МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Навчально-науковий інститут новітніх освітніх технологій
Кафедра інформаційно-обчислювальних систем і управління

НОВАК Христина Іванівна

Система управління нотатками з функціями менеджера паролів та спільного доступу / Note management system with password manager and sharing features

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки
Кваліфікаційна робота

Виконала студентка групи КНз-41
Х.І. Новак

Науковий керівник
к.т.н., доцент І.В. Турченко

Кваліфікаційну роботу допущено до
захисту «___»_____ 20___ р.

Завідувач кафедри
_____ Н.В. Дзюбановська

Тернопіль – 2026

Навчально-науковий інститут новітніх освітніх технологій
Освітній ступінь «бакалавр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
НОВАК Христині Іванівні
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Система управління нотатками з функціями менеджера паролів та спільного доступу / Note management system with password manager and sharing features

керівник роботи к.т.н., доцент І.В. Турченко
затверджений наказом по університету від 01 грудня 2025 року № 894.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити
- Провести аналіз предметної області систем управління нотатками
- Здійснити огляд наявних аналогів та визначити їх переваги та недоліки
- Сформулювати вимоги до програмної системи управління нотатками з функціями менеджера паролів та спільного доступу
- Розробити алгоритмічне та інформаційне забезпечення системи управління нотатками
- Розробити архітектуру програмної системи та структуру бази даних
- Реалізувати основні функціональні модулі інформаційної системи
- Забезпечити механізми захисту даних програмної системи
- Провести тестування розробленої програмної системи та оцінити коректність її функціонування

5. Перелік графічного матеріалу у роботі
- Діаграма варіантів використання системи SecureNotes
- Структурна схема програмної системи
- Схема алгоритму автентифікації користувача
- Схема алгоритму контролю активності сесії користувача
- ER-діаграма бази даних інформаційної системи

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Огляд літературних джерел відповідно до предметної області та складання плану роботи	до 01.01.2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02.2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05.2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06.2026 р.	

Студент _____ Х.І. Новак
підпис

Керівник роботи _____ к.т.н., доцент І.В. Турченко
підпис

АНОТАЦІЯ

Кваліфікаційна робота на тему «Система управління нотатками з функціями менеджера паролів та спільного доступу» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 70 сторінок і містить 12 ілюстрацій, 3 таблиці, 5 додатків та 28 використаних джерел.

Метою кваліфікаційної роботи є розроблення програмної системи управління нотатками, яка забезпечує зручне збереження інформації, безпечне зберігання паролів та організацію спільного доступу до даних.

Об'єктом дослідження є процеси створення, редагування та збереження інформації у вигляді нотаток і паролів, а також організація спільного доступу до даних у межах групи користувачів.

Результатом роботи є багатофункціональний настільний застосунок, що поєднує функції управління нотатками, інтегрованого менеджера паролів із криптографічним захистом, а також інструменти для створення груп і безпечної спільної роботи з інформацією.

Розроблено настільну систему управління нотатками з функціями менеджера паролів та спільного доступу, яку можуть використовувати студенти та викладачі для організації навчальних матеріалів , повсякденні користувачі для безпечного збереження конфіденційних даних і паролів , а також бізнес-структури та наукові колективи як надійний інструмент для ведення спільних проєктів і колективного доступу до інформації.

Ключові слова: СИСТЕМА УПРАВЛІННЯ НОТАТКАМИ, МЕНЕДЖЕР ПАРОЛІВ, ІНФОРМАЦІЙНА БЕЗПЕКА, ДЕСКТОПНИЙ ЗАСТОСУНОК, СПІЛЬНИЙ ДОСТУП, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

ANNOTATION

Qualification work on the topic «Notes management system with password manager and sharing features» for obtaining a bachelor's degree in the specialty 122 «Computer Science» of the educational program «Computer Science» is written on 70 pages and contains 12 illustrations, 3 tables, appendices, and 28 references.

The aim of this qualification work is to develop a software system for note management that ensures convenient information storage, secure password management, and the organization of shared access to data.

The object of research is the processes of creating, editing, and storing information in the form of notes and passwords, as well as organizing shared access to data within a group of users.

The result of the work is a multifunctional desktop application that combines convenient note management, an integrated password manager with cryptographic protection, and tools for creating groups and securely collaborating on information.

A desktop note management system with password manager and sharing features was developed. It can be used by students and teachers to organize educational materials, by everyday users to securely store confidential data and passwords, and by business structures and scientific teams as a reliable tool for managing joint projects and collective access to information.

Keywords: NOTE MANAGEMENT SYSTEM, PASSWORD MANAGER, INFORMATION SECURITY, DESKTOP APPLICATION, SHARED ACCESS, SOFTWARE.

ЗМІСТ

ВСТУП

У сучасному суспільстві, в епоху інформації, обсяг даних, з якими люди працюють щодня, постійно зростає. Саме тому люди повинні мати доступ до ефективних інструментів для зберігання, обробки та впорядкування своїх даних. Система управління нотатками – це тип інструменту, який дозволяє людям створювати та редагувати нотатки, структурувати їх, а також надає користувачам швидкий доступ до інформації.

Сучасні програмні рішення у сфері управління нотатками пропонують широкий функціонал, включаючи синхронізацію між пристроями, підтримку мультимедійного контенту та можливість спільної роботи. Водночас, більшість наявних систем не забезпечують належного рівня захисту конфіденційної інформації, зокрема паролів, або не мають інтегрованих механізмів їх безпечного збереження. Це створює додаткові ризики для користувачів та знижує ефективність використання таких систем.

Розробка настільної системи управління нотатками, яка забезпечує створення та редагування нотаток, безпечне збереження паролів із використанням механізмів шифрування, а також організацію спільних нотаток із можливістю створення груп і приєднання до них, є актуальною задачею сучасних інформаційних технологій.

Об'єктом дослідження є процеси створення, редагування, збереження та спільного використання інформації у вигляді нотаток і паролів.

Предметом дослідження є система управління нотатками з інтегрованим менеджером паролів та механізмами спільного доступу.

Метою кваліфікаційної роботи є розроблення програмної системи управління нотатками, яка забезпечує зручне збереження інформації, безпечне зберігання паролів та організацію спільного доступу до даних.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати предметну область систем управління нотатками;
- здійснити огляд наявних аналогів та визначити їх переваги та недоліки;
- сформулювати вимоги до програмної системи управління нотатками;
- розробити алгоритмічне та інформаційне забезпечення системи;

- розробити архітектуру програмного застосунку та структуру бази даних;
- реалізувати основні функціональні модулі інформаційної системи;
- забезпечити механізми захисту даних програмної системи;
- провести тестування розробленого програмного продукту.

У роботі застосовано методи аналізу предметної області, проєктування програмних систем, об'єктно-орієнтованого програмування, а також методи забезпечення інформаційної безпеки.

Практичне значення отриманих результатів полягає у створенні настільної системи, яка може бути використана для організації особистих та спільних нотаток, а також для надійного збереження конфіденційних даних.

Результати дослідження опубліковано в матеріалах І Міжнародної науково-практичної конференції «ICSPM 2026: Інтелектуальні обчислювальні системи та управління проєктами», м. Тернопіль, 21–22 травня 2026 р. (додаток Д).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ ПРОЄКТУВАННЯ

1.1 Опис предметної області систем управління нотатками

Системи управління нотатками допомагають людям зберігати, упорядковувати та знаходити інформацію у сучасному світі. Ці системи використовуються в освіті, бізнесі, наукових дослідженнях та повсякденному житті. Системи управління нотатками дають користувачам можливість створювати, впорядковувати і зберігати інформацію в цифровому вигляді, дають швидкий доступ і дозволяють робити структурований пошук [1]. Такий підхід допомагає користувачам ефективно працювати з інформацією в різних сферах, особливо в навчанні, у наукових дослідженнях і в бізнесі. Першими з цих розроблених систем були прості текстові редактори, які дозволяли користувачам створювати та змінювати файли. Потім з'явилися спеціалізовані програми, які дозволяли користувачам упорядковувати свої дані за допомогою списків, категорій та тегів. На сучасному етапі розвитку з'явилися хмарні сервіси, які надають користувачам доступ до нотаток на різних пристроях з миттєвою синхронізацією їхніх даних. «Інформаційні системи є комплексом взаємопов'язаних компонентів, які забезпечують збір, обробку, зберігання та поширення даних для підтримки управлінських рішень» [2].

Сучасні рішення для управління нотатками базуються на штучному інтелекті, автоматично генеруючи та шукаючи інформацію. Це стало важливим етапом розвитку сучасних інформаційних систем, які пропонують мобільність, зручність та багатофункціональність використання [3-4]. Ці системи мають дуже широку сферу застосування. Вони використовуються студентами та викладачами для планування навчальних матеріалів, планування занять та зберігання дослідницьких даних під час навчання. У бізнесі системи управління нотатками використовуються для управління проєктами, запису зустрічей та організації командної роботи. Вони використовуються в науковій роботі для зберігання результатів експериментів, створення баз даних та проведення спільного аналізу інформації. У повсякденному житті користувачі використовують такі системи для

планування завдань, ведення щоденників, збереження паролів та конфіденційних даних.

Особливого значення має питання інформаційної безпеки. «Більшість стандартних застосунків для нотаток не мають належних механізмів захисту, тому використання їх для паролів є небезпечним» [5]. Таким чином, інтеграція менеджера паролів у систему управління нотатками є актуальним напрямком розвитку, що дозволяє централізовано зберегти конфіденційні дані та забезпечити їх захист від несанкціонованого доступу. Прикладом є проект GitHub, де розроблено інтегровану систему для управління нотатками та паролями, що демонструє перспективність такого підходу [6].

Розвиток систем управління нотатками зумовлений потребами користувачів у комплексних інструментах для ефективної роботи з інформацією. На відміну від розглянутих систем, у розроблюваному застосунку реалізовано поєднання функцій збереження нотаток та захисту даних, що дозволяє усунути виявлені обмеження. Вивчення сучасного стану предметної області дозволяє сформулювати основні вимоги до системи, яка буде розроблена в межах цієї кваліфікаційної роботи.

1.2 Огляд і аналіз існуючих аналогів

На теперішній час існує багато аналогів програмних систем для управління інформацією. Одним із найбільш поширених та функціонально розвинених інструментів для роботи з електронними нотатками є Microsoft OneNote. Завдяки розгалуженій організаційній структурі, наявності безкоштовної версії з широким спектром можливостей та багатофункціональному інтерфейсу, цей застосунок характеризується високою універсальністю та комплексністю [7]. OneNote входить до складу пакета Microsoft 365 і забезпечує користувачам створення текстових та рукописних записів, інтеграцію графічних, аудіо- та відеоматеріалів, а також підтримку синхронізації з іншими сервісами Microsoft, зокрема Outlook та Teams. Крім того, програмне забезпечення реалізує механізми спільної роботи у режимі реального часу, що підвищує ефективність колективного використання та сприяє організації інформаційних процесів у навчальній, науковій та професійній

діяльності.

На рисунку 1.1 показано головне вікно OneNote, яке відображає структуру блокнотів, розділів та сторінок. Інтерфейс OneNote побудований за принципом "блокнот-розділ-сторінка": користувач створює окремі блокноти, які містять розділи та сторінки. Екранні форми відображають список блокнотів ліворуч, панель сторінок праворуч та робочу область для введення тексту й мультимедіа. Це дозволяє організовувати інформацію у вигляді вкладеної структури, що зручно для навчальних конспектів та бізнес-записів. Водночас у ньому відсутній інтегрований менеджер паролів, а механізми спільного доступу обмежені корпоративною екосистемою Microsoft, що не завжди зручно для індивідуальних користувачів. На основі проведеного аналізу встановлено, що Microsoft OneNote є комплексним програмним забезпеченням, орієнтоване на систематизацію та структуроване представлення інформації. Разом із тим багатокомпонентність та складність інтерфейсу можуть виступати чинником, що ускладнює процес первинної адаптації нових користувачів. Виявлена особливість засвідчує необхідність розробки більш інтуїтивно зрозумілих рішень, здатних забезпечити оптимальний баланс між функціональною насиченістю та доступністю використання [8-9].

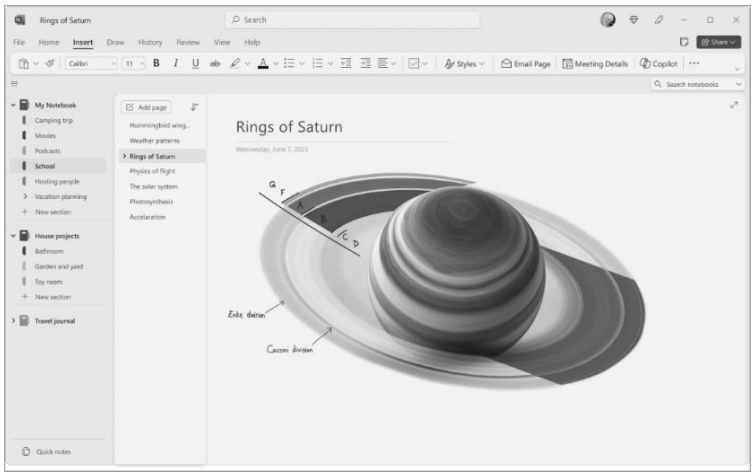

Рисунок 1.1 – Головне вікно Microsoft OneNote

Іншим прикладом програмного забезпечення для роботи з нотатками є Joplin, який позиціонується як безкоштовна альтернатива Evernote. Застосунок вирізняється орієнтацією на простоту використання, підтримку локального зберігання даних та наявність клієнтів для всіх основних платформ, що робить його

зручним для широкого кола користувачів [10]. Популярність Joplin зростає завдяки відкритому початковому коду та акценту на забезпеченні приватності. Програмне забезпечення створене з метою гарантувати повне володіння даними, використовуючи механізми шифрування та мінімальний збір інформації, що підтверджує його відповідність сучасним вимогам до захисту конфіденційності [11].

Інтерфейс складається зі списку блокнотів та нотаток із панеллю редагування Markdown, що дозволяє створювати структуровані записи з використанням форматування. Основні варіанти використання - створення технічних нотаток, імпорт даних з Evernote, локальне зберігання інформації. Екранні форми Joplin включають панель синхронізації, де користувач може обрати провайдера та налаштувати параметри шифрування, що є важливим для забезпечення приватності. На рисунку 1.2 наведено інтерфейс Joplin, де видно редактор Markdown і панель синхронізації. Попри акцент на безпекі, у Joplin немає інтегрованого менеджера паролів, а функції спільного доступу реалізовані лише частково через сторонні сервіси. У результаті аналізу встановлено, що Joplin може розглядатися як доцільний вибір для користувачів, які прагнуть зберігати контроль над власними даними. Водночас, інтерфейс програми характеризується певною багатокомпонентністю, що здатна створювати труднощі для нових користувачів [12]. Така особливість підтверджує необхідність розробки більш адаптованих та інтуїтивно зрозумілих рішень, орієнтованих на забезпечення доступності та зручності використання.

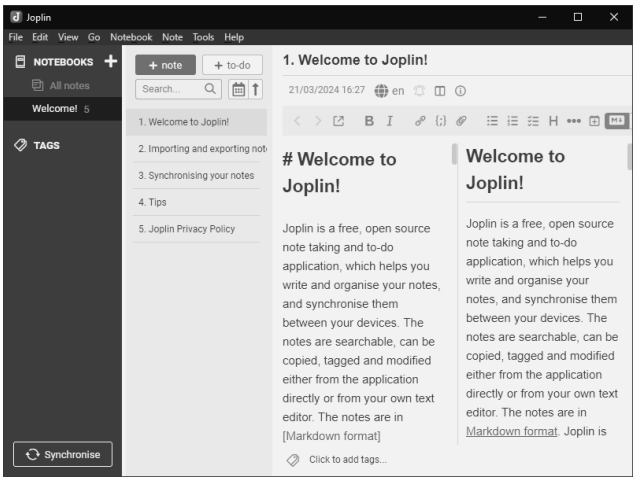

Рисунок 1.2 – Інтерфейс Joplin з редактором Markdown

Simplenote є мінімалістичним програмним забезпеченням для створення текстових нотаток, розробленим компанією Automattic. Застосунок орієнтований на користувачів, які працюють переважно з текстовими даними, пропонуючи базові функції, такі як версійність, спільна робота та синхронізація між пристроями, при цьому форматування тексту реалізовано через систему Markdown [13,14]. На рисунку 1.3 представлено головне вікно програми, що характеризується простотою інтерфейсу, відсутністю надлишкових елементів та наявністю можливості додавання тегів і пошуку [15].

Основними сценаріями використання є швидке створення коротких текстових записів, організація даних за тегами та синхронізація між пристроями. Водночас застосунок не підтримує роботу з мультимедійними даними, має обмежену політику конфіденційності та не передбачає механізмів захисту інформації на рівні менеджера паролів чи контролю прав доступу. Це свідчить про те, що Simplenote є доцільним вибором для користувачів, які цінують швидкість і простоту, проте його функціональні обмеження не дозволяють ефективно застосовувати його для роботи з конфіденційними даними [23].

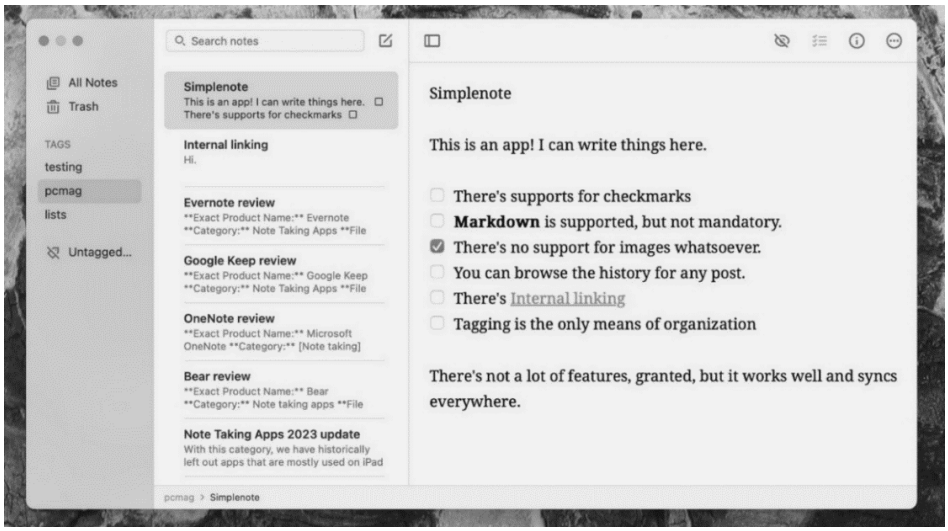

Рисунок 1.3 – Головне вікно Simplenote

Узагальнюючи, можна зазначити, що всі розглянуті аналоги мають сильні сторони, проте жоден із них не забезпечує комплексного поєднання функцій управління нотатками, інтегрованого менеджера паролів та зручного механізму спільного доступу. Це підтверджує актуальність розробки власної системи, яка

враховуватиме виявлені недоліки та потреби користувачів. Досвід провідних компаній-розробників (Microsoft, Automattic, спільнота Joplin) може бути використаний при реалізації проєктних рішень - зокрема, у частині організації інтерфейсу, синхронізації даних та підтримки форматування. Водночас унікальність розроблюваної системи полягає у поєднанні функцій управління нотатками з менеджером паролів та розвиненим механізмом спільного доступу, що забезпечує її конкурентні переваги.

1.3 Постановка задачі проєктування

Аналіз предметної області та огляд наявних аналогів показав, що сучасні системи управління нотатками мають широкий спектр функцій, проте не забезпечують комплексного поєднання зручності роботи, інформаційної безпеки та механізмів спільного доступу. Більшість програмних продуктів орієнтовані або на швидке створення текстових записів, або на організацію даних у вигляді структурованих блокнотів, але при цьому вони є обмеженими у питаннях захисту конфіденційної інформації та інтеграції функцій командної роботи. Це створює проблему для користувачів, які прагнуть мати універсальний інструмент для ведення нотаток, збереження паролів та організації спільної діяльності.

Об’єктом дослідження є процеси створення, редагування та збереження інформації у вигляді нотаток і паролів, а також організація спільного доступу до даних у межах групи користувачів. Предметом дослідження є програмна система управління нотатками, яка поєднує функції менеджера паролів та механізми спільної роботи [16].

Постановка задачі у процесі проєктування інформаційних систем є ключовим етапом, що визначає успішність усієї розробки. Вона повинна містити чітке формулювання мети, визначення об'єкта та предмета дослідження, а також опис вимог до функціональних і нефункціональних характеристик системи. Особливу увагу слід приділяти питанням інформаційної безпеки, адже сучасні системи працюють із конфіденційними даними та потребують захисту від несанкціонованого доступу. Недостатньо чітка постановка задачі призводить до

труднощів у подальшому проєктуванні, тому саме на цьому етапі закладаються основи архітектури та визначаються ключові модулі майбутньої системи [2].

Метою роботи є розробка настільної системи управління нотатками, яка забезпечує:

- зручне створення та редагування звичайних нотаток;
- безпечне збереження паролів із використанням механізмів шифрування;
- організацію спільних нотаток із можливістю створення груп та приєднання до них;
- контроль прав доступу та захист даних від несанкціонованого використання.

Для досягнення поставленої мети необхідно врахувати структурні, функціональні та конструктивні архітектурні рішення, які підвищують ефективність вирішення завдань. Зокрема:

- структурні зміни: система повинна мати чіткий поділ на три модулі – «Нотатки», «Паролі» та «Спільні нотатки». Така структура забезпечує логічну організацію даних та спрощує навігацію для користувача;
- функціональні зміни: кожен модуль має підтримувати базові операції (створення, редагування, видалення, перегляд), а також додаткові можливості – наприклад, для спільних нотаток це створення груп та приєднання до них;
- конструктивні зміни: система повинна бути реалізована як настільний застосунок із локальним збереженням даних та можливістю синхронізації. Для забезпечення безпеки передбачено використання алгоритмів шифрування паролів та механізмів автентифікації користувачів.

Загальна концепція проєктованої системи полягає у створенні комплексного інструменту, який поєднує функції управління нотатками та менеджера паролів, а також забезпечує можливість спільної роботи. На відміну від наявних аналогів, система має інтегрований модуль для захищеного збереження конфіденційних даних та розвинений механізм організації групового доступу. Це дозволяє користувачам не лише вести особисті нотатки, але й ефективно співпрацювати у навчальних, наукових та бізнес-проєктах.

У таблиці 1.1 наведено основні структурні, функціональні та безпекові вимоги до проєктованої системи

Таблиця 1.1 – Основні вимоги до системи управління нотатками з функціями менеджера паролів та спільного доступу

Категорія вимог	Зміст вимог	Очікуваний результат
Структурні вимоги	Система повинна мати три модулі: «Нотатки», «Паролі», «Спільні нотатки».	Логічна організація даних, зрозуміла навігація для користувача.
Функціональні вимоги	Кожен модуль підтримує операції: створення, редагування, видалення, перегляд. Для спільних нотаток – додатково створення груп та приєднання до них.	Повний набір базових операцій, можливість командної роботи.
Вимоги до безпеки	Використання алгоритмів шифрування для збереження паролів; автентифікація користувачів; контроль прав доступу.	Захист конфіденційних даних від несанкціонованого доступу.
Конструктивні вимоги	Реалізація у вигляді програмного застосунку для Windows; локальне збереження даних із можливістю синхронізації.	Зручність використання, мобільність, підтримка роботи на різних пристроях.
Інтерфейсні вимоги	Інтуїтивно зрозумілий графічний інтерфейс; підтримка української та англійської мов; можливість швидкого пошуку.	Дружність до користувача, зменшення бар’єру входження для новачків.
Експлуатаційні вимоги	Простота встановлення та оновлення; мінімальні системні ресурси; можливість резервного копіювання даних.	Надійність роботи системи, зручність обслуговування та підтримки.

Як видно з таблиці 1.1, проєктована система має відповідати сучасним вимогам до структури, функціональності та безпеки, що забезпечує її практичну цінність для користувачів.

На рисунку 1.4 наведено функціональну структуру системи та основні сценарії її використання. Система охоплює три ключові функціональні підсистеми: управління нотатками, управління паролями та роботу зі спільними нотатками. Кожна з підсистем підтримує виконання базових операцій зі створення, перегляду, редагування, видалення та організації відповідних даних. Крім того, система забезпечує засоби управління групами, що дозволяє впорядковувати інформацію та спрощує спільну роботу користувачів. Було визначено та закріплено назву програми - SecureNotes, що відображає її основне призначення та використовується у всіх наступних описах і характеристиках системи

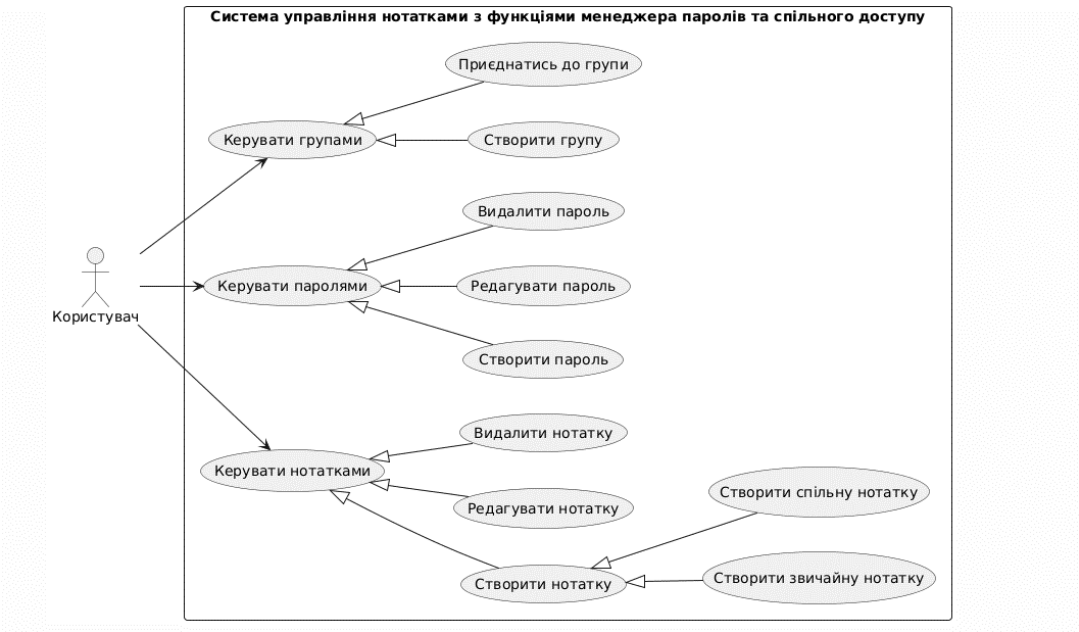

Рисунок 1.4 – Діаграма варіантів використання системи SecureNotes

Узагальнюючи, постановка задачі дослідження полягає у створенні програмної системи, яка відповідає сучасним вимогам до зручності, мобільності та безпеки, враховує недоліки наявних рішень і пропонує нові можливості для роботи з інформацією. Така система має практичну цінність для широкого кола користувачів – від студентів і викладачів до бізнес-структур та наукових колективів [17].

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ УПРАВЛІННЯ НОТАТКАМИ З ФУНКЦІЯМИ МЕНЕДЖЕРА ПАРОЛІВ ТА СПІЛЬНОГО ДОСТУПУ

2.1 Загальна структура проєктованої системи SecureNotes

Розроблений програмний застосунок належить до класу настільних інформаційних систем персонального призначення та орієнтований на зберігання, обробку й систематизацію особистих даних користувача. Актуальність створення такої системи зумовлена постійним зростанням обсягів інформації, що потребує впорядкованого зберігання, швидкого пошуку та забезпечення належного рівня захисту.

Під час проєктування програмної системи було обрано модульний підхід до побудови архітектури. Модульна архітектура програмного забезпечення передбачає поділ системи на незалежні компоненти, кожен з яких реалізує окрему функціональність [2, 16]. Загальна структура програмної системи та взаємозв’язок її основних компонентів наведені на рисунку 2.1.

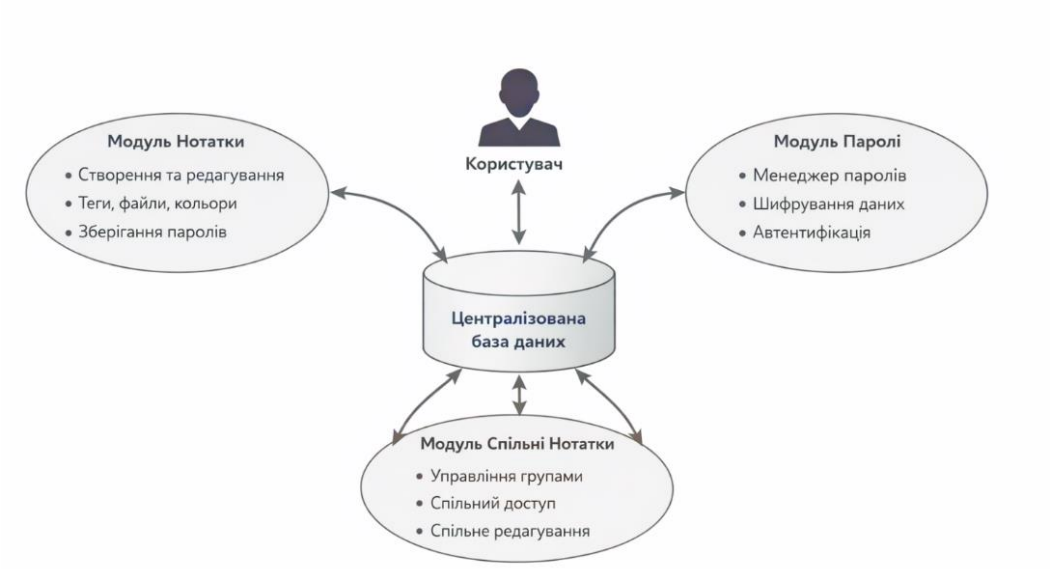

Рисунок 2.1 – Структурна схема програмної системи

Застосування такого підходу забезпечує спрощення тестування, супроводу та подальшого розширення програмної системи. Застосування модульної структури передбачає поділ програмного забезпечення на окремі логічно завершені

компоненти, кожен із яких виконує визначені функції. Такий підхід забезпечує зменшення взаємозалежності між елементами системи, спрощує процес тестування й супроводу програмного продукту, а також створює передумови для його подальшого масштабування та розширення функціональних можливостей.

Для забезпечення гнучкості та безпеки доступу до даних, у трирівневій моделі рівень доступу до даних реалізовано через проміжний шар – прикладний програмний інтерфейс (API). Це дозволяє абстрагувати логіку роботи з базою даних від клієнтської частини, забезпечуючи централізовану перевірку прав доступу та валідацію вхідної інформації на стороні сервера. Така архітектура трансформує застосунок із класичної локальної системи у клієнт-серверну структуру, де програмний клієнт взаємодіє із сервером за допомогою стандартних протоколів передачі даних.

Узагальнено архітектура системи побудована за трирівневою моделлю, яка включає рівень користувацького інтерфейсу, рівень прикладної логіки та рівень доступу до даних. Кожен із зазначених рівнів виконує чітко визначені функції та взаємодіє з іншими компонентами через формалізовані інтерфейси обміну даними.

Рівень користувацького інтерфейсу забезпечує взаємодію користувача із програмним застосунком. До його функцій належать відображення інформації у структурованому вигляді, приймання вхідних даних та ініціювання виконання відповідних операцій. Інтерфейс реалізовано у формі системи вікон, діалогових форм, меню та елементів керування. При проєктуванні особливу увагу приділено забезпеченню логічної організації елементів, мінімізації кількості дій для виконання типових операцій та підвищенню ергономічності програмного продукту.

Рівень прикладної логіки відповідає за реалізацію функціональних можливостей системи. На цьому рівні здійснюється обробка запитів, перевірка коректності введених даних, керування бізнес-логікою та формування результатів виконання операцій. Саме тут реалізовано алгоритми створення, редагування, видалення та пошуку нотаток, механізми роботи з вкладеннями, а також підсистему локалізації інтерфейсу.

Рівень доступу до даних забезпечує збереження, відновлення та оновлення інформації. Інформаційна структура охоплює тексти нотаток, відомості про вкладені файли, параметри налаштувань користувача та мовні ресурси. Взаємодія з цим рівнем здійснюється через спеціалізовані методи, що дозволяє ізолювати прикладну логіку від конкретної реалізації сховища та підвищує гнучкість системи.

У структурі прикладної логіки виокремлено низку функціональних модулів, кожен із яких реалізує визначений набір завдань. Перелік основних модулів та їх призначення наведено в таблиці 2.1.

Таблиця 2.1 – Основні модулі програмної системи

Назва модуля	Призначення
Модуль користувацького інтерфейсу	Забезпечує взаємодію користувача з програмою
Модуль керування нотатками	Реалізує створення, редагування та видалення нотаток
Модуль роботи з вкладеннями	Забезпечує додавання та відкриття файлів
Модуль локалізації	Відповідає за багатомовний інтерфейс
Модуль доступу до даних	Реалізує збереження та завантаження інформації

Модуль керування нотатками є центральним елементом системи та забезпечує повний цикл роботи із записами: створення, редагування, збереження, видалення, а також пошук і сортування. Усі операції виконуються з урахуванням перевірки коректності даних та підтримання цілісності інформаційної моделі.

Модуль роботи з вкладеннями реалізує механізм додавання та відкриття файлів різних форматів. Обробка вкладень здійснюється із використанням стандартних діалогових засобів операційної системи, що забезпечує сумісність із різними типами файлів та зручність для користувача.

Модуль локалізації відповідає за підтримку багатомовного інтерфейсу. Його функціонування базується на завантаженні мовних ресурсів та динамічній заміні

текстових елементів інтерфейсу відповідно до обраної мови. Такий підхід забезпечує адаптивність системи без внесення змін до програмного коду.

Модуль доступу до даних реалізує механізми запису та читання інформації, обробку можливих помилок та забезпечення узгодженості даних. Відокремлення цього модуля дозволяє знизити залежність між бізнес-логікою та фізичним способом зберігання інформації.

Взаємодія між модулями здійснюється за принципом чітко визначених викликів та контрольованого обміну даними. Користувацький інтерфейс ініціює виконання операцій, передаючи запити до рівня прикладної логіки, який, натомість, звертається до підсистеми доступу до даних. Отримані результати повертаються до інтерфейсу для відображення користувачеві.

Окремими функціональними компонентами системи є модулі "Нотатки", "Паролі" та "Спільні нотатки". Модуль "Нотатки" забезпечує створення, редагування, перегляд і видалення записів, а також підтримує механізми тегування, кольорового маркування та додавання вкладених файлів. Модуль "Паролі" реалізує функції менеджера паролів із застосуванням криптографічних механізмів захисту та процедур автентифікації. Модуль "Спільні нотатки" забезпечує організацію груп користувачів, розмежування прав доступу та можливість колективного редагування даних.

Усі функціональні компоненти взаємодіють із централізованою базою даних, яка виступає єдиним джерелом зберігання інформації та забезпечує її цілісність, узгодженість і захищеність [18]. Запропонована архітектура відповідає принципам модульного проєктування та забезпечує можливість подальшого розвитку програмного продукту відповідно до нових функціональних вимог.

2.2 Алгоритмічне забезпечення системи управління нотаками

Алгоритмічне забезпечення системи SecureNotes визначає порядок обробки даних та механізми забезпечення їх захисту. В основу розробки покладено комплекс взаємопов'язаних алгоритмів, що забезпечують повний життєвий цикл даних: від моменту введення користувачем до безпечного зберігання у хмарній

інфраструктурі. Побудова алгоритмічної моделі базується на принципах модульності та абстракції, що дозволяє відокремити криптографічну логіку від інтерфейсних рішень та механізмів доступу до даних. Важливим елементом алгоритмічної структури є алгоритм автентифікації та генерації ключів, який реалізує захист від несанкціонованого доступу ще на етапі входу в систему [19].

Процес автентифікації реалізовано з використанням стандарту PBKDF2 (Password-Based Key Derivation Function 2), що передбачає багаторазове хешування пароля разом із унікальним значенням солі. Такий підхід забезпечує стійкість до атак перебором. Отриманий результат використовується для перевірки автентичності користувача та формування ключа шифрування. Результуючий хеш виконує подвійну роль: він слугує еталоном для перевірки особи користувача та основою для формування сесійного ключа шифрування [18]. Логіка цього процесу відображена на алгоритмічній схемі, що демонструє послідовність трансформації вхідних даних у захищений криптографічний об'єкт. Повна послідовність кроків цієї операції, включаючи етапи трансформації даних та перевірки умов, наведена на рисунку 2.2.

Після успішної автентифікації вступає в дію алгоритм симетричного шифрування, що відповідає за конфіденційність нотаток та збережених паролів. У системі реалізовано алгоритм AES-256 (Advanced Encryption Standard). Для кожного запису генерується окремий ініціалізаційний вектор (IV) довжиною 16 байт. Перед шифруванням дані доповнюються за стандартом PKCS7, після чого формується шифротекст, який зберігається разом із відповідним IV. Це забезпечує унікальність результату шифрування навіть для однакових вхідних текстів архітектури [6, 20].

Важливим аспектом алгоритмічного забезпечення є механізм синхронізації з хмарною базою даних Aiven Cloud. Оскільки система працює з віддаленим сервером через Інтернет, алгоритм взаємодії побудований на принципах асинхронності, що дозволяє уникнути блокування інтерфейсу під час виконання мережевих запитів [21]. Алгоритм обробки запитів включає стадію перевірки стану з'єднання, ініціалізацію захищеного TLS-з'єднання та виконання параметризованих SQL-команд. Це виключає можливість проведення атак типу SQL-ін'єкцій,

оскільки всі дані користувача передаються як зашифровані параметри, а не як частина тіла запиту. У разі виникнення помилок мережі алгоритм передбачає механізм повторних спроб із експоненціальною затримкою, що забезпечує цілісність даних у нестабільних умовах зв'язку.

Рисунок 2.2 – Схема алгоритму автентифікації користувача

Алгоритм взаємодії між компонентами системи базується на принципах REST-архітектури. Процес обробки запиту користувача включає: формування HTTP-запиту клієнтським модулем, передачу параметрів у форматі JSON, автентифікацію запиту на стороні API та виконання відповідних CRUD-операцій у базі даних MySQL. Використання асинхронних запитів дозволяє підтримувати стабільну роботу інтерфейсу під час очікування відповіді від сервера, що важливо для збереження високої продуктивності застосунку [22].

Окрему увагу в алгоритмічній структурі приділено системі безпеки "активного сеансу", яка реалізована через алгоритм Idle Lock. Його робота базується на перехопленні системних подій введення через обробник подій Windows Forms. Алгоритм використовує глобальний таймер із фіксованим інтервалом. При кожному натисканні клавіші або русі миші в межах вікна програми лічильник обнуляється. Якщо ж активність відсутня протягом заданого часу, алгоритм ініціює процедуру негайного очищення статичної змінної SessionKey, яка зберігає ключ у оперативній пам'яті, та викликає примусове закриття всіх форм із конфіденційними даними. Реалізація даного механізму підвищує рівень захисту даних у разі відсутності активності користувача. Логіка переходів системи між станами активності, очікування та блокування, а також умови спрацювання захисних механізмів, детально відображені на схемі алгоритму, що наведена на рисунку 2.3.

Алгоритми управління контентом, зокрема створення груп та управління спільними нотатками, базуються на реляційній логіці зв'язків "один до багатьох" та "багато до багатьох". При створенні нової групи алгоритм генерує унікальний GUID, який стає ідентифікатором власника. Додавання співавторів до нотатки реалізується через алгоритм перевірки прав доступу, який перед кожним запитом до БД звіряє ідентифікатор поточного користувача зі списком дозволених суб'єктів для конкретного об'єкта нотатки. Це дозволяє реалізувати гнучкий механізм спільної роботи, зберігаючи при цьому ізоляцію особистих даних [8].

Для забезпечення високої ергономічності системи було розроблено алгоритм динамічної локалізації інтерфейсу. Він використовує структуру, де зберігаються пари "ключ-значення" для кожної підтримуваної мови (українська, англійська,

польська тощо). Алгоритм рекурсивно обходить дерево елементів керування поточної форми, ідентифікує текстові властивості компонентів за їхніми назвами та замінює їх на відповідні переклади зі словника. Це мінімізує використання пам'яті та дозволяє легко додавати підтримку нових мов без зміни програмного коду.

Рисунок 2.3 – Схема алгоритму контролю активності сесії користувача

Для забезпечення високої ергономічності системи було розроблено алгоритм динамічної локалізації інтерфейсу. Він використовує структуру, де зберігаються пари "ключ-значення" для кожної підтримуваної мови (українська, англійська, польська тощо). Алгоритм рекурсивно обходить дерево елементів керування поточної форми, ідентифікує текстові властивості компонентів за їхніми назвами та замінює їх на відповідні переклади зі словника. Це мінімізує використання пам'яті та дозволяє легко додавати підтримку нових мов без зміни програмного коду.

Завершальним елементом алгоритмічного забезпечення є підсистема резервного копіювання та відновлення даних. Алгоритм резервного копіювання виконує серіалізацію локальних налаштувань користувача у формат JSON та агрегацію всіх зашифрованих записів із бази даних. Отриманий масив даних піддається стисненню, що дозволяє створювати компактні архіви для зберігання на зовнішніх носіях. Алгоритм відновлення діє у зворотному порядку, включаючи обов'язкову валідацію цілісності архіву та перевірку сумісності версій бази даних, що запобігає пошкодженню інформаційної моделі при імпорті застарілих копій. Реалізований комплекс алгоритмів забезпечує функціонування системи відповідно до вимог безпеки та надійності.

2.3 Інформаційне забезпечення системи управління нотаками

Інформаційне забезпечення системи SecureNotes охоплює структуру даних, способи їх організації та механізми зберігання у базі даних. Основною метою побудови інформаційного забезпечення є організація раціональної структури даних, яка забезпечує мінімальну надмірність, високу швидкість доступу та гарантовану цілісність конфіденційних відомостей користувача. Враховуючи специфіку системи як інструменту для захищеного зберігання нотаток та паролів, інформаційна модель базується на поєднанні реляційного підходу до організації метаданих та криптографічного перетворення безпосереднього вмісту об'єктів зберігання. Такий підхід забезпечує зберігання в базі даних лише зашифрованої інформації, яка не може бути інтерпретована без відповідного ключа доступу [17].

Логічна модель даних побудована з використанням зв'язків типу "один до багатьох" та "багато до багатьох". Основна інформація користувача розподілена між таблицями notes, де зберігається зашифрований зміст нотаток у форматі LONGBLOB, та usergroups, що відповідає за логічне групування записів. Для реалізації механізму спільного доступу використана таблиця groupmembers, яка пов'язує ідентифікатори користувачів із конкретними групами та визначає рівень їхніх повноважень (Access_Level). Концептуальна схема даних, що відображає основні сутності та атрибути інформаційної моделі, представлена на рисунку 2.4 у вигляді ER-діаграми.

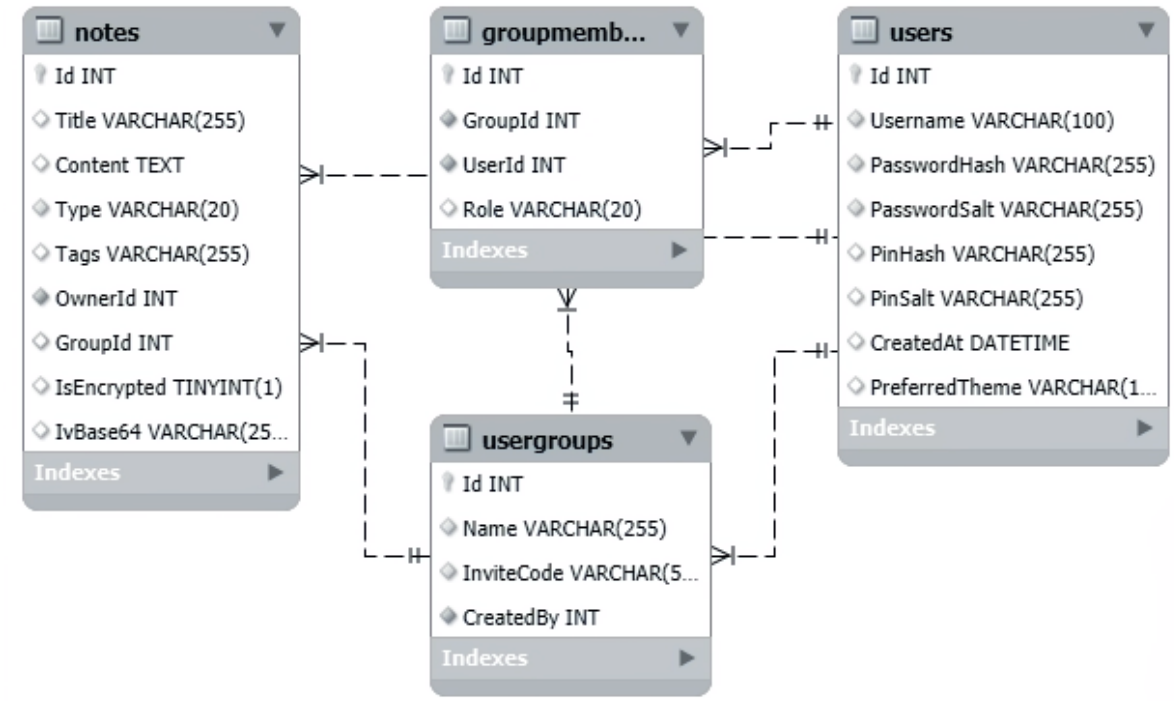

Рисунок 2.4 – ER-діаграма бази даних інформаційної системи

Для фізичної реалізації інформаційного сховища обрано хмарну інфраструктуру Aiven Cloud на базі СУБД MySQL. Такий вибір обґрунтований необхідністю забезпечення високої доступності даних та можливості масштабування ресурсів у разі зростання обсягів інформації [24]. Фізична структура таблиць оптимізована під зберігання бінарних об'єктів, оскільки всі змістовні поля (текст нотаток, логіни та паролі) передаються до бази виключно в зашифрованому вигляді. Кожна таблиця містить первинні ключі типу BIGINT або

GUID для забезпечення унікальності записів у глобальному масштабі. Додатково впроваджено механізм індексації за ключовими полями, такими як ідентифікатор користувача (User_ID), що дозволяє суттєво скоротити час виконання запитів при вибірці персональних даних.

Важливою складовою інформаційного забезпечення є механізми управління вхідними та вихідними потоками даних. Взаємодія з базою даних здійснюється через захищений канал із використанням протоколу SSL (Secure Sockets Layer), що ініціалізується на рівні конфігураційного файлу AppConfig. Алгоритм заповнення інформаційної бази передбачає обов'язкову валідацію типів даних на стороні клієнта, що запобігає пошкодженню структури таблиць. Перед передачею на сервер дані формуються у вигляді структурованих параметризованих SQL-запитів, що забезпечує коректну обробку інформації на стороні СУБД. При цьому інформаційна модель підтримує роботу з багатомовним контентом за допомогою кодування UTF-8 для коректного збереження українських та іноземних символів у змісті нотаток [24].

Особливе місце в інформаційному забезпеченні посідає організація спільних нотаток та управління групами. Для цього розроблено таблицю розмежування прав доступу, яка пов'язує унікальні ідентифікатори користувачів із ідентифікаторами груп та об'єктів. Інформаційний потік при наданні доступу включає передачу ідентифікатора співавтора та визначення рівня його повноважень (читання або редагування). Це дозволяє реалізувати динамічне оновлення інформаційного середовища для групи користувачів без необхідності дублювання самих даних. Такий підхід не лише економить дисковий простір хмарного сховища, але й гарантує цілісність інформації при одночасному доступі декількох суб'єктів.

Питання надійності зберігання інформації вирішується через впровадження процедур резервного копіювання та архівування. Інформаційне забезпечення передбачає можливість створення резервних копій даних та їх подальшого відновлення у разі потреби. Структура архівного файлу передбачає збереження не лише записів бази даних, а й пов'язаних медіа-файлів (якщо такі передбачені в нотатках), що забезпечує повне відновлення робочого середовища у разі критичного збою системи. Використання стандартних форматів обміну даними

робить систему сумісною з іншими інструментами аналізу та візуалізації інформації, що підвищує її практичну цінність [23].

Завершальним етапом проєктування інформаційного забезпечення є розробка засобів контролю та моніторингу цілісності даних. У системі реалізовано механізм контрольних сум для перевірки відповідності завантажених об'єктів їхнім оригіналам. Це дозволяє вчасно виявляти випадки пошкодження даних під час передачі або збоїв на стороні сервера. Крім того, інформаційна модель передбачає логування критичних дій користувача (вхід, зміна пароля, видалення групи), що створює необхідну доказову базу для проведення аудиту безпеки. Таким чином, розроблене інформаційне забезпечення системи SecureNotes є цілісним комплексом засобів, що забезпечує надійне зберігання та контроль цілісності даних користувача.

3 РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ НОТАТКАМИ З ФУНКЦІЯМИ МЕНЕДЖЕРА ПАРОЛІВ ТА СПІЛЬНОГО ДОСТУПУ

3.1 Обґрунтування вибору засобів і технологій та реалізація системи управління нотатками

Розроблення програмної системи здійснювалося з урахуванням сучасних підходів до створення прикладного програмного забезпечення та вимог до надійності, масштабованості й безпеки інформаційних систем. Інструменти та технології були обрані на основі аналізу функціональності, продуктивності, впровадження, сумісності один з одним та доцільності в контексті впровадження клієнт-серверної архітектури. Особлива увага приділялася забезпеченню стабільності системи, легкості розширення функціональності та керованості програмного коду.

Мова розробки C#, реалізована на стороні клієнта програмної системи, використовує об'єктно-орієнтоване програмування, що дозволяє організовувати програмний код у керовані частини, які можна повторно використовувати відповідно до принципів інкапсуляції, успадкування та поліморфізму. В результаті використання C# та платформи .NET існує чітке розмежування між логікою обробки даних та тим, як інформація подається користувачеві, що позитивно впливає на загальну якість архітектури програмного забезпечення. C# також має надійну систему типізації, багато вбудованих винятків для обробки помилок та функції автоматичного керування пам'яттю, які мінімізують помилки під час виконання програми [25].

Платформа .NET забезпечує середовище виконання програмного коду, яке відповідає за компіляцію, керування пам’яттю, перевірку типів та контроль безпеки [15]. Однією з особливостей цього фреймворку є наявність однієї базової бібліотеки класів для виконання стандартних функцій, таких як обробка файлів, мереж, потоків даних та колекцій. Ще однією перевагою розробки в керованому середовищі є те, що після написання програми вона має залишатися стабільною, оскільки кероване середовище автоматично очищає будь-які ресурси, які не використовуються, і гарантує, що виконання вашої програми буде здійснюватися

відповідно до задуму програми без винятків. Таким чином можна вільно розробляти логіку своєї програми, не турбуючись про низькорівневе управління пам'яттю.

Інтерфейс користувача реалізовано із застосуванням засобів створення настільних застосунків для операційної системи Windows. Такий підхід забезпечує повну інтеграцію із середовищем операційної системи, підтримку стандартних елементів керування та стабільну роботу на персональних комп’ютерах користувачів. Використання програмного застосунку є доцільним у випадках, коли необхідно забезпечити швидку реакцію інтерфейсу, локальну обробку даних та зменшення залежності від постійного мережевого з’єднання.

Для зберігання інформації у системі використовується реляційна система керування базами даних MySQL. Вибір саме реляційної моделі обумовлений структурованим характером даних, що обробляються програмною системою, а також необхідністю забезпечення цілісності та узгодженості інформації. MySQL підтримує механізм транзакцій, індексування, обмеження цілісності та багатокористувацький режим роботи, що є важливим для забезпечення коректності збереження даних. Підтримка стандарту SQL дозволяє формувати гнучкі запити для вибірки, оновлення та видалення записів [23]. Основні SQL-запити створення структури бази даних наведено у додатку А.

Зв’язок клієнтської частини з базою даних реалізовано через відповідний драйвер підключення для платформи .NET, що забезпечує виконання SQL-запитів та отримання результатів їх виконання у вигляді структурованих об’єктів. Така інтеграція дозволяє програмній логіці застосунку враховувати реляційну модель даних без порушення принципів модульності. Обробка запитів здійснюється з урахуванням перевірки коректності введених даних, що зменшує ризик виникнення помилок та підвищує рівень безпеки системи.

Архітектурно програмна система побудована за клієнт-серверною моделлю. Клієнтський застосунок відповідає за відображення інтерфейсу користувача та формування запитів, серверна частина здійснює централізовану обробку логіки взаємодії та забезпечує контроль доступу до даних, а база даних виконує функцію довготривалого зберігання інформації (основний програмний код клієнтської

частини наведено у додатку Б) Такий розподіл функцій дозволяє забезпечити масштабованість системи та спростити її модернізацію у разі необхідності розширення функціональних можливостей.

На стороні клієнтського застосунку реалізовано низку класів, що забезпечують його життєвий цикл та управління інформаційною моделлю. Основний клас програми Program.cs відповідає за ініціалізацію застосунку, завантаження збережених налаштувань, керування активною сесією та управління темами оформлення (код Б.1). Об'єктна модель предметної області представлена базовими класами сутностей: класом Note (код Б.2), що інкапсулює логіку роботи з атрибутами нотаток і параметрами їх шифрування, та класом User.cs (код Б.3), який зберігає облікові дані, криптографічні параметри та персональні налаштування поточного користувача.

Для підвищення ергономічності та зручності взаємодії з елементами керування було додатково розроблено клас допоміжних методів UIHelpers.cs (код Б.4). Цей компонент програмно забезпечує логіку роботи з текстовими полями, зокрема створення динамічних підказок. Вони автоматично відслідковують стан фокусу елемента та адаптуються під обрану користувачем темну чи світлу тему оформлення, що значно покращує візуальне сприйняття інтерфейсу.

Серверний компонент системи реалізований у вигляді прикладного програмного інтерфейсу, який забезпечує взаємодію між клієнтською програмою та базою даних через стандартизований механізм обміну даними (додаток В).

Серед основних класів можна виділити клас «CryptoService», який реалізує механізми криптографічного захисту паролів користувачів. Використовується алгоритм PBKDF2 з 100 000 ітераціями та 256-бітним ключем. Для кожного пароля генерується унікальна криптографічна сіль розміром 16 байт, що унеможливлює використання райдужних таблиць для несанкціонованого доступу (код В.1).

Клас «TokenService» відповідає за генерацію JSON Web Token (JWT) для автентифікації користувачів. Токен містить ідентифікатор користувача та має обмежений термін дії. Використовується симетричний алгоритм підпису HMAC-SHA256 (код В.2).

Передача інформації здійснюється за допомогою протоколу HTTP, який визначає структуру запитів і відповідей, а також коди стану виконання операцій. Клас «AuthController» забезпечує обробку HTTP-запитів для реєстрації нових користувачів та автентифікації вже зареєстрованих. Реалізовано кінцеві точки для створення облікового запису з хешуванням пароля та входу в систему з видачею JWT-токена. (код В.3).

Клас «NotesController» реалізує CRUD-операції (Create, Read, Update, Delete) для роботи з захищеними нотатками. Усі операції доступні лише автентифікованим користувачам, що забезпечується атрибутом [Authorize] (код В.4).

Використання текстового формату JSON для обміну даними забезпечує компактність передавання інформації та зручність її обробки як на стороні клієнта, так і на стороні сервера. Такий підхід відповідає принципам REST-архітектури, що передбачає чітке визначення ресурсів та операцій над ними.

Для узагальнення характеристик використаних технологій у таблиці 3.1 наведено основні програмні засоби та їх призначення [26].

Таблиця 3.1 – Характеристика використаних технологій та інструментальних засобів

Технологія	Тип	Функціональне призначення	Причина вибору
C# (.NET Core)	Мова програмування	Розробка клієнтської та серверної частин системи	Об’єктно-орієнтований підхід, строга типізація, висока безпека коду
ASP.NET Core Web API	Фреймворк	Реалізація серверної логіки та взаємодії з БД	Створення продуктивних REST-сервісів, кросплатформеність, модульність
MySQL (Cloud)	СУБД	Централізоване зберігання структурованих даних	Надійність, підтримка транзакцій, можливість стабільного хмарного розміщення
HTTP / REST	Архітектурний стиль та протокол	Організація обміну даними між клієнтом та API	Стандартизований механізм взаємодії, простота масштабування системи

Продовження таблиці 3.1

Технологія	Тип	Функціональне призначення	Причина вибору
Newtonsoft.Json	Програмна бібліотека	Серіалізація та десеріалізація об’єктів	Ефективна обробка JSON-даних при передачі через мережеві протоколи
HttpClient	Системний клас	Виконання асинхронних запитів до серверної частини	Вбудована підтримка асинхронності, що забезпечує чуйність інтерфейсу

Використання єдиного технологічного стеку для клієнтської та серверної частин забезпечує узгодженість програмної реалізації та спрощує процес розроблення. Завдяки цьому зменшується кількість потенційних конфліктів між різними технологіями та підвищується ефективність інтеграції компонентів системи. Обрана архітектура дозволяє у подальшому реалізувати розширення системи без суттєвої зміни її базової структури.

3.2 Інтерфейс користувача системи SecureNotes

SecureNotes має інтерфейс користувача для робочого столу, розроблений зі стандартними елементами керування .NET, що забезпечує нативну інтеграцію з ОС Windows, передбачувану поведінку елементів та стабільність роботи. Інтерфейс було розроблено на основі основних сценаріїв користувача системи, включаючи візуальне представлення списку нотаток, додавання нових записів, редагування існуючих записів, організацію спільного доступу та налаштування параметрів відображення для користувачів. При розміщенні різних елементів керування враховувалося як їхня функція, так і типовий порядок, у якому користувачі виконують ці завдання [2].

Головне вікно програмної системи призначене для відображення списку створених нотаток. У ньому відображаються назви записів, а також їх колірне оформлення, що дозволяє швидко візуально відрізняти нотатки одна від одної. Передбачено можливість відкриття нотатки подвійним натисканням або вибором

відповідної команди редагування. Головне вікно містить елементи керування для створення нової нотатки та доступу до налаштувань.

Інтерфейс головного вікна забезпечує компактне відображення інформації без перевантаження елементами керування, що дозволяє зосередитися на роботі зі списком записів. Зовнішній вигляд головного вікна наведено на рисунку 3.1.

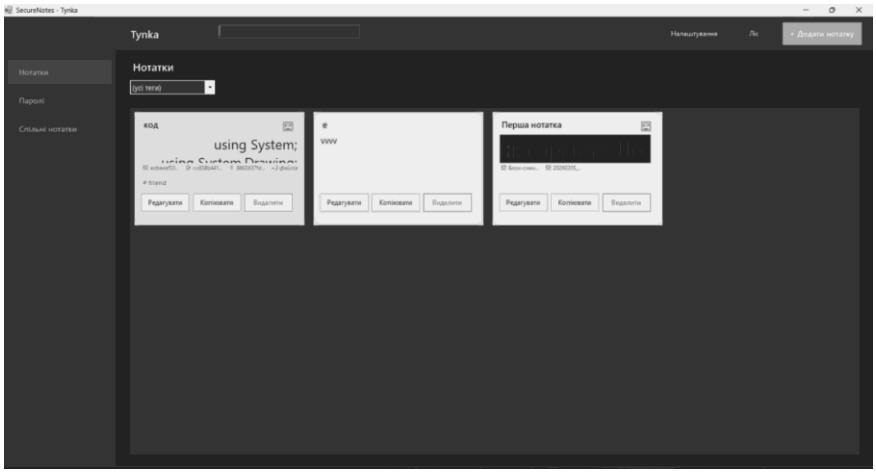

Рисунок 3.1 – Головне вікно програмної системи SecureNotes

Створення та редагування нотаток здійснюється в окремому діалоговому вікні. Вікно для звичайних нотаток та для записів типу «пароль» має однакову структуру та набір елементів керування. Відмінність полягає лише у виборі типу запису, що визначається відповідним перемикачем. Такий підхід забезпечує єдину логіку роботи з різними типами даних.

У верхній частині форми розташоване поле введення заголовка, яке використовується для ідентифікації нотатки в загальному списку. Нижче знаходиться текстова область для введення основного змісту. Над текстовим полем розміщена панель інструментів форматування, що містить кнопки для зміни стилю шрифту (жирний, курсив, підкреслення, закреслення), вибору розміру тексту, вирівнювання абзаців, створення маркованих списків, зміни кольору тексту та фону виділення, а також для скасування й повторення виконаних дій. Передбачено можливість додавання вкладених файлів через стандартне діалогове вікно вибору файлів.

У нижній частині форми розташовані додаткові параметри нотатки: вибір типу запису, встановлення кольору, введення тегів та налаштування спільного

доступу. Кнопка збереження забезпечує фіксацію змін у системі зберігання даних. Зовнішній вигляд вікна створення та редагування нотатки наведено на рисунку 3.2.

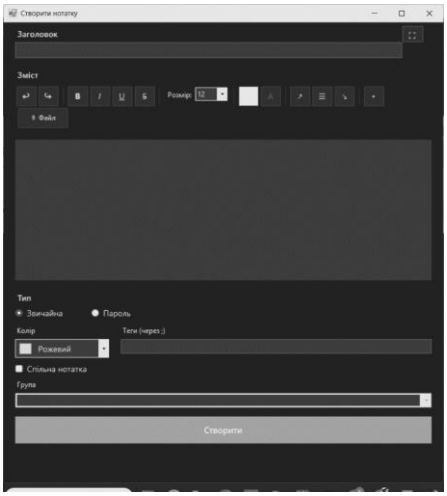

Рисунок 3.2 – Вікно створення та редагування нотатки

Для реалізації спільного доступу передбачено встановлення відповідного прапорця. Після його активації стає доступним поле вибору групи, до якої буде віднесено нотатку. Групи створюються окремо та дозволяють логічно об’єднувати записи. Такий механізм забезпечує структурованість даних і можливість організації спільної роботи [27]. Блок налаштування спільного доступу наведено на рисунку 3.3.

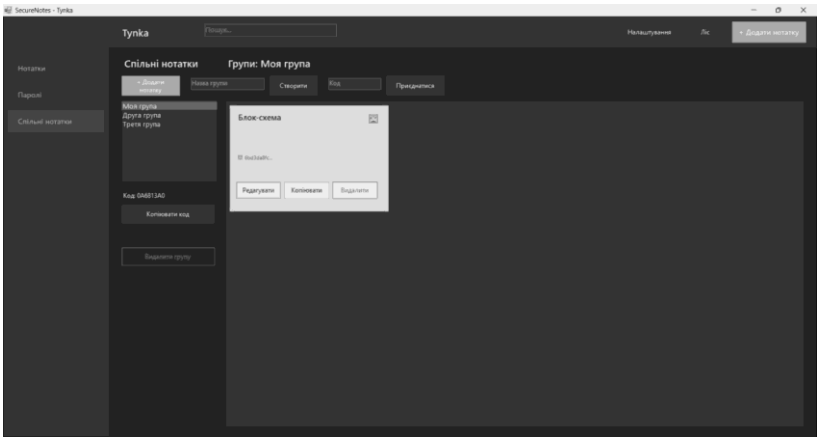

Рисунок 3.3 – Налаштування спільного доступу до нотатки

Окрім основних форм, система містить окреме вікно налаштувань, призначене для конфігурації параметрів роботи застосунку. У цьому вікні

користувач може змінювати параметри відображення інтерфейсу, зокрема тему оформлення. Зміна теми впливає на колірне оформлення елементів керування та загальний вигляд програми. Також у цьому ж вікні можна перейти на інший акаунт чи змінити пароль до свого акаунту. Додатково у SecureNotes реалізовано модуль локалізації, що забезпечує підтримку різних мов інтерфейсу [27]. Обрані параметри зберігаються та застосовуються під час подальшого запуску системи. Зовнішній вигляд вікна налаштувань наведено на рисунку 3.4.

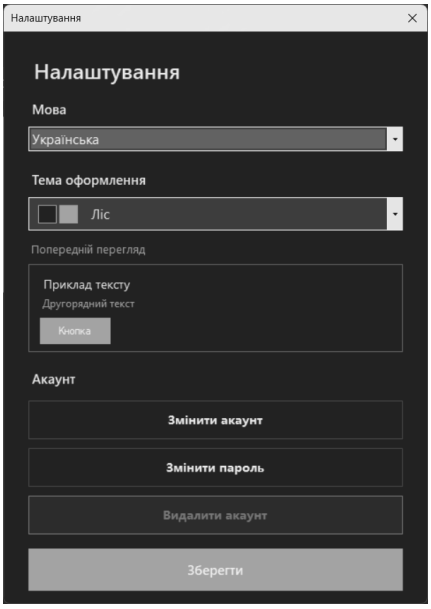

Рисунок 3.4 – Вікно налаштувань програмної системи SecureNotes

Таким чином, інтерфейс програмної системи SecureNotes забезпечує реалізацію всіх передбачених функціональних можливостей через структуровано організовані форми та елементи керування. Єдина логіка побудови вікон, однакова структура форм для різних типів записів та використання стандартних компонентів середовища Windows забезпечують зручність використання та спрощують процес освоєння програми.

3.3 Тестування та оцінка якості програмної системи SecureNotes

Завершальним та найбільш відповідальним етапом життєвого циклу розробки інформаційної системи "SecureNotes" є проведення комплексного тестування. Головною метою цього етапу є об'єктивна оцінка працездатності всіх

програмних модулів, перевірка відповідності розробленого продукту сформованим функціональним і нефункціональним вимогам, а також практичне підтвердження надійності та безпеки обраних архітектурних рішень у межах клієнт-серверної взаємодії. Процес тестування було організовано за принципом "чорної скриньки". Зазначений підхід дозволив зосередитися на аналізі вихідних даних залежно від введених користувачем параметрів, не заглиблюючись у внутрішню логіку програмних методів, що забезпечило максимальну об'єктивність оцінки. У додатку Г наведено тестові сценарії для перевірки основних модулів програмної системи.

Важливим аспектом дослідження стала перевірка функціональності роботи з нотатками (таблиця Г.1), тестування можливостей форматування тексту (таблиця Г.2) та реалізації функцій групової взаємодії (таблиця Г.3). Тестування цих модулів дозволило підтвердити коректність виконання базових операцій, зручність інтерфейсу та надійність механізмів спільної роботи, що є ключовими для практичного використання програмної системи.

Було протестовано додаткову функціональність: зміну тем інтерфейсу та мовної локалізації шляхом вибору різних кольорових тем та мов, а також перевірено збереження налаштувань після перезапуску програми (таблиця Г.4).

Враховуючи впровадження прикладного програмного інтерфейсу як проміжного шару між програмним клієнтом та хмарною базою даних MySQL, програма тестування була розширена тестами мережевого рівня. Процес верифікації охоплював не лише перевірку локальних подій графічного інтерфейсу користувача, а й контроль цілісності даних при їх трансляції через мережеві протоколи. Основна увага приділялася верифікації механізмів серіалізації об'єктів у формат JSON, стабільності обробки HTTP-запитів та здатності системи коректно обробляти виняткові ситуації, спричинені латентністю мережі або тимчасовою недоступністю віддаленого вузла.

Для досягнення максимальної повноти охоплення перевіркою, у межах дослідження було виокремлено чотири послідовні цикли тестування: функціональне, інформаційної безпеки, продуктивності та мережевої стійкості, стрес тестування та контроль цілісності.

На етапі функціонального тестування проводилася детальна перевірка коректності виконання базових CRUD-операцій для всіх ключових сутностей системи: текстових нотаток, зашифрованих записів паролів та користувацьких груп. Окремо тестувалася логіка динамічного оновлення компонентів інтерфейсу після отримання асинхронного підтвердження від API про успішне внесення змін у хмарну базу даних. Особлива увага була приділена валідації вхідних форм на предмет недопущення введення некоректних символів або порожніх значень.

Тестування інформаційної безпеки. Оскільки захист конфіденційної інформації є пріоритетною вимогою до системи, було проведено багаторівневу валідацію процедур автентифікації (таблиця Г.5). Перевірка включала контроль життєвого циклу токенів доступу, підтвердження стійкості алгоритму шифрування AES-256 (таблиця Г.6) при передачі даних через відкриті канали зв'язку та верифікації механізму розмежування прав доступу при роботі суб'єктів у межах спільних груп. Було підтверджено, що навіть за умови перехоплення мережевого трафіку, вміст нотаток залишається недоступним для дешифрування без відповідного ключа користувача.

В межах циклу тестування продуктивності та мережевої стійкості аналізувався середній час відповіді сервера на запити різної складності та оцінювалася реактивність інтерфейсу під час виконання фонових операцій. Важливим аспектом була імітація критичних умов роботи: повна відсутність зв'язку із сервером або раптовий розрив сесії. За допомогою методів обробки винятків та використання класу HttpClient, система продемонструвала здатність до самовідновлення та коректного інформування користувача про перехід в офлайн-режим без серйозних збоїв у роботі.

Під час стрес-тестування та контролю цілісності проводилася імітація інтенсивного навантаження на API шляхом генерації одночасних запитів від декількох ініціалізованих клієнтів до одного ресурсу (спільної нотатки). Це дозволило переконатися, що механізми транзакційності на стороні MySQL та логіка обробки черг на стороні серверної частини зберігають цілісність інформації при конкурентному доступі.

Аналіз підсумкових показників випробувань продемонстрував, що обраний технологічний стек забезпечує високу швидкість обробки інформаційних потоків. Використання формату JSON підтвердило свою універсальність для передачі складних зашифрованих структур [22]. Застосування асинхронних конструкцій дозволило повністю уникнути блокування головного потоку програми, що забезпечує стабільно високий рівень комфорту при взаємодії користувача із застосунком [21].

Підсумкові результати наведено у таблиці Г.7, що узагальнює кількість виконаних тестів по модулях. Загалом було реалізовано 34 тестових сценарії, усі з яких завершилися успішно, що свідчить про стабільність роботи системи, коректність реалізації ключових функцій та високий рівень надійності програмного продукту.

Підсумовуючи вищевикладене, можна стверджувати, що система SecureNotes повністю відповідає вимогам технічного завдання. Виявлені під час ітераційного тестування незначні дефекти в обробці таймаутів запитів були усунені шляхом оптимізації конфігурації HttpClient. Це дозволяє вважати систему готовою до практичного впровадження та стабільної експлуатації в умовах реального мережевого середовища [26].

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було:

1. Проведено аналіз предметної області систем управління нотатками, що здійснювався шляхом дослідження сучасних інформаційних систем та підходів до організації збереження і обробки даних та визначено основні вимоги до функціональності, зручності використання та інформаційної безпеки системи.

2. Виконано огляд і порівняльний аналіз наявних програмних рішень, який здійснювався на основі дослідження функціональних можливостей сучасних застосунків для роботи з нотатками, що дозволило виявити їх основні недоліки, зокрема відсутність інтегрованого менеджера паролів, обмежені можливості захисту даних та недостатню підтримку спільного доступу.

3. Сформульовано постановку задачі та визначено вимоги до програмної системи на основі проведеного аналізу предметної області та наявних аналогів, що забезпечило чітке визначення структури системи, її функціональних модулів та критеріїв ефективності.

4. Розроблено загальну архітектуру програмної системи шляхом побудови модульної клієнт-серверної структури з поділом на рівень користувацького інтерфейсу, прикладної логіки та доступу до даних, а також виокремленням модулів управління нотатками, збереження паролів і спільного доступу.

5. Розроблено алгоритмічне забезпечення програмної системи, що включає алгоритми автентифікації користувачів, шифрування даних, синхронізації з хмарною базою даних та контролю активності сесії.

6. Спроєктовано інформаційне забезпечення системи шляхом побудови логічної та фізичної структури бази даних, організації механізмів збереження, резервного копіювання та контролю цілісності даних.

7. Обґрунтовано вибір програмних засобів та реалізовано програмну систему управління нотатками з функціями менеджера паролів і спільного доступу із застосуванням технологій C#, .NET, ASP.NET Core Web API та MySQL.

8. Розроблено графічний інтерфейс користувача програмної системи, який забезпечує створення, редагування та організацію нотаток, керування спільним доступом і налаштування параметрів роботи застосунку.

9. Проведено тестування та оцінку якості програмної системи шляхом перевірки працездатності основних функціональних модулів, механізмів безпеки та клієнт-серверної взаємодії, що підтвердило коректність роботи системи, її надійність та відповідність поставленим функціональним і нефункціональним вимогам.

10. У порівнянні з наявними рішеннями розроблена система відрізняється комплексним підходом до поєднання функцій управління нотатками, збереження паролів та організації спільного доступу, що забезпечує підвищення ефективності роботи користувачів та рівня захисту даних.

11. Практична цінність програмної системи SecureNotes, полягає в тому, що вона може бути використана для організації особистих і спільних нотаток, а також для безпечного збереження конфіденційної інформації у навчальній, науковій та професійній діяльності.

12. У подальшому можливе розширення функціональних можливостей системи, зокрема реалізація хмарної синхронізації, створення мобільної версії застосунку, удосконалення механізмів шифрування та впровадженні гнучкої системи керування правами доступу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pitura J. Digital Note-Taking for Writing. Cham: Springer, 2023. 101–119 с.
2. Васильків Н.М. Опорний конспект лекцій з дисципліни «Ефективність інформаційних систем». Тернопіль: Економічна думка, 2005. 98 с.
3. Маргарян Т. А. Система керування нотатками з використанням технологій штучного інтелекту. Київ: КПІ ім. Ігоря Сікорського, 2023. 64 с.
4. Хлепітько К. В. Програмне забезпечення інформаційної системи для керування нотатками користувача. Запоріжжя: Нац. ун-т «Запорізька політехніка», 2025. 64 с.
5. Why You Shouldn't Store Passwords in Note-Taking Apps. MakeUseOf. 2025. URL: https://www.makeuseof.com
6. MVVM Pattern for Windows Desktop Apps. Microsoft Learn. 2024. URL: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm
7. Microsoft OneNote Review. PCMag. 2025. URL: https://www.pcmag.com
8. Microsoft OneNote Review. Cloudwards. 2025. URL: https://www.cloudwards.net
9. Microsoft OneNote Review 2025. CRM.org. 2025. URL: https://crm.org
10. Joplin Review. PCMag. 2025. URL: https://www.pcmag.com
11. Joplin Review: Features, Security, Performance. Cloudwards. 2025. URL: https://www.cloudwards.net
12. Joplin Notes Review. AcademicHelp. 2025. URL: https://academichelp.net
13. Simplenote Review. Cloudwards. 2025. URL: https://www.cloudwards.net
14. Simplenote Review. PCMag. 2023. URL: https://www.pcmag.com
15. The 5 Best Note-Taking Apps for Collaboration and Sharing. MakeUseOf. 2024. URL: https://www.makeuseof.com
16. Авраменко В. С., Авраменко А. С. Проєктування інформаційних систем: навч. посіб. Черкаси: Черкас. нац. ун-т ім. Б. Хмельницького, 2017. 434 с.
17. Date C. J. Вступ до систем баз даних. 8-ме вид. Київ: Діалектика, 2021. 1328 с.

18. PKCS #5: Password-Based Cryptography Specification Version 2.1. RFC 8018. URL: https://tools.ietf.org/html/rfc8018

19. NIST SP 800-132. Recommendation for Password-Based Key Derivation. Gaithersburg: National Institute of Standards and Technology, 2010. URL: https://csrc.nist.gov

20. FIPS PUB 197. Advanced Encryption Standard (AES). Gaithersburg: National Institute of Standards and Technology, 2001. 51 с.

21. Make HTTP requests with HttpClient in .NET. Microsoft Learn. 2024. URL: https://learn.microsoft.com/enus/dotnet/fundamentals/networking/http/httpclient

22. ASP.NET Core documentation. Microsoft Learn. 2024. URL: https://learn.microsoft.com/en-us/aspnet/core

23. Asynchronous programming with async and await. Microsoft Learn. 2024. URL: https://learn.microsoft.com/en-us/dotnet/csharp/asynchronous-programming

24. Security Management and Encryption in MySQL. MySQL Documentation. 2024. URL: https://dev.mysql.com/doc

25. .NET documentation. Microsoft Learn. 2024. URL: https://learn.microsoft.com/en-us/dotnet

26. JSON serialization and deserialization in .NET. Microsoft Learn. 2024. URL: https://learn.microsoft.com/en-us/dotnet/standard/serialization/system-text-json-overview

27. Новак Х.І., Турченко І.В. Система управління нотатками з функціями менеджера паролів та спільного доступу. CSPM 2026: Інтелектуальні обчислювальні системи та управління проєктами: збірник тез доповідей міжнародної науково-практичної конференції студентів і молодих вчених (Тернопіль, 21-22 травня 2026 р). Тернопіль: ЗУНУ, 2026. С.168-170.

28. Комар М. П., Саченко А. О., Васильків Н. М., Гладій Г. М., Коваль В .С., Ліп'яніна-Гончаренко Х. В. Методичні рекомендації до виконання кваліфікаційної роботи зі спеціальності 122 «Комп’ютерні науки». Тернопіль: ЗУНУ, 2024. 51 с.

Додаток А

Основні SQL-запити створення структури бази даних інформаційної системи "SecureNotes"

```
-- Створення бази даних
CREATE DATABASE securenotes_db
CHARACTER SET utf8mb4
COLLATE utf8mb4_0900_ai_ci;

USE securenotes_db;

-- Таблиця користувачів
CREATE TABLE users (
  Id INT NOT NULL AUTO_INCREMENT,
  Username VARCHAR(120) NOT NULL,
  PasswordHash TEXT NOT NULL,
  PasswordSalt TEXT NOT NULL,
  PreferredTheme VARCHAR(20) DEFAULT 'Light',
  CreatedAt DATETIME DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (Id),
  UNIQUE (Username)
) ENGINE=InnoDB;

-- Таблиця груп
CREATE TABLE usergroups (
  Id INT NOT NULL AUTO_INCREMENT,
  Name VARCHAR(255) NOT NULL,
  InviteCode VARCHAR(20) NOT NULL,
  CreatedBy INT NOT NULL,
  CreatedAt DATETIME DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (Id),
  UNIQUE (InviteCode),
  FOREIGN KEY (CreatedBy) REFERENCES users(Id) ON DELETE CASCADE
) ENGINE=InnoDB;

-- Таблиця учасників груп
```

```
CREATE TABLE groupmembers (
  Id INT NOT NULL AUTO_INCREMENT,
  GroupId INT NOT NULL,
  UserId INT NOT NULL,
  Role VARCHAR(20) DEFAULT 'edit',
  PRIMARY KEY (Id),
  UNIQUE (GroupId, UserId),
  FOREIGN KEY (GroupId) REFERENCES usergroups(Id) ON DELETE CASCADE,
  FOREIGN KEY (UserId) REFERENCES users(Id) ON DELETE CASCADE
) ENGINE=InnoDB;

-- Таблиця нотаток
CREATE TABLE notes (
  Id INT NOT NULL AUTO_INCREMENT,
  OwnerId INT NOT NULL,
  GroupId INT NULL,
  Title VARCHAR(255) NOT NULL,
  Content LONGTEXT NOT NULL,
  Type VARCHAR(50) NOT NULL,
  Color VARCHAR(20) DEFAULT '#FFFFFF',
  Tags TEXT,
  IvBase64 TEXT,
  Attachments LONGTEXT,
  CreatedAt DATETIME DEFAULT CURRENT_TIMESTAMP,
  UpdatedAt DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  PRIMARY KEY (Id),
  FOREIGN KEY (OwnerId) REFERENCES users(Id) ON DELETE CASCADE,
  FOREIGN KEY (GroupId) REFERENCES usergroups(Id) ON DELETE SET NULL
) ENGINE=InnoDB;
```

Додаток Б

Вихідні коди основних класів програмної системи SecureNotes

Б.1 Program.cs – код основного класу програми

```
using System;
using System.IO;
using System.Windows.Forms;

namespace SecureNotes
{
   static class Program
   {
      private static readonly string SettingsDirectory = Path.Combine(
          Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData),
         "SecureNotes",
            "users"
      );

      public static bool RestartRequested { get; set; } = false;

      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);

         do
         {
            RestartRequested = false;
            LoadSavedPreferences(null);

            User logged = null;
            using (var login = new LoginForm())
            {
               if (login.ShowDialog() != DialogResult.OK)
               {
                  return;
               }
               logged = login.LoggedInUser;
            }

            if (logged == null) return;

            CurrentUser = logged;
```

```
                LoadSavedPreferences(CurrentUser.Username);

                if (!string.IsNullOrEmpty(logged.PreferredTheme))
                {
                    CurrentTheme = ThemeManager.FromString(logged.PreferredTheme);
                }

                LastActivity = DateTime.Now;
                SessionKey = null;

                Application.ApplicationExit += (s, e) => SaveCurrentPreferences();

                using (var mainForm = new MainForm())
                {
                    Application.Run(mainForm);
                }

            } while (RestartRequested);
        }

        public static User CurrentUser { get; set; }
        public static Theme CurrentTheme { get; set; } = Theme.Light;
        public static DateTime LastActivity { get; set; } = DateTime.Now;
        public static byte[] SessionKey { get; set; }

        public static void TouchActivity()
        {
            LastActivity = DateTime.Now;
        }

        private static string GetSettingsPath(string username)
        {
            var safeUsername = string.IsNullOrWhiteSpace(username) ? "default" :
username.Trim().ToLowerInvariant();
            foreach (var c in Path.GetInvalidFileNameChars())
            {
                safeUsername = safeUsername.Replace(c, '_');
            }

            return Path.Combine(SettingsDirectory, $"{safeUsername}.ini");
        }

        private static void LoadSavedPreferences(string username)
        {
            try
            {
                var settingsPath = GetSettingsPath(username);
```

```
            if (File.Exists(settingsPath))
            {
                var lines = File.ReadAllLines(settingsPath);
                foreach (var line in lines)
                {
                    if (line.StartsWith("Theme="))
                    {
                        var themeName = line.Substring(6).Trim();
                        CurrentTheme = ThemeManager.FromString(themeName);
                    }
                    else if (line.StartsWith("Language="))
                    {
                        var languageName = line.Substring(9).Trim();
                        if (Enum.TryParse(languageName, out Language language))
                        {
                            LocalizationManager.CurrentLanguage = language;
                        }
                    }
                }
            }
        }
        catch { }
    }

    public static void SaveCurrentPreferences()
    {
        try
        {
            if (!Directory.Exists(SettingsDirectory))
Directory.CreateDirectory(SettingsDirectory);
            var lines = new[]
            {
                $"Theme={ThemeManager.GetThemeName(CurrentTheme)}",
                $"Language={LocalizationManager.CurrentLanguage}"
            };

            File.WriteAllLines(GetSettingsPath(CurrentUser?.Username), lines);
        }
        catch { }
    }

    public static void SwitchAccount()
    {
        RestartRequested = true;
        CurrentUser = null;
        SessionKey = null;
```

```
        }
    }
}
```

Б.2 Note.cs – код класу для збереження нотаток

```
using System;

namespace SecureNotes
{
    public class Note
    {
        public int Id { get; set; }
        public int OwnerId { get; set; }
        public int? GroupId { get; set; }
        public string Title { get; set; } = "";
        public string Content { get; set; } = ""; // текст/RTF або шифр
        public string Type { get; set; } = "note"; // note | password
        public string Color { get; set; } = "#FFFFFF";
        public string Tags { get; set; } = "";     // "work;todo"
        public string Attachments { get; set; } = ""; // шляхи до файлів через |
        public DateTime CreatedAt { get; set; } = DateTime.Now;
        public DateTime UpdatedAt { get; set; } = DateTime.Now;
        public string IvBase64 { get; set; } // IV для AES

        public bool IsEncrypted => Type == "password" && !string.IsNullOrEmpty(IvBase64);
    }
}
```

Б.3 User.cs – код класу користувача системи

```
using System;
using SecureNotes;
using System.Drawing;
using System.Windows.Forms;

namespace SecureNotes
{
    public class User
    {
        public int Id { get; set; }
        public string Username { get; set; } = "";
        public string PasswordHash { get; set; } = "";
        public string PasswordSalt { get; set; } = "";
        public string PinHash { get; set; } = "";
```

```
        public string PinSalt { get; set; } = "";
        public DateTime CreatedAt { get; set; } = DateTime.Now;
        public string PreferredTheme { get; set; } = "Light"; // Light|Dark
    }
}
```

Б.4 UIHelpers.cs – код допоміжних методів інтерфейсу

```
using System.Drawing;
using System.Windows.Forms;

namespace SecureNotes
{
    public static class UIHelpers
    {
        // Плейсхолдер керується через Tag, щоб не залежати від кольорів теми
        public static void SetPlaceholder(TextBox box, string placeholderText)
        {
            box.Tag = new PlaceholderState { Text = placeholderText, IsActive = true };
            ApplyPlaceholder(box);

            box.GotFocus += (s, e) =>
            {
                var st = box.Tag as PlaceholderState;
                if (st != null && st.IsActive)
                {
                    box.Text = "";
                    st.IsActive = false;
                    box.ForeColor = ThemeIsDark() ? Color.WhiteSmoke : Color.Black;
                }
            };

            box.LostFocus += (s, e) =>
            {
                var st = box.Tag as PlaceholderState;
                if (st != null && string.IsNullOrWhiteSpace(box.Text))
                {
                    st.IsActive = true;
                    ApplyPlaceholder(box);
                }
            };
        }

        public static bool IsPlaceholder(TextBox box)
        {
```

```
            var st = box.Tag as PlaceholderState;
            return st != null && st.IsActive;
        }

        private static void ApplyPlaceholder(TextBox box)
        {
            var st = box.Tag as PlaceholderState;
            if (st == null) return;
            box.Text = st.Text;
            box.ForeColor = ThemeIsDark() ? Color.Silver : Color.Gray;
        }

        private static bool ThemeIsDark() => Program.CurrentTheme == Theme.Dark;

        private class PlaceholderState
        {
            public string Text { get; set; }
            public bool IsActive { get; set; }
        }
    }
}
```

Додаток В

Основний програмний код прикладного програмного інтерфейсу

.

В.1 Код сервісу криптографічного захисту даних

```
using System.Security.Cryptography;

namespace SecureNotes.Api.Services;

public static class CryptoService
{
    public static string GenerateSalt(int bytes = 16)
    {
        var salt = new byte[bytes];
        using var rng = RandomNumberGenerator.Create();
        rng.GetBytes(salt);
        return Convert.ToBase64String(salt);
    }

    public static string HashWithPBKDF2(string value, string saltBase64, int iterations = 100_000, int bytes = 32)
    {
        var salt = Convert.FromBase64String(saltBase64);
        using var pbkdf2 = new Rfc2898DeriveBytes(value, salt, iterations, HashAlgorithmName.SHA256);
        return Convert.ToBase64String(pbkdf2.GetBytes(bytes));
    }
}
```

В.2 Код сервісу генерації JWT-токенів

```
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
using Microsoft.IdentityModel.Tokens;

namespace SecureNotes.Api.Services;

public sealed class TokenService(IConfiguration config)
{
    public string CreateToken(int userId, string username)
    {
```

```
        var issuer = config["Jwt:Issuer"] ?? throw new InvalidOperationException("Jwt:Issuer missing");
        var audience = config["Jwt:Audience"] ?? throw new InvalidOperationException("Jwt:Audience missing");
        var key = config["Jwt:Key"] ?? throw new InvalidOperationException("Jwt:Key missing");
        var expiresMinutes = int.TryParse(config["Jwt:ExpiresMinutes"], out var minutes) ? minutes : 60;

        var claims = new[]
        {
            new Claim(ClaimTypes.NameIdentifier, userId.ToString()),
            new Claim(ClaimTypes.Name, username)
        };

        var creds = new SigningCredentials(new SymmetricSecurityKey(Encoding.UTF8.GetBytes(key)), SecurityAlgorithms.HmacSha256);

        var token = new JwtSecurityToken(
            issuer: issuer,
            audience: audience,
            claims: claims,
            expires: DateTime.UtcNow.AddMinutes(expiresMinutes),
            signingCredentials: creds);

        return new JwtSecurityTokenHandler().WriteToken(token);
    }
}
```

В.3 Код контролерів автентифікації

```
using Microsoft.AspNetCore.Mvc;
using SecureNotes.Api.Models;
using SecureNotes.Api.Services;

namespace SecureNotes.Api.Controllers;

[ApiController]
[Route("api/[controller]")]
public sealed class AuthController(DbService db, TokenService tokens) : ControllerBase
{
    [HttpPost("register")]
    public ActionResult<AuthResponse> Register([FromBody] RegisterRequest req)
    {
        if (string.IsNullOrWhiteSpace(req.Username) || string.IsNullOrWhiteSpace(req.Password))
            return BadRequest("Username and password are required.");
```

```
            if (db.GetUserByUsername(req.Username) is not null)
                return Conflict("User already exists.");

            var salt = CryptoService.GenerateSalt();
            var hash = CryptoService.HashWithPBKDF2(req.Password, salt);
            var userId = db.CreateUser(req.Username.Trim(), hash, salt);
            var token = tokens.CreateToken(userId, req.Username.Trim());

            return Ok(new AuthResponse { UserId = userId, Username = req.Username.Trim(),
AccessToken = token });
        }

        [HttpPost("login")]
        public ActionResult<AuthResponse> Login([FromBody] LoginRequest req)
        {
            var user = db.GetUserByUsername(req.Username.Trim());
            if (user is null)
                return Unauthorized("Invalid credentials.");

            var calc = CryptoService.HashWithPBKDF2(req.Password, user.PasswordSalt);
            if (!string.Equals(calc, user.PasswordHash, StringComparison.Ordinal))
                return Unauthorized("Invalid credentials.");

            var token = tokens.CreateToken(user.Id, user.Username);
            return Ok(new AuthResponse { UserId = user.Id, Username = user.Username, AccessToken
= token });
        }
    }
```

В.4 Код контролерів управління нотатками

```
using System.Security.Claims;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using SecureNotes.Api.Models;
using SecureNotes.Api.Services;

namespace SecureNotes.Api.Controllers;

[ApiController]
[Authorize]
[Route("api/[controller]")]
public sealed class NotesController(DbService db) : ControllerBase
{
    private int CurrentUserId()
    {
```

```
        var id = User.FindFirstValue(ClaimTypes.NameIdentifier);
        if (!int.TryParse(id, out var userId))
            throw new UnauthorizedAccessException("Invalid token.");
        return userId;
    }

    [HttpGet]
    public ActionResult<List<NoteEntity>> GetMine()
    {
        var userId = CurrentUserId();
        return Ok(db.GetNotesForUser(userId));
    }


    [HttpGet("{id:int}")]
    public ActionResult<NoteEntity> GetById([FromRoute] int id)
    {
        var userId = CurrentUserId();
        var note = db.GetNoteByIdForUser(id, userId);
        return note is null ? NotFound() : Ok(note);
    }

    [HttpPost]
    public ActionResult<object> Create([FromBody] UpsertNoteRequest req)
    {
        var userId = CurrentUserId();
        var id = db.AddNote(userId, req);
        return Ok(new { id });
    }

    [HttpPut("{id:int}")]
    public IActionResult Update([FromRoute] int id, [FromBody] UpsertNoteRequest req)
    {
        var userId = CurrentUserId();
        var ok = db.UpdateNote(id, userId, req);
        return ok ? NoContent() : NotFound();
    }

    [HttpDelete("{id:int}")]
    public IActionResult Delete([FromRoute] int id)
    {
        var userId = CurrentUserId();
        var ok = db.DeleteNote(id, userId);
        return ok ? NoContent() : NotFound();
    }
}
```

Додаток Г

Результати тестування програмної системи SecureNotes

Таблиця Г.1 – Результати тестування створення та редагування нотаток

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Створення нотатки	Заголовок: "Тест", Вміст: "Текст"	Нотатка збережена	Нотатка створена та відображається	Пройдено
2	Створення без заголовку	Заголовок: "", Вміст: "Текст"	Попередження про введення назви	Виведено: "Введіть назву"	Пройдено
3	Редагування нотатки	Зміна заголовку та вмісту	Дані оновлено	Зміни збережено в БД	Пройдено
4	Видалення нотатки	Підтвердження: "Так"	Нотатка видалена	Нотатка видалена з БД та списку	Пройдено
5	Вибір кольору нотатки	Колір: рожевий	Нотатка з рожевим фоном	Карточка відображається з кольором #FFD6E8	Пройдено
6	Додавання тегів	Теги: "work;important"	Теги збережені	Фільтрація за тегами працює	Пройдено

Таблиця Г.2 – Результати тестування форматування

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Жирний текст (Bold)	Виділення + B	Жирний шрифт	Форматування застосовано	Пройдено
2	Курсив (Italic)	Виділення + I	Курсивний шрифт	Форматування застосовано	Пройдено
3	Підкреслення (Underline)	Виділення + U	Підкреслений текст	Форматування застосовано	Пройдено
4	Зміна розміру шрифту	Розмір: 16	Текст 16pt	Розмір змінено	Пройдено
5	Виділення маркером	Highlight	Жовтий фон тексту	Виділення застосовано	Пройдено
6	Скасування дії (Undo)	Ctrl+Z	Попередній стан	Undo працює	Пройдено

Таблиця Г.3 – Результати тестування функцій груп

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Створення групи	Назва: "Команда"	Група створена з invite-кодом	Код згенеровано, група в списку	Пройдено
2	Приєднання за кодом	Код: "100F36E7"	Додано до групи	Нотатки групи доступні	Пройдено
3	Невірний invite-код	Код: "INVALID"	Повідомлення про помилку	"Групу не знайдено"	Пройдено
4	Створення спільної нотатки	Група + нотатка	Видима всім учасникам	Синхронізація між учасниками	Пройдено
5	Вихід з групи	Кнопка виходу	Видалено з групи	Нотатки групи недоступні	Пройдено

Таблиця Г.4 – Результати тестування тем та локалізації

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Тема Light	Вибір теми	Світлі кольори	Фон #FAFBFC застосовано	Пройдено
2	Тема Dark	Вибір теми	Темні кольори	Фон #111115 застосовано	Пройдено
3	Тема Ocean	Вибір теми	Сині відтінки	Фон #0F172A застосовано	Пройдено
4	Мова українська	Вибір мови	Українська локалізація	Всі тексти українською	Пройдено
5	Мова англійська	Вибір мови	Англійська локалізація	Всі тексти англійською	Пройдено
6	Збереження налаштувань	Перезапуск програми	Тема та мова збережені	Налаштування відновлено	Пройдено

Таблиця Г.5 – Результати тестування реєстрації та авторизації

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Успішна реєстрація	Логін: "testuser", Пароль: "pass1234"	Реєстрація успішна, перехід до MainForm	Користувач створений, відкрито головну форму	Пройдено
2	Реєстрація з коротким паролем	Логін: "user2", Пароль: "123"	Повідомлення про мінімум 4 символи	Виведено: "Пароль повинен містити мінімум 4 символи"	Пройдено
3	Реєстрація з порожніми полями	Логін: "", Пароль: ""	Повідомлення про заповнення полів	Виведено: "Введіть логін та пароль"	Пройдено
4	Успішна авторизація	Логін: "testuser", Пароль: "pass1234"	Вхід успішний	Відкрито головну форму	Пройдено
5	Авторизація з невірним паролем	Логін: "testuser", Пароль: "wrong"	Повідомлення про помилку	API повертає помилку авторизації	Пройдено

Таблиця Г.6 – Результати тестування PIN-коду та шифрування

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Встановлення PIN-коду	PIN: "1234"	PIN встановлено	Секція паролів розблокована	Пройдено
2	PIN менше 4 символів	PIN: "12"	Попередження про довжину	Виведено попередження	Пройдено
3	Верифікація правильного PIN	PIN: "1234"	Доступ до паролів	Паролі розшифровано	Пройдено
4	Верифікація невірного PIN	PIN: "0000"	Відмова в доступі	Виведено повідомлення про невірний PIN	Пройдено
5	Шифрування паролю	Вміст: "secret123"	Дані зашифровані AES-256	Вміст зберігається зашифрованим	Пройдено
6	Розшифрування паролю	Збережений запис	Оригінальний текст	Дешифрування успішне	Пройдено

Таблиця Г.7 – Загальні результати тестування

Модуль	Тестів	Пройдено	Відсоток
Автентифікація	5	5	100%
Нотатки	6	6	100%
Захищені паролі	6	6	100%
Групи	5	5	100%
Інтерфейс	6	6	100%
Редактор	6	6	100%
ВСЬОГО	34	34	100%

Додаток Д

Копія опублікованих результатів

Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

Матеріали

І Міжнародної науково-практичної конференції студентів і молодих вчених

«ICSPM 2026: Intelligent Computing Systems and Project Management / Інтелектуальні обчислювальні системи та управління проєктами»

21–22 травня 2026 р.

УДК 004.912:004.056.52:004.77

СИСТЕМА УПРАВЛІННЯ НОТАТКАМИ З ФУНКЦІЯМИ МЕНЕДЖЕРА ПАРОЛІВ ТА СПІЛЬНОГО ДОСТУПУ

Новак Христина Іванівна,
здобувач першого (бакалаврського) рівня вищої освіти,
khrystynanovak3004@gmail.com

Турченко Ірина Василівна,
к.т.н., доцент, доцент кафедри
інформаційнообчислювальних систем і управління,
itu@wunu.edu.ua
ORCID: 0000-0002-9441-6669
Західноукраїнський національний університет

У сучасному світі системи управління нотатками допомагають людям зберігати, упорядковувати та знаходити інформацію у цифровому вигляді, що дає користувачам можливість ефективно працювати з інформацією у різних сферах [1]. Також, важливим аспектом є можливість спільного доступу до інформації, оскільки це необхідно для управління проєктами та організації командної роботи. Водночас, особливе значеннння має інформаційна безпека, тому інтеграція менеджера паролів у систему управління нотатками є актуальним напрямком розвитку, що дозволяє зберегти конфіденційні дані та забезпечити їх захист від несанкціонованого доступу.

На даний час існує багато аналогів програмних систем для управління інформацією. Одним із найпоширеніших інструментів для роботи з нотатками є Microsoft OneNote. Програма входить до складу пакету Microsoft 365 і забезпечує створення текстових та рукописних записів, вставку зображень, аудіо та відео, інтеграцію з Outlook та Teams, а також можливість спільної роботи у реальному часі [2], проте не забезпечує інформаційної безпеки, оскільки відсутній інтегрований менеджер паролів. Іншим прикладом є Joplin, який створений із акцентом на конфіденційність, пропонуючи повне володіння даними завдяки шифруванню та мінімальному збору інформації. Проте у ньому немає інтегрованого менеджера паролів та функції спільного доступу організовані лише частково, також інтерфейс може здатись перевантаженим. Часто використовуваним аналогом є Simplenote, на відміну від Joplin, він має максимально прості екранні форми, проте він не підтримує мультимедійні дані, має слабку політику конфіденційності та не пропонує механізмів захисту інформації на рівні менеджера паролів чи контролю прав доступу. Огляд існуючих продуктів показав, що сучасні системи управління нотатками не

забезпечують комплексного поєднання зручності роботи, інформаційної безпеки та механізмів спільного доступу.

Розробка настільної системи управління нотатками, яка забезпечує створення та редагування нотаток, безпечне збереження паролів із використанням механізмів шифрування, а також організацію спільних нотаток із можливістю створення груп і приєднання до них, є актуальною задачею сучасних інформаційних технологій.

Розроблена програмна система SecureNotes містить три модулі: модуль управління нотатками, модуль управління паролями та модуль управління спільним доступом (рисунок 1).

Рисунок 1 – Структурна схема програмної системи

На рисунку 2 наведено основні дії, які може виконувати користувач у межах програмної системи.

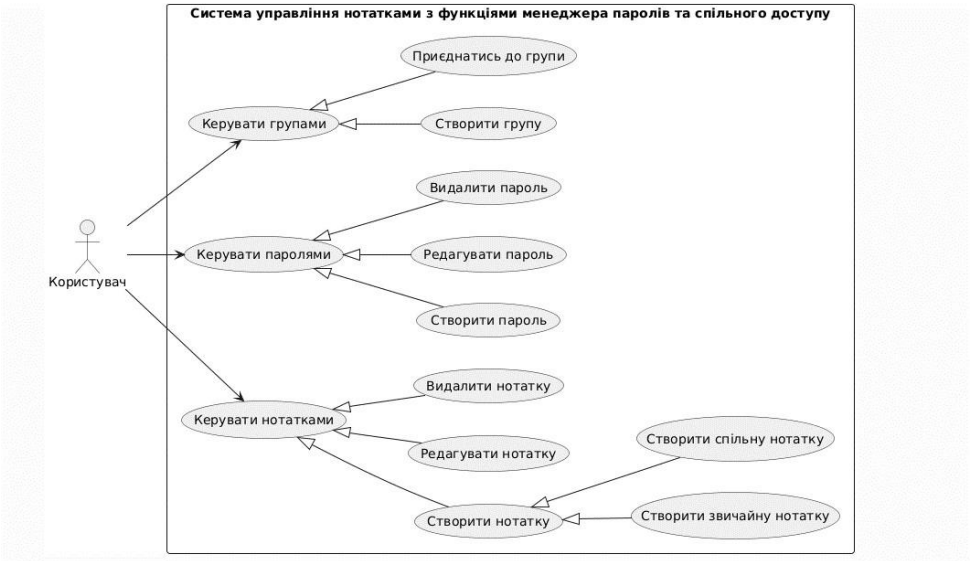

Рисунок 2 – Діаграма варіантів використання

Програмна система SecureNotes реалізована з урахуванням сучасних підходів до створення прикладного програмного забезпечення та вимог до надійності, масштабованості й безпеки інформаційних систем. Обрано мову програмування C# та платформу .NET, яка забезпечує середовище виконання програмного коду, яке відповідає за компіляцію, керування пам'яттю, перевірку типів та контроль безпеки [3]. Інтерфейс користувача розроблено з використанням технології Windows Forms, стандартного інструменту для створення настільних програм для операційної системи Windows. Для фізичного розгортання інформаційного сховища використано хмарну інфраструктуру Aiven на базі СУБД MySQL. Причиною вибору хмари Aiven є необхідність забезпечити високу доступність даних та масштабувати ресурси зі збільшенням обсягу інформації.

Архітектурно програмна система побудована за клієнт-серверною моделлю. Для забезпечення гнучкості та безпеки доступу до даних у трирівневій моделі рівень доступу до даних реалізовано через проміжний шар – прикладний програмний інтерфейс (API). Клієнтський застосунок відповідає за відображення інтерфейсу користувача та формування запитів, серверна частина здійснює централізовану обробку логіки взаємодії та забезпечує контроль доступу до даних, а база даних виконує функцію довготривалого зберігання інформації. Передача інформації здійснюється за допомогою протоколу HTTP, який визначає структуру запитів і відповідей, а також коди стану виконання операцій. Використання текстового формату JSON для обміну даними забезпечує компактність передавання інформації та зручність її обробки як на стороні клієнта, так і на стороні сервера [4].

Інтерфейс SecureNotes забезпечує компактне відображення інформації без перевантаження елементами керування. Створення та редагування нотаток здійснюється в окремому діалоговому вікні, відмінність полягає лише у виборі типу запису, що визначається відповідним перемикачем. Якщо обрано тип запису «пароль», то нотатка автоматично додається до вкладки «Паролі», яка захищена від несанкціонованого доступу. Важливим аспектом є наявність у вікні створення нотаток інструментів для форматування та редагування тексту та підтримка збереження мультимедійних даних і прикріплення зовнішніх файлів. Також є окрема вкладка «Спільні нотатки», у якій є можливість створення чи приєднання до групи, а також відображаються спільні нотатки. Окрім основних форм, система містить окреме вікно налаштувань, призначене для конфігурації параметрів роботи застосунку. У цьому вікні користувач може змінювати параметри відображення інтерфейсу, зокрема тему оформлення. Зміна теми впливає на колірне оформлення елементів керування та загальний вигляд програми. Також, у цьому ж вікні можна перейти на інший акаунт чи змінити пароль до свого акаунту. Додатково, у SecureNotes реалізовано модуль локалізації, що забезпечує підтримку різних мов інтерфейсу.

Практична цінність програмної системи SecureNotes, полягає в тому, що вона може бути використана для організації особистих і спільних нотаток, а також для безпечного збереження конфіденційної інформації у навчальній, науковій та професійній діяльності.

Список використаних джерел

1. Авраменко В. С., Авраменко А. С. Проєктування інформаційних систем: навч. посіб. Черкаси: Черкас. нац. ун-т ім. Б. Хмельницького, 2017. 434 с.
2. Microsoft OneNote Review 2025. CRM.org. 2025. URL: https://crm.org
3. .NET documentation. Microsoft Learn. 2024. URL: https://learn.microsoft.com/en-us/dotnet 4. Make HTTP requests with HttpClient in .NET. Microsoft Learn. 2024. URL: https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/http/httpclient