Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра мережних технологій

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: «**РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ ОСОБИСТИХ ФАЙЛІВ**»

Виконала: студентка 4-го року навчання,

Спеціальності

121 Інженерія програмного забезпечення

Верхогляд Катерина Ігорівна

Керівник Черкасов Д.І.

к.т.н., старший викладач

Рецензент _____

(прізвище та ініціали)

Кваліфікаційна робота захищена

з оцінкою _____

Секретар ЕК _____

«____» _____ 20____ р.

Київ – 2025

Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра мережних технологій

ЗАТВЕРДЖУЮ

Зав. кафедри мережних технологій, проф., д.ф.-м.н.

Г. І. Малашонок _____

(підпис)

“____”_____2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студентці Верхогляд Катерині Ігорівні

факультету інформатики 4 курсу бакалаврської програми

ТЕМА: Розробка програмного застосунку для організації особистих файлів

Зміст ТЧ до кваліфікаційної роботи:

Вступ

Огляд наявних рішень

Проектування власного рішення

Програмна реалізація власного рішення

Висновки

Джерела

Додатки (за потреби)

Дата видачі „___” _____ 2024 р. Керівник _____

(підпис)

Завдання отримав _____

Календарний план виконання курсової роботи

Тема: «Система для організації персональної мультимедійної інформації**»**

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання теми кваліфікаційної роботи.	31.10.2024	
2.	Огляд літератури за темою роботи.	31.01.2025	
3.	Написання вступу.	07.02.2025	
4.	Проектування власного рішення.	28.02.2025	
8.	Програмна розробка застосунку.	30.04.2025	
9.	Написання й оформлення текстової частини.	20.05.2024	
10.	Захист кваліфікаційної роботи.	02.06.2025	

Студент Верхогляд К.І.

Керівник Черкасов Д.І.

“_____”_____

ЗМІСТ

АНОТАЦІЯ

У роботі розглянуто актуальну проблему організації, зберігання та захисту персональних цифрових файлів у сучасних користувачів. Проведено аналіз особливостей користувацької поведінки та порівняльний огляд наявних програмних рішень, що виявив відсутність інструментів, які б повністю задовольняли вимоги до автоматизованої ієрархічної організації файлів із урахуванням індивідуальних запитів користувача. На основі визначених технічних вимог розроблено архітектуру, концептуальну модель бази даних та прототип програмного застосунку для ефективної організації персональних файлів. Особливу увагу приділено впровадженню механізмів автоматичного групування файлів на основі аналізу змісту, метаданих та користувацьких запитів із застосуванням векторних та мовних моделей. Створений прототип відкриває перспективи для подальшого розвитку системи, зокрема вдосконалення алгоритмів автоматичного групування, впровадження стратегій chunking та підтримки аналізу мультимедійних даних, що підвищить ефективність і гнучкість управління персональними цифровими файлами.

ВСТУП

Зручна й надійна організація персональних файлів є актуальною потребою сучасного користувача цифрових пристроїв, адже зі збільшенням обсягів даних – фотографій, відеозаписів, особистих документів, книг та інших файлів – виникає проблема ефективного керування накопиченою інформацією, її організації, захисту та швидкого доступу до неї. Проте наявні рішення здебільшого орієнтовані на організацію окремих типів файлів або зберігання даних у хмарних сховищах без надання користувачу достатнього рівня контролю над власною інформацією.

Метою роботи є розробка програмного застосунку для ефективної організації, зберігання та захисту персональних файлів.

Для досягнення цієї мети необхідно вирішити наступне ***наукове завдання***:

1. З’ясувати особливості поведінки користувачів у процесі взаємодії з персональною колекцією файлів.
2. Провести порівняльний аналіз наявних рішень у сфері організації персональних даних.
3. Визначити вимоги до розроблюваної системи.
4. Розробити архітектуру майбутнього застосунку з урахуванням гнучкості, розширюваності та безпеки.
5. Реалізувати програмний компонент спроектованого рішення.

Робота складається з чотирьох розділів.

У першому розділі проведено аналіз користувацької поведінки щодо організації цифрових файлів, на основі якого визначено основні орієнтири для розробки системи.

Другий розділ присвячено огляду п’яти релевантних програмних рішень, що використовують для організації персональних цифрових даних, а також здійснено аналіз їхніх переваг і недоліків за визначеними критеріями.

У третьому розділі, спираючись на результати попередніх досліджень, сформульовано технічні вимоги до розроблюваного застосунку та спроектовано застосунок.

Четвертий розділ присвячено обґрунтуванню обраних технологій та опису процесу розробки програмної частини застосунку.

РОЗДІЛ 1. АНАЛІЗ ПОТРЕБ КОРИСТУВАЧІВ І ОСНОВНИХ ПРОБЛЕМ ОРГАНІЗАЦІЇ ЦИФРОВОЇ ІНФОРМАЦІЇ

У цьому розділі розглянуто, які саме потреби мають користувачі під час роботи з цифровою інформацією та які труднощі найчастіше виникають. Також проаналізовано, як саме люди намагаються організувати свої файли, і що може їм допомогти оптимізувати цей процес.

1.1 Пошук, як альтернатива навігації.

Автори книги «The Science of Managing Our Digital Stuff» порівнюють два основні підходи до знаходження особистих файлів: навігацію в ієрархії тек і пошук за певними атрибутами файлу [1, с. 107-120]. Незважаючи на думку, що сучасні пошукові системи простіші та ефективніші за навігацію, дослідження показують, що більшість користувачів надають перевагу саме навігації.

Пошукові системи стали швидкими й універсальними, проте основний спосіб знаходження файлів лишається навігацією серед ієрархії тек.

Результати досліджень, проведених для різних ОС і з різними пошуковими системами, показали, що навігацію користувачі використовували у 56–68% випадків пошуку файлів, тоді як пошук лише у 7–15%. Пошук зазвичай застосовується як крайній захід, коли користувач не пам’ятає розташування потрібного файлу.

Попри постійне вдосконалення пошукових систем, користувачі здебільшого надають перевагу навігації через папки, що пов’язано із особливостями пам’яті та просторового сприйняття людини. Дослідження показують, що користувачі добре пам’ятають місце збереження своїх файлів і майже завжди можуть знайти їх саме через навігацію.

1.2 Теги, як альтернатива традиційній ієрархії тек

Автори книги розглядають теги як альтернативу текам у персональному управлінні інформацією (PIM) [1, с. 121-139]. Серед основних переваг тегів виділяють можливість множинної класифікації (один файл можна позначити кількома тегами) та відсутність ієрархічної структури, що властива текам. Теки обмежують користувача єдиною категорією для файлу, тегування ж дозволяє застосовувати кілька ключових слів і здійснювати пошук за будь-яким із них, водночас зберігаючи всі дані у спільному «пласкому» сховищі.

Попри це, дослідження показали, що коли користувачі мають вибір між організацією за тегами або теками, вони надають перевагу текам й ієрархічному зберіганню як для організації, так і для пошуку даних, а також уникають множинної класифікації.

Серед причин такої прихильності до тек: простота (однозначна класифікація простіша для запам’ятовування), звичка, а також те, що ієрархія тек дозволяє сформувати чітку структуру інформації в пам’яті користувачів. На противагу множинна класифікація тегами часто здається користувачам складною, оскільки ручне призначення тегів займає більше часу, а їхня гнучкість може призвести до плутанини через відсутність стандартизації назв.

Отже, у персональному управлінні інформацією багато користувачів залишаються прихильниками тек, ієрархії та одноразової класифікації, і відмовляються від використання тегів зокрема через те, що процес їх присвоєння файлам забирає багато часу.

1.3 Системи співпраці для організації файлів

Автори книги також розглядають групове управління інформацією (GIM) як альтернативу персональному управлінню інформацією (PIM) для можливості спільної роботи з файлами [1, с. 141-159]. GIM передбачає зберігання та організацію файлів у спільному сховищі (наприклад, Google Drive), тоді як PIM означає, що кожен користувач організовує спільні файли у власному порядку, а для обміну ними використовує, наприклад, електронну пошту.

Дослідження авторів показало, що користувачі значно швидше знаходять файли, якщо організовували їх самостійно, ніж коли структура була створена іншим користувачем.

Отже, у спільних середовищах виникають проблеми організації даних, оскільки кожен користувач має власне бачення вдалого структурування файлів, і загальна ієрархія не задовольняє всіх.

1.4 Користувацький суб’єктивний підхід до організації персональної інформації

Автори книги пропонують застосовувати суб’єктивний підхід до організації персональної інформації, що полягає у врахуванні не лише загальних характеристик, таких як формат, розмір чи дата створення, а й таких, що залежать від самого користувача, як-от особиста важливість об'єкта, проект чи тема, до якої він належить, а також контекст використання цієї інформації. [1, с. 179-201].

На основі цього підходу автори сформулювали три основні дизайн-принципи до розробки системи організації персональної інформації.

Перший принцип – *принцип суб'єктивної важливості* – передбачає, що більш важлива інформація для користувача повинна бути більш помітною та доступною в інтерфейсі. Важливі об'єкти мають бути легкодоступними («принцип промоції»), а менш важливі – менш помітними, але не видаленими («принцип демоції»).

Другий принцип – *принцип проектної класифікації* – пропонує групувати всю інформацію, що стосується одного проекту чи теми, незалежно від типу файлів чи програм, у яких вони створені.

Третій принцип – *принцип контексту* – передбачає відображення даних користувачу в тому ж контексті, у якому він працював із ними раніше, що полегшує їх знаходження в майбутньому.

Отже, суб’єктивний підхід до організації персональної інформації дозволяє створити систему, що краще відповідає індивідуальним потребам користувача.

1.5 Безпекові ризики цифрового накопичення

У статті «Digital Hoarding: The Rising Environmental and Personal Costs of Information Overload» [2] автори підкреслюють, що хаотичне зберігання великих обсягів персональних цифрових даних створює серйозні ризики для приватності, особливо коли інформація розподілена між різними сторонніми сервісами й хмарними платформами. Забуті профілі можуть залишатися відкритими для доступу навіть після того, як користувач перестав ними користуватися, а помилки синхронізації часто призводять до помилкового спільного доступу до особистих фотографій чи документів.

Таким чином, використання хмарних платформ, попри їхні переваги, серед яких спільна робота із файлами та доступ до даних з будь-яких пристроїв, має значний недолік: користувачі віддають свої дані під контроль сторонніх провайдерів.

Щоб поєднати зручність хмари з контролем над особистою інформацією, дедалі більшої популярності набуває ідеологія local-first software [4]. Такі рішення зберігають дані безпосередньо на пристрої користувача, працюють швидше, оскільки не залежать від швидкості інтернету та доступності хмарного сервісу, мають кращу інтеграцію з платформами і підвищують приватність.

1.6 Висновки до розділу

Сучасні користувачі стикаються з низкою викликів під час організації й пошуку цифрової інформації. На основі проведеного в цьому розділі дослідження визначено наступні потреби користувачів й оптимальні способи їхньої реалізації (див. Таблиця 1).

Потреба користувача	**Типове рішення**	**Виклики**	**Що важливо реалізувати**
Класифікація файлів.	Ієрархія тек, рідше – тегування.	Теки обмежують файл однією категорію, а ручне тегування вимагає багато часу.	Автоматизація класифікації файлів з можливістю множинного присвоєння.
Швидке знаходження потрібного файлу.	Навігація по теках, рідше – пошук.	Не виявлено.	Можливість навігації в ієрархії тек і гнучкий пошук.
Гнучке групування файлів	Один файл – одна тека або тегування	Теки обмежують файл однією категорію, хоча ієрархічна структура є важливою.	Можливість віднесення файлу до кількох тек (поєднання переваг тек і тегів).
Власна структура файлів у спільних середовищах.	Групове або персональне управління інформацією	Неможливість створення структури, що задовольнить усіх учасників групи.	Персональне управлення інформацією.
Приглушення видимості менш важливих файлів.	Переміщення файлів або їх видалення.	«Демоція» без зміни контексту файлів.	Механізм приглушення без зміни контексту файлів.
Суб’єктивний підхід до організації.	Групування за проектами.	Суб’єктивність критеріїв групування.	Залучення користувача до автоматизованих процесів.

Потреба користувача	**Типове рішення**	**Виклики**	**Що важливо реалізувати**
Контроль над приватністю та безпекою.	Використання хмарних сервісів.	Втрата контролю, підвищений ризик витоку чутливої інформації.	Захищене local-first рішення з можливістю інтеграції в хмарне середовище.

Таблиця 1. Виявлені потреби користувачів і оптимальні шляхи реалізації

РОЗДІЛ 2. ОГЛЯД НАЯВНИХ РІШЕНЬ

На основі виявлених проблем і потреб користувачів у контексті організації персональних цифрових файлів виділено наступні критерії для порівняння наявних рішень (див. Таблиця 2).

Критерій порівняння	**Опис**
Підтримувані формати файлів.	Чи підтримує система різноманітні типи файлів (зображення, відео, документи, аудіо тощо)?
Простота інтерфейсу користувача.	Чи система інтуїтивно зрозуміла і зручна для персонального використання технічно-недосвідченим користувачем?
Підтримка операційних систем.	Чи сумісне рішення із платформами Windows, macOS, Linus, Android, iOS?
Навігація та пошук	Чи надано механізми ефективного доступу до потрібних файлів?
Підвищення та зниження видимості файлів без зміни контексту.	Чи є можливість візуально виділити більш важливі файли і приховати менш важливі без їх переміщення з оригінальної локації?
Автоматична ієрархічна класифікація файлів.	Чи підтримується автоматичне інтелектуальне групування файлів у ієрархію тек?
Залучення користувача до прийняття автоматизованих рішень.	Чи враховує система зворотній зв’язок від користувача щодо роботи інтелектуальних функцій задля покращення їх роботи?
Сховище даних.	Чи забезпечує рішення зберігання файлів локально на пристрої користувача, у хмарі чи дозволяє обидва середовища?
Безпека даних.	Чи передаються дані у зашифрованому вигляді? Чи відповідає система стандартам безпеки?

Таблиця 2. Критерії порівняння наявних рішень

Для порівняльного аналізу обрано наступні рішення: Microsoft OneDrive у поєднанні з Microsoft 365 Copilot, Apple iCloud Drive у поєданні з Finder, IBM FileNet P8, TagSpaces та Organize.

2.1. Microsoft OneDrive у поєднанні з Microsoft 365 Copilot

Microsoft OneDrive разом із AI-помічником Microsoft 365 Copilot – це потужне й інтелектуальне хмарне рішення для зберігання й організації файлів [5]. Основна мета Microsoft 365 Copilot – скоротити час на рутинні завдання з пошуку та аналізу документів.

OneDrive має простий, інтуїтивно зрозумілий веб-інтерфейс, а також доступний для Windows, macOS, Android, iOS (проте Copilot наразі впроваджений тільки у веб-версії); файли зберігаються у хмарі з можливістю синхронізації на локальних пристроях.

OneDrive підтримує широкий спектр форматів файлів, серед яких документи, зображення, аудіо, відео тощо, проте Copilot наразі не працює з відео та зображеннями.

У OneDrive є можливість закріплювати важливі файли та додавати їх у “Вибране”, що дозволяє підвищити візуальну видимість важливих документів.

Серед можливостей Microsoft 365 Copilot: витягування інформації з кількох файлів без їх відкриття, порівняння кількох документів, створення коротких підсумків про документи як у текстовому, так і в аудіо-форматі, а також можливість поставити будь-яке запитання про файл, що знаходиться в системі (див.Рисунок 1).

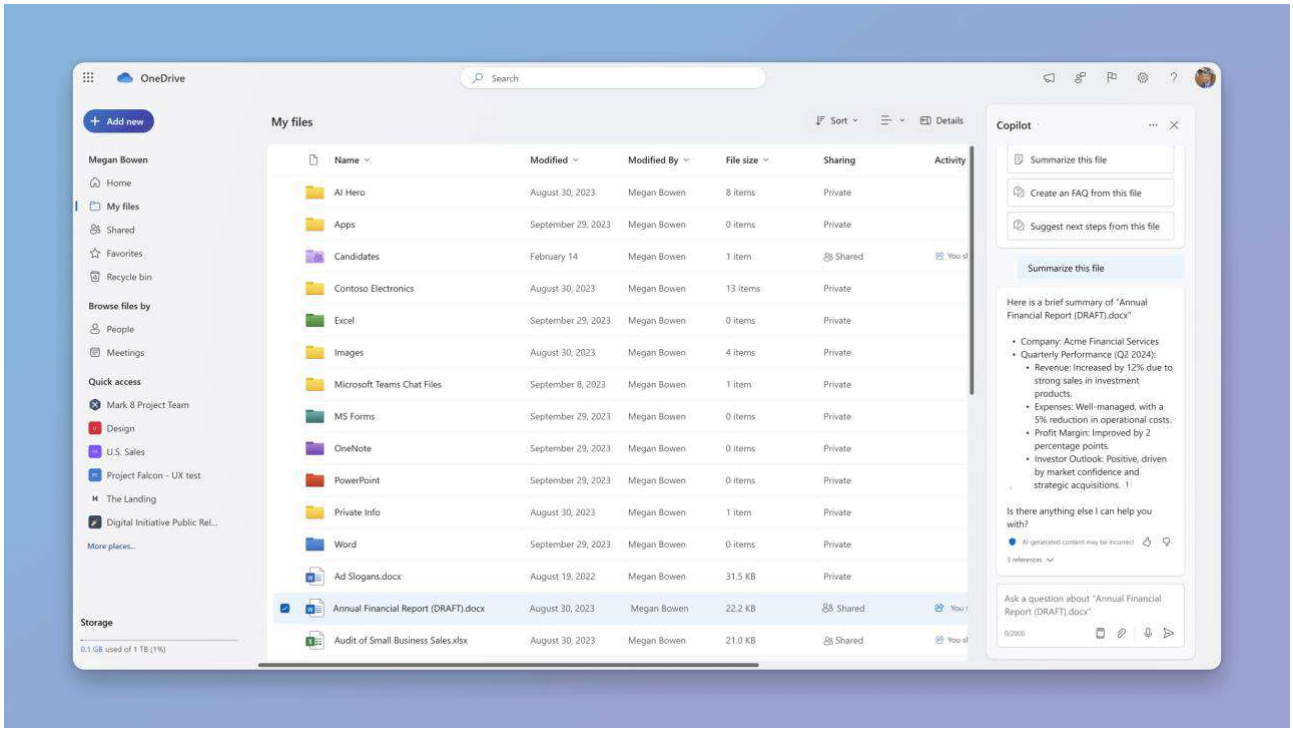


Рисунок 1. Copilot

Окрім того, Microsoft 365 Copilot надає можливість користувачу за його запитом знаходити файли, що доступні йому в середовищі OneDrive (див. Рисунок 2).

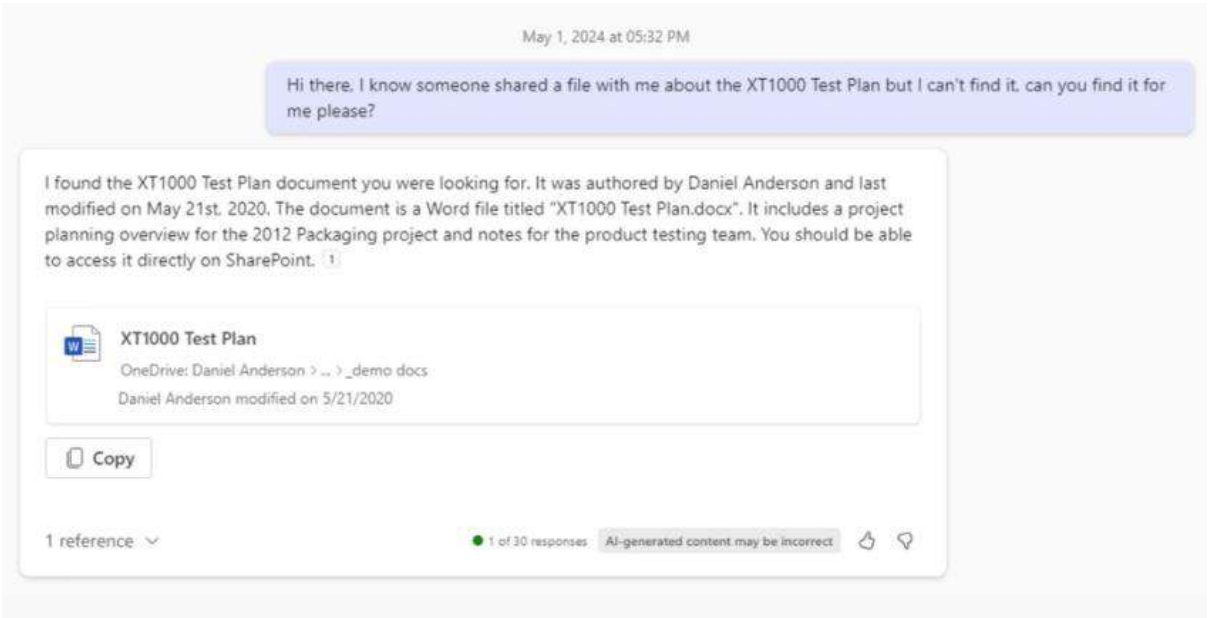


Рисунок 2. Чат з Copilot

Знаходження релевантних файлів за користувацьким запитом у Microsoft 365 Copilot реалізовано на основі лексичного та семантичного індексування файлів [6]. Система створює спеціальні структури (індекси), які містять інформацію про зміст файлів, зв’язки між ними, а також контекст їх використання. Коли користувач формулює запит, Microsoft 365 Copilot звертається до цих індексів, аналізує не лише ключові слова, а й сенс запиту, і знаходить найбільш релевантні документи.

Окрім того, Microsoft 365 Copilot може рекомендувати файли за їх актуальністю для користувача, на основі його останніх взаємодій або поточного проекту. Аналіз активності відбувається на основі даних з Microsoft Graph.

У OneDrive безпеку даних забезпечено їх шифруванням під час передачі (SSL/TLS) і зберігання (унікальні ключі генеруються для кожного файлу) і суворим контролем доступу до інформації [8].

2.2. Finder у поєднанні з Apple iCloud Drive

Finder у поєднанні з iCloud Drive – ще одне потужне рішення для зберігання файлів, інтегроване в екосистему Apple [9, 10]. Воно забезпечує зручний доступ до файлів на macOS, iOS та веб-платформі й орієнтоване на простоту використання та синхронізацію даних.

Finder та iCloud Drive підтримують усі основні типи файлів. Finder працює лише на macOS, але доступ до iCloud Drive можливий і через iOS, Windows, а також через веб-інтерфейс на будь-якій платформі. Інтерфейс користувача простий і добре знайомий для користувачів екосистеми Apple.

Finder надає потужний пошук Spotlight із підтримкою фільтрів, тегів і швидких дій, що дозволяє швидко знаходити файли за ім’ям, змістом, типом, датою чи тегом.

Система також надає механізм ручної зміни видимості файлів: вона дозволяє закріплювати файли у “Вибране”, застосовувати кольорові теги для швидкої ідентифікації важливого, а також приховувати файли.

Це рішення також забезпечує зручну організацію файлів за допомогою ручних інструментів, таких як Smart Folders (див. Рисунок 3) і теги, проте не надає можливості автоматизації процесу.

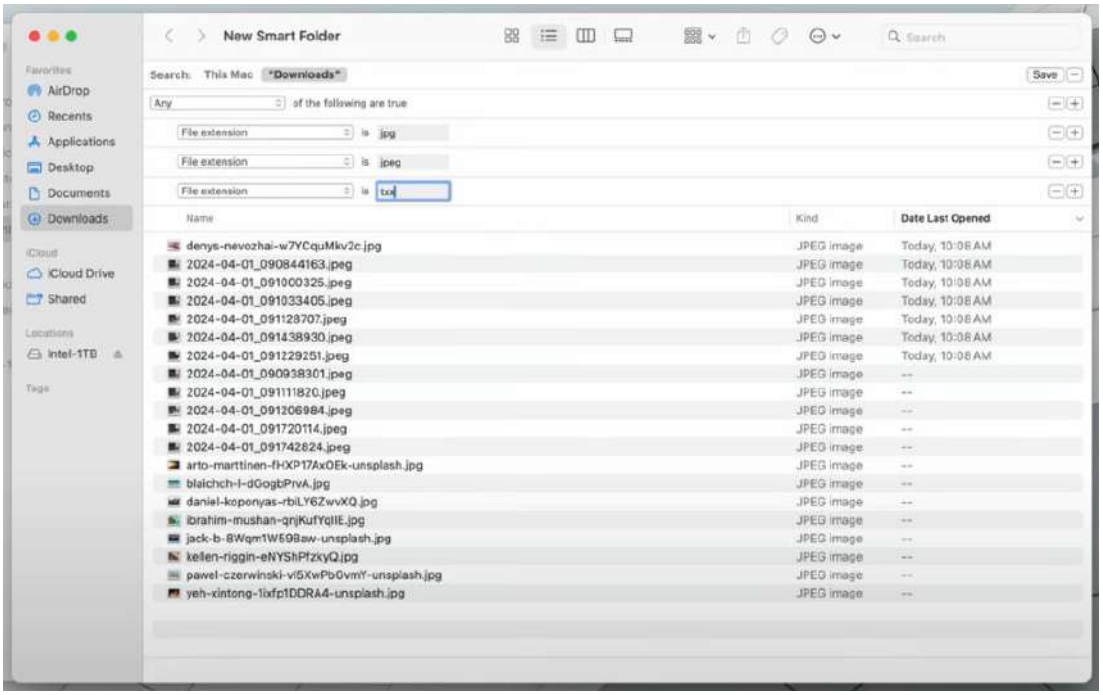


Рисунок 3. Finder. Розумні теки

Дані в iCloud Drive передаються й зберігаються у зашифрованому вигляді [12]. Apple використовує суворі політики доступу, а також двофакторну аутентифікацію для акаунтів, проте повний контроль над ключами шифрування має компанія Apple, а не кінцевий користувач.

2.3 IBM FileNet P8

IBM FileNet P8 – це корпоративна система управління документами, що надає потужне середовище для автоматичної організації документів [13]. Вона пропонує функції для класифікації документів, аналізу метаданих, управління життєвим циклом документів, і орієнтована на великий бізнес.

Ця система підтримує широкий спектр форматів файлів, зокрема документи, зображення аудіо, відео, а також спеціалізовані формати для

сканованих файлів чи CAD-документів. Система дозволяє розширювати підтримувані формати файлів під конкретні бізнес-процеси.

Інтерфейс FileNet P8 вимагає певного навчання і значно менш інтуїтивний, як у хмарних сервісах, орієнтованих на персональне використання (див. Рисунок 4).

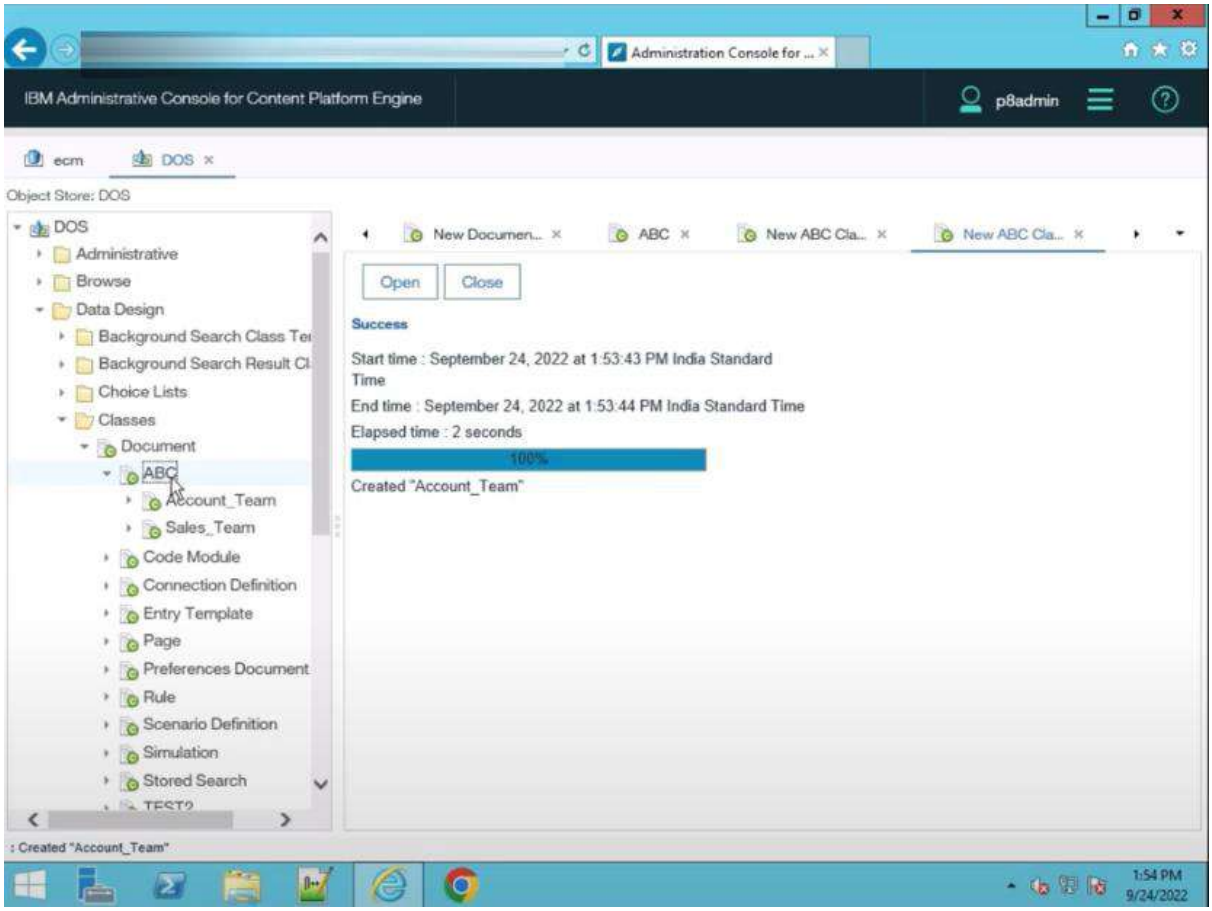


Рисунок 4. FileNet P8

FileNet P8 розгортається на серверному рівні (Windows Server, Linux), а доступ кінцевих користувачів забезпечено через браузер.

Ця система надає можливість гнучкого пошуку: повнотекстового, атрибутного, через фільтрацію за метаданими та за збереженими запитами.

IBM FileNet P8 дозволяє системі автоматично визначати тип документа й заповнювати його властивості на основі змісту (наприклад, через аналіз тексту чи структури файлу). Як результат, система здатна швидко й точно класифікувати файли. Після завантаження документа в систему класифікація відбувається

автоматично у фоні. Система дозволяє створювати користувацькі класифікатори, а «з коробки» надає тільки класифікатор для XML-файлів.

FileNet P8 також має надійну систему захисту, а саме забезпечує шифрування даних при передач, шифрує дані на рівні сховища, розмежовує доступ та інтегрується з корпоративними системами ідентифікації.

2.4 TagSpaces

TagSpaces – це локальний менеджер файлів з відкритим кодом, що дозволяє надавати файлам теги як вручну, так і автоматично за допомогою ШІ [14].

Цей застосунок підтримує майже всі основні формати файлів і надає можливість попереднього перегляду й базового редагування багатьох типів файлів (зокрема, текстових, зображень, PDF), а також прослуховування аудіо й перегляд відео всередині програми.

TagSpaces працює на Windows, macOS, Linux, а також підтримує веб-версію для самостійного розгортання. Цей застосунок має offline-first дизайн, завдяки чому за замовченням не відбувається передача користувацьких файлів у сторонні сервіси чи хмару. За бажанням можна підключити будь-яке S3-сумісне хмарне сховище. Інструменти для синхронізації та шифрування користувач обирає самостійно (напиклад, Dropbox, Nextcloud, Cryptomator тощо).

Додаток має інтуїтивний і сучасний інтерфейс, і надає такі інструменти як. перетягування файлів, додавання тегів, кольорове маркування, створення нотаток та списків (див. Рисунок 5). Система також надає швидкий пошук за загальними характеристиками файлів, а pro-версія також оснащена повнотекстовим пошуком за вмістом файлів.

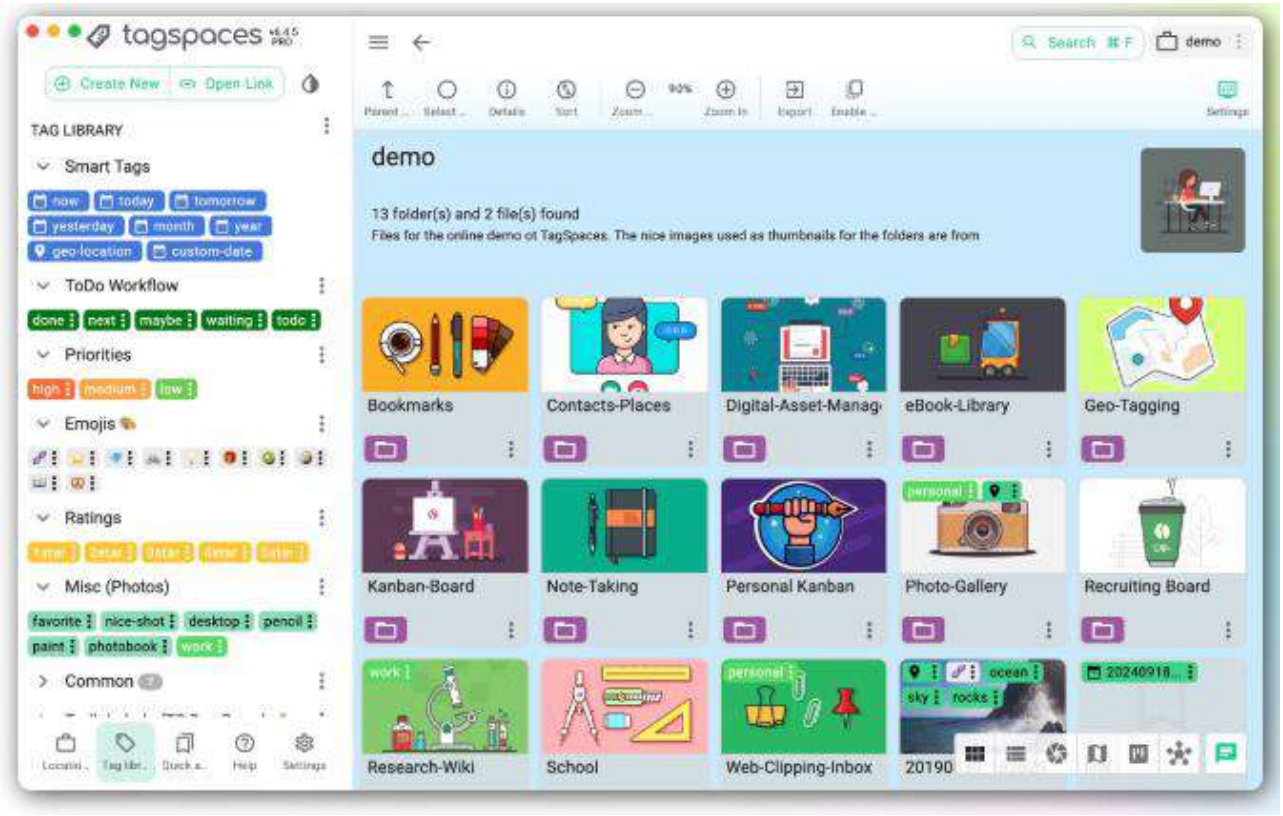


Рисунок 5. TagSpaces

Класифікація файлів у базовій версії здійснюється вручну через теги та структуру папок, водночас у pro-версії та з підключеним локальним AI через провайдер Ollama можливе автоматичне генерування тегів, описів і коротких підсумків для текстових файлів і зображень (див.Рисунок 6) [15].

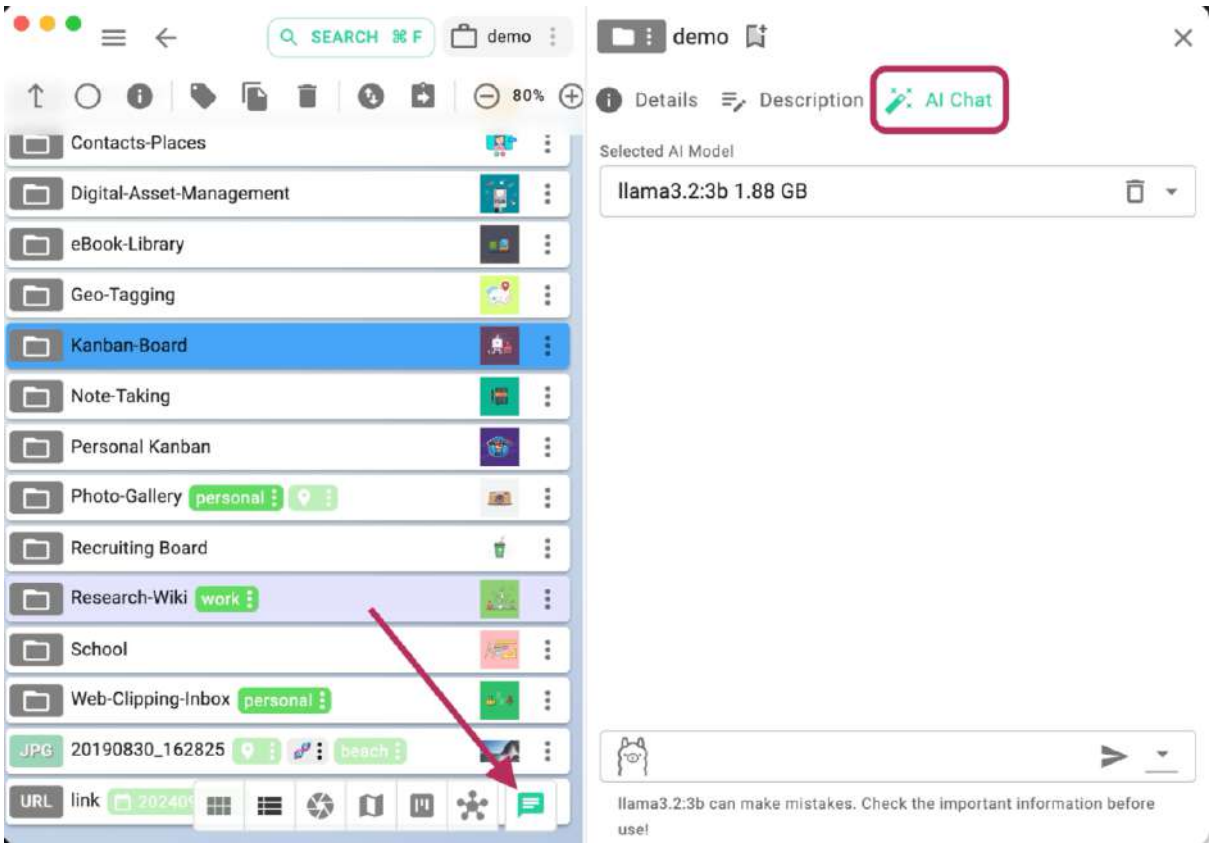


Рисунок 6. ШІ-можливості TagSpaces

2.5 Organize

Organize – це локальний інструмент для автоматизації організації файлів з відкритим кодом, що надає можливість гнучкої персоналізації за допомогою задання користувачем правил, на основі яких відбуватиметься категоризація файлів.

Ця утиліта працює на Windows, macOS і Linux і не має графічного інтерфейсу (див. Рисунок 7), що є незручним для технічно-недосвідчених користувачів. До того ж, налаштування відбувається через Python-скипти, що також вимагає від користувача технічних знань та вмінь.

Command line interface

```
organize - The file management automation tool.

Usage:
  organize run   [options] [<config>]
  organize sim   [options] [<config>]
  organize new   [<config>]
  organize edit  [<config>]
  organize check [<config>]
  organize debug [<config>]
  organize show  [--path|--reveal] [<config>]
  organize list
  organize docs
  organize --version
  organize --help

Commands:
  run         Organize your files.
  sim         Simulate organizing your files.
  new         Creates a new config.
  edit        Edit the config file with $EDITOR.
  check       Check whether the config file is valid.
  debug       Shows the raw config parsing steps.
  show        Print the config to stdout.
                Use --reveal to reveal the file in your file manager
                Use --path to show the path to the file
  list        Lists config files found in the default locations.
  docs        Open the documentation.

Options:
  <config>                          A config name or path to a config file
  -W --working-dir <dir>            The working directory
  -F --format (default|jsonl)       The output format [Default: default]
  -T --tags <tags>                  Tags to run (eg. "initial,release")
  -S --skip-tags <tags>             Tags to skip
  -h --help                         Show this help page.
```

Рисунок 7. Organize

Organize працює з будь-якими файлами, а для деяких типів, зокрема PDF, DOCX, та зображень підтримує аналіз їхнього вмісту для більш точної автоматизованої класифікації.

Автоматична класифікація файлів є основною функцією цього рішення. Файли сортуються, переміщуються, перейменовуються, видаляються чи позначаються тегами за заданими правилами, що засновані на імені, розширенні, EXIF-даних, вмісті PDF/DOCX, тощо. Також є піжтримка використання регулярних виразів, Python- та shell-скриптів для складних сценаріїв.

Наприклад, автоматизацію створення теки з чеками у PDF форматі на основі файлів розташованих у теці Downoalds можна реаліазувати за допомогою визначення наступного правила (див. Рисунок 8) [17].

```
rules:
  - name: "Sort my invoices and receipts"
    locations: ~/Downloads
    subfolders: true
    filters:
      - extension: pdf
      - name:
          contains:
            - Invoice
            - Order
            - Purchase
          case_sensitive: false
    actions:
      - move: ~/Documents/Shopping/
```

Рисунок 8. Користувацькі правила в Organize

Перевагою цього рішення є повний контроль користувача над автоматизацією через редагування YAML-конфігурацій. Також є можливість запуску симуляції командою sim (див. Рисунок 7), щоб побачити, як виглядатиме результуюча структура файлів, і лише потім виконати зміни. Всі дії можна перевірити, змінити чи відмінити в разі необхідності.

2.6 Висновки до розділу

З урахуванням сформованих критеріїв у цьому розділі проведено порівняльний аналіз п’яти програмних рішень, які вирішують подібні завдання до визначених для майбутньої системи. Розглянуто функціональні можливості Microsoft OneDrive з Copilot, Apple iCloud Drive у зв’язці з Finder, IBM FileNet P8, TagSpaces та Organize щодо зберігання та організації файлів.

Жоден із розглянутих застосунків не задовольняє всіх встановлених критеріїв (див. Таблиця 3), що підкреслює необхідність розробки власного рішення, яке відповідатиме всім зазначеним вимогам.

Критерій порівняння	**OneDrive + Copilot**	**iCloud Drive + Finder**	**IBM FileNet P8**	**Tag Spaces**	**Organize**
Підтримувані формати файлів	OneDrive: Усі основні формати Copilot: тільки текстові й зображення	Усі основні формати	Усі основні формати, можливість додавання нових	Усі основні формати	Будь-які формати
Простота інтерфейсу користувача	Простий	Простий, дизайн звичний як для рішень від Apple	Складний для технічно недосвідчених користувачів	Простий	CLI, немає GUI
Підтримка операційних систем	Win, macOS, Android, iOS, Web	macOS (Finder), iOS, Win, Web	Windows Server, Linux, Web-клієнт	Win, macOS, Linux, Web, Android (частк.)	Win, macOS, Linux
Навігація та пошук	Швидкий пошук, фільтри, AI-пошук	Spotlight, теги, Smart Folders	Розвинений пошук, фільтри, атрибути	Пошук за тегами, назвами, повнотекстовий	Пошук за правилами у конфігурації
Підвищення/зниження видимості файлів без зміни контексту	Вибране, закріплення	Вибране, теги, ручне приховування	Ролі, права доступу	Кольорові теги, вибране	Переміщення у спец. папки/теги
Автоматична ієрархічна класифікація файлів	Частково, ШІ-підсумки, рекомендації щодо файлів	Ні, тільки ручна організація	Так, автоматичне групування за змістом	Pro: ШІ-теги/описи, але не теки	Так, правила за іменем, вмістом,

Критерій порівняння	OneDrive + Copilot	iCloud Drive + Finder	IBM FileNet P8	Tag Spaces	Organize
					метадани ми
Залучення користувача до прийняття автоматизовани х рішень	ШІ враховує активність та конкретні запити користувача	Тільки ручна організація	Через користуваць кі класифікато ри	Через запити до ШІ-чату	Повний контроль через правила
Сховище даних	Хмара, синхронізація з локальним	Локально + хмара (iCloud)	Локально, хмара, гібридне	Локально, S3-хмара	Тільки локально
Безпека даних	Шифрування при передачі/збері ганні	Шифрування, 2FA, контроль Apple	Шифруванн я, корпоративн і політики	Є можливість інтеграції інструменті в шифруванн я	Локально, залежить від ОС

Таблиця 3. Порівняння наявних ріщень

РОЗДІЛ 3. ПРОЕКТУВАННЯ ВЛАСНОГО РІШЕННЯ

На основі аналізу актуальних проблем в організації персональної цифрової інформації та порівняльного аналізу наявних рішень на ринку сформульовано такі технічні вимоги до розроблюваного програмного застосунку:

1. **Інтуїтивно зрозумілий інтерфейс користувача.** Система повинна бути максимально простою у використанні, щоб її могли легко опанувати навіть технічно недосвідчені користувачі.
2. **Кросплатформеність.** Задля охоплення широкої аудиторії програмний продукт повинен бути доступним на основних десктопних операційних системах.
3. **Інтелектуальна ієрархічна класифікація файлів за користувацьким запитом**. Система повинна забезпечувати автоматизоване створення ієрархії тек на основі метаданих і вмісту файлів у системі відповідно до запитів користувача, що дозволить йому не витрачати час на ручну організацію файлів.
4. **Можливість впливати на автоматизовані процеси**. Система повинна надавати інструменти для корегування результатів автоматичного групування та класифікації файлів, щоб максимально персоналізувати систему під потреби користувача.
5. **Підтримка множинної ієрархічної категоризації файлів.** Система повинна поєднувати переваги ієрархічної структури тек та гнучкість тегування, а отже дозволяти додавати один файл до кількох категорій одночасно і водночас не перевантажувати інтерфейс альтернативними інструментами.
6. **Управління видимістю файлів за важливістю.** Необхідно впровадити механізм для зниження візуальної видимості менш важливих файлів без зміни місця їх зберігання, що допоможе зменшити інформаційне навантаження.

7. **Надання зручного пошуку і навігації між файлами.** Система повинна надавати інструменти для ефективного пошуку та просту навігацію у межах створеної структури файлів.

8. **Безпечне зберігання даних.** Користувацькі дані повинні надійно зберігатись і оброблятись локально без передачі у сторонні чи хмарні сервіси. Водночас необхідно передбачити можливість безпечної інтеграції з хмарними сервісами.

Відповідно до визначених вимог необхідно спроектувати власне рішення, а саме: розробити структурну схему застосунку, ER-модель бази даних, архітектуру програмної частини та макет користувацького інтерфейсу.

3.1 Структурна розробка застосунку

Розроблювана система повинна складатись із таких компонентів: програмного забезпечення, бази даних та безпечного сховища. Взаємодію та розміщення цих компонентів проілюстровано на структурній схемі (див. Рисунок 9).

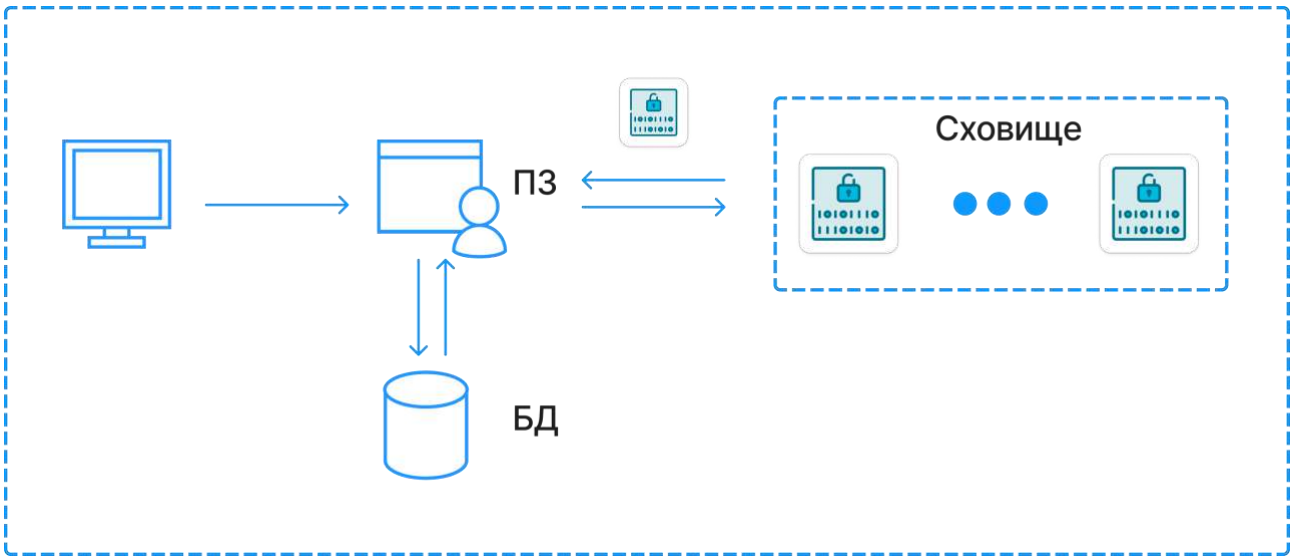


Рисунок 9. Структурна схема застосунку

Нижче поетапно описано роботу системи від моменту її встановлення до збереження даних.

Спершу користувач завантажує інсталяційний пакет відповідно до його операційної системи й встановлює застосунок на персональному комп'ютері. Після завершення інсталяції система автоматично ініціалізує локальну базу даних.

Під час першого запуску застосунку користувач проходить реєстрацію. Облікові дані користувача (логін та пароль, а також за потреби інші дані) будуть використані для ініціалізації персонального сховища даних і збережені у зашифрованому вигляді.

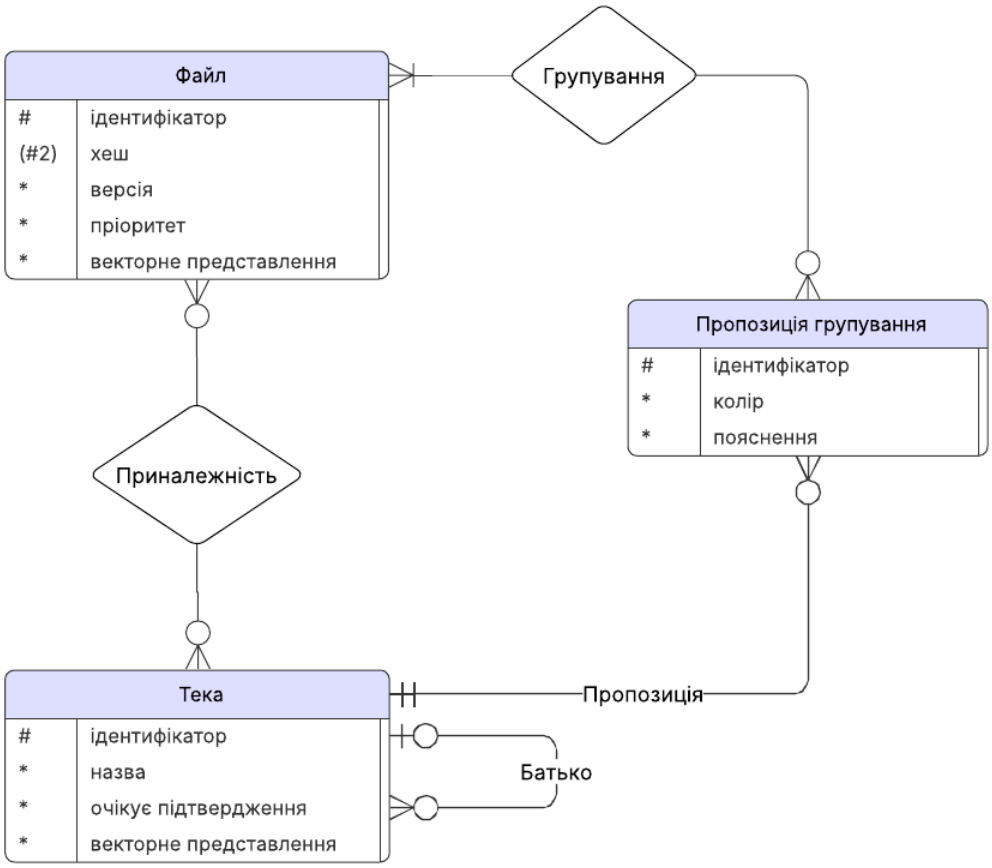
Після успішної реєстрації користувач обирає розміщення сховища. Локальне збереження можливе на диску ПК або на мережевому накопичувачі (NAS). Як альтернатива, дані можуть бути збережені в хмарному сховищі на вибір користувача. Передача даних між програмною частиною застосунку і сховищем і їх зберігання відбувається у зашифрованому вигляді. Ключі шифрування також розміщені у сховищі.

База даних у розроблюваній системі призначена для зберігання інформації про доменні сутності системи та взаємозв'язки між ними. Вміст файлів і їхні метадані розміщені у сховищі, тоді як додаткова інформація про файл та посилання на нього зберігаються в базі даних. Важливо забезпечити узгодженість між записами у базі даних та файлами у сховищі, щоб уникнути появи так званих «осиротілих файлів» – тобто таких файлів, що фізично існують у сховищі, проте не мають відповідних записів у базі даних і, відповідно, програмна частина системи їх не відображає.

Для вирішення цієї проблеми доцільно реалізувати механізм періодичної перевірки цілісності сховища та бази даних. Зокрема, можна впровадити регулярне порівняння переліку файлів у сховищі з відповідними записами у базі даних, виявляти «осиротілі файли» та пропонувати їх видалення чи відновлення інформації про них у системі. Такий підхід дозволить забезпечити цілісність даних і мінімізувати ризик накопичення неврахованих файлів у сховищі.

3.2 Розробка концептуальної моделі бази даних

У процесі розробки концептуальної моделі бази даних для системи організації персональних файлів було виділено три сутності, що відображають ключові об’єкти предметної області та взаємозв’язки між ними: файл, теку й пропозицію групування (див.Рисунок 10).



Корпоративні обмеження цілісності:

1. Екземпляр сутності "Тека", що очікує підтвердження користувача, не може бути у зв’язку "Приналежність".
2. Атрибут "Назва" для сутності "Тека" повинен бути унікальним для тек зі спільною батьківською текою.

Рисунок 10. Концептуальна модель бази даних

3.2.1. Сутність «Файл»

Сутність «Файл» репрезентує окремий файл, що зберігається у системі. Для кожного файлу фіксується унікальний ідентифікатор, який також буде використано для зберігання вмісту та метаданих самого файлу в об’єктному сховищі. З метою уникнення дублювання даних у базі зберігається хеш вмісту файлу, що дозволяє запобігти завантаженню декількох ідентичних файлів у систему. Атрибут «пріоритет» визначає рівень важливості файлу та впливає на його візуальну видимість для користувача. Окрім цього, для кожного файлу передбачено збереження його векторного представлення на основі вмісту та метаданих, що забезпечить можливість реалізації ефективного пошуку семантично схожих файлів для автоматизації процес категоризації файлів за користувацьким запитом.

3.2.2 Сутність «Тека»

Сутність «Тека» забезпечує ієрархічну організацію файлів у системі. Кожна тека має унікальний ідентифікатор та назву, яка є унікальною в межах спільної батьківської теки. Файли можуть належати до кількох тек одночасно, що дозволяє реалізувати механізм множинної ієрархічної категоризації без фізичного дублювання даних. Додатково, для тек передбачено атрибут «очікування на підтвердження», який позначає автоматично згенеровані системою теки, що потребують затвердження користувачем перед встановленням зв’язків із файлами. Як і у випадку з файлами, для тек передбачено атрибут для векторного представлення для реалізації автоматизованої категоризації файлів.

3.2.3 Сутність «Пропозиція групування»

Сутність «Пропозиція групування» створюється системою, щойно оброблено запит користувача на автоматичне групування та категоризацію файлів. Вона стосується певної теки та одного чи декількох файлів, які є

найближчими за векторною відстанню до тих, які були зазначені користувачем у запиті. Кожна така пропозиція має унікальний ідентифікатор, містить випадково обраний колір для візуального виділення запропонованих файлів у графічному інтерфейсі, а також пояснення, що описує причину формування відповідної пропозиції.

3.3 Архітектура програмної частини застосунку

У розробці програмної системи важливим етапом проектування є розробка достатньо гнучкої для розширення архітектури. Важливо обрати такий підхід, що дозволить додавати нові можливості в програмний продукт без необхідності вносити значні зміни в наявну кодову базу.

3.3.1 Вибір архітектури

За основу в розроблюваній системі взято чисту архітектуру (Clean Architecture), описану Робертом Мартіном у його однойменній книзі [18]. Вибір цього архітектурного патерну зумовлений його численними перевагами.

По-перше, чиста архітектура забезпечує *чітке розділення відповідальностей*, а отже кожен компонент системи має власну зону відповідальності, завдяки чому зміни, внесені в реалізацію одних не призведуть до необхідності змінювати інші компоненти системи.

По-друге, вона забезпечує *незалежність системи від конкретних технологій*. Як результат, бізнес-логіка застосунку жодним чином не залежить від фреймворків, бібліотек, бази даних чи будь-яких інших деталей реалізації.

Окрім того, ця архітектура дозволяє розробити систему, *гнучку до розширення*: додавання нових можливостей не призводить до змін основної логіки.

Також вона сприяє легкому покриттю бізнес-логіки *юніт-тестами* і *простій заміні технологій*, оскільки вона не залежить від зовнішньої

інфраструктури програми, і її можна легко заміняти альтернативними інструментами.

Основний принцип, що лежить в основі цієї архітектури, – інверсія залежностей, що дозволяє більш абстрактним компонентам не залежати напряму від деталей реалізації, оскільки зв'язок здійснюється через інтерфейси.

Чисту архітектуру зображають у вигляді концентричних кіл, де кожне внутрішнє коло є більш критичним і менш залежним, що мінімізує вплив зовнішніх технологій на ядро системи (див. Рисунок 11). Вона містить чотири шари: сутності (Entities), варіанти використання (Use Cases), адаптери інтерфейсу (Interface Adapters) і фреймворки та драйвери (Frameworks and Drivers).

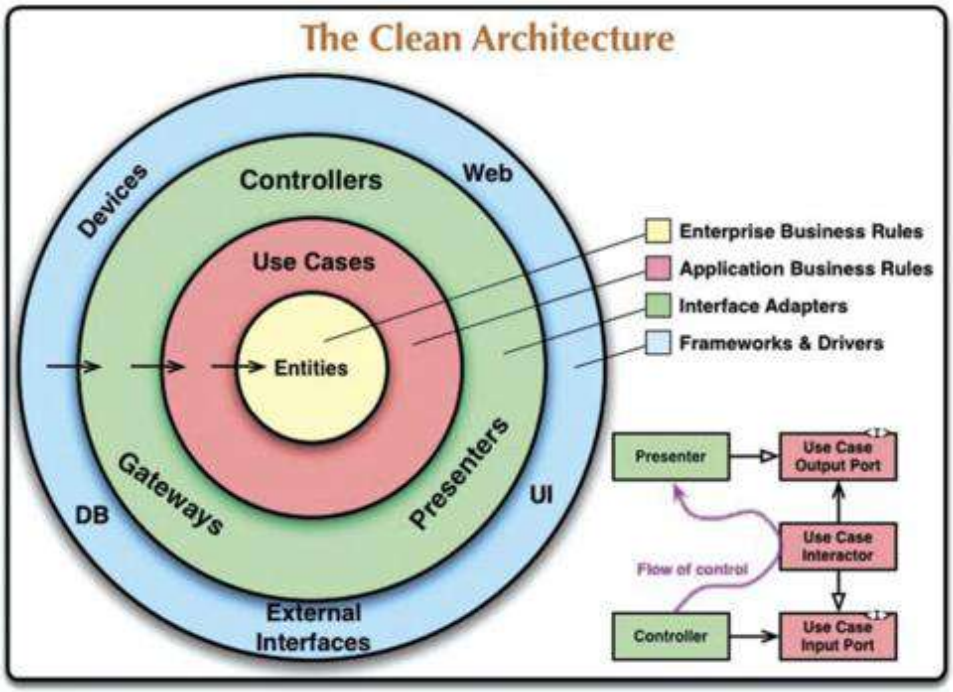


Рисунок 11. Діаграма чистої архітектури

Сутності – це найбільш абстрактні компоненти системи, що визначають критичні-бізнес дані.

Варіанти використання – це критично важливі бізнес-правила, що визначають як система має реагувати на дії. Вони координують сутності і не залежать від конкретних технологій.

Адаптери інтерфейсу – це шар, що забезпечує комунікацію між абстрактною логікою і деталями реалізації.

Фреймворки та драйвери – це конкретні інструменти та технології, що реалізують інфраструктуру застосунку, та можуть бути легко замінені.

Отже, чиста архітектура надає переваги, яких, наприклад, не має класична шарова архітектура, де верхні шари напряму залежать від нижчих, тобто бізнес-логіка залежна від конкретної реалізації шару даних, що ускладнює заміну або тестування окремих компонентів. Вибір більш гнучкого підходу є особливо важливим для проєктів, що планують довготривалий життєвий цикл та можливості розширення.

3.3.2. Застосування обраної архітектури до розроблюваної системи

Для розроблюваної системи за основу взято принципи чистої архітектури, однак із певними спрощеннями. Зокрема, бізнес-логіка реалізована без окремого шару варіантів використання, який у класичному варіанті чистої архітектури призводить до створення великої кількості класів, кожен з яких є абстракцією над однією операцією репозиторію. Хоча такий підхід забезпечує максимальну гнучкість та ізольованість, на практиці він значно ускладнює та сповільнює розробку, особливо на початкових етапах.

У запропонованій схемі (див.Рисунок 12) інтерфейси репозиторіїв містять визначення бізнес-логіки, до якої контролери з презентаційного шару мають прямий доступ. Таким чином, контролери все ще залежать від абстракцій, а не від конкретних реалізацій, що дозволяє дотримуватись основного принципу чистої архітектури – інверсії залежностей.

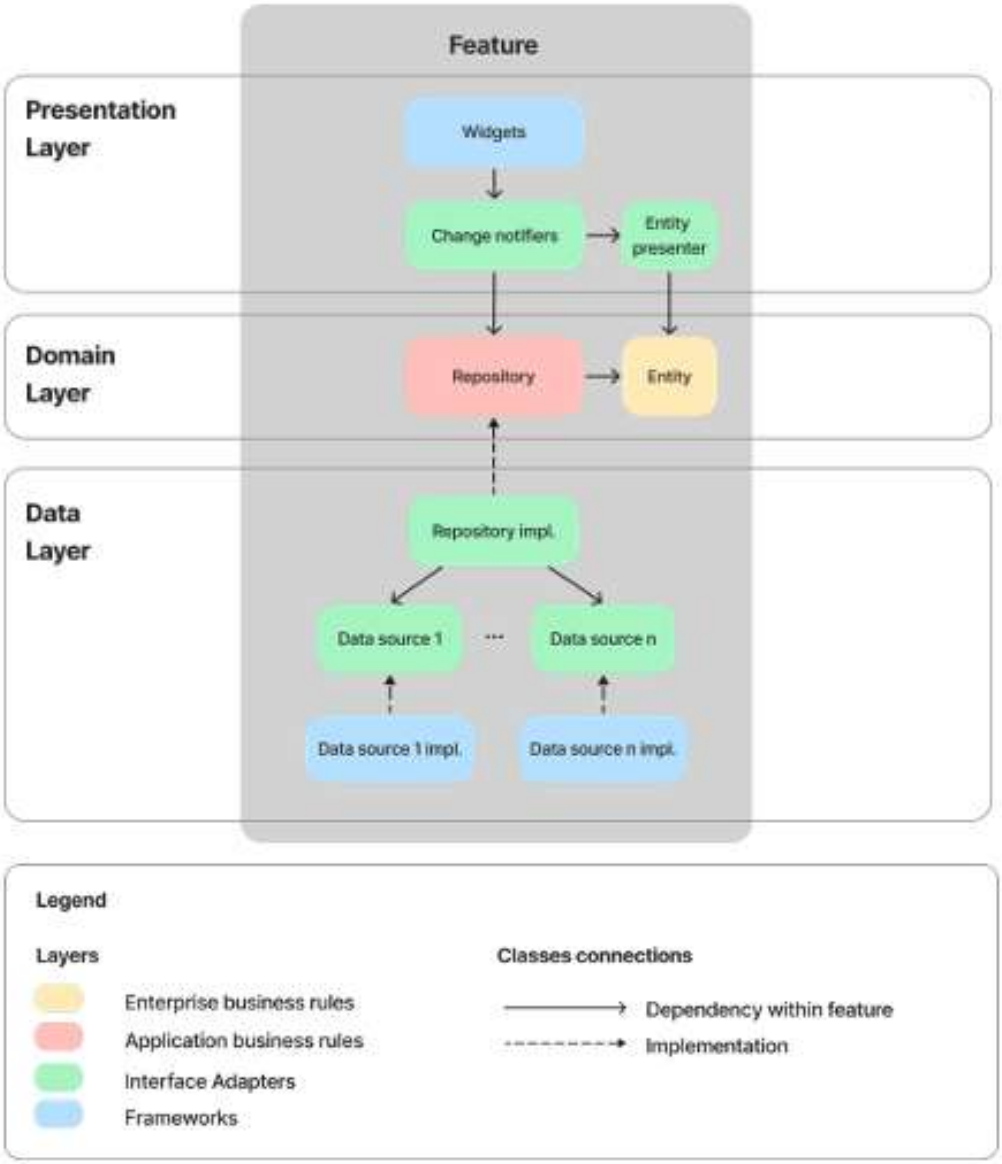


Рисунок 12. Схема архітектури розроблюваного рішення

Ще одне спрощення полягає в об’єднанні шарів бізнес-правил та бізнес-даних у доменний шар, що дозволить простіше структурувати проект на рівні папок, і водночас зберігає чітке розділення відповідальностей.

Таки чином, кожну функцію системи поділено на шар з доменною логікою, шар доступу до даних та шар з презентаційною логікою.

Доменний шар містить абстракції критичних бізнес-даних (Entity) та критичних бізнес-правил, що визначені як методи інтерфейсу репозиторію (Repository).

Шар доступу до даних містить реалізацію інтерфейсу репозиторію і залежить від принаймні одного абстрактного джерела даних. У цьому шарі також розміщені конкретні реалізації джерел даних.

Шар з презентаційною логікою надає конкретні реалізації елементів інтерфейсу користувача, перетворювач сутності для її відображення на користувацькому інтерфейсі (Presenter), а також сповіщувач змін (Change notifier), який забезпечує актуалізацію даних між презентаційним шаром і шаром доступу до даних.

У розроблюваній системі визначено 5 основних компонентів системи: управління теками (Folder management), управління файлами (File management), помічник з організації (Organizing assistant), аналізатор запитів (Prompt analyzer) та генератор векторних представлень (Embedding generator).

Діаграму, що ілюструє структуру кожного з компонентів відповідно до обраної архітектури, а також взаємодію між цими компонентами, наведено в Додатку А. Нижче надано пояснення щодо призначення кожного з компонентів.

Компонент *управління теками* відповідає за створення, редагування, видалення, оновлення та ієрархічну організацію тек.

Компонент *управління файлами* забезпечує додавання, зберігання, обробку й видалення файлів у системі.

Помічник з організації автоматизує процеси групування та категоризації файлів відповідно до потреб користувача.

Аналізатор запитів обробляє запити користувача та визначає його потреби щодо організації файлів.

Генератор векторних представлень створює векторні представлення для різних типів даних (текстових та мультимедійних).

З діаграми видно, що компоненти, яким потрібні обчислення векторних представлень певних даних, залежать від абстрактного інтерфейсу клієнта, реалізація якого залежить від бізнес-логіки генератора векторних представлень. Взаємодія ж між іншими компонентами системи відбувається через їхні сповіщувачі змін.

3.4. Проектування користувацького інтерфейсу

Оскільки цільова аудиторія охоплює широке коло користувачів, у тому числі й технічно-недосвідчених, критично важливим є створення інтуїтивного інтерфейсу, можливості якого прості для опанування.

На рисунку (див. Рисунок 13) представлено орієнтовний вигляд головної сторінки застосунку, на якій зосереджено всі основні інструменти для організації файлів.

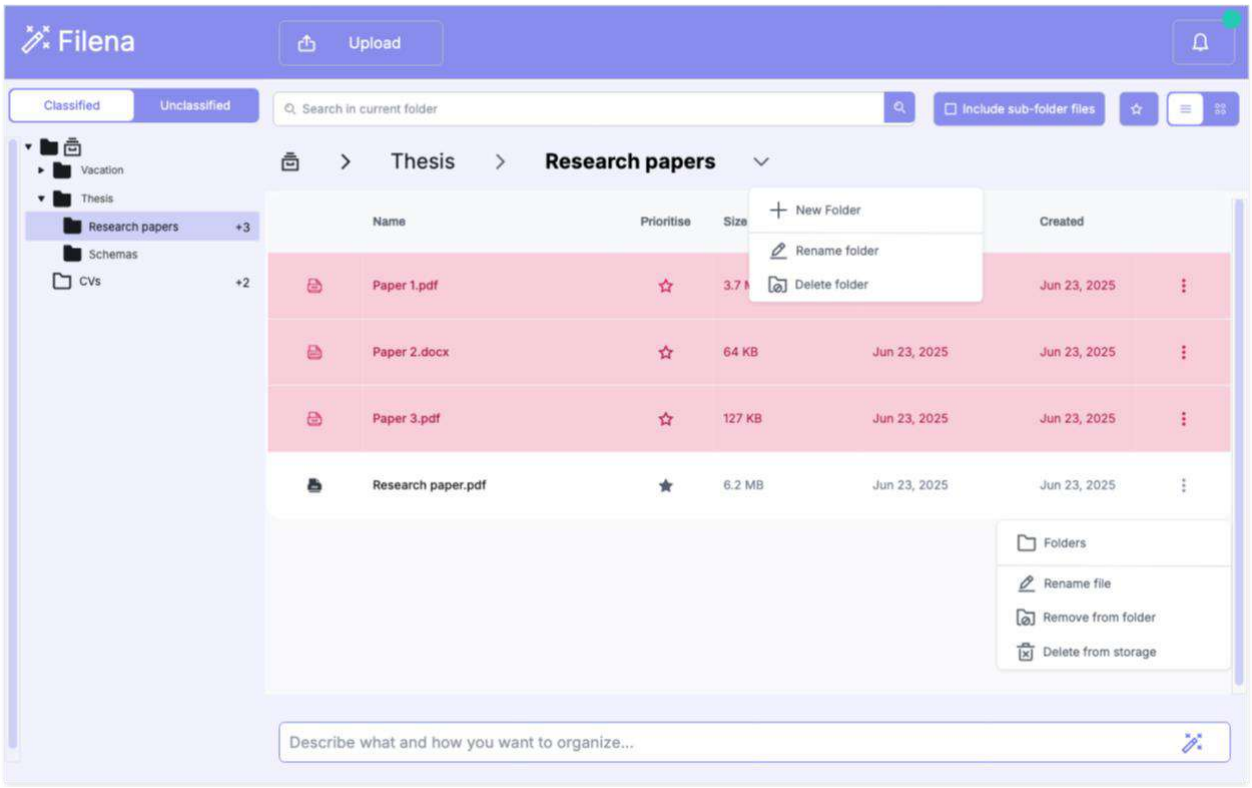


Рисунок 13. Головна сторінка розроблюваного застосунку

На цій сторінці розміщено кнопку для завантаження файлів до системи. Завантажені файли не прив’язуються до жодної теки.

Основна частина екрану призначена для відображення ієрархії тек, де користувач може переходити між теками, редагувати їхні назви, створювати нові

теки або видаляти наявні. Для зручності також відображено шлях до поточно обраної теки.

Усі файли, що знаходяться в обраній теці, відображено у вигляді таблиці з основною інформацією про кожен файл. Користувач має можливість підвищити важливість файлу, перейменувати його, видалити з поточної теки або повністю із системи, а також переглянути всі теки, у яких цей файл присутній, і додати його до існуючої теки (див. Рисунок 14). Додатково передбачено можливість показувати всі вкладені файли для обраної теки, що наближає використання тек до принципу тегів.

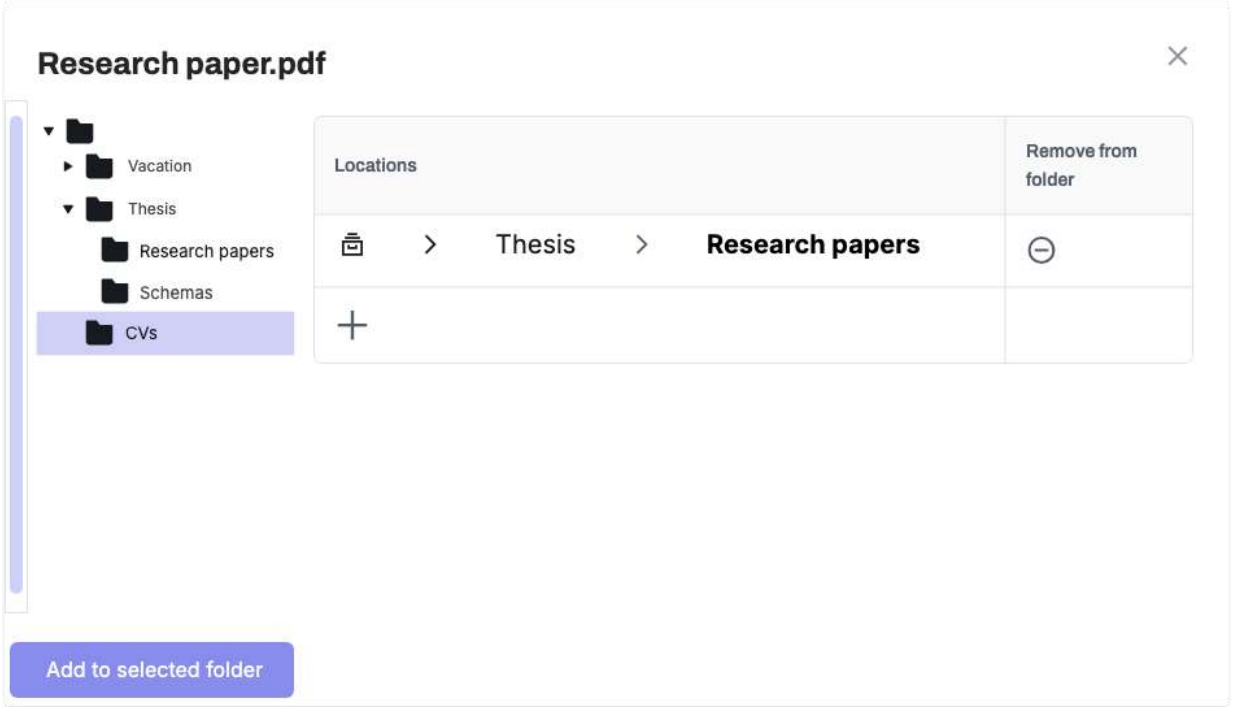


Рисунок 14. Вікно додавання файлу до тек

Для фільтрації інформації користувач може скористатися фільтром для файлів із підвищеною важливістю, що дозволяє приховати менш важливі файли і водночас зберегти доступ до них у природному контексті. Також передбачене поле для інтелектуального пошуку файлів.

Інтерфейс враховує також можливість перемикання виду відображення з табличного на плитковий, однак у межах даної роботи цей функціонал не реалізується.

Додатково реалізовано можливість перегляду файлів, що не прив’язані до жодної теки. Запропоновано групувати такі файли за типом і забезпечити доступ до них через зміну відображення дерева тек (кнопка-перемикач) (див. Рисунок 15).

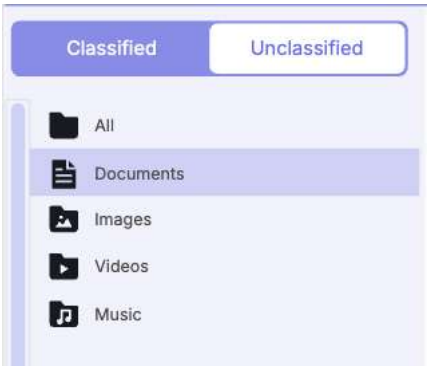


Рисунок 15. Структура некласифікованих файлів

Крім того, користувач може звернутися до системи із конкретним запитом щодо організації файлів. Після опрацювання такого запиту система формує пропозиції, в яких зазначає, до яких тек рекомендується перемістити відповідні файли. Користувач може прийняти або відхилити ці пропозиції (див.Рисунок 16).

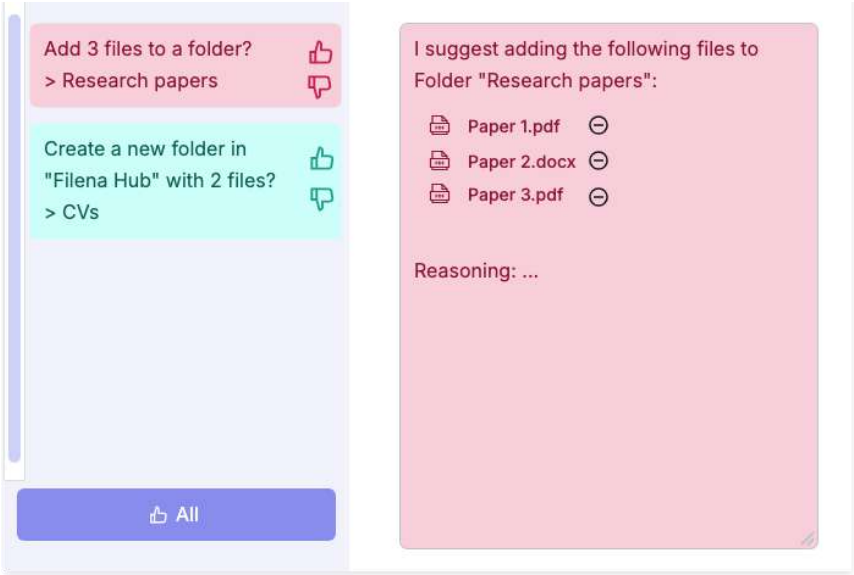


Рисунок 16. Вікно пропозицій групування

3.5 Висновки до розділу

У цьому розділі було визначено та детально обґрунтовано основні технічні вимоги до програмного застосунку для організації персональної цифрової інформації, а також запропоновано структурне, концептуальне та архітектурне рішення для його реалізації.

Проведено аналіз ключових завдань, які має вирішувати система. Серед них інтуїтивність інтерфейсу, кросплатформеність, інтелектуальна ієрархічна категоризація файлів, можливість впливу на автоматизовані процеси, підтримка множинної категоризації, управління видимістю файлів, ефективна навігація й безпечне зберігання даних.

Розроблено концептуальну модель бази даних, де виокремлено основні сутності системи: файл, теку й пропозицію групування, а також описано взаємозв’язки між ними.

Обґрунтовано вибір чистої архітектури як основи для розробки програмної частини застосунку, що забезпечує незалежність бізнес-логіки від технологічних рішень, спрощує розширення системи та підвищує її надійність і тестованість.

Також було приділено увагу проектуванню користувацького інтерфейсу, який має бути інтуїтивно зрозумілим для широкого кола користувачів, а також забезпечувати доступ до основних функцій організації, навігації та управління файлами.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ВЛАСНОГО РІШЕННЯ

4.1 Сховище даних

Вибір сховища даних для розроблюваної системи зумовлений вимогами до безпеки, продуктивності, а також можливості як локального розгортання, так й інтеграції в хмару.

Для досягнення цих цілей обрано MinIO – кросплатформене, надійне й ефективне об’єктне сховище, що має низку важливих переваг для розроблюваного програмного рішення [19]. Воно зберігає файли у вигляді об’єктів з метаданими, забезпечує їх швидку обробку та надає вбудовані механізми для підтримки цілісності й безпеки, серед яких автоматичне виявлення й відновлення пошкоджених даних, а також надійне шифрування під час зберігання й передачі. MinIO також підтримує версіонування, резервне копіювання та синхронізацію даних, а сумісність із Amazon S3 API спрощує інтеграцію з хмарними платформами.

Окремою і важливою перевагою MinIO є запропонована альтернатива класичному S3 API: *Prompt API*, який замінює традиційні PUT і GET на PUT і PROMPT [20, 21]. Це інноваційне рішення дозволяє отримувати структуровану інформацію про об’єкти напряму із запитів до них. Така можливість є особливо корисною для автоматизованої категоризації файлів, в основі якої лежить аналіз вмісту та метаданих об’єктів.

На ілюстрації (див. Рисунок 17) наведено приклад запиту до зображення із чеком і відповіді від сервера, що містить інформацію про цей об’єкт у структурованому форматі.

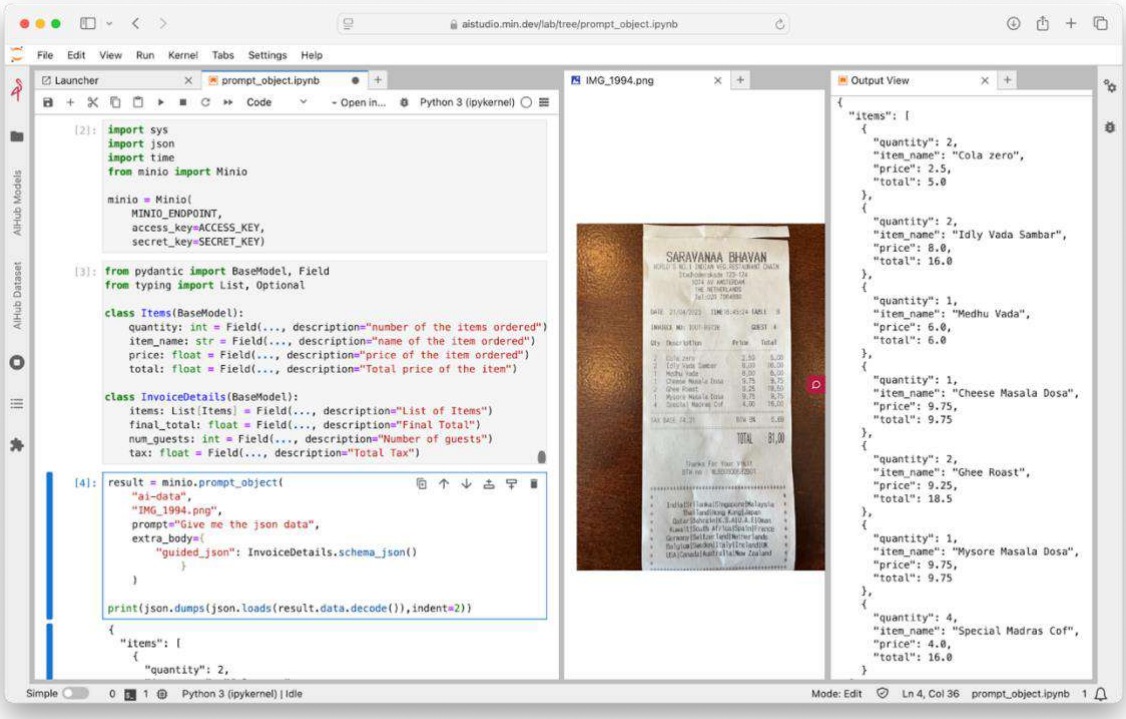


Рисунок 17. MinIO. Prompt API

Використання Prompt API дозволить отримувати необхідну інформацію про вміст об’єктів напряму зі сховища, без необхідності аналізу іншими сервісами. Наразі Prompt API працює лише з текстовими даними та зображеннями, проте розробники MinIO обіцяють реалізувати її також для відео та аудіо форматів.

Prompt API має ще одне обмеження: хоча сховище MinIO є безкоштовним, ШІ-можливості системи наразі доступні лише за передплатою, а отже для розроблюваної системи буде використано альтернативні інструменти для аналізу вмісту файлів.

Отже, MinIO повністю відповідає визначеним вимогам до сховища даних, адже забезпечує високу продуктивність, надійність, безпеку та гнучкість інтеграції. Додатковою перевагою цього рішення є вбудовані механізми обробки природньої мови та аналізу вмісту файлів, що особливо корисно в для реалізації автоматизованої категоризації та інших інтелектуальних функцій системи.

4.2 База даних

Оскільки розроблюваний застосунок повинен забезпечувати ефективне групування файлів у теки на основі користувацьких запитів, необхідно передбачити зберігання векторних представлень сутностей, а також можливість здійснення швидкого пошуку за цими представленнями. Тому вибір бази даних повинен гарантувати підтримку такого функціоналу, що забезпечить високу продуктивність, масштабованість і зручність роботи з векторними даними.

Для виконання цих вимог обрано кросплатформену NoSQL-базу даних ObjectBox, яка не тільки дозволяє зберігати векторні представлення об'єктів разом із самими об'єктами, а також має такі низку переваг над іншими популярними базами даних [22].

Так, наприклад, ObjectBox суттєво випереджає SQLite за продуктивністю. Це особливо важливо для роботи з великими обсягами даних і здійснення пошуку за схожістю. Окрім того, ця база даних гарантує цілісність, узгодженість, ізольованість та довговічність даних, що є критично важливим для надійності системи. ObjectBox також доступна для різних операційних систем, що дозволить розгортати їх локально на різних пристроях.

Додатковою перевагою цієї бази даних є підтримка синхронізації даних між пристроями, що дозволяє забезпечити актуальність даних у різних середовищах без передачі їх стороннім сервісам.

Завдяки цим перевагам ObjectBox – це оптимальний вибір для реалізації швидкого пошуку та групування файлів у розроблюваній системі.

4.3 Програмна реалізація застосунку

Програмну частину застосунку розроблено з використанням фреймворку Flutter [23], який забезпечує можливість створення кросплатформених застосунків із єдиною кодовою базою. Такий підхід значно зменшує витрати часу та ресурсів на розробку й подальшу підтримку проекту. Flutter також

вирізняється високою продуктивністю та зручністю для розробників, розвиненою екосистемою та підтримкою компанії Google і великої спільноти. На ілюстрації (див. Рисунок 18) представлено вигляд користувацького інтерфейсу розроблюваної системи, створеного із застосуванням цього фреймворку.

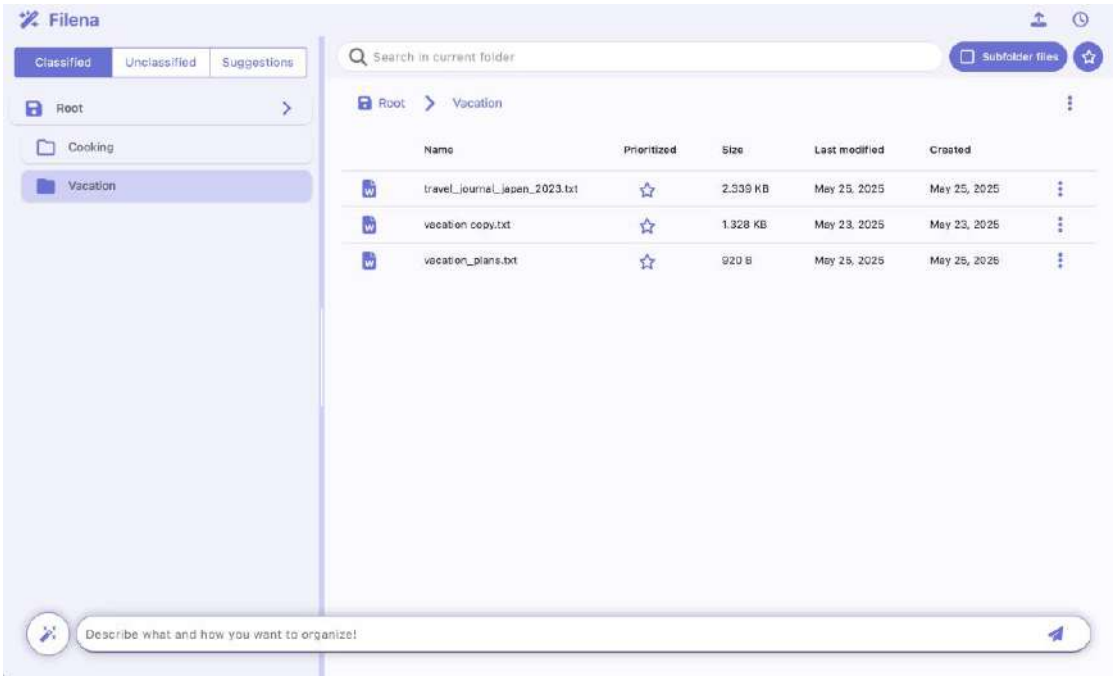


Рисунок 18. Реалізація головного екрану застосунку

З огляду на те, що необхідно забезпечити коректну синхронізацію стану компонентів системи, для їх управління було обрано сучасний фреймворк Riverpod [24], що забезпечує розробку в декларативному стилі, дає змогу відокремити бізнес-логіку від користувацького інтерфейсу, а також мінімізує необхідність використання компонентів зі станами. Такий підхід позитивно впливає на підтримуваність, масштабованість та якість розроблюваної системи.

4.4 Реалізація групування файлів у теки за користувацьким запитом

Для реалізації автоматичного групування файлів у теки за запитами користувача у системі використано сучасні підходи для обробки природної мови, векторних представлень об’єктів та векторного пошуку.

Під час завантаження файлів у систему для кожного з них створюється векторне представлення на основі їхнього вмісту. З метою зменшення шуму у векторному представленні текстових файлів перед створенням векторних представлень, із вмісту файлу велика мовна модель формує короткий огляд, який разом із метаданими передається до embedding-моделі для генерації векторного опису.

Векторні представлення зберігаються у базі даних разом із відповідними сутностями файлів. Для обробки зображень використано мультимодальну модель, якій передається base64-подання зображення з метою отримання його векторного представлення. Аналогічно для тек система створює векторні описи на основі їхніх шляхів. Таким чином, база даних ObjectBox містить векторні представлення для всіх файлів і тек у системі.

Коли користувач надсилає запит на групування файлів, система передає його великій мовній моделі (LLM), яка аналізує запит, визначає критерії групування та пропонує назву теки для об’єднання відповідних файлів. Якщо користувач не зазначив категорію, модель самостійно генерує релевантну назву теки на основі контексту запиту. Визначені критерії групування передаються embedding-моделі, що створює векторне представлення ключових слів, після чого у базі даних здійснюється векторний пошук.

Система обирає файли, векторна відстань від яких до ключового запиту не перевищує 0.5. Додатково, для визначення найкращого місця розташування нової теки база даних виконує пошук найближчої теки за векторною відстанню. Якщо ця відстань перевищує 0.1, то система створює нову дочірню теку в знайденій, інакше використовує її для додавання нових файлів.

Важливо підкреслити, що всі залучені моделі працюють локально, а отже жодні дані користувача не передаються третім сторонам, що гарантує конфіденційність інформації. До того ж, обрані моделі є оптимізованими та не потребують значних обчислювальних ресурсів користувацьких пристроїв. Для взаємодії із моделями використано провайдер Ollama [25]. Схема алгоритму наведена на ілюстрації (див. Рисунок 19).

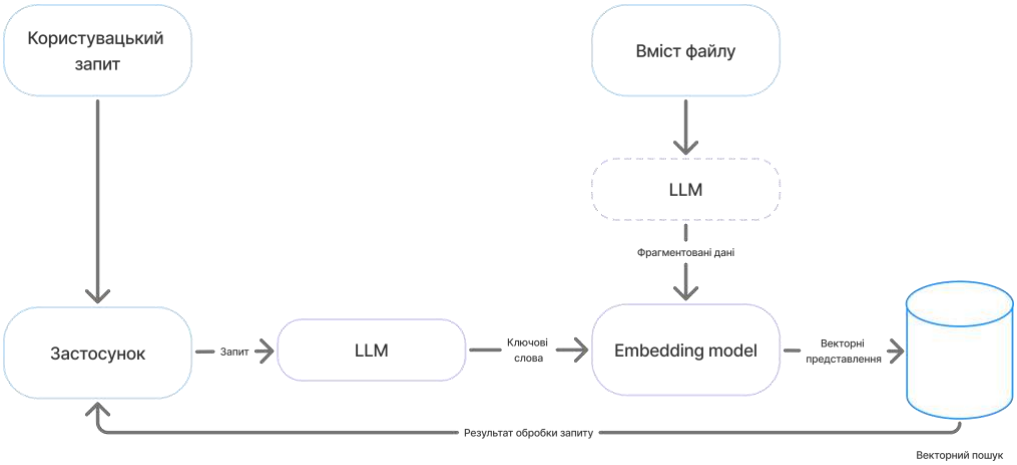


Рисунок 19. Схема алгоритму автоматизованого групування файлів

На ілюстрації (див. Рисунок 20) наведено приклад того, як виглядає неорганізована сукупність файлів у системі. Користувач може, наприклад, звернутись до застосунку із проханням організувати його файли, що стосуються відпустки, у відповідну єдину теку.

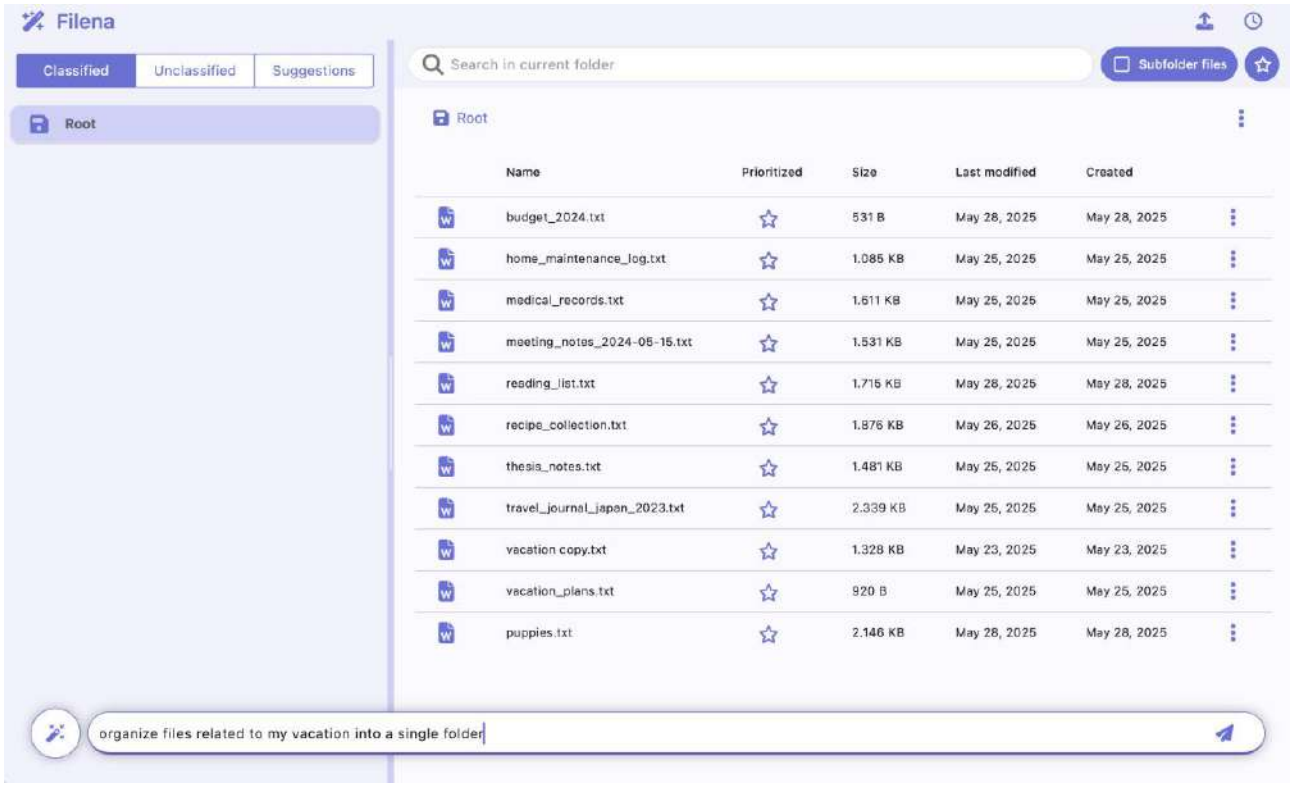


Рисунок 20. Неструктуровані файли в системі

Система визначила ключове слово «vacation» для групування файлів і назву теки «Travel Documents». На їх основі вона визначила, що сховище користувача містить п’ять файлів, що стосуються цієї категорії. Серед цих файлів дійсно усі такі, що стосуються відпустки, проте також містить два файли, що не стосуються цієї категорії (див. Рисунок 21).

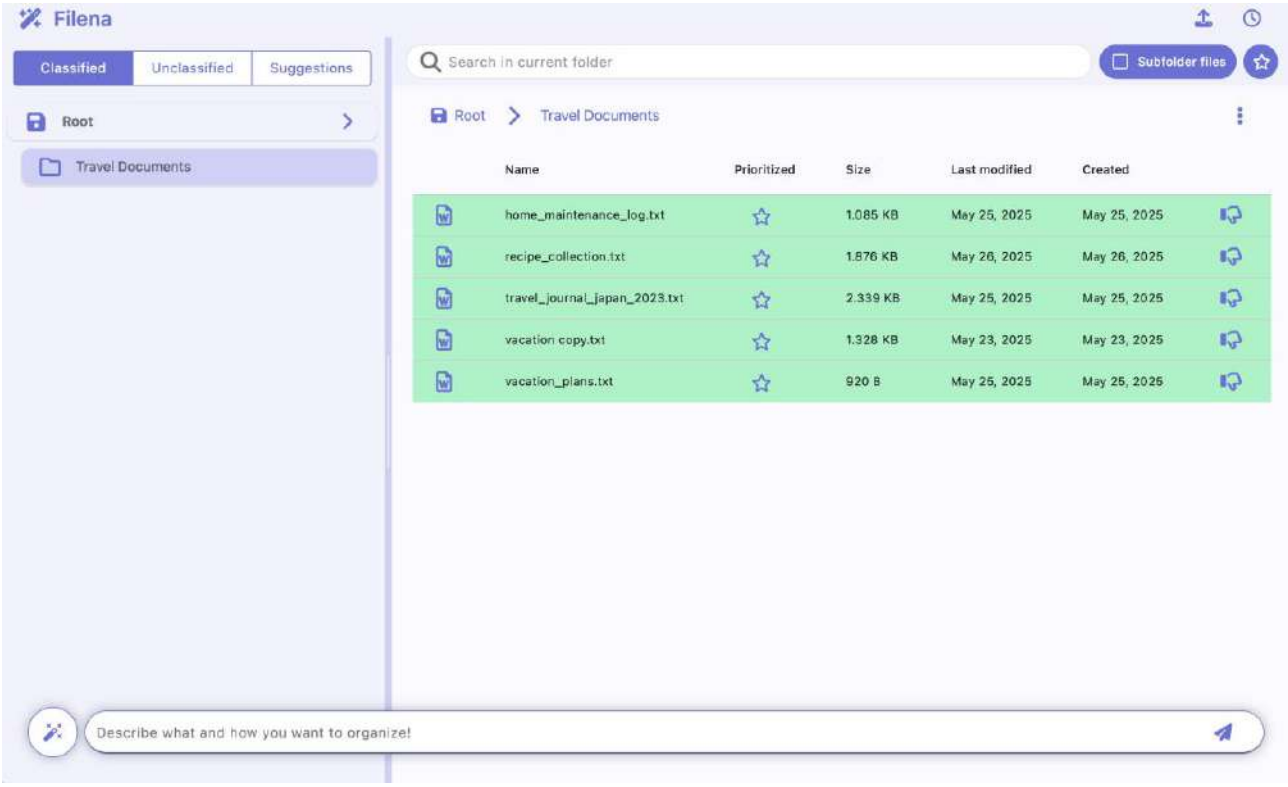


Рисунок 21. Пропозиція групування

Користувач може прийняти або відхилити пропозицію групування, а також прибрати з неї певні файли (див. Рисунок 22).

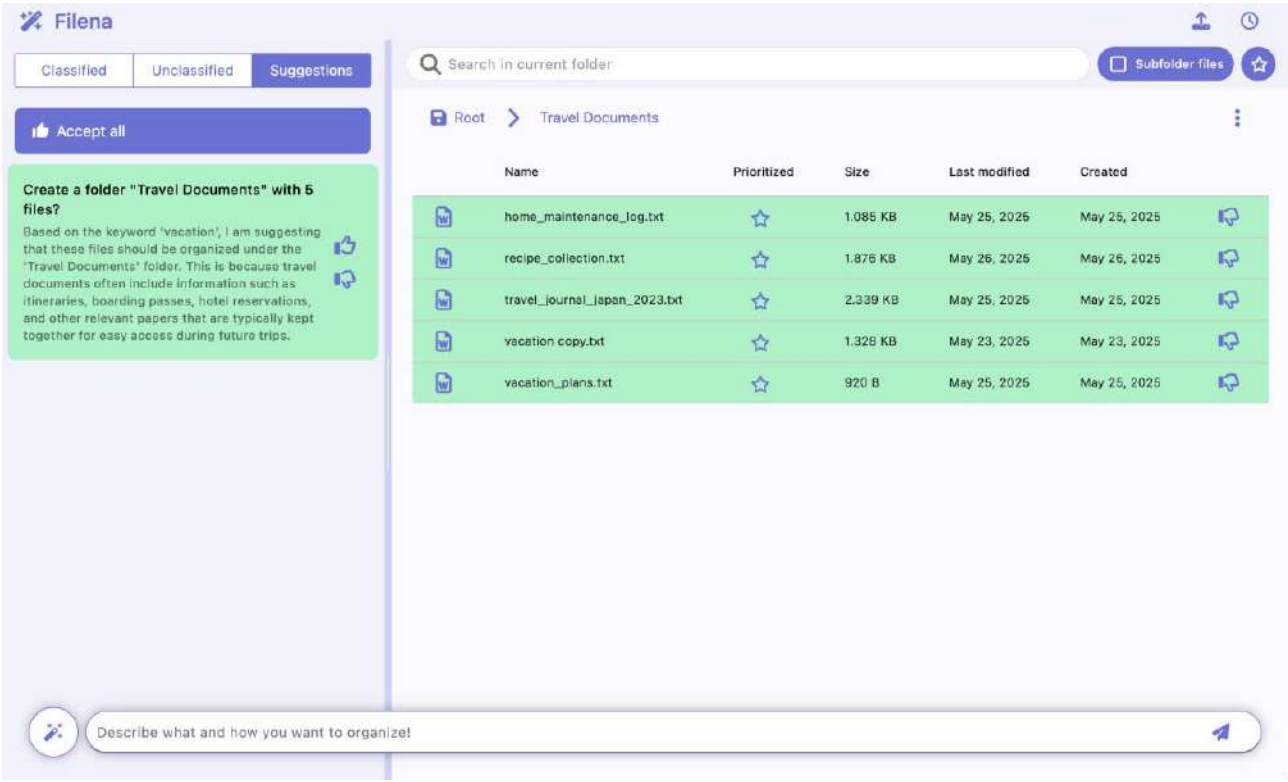


Рисунок 22. Редагування пропозиції групування

Після внесення коректив застосунок і прийняття пропозиції користувач може бачити файли в згенерованій теці (див. Рисунок 23).

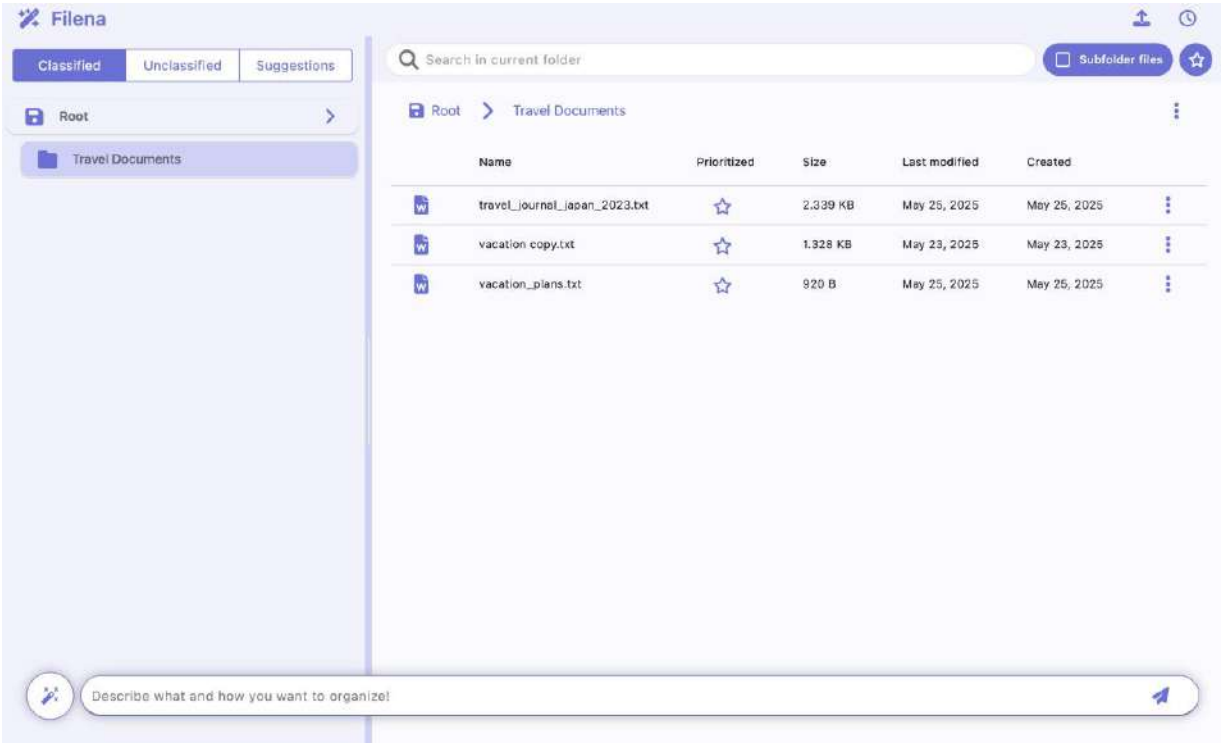


Рисунок 23. Застосування пропозиції групування

Алгоритм може показувати неточності через обмеження локальних моделей, втрату контексту під час скорочення тексту й умовність порогів при порівнянні векторних представлень.

Один з підходів, що може допомогти краще аналізувати вміст файлів – розбиття великих текстів на менші фрагменти зі збереженням їхнього сенсу [26]. Вибір правильної стратегії такого розбиття може значно зменшити шум у даних, і як результат покращити відповідність вмісту запитам користувача.

4.5 Висновки до розділу

У цьому розділі обґрунтовано вибір інструментів та технологій, що були використані для побудови програмної частини системи для організації персональних файлів. Було визначено, що для забезпечення надійного, продуктивного та безпечного сховища даних доцільно використовувати об’єктне сховище MinIO, що також забезпечує гнучкість інтеграції як з локальним, так і з хмарним сховищем.

Для зберігання та пошуку об’єктів та їх векторних представлень обрано базу даних ObjectBox, що забезпечує високу продуктивність, масштабованість, а також синхронізацію між пристроями, що може стати зручним інструментом для впровадження такої можливості в майбутньому.

Для розробки програмної частини застосунку було використано фреймворк Flutter, що дозволило реалізувати кросплатформену архітектуру з єдиною кодовою базою, а для ефективного управління станом компонентів обрано Riverpod.

Особливу увагу приділено реалізації автоматичного групування файлів у теки. Описаний підхід передбачає створення векторних представлень файлів і тек, застосування мовних моделей для аналізу текстових даних і зображень, а також використання embedding-моделей для пошуку схожості на основі векторної відстані.

Розроблений алгоритм групування має певні обмеження, пов’язані з якістю локальних моделей, втратами контексту при скороченні тексту та вибором порогових значень для пошуку за векторними представленнями. Задля покращення роботи системи розглянуто застосування стратегій розбиття тексту на осмислені фрагменти.

Отже, описані підходи дозволили створити прототип безпечної та масштабованої системи для інтелектуального управління файлами, що відкриває можливості подальшого вдосконалення автоматизованого групування даних.

ВИСНОВОК

У процесі виконання даної роботи було проведено аналіз сучасних проблем, що виникають під час організації персональної цифрової інформації, а також досліджено наявні програмні рішення у цій сфері. Було виявлено, що жоден із розглянутих застосунків не здатен у повній мірі задовольнити визначені потреби користувачів. Зокрема, жодне з наявних рішень не забезпечує автоматизованого ієрархічного групування файлів, яке б враховувало індивідуальні запити та вимоги користувача. Це обґрунтувало необхідність розробки власного рішення, що відповідатиме сучасним вимогам до організації та захисту персональної інформації.

На основі визначених технічних вимог було спроектовано архітектуру, структуру бази даних і користувацький інтерфейс програмного застосунку. Для реалізації системи обрано сучасні, продуктивні та безпечні інструменти, серед яких сховище даних MinIO, база даних ObjectBox, що підтримує зберігання векторних представлень, а для розробки програмної частини використано фреймворк Flutter.

Особливу увагу приділено впровадженню механізмів автоматичного групування файлів на основі їхнього змісту, метаданих та користувацьких запитів, із застосуванням векторних та мовних моделей.

Розроблена система забезпечує інтуїтивну організацію персональних файлів, конфіденційність даних, а також відкриває перспективи для подальшого розвитку можливостей цього рішення.

Результати роботи мають практичну цінність і можуть стати основою для розвитку нових підходів до автоматизованої класифікації та організації цифрових даних. Зокрема у майбутніх дослідженнях доцільно зосередитись на покращенні точності автоматичного групування файлів, дослідження стратегій chunking для підвищення релевантності результатів векторного пошуку, а також додавання підтримки аналізу відео та аудіо файлів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bergman O., Whittaker S. The science of managing our digital stuff. Massachusetts : MIT Press, 2016. 275 с.

2. View of digital hoarding: the rising environmental and personal costs of information overload. *Partners Universal Multidisciplinary Research Journal*. URL: https://www.pumrj.com/index.php/research/article/view/11/9.

3. Sinn D., Kim S., Syn S. Y. Personal digital archiving: influencing factors and challenges to practices. *Library hi tech*. 2017. Т. 35, № 2. С. 222–239. URL: https://doi.org/10.1108/lht-09-2016-0103.

4. Sorrel C. Local-First computing: why it's time to get your data out of the cloud. *Lifewire*. URL: https://www.lifewire.com/local-first-computing-7570378.

5. Get started with Copilot in OneDrive - Microsoft Support. *Microsoft Support*. URL: https://support.microsoft.com/en-us/office/get-started-with-copilot-in-onedrive-7fc81e10-e0cf-4da8-af2e-9876a2770e5d.

6. Semantic indexing for microsoft 365 copilot. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/microsoftsearch/semantic-index-for-copilot.

7. Microsoft 365 copilot connectors overview. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/microsoft-365-copilot/extensibility/overview-copilot-connector.

8. Data encryption in OneDrive and SharePoint. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/purview/data-encryption-in-odb-and-spo.

9. Використання finder на mac. *Apple Support*. URL: https://support.apple.com/uk-ua/guide/mac-help/mchlp2605/mac.

10. ICloud. *iCloud*. URL: https://www.icloud.com/.

11. Створення або змінення динамічної папки на Mac. *Apple Support*. URL: https://support.apple.com/uk-ua/guide/mac-help/mchlp2804/mac.

12. ICloud data security overview - Apple Support. *Apple Support*. URL: https://support.apple.com/en-us/102651.

13. FileNet P8 platform. *IBM - United States*. URL: https://www.ibm.com/docs.

14. Organize your files and folders with tags | TagSpaces. *Organize your files and folders with tags | TagSpaces*. URL: https://www.tagspaces.org/.

15. AI functionality | tagspaces docs. *User Documentation | TagSpaces Docs*. URL: https://docs.tagspaces.org/ai/.

16. Organize. *organize*. URL: https://organize.readthedocs.io/en/latest/.

17. Filters – organize 1.10.1 documentation. *organize*. URL: https://organize.readthedocs.io/en/v1.10.1/page/filters.html.

18. Martin R. Clean architecture: a craftsman's guide to software structure and design. Pearson Education, Limited, 2017.

19. MinIO | enterprise grade, high performance object storage. *MinIO*. URL: https://min.io/product/overview.

20. Radhakrishnan D. The most powerful S3 API ever? Introducing the prompt API. *MinIO Blog*. URL: https://blog.min.io/promptobjectapi/.

21. Rajaram S. Chat with your objects using the prompt API. *MinIO Blog*. URL: https://blog.min.io/chat-with-objects/.

22. Objectbox | Dart package. *Dart packages*. URL: https://pub.dev/packages/objectbox.

23. Flutter - Build apps for any screen. *Flutter - Build apps for any screen*. URL: https://flutter.dev/.

24. Riverpod. *Riverpod*. URL: https://riverpod.dev/.

25. Llama3.2. *Ollama*. URL: https://ollama.com/library/llama3.2.

26. Chunking strategies for LLM applications | pinecone. *The vector database to build knowledgeable AI | Pinecone*. URL: https://www.pinecone.io/learn/chunking-strategies/.

Додаток А

АРХІТЕКТУРА ПРОГРАМНОЇ ЧАСТИНИ ЗАСТОСУНКУ