

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Хмарне сховище з наскрізним шифруванням»

Виконав студент групи ІПЗ-21-1
КОНДРАТЮК Олег Дмитрович

Керівник роботи:
ЛОКТИКОВА Тамара Миколаївна

Рецензент:
МОРОЗОВ Андрій Васильович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна ВАКАЛЮК

«17» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КОНДРАТЮК Олег Дмитрович**

Керівник роботи: **ЛОКТИКОВА Тамара Миколаївна**

Тема роботи: **«Хмарне сховище з наскрізним шифруванням»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: об'єктом дослідження є процес проектування та розробки безпечних хмарних сховищ, а також технології та алгоритми, що забезпечують наскрізне шифрування та конфіденційність даних користувачів, а предметом дослідження є методи реалізації систем хмарного зберігання з наскрізним шифруванням; конкретні криптографічні технології (AES, RSA, PBKDF2), програмні інтерфейси (Web Crypto API), архітектурні підходи та патерни проектування; а також аналіз їх ефективності для забезпечення безпеки, приватності та цілісності даних у порівнянні з традиційними рішеннями.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Локтікова Т.М.	17.02.2025р.	17.02.2025р.
2	Локтікова Т.М.	17.02.2025р.	17.02.2025р.
3	Локтікова Т.М.	17.02.2025р.	17.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебдодатку хмарного сховища та пояснювальної записки. Пояснювальна записка до випускної роботи містить 67 сторінок, 34 ілюстрацій та 13 таблиць.

Виконано розробку хмарного сховища з реалізацією наскрізного шифрування, яке забезпечить високий рівень безпеки та конфіденційності даних користувачів. Архітектура системи побудована таким чином, що всі криптографічні операції (шифрування та дешифрування файлів, імен файлів, ключів) виконуються виключно на стороні клієнта. Це гарантує, що серверна частина не має доступу до незашифрованих даних чи майстер-ключів користувачів, що мінімізує ризики витоку інформації.

Для реалізації використано сучасний технологічний стек: C++ з веб-фреймворком Drogon для розробки серверної частини та сервера зберігання статичних файлів, JavaScript (Vanilla JS) з Web Crypto API та бібліотеками aes.js/aesasm.js для клієнтської логіки та криптографії, та СУБД PostgreSQL для зберігання метаданих. Система підтримує шифрування файлів та шифрування імен файлів/папок. Також реалізовано безпечний спільний доступ до файлів та папок з передачею ключів дешифрування через URL-фрагмент.

Додаток має інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам легко реєструватися, автентифікуватися, завантажувати, скачувати, організовувати файли та папки, керувати спільним доступом, а також експортувати ключ відновлення та керувати сесіями.

КЛЮЧОВІ СЛОВА: БЕЗПЕКА ДАНИХ, НАСКРІЗНЕ ШИФРУВАННЯ, ХМАРНЕ СХОВИЩЕ, AES, C++, DROGON, JAVASCRIPT, PBKDF2, POSTGRESQL, RSA, WEB CRYPTO API.

					ІПЗ.КР.Б-121-25-ПЗ							
Змн.	Арк.	№ докум.	Підпис	Дата								
Розроб.		Кондратюк.О.Д.			Хмарне сховище з наскрізним шифруванням			Лім.	Арк.	Аркушів		
Керівник		Локтікова Т.М								3	67	
Н. контр.		Тичина О.П.										
Зав. каф.		Вакалюк Т.А.						ФІКТ Гр. ІПЗ-21-1				

ABSTRACT

The bachelor's thesis consists of the cloud storage web application and an explanatory note. The explanatory note to the final thesis contains 67 pages, 34 illustrations, and 13 tables.

A cloud storage system has been developed with the implementation of end-to-end encryption, ensuring a high level of security and confidentiality for user data. The system architecture is designed so that all cryptographic operations (encryption and decryption of files, file names, keys) are performed exclusively on the client-side. This guarantees that the server-side has no access to unencrypted user data or master keys, minimizing information leakage risks.

The project utilizes a modern technology stack: C++ with the Drogon web framework for the backend and static file storage server development, JavaScript (Vanilla JS) with the Web Crypto API and aes.js/aesasm.js libraries for client-side logic and cryptography, and PostgreSQL DBMS for metadata storage. The system supports file encryption and file/folder name encryption. Secure file and folder sharing is also implemented, with decryption keys transmitted via URL fragments.

The application features an intuitive interface that allows users to easily register, authenticate, upload, download, organize files and folders, manage sharing, as well as export their recovery key and manage sessions.

KEYWORDS: AES, C++, CLOUD STORAGE, DATA SECURITY, DROGON, END-TO-END ENCRYPTION, JAVASCRIPT, PBKDF2, POSTGRESQL, RSA, WEB CRYPTO API.

		Кондратюк.О.Д			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ХМАРНОГО СХОВИЩА З НАСКРІЗНИМ ШИФРУВАННЯМ	9
1.1. Постановка задачі	9
1.2. Аналіз аналогів програмного продукту.....	10
1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	15
Висновки до першого розділу	17
РОЗДІЛ 2. ПРОЄКТУВАННЯ ХМАРНОГО СХОВИЩА.....	19
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	19
2.2. Розробка бази даних системи	27
2.3. Проєктування та реалізація алгоритмів роботи системи.....	35
2.4. Реалізація функціоналу хмарного сховища	37
Висновки до другого розділу.....	42
РОЗДІЛ 3. ІНТЕРФЕЙС КОРИСТУВАЧА ТА ПОРЯДОК РОБОТИ З ХМАРНИМ СХОВИЩЕМ З НАСКРІЗНИМ ШИФРУВАННЯМ.....	43
3.1. Порядок встановлення та налаштування параметрів застосунку	43
3.2. Загальна структура інтерфейсу користувача	45
3.3. Порядок роботи користувача з системою	51
3.4. Порядок роботи гостя зі спільним вмістом.....	59
Висновки до третього розділу	61
ВИСНОВКИ.....	63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	65
ДОДАТКИ.....	67

		Кондратюк. О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

БД – база даних

ПЗ – програмне забезпечення

E2EE – наскрізне шифрування (End-to-End Encryption)

AES – просунутий стандарт шифрування (Advanced Encryption Standard)

RSA – алгоритм шифрування Рівста-Шаміра-Адлемана (Rivest-Shamir-Adleman)

PBKDF2 – функція походження ключа на основі пароля (Password-Based Key Derivation Function)

MAC – код автентифікації повідомлення (Message Authentication Code)

IV – ініціалізаційний вектор (Initialization Vector)

СУБД – система управління базами даних

HTTP – протокол передачі гіпертексту (HyperText Transfer Protocol)

JSON – об'єктний запис JavaScript (JavaScript Object Notation)

CRUD – основні операції, що використовуються для управління даними (Create, Read, Update, Delete)

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми пов'язана зі стрімким зростанням обсягів цифрової інформації та нагальною потребою користувачів у постійному та безпечному доступі до своїх файлів. Існуючі хмарні сховища, хоча й зручні, часто викликають занепокоєння щодо конфіденційності та безпеки даних, оскільки не завжди гарантують захист від несанкціонованого доступу чи використання інформації третіми сторонами, включаючи самих провайдерів послуг. Успішні хакерські атаки та витoki даних великих компаній підкреслюють ризики зберігання чутливої інформації на централізованих серверах без належного захисту. У зв'язку з цим зростає попит на прозорі та надійні рішення, що використовують наскрізне шифрування, де контроль над доступом до даних повністю залишається у користувача.

Метою випускної кваліфікаційної роботи бакалавра є розробка хмарного сховища з наскрізним шифруванням, яке забезпечить високий рівень безпеки, конфіденційності, масштабованості та зручності використання для зберігання, завантаження та обміну файлами. Ключовим аспектом є гарантія того, що ключі шифрування генеруються, зберігаються та використовуються виключно на стороні клієнта, унеможливлюючи доступ до незашифрованих даних з боку сервера.

Для досягнення цієї мети необхідно виконати такі **завдання**:

1. Провести аналіз існуючих хмарних сховищ, визначити їх переваги та недоліки, особливо в контексті безпеки та наскрізного шифрування.
2. Сформулювати функціональні та нефункціональні вимоги до системи, зосереджуючись на безпеці, приватності даних, надійності та зручності інтерфейсу.
3. Обґрунтувати вибір технологічного стеку та інструментів моделювання для реалізації проєкту (наприклад, C++, JavaScript, Web Crypto API, PostgreSQL, OpenSSL).
4. Розробити архітектуру системи, включаючи логічну та фізичну моделі,

ВИЗНАЧИТИ ОСНОВНІ КОМПОНЕНТИ ТА ЇХ ВЗАЄМОЛІЮ.

		Кондратюк О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

5. Спроекувати та описати ключові алгоритми роботи системи: реєстрації, автентифікації користувачів, генерації та управління ключами, наскрізного шифрування та дешифрування файлів.
6. Реалізувати програмний продукт відповідно до розробленої архітектури та алгоритмів.
7. Провести тестування розробленої системи для перевірки її функціональності, надійності та безпеки.

Об'єктом дослідження є процес проєктування та розробки безпечних хмарних сховищ, а також технології та алгоритми, що забезпечують наскрізне шифрування та конфіденційність даних користувачів.

Предметом дослідження є методи реалізації систем хмарного зберігання з наскрізним шифруванням; конкретні криптографічні технології (AES, RSA, PBKDF2), програмні інтерфейси (Web Crypto API), архітектурні підходи та патерни проєктування; а також аналіз їх ефективності для забезпечення безпеки, приватності та цілісності даних у порівнянні з традиційними рішеннями.

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ХМАРНОГО СХОВИЩА З НАСКРІЗНИМ ШИФРУВАННЯМ

1.1. Постановка задачі

Сучасний світ характеризується експоненційним зростанням обсягів інформації. У таких умовах виникає гостра потреба у надійних та завжди доступних засобах для зберігання файлів, які б не обтяжували пам'ять персональних пристроїв користувачів. Хмарні сховища стали фактичним стандартом для вирішення цієї проблеми, пропонуючи зручний доступ до даних з будь-якої точки світу.

Проте, разом зі зручністю постають серйозні питання щодо конфіденційності та безпеки. Чи можна повністю довіряти великим ІТ-корпораціям, які не завжди прозоро описують механізми роботи своїх сервісів та потенційно можуть аналізувати або використовувати дані користувачів у власних цілях без їхнього відома? Зростаюча кількість новин про успішні хакерські атаки та масштабні витоки приватних даних лише підсилює ці побоювання. Користувачі все частіше шукають альтернативи, які б гарантували повний контроль над їхньою інформацією.

Існуючі популярні рішення, хоча й функціональні, часто не забезпечують належного рівня приватності через відсутність справжнього наскрізного шифрування, де тільки користувач володіє ключами до своїх даних. Також можуть виникати питання щодо стійкості до певних типів атак та забезпечення цілісності даних.

Зважаючи на ці виклики, виникає необхідність у розробці хмарного сховища нового покоління, яке б поєднувало зручність використання з найвищим рівнем безпеки. Таке рішення повинно базуватися на принципах наскрізного шифрування, гарантуючи, що доступ до даних має виключно їх власник, а сервер виступає лише як захищене "транзитне" або "зберігаюче" середовище для зашифрованих даних.

		Кондратюк О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

Основною метою цього проєкту є розробка веб-додатку – хмарного сховища з реалізацією наскрізного шифрування, яке гарантує односторонній доступ до даних криптографічним шляхом, унеможлиблює доступ до даних з боку сервера, забезпечує високу стійкість до витоків даних та пропонує зручний і сучасний інтерфейс користувача.

Для досягнення цієї мети необхідно виконати такі основні етапи:

- провести аналіз існуючих хмарних сховищ, виявити їх переваги та недоліки, особливо щодо моделей безпеки та шифрування;
- сформулювати вимоги до системи, зосереджуючись на безпеці, наскрізному шифруванні, приватності, зручності використання та основних функціях (завантаження, скачування, керування файлами, можливий обмін);
- обґрунтувати вибір технологічного стеку для реалізації клієнтської та серверної частин, бази даних та криптографічних бібліотек;
- спроектувати архітектуру системи, включаючи взаємодію клієнта та сервера, структуру бази даних, компоненти системи;
- розробити та описати ключові криптографічні алгоритми та процеси: реєстрація та автентифікація користувача, генерація та управління ключами (майстер-ключ, ключ файлу, похідні ключі), шифрування/дешифрування файлів та метаданих, забезпечення цілісності даних;
- реалізувати систему онлайн-сховища, використовуючи обрані технології та спроектовану архітектуру;
- провести тестування розробленого додатку для перевірки функціональності, безпеки та зручності використання.

1.2. Аналіз аналогів програмного продукту

Ринок хмарних сховищ пропонує безліч сервісів, однак для порівняння та виявлення ключових відмінностей у підходах до безпеки було обрано три відомі аналоги: Dropbox, Google Drive та OneDrive. Аналіз цих платформ дозволяє

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

краще зрозуміти поточні стандарти ринку та визначити нішу для розроблюваного продукту з акцентом на приватність.

Dropbox є одним із піонерів ринку хмарних сховищ [1]. Сервіс пропонує зручний інтерфейс (рис. 1.1), широкий функціонал для синхронізації файлів між пристроями та можливості для спільної роботи. Dropbox декларує використання надійних засобів захисту для даних користувачів. Однак, аналіз їхньої політики та звітів прозорості (рис. 1.2) показує, що компанія може надавати доступ до вмісту файлів користувачів на запити правоохоронних органів. Це свідчить про те, що серверна частина має технічну можливість доступу до незашифрованих даних або ключів шифрування, хоча компанія обіцяє не робити цього без законних підстав.

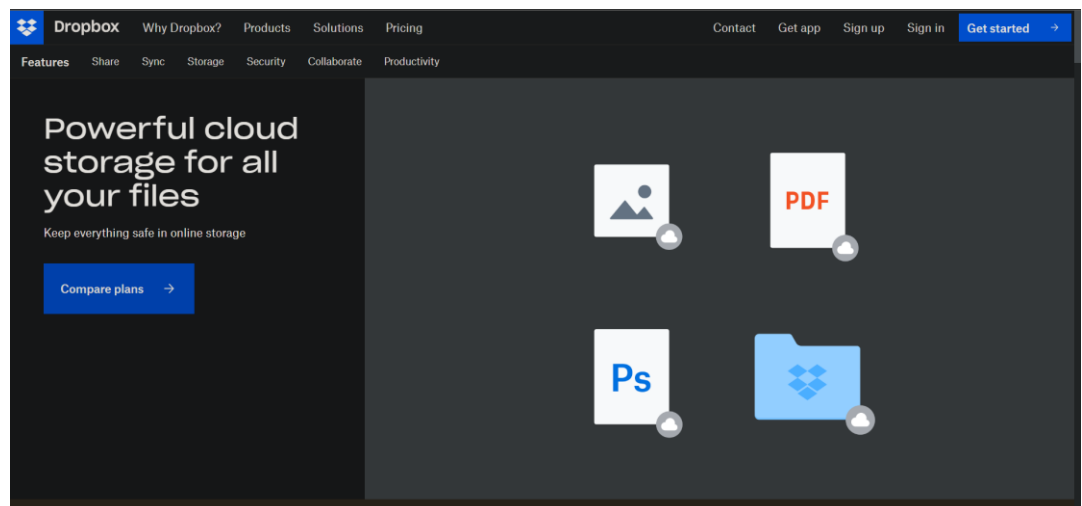


Рисунок 1.1 – Головна сторінка веб-сайту dropbox.com

Search warrants

1332 requests

How we responded

Does not exist	234
No information provided	100
Non-content provided	5
Notice provided	3
Content provided	993
Notice provided	503

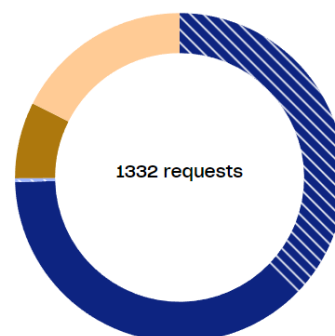


Рисунок 1.2 – Звіт прозорості за адресою help.dropbox.com/transparency/reports

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

Dropbox має низку переваг, серед яких зручний інтерфейс та простота використання, а також потужні можливості синхронізації та спільного доступу. Крім того, сервіс є кросплатформним, підтримуючи Windows, macOS, Linux, iOS, Android та веб-інтерфейс.

Однак, існують і певні недоліки. Зокрема, безкоштовний обсяг сховища обмежений всього двома гігабайтами. Також варто відзначити відсутність наскрізного шифрування за замовчуванням та питання конфіденційності, пов'язані з можливістю доступу до даних компанією.

Google Drive – ще один надзвичайно популярний сервіс, тісно інтегрований в екосистему Google [2]. Він пропонує великий обсяг безкоштовного сховища та зручний інтерфейс (рис. 1.3). Google також заявляє про високі стандарти безпеки. Проте, в офіційній документації щодо безпеки зазначено, що сервіс обробляє вміст файлів користувачів для надання таких послуг, як фільтрація спаму, виявлення вірусів та шкідливого ПЗ, а також для індексації та пошуку всередині файлів [3]. Це прямо вказує на те, що Google Drive не використовує наскрізне шифрування за замовчуванням, і сервер має доступ до вмісту файлів.

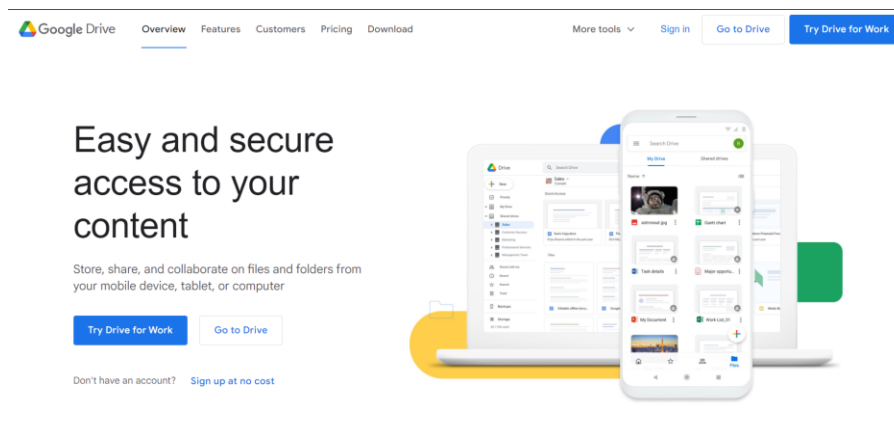


Рисунок 1.3 – Головна сторінка веб-сайту google.com/drive

Google Drive пропонує значні переваги, зокрема великий безкоштовний обсяг сховища у 15 ГБ та тісну інтеграцію з екосистемою Google, що включає Docs, Sheets, Gmail та інші сервіси. Також варто відзначити потужні інструменти для спільної роботи в реальному часі та доступність на всіх основних платформах, таких як Web, Windows, macOS, iOS та Android.

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Проте, Google Drive має і певні недоліки. До них належить відсутність наскрізного шифрування за замовчуванням та обробка вмісту файлів сервером для різних цілей, що може викликати питання щодо конфіденційності. Крім того, можливості офлайн-доступу дещо обмежені порівняно з конкурентами.

OneDrive – хмарний сервіс від Microsoft (рис. 1.4), тісно інтегрований з операційною системою Windows та пакетом Microsoft 365 [4]. Він дозволяє зберігати файли, синхронізувати їх між пристроями та надавати спільний доступ до документів, фотографій, відео та інших типів даних. OneDrive пропонує доступ до файлів з будь-якого пристрою, підключеного до Інтернету, включаючи комп'ютери, смартфони та планшети. Сервіс підтримує автоматичне резервне копіювання важливих даних, зокрема робочих документів або фотоальбомів, що забезпечує додаткову безпеку у випадку втрати чи пошкодження пристрою. Крім того, OneDrive має функції спільної роботи над файлами в реальному часі.

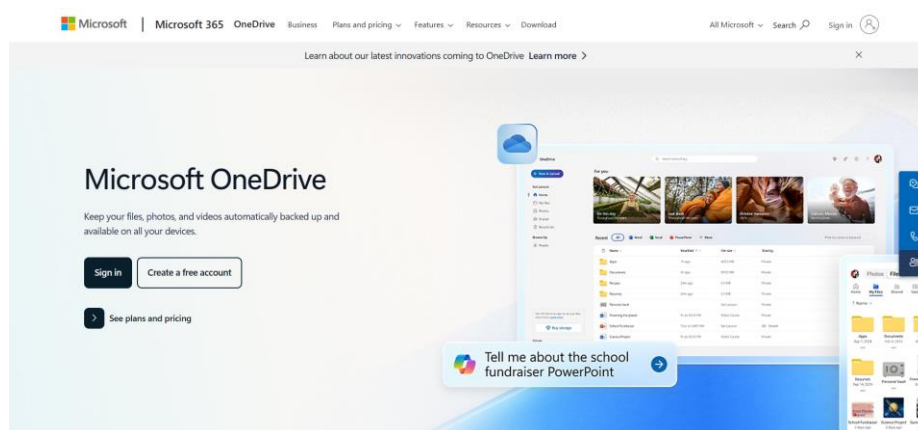


Рисунок 1.4 – Головна сторінка Microsoft OneDrive

OneDrive вирізняється низкою переваг, серед яких глибока інтеграція з операційною системою Windows та пакетом Microsoft 365. Сервіс доступний на всіх ключових платформах: Windows, macOS, iOS, Android, Web та Linux. Корисною є функція "Files On-Demand", яка дозволяє бачити всі файли без їх завантаження, економлячи таким чином місце на диску. OneDrive також надає можливості для спільної роботи над документами Office у реальному часі та пропонує надійні засоби безпеки, такі як шифрування даних під час передачі та зберігання (AES 256-біт), двофакторна автентифікація та Personal Vault – особисте сховище з додатковим захистом.

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						13

До недоліків можна віднести безкоштовний обсяг сховища, який становить 5 ГБ, що відносно небагато порівняно з Google Drive. Також виникають питання щодо конфіденційності, оскільки Microsoft контролює ключі шифрування і може отримати доступ до даних за вимогою законодавства, а наскрізне шифрування за замовчуванням відсутнє.

Виходячи з наявних переваг та недоліків, складемо порівняльну таблицю вищезгаданих програмних аналогів (табл. 1.1).

Таблиця 1.1

Порівняння аналогів хмарних сховищ

Характеристика	Dropbox	Google Drive	OneDrive
Безкоштовне сховище	2 ГБ	15 ГБ	5 ГБ
Безпека	Шифрування при передачі (TLS/SSL) та зберіганні (AES 256-біт).	Шифрування при передачі та зберіганні.	Шифрування при передачі та зберіганні (AES 256-біт), 2FA, Personal Vault.
Наскрізне шифрування (E2EE)	Немає за замовчуванням.	Немає за замовчуванням.	Немає за замовчуванням.
Можливості співпраці	Синхронізація, спільні папки, коментарі.	Спільне редагування в реальному часі.	Спільне редагування в реальному часі (Office apps), спільні папки, контроль версій.
Доступність на платформах	Web, Windows, macOS, Linux, iOS, Android.	Web, Windows, macOS, iOS, Android.	Web, Windows, macOS, Linux, iOS, Android.
Ціноутворення (платні плани)	Різні плани з більшим обсягом сховища та додатковими функціями.	Різні плани (Google One) з більшим обсягом та додатковими перевагами.	Різні плани, включаючи вигідні пакети Microsoft 365 з Office apps та 1 ТБ+ сховища.
Офлайн доступ / Синхронізація	Потужна синхронізація, вибіркового офлайн доступ.	Обмежений офлайн доступ, синхронізація через додаток.	Files On-Demand для доступу без завантаження, можливість зробити файли/папки доступними офлайн.

Аналіз цих провідних сервісів показує важливу деталь: незважаючи на заяви про безпеку, стандартні моделі їх роботи не передбачають наскрізного шифрування, при якому тільки користувач контролює ключі шифрування. Сервер так чи інакше має доступ до даних або ключів. Це створює ризики для конфіденційності, чи то через внутрішні політики компанії, запити третіх сторін, чи потенційні витоки даних.

Отже, існує чітка потреба у хмарному сховищі, яке б реалізовувало справжнє наскрізне шифрування за замовчуванням. Розроблюваний продукт має на меті задовольнити цю потребу, пропонуючи користувачам максимальний рівень приватності, де безпека гарантується криптографічно, а не лише політиками компанії. При цьому важливо зберегти зручність використання, властиву аналогам, та забезпечити основний функціонал для керування файлами.

1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вибір правильного набору технологій є критично важливим для створення надійного, безпечного та продуктивного хмарного сховища з наскрізним шифруванням. Технологічний стек для цього проєкту обирався з урахуванням вимог до безпеки, максимальної продуктивності серверної частини та можливості реалізації складної клієнтської логіки шифрування.

Для серверної частини (Backend) обрано мову програмування C++ через необхідність досягнення максимальної продуктивності, низькорівневого контролю над системними ресурсами та можливості тонкого налаштування безпеки [5]. Використання високопродуктивного асинхронного веб-фреймворку, такого як Drogen, дозволяє ефективно обробляти велику кількість одночасних запитів та забезпечує зручну інтеграцію компонентів [6].

Клієнтська частина (Frontend) буде реалізована на чистому JavaScript (Vanilla JS) для максимального контролю над кодом, оптимізації продуктивності та мінімізації розміру додатку, що важливо для криптографічних функцій [7]. Структура та візуальне оформлення створюватимуться за допомогою

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

стандартних HTML та CSS [8-9]. Ключову роль у реалізації наскрізного шифрування на клієнті відіграє Web Crypto API, сучасний веб-стандарт, що надає браузеру доступ до необхідних криптографічних примітивів (генерація випадкових чисел, хешування, шифрування, робота з ключами) [10]. Додатково будуть залучені легковагові бібліотеки для специфічних алгоритмів, такі як jszip для zip-архівування файлів на клієнті, aesasm для алгоритму AES-CCM, та aesjs для алгоритму AES-ECB [11-13].

Як базу даних було обрано PostgreSQL [14]. Це вільна та відкрита реляційна система управління базами даних (СУБД), що дозволяє зберігати та керувати великими обсягами даних. PostgreSQL є однією з найбільш потужних та розширюваних СУБД на ринку, з високим рівнем надійності та безпеки. Вона підтримує стандарт SQL та включає багато розширень для розширених функціональних можливостей, таких як географічні запити, повнотекстовий пошук, JSON-операції та багато інших [15]. PostgreSQL також підтримує розширення, що дозволяють розробникам створювати власні функції та типи даних. Однією з ключових рис є розширюваність, що дозволяє спільноті постійно вдосконалювати СУБД. Postgres відома своєю безпекою та надійністю, маючи широкий набір функцій безпеки та високий рівень стійкості до помилок. У підсумку, PostgreSQL є потужною та надійною СУБД з широким набором функцій, високим рівнем безпеки, підтримкою стандарту SQL та багатьма розширеннями, що робить її популярним вибором для різноманітних додатків.

Основні криптографічні рішення та механізми безпеки включають: реалізацію наскрізного шифрування, де дані шифруються та дешифруються виключно на клієнті; використання ієрархії ключів (майстер-ключ, ключ файлу, похідні ключі); шифрування файлів за допомогою AES (AES-CCM для конфіденційності та цілісності) [16]; шифрування ключів (майстер-ключа та ключів файлів) за допомогою AES та RSA для сесійних ключів; надійне походження майстер-ключа з пароля за допомогою PBKDF2-HMAC-SHA512 з криптографічною сіллю та достатньою кількістю ітерацій [17]; безпарольну автентифікацію на сервері з використанням похідного ключа та збереженням

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

лише його хешу; забезпечення цілісності даних через MAC-коди; захист від атак за часом; використання надійної C++ бібліотеки (OpenSSL) для криптографічних операцій на сервері [18].

Обраний стек технологій та підходи до безпеки дозволяють створити систему, яка відповідає поставленим цілям: забезпечує високий рівень приватності завдяки наскрізному шифруванню, є достатньо продуктивною для веб-додатку та використовує поширені, добре документовані технології.

Висновки до першого розділу

У першому розділі було проведено детальний аналіз предметної області, пов'язаної зі зберіганням даних у хмарі, що дозволило чітко визначити ключові проблеми та вимоги до розроблюваної системи хмарного сховища. Основною метою проєкту визначено створення безпечного, надійного та зручного для користувача рішення, яке б гарантувало конфіденційність даних завдяки імплементації наскрізного шифрування.

У процесі дослідження існуючих аналогів, зокрема таких популярних сервісів як Dropbox та Google Drive, було виявлено їхні сильні сторони у зручності та функціональності, але водночас підтверджено обмеження щодо гарантій приватності даних та відсутність повного контролю користувача над ключами шифрування. Цей аналіз обґрунтував необхідність розробки системи, де безпека та приватність забезпечуються криптографічно на рівні архітектури.

На основі аналізу проблематики та аналогів було сформульовано вимоги до власної системи та обґрунтовано вибір технологічного стеку. Для серверної частини було обрано мову C++ з використанням фреймворку Drogon задля досягнення високої продуктивності та контролю над ресурсами. Клієнтська частина буде реалізована на Vanilla JavaScript з використанням Web Crypto API для забезпечення наскрізного шифрування безпосередньо у браузері користувача. Як система управління базами даних обрано PostgreSQL завдяки її надійності та відповідності вимогам до цілісності даних. Також визначено ключові криптографічні підходи та механізми безпеки (AES, RSA, PBKDF2,

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

SHA-256, керування ключами на клієнті), що будуть використані для захисту даних.

Таким чином, у першому розділі закладено теоретичну основу для подальшого проектування та розробки системи, визначено її основні характеристики та технологічні рішення, спрямовані на створення конкурентоспроможного та безпечного хмарного сховища.

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

РОЗДІЛ 2. ПРОЄКТУВАННЯ ХМАРНОГО СХОВИЩА

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Система хмарного сховища розроблена для забезпечення безпечного зберігання та обміну файлами з використанням наскрізного шифрування. Взаємодія з системою відбувається через веб інтерфейс, де користувачі виконують різноманітні операції залежно від своїх ролей та потреб. Основними дійовими особами (акторами) системи є користувач та гість. Їхні можливі дії та взаємозв'язки представлені на діаграмі варіантів використання (рис. 2.1).

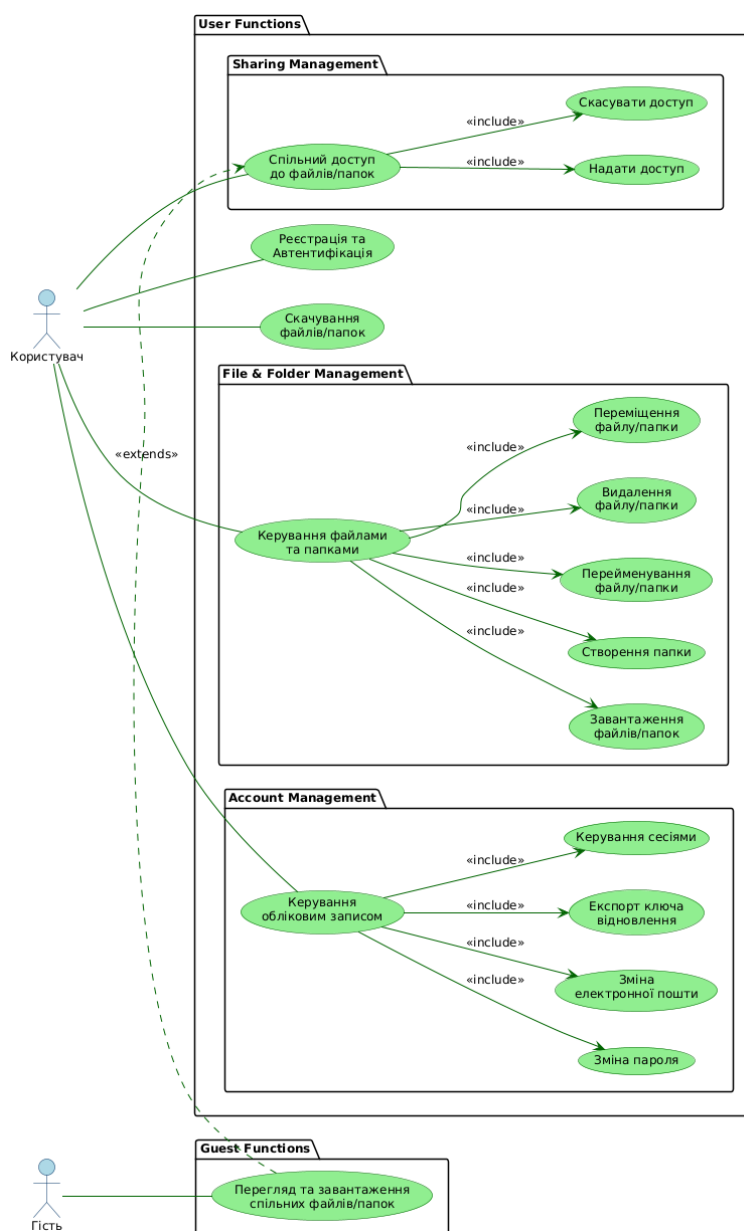


Рисунок 2.1. – Діаграма використання

Опис діаграми варіантів використання наведено в табл. 2.1 – 2.6.

Таблиця 2.1

Опис прецеденту «Реєстрація та Автентифікація»

Назва варіанту використання	Реєстрація та Автентифікація
Основні актори	Користувач
Короткий опис	Користувач створює новий обліковий запис, надаючи адресу електронної пошти та пароль. У процесі реєстрації на стороні клієнта генеруються унікальні криптографічні ключі, що забезпечують наскрізне шифрування. Пароль користувача у відкритому вигляді на сервер не передається і не зберігається. При автентифікації клієнт розшифровує майстер-ключ.

Таблиця 2.2

Опис прецеденту «Керування файлами та папками»

Назва варіанту використання	Керування файлами та папками
Основні актори	Користувач
Короткий опис	Користувач має можливість завантажувати файли та цілі папки у своє хмарне сховище, створювати нові папки для організації даних, перейменовувати існуючі файли та папки, видаляти їх, а також переміщувати файли та папки між різними директоріями у сховищі. Усі операції, що стосуються імен файлів/папок, виконуються з попереднім шифруванням/дешифруванням на стороні клієнта.

Таблиця 2.3

Опис прецеденту «Скачування файлів/папок»

Назва варіанту використання	Скачування файлів/папок
Основні актори	Користувач
Короткий опис	Користувач може вибирати файли або папки зі свого сховища для завантаження на свій пристрій. Усі дані дешифруються безпосередньо на клієнтському пристрої після отримання їх із сервера. При завантаженні папки її вміст автоматично архівується у ZIP-файл на стороні клієнта.

Таблиця 2.4

Опис прецеденту «Спільний доступ до файлів/папок»

Назва варіанту використання	Скачування файлів/папок
Основні актори	Користувач
Короткий опис	Користувач може надавати публічний доступ до своїх файлів або папок шляхом генерації унікального посилання. Це посилання містить спеціальний ключ для дешифрування, який передається через URL-фрагмент і не доступний серверу. Користувач також може в будь-який момент скасувати наданий спільний доступ.

Таблиця 2.5

Опис прецеденту «Керування обліковим записом»

Назва варіанту використання	Керування обліковим записом
Основні актори	Користувач

		Кондратюк. О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

Короткий опис	Користувач може змінювати пароль свого облікового запису та адресу електронної пошти (з обов'язковим підтвердженням нової адреси). Також надається можливість експортувати майстер-ключ відновлення для забезпечення доступу до даних у випадку втрати пароля. Додатково, користувач може переглядати список своїх активних сесій та деактивувати будь-яку з них для підвищення безпеки.
---------------	--

Таблиця 2.6

Опис прецеденту «Перегляд та завантаження спільних файлів/папок»

Назва варіанту використання	Скачування файлів/папок
Основні актори	Гість
Короткий опис	Гість, маючи спеціальне посилання, надане Користувачем, може отримати доступ до зашифрованого файлу або вмісту папки. Використовуючи ключ, вбудований у фрагмент посилання, клієнтський додаток Гостя дешифрує отримані дані, дозволяючи їх переглянути (якщо тип файлу підтримується для попереднього перегляду) або завантажити на пристрій.

Функціональні вимоги до системи є такими.

Система хмарного сховища повинна забезпечувати наскрізне шифрування (E2EE), що означає, що всі файли шифруються на пристрої користувача перед завантаженням на сервер і дешифруються лише на пристрої користувача після

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

скачування; сервер не повинен мати фізичної можливості отримувати незашифровані дані користувача, реалізуючи принцип нульової довіри (Zero Trust). Керування криптографічними ключами повністю покладено на клієнтську сторону: генерація майстер-ключа та ключів файлів/папок відбувається на пристрої користувача, і ці ключі у незашифрованому вигляді ніколи не передаються на сервер.

Процес автентифікації користувача має бути безпечним і не повинен передавати пароль користувача на сервер; натомість він базується на криптографічних похідних від пароля. Важливі метадані, такі як назви файлів та папок, також підлягають шифруванню перед збереженням на сервері. Для забезпечення цілісності даних використовується алгоритм AES-CCM, який перевіряє дані під час завантаження та скачування, як для кожної окремої частини файлу, так і для всього файлу в цілому.

Система повинна підтримувати безпечний спільний доступ до файлів та папок, де ключ дешифрування передається разом із посиланням через URL-фрагмент, унеможливаючи доступ сервера до нього. Користувач також має можливість відкликати наданий спільний доступ. Передбачено стандартні функції керування файловою системою: створення папок, перейменування та видалення файлів і папок, а також переміщення файлів і папок між директоріями.

Для відновлення доступу у випадку втрати пароля система надає користувачу можливість експортувати свій майстер-ключ. Користувачі можуть керувати своїми сесіями, переглядаючи список активних та неактивних сесій, та деактивувати будь-яку з них; цей список повинен включати IP-адресу та User-Agent пристрою. Також реалізовано можливість зміни пароля та адреси електронної пошти (з обов'язковим підтвердженням нової адреси).

Нефункціональні вимоги до системи наведені нижче.

Час виконання операцій шифрування та дешифрування на клієнтській стороні має бути в прийнятних межах, для чого використовуються Web Workers, що дозволяють виконувати обчислювально складні задачі у фоновому режимі без блокування основного потоку інтерфейсу користувача. Швидкість

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

завантаження та скачування файлів повинна обмежуватися головним чином пропускнуою здатністю мережі користувача та сервера, а не процесами шифрування. Інтерфейс користувача має швидко реагувати на дії користувача. Система повинна бути доступною для користувачів переважно більшість часу.

Архітектурні рішення мають дозволяти горизонтальне масштабування серверної частини та інфраструктури зберігання файлів для підтримки зростаючої кількості користувачів та обсягу даних, зокрема через використання окремих статичних серверів для зберігання файлів. З точки зору безпеки, система має бути стійкою до відомих векторів атак, таких як спроби підбору паролів (ускладнено використанням PBKDF2) та атаки на активні сесії. Використовуються лише перевірені криптографічні бібліотеки та стандартизовані алгоритми, такі як AES-CCM для шифрування файлів, AES-ECB для шифрування ключів, PBKDF2-HMAC-SHA512 для отримання ключів з паролів та RSA-OAEP для захисту сесійних даних.

Інтерфейс користувача має бути інтуїтивно зрозумілим, особливо для користувачів, які мають досвід роботи з іншими популярними хмарними сховищами. Процеси шифрування та дешифрування даних повинні відбуватися максимально прозоро для користувача, не вимагаючи від нього спеціальних знань у галузі криптографії.

Системні вимоги:

Клієнт:

- сучасний веббраузер з підтримкою JavaScript та Web Crypto API;
- достатньо ресурсів (CPU, RAM) для виконання операцій шифрування/дешифрування на клієнті (залежить від розміру файлів);
- стабільне інтернет-з'єднання;

Сервер (Backend - C++):

- операційна система Linux (рекомендовано);
- встановлений компілятор C++ з підтримкою C++20 або новіше.
- встановлені залежності (OpenSSL, фреймворк Drogon);

		Кондратюк О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

- достатньо ресурсів CPU, RAM та дискового простору для обробки запитів;

Сервер (База даних - PostgreSQL):

- встановлений сервер PostgreSQL;
- достатньо дискового простору для зберігання метаданих;
- налаштовані резервні копії;

Статичний сервер (Сховище файлів):

- спеціалізоване рішення для зберігання об'єктів;
- достатньо дискового простору для зберігання зашифрованих файлів;
- висока пропускну здатність мережі;

Для представлення логічної структури системи та взаємозв'язків між її основними сутностями використовується діаграма класів (рис. 2.2). Ця діаграма відображає основні сутності серверної частини та їх взаємозв'язки, базуючись на моделях даних та сервісах, реалізованих у проєкті.

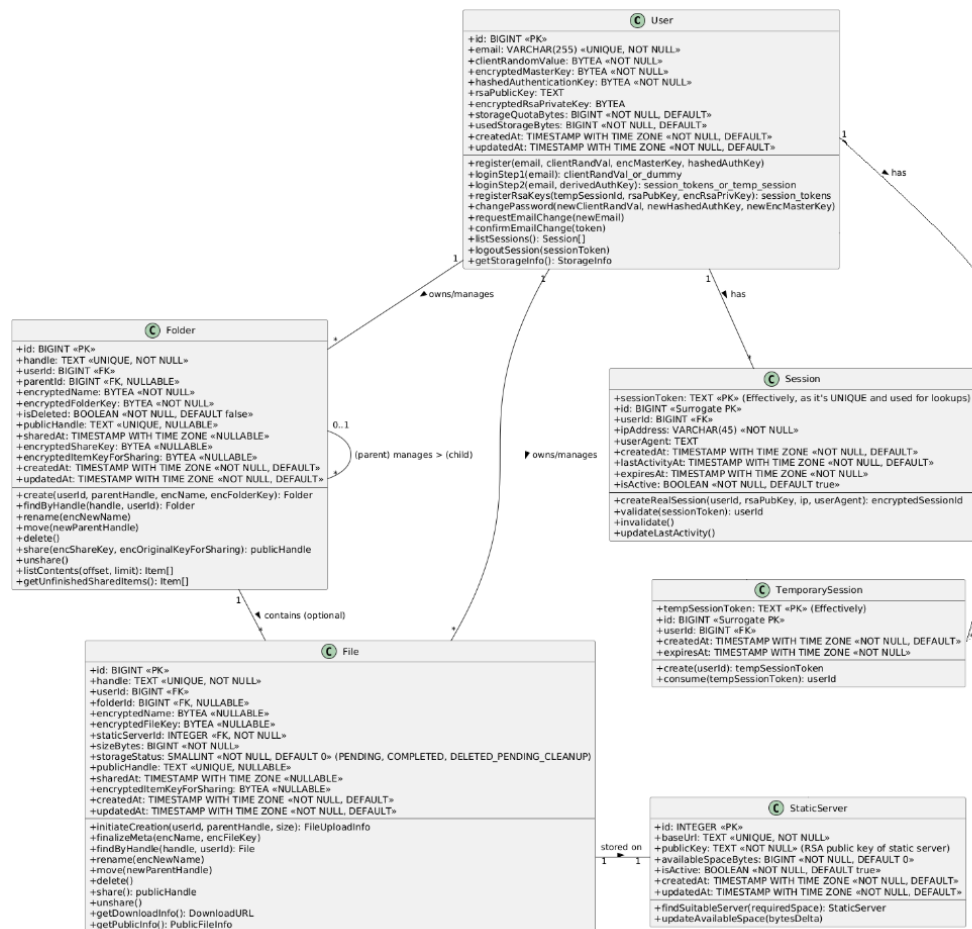


Рисунок 2.2. – Діаграма класів

		Кондратюк, О.Д.			Арк.
		Локтікова Т.М.			25
Змн.	Арк.	№ докум.	Підпис	Дата	

Ключовим класом є User, який представляє зареєстрованого користувача. Він містить такі атрибути, як id, email, clientRandomValue, encryptedMasterKey, hashedAuthenticationKey, а також опціональні rsaPublicKey та encryptedRsaPrivateKey для шифрування сесійних даних. Також клас зберігає інформацію про квоту зберігання (storageQuotaBytes, usedStorageBytes) та часові мітки (createdAt, updatedAt). Методи цього класу, реалізовані через відповідні сервіси, забезпечують реєстрацію, автентифікацію, зміну пароля та електронної пошти, керування RSA-ключами.

Для організації файлової системи використовуються класи Folder та File.

Клас Folder має атрибути id, handle (унікальний текстовий ідентифікатор), userId (власник), parentId (для реалізації ієрархічної структури, може бути NULL), encryptedName (зашифроване ім'я папки), encryptedFolderKey (ключ папки, зашифрований майстер-ключем користувача). Також він містить поля для реалізації спільного доступу: publicHandle, sharedAt, encryptedShareKey (ключ спільного доступу до папки, зашифрований майстер-ключем) та encryptedItemKeyForSharing (ключ цієї папки, зашифрований ключем спільного доступу). Атрибути isDeleted, createdAt, updatedAt відслідковують стан та часові мітки. Методи дозволяють створювати, перейменовувати, переміщувати, видаляти папки та керувати спільним доступом до них.

Клас File характеризується атрибутами id, handle, userId, folderId (батьківська папка, може бути NULL), encryptedName (зашифроване ім'я файлу), encryptedFileKey (ключ файлу, зашифрований майстер-ключем), staticServerId (вказує на сервер зберігання), sizeBytes (оригінальний розмір файлу), storageStatus (статус зберігання: PENDING, COMPLETED, DELETED_PENDING_CLEANUP). Для спільного доступу передбачені publicHandle, sharedAt та encryptedItemKeyForSharing (ключ файлу, зашифрований ключем спільного доступу батьківської папки). Часові мітки createdAt та updatedAt також присутні. Методи класу забезпечують ініціалізацію завантаження, фіналізацію метаданих після завантаження, отримання інформації

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

для скачування, перейменування, переміщення, видалення та керування спільним доступом до файлу.

Для керування сесіями користувачів існують класи, що відображають таблиці sessions та temporary_sessions.

Клас Session (на основі SessionRecord) містить sessionToken (унікальний токен), userId, ipAddress, userAgent, createdAt, lastActivityAt, expiresAt та isActive. Методи цього класу дозволяють створювати, валідувати, оновлювати та деактивувати сесії.

Клас TemporarySession (на основі TempSessionRecord) з атрибутами tempSessionToken, userId, createdAt, expiresAt використовується для короткотривалих сесій, наприклад, під час реєстрації RSA-ключів.

Клас StaticServer описує доступні сервери для зберігання файлів, маючи атрибути id, baseUrl, publicKey (публічний RSA-ключ сервера для захисту запитів до нього), availableSpaceBytes та isActive.

Зв'язки між класами такі: User є власником багатьох Folder та File. Folder може містити інші Folder (рекурсивний зв'язок parentId) та багато File. Кожен File зберігається на одному StaticServer. Користувач може мати декілька активних Session та тимчасових TemporarySession.

Описані вище сутності та їх зв'язки формують повноцінну модель, що забезпечує основні функції хмарного сховища з підтримкою наскрізного шифрування, ієрархічної структури файлів і папок, керування сесіями користувача та механізмами безпечного спільного доступу. Така модель дозволяє ефективно управляти даними, гарантувати конфіденційність і масштабованість системи.

2.2. Розробка бази даних системи

Для зберігання метаданих користувачів, інформації про файли, папки, сесії та конфігурації статичних серверів використовується реляційна система управління базами даних PostgreSQL. Вибір PostgreSQL обґрунтований її надійністю, продуктивністю, підтримкою стандартів SQL, розширюваністю та

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

відкритим вихідним кодом, що відповідає вимогам проєкту до безпеки та цілісності даних.

Важливо наголосити, що в базі даних не зберігаються:

- незашифровані файли користувачів;
- паролі користувачів у відкритому вигляді;
- незашифровані майстер-ключі, ключі файлів або папок.

Усі чутливі дані або шифруються перед збереженням (наприклад, майстер-ключ, ключі файлів/папок, імена файлів/папок), або зберігаються у вигляді хеш-значень (похідний ключ автентифікації).

Опис кожної з таблиць бази даних наведений у табл. 2.7 – 2.12.

Таблиця 2.7

Структура таблиці users

Поле	Тип	Призначення
id	BIGINT	Унікальний ідентифікатор користувача (Первинний ключ)
email	VARCHAR (255)	Адреса електронної пошти користувача (логін)
client_random_value	BYTEA	Випадкове значення, згенероване клієнтом (16 байт), використовується для генерації солі при вході до системи
encrypted_master_key	BYTEA	Майстер-ключ користувача, зашифрований похідним ключем шифрування
hashed_authentication_key	BYTEA	Хеш похідного ключа автентифікації (SHA-256, усічений до 16 байт)
rsa_public_key	TEXT	Публічний ключ RSA користувача для шифрування сесійних даних

Продовження табл. 2.7

encrypted_rsa_private_key	BYTEA	Приватний ключ RSA користувача, зашифрований майстер-ключем
storage_quota_bytes	BIGINT	Максимально дозволений обсяг сховища для користувача в байтах
used_storage_bytes	BIGINT	Поточний обсяг використаного сховища користувачем в байтах
created_at	TIMESTAMP WITH TIME ZONE	Дата та час реєстрації користувача
updated_at	TIMESTAMP WITH TIME ZONE	Дата та час останнього оновлення даних користувача

Таблиця 2.8

Структура таблиці folders

Поле	Тип	Призначення
id	BIGINT	Унікальний ідентифікатор папки (Первинний ключ)
handle	TEXT	Унікальний текстовий ідентифікатор папки (генерується системою)
user_id	BIGINT	Ідентифікатор користувача-власника
parent_id	BIGINT	Ідентифікатор батьківської папки (NULL для кореневого рівня)
encrypted_name	BYTEA	Ім'я папки, зашифроване ключем папки

encrypted_folder_key	BYTEA	Ключ шифрування папки, зашифрований майстер-ключем користувача
is_deleted	BOOLEAN	Прапорець, що вказує, чи папка "м'яко" видалена (очікує на остаточне видалення сервісом очищення)
public_handle	TEXT	Унікальний публічний ідентифікатор для спільного доступу до папки, або NULL, якщо папка не надає можливості для спільного доступу
shared_at	TIMESTAMP WITH TIME ZONE	Дата та час, коли було надано спільний доступ до папки
encrypted_share_key	BYTEA	Ключ спільного доступу до папки (згенерований користувачем), зашифрований майстер-ключем користувача
encrypted_item_key for_sharing	BYTEA	Ключ цієї папки, зашифрований ключем батьківської спільної папки
created_at	TIMESTAMP WITH TIME ZONE	Дата та час створення папки
updated_at	TIMESTAMP WITH TIME ZONE	Дата та час останнього оновлення папки

Структура таблиці files

Поле	Тип	Призначення
id	BIGINT	Унікальний ідентифікатор файлу (Первинний ключ)
handle	TEXT	Унікальний текстовий ідентифікатор файлу (генерується системою)
user_id	BIGINT	Ідентифікатор користувача- власника
folder_id	BIGINT	Ідентифікатор батьківської папки (NULL для кореневого рівня)
encrypted_name	BYTEA	Ім'я файлу, зашифроване ключем файлу (може бути NULL до фіналізації завантаження)
encrypted_file_key	BYTEA	Ключ шифрування файлу, зашифрований майстер-ключем користувача (може бути NULL до фіналізації завантаження)
static_server_id	INTEGER	Ідентифікатор статичного сервера, де зберігається файл
size_bytes	BIGINT	Розмір файлу в байтах
storage_status	SMALLINT	Статус зберігання файлу (PENDING, DELETED PENDING CLEANUP, COMPLETED)
public_handle	TEXT	Унікальний публічний ідентифікатор для спільного доступу до файлу

Продовження табл. 2.9

shared_at	TIMESTAMP WITH TIME ZONE	Дата та час, коли було надано спільний доступ до файлу
encrypted_item_key for_sharing	BYTEA	Ключ цього файлу, зашифрований ключем спільного доступу батьківської папки
created_at	TIMESTAMP WITH TIME ZONE	Дата та час створення запису про файл
updated_at	TIMESTAMP WITH TIME ZONE	Дата та час останнього оновлення запису про файл

Таблиця 2.10

Структура таблиці sessions

Поле	Тип	Призначення
id	BIGINT	Унікальний числовий ідентифікатор сесії (Первинний ключ)
session_token	TEXT	Унікальний токен сесії (використовується для автентифікації)
user_id	BIGINT	Ідентифікатор користувача, якому належить сесія
ip_address	VARCHAR(45)	IP-адреса, з якої створено сесію
user_agent	TEXT	User-Agent клієнта

created_at	TIMESTAMP WITH TIME ZONE	Дата та час створення сесії
last_activity_at	TIMESTAMP WITH TIME ZONE	Дата та час останньої активності в сесії
expires_at	TIMESTAMP WITH TIME ZONE	Дата та час, коли сесія стане недійсною
is_active	BOOLEAN	Прапорець, що вказує, чи активна сесія

Таблиця 2.11

Структура таблиці temporary_sessions

Поле	Тип	Призначення
id	BIGINT	Унікальний числовий ідентифікатор тимчасової сесії (Первинний ключ)
temp_session_token	TEXT	Унікальний токен тимчасової сесії
user_id	BIGINT	Ідентифікатор користувача, якому належить сесія
created_at	TIMESTAMP WITH TIME ZONE	Дата та час створення тимчасової сесії
expires_at	TIMESTAMP WITH TIME ZONE	Дата та час, коли тимчасова сесія стане недійсною

Структура таблиці static_servers

Поле	Тип	Призначення
id	INTEGER	Унікальний ідентифікатор статичного сервера (Первинний ключ)
base_url	TEXT	Базова URL-адреса статичного сервера
public_key	TEXT	Публічний ключ RSA статичного сервера
available_space_bytes	BIGINT	Доступний обсяг сховища на сервері в байтах
is_active	BOOLEAN	Прапорець, чи активний сервер для нових завантажень
created_at	TIMESTAMP WITH TIME ZONE	Дата та час додавання сервера до системи
updated_at	TIMESTAMP WITH TIME ZONE	Дата та час останнього оновлення інформації про сервер

Ця структура бази даних розроблена для підтримки наскрізного шифрування, зберігаючи лише необхідні метадані та зашифровані ключі на сервері, тоді як самі дані файлів та їх розшифровані ключі залишаються під контролем користувача. Зв'язки ON DELETE CASCADE забезпечують цілісність даних при видаленні користувачів або батьківських папок. Індeksi (idx_*) додані для оптимізації типових запитів.

2.3. Проєктування та реалізація алгоритмів роботи системи

Функціонування хмарного сховища базується на послідовному виконанні алгоритмів, що охоплюють типовий цикл взаємодії користувача з системою: від створення облікового запису до керування файлами та завершення сесії. Усі криптографічні операції, пов'язані з даними користувача та його ключами, виконуються виключно на стороні клієнта, забезпечуючи принцип наскрізного шифрування.

Типовий цикл використання починається з реєстрації нового користувача. Користувач надає свою електронну пошту та створює пароль. Клієнтський додаток локально генерує унікальний майстер-ключ та інші необхідні криптографічні матеріали, похідні від пароля користувача та випадкової солі. Зашифрована версія майстер-ключа та хеш ключа автентифікації надсилаються на сервер для створення облікового запису. Після цього користувачу необхідно підтвердити свою електронну пошту для активації акаунта.

Наступним кроком є автентифікація. Користувач вводить пошту та пароль. Клієнт, використовуючи отриману з сервера інформацію (спеціальне випадкове значення, збережене при реєстрації), відтворює похідні ключі. Один з них використовується для підтвердження особистості перед сервером, а інший – для дешифрування майстер-ключа, який сервер надсилає у зашифрованому вигляді. Якщо це перший вхід після реєстрації або якщо RSA-ключі ще не були налаштовані, система ініціює процес їх генерації на клієнті та безпечної реєстрації на сервері. Ці RSA-ключі надалі використовуються для шифрування сесійних даних.

Після успішної автентифікації користувач отримує доступ до свого файлового менеджера. Він може завантажувати файли та папки. Перед завантаженням кожен файл шифрується на клієнті унікальним ключем файлу, а ім'я файлу також шифрується. Ключ файлу, у свою чергу, шифрується майстер-ключем користувача. Тільки після цього зашифровані дані та метадані передаються на сервер для зберігання. При завантаженні папок, їх структура

		Кондратюк О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

відтворюється на сервері, а кожен файл у папці проходить такий самий процес шифрування та завантаження.

Користувач може скачувати файли та папки. Для цього клієнт запитує у сервера зашифрований файл та зашифрований ключ файлу. Ключ файлу дешифрується на клієнті майстер-ключем, після чого цим ключем дешифрується сам файл. Також перевіряється цілісність файлу. При скачуванні папки, всі файли з неї та її підпапок завантажуються, дешифруються та архівуються у ZIP-файл на стороні клієнта.

Система дозволяє керувати файлами та папками: створювати нові папки (їхні імена та ключі шифруються аналогічно файлам), перейменовувати (нове ім'я шифрується на клієнті відповідним ключем елемента), переміщувати елементи між папками та видаляти їх. При видаленні елементи спочатку позначаються як видалені, а їх фактичне видалення зі сховища відбувається пізніше фоновим процесом.

Важливою функцією є спільний доступ. Користувач може згенерувати унікальне посилання на файл або папку. Це посилання містить публічний ідентифікатор елемента, а ключ дешифрування передається через фрагмент URL, що робить його недоступним для сервера. Гість, маючи таке посилання, може завантажити та дешифрувати дані на своєму пристрої. Користувач може в будь-який момент відкликати спільний доступ.

Для підвищення безпеки та контролю користувач може керувати своїм обліковим записом: змінювати пароль (що призводить до перешифрування майстер-ключа на клієнті новими похідними ключами), змінювати адресу електронної пошти (з підтвердженням), переглядати список активних сесій та деактивувати їх, а також експортувати свій зашифрований майстер-ключ для можливості відновлення доступу.

Завершення роботи з системою відбувається через функцію виходу (logout), яка деактивує поточну сесію на сервері та видаляє збережені сесійні дані (майстер-ключ, сесійний токен) з локального сховища браузера.

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

2.4. Реалізація функціоналу хмарного сховища

Реалізація функціоналу хмарного сховища базується на чіткому розподілі відповідальності між клієнтською та серверною частинами, де клієнт відповідає за всі криптографічні операції з даними користувача, а сервер виступає у ролі захищеного сховища зашифрованих даних та координатора запитів.

Реєстрація нового користувача починається з введення адреси електронної пошти та пароля на клієнтській стороні. На клієнтській стороні реалізовано функцію для оцінки надійності введеного користувачем пароля, яка аналізує його довжину та наявність різних типів символів (додаток А). Після валідації цих даних клієнт генерує 16-байтне випадкове значення `clientRandomValue` та 16-байтний майстер-ключ `masterKeyBytes`. Далі, з `clientRandomValue` та константної строки домену (`lockbox.com`) з доповненням до 200 символів формується префікс, який конкатенується з `clientRandomValue` та хешується за допомогою SHA-256 для отримання солі `saltBytes`. Використовуючи введений пароль та отриману сіль, за допомогою функції PBKDF2-HMAC-SHA512 зі 100000 ітераціями генерується 32-байтний похідний ключ `derivedKeyBytes` (додаток Б). Цей ключ розділяється: перші 16 байт стають ключем шифрування `derivedEncryptionKeyBytes`, наступні 16 байт – ключем автентифікації `derivedAuthenticationKeyBytes`. Ключ автентифікації `derivedAuthenticationKeyBytes` хешується за допомогою SHA-256, і перші 16 байт цього хешу (`hashedAuthenticationKeyBytes`) призначені для зберігання на сервері. Майстер-ключ `masterKeyBytes` шифрується ключем `derivedEncryptionKeyBytes` за алгоритмом AES-ECB, утворюючи `encryptedMasterKeyBytes` (додаток В). На сервер надсилаються: адреса пошти, `clientRandomValue` (у Base64), `encryptedMasterKeyBytes` (у Base64) та `hashedAuthenticationKeyBytes` (у Base64) [19]. Сервер зберігає ці дані та ініціює процес підтвердження пошти надсилаючи лист з посиланням для підтвердження за вказаною адресою. Якщо користувач з такою адресою вже існує, клієнту так само сповіщається, що було надіслано лист для підтвердження реєстрації, щоб запобігти енумерації зареєстрованих адрес електронних пошт.

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Після переходу по посиланню в листі – користувач вважається зареєстрованим, і його буде перенаправлено на сторінку логіну.

Автентифікація користувача відбувається у два етапи. На першому етапі (/api/login1) клієнт надсилає адресу електронної пошти. Сервер, знайшовши користувача, повертає збережене для нього значення clientRandomValue (або генерує фіктивне значення, якщо користувач не знайдений, щоб запобігти перебору імейлів). На другому етапі (/api/login2) користувач вводить пароль. Клієнт, використовуючи отриманий clientRandomValue, відтворює сіль saltBytes та похідний ключ derivedKeyBytes (і, відповідно, derivedEncryptionKeyBytes та derivedAuthenticationKeyBytes) так само, як при реєстрації. Потім клієнт надсилає на сервер адресу пошти та derivedAuthenticationKeyBytes (у Base64). Сервер хешує отриманий derivedAuthenticationKeyBytes (SHA-256, усічений до 16 байт) і порівнює результат зі збереженим hashedAuthenticationKey. Якщо хеші збігаються, автентифікація вважається успішною. Якщо у користувача ще немає зареєстрованої пари RSA-ключів, сервер створює тимчасову сесію (temporary_session_id_b64) і повертає її разом із зашифрованим майстер-ключем користувача (encrypted_master_key_b64). Клієнт генерує 2048-бітну пару RSA-ключів (публічний у форматі SPKI, приватний – PKCS#8) [20-21]. Приватний ключ RSA шифрується за допомогою derivedEncryptionKeyBytes (AES-ECB з PKCS7-доповненням) і разом з публічним ключем RSA та тимчасовим ID сесії надсилається на сервер (/api/register_rsa) [22-23] (додаток Г). Сервер зберігає RSA-ключі, анулює тимчасову сесію, створює постійну сесію, шифрує ID цієї сесії (encrypted_session_id_b64) та зашифрований приватний ключ RSA користувача (encrypted_rsa_private_key_b64 з БД) публічним ключем RSA користувача і повертає їх. Якщо RSA-ключі вже існували, сервер одразу створює постійну сесію і повертає зашифрований майстер-ключ, зашифрований ID сесії та зашифрований приватний ключ RSA з БД. На клієнті майстер-ключ дешифрується за допомогою derivedEncryptionKeyBytes. Зашифрований приватний ключ RSA спершу дешифрується майстер-ключем (AES-ECB, зняття PKCS7-доповнення), а потім отриманим приватним ключем RSA дешифрується

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

ID сесії. Отримані майстер-ключ, приватний RSA-ключ та ID сесії зберігаються в localStorage [24].

Завантаження файлу починається з того, що клієнт надсилає запит на сервер (/api/files/ POST) з розміром файлу та, опціонально, ідентифікатором батьківської папки. Сервер перевіряє квоту, обирає доступний статичний сервер, створює попередній запис про файл у таблиці files зі статусом PENDING та повертає file_handle і upload_url. upload_url містить зашифрований (публічним ключем статичного сервера) запит до статичного сервера. Клієнт генерує випадковий 24-байтний ключ файлу (представлений як 6 слів по 32 біти – fileKeyWords). Ім'я файлу шифрується першими 16 байтами (4 словами) fileKeyWords як ключем та наступними 8 байтами (2 словами) fileKeyWords як основою для IV (де IV формується дублюванням цих 8 байт) за алгоритмом AES-CBC з PKCS7-доповненням (encryptFileName в cryptoUtils.js) [25] (додаток Д). Далі файл читається частинами. Кожна частина шифрується у Web Worker (encrypter.js) за алгоритмом AES-CCM, використовуючи ті ж 6 слів fileKeyWords та інкрементальний лічильник для забезпечення унікальності nonce для кожного блоку (додаток Е). Worker для кожної частини повертає зашифровані дані та 4 слова MAC-тегу. Зашифровані частини надсилаються на статичний сервер за upload_url, до якого додається номер частини та її MAC-тег у Base64URL форматі. Після завантаження всіх частин, клієнт агрегує всі 4-слівні MAC-теги частин в єдиний 4-слівний MAC (condenseMACs в cryptoUtils.js). Потім 6 слів ключа файлу fileKeyWords об'єднуються з 2 словами, що представляють першу половину агрегованого MAC (останні 2 слова MAC не використовуються для обфускації), формуючи 8-слівний обфускований ключ файлу (obfuscateFileKey в cryptoUtils.js). Цей 32-байтний обфускований ключ файлу шифрується майстер-ключем користувача за допомогою AES-ECB і у вигляді Base64 (encrypted_file_key) разом із зашифрованим ім'ям файлу (encrypted_name) надсилається на бекенд-сервер для фіналізації (/api/files/{file_handle}/complete POST) (додаток Ж). Сервер оновлює запис файлу, встановлюючи статус

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

COMPLETED та зберігаючи зашифровані метадані, а також оновлює лічильник використаного місця для користувача.

Скачування файлу ініціюється запитом клієнта на `/api/files/{file_handle}/download`. Сервер повертає `download_url` для статичного сервера, аналогічний тому, що використовується при завантаженні. Клієнт отримує зашифрований/обфускований ключ файлу (`encryptedFileKey`) з метаданих файлу (які він міг завантажити раніше при лістингу папки). Цей ключ дешифрується майстер-ключем користувача (AES-ECB), отримуючи 8-слівний обфускований ключ. З нього відновлюються оригінальні 6 слів ключа файлу (`fileKeyWordsForDecrypter`) та 2 слова очікуваного агрегованого MAC `obfuscatedFileKeyMacPartWords` за допомогою `getFileKeyFromObfuscatedFileKey` та `getCondensedMacWordsFromObfuscatedFileKey` (додаток К). Файл завантажується частинами зі статичного сервера. Кожна частина дешифрується у Web Worker (`decrypter.js`) за алгоритмом AES-CCM, використовуючи `fileKeyWordsForDecrypter` (додаток Л). Worker також повертає 4-слівний MAC-тег для кожної дешифрованої частини. Клієнт агрегує всі отримані MAC-теги частин (`condenseMACs`) і порівнює перші 2 слова результату з `obfuscatedFileKeyMacPartWords` для перевірки цілісності всього файлу. Якщо перевірка успішна, зашифроване ім'я файлу дешифрується (`aesCbcDecrypt` в `cryptoUtils.js`) і користувачеві пропонується зберегти файл.

Створення папки передбачає генерацію на клієнті 24-байтного ключа папки (`folderKeyWords`). Ім'я папки шифрується аналогічно імені файлу (AES-CBC, перші 4 слова ключа папки як ключ, наступні 2 слова як основа для IV). Ключ папки, доповнений до кратності 16 байт за допомогою PKCS7, шифрується майстер-ключем користувача (AES-ECB) (додаток М). Зашифроване ім'я та зашифрований ключ папки, разом з `handle` батьківської папки (якщо є), надсилаються на сервер (`/api/folders` POST). Сервер створює запис у таблиці `folders`.

Спільний доступ до файлу ініціюється користувачем. Клієнт надсилає запит на сервер (`/api/files/{file_handle}/share` POST). Сервер генерує (або

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

повертає існуючий) `public_handle` для файлу та зберігає його в БД. Клієнт отримує цей `public_handle`. Потім клієнт дешифрує зашифрований ключ файлу (`encryptedFileKey`) майстер-ключем, відновлює оригінальний 24-байтний ключ файлу. Цей ключ файлу кодується у Base64URL і додається до URL як фрагмент (#): `shared_file.html#{public_handle}!{base64url_file_key}` (додаток Н). Коли гість відкриває таке посилання, `public_handle` використовується для запиту до сервера (`/api/public/files/{public_handle}` GET) інформації про файл (зашифроване ім'я, розмір, `download_url` до статичного сервера). Ключ файлу з URL-фрагмента використовується для дешифрування імені та, при скачуванні, для дешифрування вмісту файлу (AES-CCM) та перевірки його цілісності.

Спільний доступ до папки працює схожим чином, але з додатковими кроками. Користувач ініціює спільний доступ до папки. Клієнт генерує випадковий 16-байтний ключ спільного доступу до папки (`rawFolderShareKeyBytes`). Цей ключ шифрується майстер-ключем користувача (`encryptedFolderShareSecretKeyForStorageB64`) (додаток О). Також, ключ самої кореневої спільної папки (`folderKeyWords`) дешифрується майстер-ключем, а потім шифрується за допомогою `rawFolderShareKeyBytes` (`encryptedFolderKeyForSharingB64`). Ці два зашифровані ключі надсилаються на сервер (`/api/folders/{folder_handle}/share` POST). Сервер зберігає їх та генерує `public_handle` для папки. Посилання для спільного доступу формується як `shared_folder.html#{public_handle}!{base64url_raw_folder_share_key}`. Після цього клієнт рекурсивно обходить вміст папки. Для кожної вкладеної папки та файлу їхні оригінальні ключі (папки або файлу) дешифруються майстер-ключем, а потім шифруються за допомогою `rawFolderShareKeyBytes`. Ці нові зашифровані ключі (`encrypted_item_key_for_sharing`) надсилаються на сервер для оновлення відповідних записів у БД (`/api/shared-folders/{public_handle}/items` POST). Коли гість відкриває посилання на спільну папку, він використовує `rawFolderShareKeyBytes` з URL-фрагмента для дешифрування `encrypted_item_key_for_sharing` кожного елемента в папці, що дозволяє отримати оригінальний ключ елемента для подальшого дешифрування імені та вмісту.

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

Зміна пароля вимагає від користувача введення нового пароля. Клієнт генерує нове `clientRandomValue`, нову сіль, новий похідний ключ (`newDerivedKeyBytes`) та, відповідно, новий ключ шифрування (`newDerivedEncryptionKeyBytes`) і новий хеш ключа автентифікації (`newHashedAuthenticationKeyForServer`). Поточний майстер-ключ (який зберігається в пам'яті клієнта) шифрується новим `newDerivedEncryptionKeyBytes`, утворюючи `newEncryptedMasterKeyBytes`. На сервер (`/api/change_password` POST) надсилаються: `new_client_random_value_b64`, `new_hashed_authentication_key_b64` та `new_encrypted_master_key_b64`. Сервер оновлює відповідні поля для користувача та деактивує всі інші активні сесії цього користувача, крім поточної.

Висновки до другого розділу

У цьому розділі було детально описано процес проєктування хмарного сховища з наскрізним шифруванням. Було визначено вимоги користувачів та системи, побудовано діаграми варіантів використання та класів для візуалізації взаємодії та структури. Спроектовано схему бази даних PostgreSQL для зберігання метаданих, наголошуючи на тому, що чутливі дані залишаються зашифрованими або не зберігаються на сервері взагалі.

Розглянуто ключові алгоритми роботи системи: реєстрація, автентифікація, завантаження/шифрування та скачування/дешифрування файлів. Наведено приклади реалізації цих алгоритмів на стороні клієнта за допомогою JavaScript коду, що демонструють використання Web Crypto API, Web Workers та криптографічних бібліотек для забезпечення наскрізного шифрування та перевірки цілісності даних. Описано взаємодію між основними компонентами системи згідно з діаграмою компонентів.

Таким чином, розроблена логічна модель та описані алгоритми створюють надійну основу для подальшої реалізації безпечного та функціонального хмарного сховища, що ставить на перше місце конфіденційність даних користувача.

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

РОЗДІЛ 3. ІНТЕРФЕЙС КОРИСТУВАЧА ТА ПОРЯДОК РОБОТИ З ХМАРНИМ СХОВИЩЕМ З НАСКРІЗНИМ ШИФРУВАННЯМ

3.1. Порядок встановлення та налаштування параметрів застосунку

Встановлення та налаштування хмарного сховища з наскрізним шифруванням вимагає розгортання всіх його компонентів та відповідного конфігурування для забезпечення коректної взаємодії та функціонування системи в цілому. Архітектура розгортання платформи, що ілюструє фізичне розміщення компонентів та їх взаємозв'язки, наведена на рис. 3.1.

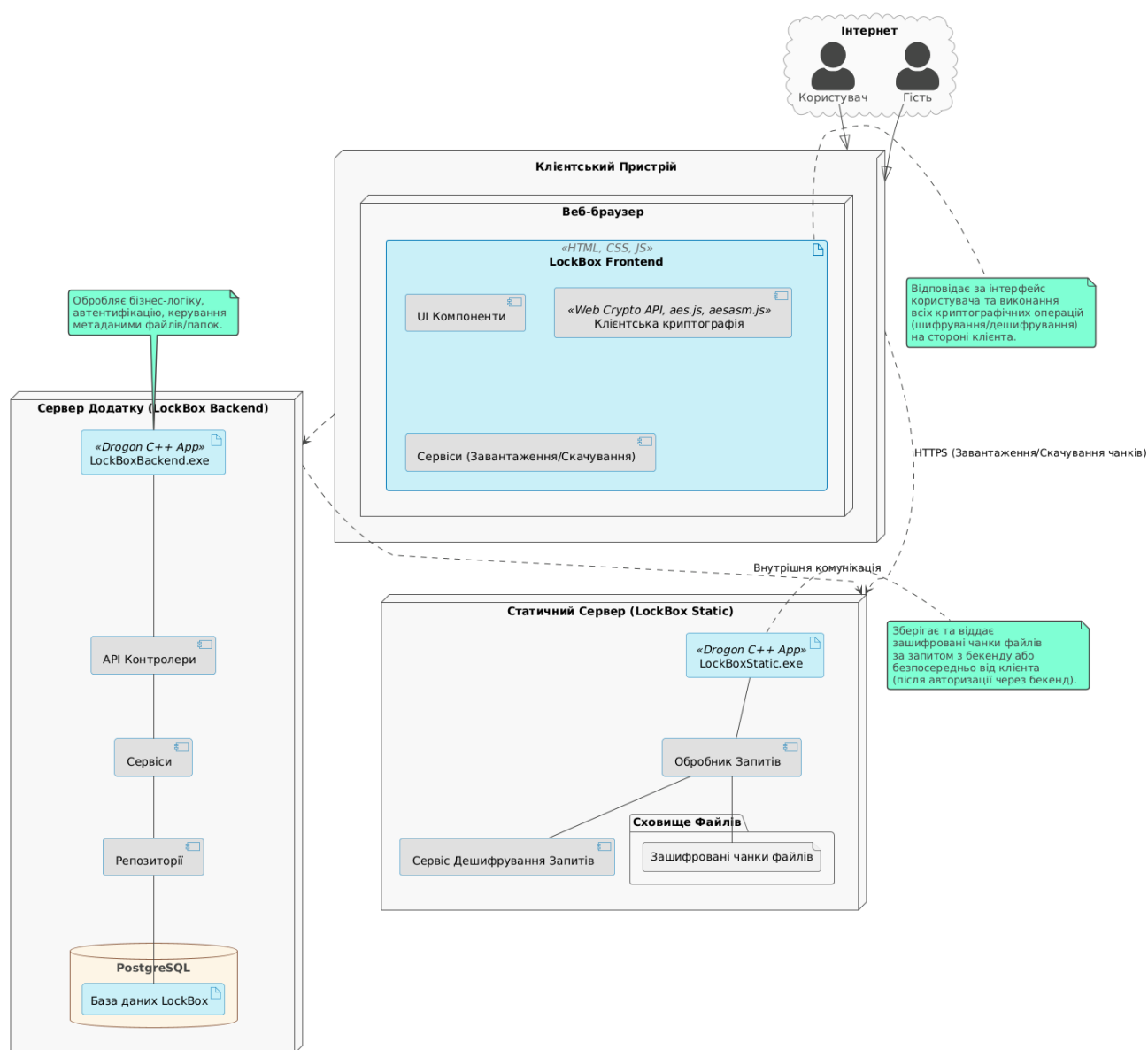


Рисунок 3.1 – Діаграма розгортання

Процес розгортання ключових компонентів системи є наступним.

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						43

Клієнтський пристрій (Веб-браузер) не вимагає спеціального встановлення програмного забезпечення, окрім наявності сучасного веббраузера з підтримкою JavaScript та Web Crypto API. Клієнтська частина додатку "LockBox Frontend", що складається з файлів HTML, CSS та JavaScript, розгортається на будь-якому статичному вебхостингу або може бути запущена локально. Всі криптографічні операції, пов'язані з даними користувача, такі як шифрування, дешифрування та генерація ключів, виконуються безпосередньо у браузері користувача.

Сервер додатку (LockBox Backend), реалізований як скомпільований виконуваний файл LockBoxBackend.exe на C++ з використанням фреймворку Drogon, розгортається на серверному вузлі під керуванням операційної системи, для якої був скомпільований. Для його роботи необхідна конфігурація через файл config.json, де вказуються параметри підключення до бази даних PostgreSQL, налаштування логування, мережеві параметри (IP-адреса, порт), а також параметри для сервісу надсилання електронних листів, наприклад, API ключ для сервісу Resend. Важливою частиною серверної інфраструктури є База даних PostgreSQL, для якої розгортається відповідний екземпляр СУБД. Схема бази даних, що включає таблиці users, folders, files, sessions, temporary_sessions та static_servers з відповідними індексами та тригерами, створюється згідно з описом у пункті 2.2. Також налаштовується доступ до БД для користувача, від імені якого працюватиме бекенд-додаток.

Статичний Сервер (LockBox Static), також реалізований як виконуваний файл LockBoxStatic.exe на C++ з Drogon, розгортається на одному або декількох серверних вузлах. Його конфігурація через config.json включає шлях до локального сховища файлів (storage_base_path), розмір чанків за замовчуванням та шлях до приватного RSA-ключа сервера, який використовується для дешифрування авторизаційних токенів у запитах. Для зберігання зашифрованих чанків файлів готується відповідна директорія на файловій системі сервера (storage_base_path).

Для забезпечення мережевої взаємодії необхідно налаштувати доступність Сервера Додатку та Статичного Сервера через Інтернет, переважно з

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

використанням HTTPS для захисту каналу передачі даних. Це може включати налаштування DNS-записів, конфігурацію брандмауерів та, за потреби, зворотних проксі-серверів для обробки вхідних запитів, термінації SSL/TLS та балансування навантаження.

Після розгортання та налаштування всіх компонентів, хмарне сховище з наскрізним шифруванням готове до роботи, забезпечуючи користувачам можливість безпечно зберігати та обмінюватися файлами.

3.2. Загальна структура інтерфейсу користувача

Інтерфейс користувача розроблено з акцентом на інтуїтивність, простоту використання та забезпечення швидкого доступу до основного функціоналу. Він складається з кількох ключових сторінок та компонентів.

Основною робочою областю після автентифікації користувача є файловий менеджер (реалізований у fm.html та керований fm_ui.js). Його інтерфейс можна умовно розділити на такі зони:

Бічна панель навігації: Розташована ліворуч, містить посилання для переходу до основного розділу "Мої файли", кнопки для доступу до налаштувань безпеки ("Безпека"), експорту ключа відновлення, керування активними сесіями, інформаційний блок про використання дискового простору (квота та зайняте місце) та кнопку виходу з системи (рис. 3.2).

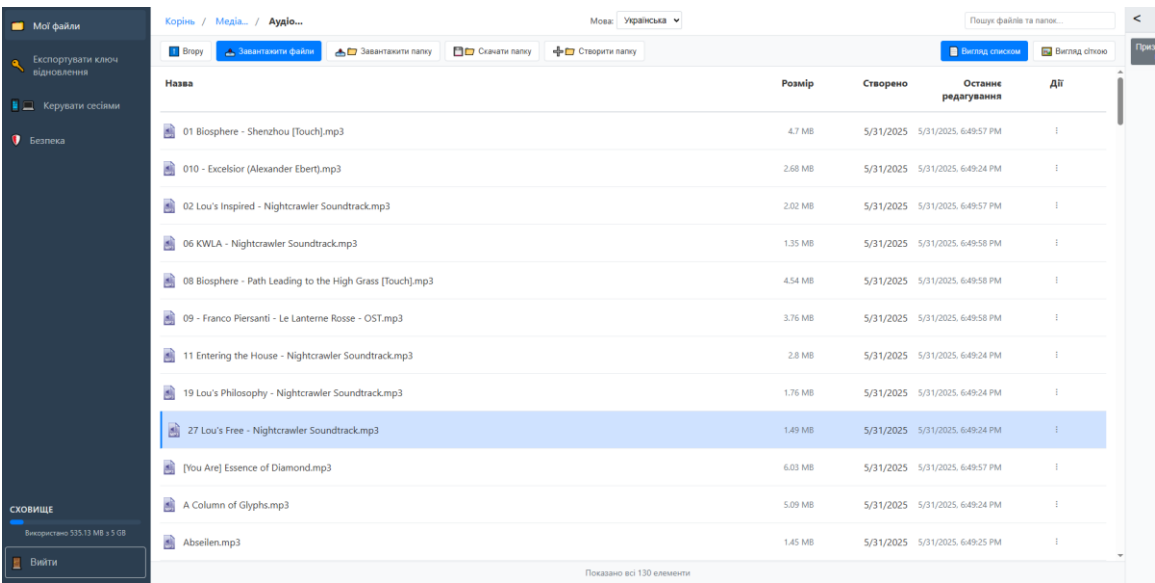


Рисунок 3.2 – Файловий менеджер

Основна область контенту: Займає центральну частину екрана та динамічно змінюється залежно від дій користувача.

Заголовок основної області: Відображає "хлібні крихти" для навігації по ієрархії папок; поле для пошуку файлів та папок по поточній директорії; та перемикач мови інтерфейсу (рис. 3.2).

Панель інструментів: Знаходиться під заголовком та містить кнопки для виконання основних операцій: "Вгору" (перехід до батьківської папки), "Завантажити Файли", "Завантажити Папку", "Створити Папку", "Скачати Поточну Папку" (як ZIP-архів), а також кнопки для перемикання вигляду відображення елементів – "Список" або "Сітка" (рис. 3.2).

Область відображення файлів та папок: Центральна частина, де відображається вміст поточної папки у вигляді списку або сітки мініатюр. Для кожного елемента відображається іконка, ім'я, а для файлів – розмір та дата модифікації/створення (рис. 3.2 – рис. 3.3).

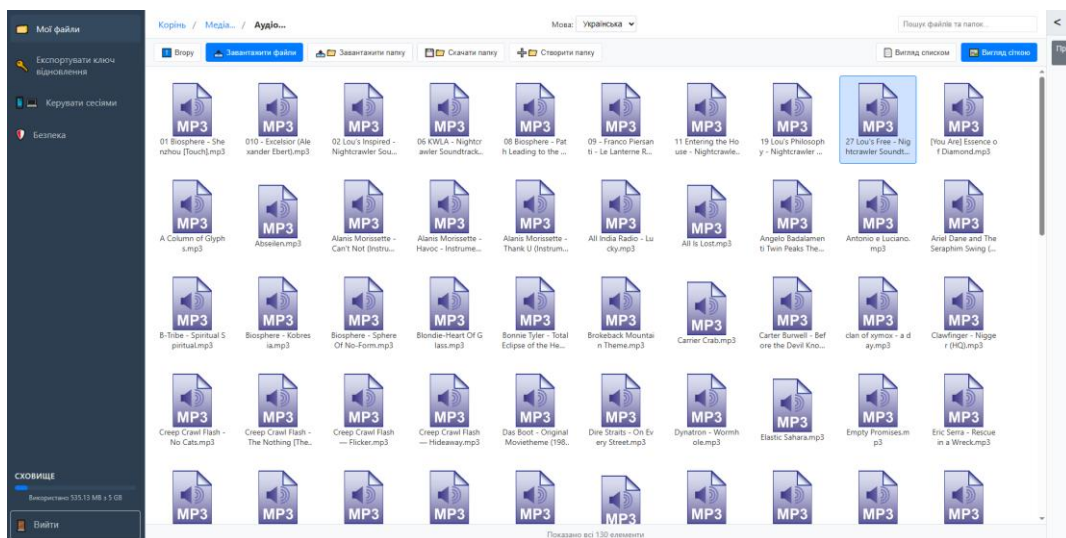


Рисунок 3.3 – Вигляд "Сітка"

Нижній колонтитул (статус-бар): Відображає інформацію про кількість елементів у поточній папці та загальну кількість вибраних елементів.

Панель активних передач: Розташована праворуч (може бути згорнута), відображає прогрес поточних операцій завантаження файлів на сервер та скачування файлів з сервера, надає можливість призупинити, відновити або скасувати окремі чи всі передачі (рис. 3.4).

		Кондратюк О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

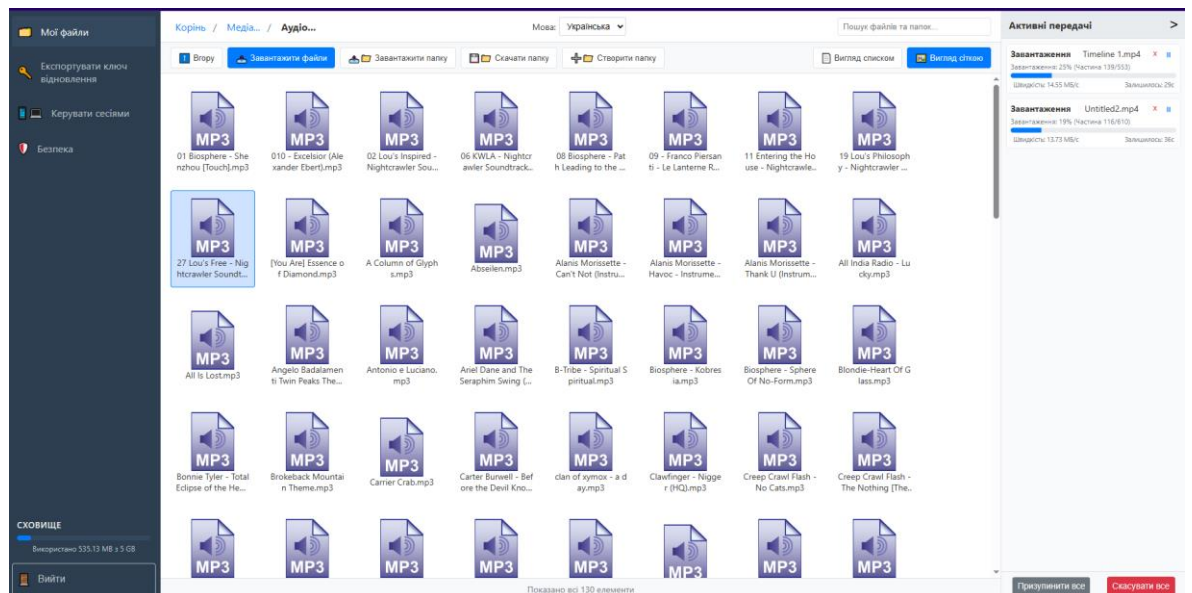


Рисунок 3.4 – Панель активних передач

Сторінки реєстрації (рис. 3.5) (register.html) та входу (рис. 3.6) (login.html) мають стандартний для вебдодатків дизайн: форми для введення email та пароля, поля для повторного введення пароля (при реєстрації), чекбокс для згоди з умовами використання, кнопки для надсилення форми та посилення для переходу між сторінками реєстрації та входу. Передбачено відображення повідомлень про помилки валідації або помилки сервера. На сторінці реєстрації також присутній індикатор надійності пароля.

Рисунок 3.5 – Сторінка реєстрації

		Кондратюк.О.Д						Арк.
		Локтікова Т.М.						47
Змн.	Арк.	№ докум.	Підпис	Дата				

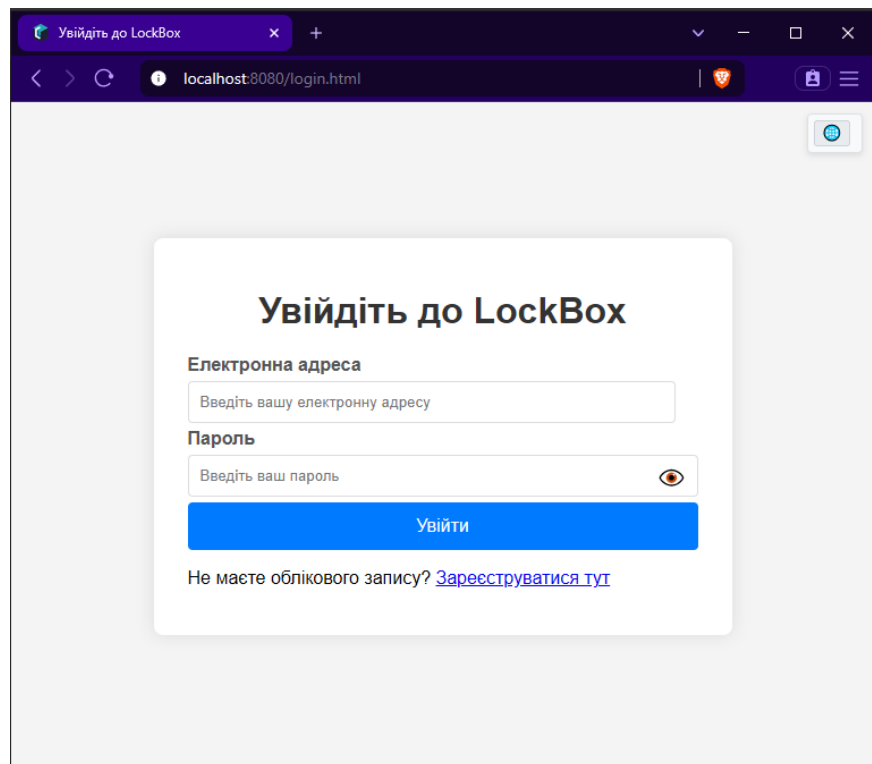


Рисунок 3.6 – Сторінка входу

Сторінка спільного файлу (рис. 3.7) (shared_file.html): Призначена для гостей, які переходять за посиланням на окремий файл. Відображає назву файлу (дешифровану на клієнті), його розмір та кнопку для завантаження. Також показує прогрес дешифрування та скачування.

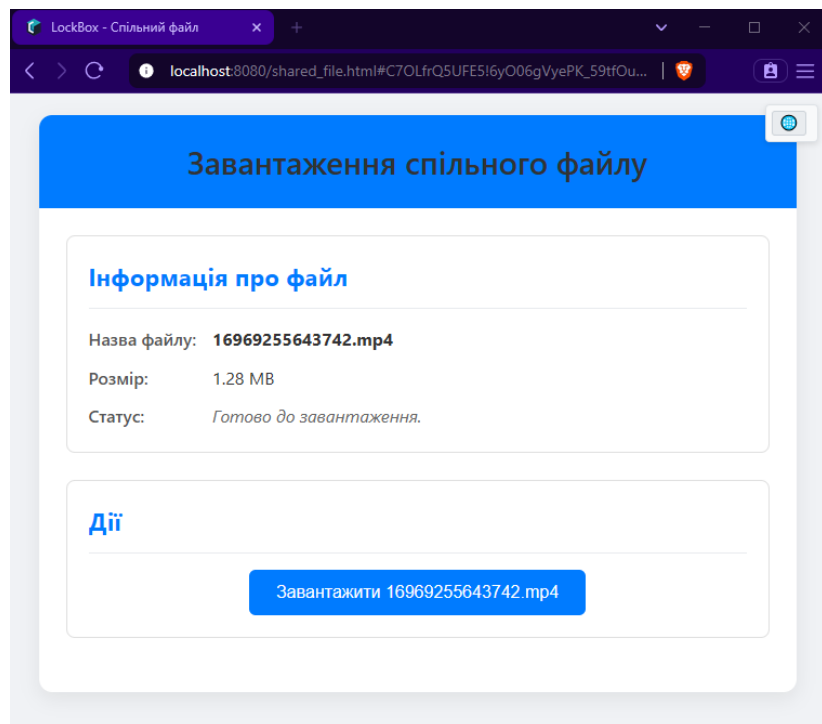


Рисунок 3.7 – Сторінка спільного файлу

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

Сторінка спільної папки (рис. 3.8) (shared_folder.html): Аналогічно файловому менеджеру, дозволяє гостю переглядати вміст спільної папки (імена дешифруються на клієнті), переходити у підпапки та завантажувати окремі файли або всю поточну папку як ZIP-архів. Функціонал редагування та завантаження нового контенту для гостя відсутній.

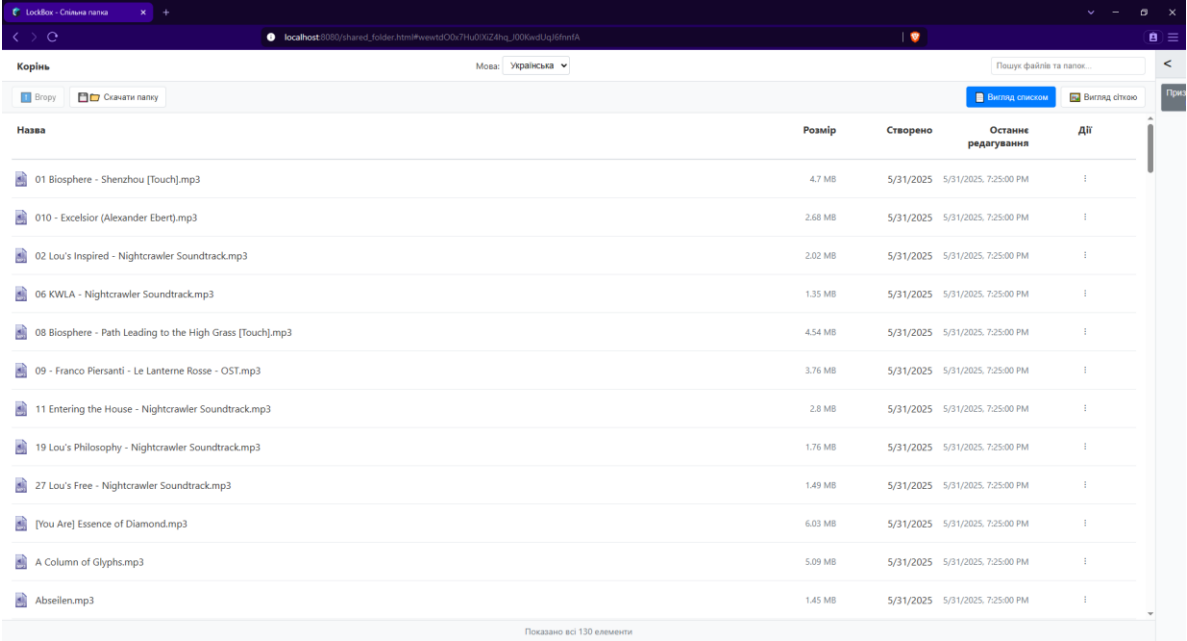


Рисунок 3.8 – Сторінка спільної папки

Система використовує модальні вікна для виконання специфічних дій та отримання підтверджень від користувача. Наприклад:

- створення нової папки (рис. 3.9) (запит імені папки);
- перейменування файлу/папки (запит нового імені);
- підтвердження видалення одного або декількох елементів;
- відображення згенерованого посилання для спільного доступу до файлу або папки з кнопкою копіювання (рис. 3.10);
- експорт майстер-ключа відновлення (відображення ключа та кнопка копіювання);
- керування сесіями (рис. 3.11) (список активних сесій з інформацією про IP, User-Agent, час активності та кнопками для деактивації);
- підтвердження виходу з системи.

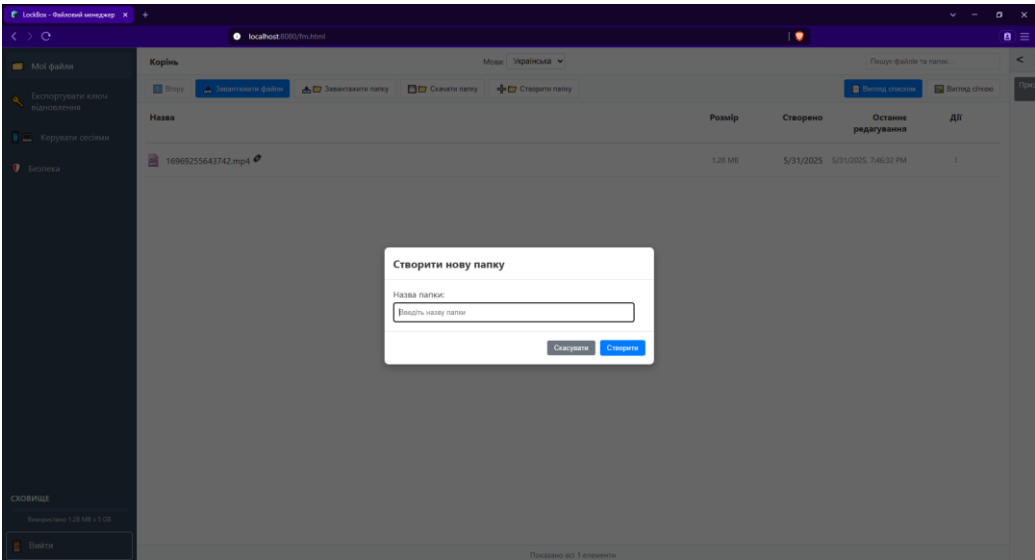


Рисунок 3.9 – Створення нової папки

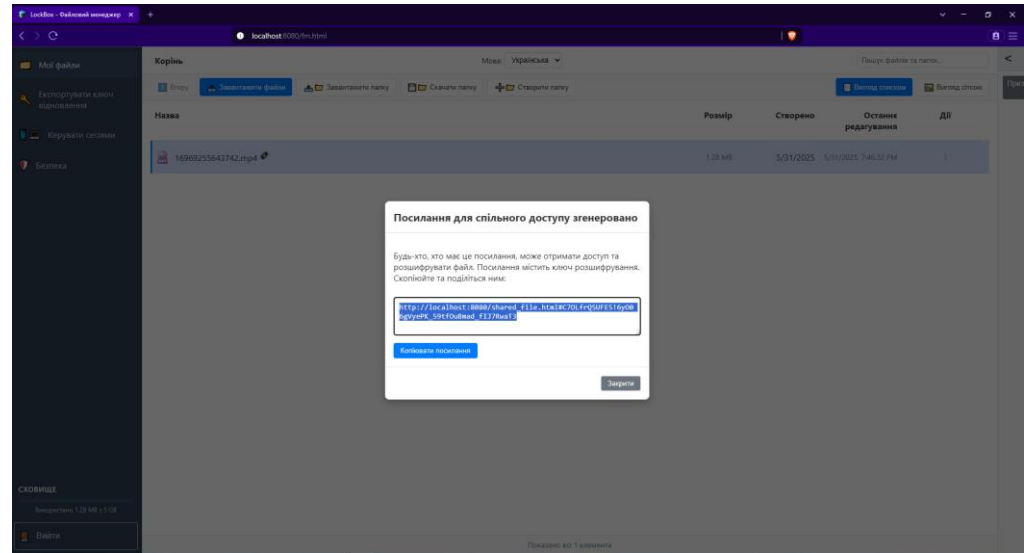


Рисунок 3.10 – Відображення посилання для спільного доступу

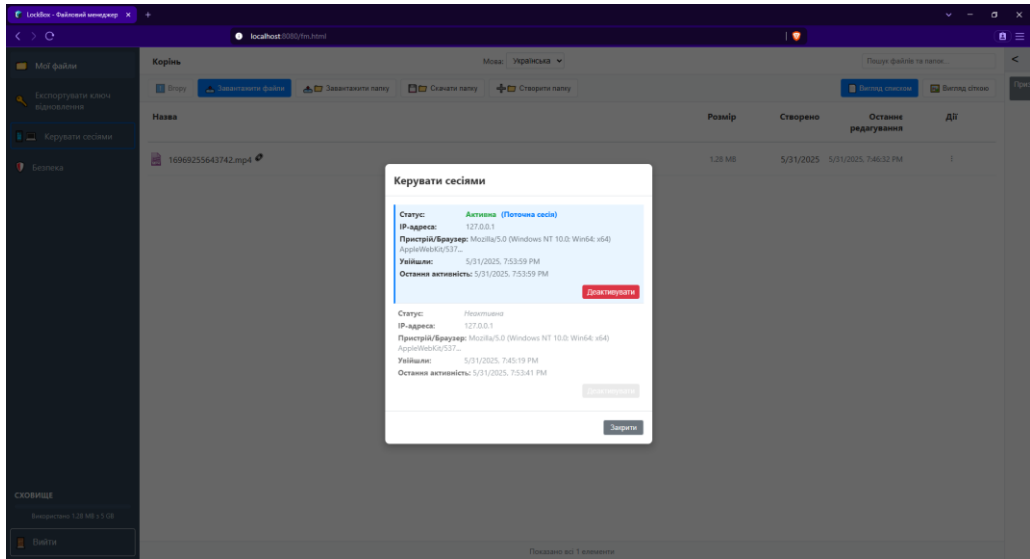


Рисунок 3.11 – Керування сесіями

Окремий розділ інтерфейсу в файловому менеджері присвячений налаштуванням безпеки (рис. 3.12) (fm.html, активується через бічну панель). Тут користувач може змінити свою адресу електронної пошти (з обов'язковим підтвердженням нової адреси через лист) та змінити свій пароль (з підтвердженням нового пароля та індикатором його надійності).

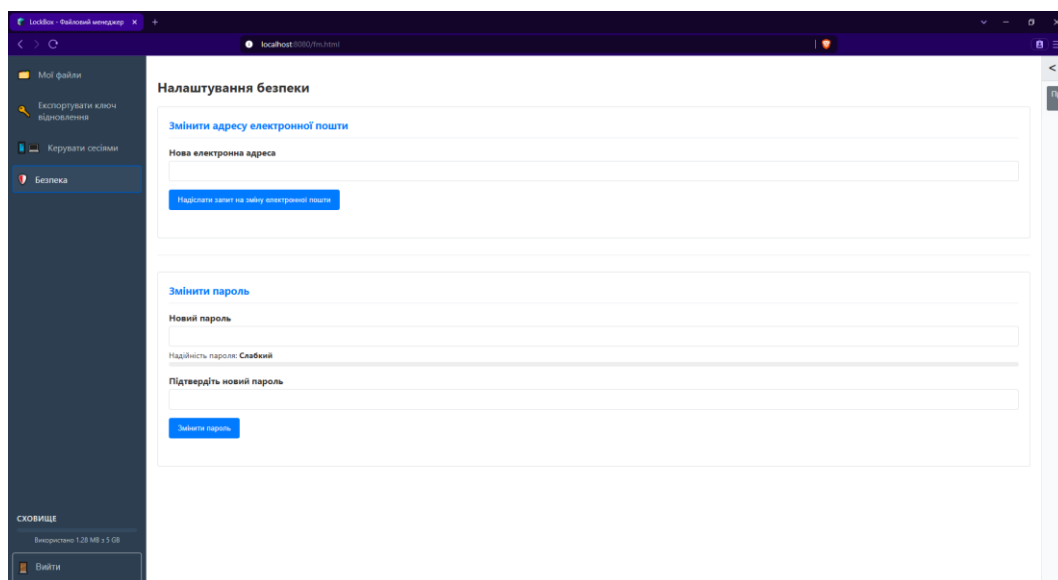


Рисунок 3.12 – Налаштування безпеки

3.3. Порядок роботи користувача з системою

Взаємодія Користувача з хмарним сховищем охоплює повний цикл роботи з даними, починаючи від створення облікового запису.

Реєстрація та перший вхід в систему передбачає наступні кроки; користувач переходить на сторінку реєстрації (рис. 3.13) (register.html), вводить свою актуальну адресу електронної пошти, створює надійний пароль та повторює його для підтвердження; погоджується з умовами використання сервісу та натискає кнопку створення акаунту; на клієнтській стороні відбувається генерація криптографічних ключів, і запит надсилається на сервер; система інформує користувача про необхідність підтвердити електронну пошту (рис. 3.14), надіславши лист з унікальним посиланням.

		Кондратюк, О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

Створіть свій обліковий запис LockBox

Електронна адреса
privacy-enjoyer@proton.me

Пароль
.....

Надійність пароля: **Сильний**

Повторіть пароль
.....

☒ Я погоджуюся з [Умовами та Положеннями](#)

Створити обліковий запис

Вже маєте обліковий запис? [Увійти тут](#)

Рисунок 3.13 – Реєстрація нового користувача

Створіть свій обліковий запис LockBox

На вашу адресу надіслано електронний лист із підтвердженням. Будь ласка, перевірте свою поштову скриньку та виконайте інструкції для завершення реєстрації.

[Перейти на сторінку входу](#)

Рисунок 3.14 – Підтвердження пошти

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

Після переходу за посиланням обліковий запис активується (рис. 3.15); при першій успішній автентифікації на сторінці входу (login.html), якщо для акаунту ще не зареєстровано RSA-ключі, система автоматично генерує їх на клієнті та безпечно реєструє на сервері для подальшого захисту сесійних даних; після цього користувач потрапляє до основного інтерфейсу файлового менеджера.

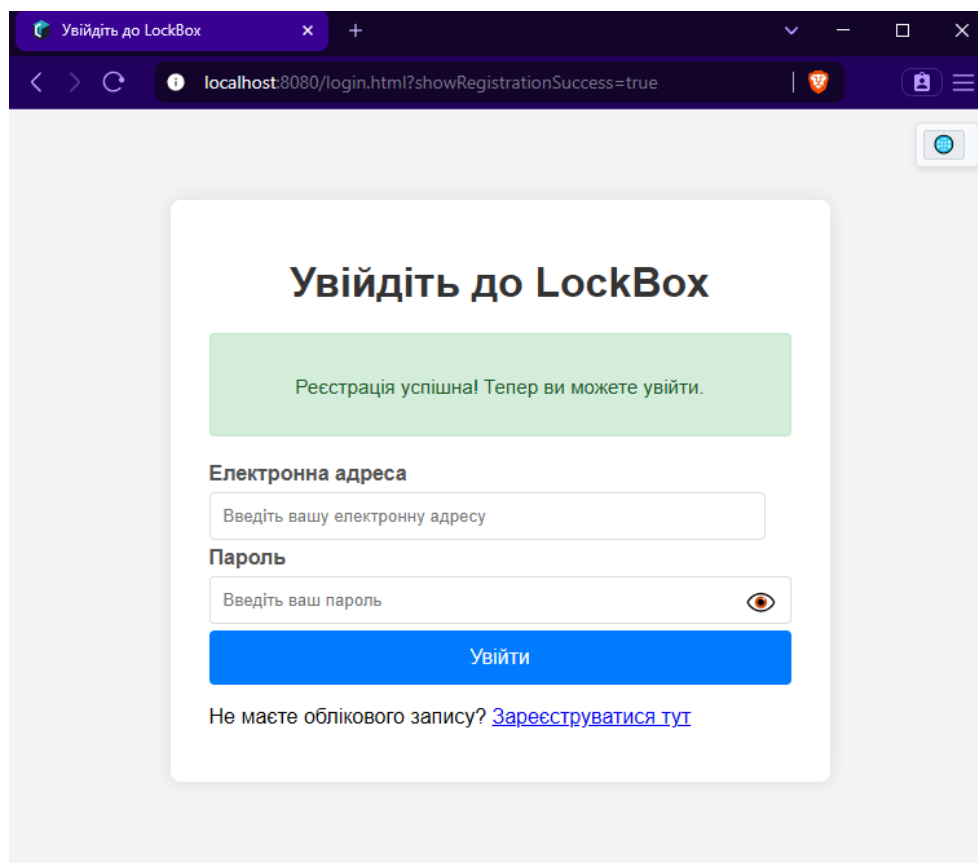


Рисунок 3.15 – Успішна реєстрація

Робота з файлами та папками в файловому менеджері (рис. 3.2) (fm.html) відбувається наступним чином; користувач може переміщуватися по ієрархії папок, використовуючи список елементів у центральній області, "хлібні крихти" у заголовку або кнопку "Вгору" на панелі інструментів; для створення нової папки користувач натискає відповідну кнопку на панелі інструментів, вводить ім'я папки у модальному вікні (рис. 3.9) та підтверджує створення; завантаження файлів можливе як через натискання кнопки "Завантажити Файли" та вибір файлів у системному діалозі, так і шляхом перетягування файлів (рис. 3.16) з комп'ютера користувача безпосередньо в область списку файлів.

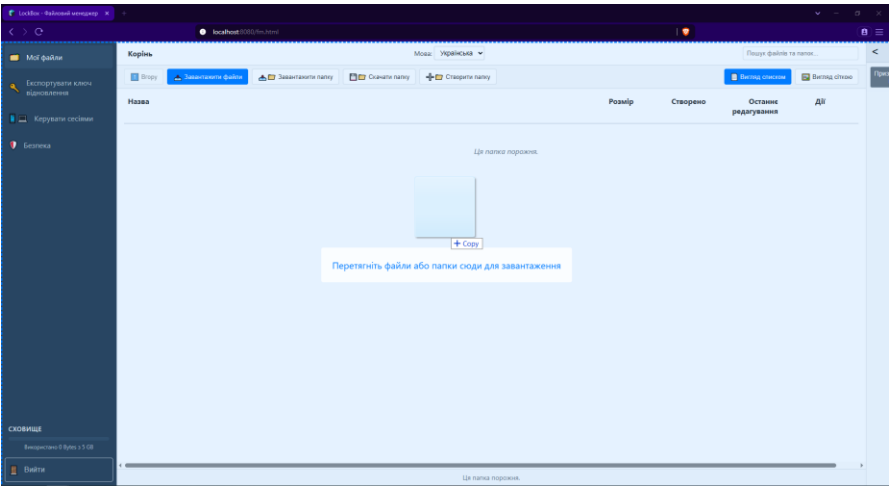


Рисунок 3.16 – Завантаження файлів методом перетягування

Аналогічно, завантаження цілої папки (рис. 3.17 – рис. 3.18) з усім її вмістом ініціюється кнопкою "Завантажити Папку". Прогрес завантаження відображається на панелі активних передач.

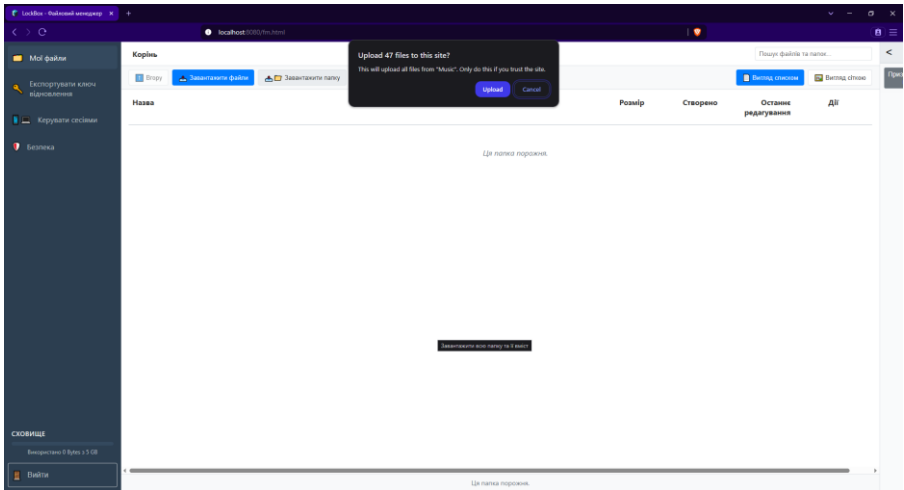


Рисунок 3.17 – Підтвердження на завантаження папки

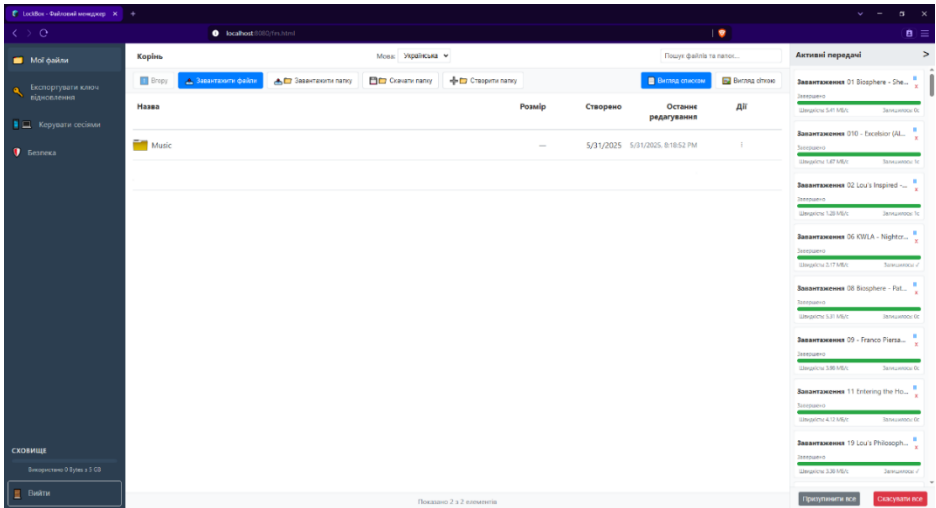


Рисунок 3.18 – Прогрес завантаження папки

Для скачування файлу користувач обирає його у списку та використовує відповідний пункт контекстного меню (рис. 3.19) або натискає на нього подвійним кліком.

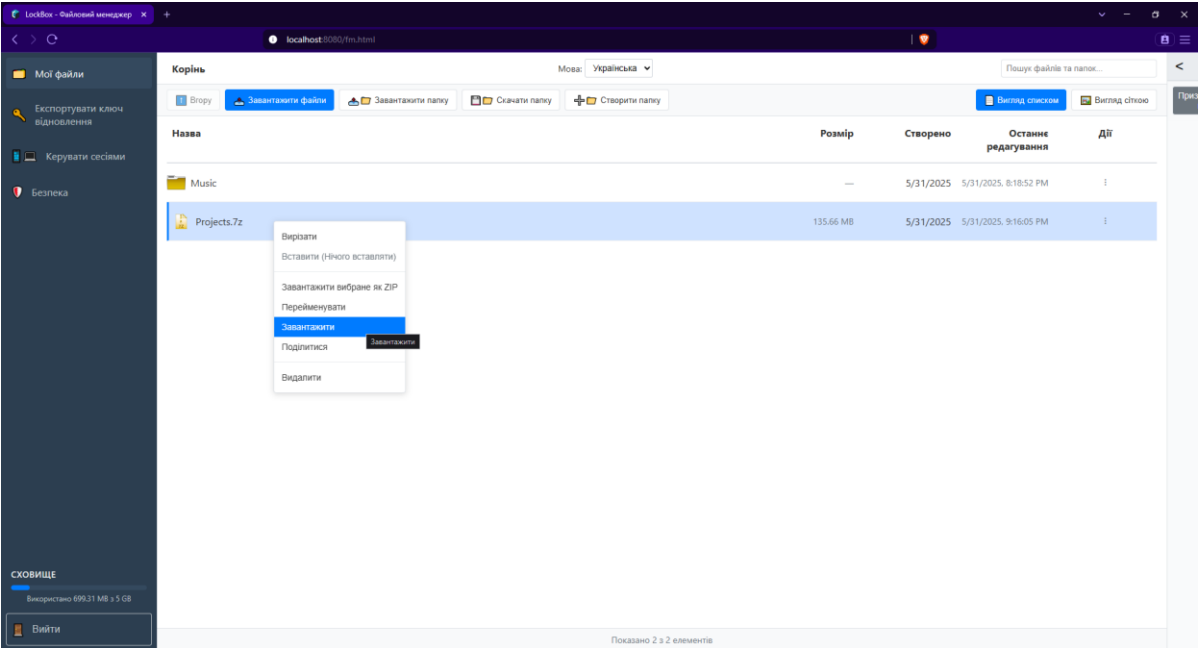


Рисунок 3.19 – Контекстне меню файлового менеджера

Для скачування вмісту поточної папки або кількох вибраних елементів як ZIP-архіву (рис. 3.20), користувач використовує кнопку "Скачати папку" або відповідну опцію контекстного меню.

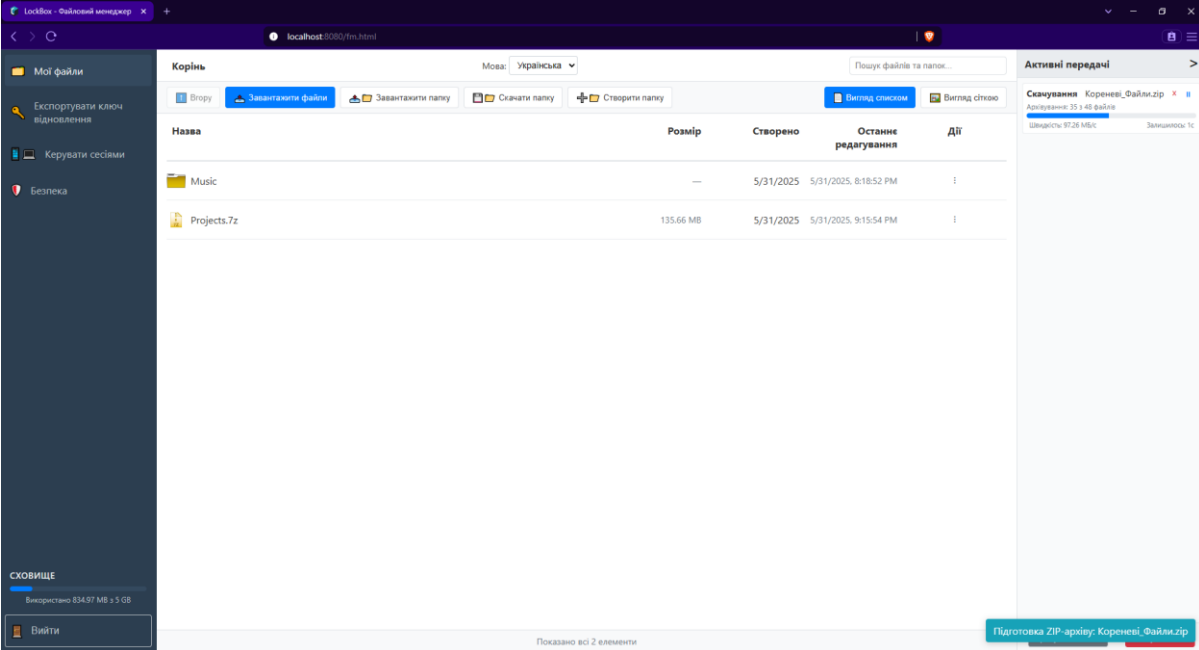


Рисунок 3.20 – Скачування файлів як ZIP-архів

Перейменування файлу чи папки здійснюється через контекстне меню обраного елемента, де у модальному вікні вводиться нове ім'я (рис. 3.21).

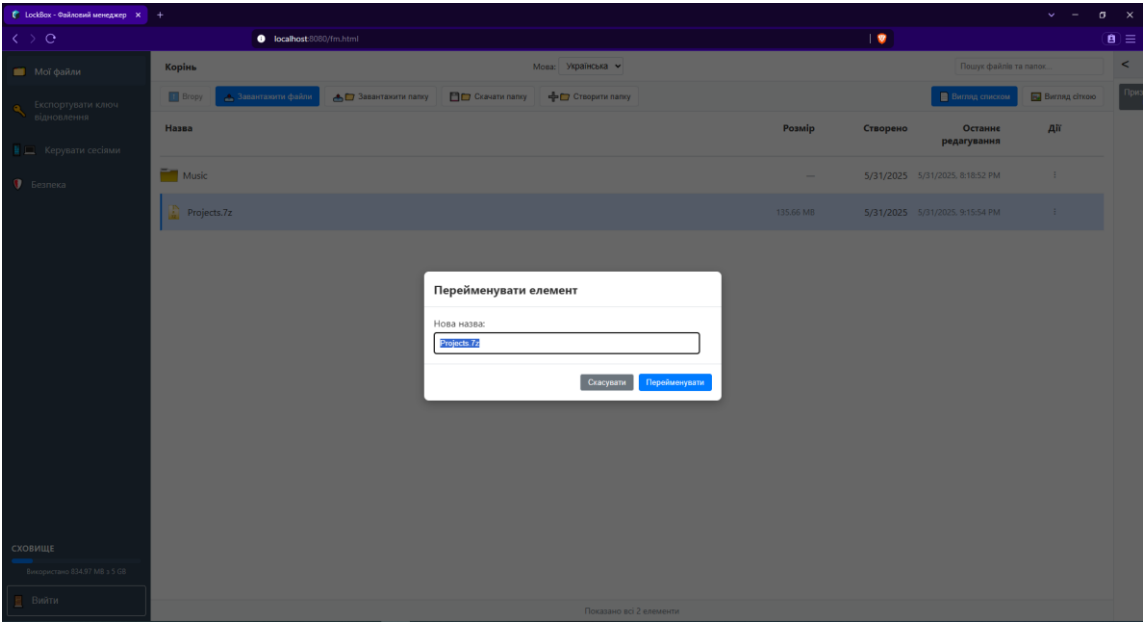


Рисунок 3.21 – Перейменування елемента

Переміщення файлів та папок можливе як за допомогою функцій "Вирізати" та "Вставити" з контекстного меню, так і шляхом прямого перетягування (drag-and-drop) обраних елементів на цільову папку або на один із сегментів "хлібних крихт" (рис. 3.22).

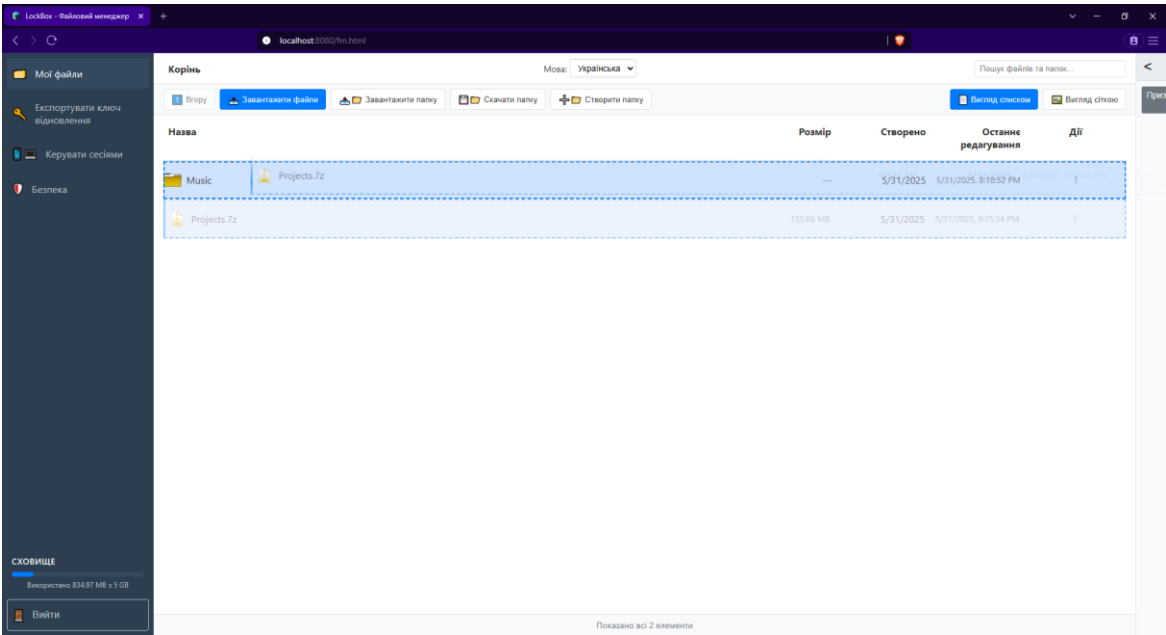


Рисунок 3.22 – Використання drag-and-drop для переміщення елемента(ів)

Видалення файлів або папок також ініціюється через контекстне меню, після чого з'являється модальне вікно для підтвердження операції (рис. 3.23).

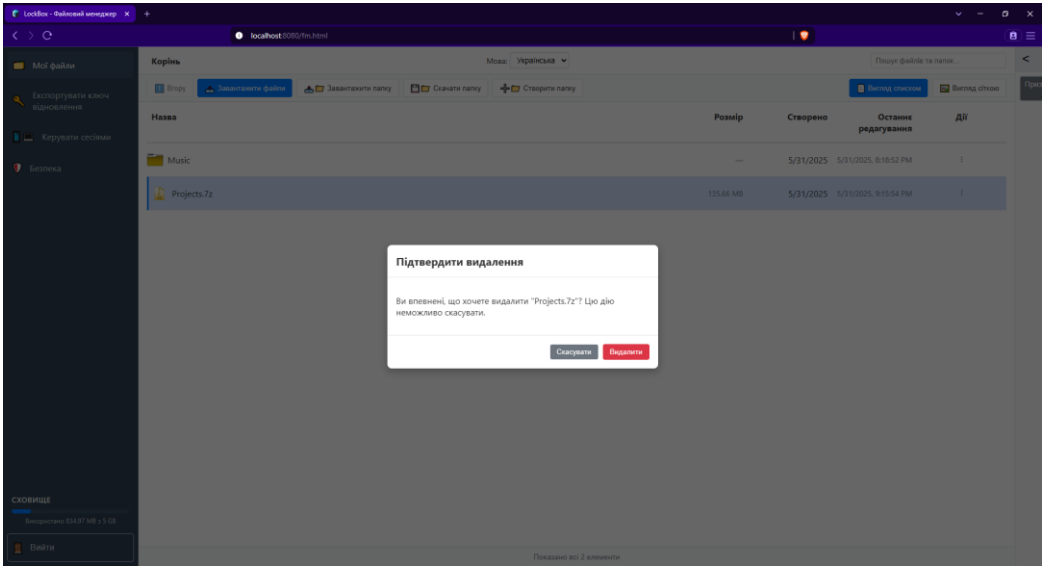


Рисунок 3.23 – Видалення елемента

Процедура надання спільного доступу (рис. 3.10) до файлу або папки є простою; користувач обирає потрібний елемент, викликає контекстне меню та обирає пункт "Поділитися" (або "Поділитися папкою"); система генерує унікальне посилання, яке включає ключ дешифрування у URL-фрагменті, та відображає його у модальному вікні разом з кнопкою для копіювання. Для скасування спільного доступу (рис. 3.24) користувач обирає відповідний пункт у контекстному меню вже поширеного елемента, після чого публічне посилання стає недійсним.

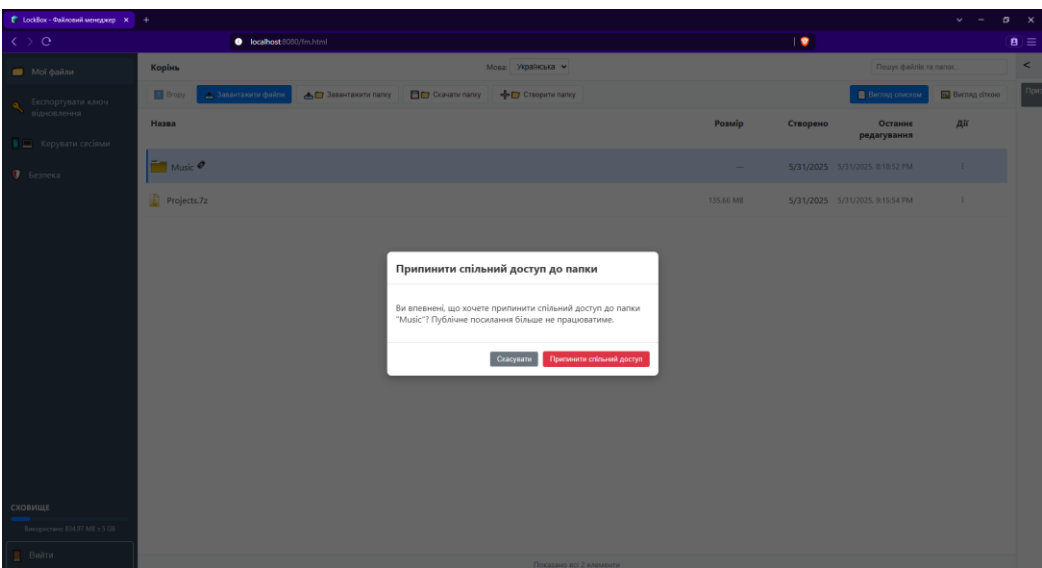
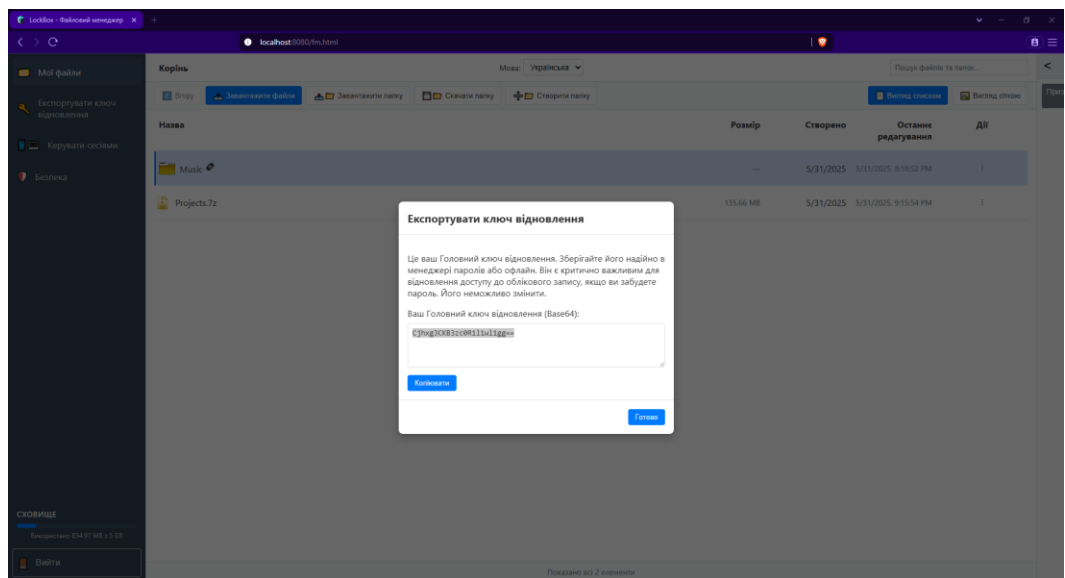


Рисунок 3.24 – Припинення спільного доступу

Експорт ключа відновлення (рис. 3.25) ініціюється відповідною кнопкою, після чого у модальному вікні відображається майстер-ключ у форматі Base64 для копіювання та безпечного зберігання.



Керування сесіями дозволяє переглянути список усіх активних пристроїв та браузерів, з яких здійснювався вхід, з інформацією про IP-адресу, тип пристрою та час останньої активності, та надає можливість деактивувати будь-яку сесію, включаючи поточну.

Вихід з системи (рис. 3.26) здійснюється натисканням кнопки "Вихід" на бічній панелі, після чого поточна сесія користувача деактивується на сервері, а локальні сесійні дані (майстер-ключ, токен сесії) видаляються з браузера.

		Кондратюк.О.Д			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

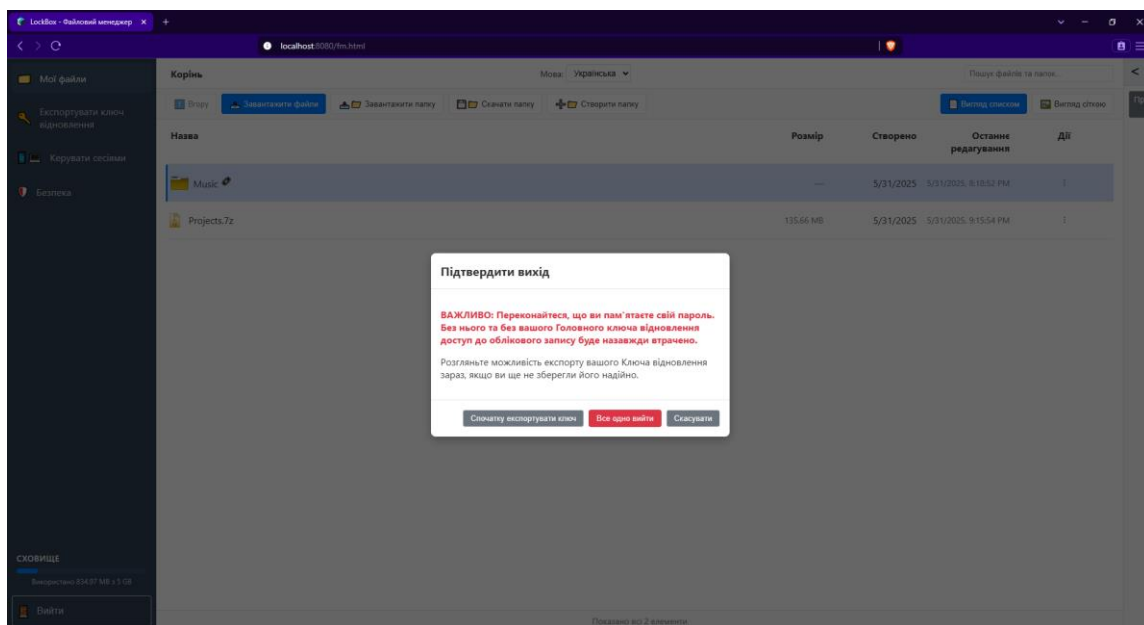


Рисунок 3.26 – Підтвердження виходу

3.4. Порядок роботи гостя зі спільним вмістом

Гість, не маючи облікового запису в системі, може взаємодіяти з файлами та папками, до яких йому надав доступ інший користувач.

При переході за посиланням на спільний файл, браузер гостя завантажує спеціальну сторінку; клієнтський скрипт витягує публічний ідентифікатор файлу (`public_handle`) з шляху URL та ключ дешифрування файлу з URL-фрагмента (`key_fragment`); за допомогою `public_handle` запитується інформація про файл у сервера (зашифроване ім'я, розмір) та посилання для завантаження зашифрованого контенту зі статичного сервера; отримане зашифроване ім'я файлу дешифрується на стороні Гостя ключем з URL-фрагмента та відображається разом з розміром файлу. При натисканні кнопки "Завантажити" починається процес скачування зашифрованих частин файлу зі статичного сервера (рис. 3.27), їх дешифрування на льоту за допомогою ключа з URL-фрагмента та перевірка цілісності кожної частини. Після успішного завантаження та дешифрування всіх частин, файл зберігається на пристрій гостя.

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

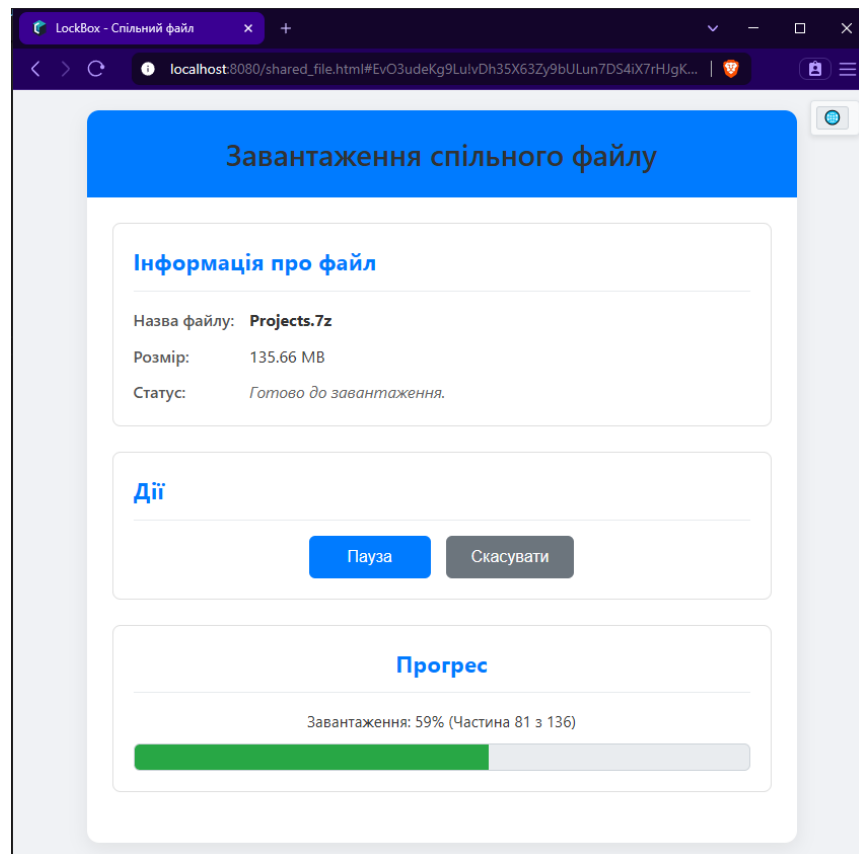


Рисунок 3.27 – Процес завантаження спільного файлу

При переході за посиланням на спільну папку, браузер гостя завантажує інтерфейс, схожий на файловий менеджер, але з обмеженим функціоналом; клієнтський скрипт отримує `public_handle` папки та ключ спільного доступу до папки (`rawFolderShareKeyBytes`) з URL-фрагмента; використовуючи `public_handle`, запитується список вмісту папки у сервера; сервер повертає список елементів, де для кожного елемента вказано його зашифроване ім'я та зашифрований ключ цього елемента (який був зашифрований `rawFolderShareKeyBytes` власником при наданні доступу); клієнт гостя використовує `rawFolderShareKeyBytes` для дешифрування ключа кожного елемента, а потім цим дешифрованим ключем елемента дешифрує його ім'я. Гостю відображається список файлів та підпапок всередині спільної папки. Гість може переходити до підпапок (де процес отримання та дешифрування вмісту повторюється для нової поточної папки, використовуючи той самий початковий `rawFolderShareKeyBytes` для дешифрування ключів елементів), а також скачувати окремі файли (процес аналогічний скачуванню окремого спільного файлу, де ключ файлу отримується шляхом дешифрування

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

encrypted_item_key_for_sharing за допомогою rawFolderShareKeyBytes) або всю поточну видиму папку як ZIP-архів (архівація відбувається на стороні клієнта після дешифрування всіх файлів) (рис. 3.28).

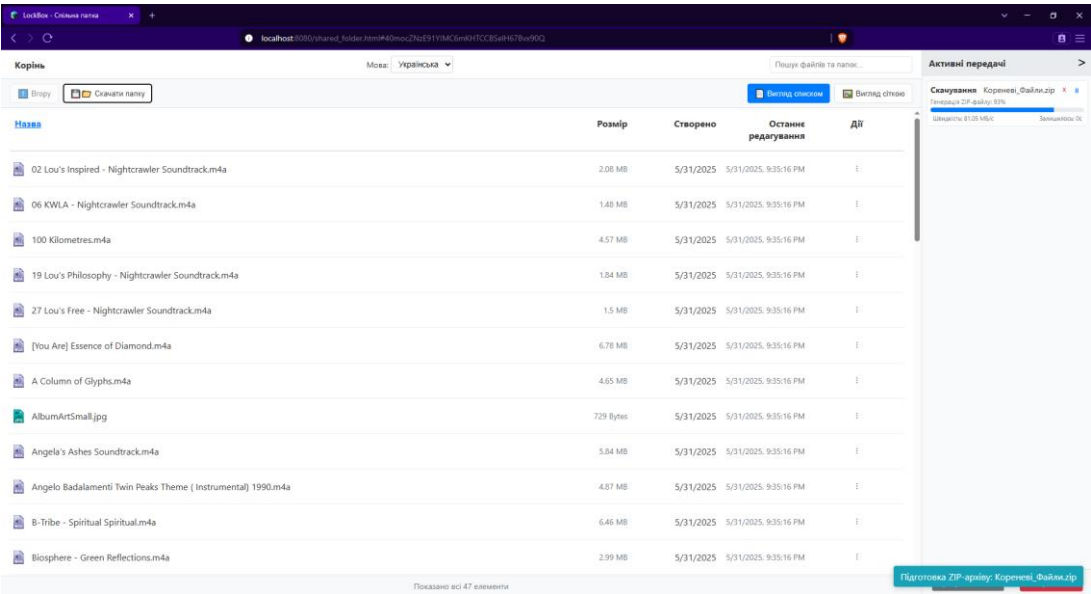


Рисунок 3.28 – Процес скачування спільної папки

Висновки до третього розділу

У цьому розділі було детально розглянуто користувацький інтерфейс хмарного сховища з наскрізним шифруванням та основні сценарії взаємодії з ним як для зареєстрованих користувачів, так і для гостей, що отримують доступ до спільного контенту. Описано структуру ключових сторінок: файлового менеджера з його компонентами (бічна панель, основна область контенту, панель інструментів, панель передач), сторінок реєстрації та автентифікації, сторінок перегляду спільних файлів та папок, а також розділу налаштувань безпеки.

Наведено порядок виконання основних операцій: реєстрації нового користувача з генерацією криптографічних ключів на клієнті та підтвердженням електронної пошти; двоетапної автентифікації з отриманням та дешифруванням майстер-ключа; керування файлами та папками, включаючи завантаження (з шифруванням на клієнті), скачування (з дешифруванням на клієнті), створення, перейменування, переміщення та видалення елементів. Окремо розглянуто механізми надання та отримання спільного доступу до файлів та папок, що

базуються на передачі ключів дешифрування через URL-фрагмент. Також описано функції керування обліковим записом, такі як зміна пароля та пошти, експорт ключа відновлення та керування сесіями.

Деталізація сценаріїв роботи дозволяє зрозуміти логіку взаємодії користувача з інтерфейсом та базові принципи роботи системи з точки зору кінцевого споживача послуги, підкреслюючи наскрізне шифрування та контроль користувача над своїми даними на кожному етапі.

		Кондратюк. О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було успішно спроектовано та розроблено програмну систему – хмарне сховище з реалізацією наскрізного шифрування. Головною метою роботи було створення безпечного, приватного та зручного у використанні рішення для зберігання, синхронізації та обміну файлами, де контроль над криптографічними ключами та, відповідно, доступом до даних, повністю належить користувачеві.

На першому етапі роботи було проведено детальний аналіз предметної області та існуючих ринкових аналогів хмарних сховищ. Це дозволило виявити ключову проблему сучасних популярних сервісів – відсутність наскрізного шифрування за замовчуванням, що створює потенційні ризики для конфіденційності даних користувачів. На основі цього аналізу було сформульовано функціональні та нефункціональні вимоги до розроблюваної системи, з особливим акцентом на механізми безпеки, приватності, надійності та зручності користувацького інтерфейсу. Також було обґрунтовано вибір технологічного стеку: C++ з веб-фреймворком Drogon для серверної частини та сервера зберігання статичних файлів, JavaScript (Vanilla JS) з Web Crypto API для клієнтської частини, та СУБД PostgreSQL для зберігання метаданих.

Другий розділ роботи був присвячений проектуванню архітектури системи. Було визначено основних акторів (Користувач, Гість) та розроблено варіанти їх використання, що охоплюють усі ключові функції системи: від реєстрації та автентифікації до складних операцій з файлами та папками, включаючи безпечний спільний доступ. Розроблено логічну структуру класів, що представляють основні сутності системи, такі як User, File, Folder, Session. Детально спроектовано структуру бази даних PostgreSQL, оптимізовану для зберігання метаданих та зашифрованих ключів, забезпечуючи при цьому цілісність та безпеку інформації. Ключову увагу було приділено розробці та опису алгоритмів, що лежать в основі функціонування системи: процесів реєстрації та автентифікації користувача з генерацією та управлінням криптографічними ключами на стороні клієнта (майстер-ключ, похідні ключі).

		Кондратюк О.Д.			ІПЗ.КР.Б-121-25-ІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

RSA-ключі для сесій), наскрізного шифрування (AES-CCM для файлів, AES-CBC для імен, AES-ECB для ключів) та дешифрування файлів і метаданих, а також забезпечення цілісності даних за допомогою MAC-тегів.

У третьому розділі було описано реалізований користувацький інтерфейс та порядок взаємодії користувачів з платформою. Представлено структуру основних сторінок (файловий менеджер, сторінки реєстрації/входу, сторінки спільного доступу, налаштування безпеки) та модальних вікон, що забезпечують інтуїтивно зрозумілий доступ до всього функціоналу. Детально розглянуто сценарії виконання ключових операцій, таких як завантаження та скачування файлів і папок (з архівацією у ZIP на клієнті для папок), створення та організація структури папок, механізми надання та отримання спільного доступу, а також функції керування обліковим записом, включаючи зміну пароля, електронної пошти, експорт ключа відновлення та керування сесіями.

Розроблена система хмарного сховища успішно реалізує поставлену мету – забезпечення високого рівня безпеки та конфіденційності даних користувачів завдяки архітектурі наскрізного шифрування, де всі криптографічні операції з ключами та даними відбуваються на стороні клієнта. Серверна частина не має доступу до незашифрованих даних чи майстер-ключів користувачів, що мінімізує ризики витоку інформації. Проєкт демонструє практичне застосування сучасних криптографічних стандартів та вебтехнологій для створення надійного та приватного хмарного рішення.

Подальший розвиток проєкту може включати розширення функціоналу спільного доступу (наприклад, встановлення прав доступу, часових обмежень на посилання), реалізацію версіонування файлів, інтеграцію з іншими сервісами, а також розробку мобільних додатків для різних платформ.

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dropbox [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.dropbox.com/>
2. Google Drive [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.google.com/drive/>
3. How Drive protects your privacy & keeps you in control. Google. URL:
<https://support.google.com/drive/answer/10375054>.
4. Microsoft OneDrive [Електронний ресурс] – Режим доступу до ресурсу:
<https://onedrive.live.com/>
5. C++ Standard [Електронний ресурс] – Режим доступу до ресурсу:
<https://isocpp.org/>
6. Drogon Web Framework [Електронний ресурс] – Режим доступу до ресурсу:
<https://drogon.org/>
7. JavaScript (MDN Web Docs) [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
8. HTML (MDN Web Docs) [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTML>
9. CSS (MDN Web Docs) [Електронний ресурс] – Режим доступу до ресурсу:
<https://developer.mozilla.org/en-US/docs/Web/CSS>
10. Web Crypto API (MDN Web Docs) [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/Web_Crypto_API
11. JSZip [Електронний ресурс] – Режим доступу до ресурсу:
<https://stuk.github.io/jszip/>
12. asmCrypto.js (AES module) [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/vibornoff/asmcrypto.js/>
13. aes-js [Електронний ресурс] – Режим доступу до ресурсу:
<https://github.com/ricmoo/aes-js>
14. PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.postgresql.org/>

		Кондратюк О.Д.			ІПЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

15. SQL Standard (ISO/IEC 9075) [Електронний ресурс] – Режим доступу до ресурсу: <https://www.iso.org/standard/76583.html>
16. Advanced Encryption Standard (AES) [Електронний ресурс] – Режим доступу до ресурсу: <https://doi.org/10.6028/NIST.FIPS.197-upd1>
17. PBKDF2 (RFC 8018) [Електронний ресурс] – Режим доступу до ресурсу: <https://datatracker.ietf.org/doc/html/rfc8018>
18. OpenSSL Project [Електронний ресурс] – Режим доступу до ресурсу: <https://www.openssl.org/>
19. Base64 Encoding Scheme (RFC 4648) [Електронний ресурс] – Режим доступу до ресурсу: <https://datatracker.ietf.org/doc/html/rfc4648>
20. SPKI (Simple Public Key Infrastructure) [Електронний ресурс] – Режим доступу до ресурсу: <https://datatracker.ietf.org/doc/html/rfc2693>
21. PKCS #8: Private-Key Information Syntax Standard [Електронний ресурс] – Режим доступу до ресурсу: <https://datatracker.ietf.org/doc/html/rfc5208>
22. AES ECB Mode (NIST SP 800-38A) [Електронний ресурс] – Режим доступу до ресурсу: <https://doi.org/10.6028/NIST.SP.800-38A>
23. PKCS #7: Cryptographic Message Syntax [Електронний ресурс] – Режим доступу до ресурсу: <https://datatracker.ietf.org/doc/html/rfc2315>
24. Web Storage API (MDN Web Docs) [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API
25. AES CBC Mode (NIST SP 800-38A) [Електронний ресурс] – Режим доступу до ресурсу: <https://doi.org/10.6028/NIST.SP.800-38A>
26. MEGA Security White Paper Third Edition [Електронний ресурс] – Режим доступу до ресурсу: <https://mega.nz/SecurityWhitepaper.pdf>

		Кондратюк. О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

ДОДАТКИ

		Кондратюк. О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

```

function calculatePasswordStrength(password) {
  let score = 0;
  let feedback = {
    messageKey: 'strength_weak',
    score: 0,
  };
  if (!password || password.length === 0) {
    feedback.score = 0;
    feedback.messageKey = 'strength_weak';
    return feedback;
  }
  if (password.length >= 8) score += 1;
  if (password.length >= 10) score += 1;
  if (password.length >= 12) score += 1;

  let types = 0;
  if (/[a-z]/.test(password)) types++;
  if (/[A-Z]/.test(password)) types++;
  if (/[0-9]/.test(password)) types++;
  if (/^[^a-zA-Z0-9]/.test(password)) types++;
  if (types === 2) score += 1;
  if (types === 3) score += 2;
  if (types === 4) score += 3;
  if (password.length > 0) {
    if (password.length > 6 && (/^[0-9]+$/.test(password) || /^[a-zA-Z]+$/.test(password))) {
      score -= 2;
    }
    for (let i = 0; i < password.length - 2; i++) {
      const char1 = password.charCodeAt(i);
      const char2 = password.charCodeAt(i + 1);
      const char3 = password.charCodeAt(i + 2);
      if ((char1 + 1 === char2 && char2 + 1 === char3) ||
        (char1 - 1 === char2 && char2 - 1 === char3)) {
        score -= 1;
        break;
      }
    }
    if (/(\.)\1\1/.test(password)) score -= 1;
  }

  score = Math.max(0, Math.min(score, 5));
  feedback.score = score;
  if (score <= 1) feedback.messageKey = 'strength_weak';
  else if (score <= 3) feedback.messageKey = 'strength_medium';
  else feedback.messageKey = 'strength_strong';
  return feedback;
}

```

```

async createSalt(clientRandomValue) {
    const domain = "lockbox.com";
    const targetLength = 200;

    const domainBytes = this.stringToUint8Array(domain)
    const paddingChar = 'P';
    const paddingLength = targetLength - domainBytes.length;
    const paddingBytes =
this.stringToUint8Array(paddingChar.repeat(paddingLength));

    const combined = new Uint8Array(targetLength + clientRandomValue.length);
    combined.set(domainBytes, 0);
    combined.set(paddingBytes, domainBytes.length);
    combined.set(clientRandomValue, domainBytes.length + paddingLength);

    return this.sha256(combined);
}

async pbkdf2HmacSha512(password, salt, iterations, keyByteLength) {
    const passwordKey = await crypto.subtle.importKey(
        'raw',
        this.stringToUint8Array(password),
        { name: 'PBKDF2' },
        false,
        ['deriveBits']
    );

    const derivedBits = await crypto.subtle.deriveBits(
        {
            name: 'PBKDF2',
            salt: salt,
            iterations: iterations,
            hash: 'SHA-512',
        },
        passwordKey,
        keyByteLength * 8
    );
    return new Uint8Array(derivedBits);
}

stringToUint8Array(str) {
    return new TextEncoder().encode(str);
}

async sha256(data) {
    const hashBuffer = await crypto.subtle.digest('SHA-256', data);
    return new Uint8Array(hashBuffer);
}

```

		Кондратюк О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

```

registerForm.addEventListener('submit', async (event) => {
    event.preventDefault();
    clearErrors();
    if (!validateInputs()) {
        return;
    }

    const registerButton = document.getElementById('registerButton');
    registerButton.disabled = true;
    registerButton.textContent = ltr('loading');

    const email = emailInput.value;
    const password = passwordInput.value;

    try {
        const clientRandomValueBytes = cryptoUtils.generateRandomBytes(16);
        const masterKeyBytes = cryptoUtils.generateRandomBytes(16);
        const saltBytes = await
cryptoUtils.createSalt(clientRandomValueBytes);
        const derivedKeyBytes = await
cryptoUtils.pbkdf2HmacSha512(password, saltBytes, 100000, 32);

        const derivedEncryptionKeyBytes = derivedKeyBytes.slice(0, 16);
        const derivedAuthenticationKeyBytes = derivedKeyBytes.slice(16,
32);

        const cipherObject = new
aesjs.ModeOfOperation.ecb(derivedEncryptionKeyBytes);
        const encryptedMasterKeyBytes =
cipherObject.encrypt(masterKeyBytes);

        const hashedAuthKeyFull = await
cryptoUtils.sha256(derivedAuthenticationKeyBytes);
        const hashedAuthenticationKeyBytes = hashedAuthKeyFull.slice(0,
16);

        const payload = {
            email: email,
            clientRandomValue:
cryptoUtils.uint8ArrayToBase64(clientRandomValueBytes),
            encryptedMasterKey:
cryptoUtils.uint8ArrayToBase64(encryptedMasterKeyBytes),
            hashedAuthenticationKey:
cryptoUtils.uint8ArrayToBase64(hashedAuthenticationKeyBytes),
        };
        const response = await fetch('/api/register', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(payload),

```

		Кондратюк. О.Д.			ІПЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

```

    });
    const responseText = await response.text();
    let responseData;
    try {
        responseData = JSON.parse(responseText);
    } catch (e) {
        responseData = { message: responseText ||
ltr('error_registration_failed') };
    }

    if (response.ok) {
        registerForm.style.display = 'none';
        if (loginLinkParagraph) loginLinkParagraph.style.display =
'none';

        if (confirmationMessageDiv) {
            const pElement =
confirmationMessageDiv.querySelector('p[data-ltr-
key="confirmation_email_sent_message"]');
            if (pElement) {
                pElement.textContent =
ltr('confirmation_email_sent_message', email);
            } else {
                confirmationMessageDiv.textContent =
ltr('confirmation_email_sent_message', email);
            }
            confirmationMessageDiv.style.display = 'block';
        }

    } else {
        const errorMessageKey = `error_api_${response.status}`;
        let displayMessage = responseData.message ||
ltr(errorMessageKey, ltr('error_registration_failed'));

        if (displayMessage === errorMessageKey && responseData.message)
{
            displayMessage = responseData.message;
        } else if (displayMessage === errorMessageKey) {
            displayMessage = ltr('error_registration_failed');
        }
        showGlobalError(displayMessage);
    }

    } catch (error) {
        console.error("Registration submission error:", error);
        showGlobalError(ltr('error_registration_failed') + (error.message ?
` : ${error.message}` : ''));
    } finally {
        registerButton.disabled = false;
        registerButton.textContent = ltr('button_register');
    }
});

```

		Кондратюк.О.Д			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

```

loginForm.addEventListener('submit', async (event) => {
  event.preventDefault();
  clearErrors();
  if (!validateInputs()) return;
  setLoadingState(true);
  const email = emailInput.value;
  const password = passwordInput.value;
  try {
    const step1Response = await fetch('/api/login1', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ email: email }),
    });
    if (!step1Response.ok) {
      const errorText = await step1Response.text();
      showGlobalError(ltr('error_login1_failed') + (errorText ? `: ${errorText}` : ''));
      setLoadingState(false);
      return;
    }
    const saltBase64 = await step1Response.text();
    const saltBytes = cryptoUtils.base64ToUint8Array(saltBase64);
    const derivedKeyBytes = await cryptoUtils.pbkdf2HmacSha512(password, saltBytes, 100000, 32);
    const derivedEncryptionKeyBytes = derivedKeyBytes.slice(0, 16);
    const derivedAuthenticationKeyBytes = derivedKeyBytes.slice(16, 32);
    const step2Payload = {
      email: email,
      derivedAuthenticationKey:
cryptoUtils.uint8ArrayToBase64(derivedAuthenticationKeyBytes),
    };
    const step2Response = await fetch('/api/login2', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(step2Payload),
    });
    if (!step2Response.ok) {
      const errorText = await step2Response.text();
      let parsedError;
      try { parsedError = JSON.parse(errorText); } catch (e) { }
      showGlobalError(ltr('error_login2_failed') + (parsedError?.message || errorText ? `: ${parsedError?.message || errorText}` : ''));
      setLoadingState(false);
      return;
    }
    const loginData = await step2Response.json();
    const encryptedMasterKeyBytes = cryptoUtils.base64ToUint8Array(loginData.cmk);
    let masterKeyBytes;
    try {

```

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

        const cipherObject = new
aesjs.ModeOfOperation.ecb(derivedEncryptionKeyBytes);
        masterKeyBytes = cipherObject.decrypt(encryptedMasterKeyBytes);
    } catch (e) {
        showGlobalError(ltr('error_master_key_decryption_failed'));
        setLoadingState(false);
        return;
    }
    localStorage.setItem('mk', cryptoUtils.uint8ArrayToBase64(masterKeyBytes));
    if (loginData.tsid) {
        setLoadingState(true, ltr('info_setting_up_keys'));
        try {
            const tsid = loginData.tsid;
            const rsaKeyPair = await cryptoUtils.generateRsaKeyPair('SHA-256');
            const paddedPrivateKeyBytes =
cryptoUtils.pkcs7Pad(rsaKeyPair.privateKeyPkcs8Bytes, 16);
            const encryptedRsaPrivateKeyBytes = await
cryptoUtils.aesEcbEncrypt(masterKeyBytes, paddedPrivateKeyBytes);
            const encryptedRsaPrivateKeyBase64 =
cryptoUtils.uint8ArrayToBase64(encryptedRsaPrivateKeyBytes);
            const registerRsaPayload = {
                temporarySessionId: tsid,
                rsaPublicKey: rsaKeyPair.publicKeySpkiBase64,
                encryptedRsaPrivateKey: encryptedRsaPrivateKeyBase64,
            };
            const rsaResponse = await fetch('/api/register_rsa', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(registerRsaPayload),
            });
            if (!rsaResponse.ok) {
                const errorText = await rsaResponse.text();
                let parsedError;
                try { parsedError = JSON.parse(errorText); } catch (e) { }
                throw new Error(ltr('error_rsa_registration_failed') +
(parsedError?.message || errorText ? `: ${parsedError?.message || errorText}` :
''));
            }
            const rsaData = await rsaResponse.json();
            localStorage.setItem('pk',
cryptoUtils.uint8ArrayToBase64(rsaKeyPair.privateKeyPkcs8Bytes));
            const encryptedSessionIdBytes =
cryptoUtils.base64ToUint8Array(rsaData.csid);
            const sessionIdBytes = await
cryptoUtils.rsaDecrypt(rsaKeyPair.privateKey, encryptedSessionIdBytes);
            const sessionId = cryptoUtils.uint8ArrayToString(sessionIdBytes);
            localStorage.setItem('sid', sessionId);
            localStorage.removeItem('tsid');
            window.location.href = 'fm.html';
        } catch (error) {
            console.error('RSA registration process failed:', error);

```

		Кондратюк.О.Д			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

```

        showGlobalError(error.message || ltr('error_rsa_registration_failed'));
        setLoadingState(false);
    }
} else if (loginData.csid && loginData.cpk) {
    const encryptedCpkBytes = cryptoUtils.base64ToUint8Array(loginData.cpk);
    let decryptedPaddedCpkBytes;
    try {
        decryptedPaddedCpkBytes = await
cryptoUtils.aesEcbDecrypt(masterKeyBytes, encryptedCpkBytes);
    } catch (e) {
        console.error("RSA Private Key decryption failed:", e);
        showGlobalError(ltr('error_private_key_decryption_failed'));
        setLoadingState(false);
        return;
    }
    let rsaPrivateKeyBytes;
    try {
        rsaPrivateKeyBytes = cryptoUtils.pkcs7Unpad(decryptedPaddedCpkBytes);
    } catch (e) {
        console.error("CPK decryption failed (Unpadding part):", e);
        showGlobalError(ltr('error_private_key_decryption_failed') + " Invalid
key padding.");
        setLoadingState(false);
        return;
    }
    localStorage.setItem('pk',
cryptoUtils.uint8ArrayToBase64(rsaPrivateKeyBytes));
    let rsaPrivateKeyForDecryption;
    try {
        rsaPrivateKeyForDecryption = await
cryptoUtils.importRsaPrivateKey(rsaPrivateKeyBytes);
    } catch (e) {
        console.error("RSA Private Key import for CSID decryption failed:", e);
        showGlobalError(ltr('error_crypto_operation') + " Key import failed.");
        setLoadingState(false);
        return;
    }
    let sessionIdBytes;
    try {
        const encryptedSessionIdBytes =
cryptoUtils.base64ToUint8Array(loginData.csid);
        sessionIdBytes = await
cryptoUtils.rsaDecrypt(rsaPrivateKeyForDecryption, encryptedSessionIdBytes);
    } catch (e) {
        console.error("Session ID decryption failed:", e);
        showGlobalError(ltr('error_session_id_decryption_failed'));
        setLoadingState(false);
        return;
    }
    const sessionId = cryptoUtils.uint8ArrayToString(sessionIdBytes);
    localStorage.setItem('sid', sessionId);

```

		Кондратюк.О.Д			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ док.ум.	Підпис	Дата		74

```

        window.location.href = 'fm.html';
    } else {
        showGlobalError(ltr('error_login_generic') + ': Invalid response from
server.');
```

```

        setLoadingState(false);
    }
} catch (error) {
    console.error('Login process error:', error);
    showGlobalError(ltr('error_login_generic') + (error.message ? `:
${error.message}` : ''));
    setLoadingState(false);
}

```

Додаток Д

```

async encryptFileName(fileName, fileKeyWords) {
    const keyBytes = this.wordsToBytes(new Uint32Array(fileKeyWords.slice(0,
4)));
    const ivPart1 = this.wordsToBytes(new Uint32Array(fileKeyWords.slice(4,
6)));
    const ivBytes = new Uint8Array(16);
    ivBytes.set(ivPart1, 0);
    ivBytes.set(ivPart1, 8);

    const fileNameBytes = this.stringToUint8Array(fileName);
    const paddedFileNameBytes = this.pkcs7Pad(fileNameBytes, 16);

    const encryptedBytes = await this.aesCbcEncrypt(keyBytes, ivBytes,
paddedFileNameBytes);
    return this.uint8ArrayToBase64(encryptedBytes);
}

wordsToBytes(words) {
    const bytes = new Uint8Array(words.length * 4);
    const view = new DataView(bytes.buffer);
    for (let i = 0; i < words.length; i++) {
        view.setUint32(i * 4, words[i], false);
    }
    return bytes;
}

pkcs7Pad(data, blockSize) {
    const paddingLength = blockSize - (data.length % blockSize);
    const padding = new Uint8Array(paddingLength).fill(paddingLength);
    const paddedData = new Uint8Array(data.length + paddingLength);
    paddedData.set(data, 0);
    paddedData.set(padding, data.length);
    return paddedData;
}

```

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

importScripts('aesasm.js');

postMessage = self.webkitPostMessage || self.postMessage;

var heap = new Uint8Array(0x200000),
    asm = aes_asm(self, null, heap.buffer),
    nonce = new Uint8Array(8),
    iv = new Uint8Array(16),
    ctr = 0;

onmessage = function(e)
{
    if (typeof(e.data) == 'string')
    {
        var arr = JSON.parse(e.data);

        var nonceView = new DataView(nonce.buffer);
        nonceView.setUint32( 0, arr[4], false );
        nonceView.setUint32( 4, arr[5], false );
        iv.set( nonce, 0 );
        iv.set( nonce, 8 );

        var key = new Uint8Array(16);
        var keyView = new DataView(key.buffer);
        keyView.setUint32( 0, arr[0], false );
        keyView.setUint32( 4, arr[1], false );
        keyView.setUint32( 8, arr[2], false );
        keyView.setUint32( 12, arr[3], false );

        asm.init_key_128.apply( asm, key );
    }
    else if (typeof(e.data) == 'number')
    {
        ctr = e.data;
    }
    else
    {
        var data = new Uint8Array( e.data.buffer || e.data );
        var heapView = new DataView( heap.buffer );
        var macs = [];

        for ( var i = 0; i < data.length; i += 0x100000 )
        {
            var j = ( i + 0x100000 < data.length ) ? i + 0x100000 : data.length;
            heap.set( data.subarray(i,j), 0x1000 );
            asm.init_state.apply( asm, iv );
            asm.ccm_encrypt(0x1000, j-i,nonce[0], nonce[1], nonce[2], nonce[3],
            nonce[4], nonce[5], nonce[6], nonce[7],0, 0, 0, 0, 0, 0,(ctr/0x100000000) >>> 0,ctr
            >>> 0);
        }
    }
}

```

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

        data.set( heap.subarray( 0x1000, 0x1000+j-i ), i );
        asm.save_state(0x1000);
        macs.push( heapView.getUint32( 0x1000, false ) );
        macs.push( heapView.getUint32( 0x1004, false ) );
        macs.push( heapView.getUint32( 0x1008, false ) );
        macs.push( heapView.getUint32( 0x100c, false ) );
        ctr += Math.ceil( (j-i)/16 );
    }

    postMessage(JSON.stringify(macs));
    postMessage(data.buffer);
}
};

```

Додаток Ж

```

condenseMACs(macsArray, keyWords) {
    try {
        const keyBytes = this.wordsToBytes(keyWords);
        const aesEcb = new aesjs.ModeOfOperation.ecb(keyBytes);
        let currentMac = new Uint32Array(4);
        for (let i = 0; i < macsArray.length; i += 4) {
            const mBlk = new Uint32Array([
                macsArray[i], macsArray[i + 1], macsArray[i + 2], macsArray[i + 3]
            ]);
            currentMac[0] ^= mBlk[0];
            currentMac[1] ^= mBlk[1];
            currentMac[2] ^= mBlk[2];
            currentMac[3] ^= mBlk[3];
            const encryptedMacBytes =
aesEcb.encrypt(this.wordsToBytes(currentMac));
            currentMac = this.bytesToWords(encryptedMacBytes);
        }
        return currentMac;
    }
    catch (error) {
        console.error("Condense MACs Error:", error);
        throw error;
    }
}

obfuscateFileKey(fileKeyWords, condensedMacWords) {
    const obfuscated = new Uint32Array(8);
    obfuscated[0] = fileKeyWords[0] ^ fileKeyWords[4];
    obfuscated[1] = fileKeyWords[1] ^ fileKeyWords[5];
    obfuscated[2] = fileKeyWords[2] ^ condensedMacWords[0] ^
condensedMacWords[1];
    obfuscated[3] = fileKeyWords[3] ^ condensedMacWords[2] ^
condensedMacWords[3];
    obfuscated[4] = fileKeyWords[4];

```

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

obfuscated[5] = fileKeyWords[5];
obfuscated[6] = condensedMacWords[0] ^ condensedMacWords[1];
obfuscated[7] = condensedMacWords[2] ^ condensedMacWords[3];
return obfuscated;
}

```

Додаток К

```

getFileKeyFromObfuscatedFileKey(obfuscatedFileKeyWords) {
    if (obfuscatedFileKeyWords.length !== 8) {
        throw new Error("Obfuscated File Key must be 8 words (Uint32Array).");
    }
    const fileKey = new Uint32Array(6);
    fileKey[0] = obfuscatedFileKeyWords[0] ^ obfuscatedFileKeyWords[4];
    fileKey[1] = obfuscatedFileKeyWords[1] ^ obfuscatedFileKeyWords[5];
    fileKey[2] = obfuscatedFileKeyWords[2] ^ obfuscatedFileKeyWords[6];
    fileKey[3] = obfuscatedFileKeyWords[3] ^ obfuscatedFileKeyWords[7];
    fileKey[4] = obfuscatedFileKeyWords[4];
    fileKey[5] = obfuscatedFileKeyWords[5];
    return fileKey;
}

getCondensedMacWordsFromObfuscatedFileKey(obfuscatedFileKeyWords) {
    if (obfuscatedFileKeyWords.length !== 8) {
        throw new Error("Obfuscated File Key must be 8 words (Uint32Array).");
    }
    const macWords = new Uint32Array(2);
    macWords[0] = obfuscatedFileKeyWords[6];
    macWords[1] = obfuscatedFileKeyWords[7];
    return macWords;
}

```

Додаток Л

```

importScripts('aesasm.js');
postMessage = self.webkitPostMessage || self.postMessage;
var heap = new Uint8Array(0x200000),
    asm = aes_asm(self, null, heap.buffer),
    nonce = new Uint8Array(8),
    iv = new Uint8Array(16),
    ctr = 0;

onmessage = function(e)
{
    if (typeof(e.data) == 'string')
    {
        var arr = JSON.parse(e.data);
        var nonceView = new DataView(nonce.buffer);
        nonceView.setUint32( 0, arr[4], false );
        nonceView.setUint32( 4, arr[5], false );
    }
}

```

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        iv.set( nonce, 0 );
        iv.set( nonce, 8 );
        var key = new Uint8Array(16);
        var keyView = new DataView(key.buffer);
        keyView.setUint32( 0, arr[0], false );
        keyView.setUint32( 4, arr[1], false );
        keyView.setUint32( 8, arr[2], false );
        keyView.setUint32( 12, arr[3], false );
        asm.init_key_128.apply( asm, key );
    }
    else if (typeof(e.data) == 'number') ctr = e.data;
    else
    {
        var data = new Uint8Array( e.data.buffer || e.data );
        var heapView = new DataView( heap.buffer );
        var macs = [];
        for ( var i = 0; i < data.length; i += 0x100000 )
        {
            var j = ( i + 0x100000 < data.length ) ? i + 0x100000 : data.length;
            heap.set( data.subarray(i,j), 0x1000 );
            asm.init_state.apply( asm, iv );
            asm.ccm_decrypt(0x1000, j-i,nonce[0], nonce[1], nonce[2], nonce[3],
nonce[4], nonce[5], nonce[6], nonce[7],0, 0, 0, 0, 0, 0,(ctr/0x100000000) >>> 0,ctr
>>> 0);
            data.set( heap.subarray( 0x1000, 0x1000+j-i ), i );
            asm.save_state(0x1000);
            macs.push( heapView.getUint32( 0x1000, false ) );
            macs.push( heapView.getUint32( 0x1004, false ) );
            macs.push( heapView.getUint32( 0x1008, false ) );
            macs.push( heapView.getUint32( 0x100c, false ) );
            ctr += Math.ceil( (j-i)/16 );
        }

        postMessage(JSON.stringify(macs));
        postMessage(data.buffer);
    }
};

```

Додаток М

```

async function createFolderOnServer(folderName, parentApiHandle) {

    const newFolderKeyMaterialBytes = cryptoUtils.generateRandomBytes(24);
    const newFolderKeyWords =
cryptoUtils.bytesToWords(newFolderKeyMaterialBytes);

    const nameEncryptionKeyBytes = cryptoUtils.wordsToBytes(new
Uint32Array(newFolderKeyWords.slice(0, 4)));

```

		Кондратюк.О.Д			ІПЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

        const nameNoncePartBytes = cryptoUtils.wordsToBytes(new
Uint32Array(newFolderKeyWords.slice(4, 6)));
        const nameIvBytes = new Uint8Array(16);
        nameIvBytes.set(nameNoncePartBytes, 0); nameIvBytes.set(nameNoncePartBytes,
8);

        const folderNameBytes = cryptoUtils.stringToUint8Array(folderName);
        const paddedFolderNameBytes = cryptoUtils.pkcs7Pad(folderNameBytes, 16);
        const encryptedFolderNameBytes = await
cryptoUtils.aesCbcEncrypt(nameEncryptionKeyBytes, nameIvBytes,
paddedFolderNameBytes);
        const encryptedFolderNameB64 =
cryptoUtils.uint8ArrayToBase64(encryptedFolderNameBytes);

        const paddedFolderKeyMaterialBytes =
cryptoUtils.pkcs7Pad(newFolderKeyMaterialBytes, 16);
        const encryptedFolderKeyWithMasterKeyBytes = await
cryptoUtils.aesEcbEncrypt(userMasterKey, paddedFolderKeyMaterialBytes);
        const encryptedFolderKeyB64 =
cryptoUtils.uint8ArrayToBase64(encryptedFolderKeyWithMasterKeyBytes);

        const payload = {
            parent_folder_handle: parentApiHandle,
            encrypted_name: encryptedFolderNameB64,
            encrypted_folder_key: encryptedFolderKeyB64
        };

        const response = await authenticatedFetch('/api/folders', {
            method: 'POST',
            body: JSON.stringify(payload)
        });

        if (!response.ok) {
            const errorData = await response.json().catch(() => ({ message:
response.statusText }));
            throw new Error(errorData.message || `HTTP ${response.status}`);
        }
        const newItemDataFromApi = await response.json();

        if (newItemDataFromApi && newItemDataFromApi.h) {
            newItemDataFromApi.n = encryptedFolderNameB64;
            newItemDataFromApi.k = encryptedFolderKeyB64;
            newItemDataFromApi.f = 1;
            newItemDataFromApi.c = Math.floor(Date.now() / 1000);
            newItemDataFromApi.u = Math.floor(Date.now() / 1000);
            await addOrUpdateItemInView(newItemDataFromApi, 'add',
parentApiHandle);
        }

        return newItemDataFromApi;
    }
}

```

		Кондратюк.О.Д.			ІПЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

textKey: 'fm_button_share', class: 'primary', onClick: async () => {
  const shareButton = document.getElementById('share-folder-button');
  if (shareButton) shareButton.disabled = true;
  document.getElementById('sharing-progress-indicator').style.display = 'block';
  updateSharingProgressText(ltr('fm_sharing_folder_generating_keys'));
  try {
    const rawFolderShareKeyBytes = cryptoUtils.generateRandomBytes(16);
    const encryptedFolderShareSecretKeyForStorageBytes = await
cryptoUtils.aesEcbEncrypt(userMasterKey, rawFolderShareKeyBytes);
    const encryptedFolderShareSecretKeyForStorageB64 =
cryptoUtils.uint8ArrayToBase64(encryptedFolderShareSecretKeyForStorageBytes);
    const folderItemKeyMaterialBytes = await
getItemKeyMaterial(folderData.type, folderData.encrypted_key_b64, userMasterKey);
    const paddedFolderItemKeyMaterial =
cryptoUtils.pkcs7Pad(folderItemKeyMaterialBytes, 16);
    const encryptedFolderKeyForSharingBytes = await
cryptoUtils.aesEcbEncrypt(rawFolderShareKeyBytes, paddedFolderItemKeyMaterial);
    const encryptedFolderKeyForSharingB64 =
cryptoUtils.uint8ArrayToBase64(encryptedFolderKeyForSharingBytes);
    updateSharingProgressText(ltr('fm_sharing_folder_contacting_server'));
    const shareApiEndpoint = `/api/folders/${folderData.handle}/share`;
    const response = await authenticatedFetch(shareApiEndpoint, {
      method: 'POST',
      body: JSON.stringify({
        encrypted_share_key_b64:
encryptedFolderShareSecretKeyForStorageB64,
        encrypted_original_key_for_sharing_b64:
encryptedFolderKeyForSharingB64
      })
    });
    if (!response.ok) {
      const errorData = await response.json().catch(() => ({ message:
response.statusText }));
      throw new Error(errorData.message || `Share API call failed:
${response.status}`);
    }
    const shareResponseData = await response.json();
    if (!shareResponseData.public_handle) {
      throw new Error("Backend did not return a public_handle for the shared
folder.");
    }
    folderData.publicHandle = shareResponseData.public_handle;
    folderData.encrypted_share_key_b64 =
encryptedFolderShareSecretKeyForStorageB64;
    const itemElement = itemListContainer.querySelector(`.item-entry[data-
handle="${folderData.handle}"]`);
    if (itemElement) {
      itemElement.dataset.publicHandle = folderData.publicHandle;
      const nameElement = itemElement.querySelector('.item-name');

```

		Кондратюк.О.Д.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Локтікова Т.М.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (nameElement && !nameElement.querySelector('.shared-indicator')) {
            const indicator = document.createElement('span');
            indicator.className = 'shared-indicator';
            indicator.title = ltr('fm_item_shared_indicator_tooltip');
            nameElement.appendChild(indicator);
        }
    }
    sortAndReRenderItems();
    showToast(ltr('fm_folder_shared_successfully_toast',
folderData.decrypted_name), 'success');

```

		Кондратюк. О.Д			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		82