Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи захищеного документообігу, розгорнутої на мобільних пристроях”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Щербаков В.Г.
«____» _____ 2025 р.

Керівник проекту
доктор філософії (PhD)
_____ Усік П.С.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Щербакову Владиславу Георгійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи захищеного документообігу, розгорнутої на мобільних пристроях*

2. Керівник роботи *Усік Павло Сергійович, доктор філософії (PhD)*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 46-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи захищеного документообігу, розгорнутої на мобільних пристроях*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Усік П.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Щербаков В.Г.
(прізвище та ініціали)

АНОТАЦІЯ

Щербаков В.Г. Програмне забезпечення системи захищеного документообігу, розгорнутої на мобільних пристроях. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи захищеного документообігу, розгорнутої на мобільних пристроях.

Метою розробки є програмне забезпечення системи захищеного документообігу, розгорнутої на мобільних пристроях.

Результат роботи – програмна реалізація системи захищеного документообігу, розгорнутої на мобільних пристроях.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на мобільних пристроях під керуванням ОС Android.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, захищений документообіг

ABSTRACT

Shcherbakov V.G. Software for a secure document management system deployed on mobile devices. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a secure document management system deployed on mobile devices.

The purpose of the development is software for a secure document management system deployed on mobile devices.

The result of the work is a software implementation of a secure document management system deployed on mobile devices.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on mobile devices running the Android OS.

The program was developed in the Python environment.

Keywords: computer engineering, secure document management

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

КСЗ	–	комплексна система захисту
ПЕОМ	–	персональна електронно-обчислювальна машина
СУБД	–	система управління базами даних
PGP	–	Pretty Good Privacy

ВСТУП

Актуальність теми. Захисту систем електронного документообігу сьогодні приділяється не менше уваги, ніж їхнім функціональним можливостям. З поширенням мобільного доступу до корпоративних систем виникають нові ризики, однак індустрія інформаційної безпеки розвивається так само успішно й здатна відповісти на ці виклики. За два десятки років свого існування український ринок систем електронного документообігу (СЕД; Enterprise Content Management, ECM) пройшов шлях від простих систем реєстрації й обліку до багатофункціональних рішень, що поєднують засоби керування документами, бізнес-процесами й можливості колективної роботи. Потреби багатьох організацій не обмежуються автоматизацією традиційних завдань – СЕД використовується для підтримки самих різних бізнес-процесів: керування закупівлями, тендерами, маркетинговими кампаніями, нормативною документацією, договорами й т.д. Нерідко цей додаток є критично важливим для організації.

Мобільність стає одним з істотних засобів підвищення ефективності роботи в компаніях різного профілю й з різною чисельністю співробітників. Як позначається «корпоративна мобільність» на системах електронного документообігу? Наскільки підходить форм-фактор планшета/смартфону для завдань СЕД? Підтримка мобільності в системах електронного документообігу досить перспективна, оскільки такий підхід дуже зручний при виконанні певних завдань. Практично в кожній організації є «мобільні» співробітники, що працюють поза офісом, і вони теж повинні мати доступ до актуальної інформації й брати участь у робочих процесах».

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи захищеного документообігу, розгорнутої на мобільних пристроях.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем захищеного документообігу, розгорнутої на мобільних пристроях.

– Дослідження системи захищеного документообігу, розгорнутої на мобільних пристроях.

– Програмна реалізація системи захищеного документообігу, розгорнутої на мобільних пристроях.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі захищеного документообігу, розгорнутої на мобільних пристроях.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи захищеного документообігу, розгорнутої на мобільних пристроях, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Перспективність розвитку мобільних інтерфейсів до СЕД не підлягає сумніву. За умови забезпечення інформаційної безпеки (ІБ) багато етапів процесів роботи з документами будуть переноситися на мобільні платформи. Зараз співробітники найчастіше працюють зі своєю електронною поштою, використовуючи смартфони й планшети. Точно так само вони зможуть (і багато хто вже це роблять) погоджувати документи й виконувати ті або інші завдання за допомогою мобільних пристроїв.

Однак форм-фактор планшетів і смартфонів зручний не завжди. Наприклад, для керівників вищої й середньої ланки, відповідальних за постановку завдань, контроль і виконання доручень, узгодження документів і т.д., він цілком прийнятний, але для тих, хто займається реєстрацією документів, таких форм-фактор не підходить і навряд чи коли-або стане придатним, у всякому разі поки зберігається традиційний підхід до цього процесу. Офіційні документи, що підлягають реєстрації, містять безліч реквізитів, які незручно вносити з мобільного пристрою. Крім того, звичайно для діловода спеціально обладнається стаціонарне місце, оснащене, зокрема, засобами уведення-виводу документів.

Уважається, що складні багатофункціональні мобільні робітники місця вимагають застосування ноутбуків/ультрабуків. У дійсності це залежить від того, хто використовує СЕД/ECM. Професійному діловодові навряд чи необхідно мобільне робоче місце. Якщо ж співробітник звертається віддалено (у відрядженнях або на виїздах) до інших корпоративних систем (наприклад, системам обліку, проектного керування, CRM, BI і т.д.) або створює в процесі своєї діяльності документи (специфікації, договори, пропозиції, презентації), то,

швидше за все, його робітником інструментом є ноутбук або ультрабук, а не планшет. У цьому випадку ніщо не заважає встановити повнофункціональний клієнт до СЕД, що надасть максимум можливостей для роботи із системою. Якщо ж основні потреби обмежуються роботою з поштою, контролем завдань і узгодженням документів, причому співробітник не користується постійно ноутбуком, віддаючи перевагу планшету або телефону, то мобільного клієнта СЕД цілком достатньо.

1.2 Область застосування

Яка сьогодні ситуація з підтримкою мобільності в системах СЕД/ECM? Наскільки широко компанії використовують мобільні робочі місця і які завдання вирішуються з їхньою допомогою? Попит на мобільність є – відповідно, є й пропозиції від виробників СЕД і тих, хто впроваджує: як стандартні «коробкові» варіанти, так і замовлені розробки. Зараз при виборі СЕД практично кожна організація цікавиться засобами мобільного доступу поза залежністю від того, чи будуть вони реально застосовуватися. Таким чином, їхня наявність у портфелі пропозицій виробника стає обов'язковим.

Без підтримки мобільних місць СЕД уже не може вважатися сучасною. У ряді ринкових ніш, наприклад в органах державного керування, необхідність у мобільних робочих місцях виражена досить слабко, але й там є керівники вищого рангу, для яких розробляються відповідні АРМ. У корпоративному секторі практично всі впроваджувані СЕД мають повноцінних мобільних клієнтів, оскільки їхня наявність – один з вагомих критеріїв вибору системи.

У корпоративному секторі мобільні робочі місця є приблизно в однієї компанії з п'яти. При цьому майже всі працюють із СЕД віддалено, підключаючись до неї по VPN або за допомогою термінального доступу. Є й виключення – організації із закритої для зовнішніх користувачів корпоративною мережею передачі даних, наприклад банки. Основними користувачами мобільних

робочих місць є в першу чергу топ-менеджери й уже потім – керівники середньої ланки. І ті й інші проводять досить багато часу поза своїм офісом, і більшість типових для таких користувачів завдань СЕД – технічно нескладні дії, що вимагають швидкого виконання (робота з узгодженнями й завданнями).

Планшети й смартфони не можуть повністю замінити ноутбуки або робочі станції, але відмінно підійдуть споживачам контенту. До таких користувачів відносяться топ-менеджери, від яких залежить оперативне прийняття рішень. У керівників, що не перебувають постійно в офісі, потреба в таких рішеннях дійсно дуже висока. Як правило, вони не створюють документи самостійно, скоріше навпаки – читають, вивчають, візують уже готові або відносять завдання своїм співробітникам. Смартфон з мобільним клієнтом СЕД прекрасно підходить для цих цілей. На сьогоднішній день підтримка мобільних клієнтів СЕД популярна в першу чергу саме серед керівників організацій. Їм потрібно забезпечити безпечний доступ до оперативної інформації й аналітики, надати можливість переглядати документи й приймати рішення.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи захищеного документообігу, розгорнутої на мобільних пристроях, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

AVACCO корпоративне керування

AVACCO-корпоративне керування – це комплексна інформаційна система, що не тільки забезпечує ефективне функціонування підприємства, але й здатна супроводжувати й стимулювати його майбутній розвиток. Система дозволяє організувати автоматизований управлінський, фінансовий, складський і виробничий облік; забезпечити електронний документообіг у рамках єдиного інформаційного простору. Гнучкість і відкритість системи дає можливість адаптувати її практично для будь-якої моделі діяльності й динамічно розвивати надалі. Виконуючи «штатні» функції класичної інформаційної системи, AVACCO-корпоративне керування містить у собі автоматизовані робочі місця фахівців планово-економічних відділів, кадрових служб, бухгалтерів, логістів, складських працівників.

Система AVACCO-корпоративне керування розроблявся як інструмент для оптимізації управлінської діяльності, тому орієнтовано вона, насамперед, на менеджерів різних рівнів. Оскільки якість управлінських рішень засновано на точної, вчасно отриманої й належним чином представленої інформації, системою відслідковуються й збираються оперативні дані про рух фінансових і матеріальних потоків, а також про показники виконання користувачами поточних завдань. Зібрана інформація бути основою для побудови системи збалансованих показників, а також регулярного моніторингу ефективності роботи співробітників і підрозділів. Велика увага розроблювачі системи приділили постановці

управлінського й фінансового обліку. Аналітичний блок системи дозволяє надати різну управлінську звітність, а також звітність по стандартах МСФО (GAAP).

Система легко адаптується для підприємств із різними формами власності й типами структур, незалежно від сфери діяльності, вирішуючи загальні завдання обліку, контролю й керування.

Загальні завдання, розв'язувані системою AVACCO-корпоративне керування

Керування товарними потоками

Ефективне керування товарними потоками неможливо без побудови інтегрованої інформаційної моделі руху товарів, що охоплює всі ділянки діяльності підприємства. У рамках системи AVACCO-корпоративне керування, що реалізує таку модель на базі єдиного інформаційного простору, є можливість організувати:

– Керування складами.

– Керування продажами.

– Керування закупівлями.

– Керування перевезеннями.

– Облік товарно-матеріальних цінностей.

– Взаємодія між окремими процесами руху товарів.

Система забезпечує ведення довідника товарів, що містить не тільки статичну довідкову інформацію, але й обновлювані в режимі реального часу детальні дані про товарні залишки, очікувані надходження, динамік продажів.

Система підтримує ведення будь-якої кількості прайс-аркушів і гнучкої політики ціноутворення.

Система дозволяє формувати стандартний пакет взаємопогоджуваних документів, що супроводжують процеси продажу, закупівлі, доставки, складські операції (рахунку, накладні, ф'ючерси, заявки на закупівлю й ін.).

У системі реалізований унікальний механізм динамічного резервування товарів, при якому необхідна кількість товару резервується на конкретну дату

його майбутнього відвантаження (надходження). Алгоритм динамічного резервування дозволяє обробити такі складні ситуації, коли відбувається зрив поставки, відмова клієнта від товару й ін. Поняття динамічного резервування в системі AVACCO розширюється поняттям черги на резерв, що дає можливість вирішувати завдання планування закупівель, управляти поставками за принципом «точно в строк», управляти забезпеченням виробництва сировиною.

Реалізований у системі AVACCO-корпоративне керування підхід до керування товарними потоками дозволяє:

– ефективно управляти надходженнями й відвантаженнями;

– завжди мати в наявність товар необхідний до відвантаження;

– мати план відвантажень на майбутній період, що полегшує процес закупівель і фінансового планування;

– збільшити оборотність засобів, оскільки скорочується час перебування товару на складі;

– управляти забезпеченням виробництва комплектуючими виробами;

– одержувати різноманітну інформацію з руху товару.

Керування фінансовими потоками й ведення фінансового обліку

Система AVACCO-корпоративне керування оптимізує фінансову діяльність підприємства за рахунок упорядкування фінансових потоків, а також автоматизації діяльності бухгалтерії й фінансового відділу. Основу фінансового обліку становить облік господарських операцій, здійснюваний виходячи із загальноприйнятих концепцій бухгалтерського обліку. У системі реалізовані оригінальні механізми обліку, що розширюють типові механізми й поняття бухгалтерського обліку. Стандартна операція подвійного запису здійснюється не тільки між двома рахунками, але й між двома контрагентами й валютами, із вказівкою переліку товарів. Для здійснення обліку господарських операцій у системі AVACCO існує модуль «Фінансові операції».

AVACCO-корпоративне керування забезпечує:

– облік руху грошових коштів, відповідно до прийнятого в компанії політиці обліку;

– ведення мультивалютного обліку;

– моніторинг стану взаєморозрахунків з постачальниками й покупцями;

– автоматизацію бухгалтерського, фінансового, управлінського й податкового обліку;

– ведення обліку й надання звітності (звіт про прибутки й збитки, балансовий звіт, баланс у системі МСФО);

– консолідацію даних по всіх підрозділах організації, і по всіх організаціях, що входить у територіально-розподілену структуру, формування зведеної звітності;

– можливість розрахунку показників, що відбивають ефективність роботи компанії.

Система підтримує ведення необхідних фінансових довідників: довідника валют, довідника рахунків, довідника категорій, дозволяє формувати необхідні банківські й касові документи, а також будувати різноманітні персоніфіковані звіти. Для ведення бухгалтерської звітності відповідно до українських стандартів система може бути інтегрована із продуктами компанії 1С.

Управлінський облік і контролінг

AVACCO-корпоративне керування – це комплексний програмний продукт, що охоплює всі потреби обліку первинної інформації, у тому числі й інформації, що є основою для ведення управлінського обліку.

У системі реалізована концепція «конвеєрного керування» – керування по завданнях, відповідно до якого:

– Усі бізнес-процеси організації являють собою строго певну послідовність завдань, за виконання кожного завдання відповідає закріплений співробітник.

– Кожний користувач системи може бачити поточний список своїх завдань, що підлягають виконанню.

– Система контролює відповідність дій кожного виконавця логіці виконуваного бізнесу-процесу.

– Виконання завдання приводить до автоматичного виникнення завдання в наступного виконавця.

– Система надає всю необхідну інформацію для ухвалення рішення на будь-якому рівні керування.

– Сервер бізнес-процесів зберігає інформацію про час появи, часу і якості виконання завдання.

– Менеджери одержують можливість у режимі реального часу відслідковувати завантаженість своїх підлеглих, що дозволяє оперативно реагувати на виникаючі проблеми.

Система забезпечує інформаційну підтримку контролінгу, як найбільш передовий на сьогоднішній день технології менеджменту.

Фахівці компанії AVACCO Soft роблять послуги з розробки системи показників управлінського обліку, орієнтованих на потребі керівництва конкретного підприємства, і які є методологічною основою контролінгу. У процесі функціонування системи виробляється збір, узагальнення, формалізація й консолідація даних управлінського обліку. Сервер бізнес-процесів системи ефективно вирішує такі управлінські проблеми, як пошук інформації для ухвалення рішення, відстеження руху документів (docflow), самостійна постановка завдань співробітниками (workflow), візуальний/системний контроль над роботою окремого співробітника (підрозділу, всієї організації) і т.д. Автоматичне декларування завдань дозволяє планувати людські ресурси, товарні й фінансові операції, строки виконання завдань.

Залежно від настроєних прав доступу керівник будь-якого рівня в on-line режимі може побачити:

– перевантажені й вільні підрозділ/співробітник;

– чергові завдання для кожного співробітника;

– на якій стадії обробки перебуває той або інший документ та ін.

Можливості системи дозволяють наочно відобразити:

– Організаційну структуру підприємства (відділи підрозділу, філії, співробітників кожного відділу й ін.).

– Структуру балансу підприємства.

– Структуру бюджету підприємства.

– Показники управлінського обліку.

– Персоніфіковану звітну інформацію для керівників.

Система забезпечує інформаційну прозорість фінансово-господарської діяльності, що особливо важливо для підвищення інвестиційної привабливості підприємства.

Basware Procurement

Це керування закупівлями, завдяки відмові від ручної обробки документів і, що немаловажно, забезпеченню набагато більше високого рівня відповідності закупівель умовам угод з постачальниками. Серед переваг можна виділити додаткові можливості економії, усунення паперової роботи, а також скорочення часу, що йде на відстеження операцій з постачальниками й виправлення помилок. Широкі масштаби впровадження й високий рівень контролю за витратами також привносять значні переваги. Basware Procurement також дозволяє здійснювати зіставлення рахунків на закупівлю із замовленнями за допомогою модуля Basware Order Matching, що є ключовим чинником автоматизації й ефективності. Інформація відносно готівок коштів, витрати засобів і зобов'язань, що фіксується в базі дані рішення по закупівлях, дає керівництву більше повне розуміння грошових потоків організації й положення з оборотним капіталом, що сприяє істотному підвищенню ефективності, точності й маневреності в керуванні фінансами й корпоративною діяльністю.

Bellview SCAN

Bellview SCAN – це високопродуктивна інтегрована система автоматизованого уведення й обробки більших обсягів однотипних друкованих форм за допомогою сканування. Система заснована на архітектурі клієнт-сервер, добре масштабується й може бути використана в якості Front-Enda практично будь-якої системи документообігу або корпоративного сервера баз даних.

Модулі Bellview SCAN дають можливість провести повний цикл обробки оптувальників і інших форм (сканування, розпізнавання, редагування, перевірка й експорт даних) швидко, ефективно, точно й з мінімальними витратами

CompanyMedia

CompanyMedia – корпоративна система керування документами, завданнями й особистою продуктивністю. Зберігаючи функції діловодства, система сфальцьована на роботі керівників і бізнес-фахівців. Цим категоріям працівників CompanyMedia надає інструменти для аналізу й прийняття управлінських рішень, оцінки ефективності персоналу, підвищення результативності основної діяльності організації. Найбільші переваги від впровадження системи одержують територіально розподілені організації.

Система підтримує платформну інваріантність: у якості ПЗ системного й проміжного рівнів можуть використовуватися різні СУБД, ECM-платформи, операційні системи й офісні пакети, включаючи рішення на базі СПО. Відкритість архітектури забезпечується за рахунок підтримки індустріальних концепцій, специфікацій і стандартів, таких як BPMN 2.0, CMIS, HTML5, REST. Мультиплатформеність програмних продуктів «ІнтерТраст» є одним із ключових переваг, що забезпечує замовникам нашої компанії більшу гнучкість при побудові своєї інформаційної стратегії.

На прикладному рівні система являє собою набір інтегрованих модулів і бізнес-рішень, призначених для роботи з певним типом документів: вхідними й вихідними, договорами, зверненнями громадян, повістками дня, протоколами засідань і іншим значимим контентом. Гнучке сполучення функціональності, закладеної в модулях системи, дозволяє створювати бізнес-рішення з

урахуванням організаційної структури, стилю керування й галузевих особливостей організації-замовника.

Ефективну роботу й взаємодію бізнес-рішень CompanyMedia забезпечують базові сервіси: загалькорпоративні й локальні довідники й класифікатори, служба керування контентом, служба пошуку, служба колективної роботи з документами, служба аналітичної обробки даних і побудови звітів, служба керування потоками операцій, служба єдиного автоматизованого контролю виконання доручень і резолюцій і ін.:

– Відкрита, довірена й захищена СЕД. CompanyMedia у її актуальній версії – повністю українська розробка, що відповідає вимогам в області імпортозаміщення й принципам відкритості, дорученню й захищеності.

– Перша на українському ринку реалізація технології адаптивного кейса-менеджменту. Технологія адаптивний кейс-менеджменту призначена для керування неструктурованими й частково структурованими бізнесами-процесами.

– Персоналізація контенту й інтерфейсу. У єдиному web-інтерфейсі користувачам надається доступ до зовнішньої ділової кореспонденції, внутрішній переписці, матеріалам проектних груп, колегіальних органів, дорученням і розпорядницьким документам, договорам і іншому контенту, необхідному для повсякденної роботи.

– Широкий вибір мобільних додатків. Мобільне робоче місце CompanyMedia призначене для керівників і бізнес-фахівців.

– Елементи соціальності. Соціальні інструменти системи забезпечують горизонтальні зв'язки між функціональними підрозділами й філіями організації, руйнують організаційні бар'єри, що заважають колективній роботі.

– СЕД як сервіс. Організація електронного документообігу – необхідний технологічний сервіс, що супроводжує діяльності будь-якої організації.

– Реальна корпоративність. Система орієнтована на автоматизацію документообігу, процесів і керування проектами в територіально розподілених організаціях зі складною структурою.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ.

Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді для всіх основних платформ на веб-сайті Python https://www.python.org/ і можуть вільно поширюватися. Цей же сайт також містить дистрибутиви та вказівники на багато безкоштовних сторонніх модулів Python, програм і інструментів, а також додаткову документацію.

Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для налаштовуваних програм.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи захищеного документообігу, розгорнутої на мобільних пристроях.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи

технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Мобільність перестала бути перевагою й перетворилася, скоріше, в обов'язковий фактор в умовах загальної й всебічної «мобілізації». Тому без підтримки мобільності рішення не буде сучасним. Основні розроблювачі СЕД затурбувалися написанням мобільних клієнтів, тому що це стало одним з найпоширеніших вимог замовників. Проте деякі організації не планували й не планують використовувати мобільні клієнти через погрози витоку конфіденційної інформації. Причина – висока оцінка ризиків або надмірна вартість дійсно захищених комплексних рішень.

Поряд з мобільністю сьогодні можна говорити про те, що в СЕД відбувається зсув фокуса з документа на взаємодію, колективну роботу для досягнення результатів бізнесу. Даний процес іде вже давно, і найбільше яскраво це проявляється в корпоративному секторі. Стосовно до завдань бізнесу комерційні організації більше орієнтовані на кінцеві результати використання СЕД, а не на автоматизацію документообігу в класичному розумінні, тобто процеси реєстрації й обліку документів відходять на другий план. Концепція «документо-орієнтованих» СЕД починає розмиватися: у центрі системи вже перебувають безпосередньо дії й досягнуті результати.

Корпоративні шини доручень, колективна робота, підтримка робочих груп, супровід клієнтів – передній край розвитку СЕД і найбільш обговорювані серед їхніх виробників тенденції.

За останнім часом у багатьох лідируючих на ринку українських систем з'явилися розвинені функції відповідного призначення, які будуть постійно вдосконалюватися.

ЕЦП і автентифікація в СЕД

Дуже важливо забезпечити довіра до операцій, виконаним з мобільного пристрою, і це завдання вирішується шляхом використання механізмів електронного підпису (ЕЦП). Електронний підпис дозволяє забезпечити незмінність підписаних документів і однозначно встановити, ким вони створені.

Функції електронного підпису потрібні в першу чергу там, де необхідно виконати вимоги законодавства і юридичний статус підпису дуже важливий. У цих випадках для її створення використовуються сертифіковані криптографічні засоби.

Існує кілька варіантів роботи з ЕЦП на мобільному пристрої, але найбільш безпечним є використання персонального засобу для формування підпису у вигляді відчужуваного модуля. Наприклад, у випадку із платформою iOS таким пристроєм може бути смарт-карта, підключена через контактний або бездротовий зчитувач, а для Android-пристроїв – звичайний USB-токен (підключений через перехідник) або компактний microUSB-токен.

За допомогою смарт-карти або USB-токена керівник може автентифікуватися в мобільному додатку, установлювати захищене з'єднання до сервера або хмари, а також перевіряти й формувати електронний підпис на документах. Крім виконання двофакторної автентифікації, ці захищені пристрої можуть безпечно зберігати ключі шифрування для програмних СКЗІ, установлених на смартфоні або планшеті, і для шифрування конфіденційної інформації, збереженої на мобільному пристрої.

Функції електронного підпису й забезпечення інформаційної безпеки при роботі з мобільної СЕД/ECM уважаються одними із самих затребуваних. Однак практика показує, що на мобільних пристроях ЕЦП поки мало популярна. Видимо, це пов'язане з тим, що її використання в документообігу ще не одержало широкого поширення навіть у великих організаціях. Проте ряд засобів дозволяють забезпечити наскрізну підтримку застосування ЕЦП у процесах документообігу, у тому числі за допомогою мобільних пристроїв.

Технічну проблему представляє оптимізація трафіку між СЕД і мобільним пристроєм при передачі даних для підписання, адже звичайно СЕД надає багато непотрібної мобільному клієнтові інформації. Однак модифікація переданого для підпису вмісту неможлива, тому що це зробить ЕЦП недійсної. Тому на підпис доводиться передавати весь контент у незмінному виді, а для оптимізації трафіку шукати інші шляхи.

Крім того, для повноцінної перевірки ЕЦП на планшеті необхідно вмонтувати в додаток додаткові засоби, інтегровані з мобільним криптопровайдером. Математичні алгоритми підпису забезпечуються засобами криптопровайдера, а побудова ланцюжка довіри для сертифікатів і керування ключами – це функції, які реалізуються безпосередньо в захищеному мобільному клієнті.

У будь-якому мобільному рішенні, де використовується сертифікована криптографія, застосовується двофакторна автентифікація. Перший фактор – сам мобільний пристрій, другий – PIN-код. Що стосується необхідності застосування фізичних токенів, які привносять третього фактора, те тут вибір за замовником.

Як показує практика, зараз багато комерційних організацій створюють системи документообігу, що припускають обов'язкове узгодження документів. Це робиться на базі порталів Microsoft SharePoint або на основі інших спеціалізованих Web-Систем. Деякі компанії використовують схеми доступу через Web-браузери (що більшою мірою затребувано в банківській галузі), і в цьому випадку теж не обійтися без двофакторної авторизації, що дозволяє бізнесу бути більше мобільним. Двофакторна автентифікація вже стала стандартом де-факто в системах вилученого доступу, де ім'я й пароля недостатньо для забезпечення належного рівня безпеки. Додатково задіються токени, одноразові паролі, SMS-паролі й т.ін. Звичайно, приймається в розрахунок критичність даних. Деякі великі компанії за допомогою другого фактора здійснюють захист вилученого доступу до корпоративних ресурсів, у тому числі до електронної пошти».

Електронний підпис для мобільних пристроїв дозволяє організувати безпечний доступ до корпоративних систем і сервісів з мобільних пристроїв і забезпечує юридичну значимість електронних документів, що підписуються, і вироблених операцій. Замовникам пропонується цілий ряд функцій: взаємна двофакторна автентифікація, формування посиленого кваліфікованого електронного підпису й безпечне зберігання ключів і цифрових сертифікатів на відчужуваному модулі безпеки (смарт-карті або токені Secure MicroSD).

Украй бажані також механізми видалення конфіденційної інформації. Наприклад використовується захист від злому пристрою (Jailbreak, Root), багаторазового неправильного уведення PIN-коду й порушення цілісності дистрибутивів. При настанні цих подій всі ключі й документи віддаляються, додаток блокується й не запускається.

Що стосується захисту переданих даних, то для пристроїв під керуванням iOS вирішують це завдання як шляхом вбудовування захисту в мобільний додаток, так і за допомогою окремого додатка «Захищений тунель». Обидва варіанти використовують реалізацію протоколу TLS, що поставляється в складі СКЗІ для iOS. Цей протокол забезпечує криптографічну двосторонню автентифікацію клієнта й сервера, контроль цілісності й шифрування даних у процесі інформаційного обміну. Застосування «Захищеного тунелю» зручно тим, що дозволяє захищати трафік сторонніх СЕД-клієнтів для iOS без якої-небудь їхньої модифікації.

Головні перешкоди для мобільності

Які технологічні, організаційні, законодавчі проблеми перешкоджають широкому поширенню мобільності й, зокрема, використанню мобільних пристроїв у системах документообігу?

Незважаючи на те що багато виробників систем документообігу представляють «коробкові» мобільні робочі місця, на практиці стандартні мобільні клієнти підходять лише для базових сценаріїв роботи й застосовні в основному для невеликих організацій. А для таких замовників вартість мобільних

клієнтів може виявитися непомірно високою – іноді вона порівнянна з вартістю СЕД у цілому. Можливо, згодом удасться знизити ціну стандартної функціональності до рівня ПЗ, продаваного через магазини додатків, але поки – через невисокий попит – розробка й підтримка таких мобільних клієнтів досить коштовні.

У великих організаціях ситуація інша – там процеси документообігу істотно складніше, СЕД звичайно сильно кастомизировані, а мобільні робочі місця найчастіше створюються розраховуючи на конкретних топ-менеджерів, що є їхніми основними користувачами. У підсумку вартість впровадження мобільних клієнтів може бути досить великий, що трохи обмежує поширення таких рішень.

Питання мобільної безпеки в корпоративному секторі також роблять свій вплив: у деяких організаціях забороняється працювати з корпоративною інформаційною системою з використанням мобільних пристроїв. Однак у цій області спостерігається певний прогрес, оскільки в міру розвитку вітчизняних мобільних засобів ІБ захищеність систем росте, а ризики втрати конфіденційних даних знижуються.

Причин, що перешкоджають масовому поширенню мобільних клієнтів у системах документообігу, може бути трохи, і в кожній організації вони свої. Найчастіше мова йде про велику розмаїтість пристроїв, їхньої високої вартості, а також короткому життєвому циклі (коли кожні 8-10 місяців випускаються нові моделі). Сюди ж можна віднести питання забезпечення безпеки інформації на мобільних пристроях і керування ними (MDM), підтримку мобільних платформ / операційних систем, наявність відповідних сертифікатів, виданих регуляторами».

3.2 Розробка структурної схеми

На мобільних пристроях співробітників (особливо тих, у кого є мобільні додатки, підключені до інформаційних систем підприємства) перебуває безліч конфіденційних даних, у першу чергу корпоративна пошта й службові

документи. Параметри й права доступу до захищених ресурсів звичайно зберігаються там же.

Тому наявність мобільних пристроїв – це не тільки можливість ефективно працювати поза офісом, але й великий ризик для організації. Якщо раніше доступ до закритих даних вимагав проникнення в мережу компанії, то зараз чимало такої інформації перебуває «зовні» – у співробітників. А смартфони й планшети можуть втратитися або бути украдені разом з усім їхнім умістом. Тому консолідація в руках служби ІБ хоча б таких функцій, як блокування доступу й видалення даних з мобільного пристрою при його втраті, може серйозно знизити рівень погроз.

Ще одна корисна функція централізованого керування – можливість синхронізації настроювань, версій ПЗ й конфігурації корпоративних додатків. Пристрою (якщо вони перебувають в особистій власності) міняються часто, і при заміні можна відразу застосувати необхідні настроювання й установити всі потрібні додатки, причому їхні актуальні версії.

Якщо для роботи із СЕД використовуються мобільні пристрої, необхідний комплексний підхід, а виходить, і MDM.

Потреба в MDM залежить від сценарію доступу. Якщо використовуються Web-технології й на мобільному пристрої документи не зберігаються, то без впровадження MDM цілком можна обійтися. При наявності конфіденційних документів на самому пристрої створення шифрованого й керованого контейнера обов'язково, і в цьому випадку активно використовуються рішення MDM. Цей підхід відносно легко реалізувати, коли мова йде про контроль мобільних пристроїв, що належать співробітникам однієї компанії. Однак управляти сторонніми пристроями, наприклад за підтримкою мобільного банкінгу, набагато складніше навіть на організаційному рівні».

Централізоване керування мобільними пристроями – дуже важливе завдання. Використання такого встаткування при роботі з електронними документами «розмиває» захищений периметр організації й створює передумови

для витоку даних, що практично неможливо відстежити. Складність складається у величезній розмаїтості версій операційних систем, апаратних платформ і їхніх сполучень. Тобто розроблювачеві системи захисту мобільних пристроїв потрібно забезпечувати підтримку великої кількості версій операційних систем (Windows Phone, Android, iOS) і моделей смартфонів (Nokia, Samsung, iPhone, HTC, Blackberry і т.д.).

Є три підходи до рішення цієї проблеми:

– У першому випадку компанія централізовано здобуває й роздає співробітникам корпоративні мобільні пристрої (Choose Your Own Device, CYOD). Співробітники можуть користуватися тільки обмеженим спектром мобільних пристроїв із установленими на них рішеннями по захисту інформації. Деталізувавши правила користування такими пристроями в корпоративній політиці ІБ і настроївши на них систему моніторингу, можна значно знизити ризики.

– Другий варіант – замовлена розробка ПЗ під корпоративний стандарт окремого замовника.

– Третій – просто видати (без можливості вибору) мобільні пристрої із передвстановленим захисним ПЗ.

Як привести можливості пропонованих рішень у відповідність із реальними потребами замовників? Практика показує, що універсального рішення немає. У СЕД застосовуються типові сценарії, однак є й нюанси, і особливості. Це стосується бізнес-процесів, ИТ- і ІБ-інфраструктури, прийнятої в кожного окремого замовника, а також всіх мобільних пристроїв, використовуваних співробітниками.

Щоб корпоративні мобільні клієнти відповідали реальним, а не «стандартним» (передбачуваним виробником) потребам замовників, при впровадженні вони піддаються кастомизації для потреб електронного документообігу, що є досить розповсюдженою практикою. Найчастіше адаптується інтерфейс клієнта. Коннектори же для мобільних клієнтів від

виробників СЕД зберігаються й можуть підтримувати змінену логіку бізнес-процесів СЕД.

Крім того, існують універсальні мобільні клієнти, що потенційно підходять до будь-який СЕД і вже мають продуманий і зручний інтерфейс. У цьому випадку вони кастомизируються на технологічному рівні – у частині розробки коннекторів до певного СЕД.

Багатофункціональні мобільні АРМ досить затребувані. Класичні завдання документообігу, зведена аналітична звітність, надання даних для нарад, контроль виконання можуть бути об'єднані в одному мобільному автоматизованому робочому місці керівника. При цьому комплексне АРМ може бути реалізоване за допомогою або одного додатка, або їхнього набору. Консолідація різних функцій на базі мобільних клієнтів СЕД цілком логічна, оскільки в корпоративному секторі давно застосовуються повноцінні багатофункціональні мобільні клієнти СЕД.

Що стосується реальної потреби в мобільних СЕД серед різних груп користувачів (топ-менеджерів, керівників підрозділів і проектів, а також рядових співробітників), те все залежить від їхньої мобільності, поставлених завдань і структури самої організації. Чимале значення має й функціональність СЕД, яку необхідно використовувати віддалено.

Крім того, захищати потрібно не окремо взяту систему, а всю корпоративну інформаційну інфраструктуру. Проблему безпеки краще вирішувати на рівні архітектури підприємства, а не перевантажувати СЕД непрофільними для неї завданнями. Уважається також, що основні погрози для СЕД – внутрішні, зокрема витоку даних з вини допущених до документів співробітників. Це вимагає відповідних організаційних мір, реалізація яких часом виявляється ефективніше, ніж впровадження багатоступінчастих процедур по керуванню доступом або складними технічними рішеннями DLP.

Забезпечення безпеки інформації при зберіганні й обробці більших інформаційних масивів у захищеній системі документообігу, розгорнутої на мобільних пристроях – одна із самих актуальних проблем сучасних

інформаційних технологій. Інтенсивний розвиток методів розподіленої обробки даних і різке збільшення обсягів інформації, що накопичується в комп'ютерних системах, привели останнім часом до кардинальної зміни методів довгострокового зберігання даних. Традиційні підходи до організації зберігання великих інформаційних масивів перестали задовольняти зрослим вимогам до ємності носіїв і швидкості доступу до даних. Все частіше споживач довіряє зберігання своєї власної інформації зовнішнім центрам або мережам зберігання даних (так званий аутсорсинг). Одна з основних сфер застосування мережного зберігання даних – формування банків даних електронних документів, а також електронних архівів і бібліотек. Такі сховища даних можуть бути як публічними, так і обмеженого доступу, залежно від характеру документів, що накопичуються в них.

Для шифрування великих масивів даних захищеної системи документообігу, розгорнутої на мобільних пристроях, що поміщаються в зовнішні стосовно власника інформації сховища, ефективні лише симетричні схеми шифрування. Можливості їхнього практичного застосування, мабуть, визначаються можливостями організації схеми керування секретними ключами, для яких необхідно забезпечити виконання двох почасти суперечливих вимог: забезпечення високої схоронності ключів (зокрема, за рахунок резервування) і обмеження середовища їхнього поширення тільки тими пристроями, яким довіряє власник інформації. У зв'язку із цим у деяких випадках більше раціональним виглядає застосування схем відкритого шифрування, що дає можливість невизначеному колу осіб поміщати свої дані в сховище, але доступ до них залишати лише для власників секретного ключа. Така схема може бути корисна, наприклад, для систем електронної пошти або систем планування потоків завдань (workflows), де циркулюють переважно повідомлення невеликої довжини. Для таких схем тим більше необхідні механізми пошуку за шифрованим даними, що операції розшифрування в асиметричних криптосхемах, як відомо, виконуються на кілька порядків повільніше в порівнянні із симетричними.

Інша проблема, пов'язана із забезпеченням конфіденційності пошуку в масивах даних, пов'язана з бажанням унеможливити одержання адміністратором

СУБД і сторонніми особами відомостей про те, до яких саме записів (або фрагментів) бази даних здійснювався доступ при кожному конкретному запиті. У закордонній літературі це завдання зветься “Private Information Retrieval» (PIR). Вона особливо актуальна, наприклад, при обробці й зберіганні електронних документів, що містять відомості приватного характеру: фінансові, юридичні, майнові, медичні й інші.

Якщо навіть самі поля бази даних зашифровані, характер і частота запитів до них уже можуть дати зловмисникові деяку непряму інформацію, розголошення якої небажано для власника. Ці завдання до визначеної міри аналогічні виникаючої в телекомунікаційних системах завданню маскування інтенсивності трафіку між вузлами, що, як відомо, вирішується шляхом суцільного заповнення каналу псевдовипадковими послідовностями.

Нерідко перед приміщенням документів у мережні сховища вони піддаються стиску або іншим спеціальним видам кодування. У зв'язку із цим загострюється необхідність забезпечення керованості, надійності й безпеці зберігання й доступу до електронних документів, а також процедур їхньої передачі між прикладними програмами й пристроями зберігання.

Якщо навіть дані зберігаються локально, виникає інша проблема: адміністратори, що управляють СУБД і персонал так чи інакше звичайно мають права доступу до всієї збереженої інформації. Для захисту від їхніх несанкціонованих дій у деяких випадках доцільне застосування апаратно-програмних засобів шифрування даних перед записом їх на засоби зберігання.

Часто шифрувальні модулі вбудовуються, наприклад, у засоби резервного копіювання даних. Однак при зберіганні шифрованих масивів утруднений пошук окремих файлів і оперативний доступ до елементів масиву, необхідним для роботи прикладних програм. Так як масив зберігається в зашифрованому виді, і серверу, на якому він зберігається (або СУБД), не можуть бути довірені ключі шифрування, користувач (або прикладна програма від його ім'я) змушений завантажувати копії всіх файлів масиву, розшифровувати їх і потім виконувати пошук на локальній машині. Очевидно, що такий спосіб пошуку дуже

неекономічний. У зв'язку із цим вимальовується проблема забезпечення можливості пошуку даних по шифрованим і (або) стислим даним, що може бути конкретизована залежно від застосовуваної в системі моделі шифрування даних.

Для реалізації захищеної системи документообігу, розгорнутої на мобільних пристроях використовуються команди алгоритму шифрування AES. Вони складаються із:

– інструкцій для шифрування AES (AESENC, AESENCLAST);

– інструкцій для розшифровки AES (AESDEC, AESDECLAST);

– інструкції для роботи із ключем AES (AESIMC, AESKEYGENASSIST).

Підтримуються всі три ключі AES (128, 192 і 256 біт з 10, 12 і 14 проходами підстановки й перестановки). Оскільки всі інструкції AES мають фіксовану затримку, що не залежить від даних, тобто час фіксований й доступ до пам'яті не потрібно. Крім того, модель програмування така ж, як і у випадку інших інструкцій SSE з первісного стандарту SSE4. Таким чином, всі операційні системи, які підтримують роботу з SSE, зможуть використовувати й інструкції AES New Instructions.

На рисунку 3.1 зображена структурна схема захищеної системи документообігу, розгорнутої на мобільних пристроях. Виходячи зі структурної схеми системи зображеної на рисунку 3.1, захищена система документообігу, розгорнута на мобільних пристроях, працює наступним чином.

Спершу при вході в систему, користувач звертається до блоку розмежування доступу захищеної системи документообігу, розгорнутої на мобільних пристроях. Блок розмежування доступу отримує пароль користувача, та звертається до менеджеру паролів, де отримує сеансовий пароль перевірки правильності паролю користувача, та правильності прав доступу користувача, які зберігаються у відповідних зашифрованих базах даних. Розмежування цих баз зроблено з метою підвищення стійкості системи захищеного документообігу. Після підтвердження прав доступу, та правильності введеного паролю, користувачеві видається сеансовий ключ AES для роботи з інформацією.

У блоці шифрування згідно прав доступу, з отриманого ключа AES

відбувається його розширення, та обирається ключ ітерації, за допомогою яких й відбувається шифрування інформації алгоритмом AES.

Процедура дешифрування відбувається аналогічним чином.

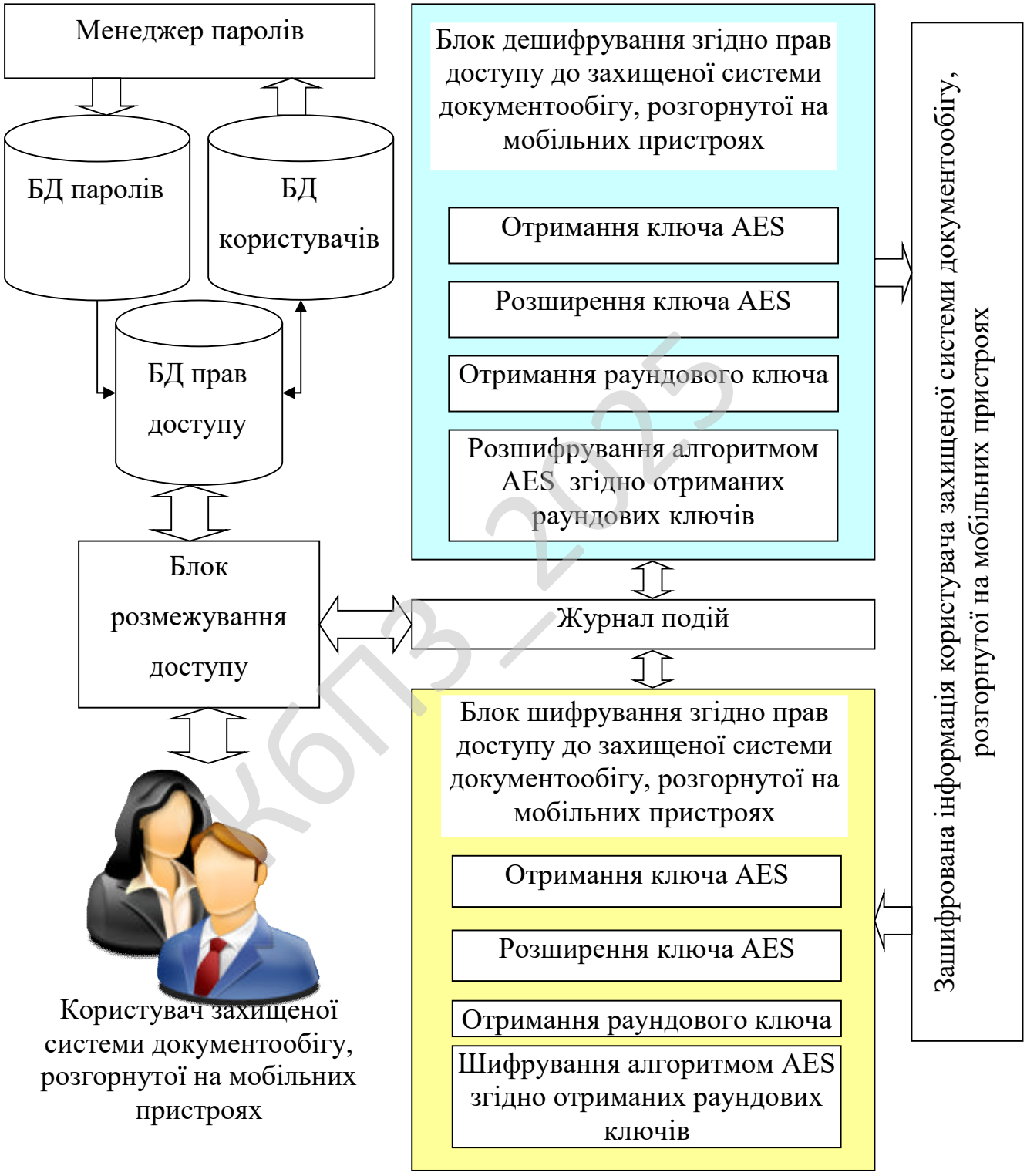

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема складається з наступних блоків:

– Головне вікно програми.

– Блоки шифрування та дешифрування інформації згідно алгоритму AES.

– Нормативні й розпорядницькі документи.

– Доручення.

– Клієнти й Контакти.

– Звернення громадян.

– Допомога.

– Планування.

– Керування Персоналом.

– Корпоративний тренінг.

– WorkFlow.

– Захист.

– Універсальне робоче місце.

– Мобільне робоче місце.

– Діловодство.

– Факс.

– Засідання.

– Договори.

Розглянемо ці блоки більш детально.

Головне вікно програми

Головне вікно призначене для швидкого доступу до основних функцій програми й меню. Програма складається з головного вікна, розташованого у верхній частині екрана й набору незалежних дочірніх вікон.

Розташування й розміри вікон можна змінювати за допомогою миші. Також існує можливість закрити непотрібні дочірні вікна (знову відобразити їх можна шляхом вибору відповідних пунктів у меню натисканням на аналогічні кнопки в головному вікні програми). Всі зроблені зміни збережуться в наступному сеансі роботи. Призначення всіх кнопок у програмі пояснюється спливаючими підказками: підведіть покажчик миші до будь-якої кнопки й затримаєте його – з'явиться спливаюча підказка із призначенням кнопки. Головне меню надає доступ до основних списків і функцій системи.

Блоки шифрування та дешифрування інформації згідно алгоритму AES

Призначені для шифрування та дешифрування інформації, до якої користувач має доступ, згідно прав доступу.

При реалізації шифрування та дешифрування виконуються такі основні операції:

– Key Expansion – процедура використовувана для генерації Round Keys з Cipher Key.

– Cipher Key – секретний, криптографічний ключ, що використовується Key Expansion процедурою, щоб зробити набір ключів для раундів (Round Keys); може бути представлений як прямокутний масив байтів, що має чотири рядки й Nk колонок.

– Round Key – Round Keys виходять із Cipher Key використовуючи процедуру Key Expansion. Вони застосовуються до State при шифруванні й розшифруванні.

– State – проміжний результат шифрування, що може бути представлений як прямокутний масив байтів що має 4 рядки й Nb колонок.

– AddRoundKey() – трансформація при шифруванні й зворотному шифруванні, при якій Round Key XOR’иться с State. Довжина RoundKey дорівнює розміру State (тобто, якщо Nb = 4, то довжина RoundKey дорівнює 128 біт або 16 байт).

– SubBytes() – трансформації при шифруванні які обробляють State використовуючи нелінійну таблицю заміщення байтів (S-box), застосовуючи її незалежно до кожного байта State.

– ShiftRows() – трансформації при шифруванні, які обробляють State, циклічно зміщаючи останні три рядки State на різні величини.

– MixColumns() – трансформація при шифруванні яка бере всі стовпці State і змішує їх дані (незалежно друг від друга), щоб одержати нові стовпці.

– InvShiftRows() – трансформація при розшифруванні яка є зворотною стосовно ShiftRows().

– InvSubBytes() – трансформація при розшифруванні яка є зворотною стосовно SubBytes().

– InvMixColumns() – трансформація при розшифруванні яка є зворотною стосовно MixColumns().

– RotWord() – функція, що використовується в процедурі Key Expansion, що бере 4-х байтне слово й робить над ним циклічну перестановку.

– SubWord() – функція, використовувана в процедурі Key Expansion, що бере на вході 4-х байтне слово й застосовуючи S-box до кожного із чотирьох байтів видає вихідне слово.

– Block – послідовність біт, з яких складається input, output, State і Round Key. Також під Block можна розуміти послідовність байт.

– Ciphertext – вихідні дані алгоритму шифрування.

– S-box – нелінійна таблиця замін, що використовується в декількох трансформаціях заміни байт і в процедурі Key Expansion для взаємнооднозначної заміни значення байта.

– Nb – число стовпців(32-ух бітних слів), що становлять State. Для, AES Nb = 4.

– Nk – число 32-ух бітних слів, що становлять шифроключ. Для AES, Nk = 4,6, або 8.

– Nr – число раундів, що є функцією Nk і Nb. Для AES, Nr = 10, 12, 14.

– Rcon[] – масив, що складається з бітів 32-х розрядного слова і є постійним для даного раунду.

Крім того функціонально «Захищена система документообігу, розгорнута на мобільних пристроях» включає системи:

– що автоматизують ведення діловодства й керування документообігом («Захищена система документообігу, розгорнута на мобільних пристроях – Діловодство», «Захищена система документообігу, розгорнута на мобільних пристроях – Документи», «Захищена система документообігу, розгорнута на мобільних пристроях – Інформаційно-довідкова система», «Захищена система документообігу, розгорнута на мобільних пристроях – Факс», «Захищена система документообігу, розгорнута на мобільних пристроях – Договори», «Захищена система документообігу, розгорнута на мобільних пристроях – Звернення громадян», «Захищена система документообігу, розгорнута на мобільних пристроях – Workflow»);

– що забезпечують інформаційну підтримку робочих процесів, властивих будь-який організації («Захищена система документообігу, розгорнута на мобільних пристроях – Клієнти й контакти», «Захищена система документообігу, розгорнута на мобільних пристроях – Планування», «Захищена система документообігу, розгорнута на мобільних пристроях – Засідання»);

– що автоматизують роботу кадрової служби («Захищена система документообігу, розгорнута на мобільних пристроях – Керування персоналом»);

– що автоматизують взаємодію користувачів з відділом технічної підтримки (система «Захищена система документообігу, розгорнута на мобільних пристроях – HelpDesk»);

– що забезпечують можливість навчання роботі в системі «Захищена система документообігу, розгорнута на мобільних пристроях» («Захищена система документообігу, розгорнута на мобільних пристроях – Корпоративний тренінг»).

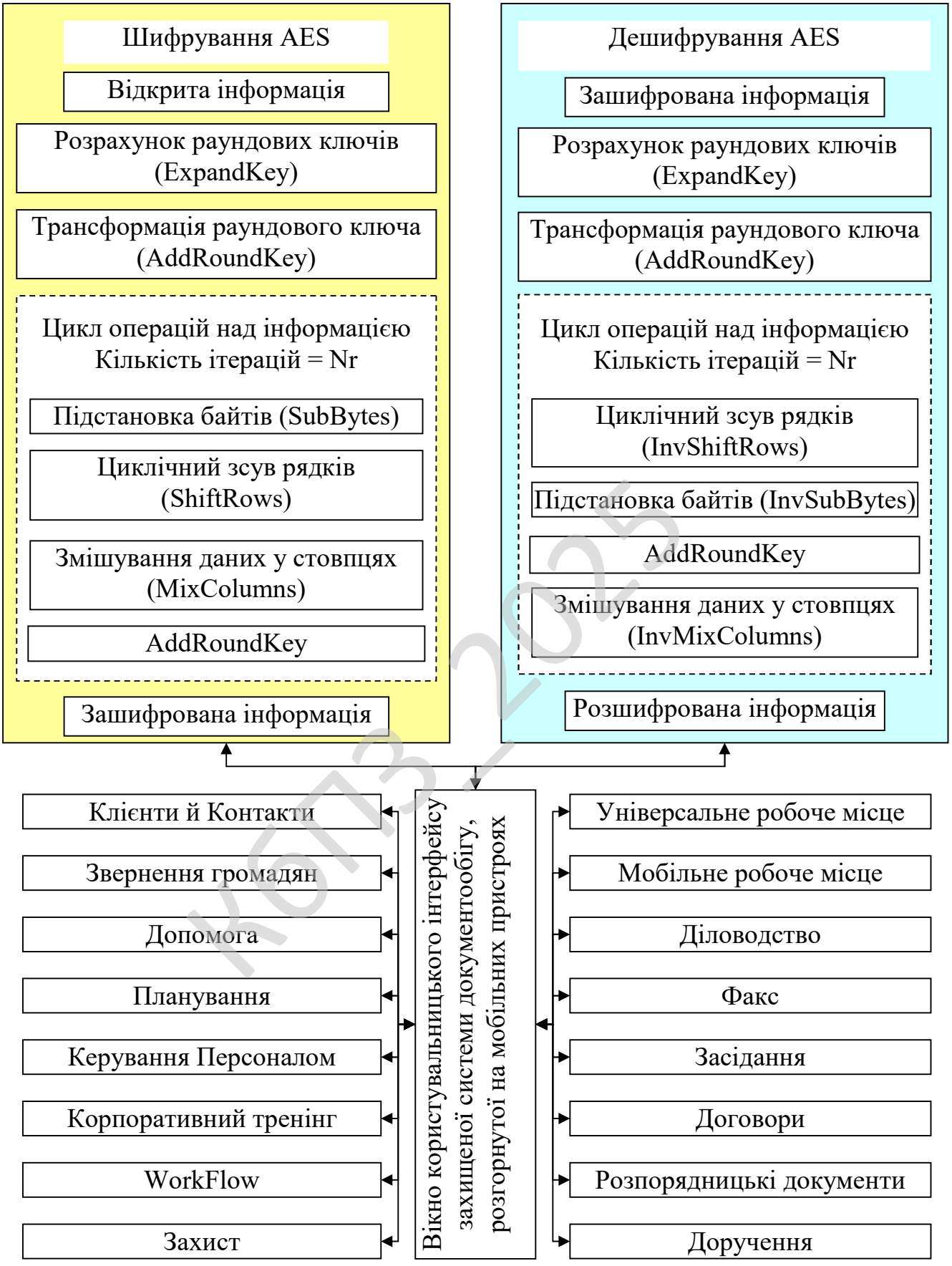

Рисунок 3.2 – Функціональна схема системи

Захищена система документообігу, розгорнута на мобільних пристроях – Факс

Система факс-серверної служби організації в складі системи корпоративного документообігу й діловодства «Захищена система документообігу, розгорнута на мобільних пристроях» істотно підвищує ефективність і зручність роботи з факсимільними повідомленнями.

Захищена система документообігу, розгорнута на мобільних пристроях – Засідання

Підсистема призначена для автоматизації процесу документообігу, що супроводжує проведення засідань і нарад колегіальних органів керування організації.

Захищена система документообігу, розгорнута на мобільних пристроях – Договори

Система призначена для ведення реєстру договорів і контролю виконання стосовних до них доручень.

Захищена система документообігу, розгорнута на мобільних пристроях – Нормативні й розпорядницькі документи

Система «Захищена система документообігу, розгорнута на мобільних пристроях – Нормативно-розпорядницькі документи» призначена для публікації й зберігання офіційних діючих і застарілих нормативних, а також організаційно-розпорядницьких документів (уставів, положень, наказів, правил, інструкцій і т.д.), призначених для використання в інформаційно-довідкових цілях.

Захищена система документообігу, розгорнута на мобільних пристроях – Доручення

Підсистема «Захищена система документообігу, розгорнута на мобільних пристроях – Доручення» призначена для створення й контролю виконання усних доручень, не пов'язаних з документами.

Захищена система документообігу, розгорнута на мобільних пристроях – Клієнти й Контакти

«Захищена система документообігу, розгорнута на мобільних пристроях – Клієнти й контакти» дозволяє впорядкувати всю інформацію про клієнтів і дає можливість зберігати інформацію із всіх зовнішніх організацій, з якими коли-або контактувала компанія.

Захищена система документообігу, розгорнута на мобільних пристроях – Звернення громадян

Система орієнтована на підприємства й організації, що ведуть діловодство на підставі звернень громадян і дозволяє скоротити часові витрати на реєстрацію, обробку й контроль виконання заявок фізичних осіб.

Захищена система документообігу, розгорнута на мобільних пристроях – Допомога

Система призначена для автоматизації процесу рішення технічних проблем, які виникають у користувачів компанії при експлуатації технічного й програмного забезпечення, для ведення архіву усунутих проблем з метою їхнього подальшого аналізу, для узагальнення й використання результатів при виникненні аналогічних ситуацій, для обліку завантаження співробітників по проектах, для короткострокового планування роботи на майбутній період.

Захищена система документообігу, розгорнута на мобільних пристроях – Планування

Система призначена для планування й координації робіт, для здійснення контролю за виконанням поставлених планів, для нагромадження інформації про успішні й провалені проекти з метою їхній наступного аналізу. Система дозволяє підвищити ефективність управлінської діяльності компанії.

Захищена система документообігу, розгорнута на мобільних пристроях – Керування Персоналом

Система призначена для автоматизації виробничих процесів, пов'язаних з кадровим обліком і кадровим документообігом.

Захищена система документообігу, розгорнута на мобільних пристроях – Корпоративний тренінг

«Захищена система документообігу, розгорнута на мобільних пристроях – Корпоративний тренінг» – це програмно-методичний комплекс навчання й атестації користувачів і інструкторів системи «Захищена система документообігу, розгорнута на мобільних пристроях».

Захищена система документообігу, розгорнута на мобільних пристроях – WorkFlow

Термін workflow дослівно означає «потік робіт». Однак технологія workflow розглядається набагато ширше – це автоматизація робітників бізнес-процесів. Бізнес-процес, по суті справи, поєднує в собі все: потік робіт і функції, людей і встаткування, що реалізує ці функції, а також правила, керуючі послідовністю цих функцій.

Захищена система документообігу, розгорнута на мобільних пристроях – Захист

Програмний продукт Захист призначений для підключення зовнішніх засобів криптозахисту інформації.

Захищена система документообігу, розгорнута на мобільних пристроях – Центр Звітів

Система «Центр звітів» призначена для створення звітів за даними, що зберігається в базах даних. За допомогою цієї системи можна створювати довільно оформлені звіти, що дозволяє швидко одержати інформацію в зручній для користувача формі. Можливість створення власних звітів дозволяє враховувати особливості роботи будь-якої організації.

Захищена система документообігу, розгорнута на мобільних пристроях – Універсальне робоче місце

«Захищена система документообігу, розгорнута на мобільних пристроях – Універсальне робоче місце» – це єдиний інтерфейс для всіх співробітників

організації, що беруть участь в електронному документообігу: більш зручно ніж файлова система, і більш надійно ніж електронна пошта.

Захищена система документообігу, розгорнута на мобільних пристроях – Мобільне робоче місце

«Захищена система документообігу, розгорнута на мобільних пристроях – Мобільний портал» – робоче місце, що дозволяє здійснювати мобільний доступ до баз дані системи електронного документообігу «Захищена система документообігу, розгорнута на мобільних пристроях».

Захищена система документообігу, розгорнута на мобільних пристроях – Діловодство

Модуль «Захищена система документообігу, розгорнута на мобільних пристроях – Діловодство» призначений для автоматизації документообігу в територіально-розподілених організаціях, технологія діловодства яких припускає централізоване відстеження руху документів у реальному масштабі часу.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврської дипломної роботи, наведена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі. Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

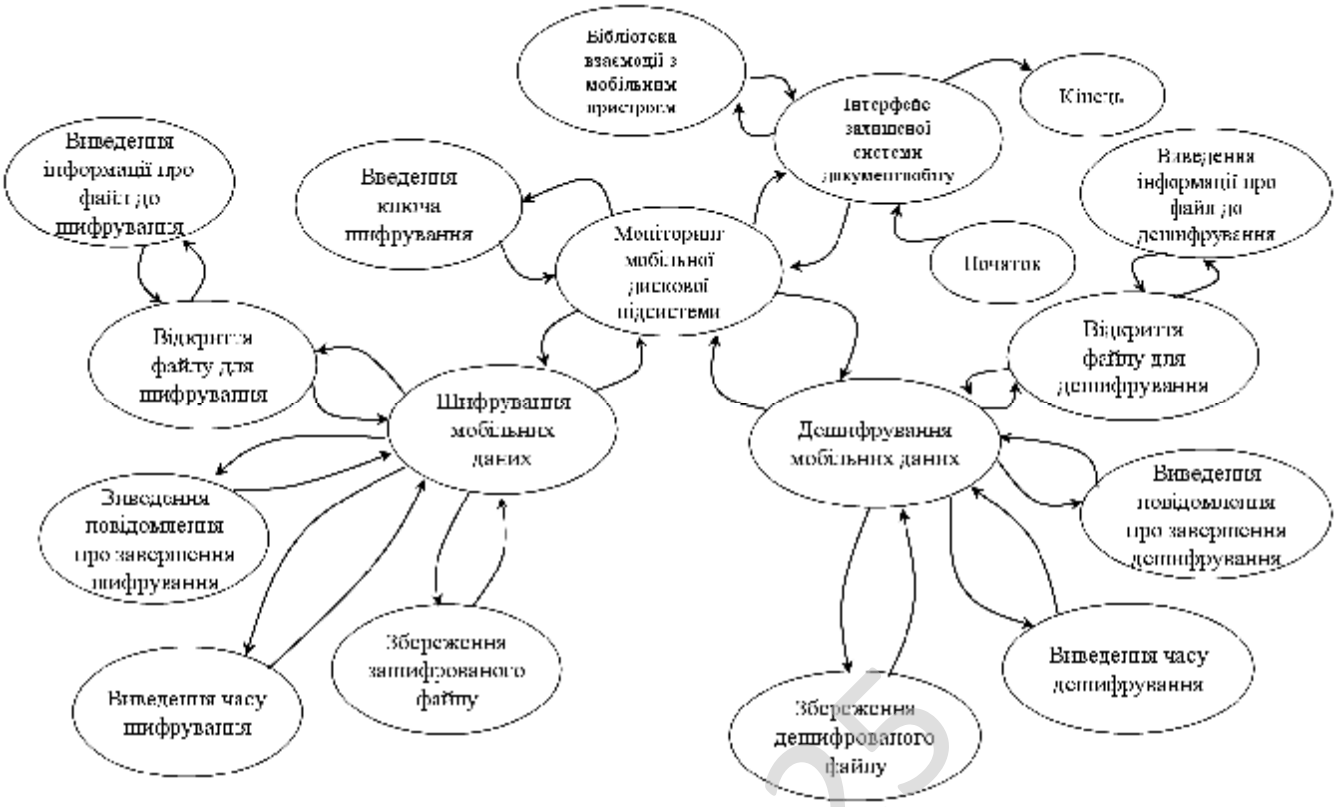

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю захищеної системи документообігу, розгорнутої на мобільних пристроях.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми. З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

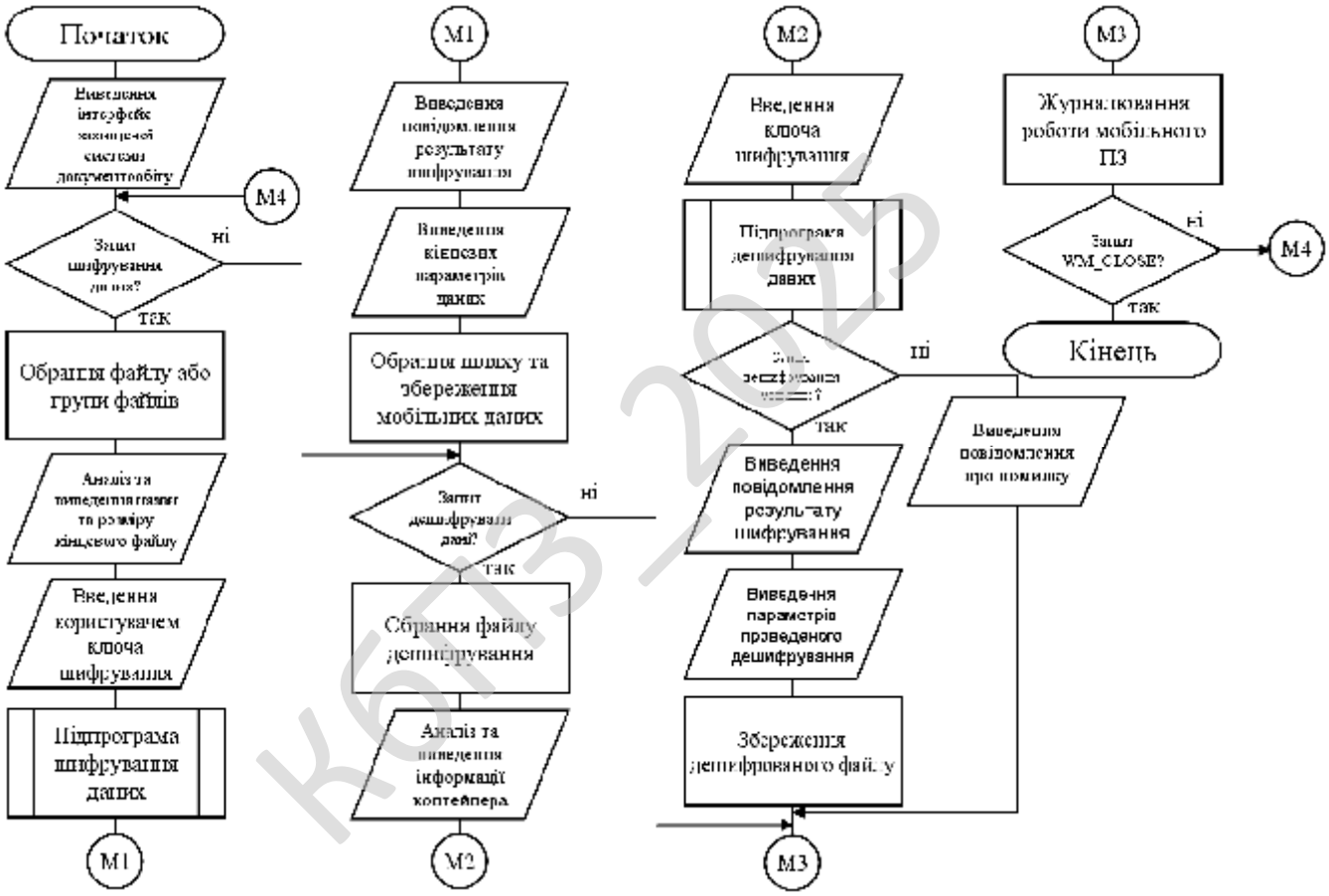

Рисунок 4.1 – Блок-схема основної програми

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

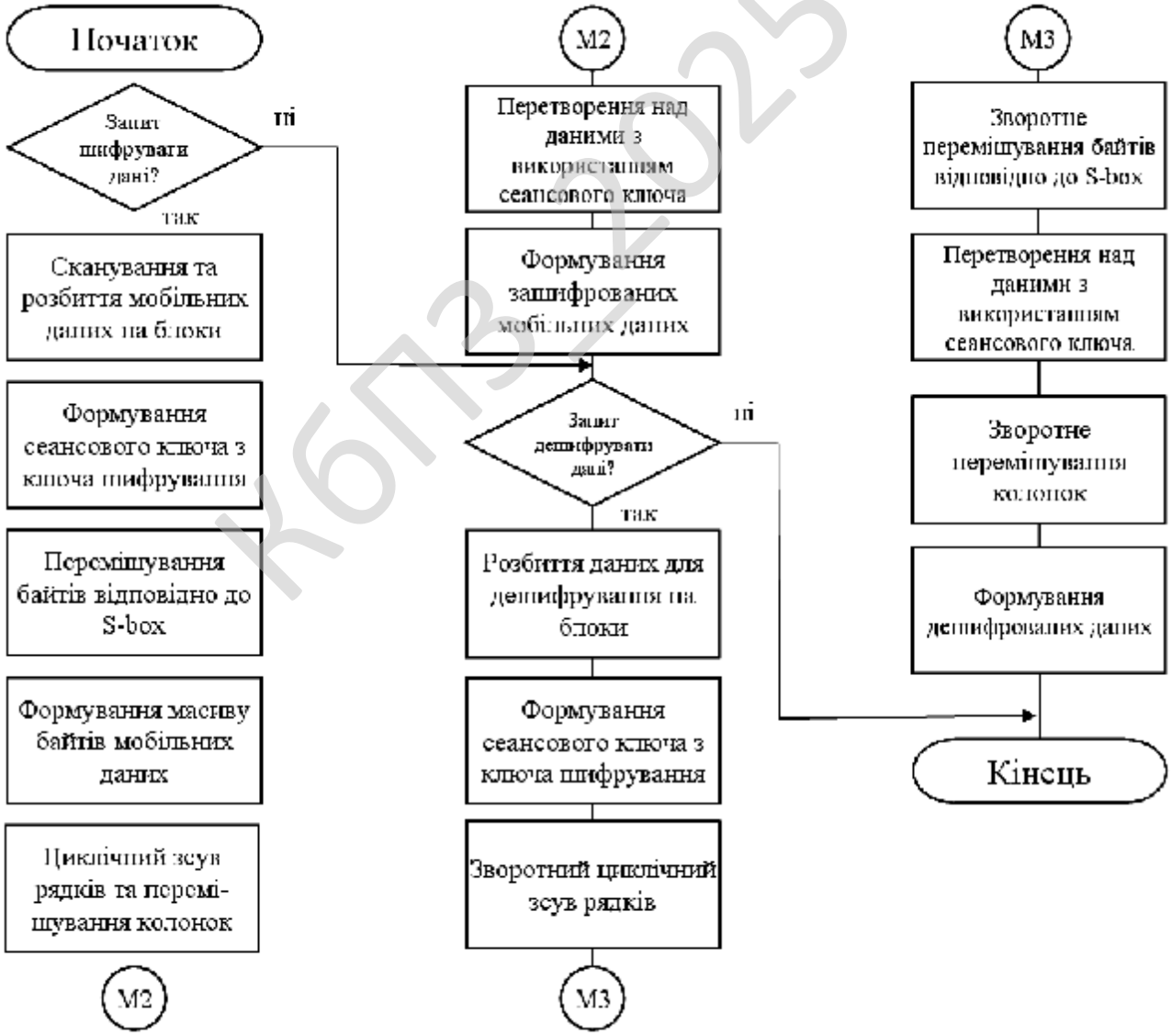

Рисунок 4.2 – Блок-схема роботи підпрограми

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Система захищеного документообігу для мобільних пристроїв призначається для забезпечення безпечного зберігання, передачі та управління документами. Вона включає серверну частину для централізованого управління та мобільний клієнт для взаємодії з користувачем.

Архітектура системи базується на клієнт-серверній моделі, де сервер виконує функції обробки запитів, а мобільні пристрої є клієнтами. Серверна частина розгортається з використанням Python та фреймворку Flask для організації веб-сервісу. Модуль SQLAlchemy забезпечує роботу з базою даних. Аутентифікація та шифрування реалізуються за допомогою бібліотек bcrypt та cryptography.

Клієнтська частина представлена мобільним додатком, розробленим з використанням Python та Kivy. Ця частина дозволяє користувачам переглядати документи, здійснювати пошук, надсилати запити на сервер та отримувати відповіді. Використання асинхронних запитів на основі модуля asyncio забезпечує швидку та безпечну взаємодію з сервером.

Основні модулі системи включають:

– Серверна частина з обробкою запитів на автентифікацію, управління доступом до документів та логуванням активності користувачів.

– Клієнтська частина, що містить графічний інтерфейс користувача та механізм зв'язку з сервером.

– Модуль безпеки для шифрування файлів і контроль доступу.

– Модуль бази даних для зберігання користувацьких даних, прав доступу та логів.

Функціональні можливості системи охоплюють авторизацію користувачів, управління ролями, захищену передачу файлів та перегляд документів у зашифрованому вигляді. Використання алгоритму AES-256 гарантує високий рівень безпеки. Перевірка автентичності реалізується за допомогою токенів JWT.

Розрахунки продуктивності системи включають аналіз часу відповіді сервера, середнє навантаження на процесор та використання пам'яті.

У середньому, передача документа розміром 1 МБ займає 0.2 секунди при використанні шифрування. Підтримка одночасної роботи до 100 клієнтів без втрати продуктивності забезпечується оптимізацією запитів та кешуванням.

Застосування цієї системи у навчальному процесі підвищує рівень безпеки документів, зменшує ризики несанкціонованого доступу та забезпечує ефективне управління електронними матеріалами.

Серверна частина (частина коду):

```
from flask import Flask, request, jsonify, send_file
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash
import jwt
import datetime
```

```
import os
from cryptography.fernet import Fernet

# Ініціалізація сервера
app = Flask(__name__)
app.config['SECRET_KEY'] = 'your_secret_key'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
db = SQLAlchemy(app)

# Генерація ключа шифрування
if not os.path.exists("secret.key"):
    key = Fernet.generate_key()
    with open("secret.key", "wb") as key_file:
        key_file.write(key)
else:
    with open("secret.key", "rb") as key_file:
        key = key_file.read()

cipher = Fernet(key)

# Модель користувача
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), unique=True, nullable=False)
    password = db.Column(db.String(100), nullable=False)

# Модель документів
class Document(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    filename = db.Column(db.String(100), nullable=False)
    content = db.Column(db.LargeBinary, nullable=False)

# Реєстрація користувача
@app.route('/register', methods=['POST'])
def register():
    data = request.get_json()
    hashed_password = generate_password_hash(data['password'])
    new_user = User(username=data['username'], password=hashed_password)
    db.session.add(new_user)
    db.session.commit()
    return jsonify({'message': 'User registered successfully'})
```

```
        # Авторизація користувача
        @app.route('/login', methods=['POST'])
        def login():
            data = request.get_json()
            user = User.query.filter_by(username=data['username']).first()
            if user and check_password_hash(user.password, data['password']):
                token = jwt.encode({'user': user.username, 'exp':
datetime.datetime.utcnow() + datetime.timedelta(hours=1)},
app.config['SECRET_KEY'])
                return jsonify({'token': token})
            return jsonify({'message': 'Invalid credentials'}), 401

        # Завантаження документа
        @app.route('/upload', methods=['POST'])
        def upload():
            token = request.headers.get('Authorization')
            try:
                jwt.decode(token, app.config['SECRET_KEY'], algorithms=["HS256"])
            except:
                return jsonify({'message': 'Unauthorized'}), 401

            file = request.files['file']
            encrypted_content = cipher.encrypt(file.read())
            new_document = Document(filename=file.filename,
content=encrypted_content)
            db.session.add(new_document)
            db.session.commit()
            return jsonify({'message': 'File uploaded successfully'})

        # Завантаження документа користувачем
        @app.route('/download/<int:doc_id>', methods=['GET'])
        def download(doc_id):
            token = request.headers.get('Authorization')
            try:
                jwt.decode(token, app.config['SECRET_KEY'], algorithms=["HS256"])
            except:
                return jsonify({'message': 'Unauthorized'}), 401

            document = Document.query.get(doc_id)
            if document:
                decrypted_content = cipher.decrypt(document.content)
                with open(document.filename, "wb") as file:
```

```
                file.write(decrypted_content)
            return send_file(document.filename, as_attachment=True)
        return jsonify({'message': 'Document not found'}), 404

    # Запуск сервера
    if __name__ == '__main__':
        db.create_all()
        app.run(debug=True)
```

Частина коду, клієнтська частина:

```
    import requests

    # URL сервера
    SERVER_URL = "http://127.0.0.1:5000"

    # Реєстрація користувача
    def register(username, password):
        response = requests.post(f"{SERVER_URL}/register", json={'username':
username, 'password': password})
        return response.json()

    # Авторизація користувача
    def login(username, password):
        response = requests.post(f"{SERVER_URL}/login", json={'username':
username, 'password': password})
        return response.json().get('token')

    # Завантаження файлу
    def upload_file(token, filepath):
        headers = {'Authorization': token}
        files = {'file': open(filepath, 'rb')}
        response = requests.post(f"{SERVER_URL}/upload", headers=headers,
files=files)
        return response.json()

    # Завантаження файлу з сервера
    def download_file(token, doc_id):
        headers = {'Authorization': token}
        response = requests.get(f"{SERVER_URL}/download/{doc_id}",
headers=headers)
        if response.status_code == 200:
            with open("downloaded_file", "wb") as file:
```

```
                file.write(response.content)
            return "File downloaded successfully"
        return response.json()

    # Тестування функцій
    if __name__ == '__main__':
        username = "testuser"
        password = "testpassword"
        print(register(username, password))
        token = login(username, password)
        print(f"Token: {token}")

        if token:
            print(upload_file(token, "test_document.txt"))
            print(download_file(token, 1))
```

Система складається з двох основних частин – серверної та клієнтської. Сервер використовує Flask для обробки HTTP-запитів, а база даних зберігає користувачів та зашифровані документи. Клієнт взаємодіє з сервером через API-запити.

Основні модулі:

– flask використовується для створення веб-сервера.

– flask_sqlalchemy застосовується для роботи з базою даних SQLite.

– werkzeug.security забезпечує безпеку паролів.

– jwt реалізує автентифікацію за допомогою токенів.

– cryptography.fernet виконує шифрування файлів.

Алгоритм роботи:

1. Користувач реєструється, а його пароль хешується та зберігається у базі

2. Під час авторизації користувач отримує JWT-токен.

3. Користувач завантажує файл, який шифрується та зберігається у базі.

4. При запиті на завантаження сервер розшифровує файл та надсилає його клієнту.

5. Система забезпечує безпеку завдяки використанню шифрування, аутентифікації токенами та хешування паролів.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою IDEA – симетричний блоковий алгоритм шифрування даних, запатентований швейцарською фірмою Ascom. Відомий тим, що застосовувався в пакеті програм шифрування PGP. У листопаді 2000 року IDEA був представлений як кандидат у проекті NESSIE в рамках програми Європейської комісії IST (англ. Information Societes Technology, інформаційні громадські технології).

Першу версію алгоритму розробили в 1990 році Лай Сюецзя (Xuejia Lai) і Джеймс Мессі (James Massey) зі Швейцарського інституту ETH Zürich (за контрактом з Hasler Foundation, яка пізніше влилася в Ascom-Tech AG) як заміна DES (англ. Data Encryption Standard, стандарт шифрування даних) і назвали її PES (англ. Proposed Encryption Standard, запропонований стандарт шифрування). Потім, після публікації робіт Біхамом і Шаміра по диференціальному криптоаналізу PES, алгоритм був поліпшений з метою посилення криптостійкості і названий IPES (англ. Improved Proposed Encryption Standard, покращений запропонований стандарт шифрування). Через рік його перейменували в IDEA (англ. International Data Encryption Algorythm).

Так як IDEA використовує 128-бітний ключ і 64-бітний розмір блоку, відкритий текст розбивається на блоки по 64 біт. Якщо таке розбиття неможливо, останній блок доповнюється різними способами певною послідовністю біт. Для уникнення витоку інформації про кожному окремому блоці використовуються різні режими шифрування. Кожен вихідний незашифрований 64 – біт ний блок ділиться на чотири підблока по 16 біт кожен, так як всі алгебраїчні операції, що використовуються в процесі шифрування, відбуваються над 16-бітними числами. Для шифрування і розшифрування IDEA використовує один і той же алгоритм.

Позначення операцій:

- $\boxplus$ Додавання за модулем 2^{16}.
- $\odot$ Множення за модулем $2^{16}+1$.

– $\oplus$ Побітова виключна диз'юнкція.

Фундаментальним нововведенням в алгоритмі є використання операцій з різних алгебраїчних груп, а саме:

Додавання за модулем 2^{16}.

Множення за модулем $2^{16}+1$.

Побітова виключна диз'юнкція (XOR).

Ці три операції несумісні в тому сенсі, що ніякі дві з них не задовольняють дистрибутивному закону, тобто:

$$a \odot (b \oplus c) \neq (a \odot b) \oplus (a \odot c).$$

Застосування цих трьох операцій ускладнює криптоаналіз IDEA в порівнянні з DES, який базується виключно на операції виключає АБО, а також дозволяє відмовитися від використання S-блоків і таблиць заміни. IDEA є модифікацією мережі Фейстеля.

Генерація ключів

З 128-бітного ключа для кожного з восьми раундів шифрування генерується по шість 16-бітних подключів, а для вихідного перетворення генерується чотири 16-бітних підключа. Всього буде потрібно 52 = 8 x 6 + 4 різних подключів по 16 біт кожен. Процес генерації п'ятдесяти двох 16-бітних ключів полягає в наступному:

Насамперед, 128-бітний ключ розбивається на вісім 16-бітних блоків. Це будуть перші вісім подключів по 16 біт кожен – $(K_1^{(1)}K_2^{(1)}K_3^{(1)}K_4^{(1)}K_5^{(1)}K_6^{(1)}K_1^{(2)}K_2^{(2)})$

Потім цей 128-бітний ключ циклічно зсувається вліво на 25 позицій, після чого новий 128-бітний блок знову розбивається на вісім 16-бітних блоків. Це вже наступні вісім подключів по 16 біт кожен – $(K_3^{(2)}K_4^{(2)}K_5^{(2)}K_6^{(2)}K_1^{(3)}K_2^{(3)}K_3^{(3)}K_4^{(3)})$

Процедура циклічного зсуву і розбивки на блоки триває до тих пір, поки не будуть згенеровані всі 52 16-бітних підключа.

Шифрування

Структура алгоритму IDEA показана на рисунку 4.3.

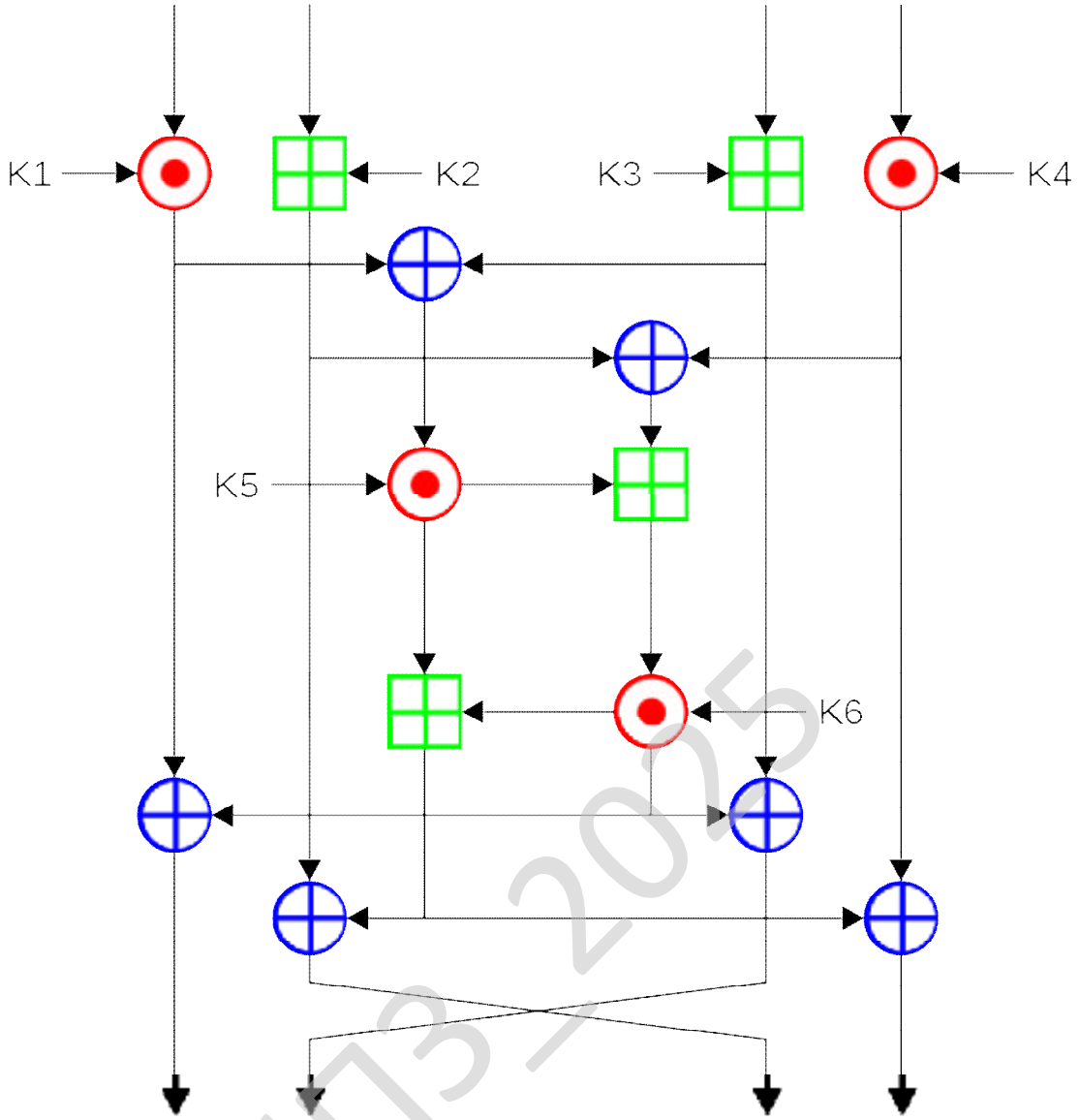

Рисунок 4.3 – Структура алгоритму IDEA

Процес шифрування складається з восьми однакових раундів шифрування і одного вихідного перетворення. Вихідний незашифрований текст ділиться на блоки по 64 біта. Кожен такий блок ділиться на чотири підблока по 16 біт кожен. На рисунку ці підблоки позначені D_1, D_2, D_3, D_4. У кожному раунді використовуються свої підключі згідно з таблицею підключів. Над 16-бітними підключами і підблока незашифрованого тексту проводяться наступні операції:

– Множення за модулем $2^{16}+1 = 65537$, причому замість нуля використовується 2^{16}.

– Додавання за модулем 2^{16}.

– Побітове виключне АБО.

В кінці кожного раунду шифрування є чотири 16-бітних підблоки, які потім використовуються як вхідні підблоки для наступного раунду шифрування. Вихідна перетворення являє собою скорочений раунд, а саме, чотири 16-бітних підблоки на виході восьмого раунду і чотири відповідних підключа піддаються операціям:

– Множення за модулем $2^{16}+1$.

– Додавання за модулем 2^{16}.

Після виконання вихідного перетворення конкатенація підблоків D_1', D_2', D_3' і D_4' являє собою зашифрований текст. Потім береться наступний 64-бітний блок незашифрованого тексту і алгоритм шифрування повторюється. Так продовжується до тих пір, поки не зашифрують всі 64-бітові блоки вихідного тексту.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи.

Розроблене програмне забезпечення захищеної системи документообігу, розгорнутої на мобільних пристроях складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Шаблон; Налаштування; Довідка.

– Вікно розподілено на чотири підрозділи, а саме: зв'язок з мобільним пристроєм; шифрування/дешифрування; функції; поточні дії програми.

– Вікно виведення результату роботи ПЗ.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

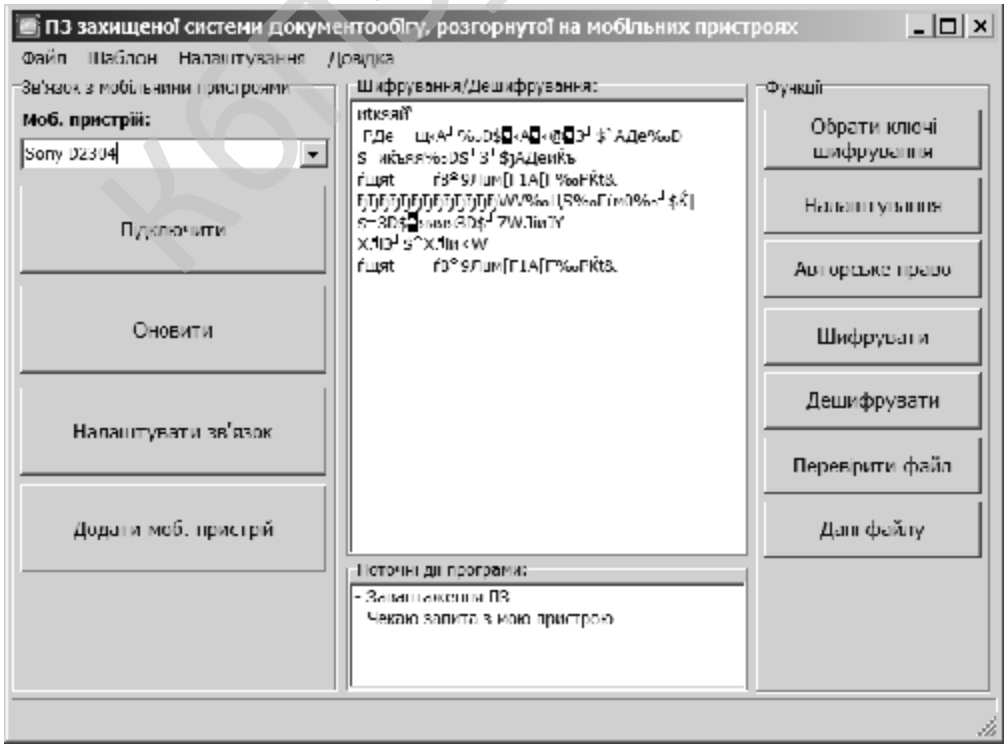

Рисунок 5.1 – Головне вікно розробленого ПЗ (серверна частина)

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

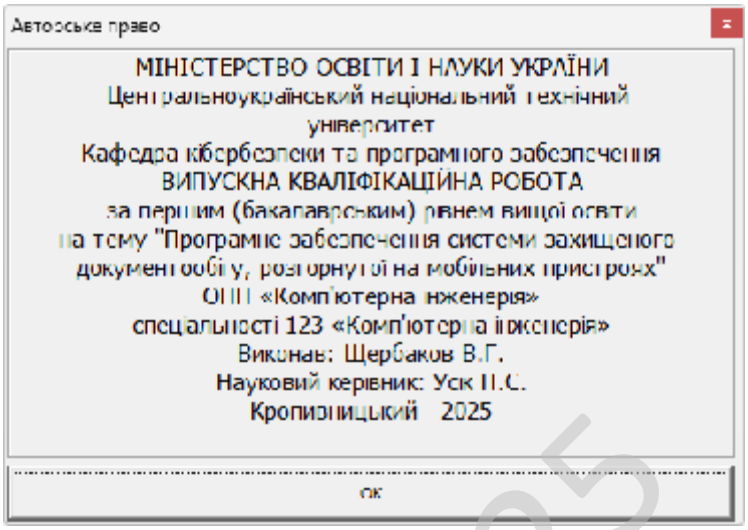

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи захищеного документообігу, розгорнутої на мобільних пристроях.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем захищеного документообігу, розгорнутої на мобільних пристроях.

– Досліджена система захищеного документообігу, розгорнутої на мобільних пристроях.

– На основі отриманих результатів досліджень створена програмна реалізація системи захищеного документообігу, розгорнутої на мобільних пристроях.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання захищеного документообігу, розгорнутої на мобільних пристроях.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи захищеного документообігу, розгорнутої на мобільних пристроях. Це

дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Android.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм IDEA.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

2. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

3. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

4. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

5. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

6. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

7. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnova, T., Prokopov, S., Bilanovych, A. «New Cost Function for S-boxes Generation by Simulated Annealing Algorithm». Lecture Notes on Data Engineering and Communications Technologies, 2023. vol 180. pp. 310-320. Springer, Cham.

8. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnov, O., Ulianovska, Y., Kobylianska, O. «Heuristic Search for Nonlinear Substitutions for Cryptographic

Applications». *Lecture Notes on Data Engineering and Communications Technologies,* 2023. vol 180. Springer, Cham. pp. 288-298.

9. Kuznetsov, O., Kuznetsova, Y., Smirnov, O., Kostenko, O., Zvieriev, V. «Evaluating Hashing Algorithms in the Age of ASIC Resistance». *CEUR Workshop Proceedings*, 2023, 3628, pp. 93-105.

10. Kuznetsov O., Frontoni E., Kuznetsova Ye., Smirnov O., Chevardin V. «Achieving Enhanced Security in Biometric Authentication: A Rigorous Analysis of Code-Based Fuzzy Extractor». *CEUR Workshop Proceedings,* Volume 3624, 2023, pp. 330-339.

11. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

12. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

13. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

14. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

15. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

16. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

17. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

18. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

19. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

20. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

21. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

22. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

23. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

24. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

25. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

26. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

27. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

28. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

29. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

30. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

31. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

32. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

33. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

34. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

35. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

36. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

37. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

38. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

39. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

40. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

41. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

42. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

43. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

44. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in

Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

45. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

46. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

47. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

48. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

49. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

50. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи захищеного документообігу, розгорнутої на мобільних пристроях.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 46-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи захищеного документообігу, розгорнутої на мобільних пристроях.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи захищеного документообігу, розгорнутої на мобільних пристроях;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на мобільних пристроях під керуванням ОС Android і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Android.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 69 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 2.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Усік П.С.

Програмне забезпечення системи захищеного документообігу, розгорнутої на мобільних пристроях

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 24

Літера: РП

Кропивницький – 2025 року

Основна програма

```
import os
import base64
import hashlib
import json
import sqlite3
import random
import string
import time
import zipfile
import qrcode
import shutil
import dropbox
import smtplib
import face_recognition
import matplotlib.pyplot as plt
from datetime import datetime
from flask import Flask, request, jsonify, session
from flask_session import Session
from cryptography.fernet import Fernet
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from deep_translator import GoogleTranslator
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans

# Генеруємо ключ для шифрування файлів
def generate_key():
    key = Fernet.generate_key()
    with open("secret.key", "wb") as key_file:
        key_file.write(key)

# Завантажуємо ключ шифрування
def load_key():
    return open("secret.key", "rb").read()

# Функція для створення хешу пароля
def hash_password(password):
    salt = os.urandom(32)
    key = hashlib.pbkdf2_hmac('sha256', password.encode(), salt, 100000)
    return salt + key

# Функція перевірки пароля
def verify_password(stored_password, provided_password):
    salt = stored_password[:32]
    key = stored_password[32:]
    new_key = hashlib.pbkdf2_hmac('sha256', provided_password.encode(), salt,
100000)
    return key == new_key

# Функція генерації 2FA-коду
def generate_2fa_code():
    return "".join(str(random.randint(0, 9)) for _ in range(6))

# Функція надсилання коду через email
def send_2fa_email(email, code):
    server = smtplib.SMTP("smtp.example.com", 587)
    server.starttls()
    server.login("noreply@example.com", "password")
    message = f"Subject: Your 2FA Code\n\nYour code is: {code}"
    server.sendmail("noreply@example.com", email, message)
    server.quit()

# Клас користувача
class User:
    def __init__(self, username, password):
        self.username = username
        self.password = hash_password(password)
```

```
# База даних
class Database:
    def __init__(self):
        self.conn = sqlite3.connect("secure_docs.db", check_same_thread=False)
        self.cursor = self.conn.cursor()
        self.create_tables()

    def create_tables(self):
        self.cursor.execute('''CREATE TABLE IF NOT EXISTS users
                               (id INTEGER PRIMARY KEY, username TEXT, password
BLOB)''')
        self.cursor.execute('''CREATE TABLE IF NOT EXISTS documents
                               (id INTEGER PRIMARY KEY, user_id INTEGER, filename
TEXT, data BLOB, timestamp TEXT)''')
        self.conn.commit()

# Запуск сервера
app = Flask(__name__)
app.config["SESSION_TYPE"] = "filesystem"
Session(app)
db = Database()

@app.route("/register", methods=["POST"])
def register():
    data = request.json
    username = data.get("username")
    password = data.get("password")
    db.cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)",
(username, hash_password(password)))
    db.conn.commit()
    return jsonify({"message": "User registered successfully"})

@app.route("/login", methods=["POST"])
def login():
    data = request.json
    username = data.get("username")
    password = data.get("password")
    db.cursor.execute("SELECT * FROM users WHERE username=?", (username,))
    user = db.cursor.fetchone()
    if not user:
        return jsonify({"message": "User not found"})
    if not verify_password(user[2], password):
        return jsonify({"message": "Invalid password"})
    session["user"] = username
    return jsonify({"message": "Login successful"})

@app.route("/logout", methods=["POST"])
def logout():
    session.pop("user", None)
    return jsonify({"message": "Logged out"})

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=5000)
```

Файл two_factor_auth.py

```
import random
import smtplib
import time

# Функція генерації 6-значного коду
def generate_2fa_code():
code = "".join(str(random.randint(0, 9)) for _ in range(6))
return code

# Функція надсилання коду через email
def send_2fa_email(email, code):
server = smtplib.SMTP("smtp.example.com", 587)
server.starttls()
server.login("noreply@example.com", "password")
message = f"Subject: Your 2FA Code\n\nYour code is: {code}"
server.sendmail("noreply@example.com", email, message)
server.quit()

# Використання 2FA-коду
def two_factor_auth(email):
code = generate_2fa_code()
send_2fa_email(email, code)
print("2FA code sent to your email.")
time.sleep(5)  # Очікування перед введенням
entered_code = input("Enter the 2FA code: ")
return entered_code == code
```

Файл roles_permissions.py

```
class RoleManager:
def __init__(self):
self.roles = {
"admin": {"read": True, "write": True, "delete": True},
"user": {"read": True, "write": True, "delete": False},
"moderator": {"read": True, "write": True, "delete": True},
}

# Функція отримання прав ролі
def get_permissions(self, role):
return self.roles.get(role, {"read": False, "write": False, "delete": False})

# Функція перевірки доступу
def has_permission(self, role, action):
permissions = self.get_permissions(role)
return permissions.get(action, False)

# Приклад використання
manager = RoleManager()
print(manager.has_permission("admin", "delete"))  # True
print(manager.has_permission("user", "delete"))  # False
```

Файл test_data_generator.py

```
import random
import string
import os

# Генерація випадкових імен користувачів
def generate_random_username():
return "".join(random.choices(string.ascii_lowercase, k=8))

# Генерація масиву користувачів
def generate_users(count=10):
users = [generate_random_username() for _ in range(count)]
return users

# Генерація тестових документів
def generate_test_documents(folder="test_docs", count=10):
os.makedirs(folder, exist_ok=True)
for i in range(count):
filename = os.path.join(folder, f"document_{i}.txt")
with open(filename, "w") as file:
file.write(f"Test content for document {i}")
print(f"Generated: {filename}")

generate_test_documents()
print(generate_users(5))
```

Файл bulk_upload.py

```
import os
import zipfile

# Функція розпакування ZIP-архіву
def unzip_files(zip_path, extract_to="extracted_files"):
os.makedirs(extract_to, exist_ok=True)
with zipfile.ZipFile(zip_path, "r") as zip_ref:
zip_ref.extractall(extract_to)
print(f"Extracted files to {extract_to}")

# Використання
unzip_files("documents.zip")
```

Файл blockchain_verification.py

```
import hashlib

# Функція створення хешу документа
def generate_document_hash(content):
hash_object = hashlib.sha256(content.encode())
return hash_object.hexdigest()

# Функція перевірки документа
def verify_document(content, stored_hash):
return generate_document_hash(content) == stored_hash

# Приклад використання
doc_content = "Secure document example."
doc_hash = generate_document_hash(doc_content)
print("Document verified:", verify_document(doc_content, doc_hash))
```

Файл file_recovery.py

```
import os
import shutil

RECYCLE_BIN = "recycle_bin"

# Функція переміщення в кошик
def move_to_recycle_bin(filepath):
os.makedirs(RECYCLE_BIN, exist_ok=True)
shutil.move(filepath, RECYCLE_BIN)
print(f"Moved {filepath} to recycle bin.")

# Функція відновлення файлу
def restore_file(filename):
filepath = os.path.join(RECYCLE_BIN, filename)
if os.path.exists(filepath):
shutil.move(filepath, ".")
print(f"Restored {filename}")
else:
print("File not found in recycle bin")

# Використання
move_to_recycle_bin("test.txt")
restore_file("test.txt")
```

Файл chat_system.py

```
import socket
import threading

clients = []

# Функція обробки клієнтів
def handle_client(client_socket):
while True:
try:
message = client_socket.recv(1024).decode()
print(f"Received: {message}")
broadcast(message, client_socket)
except:
clients.remove(client_socket)
break

# Функція надсилання повідомлень усім клієнтам
def broadcast(message, sender_socket):
for client in clients:
if client != sender_socket:
client.send(message.encode())

# Сервер чату
def start_chat_server():
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(("0.0.0.0", 5555))
server.listen(5)
print("Chat server started...")
while True:
client_socket, addr = server.accept()
clients.append(client_socket)
threading.Thread(target=handle_client, args=(client_socket,)).start()

start_chat_server()
```

Файл backup_system.py

```
import shutil
import os
import time
```

```
# Функція створення резервної копії
def create_backup(source_folder, backup_folder="backup"):
os.makedirs(backup_folder, exist_ok=True)
timestamp = time.strftime("%Y%m%d%H%M%S")
backup_path = os.path.join(backup_folder, f"backup_{timestamp}")
shutil.copytree(source_folder, backup_path)
print(f"Backup created at: {backup_path}")

# Автоматичне резервне копіювання кожні 24 години
def schedule_backup(source_folder, interval=86400):
while True:
create_backup(source_folder)
time.sleep(interval)

# Використання
# schedule_backup("documents")
```

Файл ml_anomaly_detection.py

```
import random

# Функція генерації випадкових дій користувачів
def generate_user_activity():
actions = ["upload", "delete", "download", "modify"]
return [random.choice(actions) for _ in range(1000)]

# Алгоритм виявлення аномалій
def detect_anomalies(activity_log):
anomaly_threshold = 50  # Поріг аномалій
activity_counts = {action: activity_log.count(action) for action in
set(activity_log)}
anomalies = {action: count for action, count in activity_counts.items() if count
> anomaly_threshold}
return anomalies

# Використання
activity_log = generate_user_activity()
print(detect_anomalies(activity_log))
```

Файл encryption_key_generator.py

```
from cryptography.fernet import Fernet

# Функція генерації ключа
def generate_encryption_key():
key = Fernet.generate_key()
with open("encryption.key", "wb") as key_file:
key_file.write(key)
print("Encryption key generated and saved.")

# Використання
generate_encryption_key()
```

Файл websocket_updates.py

```
import asyncio
import websockets

clients = set()

# Функція обробки клієнтів
async def handler(websocket, path):
clients.add(websocket)
try:
async for message in websocket:
for client in clients:
await client.send(message)
finally:
clients.remove(websocket)

# Запуск сервера
start_server = websockets.serve(handler, "localhost", 6789)
asyncio.get_event_loop().run_until_complete(start_server)
asyncio.get_event_loop().run_forever()
```

Файл api_integration.py

```
from flask import Flask, request, jsonify

app = Flask(__name__)

# API маршрут для отримання файлів
@app.route("/api/files", methods=["GET"])
def get_files():
return jsonify(["file1.txt", "file2.pdf"])

# API маршрут для завантаження файлу
@app.route("/api/upload", methods=["POST"])
def upload_file():
file = request.files["file"]
file.save(f"uploads/{file.filename}")
return jsonify({"message": "File uploaded successfully"})

# Запуск API
if __name__ == "__main__":
app.run(debug=True)
```

Файл document_statistics.py

```
import matplotlib.pyplot as plt

# Функція генерації статистики
def generate_statistics():
data = {"uploads": 50, "downloads": 30, "deletes": 10}
plt.bar(data.keys(), data.values())
plt.xlabel("Actions")
plt.ylabel("Count")
plt.title("User Activity Statistics")
```

```
plt.show()

# Використання
generate_statistics()
```

Файл advanced_search.py

```
import os

# Функція пошуку за ключовим словом
def search_files(keyword, folder="documents"):
results = []
for root, dirs, files in os.walk(folder):
for file in files:
filepath = os.path.join(root, file)
with open(filepath, "r", errors="ignore") as f:
if keyword in f.read():
results.append(filepath)
return results

# Використання
print(search_files("confidential"))
```

Файл cloud_integration.py

```
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
import dropbox

# Функція підключення до Google Drive
def google_drive_connect():
gauth = GoogleAuth()
gauth.LocalWebserverAuth()
drive = GoogleDrive(gauth)
return drive

# Функція завантаження файлу в Google Drive
def upload_to_google_drive(file_path):
drive = google_drive_connect()
file_drive = drive.CreateFile({"title": file_path})
file_drive.SetContentFile(file_path)
file_drive.Upload()
print(f"Uploaded {file_path} to Google Drive.")

# Функція підключення до Dropbox
def dropbox_connect(access_token):
dbx = dropbox.Dropbox(access_token)
return dbx

# Функція завантаження файлу в Dropbox
def upload_to_dropbox(file_path, access_token):
dbx = dropbox_connect(access_token)
with open(file_path, "rb") as f:
dbx.files_upload(f.read(), f"/{file_path}")
print(f"Uploaded {file_path} to Dropbox.")

# Використання (додати токен Dropbox перед запуском)
# upload_to_google_drive("test.txt")
# upload_to_dropbox("test.txt", "your_dropbox_access_token")
```

Файл document_translation.py

```
from deep_translator import GoogleTranslator

# Функція перекладу документа
def translate_document(file_path, target_language="en"):
with open(file_path, "r", encoding="utf-8") as file:
content = file.read()
translated_content = GoogleTranslator(source="auto",
target=target_language).translate(content)
translated_file_path = f"{file_path.split('.')[0]}_translated.txt"
with open(translated_file_path, "w", encoding="utf-8") as file:
file.write(translated_content)
print(f"Translated document saved as {translated_file_path}")

# Використання
# translate_document("document.txt", "fr")
```

Файл session_management.py

```
from flask import Flask, session
from flask_session import Session

app = Flask(__name__)
app.config["SESSION_TYPE"] = "filesystem"
Session(app)

# Функція входу користувача
@app.route("/login/<username>")
def login(username):
session["user"] = username
return f"User {username} logged in."

# Функція виходу користувача
@app.route("/logout")
def logout():
session.pop("user", None)
return "User logged out."

# Функція перевірки активної сесії
@app.route("/status")
def session_status():
if "user" in session:
return f"User {session['user']} is logged in."
return "No active session."

# Запуск сервера
if __name__ == "__main__":
app.run(debug=True)
```

Файл DocxData.py

```
from __future__ import annotations

from typing import TYPE_CHECKING, Iterator, cast, overload

from typing_extensions import TypeAlias

from docx.blkcntnr import BlockItemContainer
from docx.enum.style import WD_STYLE_TYPE
from docx.enum.table import WD_CELL_VERTICAL_ALIGNMENT
from docx.oxml.simpletypes import ST_Merge
from docx.oxml.table import CT_TblGridCol
from docx.shared import Inches, Parented, StoryChild, lazyproperty

if TYPE_CHECKING:
    import docx.types as t
    from docx.enum.table import WD_ROW_HEIGHT_RULE, WD_TABLE_ALIGNMENT,
WD_TABLE_DIRECTION
    from docx.oxml.table import CT_Row, CT_Tbl, CT_TblPr, CT_Tc
    from docx.shared import Length
    from docx.styles.style import (
        ParagraphStyle,
        _TableStyle,  # pyright: ignore[reportPrivateUsage]
    )

TableParent: TypeAlias = "Table | _Columns | _Rows"


class Table(StoryChild):
    """Proxy class for a WordprocessingML ``<w:tbl>`` element."""

    def __init__(self, tbl: CT_Tbl, parent: t.ProvidesStoryPart):
        super(Table, self).__init__(parent)
        self._element = tbl
        self._tbl = tbl

    def add_column(self, width: Length):
        """Return a |_Column| object of `width`, newly added rightmost to the
table."""
        tblGrid = self._tbl.tblGrid
        gridCol = tblGrid.add_gridCol()
        gridCol.w = width
        for tr in self._tbl.tr_lst:
            tc = tr.add_tc()
            tc.width = width
        return _Column(gridCol, self)

    def add_row(self):
        """Return a |_Row| instance, newly added bottom-most to the table."""
        tbl = self._tbl
        tr = tbl.add_tr()
        for gridCol in tbl.tblGrid.gridCol_lst:
            tc = tr.add_tc()
            if gridCol.w is not None:
                tc.width = gridCol.w
        return _Row(tr, self)

    @alignment.setter
    def alignment(self, value: WD_TABLE_ALIGNMENT | None):
        self._tblPr.alignment = value

    @property
    def autofit(self) -> bool:
        """|True| if column widths can be automatically adjusted to improve the
fit of
        cell contents.
```

```
        |False| if table layout is fixed. Column widths are adjusted in either
case if
        total column width exceeds page width. Read/write boolean.
        """
        return self._tblPr.autofit

    @autofit.setter
    def autofit(self, value: bool):
        self._tblPr.autofit = value

    def cell(self, row_idx: int, col_idx: int) -> _Cell:
        """|_Cell| at `row_idx`, `col_idx` intersection.

        (0, 0) is the top, left-most cell.
        """
        cell_idx = col_idx + (row_idx * self._column_count)
        return self._cells[cell_idx]

    def column_cells(self, column_idx: int) -> list[_Cell]:
        """Sequence of cells in the column at `column_idx` in this table."""
        cells = self._cells
        idxs = range(column_idx, len(cells), self._column_count)
        return [cells[idx] for idx in idxs]

    @lazyproperty
    def columns(self):
        """|_Columns| instance representing the sequence of columns in this
table."""
        return _Columns(self._tbl, self)

    def row_cells(self, row_idx: int) -> list[_Cell]:
        """DEPRECATED: Use `table.rows[row_idx].cells` instead.

        Sequence of cells in the row at `row_idx` in this table.
        """
        column_count = self._column_count
        start = row_idx * column_count
        end = start + column_count
        return self._cells[start:end]

    @lazyproperty
    def rows(self) -> _Rows:
        """|_Rows| instance containing the sequence of rows in this table."""
        return _Rows(self._tbl, self)

    @property
    def style(self) -> _TableStyle | None:
        """|_TableStyle| object representing the style applied to this table.
        """
        style_id = self._tbl.tblStyle_val
        return cast("_TableStyle | None", self.part.get_style(style_id,
WD_STYLE_TYPE.TABLE))

    @style.setter
    def style(self, style_or_name: _TableStyle | str | None):
        style_id = self.part.get_style_id(style_or_name, WD_STYLE_TYPE.TABLE)
        self._tbl.tblStyle_val = style_id

    @property
    def table(self):
        """Provide child objects with reference to the |Table| object they
belong to,
        """
        return self

    @table_direction.setter
    def table_direction(self, value: WD_TABLE_DIRECTION | None):
        self._element.bidiVisual_val = value
```

```
    @property
    def _cells(self) -> list[_Cell]:
        """
        """
        col_count = self._column_count
        cells: list[_Cell] = []
        for tc in self._tbl.iter_tcs():
            for grid_span_idx in range(tc.grid_span):
                if tc.vMerge == ST_Merge.CONTINUE:
                    cells.append(cells[-col_count])
                elif grid_span_idx > 0:
                    cells.append(cells[-1])
                else:
                    cells.append(_Cell(tc, self))
        return cells

    @property
    def _column_count(self):
        """The number of grid columns in this table."""
        return self._tbl.col_count

    @property
    def _tblPr(self) -> CT_TblPr:
        return self._tbl.tblPr


class _Cell(BlockItemContainer):
    """Table cell."""

    def __init__(self, tc: CT_Tc, parent: TableParent):
        super(_Cell, self).__init__(tc, cast("t.ProvidesStoryPart", parent))
        self._parent = parent
        self._tc = self._element = tc

    def add_paragraph(self, text: str = "", style: str | ParagraphStyle | None =
None):
        """Return a paragraph newly added to the end of the content in this
cell.
        """
        return super(_Cell, self).add_paragraph(text, style)

    def add_table(  # pyright: ignore[reportIncompatibleMethodOverride]
        self, rows: int, cols: int
    ) -> Table:
        """Return a table newly added to this cell after any existing cell
content.
        """
        width = self.width if self.width is not None else Inches(1)
        table = super(_Cell, self).add_table(rows, cols, width)
        self.add_paragraph()
        return table

    @property
    def grid_span(self) -> int:
        """Number of layout-grid cells this cell spans horizontally.
        A "normal" cell has a grid-span of 1. A horizontally merged cell has a
grid-span of 2 or
        more.
        """
        return self._tc.grid_span

    def merge(self, other_cell: _Cell):
        """Return a merged cell created by spanning the rectangular region
having this
        cell and `other_cell` as diagonal corners.

        Raises |InvalidSpanError| if the cells do not define a rectangular
region.
        """
```

```
        tc, tc_2 = self._tc, other_cell._tc
        merged_tc = tc.merge(tc_2)
        return _Cell(merged_tc, self._parent)

    @property
    def paragraphs(self):
        """List of paragraphs in the cell.
        """
        return super(_Cell, self).paragraphs

    @property
    def tables(self):
        """List of tables in the cell, in the order they appear.

        Read-only.
        """
        return super(_Cell, self).tables

    @property
    def text(self) -> str:
        """The entire contents of this cell as a string of text.

        Assigning a string to this property replaces all existing content with a
single
        paragraph containing the assigned text in a single run.
        """
        return "\n".join(p.text for p in self.paragraphs)

    @text.setter
    def text(self, text: str):
        """Write-only.

        Set entire contents of cell to the string `text`. Any existing content
or
        revisions are replaced.
        """
        tc = self._tc
        tc.clear_content()
        p = tc.add_p()
        r = p.add_r()
        r.text = text

    @property
    def vertical_alignment(self):
        """Member of :ref:`WdCellVerticalAlignment` or None.
        """
        tcPr = self._element.tcPr
        if tcPr is None:
            return None
        return tcPr.vAlign_val

    @vertical_alignment.setter
    def vertical_alignment(self, value: WD_CELL_VERTICAL_ALIGNMENT | None):
        tcPr = self._element.get_or_add_tcPr()
        tcPr.vAlign_val = value

    @property
    def width(self):
        """The width of this cell in EMU, or |None| if no explicit width is
set."""
        return self._tc.width

    @width.setter
    def width(self, value: Length):
        self._tc.width = value


class _Column(Parented):
    """Table column."""
```

```
    def __init__(self, gridCol: CT_TblGridCol, parent: TableParent):
        super(_Column, self).__init__(parent)
        self._parent = parent
        self._gridCol = gridCol

    @property
    def cells(self) -> tuple[_Cell, ...]:
        """Sequence of |_Cell| instances corresponding to cells in this
column."""
        return tuple(self.table.column_cells(self._index))

    @property
    def table(self) -> Table:
        """Reference to the |Table| object this column belongs to."""
        return self._parent.table

    @property
    def width(self) -> Length | None:
        """The width of this column in EMU, or |None| if no explicit width is
set."""
        return self._gridCol.w

    @width.setter
    def width(self, value: Length | None):
        self._gridCol.w = value

    @property
    def _index(self):
        """Index of this column in its table, starting from zero."""
        return self._gridCol.gridCol_idx


class _Columns(Parented):
    """Sequence of |_Column| instances corresponding to the columns in a table.

    Supports ``len()``, iteration and indexed access.
    """

    def __init__(self, tbl: CT_Tbl, parent: TableParent):
        super(_Columns, self).__init__(parent)
        self._parent = parent
        self._tbl = tbl

    def __getitem__(self, idx: int):
        """Provide indexed access, e.g. 'columns[0]'."""
        try:
            gridCol = self._gridCol_lst[idx]
        except IndexError:
            msg = "column index [%d] is out of range" % idx
            raise IndexError(msg)
        return _Column(gridCol, self)

    def __iter__(self):
        for gridCol in self._gridCol_lst:
            yield _Column(gridCol, self)

    def __len__(self):
        return len(self._gridCol_lst)

    @property
    def table(self) -> Table:
        """Reference to the |Table| object this column collection belongs to."""
        return self._parent.table

    @property
    def _gridCol_lst(self):
        """Sequence containing ``<w:gridCol>`` elements for this table, each
        representing a table column."""
```

```
        tblGrid = self._tbl.tblGrid
        return tblGrid.gridCol_lst


    @property
    def height(self) -> Length | None:
        """Return a |Length| object representing the height of this cell, or
|None| if
        no explicit height is set."""
        return self._tr.trHeight_val

    @height.setter
    def height(self, value: Length | None):
        self._tr.trHeight_val = value



    @height_rule.setter
    def height_rule(self, value: WD_ROW_HEIGHT_RULE | None):
        self._tr.trHeight_hRule = value

    @property
    def table(self) -> Table:
        """Reference to the |Table| object this row belongs to."""
        return self._parent.table

    @property
    def _index(self) -> int:
        """Index of this row in its table, starting from zero."""
        return self._tr.tr_idx


class _Rows(Parented):
    """Sequence of |_Row| objects corresponding to the rows in a table.

    Supports ``len()``, iteration, indexed access, and slicing.
    """

    def __init__(self, tbl: CT_Tbl, parent: TableParent):
        super(_Rows, self).__init__(parent)
        self._parent = parent
        self._tbl = tbl

    def __iter__(self):
        return (_Row(tr, self) for tr in self._tbl.tr_lst)

    def __len__(self):
        return len(self._tbl.tr_lst)

    @property
    def table(self) -> Table:
        """Reference to the |Table| object this row collection belongs to."""
        return self._parent.table
```

Файл UNPdoc.py

```
import os
import re
import sys
import tempfile
import mimetypes
import subprocess

import click

FILENAME = object()
OUTPUT_FOLDER = object()
unpackers = []

def register_unpacker(cls):
    unpackers.append(cls)
    return cls

def fnmatch(pattern, filename):
    filename = os.path.basename(os.path.normcase(filename))
    pattern = os.path.normcase(pattern)
    bits = '(%s)' % re.escape(pattern).replace('\\*', ')(.*?)(')
    return re.match('^%s$' % bits, filename)

def which(name):
    path = os.environ.get('PATH')
    if path:
        for p in path.split(os.pathsep):
            p = os.path.join(p, name)
            if os.access(p, os.X_OK):
                return p

def increment_string(string):
    m = re.match(r'(.*?)(\d+)$', string)
    if m is None:
        return string + '-2'
    return m.group(1) + str(int(m.group(2)) + 1)

def get_mimetype(filename):
    file_executable = which('file')
    if file_executable is not None:
        rv = subprocess.Popen(['file', '-b', '--mime-type', filename],
                              stdout=subprocess.PIPE,
                              stderr=subprocess.PIPE).communicate()[0].strip()
        if rv:
            return rv
    return mimetypes.guess_type(filename)[0]

def line_parser(format):
    pass

class StreamProcessor(object):

    def __init__(self, format, stream):
        self.regex = re.compile(format)
        self.stream = stream

    def process(self, p):
        stream = getattr(p, self.stream)
        while 1:
            line = stream.readline()
            if not line:
                break
            match = self.regex.search(line)
            if match is not None:
                yield match.group(1)
```

```
class UnpackerBase(object):
    id = None
    name = None
    executable = None
    filename_patterns = ()
    mimetypes = ()
    brew_package = None
    args = ()
    cwd = OUTPUT_FOLDER

    def __init__(self, filename, silent=False):
        self.filename = filename
        self.silent = silent
        self.assert_available()

    @classmethod
    def filename_matches(cls, filename):
        for pattern in cls.filename_patterns:
            if fnmatch(pattern, filename) is not None:
                return True

    @classmethod
    def mimetype_matches(cls, filename):
        mt = get_mimetype(filename)
        return mt in cls.mimetypes

    @classmethod
    def find_executable(cls):
        return which(cls.executable)

    @property
    def basename(self):
        for pattern in self.filename_patterns:
            match = fnmatch(pattern, self.filename)
            if match is None:
                continue
            pieces = match.groups()
            if pieces and pieces[-1].startswith('.'):
                return ''.join(pieces[:-1])
        return os.path.basename(self.filename).rsplit('.', 1)[0]

    def assert_available(self):
        if self.find_executable() is not None:
            return

        msgs = ['Cannot unpack "%s" because %s is not available.' % (
            click.format_filename(self.filename),
            self.executable,
        )]
        if sys.platform == 'darwin' and self.brew_package is not None:
            msgs.extend((
                'You can install the unpacker using brew:',
                '',
                '    $ brew install %s' % self.brew_package,
            ))

        raise click.UsageError('\n'.join(msgs))

    def get_args_and_cwd(self, dst):
        def convert_arg(arg):
            if arg is FILENAME:
                return self.filename
            if arg is OUTPUT_FOLDER:
                return dst
            return arg

        args = [self.find_executable()]
        for arg in self.args:
            args.append(convert_arg(arg))
```

```
    cwd = convert_arg(self.cwd)
    if cwd is None:
        cwd = '.'
    return args, cwd

def report_file(self, filename):
    if not self.silent:
        click.echo(click.format_filename(filename), err=True)

def real_unpack(self, dst, silent):
    raise NotImplementedError()

def finish_unpacking(self, tmp_dir, dst):
    # Calculate the fallback destination
    basename = self.basename
    fallback_dst = os.path.join(os.path.abspath(dst), basename)
    while os.path.isdir(fallback_dst):
        fallback_dst = increment_string(fallback_dst)

    # Find how many unpacked files there are.  If there is more than
    # one, then we have to go to the fallback destination.  Same goes
    # if the intended destination already exists.
    contents = os.listdir(tmp_dir)
    if len(contents) == 1:
        the_one_file = contents[0]
        intended_dst = os.path.join(dst, the_one_file)
    else:
        intended_dst = None
    if intended_dst is None or os.path.exists(intended_dst):
        os.rename(tmp_dir, fallback_dst)
        return fallback_dst

    # Otherwise rename the first thing to the intended destination
    # and remove the temporary directory.
    os.rename(os.path.join(tmp_dir, the_one_file), intended_dst)
    os.rmdir(tmp_dir)
    return intended_dst

def cleanup(self, dst):
    try:
        os.remove(dst)
    except Exception:
        pass

    try:
        import shutil
        shutil.rmtree(dst)
    except Exception:
        pass

def unpack(self, dst):
    if not self.silent:
        click.secho('Unpacking "%s" with %s' % (
            self.filename,
            self.executable,
        ), fg='yellow')

    dst = os.path.abspath(dst)
    try:
        os.makedirs(dst)
    except OSError:
        pass

    tmp_dir = tempfile.mkdtemp(prefix='.' + self.basename, dir=dst)
    try:
        if self.real_unpack(tmp_dir) != 0:
            click.secho('Error: unpacking through %s failed.'
                        % self.executable, fg='red')
            sys.exit(2)
```

```
            final = self.finish_unpacking(tmp_dir, dst)
            if not self.silent:
                click.secho('Extracted to %s' % final, fg='green')
        finally:
            self.cleanup(tmp_dir)

    def dump_command(self, dst):
        args, cwd = self.get_args_and_cwd(dst)
        for idx, arg in enumerate(args):
            if arg.split() != [arg]:
                args[idx] = '"%s"' % \
                    arg.replace('\\', '\\\\').replace('"', '\\"')
        click.echo(' '.join(args))

    def __repr__(self):
        return '<Unpacker %r>' % (
            self.name,
        )

class Unpacker(UnpackerBase):
    stream_processor = None

    def real_unpack(self, dst):
        args, cwd = self.get_args_and_cwd(dst)
        extra = {}
        extra[self.stream_processor.stream] = subprocess.PIPE
        c = subprocess.Popen(args, cwd=cwd, **extra)
        for filename in self.stream_processor.process(c):
            self.report_file(filename)
        return c.wait()

class SingleInplaceUnpacker(UnpackerBase):

    def real_unpack(self, dst):
        args, cwd = self.get_args_and_cwd(dst)
        filename = os.path.join(dst, self.basename)
        with open(filename, 'wb') as f:
            rv = subprocess.Popen(args, cwd=cwd, stdout=f).wait()
        self.report_file(filename)
        return rv

tar_stream_processor = StreamProcessor(
    format=r'^x (.*?)$',
    stream='stderr',
)

@register_unpacker
class TarUnpacker(Unpacker):
    id = 'tar'
    name = 'Uncompressed Tarballs'
    filename_patterns = ['*.tar']
    executable = 'tar'
    args = ['xvf', FILENAME]
    stream_processor = tar_stream_processor
    mimetypes = ['application/x-tar']

@register_unpacker
class TarGzUnpacker(Unpacker):
    id = 'tgz'
    name = 'Gzip Compressed Tarballs'
    filename_patterns = ['*.tar.gz', '*.tgz']
    executable = 'tar'
    args = ['xvzf', FILENAME]
    stream_processor = tar_stream_processor

@register_unpacker
class TarBz2Unpacker(Unpacker):
    id = 'tbz2'
```

```
    name = 'Bz2 Compressed Tarballs'
    filename_patterns = ['*.tar.bz2']
    executable = 'tar'
    args = ['xvjf', FILENAME]
    stream_processor = tar_stream_processor

@register_unpacker
class TarXZUnpacker(UnpackerBase):
    id = 'txz'
    name = 'XZ Compressed Tarballs'
    filename_patterns = ['*.tar.xz']
    executable = 'unxz'
    args = ['-c', FILENAME]
    brew_package = 'xz'

    def real_unpack(self, dst):
        args, cwd = self.get_args_and_cwd(dst)
        tar = subprocess.Popen(['tar', 'x'], cwd=cwd,
                               stderr=subprocess.PIPE,
                               stdin=subprocess.PIPE)
        xz = subprocess.Popen(args, stdout=subprocess.PIPE, cwd=cwd)
        while 1:
            chunk = xz.stdout.read(131072)
            if not chunk:
                break
            tar.stdin.write(chunk)
        tar.stdin.close()
        xz.stdout.close()
        for proc in tar, xz:
            rv = proc.wait()
            if rv != 0:
                return rv
            return 0

@register_unpacker
class GzipUnpacker(SingleInplaceUnpacker):
    id = 'gz'
    name = 'Gzip Compressed Files'
    filename_patterns = ['*.gz']
    executable = 'gunzip'
    args = ['-c', FILENAME]
    mimetypes = ['application/x-gzip']

@register_unpacker
class Bz2Unpacker(SingleInplaceUnpacker):
    id = 'bz2'
    name = 'Bz2 Compressed Files'
    filename_patterns = ['*.bz2']
    executable = 'bunzip2'
    args = ['-c', FILENAME]
    mimetypes = ['application/x-bzip2']

@register_unpacker
class XZUnpacker(SingleInplaceUnpacker):
    id = 'xz'
    name = 'XZ Compressed Files'
    filename_patterns = ['*.xz']
    executable = 'unxz'
    args = ['-c', FILENAME]
    brew_package = 'xz'
    mimetypes = ['application/x-xz']

@register_unpacker
class ZipUnpacker(Unpacker):
    id = 'zip'
    name = 'Zip Archives'
    filename_patterns = ['*.zip', '*.egg', '*.whl', '*.jar']
    executable = 'unzip'
    args = [FILENAME]
```

```
    mimetypes = ['application/zip']
    stream_processor = StreamProcessor(
        format=r'^  inflating: (.*?)$',
        stream='stdout',
    )

@register_unpacker
class RarUnpacker(Unpacker):
    id = 'rar'
    name = 'WinRAR Archives'
    filename_patterns = ['*.rar']
    executable = 'unrar'
    args = ['-idp', '-y', 'x', FILENAME]
    mimetypes = ['application/zip']
    brew_package = 'unrar'
    stream_processor = StreamProcessor(
        format=r'^Extracting  (.*?)\s+OK\s*$',
        stream='stdout',
    )

@register_unpacker
class P7ZipUnpacker(Unpacker):
    id = '7z'
    name = '7zip Archives'
    filename_patterns = ['*.7z']
    executable = '7z'
    args = ['-bd', 'x', FILENAME]
    mimetypes = ['application/zip']
    brew_package = 'p7zip'
    stream_processor = StreamProcessor(
        format=r'^Extracting  (.*?)$',
        stream='stdout',
    )

@register_unpacker
class CabUnpacker(Unpacker):
    id = 'cab'
    name = 'Windows Cabinet Archive'
    filename_patterns = ['*.cab']
    executable = 'cabextract'
    args = ['-f', FILENAME]
    mimetypes = ['application/vnd.ms-cab-compressed']
    brew_package = 'cabextract'
    stream_processor = StreamProcessor(
        format=r'^  extracting (.*?)$',
        stream='stdout',
    )
@register_unpacker
class ArUnpacker(Unpacker):
    id = 'ar'
    name = 'AR Archives'
    filename_patterns = ['*.a']
    executable = 'ar'
    args = ['-vx', FILENAME]
    mimetypes = ['application/x-archive']
    stream_processor = StreamProcessor(
        format=r'^x - (.*?)$',
        stream='stdout',
    )

class DMGUnpacker(UnpackerBase):
    id = 'dmg'
    name = 'Apple Disk Image'
    filename_patterns = ['*.dmg', '*.sparseimage']
    executable = 'hdiutil'
    args = ['attach', '-nobrowse', FILENAME]
    def real_unpack(self, dst):
        mp = dst + '---mp'
        args, cwd = self.get_args_and_cwd(dst)
```

```
        args.append('-mountpoint')
        args.append(mp)
        with open('/dev/null', 'wb') as devnull:
            rv = subprocess.Popen(args, cwd=cwd,
                                  stdout=devnull,
                                  stderr=devnull).wait()
            if rv != 0:
                return rv
        p = subprocess.Popen(['cp', '-vpR', mp + '/', dst],
                             stdout=subprocess.PIPE)
        while 1:
            line = p.stdout.readline()
            if not line:
                break
            line = line.rstrip('\r\n').split(' -> ', 1)[1]
            if line.startswith(dst + '/'):
                line = line[len(dst) + 1:].strip()
                if line:
                    self.report_file(line)
        return p.wait()
    def cleanup(self, dst):
        with open('/dev/null', 'wb') as devnull:
            subprocess.Popen(['umount', dst + '---mp'],
                             stderr=devnull, stdout=devnull).wait()
        UnpackerBase.cleanup(self, dst)
if sys.platform == 'darwin':
    register_unpacker(DMGUnpacker)

def get_unpacker_class(filename):
    uifn = click.format_filename(filename)

    for unpacker_cls in unpackers:
        if unpacker_cls.filename_matches(filename):
            return unpacker_cls
    for unpacker_cls in unpackers:
        if unpacker_cls.mimetype_matches(filename):
            return unpacker_cls
   raise click.UsageError('Could not determine unpacker for "%s".' % uifn)
def list_unpackers(ctx, param, value):
    if not value:
        return
    for unpacker in sorted(unpackers, key=lambda x: x.name.lower()):
        if unpacker.find_executable() is None:
            continue
        click.echo('- %- 5s %s (%s)' % (
            unpacker.id,
            unpacker.name,
            '; '.join(unpacker.filename_patterns),
        ))
    ctx.exit()
    for unpacker in unpackers:
        if dump_command:
            unpacker.dump_command(output)
        else:
            unpacker.unpack(output)
```