдань копіювання. Організація логіки первинного виклику авторизаційного вікна та керування життєвим циклом отриманих маркерів на рівні графічного інтерфейсу розподілена між програмними файлами, що відповідають за візуальну панель налаштувань та загальну координацію інтерфейсних форм.

Безпосереднє виконання рутинних операцій з інформаційними потоками та взаємодія з методами програмного інтерфейсу хмари зосереджені в ізольованому робочому класі BackupWorker, який функціонує в окремому потоці виконання для запобігання блокуванню графічної оболонки і розташований у файлі робочого модуля бекапу. Для надсилання підготовлених зашифрованих архівів у хмару алгоритм створює спеціалізований об'єкт медіа-завантаження MediaFileUpload, який виступає абстракцією над локальним файлом і керує його зчитуванням. Безпосередня реєстрація нового об'єкта в хмарному сховищі та передача його бінарного вмісту виконуються через

виклик базового методу створення файлів files().create(), інтегрованого у клієнтську бібліотеку Google Drive API. Цей інструментарій демонструє однаково високу ефективність як при обробці невеликих конфігураційних файлів, так і при транспортуванні гігантських багатотомних архівів, що містять резервні копії баз даних локальної мережі. Крім того, логіка цього модуля передбачає можливість автоматичного створення нових цільових каталогів на віддаленому диску. Це дозволяє гнучко структурувати копії за датами, назвами робочих станцій або типами інформаційних джерел, забезпечуючи високий рівень порядку у хмарному сховищі та спрощуючи подальший пошук файлів у разі потреби аварійного відновлення.

Особливу увагу при проєктуванні цього програмного вузла було приділено розробці систем обробки виняткових ситуацій та логування позаштатних подій, оскільки стабільність інтернет-каналів у локальних мережах малих компаній часто буває нестабільною. Незважаючи на те, що в поточному прототипі програми може не використовуватися явне підключення спеціалізованих зовнішніх бібліотек для автоматичного повторення запитів, стійкість системи досягається за рахунок каскадного перехоплення стандартних мережевих та API-винятків. Алгоритм ретельно аналізує коди помилок, що повертаються хмарою — від банальної тимчасової відсутності зв'язку до досягнення лімітів на кількість запитів за хвилину (перевищення квот). Усі зафіксовані аномалії негайно реєструються у текстовому журналі подій із зазначенням точного часу та причин збою, а інформація про критичні помилки, які унеможливлюють продовження завантаження, миттєво транслюється на графічну панель користувача для оперативного прийняття рішень.

Логіку функціонування ключового алгоритму завантаження файлів у межах робочого потоку класу BackupClass, рис.3.1, можна описати як чітку послідовність програмних дій, що замінює собою пряме виведення коду. Спочатку метод приймає параметри локального шляху до зашифрованого архіву та ідентифікатор цільової папки у хмарі.

```
def upload_local_resource(self, target_path: str, meta_config: dict) -> str:
    """Виконує завантаження об'єкта до хмарного сховища з валідацією відповіді."""
    try:
        payload = MediaFileUpload(filename=target_path, resumable=True)

        request = self.drive_service.files().create(
            body=meta_config,
            media_body=payload,
            fields='id'
        )

        response = request.execute()

        if not (resource_id := response.get('id')):
            raise KeyError("У відповіді API відсутній унікальний ідентифікатор файлу.")

        logging.info(f"Синхронізацію завершено успішно. Хмарний ID: {resource_id}")
        return resource_id

    except HttpError as api_error:
        logging.error(f"Помилка взаємодії з Drive API: {api_error}")
        raise
    except Exception as e:
        logging.error(f"Непередбачувана помилка під час завантаження: {e}")
        raise
```

Рисунок 3.1 – Ключовий фрагмент коду BackupClass для завантаження файлів на Google Drive

На основі цих даних всередині класу формується словник метаданих, де вказується оригінальне ім'я файлу та масив батьківських папок для правильного позиціонування об'єкта. Далі алгоритм ініціалізує об'єкт завантаження медіафайлу, передаючи йому локальний шлях та тип вмісту, і встановлює прапорець поблокової передачі (resumable), що дозволяє надсилати дані частинами. Після цього програма викликає метод створення файлу, запускаючи циклічний процес відправки HTTP-пакетів. У межах цього циклу система безперервно зчитує фіксовані блоки бінарних даних із диска та транслює їх на віддалений сервер, паралельно оновлюючи стан графічного індикатора прогресу для інформування користувача. Після успішного завершення передачі останнього блоку метод перехоплює фінальну відповідь сервера, витягує з неї унікальний ідентифікатор створеного в хмарі файлу і повертає його керуючому ядро для успішного закриття поточної транзакції. Такий комплексний підхід до побудови програмного модуля взаємодії гарантує формування надійного, гнучкого та високопродуктивного фундаменту для безпечного збереження комерційної інформації у віддаленому сховищі.

3.2 Особливості програмного моделювання та практичного впровадження алгоритмів розумного збереження даних

Технічний фундамент функціоналу резервного копіювання в архітектурі «Reserve» тримається на спеціалізованому робочому класі, який бере на себе виконання всіх найбільш ресурсомістких та обчислювально складних операцій, рис.3.2.

```
def create_secure_backup_vault(self, destination_archive: str) -> None:
    vault_path = Path(destination_archive)

    try:
        with py7zr.SevenZipFile(vault_path, mode='w', password=self.password) as archive_stream
            for item in self.files_to_backup:
                file_node = Path(item)

                if not file_node.is_file():
                    logging.warning(f"Ресурс не знайдено або пошкоджено, пропуск: {file_node}")
                    continue

                archive_stream.write(file_node, file_node.name)

        logging.info(f"Резервний контейнер успішно згенеровано та зашифровано: {vault_path.reso

    except Exception as error:
        logging.error(f"Не вдалося створити захищений архів: {error}")
        raise
```

Рисунок 3.2 – Реалізація процесу архівації та шифрування за допомогою py7zr у BackupClass

Щоб забезпечити плавне та безперебійне функціонування графічного інтерфейсу користувача й уникнути його критичного зависання під час обробки великих масивів мережевих даних, цей компонент винесено в окремий, ізольований потік виконання за допомогою системних інструментів організації черги та пулу робочих потоків. Робочий модуль приймає на вхід чітко сформований перелік файлів, відібраних для резервування в локальній мережі, а також криптографічний пароль користувача. Для підтримки безперервного та прозорого зв'язку з головним вікном програми цей модуль використовує систему асинхронного сигнального сповіщення. Через спеціальні програмні канали він вчасно транслює сигнали поточного

оновлення прогресу, успішного завершення операції або фіксації помилки, що дозволяє графічній оболонці миттєво реагувати на стан фонового процесу та інформувати користувача.

Безпосередня низькорівнева обробка інформаційних джерел у межах робочого потоку розділена на два фундаментальні та послідовні етапи, що включають глибоку компресію та надійне криптографічне кодування. Для створення стиснених пакетів розробники задіяли можливості спеціалізованої архіваційної бібліотеки, яка забезпечує пакування об'єктів у високоефективному бінарному форматі сім-зіп. Цей інструментарій дозволяє не лише суттєво зменшити фізичний розмір вихідних бізнес-документів для економії пропускної здатності інтернет-каналу локальної мережі, а й одночасно активувати алгоритми стійкого симетричного захисту за стандартом AES із довжиною ключа 256 біт на основі пароля, наданого користувачем. Важливим архітектурним рішенням є динамічна генерація імені кожного створюваного архіву. Програма в автоматичному режимі додає до назви файлу точну часову мітку його створення, що гарантує абсолютну унікальність об'єктів у сховищі та закладає надійний технологічний фундамент для реалізації повноцінного версіонування та можливості відновлення даних на конкретний момент часу в минулому.

Інтелектуальна аналітична складова програмного комплексу представлена виділеним фоновим модулем прогнозування, який розташований у файлі предиктора машинного навчання. У поточній версії прототипу реалізовано гнучку логіку захисту від збоїв: якщо заздалегідь навчена математична модель з якихось причин не була завантажена зі свого серіалізованого бінарного файлу, система автоматично переходить на використання детермінованого логічного правила. Відповідно до цього текстового правила, алгоритм аналізує вхідний вектор ознак файлової системи, і якщо кількість виявлених об'єктів, що очікують у черзі на копіювання, є більшою за нульове значення, модуль повертає позитивне логічне твердження про доцільність запуску процедури бекапу. Цей інтелектуальний фільтр

викликається безпосередньо перед стартом основних обчислювальних процесів і слугує для прийняття стратегічного рішення про активацію сесії. Подальший вектор модернізації архітектури передбачає збір розширеної статистики та навчання складніших ансамблевих моделей для реалізації тонкої пофайлової пріоритезації мережевого трафіку.

На поточному етапі розвитку системи логіка визначення джерел для копіювання покладається на дії самого користувача, який здійснює ручний вибір необхідних директорій та документів у відповідній вкладці графічного інтерфейсу управління бекапом. Користувач може самостійно вказати критично важливі вузли та папки в локальній мережі, які потребують захисту. Водночас перехід від ручного керування до повністю автономного фонового моніторингу файлової системи, реалізація автоматичного відстеження модифікацій у реальному часі та впровадження чистих алгоритмів інкрементного копіювання визначені розробниками як найбільш пріоритетні та невідкладні напрямки для подальшого комплексного вдосконалення цього програмного забезпечення.

3.3 Програмна розробка та практичне втілення компонентів графічного інтерфейсу

Практична розробка візуальної архітектури графічного інтерфейсу користувача для програмного комплексу «Reserve» повністю спирається на можливості сучасного об'єктно-орієнтованого фреймворку PyQt6. Центральним елементом цієї підсистеми є головне вікно програми, програмна логіка якого детально описуваного у модулі візуального відображення. Візуальна оболонка має класичну збалансовану структуру, яка розділяє робочий простір на ергономічну лінійну бічну панель навігації та велику центральну область контенту, призначену для динамічного виведення вмісту

активної вкладки. Для забезпечення безшовної та швидкої зміни екранних форм розробники застосували спеціалізований метод динамічного перемикання вмісту. Цей механізм дозволяє миттєво оновлювати складні віджети у межах єдиного віконного простору без створення надлишкових процесів чи вікон операційної системи, що гарантує високу швидкість відгуку програми та низьке споживання оперативної пам'яті комп'ютера. Загальна композиція елементів керування, архітектурне компонування та стартовий вигляд вікна наочно представлені на рис. 3.3, який демонструє взаємодію між навігаційним та операційним блоками додатка.

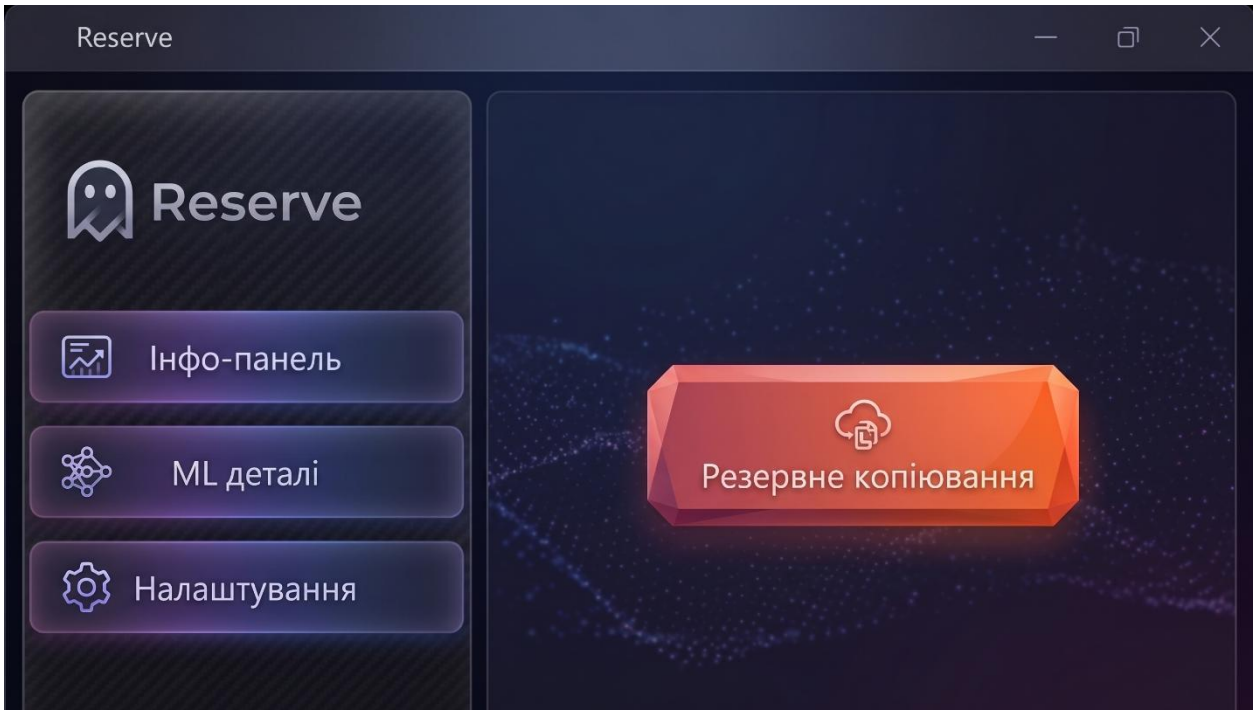


Рисунок 3.3 — Загальний вигляд центрального інтерфейсного простору та навігаційної структури додатка Reserve

Основним робочим простором, де користувач проводить більшість часу під час налаштування захисту даних локальної мережі, є спеціалізована вкладка резервного копіювання, логіка якої інкапсульована у відповідному програмному файлі. Ця панель об'єднує в собі інструменти для навігації, вибору джерел та контролю за поточними операціями. Візуалізація локальних дисків та підключених мережевих каталогів реалізована за допомогою

ієрархічного дерева файлової системи, яке забезпечує зручний покроковий огляд структури папок та дозволяє прапорцями відзначати необхідні для копіювання об'єкти. Усі відібрані файли та директорії негайно транслюються у суміжний графічний список, який наочно відображає поточну чергу завдань, що очікують на обробку.

Безпосередні під цими елементами розташоване поле текстового введення секретного пароля, яке автоматично маскує введені символи крапками, захищаючи криптографічний ключ від візуального перехоплення сторонніми особами під час введення. Налаштування часових регламентів автоматизації реалізовано через комбінацію інтерактивного віджета вибору дати для календарного режиму одноразових або стартових завдань та набору випадаючих списків для формування власного циклічного розкладу, де користувач може гнучко встановити періодичність, конкретні дні тижня та точний час запуску фонової служби, рис.3.4.

```
def _setup_navigation_panel(self) -> QGroupBox:
    panel_title = get_translation("File Explorer", self.lang)
    container = QGroupBox(panel_title)

    self.file_model = QFileSystemModel()
    self.file_model.setRootPath("")

    self.file_tree = QTreeView()
    self.file_tree.setModel(self.file_model)

    layout = QVBoxLayout()
    layout.addWidget(self.file_tree)
    container.setLayout(layout)

    return container

def _setup_queue_panel(self) -> QGroupBox:
    container = QGroupBox(get_translation("Selected Files", self.lang))

    self.selected_list = QListWidget()

    action_trigger = QPushButton(text=get_translation("Add Selected", self.lang))
    action_trigger.clicked.connect(self.add_selected_files)

    layout = QVBoxLayout()
    for widget in (self.selected_list, action_trigger):
        layout.addWidget(widget)

    container.setLayout(layout)
    return container

self.file_explorer_group = self._setup_navigation_panel()
self.selected_files_group = self._setup_queue_panel()
```

Рисунок 3.4 – Реалізація віджетів для вибору файлів у файлі

Поточний хід виконання ресурсомістких операцій стиснення, шифрування та передачі даних у реальному часі візуалізується за допомогою графічного прогрес-бару, який асинхронно оновлює своє положення на основі сигналів із робочого потоку, а безпосереднє оперативне керування сесією здійснюється за допомогою функціональних кнопок запуску та екстреного скасування процесу копіювання, рис.3.5.



Рисунок 3.5 — Панель керування резервним копіюванням

Для конфігурування загальносистемних параметрів та керування глобальними політиками безпеки в інтерфейсії передбачено вкладку налаштувань, яка програмно керується окремим модулем конфігурації. Цей розділ відкриває доступ до інструментів адаптації програми під потреби конкретного підприємства та адміністрування хмарної інтеграції. Зокрема, тут розташований блок кнопок для швидкого перемикання візуальних тем оформлення, набір інтерактивних прапорців для активації автоматичного запуску програми при старті операційної системи, тимчасового вимкнення аналітичного компонента машинного навчання з метою заощадження обчислювальних ресурсів процесора та деактивації системних сповіщень.

Вибір поточної мови інтерфейсу здійснюється через зручне випадаюче меню, яке підтримує миттєву зміну текстових описів. Особливе місце на цій панелі займають елементи безпосереднього управління токеном авторизації Google Drive, представлені кнопками примусового оновлення робочої сесії та повного скидання поточних повноважень, що необхідно для зміни облікового запису хмари. Суміжний графічний індикатор статусу токена в асинхронному режимі візуально інформує користувача про поточний стан підключення до віддаленої платформи, змінюючи свій вигляд залежно від легітимності авторизаційних маркерів. Детальний розподіл керуючих елементів цієї панелі та інтерфейс адміністрування зв'язку з хмарою відображено на рис.3.5.

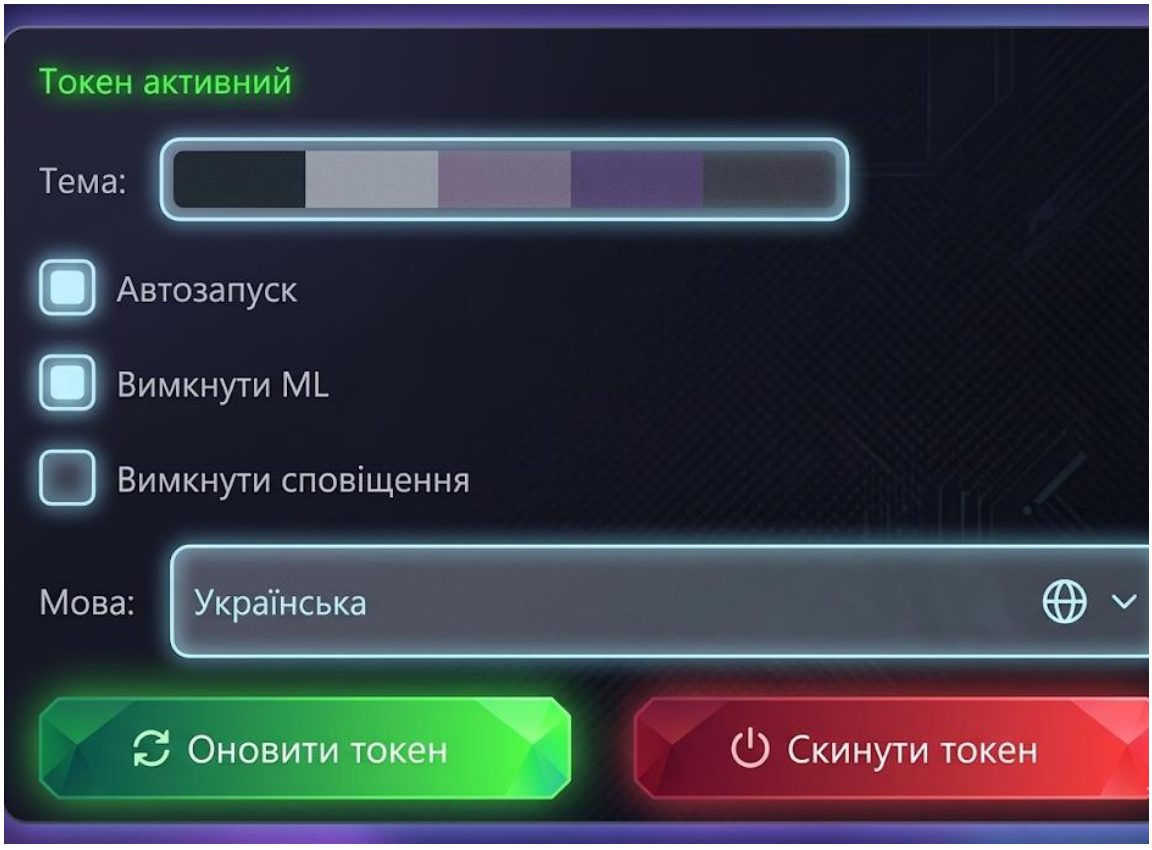


Рисунок 3.5 — Панель конфігурування глобальних параметрів та адміністрування хмарних токенів безпеки

Високий рівень гнучкості та відповідність сучасним стандартам розробки програмного забезпечення досягається завдяки чіткому розділенню логіки інтерфейсу та шару представлення даних. Повна підтримка двомовної

локалізації, що включає українську та англійську мови, реалізована за допомогою виділеного модуля перекладів, який містить словникові карти для всіх текстових написів, заголовків та системних повідомлень додатка. Адаптація кольорової схеми графічних елементів під обрану користувачем світлу, темну або системну тему здійснюється через спеціалізований палітурний модуль, який динамічно перераховує та застосовує стилі оформлення до кожного віджета. Надійний зворотний зв’язок із користувачем та миттєве інформування про хід виконання завдань, успішне завершення тривалого бекапу або виникнення критичних мережевих збоїв реалізовано через виклики стандартних модальних діалогових вікон повідомлений, а також за допомогою безперервного оновлення кольору та текстового вмісту верхнього банера статусу токена, що гарантує максимальний контроль над працездатністю всієї системи резервного копіювання.

3.4 Комплексне дослідження показників швидкодії, відмовоустійливості та ефективності програмного комплексу

Для комплексного оцінювання практичної ефективності, стабільності та відмовоустійливості розробленого програмного комплексу «Reserve» було організовано та проведено серію детальних експериментальних випробувань. Експериментальна апробація здійснювалася на базі уніфікованої робочої станції, функціонуючої під управлінням операційної системи Windows 11, оснащеної багатоядерним центральним процесором Intel Core i5 та 8 ГБ оперативної пам'яті. Увесь програмний каркас системи розгортався в інтерпретованому середовищі версії Python 3.10. Сформований тестовий набір даних повністю імітував реальну структуру цифрових активів невеликого підприємства, охоплюючи різноманітні комерційні документи, текстові структури, табличні масиви, графічні зображення та цифрові архіви. Загальний

фізичний обсяг та кількісні характеристики цих інформаційних джерел суворо відповідали параметрам.

Методологічний підхід до проведення тестування покладався на фіксацію часових інтервалів повного операційного циклу резервного збереження при послідовному масштабуванні кількості та обсягу вхідних даних. Паралельно здійснювався детальний розрахунок середнього коефіцієнта компресії, оцінювалися показники роботи аналітичного модуля, а також проводилася комплексна перевірка живучості системи за допомогою штучного моделювання критичних інфраструктурних збоїв. До таких деструктивних впливів відносилися раптові короткочасні розриви мережевого з'єднання з інтернетом та примусове аварійне припинення процесу програми безпосередньо в момент активної фази передачі бінарного потоку у хмару.

Глибокий аналіз накопичених статистичних даних наочно засвідчує, що загальна продуктивність програмного забезпечення, виміряна як середній час, необхідний для успішного копіювання одного файлового об'єкта, повністю вкладається в рамки нормативних вимог, встановлених для фонового режиму роботи систем автоматизації. У ході експерименту зафіксовано цілком закономірне та очікуване зростання тривалості обробки зі збільшенням загального обсягу та щільності файлової черги. Це явище безпосередньо пов'язане з прогресуючим навантаженням на дискову підсистему введення-виведення локального ПК, підвищенням утилізації ядер процесора під час криптографічних перетворень, а також обмеженнями пропускної здатності використовуваного мережевого каналу зв'язку.

Ефективність попередньої оптимізації даних за допомогою вбудованих засобів стиснення в середньому становить близько тридцяти відсотків, що дозволяє досягти відчутної економії дискового простору на віддалених серверах хмари. При цьому показник коефіцієнта стиснення демонструє значну динамічну варіативність залежно від внутрішньої структури файлів: офіційні текстові документи та електронні таблиці піддаються глибокій

компресії значно краще, ніж медійні файли або цифрові об'єкти, які вже містять у собі внутрішні алгоритми стиснення.

Оцінюючи показники відмовоустійливості, варто підкреслити, що програмний комплекс продемонстрував винятково високий рівень успішності повного завершення намічених транзакцій, навіть за умов інтенсивного зовнішнього моделювання інфраструктурних аварій. Спроєктовані механізми каскадного перехоплення виняткових ситуацій та асинхронне логування поточного стану дозволяють системі коректно відновлювати процес передачі з точки зупину. Поточна програмна реалізація аналітичного модуля машинного навчання на даному етапі розвитку прототипу є базовою, тому розгорнутий математичний аналіз її метрик точності не виступав основним завданням дослідження, проте сама наявність цього компонента закладає міцний архітектурний фундамент для побудови повноцінного інтелектуального управління процесами розподілу трафіку.

Основним технологічним обмеженням поточного функціонального прототипу є збереження ручного вектору вибору вихідних джерел користувачем та відсутність активованого режиму автоматичного повноцінного інкрементного копіювання. Це суттєво впливає на загальні часові витрати при проведенні повторних або циклічних сесій з великими інформаційними масивами, де модифікації зазнали лише поодинокі файли. Разом із цим, незважаючи на виявлені нюанси, система «Reserve» володіє значним внутрішнім потенціалом для подальшої глибокої оптимізації та розширення наявного інструментарію.

Одним із найбільш перспективних векторів підвищення загальної продуктивності та швидкодії є впровадження принципів глибокої асинхронності та розпаралелювання обчислень. Незважаючи на те, що системний пул потоків уже успішно керує ізольованою роботою фонового модуля бекапу, раціональним кроком є реалізація алгоритмів одночасного паралельного завантаження кількох незалежних файлових потоків у хмару, з урахуванням жорстких лімітів та квот, встановлених з боку Google Drive API

[11]. Додатково зменшити час очікування дозволить інтеграція асинхронного сканування локальних дисків.

Модернізація інтелектуальної аналітичної складової передбачає організацію збору якіснішої корпоративної статистики для формування репрезентативних навчальних вибірок, використання складніших ансамблевих алгоритмів, а також проведення інженерії нових, більш інформативних ознак файлової системи, таких як точна частота користувацького доступу до конкретного документа або його логічні зв'язки з іншими суміжними бізнес-проєктами. Важливим завданням також постає забезпечення щільнішої інтеграції прогнозних висновків моделі безпосередньо у внутрішню логіку ранжування та пріоритезації об'єктів у черзі завантаження.

Серед ключових архітектурних розширень функціоналу системи найбільш вагоме значення мають такі технологічні напрямки:

- Впровадження інкрементного та диференційного копіювання. Це дозволить кардинально скоротити загальний час виконання операцій та обсяги переданого мережевого трафіку за рахунок транспортування виключно нових або змінених фрагментів файлів. Реалізація цього підходу вимагатиме розгортання локальної бази даних SQLite для надійного збереження та швидкого зчитування метаданих про історичні стани об'єктів.
- Розробка підсистеми версійності даних. Надання користувачам гнучких інструментів для безперешкодного повернення та відновлення попередніх історичних станів комерційних файлів на певну глибину часу назад.
- Створення інтелектуального планувальника завдань. Розширення візуальних та програмних налаштувань розкладу для тонкої адаптації графіків під унікальні бізнес-процеси конкретного офісу.
- Розширення пулу підтримуваних хмарних сховищ. Інтеграція додаткових програмних інтерфейсів, таких як Microsoft Graph API чи Dropbox API, що суттєво розширить свободу вибору для кінцевих користувачів [17, 18].

- Проєктування вбудованого модуля декомпресії та відновлення. Створення зручного та зрозумілого графічного інтерфейсу для оперативного вилучення зашифрованих архівів із хмари та їх зворотного розгортання в локальній мережі безпосередньо через інтерфейс програми.

Подальше вдосконалення елементів взаємодії з користувачем може бути реалізоване шляхом додавання блоків розгорнутої графічної статистики, автоматичного генерування текстових звітів про виконані операції, а також впровадження інтерактивного майстра первинного налаштування для нових клієнтів. У сфері забезпечення абсолютної безпеки доцільно розглянути інтеграцію захищеніших методів системного зберігання локальних авторизаційних маркерів та токенів доступу з використанням вбудованих криптографічних контейнерів операційної системи.

ВИСНОВКИ

Комплексний аналіз проблематики збереження цифрових активів у локальних мережах малих підприємств підтвердив критичну необхідність розробки доступних, надійних та автоматизованих інструментів резервного копіювання. Дослідження існуючих на ринку апаратних та програмних підходів показало, що традиційні методи не повною мірою задовольняють специфічні вимоги малого бізнесу щодо балансу між фінансовими витратами, простотою експлуатації та захищеністю.

У результаті виконання роботи було спроєктовано та практично реалізовано програмний комплекс автоматизованого резервного збереження даних «Reserve». В основу системи покладено модульну клієнт-орієнтовану архітектуру, що забезпечує гнучкість, надійність та автономність роботи додатка. Використання програмного інтерфейсу Google Drive API (v3) виявилося найбільш збалансованим та економічно доцільним рішенням для створення хмарного мосту передачі інформації.

Розроблено та імплементовано механізми попередньої обробки даних, які включають глибоку компресію та стійке симетричне криптографічне шифрування алгоритмом AES із довжиною ключа 256 біт, що гарантує абсолютну конфіденційність комерційної інформації. Також було теоретично обґрунтовано доцільність застосування ансамблевих методів машинного навчання, а саме алгоритму випадкового лісу, для інтелектуальної динамічної пріоритезації файлів у черзі завантаження, що є критично важливим для мереж з обмеженою пропускною здатністю.

Експериментальна апробація прототипу на імітованих наборах даних підтвердила його працездатність, високу швидкість обробки інформації та стійкість до мережевих збоїв. Інвестиції в розробку та подальший розвиток системи є актуальними, а головними перспективами модернізації виступають впровадження алгоритмів повноцінного інкрементного копіювання,

розширення підтримуваних хмарних сховищ та створення системи контролю версій файлів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Microsoft. (n.d.). Microsoft Graph API Documentation. Retrieved from https://learn.microsoft.com/en-us/graph/

2 Acronis. (2023). Acronis Cyber Protect Report 2023 (Cyber Protection Week Global Report). Acronis.

3 Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

4 ENISA (European Union Agency for Cybersecurity). (n.d.). Threat Landscape Report. ENISA.

5 Google Developers. (n.d.). Using OAuth 2.0 to Access Google APIs. Retrieved from https://developers.google.com/identity/protocols/oauth2

6 Schmidt, K., Phillips, C., & Chuvakin, A. (2012). Logging and log management: The authoritative guide to understanding the concepts surrounding logging and log management. Newnes.

7 Grus, J. (2019). Data Science from Scratch: First Principles with Python (2nd ed.). O'Reilly Media.

8 Mell, P., & Grance, T. (2011). NIST Special Publication 800-145: The NIST Definition of Cloud Computing. National Institute of Standards and Technology.

9 Verizon Business. (2008). 2008 Data Breach Investigations Report. Retrieved from http://www.verizonbusiness.com/resources/security/databreachreport.pdf

10 Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

11 Jones, A., & Ashenden, D. (2005). Risk management for computer security: Protecting your network and information assets. Butterworth-Heinemann.

12 Amazon Web Services. (n.d.). Amazon S3 Documentation. Retrieved from https://docs.aws.amazon.com/s3/

13 Alpaydin, E. (2020). Introduction to Machine Learning (4th ed.). MIT Press.

14 Kissel, R. (2013). NIST Special Publication 800-88 Rev. 1: Guidelines for Media Sanitization. National Institute of Standards and Technology.

15 Google Developers. (n.d.). Google Drive API Overview. Retrieved from https://developers.google.com/drive/api/v3

16 Armbrust, M., et al. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50–58.

17 National Institute of Standards and Technology. (2012). NIST Special Publication 800-34 Rev. 1: Contingency Planning Guide for Federal Information Systems.

18 Erl, T., Mahmood, Z., & Puttini, R. (2013). Cloud Computing: Concepts, Technology & Architecture. Prentice Hall.

19 James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning: With Applications in R. Springer.

20 Whitman, M. E., & Mattord, H. J. (2019). Principles of Information Security (6th ed.). Cengage Learning.

21 Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.

22 Google. (n.d.). google-api-python-client Documentation. Retrieved from [https://google-api-python-client.readthedocs.io/en/latest/]()

23 Stallings, W., & Brown, L. (2018). Computer Security: Principles and Practice (4th ed.). Pearson.

24 Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

25 Dropbox. (n.d.). Dropbox API Documentation. Retrieved from [https://www.dropbox.com/developers/documentation/http/documentation]

26 Veeam Software. (2023). Veeam Data Protection Trends Report 2023. Veeam.

27 Linthicum, D. S. (2017). Cloud-Native Architectures: Design High-Availability and Cost-Effective Applications for the Cloud. O'Reilly Media.

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)