НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Кросплатформний файлообмінник для локальних середовищ

Виконав студент IV курсу, групи ІТ-91
(шифр групи)
Плескач Михайло Тарасович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. викл., Марченко О. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації ст. викл., Головченко М. М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент ст. викл., Шимкович В. М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення комп'ютерних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Плескачу Михайлу Тарасовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Кросплатформний файлообмінник для локальних середовищ
керівник проєкту Марченко Олена Іванівна, ст. викл.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 31 »травня 2023 р. №2101-с

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: змістовний опис і аналіз предметної області, аналіз існуючих технологій та успішних IT-проєктів, аналіз вимог до програмного забезпечення, постановка задачі.
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, конструювання програмного забезпечення, аналіз безпеки даних.
3) Аналіз якості та тестування програмного забезпечення: аналіз якості програмного забезпечення, опис процесів тестування, опис контрольного прикладу.
4) Впровадження та супровід програмного забезпечення: розгортання програмного забезпечення, підтримка програмного забезпеченн.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема структурна класів програмного забезпечення

3) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2023 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв’язання задачі	17.03.2023	
3	Постановка та формалізація задачі	24.03.2023	
4	Розробка інформаційного забезпечення	12.04.2023	
5	Алгоритмізація задачі	19.04.2023	
6	Обґрунтування вибору використаних технічних засобів	21.04.2023	
7	Розробка програмного забезпечення	12.05.2023	
8	Налагодження програми	14.05.2023	
9	Виконання графічних документів	16.05.2023	
10	Оформлення пояснювальної записки	17.05.2023	
11	Подання ДП на попередній захист	02.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент _____ Михайло ПЛЕСКАЧ
(підпис) (ініціали, прізвище)

Керівник _____ Олена МАРЧЕНКО
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 49 таблиць, 32 рисунків та 34 джерела – загалом 64 сторінки.

Дипломний проєкт присвячений створенню кросплатформного файлообміннику для локальних середовищ.

Мета: розробка кросплатформного додатку для обміну файлів для локальних середовищ.

Об'єкт дослідження: файли даних.

Предмет дослідження: додаток для обміну файлами.

У першому розділі виконано аналіз вимог до програмного забезпечення. Досліджено алгоритмічні і технічні рішення, засоби розробки і відомі програмні продукти з області дослідження. В результаті створено функціональні і нефункціональні вимоги до програмного забезпечення, поставлено цілі і задачі розробки.

Другий розділ присвячений моделюванню та конструюванню програмного забезпечення. Описано бізнес-процеси. Проведено визначення архітектури і конструювання програмного забезпечення, а також аналіз безпеки даних.

У третьому розділі проведено аналіз якості та тестування програмного забезпечення. Описано методи і процеси тестування. Наведено детальний розбір тесту у вигляді контрольного прикладу.

У четвертому розділі виконано впровадження та супровід програмного забезпечення. Задокументовано процес розгортання програмного забезпечення. Визначено процес підтримки програмного забезпечення.

КЛЮЧОВІ СЛОВА: ФАЙЛООБМІННИК, ЛОКАЛЬНЕ СЕРЕДОВИЩЕ, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, ВЕБ-ЗАСТОСУНОК, ФАЙЛ, ПРОФІЛЬ, ДОСТУП.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 49 tables, 32 figures and 34 sources – in total 64 pages.

The purpose of the diploma project is to create cross-platform file hosting for local environments.

Goal: development of cross-platform file exchange application for local environments.

Research object: data files.

Research subject: file exchange application.

In the first section, an analysis of software requirements is performed. Algorithmic and technical solutions, development tools and well-known software products from the research area were studied. As a result, functional and non-functional requirements for the software were created, development goals and objectives were set.

The second chapter is devoted to software modeling and design. Business processes are described. The definition of software architecture and design, as well as data security analysis, was carried out.

In the third chapter, the quality analysis and testing of the software is carried out. Test methods and processes are described. A detailed analysis of the test is given in the form of a control example.

In the fourth chapter, the implementation and maintenance of the software is performed. The software deployment process is documented. Software support process is defined.

KEYWORDS: FILE HOSTING, LOCAL ENVIRONMENT, CLIENT-SERVER ARCHITECTURE, WEB APPLICATION, FILE, PROFILE, ACCESS.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний	2	
			проєкт		
2	А4	КПІ.ІТ-9121.045440.01.91	Технічне завдання	14	
3	А4	КПІ.ІТ-9121.045440.02.81	Пояснювальна записка	57	
4	А4	КПІ.ІТ-9121.045440.03.12	Текст програми	34	
5	А4	КПІ.ІТ-9121.045440.04.51	Програма та методика	6	
			тестування		
6	А4	КПІ.ІТ-9121.045440.05.34	Керівництво користувача	11	
7	А3	КПІ.ІТ-9121.045440.06.99.СС	Схема структурна варіантів	1	
			використань		
8	А3	КПІ.ІТ-9121.045440.06.99.СС	Схема структурна класів	1	
			програмного забезпечення		
9	А3	КПІ.ІТ-9121.045440.06.99.КЕ	Креслення вигляду екранних	1	
			форм		

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ-9121.045440.00.90		
Розроб.		Плескач М.Т.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Марченко О.І.					
Н.контр.		Головченко М.М.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-91	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

КРОСПЛАТФОРМНИЙ ФАЙЛООБМІННИК ДЛЯ ЛОКАЛЬНИХ СЕРЕДОВИЩ

Технічне завдання

КПІ.ІТ-9121.045440.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена МАРЧЕНКО

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Михайло ПЛЕСКАЧ

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: КРОСПЛАТФОРМНИЙ ФАЙЛООБМІННИК ДЛЯ ЛОКАЛЬНИХ СЕРЕДОВИЩ

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення «Кросплатформний файлообмінник для локальних середовищ» КПІ.ІТ-9121.045440.01.91, котра використовується для обміну файлами між файловими системами користувачів та призначена для вирішення таких задач, як обмін файлами між комп’ютерами ІТ-фірми, віддалене збереження і поширення бухгалтерських документів, створення спільного файлового простору для команди дизайнерів.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки «Кросплатформного файлообмінника для локальних середовищ» є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для обміну файлами між файловими системами користувачів.

Метою розробки є надання користувачам можливості зручного і надійного обміну файлами між своїми локальними середовищами без використання хмарних сховищ.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Функції користувацького інтерфейсу

– сторінка з формою для авторизації;

– сторінка зі списком поширених файлів для користувача;

– сторінка зі списком поширених файлів для адміністратора системи;

– сторінка для керування профілями користувачів для адміністратора системи;

– сторінка для керування доступом користувача до файлів для адміністратора системи.

Прототипи сторінок користувацького інтерфейсу зображено на рисунках 4.1 – 4.5.

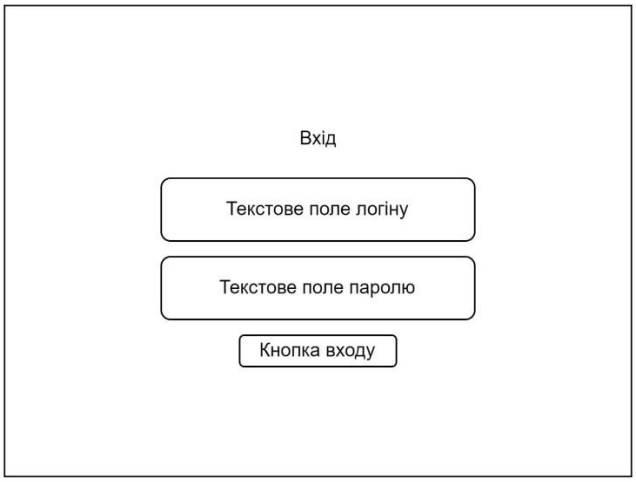


Рисунок 4.1 – Прототип сторінки авторизації

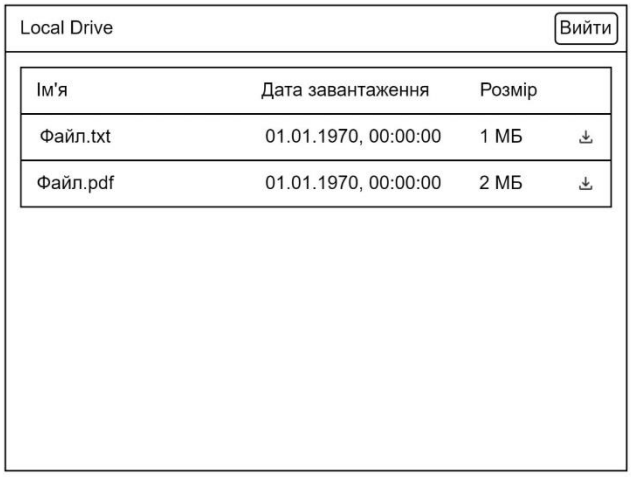


Рисунок 4.2 – Прототип сторінки доступних файлів

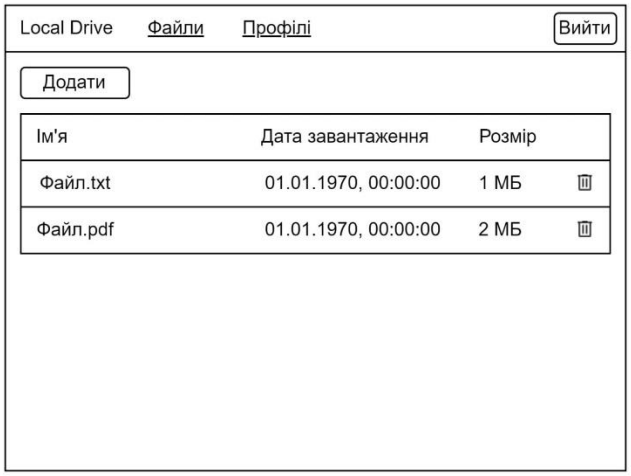


Рисунок 4.3 – Прототип сторінки файлів

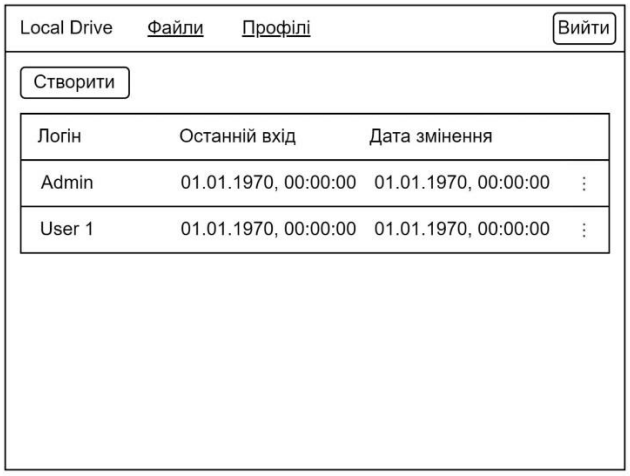


Рисунок 4.4 – Прототип сторінки профілів

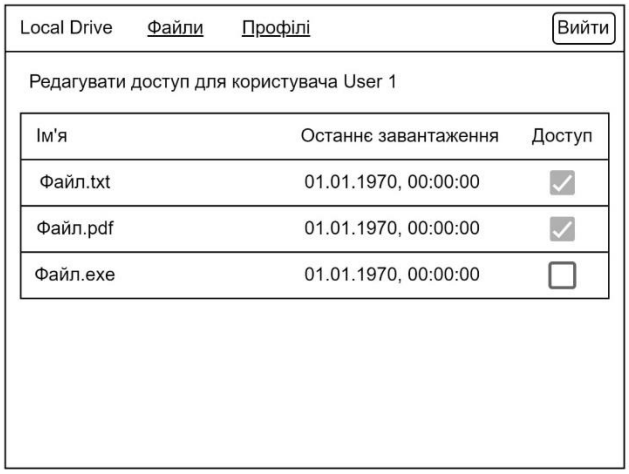


Рисунок 4.5 – Прототип сторінки доступу користувача до файлів

4.1.2 Для користувача:

– авторизація на віддаленому сервері з обліковими даними, наданими адміністратором системи;

– перегляд списку поширених йому файлів та їх характеристик;

– завантаження поширених йому файлів.

4.1.3 Для адміністратора системи:

– створення та редагування облікових записів користувачів;

– завантаження файлів у систему;

– видалення файлів з системи;

– надання або обмеження доступу до завантажених файлів певному користувачу.

4.1.4 Додаткові вимоги:

– хешування паролів;

– збереження дати і часу останнього завантаження файлів і зміни облікових даних користувача;

– відображення користувачу повідомлення при введенні некоректних даних.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити перевірку отриманих запитів і відхиляти їх у випадку некоректних даних. Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

Вимоги до умов експлуатації не висуваються.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i3;
– об’єм ОЗП: 1 ГБ;
– об’єм запам’ятовувального пристрою від 256 ГБ;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт.

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об’єм ОЗП: 8 ГБ;
– об’єм запам’ятовувального пристрою від 512 ГБ;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейств Windows NT, Linux Debian або MacOS. Обов’язковим є підтримка системою платформи ASP.NET Core 7.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: ввід користувачем інформації у відповідні текстові поля графічного користувацького інтерфейсу через веб-браузер.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: елементи графічного користувацького інтерфейсу і текстова репрезентація сутностей у вікні веб-браузера, бінарний файловий потік.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування C# і JavaScript.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі ASP.NET Core 7 і React.js 18.2.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді документу з текстом програми КПІ.ІТ-9121.045440.03.12 з розширенням .docx. Файли сирцевого коду мають розширення .cs, .js, jsx, .json.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Згенерувати випуски програмного забезпечення для операційних систем сімейств Windows NT, Linux Debian і MacOS.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– схема структурна класів програмного забезпечення;
– креслення вигляду екранних форм.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	10.03	
2.	Розробка технічного завдання	01.04	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	12.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	22.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	12.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	15.05	Тести, результати тестування
7.	Розробка матеріалів графічної частини проекту	16.05	Графічний матеріал проекту
8.	Розробка матеріалів текстової частини проекту	17.05	Пояснювальна записка
9.	Оформлення технічної документації проекту	18.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка

до дипломного проєкту

на тему: Кросплатформний файлообмінник для локальних середовищ

КПІ.ІТ-9121.045440.02.81

Київ-2023р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application programming interface, прикладний програмний Інтерфейс
HTTPS	–	HyperText Transfer Protocol Secure
FTP	–	File Transfer Protocol
IT	–	Інформаційні технології
ER	–	Entity-Relation diagram
ОС	–	Операційна система.
БД	–	База даних.
СУБД	–	Система управління базами даних.
ORM	–	Object-relational mapping

ВСТУП

З розвитком науки і технології електронно-обчислювальні машини набули масового поширення. Наразі багато людей мають персональні комп’ютери, майже у кожного є особистий мобільний телефон. Високий рівень комунікації був би неможливий без Інтернету, який складається з багатьох поєднаних між собою серверів. Таке повсякденне використання комп'ютерних і мобільних пристроїв потребує вирішення різноманітних проблем зі збереження і обміну даних: від завантаження в Інтернет щойно зробленої на камеру смартфону фотографії до резервного копіювання бази даних банку. Хмарні технології – зручний інструмент для розв’язання цих задач, популярність якого стрімко зростає з кожним днем. Сховища такого типу дозволяють надійно зберігати інформацію на захищених серверах, доступ до неї можливий з будь-якого іншого пристрою, а також за потреби її можна поширювати іншим користувачам[1]. Однак у них є і недоліки – диски, Інтернет-з’єднання і обслуговування серверів мають велику вартість для компанії, яка надає послуги з хмарного збереження даних, тому зазвичай є ліміт з сумарного розміру файлів, які можна туди завантажити, а додаткове місце може коштувати велику суму для користувача[2]. З альтернативних технологій – поширення файлів інструментами операційних систем або віддалене підключення. Але у них також є недоліки. Поширення файлів не має можливості працювати з багатьма платформами, а віддалене підключення надає користувачу повний контроль над цільовим середовищем, тож не є захищеним. Тому розробка додатку для обміну файлів, що вирішує перелічені проблеми, є важливою і актуальною. Необхідними для нього є наступні критерії: багатоплатформність, захищеність, можливість встановити на локальне середовище.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Хмарні сховища – це різновид технологій хмарних обчислень, що відповідають за збереження і поширення файлів. В їх основі – файловий сервер, доступ до якого зазвичай можна отримати з будь-якого іншого пристрою через графічний користувацький інтерфейс, інколи – через консольний. Обов'язковим є мережеве з'єднання з цільовим сховищем, оскільки обмін даними відбувається якраз мережевими протоколами. Власники таких серверів надають доступ до них зовнішнім користувачам за певним тарифом.

Файлообмінники – це сервіси для поширення файлів мережею Інтернет. Їх ціль – надавати користувачам доступ до необхідних їм даних, що зберігаються на віддаленому сервері. Поширювані файли публікуються адміністраторами цих сервісів. Файлообмінники – це функціонально обмежені хмарні сховища, але їх ціль полягає у вирішенні задач дещо іншого напрямку.

Перед розробниками подібного програмного забезпечення стоять типові задачі з впровадження якісного обміну даними. Перш за все, власне обмін даними, для якого вимагаються знання файлових систем і розуміння роботи мережевих протоколів. Дуже важливим є дотримання високого рівня безпеки даних для уникнення несанкціонованого доступу до файлів, для чого розробник повинен мати розширені знання з кібербезпеки. Також стоїть задача зі створення зручного користувацького інтерфейсу, а оскільки хмарні сховища і файлообмінники є найпоширенішими видами програмного забезпечення у досліджуваній області, то вітається вміння створювати для них сучасний дизайн.

1.2 Змістовний опис і аналіз предметної області

Прикладів застосування хмарних сховищ можна знайти безліч. Будь-який веб-застосунок з високою ймовірністю використовуватиме хмарні БД, а також зберігатиме їх резервні копії. Для зберігання файлів сховища такого типу використовуються при роботі з системами контролю версій, мессенджерами, файлообмінниками, застосунками для перегляду фотографій у смартфонах. Деякі сервіси просто дозволяють віддалено зберігати певний обсяг документів на їх серверах. Але всі перелічені підходи мають свої недоліки. Найбільшою проблемою є обмежений обсяг даних, який можна зберігати в хмарних сховищах. Збільшення цього обсягу також збільшить витрати на користування подібними сервісами. Тому в цій роботі пропонується один з шляхів вирішення вказаної проблеми. Планується розробити програмне забезпечення для зберігання і обміну файлами, враховуючи переваги альтернативних рішень. Готовий додаток має розгортатись на локальному середовищі користувача, і, після надання відповідних доступів, давати іншим користувачам можливість завантажувати поширені файли.

1.3 Аналіз існуючих технологій та успішних ІТ-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації локального файлообмінника. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

При аналізі алгоритмічних рішень виявлено, що для підвищення ефективності системи безпеки даних додатку варто розглянути можливість використання алгоритмів хешування для збереження паролів у БД. Серед них

можна виділити bcrypt[3], scrypt[4] і argon2[5]. Прийнято рішення про використання алгоритму bcrypt, оскільки він надає необхідний рівень захисту паролів і найбільш сумісний з обраними в наступному підрозділі інструментами.

Щодо технічних рішень було розглянуто переваги різних архітектур ПЗ та патернів проектування. Чиста архітектура має всі якості, необхідні для зручної розробки програмного забезпечення: гнучкість, незалежність компонентів і простота в тестуванні[6]. Але для файлообмінника її впровадження лише ускладнить процес розробки. Тому вирішено використовувати багатошарову архітектуру[7].

Після оцінки інших технічних рішень для програмного забезпечення, задачею якого є мережевий обмін даними, було вирішено впровадити клієнт-серверну архітектуру. Це підтримає чистоту коду і якість тестування. Також надано перевагу розробці веб-застосунку над програмою для робочого стола, оскільки в майбутньому це дозволить додавати інтеграції зі сторонніми сервісами[8]. При цьому сервер має підтримувати принципи REST[9], а клієнт має надавати користувачеві неблокуючий інтерфейс.

Для доступу до даних вирішено впровадити патерни Репозиторій[10] і Одиниця Роботи[11]. Їх перевага полягає у високому рівні абстракцій і гарантії узгодженості[12] даних у БД після транзакції. Поєднання цих патернів разом з ORM-фреймворком призводить до спрощення роботи з даними, що позитивно впливає на гнучкість розробки, простоту в підтриманні і чистоту коду програмного застосунку. Тому цей вибір є найкращим для моделі даних невеликих додатків, такі як файлообмінник.

На відміну від моделі даних, рішення для якої наведено в попередньому абзаці, файли зберігаються засобами ОС. Однією з вимог до додатку, що розробляється, є кросплатформність, тому програмне забезпечення повинно розгортатися на декількох цільових платформах. Єдине обмеження – ОС повинна мати змогу запускати веб сервери для веб

застосунків. Такими є представники сімейств Windows, Linux Debian, Mac OS.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Оскільки додаток для обміну файлів є веб застосунком, в першу чергу варто розглянути платформи для створення програм подібного типу. Одним з рішень є API сокетів[13], що надає повний контроль над мережевим обміном даних і дозволяє впроваджувати високопродуктивні алгоритми мережевого спілкування, в тому числі і обміну файлів. Великим недоліком є суттєве зростання складності розробки. Протилежно попередньому підходу ще одним рішенням є Low-code платформи[14], яким притаманне зменшення складності розробки, але при цьому обмежується доступ розробника до багатьох функцій і налаштувань. Найбільше це помітно при роботі з СУБД і системою контролю версій. Усередненим варіантом є використання бібліотек мови програмування JavaScript для клієнтської частини і MVC-фреймворків інших популярних мов для серверної[15]. Вони повністю покривають вимоги для задач зі створення веб застосунків.

Аналізуючи реляційні[16] і нереляційні[17] СУБД було помічено, що перші, незалежно від їх функціоналу, повністю задовольняють вимоги до моделі даних майбутнього застосунку, в той час як другі не мають жодних переваг. Файлообмінник міститиме сутності зі сталим набором атрибутів, а також очевидне існування відношення між файлами та користувачами, тому вибір зроблений на користь реляційних СУБД.

Щодо засобів розробки, то мовами програмування було обрано C# і JavaScript. Відповідно, при розробці додатку використовуватимуться такі утиліти, як React.js[18], React Router 6[19], Material UI[20], ASP.NET Core[21], Entity Framework Core[22], Visual Studio[23], SQL Server[24].

1.3.3 Аналіз відомих програмних продуктів

При пошуку аналогів додатку виявлено три найбільш схожі за функціоналом сервіси. Головною відмінністю цих продуктів є неможливість розгорнути їх на локальному середовищі, доступ обмежений і можливий лише при наявності інтернет-з'єднання.

Google Drive[25] – найбільш популярний сервіс для хмарного збереження даних. Основною його перевагою серед аналогів є те, що користувачеві з безкоштовним тарифом надається 15 ГБ вільного місця. Серед послуг, що надає сервіс, є попередній перегляд файлів та їх властивостей, налаштування синхронізації з комп'ютерами, перегляд доступних файлів інших користувачів, а для файлів певних розширень передбачені плагіни безкоштовних сервісів Google або інших сторонніх сервісів для перегляду та редагування документів у вікні веб-браузера.

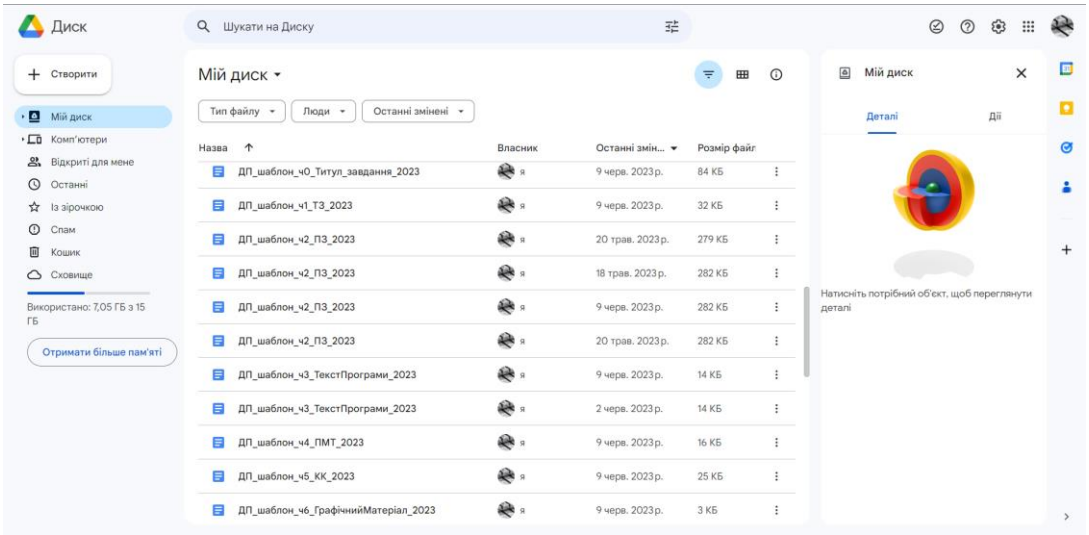


Рисунок 1.1 – Сервіс Google Drive

OneDrive[26] – схоже рішення від Microsoft, основною перевагою якого є повна інтеграція з іншими продуктами компанії. Також додаток встановлюється разом із Windows 10. Користувачеві з безкоштовним тарифом надається 5 ГБ вільного місця. Серед послуг, що надає сервіс, є попередній перегляд файлів та їх властивостей, перегляд доступних файлів інших користувачів. Для файлів певних розширень передбачений

безкоштовний сервіс Microsoft 365 Web, який надає потужні інструменти для редагування документів, таблиць, презентацій тощо.

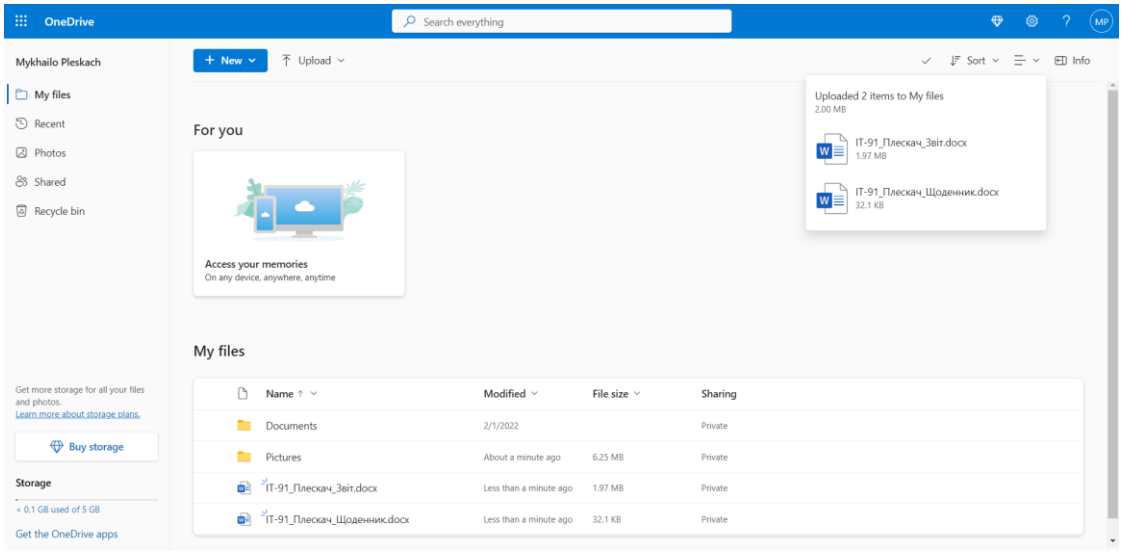


Рисунок 1.2 – Сервіс OneDrive

Dropbox[27] – в порівнянні з іншими продуктами сервіс надає більше функціоналу для керування сховищем. Серед них консоль адміністратора, журнал переглядів, резервні копії, функції захисту і багато іншого. Серед послуг, що надає сервіс, є попередній перегляд файлів та їх властивостей, перегляд доступних файлів інших користувачів. Значним недоліком є те, що безкоштовний тариф включає в себе всього 2 ГБ вільного місця. Також відсутні власні інструменти для редагування файлів.

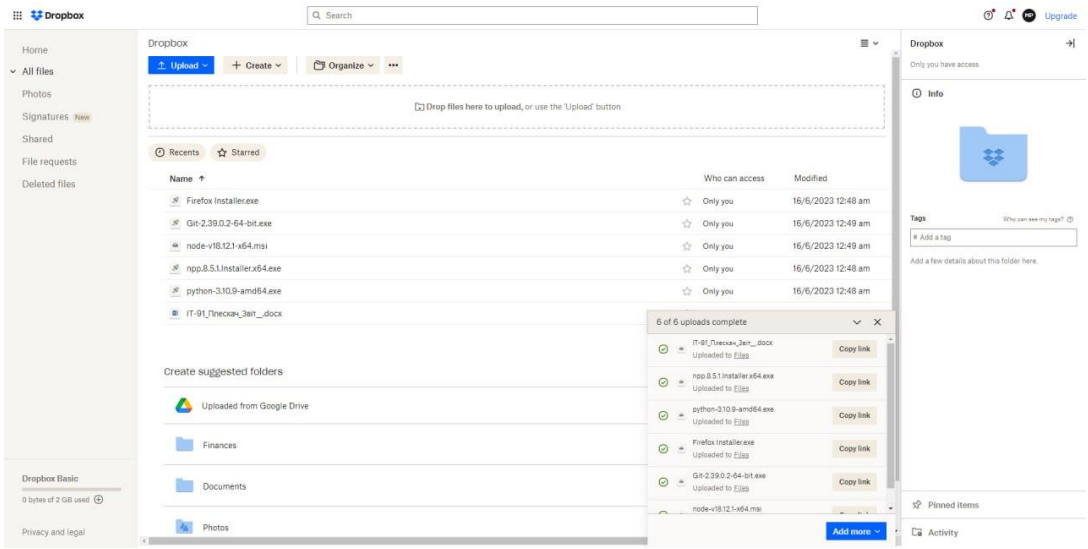


Рисунок 1.3 – Сервіс Dropbox

Для порівняння програмного забезпечення з аналогами можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогами

Критерії	**Файлообмінник**	Google Drive	One Drive	Dropbox	Пояснення
Обсяг даних	Обмежений розміром накопичувача	Залежить від тарифного плану, макс. – 2 ТБ	Залежить від тарифного плану, макс. – 1 ТБ	Залежить від тарифного плану, макс. – 3 ТБ	Для аналогів обирався індивідуальний план
Швидкість обміну даними	Обмежена пропускною здатністю з'єднання	Обмежена сервісом	Обмежена сервісом	Обмежена сервісом	
Надійність роботи	Середня	Висока	Висока	Висока	
Безпека даних	Найвища	Висока	Висока	Висока	
Керування файлами	Немає	Є	Є	Є	Не передбачено
Плагіни для редагування файлів	Немає	Власні, сторонні	Власні	Сторонні	Не передбачено
Розгортка додатку	Локально	Хмарно	Хмарно	Хмарно	Основна перевага

1.4 Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є обмін файлами, більше функцій можна побачити в розділі "Графічний матеріал" на схемі структурній варіантів використання, документ КПІ.ІТ-9121.045440.06.99.СС.

В таблицях 1.2 – 1.13 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 – Варіант використання UC-1

Use case name	Переглянути профілі
Use case ID	UC-01
Goals	Відобразити користувачеві список наявних профілів
Actors	Користувач (власник)

Продовження таблиці 1.2

Trigger	Користувач бажає переглянути наявні профілі
Pre-conditions	-
Flow of Events	На сторінці поширених файлів користувач натискає на кнопку переходу на сторінку профілів. На ній відображається список усіх профілів додатку.
Extension	На сторінці списку можна обрати профіль для редагування або видалення
Post-Condition	Користувач перейшов на сторінку профілів.

Таблиця 1.3 – Варіант використання UC-2

Use case name	Створити профіль
Use case ID	UC-02
Goals	Додати профіль зовнішнього користувача до списку
Actors	Користувач (власник)
Trigger	Користувач бажає додати профіль
Pre-conditions	-
Flow of Events	Користувач натискає на кнопку додавання профілю. У формі він заповнює облікові дані профілю і натискає кнопку підтвердження. У випадку, якщо існує профіль зі вказаним логіном, користувач отримає повідомлення про неможливість виконання операції. Інакше профіль успішно додається до списку.
Extension	У випадку вводу недопустимих символів користувач отримує відповідне повідомлення.
Post-Condition	У випадку успіху профіль з’являється на сторінці профілів.

Таблиця 1.4 – Варіант використання UC-3

Use case name	Видалення профілю
Use case ID	UC-03
Goals	Видалити профіль зі списку
Actors	Користувач (власник)
Trigger	Користувач бажає видалити профіль
Pre-conditions	Користувач знаходиться на сторінці профілів
Flow of Events	Користувач обирає профіль для видалення і натискає на відповідну кнопку. Після цього профіль видаляється зі списку.
Extension	Зі списку доступів видаляються об’єкти доступу профілю до файлів.
Post-Condition	Профіль зникає зі сторінки профілів. Зі сторінки доступів видаляються пов’язані з профілем записи.

Таблиця 1.5 – Варіант використання UC-4

Use case name	Редагувати профіль
Use case ID	UC-04
Goals	Відобразити користувачеві елементи управління для редагування профілю
Actors	Користувач (власник)
Trigger	Користувач бажає редагувати профіль
Pre-conditions	Користувач знаходиться на сторінці профілів
Flow of Events	Користувач обирає профіль для редагування а після цього обирає, що саме він хоче редагувати. Йому відображується відповідний елемент управління.
Extension	У відкритому елементі управління можна редагувати облікові дані профілю або його доступ до файлів.
Post-Condition	Користувачеві доступний інтерфейс для редагування профілю

Таблиця 1.6 – Варіант використання UC-5

Use case name	Редагувати облікові дані профіля
Use case ID	UC-05
Goals	Зміна облікових даних профіля
Actors	Користувач (власник)
Trigger	Користувач бажає редагувати облікові дані профіля
Pre-conditions	Користувачеві доступний інтерфейс для редагування облікових даних профілю.
Flow of Events	У формі користувач заповнює облікові дані профілю і натискає кнопку підтвердження. У випадку, якщо існує профіль зі вказаним логіном, користувач отримає повідомлення про неможливість виконання операції. Інакше облікові дані профілю успішно змінюються.
Extension	У випадку вводу недопустимих символів користувач отримує відповідне повідомлення.
Post-Condition	Оновлення облікових даних профілю

Таблиця 1.7 – Варіант використання UC-6

Use case name	Редагувати доступ профіля до файлу
Use case ID	UC-06
Goals	Зміна доступу профіля до файлу
Actors	Користувач (власник)
Trigger	Користувач бажає редагувати доступ профіля до файлу
Pre-conditions	Користувачеві доступний інтерфейс для редагування доступу профіля до файлу.

Продовження таблиці 1.7

Flow of Events	В інтерфейсі редагування профілю в списку поширених файлів користувач надає або обмежує доступ до обраного файлу через прапорець. Після натискання на нього статус доступу змінюється на протилежний
Extension	-
Post-Condition	Оновлення доступу профіля до файлу

Таблиця 1.8 – Варіант використання UC-7

Use case name	Переглянути поширені файли
Use case ID	UC-07
Goals	Відобразити користувачеві список поширених файлів
Actors	Користувач (власник)
Trigger	Користувач бажає переглянути поширені файли
Pre-conditions	-
Flow of Events	На головній сторінці користувач одразу бачить список поширених файлів. Якщо він знаходиться на іншій сторінці, то має натиснути на кнопку переходу на головну.
Extension	На сторінці списку можна видалити файл зі списку поширених.
Post-Condition	Користувач перейшов на сторінку поширених файлів (головну)

Таблиця 1.9 – Варіант використання UC-8

Use case name	Додати файл до поширених
Use case ID	UC-08
Goals	Додати файл до списку поширених
Actors	Користувач (власник)
Trigger	Користувач бажає додати файл до списку поширених
Pre-conditions	-
Flow of Events	Користувач натискає на кнопку додавання файлу. Йому відображається вікно вибору файлу файлового менеджера ОС. У вікні він обирає файл і натискає кнопку підтвердження. У випадку, якщо файл вже поширено, користувач отримає повідомлення про неможливість виконання операції. Інакше файл додається до списку.
Extension	-
Post-Condition	Файл додається до списку поширених

Таблиця 1.10 – Варіант використання UC-9

Use case name	Вилучити файл з поширених
Use case ID	UC-09
Goals	Вилучити файл зі списку поширених

Продовження таблиці 1.10

Actors	Користувач (власник)
Trigger	Користувач бажає вилучити файл зі списку поширених
Pre-conditions	Користувач знаходиться на сторінці поширених файлів
Flow of Events	Користувач обирає файл для видалення і натискає на відповідну кнопку. Після цього файл видаляється зі списку.
Extension	Зі списку доступів видаляються об’єкти доступу профілів до файлу.
Post-Condition	Файл зникає зі списку поширених файлів на головній сторінці. Зі сторінки доступів видаляються пов’язані з файлом записи.

Таблиця 1.11 – Варіант використання UC-10

Use case name	Авторизуватися на віддаленому сервері
Use case ID	UC-10
Goals	Авторизація зовнішнього користувача в системі
Actors	Користувач (зовнішній)
Trigger	Користувач бажає авторизуватися в системі
Pre-conditions	-
Flow of Events	У формі авторизації користувач вводить облікові дані. У випадку їх повного збігу його авторизує під відповідним профілем. Інакше відображається повідомлення про невдалу авторизацію.
Extension	На сторінці доступів користувач має можливість завантажити доступні йому файли.
Post-Condition	Користувач перейшов на сторінку доступів.

Таблиця 1.12 – Варіант використання UC-11

Use case name	Переглянути доступні файли
Use case ID	UC-11
Goals	Відобразити користувачеві список доступних файлів
Actors	Користувач (зовнішній)
Trigger	Користувач бажає переглянути доступні файли
Pre-conditions	Користувач авторизований
Flow of Events	Після авторизації зовнішній користувач знаходиться на сторінці доступів і одразу бачить список доступних йому файлів.
Extension	На сторінці доступів користувач може завантажити доступний йому файл.
Post-Condition	Користувач перейшов на сторінку доступних файлів

Таблиця 1.13 – Варіант використання UC-12

Use case name	Завантажити файл
Use case ID	UC-12
Goals	Завантажити файл у файлову систему зовнішнього користувача
Actors	Користувач (зовнішній)
Trigger	Користувач бажає завантажити доступний йому файл
Pre-conditions	Користувач знаходиться на сторінці доступних файлів
Flow of Events	Користувач обирає файл для завантаження зі списку доступних і натискає на відповідну кнопку. Після цього починається процес завантаження файлу через локальну мережу або Інтернет.
Extension	-
Post-Condition	Файл завантажено у файлову систему зовнішнього користувача

1.4.1 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. Нижче наведено загальну модель вимог:

- Керування профілями, FR-1: високий пріоритет;
  - Перегляд профілів, FR-2: середній пріоритет;
  - Створення профілю, FR-3: високий пріоритет;
  - Видалення профілю, FR-4: низький пріоритет;
  - Редагування облікових даних профілю, FR-5: низький пріоритет;
  - Редагування доступу профіля до файлу, FR-6: високий пріоритет;
- Керування поширеними файлами, FR-7: високий пріоритет
  - Перегляд поширених файлів, FR-8: середній пріоритет;
  - Додавання файлу до поширених, FR-9: високий пріоритет;
  - Вилучення файлу з поширених, FR-10: низький пріоритет;
- Авторизація користувача, FR-11: високий пріоритет;
- Доступ до файлів, FR-12: високий пріоритет;
  - Перегляд доступних файлів, FR-13: високий пріоритет;
  - Завантаження файлу, FR-14: високий пріоритет.

В таблицях 1.14 – 1.27 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 1.4.

Таблиця 1.14 – Функціональна вимога FR-1

Назва	Керування профілями
Опис	Система повинна надавати користувачу можливість керувати профілями: створювати, переглядати, редагувати і видаляти

Таблиця 1.15 – Функціональна вимога FR-2

Назва	Перегляд профілів
Опис	Система повинна надавати користувачу можливість переглядати список створених ним профілів.

Таблиця 1.16 – Функціональна вимога FR-3

Назва	Створення профілю
Опис	Система повинна надавати користувачу можливість створювати профілі для зовнішніх користувачів шляхом введення логіну і паролю.

Таблиця 1.17 – Функціональна вимога FR-4

Назва	Видалення профілю
Опис	Система повинна надавати користувачу можливість видаляти створені профілі.

Таблиця 1.18 – Функціональна вимога FR-5

Назва	Редагування облікових даних профілю
Опис	Система повинна надавати користувачу можливість редагувати логін і пароль профілю.

Таблиця 1.19 – Функціональна вимога FR-6

Назва	Редагування доступу профіля до файлу
Опис	Система повинна надавати користувачу можливість надавати або обмежувати доступ профілів до файлу зі списку поширених.

Таблиця 1.20 – Функціональна вимога FR-7

Назва	Керування поширеними файлами
Опис	Система повинна надавати користувачу можливість керувати файлами:переглядати, додавати до поширених і видаляти їх

Таблиця 1.21 – Функціональна вимога FR-8

Назва	Перегляд поширених файлів
Опис	Система повинна надавати користувачу можливість переглядати поширені ним файли.

Таблиця 1.22 – Функціональна вимога FR-9

Назва	Додавання файлу до поширених
Опис	Система повинна надавати користувачу можливість поширювати файли, що знаходяться у файловій системі його середовища.

Таблиця 1.23 – Функціональна вимога FR-10

Назва	Вилучення файлу з поширених
Опис	Система повинна надавати користувачу можливість вилучати файли зі списку поширених.

Таблиця 1.24 – Функціональна вимога FR-11

Назва	Авторизація користувача на віддаленому сервері
Опис	Система повинна надавати зовнішньому користувачу можливість шляхом введення логіну і паролю.

Таблиця 1.25 – Функціональна вимога FR-12

Назва	Доступ до файлів
Опис	Система повинна надавати зовнішньому користувачу доступ до поширених йому файлів.

Таблиця 1.26 – Функціональна вимога FR-13

Назва	Перегляд доступних файлів
Опис	Система повинна надавати зовнішньому користувачу можливість переглядати список доступних йому файлів.

Таблиця 1.27 – Функціональна вимога FR-14

Назва	Завантаження файлу
Опис	Система повинна надавати зовнішньому користувачу можливість завантажувати доступні йому файли.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14
UC-1	+	+												
UC-2	+		+											
UC-3	+			+										
UC-4	+				+	+								
UC-5	+				+									
UC-6	+					+								
UC-7							+	+						
UC-8							+		+					
UC-9							+			+				
UC-10											+			
UC-11												+	+	
UC-12												+		+

Рисунок 1.4 – Матриця трасування вимог

1.4.2 Розроблення нефункціональних вимог

Для підтримки високої якості програмного забезпечення визначено необхідні вимоги для забезпечення його зручності, безпеки, швидкодії та сумісності. Було складено список наступних нефункціональних вимог:

– Передбачити контроль введення інформації та захист від некоректних дій користувача;

– Забезпечити перевірку отриманих запитів і відхиляти їх у випадку некоректних даних;

– Забезпечити цілісність інформації в базі даних;

– Забезпечити сумісність серверної частини програмного забезпечення з операційними системами сімейств Windows NT, Linux Debian, MacOS.

1.5 Постановка задачі

Метою проекту є розробка програмного забезпечення для обміну файлів, що поєднує властивості хмарних сховищ і надає користувачеві можливість встановлення додатку на своє локальне середовище.

Було визначено наступні функціональні задачі проекту:

– Розробити систему авторизації користувачів;

– Надати користувачам можливість отримати доступ до файлів;

– Надати адміністратору можливість керувати файлами;

– Надати адміністратору можливість керувати обліковими записами користувачів.

Висновки до розділу

У цьому розділі було проведено аналіз вимог до програмного забезпечення для обміну файлів.

Досліджена предметна область, до якої відноситься застосунок – хмарні сховища. Це дуже перспективний напрямок розвитку IT-технологій, оскільки такі інструменти використовуються всюди. Тому можна знайти чимало аналогів додатку, що планується розробити, наприклад Google Drive, OneDrive, Dropbox. Застосунок має суміщати переваги цих сервісів зі своїм унікальним функціоналом.

Визначено найкращі алгоритмічні та технічні рішення для розробки додатку – тришарова архітектура, клієнт-серверна архітектура, патерни Репозиторій і Одиниця Роботи, алгоритм хешування паролів bcrypt.

Також встановлено вимоги до операційних систем – Windows, Linux Debian, Mac OS. Визначено основні інструменти для розробки додатку – React.js, React Router 6, MaterialUI, Visual Studio Code, ASP.NET Core, Entity Framework Core, Visual Studio, SQL Server.

В результаті аналізу вимог було побудовано діаграму варіантів використання (КПІ.ІТ-9121.045440.06.99.СС).

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Провівши аналіз вимог і визначивши варіанти використання програмного забезпечення слід описати його бізнес процеси. Їх можна розділити на три види: робота з профілями, робота з доступом до файлів і робота з обміном файлів. Останній описує процеси з участю зовнішніх користувачів.

2.1.1 Моделювання роботи з профілями

Важливою складовою розробленого додатку є профілі – облікові записи. Вони обмежують поширення файлів, надаючи доступ лише тим користувачам, які мають відповідні облікові дані. Такий підхід впроваджує захист інформації від несанкціонованого доступу і є основою безпеки файлообмінника. За керування профілями відповідає власник серверу.

2.1.1.1 Процес створення профілю

Процес розпочинається з перевірки на його існування. Об’єкти порівнюються наступним чином – якщо обліковий запис з вказаним при створенні логіном вже існує, то вважається, що також існує і профіль. В такому випадку процес завершується повідомленням про його зупинку. Якщо вказаний логін є унікальним для серверу, то профіль успішно додається до БД. Після цього процес завершується.

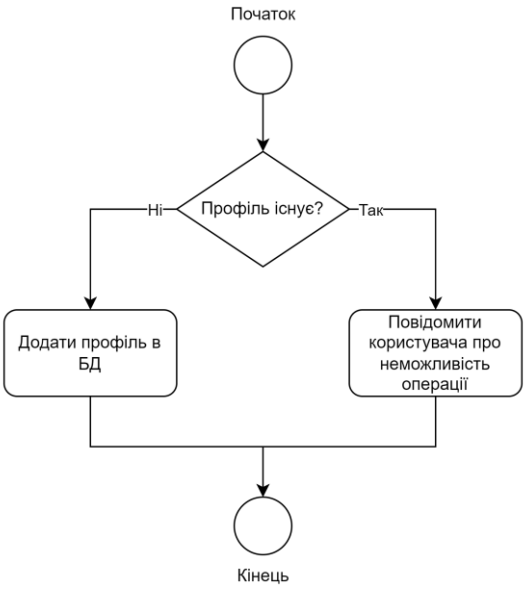


Рисунок 2.1 – Блок-схема процесу створення профілю

2.1.1.2 Процес видалення профілю

Схожий зі створенням профілю, цей процес так само перевіряє існування облікового запису, але для порівняння використовується унікальний ідентифікатор. Якщо такий є, то спочатку видаляються всі записи про поширені файли, потім – сам об’єкт. В іншому випадку процес завершується повідомленням про відсутність профіля, а БД не змінюється.

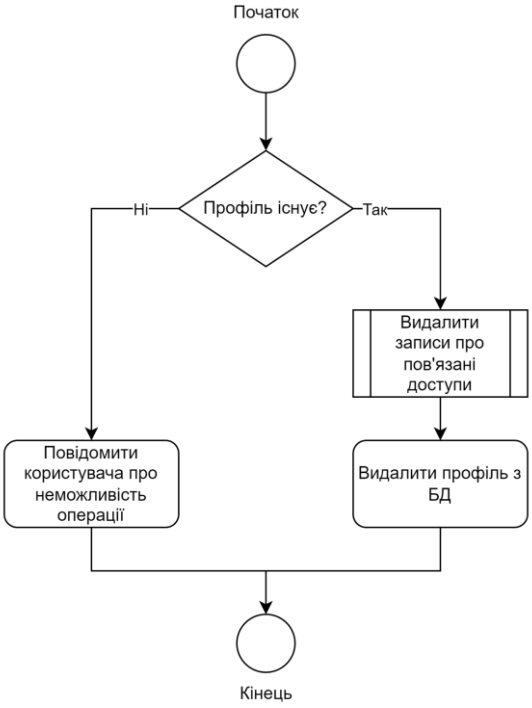


Рисунок 2.2 – Блок-схема процесу видалення профілю

2.1.1.3 Процес редагування профілю

Цей процес потребує додаткових даних – нові логін і пароль, що мають бути присвоєні профілю з вказаним унікальним ідентифікатором. Спочатку облікові дані перевіряються на недопустимі символи, і, якщо такі знайдено, процес зупиняється з відповідним повідомленням. Далі відбувається пошук профіля по ідентифікатору. У випадку успіху програма змінює його облікові дані, інакше – нічого не робить. Після цього процес завершується.

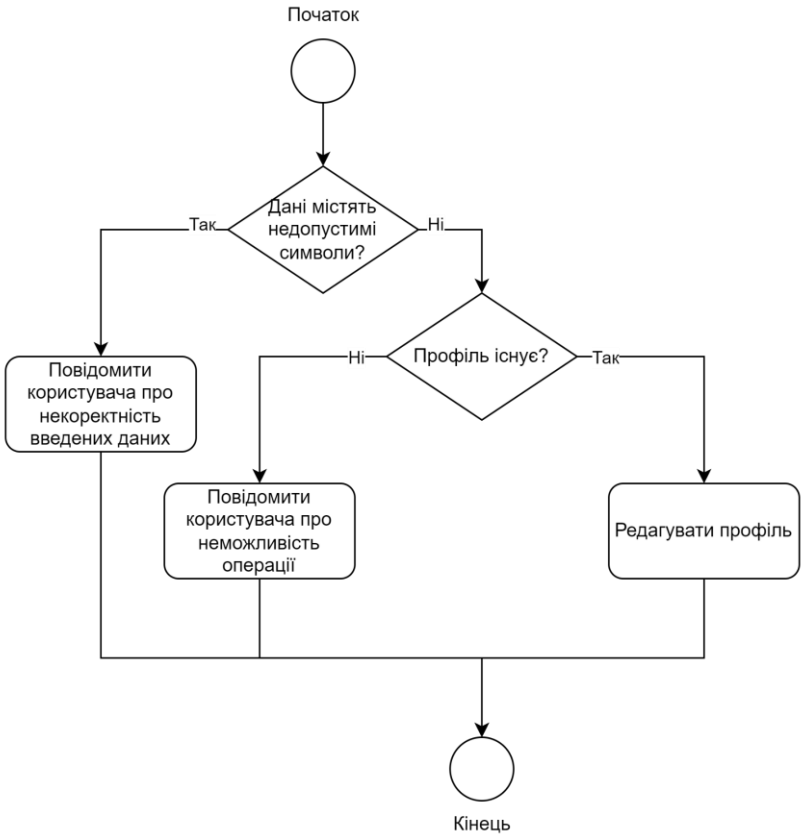


Рисунок 2.3 – Блок-схема процесу редагування профілю

2.1.2 Моделювання роботи з доступами до файлів

Головними компонентами локального сховища є файли та їх метадані. При плануванні архітектури і аналізі існуючих баз даних автором записки було вирішено використовувати реляційні СУБД, оскільки вони гарантують високу продуктивність роботи з сутностями та консистентність транзакцій. Тому для покращення швидкодії за збереження файлів відповідає цільова ОС. А інформація про метадані програмно керується СУБД. Користувач має

можливість додати файли до поширених або вилучити їх, а також редагувати доступ до них обраним профілям.

2.1.2.1 Процес поширення файлу

Для додавання файлу до списку поширених користувач має вказати до нього шлях на своєму локальному середовищі (ОС). Після цього у випадку відсутності в БД файлу з ідентичним шляхом додається запис про його метадані – назва, розмір, дата завантаження тощо. На цьому процес завершується.

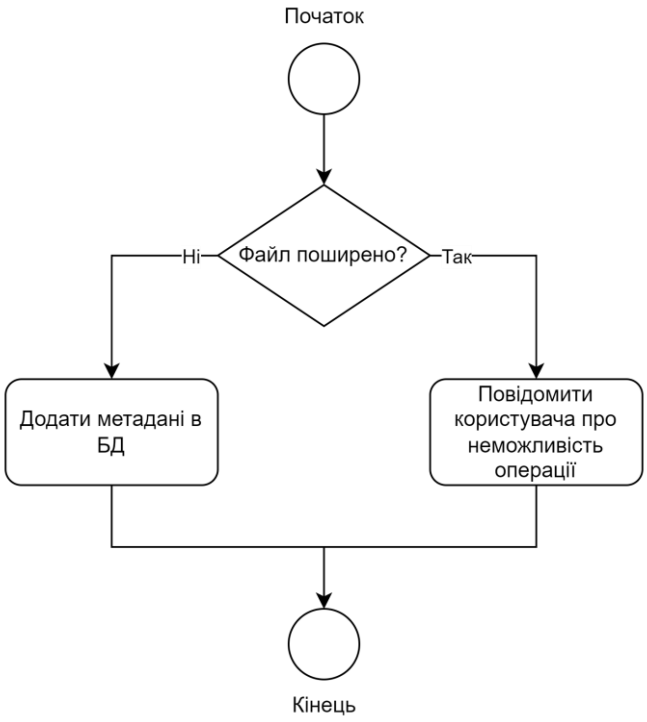


Рисунок 2.4 – Блок-схема процесу поширення файлу

2.1.2.2 Процес вилучення файлу з поширених

Процес, обернений до попереднього. Користувач вказує ідентифікатор метаданих, після чого з БД видаляються сам об’єкт і записи про доступ профілів до нього. Сам файл сервером не видаляється і в подальшому його можна знову додати до поширених.

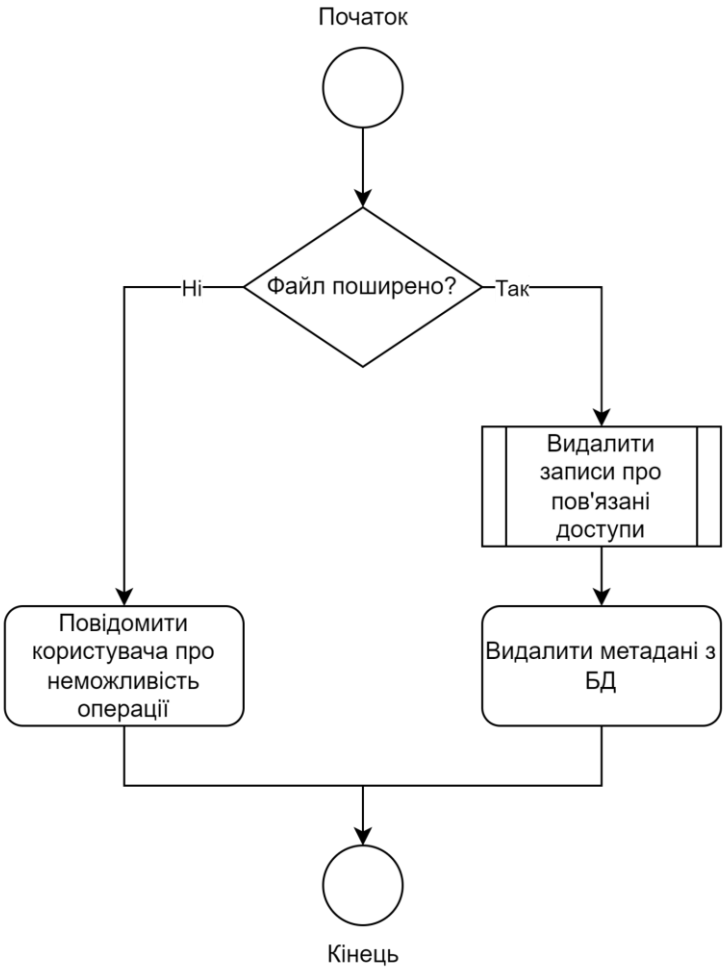


Рисунок 2.5 – Блок-схема процесу вилучення файлу з поширених

2.1.2.3 Процес надання профілю доступу до файлу

Саме цей процес відповідає за надання користувачам доступу до файлів з метою завантаження їх на клієнтське середовище. Спочатку перевіряються ідентифікатори профілю і метаданих, вказані користувачем на серверній частині. Якщо вони збігаються з даними будь-якого запису, то вважається, що доступ до файлу вже надано. В іншому випадку створюється новий і процес завершується.

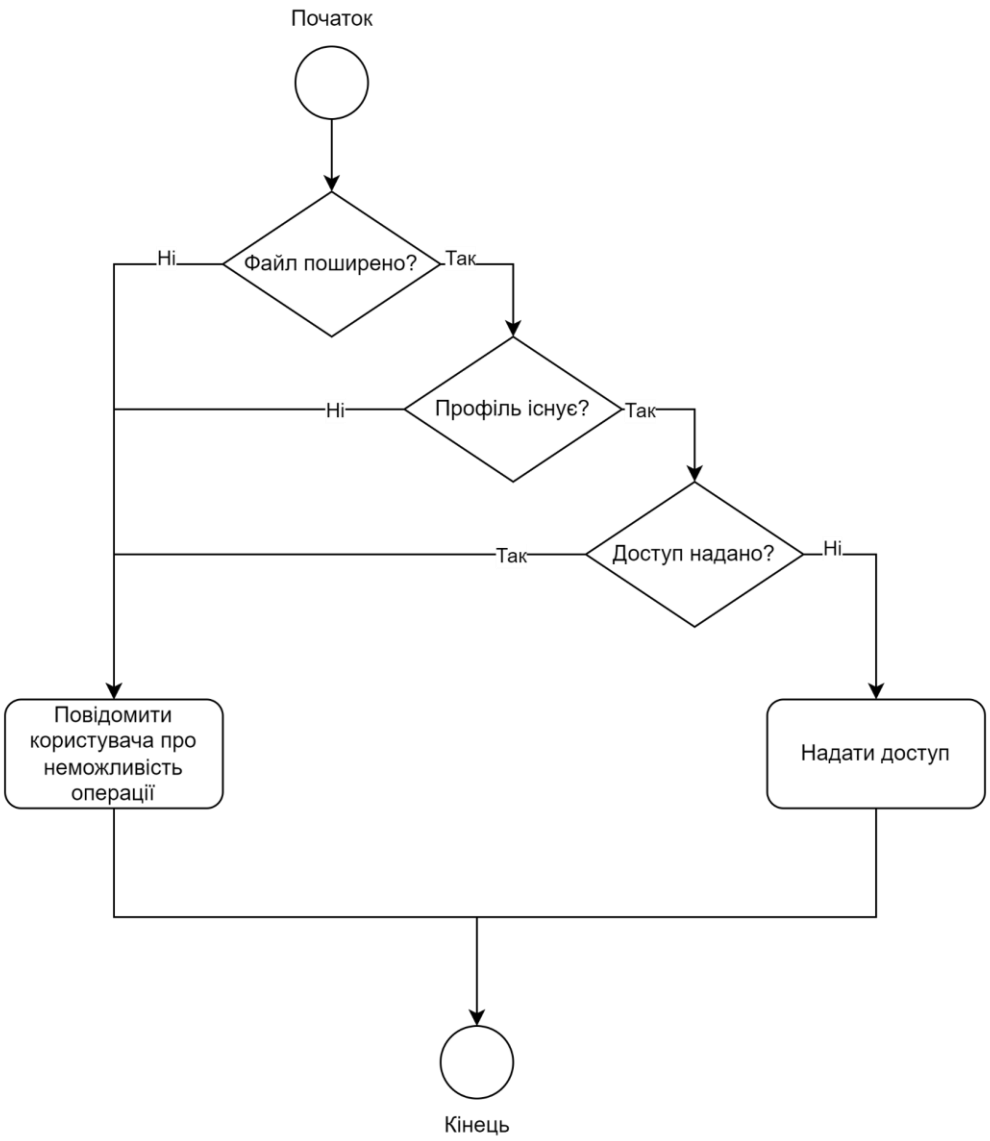


Рисунок 2.6 – Блок-схема процесу надання профілю доступу до файлу

2.1.2.4 Процес видалення у профіля доступу до файлу

Цей процес використовується для обмеження доступу профіля до файлу, якщо він є. Користувач вказує ідентифікатор доступу, і якщо відповідний запис існує, то він видаляється з БД. Цей процес також активується автоматично. У випадку видалення профіля пов’язані з ним записи про доступ визнаються недійсними і вилучаються з БД. У випадку видалення метаданих доступ до файлу обмежується для всіх профілів, яким він був наданий.

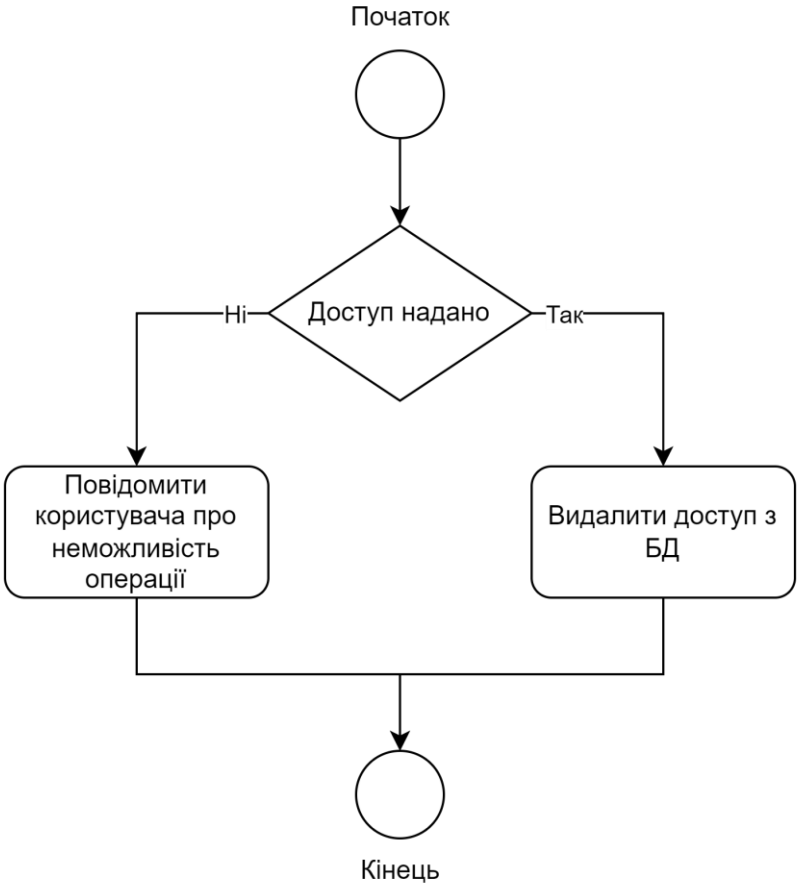


Рисунок 2.7 – Блок-схема процесу видалення у профіля доступу до файлу

2.1.3 Моделювання обміну файлів

Окрім роботи з доступами додаток надає зовнішнім користувачам можливість завантажувати поширені файли. Для цього вони мають встановити інтернет-з’єднання з сервером, на якому розгорнутий локальний файлообмінник і авторизуватись під створеними для них профілями.

2.1.3.1 Процес авторизації

Для отримання доступу до файлів зовнішній користувач має авторизуватися за обліковими даними профілю, наданими йому локальним користувачем. Процес відбувається наступним чином. Після відправки форми відбувається пошук в БД профіля по логіну, за цим йде порівняння паролей. Якщо логін не є дійсним або пароль не збігається з указаним, то користувач отримує повідомлення про невдалу авторизацію. Якщо облікові

дані виявляються дійсними, то користувачеві надається доступ до файлів, які були надані відповідному профілю. На цьому процес завершується.

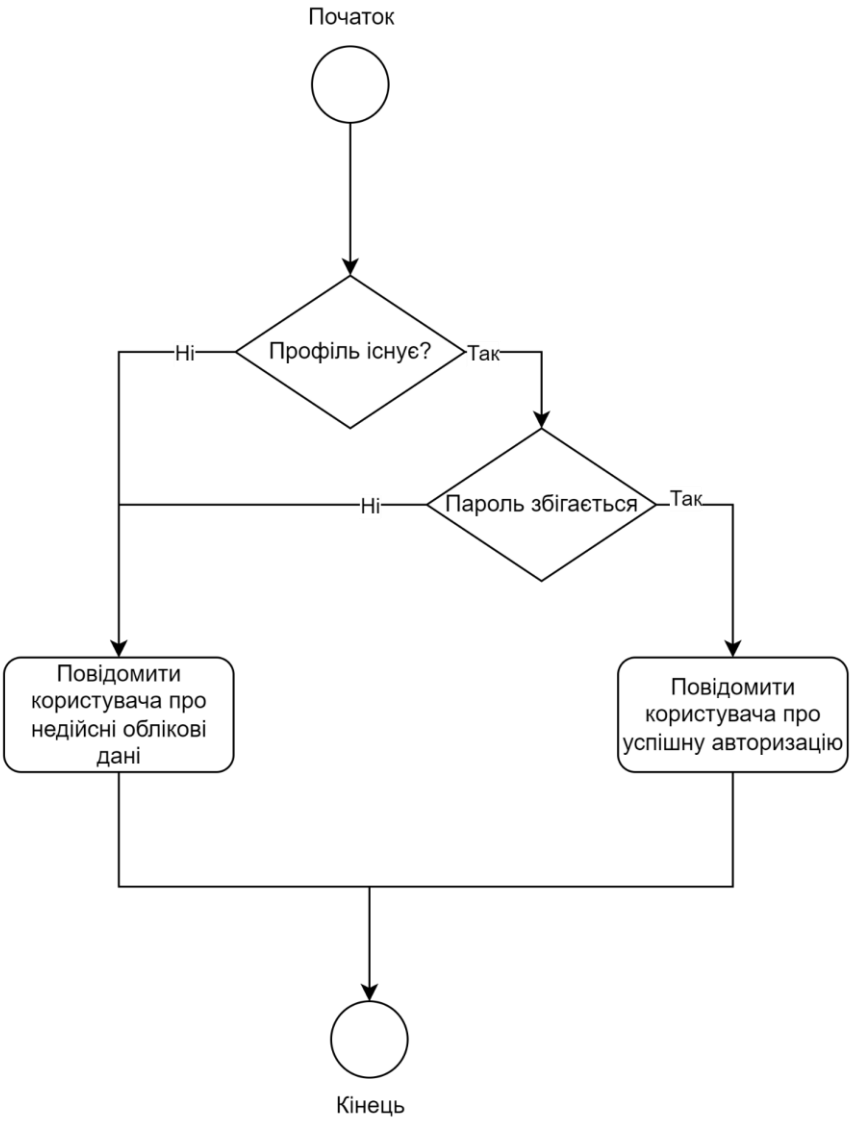


Рисунок 2.8 – Блок-схема процесу авторизації

2.1.3.2 Процес завантаження файлу

Для завантаження файлу зовнішній користувач має бути авторизованим. В іншому випадку доступ буде неможливий. Після вибору файлу, який користувач бажає завантажити, відбувається пошук в БД запису про доступ. Якщо такий існує, з метаданих обирається шлях файлу. Після цього документ передається по мережевому протоколу з серверу на клієнт, що запросив завантаження. На цьому процес завершується.

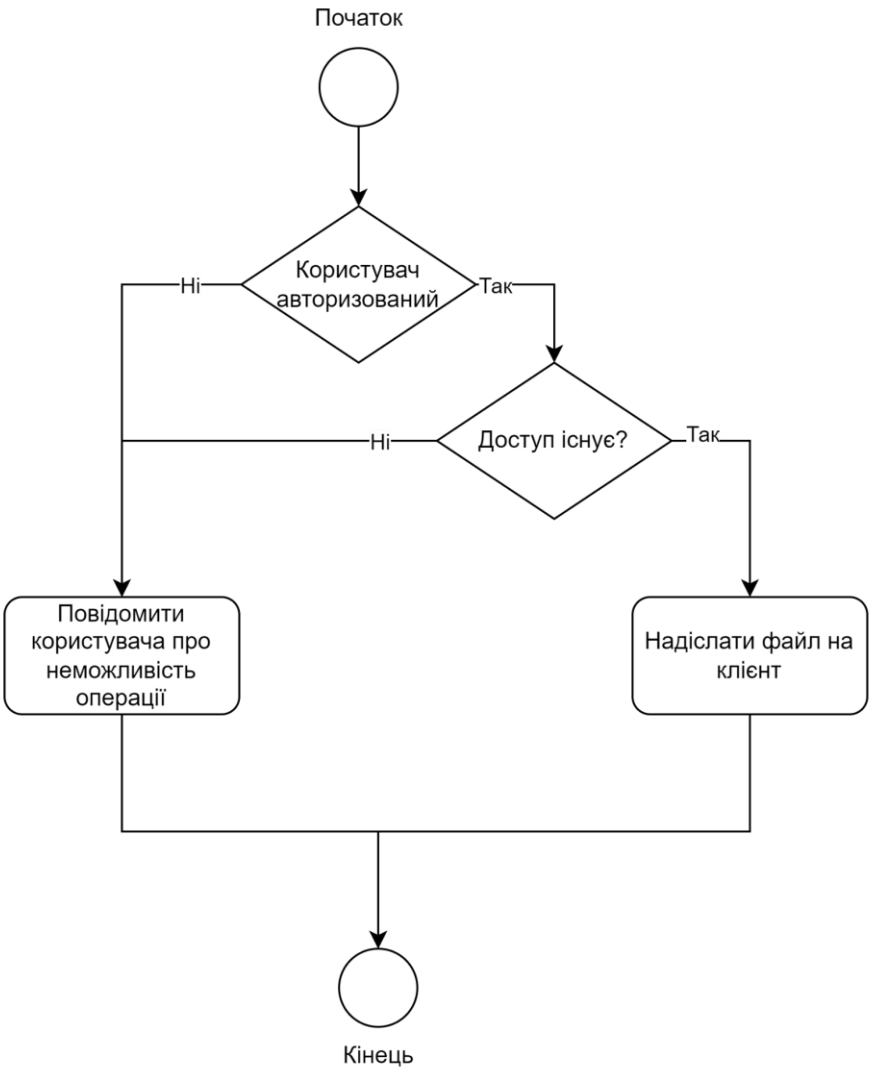


Рисунок 2.9 – Блок-схема процесу завантаження файлу

2.2 Архітектура програмного забезпечення

Для додатку з обміну файлів автором записки обрано тришарову архітектуру. Вона забезпечує розширюваність і ефективну підтримку програмного забезпечення.

Найнижчим рівнем тришарової архітектури є шар доступу до даних (Data Access Layer). Ціль його компонентів полягає у наданні узагальненого інтерфейсу для керування даними. На цьому рівні абстракції надають шару вище методи для роботи з сутностями, не конкретизуючи яка саме СУБД використовується. Щоб цього досягти для розробки файлообмінника

використовуються патерни Репозиторій для надання доступу до таблиць або колекцій і Одиниця Роботи для виконання операцій запису однією транзакцією. Це дозволить додатку видаляти профілі і метадані файлів з пов’язаними з ними доступами без ризику що залишаться зайві дані.

Середнім рівнем тришарової архітектури є шар бізнес-логіки (Business Logic Layer). В ньому імплементуються описані вище бізнес процеси. Уся робота з даними, які зберігаються в БД, описується в сервісах – основних компонентах шару. До неї належить керування профілями, метаданими та доступами через абстракції репозиторіїв цих сутностей.

Найвищим рівнем тришарової архітектури є шар презентацій (Presentation Layer). Ціль його компонентів у спілкуванні користувацького інтерфейсу з програмним кодом додатку і запуску відповідних бізнес процесів. При проектуванні було обрано патерн MVC, тому основними компонентами цього шару є контролери. Їх ціль – перетворювати введені користувачем дані у формат, який використовується в шарі бізнес логіки. У файлообміннику вони виконують додаткову функцію – запускають процес завантаження файлу на клієнт. Також шар містить проміжне програмне забезпечення (Middleware). Його ціль в додатку – авторизація користувачів.

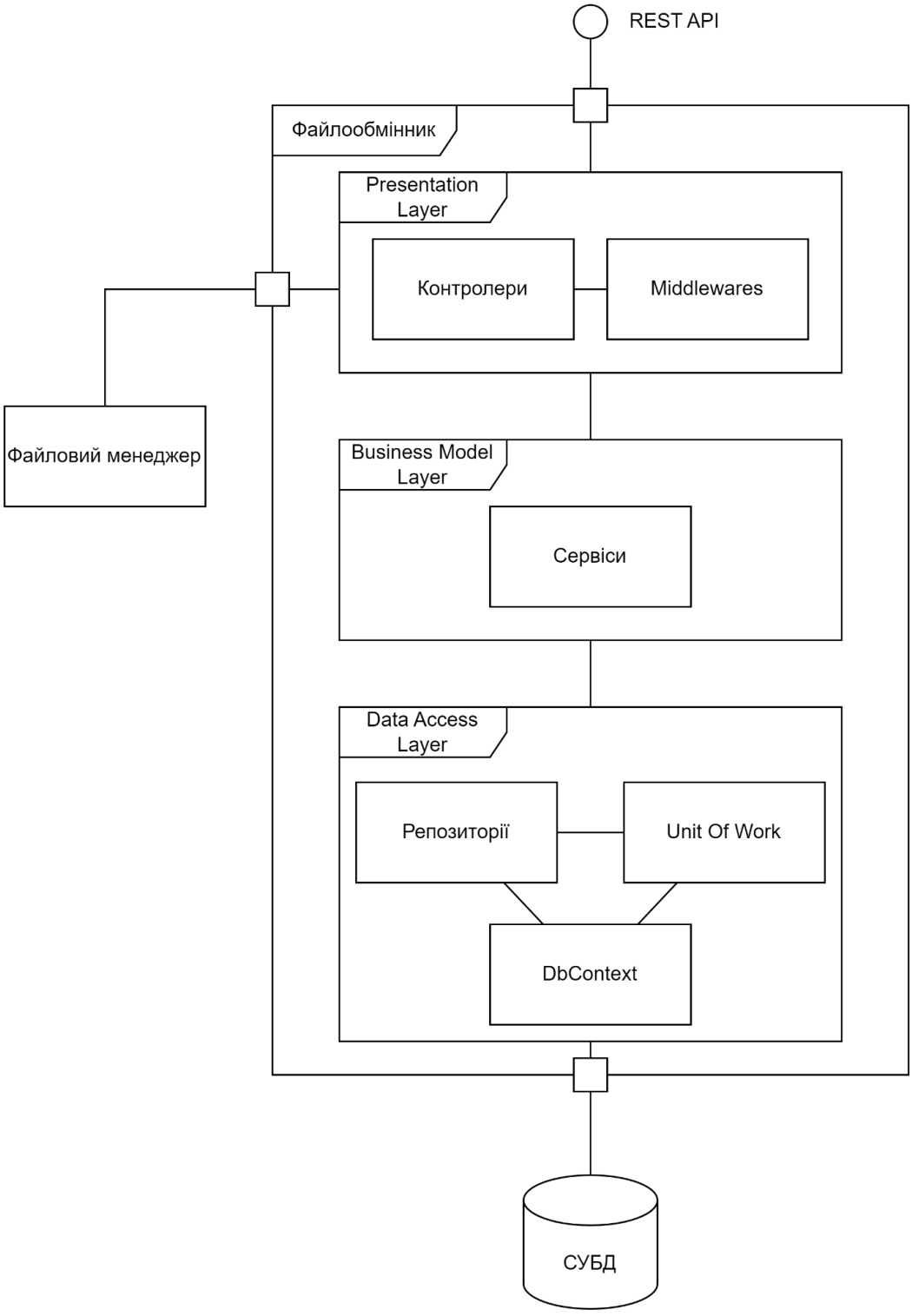


Рисунок 2.10 – Архітектура додатку

Окрім цього додаток також слідує принципам REST, що дозволить створювати інтеграції зі сторонніми сервісами.

Розберемо архітектуру додатку докладніше на прикладі процесу завантаження файлу.

Спершу по протоколу HTTP надходить запит на завантаженняя файлу у форматі JSON. Спочатку він проходить через обробку сервером та веб-платформою, серед яких валідація і перетворення документа на об’єкт класу. Після цього він потрапляє проміжне ПЗ, що відповідає за авторизацію. Після парсингу і підтвердження веб-токену запит надсилається на контролер.

Контролер надсилає запит сервісу на перевірку доступу профіля до файлу. Через компонент Unit of Work сервіс обирає репозиторій доступів, який через контекст даних шукає відповідний запит. Репозиторій сигналізує сервісу про успіх операції, а сервіс – контролеру.

Аналогічно контролер надсилає запит сервісу на отримання метаданих файлу, далі – на репозиторій, звідти – на контекст який повертає шуканий запис.

З цього запису контролер отримує шлях і викликає функцію завантаження файлу на клієнт. Після підтвердження зі сторони користувача додаток засобами платфоми передає вміст файлу через протокол FTP.

2.3 Конструювання програмного забезпечення

2.3.1 Модель даних

В попередньому розділі задля збереження консистентності даних було вирішено використовувати реляційну СУБД. Отже, модель будується з урахуванням зв’язків між сутностями. Після аналізу предметної області і нормалізації даних виявлено три необхідні сутності. Їх зображено на наступній діаграмі:

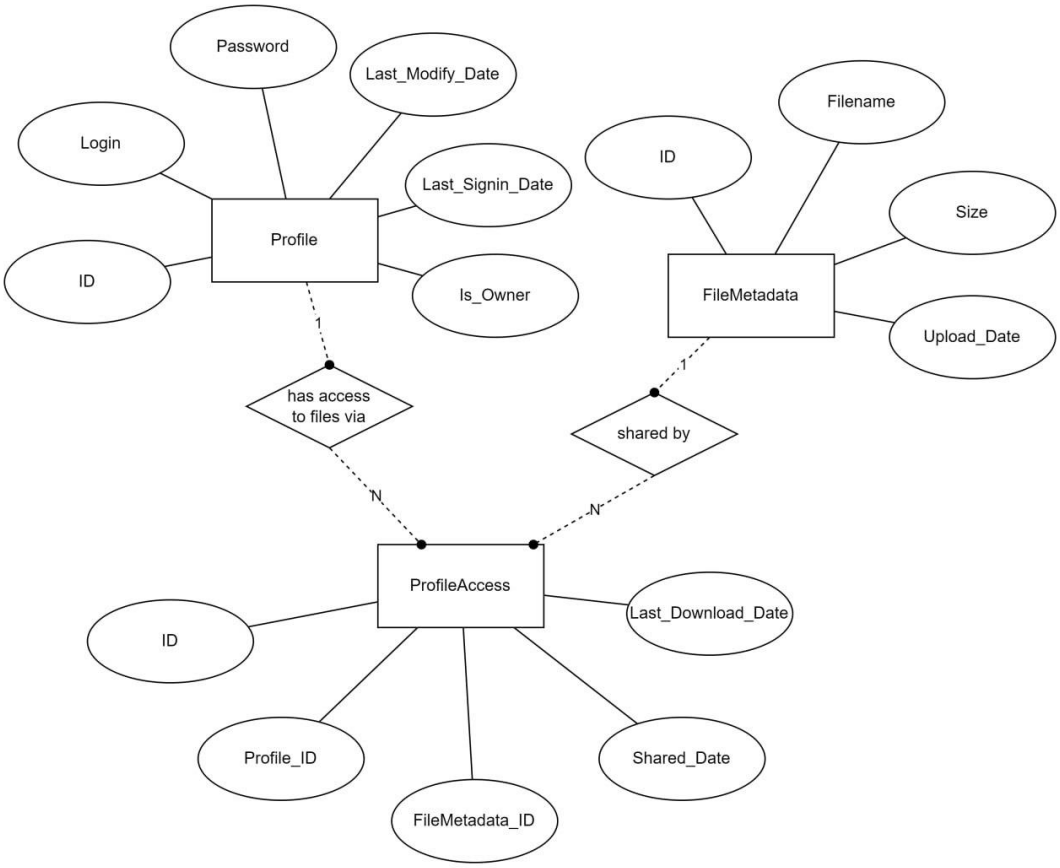


Рисунок 2.11 – ER діаграма сутностей FileMetadata, Profile, ProfileAccess

Нижче наведено опис атрибутів кожної сутності.

Таблиця 2.1 – сутність Profile

Атрибут	Тип даних	Опис	Примітка
ID	Ідентифікатор	Унікальний ідентифікатор	Унікальний
Login	Текст	Логін профіля	Унікальний
Password	Текст	Пароль профіля	Хешується на рівні бізнес логіки
Last_Modify_Date	Дата і час	Дата останнього редагування профіля	
Last_Signin_Date	Дата і час	Дата останньої авторизації	
Is_Owner	Логічна одиниця	Чи належить профіль власнику?	I

Таблиця 2.2 – сутність FileMetadata

Атрибут	Тип даних	Опис	Примітка
ID	Ідентифікатор	Унікальний ідентифікатор	Унікальний
Filename	Текст	Назва файлу	Унікальний
Size	Ціле число	Розмір у байтах	
Upload_Date	Дата і час	Дата поширення файлу	

Таблиця 2.3 – сутність ProfileAccess

Атрибут	Тип даних	Опис	Примітка
ID	Ідентифікатор	Унікальний ідентифікатор	Унікальний
Profile_ID	Ідентифікатор	Ідентифікатор профіля	Унікальна пара з FileMetadata_ID
FileMetadata_ID	Ідентифікатор	Ідентифікатор метаданих	Унікальна пара з Profile_ID
Shared_Date	Дата і час	Дата надання доступу	
Last_Download_Date	Дата і час	Дата останнього завантаження	

2.3.2 Структура програми

З урахуванням побудованої моделі даних, а також обраних патернів проектування і типу архітектури, було створено діаграму класів: Її можна знайти в розділі “Графічний матеріал” на схемі структурній класів програмного забезпечення, документ КПІ.ІТ-9121.045440.06.99.СС.

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.4.

Таблиця 2.4 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Visual Studio	Середовище розробки програмного забезпечення. Редактор сирцевого коду серверної частини застосунку.
2	ASP.NET	Платформа для створення кросплатформних веб застосунків. Підтримує MVC і DI, які є невід'ємною частиною додатку. Обробляє всі веб-запити до початку роботи контролерів.
3	Entity Framework Core	Фреймворк для керування даними в базах даних. Може бути використаний для створення БД з коду (code-first), надає можливість змінювати модель шляхом міграцій[28]. В додатку використовується як ORM для доступу до таблиць з сутностями.
4	SQL Server	Реляційна СУБД. Має багатий функціонал і легко підтримується з іншими утилітами Microsoft для розробки ПЗ. В файлообміннику використовується через розширення EF Core. Зберігає записи про профілі, метадані і доступи до файлів.
5	bcrypt.net	Стороння бібліотека для хешування паролів. Імплементує алгоритм з високим рівнем захищеності. В додатку використовується для хешування паролю перед додаванням у БД і порівнянням хешу і паролю, отриманого з форми авторизації.
6	React.js	Бібліотека для створення користувацьких інтерфейсів. Надає зручні інструменти для часткового оновлення сторінок.
7	React Router 6	Бібліотека для навігації між сторінками на стороні клієнта. Дозволяє змінювати вміст HTML-документа без HTTP-запитів.
8	Material UI	Бібліотека для створення сторінок з компонентів. Прибирає необхідність роботи з HTML-тегами. В додатку використовується для створення зручного і естетичного користувацького інтерфейсу.

Продовження таблиці 2.4

9	Visual Studio Code	Середовище розробки програмного забезпечення. Редактор сирцевого коду клієнтської частини застосунку.

2.4 Аналіз безпеки даних

Під час розробки додатку проводився аналіз безпеки даних. Оцінювалися різні складові програмного забезпечення: авторизація, зберігання даних в БД, мережеві протоколи, допоміжне ПЗ тощо. Було визначено необхідні покращення системи безпеки і виконано їх реалізацію:

– Усе програмне забезпечення, утиліти, фреймворки і бібліотеки, які використовуються при розробці додатку або використовуються ним, оновлені до останньої стабільної версії;

– Додаток підтримує спілкування протоколом HTTPS;

– Для авторизації користувачів використовується JWT токен, підписаний серверною частиною;

– Перед додаванням в БД паролі хешуються за допомогою алгоритму Bcrypt засобами бібліотеки bcrypt.net;

– Додаток виключає можливість атак з використанням включення коду і міжсайтового скриптингу[29];

– Транзакції бази даних атомарні і консистентні завдяки використанню Entity Framework Core в поєднанні з патернами Одиниця Роботи і Репозиторій.

Висновки до розділу

У цьому розділі виконано моделювання та конструювання програмного забезпечення для обміну файлів.

Спершу було визначено усі бізнес процеси застосунку. Для кожного з них побудована блок-схема.

Визначено архітектуру програмного забезпечення – тришарова. Шар доступу до даних містить реалізації патернів Репозиторій і Одиниця Роботи щоб досягти високого рівня абстракцій і гарантії узгодженості даних у БД після транзакції. Шар презентацій реалізує патерн MVC і містить контролери веб застосунку. Архітектура додатку наведена на рисунку 2.10. Окрім цього додаток також слідує принципам REST, що дозволить створювати інтеграції зі сторонніми сервісами.

Під час конструювання програмного забезпечення було створено ER діаграму сутностей моделі даних (Рисунок 2.11), а також описано кожен атрибут кожної сутності. Також побудовано структуру програми, що виражена діаграмою класів. Структурна схему класів наведена в документі КПІ.ІТ-9121.045440.06.99.СС

Проведено аналіз безпеки даних і реалізовано необхідні покращення системи безпеки.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Щоб виконати аналіз якості ПЗ необхідно визначити метрики якості, які найбільше підходять для оцінки розробленого файлообмінника.

Найважливішою складовою такого типу програмного забезпечення є безпека даних. Застосунок має гарантувати відсутність можливості несанкціонованого доступу до файлів або до процесів адміністрування, якщо користувач не має на це права.

При великому часі відгуку програмним забезпеченням користуватись незручно, тому повинен підтримуватись достатній рівень швидкодії. При роботі з застосунком має бути мінімізований час виконання команд і об’єм пам’яті для ресурсів процесу і для клієнта, і для сервера. Тому проводиться аналіз продуктивності.

Для спрощення підтримки програмного забезпечення має дотримуватись високий рівень якості коду. Тому проводиться оцінка чистоти коду, до якої входять аналіз форматування і наявності антипатернів. Виправлення помилок такого типу дозволить швидше додавати новий функціонал і виправляти некоректний.

Для аналізу якості програмного забезпечення використовувався статичний аналізатор коду сервісу DeepSource[30]. На рисунку 3.1 наведено результати аналізу.

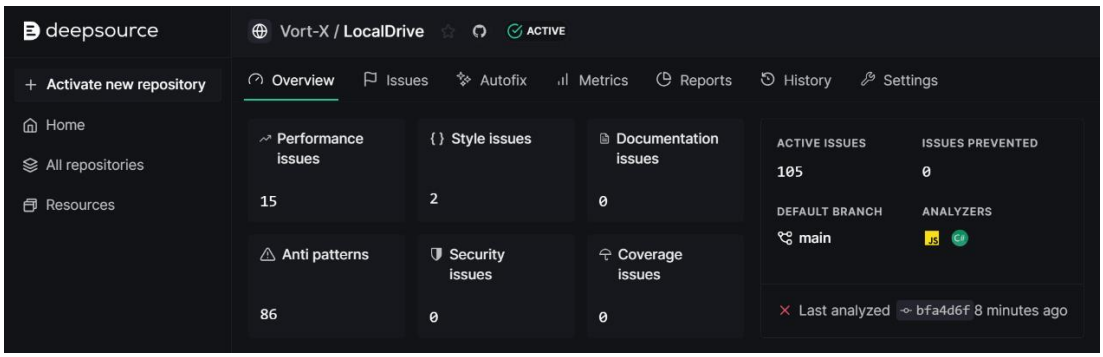


Рисунок 3.1 – Результати статичного аналізу коду

Можна зробити висновок, що програмне забезпечення гарантує найвищий рівень безпеки даних. Не виявлено жодної пов’язаної з цим проблеми.

Виявлені деякі проблеми з продуктивністю програмного забезпечення. Більшість з них викликана неправильним використанням функцій, що надають бібліотеки користувацького інтерфейсу. З боку серверної частини застосунку проблем майже не виявлено.

Виявлено багато порушень чистоти коду. Майже всі стосуються використання антипатернів. Це сильно завадить процесам підтримки програмного забезпечення, тому необхідним є проведення рефакторингу коду. Проблем з форматування майже не виявлено.

На рисунках 3.2 – 3.3 наведено детальний аналіз безпеки даних програмного забезпечення. Оцінка не виявила серйозних порушень.

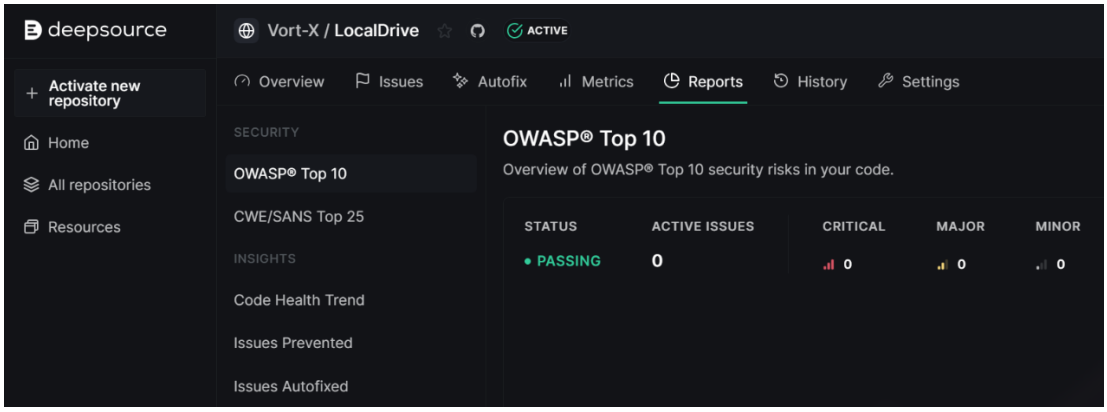


Рисунок 3.2 – Аналіз ризиків порушень безпеки

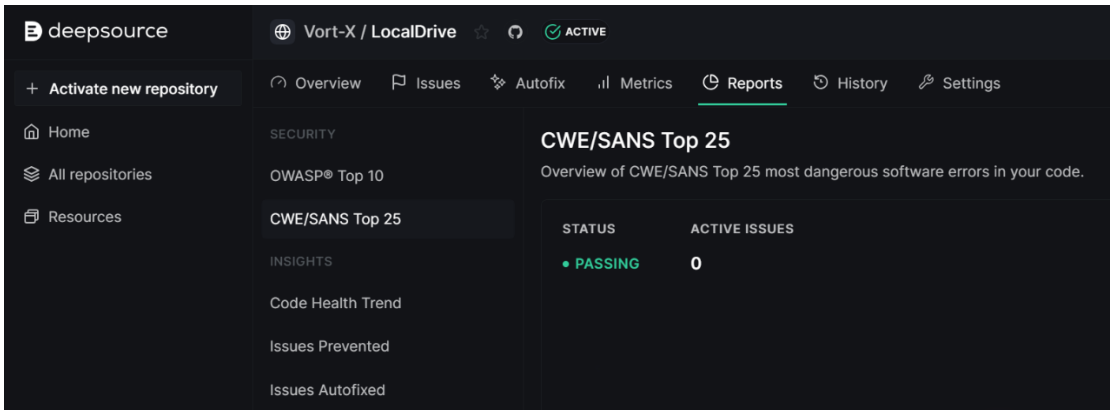


Рисунок 3.3 –Аналіз критичних помилок в коді програми

Також були виконані всі нефункціональні вимоги.

На клієнтській частині програмного забезпечення є контроль введення інформації та захист від некоректних дій користувача. На формах створення і оновлення облікового запису присутня перевірка введених даних. Застосунок забороняє виконувати дії якщо вони виконуються над недопустимими даними і повідомляє про це користувача. В описі контрольного прикладу наведені рисунки, на яких видно повідомлення про помилку.

На серверній частині програмного забезпечення є перевірка отриманих запитів. Якщо вони містять некоректні дані, то сервер їх відхиляє. Перевірка була здійснена з надсиланням HTTP-запитів на API програмного забезпечення, але в розділі не розглядається.

Цілісність інформації в базі даних здобувається завдяки встановленню обмежень в коді і властивостями обраної СУБД. Тому програмно неможливо додати запис в БД, який би порушував її цілісність. Перевірка була здійснена з надсиланням HTTP-запитів на API програмного забезпечення, але в розділі не розглядається.

Забезпечено сумісність серверної частини програмного забезпечення з операційними системами сімейств Windows NT, Linux Debian, MacOS. Код програми є платформо-незалежним. Створено випуски програмного забезпечення для перелічених сімейств, проведено тести їх працездатності.

3.2 Опис процесів тестування

Мануальне тестування було виконано через користувацький інтерфейс, наданий клієнтською частиною додатку. Тести засновані на функціональних вимогах програмного забезпечення, а для підтвердження стійкості системи деякі з них проведено з використанням некоректного вводу. Опис тестів наведено у таблицях 3.1 – 3.17.

Таблиця 3.1 – Тест 1.1

Тест	Переглянути профілі
Модуль	Переглянути профілі
Номер тесту	1.1

Продовження таблиці 3.1

Початковий стан системи	Користувач, авторизований як власник, не знаходиться на сторінці профілів
Вхідні дані	
Опис проведення тесту	Користувач переходить на сторінку профілів. Список профілів автоматично завантажується з серверу.
Очікуваний результат	Користувачу відображується список профілів.
Фактичний результат	Користувачу відображується список профілів.

Таблиця 3.2 – Тест 1.2

Тест	Створити профіль з неунікальним логіном
Модуль	Створити профіль
Номер тесту	1.2
Початковий стан системи	Користувач, авторизований як власник, знаходиться на сторінці профілів
Вхідні дані	Логін, пароль
Опис проведення тесту	Користувач натискає на кнопку створення профілю. У новому вікні він вводить неунікальний логін і пароль і натискає на кнопку збереження.
Очікуваний результат	Користувач отримує повідомлення про те, що профіль вже існує.
Фактичний результат	Користувач отримує повідомлення про те, що профіль вже існує.

Таблиця 3.3 – Тест 1.3

Тест	Створити профіль з некоректними даними
Модуль	Створити профіль
Номер тесту	1.3
Початковий стан системи	Користувач, авторизований як власник, знаходиться на сторінці профілів
Вхідні дані	Логін, пароль
Опис проведення тесту	Користувач натискає на кнопку створення профілю. У новому вікні він вводить некоректні логін і пароль. Логін має бути пустим, містити менше чотирьох символів або недопустимі знаки. Пароль має бути пустим або містити менше шести символів. Користувач натискає на кнопку збереження.
Очікуваний результат	Користувач отримує повідомлення про некоректний ввід.

Продовження таблиці 3.3

Фактичний результат	Користувач отримує повідомлення про некоректний ввід.

Таблиця 3.4 – Тест 1.4

Тест	Створити профіль
Модуль	Створити профіль
Номер тесту	1.4
Початковий стан системи	Користувач, авторизований як власник, знаходиться на сторінці профілів
Вхідні дані	Логін, пароль
Опис проведення тесту	Користувач натискає на кнопку створення профілю. У новому вікні він вводить коректні логін і пароль. Логін має бути унікальним. Користувач натискає на кнопку збереження.
Очікуваний результат	Профіль створено і додано в список
Фактичний результат	Профіль створено і додано в список

Таблиця 3.5 – Тест 1.5

Тест	Видалення профілю
Модуль	Видалення профілю
Номер тесту	1.5
Початковий стан системи	Користувач, авторизований як власник, знаходиться на сторінці профілів
Вхідні дані	Список профілів
Опис проведення тесту	Користувач обирає довільний профіль зі списку, відкриває для нього контекстне меню і натискає на кнопку видалення.
Очікуваний результат	Профіль видалено і вилучено зі списку
Фактичний результат	Профіль видалено і вилучено зі списку

Таблиця 3.6 – Тест 1.6

Тест	Присвоїти профілю некоректні облікові дані
Модуль	Редагувати облікові дані профіля
Номер тесту	1.6
Початковий стан системи	Користувач, авторизований як власник, знаходиться на сторінці профілів
Вхідні дані	Логін, пароль

Продовження таблиці 3.6

Опис проведення тесту	Користувач обирає довільний профіль зі списку, відкриває для нього контекстне меню і натискає на кнопку редагування. У новому вікні він вводить некоректні логін і пароль. Логін має бути пустим, містити менше чотирьох символів або недопустимі знаки. Пароль має бути пустим або містити менше шести символів. Користувач натискає на кнопку збереження.
Очікуваний результат	Користувач отримує повідомлення про некоректний ввід.
Фактичний результат	Користувач отримує повідомлення про некоректний ввід.

Таблиця 3.7 – Тест 1.7

Тест	Присвоїти профілю коректні облікові дані
Модуль	Редагувати облікові дані профіля
Номер тесту	1.7
Початковий стан системи	Користувач, авторизований як власник, знаходиться на сторінці профілів
Вхідні дані	Логін, пароль
Опис проведення тесту	Користувач обирає довільний профіль зі списку, відкриває для нього контекстне меню і натискає на кнопку редагування. У новому вікні він вводить коректні логін і пароль. Логін має бути унікальним. Користувач натискає на кнопку збереження.
Очікуваний результат	Профіль оновлено. Зміни відображені в списку
Фактичний результат	Профіль оновлено. Зміни відображені в списку

Таблиця 3.8 – Тест 1.8

Тест	Надати профілю доступ до файлу
Модуль	Редагувати доступ профіля до файлу
Номер тесту	1.8
Початковий стан системи	Користувач, авторизований як власник, знаходиться на сторінці профілів
Вхідні дані	
Опис проведення тесту	Користувач обирає довільний профіль зі списку, відкриває для нього контекстне меню і натискає на кнопку редагування доступу. У новому вікні він обирає довільний файл, доступ до якого не надано. Користувач натискає на відповідний прапорець.
Очікуваний результат	Обраному профілю надано доступ до файлу

Продовження таблиці 3.8

Фактичний результат	Обраному профілю надано доступ до файлу

Таблиця 3.9 – Тест 1.9

Тест	Обмежити профілю доступ до файлу
Модуль	Редагувати доступ профіля до файлу
Номер тесту	1.9
Початковий стан системи	Користувач, авторизований як власник, знаходиться на сторінці профілів
Вхідні дані	
Опис проведення тесту	Користувач обирає довільний профіль зі списку, відкриває для нього контекстне меню і натискає на кнопку редагування доступу. У новому вікні він обирає довільний файл, доступ до якого надано. Користувач натискає на відповідний прапорець.
Очікуваний результат	Обраному профілю обмежено доступ до файлу
Фактичний результат	Обраному профілю обмежено доступ до файлу

Таблиця 3.10 – Тест 1.10

Тест	Переглянути поширені файли
Модуль	Переглянути поширені файли
Номер тесту	1.10
Початковий стан системи	Користувач, авторизований як власник, не знаходиться на сторінці файлів
Вхідні дані	
Опис проведення тесту	Користувач переходить на сторінку файлів. Список поширених файлів автоматично завантажується з серверу.
Очікуваний результат	Користувачу відображується список файлів.
Фактичний результат	Користувачу відображується список файлів.

Таблиця 3.11 – Тест 1.11

Тест	Додати поширений файл
Модуль	Додати файл до поширених
Номер тесту	1.11
Початковий стан системи	Користувач, авторизований як власник, знаходиться на сторінці файлів
Вхідні дані	Файл

Продовження таблиці 3.11

Опис проведення тесту	Користувач натискає на кнопку додавання файлу. У вікні файлового менеджера він обирає файл з неунікальним ім’ям і натискає на кнопку збереження.
Очікуваний результат	Користувач отримує повідомлення про те, що файл вже існує.
Фактичний результат	Користувач отримує повідомлення про те, що файл вже існує.

Таблиця 3.12 – Тест 1.12

Тест	Додати непоширений файл
Модуль	Додати файл до поширених
Номер тесту	1.12
Початковий стан системи	Користувач, авторизований як власник, знаходиться на сторінці файлів
Вхідні дані	
Опис проведення тесту	Користувач натискає на кнопку додавання файлу. У вікні файлового менеджера він обирає файл, який ще не було поширено, і натискає на кнопку збереження.
Очікуваний результат	Файл завантажено на сервер і додано в список
Фактичний результат	Файл завантажено на сервер і додано в список

Таблиця 3.13 – Тест 1.13

Тест	Вилучити файл з поширених
Модуль	Вилучити файл з поширених
Номер тесту	1.13
Початковий стан системи	Користувач, авторизований як власник, знаходиться на сторінці файлів
Вхідні дані	
Опис проведення тесту	Користувач обирає довільний файл зі списку і натискає на кнопку видалення.
Очікуваний результат	Файл видалено і вилучено зі списку
Фактичний результат	Файл видалено і вилучено зі списку

Таблиця 3.14 – Тест 1.14

Тест	Авторизація користувача з некоректними обліковими даними
Модуль	Авторизація користувача
Номер тесту	1.14

Продовження таблиці 3.14

Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні дані	Логін, пароль
Опис проведення тесту	Користувач вводить логін і пароль, які не відповідають жодному профілю в системі, у відповідні поля і натискає кнопку входу.
Очікуваний результат	Користувач отримує повідомлення про некоректні облікові дані
Фактичний результат	Користувач отримує повідомлення про некоректні облікові дані

Таблиця 3.15 – Тест 1.15

Тест	Авторизація користувача з коректними обліковими даними
Модуль	Авторизація користувача
Номер тесту	1.15
Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні дані	Логін, пароль
Опис проведення тесту	Користувач вводить логін і пароль, які відповідають одному профілю в системі, у відповідні поля і натискає кнопку входу.
Очікуваний результат	Користувач авторизувався
Фактичний результат	Користувач авторизувався

Таблиця 3.16 – Тест 1.16

Тест	Переглянути доступні файли
Модуль	Переглянути доступні файли
Номер тесту	1.16
Початковий стан системи	Користувач авторизований як зовнішній
Вхідні дані	
Опис проведення тесту	Після успішної авторизації користувач опиняється на сторінці доступних файлів, де автоматично завантажується список
Очікуваний результат	Користувачу відображується список доступних файлів.
Фактичний результат	Користувачу відображується список доступних файлів.

Таблиця 3.17 – Тест 1.17

Тест	Завантажити файл
Модуль	Завантажити файл
Номер тесту	1.17
Початковий стан системи	Користувач, авторизований як зовнішній, знаходиться на сторінці доступних файлів
Вхідні дані	
Опис проведення тесту	Користувач обирає довільний файл зі списку і натискає на кнопку завантаження.
Очікуваний результат	Файл завантажується до файлової системи користувача
Фактичний результат	Файл завантажується до файлової системи користувача

Фактичні результати тестування відповідають очікуваним, отже система працює правильно.

3.3 Опис контрольного прикладу

Для контрольного прикладу було вирішено створити новий профіль.

Спершу потрібно впевнитись, що в системі вже знаходиться щонайменше один профіль. Для цього список профілів заповнено декількома обліковими записам. Це потрібно щоб в подальших кроках впевнитись, що система не дозволяє стан, при якому одночасно існують два профілі з однаковими логінами. Цей список можна побачити на рисунку 3.4.

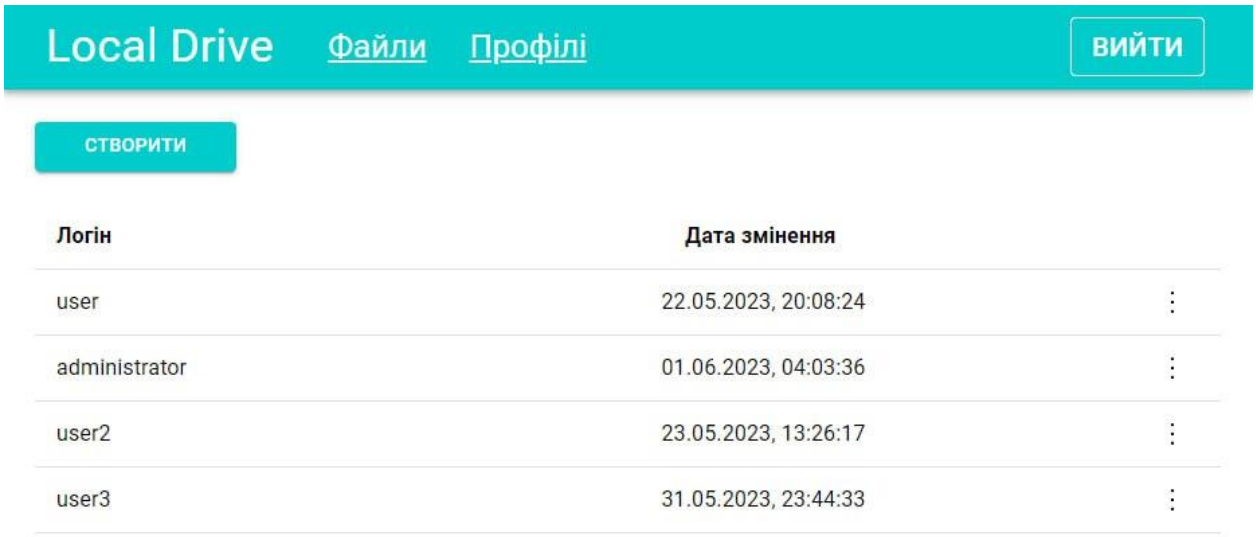


Рисунок 3.4 – Список профілів

Першим кроком є натискання на кнопку “Створити”. Після цього з’являється діалогове вікно для створення профілю. Його наведено на рисунку 3.5.

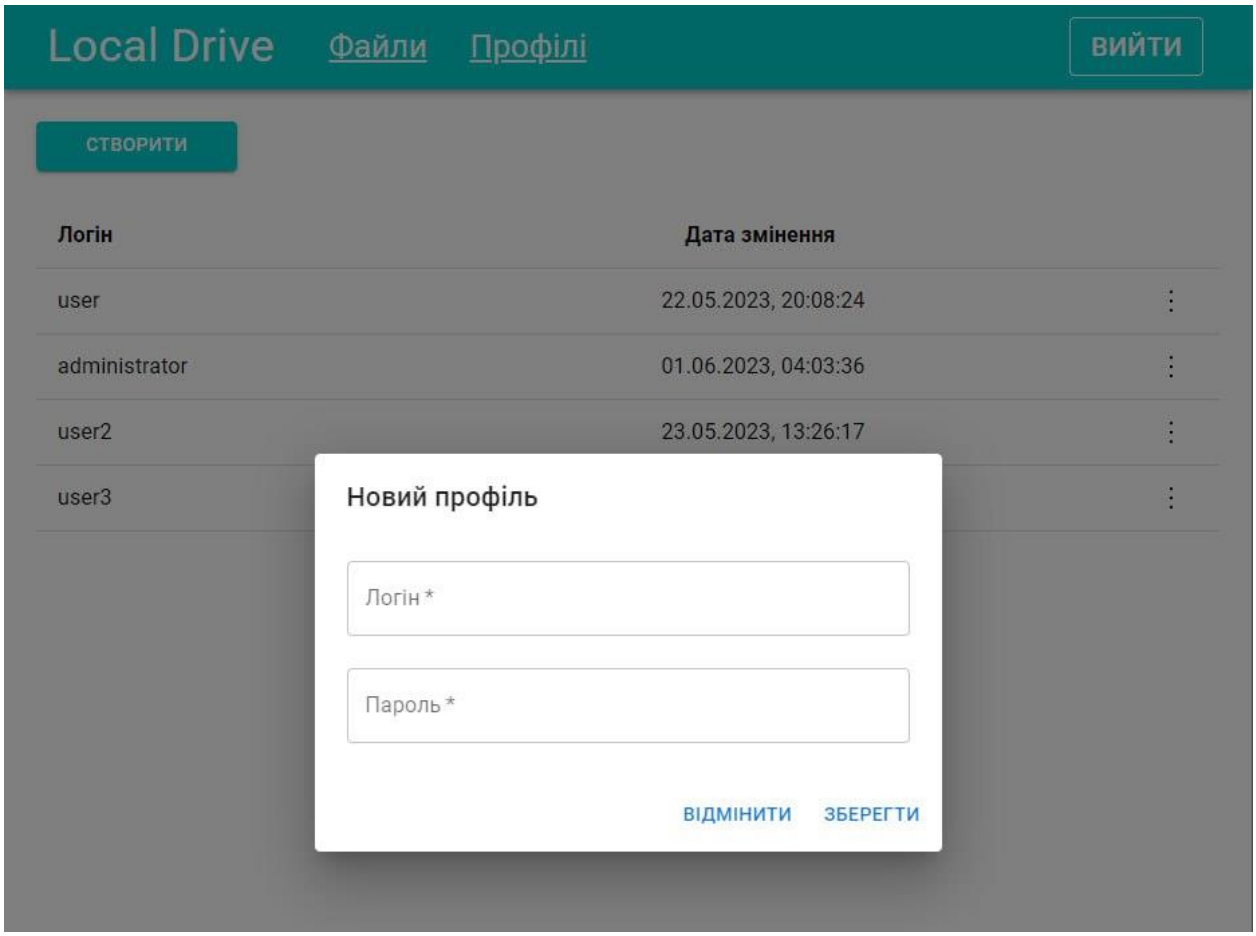


Рисунок 3.5 – Вікно створення профілю

Наступним кроком є ввід облікових даних у відповідні текстові поля. Користувацький інтерфейс дозволяє ввід будь-яких даних текстового формату. Пароль при цьому приховується.

Останнім кроком є натискання кнопки “Зберегти”. Спочатку відбувається перевірка введених даних, і у випадку невдачі користувач отримує повідомлення про це. Результат перевірки залежить від змісту даних. В таблиці 3.18 детально описано дії, що виконуються програмою при всіх можливих варіантах вводу.

Таблиця 3.18 – Результати перевірки вводу

	Пароль пустий	Довжина паролю менше 4 символів	Пароль допустимий
Логін пустий	Повідомлення про пустий логін; повідомлення про пустий пароль	Повідомлення про пустий логін; повідомлення про короткий пароль	Повідомлення про пустий логін
Довжина логіну менше 4 символів	Повідомлення про короткий логін; повідомлення про пустий пароль	Повідомлення про короткий логін; повідомлення про короткий пароль	Повідомлення про короткий логін
Логін містить недопустимі символи	Повідомлення про перший недопустимий символ у логіні; повідомлення про пустий пароль	Повідомлення про перший недопустимий символ у логіні; повідомлення про короткий пароль	Повідомлення про перший недопустимий символ у логіні
Профіль з введеним логіном вже існує	Повідомлення про існування профілю з введеним логіном; повідомлення про пустий пароль	Повідомлення про існування профілю з введеним логіном; повідомлення про короткий пароль	Повідомлення про існування профілю з введеним логіном
Логін допустимий	Повідомлення про пустий пароль	Повідомлення про короткий пароль	Перевірка пройдена успішно

Оскільки перевірки логіну і паролю відбуваються окремо, то необов’язково тестувати кожен варіант вводу. Для прикладу було обрано декілька варіантів, результати яких зображено на рисунках 3.6 – 3.10.

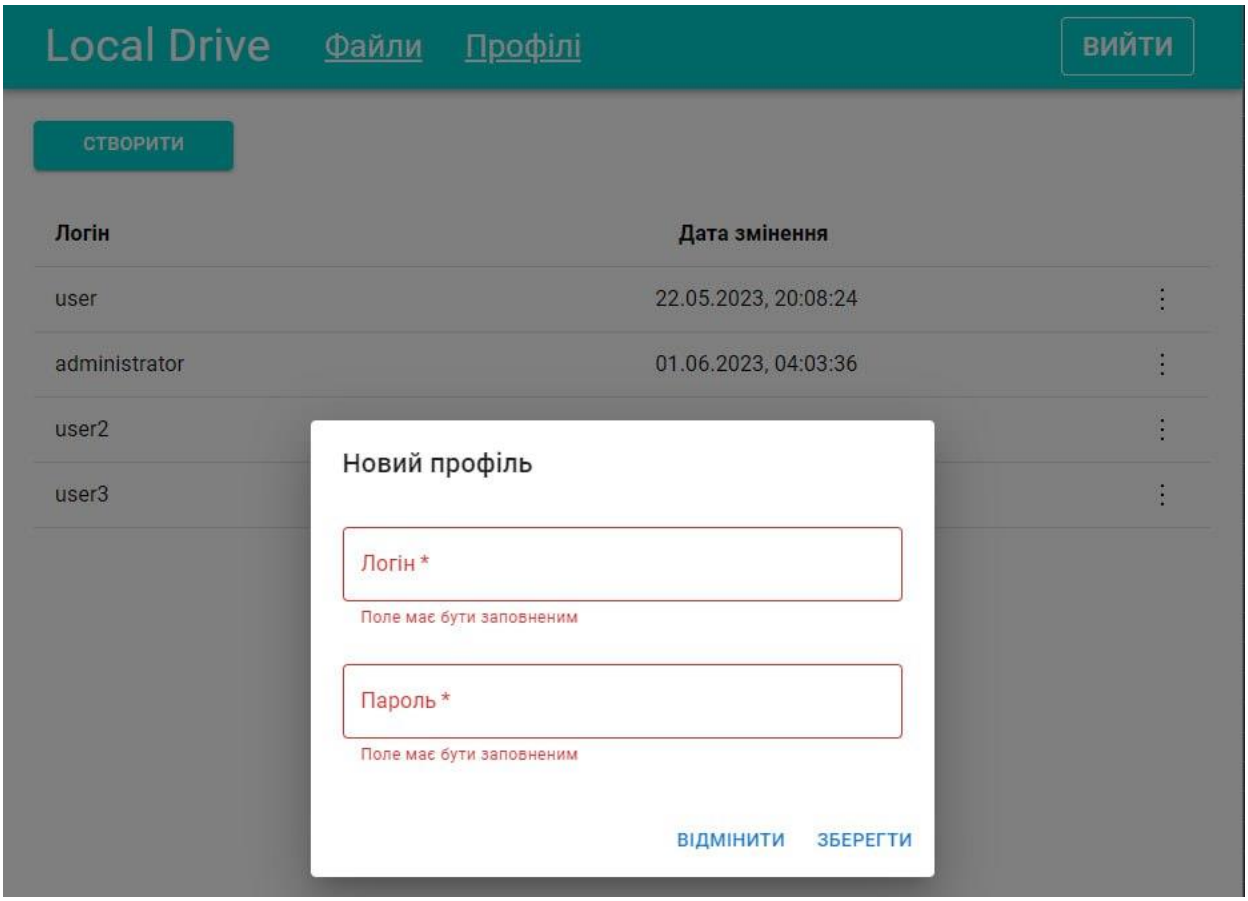


Рисунок 3.6 – Пустий логін, пустий пароль

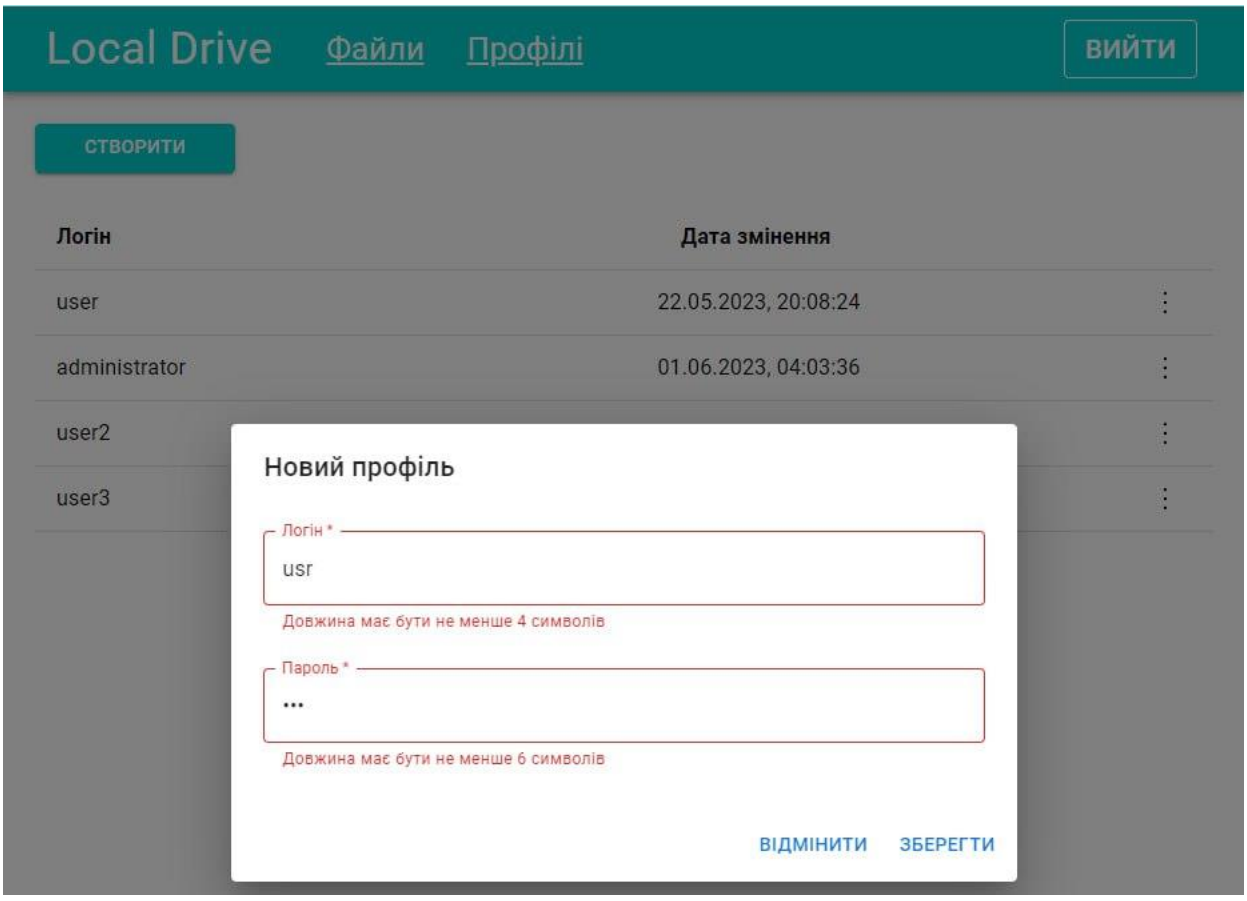


Рисунок 3.7 – Короткий логін, короткий пароль

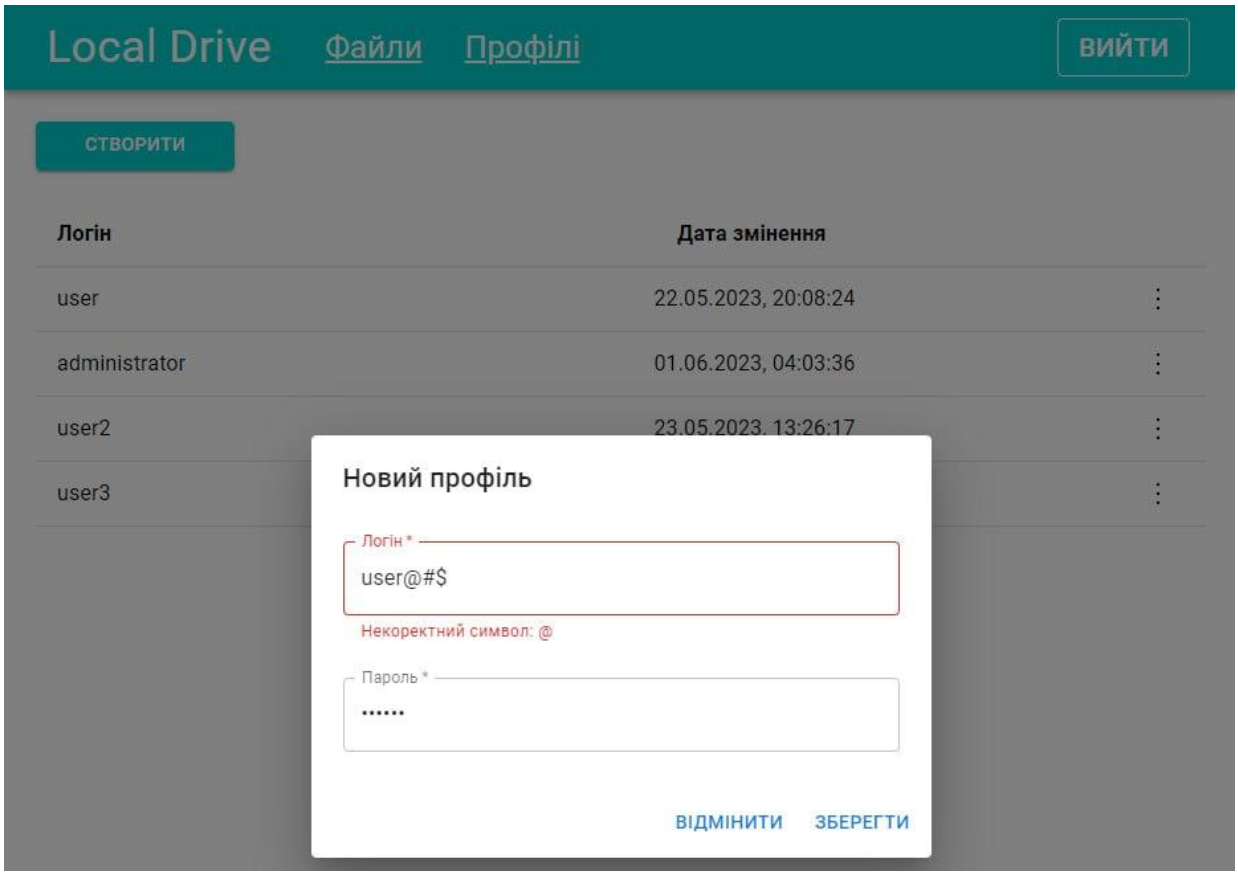


Рисунок 3.8 – Логін з недопустимими символами, допустимий пароль

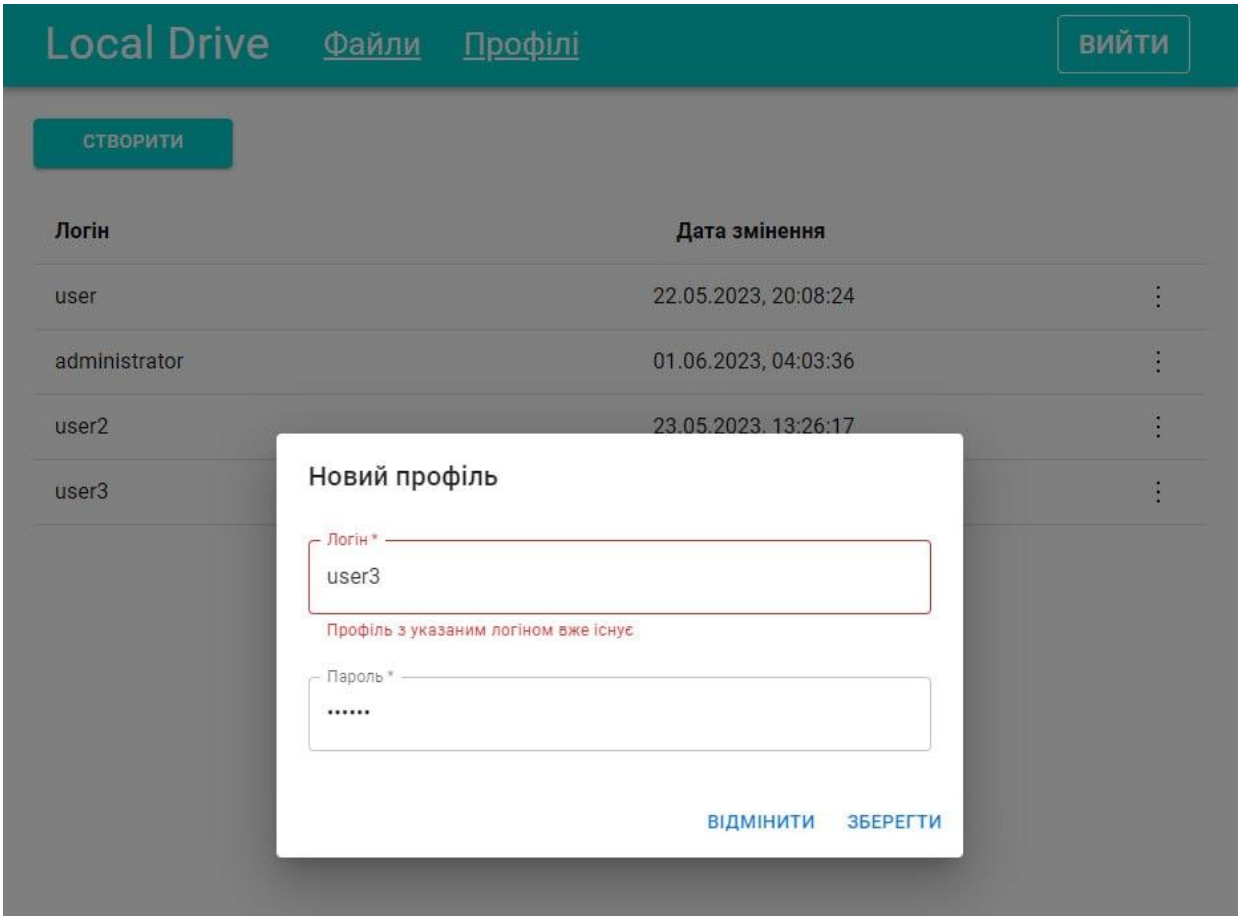


Рисунок 3.9 – Профіль з ідентичним логіном, допустимий пароль

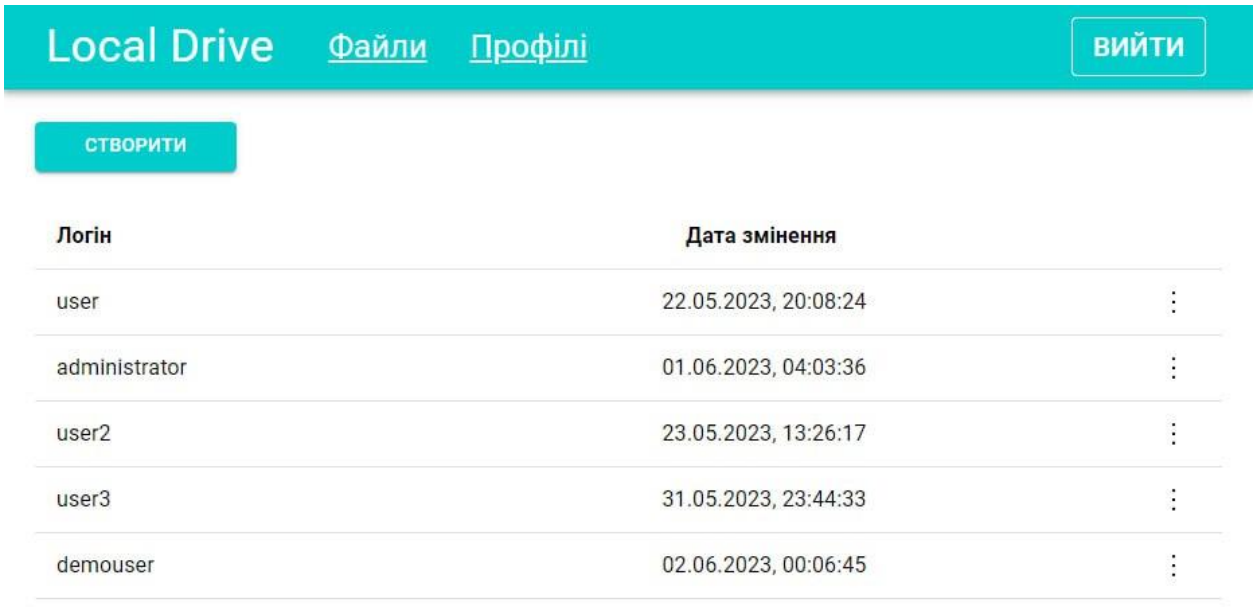


Рисунок 3.10 –Допустимий логін, допустимий пароль

Як видно з рисунку 3.10, профіль з’явився у списку, що свідчить про успішні перевірки вводу. Користувач не отримує повідомлення про успішну операцію, і вікно закривається одразу після надсилання HTTP-запиту.

На цьому тест завершено. Фактична поведінка відповідає очікуваній, отже реалізація цієї функціональної вимоги є успішною.

Висновки до розділу

У даному розділі проведено аналіз якості розробленого програмного забезпечення. Для цього було визначено метрики якості, за якими має оцінюватись файлообмінник, і проведено статичний аналіз коду.

Через користувацький інтерфейс було перевірено правильність реалізації функціональних вимог. У таблицях 3.1 – 3.17 наведено мануальні тести, за якими визначалась якість розробленого додатку. Тестування виявило, що фактична поведінка відповідає очікуваній.

Описано контрольний приклад, в якому покроково для усіх можливих вхідних даних перевірено поведінку програми при створенні нового профілю. Якщо вхідні логін і пароль допустимі, то профіль додається в список, інакше – користувач отримує повідомлення з помилкою вводу. Тестування виявило, що фактична поведінка системи при створені профілю з

усіма можливими вхідними даними відповідає очікуваній, яка надана в таблиці 3.18. Можна зробити висновок, що ця функціональність реалізована правильно.

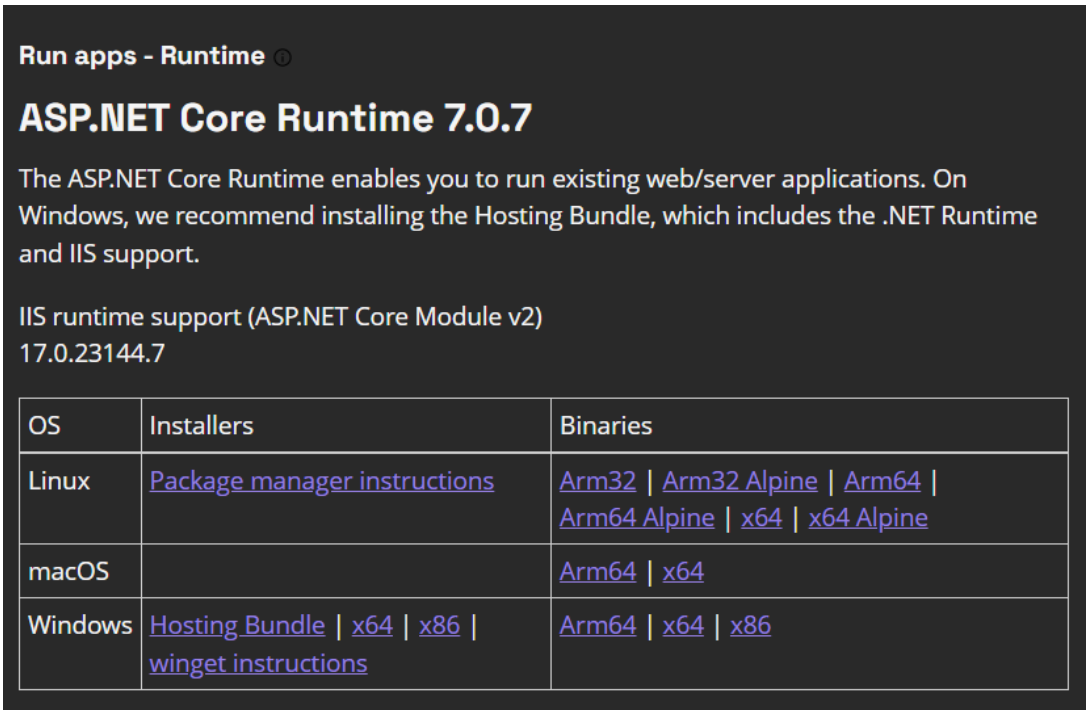
4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Згідно вимог до додатку для обміну файлів програма має працювати на локальному середовищі користувача. Він може завантажити готові збірки, опубліковані на платформі GitHub[31]. Налаштування і розгортанню додатку. складається з п’яти кроків.

Першим кроком є встановлення необхідних утиліт і платформ.

З офіційного сайту Microsoft[32] необхідно завантажити та інсталювати ASP.NET Core Runtime 7 для своєї платформи.



Run apps - Runtime

ASP.NET Core Runtime 7.0.7

The ASP.NET Core Runtime enables you to run existing web/server applications. On Windows, we recommend installing the Hosting Bundle, which includes the .NET Runtime and IIS support.

IIS runtime support (ASP.NET Core Module v2)
17.0.23144.7

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 | Arm32 Alpine | Arm64 | Arm64 Alpine | x64 | x64 Alpine
macOS		Arm64 | x64
Windows	Hosting Bundle | x64 | x86 | winget instructions	Arm64 | x64 | x86

Рисунок 4.1 – Посилання на ASP.NET Core Runtime

SQL Server завантажується також з офіційного сайту[33]. Для роботи застосунку достатньо буде версії Express/

Рисунок 4.2 – Посилання на SQL Server

Node.js завантажується з офіційного сайту[34]. Рекомендується версія LTS.

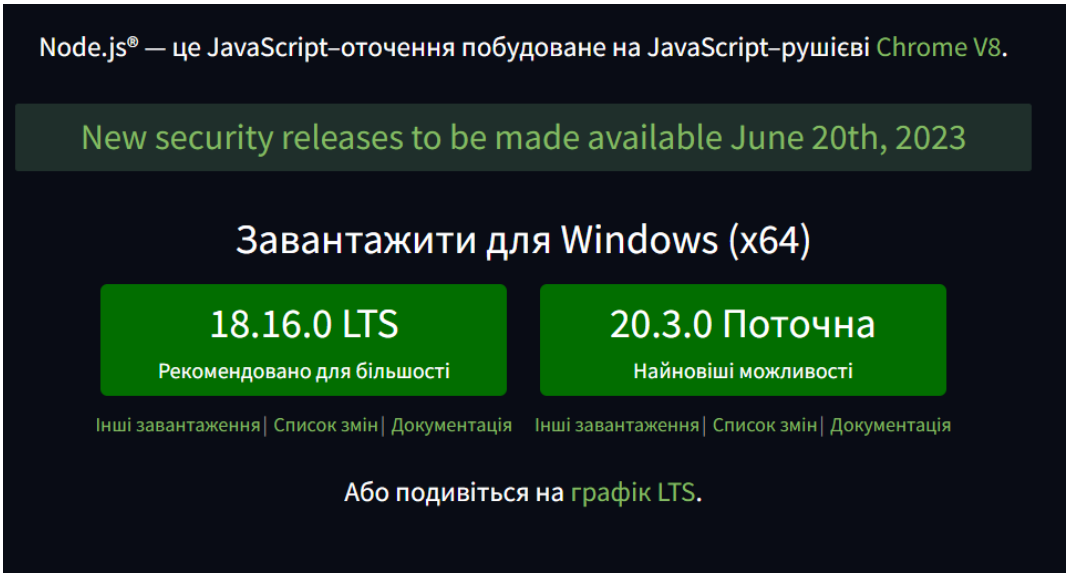


Рисунок 4.3 – Посилання на Node.js

Другий крок – налаштування додатку. До цього відноситься редагування файлу appsettings.json серверної частини програмного забезпечення, де вказуються секретний ключ, коренева директорія для збереження файлів і рядок з'єднання з БД.

Рисунок 4.4 – Файл appsettings.json

Третій крок – запуск серверної частини додатку через виконуваний файл LocalDrive.WebServer.exe.

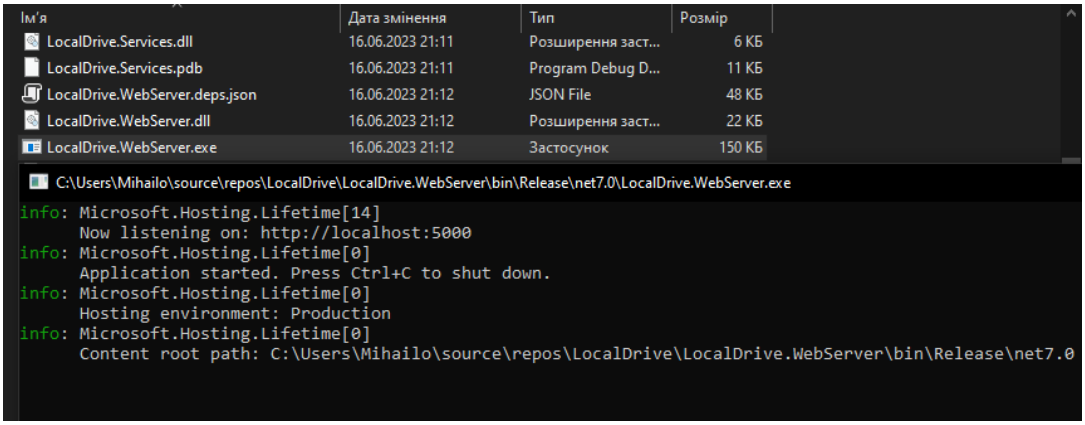


Рисунок 4.5 – Виконуваний файл LocalDrive.WebServer.exe

На четвертому кроці користувач має з допомогою npm завантажити пакети для клієнтської частини додатку і запустити її. Для цього в консолі потрібно виконати команди “npm i” та “npm start”.

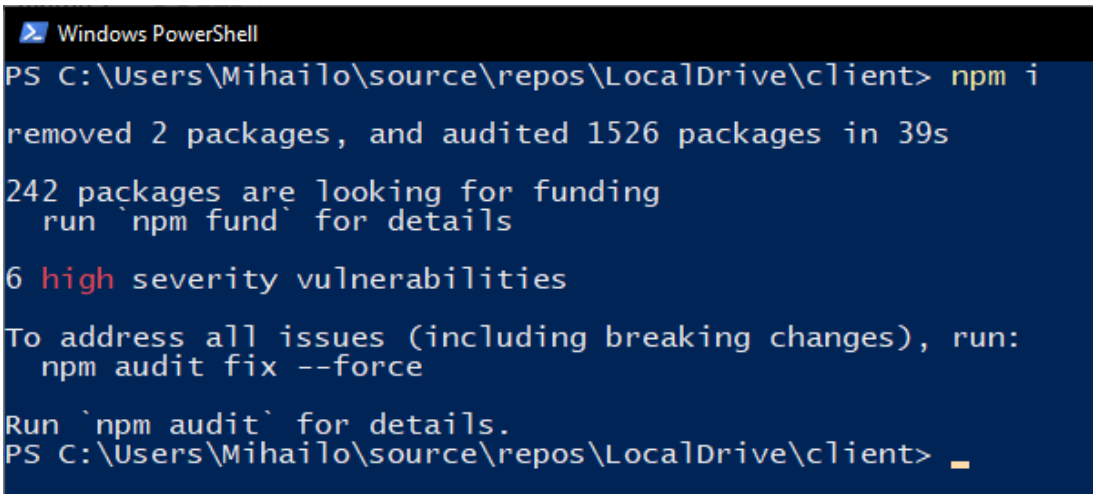


Рисунок 4.6 – Результат виконання команди npm i

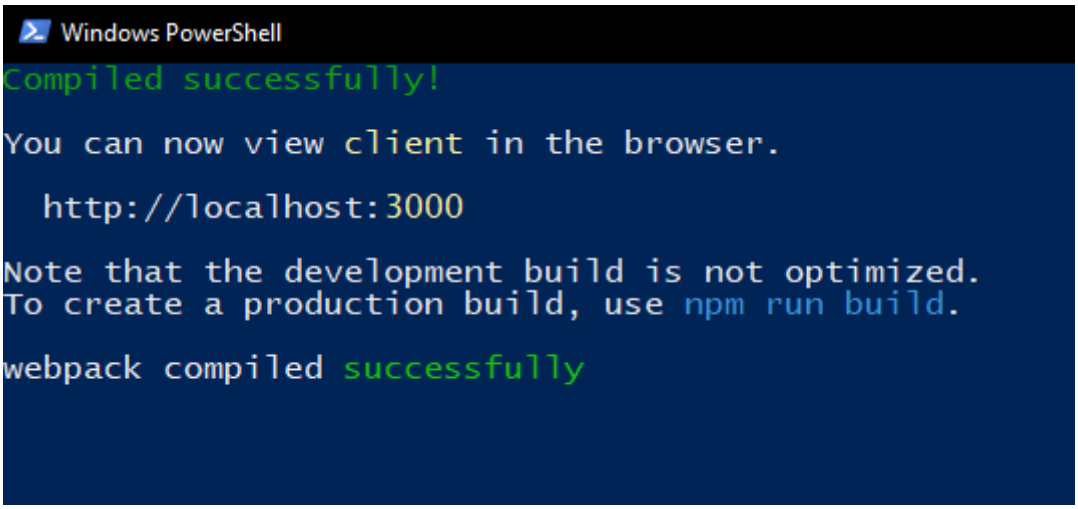


Рисунок 4.7 – Результат виконання команди npm start

П’ятий крок – переадресація портів. Щоб на файлообмінник могли надходити запити ззовні потрібно налаштувати мережу на перенаправлення запитів до обох частин додатків. Для кожної моделі роутера цей процес може відрізнятись, тому для виконання подальших дій варто звернутись до виробника.

4.2 Підтримка програмного забезпечення

Планується довгострокова підтримка програмного забезпечення. Кожного циклу оновлення буде відбуватися розширення функціоналу, виправлення помилок або покращення користувацького інтерфейсу. Програмне забезпечення має відкритий код, тому до підтримки зможуть долучитися сторонні розробники. Після огляду запропонованих ними змін, новий код може бути включено в наступні випуски.

Створення нового випуску відбувається після додавання змін з робочої гілки у гілку main. Після цього починається процес збірки проектів клієнтської і серверної частин додатку для операційних систем Windows, Linux Debian і MacOS. Для кожного сімейства ОС репозиторій надає окремий випуск. Збірки додаються у відповідний розділ під новим тегом – наступною версією.

Висновки до розділу

У цьому розділі описано процеси розгортання і підтримки програмного забезпечення.

На платформі GitHub опубліковано збірки для різних сімейств ОС. Описано процес встановлення утиліт і платформ для роботи додатку, налаштування і запуск та переадресацію портів.

Визначено процес підтримки програмного забезпечення. Він передбачає оновлення сирцевого коду, а саме розширення функціоналу, виправлення помилок або покращення користувацького інтерфейсу. До цього можуть долучитись сторонні розробники.

Описано процес створення нових випусків програмного забезпечення. Він передбачає оновлення гілки репозиторію і збірка нових версій додатку.

ВИСНОВКИ

Результатом роботи над дипломним проєктом стало програмне забезпечення. Воно дозволяє поширювати файли з локального середовища зовнішнім користувачам, що є його унікальною особливістю, яка відрізняє його від аналогів. Програмне забезпечення надає достатньо функціональних можливостей для виконання поставленої мети, зручне в користуванні і дотримується вимог до безпеки і швидкодії. Це забезпечується використанням сучасних технологій розробки ПЗ і відомих архітектурних рішень.

Розроблене програмне забезпечення може бути використано у вирішенні більшості задач з обміну і збереження файлів. Серед них є такі як обмін файлами між комп'ютерами ІТ-фірми, віддалене збереження і поширення бухгалтерських документів, створення спільного файлового простору для команди дизайнерів. Також це рішення підходить і для окремих користувачів, які хочуть мати доступ до своїх файлів з інших пристроїв, але для них обсяг даних, що надають хмарні сервіси, недостатній.

Програмне забезпечення, що було розроблене, є безкоштовним і не накладає обмежень у використанні. Воно підходить для окремих користувачів і невеликих підприємств, які не можуть фінансово забезпечити використання аналогічних рішень, а регулярна підтримка і дотримання високих стандартів якості гарантує подальший розвиток продукту.

В процесі роботи над проєктом було виконано всі поставлені в дипломному проєктуванні задачі.

Досліджено процеси мережевого обміну файлами і питання з безпеки, що дозволило визначити найкращі інструменти, алгоритмічні і технічні рішення, які використовувалися в проектуванні і реалізації функціональних вимог до програмного забезпечення.

Програмне забезпечення змодельоване і розроблене згідно вказаного функціоналу і вимог. В результаті отримано якісний і ефективний код додатку, його легко розширювати і виправляти.

Створено збірки проекту для різних ОС.

Описано процес встановлення програмного забезпечення на різні ОС. Розгортка і налаштування описані у покроковій інструкції.

Опубліковано збірки у відкритому доступі на платформі GitHub.

В якості середовища розробки обрано Visual Studio і Visual Studio Code. Вказані IDE надають широкий набір інструментів для розробки веб-застосунків.

В якості БД використано SQL Server – реляційна СУБД, що підтримується утилітами Microsoft, які використовувалися при розробці додатку.

В якості інструментів розробки клієнтської частини додатку обрано бібліотеки React.js і Material UI для створення зручного користувацького інтерфейсу з підтримкою часткового оновлення.

В якості інструментів розробки серверної частини додатку обрано платформу ASP.NET Core. Вона надає можливість ефективно створювати веб-застосунки з використанням шаблону MVC і підтримкою принципів REST API.

Проведено статичний аналіз коду програмного забезпечення, який виявив деякі проблеми з продуктивністю і чистотою коду, але підвердив повне дотримання вимог з безпеки даних.

Проведено мануальне тестування програмного забезпечення, яке виявило, що всі функціональні вимоги були реалізовані правильно.

Визначено процеси розгортання і підтримки програмного забезпечення. Написана інструкція по встановленню, налаштуванню і запуску програми.

Розроблене програмне забезпечення реалізує основні вимоги до нього: обмін файлами, кросплатформність, локальне розгортання, безпека даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Єніна І.І., Мороз А.С. Перспективи застосування хмарних технологій для зберігання інформації. *Наукові записки*. 2017. Вип. 21, С. 81–83.

2) Judiths H. Cloud Computing for Dummies. JOHN WILEY & Sons, 2019. 320 с.

3) Provos N., and Mazières D. A Future Adaptable Password Scheme. Wayback Machine. URL: https://web.archive.org/web/20180615201340/http://www.openbsd.org/papers/bcrypt-paper.pdf (дата звернення 05.04.2023).

4) Percival C. Stronger Key Derivation Via Sequential Memory-Hard Functions. URL: http://www.tarsnap.com/scrypt/scrypt.pdf (дата звернення 05.04.2023).

5) Biryukov A., Dinu D., and Khovratovich D. Argon2: the memory-hard function for password hashing and other applications. URL: https://www.cryptolux.org/images/0/0d/Argon2.pdf (дата звернення 05.04.2023).

6) Martin R. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston, Prentice Hall, 2018. 432 с.

7) Common web application architectures // Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures (дата звернення 01.04.2023).

8) Berson A. Client/Server Architecture. New York, McGraw-Hill, 1996. 569 с.

9) Masse M. REST API Design Rulebook. Beijing: O’Reilly, 2012. 94 с.

10) Aroraa G., Chilberto J. Hands-On Design Patterns with C# and .Net Core: Write Clean and Maintainable Code by Using Reusable Solutions to Common Software Design Problems. Birmingham, UK: Packt Publishing, 2019. 410 с.

11) Millett S., Tune N. Patterns Principles and Practices of Domain-Driven Design. Indianapolis, Indiana: Wrox, 2015. 800 с.

12) ACID properties of transactions // IBM Documentation. URL: https://www.ibm.com/docs/en/cics-ts/5.4?topic=processing-acid-properties-transactions (дата звернення 03.04.2023).

13) Lombardi A. Websocket: Lightweight Client-Server Communications. First ed. Sebastopol, CA: O'Reilly Media, 2015. 144 с.

14) Frank U., Maier P., Bock A. Low Code Platforms: Promises, Concepts and Prospects. A Comparative Study of Ten Systems. Institut für Informatik und Wirtschaftsinformatik (ICB), Universität Duisburg-Essen, Essen, 2021. 170 с.

15) Dresher T., Zuker A., Friedman S. Hands-On Full Stack Web Development with Asp.net Core: Learn End-To-End Web Development with Leading Frontend Frameworks Such As Angular React and Vue. Birmingham UK: Packt Publishing, 2018.

16) Chaudhary H. Handy On Relational Database Management System RDBMS-1000+ MCQ. STCD Company, 2018. 200 с.

17) Sadalage P., Fowler M. NoSQL Distilled a Brief Guide to the Emerging World of Polyglot Persistence. Hoboken: Pearson Education Limited, 2012. 192 с.

18) React. URL: https://react.dev/ (дата звернення 02.04.2023).

19) Home v6.13.0 // React Router. URL: https://reactrouter.com/en/main (дата звернення 02.04.2023).

20) MUI: The React component library you always wanted. URL: https://mui.com/ (дата звернення 02.04.2023).

21) ASP.NET // Open-source web framework for .NET. URL: https://www.asp.net/ (дата звернення 02.04.2023).

22) Overview of Entity Framework Core - EF Core // Microsoft Learn. URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення 02.04.2023).

23) Visual Studio: IDE and Code Editor for Software Developers and Teams. URL: https://visualstudio.microsoft.com/ (дата звернення 02.04.2023).

24) SQL Server 2022 // Microsoft. URL: https://www.microsoft.com/en-us/sql-server/sql-server-2022 (дата звернення 02.04.2023).

25) Платформа для обміну файлами й особисте хмарне сховище // Google. URL: https://www.google.com/intl/uk_UA/drive/ (дата звернення 15.03.2023).

26) Особисте хмарне сховище // Microsoft OneDrive. URL: https://www.microsoft.com/uk-ua/microsoft-365/onedrive/online-cloud-storage (дата звернення 15.03.2023).

27) Dropbox.com. URL: https://www.dropbox.com/ (дата звернення 15.03.2023).

28) Smith J. P. Entity Framework Core in Action. O'REILLY MEDIA, 2021. 624 c.

29) Fogie S., Grossman J., Hansen R., Rager A., Petkov P.D. XSS Attacks: Cross Site Scripting Exploits and Defense. Burlington: Elseveir Science, 2014. 480 c.

30) DeepSource: The Code Health Platform. URL: https://deepsource.com/ (дата звернення 13.05.2023).

31) GitHub - Vort-X/LocalDrive. URL: https://github.com/Vort-X/LocalDrive (дата звернення 13.05.2023).

32) Download .NET 7.0 (Linux, macOS and Windows). URL: https://dotnet.microsoft.com/en-us/download/dotnet/7.0/ (дата звернення 19.03.2023).

33) SQL Server Downloads // Microsoft. URL: https://www.microsoft.com/en-us/sql-server/sql-server-downloads/ (дата звернення 19.03.2023).

34) Node.js. URL: https://nodejs.org/uk/ (дата звернення 19.03.2023).

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

КРОСПЛАТФОРМНИЙ ФАЙЛООБМІННИК ДЛЯ ЛОКАЛЬНИХ СЕРЕДОВИЩ

Текст програми

КПІ.ІТ-9121.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена МАРЧЕНКО

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Михайло ПЛЕСКАЧ

Київ – 2023

Файл FileMetadata.cs

```
namespace LocalDrive.Entities
{
    public class FileMetadata
    {
        public int Id { get; set; }
        public string? Filename { get; set; }
        public long Size { get; set; }
        public DateTime UploadDate { get; set; }
    }
}
```

Файл Profile.cs

```
namespace LocalDrive.Entities
{
    public class Profile
    {
        public int Id { get; set; }
        public string? Login { get; set; }
        public string? PasswordHash { get; set; }
        public DateTime LastModifyDate { get; set; }
        public DateTime LastSignInDate { get; set; }
        public bool IsOwner { get; set; }
    }
}
```

Файл ProfileAccess.cs

```
namespace LocalDrive.Entities
{
    public class ProfileAccess
    {
        public int Id { get; set; }
        public int ProfileId { get; set; }
        public int FileMetadataId { get; set; }
        public DateTime SharedDate { get; set; }
        public DateTime LastDownloadDate { get; set; }
    }
}
```

Файл IRepository.cs

```
namespace LocalDrive.DataAccess
{
    public interface IRepository<TEntity, TKey> where TEntity : class where TKey :
IEquatable<TKey>
    {
        void Create(TEntity entity);
        TEntity? Get(TKey key);
        IEnumerable<TEntity> GetAll();
        void Remove(TKey key);
        void Update(TKey key, Action<TEntity> setFields);
    }
}
```

Файл IUnitOfWork.cs

```
using LocalDrive.Entities;

namespace LocalDrive.DataAccess
{
    public interface IUnitOfWork : IDisposable
    {
        IRepository<FileMetadata, int> FileMetadataRepository { get; }
        IRepository<Profile, int> ProfileRepository { get; }
        IRepository<ProfileAccess, int> ProfileAccessRepository { get; }

        void SaveChanges();
    }
}
```

Файл DefaultRepository.cs

```
namespace LocalDrive.DataAccess
{
    public class DefaultRepository<TEntity, TKey> : IRepository<TEntity, TKey>
        where TEntity : class
        where TKey : IEquatable<TKey>
    {
        protected readonly LocalDriveDbContext context;

        public DefaultRepository(LocalDriveDbContext context)
        {
            this.context = context;
        }

        public void Create(TEntity entity)
        {
            context.Add(entity);
        }

        public TEntity? Get(TKey key)
        {
            return context.Find<TEntity>(key);
        }

        public IEnumerable<TEntity> GetAll()
        {
            return context.Set<TEntity>().AsEnumerable();
        }

        public void Remove(TKey key)
        {
            var entity = Get(key);
            if (entity == null) return;
            context.Remove(entity);
        }

        public void Update(TKey key, Action<TEntity> setFields)
        {
            var entity = Get(key);
            if (entity == null) return;
            setFields(entity);
            context.Update(entity);
        }
    }
}
```

Файл DefaultUnitOfWork.cs

```
using LocalDrive.Entities;

namespace LocalDrive.DataAccess
{
    public class DefaultUnitOfWork : IUnitOfWork
    {
        private readonly LocalDriveDbContext context;
        private bool disposedValue;

        public DefaultUnitOfWork(LocalDriveDbContext context)
        {
            this.context = context;
        }

        public IRepository<FileMetadata, int> FileMetadataRepository => new
DefaultRepository<FileMetadata, int>(context);
        public IRepository<Profile, int> ProfileRepository => new
DefaultRepository<Profile, int>(context);
        public IRepository<ProfileAccess, int> ProfileAccessRepository => new
DefaultRepository<ProfileAccess, int>(context);

        public void SaveChanges()
        {
            context.SaveChanges();
        }

        protected virtual void Dispose(bool disposing)
        {
            if (!disposedValue)
            {
                if (disposing)
                {
                    context.Dispose();
                }
                disposedValue = true;
            }
        }

        public void Dispose()
        {
            Dispose(disposing: true);
            GC.SuppressFinalize(this);
        }
    }
}
```

Файл LocalDriveDbContext.cs

```
using LocalDrive.Entities;
using Microsoft.EntityFrameworkCore;
using System.Diagnostics.CodeAnalysis;

namespace LocalDrive.DataAccess
{
    public class LocalDriveDbContext : DbContext
    {
        public LocalDriveDbContext([NotNull] DbContextOptions<LocalDriveDbContext>
options) : base(options)
        {
            Database.EnsureCreated();
        }
```

```
        public DbSet<FileMetadata> FileMetadata { get; set; }
        public DbSet<Profile> Profiles { get; set; }
        public DbSet<ProfileAccess> ProfileAccesses { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
        }
    }
}
```

Файл IFileService.cs

```
namespace LocalDrive.Services
{
    public interface IFileService
    {
        IEnumerable<FileMetadata> GetSharedFiles(string? login = null);
        FileMetadata? GetFile(int id);
        void UpdateFileLastDownload(int profileId, int fileId);
        bool ShareFile(string path, long size);
        bool UnshareFile(int id);
        IEnumerable<object> GetSharedFilesForUser(int id);
        bool HasAccess(int profileId, int fileId);
        bool GrantAccess(int profileId, int fileId);
        bool RevokeAccess(int profileId, int fileId);
    }
}
```

Файл IPasswordHashService.cs

```
namespace LocalDrive.Services
{
    public interface IPasswordHashService
    {
        string Hash(string password);
        bool CheckPassword(string password, string hash);
    }
}
```

Файл IProfileService.cs

```
using LocalDrive.Entities;

namespace LocalDrive.Services
{
    public interface IProfileService
    {
        IEnumerable<Profile> GetProfiles();
        Profile? GetProfile(int id);
        Profile? GetProfile(string login);
        bool CreateProfile(string login, string password);
        bool DeleteProfile(int id);
        bool UpdateProfile(int id, string login, string password);
        void UpdateProfileLastLogin(int id);
        bool CheckCredentials(string login, string password);
    }
}
```

Файл DefaultFileService.cs

```
using LocalDrive.DataAccess;
using LocalDrive.Entities;

namespace LocalDrive.Services
{
    public class DefaultFileService : IFileService
    {
        private readonly IUnitOfWork unitOfWork;

        public DefaultFileService(IUnitOfWork unitOfWork)
        {
            this.unitOfWork = unitOfWork;
        }

        public bool HasAccess(int profileId, int fileId)
        {
            return unitOfWork.ProfileAccessRepository.GetAll().Any(
                a => a.ProfileId == profileId && a.FileMetadataId == fileId);
        }

        public FileMetadata? GetFile(int id)
        {
            return unitOfWork.FileMetadataRepository.Get(id);
        }

        public IEnumerable<FileMetadata> GetSharedFiles(string? login = null)
        {
            if (login == null)
            {
                return unitOfWork.FileMetadataRepository.GetAll();
            }
            var profile = unitOfWork.ProfileRepository.GetAll().FirstOrDefault(p =>
p.Login == login);
            if (profile == null)
            {
                return Enumerable.Empty<FileMetadata>();
            }
            var fileIds = unitOfWork.ProfileAccessRepository.GetAll()
                .Where(a => a.ProfileId == profile.Id).Select(a =>
a.FileMetadataId).ToList();
            return unitOfWork.FileMetadataRepository.GetAll().Where(f =>
fileIds.Contains(f.Id));
        }

        public IEnumerable<object> GetSharedFilesForUser(int id)
        {
            var files = unitOfWork.FileMetadataRepository.GetAll();
            var accesses = unitOfWork.ProfileAccessRepository.GetAll().Where(a =>
a.ProfileId == id).ToList();
            return files.Select(f => new {
                f.Id,
                f.Filename,
                LastDownload = accesses.FirstOrDefault(a => a.FileMetadataId ==
f.Id)?.LastDownloadDate,
                HasAccess = accesses.Exists(a => a.FileMetadataId == f.Id),
            });
        }

        public bool GrantAccess(int profileId, int fileId)
        {
            if (unitOfWork.ProfileRepository.Get(profileId) == null ||
                unitOfWork.FileMetadataRepository.Get(fileId) == null)
            {
```

```
                return false;
            }
            var accesses = unitOfWork.ProfileAccessRepository.GetAll();
            if (accesses.Any(a => a.ProfileId == profileId && a.FileMetadataId ==
fileId))
            {
                return false;
            }
            unitOfWork.ProfileAccessRepository.Create(new()
            {
                FileMetadataId= fileId,
                ProfileId= profileId,
                SharedDate= DateTime.Now,
            });
            unitOfWork.SaveChanges();
            return true;
        }

        public bool RevokeAccess(int profileId, int fileId)
        {
            var accesses = unitOfWork.ProfileAccessRepository.GetAll();
            var access = accesses.FirstOrDefault(a => a.ProfileId == profileId &&
a.FileMetadataId == fileId);
            if (access == null)
            {
                return false;
            }
            unitOfWork.ProfileAccessRepository.Remove(access.Id);
            unitOfWork.SaveChanges();
            return true;
        }

        public bool ShareFile(string path, long size)
        {
            var files = unitOfWork.FileMetadataRepository.GetAll();
            if (files.Any(a => a.Filename == path))
            {
                return false;
            }
            unitOfWork.FileMetadataRepository.Create(new()
            {
                Filename = path,
                UploadDate = DateTime.Now,
                Size = size,
            });
            unitOfWork.SaveChanges();
            return true;
        }

        public bool UnshareFile(int id)
        {
            if (unitOfWork.FileMetadataRepository.Get(id) == null)
            {
                return false;
            }
            unitOfWork.ProfileAccessRepository.GetAll()
                .Where(a => a.FileMetadataId == id)
                .Select(a => a.Id)
                .ToList()
                .ForEach(unitOfWork.ProfileAccessRepository.Remove);
            unitOfWork.FileMetadataRepository.Remove(id);
            unitOfWork.SaveChanges();
            return true;
        }
```

```
        public void UpdateFileLastDownload(int profileId, int fileId)
        {
            var access = unitOfWork.ProfileAccessRepository.GetAll().FirstOrDefault(
                a => a.ProfileId == profileId && a.FileMetadataId == fileId);
            if (access == null)
            {
                return;
            }
            unitOfWork.ProfileAccessRepository.Update(access.Id, f =>
            {
                f.LastDownloadDate = DateTime.Now;
            });
            unitOfWork.SaveChanges();
        }
    }
}
```

Файл DefaultPasswordHashService.cs

```
using BC = BCrypt.Net.BCrypt;

namespace LocalDrive.Services
{
    public class DefaultPasswordHashService : IPasswordHashService
    {
        public bool CheckPassword(string password, string hash)
        {
            return BC.Verify(password, hash);
        }

        public string Hash(string password)
        {
            return BC.HashPassword(password);
        }
    }
}
```

Файл DefaultProfileService.cs

```
using LocalDrive.DataAccess;
using LocalDrive.Entities;

namespace LocalDrive.Services
{
    public class DefaultProfileService : IProfileService
    {
        private readonly IUnitOfWork unitOfWork;
        private readonly IPasswordHashService passwordHashService;

        public DefaultProfileService(IUnitOfWork unitOfWork, IPasswordHashService
passwordHashService)
        {
            this.unitOfWork = unitOfWork;
            this.passwordHashService = passwordHashService;

            Seed();
        }

        private void Seed()
        {
            if (!GetProfiles().Any(p => p.IsOwner))
            {
                unitOfWork.ProfileRepository.Create(new()
```

```
                {
                    Login = "admin",
                    PasswordHash = passwordHashService.Hash("admin"),
                    IsOwner = true,
                    LastModifyDate = DateTime.UtcNow
                });
                unitOfWork.SaveChanges();
            }
        }

        public bool CheckCredentials(string login, string password)
        {
            var profile = GetProfiles().FirstOrDefault(p => p.Login == login);
            return profile != null && passwordHashService.CheckPassword(password,
profile.PasswordHash!);
        }

        public bool CreateProfile(string login, string password)
        {
            if (GetProfiles().Any(p => p.Login == login))
            {
                return false;
            }
            unitOfWork.ProfileRepository.Create(new()
            {
                Login= login,
                PasswordHash= passwordHashService.Hash(password),
                IsOwner= false,
                LastModifyDate= DateTime.UtcNow
            });
            unitOfWork.SaveChanges();
            return true;
        }

        public bool DeleteProfile(int id)
        {
            if (unitOfWork.ProfileRepository.Get(id) == null)
            {
                return false;
            }
            unitOfWork.ProfileAccessRepository.GetAll()
                .Where(a => a.ProfileId == id)
                .Select(a => a.Id)
                .ToList()
                .ForEach(unitOfWork.ProfileAccessRepository.Remove);
            unitOfWork.ProfileRepository.Remove(id);
            unitOfWork.SaveChanges();
            return true;
        }

        public Profile? GetProfile(int id)
        {
            return unitOfWork.ProfileRepository.Get(id);
        }

        public Profile? GetProfile(string login)
        {
            return GetProfiles().FirstOrDefault(p => p.Login == login);
        }

        public IEnumerable<Profile> GetProfiles()
        {
            return unitOfWork.ProfileRepository.GetAll();
        }
```

```
        public bool UpdateProfile(int id, string login, string password)
        {
            if (unitOfWork.ProfileRepository.Get(id) == null)
            {
                return false;
            }
            unitOfWork.ProfileRepository.Update(id, p =>
            {
                p.Login = login;
                p.PasswordHash = passwordHashService.Hash(password);
                p.LastModifyDate = DateTime.Now;
            });
            unitOfWork.SaveChanges();
            return true;
        }

        public void UpdateProfileLastLogin(int id)
        {
            if (unitOfWork.ProfileRepository.Get(id) == null)
            {
                return;
            }
            unitOfWork.ProfileRepository.Update(id, p =>
            {
                p.LastSignInDate = DateTime.Now;
            });
            unitOfWork.SaveChanges();
        }
    }
}
```

Файл FileController.cs

```
using LocalDrive.Services;
using LocalDrive.WebServer.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace LocalDrive.WebServer.Controllers
{
    [Route("/files")]
    [ApiController]
    public class FileController : ControllerBase
    {
        private readonly ILogger<FileController> _logger;
        private readonly IFileService _fileService;
        private readonly IProfileService _profileService;
        private readonly IConfiguration configuration;

        public FileController(ILogger<FileController> logger, IFileService
fileService, IProfileService profileService, IConfiguration configuration)
        {
            _logger = logger;
            _fileService = fileService;
            _profileService = profileService;
            this.configuration = configuration;
        }

        [HttpGet]
        [Route("/download/{fileId}")]
        [Authorize]
        public IActionResult DownloadFile(int fileId)
        {
            var login = User.Identity!.Name!;
```

```
            var profile = _profileService.GetProfile(login);
            if (!IsOwner())
            {
                if (profile == null || !_fileService.HasAccess(profile.Id, fileId))
                {
                    return NotFound();
                }
            }

            var file = _fileService.GetFile(fileId);
            if (file == null)
            {
                return NotFound();
            }

            var root = configuration["FileRoot"];
            if (root == null)
            {
                return StatusCode(StatusCodes.Status500InternalServerError);
            }

            var stream = System.IO.File.OpenRead(Path.Combine(root,
file.Filename!));
            if (profile != null)
            {
                _fileService.UpdateFileLastDownload(profile.Id, fileId);
            }
            return File(stream, "application/octet-stream", file.Filename);
        }

        [HttpGet]
        [Route("/shared")]
        [Authorize]
        public IActionResult GetSharedFiles()
        {
            var login = IsOwner() ? null : User.Identity!.Name;
            return Ok(_fileService.GetSharedFiles(login));
        }

        [HttpPost]
        [Route("/share")]
        [Authorize]
        [RequestSizeLimit(1024 * 1024 * 1024)]
        public IActionResult ShareFile([FromForm] ShareFileViewModel fileViewModel)
        {
            if (!IsOwner())
            {
                return Forbid();
            }
            var file = fileViewModel.FormFile;
            if (file == null)
            {
                return BadRequest();
            }

            var root = configuration["FileRoot"];
            if (root == null)
            {
                return StatusCode(StatusCodes.Status500InternalServerError);
            }
            Directory.CreateDirectory(root);
            var success = _fileService.ShareFile(file.FileName, file.Length);
            if (success)
            {
```

```
                using (var stream = System.IO.File.Create(Path.Combine(root,
file.FileName)))
                {
                    file.CopyTo(stream);
                }
                return Ok();
            }
            return BadRequest();
        }

        [HttpPost]
        [Route("/unshare")]
        [Authorize]
        public IActionResult UnshareFile([FromBody] int fileId = 0)
        {
            if (!IsOwner())
            {
                return Forbid();
            }

            var root = configuration["FileRoot"];
            if (root == null)
            {
                return StatusCode(StatusCodes.Status500InternalServerError);
            }

            var file = _fileService.GetFile(fileId);
            var success = _fileService.UnshareFile(fileId);
            if (success)
            {
                System.IO.File.Delete(Path.Combine(root, file!.Filename!));
                return Ok();
            }
            return BadRequest();
        }

        [HttpGet]
        [Route("/shared/{profileId}")]
        [Authorize]
        public IActionResult GetSharedFilesForUser(int profileId)
        {
            if (!IsOwner())
            {
                return Forbid();
            }

            var profile = _profileService.GetProfile(profileId);
            if (profile == null)
            {
                return BadRequest();
            }

            return Ok(new { User = profile.Login,  Files =
_fileService.GetSharedFilesForUser(profileId) });
        }

        [HttpPost]
        [Route("/grant")]
        [Authorize]
        public IActionResult GrantAccess([FromBody] UpdateAccessViewModel
updateAccessViewModel)
        {
            if (!IsOwner())
            {
                return Forbid();
```

```
            }
            var success = _fileService.GrantAccess(updateAccessViewModel.ProfileId,
updateAccessViewModel.FileId);
            return success ? Ok() : BadRequest();
        }

        [HttpPost]
        [Route("/revoke")]
        [Authorize]
        public IActionResult RevokeAccess([FromBody] UpdateAccessViewModel
updateAccessViewModel)
        {
            if (!IsOwner())
            {
                return Forbid();
            }
            var success = _fileService.RevokeAccess(updateAccessViewModel.ProfileId,
updateAccessViewModel.FileId);
            return success ? Ok() : BadRequest();
        }

        private bool IsOwner()
        {
            var login = User.Identity!.Name;
            var user = _profileService.GetProfiles().FirstOrDefault(p => p.Login ==
login);
            return user != null && user.IsOwner;
        }
    }
}
```

Файл ProfileController.cs

```
using LocalDrive.Services;
using LocalDrive.WebServer.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace LocalDrive.WebServer.Controllers
{
    [Route("profiles")]
    [ApiController]
    public class ProfileController : ControllerBase
    {
        private readonly ILogger<ProfileController> _logger;
        private readonly IFileService _fileService;
        private readonly IProfileService _profileService;
        private readonly IConfiguration configuration;

        public ProfileController(ILogger<ProfileController> logger, IFileService
fileService, IProfileService profileService, IConfiguration configuration)
        {
            _logger = logger;
            _fileService = fileService;
            _profileService = profileService;
            this.configuration = configuration;
        }

        [HttpGet]
        [Route("/all")]
```

```
        [Authorize]
        public IActionResult GetProfiles()
        {
            if (!IsOwner())
            {
                return Forbid();
            }
            return Ok(_profileService.GetProfiles());
        }

        [HttpPost]
        [Route("/create")]
        [Authorize]
        public IActionResult CreateProfile([FromBody] CreateProfileViewModel
createProfileViewModel)
        {
            if (!IsOwner())
            {
                return Forbid();
            }
            if (string.IsNullOrEmpty(createProfileViewModel.Login) ||
string.IsNullOrEmpty(createProfileViewModel.Password))
            {
                return BadRequest();
            }
            var success =
_profileService.CreateProfile(createProfileViewModel.Login,
createProfileViewModel.Password);
            return success ? Ok() : BadRequest();
        }

        [HttpPost]
        [Route("/delete")]
        [Authorize]
        public IActionResult DeleteProfile([FromBody] int id = 0)
        {
            if (!IsOwner())
            {
                return Forbid();
            }
            var success = _profileService.DeleteProfile(id);
            return success ? Ok() : BadRequest();
        }

        [HttpPost]
        [Route("/update")]
        [Authorize]
        public IActionResult UpdateProfile(UpdateProfileViewModel
updateProfileViewModel)
        {
            if (!IsOwner())
            {
                return Forbid();
            }
            if (string.IsNullOrEmpty(updateProfileViewModel.Login) ||
string.IsNullOrEmpty(updateProfileViewModel.Password))
            {
                return BadRequest();
            }
            var success = _profileService.UpdateProfile(
                updateProfileViewModel.Id,
                updateProfileViewModel.Login,
                updateProfileViewModel.Password);
            return success ? Ok() : BadRequest();
        }
```

```
        [HttpPost]
        [Route("/login")]
        public IActionResult Login([FromBody] LoginViewModel loginViewModel)
        {
            var login = loginViewModel.Login;
            if (string.IsNullOrEmpty(login) ||
string.IsNullOrEmpty(loginViewModel.Password))
            {
                return BadRequest();
            }
            if (_profileService.CheckCredentials(login, loginViewModel.Password))
            {
                var profile = _profileService.GetProfile(login);
                _profileService.UpdateProfileLastLogin(profile!.Id);
                var token = CreateToken(login);
                return Ok(new {
                    access_token = token.Item1,
                    expired_at = token.Item2,
                    name = login,
                    role = profile!.IsOwner ? "Owner" : "User",
                });
            }
            return BadRequest();
        }

        private (string, long) CreateToken(string login)
        {
            var authClaims = new List<Claim>()
            {
                new Claim(ClaimTypes.Name, login),
                new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
            };

            var authSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(configuration["JWT:Secret"]!));

            var period = int.TryParse(configuration["JWT:TimeToLive"], out int
@unchecked) ? @unchecked : 240;
            var expires = DateTime.Now.AddMinutes(period);
            var token = new JwtSecurityToken(
                issuer: configuration["JWT:ValidIssuer"],
                expires: expires,
                claims: authClaims,
                signingCredentials: new(authSigningKey,
SecurityAlgorithms.HmacSha256)
                );
            return (new JwtSecurityTokenHandler().WriteToken(token), expires.Ticks);
        }

        private bool IsOwner()
        {
            var login = User.Identity!.Name;
            var user = _profileService.GetProfiles().FirstOrDefault(p => p.Login ==
login);
            return user != null && user.IsOwner;
        }
    }
}
```

Файл CreateProfileViewModel.cs

```
namespace LocalDrive.WebServer.ViewModels
{
    public class CreateProfileViewModel
    {
        public string Login { get; set; }
        public string Password { get; set; }
    }
}
```

Файл LoginViewModel.cs

```
namespace LocalDrive.WebServer.ViewModels
{
    public class LoginViewModel
    {
        public string Login { get; set; }
        public string Password { get; set; }
    }
}
```

Файл ShareFileViewModel.cs

```
namespace LocalDrive.WebServer.ViewModels
{
    public class ShareFileViewModel
    {
        public IFormFile? FormFile { get; set; }
    }
}
```

Файл UpdateAccessViewModel.cs

```
namespace LocalDrive.WebServer.ViewModels
{
    public class UpdateAccessViewModel
    {
        public int ProfileId { get; set; }
        public int FileId { get; set; }
    }
}
```

Файл UpdateProfileViewModel.cs

```
namespace LocalDrive.WebServer.ViewModels
{
    public class CreateProfileViewModel
    {
        public string Login { get; set; }
        public string Password { get; set; }
    }
}
```

Файл DefaultPasswordHashService.cs

```
namespace LocalDrive.WebServer.ViewModels
{
    public class UpdateProfileViewModel
    {
```

```
        public int Id { get; set; }
        public string Login { get; set; }
        public string Password { get; set; }
    }
}
```

Файл appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "EntityContext": "Data Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=LocalDriveDB"
  },
  "JWT": {
    "ValidIssuer": "localhost:44328",
    "Secret": "set-your-secret-key-here",
    "TimeToLive": 240
  },
  "FileRoot": "C:\\Users\\Mihailo\\Documents\\localdrive-demo"
}
```

Файл Program.cs

```
using LocalDrive.DataAccess;
using LocalDrive.Services;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;
using System.Text;

var builder = WebApplication.CreateBuilder(args);
var services = builder.Services;

services.AddControllers();
services.AddEndpointsApiExplorer();
services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "LocalDrive", Version = "v1" });
    c.AddSecurityDefinition("Bearer", new()
    {
        Name = "Authorization",
        Type = SecuritySchemeType.ApiKey,
        Scheme = "Bearer",
        BearerFormat = "JWT",
        In = ParameterLocation.Header,
    });
    c.AddSecurityRequirement(new OpenApiSecurityRequirement
    {
        {
            new OpenApiSecurityScheme
            {
                Reference = new OpenApiReference
                {
                    Type = ReferenceType.SecurityScheme,
                    Id = "Bearer"
```

```
                }
            },
            Array.Empty<string>()
        }
    });
});

services.AddScoped<IFileService, DefaultFileService>();
services.AddScoped<IProfileService, DefaultProfileService>();
services.AddScoped<IPasswordHashService, DefaultPasswordHashService>();
services.AddScoped<IUnitOfWork, DefaultUnitOfWork>();
services.AddDbContext<LocalDriveDbContext>(
    options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("EntityContext"))
);

services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    options.SaveToken = true;
    options.RequireHttpsMetadata = true;
    options.TokenValidationParameters = new TokenValidationParameters()
    {
        ValidateIssuer = true,
        ValidateAudience = false,
        ValidIssuer = builder.Configuration["JWT:ValidIssuer"],
        IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["JWT:Secret"]!)),
    };
});

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseCors(c => c
    .AllowAnyOrigin()
    .AllowAnyHeader());

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

Файл BoldText.jsx

```
import { Typography } from "@mui/material";

const BoldText = ({children}) => {
```

```
    return (
        <Typography sx={{ fontWeight: '700' }}>
            {children}
        </Typography>
    )
}

export {BoldText}
```

Файл Layout.jsx

```
import { AppBar, Toolbar, Typography, Link, Button, Box } from "@mui/material"
import { Link as RouterLink, Outlet } from "react-router-dom"
import { useAuth } from "../hook/useAuth"

const Layout = () => {
    const {user, signout} = useAuth();

    return (
        <>
            <AppBar position="static" sx={{ backgroundColor: "#00BBBB"}}>
                <Toolbar>
                    <Typography component="h1" variant="h5" sx={{ flexGrow: 0.05,
ml: 1 }}>
                        Local Drive
                    </Typography>
                    <Box sx={{ flexGrow: 1, ml: 1 }}>
                        <nav hidden={user.role !== 'Owner'}>
                            <Link
                                variant="h6"
                                component={RouterLink}
                                to='/file'
                                sx={{ my: 1, mx: 2, color: "#FFFFFF",
textDecorationColor: "#FFFFFF" }}
                                >
                                Файли
                            </Link>
                            <Link
                                variant="h6"
                                component={RouterLink}
                                to='/profile'
                                sx={{ my: 1, mx: 2, color: "#FFFFFF",
textDecorationColor: "#FFFFFF" }}
                                >
                                Профілі
                            </Link>
                        </nav>
                    </Box>
                    <Button variant="text" sx={{ my: 1, mx: 1.5, color: "#FFFFFF",
border: "1px solid white" }} onClick={signout}>
                        <Typography variant="h7">
                            Вийти
                        </Typography>
                    </Button>
                </Toolbar>
            </AppBar>
            <Outlet/>
        </>
    )
}

export { Layout }
```

Файл AuthProvider.jsx

```
import { createContext, useState } from "react";

export const AuthContext = createContext(null);

export const AuthProvider = ({children}) => {
    const [user, setUser] = useState(null);
    const signin = (newUser, callbackfn) => {
        setUser(newUser);
        window.sessionStorage.setItem('token', newUser.access_token);
        callbackfn();
    }

    const signout = () => {
        setUser(null);
        window.sessionStorage.removeItem('token');
    }

    const value = {user, signin, signout};

    return <AuthContext.Provider value={value}>
        {children}
    </AuthContext.Provider>
}
```

Файл RequireAuth.jsx

```
import { useLocation, Navigate } from "react-router-dom";
import { useAuth } from '../hook/useAuth';

const RequireAuth = ({children}) => {
    const location = useLocation();
    const {user} = useAuth();

    if (!user) {
        return <Navigate to='/login' state={{from: location}}/>
    }

    return children;
}

export { RequireAuth }
```

Файл useAuth.js

```
import { useContext } from "react";
import { AuthContext } from "../hoc/AuthProvider";

export function useAuth() {
    return useContext(AuthContext);
}
```

Файл Accesspage.jsx

```
import { useState, useEffect } from "react";
import { useParams } from "react-router-dom"
```

```
import { Checkbox, Container, List, ListItem, ListItemText, Divider, Typography }
from "@mui/material";
import { sendGetAccessesRequest, sendGrantAccessRequest, sendRevokeAccessRequest }
from "../scripts/requestProvider";
import { BoldText } from "../components/BoldText";

const Access = () => {
    const {id} = useParams();
    const [login, setLogin] = useState(null);
    const [accessList, setAccessList] = useState([]);

    const date = (access) => {
        return access.hasAccess && (new Date(access.lastDownload || '0001-01-
01T00:00:00')).toLocaleString()
    }

    const onAccessChange = (access) => {
        access.hasAccess = !access.hasAccess;
        access.hasAccess ? sendGrantAccessRequest({
            profileId: id,
            fileId: access.id,
        }, () => {}, (error) => {
            alert("Unexpected error");
            console.log(error);
        }) : sendRevokeAccessRequest({
            profileId: id,
            fileId: access.id,
        }, () => {}, (error) => {
            alert("Unexpected error");
            console.log(error);
        })
        setAccessList(accessList.map((a) => a));
    }

    useEffect(() => {
        sendGetAccessesRequest({id}, (data) => {
            setLogin(data.user);
            setAccessList(data.files);
        }, () => {})
    }, [id]);

    return (
        <Container component="main">
            <Typography component="h1" variant="h4" sx={{ mt: 3, mb: 2}}>
                Редагувати доступ для користувача {login}
            </Typography>
            <List sx={{ width: "100%"}}>
                { accessList && (
                    <>
                        <ListItem>
                            <ListItemText primary={<BoldText>Назва файлу</BoldText>}
sx={{ width: "70%"}} />
                            <ListItemText primary={<BoldText>Останнє
завантаження</BoldText>} sx={{ width: "30%"}} />
                            <ListItemText primary={<BoldText>Доступ</BoldText>} />
                        </ListItem>
                        <Divider/>
                        {
                            accessList.map((access) => (
                                <>
                                    <ListItem>
                                        <ListItemText primary={access.filename}
sx={{ width: "70%"}} />
                                        <ListItemText primary={date(access)} sx={{
width: "30%"}} />
```

```
                                        <Checkbox checked={access.hasAccess}
onChange={() => {onAccessChange(access)}} sx={{ right: "0px"}} />
                                    </ListItem>
                                    <Divider/>
                                </>
                            ))
                        }
                    </>
                )}
            </List>
        </Container>
    )
}

export { Access }
```

Файл Filepage.jsx

```
import { useState, useEffect } from "react";
import { Button, Container, Divider, IconButton, Input, List, ListItem,
ListItemIcon, ListItemText, SvgIcon } from "@mui/material";
import { sendGetFilesRequest, sendShareFileRequest, sendUnshareFileRequest } from
"../scripts/requestProvider";
import { sizeToString } from "../scripts/sizeToString";
import { BoldText } from "../components/BoldText";
import { ReactComponent as DeleteIcon } from "../static/delete.svg"
import { ReactComponent as FileIcon } from "../static/file.svg"

const File = () => {
    const [fileList, setFileList] = useState([]);

    const browseFiles = () => {
        window.document.getElementById('file-input').click();
    }

    const uploadFile = (e) => {
        const selectedFile = e.target.files[0];
        if (!selectedFile) {
            return;
        }
        const form = new FormData();
        form.append("formFile", selectedFile);

        sendShareFileRequest(form, () => {
            sendGetFilesRequest({}, setFileList, () => {});
        }, (error) => {
            alert("Unexpected error");
            console.log(error);
        });
    }

    const deleteFile = (file) => {
        sendUnshareFileRequest(file.id, () => {
            sendGetFilesRequest({}, setFileList, () => {});
        }, (error) => {
            alert("Unexpected error");
            console.log(error);
        })
    }

    useEffect(() => {
        sendGetFilesRequest({}, setFileList, () => {});
    }, [])
```

```
    return (
        <Container component="main">
            <Button
                type="button"
                variant="contained"
                onClick={browseFiles}
                sx={{ mt: 3, mb: 2, backgroundColor: '#00BBBB', color: '#FFFFFF',
minWidth: '150px' }}
            >
                Додати
            </Button>
            <Input
                id="file-input"
                type="file"
                onChange={uploadFile}
                sx={{ display: "none"}}
            />
            <List sx={{ width: "100%"}}>
                { fileList && (
                    <>
                        <ListItem secondaryAction={<IconButton disabled/>}>
                            <ListItemIcon/>
                            <ListItemText primary={<BoldText>Ім'я</BoldText>} sx={{
width: "60%"}}/>
                            <ListItemText primary={<BoldText>Дата
завантаження</BoldText>} sx={{ width: "30%"}}/>
                            <ListItemText primary={<BoldText>Розмір</BoldText>}
sx={{ width: '10%' }} />
                        </ListItem>
                        <Divider/>
                        {
                            fileList.map((file) => (
                                <>
                                    <ListItem
                                        secondaryAction={
                                            <IconButton edge="end" aria-
label="delete" onClick={() => deleteFile(file)}>
                                                <SvgIcon component={DeleteIcon}
sx={{ color: '#000000' }} inheritViewBox />
                                            </IconButton>
                                        }
                                    >
                                        <ListItemIcon>
                                            <SvgIcon component={FileIcon} sx={{
color: '#000000' }} inheritViewBox  />
                                        </ListItemIcon>
                                        <ListItemText primary={file.filename} sx={{
width: '60%' }} />
                                        <ListItemText primary={(new
Date(file.uploadDate)).toLocaleString()} sx={{ width: '30%' }} />
                                        <ListItemText
primary={sizeToString(file.size)} sx={{ width: '10%' }} />
                                    </ListItem>
                                    <Divider/>
                                </>
                            ))
                        }
                    </>
                )}
            </List>
        </Container>
    )
}

export { File };
```

Файл Homepage.jsx

```
import { useState, useEffect } from "react";
import { Container, List, ListItem, IconButton, ListItemText, Divider, SvgIcon,
ListItemIcon } from "@mui/material";
import { BoldText } from "../components/BoldText";
import { sendDownloadFileRequest, sendGetFilesRequest } from
"../scripts/requestProvider";
import { sizeToString } from "../scripts/sizeToString";
import { ReactComponent as DownloadIcon } from "../static/download.svg"
import { ReactComponent as FileIcon } from "../static/file.svg"

const Home = () => {
    const [fileList, setFileList] = useState([]);

    const downloadFile = (file) => sendDownloadFileRequest({
        id: file.id,
    }, (blob) => {
        const url = window.URL.createObjectURL(new Blob([blob]));
        const link = document.createElement('a');
        link.href = url;
        link.setAttribute('download', file.filename);
        document.body.appendChild(link);
        link.click();
        link.parentNode.removeChild(link);
    }, (err) => {
        console.log(err);
    });

    useEffect(() => {
        sendGetFilesRequest({}, (data) => {
            console.log(data);
            setFileList(data);
        }, () => {});
    }, []);

    return (
        <Container component="main">
            <List sx={{ width: '100%' }}>
                {fileList && (
                    <>
                        <ListItem secondaryAction={<IconButton disabled/>}>
                            <ListItemIcon />
                            <ListItemText primary={<BoldText>Ім'я</BoldText>} sx={{
width: '60%' }} />
                            <ListItemText primary={<BoldText>Дата
завантаження</BoldText>} sx={{ width: '30%' }} />
                            <ListItemText primary={<BoldText>Розмір</BoldText>}
sx={{ width: '10%' }} />
                        </ListItem>
                        <Divider />
                        {
                            fileList.map((file) => (
                                <>
                                    <ListItem
                                        secondaryAction={
                                            <IconButton edge="end" aria-
label="download" onClick={() => downloadFile(file)}>
                                                <SvgIcon component={DownloadIcon}
sx={{ color: '#000000' }} inheritViewBox />
                                            </IconButton>
                                        }
                                    >
```

```
                                        <ListItemIcon>
                                            <SvgIcon component={FileIcon} sx={{
color: '#000000' }} inheritViewBox  />
                                        </ListItemIcon>
                                        <ListItemText primary={file.filename} sx={{
width: '60%' }} />
                                        <ListItemText primary={(new
Date(file.uploadDate)).toLocaleString()} sx={{ width: '30%' }} />
                                        <ListItemText
primary={sizeToString(file.size)} sx={{ width: '10%' }} />
                                    </ListItem>
                                    <Divider />
                                </>
                            ))
                        }
                    </>
                )

                }
            </List>

        </Container>
    )
}

export { Home };
```

Файл Loginpage.jsx

```
import { useNavigate } from "react-router-dom";
import { useAuth } from "../hook/useAuth";

import { Box, Typography, TextField, Button, Container } from "@mui/material";

import { sendLoginRequest } from "../scripts/requestProvider";

const Login = () => {
    const navigate = useNavigate();
    const {signin} = useAuth();

    const handleSubmit = (event) => {
        event.preventDefault();
        const form = event.target;
        const login = form.login.value;
        const password = form.password.value;

        const data = {
            "login": login,
            "password": password,
        }

        sendLoginRequest(data, (user) => {
            const toPage = user.role === 'Owner' ? '/file' : '/home';
            signin(user, () => navigate(toPage, {replace: true}));
        }, (error) => {
            alert("Bad credentials");
        });
    }

    return (
        <Container component="main" maxWidth="xs">
            <Box
                sx={{
                    marginTop: 16,
```

```
                    display: 'flex',
                    flexDirection: 'column',
                    alignItems: 'center',
                }}
            >
                <Typography component="h1" variant="h4">
                    Sign In
                </Typography>
                <Box component="form" onSubmit={handleSubmit} sx={{ mt:1 }}>
                    <TextField margin="normal" required fullWidth name="login"
label="Login" />
                    <TextField margin="normal" required fullWidth name="password"
label="Password" autoComplete="off" type="password" />
                    <Button
                        type="submit"
                        fullWidth
                        variant="contained"
                        sx={{ mt: 3, mb: 2, backgroundColor: '#00BBBB' }}
                    >
                        Sign In
                    </Button>
                </Box>
            </Box>
        </Container>
    )
}

export { Login };
```

Файл Profilepage.jsx

```
import { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import { Container, List, ListItem, ListItemText, Divider, IconButton, SvgIcon,
Button, TextField, Menu, MenuItem, ListItemIcon } from "@mui/material";
import { Dialog, DialogTitle, DialogContent, DialogActions } from "@mui/material"
import { BoldText } from "../components/BoldText";
import { sendGetProfilesRequest, sendCreateProfileRequest, sendUpdateProfileRequest,
sendDeleteProfileRequest } from "../scripts/requestProvider";
import { ReactComponent as MoreIcon } from "../static/more_vert.svg";
import { ReactComponent as UserIcon } from "../static/user.svg";
import { ReactComponent as AdminIcon } from "../static/admin.svg";
import { useAuth } from "../hook/useAuth";

const Profile = () => {
    const {user, signout} = useAuth();
    const navigate = useNavigate();
    const [profileList, setProfileList] = useState([]);
    const [openDialog, setOpenDialog] = useState(false);
    const [selectedProfile, setSelectedProfile] = useState(0);
    const [loginError, setLoginError] = useState("");
    const [passwordError, setPasswordError] = useState("");
    const [anchorEl, setAnchorEl] = useState(null);
    const [oldLogin, setOldLogin] = useState("");
    const open = Boolean(anchorEl);

    const getLoginErrorText = (str) => {
        if (!str) return "Поле має бути заповненим";
        if (str.length < 4) return "Довжина має бути не менше 4 символів";

        const special = '`~@#$%^&()[]{}+-*=/\\"\'<>!?,.;:';

        for (let index = 0; index < str.length; index++) {
            const element = str[index];
```

```
            if (special.includes(element)) return `Некоректний символ: ${element}`;
        }

        if (!selectedProfile && profileList.find((p) => p.login === str)) return
"Профіль з указаним логіном вже існує";

        return "";
    }

    const getPasswordErrorText = (str) => {
        if (!str) return "Поле має бути заповненим";
        if (str.length < 6) return "Довжина має бути не менше 6 символів";

        return "";
    }

    const openMenu = (event, profileId) => {
        setSelectedProfile(profileId)
        setAnchorEl(event.currentTarget);
    }

    const closeMenu = () => {
        setSelectedProfile(0)
        setAnchorEl(null);
    }

    const EditProfile = () => {
        setAnchorEl(null);
        openProfileDialog();
    }

    const DeleteProfile = () => {
        sendDeleteProfileRequest(selectedProfile, () => {
            sendGetProfilesRequest({}, setProfileList, () => {});
        }, (error) => {
            alert("Unexpected error");
            console.log(error);
        });
        closeMenu();
    }

    const openProfileDialog = () => {
        if (selectedProfile) {
            setOldLogin(profileList.find(p => p.id === selectedProfile).login);
        }
        setOpenDialog(true);
    }

    const closeProfileDialog = (isSave) => {
        if (isSave) {
            const newLogin = document.getElementById("new-login").value;
            const newPassword = document.getElementById("new-password").value;
            const lErr = getLoginErrorText(newLogin);
            const pErr = getPasswordErrorText(newPassword);
            setLoginError(lErr);
            setPasswordError(pErr);
            if (lErr || pErr) {
                return;
            }
            if (selectedProfile) {
                sendUpdateProfileRequest({
                    'id': selectedProfile,
                    'login': newLogin,
                    'password': newPassword,
                }, (_) => {
```

```
                    const profile = profileList.find((p) => p.id ===
selectedProfile);
                    profile.isOwner ? signout() : sendGetProfilesRequest({},
setProfileList, () => {});
                }, (error) => {
                    alert("Unexpected error");
                    console.log(error);
                });
            } else {
                sendCreateProfileRequest({
                    'login': newLogin,
                    'password': newPassword,
                }, (_) => {
                    sendGetProfilesRequest({}, setProfileList, () => {});
                }, (error) => {
                    alert("Unexpected error");
                    console.log(error);
                });
            }
        }
        setOpenDialog(false);
        setLoginError("");
        setPasswordError("");
        setSelectedProfile(0);
    }

    useEffect(() => {
        sendGetProfilesRequest({}, setProfileList, () => {});
    }, [])

    return (
        <Container component="main">
            <Button
                type="button"
                variant="contained"
                onClick={openProfileDialog}
                sx={{ mt: 3, mb: 2, backgroundColor: '#00BBBB', color: '#FFFFFF',
minWidth: '150px' }}
            >
                Створити
            </Button>
            <List sx={{ width: "100%"}}>
                { profileList && (
                    <>
                        <ListItem secondaryAction={<IconButton disabled/>}>
                            <ListItemIcon/>
                            <ListItemText primary={<BoldText>Логін</BoldText>} sx={{
width: '40%' }} />
                            <ListItemText primary={<BoldText>Останній
вхід</BoldText>} sx={{ width: '30%' }} />
                            <ListItemText primary={<BoldText>Дата
змінення</BoldText>} sx={{ width: '30%' }} />
                        </ListItem>
                        <Divider />
                        {
                            profileList.map((profile) => (
                                <>
                                    <ListItem
                                        secondaryAction={
                                            <IconButton
                                                aria-haspopup="true"
                                                aria-expanded={open ? 'true' :
undefined}
                                                onClick={(e) => openMenu(e,
profile.id)}
```

```
                                        >
                                            <SvgIcon component={MoreIcon} sx={{
color: '#000000' }} inheritViewBox />
                                        </IconButton>
                                    }
                                >
                                    <ListItemIcon>
                                        <SvgIcon component={profile.login ===
user.name ? AdminIcon : UserIcon} sx={{ color: '#000000' }} inheritViewBox  />
                                    </ListItemIcon>
                                    <ListItemText primary={profile.login} sx={{
width: '40%' }} />
                                    <ListItemText primary={(new
Date(profile.lastSignInDate)).toLocaleString()} sx={{ width: '30%' }} />
                                    <ListItemText primary={(new
Date(profile.lastModifyDate)).toLocaleString()} sx={{ width: '30%' }} />
                                </ListItem>
                                <Divider />
                            </>
                        ))
                    }
                </>
            )}
        </List>

        <Menu
            anchorEl={anchorEl}
            open={open}
            onClose={closeMenu}
        >
        <MenuItem onClick={() =>
{navigate(`/access/${selectedProfile}`)}}>Редагувати доступ</MenuItem>
            <MenuItem onClick={EditProfile}>Редагувати профіль</MenuItem>
            <MenuItem onClick={DeleteProfile}>Видалити</MenuItem>
        </Menu>

        <Dialog open={openDialog} onClose={() => {closeProfileDialog(false)}}>
            <DialogTitle>{selectedProfile ? `Редагування профілю ${oldLogin}` :
"Новий профіль"}</DialogTitle>
            <DialogContent>
                <TextField
                    margin="normal"
                    required
                    fullWidth
                    id="new-login"
                    label="Логін"
                    error={!!loginError}
                    helperText={loginError}
                    autoComplete="off"
                />
                <TextField
                    margin="normal"
                    required
                    fullWidth
                    id="new-password"
                    label="Пароль"
                    error={!!passwordError}
                    helperText={passwordError}
                    autoComplete="off"
                    type="password"
                />
            </DialogContent>
            <DialogActions>
                <Button onClick={() =>
{closeProfileDialog(false)}}>Відмінити</Button>
```

```
                    <Button onClick={() =>
{closeProfileDialog(true)}}>Зберегти</Button>
                </DialogActions>
            </Dialog>
        </Container>
    )
}

export { Profile };
```

Файл requestProvider.js

```
const same = (response) => response;
const json = (response) => response.json();
const blob = (response) => response.blob();

const appjson = (data) => JSON.stringify(data);
const appstream = (data) => data;

const sendRequest = ({path, method, headers, auth, type, format, data, successcb,
failcb}) => {
    const access_token = window.sessionStorage.getItem('token');

    if (auth && headers) {
        headers['Authorization'] = 'Bearer ' + access_token;
    }

    fetch('https://localhost:7203/' + path, {
        method: method || 'GET',
        headers: headers || {},
        body: data && type(data),
    }).then(
        (response) => {
            if (!response.ok) {
                throw new Error();
            }
            return format(response);
        }
    ).then(successcb).catch(failcb);
}

// File Controller
// GET /download/{fileId}
export const sendDownloadFileRequest = (params, successcb, failcb) => {
    sendRequest({
        path: `download/${params.id}`,
        method: 'GET',
        headers: {},
        auth: true,
        type: appjson,
        format: blob,
        data: null,
        successcb: successcb,
        failcb: failcb,
    });
}

// GET /shared
export const sendGetFilesRequest = (params, successcb, failcb) => {
    sendRequest({
        path: 'shared',
        method: 'GET',
        headers: {
            'Content-Type': 'application/json',
```

```
        },
        auth: true,
        type: appjson,
        format: json,
        data: null,
        successcb: successcb,
        failcb: failcb,
    });
}

// POST /share
export const sendShareFileRequest = (data, successcb, failcb) => {
    sendRequest({
        path: 'share',
        method: 'POST',
        headers: {},
        auth: true,
        type: appstream,
        format: same,
        data: data,
        successcb: successcb,
        failcb: failcb,
    });
}

// POST /unshare
export const sendUnshareFileRequest = (data, successcb, failcb) => {
    sendRequest({
        path: 'unshare',
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        auth: true,
        type: appjson,
        format: same,
        data: data,
        successcb: successcb,
        failcb: failcb,
    });
}

// GET /shared/{profileId}
export const sendGetAccessesRequest = (params, successcb, failcb) => {
    sendRequest({
        path: `shared/${params.id}`,
        method: 'GET',
        headers: {
            'Content-Type': 'application/json',
        },
        auth: true,
        type: appjson,
        format: json,
        data: null,
        successcb: successcb,
        failcb: failcb,
    });
}

// POST /grant
export const sendGrantAccessRequest = (data, successcb, failcb) => {
    sendRequest({
        path: 'grant',
        method: 'POST',
        headers: {
```

```
            'Content-Type': 'application/json',
        },
        auth: true,
        type: appjson,
        format: same,
        data: data,
        successcb: successcb,
        failcb: failcb,
    });
}

// POST /revoke
export const sendRevokeAccessRequest = (data, successcb, failcb) => {
    sendRequest({
        path: 'revoke',
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        auth: true,
        type: appjson,
        format: same,
        data: data,
        successcb: successcb,
        failcb: failcb,
    });
}

// Profile Controller
// GET /all
export const sendGetProfilesRequest = (params, successcb, failcb) => {
    sendRequest({
        path: 'all',
        method: 'GET',
        headers: {
            'Content-Type': 'application/json',
        },
        auth: true,
        type: appjson,
        format: json,
        data: null,
        successcb: successcb,
        failcb: failcb,
    });
}

// POST /create
export const sendCreateProfileRequest = (data, successcb, failcb) => {
    sendRequest({
        path: 'create',
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        auth: true,
        type: appjson,
        format: same,
        data: data,
        successcb: successcb,
        failcb: failcb,
    });
}

// POST /delete
export const sendDeleteProfileRequest = (data, successcb, failcb) => {
```

```
    sendRequest({
        path: 'delete',
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        auth: true,
        type: appjson,
        format: same,
        data: data,
        successcb: successcb,
        failcb: failcb,
    });
}

// POST /update
export const sendUpdateProfileRequest = (data, successcb, failcb) => {
    sendRequest({
        path: 'update',
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        auth: true,
        type: appjson,
        format: same,
        data: data,
        successcb: successcb,
        failcb: failcb,
    });
}

// POST /login
export const sendLoginRequest = (data, successcb, failcb) => {
    sendRequest({
        path: 'login',
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        auth: false,
        type: appjson,
        format: json,
        data: data,
        successcb: successcb,
        failcb: failcb,
    });
}
```

Файл sizeToString.js

```
const prefix = ['', 'К', 'М', 'Г', 'Т', 'Р']

export const sizeToString = (size) => {
    let i = 0;
    for (; i < prefix.length; i++) {
        if (size < 1024) {
            break;
        }
        size = size >>> 10;
    }
    return `${size} ${prefix[i]}Б`;
}
```

Файл App.jsx

```
import { Route, Routes } from 'react-router-dom';

import { Login } from './pages/Loginpage';
import { File } from './pages/Filepage';
import { Profile } from './pages/Profilepage';
import { Home } from './pages/Homepage';
import { Access } from './pages/Accesspage';

import { Layout } from './components/Layout';

import { RequireAuth } from './hoc/RequireAuth';
import { AuthProvider } from './hoc/AuthProvider';

function App() {
  return (
    <AuthProvider>
      <Routes>
        <Route path='/login' element={<Login/>}/>
        <Route path='/' element={
          <RequireAuth>
            <Layout/>
          </RequireAuth>
        }>
          <Route path='/file' element={<File/>}/>
          <Route path='/profile' element={<Profile/>}/>
          <Route path='/access/:id' element={<Access/>}/>
          <Route path='/home' element={<Home/>}/>
        </Route>
      </Routes>
    </AuthProvider>
  );
}

export default App;
```

Файл index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { BrowserRouter } from 'react-router-dom';
import './index.css';
import App from './App';

import '@fontsource/roboto/300.css';
import '@fontsource/roboto/400.css';
import '@fontsource/roboto/500.css';
import '@fontsource/roboto/700.css';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

КРОСПЛАТФОРМНИЙ ФАЙЛООБМІННИК ДЛЯ ЛОКАЛЬНИХ СЕРЕДОВИЩ

Програма та методика тестування

КПІ.ІТ-9121.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена МАРЧЕНКО

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Михайло ПЛЕСКАЧ

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є розроблений програмний застосунок для обміну файлами «Local Drive». Тестування проводиться на клієнтській і серверній частинах додатку. Згідно системних вимог, на тестове середовище повинна бути встановлена операційна система сімейства Windows NT, Linux Debian або MacOS. Також має бути підтримка платформ ASP.NET Core 7 і Node.js 18.2.0.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка правильності роботи програмного забезпечення відповідно до нефункціональних вимог;

– перевірка збереження даних;

– перевірка сумісності формату даних, якими обмінюється клієнтська і серверна частини застосунку;

– перевірка доступності сервера в мережі Інтернет;

– перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Chrome, Firefox, Safari);

– перевірка сумісності застосунку з різними операційними системами (Windows, Linux, MacOS);

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Статичне тестування виконується з метою оцінки якості програмного коду і змістовності документації. Вихідний код оцінюється на наявність алгоритмічних і технічних рішень, прийнятих в результаті аналізу вимог до програмного забезпечення, та їх відповідність теоретичним моделям. Опис цих рішень наведений у розділі 1.2.1 документу КПІ-ІТ-9121.045440.02.81. Також перевіряється змістовність, вичерпність і відповідність інструкції по встановленню і налаштуванню.

Функціональне тестування виконується з метою перевірки узгодженості даних і працездатності функціональної частини програмного забезпечення. Проводяться тести серверної частини застосунку. Через Swagger UI надсилаються запити і перевіряється відповідь на правильність статусу коду і формату даних. У випадку надсилання некоректного або забороненого запиту відповідь має містити про це повідомлення. Додатково перевіряється база даних на наявність очікуваних записів і відсутності неузгоджених даних.

Мануальне тестування виконується з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Воно проводиться з допомогою графічного користувацького інтерфейсу програми. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

– Тестування комунікації клієнтської і серверної частин застосунку;

– тестування інтерфейсу користувача;

– тестування зручності використання;

– тестування на виведення повідомлень про помилку, коли це необхідно.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Кросплатформний файлообмінник для локальних середовищ

Керівництво користувача

КПІ.ІТ-9121.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена МАРЧЕНКО

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Михайло ПЛЕСКАЧ

Київ – 2023

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«Local Drive» - це застосунок для робочого стола призначений для зручного і безпечного обміну файлами. Програма дозволяє користувачу поширювати свої файли мережею Інтернет. Інші користувачі можуть завантажити їх, попередньо авторизувавшись в системі за наданими йому адміністратором системи обліковими даними. Для кожного облікового запису можна обирати свої налаштовування доступу до конкретних файлів.

Програмне забезпечення складається з клієнтської та серверної частин.

Клієнтська частина складається з файлів вихідного коду мови програмування JavaScript. Під час інсталяції до них додаються бібліотеки, серед яких React.js, React Router, Material UI. Клієнт запускається програмою npm, яка постачається при встановленні програми разом з платформою Node.js.

Серверна частина складається з бібліотек платформи .NET, файлу налаштувань appsettings.json і виконуваного файлу для операційних систем сімейства Windows. Для роботи програми необхідно інсталювати runtime версію платформи ASP.NET Core.

Програмне забезпечення і інструкцію по налаштуванню і розгортанню можна завантажити з сервісу GitHub. На вибір користувачеві дається три випуски для різних платформ. Посилання на репозиторій: https://github.com/Vort-X/LocalDrive

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

- тип процесору: Intel Core i5;
- об’єм ОЗП: 8 ГБ;
- об’єм запам’ятовувального пристрою від 512 ГБ;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт;
- підтримка системою платформи ASP.NET Core 7.

2.2 Завантаження застосунку

Застосунок можна завантажити з платформи GitHub. Кожен випуск містить інструкцію по налаштуванню і розгортанню. Для початку роботи спершу необхідно розпакувати архів у своїй файловій системі. Далі, слідуючи інструкції, потрібно встановити платформи ASP.Net Core і Node.js. Після цього для встановлення бібліотек клієнта необхідно запустити наданий разом зі збіркою скрипт. Також для можливості встановлення Інтернет-з’єднання зовнішніх користувачів з програмою необхідно налаштувати переадресацію портів, що також описано в інструкції.

2.3 Перевірка коректної роботи

Якщо після запуску клієнтської частини застосунку з’явилось вікно веб-браузера і користувач не отримав повідомлення про помилку, то установка клієнтської частини пройшла успішно. Якщо після авторизації за обліковими даними користувача за замовчуванням з’являється сторінка з файлами, то установка серверної частини пройшла успішно.

Логін за замовчуванням: admin.

Пароль за замовчуванням: admin.

3 ВИКОНАННЯ ПРОГРАМИ

Для початку користування програмним застосунком необхідно відкрити його в браузері. Користувач одразу потрапляє на сторінку авторизації.

Рисунок 3.1 – Сторінка авторизації

Якщо користувач авторизується під обліковими даними адміністратора, то його переправить на сторінку файлів.

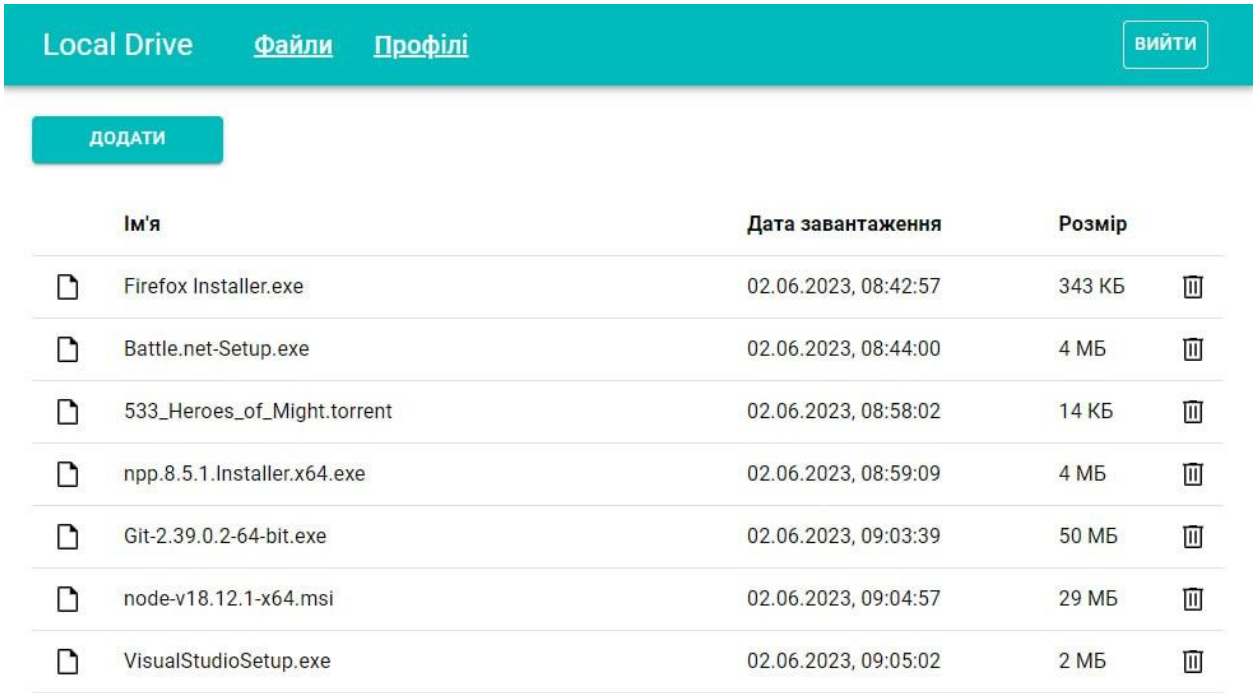


Рисунок 3.2 – Сторінка файлів

На цій сторінці знаходиться список поширених файлів. Також після авторизації адміністратору доступний заголовок з посиланням на сторінки файлів і профілів та кнопка виходу.

Щоб додати файл до списку поширених необхідно натиснути кнопку «Додати». Після цього з’явиться вікно файлового менеджера системи.

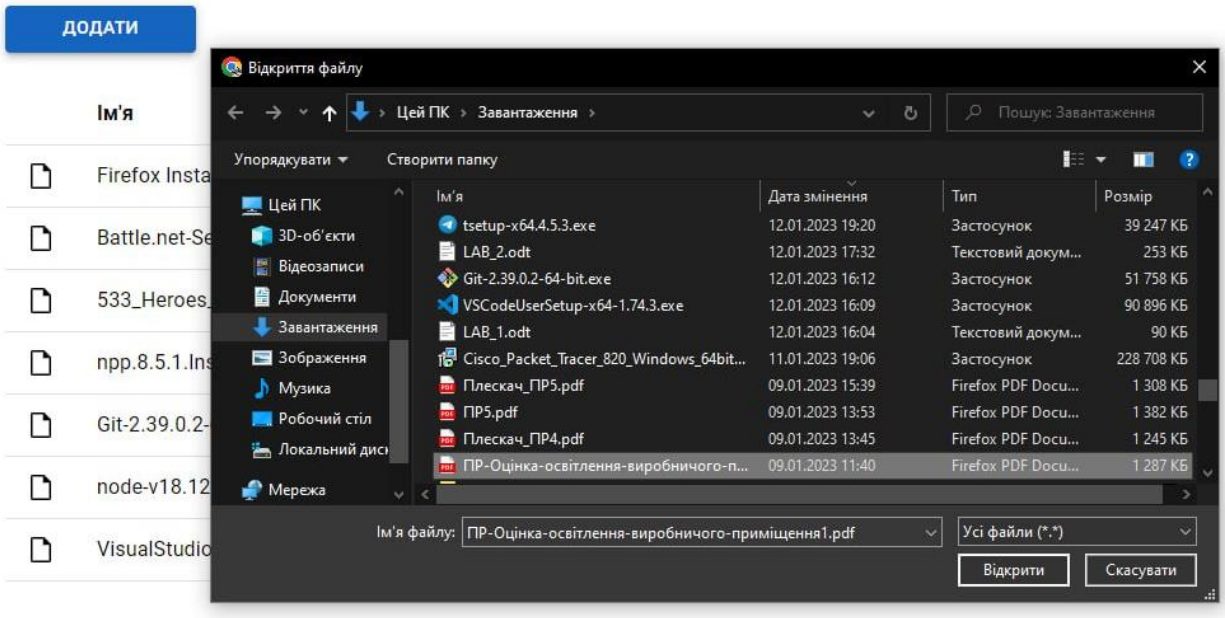


Рисунок 3.3 – Вікно вибору файлу

Після завантаження файлу він автоматично потрапляє у список.

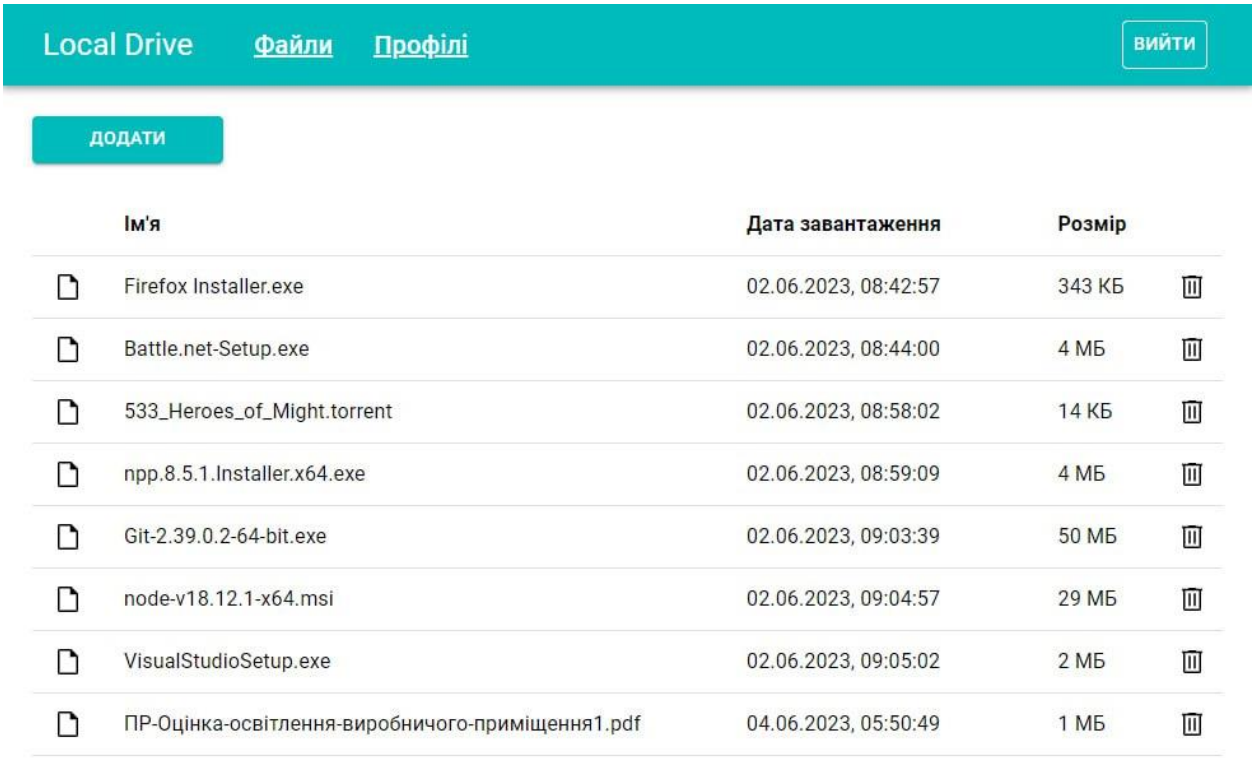


Рисунок 3.4 – Оновлений список файлів

Також при натисканні «Корзини» відповідний файл видаляється з серверу.

Перейшовши на сторінку профілів адміністратор отримує доступ до облікових записів користувачів, в тому числі і до свого.

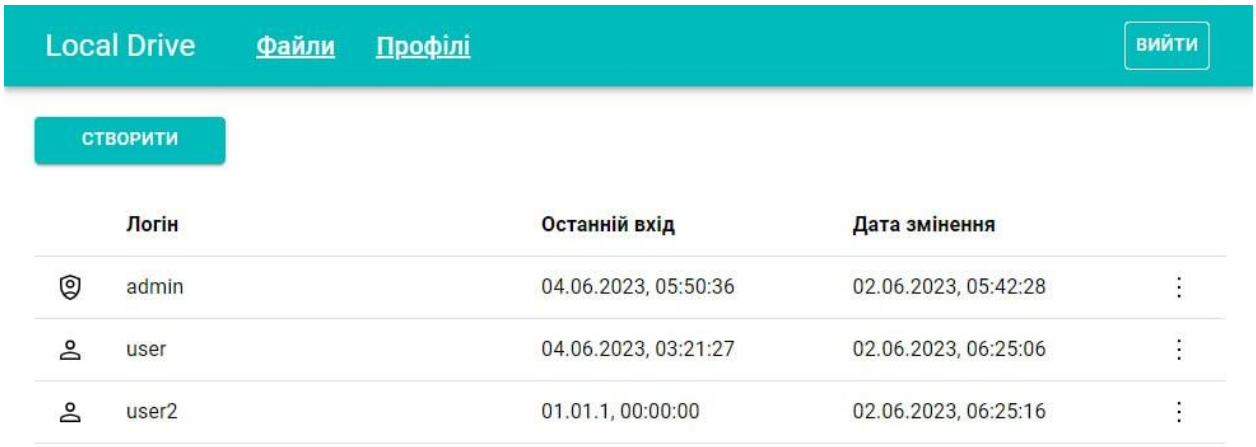


Рисунок 3.5 – Сторінка профілів

Щоб додати профіль спочатку необхідно натиснути на кнопку «Створити». Відкриється діалогове вікно з формою для введення облікових даних.

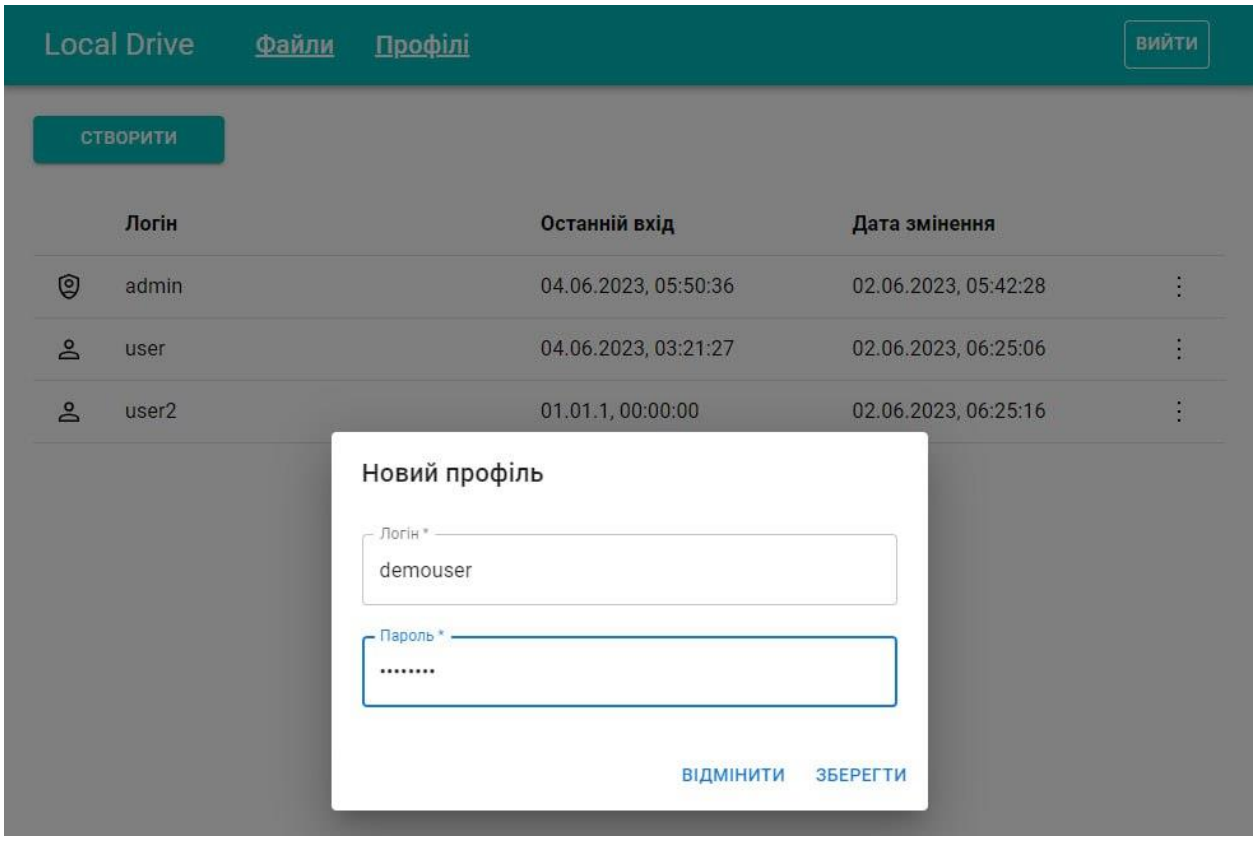


Рисунок 3.6 – Діалогове вікно створення профілю

При відправленні форми до списку додається щойно створений профіль.

Щоб керувати профілями передбачене контекстне меню. Воно з’являється при натисканні на іконку трьох вертикальних крапок. З доступних дій – редагування доступу, редагування профілю і видалення профілю.

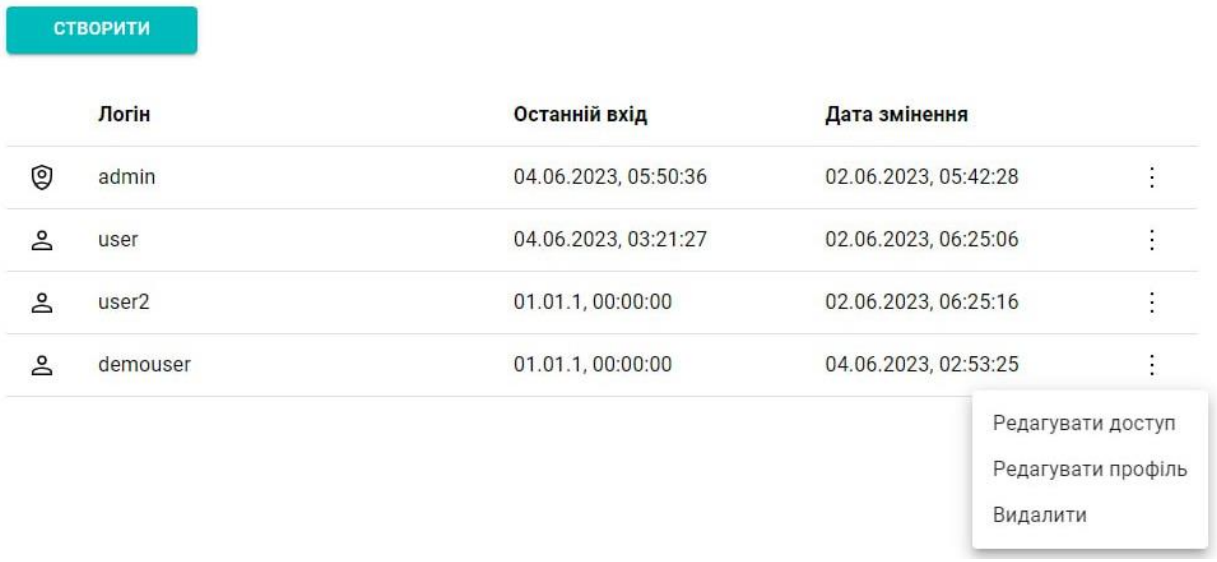


Рисунок 3.7 – Контекстне меню керування профілем

Для редагування профілю використовується ідентичне попередньому діалогове вікно.

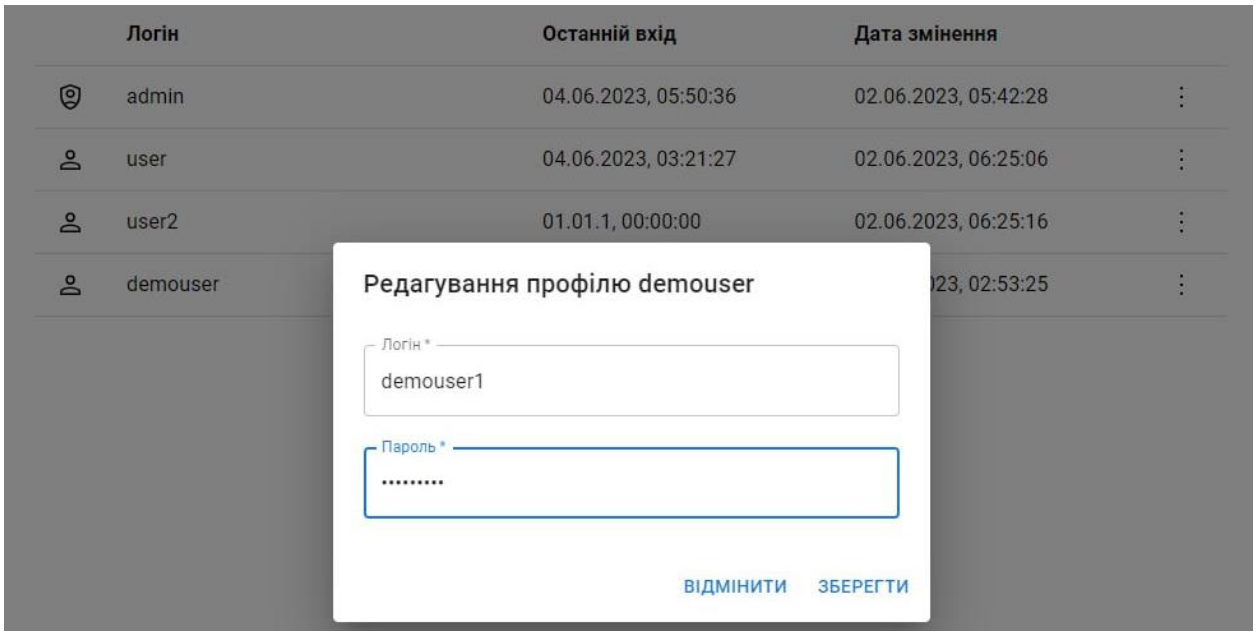


Рисунок 3.8 – Діалогове вікно редагування профілю

Якщо в контекстному меню керування профілем обрати «Редагувати доступ», то адміністратор потрапить на відповідну сторінку. Тут можна дізнатися дату і час останнього завантаження і надавати або обмежувати доступ профіля до файлів.

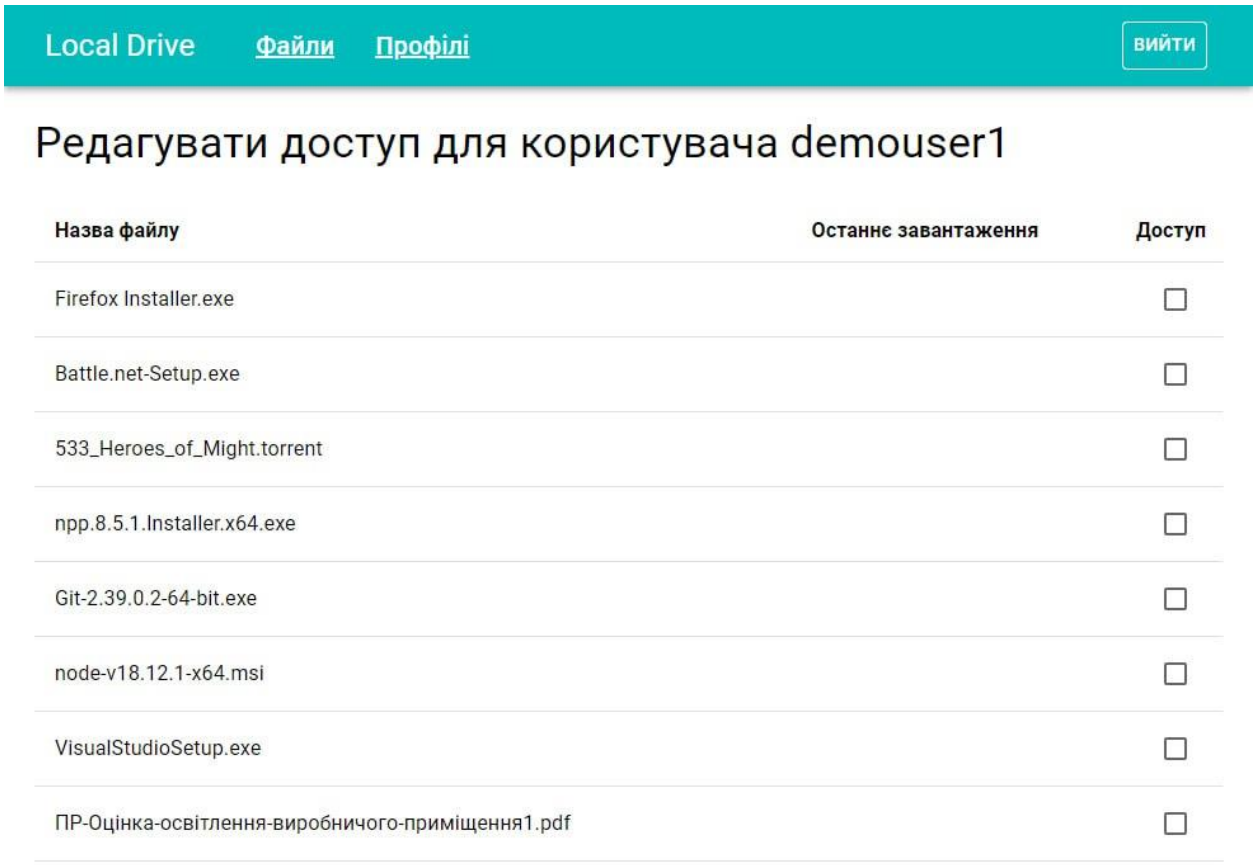


Рисунок 3.9 – Сторінка редагування доступу профіля

Для надання профілю доступу до файлу необхідно активувати прапорець. І навпаки, для обмеження доступу прапорець потрібно деактивувати.

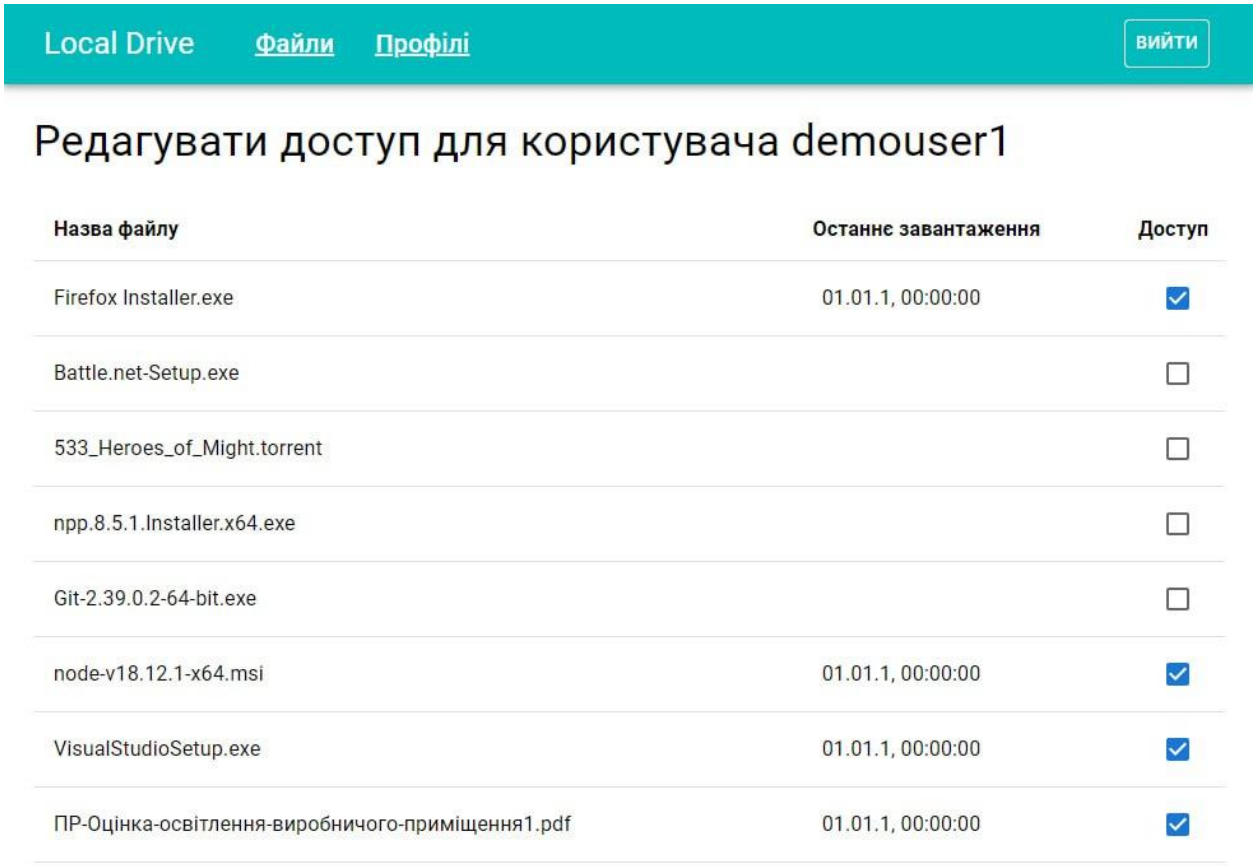


Рисунок 3.10 – Редагування доступу до файлу через прапорці

Щоб перевірити, чи наданий був доступ до файлів, адміністратор може авторизуватись під створеним ним профілем. Для цього потрібно натиснути на кнопку «Вийти». Відкриється сторінка авторизації.

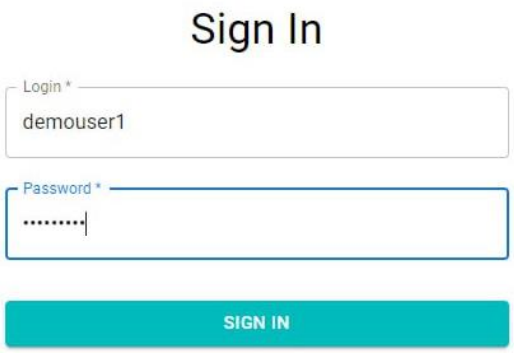


Рисунок 3.11 – Сторінка авторизації після виходу

Після авторизації за обліковими даними звичайного користувача відкривається сторінка доступних йому файлів. На ній є список з даними про файли і іконки завантаження для кожного з них.

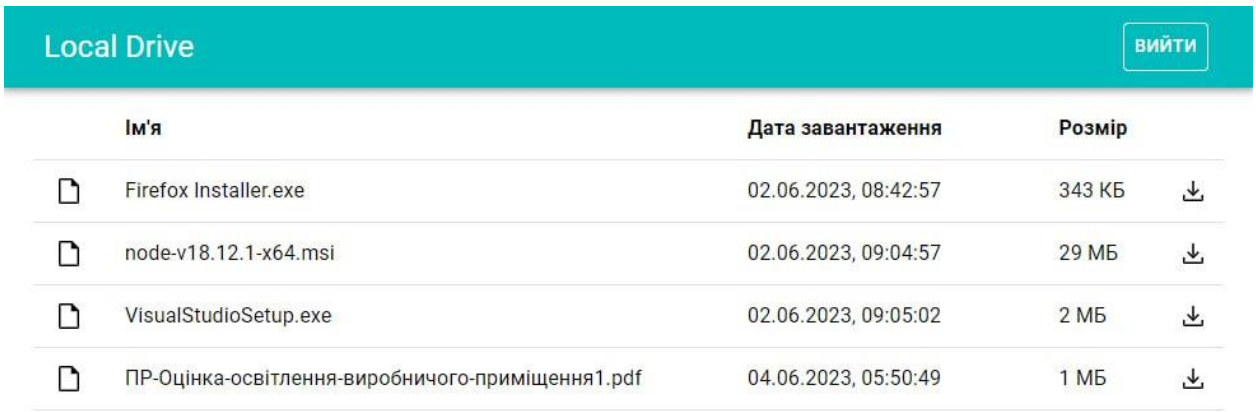


Рисунок 3.12 – Сторінка доступних файлів

Після натискання на іконку браузером почнеться автоматична загрузка обраного файла.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

КРОСПЛАТФОРМНИЙ ФАЙЛООБМІННИК ДЛЯ ЛОКАЛЬНИХ СЕРЕДОВИЩ

Графічний матеріал

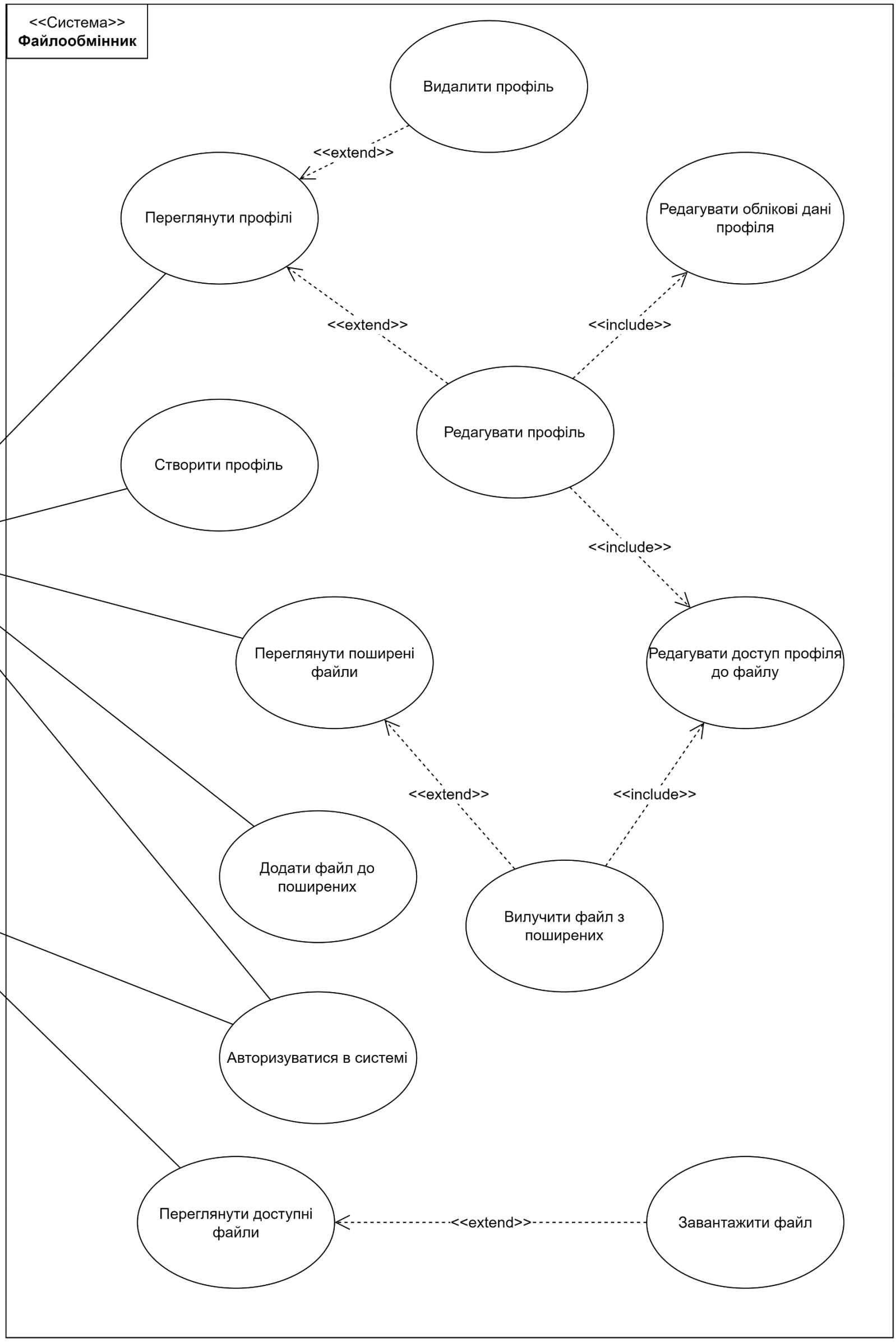
КПІ.ІТ-9121.045440.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена МАРЧЕНКО

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

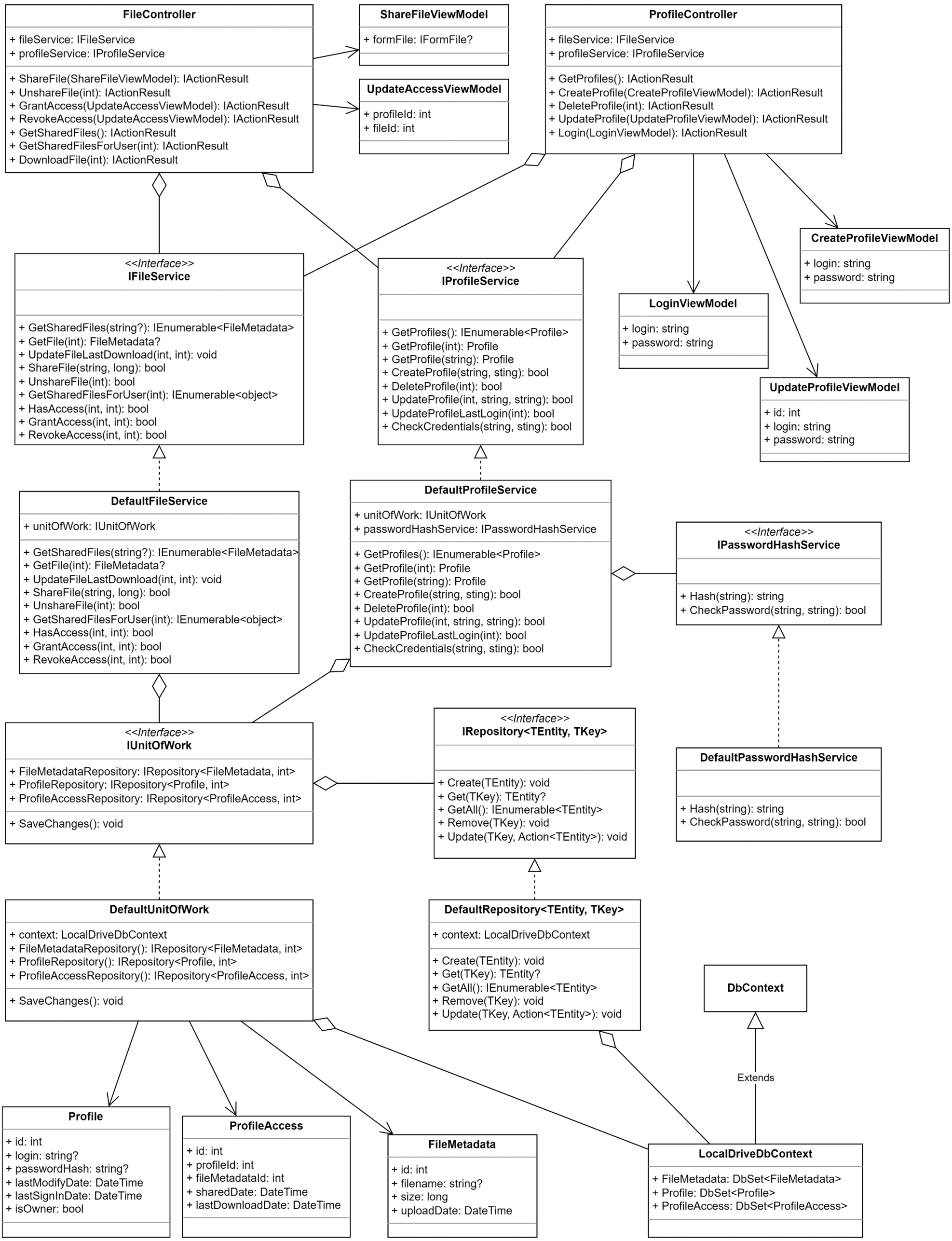
Виконавець:
_____ Михайло ПЛЕСКАЧ

Київ – 2023



					КПІ.ІТ-9121.045440.06.99.СС	
					Схема структурна варіантів використань	*Лит.* / *Арк.* 1 / *Аркушів* 1
Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*		*Аркуш* / *Аркушів*
Розроб.		*Плескач М.Т.*			*Кросплатформний файлообмінник для локальних середовищ*	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-91
Перев.		*Марченко О.І.*				
Т. Кон.						
Н. Кон.		*Головченко М.М.*				
Затв.		*Жаріков Е. В.*				

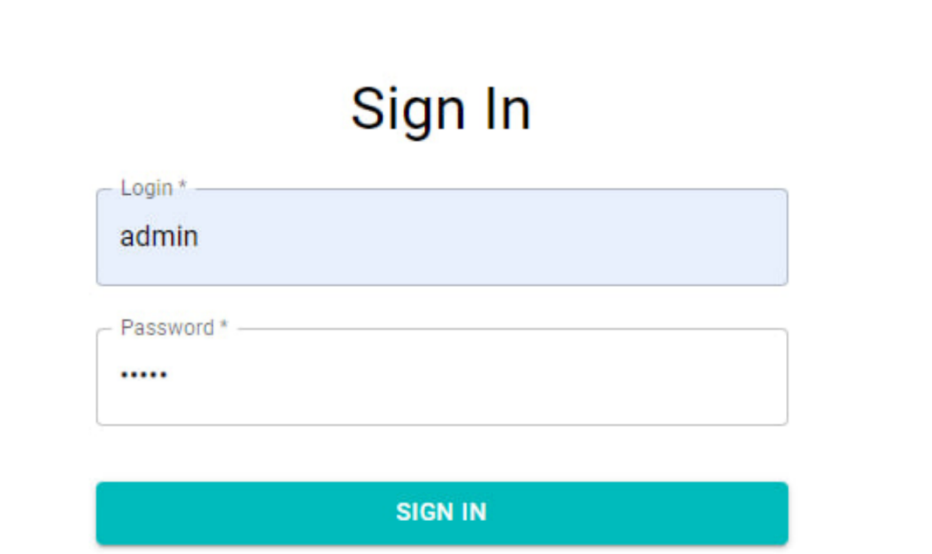
КПІ.ІТ-9121.045440.06.99.СС

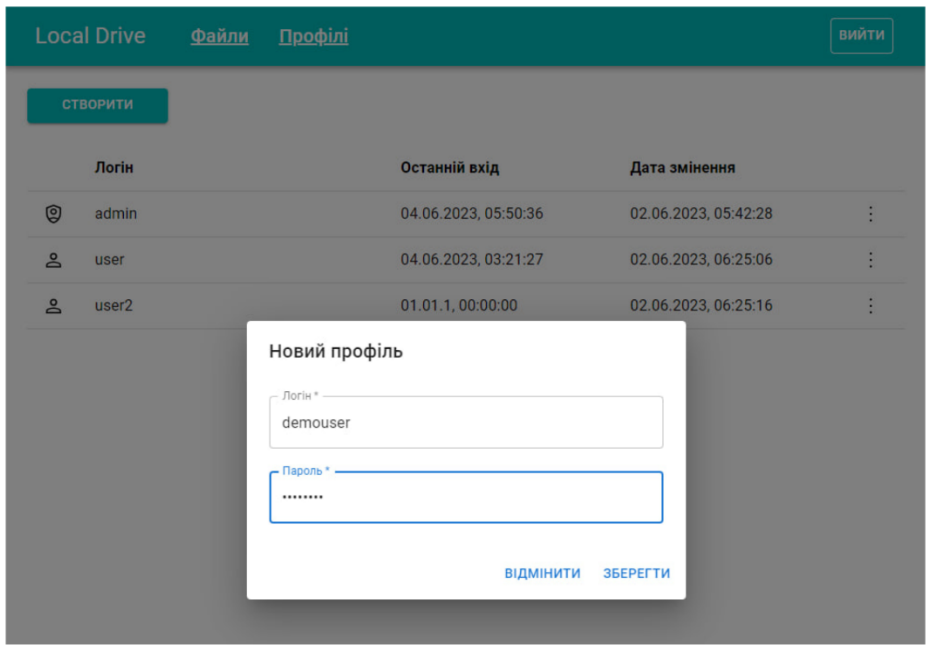


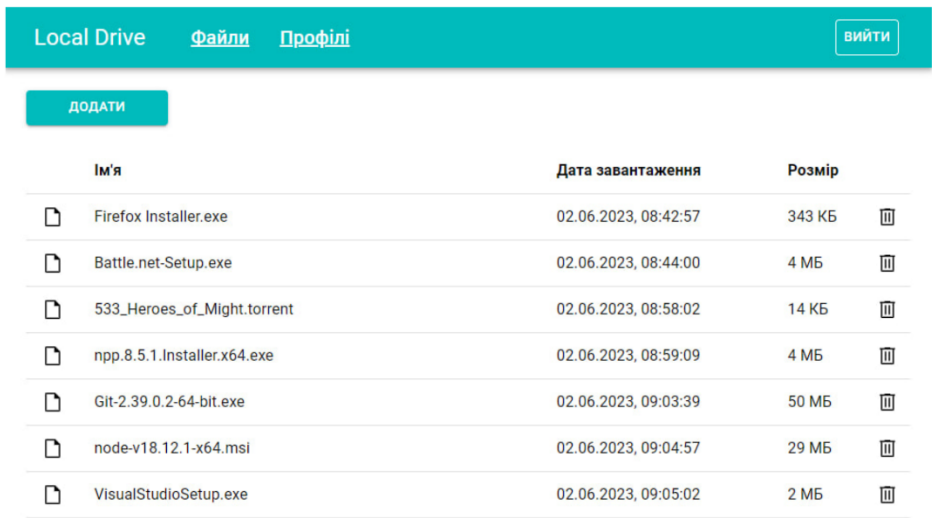
Зм.	Арк.	№ докум.	Підп.	Дата
Розроб.		Плескач М.Т.		
Перев.		Марченко О.І.		
Т. Кон.				
Н. Кон.		Головченко М.М.		
Затв.		Жаріков Е. В.		

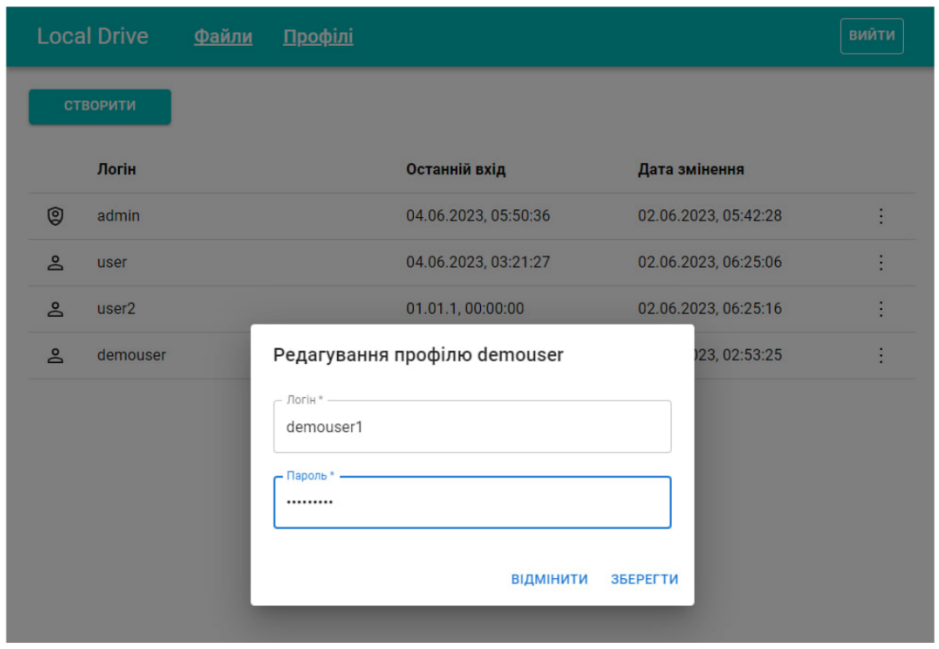
КПІ.ІТ-9121.045440.06.99.СС

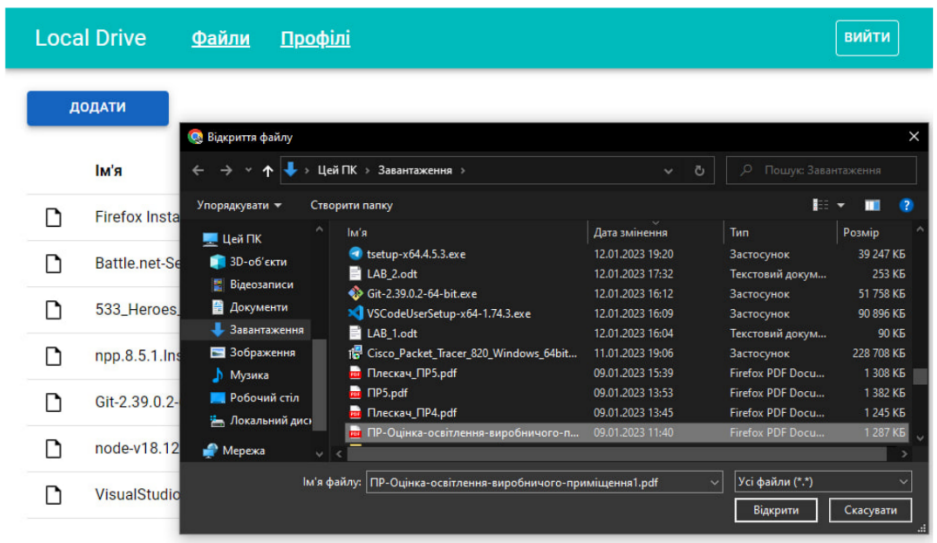
Схема структурна класів програмного забезпечення	Лит.	Арк.	Аркушів
		1	1
	Аркуш	Аркушів	
Кросплатформний файлообмінник для локальних середовищ	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-91		

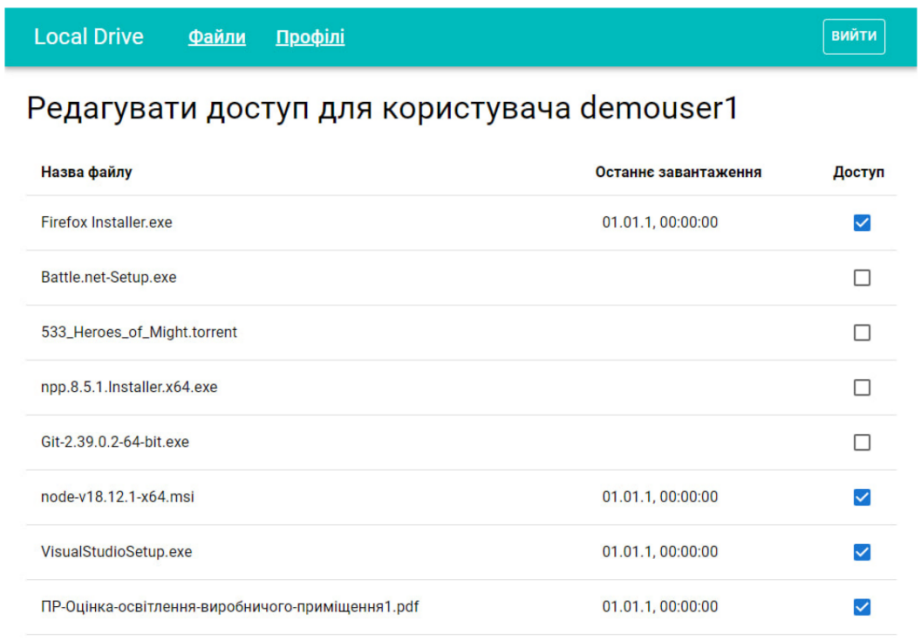


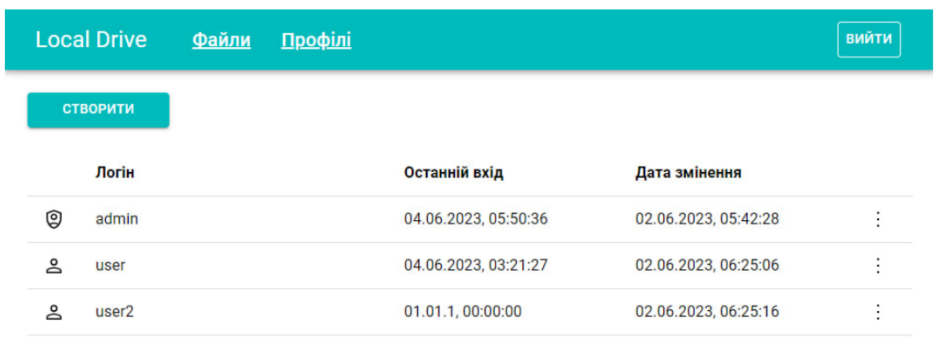


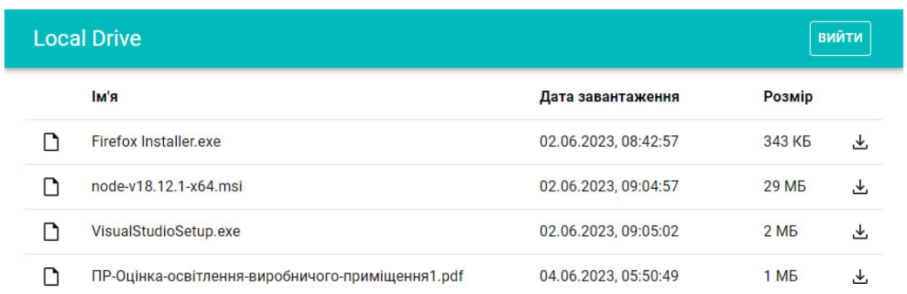












					КПІ.ІТ-9121.045440.06.99.КЕ
Зм.	Арк.	№ докум.	Підп.	Дата	
Розроб.		Плескач М.Т.			Креслення вигляду екранних форм
Перев.		Марченко О.І.			Лит.: — / Арк.: 1 / Аркушів: 1
Т. Кон.					Аркуш / Аркушів
					Кросплатформний файлообмінник для локальних середовищ
Н. Кон.		Головченко М.М.			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-91
Затв.		Жаріков Е. В.			