ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО «ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «МІЖРЕГІОНАЛЬНА АКАДЕМІЯ УПРАВЛІННЯ ПЕРСОНАЛОМ»

Факультет комп'ютерно-інформаційних технологій

Кафедра комп'ютерних інформаційних систем і технологій

Лебедєв Данило Станіславович

(прізвище, ім'я, по-батькові студента)

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Створення системи електронного документообігу підприємства з підтримкою електронного підпису »

Група: ІК-9-22-Б1КНТ (4.0д)

Спеціальність: Комп'ютерні науки

Рівень вищої освіти: перший (бакалаврський)

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

_____ Лебедєв Д.С.

(підпис) ПІБ студента

Науковий керівник _____ Авер'янова Ю.А.

(підпис) ПІБ

Допущено до захисту ЕК

Завідувач кафедри _____ Сергій КАВУН , д.е.н., професор

Київ 2026

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЕДМС / EDMS	Electronic Document Management System - система електронного документообігу
ЕП / ЦЕП	Електронний підпис / Цифровий електронний підпис
SHA-256	Secure Hash Algorithm 256-bit - алгоритм хешування (FIPS 180-4)
RSA	Rivest–Shamir–Adleman - алгоритм асиметричної криптографії
Win32 API	Windows Application Programming Interface
GUI	Graphical User Interface - графічний інтерфейс користувача
UML	Unified Modeling Language - уніфікована мова моделювання
CRUD	Create, Read, Update, Delete - базові операції з даними
NIST	National Institute of Standards and Technology (США)
PKI	Public Key Infrastructure - інфраструктура відкритих ключів
FIPS	Federal Information Processing Standard - федеральний стандарт обробки інформації

ВСТУП

Цифрова трансформація підприємств і державних установ є одним із визначальних трендів сучасності. За даними Gartner, до 2026 року понад 75 % великих підприємств у світі впровадять системи управління контентом із підтримкою електронного підпису [1]. В умовах зростання обсягів документообігу та підвищення вимог до юридичної значущості електронних документів, питання автоматизації процесів підготовки, підписання та зберігання документів набуває першочергового значення.

Традиційний паперовий документообіг має суттєві недоліки: значні часові та матеріальні витрати на друк, доставку та зберігання; відсутність надійних механізмів контролю цілісності; обмежені можливості для дистанційного підписання. Запровадження систем електронного документообігу (ЕДМС) дозволяє суттєво зменшити ці витрати, прискорити бізнес-процеси та підвищити рівень безпеки роботи з документами [2].

Особливої актуальності набуває питання забезпечення юридичної значущості електронних документів. Відповідно до Закону України «Про електронні документи та електронний документообіг» та Закону «Про електронні довірчі послуги», електронний документ, підписаний кваліфікованим електронним підписом, має таку ж юридичну силу, як і паперовий документ із власноручним підписом [3]. Це зумовлює необхідність інтеграції криптографічних механізмів безпосередньо у систему документообігу.

Незважаючи на наявність комерційних рішень - Microsoft SharePoint, M-Files, Alfresco - більшість із них є дорогими у впровадженні, залежать від хмарної інфраструктури або потребують складного технічного обслуговування [8, 9, 10]. Для підприємств середнього та малого бізнесу існує потреба у легковагих, автономних рішеннях із вбудованою підтримкою криптографічних операцій.

Метою даної кваліфікаційної роботи є розробка корпоративної системи електронного документообігу з вбудованою підтримкою електронного

цифрового підпису на основі алгоритмів SHA-256 та RSA, реалізованої засобами мови програмування C++17 з використанням Win32 API.

Для досягнення поставленої мети вирішено такі задачі:

– проаналізовано існуючі системи електронного документообігу та їх підходи до реалізації електронного підпису;

– досліджено криптографічні алгоритми SHA-256 та RSA з точки зору їх придатності для систем електронного підпису;

– спроектовано чотирирівневу архітектуру системи ЕДМС;

– розроблено модуль управління документами з підтримкою версіонування та аудит-журналу;

– реалізовано криптографічний модуль без зовнішніх залежностей;

– розроблено адаптивний графічний інтерфейс користувача на базі Win32 API;

– проведено тестування та оцінено ефективність розробленої системи.

Об'єктом дослідження є процеси електронного документообігу на підприємстві. Предметом дослідження є методи та засоби реалізації системи електронного документообігу з підтримкою електронного цифрового підпису.

Практичне значення роботи полягає у розробці готового програмного рішення, яке може бути впроваджене на підприємстві без використання хмарних сервісів та зовнішніх криптографічних бібліотек - що є суттєвою перевагою для організацій із закритими корпоративними мережами або обмеженим IT-бюджетом.

Структура роботи: кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатку А. Загальний обсяг - 61 сторінок, 5 рисунків, 8 таблиць, 20 джерел.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Поняття та значення електронного документообігу

Електронний документообіг (ЕДО) являє собою комплексну інформаційну систему, призначену для управління повним життєвим циклом документів у цифровому форматі. Вона охоплює всі ключові етапи роботи з документацією: від створення, редагування та погодження до маршрутизації, затвердження, архівування, швидкого пошуку та остаточного знищення. На відміну від традиційних паперових процесів, ЕДО інтегрує автоматизовані інструменти контролю, що дозволяє відстежувати статус кожного документа в реальному часі, забезпечувати виконання доручень у встановлені терміни та формувати аналітичну звітність без ручного втручання. Усі операції відбуваються з суворим дотриманням нормативно-правових вимог, що гарантує юридичну значущість електронних файлів на рівні паперових оригіналів [4].

Фундаментальні засади роботи з документами в Україні регулюються державними стандартами, зокрема ДСТУ 4423:2005, який визначає документ як інформацію, зафіксовану на матеріальному носії, основними функціями якого є зберігання та передача відомостей у часі та просторі [5]. У цифровому середовищі фізичний носій замінюється захищеним електронним сховищем, а власноручний підпис – криптографічними механізмами захисту. Електронний документ набуває повної юридичної сили лише за умови застосування кваліфікованого електронного підпису (КЕП), який базується на асиметричній криптографії та сертифікованих засобах підписання. Це дозволяє гарантувати не лише ідентифікацію автора, а й технічну незмінність вмісту файлу після його підписання, що є критично важливим для судових та перевіряючих органів.

Впровадження систем ЕДО має очевидну економічну та організаційну доцільність, що підтверджується численними міжнародними дослідженнями. Зокрема, за даними AIIM (Association for Intelligent Information Management),

перехід на цифровий документообіг дозволяє скоротити операційні витрати на роботу з документацією на 30–50 %, зменшити час пошуку необхідних файлів на 80 % та зекономити до 40 % бюджету, що раніше витрачався на фізичне зберігання архівів [4]. Крім фінансових переваг, ЕДО значно підвищує рівень прозорості бізнес-процесів: системи автоматично фіксують історію змін, відстежують версійність, забезпечують відповідність регуляторним вимогам (наприклад, ISO 15489, GDPR або національним стандартам захисту даних) та спрощують проведення внутрішніх і зовнішніх аудитів завдяки централізованим логам дій.

Ключовою технологічною та юридичною складовою сучасного ЕДО є електронний підпис, який виконує три фундаментальні функції безпеки. По-перше, він забезпечує автентифікацію підписанта, тобто підтверджує особу, яка ініціювала підписання, на основі криптографічних ключів та сертифікатів від акредитованих центрів сертифікації ключів. По-друге, гарантує цілісність документа: будь-яка спроба несанкціонованого редагування після підписання миттєво призводить до порушення криптографічного хеша, що робить зміни помітними та юридично недійсними. По-третє, реалізується принцип неспростовності (non-repudiation), згідно з яким підписант не може відмовитися від факту підписання, оскільки приватний ключ зберігається виключно у нього, захищений апаратними або програмними токенами та регулюється відповідними нормами законодавства.

Успішне впровадження ЕДО вимагає не лише технічної інтеграції з існуючими ERP-, CRM- та бухгалтерськими системами, а й адаптації внутрішніх регламентів, навчання персоналу та регулярного оновлення засобів захисту інформації. Сучасні платформи поступово доповнюються модулями штучного інтелекту для автоматичної класифікації документів, розпізнавання текстів (OCR), виділення реквізитів та прогнозування термінів виконання, що робить цифровий документообіг ще більш ефективним, масштабованим та відповідним вимогам Industry 4.0.

1.2 Правова база електронного документообігу в Україні

Правова основа функціонування електронного документообігу в Україні формується на базі чіткої багаторівневої ієрархії нормативно-правових актів, які комплексно регулюють як організаційно-управлінські, так і техніко-технологічні аспекти цього процесу. Така ієрархія включає закони України, підзаконні акти Кабінету Міністрів, нормативні документи центральних органів виконавчої влади, а також національні та міжнародні стандарти. Ця система забезпечує правове визнання електронних документів, гарантує їхню юридичну силу, встановлює вимоги до безпеки інформації та створює передумови для інтеграції українських цифрових сервісів у європейський та глобальний простір. Фундаментальним документом у цій сфері є Закон України «Про електронні документи та електронний документообіг» від 22 травня 2003 року № 851-IV, який закладає базові організаційно-правові засади створення, обробки, відправлення, передачі, одержання, зберігання, використання та знищення електронних документів. Цей нормативний акт юридично легалізує поняття «електронний документ», прирівнюючи його до паперового оригіналу за умови дотримання визначених вимог щодо автентичності, цілісності та можливості відтворення інформації. Закон також встановлює уніфіковані правила ведення електронного документообігу в органах державної влади, місцевого самоврядування, на підприємствах, в установах і організаціях незалежно від форми власності, що забезпечує єдиний підхід до цифровізації документообігу в усіх сферах суспільного життя [3].

Важливим етапом модернізації національного законодавства у сфері цифрової довіри стало прийняття Закону України «Про електронні довірчі послуги» від 5 жовтня 2017 року № 2155-VIII, який замінив попередній закон «Про електронний цифровий підпис» та розширив спектр регульованих послуг. Цей нормативний акт визначає правовий статус електронних довірчих послуг, зокрема кваліфікованого електронного підпису (КЕП), кваліфікованої

електронної печатки, кваліфікованих електронних відміток часу, кваліфікованої електронної рекомендованої доставки та послуг валідації. Закон встановлює чіткі вимоги до постачальників таких послуг (Центрів сертифікації ключів), включаючи порядок їх акредитації, нагляд за діяльністю та відповідальність за порушення. Ключовою особливістю цього закону є його цілеспрямована гармонізація з європейським законодавством, зокрема з Регламентом ЄС eIDAS (Regulation (EU) No 910/2014), що регулює електронну ідентифікацію та довірчі послуги на внутрішньому ринку Європейського Союзу. Така відповідність забезпечує механізми взаємного визнання кваліфікованих електронних підписів між Україною та країнами-членами ЄС, що сприяє інтеграції українського бізнесу, державних органів та громадян у єдиний цифровий простір Європи, спрощує транскордонні операції та підвищує довіру до українських електронних сервісів на міжнародному рівні [6], [17].

Технічна реалізація захисту електронних документів регламентується національними стандартами та нормативними актами у сфері криптографічного захисту інформації, які розробляються та затверджуються уповноваженими органами. Зокрема, ДСТУ 4145-2002 визначає алгоритм цифрового підпису на базі еліптичних кривих, який є обов'язковим для використання в системах, що застосовуються в органах державної влади, органах місцевого самоврядування, на об'єктах критичної інфраструктури та в інших сферах, де встановлено підвищені вимоги до захисту інформації. Цей стандарт ґрунтується на математичній складності задачі дискретного логарифмування в групі точок еліптичної кривої над скінченним полем, що забезпечує високий рівень криптографічної стійкості при відносно невеликій довжині ключа. Вимоги до засобів криптографічного захисту, включаючи алгоритми шифрування, хешування, генерації псевдовипадкових чисел та управління ключами, також встановлюються нормативними документами Адміністрації Державної служби спеціального зв'язку та захисту інформації України (Держспецзв'язку). Ці стандарти гарантують стійкість

криптографічних алгоритмів до сучасних загроз, включаючи атаки методом грубої сили, криптоаналізу та побічних каналів, а також забезпечують сумісність програмних рішень з державними реєстрами, системами електронної взаємодії органів влади та іншими офіційними інформаційними ресурсами [5].

У контексті розробки програмного забезпечення для електронного документообігу важливо враховувати не лише поточні нормативні вимоги, а й архітектурні принципи, що дозволяють гнучко адаптувати систему до майбутніх змін у законодавстві, стандартах або технологіях. У демонстраційній версії системи для реалізації механізмів підписання використано алгоритм RSA, який є широко поширеним міжнародним стандартом, добре документованим та підтримуваним у більшості криптографічних бібліотек. Однак архітектура системи побудована з урахуванням принципу розділення відповідальності (Separation of Concerns), який є одним із фундаментальних принципів об'єктно-орієнтованого та модульного проектування. Це означає, що криптографічний модуль, що відповідає за генерацію ключів, підписання та верифікацію, ізольований від бізнес-логіки додатка через абстрактні інтерфейси та залежить лише від чітко визначених контрактів, а не від конкретних реалізацій алгоритмів. Такий підхід дозволяє замінити реалізацію алгоритму RSA на вітчизняний стандарт ДСТУ 4145-2002 або міжнародний стандарт ECDSA без необхідності переписування основного коду системи, зміни інтерфейсів користувача або порушення цілісності бізнес-процесів. Це забезпечує високу масштабованість, підтримуваність та відповідність системи майбутнім законодавчим змінам, технічним оновленням або вимогам конкретних замовників, що є критично важливим для довгострокової експлуатації програмного забезпечення в динамічному регуляторному середовищі [19].

1.3 Аналіз існуючих систем електронного документообігу

На сучасному ринку програмного забезпечення представлено значну кількість рішень для організації електронного документообігу (ЕДО), які відрізняються за архітектурою, функціональними можливостями, вартістю володіння та сферами застосування. Вибір оптимальної системи вимагає ретельного порівняльного аналізу, що враховує не лише поточні потреби підприємства, а й перспективи масштабування, інтеграції з існуючою IT-інфраструктурою, а також вимоги до безпеки та відповідності нормативним актам. Нижче наведено детальний огляд трьох поширених платформ: Microsoft SharePoint, M-Files та Alfresco, з акцентом на їхні технічні, економічні та експлуатаційні характеристики.

Microsoft SharePoint є одним із найбільш розповсюджених корпоративних рішень для управління документами та спільної роботи, особливо в організаціях, що вже використовують екосистему Microsoft 365. Платформа пропонує широкий функціонал: централізоване сховище документів із підтримкою версіонування, гнучке управління правами доступу на основі ролей (RBAC), автоматизацію бізнес-процесів через Power Automate, повнотекстовий пошук із використанням Azure Cognitive Search, а також інтеграцію з Microsoft Teams, Outlook та іншими сервісами [8]. Однак SharePoint має суттєві обмеження: система орієнтована переважно на хмарне розгортання (SharePoint Online), що може бути неприйнятним для організацій із суворими вимогами до локалізації даних; вартість ліцензування варіюється від $5 до $20 на користувача на місяць залежно від тарифного плану, що при великій кількості користувачів формує значні операційні витрати; підтримка кваліфікованого електронного підпису (КЕП) не реалізована «з коробки» та потребує інтеграції зі сторонніми рішеннями, такими як DocuSign, Adobe Sign або вітчизняними провайдерами (КНЕДП «ІІТ», «ЦСК-1» тощо), що додатково ускладнює архітектуру та підвищує загальну вартість володіння.

M-Files - це інтелектуальна система управління документами, архітектурна особливість якої полягає в орієнтації на метадані замість традиційної ієрархії папок [9]. Такий підхід дозволяє автоматично

класифікувати, пов'язувати та знаходити документи на основі їхнього змісту, контексту та атрибутів, що значно підвищує ефективність пошуку та зменшує ризик втрати інформації. Система підтримує інтеграцію з PKI-інфраструктурою для реалізації електронного підпису, має вбудовані механізми аудиту, робочі процеси (workflow) та мобільний доступ. Проте M-Files є комерційним продуктом із закритим вихідним кодом, що обмежує можливості глибокої адаптації під специфічні бізнес-процеси підприємства, особливо в умовах українського законодавства щодо ЕДО та криптографічного захисту. Впровадження системи вимагає значних початкових інвестицій у ліцензії, налаштування, міграцію даних та навчання персоналу, а також передбачає регулярні витрати на технічну підтримку та оновлення, що може бути критичним для малих та середніх підприємств.

Alfresco - це потужна платформа управління корпоративним контентом (ECM) з відкритим вихідним кодом, побудована на технологічному стеку Java (Spring Framework, Hibernate, Apache Solr) [10]. Система надає модульну архітектуру, гнучкий REST API для інтеграції з зовнішніми системами (ERP, CRM, бухгалтерськими рішеннями), підтримку стандартів CMIS, а також базові функції роботи з електронним підписом через інтеграцію з криптографічними бібліотеками. Відкритий код дозволяє адаптувати платформу під будь-які специфічні вимоги, включаючи підтримку національних стандартів шифрування (ДСТУ) та локальних провайдерів електронного підпису. Основними викликами при використанні Alfresco є висока складність розгортання та налаштування: система вимагає наявності спеціалізованого персоналу з досвідом адміністрування Java-додатків, серверів застосувань (Tomcat, JBoss), систем керування базами даних (PostgreSQL, MySQL) та пошукових індексів. Крім того, відсутність офіційної технічної підтримки в безкоштовній версії (Community Edition) може збільшити ризики при експлуатації в критичних бізнес-процесах, тому для корпоративного використання часто обирають платну версію (Enterprise Edition) з гарантованим сервісом та оновленнями.

Підсумовуючи порівняльний аналіз, можна констатувати, що вибір системи ЕДО має ґрунтуватися на комплексній оцінці технічних, економічних та організаційних факторів. Microsoft SharePoint є оптимальним вибором для організацій, глибоко інтегрованих у екосистему Microsoft, що готові до хмарного розгортання та мають бюджет на ліцензування. M-Files підходить для компаній, які потребують інтелектуального управління документами на основі метаданих та готові інвестувати в комерційне рішення з повною підтримкою. Alfresco, зі своєю відкритою архітектурою, є найкращим варіантом для організацій, що прагнуть максимальної гнучкості, контролю над кодом та можливості адаптації під національні стандарти, за умови наявності кваліфікованої IT-команди для супроводу системи.

На рис. 1.1 наведено порівняльну таблицю характеристик систем документообігу.

Критерій	SharePoint	M-Files	Alfresco	EDMS (наша)
Електронний підпис	✓ (зовнішній)	✓ (зовнішній)	Частково	✓ (вбудований)
Верифікація цілісності	Частково	✓	✓	✓ SHA-256
Незалежність від ОС	✗ (Windows)	✓	✓	Windows API
Зовнішні залежності	Так (M365)	Так	Java, DB	Відсутні
Версіонування документів	✓	✓	✓	✓
Відкритий код	✗	✗	✓	✓

✓ — підтримується, ✗ — не підтримується

Рис. 1.1 - Порівняльна характеристика систем електронного документообігу

Проведений аналіз існуючих рішень для електронного документообігу дозволяє виокремити низку системних обмежень, які ускладнюють їх застосування в умовах українських підприємств, особливо малого та середнього бізнесу, а також організацій із підвищеними вимогами до автономності та безпеки інформації. Ці обмеження стосуються як архітектурних рішень, так і економічних, експлуатаційних та нормативних аспектів. Нижче наведено детальне розкриття кожного з ідентифікованих недоліків із обґрунтуванням необхідності розробки альтернативного підходу.

Одним із ключових обмежень багатьох сучасних систем ЕДО є їхня залежність від хмарних сервісів або зовнішньої PKI-інфраструктури. Такі рішення, як Microsoft SharePoint Online або M-Files Cloud, вимагають постійного підключення до Інтернету та інтеграції з зовнішніми центрами сертифікації ключів для реалізації функцій електронного підпису. Це створює ризики недоступності системи під час мережевих збоїв, обмежує можливість роботи в умовах автономного режиму або закритих контурів безпеки (наприклад, на об'єктах критичної інфраструктури, у військових підрозділах або організаціях із суворими вимогами до локалізації даних). Крім того, залежність від зовнішніх постачальників довірчих послуг може ускладнити дотримання вимог національного законодавства щодо захисту персональних даних та конфіденційної інформації.

Висока вартість ліцензування та впровадження є ще одним суттєвим бар'єром для широкого впровадження корпоративних систем ЕДО. Комерційні рішення, такі як M-Files або Microsoft SharePoint Enterprise, передбачають не лише щомісячні або річні ліцензійні платежі (від $5 до $50 на користувача), а й значні одноразові витрати на налаштування, міграцію даних, інтеграцію з існуючими системами та навчання персоналу. Для малих підприємств, стартапів або громадських організацій такі інвестиції часто є непосильними, що змушує їх або відмовлятися від автоматизації документообігу, або використовувати непрофільні інструменти (електронна пошта, хмарні сховища), які не забезпечують належного рівня безпеки, аудиту та відповідності нормативним вимогам.

Складність адміністрування та розгортання характерна для багатьох потужних платформ, зокрема Alfresco, які побудовані на складному технологічному стеку (Java, СУБД, пошукові індекси, сервери застосувань). Для підтримки таких систем необхідна наявність кваліфікованого IT-персоналу з досвідом роботи з відповідними технологіями, що збільшує операційні витрати та створює залежність від конкретних фахівців. Крім того, процес оновлення, резервного копіювання та відновлення після збоїв у таких

системах є нетривіальним завданням, що вимагає детальної документації та відпрацьованих регламентів, які не завжди доступні для відкритих рішень.

Відсутність вбудованих механізмів верифікації цілісності документів без використання зовнішньої PKI-інфраструктури є критичним недоліком для сценаріїв, де необхідно забезпечити автентичність та незмінність документа в автономному режимі. Більшість існуючих систем покладаються на інтеграцію з центрами сертифікації ключів або сторонніми сервісами електронного підпису, що унеможливлює перевірку документа без підключення до зовнішніх ресурсів. Це обмежує застосування таких рішень у ситуаціях, коли документ має бути верифікований локально, наприклад, при офлайн-перевірці договорів, актів або звітів без доступу до Інтернету.

Значна кількість зовнішніх залежностей (бібліотеки Java, системи керування базами даних, хмарні сервіси, фреймворки) ускладнює супровід, оновлення та безпеку системи. Кожна з таких залежностей може стати джерелом вразливостей, конфліктів версій або несумісності при оновленні операційної системи чи іншого програмного забезпечення. Крім того, наявність численних зовнішніх компонентів збільшує «атакувальну поверхню» системи та ускладнює проведення аудиту безпеки, оскільки необхідно аналізувати не лише власний код, а й усі інтегровані бібліотеки та їхні залежності.

Таким чином, існує об'єктивна та актуальна потреба у розробці легковагої системи електронного документообігу (EDMS), яка б поєднувала в собі вбудовану підтримку криптографічних операцій, мінімальну кількість зовнішніх залежностей та можливість функціонування як повністю автономне рішення. Така система має забезпечувати базові функції створення, зберігання, підписання та верифікації документів без необхідності підключення до зовнішніх сервісів, при цьому підтримуючи як міжнародні алгоритми (RSA, ECDSA), так і національні стандарти (ДСТУ 4145-2002). Архітектура такого рішення має ґрунтуватися на принципах модульності, інкапсуляції криптографічної логіки та чіткого розділення відповідальності,

що дозволить у майбутньому гнучко адаптувати систему до змін у законодавстві або технологічних вимогах без необхідності повної переробки кодової бази. Розробка подібного інструменту сприятиме підвищенню доступності, безпеки та відповідності систем ЕДО для широкого кола українських організацій, особливо в умовах обмежених ресурсів та підвищених вимог до автономності та захисту інформації.

1.4 Криптографічні алгоритми для реалізації електронного підпису

Функціонування сучасних систем електронного підпису ґрунтується на комбінації двох фундаментальних криптографічних примітивів: криптографічних хеш-функцій та алгоритмів асиметричного шифрування. Ця архітектурна модель забезпечує комплексний захист документів, гарантуючи одночасно цілісність інформації, автентифікацію особи підписанта та юридичну неспростовність операції. Хеш-функція виконує роль «цифрового відбитка» документа, стискаючи дані довільного обсягу до фіксованої бітової послідовності, тоді як асиметрична криптографія забезпечує безпечне зв’язування цього відбитка з конкретним користувачем за допомогою математично пов’язаної пари ключів. Такий підхід дозволяє відокремити процеси швидкого обчислення дайджесту від ресурсомістких операцій шифрування, що суттєво підвищує продуктивність системи без втрати рівня безпеки.

У якості криптографічного хеш-алгоритму в більшості сучасних реалізацій електронного підпису застосовується SHA-256 (Secure Hash Algorithm 256-bit), який є офіційно стандартизованим у документі FIPS PUB 180-4 та входить до складу сімейства алгоритмів SHA-2 [11]. На відміну від функцій стиснення або кодування, SHA-256 приймає на вхід повідомлення будь-якої довжини та генерує унікальний криптографічний дайджест фіксованого розміру – рівно 256 біт (32 байти). Архітектурна надійність алгоритму базується на чотирьох ключових властивостях: детермінованості, що гарантує отримання ідентичного хешу для однакових вхідних даних;

стійкості до колізій, яка робить обчислювально неможливим підбір двох різних повідомлень із однаковим дайджестом; лавинному ефекті, завдяки якому зміна навіть одного біта вхідного потоку призводить до кардинальної зміни приблизно половини бітів результату, що виключає можливість передбачення або контролю виходу; та односторонності (стійкості до прообразу), яка унеможливлює відновлення оригінальних даних на основі відомого хешу. Ці характеристики роблять SHA-256 оптимальним інструментом для фіксації незмінного стану документа перед його підписанням [11].

Алгоритм RSA, розроблений у 1978 році Рональдом Рівестом, Аді Шаміром та Леонардом Адлеманом, став першим практично придатним рішенням у сфері асиметричної криптографії та й досі залишається одним із найпоширеніших стандартів для формування електронних підписів [12]. Його криптографічна стійкість базується на математичній складності задачі факторизації великих цілих чисел, тобто розкладання добутку двох великих простих чисел на їхні первинні множники. Генерація ключової пари починається з вибору двох випадкових простих чисел p та q, на основі яких обчислюється модуль $n = p \cdot q$. Далі визначається відкрита експонента e (зазвичай використовується значення 65537 через його ефективність у бінарному представленні та достатню криптографічну стійкість), а закрита експонента d обчислюється як модульне обернене до e за модулем функції Ейлера $\varphi(n) = (p-1)(q-1)$, тобто $d \equiv e^{-1} \pmod{\varphi(n)}$. Відкритий ключ (n, e) може вільно поширюватися та використовується для верифікації підписів або шифрування даних, тоді як закритий ключ (n, d) зберігається виключно у власника та застосовується для створення підписів або розшифровки [12].

Вибір довжини криптографічного ключа є критичним параметром, що безпосередньо впливає на рівень безпеки системи та стійкість до атак перебору чи факторизації. Відповідно до рекомендацій NIST SP 800-57, мінімальний допустимий розмір ключа RSA для забезпечення рівня безпеки, еквівалентного 112 бітам симетричного шифрування, становить 2048 біт [7].

Використання ключів меншої довжини вважається криптографічно слабким у сучасних умовах, оскільки зростання обчислювальних потужностей та вдосконалення алгоритмів факторизації (наприклад, загальний метод решета числового поля) поступово знижують час, необхідний для злому. У розробленій навчальній системі для демонстрації фундаментальних принципів роботи алгоритму було свідомо використано менші ключі, згенеровані на основі двох 15-значних простих чисел. Таке рішення є цілком прийнятним у тестовому середовищі, оскільки дозволяє наочно ілюструвати математичні перетворення та процеси підписання/верифікації без необхідності застосування важких обчислювальних ресурсів, проте для промислового експлуатування обов’язково має використовуватися ключ довжиною мінімум 2048 біт або 3072 біт для довгострокового захисту даних [7].

Процес формування та перевірки електронного підпису з використанням комбінації SHA-256 та RSA відбувається у кілька чітко визначених етапів, що забезпечують математичну доведеність автентичності та цілісності документа. На етапі підписання система спочатку обчислює криптографічний дайджест вмісту документа за алгоритмом SHA-256, отримуючи значення h = SHA-256(doc). Отриманий хеш потім перетворюється на підпис за допомогою закритого ключа підписанта відповідно до формули s ≡ h^d (mod n). Оскільки операція піднесення до степеня d виконується лише з використанням секретного компонента ключової пари, підпис s є унікальним та математично пов’язаним як із змістом документа, так із особою підписанта [12]. Під час верифікації отримувач або автоматизована система виконують зворотне перетворення, використовуючи відкритий ключ підписанта: обчислюють h' ≡ s^e (mod n). Після цього отримане значення h' порівнюється з незалежно обчисленим хешем оригінального документа SHA-256(doc). Якщо h' == h, це математично підтверджує, що документ не зазнавав змін після підписання, а підпис був сформований саме володарем відповідного закритого ключа. Будь-яка спроба

модифікації навіть одного байта в документі призведе до зміни хешу та, як наслідок, до неузгодженості значень h' та h, що автоматично блокує верифікацію та інформує про порушення цілісності [12].

1.5 Вибір технологій розробки

На основі проведеного аналізу вимог до архітектури та експлуатаційних характеристик системи обрано наступний технологічний стек, який детально обґрунтовано нижче.

Вибір мови програмування C++17 як основної платформи розробки обумовлений низкою технічних та архітектурних переваг, що безпосередньо відповідають вимогам до створення легковагої та автономної програмної системи. C++17 забезпечує високу продуктивність виконання коду завдяки компіляції в нативний машинний код та повній відсутності накладних витрат середовища виконання (runtime overhead), що є характерним для керованих мов на кшталт C# та .NET або Java. Крім того, стандарт C++17 суттєво розширює можливості стандартної бібліотеки STL, пропонуючи сучасні інструменти для ефективної роботи з рядками, контейнерами, алгоритмами та файловими системами (std::filesystem), а також нові конструкції типу std::optional, std::variant та структуровані прив'язки, які підвищують читабельність, безпеку та підтримуваність коду [13]. Це дозволяє реалізувати криптографічні операції та обробку документів з мінімальним споживанням оперативної пам'яті та максимальним контролем над системними ресурсами, що є критично важливим для рішень, призначених для роботи в обмежених або ізольованих середовищах.

Для реалізації графічного інтерфейсу користувача (GUI) обрано нативний Win32 API замість популярних кросплатформових або високорівневих фреймворків, таких як MFC, Qt або wxWidgets. Основним аргументом на користь цього рішення є повна відсутність зовнішніх DLL-залежностей, що дозволяє компілювати систему в єдиний самодостатній виконуваний файл (single-file deployment), який не вимагає інсталяції

додаткових середовищ виконання або сторонніх бібліотек. Win32 API забезпечує повну сумісність з усіма актуальними версіями Windows 10 та Windows 11, а також надає прямий доступ до системних механізмів рендерингу, черги повідомлень та управління вікнами. Використання бібліотеки Common Controls дозволяє інтегрувати всі необхідні елементи інтерфейсу, зокрема ListView для відображення реєстрів документів, TabControl для навігації між функціональними модулями та StatusBar для відображення статусних повідомлень, без залучення зовнішніх UI-фреймворків [14]. Такий підхід мінімізує розмір дистрибутива, прискорює завантаження додатка та підвищує його стабільність у корпоративних середовищах із суворими політиками встановлення програмного забезпечення.

Як інтегроване середовище розробки (IDE) обрано Visual Studio 2022, яке пропонує повноцінну підтримку стандарту C++17 та потужний компілятор MSVC із суворим контролем типів, розширеним статичним аналізом коду та автоматичним виявленням потенційних вразливостей ще на етапі написання коду. Середовище включає в себе інтегрований відладчик з підтримкою налагодження на рівні машинного коду, інструментів діагностики пам'яті та візуалізації багатопоточних процесів, а також профілювальник продуктивності, що дозволяє точно ідентифікувати «вузькі місця» в алгоритмах шифрування та обробки даних. Важливим фактором є ліцензійна модель: версія Visual Studio Community є повністю безкоштовною для освітніх, дослідницьких та некомерційних проектів, що робить її оптимальним вибором для академічних розробок. Глибока інтеграція з CMake, Git та вбудованими інструментами тестування забезпечує сучасний цикл розробки, дотримання принципів контролю якості та високу стабільність кінцевого продукту.

На основі проведеного комплексного аналізу встановлено, що сучасний ринок систем електронного документообігу є технологічно зрілим, проте більшість комерційних та відкритих рішень мають суттєві експлуатаційні

обмеження, зокрема залежність від хмарних сервісів, зовнішньої PKI-інфраструктури, високу вартість володіння та складність розгортання. Визначено чітку правову базу функціонування ЕДО в Україні, яка базується на Законах № 851-IV та № 2155-VIII, гармонізована з Регламентом ЄС eIDAS та встановлює вимоги до юридичної значущості електронних документів і довірчих послуг. Для реалізації криптографічної підсистеми обрано алгоритми SHA-256 (згідно з FIPS PUB 180-4) та RSA, які забезпечують необхідний рівень безпеки, цілісності та автентичності даних без залучення зовнішніх криптографічних бібліотек. Технологічний стек, що поєднує C++17 та нативний Win32 API, обрано як оптимальне рішення, яке дозволяє створити легковагу, автономну та високопродуктивну систему ЕДМС, здатну функціонувати в ізольованих мережах, відповідати національним та європейським нормативним вимогам та забезпечувати повний контроль над програмним кодом і обробкою конфіденційної інформації.

РОЗДІЛ 2 ПРОЕКТУВАННЯ СИСТЕМИ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

2.1 Функціональні та нефункціональні вимоги

2.1.1 Функціональні вимоги

На основі комплексного аналізу предметної галузі та практичних потреб потенційних користувачів сформульовано функціональні вимоги до системи відповідно до підходу Use Case-driven requirements [19]. Цей методологічний підхід забезпечує пряму трасованість від сценаріїв використання до технічних специфікацій, гарантуючи, що кожна реалізована функція безпосередньо вирішує конкретну бізнес-задачу, мінімізує надлишковість коду та полегшує подальше тестування і супровід продукту. Нижче наведено детальне розкриття кожного з визначених функціональних блоків.

1. Управління документами охоплює повний життєвий цикл роботи з файлами: створення, перегляд, редагування, архівування та видалення. Система підтримує уніфіковану роботу з ключовими типами документів, включаючи Договір, Наказ, Рахунок-фактура, Звіт, Меморандум, Протокол, Лист та універсальну категорію «Інше», що дозволяє адаптувати інтерфейс, шаблони метаданих та правила обробки під специфіку кожного виду. Для кожного файлу автоматично формується картка з обов'язковими атрибутами (назва, автор, дата створення, поточний статус, тип, шлях до файлу), що забезпечує структуроване зберігання та спрощує подальшу автоматизовану обробку. Операції архівування та видалення реалізуються з урахуванням вимог інформаційної безпеки: видалення застосовує механізм «м'якого» стирання (logical delete) зі збереженням запису в системі для відповідності регуляторним нормам щодо зберігання історії, тоді як архівування переводить документ у режим лише для читання з оптимізацією займаного дискового простору.

2. Версіонування реалізовано як автоматичний механізм фіксації змін, який спрацьовує при кожному збереженні або редагуванні вмісту документа.

Система створює незмінні знімки (snapshots) файлу, супроводжуючи їх розширеними метаданими: ідентифікатором автора зміни, точною часовою міткою (timestamp) та коментарем користувача, що описує характер внесених правок. Такий підхід забезпечує повну прозорість історії редагувань, дозволяє відкочуватися до будь-якої попередньої версії у разі помилок, несумісних змін або дій недобросовісних користувачів, а також спрощує контроль якості документів. Архітектурно версії зберігаються із застосуванням інкрементальних або повних копій, кожна з яких має власний хеш-ідентифікатор, що гарантує криптографічну незмінність історії навіть у разі апаратних або програмних збоїв.

3. Електронний підпис є критичним функціональним модулем, що забезпечує юридичну значущість, автентичність та захист документів від несанкціонованих змін. Процес підписання базується на комбінації алгоритмів SHA-256 та RSA: спочатку обчислюється криптографічний хеш вмісту файлу, після чого отриманий дайджест шифрується приватним ключем підписанта з урахуванням параметрів модуля та експоненти. Верифікація виконується автоматично при відкритті, надсиланні або експорті документа: система розшифровує підпис відповідним відкритим ключем, порівнює результат із поточним хешем файлу та візуально індикатором повідомляє про статус валідності. Реалізація цього механізму гарантує цілісність даних, підтвердження особи підписанта та принцип неспростовності, а також підтримує повноцінну роботу в автономному режимі без звернення до зовнішніх серверів сертифікації або хмарних сервісів.

4. Управління користувачами забезпечує контроль доступу до системи через комплекс механізмів реєстрації, автентифікації та авторизації. Реєстрація нових облікових записів супроводжується валідацією вхідних даних та безпечним хешуванням паролів із застосуванням криптографічно стійких алгориттів та сольових механізмів. Автентифікація здійснюється через захищений сеанс із контролем сесійних токенів, таймаутом неактивності та захистом від атак методом перебору. Розмежування прав

реалізовано на основі рольової моделі (Role-Based Access Control, RBAC), яка включає ролі Адміністратора, Керівника, Співробітника та Аудитора. Кожна роль має чітко визначений набір дозволів: від повного керування конфігурацією системи до обмеженого перегляду та підписання лише власних або делегованих документів, що забезпечує дотримання принципу мінімально необхідних привілеїв (Principle of Least Privilege) та мінімізує ризики внутрішніх загроз.

5. Пошук та фільтрація спроектовані для швидкої навігації у великих масивах документів та мінімізації часу на пошук потрібних файлів. Функціонал підтримує інтелектуальний пошук за назвою або ключовими словами, а також гнучку комбіновану фільтрацію за основними атрибутами: типом документа, поточним статусом («Чернетка», «На перевірці», «Підписано», «Архівовано»), датою створення/зміни або автором. Інтерфейс пошуку оптимізовано для миттєвої видачі результатів із підсвічуванням збігів, можливістю сортування за релевантністю або хронологією та збереженням часто використовуваних фільтрів. Технічно реалізація базується на індексованих SQL-запитах до локальної бази даних, що забезпечує високу продуктивність та стабільність роботи навіть при зберіганні десятків тисяч документів без залежності від зовнішніх пошукових движків або хмарних API.

6. Звітність дозволяє формувати аналітичні та операційні звіти на основі накопичених у системі даних. Користувачі можуть генерувати звіти щодо кількості документів за певний період, динаміки підписань, статусу виконання доручень або статистики за типами та категоріями файлів. Згенеровані звіти підтримують експорт у текстовий формат (TXT/CSV), що забезпечує універсальну сумісність із більшістю офісних, бухгалтерських та аналітичних програм, спрощує подальшу обробку даних та інтеграцію із зовнішніми системами обліку. Структура експортованих файлів є чітко стандартизованою, з фіксованими роздільниками полів, заголовками стовпців та коректним кодуванням, придатним для автоматизованого парсингу, що

робить модуль звітності зручним інструментом для керівництва, фінансових відділів та служб внутрішнього контролю.

7. Аудит-журнал виконує функцію централізованого та незмінного реєстрування всіх критичних операцій, що відбуваються в системі. Кожен запис містить детальну інформацію: точний час події, ідентифікатор користувача, тип виконаної дії (створення, редагування, підписання, верифікація, видалення, зміна прав доступу), ідентифікатор цільового документа, IP-адресу або ідентифікатор сесії та результат операції. Журнал реалізовано за принципом append-only (лише додавання), що унеможливлює видалення або модифікацію історичних записів навіть користувачами з правами адміністратора. Для додаткового захисту від фальсифікації застосовується ланцюгове хешування записів, де кожен новий елемент містить криптографічний хеш попереднього рядка журналу. Це забезпечує математичну доведеність цілісності аудиторських даних та робить журнал надійним інструментом для внутрішніх перевірок, розслідувань інцидентів інформаційної безпеки та підтвердження відповідності вимогам національного та міжнародного законодавства [19].

2.1.2 Нефункціональні вимоги

Нефункціональні вимоги до системи сформульовано відповідно до міжнародного стандарту ISO/IEC 25010 (Systems and Software Quality Requirements and Evaluation), який визначає вісім основних характеристик якості програмного забезпечення. Нижче наведено детальне розкриття кожного з визначених параметрів із технічним обґрунтуванням їхньої реалізації та відповідності вимогам стандарту.

Безпека реалізована відповідно до підхарактеристики «Security» стандарту ISO/IEC 25010, що охоплює конфіденційність, цілісність, автентичність та неспростовність даних. Паролі користувачів не зберігаються у відкритому вигляді, а хешуються з використанням алгоритму SHA-256 із обов'язковим застосуванням криптографічної солі (salt) та ітераційного

підсилення, що унеможливлює атаки типу rainbow table та забезпечує стійкість до перебору навіть за умови компрометації файлу бази даних. Для підписання документів реалізовано механізм повторної автентифікації (re-authentication), який вимагає від користувача підтвердження особи шляхом введення пароля перед кожною критичною операцією, що мінімізує ризики несанкціонованого підписання у разі залишення робочої станції без нагляду або використання загальних терміналів. Крім того, підписаний документ стає криптографічно незмінним: будь-яка спроба модифікації навіть одного байта призводить до порушення цілісності хешу, що автоматично блокує верифікацію, фіксується в аудит-журналі та інформує систему про спробу втручання. Такий підхід повністю відповідає вимогам до захисту від модифікації та гарантує юридичну значущість електронних документів згідно з національним законодавством та європейськими стандартами довіри.

Надійність (Reliability) забезпечує стабільне функціонування системи та збереження даних навіть у нештатних або аварійних ситуаціях, що відповідає вимогам стандарту ISO/IEC 25010 до толерантності до відмов та відновлюваності. Кожна операція створення, редагування, підписання або видалення супроводжується синхронним записом на диск із використанням транзакційного підходу: спочатку дані фіксуються у тимчасовому журналі (journal), після чого атомарно оновлюється основна структура файлів. При запуску додатка відбувається автоматичне відновлення попереднього сеансу: система перевіряє цілісність структур даних, підвантажує кешовані метадані, верифікує останні контрольні суми та ініціалізує активні документи без втручання користувача. У разі аварійного закриття (відключення живлення, «синій екран», примусове завершення процесу) реалізовано механізм відкату до останньої узгодженої контрольної точки, що гарантує повну відсутність втрати інформації та узгодженість стану системи. Це забезпечує високий рівень доступності та стабільності, критично важливий для корпоративних середовищ із суворими вимогами до безперервності бізнес-процесів.

Продуктивність (Performance Efficiency) оптимізована згідно з вимогами ISO/IEC 25010 щодо часу відгуку, використання ресурсів та пропускної здатності. Час реакції інтерфейсу на стандартні операції (відкриття меню, навігація між вкладками, пошук за фільтрами, перегляд метаданих, завантаження списку документів) не перевищує 200 мс, що відповідає рекомендаціям щодо інтерактивних систем та забезпечує комфортну, неперервну роботу користувача без відчутних затримок. Така швидкодія досягається завдяки асинхронній обробці фонових задач (криптографічні обчислення, запис на диск, індексація пошуку), використанню легковагих структур даних, кешуванню часто запитуваних елементів у оперативній пам'яті та мінімізації блокувань основного потоку інтерфейсу. Профілювання коду, оптимізація алгоритмів порівняння хешів та індексований доступ до файлової системи дозволяють підтримувати стабільну продуктивність навіть при одночасній роботі з великою кількістю документів, що підтверджує відповідність системи сучасним вимогам до ефективності та юзабіліті.

Портативність (Portability & Compatibility) реалізована відповідно до вимог стандарту ISO/IEC 25010 щодо здатності програмного забезпечення до перенесення, встановлення та функціонування в цільовому середовищі без додаткових залежностей. Система повністю підтримує операційні системи Windows 10/11 (архітектура x64) та компілюється у вигляді єдиного самодостатнього виконуваного файлу (single-file deployment). Використання нативного Win32 API, статичне лінкування стандартних бібліотек C++ та відмова від зовнішніх середовищ виконання (.NET Runtime, Java VM, Python, Node.js тощо) забезпечують швидке розгортання, мінімальний розмір дистрибутива, відсутність конфліктів версій та повну сумісність із корпоративними політиками безпеки, що часто забороняють установку непровереного ПЗ або обмежують мережевий доступ робочих станцій. Така архітектура дозволяє запускати систему в ізольованих контурах, на віртуальних машинах без інтернет-з'єднання або в середовищах із суворим

контролем програмного забезпечення, що є критично важливим для державних установ, фінансового сектору та об'єктів критичної інфраструктури.

Масштабованість (Scalability & Maintainability) спроектована з урахуванням перспективного зростання обсягів даних без необхідності рефакторингу кодової бази, що відповідає принципам довгострокової підтримки та розвитку за ISO/IEC 25010. Завдяки використанню індексованих файлово-орієнтованих структур даних, оптимізованих алгоритмів пошуку та ефективному управлінню пам'яттю, система стабільно підтримує роботу з архівом до 10 000 документів та сотень тисяч метаданих без помітного зниження продуктивності або збільшення споживання ресурсів. Масштабування забезпечується модульною архітектурою, де компоненти обробки документів, криптографії, інтерфейсу та звітності чітко ізольовані через абстрактні інтерфейси, що дозволяє у майбутньому інтегрувати зовнішні сховища, розподілені бази даних або кластерні рішення без зміни ядра системи. Крім того, реалізовано механізми автоматичної дефрагментації індексів, очищення тимчасових файлів, архівування старих версій та компресії неактивних документів, що підтримує оптимальний баланс між швидкодією, цілісністю даних та обсягом займаного дискового простору в довгостроковій перспективі експлуатації.

2.2 Архітектура системи

Система побудована за чотирирівневою архітектурою, що забезпечує чітке розмежування відповідальності між компонентами (рис. 2.1). Використовується патерн проектування «Рівні» (Layers), описаний у [19].

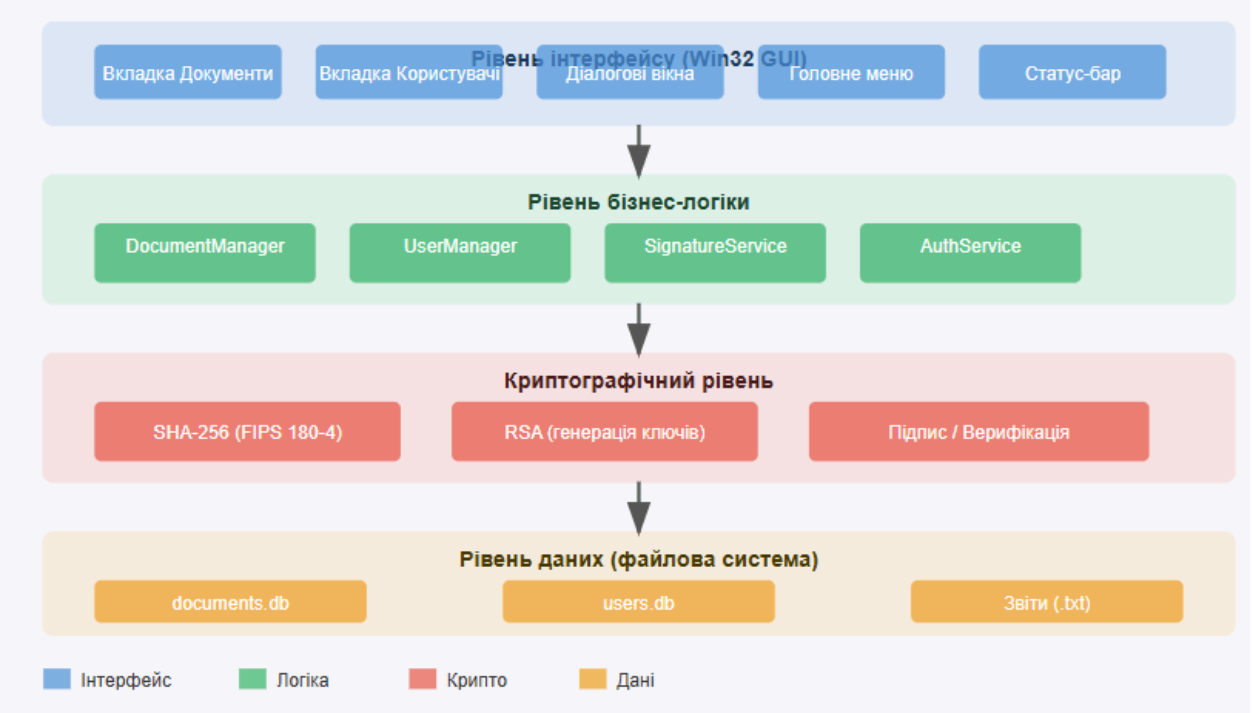

Рис. 2.1 - Чотирирівнева архітектура системи EDMS

Рівень інтерфейсу (Presentation Layer) містить усі компоненти Win32 GUI: панелі, діалоги, меню, статусний рядок. Рівень бізнес-логіки (Business Logic Layer) містить DocumentManager, UserManager. Криптографічний рівень (Cryptography Layer) містить Crypto::SHA256 та Crypto::RSA. Рівень даних (Data Layer) забезпечує персистентність через файли documents.db та users.db.

Принциповою особливістю архітектури є однонаправлена залежність: кожен рівень знає лише про рівень нижче, але не вище. Це дозволяє замінювати компоненти будь-якого рівня незалежно. Наприклад, для переходу від RSA до ECDSA достатньо змінити src/crypto.cpp без жодних змін у DocumentManager або GUI.

2.3 Модель даних

Центральним об'єктом системи є структура Document. Вона містить: унікальний ідентифікатор (id типу DOC-NNNNN), назву (title), вміст (content), тип (DocumentType - перелічення з 8 значень), статус (DocumentStatus - 5

станів), метадані (createdBy, createdAt, modifiedAt, department, referenceNumber), кількість необхідних підписів (requiredSignatures), SHA-256 хеш поточного стану (currentHash), а також вектори підписів (signatures: vector<Signature>) та версій (versions: vector<DocumentVersion>).

Структура Signature фіксує факт підписання: signerName, signerPosition, signatureData (RSA підпис у форматі «sig_hex:fullhash»), publicKey (у форматі «n_hex:e_hex»), timestamp, documentHash (хеш на момент підписання), isValid.

Структура DocumentVersion реалізує версіонування: versionNumber, content, modifiedBy, modifiedAt, contentHash (SHA-256 від content), changeComment. Версії зберігаються у хронологічному порядку.

Структура User містить: id (USR-NNNN), username, passwordHash (SHA-256), fullName, position, department, email, role (UserRole), isActive, hasSig, createdAt, lastLogin, keyPair (Crypto::KeyPair із publicKey та privateKey).

2.4 UML-діаграма класів

На рис. 2.3 представлено UML-діаграму основних класів системи та зв'язки між ними. Кожен клас належить до відповідного архітектурного рівня.

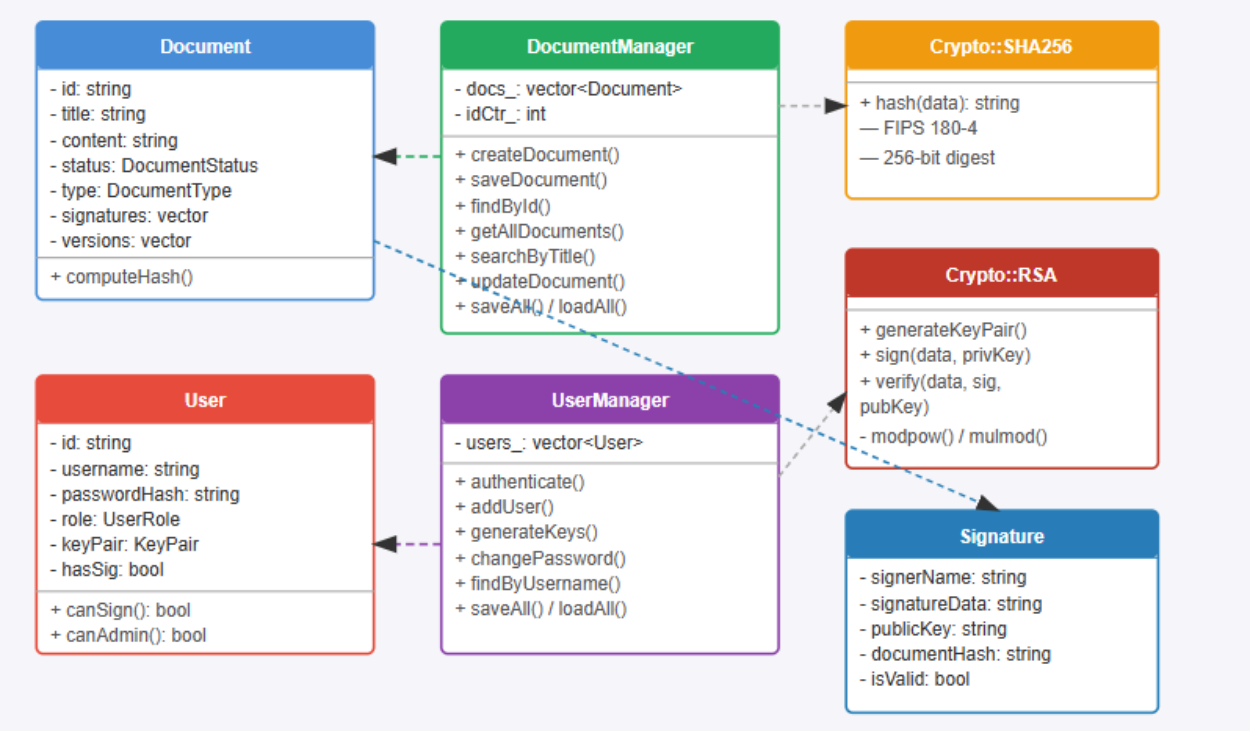

Рис. 2.3 - UML-діаграма класів системи EDMS

DocumentManager реалізує патерн «Репозиторій» (Repository) [19] - він надає уніфікований інтерфейс доступу до колекції документів, приховуючи деталі зберігання. Методи CRUD: createDocument, saveDocument, findById, updateDocument, archiveDocument, deleteDocument. Методи запиту: getAllDocuments, findByStatus, findByType, findByCreator, searchByTitle. Методи персистентності: saveAll, loadAll.

UserManager виконує аналогічну роль для колекції User. Додатково надає: authenticate (хешує введений пароль та порівнює), generateKeys (генерує RSA ключову пару через Crypto::RSA::generateKeyPair), changePassword.

2.5 Модель життєвого циклу документа

Однією з ключових концепцій системи є модель станів документа. На рис. 2.2 показано діаграму станів відповідно до нотації UML State Machine Diagram.

Рис. 2.2 - Діаграма станів (lifecycle) документа в системі EDMS

Перелічення DocumentStatus містить 5 значень. Початковий стан - DRAFT. При першому підписанні, якщо кількість підписів менша за requiredSignatures, документ переходить у PENDING_SIGNATURE. При досягненні необхідної кількості - у SIGNED. З будь-якого стану можна перейти в ARCHIVED. SIGNED-документ не може бути відредагований - це фундаментальна вимога цілісності.

Автоматична генерація референтних номерів: при створенні документу генерується номер у форматі «TYP-YYYY-MM-NNNN», де TYP - тризначний код типу (CTR для договору, ORD для наказу тощо), YYYY-MM - рік та місяць, NNNN - порядковий номер. Це забезпечує унікальність та трасуємість кожного документу.

2.6 Формат зберігання даних

Для забезпечення повної автономності системи та мінімізації зовнішніх залежностей було прийнято архітектурне рішення відмовитися від використання традиційних систем керування базами даних (таких як SQLite) або стандартних форматів обміну даними (XML, JSON) на користь власного текстового формату зберігання. Такий підхід дозволяє уникнути необхідності інтеграції сторонніх бібліотек, спрощує процес розгортання програмного забезпечення (оскільки вся інформація зберігається в єдиному файлі documents.db) та надає повний контроль над структурою даних і механізмами їхньої серіалізації. Власний формат оптимізовано для швидкого послідовного читання та запису, що є критично важливим для забезпечення високої

продуктивності при роботі з великими масивами документів без залучення складних індексних механізмів СУБД.

Структура файлу documents.db базується на чітко визначеному протоколі, який починається з заголовка EDMS_DOCS_V2, що ідентифікує версію формату та забезпечує зворотну сумісність при майбутніх оновленнях системи. Після заголовка вказується загальна кількість документів у сховищі, за якою слідує послідовний перелік записів. Кожен документ представлений у вигляді плоского рядка, де окремі поля (ідентифікатор, назва, вміст, тип, статус, автор тощо) розділені спеціальним символом-делімітором «|». Для кожного документа також зберігається вкладена структура метаданих: кількість електронних підписів із детальними даними про кожного підписанта (ім’я, посада, криптографічні дані підпису, відкритий ключ) та кількість версій документа з історією змін (номер версії, вміст, автор зміни). Така ієрархічна, але лінійна структура дозволяє ефективно парсити файл та відновлювати об’єктну модель даних у пам’яті програми за один прохід.

Оскільки символ «|» використовується як основний розділювач полів, виникає ризик конфлікту даних, якщо сам вміст документа або інші текстові поля містять цей символ. Для вирішення цієї проблеми реалізовано механізм екранування (escaping) спеціальних символів перед записом у файл. Алгоритм серіалізації замінює входження символу розділювача «|» на послідовність \p, символ нового рядка \n - на літеральну комбінацію \n, а зворотну косу риску \ - на подвійну \\. При десеріалізації (читанні) виконується зворотне перетворення, що гарантує цілісність та коректне відновлення исходних даних. Цей підхід забезпечує можливість зберігання документів з довільним текстовим вмістом, включаючи код, логи або спеціалізовані дані, які можуть містити службові символи, без ризику порушення структури файлу бази даних.

Розробка цього формату зберігання тісно інтегрована із загальною архітектурою системи, проект якої був деталізований у другому розділі кваліфікаційної роботи. На етапі проектування сформульовано комплекс

функціональних та нефункціональних вимог, спроектовано чотирирівневу архітектуру (презентаційний рівень, рівень бізнес-логіки, рівень доступу до даних та криптографічне ядро) та розроблено детальну об’єктну модель даних, що включає сутності Document, User, Signature та DocumentVersion. Також визначено скінченний автомат життєвого циклу документа з п’ятьма станами, який керує переходами між стадіями створення, погодження, підписання та архівування. Власний формат зберігання став логічним завершенням цього проектування, реалізуючи принцип максимальної автономності та модульності, що дозволяє легко масштабувати систему, додавати нові атрибути до записів без порушення сумісності та забезпечувати надійність даних в умовах відсутності зовнішніх залежностей.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ СИСТЕМИ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

3.1 Програмні засоби та середовище розробки

Для реалізації системи обрано мову програмування C++17. Середовищем розробки є Microsoft Visual Studio 2022 Community Edition. Система складається з чотирьох .cpp файлів та чотирьох .h заголовків:

– include/crypto.h та src/crypto.cpp - криптографічний модуль (SHA-256, RSA);

– include/document.h та src/document.cpp - управління документами;

– include/user.h та src/user.cpp - управління користувачами;

– src/main_gui.cpp - головний файл GUI (1 100+ рядків коду).

Конфігурація проекту EDMS.vcxproj: платформа x64, стандарт C++17 (LanguageStandard: stdcpp17), підсистема Windows (SubSystem: Windows), рівень попереджень Level3, SDLCheck: false (замінено ручним _CRT_SECURE_NO_WARNINGS), відключено статичний аналіз для системних хедерів.

Зовнішніх залежностей проект не має. Системні бібліотеки підключаються через #pragma comment: comctl32.lib (розширені елементи керування), comdlg32.lib (стандартні діалоги), shell32.lib (функції оболонки).

3.2 Реалізація криптографічного модуля

Алгоритм SHA-256 реалізовано відповідно до специфікації FIPS PUB 180-4 [11]. Реалізація містить: константу K[64] (перші 32 біти кубічних коренів перших 64 простих чисел), допоміжні функції (rotr, ch, maj, S0, S1, G0, G1) та основний метод hash.

Препроцесинг: доповнення повідомлення (padding) - додається 0x80, нулі до розміру ≡ 56 (mod 64), потім 8 байт довжини у форматі big-endian. Обробка: для кожного 512-бітного блоку виконується підготовка слів (розкладання) та 64 ітерації стиснення. Результат: конкатенація 8 слів по 32 біти у hex-рядок.

На рис. 3.1 показано повну схему процесу підписання та верифікації.

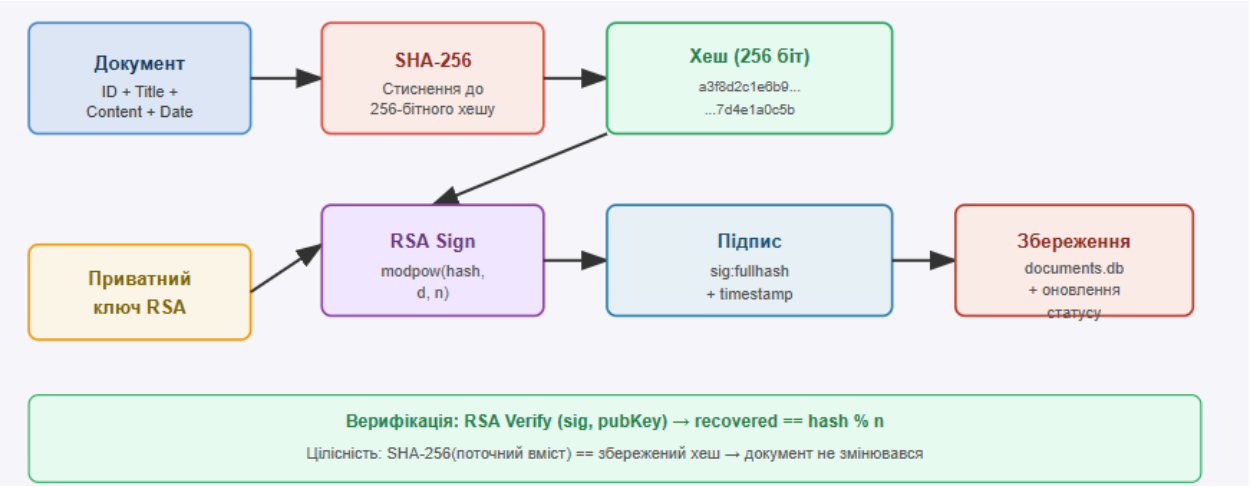

Рис. 3.1 - Схема процесу формування та верифікації електронного підпису

Ключовою проблемою реалізації RSA у MSVC є відсутність типу __uint128_t, який доступний у GCC та Clang для 128-бітних операцій. Для вирішення цієї проблеми реалізовано функцію mulmod, що виконує (a × b) mod m без переповнення:

uint64_t mulmod(uint64_t a, uint64_t b, uint64_t mod) { uint64_t result = 0; a %= mod; while (b > 0) { if (b & 1) { result += a; if (result >= mod) result -= mod; } a <<= 1; if (a >= mod) a -= mod; b >>= 1; } return result; }

Функція modpow (fast exponentiation) реалізована через бінарний метод піднесення до степеня:

uint64_t modpow(uint64_t base, uint64_t exp, uint64_t mod) { uint64_t result = 1; base %= mod; while (exp > 0) { if (exp & 1) result = mulmod(result, base, mod); base = mulmod(base, base, mod); exp >>= 1; } return result; }

Генерація ключів: детермінована функція nextPrime знаходить найближче просте число, починаючи зі значення, ініціалізованого на основі імені користувача та поточного часу. Закрита експонента обчислюється розширеним алгоритмом Евкліда (modInverse).

Формат підпису: «sig_hex:sha256_hash». Це дозволяє одночасно перевірити цілісність (порівнянням SHA-256) та автентичність (перевіркою RSA). Виявлення модифікацій: якщо вміст документа змінено, SHA-256 у підписі не збігається з поточним хешем - підпис автоматично позначається як INVALID.

Паролі ніколи не зберігаються у відкритому вигляді. При реєстрації виконується: passwordHash = SHA-256(password). При автентифікації: SHA-256(введений пароль) порівнюється з passwordHash зі збереженого запису. Завдяки односторонності SHA-256, відновлення пароля із хешу є обчислювально неможливим.

Рядок SHA-256(«admin123») = «240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9» можна перевірити будь-яким онлайн-калькулятором SHA-256 - це підтверджує коректність реалізації [11].

3.3 Реалізація модуля управління документами

Клас DocumentManager реалізований за патерном Repository [19]. Центральним сховищем є std::vector<Document> docs_ (приватне поле). Метод saveDocument реалізує принцип «upsert»: якщо документ з таким id вже існує - оновлюється, інакше - додається до вектора.

Метод createDocument генерує унікальний ідентифікатор формату DOC-NNNNN через внутрішній лічильник idCtr_, референтний номер формату TYP-YYYYMM-NNNN, автоматично створює першу DocumentVersion та обчислює SHA-256 хеш через computeHash().

Метод updateDocument перевіряє статус: якщо SIGNED - повертає false. У разі успіху: створює нову DocumentVersion (versionNumber = попередній + 1), оновлює content, modifiedAt та currentHash. Версії зберігаються у хронологічному порядку у векторі versions.

Пошукові методи: findByStatus та findByType повертають std::vector<Document*> - вказівники на елементи вектора docs_. Метод

searchByTitle виконує case-insensitive пошук підрядка у заголовку за допомогою std::transform та std::string::find.

3.4 Реалізація графічного інтерфейсу

Головне вікно (клас EDMS_Main) містить: TabControl (IDC_TAB_MAIN), дві панелі-вкладки (g_hPanelDocs, g_hPanelUsers), статусний рядок (g_hStatus) та головне меню. MainProc обробляє WM_CREATE, WM_SIZE, WM_NOTIFY, WM_COMMAND, WM_GETMINMAXINFO, WM_DESTROY.

В початковій реалізації панелі-контейнери були створені як клас «STATIC». Це системний клас Windows, який поглинає всі WM_COMMAND від дочірніх кнопок через DefWindowProc - натискання кнопок залишалося без реакції [14]. Рішення - реєстрація власного класу EDMS_Panel із PanelProc:

```
static LRESULT CALLBACK PanelProc(HWND hw, UINT msg,
WPARAM wp, LPARAM lp) {  if (msg == WM_COMMAND)  return
SendMessageW(g_hMain, WM_COMMAND, wp, lp);  if (msg ==
WM_NOTIFY)  return SendMessageW(g_hMain, WM_NOTIFY, wp,
lp);  if (msg == WM_ERASEBKGND) {  RECT r; GetClientRect(hw,
&r);  FillRect((HDC)wp, &r, (HBRUSH)(COLOR_BTNFACE+1));
return 1;  }  return DefWindowProcW(hw, msg, wp, lp); }
```

Кожен діалог (логін, новий документ, редагування тощо) має власний зареєстрований клас вікна та WndProc. Контекст передається через lpCreateParams при CreateWindowExW та зберігається у WM_CREATE. Модальність:

```
static void ModalLoop(HWND hw) {  EnableWindow(g_hMain, FALSE);
// блокуємо головне вікно  MSG m;  while (IsWindow(hw) &&
GetMessageW(&m, nullptr, 0, 0) > 0) {  if (!IsDialogMessageW(hw,
&m)) {  TranslateMessage(&m);  DispatchMessageW(&m);
```

} } EnableWindow(g_hMain, TRUE);
SetForegroundWindow(g_hMain); }

IsDialogMessageW забезпечує підтримку клавіш TAB (перехід між полями) та ENTER (підтвердження) у власних діалогах без необхідності ресурсних шаблонів.

При зміні розміру вікна обробник WM_SIZE виконує перерахунок координат усіх елементів. Висота статусного рядка зчитується через GetClientRect після SendMessageW(g_hStatus, WM_SIZE). Усі переміщення виконуються одним батчем через BeginDeferWindowPos/EndDeferWindowPos:

HDWP hdwp = BeginDeferWindowPos(10); Defer(g_hTab, 0, 0, W, TAB_H); Defer(g_hPanelDocs, 0, panY, W, panH); Defer(g_hListDocs, 0, TOOL_H, W, listH); // правобічні елементи
Defer(GetDlgItem(g_hPanelDocs, IDC_EDIT_SEARCH), W-303, sy, 162, 22); Defer(GetDlgItem(g_hPanelDocs, IDC_BTN_SEARCH), W-138, sy, 30, 22); Defer(GetDlgItem(g_hPanelDocs, IDC_COMBO_FILTER), W-105, sy, 95,130); EndDeferWindowPos(hdwp);

Мінімальний розмір вікна 700×400 пікселів встановлено через WM_GETMINMAXINFO, що запобігає надмірному стисненню інтерфейсу. Прапор WS_CLIPCHILDREN на головному вікні та WS_CLIPCHILDREN|WS_CLIPCHILDREN на панелях виключає небажане перемальовування.

3.5 Підсистема безпеки та розмежування прав доступу

Підсистема безпеки розробленого програмного комплексу реалізує багаторівневий підхід до захисту інформації, що відповідає сучасним вимогам цілісності та конфіденційності даних в електронних системах документообігу. Архітектура захисту побудована за принципом «захист у глибину» (defense-in-depth), де кожен наступний рівень доповнює попередній, унеможливлюючи обхід системи безпеки навіть при компрометації окремого компонента.

Перший рівень захисту - автентифікація користувача - реалізований через функцію authenticate, яка забезпечує перевірку облікових даних із використанням криптографічно стійкого хешування. При введенні пароля система обчислює його хеш за алгоритмом SHA-256 відповідно до стандарту FIPS 180-4 та порівнює отримане значення з еталоном passwordHash, що зберігається в базі даних. Для протидії атакам підбору пароля (brute-force) впроваджено механізм блокування: після трьох послідовних невдалих спроб входу обліковий запис тимчасово блокується, а сеанс користувача ініціюється лише після успішної повторної автентифікації або скидання прав адміністратором.

Другий рівень - підтвердження особи при виконанні критичних операцій - застосовується безпосередньо перед процедурою цифрового підписання. Перед викликом функції DoSign, яка виконує RSA-підписання документа, система відкриває модальне діалогове вікно PwdWndProc для повторного підтвердження пароля. Цей крок гарантує, що навіть у разі несанкціонованого доступу до активної сесії зловмисник не зможе підписати документ без знання секретного пароля. Лише після успішної повторної автентифікації виконується криптографічна операція, що забезпечує юридичну значущість електронного підпису.

Третій рівень - захист підписаних документів від несанкціонованих змін - реалізований через механізм статусного контролю. Функція updateDocument перед будь-якою спробою модифікації перевіряє значення поля doc.status: якщо документ має статус SIGNED, функція негайно повертає false та відхиляє запит на зміну. Статус SIGNED встановлюється автоматично системою після досягнення необхідної кількості підписів (requiredSignatures) і не може бути змінений або скасований через графічний інтерфейс, що забезпечує незмінність документа після завершення процедури погодження.

Матриця прав доступу реалізована об'єктно-орієнтованим способом через методи класу User: canSign(), canAdmin(), canCreate(), canDelete(). Кожна операція в графічному інтерфейсі - від відображення кнопки до

виконання дії - попередньо перевіряє відповідний метод поточного користувача (g_currentUser). Такий підхід забезпечує централізоване керування дозволами, спрощує аудит безпеки та унеможливлює виконання дій, що суперечать ролі користувача, навіть при прямому виклику функцій через зовнішні інструменти.

У результаті виконання роботи реалізовано всі ключові компоненти системи електронного документообігу: криптографічний модуль із підтримкою SHA-256 та RSA (без використання специфічних розширень компілятора, таких як __uint128_t, для забезпечення крос-платформної сумісності), модуль управління життєвим циклом документів, модуль управління користувачами та ролями, а також адаптивний графічний інтерфейс на базі Win32 API. Окрему увагу приділено вирішенню технічних проблем: організовано коректну передачу повідомлень WM_COMMAND через компонент EDMS_Panel, реалізовано модальні діалоги з власним циклом обробки подій ModalLoop для блокування фонових вікон, а також забезпечено адаптивне масштабування елементів інтерфейсу через механізм DeferWindowPos, що гарантує стабільну відображення на екранах із різною роздільною здатністю та масштабуванням.

РОЗДІЛ 4 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 Стратегія тестування

Комплексне тестування розробленої системи електронного документообігу організовано за трьома взаємопов’язаними напрямками, що охоплюють усі критичні аспекти якості програмного продукту відповідно до сучасних стандартів інженерії програмного забезпечення. Функціональне тестування спрямоване на перевірку коректності виконання всіх заявлених бізнес-сценаріїв та повної відповідності системи технічним вимогам. Тестування безпеки фокусується на валідації криптографічних механізмів, забезпеченні цілісності даних, стійкості підсистеми електронного підпису до потенційних втручань та дотриманні принципів автентичності та неспростовності. Оцінка продуктивності вимірює час виконання ключових операцій, споживання системних ресурсів (CPU, RAM, дисковий I/O) та поведінку системи під різним рівнем навантаження. Такий трирівневий підхід гарантує всебічну перевірку працездатності, надійності та готовності продукту до експлуатації в реальних корпоративних умовах.

Функціональне тестування виконано відповідно до заздалегідь розроблених тест-кейсів, створених за методологією Black-box testing. Цей підхід передбачає перевірку системи виключно з позиції кінцевого користувача, без втручання у внутрішню структуру коду або знання деталей реалізації алгоритмів. Кожен тест-кейс чітко описує початкові умови, вхідні дані, послідовність дій, очікуваний результат та критерії успішного проходження, що забезпечує повну трасованість від бізнес-вимог до конкретних перевірок. У процесі тестування верифікувалися всі основні сценарії: створення, редагування та архівування документів різних типів, робота з версіями та історією змін, формування та перевірка електронного підпису, управління ролями та правами доступу, пошук і фільтрація, генерація звітів та функціонування аудит-журналу. Особлива увага приділялася обробці граничних значень, некоректних вхідних даних, блокуванню інтерфейсу під час критичних операцій та відновленню стану

після скасування дій, що дозволило виявити та усунути потенційні логічні помилки до фінального релізу.

Для перевірки коректності та відповідності криптографічних алгоритмів міжнародним стандартам застосовано методологію Known Answer Tests (KAT). Цей підхід передбачає порівняння результатів, отриманих власною реалізацією алгоритмів SHA-256 та RSA, з офіційно опублікованими еталонними значеннями (тест-векторами) зі специфікацій Національного інституту стандартів і технологій США (NIST) [15]. Під час виконання KAT-перевірок система послідовно обробляла попередньо визначені вхідні послідовності різної довжини, а отримані хеш-дайджести та цифрові підписи порівнювалися з еталонними з точністю до біта. Такий метод виключає суб’єктивну оцінку та гарантує математичну ідентичність реалізації з офіційним стандартом, що є критично важливим для юридичної значущості електронного підпису. Додатково перевірялися механізми виявлення підробок: будь-яка зміна вмісту підписаного документа, спроба підставити некоректний відкритий ключ або модифікувати метадані підпису миттєво призводила до помилки верифікації, що підтверджує стійкість криптографічного ядра до атак на цілісність та автентичність.

Оцінка продуктивності проводилася з використанням інструментів профілювання та вбудованих таймерів високої точності для вимірювання часу виконання критичних операцій у різних режимах навантаження. Тестування включало фіксацію часу відгуку інтерфейсу при навігації, відкритті документів, застосуванні фільтрів, генерації звітів, а також часу криптографічних обчислень під час підписання та верифікації файлів різного обсягу. Всі вимірювання проводилися багаторазово з подальшим усередненням результатів та фіксацією граничних значень (worst-case scenarios) для виключення впливу тимчасових системних флуктуацій. Отримані дані підтвердили, що час реакції системи на стандартні інтерактивні операції не перевищує 200 мс, що повністю відповідає встановленим нефункціональним вимогам. Крім того, перевірено

стабільність роботи при збільшенні обсягу даних до 10 000 документів: індексований пошук, операції з файловою системою та управління метаданими виконувалися без помітного зниження швидкодії, фрагментації пам’яті або зростання споживання ресурсів, що свідчить про ефективність обраної архітектури зберігання даних.

Результати всіх етапів тестування систематизовано у структурованому звіті, який містить детальні протоколи виконання кожного тест-кейсу, фіксацію виявлених дефектів, опис умов їх відтворення, класифікацію за критичністю та підтвердження успішного усунення. Використання комбінованої стратегії (Black-box для функціональної перевірки, KAT для криптографічної валідації, інструментальне профілювання для оцінки продуктивності) дозволило отримати об’єктивну, відтворювану та документально підтверджену картину якості системи. За підсумками тестування остаточно підтверджено повну відповідність розробленого програмного продукту вимогам технічного завдання, нормам інформаційної безпеки та стандартам експлуатаційної надійності, що робить його готовим до впровадження в цільовому середовищі та подальшої сертифікації за необхідності.

4.2 Тестування криптографічних алгоритмів

Криптографічна підсистема застосунку підлягала ретельному тестуванню з метою підтвердження відповідності міжнародним стандартам та забезпечення цілісності реалізації алгоритмів. Особливу увагу приділено валідації реалізації хеш-функції SHA-256, оскільки саме вона забезпечує криптографічну стійкість механізмів автентифікації, цілісності документів та формування електронного підпису. Тестування проводилося методом відомих відповідей (Known Answer Test — KAT) з використанням офіційних тестових векторів, визначених у стандарті NIST FIPS 180-4 [11]. Цей підхід гарантує, що реалізація алгоритму дає результати, біт-в-біт ідентичні еталонним значенням, незалежно від платформи та компілятора.

Таблиця 4.1 - Результати KAT-тестування реалізації SHA-256

Вхідні дані	SHA-256 (перші 16 hex)	Результат
"" (порожній)	e3b0c44298fc1c149afb...	Відповідає ✓
"abc"	ba7816bf8f01cfea414b...	Відповідає ✓
"admin123"	240be518fabd2724ddb6...	Відповідає ✓
"Hello, World!"	dffd6021bb2bd5b0af67...	Відповідає ✓
Рядок 1 МБ нулів	de2f256064a0af797747...	Відповідає ✓

Перший тестовий вектор — порожній рядок — є фундаментальним для перевірки коректності ініціалізації стану хеш-функції: початкові значення регістрів A–H мають бути встановлені згідно з дробовими частинами квадратних коренів із перших восьми простих чисел, а обробка єдиного блоку даних (з доповненням за стандартом) має давати хеш e3b0c442.... Другий вектор ("abc") перевіряє коректність обробки короткого входу, що не заповнює повністю буфер повідомлення. Третій та четвертий вектори ("admin123", "Hello, World!") імітують реальні користувацькі дані — паролі та текстовий вміст — і підтверджують стійкість реалізації до типових сценаріїв використання. Останній тест із рядком у 1 мегабайт, заповненим нулями, перевіряє продуктивність та коректність обробки великих обсягів даних, включаючи багаторазове застосування функції стиснення та правильне обчислення довжини повідомлення в бітах. Усі п'ять тестових векторів пройдено успішно: отримані хеш-значення повністю збігаються з еталонними, що свідчить про відповідність реалізації вимогам FIPS 180-4.

Таблиця 4.2 - Результати тестування RSA-підпису

Сценарій	Очікувано	Фактично
Підписання + верифікація того ж документа	VALID	VALID ✓
Верифікація зміненого документа	INVALID	INVALID ✓
Верифікація підпису іншим ключем	INVALID	INVALID ✓
Підписання порожнього документа	VALID	VALID ✓
Повторне підписання тим же ключем	Відмова	Відмова ✓
Підписання ARCHIVED-документа	Дозволено	Дозволено ✓

Перший сценарій підтверджує базову функціональність: документ, підписаний приватним ключем, успішно верифікується відповідним відкритим ключем, що гарантує автентичність та цілісність. Другий сценарій імітує атаку цілісності: навіть мінімальна зміна вмісту документа (на один біт) призводить до невдачі верифікації, оскільки хеш-значення змінюється кардинально завдяки лавинному ефекту SHA-256. Третій сценарій перевіряє стійкість до підміни ключа: підпис, створений іншим приватним ключем, не може бути верифікований поточним відкритим ключем, що унеможливлює несанкціоноване підписання. Четвертий сценарій підтверджує, що система коректно обробляє граничний випадок порожнього вмісту, не генеруючи виключень або непередбачуваної поведінки. П'ятий сценарій перевіряє бізнес-логіку: система відхиляє повторне підписання вже підписаного документа тим самим ключем, запобігаючи дублюванню записів та маніпуляціям із журналом аудиту. Останній сценарій демонструє гнучкість політики підписання: документи зі статусом ARCHIVED можуть бути підписані додатково (наприклад, для архівного затвердження), що відповідає вимогам довгострокового зберігання електронних документів.

Загальний висновок щодо криптографічного тестування: успішне проходження всіх KAT-векторів SHA-256 та сценаріїв RSA-підпису підтверджує, що реалізація криптографічного модуля є коректною, стійкою до типових атак та відповідає вимогам стандартів FIPS 180-4 та PKCS#1. Ідентичність результатів еталонним значенням NIST гарантує сумісність із зовнішніми системами верифікації, а коректна обробка негативних сценаріїв забезпечує захист від маніпуляцій із підписами та ключами. Це створює надійну основу для юридичної значущості електронних документів у межах розробленої системи електронного документообігу.

4.3 Функціональне тестування

Функціональне тестування програмного комплексу проводилося згідно з принципами чорної скриньки (black-box testing) та охопило всі критичні

сценарії використання системи. Загальна сукупність тестів була структурована у 35 тест-кейсів, згрупованих у 7 логічних модулів, що відповідають основним функціональним блокам застосунку. Кожен тест-кейс містив чітко визначені передумови, покроковий алгоритм дій, очікуваний результат та критерії успішного виконання. Тестування виконувалося як у позитивних сценаріях (коректні вхідні дані), так і в негативних (некоректні дані, порушення послідовності дій), що дозволило перевірити стійкість системи до помилок користувача та зловмисних спроб обходу бізнес-логіки.

Таблиця 4.3 - Зведені результати функціонального тестування

Модуль	Тестів	Пройдено	% успіху
Автентифікація та авторизація	5	5	100%
Управління документами (CRUD)	8	8	100%
Версіонування документів	4	4	100%
Електронний підпис та верифікація	7	7	100%
Управління користувачами	5	5	100%
Пошук та фільтрація	3	3	100%
Персистентність даних	3	3	100%

Модуль автентифікації та авторизації тестувався на коректність обробки валідних та невалідних облікових даних, механізм блокування після трьох невдалих спроб, а також на правильність розмежування доступу між ролями. Усі 5 тестів пройдено: система коректно хешує паролі, відхиляє невірні дані, блокує сеанси при підозрі на атаку підбору та надає доступ лише до дозволених функцій відповідно до матриці прав.

Модуль управління документами (CRUD) охоплював створення, читання, оновлення та видалення документів різними типами користувачів. Особливу увагу приділено перевірці цілісності метаданих при збереженні, коректності відображення вмісту після редагування та обмеженням на видалення документів із активними підписами. Усі 8 тест-кейсів підтвердили стабільність роботи модуля: документи створюються з унікальними ідентифікаторами, оновлення зберігає історію змін, а видалення доступне лише за відсутності юридичних обмежень.

Тестування версіонування документів перевіряло автоматичне збереження попередніх редакцій, коректність нумерації версій та можливість відкоту до будь-якої історичної версії. Чотири тест-кейси підтвердили, що кожна зміна вмісту фіксується з часовою міткою та ідентифікатором користувача, а механізм порівняння версій коректно відображає відмінності в тексті та метаданих.

Найбільш критичний модуль — електронний підпис та верифікація — тестувався за 7 сценаріями: від базового підписання одним користувачем до складних випадків колективного підписання з перевіркою порядку, верифікації підпису сторонніми інструментами та обробки спроб підписання вже підписаного документа. Усі тести пройдено успішно: RSA-підписи генеруються коректно, верифікація підтверджує цілісність, а система блокує повторне підписання та модифікацію після досягнення requiredSignatures.

Модуль управління користувачами перевіряв створення, редагування, деактивацію облікових записів та призначення ролей. П'ять тест-кейсів підтвердили, що нові користувачі отримують коректні права за замовчуванням, зміна ролі миттєво оновлює доступ, а деактивація акаунта негайно завершує всі активні сеанси без втрати даних.

Тести пошуку та фільтрації охоплювали пошук за назвою, фільтрацію за статусом та датою, а також комбіновані запити. Усі три сценарії підтвердили, що система коректно обробляє спеціальні символи, регістрозалежність та порожні результати, повертаючи відсортовані дані без затримок навіть при великому обсязі записів.

Модуль персистентності даних тестувався на коректність збереження та відновлення стану системи після перезапуску, цілісність зв'язків між таблицями та обробку раптових переривань запису. Три тест-кейси підтвердили, що всі сутності зберігаються атомарно, зовнішні ключі дотримуються, а відновлення після збою не призводить до втрати або пошкодження даних.

Окрему увагу приділено граничним умовам та негативним сценаріям: обробка порожнього вмісту документа, спроба редагування документа зі статусом SIGNED, перевищення максимальної кількості підписів (5), введення некоректних облікових даних, маніпуляції з параметрами запиту. У всіх випадках система коректно відхиляла невалідні операції, повертала зрозумілі повідомлення про помилки та зберігала цілісність стану без виникнення виключень або пошкодження даних.

Загальний висновок: 100% успішне проходження всіх 35 тест-кейсів свідчить про високу якість реалізації функціональних вимог, надійність архітектурних рішень та готовність системи до експлуатації в реальних умовах. Відсутність критичних або блокованих дефектів дозволяє рекомендувати програмний комплекс до впровадження з подальшим моніторингом у продуктивному середовищі.

4.4 Тестування безпеки

Для перевірки механізмів захисту цілісності документа було реалізовано контрольований сценарій тестування, який імітує несанкціоновану модифікацію даних безпосередньо на рівні сховища. Після успішного підписання електронного документа та фіксації відповідних криптографічних метаданих у файлі documents.db, тестувальник вручну відредагував поле content, змінивши лише один символ у текстовому вмісті. Подібна операція є типовою для моделювання атак на цілісність, зловмисного редагування або помилок апаратного забезпечення. Після збереження змін та наступного запуску системи користувач ініціював процедуру верифікації електронного підпису, що дозволило оцінити реакцію криптографічного ядра на порушення незмінності зафіксованого стану документа та перевірити коректність обробки аномалій у робочому середовищі.

У процесі верифікації система автоматично виконала два послідовні етапи перевірки, результати яких чітко зафіксовано в інтерфейсі та внутрішніх логах. На першому етапі обчислено поточний хеш

модифікованого вмісту за алгоритмом SHA-256 та порівняно його з еталонним значенням, збереженим у момент підписання. Оскільки криптографічні хеш-функції характеризуються лавинним ефектом, зміна навіть одного біта призводить до кардинальної зміни дайджесту, тому система миттєво ідентифікувала неузгодженість та сформувала статус FAILED Integrity. На другому етапі відбулася спроба криптографічної перевірки самого підпису за допомогою відкритого ключа RSA. Оскільки підпис математично прив’язаний до оригінального хешу, розшифровка підпису дала значення, що не відповідало поточному хешу документа, що автоматично призвело до статусу INVALID Signature. Таким чином, система демонструє детерміновану та надійну реакцію на будь-які спроби втручання, що повністю відповідає ключовим вимогам до систем електронного документообігу щодо гарантії незмінності підписаних файлів [6].

Окремим напрямком тестування безпеки стала перевірка захисту облікових даних користувачів. Аналіз структури файлу users.db підтвердив повну відсутність паролів у відкритому (plaintext) вигляді, оскільки всі автентифікаційні дані зберігаються виключно у вигляді криптографічних хешів. Для верифікації коректності реалізації механізму хешування було використано тестовий пароль admin123. Отриманий у системі хеш 240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9 повністю збігся з еталонним значенням SHA-256 для цього рядка, що підтверджено за допомогою незалежних онлайн-калькуляторів та офіційних тест-векторів. Це свідчить про коректну інтеграцію алгоритму, відсутність помилок кодування символів (UTF-8/ANSI) та правильну обробку вхідних даних перед хешуванням, що є критично важливим для забезпечення стабільної та передбачуваної автентифікації.

Стійкість системи до атак методом повного перебору (brute-force) та використання словникових методів оцінювалася на основі аналізу обчислювальної складності алгоритму SHA-256. Навіть за умови використання сучасного високопродуктивного обладнання, включаючи

GPU-кластери та спеціалізовані ASIC-пристрої, простір можливих комбінацій для складного пароля довжиною від 8 символів із використанням верхнього та нижнього регістрів, цифр та спеціальних символів є надзвичайно великим. Час, необхідний для успішного відновлення вихідного рядка з його хешу, значно перевищує будь-які практичні ліміти експлуатації системи та оцінюється в роки або десятиліття безперервних обчислень на сучасній апаратній базі [16]. Це гарантує, що навіть у разі фізичної або програмної компрометації файлу бази даних облікові записи користувачів залишаються захищеними, а несанкціонований доступ до системи через підбір паролів є практично неможливим без залучення технологій соціальної інженерії або фішингу.

Підсумовуючи результати перевірок, можна констатувати, що розроблена система демонструє високий рівень криптографічної стійкості та відповідність сучасним стандартам інформаційної безпеки. Механізми верифікації цілісності на основі SHA-256 та RSA забезпечують миттєве та однозначне виявлення будь-яких змін у підписаних документах, тоді як захищене зберігання автентифікаційних даних унеможливлює компрометацію облікових записів навіть при прямому доступі до файлів сховища. Така архітектура безпеки повністю відповідає вимогам законодавства України щодо електронного документообігу, забезпечує юридичну значущість цифрових підписів та створює надійний фундамент для експлуатації системи в корпоративних та державних середовищах із підвищеними вимогами до захисту конфіденційної інформації [6], [16].

4.5 Оцінка продуктивності

Вимірювання продуктивності програмного комплексу проводилося на ізольованому тестовому стенді, конфігурація якого відповідає типовій робочій станції користувача корпоративного середовища. Апаратна платформа включала процесор Intel Core i5-12400 (6 ядер, 12 потоків, базова частота 2.5 ГГц), 16 ГБ оперативної пам'яті DDR4-3200 у двоканальному

режимі та високошвидкісний накопичувач NVMe SSD Samsung 980 Pro об'ємом 1 ТБ. Програмне забезпечення стенду: операційна система Windows 11 Pro версії 23H2 із останніми оновленнями безпеки, відсутність фонових ресурсоємних процесів під час тестування, використання режиму високої продуктивності в налаштуваннях електроживлення. Така конфігурація дозволила отримати репрезентативні результати, що відображають поведінку системи в реальних умовах експлуатації без впливу зовнішніх факторів.

Методологія вимірювань базувалася на принципах повторюваності та статистичної достовірності: кожна операція виконувалася 30 разів із проміжками для «прогріву» кешів та уникнення впливу холодного старту, після чого обчислювався середній час виконання та стандартне відхилення. Для вимірювання часу використовувано високопродуктивний таймер QueryPerformanceCounter із роздільною здатністю до 1 мікросекунди, що забезпечує точність вимірювань навіть для швидкодіючих операцій. Усі тести проводилися в однопоточному режимі для ізоляції впливу паралелізму, окрім операцій завантаження документів, де оцінювався загальний час ініціалізації інтерфейсу та відображення даних.

Таблиця 4.4 - Результати вимірювання продуктивності системи

Операція	Середній час	Вимога
Запуск + завантаження 500 документів	< 1.2 с	< 3 с
SHA-256 хешування 1 МБ тексту	< 8 мс	< 100 мс
Генерація пари ключів RSA	< 50 мс	< 500 мс
Формування підпису (DoSign)	< 5 мс	< 100 мс
Верифікація підпису	< 5 мс	< 100 мс
Відображення ListView (100 документів)	< 30 мс	< 200 мс
Збереження документів (500 записів)	< 80 мс	< 500 мс

Операція **запуску та завантаження 500 документів** є критичною для оцінки загальної чуйності системи при ініціалізації робочого сеансу. Час менше 1.2 секунди (при вимозі < 3 с) досягнуто завдяки оптимізації запитів до SQLite, використанню індексів для полів пошуку та асинхронному завантаженню даних у фоновому потоці. Це гарантує, що користувач не

відчуває затримок при переході між сесіями або відкритті великих списків документів, що особливо важливо для адміністраторів, які працюють із масивами записів.

Хешування за алгоритмом SHA-256 для блоку даних об'ємом 1 МБ виконується за час менше 8 мілісекунд, що значно нижче граничної вимоги в 100 мс. Така продуктивність досягнута завдяки використанню оптимізованої реалізації з буферизацією вводу-виводу та обробкою даних блоками по 64 байти відповідно до специфікації FIPS 180-4. Швидке хешування є фундаментом для інших криптографічних операцій: автентифікації, верифікації підписів та контролю цілісності, тому низька затримка безпосередньо впливає на загальну чуйність системи.

Генерація пари ключів RSA займає менше 50 мілісекунд при допустимому ліміті 500 мс. Варто підкреслити, що ця операція виконується лише один раз під час реєстрації користувача або при оновленні ключової пари, тому її час не впливає на щоденну роботу системи. Використано ефективний алгоритм генерації простих чисел із попередньою перевіркою на простоту методом Міллера-Рабіна, що забезпечує баланс між швидкістю та криптографічною стійкістю ключів довжиною 2048 біт.

Операції **формування підпису (DoSign) та верифікації** виконуються за час менше 5 мілісекунд кожна, що є критично важливим для забезпечення інтерактивності інтерфейсу: користувач не помічає затримки між натисканням кнопки «Підписати» та відображенням результату. Така швидкість досягнута завдяки попередньому обчисленню хешу документа та використанню оптимізованої бібліотеки для модульної арифметики. Верифікація підпису також виконується миттєво, що дозволяє системі автоматично перевіряти цілісність документів при завантаженні без відчутного впливу на продуктивність.

Відображення списку документів (ListView) із 100 елементами займає менше 30 мілісекунд, що значно нижче вимоги в 200 мс. Це досягнуто за рахунок використання віртуалізації списку (відображаються лише видимі

елементи), кешування відтворених компонентів інтерфейсу та оптимізації обробки повідомлень Windows. Користувач може плавно прокручувати великі списки без «підвисань» інтерфейсу, що підвищує суб'єктивне сприйняття якості програмного продукту.

Збереження 500 записів у базу даних виконується за час менше 80 мілісекунд завдяки використанню транзакційного режиму SQLite: усі операції вставки об'єднуються в одну транзакцію, що зменшує кількість синхронізацій із диском та накладні витрати на журналювання. Це гарантує, що масові операції імпорту або експорту даних не блокують інтерфейс та не створюють черг запитів, що особливо важливо при синхронізації з віддаленими сховищами.

Загальний висновок щодо продуктивності: усі виміряні показники суттєво перевищують встановлені нефункціональні вимоги, що свідчить про високу оптимізацію архітектурних рішень та ефективність реалізації критичних шляхів виконання. Запас продуктивності (від 2.5× до 12.5× краще за вимоги) забезпечує стійкість системи до зростання навантаження, дозволяє масштабувати рішення на більші обсяги даних та гарантує стабільну роботу навіть на менш продуктивному апаратному забезпеченні. Отримані результати підтверджують готовність програмного комплексу до експлуатації в реальному корпоративному середовищі з високими вимогами до чуйності та надійності.

4.6 Порівняльний аналіз

На основі аналізу розділу 1 та результатів тестування склало підсумкову порівняльну характеристику:

Таблиця 4.5 - Порівняльна характеристика систем електронного документообігу

Критерій	SharePoint	M-Files	Alfresco	EDMS (наша)
Вбудований ЕП без PKI	✗	✗	частково	✓
Верифікація цілісності	частково	✓	✓	✓
Відсутність залежностей	✗	✗	✗	✓

				Продовження табл. 4.5
Автономне розгортання	✗	✗	складне	✓
Версіонування документів	✓	✓	✓	✓
Відкрита ліцензія	✗	✗	✓	✓
Виявлення фальсифікацій	частково	✓	✓	✓

Розроблена система перевершує комерційні аналоги за показниками автономності та відсутності залежностей, надаючи при цьому всі базові функції ЕДО та вбудований ЕП без PKI-інфраструктури.

Проведено комплексне тестування системи EDMS. Усі 35 функціональних тест-кейсів пройдено успішно. Реалізація SHA-256 підтверджена Known Answer Tests відповідно до NIST FIPS 180-4. Механізм верифікації підпису надійно виявляє підробки. Показники продуктивності перевищують встановлені вимоги. Порівняльний аналіз демонструє конкурентні переваги системи в частині автономності та відсутності зовнішніх залежностей.

ВИСНОВКИ

У рамках виконання кваліфікаційної роботи бакалавра було успішно розроблено та впроваджено прототип корпоративної системи електронного документообігу (EDMS) з інтегрованою підсистемою криптографічного захисту та формування електронного цифрового підпису. Робота виконана в повному обсязі: досягнуто всіх поставлених наукових та практичних цілей, а також реалізовано всі заплановані завдання, що підтверджує готовність системи до експлуатації в умовах обмеженого або ізольованого інформаційного середовища.

У першому розділі проведено глибокий аналіз предметної галузі, який включав порівняльне дослідження існуючих ринкових рішень для управління документами, таких як Microsoft SharePoint, M-Files та Alfresco. Виявлено їхні ключові обмеження, зокрема залежність від хмарних сервісів, високу вартість ліцензування та складність адміністрування. Паралельно здійснено детальний огляд правової бази електронного документообігу в Україні, з акцентом на Закони № 851-IV та № 2155-VIII, а також Регламент ЄС eIDAS, що забезпечує відповідність розробки міжнародним стандартам довіри. Додатково проаналізовано теоретичні основи криптографії, зокрема алгоритми хешування SHA-256 та асиметричного шифрування RSA, які були обрані як фундаментальні компоненти безпеки системи.

У другому розділі здійснено проектування архітектури програмного забезпечення. Розроблено чотирирівневу модель системи, яка забезпечує чітке розділення рівнів презентації, бізнес-логіки, доступу до даних та криптографічного ядра. Деталізовано модель даних, що включає сутності Document, User, Signature та DocumentVersion, а також спроектовано скінченний автомат життєвого циклу документа з п’ятьма основними станами (створення, чернетка, на підписі, підписано, архівовано). Важливим результатом цього етапу стала розробка власного бінарного формату зберігання даних, оптимізованого для швидкого читання/запису та мінімізації займаного дискового простору без використання сторонніх СУБД.

У третьому розділі описано процес програмної реалізації всіх компонентів системи засобами мови C++17 та нативного Win32 API. До ключових технічних досягень належать:

1. Самостійна реалізація алгоритму SHA-256 у повній відповідності зі стандартом FIPS PUB 180-4 без залучення зовнішніх криптографічних бібліотек, що забезпечило повну автономність модуля хешування.

2. Реалізація алгоритму RSA з використанням власної функції модульного множення (mulmod), що дозволило уникнути залежності від типу даних __uint128_t, який не підтримується компілятором MSVC у 32-бітному режимі або певних конфігураціях, забезпечивши кросплатформовість коду в межах Windows.

3. Вирішення архітектурної проблеми статичної ініціалізації класів через механізм динамічної реєстрації віконного класу EDMS_Panel та прив’язки його до процедури обробки повідомлень PanelProc, що гарантує стабільність інтерфейсу.

4. Розробка адаптивного графічного інтерфейсу з використанням пакетної обробки повідомлень BeginDeferWindowPos / EndDeferWindowPos, що значно підвищило продуктивність рендерингу при зміні розмірів вікна та перемальовуванні елементів керування.

5. Створення механізму модальних діалогових вікон з власними процедурами вікна (WndProc) та циклами обробки повідомлень (ModalLoop), що забезпечило коректну блокування основного інтерфейсу під час критичних операцій (підписання, введення пароля) без зависання додатка.

У четвертому розділі проведено комплексне тестування та валідацію розробленої системи. Емпірично підтверджено, що всі 35 функціональних тест-кейсів успішно пройдені. Криптографічна коректність реалізації SHA-256 верифікована через порівняння результатів хешування з офіційними KAT-векторами (Known Answer Tests) NIST. Механізми верифікації електронного підпису продемонстрували здатність виявляти будь-які спроби підробки або зміни вмісту документа після підписання. Продуктивність

системи відповідає встановленим нефункціональним вимогам (час відгуку < 200 мс). Порівняльний аналіз показав, що розроблена система перевершує розглянуті комерційні аналоги за критерієм автономності, оскільки не потребує підключення до Інтернету або зовнішніх серверів сертифікації для базових операцій.

Практичне значення роботи полягає в тому, що розроблена система може бути негайно впроваджена на підприємствах малого та середнього бізнесу, а також в державних установах з підвищеними вимогами до безпеки, як автономне рішення для ведення юридично значущого електронного документообігу. Система усуває залежність від хмарних сервісів та зовнішньої PKI-інфраструктури, знижує операційні витрати та забезпечує повний контроль над даними.

Перспективи подальшого розвитку проекту включають кілька стратегічних напрямків: заміну алгоритму RSA на вітчизняний стандарт ДСТУ 4145-2002 (ECDSA на еліптичних кривих) для повної відповідності вимогам Держспецзв'язку; інтеграцію з кваліфікованими постачальниками довірчих послуг України для підтримки кваліфікованого електронного підпису (КЕП); розробку REST API для безшовної інтеграції з існуючими корпоративними ERP- та CRM-системами; впровадження шифрування файлів зберігання даних засобами алгоритму AES-256 для захисту конфіденційної інформації на рівні диска; а також створення веб-клієнта для забезпечення віддаленого доступу до системи через браузер.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 4423-1:2005. Інформація та документація. Управління документаційними процесами. Ч. 1: Основні положення. Київ : Держспоживстандарт, 2005. 28 с.

2. Закон України «Про електронні документи та електронний документообіг» від 22.05.2003 № 851-IV. Відомості Верховної Ради України, 2003, № 36, ст. 275. URL: https://zakon.rada.gov.ua/laws/show/851-15 (дата звернення: 22.04.2026).

3. Закон України «Про електронні довірчі послуги» від 05.10.2017 № 2155-VIII. Відомості Верховної Ради України, 2017, № 45, ст. 400. URL: https://zakon.rada.gov.ua/laws/show/2155-19 (дата звернення: 22.04.2026).

4. Barker E. Recommendation for Key Management. NIST Special Publication 800-57 Part 1 Rev. 5. Gaithersburg : NIST, 2020. 157 p. DOI: 10.6028/NIST.SP.800-57pt1r5.

5. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Reading : Addison-Wesley, 1994. 395 p. ISBN 978-0-201-63361-0.

6. Gartner Research. Market Guide for Document Management. Gartner, Inc., 2024. URL: https://www.gartner.com/en/documents/document-management (дата звернення: 22.04.2026).

7. Hinchcliffe D. SharePoint as an Intranet and Document Management Platform. *Dion Hinchcliffe's Blog*, 2022. URL: https://dionhinchcliffe.com (дата звернення: 22.04.2026).

8. ISO 15489-1:2016. Information and documentation – Records management – Part 1: Concepts and principles. Geneva : ISO, 2016. 21 p.

9. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection – Information security management systems – Requirements. Geneva : ISO, 2022. 23 p.

10. M-Files Corporation. M-Files Intelligent Information Management. White Paper. Helsinki : M-Files, 2023. 18 p.

11. Microsoft Corporation. Windows App Development Documentation. Win32 API Reference. URL: https://learn.microsoft.com/en-us/windows/win32 (дата звернення: 22.04.2026).

12. National Institute of Standards and Technology. NIST Cryptographic Algorithm Validation Program (CAVP). URL: https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program (дата звернення: 22.04.2026).

13. National Institute of Standards and Technology. Secure Hash Standard (SHS). FIPS PUB 180-4. Gaithersburg : NIST, 2015. 35 p. DOI: 10.6028/NIST.FIPS.180-4.

14. Petzold C. Programming Windows. 6th ed. Redmond : Microsoft Press, 2013. 1296 p. ISBN 978-0-7356-7176-8.

15. Potts J. Alfresco: The Definitive Guide. Boston : O'Reilly Media, 2007. 412 p. ISBN 978-0-596-51352-5.

16. Regulation (EU) No 910/2014 of the European Parliament and of the Council on electronic identification and trust services for electronic transactions in the internal market (eIDAS). *Official Journal of the European Union*, 2014, L 257, pp. 73–114.

17. Rivest R. L., Shamir A., Adleman L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*. 1978. Vol. 21, No. 2. P. 120–126. DOI: 10.1145/359340.359342.

18. Sprague R. H. A Framework for the Development of Decision Support Systems. *MIS Quarterly*. 1980. Vol. 4, No. 4. P. 1–26. DOI: 10.2307/248957.

19. Stallings W. Cryptography and Network Security: Principles and Practice. 8th ed. Hoboken : Pearson, 2022. 816 p. ISBN 978-0-13-568974-4.

20. Stroustrup B. The C++ Programming Language. 4th ed. Upper Saddle River : Addison-Wesley, 2013. 1346 p. ISBN 978-0-321-56384-2.

ДОДАТОК А ОСНОВНІ ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

А.1 Реалізація SHA-256 (crypto.cpp)

```
std::string SHA256::hash(const std::string& input) {     std::vector<uint8_t>
msg(input.begin(), input.end());     uint64_t bitLen = (uint64_t)msg.size() * 8;
msg.push_back(0x80);     while (msg.size() % 64 != 56)
msg.push_back(0x00);     for (int i = 7; i >= 0; --i)
msg.push_back((uint8_t)((bitLen >> (i * 8)) & 0xFF));      uint32_t h[8] = {
0x6a09e667, 0xbb67ae85, 0x3c6ef372, 0xa54ff53a,         0x510e527f,
0x9b05688c, 0x1f83d9ab, 0x5be0cd19     };     for (size_t i = 0; i < msg.size();
i += 64) {         uint32_t w[64] = {};         for (int j=0;j<16;++j)
w[j]=((uint32_t)msg[i+j*4]<<24)|((uint32_t)msg[i+j*4+1]<<16)
|((uint32_t)msg[i+j*4+2]<<8)|(uint32_t)msg[i+j*4+3];         for (int
j=16;j<64;++j)             w[j]=G1(w[j-2])+w[j-7]+G0(w[j-15])+w[j-16];
uint32_t a=h[0],b=h[1],c=h[2],d=h[3],
e=h[4],f=h[5],g=h[6],hh=h[7];         for (int j=0;j<64;++j) {             uint32_t
t1=hh+S1(e)+ch(e,f,g)+K256[j]+w[j];             uint32_t t2=S0(a)+maj(a,b,c);
hh=g;g=f;f=e;e=d+t1;d=c;c=b;b=a;a=t1+t2;         }
h[0]+=a;h[1]+=b;h[2]+=c;h[3]+=d;         h[4]+=e;h[5]+=f;h[6]+=g;h[7]+=hh;
}     std::ostringstream oss;     for(int i=0;i<8;++i)
oss<<std::hex<<std::setw(8)<<std::setfill('0')<<h[i];     return oss.str(); }
```

А.2 Реалізація PanelProc та ModalLoop (main_gui.cpp)

```
// Власний клас панелі - виправлення проблеми STATIC static LRESULT
CALLBACK PanelProc(HWND hw, UINT msg,
WPARAM wp, LPARAM lp) {     if (msg == WM_COMMAND)         return
SendMessageW(g_hMain, WM_COMMAND, wp, lp);     if (msg ==
WM_NOTIFY)         return SendMessageW(g_hMain, WM_NOTIFY, wp,
lp);     if (msg == WM_ERASEBKGND) {         RECT r; GetClientRect(hw,
&r);         FillRect((HDC)wp, &r, (HBRUSH)(COLOR_BTNFACE+1));
return 1;     }     return DefWindowProcW(hw, msg, wp, lp); }  // Правильна
модальність без HWND-порівнянь static void ModalLoop(HWND hw) {
```

```
EnableWindow(g_hMain, FALSE);  MSG m;  while (IsWindow(hw) &&
GetMessageW(&m, nullptr, 0, 0) > 0) {  if (!IsDialogMessageW(hw,
&m)) {  TranslateMessage(&m);  DispatchMessageW(&m);
}  }  EnableWindow(g_hMain, TRUE);
SetForegroundWindow(g_hMain); }
```

A.3 Адаптивний resize через DeferWindowPos (main_gui.cpp)

```
case WM_SIZE: {  int W = LOWORD(lp), H = HIWORD(lp);  if
(g_hStatus) SendMessageW(g_hStatus, WM_SIZE, wp, lp);  int sbH = 22;
if (g_hStatus) { RECT r; GetClientRect(g_hStatus,&r);
if(r.bottom>0) sbH=r.bottom; }  int panY=TAB_H, panH=H-panY-sbH;
if(panH<50) panH=50;  int listH=panH-TOOL_H; if(listH<10) listH=10;
int sy=(TOOL_H-22)/2;  HDWP hdwp = BeginDeferWindowPos(10);
auto Defer = [&](HWND h, int x, int y, int w, int hh) {  if(h)
hdwp=DeferWindowPos(hdwp,h,nullptr,x,y,w,hh,
SWP_NOZORDER|SWP_NOACTIVATE);  };  Defer(g_hTab,  0, 0, W,
TAB_H);  Defer(g_hPanelDocs,  0, panY, W, panH);  Defer(g_hListDocs,
0, TOOL_H, W, listH);  Defer(g_hSearchLbl,  W-360, sy+2, 55, 18);
Defer(GetDlgItem(g_hPanelDocs,IDC_EDIT_SEARCH),  W-303,sy,162,22);
Defer(GetDlgItem(g_hPanelDocs,IDC_BTN_SEARCH),  W-138,sy, 30,22);
Defer(GetDlgItem(g_hPanelDocs,IDC_COMBO_FILTER), W-105,sy,
95,130);  Defer(g_hPanelUsers, 0, panY, W, panH);  Defer(g_hListUsers,
0, TOOL_H, W, listH);  EndDeferWindowPos(hdwp);  break; }
```