Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Бакалаврська дипломна робота на тему:
«Засіб захищеного зберігання паролів»

08-20.БДР.007.00.000 ПЗ

Виконав: студент 4 курсу, групи 1БС-19б
спеціальності 125 Кібербезпека
_____ Володимир КЛИМЕНКО

Керівник: к. т. н., доцент, доцент каф. ЗІ
_____ Юрій БАРИШЕВ
«16» червня 2023 р.

Рецензент: к. т. н., доцент, доцент каф. ПЗ
_____ Наталя БАБЮК
«16» червня 2023 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., професор
_____ Лужецький В. А.
«16» червня 2023 р.

Вінниця ВНТУ – 2023 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти I (бакалаврський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма – «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ,
д. т. н., професор
Володимир ЛУЖЕЦЬКИЙ
«20» березня 2023 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Клименку Володимиру Олександровичу

1. Тема роботи: «Засіб захищеного зберігання паролів»
керівник роботи: Баришев Юрій Володимирович, к. т. н., доцент, доцент кафедри ЗІ,
затверджено наказом ректора ВНТУ від 20 березня 2023 року №67.

2. Строк подання студентом роботи 16 червня 2023 р.

3. Вихідні дані до роботи:
   - вид автентифікації користувачів – двофакторна: парольна та носій інформації;
   - тип шифрування – блокове;
   - нефункціональні вимоги – кросплатформений застосунок сумісний з Windows, Linux, MacOS.

4. Зміст текстової частини: Вступ. 1. Аналіз методів захисту паролів. 2. Архітектура засобу захищеного зберігання паролів. 3. Алгоритми роботи засобу захищеного зберігання паролів. 4. Тестування засобу захищеного зберігання паролів. Висновки. Список використаних джерел. Додатки.

5. Перелік ілюстративного матеріалу: порівняльний аналіз засобів захисту паролів (плакат А4), узагальнена архітектура засобу захищеного зберігання паролів (плакат А4), структура модуля захисту конфіденційності (плакат А4), структура модуля захисту цілісності (плакат А4), узагальнений алгоритм роботи засобу (плакат А4), алгоритм захисту конфіденційності

(плакат А4), алгоритм захисту цілісності паролів (плакат А4), порівняльний аналіз засобів розробки (плакат А4), результати блокового тестування (плакат А4), результати інтеграційного тестування (плакат А4)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.03.23	10.04.23
2	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.05.23	08.05.23
3	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.03.23	25.05.23
4	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.03.23	10.06.23

7. Дата видачі завдання 20 березня 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	20.03.23 – 26.03.23	
2	Аналіз джерел за напрямком бакалаврської дипломної роботи	27.03.23 – 09.04.23	
3	Розробка рішень, моделей, алгоритмів	10.04.23 – 23.04.23	
4	Практична реалізація, моделювання, експериментування, результати	24.04.23 – 21.05.23	
5	Аналіз виконання ТЗ, висновки	22.05.23 – 24.05.23	
6	Оформлення пояснювальної записки	25.05.23 – 31.05.23	
7	Попередній захист та доопрацювання БДР	01.06.23 – 15.06.23	
8	Представлення БДР на захист	16.06.23 – 19.06.23	
9	Захист БДР	20.06.23 – 23.06.23	

Студент _____ Володимир КЛИМЕНКО

Керівник роботи _____ Юрій БАРИШЕВ

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 95 сторінок формату А4, на яких є 37 рисунків, 3 таблиці, список використаних джерел містить 36 найменувань.

У даній роботі було проведено порівняльний аналіз відомих засобів захищеного зберігання паролів, аналіз атак на засоби захищеного зберігання паролів, сформовано вимоги для розробки програмного засобу. Розроблено узагальнену архітектуру засобу. Розроблено алгоритм роботи засобу захищеного зберігання паролів. Розроблено програмний застосунок з графічним інтерфейсом, функцією автентифікації користувача, створення, редагування та видалення входів, функцією ручного блокування та автоблокування застосунку після певного часу бездіяльності, функції копіювання імені користувача та його паролю до буферу обміну та очищення буферу обміну через невеликий проміжок часу.

Ключові слова: пароль, захист конфіденційності злам паролів, менеджер паролів, автентифікація, багатофакторна автентифікація, шифрування.

ABSTRACT

The bachelor thesis consists of 95 pages of A4 format, on which there are 37 figures, 3 tables, the list of used sources contains 36 items.

In this work, a comparative analysis of known password managers was carried out, an analysis of attacks on password managers was carried out, and requirements for the development of a software tool were formed. A generalized architecture of the software tool has been developed. An algorithm of tool for password secure storing has been developed. A software application has been developed with a graphical user interface, the function of user authentication, creation, editing and deletion of entries, the function of manual blocking and auto-blocking of the application after a certain period of inactivity, the function of copying the user name and password to the clipboard and clearing the clipboard after a short period of time.

Keywords: password, privacy protection, password breaking, password manager, authentication, multifactor authentication, encryption.

ЗМІСТ

ВСТУП 8

1 АНАЛІЗ МЕТОДІВ ЗАХИСТУ ПАРОЛІВ 9

1.1 Аналіз методів парольної автентифікації 9

1.2 Аналіз атак на засоби зберігання паролів 10

1.3 Порівняльний аналіз відомих засобів захищеного зберігання паролів 14

1.4 Постановка задачі 16

1.5 Висновки з розділу 17

2 АРХІТЕКТУРА ЗАСОБУ ЗАХИЩЕНОГО ЗБЕРІГАННЯ ПАРОЛІВ 18

2.1 Узагальнена архітектура засобу 18

2.2 Структура модуля захисту конфіденційності 19

2.3 Обґрунтування вибору криптографічних алгоритмів 20

2.4 Структура модуля захисту цілісності 22

2.5 Структура модуля взаємодії з користувачем 24

2.6 Висновки з розділу 25

3 АЛГОРИТМИ РОБОТИ ЗАСОБУ ЗАХИЩЕНОГО ЗБЕРІГАННЯ ПАРОЛІВ 26

3.1 Узагальнений алгоритм роботи засобу 26

3.2 Алгоритм захисту конфіденційності 28

3.3 Алгоритм роботи модуля криптографічного захисту 29

3.4 Алгоритм захисту цілісності 30

3.5 Алгоритми модуля взаємодії з користувачем 32

3.6 Висновки з розділу 37

4 ТЕСТУВАННЯ ЗАСОБУ ЗАХИЩЕНОГО ЗБЕРІГАННЯ ПАРОЛІВ 38

4.1 Обґрунтування засобів розробки 38

4.2 Основні семантичні одиниці програмного коду 44
4.3 Блокове тестування засобу 50
4.4 Інтеграційне тестування засобу 53
4.5 Висновки з розділу 60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 63
ДОДАТКИ 67
Додаток А. Код основних функціональних можливостей засобу 68
Додаток Б. Коди блочного та інтеграційного тестувань 80
Додаток В. Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень 84

ВСТУП

Цифровізація суспільства вимагає від користувача заводити облікові записи в різних інформаційних системах. Кожна з цих систем потребує автентифікації користувача перед тим, як надавати йому інформаційні послуги.

Для цього використовуються різні види автентифікації, основною з яких є парольна. Люди, які хоча б трохи знаються на кібербезпеці, або кібергігієні намагаються створювати стійкі паролі та знаходити ефективні способи їх зберігання. Особливо, якщо вони підключені до десятків сайтів або сервісів, може бути важко запам’ятати унікальні паролі для кожного облікового запису. Велика кількість порушень безпеки пов’язана з використанням слабких паролів, або використанні одного й того ж паролю для різних сайтів та сервісів.

Виходом з цієї ситуації є засоби захищеного зберігання паролів – менеджери паролів. Вони спрощують процес створення, керування та захисту паролів. Вони зберігають паролі в зашифрованому вигляді й мають функції автоматичного введення та заповнення форм. Саме тому актуально створювати засоби для керування захищеного зберігання паролів.

Об’єктом дослідження є процес захисту паролів.

Предметом дослідження є методи та засоби захищеного зберігання паролів.

Метою дослідження є покращення процесу захисту паролів, шляхом розробки засобу захищеного зберігання паролів.

Для досягнення мети необхідно розв’язати такі задачі:

- аналіз методів захисту паролів;
- аналіз атак на менеджери паролів;
- розробити архітектуру програмного засобу;
- реалізувати алгоритми роботи засобу;
- виконати тестування.

Результати доповідались та були опубліковані у вигляді двох тез доповідей науково-практичних конференцій [1, 2].

1 АНАЛІЗ МЕТОДІВ ЗАХИСТУ ПАРОЛІВ

1.1 Аналіз методів парольної автентифікації

Парольна автентифікація є одним з найпоширеніших методів автентифікації, що використовуються для підтвердження ідентифікатора користувача. Даний метод автентифікації належить до категорії факторів автентифікації користувачів на основі знання чого-небудь [3, 5]. Парольна автентифікація є доволі легкою в розробці та використанні, саме тому вона набула такої популярності, однак, і в неї є недоліки. Для використання даного методу користувачі повинні слідкувати певним принципам задля безпеки їх облікових записів [5]:

- для кожного сайту або сервісу необхідним є створення різних паролів;
- необхідно регулярно замінювати паролі на нові, чим частіше, тим краще;
- не повідомляти нікому свої паролі;
- не пересилати паролі по пошті, або повідомленням через месенджери;
- паролі можна записувати, але тільки в тому випадку, якщо місце де вони записані надійне і доступ до нього маєте тільки Ви;
- не писати паролі на нотатках або картках, які тримаєте біля робочого місця, і не намагайтесь їх заховати в ящики, чи під клавіатурою;
- паролі повинні відповідати певним вимогам безпеки.

Основними вимогами, що висуваються до паролю є такі:

- довжина паролів повинна бути принаймні 12 символів;
- паролі повинні складатись з комбінації літер верхнього та нижнього регістру, цифр та спецсимволів;
- пароль не повинен бути або містити в собі слово, яке можна знайти в словнику, бути ім’ям людини, персонажа, або назвою якого-небудь продукту чи організації.

З цих рекомендацій можна зробити висновок, що користувачеві для того, щоб виконувати всі ці вимоги потрібно мати феноменальну пам’ять, щоб запам’ятовувати паролі, які відповідають вимогам безпеки, і до якого сайту кожен з них потрібен, або використовувати менеджер паролів.

Якщо ж говорити про менеджери паролів, то їх використання хоч і полегшує створення та використання паролів, але має свої недоліки [1]. І головним з них є те, що користувачеві все ще потрібно запам’ятовувати майстер пароль (master password) і регулярно його змінювати, адже якщо зловмисник дізнається його і отримає доступ до менеджеру паролів, він матиме доступ до всіх паролів і облікових записів користувача, які зберігаються в даному засобі зберігання паролів. Тому необхідним є проведення аналізу атак на засоби зберігання паролів.

1.2 Аналіз атак на засоби зберігання паролів

Враховуючи те, що засоби зберігання паролів зберігають велику кількість конфіденційної інформації, в тому числі й самі паролі, вони є привабливою ціллю для хакерів. Існує декілька способів зламати дані засоби. Всі вони залежать від методу зберігання паролів, адже деякі засоби зберігають паролі локально, інші зберігають їх на хмарі, або веб-сайті даного застосунку. Враховуючи це, виділяють такі основні типи атак на засоби зберігання паролів [6]:

- атаки на основі безпосереднього доступу до програмного засобу;
- атаки на основі словників;
- атаки на основі маніпуляції користувачами;
- атаки на постачальника ПЗ.

Для проведення атак на основі безпосереднього доступу до програмного засобу зловмиснику треба отримати доступ до комп’ютера, на якому використовується даний засіб. Зробити це зловмисник може за допомогою соціальної інженерії, або помилок в програмному забезпеченні. Отримавши

доступ до комп’ютера, зловмисник може потрапити до менеджера паролів, якщо в той час він не був заблокований. Саме тому необхідним є встановлення автоблокування менеджеру паролів після певного часу бездіяльності. Якщо ж менеджер паролів заблоковано, зловмисник може встановити на комп’ютер «keylogging program» [7], що є шпигунсьим програмним забезпеченням для викрадення інформації введеної через клавіатуру, в тому числі й паролів. Для перешкодження цьому необхідним є використання багатофакторної автентифікації.

Ще одним методом, за допомогою якого зловмисник може отримати пароль, є зняття з буферу обміну. Якщо в менеджері паролів є можливість копіювати пароль до буферу обміну, наприклад для його редагування, то зловмисник зможе витягнути пароль, переглянувши даний буфер. Тому, якщо залишати можливість копіювання паролю до буферу обміну, то необхідним є встановлення часу, через який буфер буде очищено.

З наведеного вище можна зробити висновок, якщо зловмиснику вдалося отримати доступ до комп’ютера, то здебільшого витоку інформації вже не уникнути, але це не є причиною для ігнорування рекомендацій розробки безпечного програмного забезпечення.

Наступним типом атак на засоби зберігання паролів є атаки на основі словників [8]. Однією з найпоширеніших вразливостей, яку зловмисник може використати є сам користувач, який створює слабкі паролі та використовує їх багаторазово для різних сайтів та сервісів. Це спрощує можливість зламу паролю за допомогою простого перебору за словником (рис. 1.1).

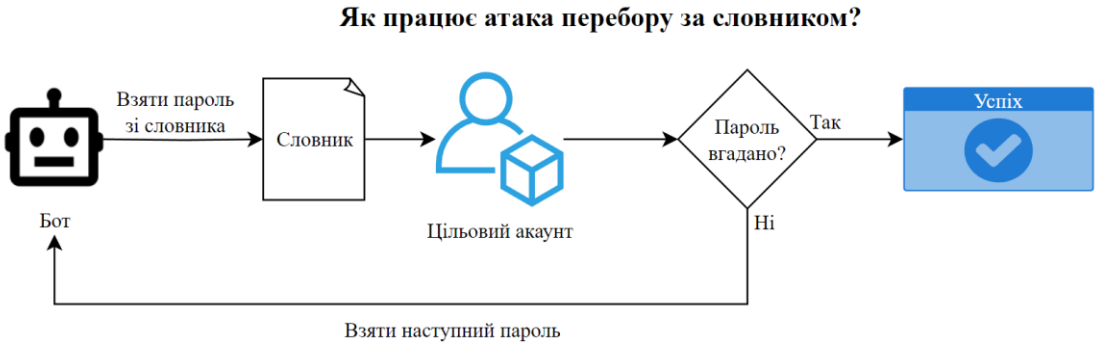


Рисунок 1.1 – Схема атаки на основі словнику

Саме тому для менеджерів паролів важливим є використання функції генерації паролів та відображення підказок користувачеві, щодо стійкості його паролю, коли він його створює або використовує. Також необхідним є нагадування користувачеві, що його пароль вже давно використовується і його необхідно змінити.

Що до атак на основі маніпуляцій користувачами [2], то до даного типу атак можна віднести соціальну інженерію. Наприклад, коли користувачеві на пошту приходить повідомлення нібито від розробників якого-небудь сервісу зі змістом як на рисунку 1.2 [9].

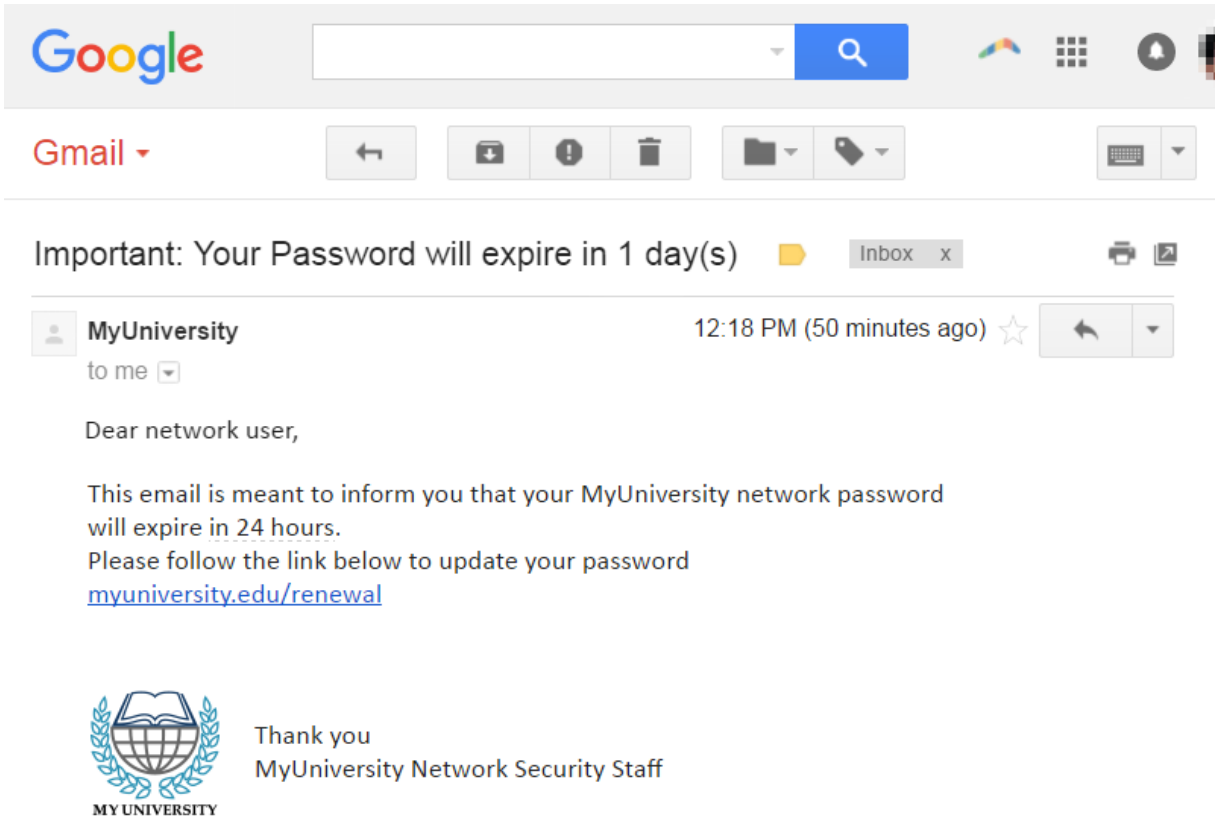


Рисунок 1.2 – Приклад фішингового електронного листа

І якщо користувач перейде за посиланням, то воно перенаправить його на фішинговий сайт, де він сам повідомить зловмиснику свої дані.

Також популярною атакою є фішингові сайти, які маскуються під сайти постачальників менеджерів паролів. Наприклад, у січні 2023 року дану атаку було проведено проти Bitwarden та 1Password, які є одними з провідних

постачальників менеджерів паролів [10]. Зловмисники не тільки створили копії даних сайтів, які майже не можливо було відрізнити від справжніх ні візуально, ні за доменним іменем (рис. 1.3), а ще й спромоглись просунути свої сайти через Google Ads таким чином, щоб їх фішингові сайти відображались першими при запитах у пошуковій системі Google.

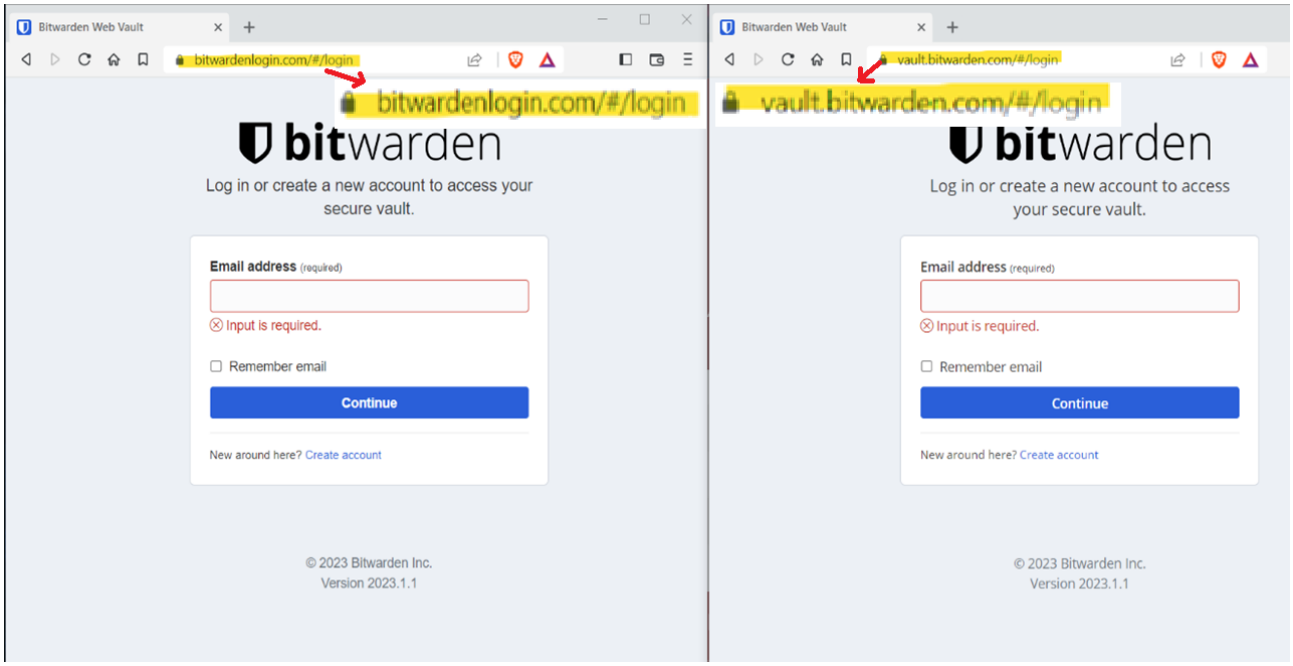


Рисунок 1.3 – Порівняння фішингового сайту Bitwarden зі справжнім

Останнім типом атак на засоби зберігання паролів варто розглянути атаки на постачальника програмного забезпечення. Якщо ж попередні типи атак були спрямовані на користувача менеджерів паролів, то даний тип атак спрямований на постачальників або розробників даного програмного забезпечення. Веб-сайти та сервіси постачальників можуть бути скомпрометовані атаками зловмисників, або містити помилки, якими зловмисники зможуть скористатись. Постачальники програмного забезпечення можуть неправильно або погано реалізувати шифрування. Працівники компанії, що постачає менеджери паролів можуть стати ціллю атак соціальної інженерії та випадково надати доступ зловмисникам до кодів програм, або іншої конфіденційної інформації.

Гарним прикладом цього є атака на менеджер паролів LastPass у серпні 2022 року. Кібератаки скомпрометували внутрішні системи LastPass, викравши вихідний код та інтелектуальну власність. Компанія з керування паролями

заявила, що виявила аномальну активність у своєму середовищі розробки два тижні тому. Дослідивши дані судової експертизи, слідчі встановили, що хтось зламав обліковий запис розробника, щоб отримати доступ до мережі, взявши «частини вихідного коду та деяку технічну інформацію LastPass» [11]. Після чого у грудні 2022 року компанія LastPass знову була атакована. Ось як компанія відреагувала на даний інцидент: «На сьогоднішній день ми визначили, що після отримання ключа доступу до хмарного сховища та ключів дешифрування подвійного контейнера сховища зловмисник скопіював інформацію з резервної копії, яка містила основну інформацію про обліковий запис клієнта та відповідні метадані, зокрема назви компаній, імена кінцевих користувачів, платіжні адреси, адреси електронної пошти, номери телефонів та IP-адреси, з яких клієнти отримували доступ до служби LastPass.» [11].

Провівши аналіз атак на засоби збереження паролів, доцільно провести порівняльний аналіз відомих засобів захищеного зберігання паролів.

1.3 Порівняльний аналіз відомих засобів захищеного зберігання паролів

Для якісного аналізу відомих засобів захищеного зберігання паролів варто розглянути, як засоби що існують на ринку протягом тривалого часу, оскільки вони здобули увагу користувачів завдяки гарному захисту та перевіреними роками функціональними можливостями, так і сучасні засоби, розробники, яких шукають не стандартні рішення, щоб закріпитися на ринку. Для аналізу було обрано 8 популярних менеджерів паролів [1]:

- NordPass [12];
- RoboForm [13];
- 1Password [14];
- Keeper [15];
- Dashlane [16];
- LastPass [17];

– Bitwarden [18];

– Enpass [19].

В таблиці 1.1 наведено результати аналізу обраних менеджерів паролів та їх властивості.

Таблиця 1.1 – Результати порівняльного аналізу менеджерів паролів

	RoboForm	1Password	Keeper	LastPass
Обмеження для безкоштовної версії	Відсутня можливість резервного копіювання в хмару	Тільки платна	Тільки для мобільного пристрою	Синхронізація між пристроями одного типу
Багатофакторна автентифікація	В платній версії	В платній версії	+	+
Біометричний вхід	+	В платній версії	+	+
Заповнення форм	+	В платній версії	+	+
Генератор паролів	+	В платній версії	+	+
Аварійний доступ	В платній версії	В платній версії	В платній версії	В платній версії
Надійність паролю	+	В платній версії	В платній версії	+
Платформи	Windows, Mac, iOS, Android, Linux	Windows, Mac, iOS, Android. Linux	Windows, Mac, iOS, Android, Linux	Windows, Mac, iOS, Android, Linux
Браузери	Chrome, Firefox, Safari, Edge, IE	Chrome, Firefox, Safari, Edge, Brave	Chrome, Firefox, Safari, Edge, IE, Opera	Chrome, Firefox, Safari, Edge, Opera

Продовження таблиці 1.1.

	Bitwarden	Enpass	NordPass	Dashlane
Обмеження для безкоштовної версії	Відсутній TOTP автентифікатор та можливість шифрування файлів	25 паролів для мобільної версії	Для одного пристрою	Для одного пристрою
Багатофакторна автентифікація	+	+	+	+
Біометричний вхід	+	+	+	+
Заповнення форм	+	+	+	+
Генератор паролів	+	+	+	+
Аварійний доступ	В платній версії	–	В платній версії	–
Надійність паролю	+	+	В платній версії	+
Платформи	Windows, Mac, iOS, Android, Linux	Windows, Mac, iOS, Android, Linux	Windows, Mac, iOS, Android, Linux	Windows, Mac, iOS, Android, Linux
Браузери	Chrome, Firefox, Safari, Edge, Brave, Opera, Vivaldi, Tor	Chrome, Firefox, Safari, Edge, Opera, Vivaldi	Chrome, Firefox, Safari, Edge, Brave, Opera	Chrome, Firefox, Safari, Edge, IE

З порівняльного аналізу, формалізованого у таблиці 1.1, можна зробити такі висновки:

– у всіх менеджерів паролів, окрім 1Password є безкоштовна версія;
– всі менеджери паролів мають функції багатофакторної автентифікації, біометричного входу, заповнення форм, генерації паролів і перевірку стійкості паролів;
– всі менеджери паролів, окрім Dashlane та Enpass, мають можливість аварійного доступу до облікового запису, хоча ця функція є тільки в платних версіях;
– всі менеджери паролів використовують симетричне шифрування AES з 256-бітним ключем, яке є більш швидким, але не має математичного доведення стійкості порівняно з асиметричними алгоритмами, що не дозволяє виключити ризик його зламу у майбутньому;
– всі менеджери паролів є кросплатформеними, хоча багато з них обмежені однією платформою в безкоштовній версії;
– всі менеджери паролів підтримують багато популярних браузерів;
– RoboForm має можливість резервного копіювання в хмару, але в платній версії.

Таким чином, розробка нового засобу захищеного зберігання паролів повинна покращувати шифрування та включати основні функціональні можливості, притаманні відомим засобам.

1.4 Постановка задачі

Виходячи з попередніх підрозділів, задачею даної роботи є покращення процесу захисту паролів, шляхом розробки засобу захищеного зберігання паролів, з функціями:

– багатофакторної автентифікації;
– автоблокуванням менеджеру паролів після певного часу бездіяльності;

– очищенням буферу обміну через певні проміжки часу;
– генерацією надійних паролів;
– відслідковуванням часу використання паролів, з метою їх регулярної заміни;
– використання асиметричного шифрування паролів, яке має математичне доведення стійкості порівняно з симетричним.

1.5 Висновки з розділу

У даному розділі було проведено аналіз методів парольної автентифікації, який дозволив визначити особливості перебігу цього процесу. Проаналізовано основні види атак на засоби зберігання паролів, що дозволило виділити основні моделі та вектори атак та сформулювати рекомендації щодо їх зменшення ризиків від їх впливу. Внаслідок виконаного порівняльного аналізу відомих засобів захисту паролів зроблено висновки щодо особливостей їх реалізації та спільних недоліків. Це дозволило виконати постановку задачі дослідження.

2 АРХІТЕКТУРА ЗАСОБУ ЗАХИЩЕНОГО ЗБЕРІГАННЯ ПАРОЛІВ

2.1 Узагальнена архітектура засобу

На рисунку 2.1 зображено узагальнену архітектуру засобу. Вона складається з 7 модулів:

– роботи з файлами;
– захист цілісності;
– захист конфіденційності;
– модуль шифрування;
– формування таблиці зі входами;
– взаємодії з користувачем;
– перевірки стійкості паролів.

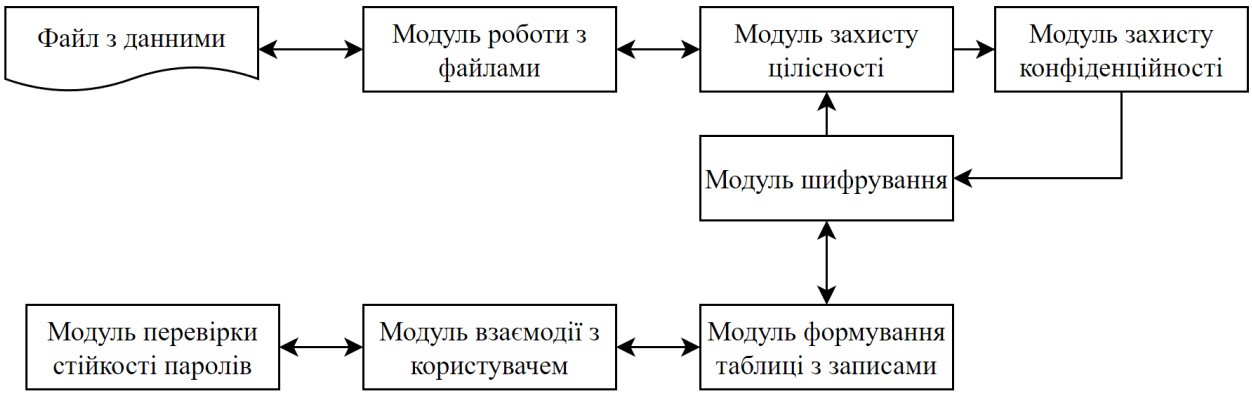


Рисунок 2.1 – Узагальнена архітектура засобу

Модуль роботи з файлами відповідає за роботу з файлом, у якому зберігаються дані користувача. Даний модуль зчитує інформацію з файлу та передає її до модуля захисту цілісності, в якому перевіряється чи не відбулось у файлі несанкціонованих модифікацій. Після того, як цілісність була підтверджена, відбувається автентифікація користувача у модулі захисту конфіденційності. Якщо користувача авторизовано, то наступним кроком дані передаються до модуля шифрування, де їх буде розшифровано та передано до модуля формування таблиці з записами. В даному модулі дані виводяться на екран в виді таблиці, з якою користувач може взаємодіяти. В модулі взаємодії

користувач може додавати нові записи, видаляти вже створені або ж змінювати дані в них. Також користувач може використати функцію генерації паролю та створити новий запис з даним паролем. Для використання записів користувачу надається можливість копіювання логіну та паролю. При ручному створенні, або зміні паролів вони передаються до модуля перевірки стійкості паролів, де відбувається перевірка паролів та виводиться відповідне повідомлення користувачу.

Коли користувач завершує роботу з програмою дані з таблиці зашифровуються у відповідному модулі, після чого передаються до модуля захисту цілісності та записуються у відповідний файл.

Після розробки загальної архітектури засобу необхідно розробити структуру основних модулів.

2.2 Структура модуля захисту конфіденційності

Для захисту модуля конфіденційності варто використати двофакторну автентифікацію за допомогою паролю та носія, на якому був створений файл. Якщо хоча б один з факторів не буде збігатись, то дані не будуть вірно розшифровані. На рисунку 2.2 зображено структуру даного модуля.

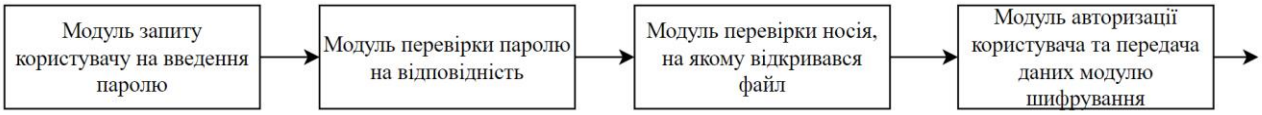


Рисунок 2.2 – Структура модуля захисту конфіденційності

Для створення структури модуля захисту конфіденційності, було виділено його основні процеси, для кожного з яких було створено окремий модуль (рис. 2.2):

– модуль запиту користувачу на введення паролю, який необхідний для того, щоб користувач авторизувався, після введення паролю він зчитується та передається наступному модулю;

– модуль перевірки паролю необхідний, щоб підтвердити перший фактор автентифікації користувача, після перевірки паролю дані передаються до наступного модуля;
– модуль перевірки носія, на якому відкривався файл є модулем перевірки другого фактору автентифікації, після виконання даного модуля;
– модуль авторизації користувача та передача даних модулю шифрування, даний модуль виконується останнім та забезпечує авторизацію користувача та передачу даних модулю шифрування.

Таким чином було розроблено структуру модуля захисту конфіденційності.

2.3 Обґрунтування вибору криптографічних алгоритмів

Всі менеджери паролів, які були проаналізовані у попередньому розділі використовували симетричне шифрування AES з 256-бітним ключем. Звісно дане шифрування виграє за швидкістю виконання у асиметричного, і його легко використовувати для шифрування великих файлів даних. Проте для шифрування паролів можна використати асиметричне шифрування. Асиметричне шифрування має такі переваги над симетричним [20]:

– конфіденційність передачі ключів;
– відсутня потреба в розв’язанні задачі передачі ключів;
– довіра і автентифікація.

Конфіденційність передачі ключів. У симетричному шифруванні обидві сторони повинні мати спільний секретний ключ, який вони використовують для шифрування та дешифрування повідомлень. Проблема полягає в безпечній передачі цього ключа між сторонами. У асиметричному шифруванні використовується публічний ключ для шифрування та приватний ключ для дешифрування. Публічний ключ може бути безпечно розповсюдженим, тоді як приватний ключ залишається виключно у власника. Це дозволяє забезпечити

конфіденційність передачі ключів без необхідності захищати секретний ключ від проникнення третіх осіб.

Відсутня потреба в розв’язанні задачі передачі ключів. Симетричне шифрування вимагає використання того ж самого ключа для шифрування і розшифрування повідомлень. Це означає, що кожен з учасників спілкування повинен мати копію секретного ключа, що може бути проблематично, особливо якщо кількість учасників зростає. У асиметричному шифруванні ж використовується публічний ключ для шифрування, і цей процес може бути виконаний всіма учасниками. Таким чином, немає необхідності передавати секретний ключ, що робить асиметричне шифрування більш ефективним у використанні ресурсів, а у випадку засобу захищеного зберігання паролів це усуває потребу в необхідності його захищеного зберігання. Більше того, якщо ключ захищати шируванням, то все одно доведеться звертатись до можливостей асиметричного шифрування, оскільки вирішення цієї задачі лише симетричними алгоритмами приведе або до необхідності зберігання ключа як константи в коді, що небезпечно з огляду на можливість зловмисника виконувати зворотну інженерію застосунка, або до нескінченного ланцюга із застосуванням нескінченної кількості одних ключів для шифрування інших ключів.

Довіра і автентифікація. Асиметричне шифрування може бути використане для забезпечення довіри та автентифікації сторін. Кожен користувач має унікальну пару ключів: публічний ключ, який він розповсюджує, і приватний ключ, який лишається виключно у нього. Коли сторони обмінюються шифрованими повідомленнями, вони можуть використовувати публічні ключі для підтвердження автентичності один одного та забезпечення недоступності третіх осіб. У симетричному шифруванні важче досягти такої рівня автентифікації та довіри без додаткових механізмів.

Серед алгоритмів асиметричного шифрування можна виділити RSA (Rivest-Shamir-Adleman). Даний тип шифрування є одним з найпоширеніших. Хоча раніше на нього був оформлений патент [21, 22], зараз даний алгоритм підтримується багатьма криптографічними бібліотеками та платформами, що

дозволяє легко інтегрувати його в різні системи та програми. RSA є одним з найбільш досліджених і використовуваних алгоритмів у криптографії. Він має витривалість щодо різних атак, таких як факторизація чисел та атаки на основі криптоаналізу.

Саме тому для шифрування паролів буде виконуватись за допомогою асиметричного шифрування RSA.

2.4 Структура модуля захисту цілісності

Для захисту цілісності варто використати контрольну суму (checksum) [23]. Контрольна сума – це число, що обчислюється з вмісту файлу або даних для перевірки їх цілісності. Дане число використовується для виявлення пошкоджень, або змін у файлі, або даних. Для перевірки цілісності файлу необхідно:

– обчислити контрольну суму за допомогою одного з відомих алгоритмів (MD5, SHA-1, SHA-256, CRC, або інших), тому в структурі модуля захисту потрібно передбачити модуль обчислення контрольної суми;
– зберегти контрольну суму або окремо від файлу, або ж вбудувати її в файл;
– перевірити контрольну суму, обчисливши її знову та порівнявши зі збереженою, якщо вони збігаються, то пошкоджень або змін не було, для чого потрібно використати окремий структурний модуль;
– забезпечити інтерфейси з іншими структурними елементами засобу, що породжує необхідність у блоках отримання даних зчитаних з файлу та блоку виведення результату перевірки контрольної суми.

На рисунку 2.3 зображено структуру модуля захисту цілісності.

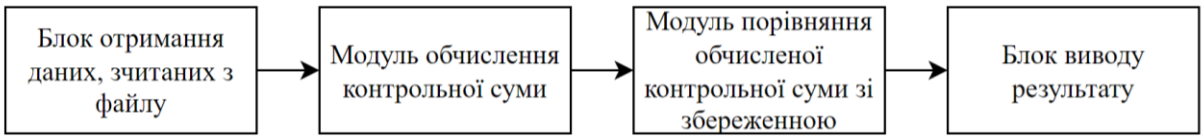


Рисунок 2.3 – Структура модуля захисту цілісності

Окрім контрольної суми, для забезпечення цілісності можна використовувати геш-функції або коди, що виявляють та виправляють помилки.

Геш-функції [25] приймають вхідні дані і генерують унікальний геш-код фіксованої довжини. Якщо навіть невеликі зміни вносяться до файлу, це призведе до значних змін в геш-коді. Геш-функції використовуються для перевірки цілісності файлу, порівнюючи геш-коди вихідного і отриманого файлу. Якщо геш-коди не співпадають, це свідчить про пошкодження або зміни файлу.

Коди, що виявляють та виправляють помилки [25] використовуються для виявлення та виправлення помилок, які можуть виникати під час передачі або збереження файлів. Ці коди включають додаткові біти інформації, які дозволяють виявити та виправити певні типи помилок. Наприклад, код Гемінга [25] виявляє та виправляє одиночні бітові помилки. Такі коди можуть виправити помилки в певному обсязі, але мають обмежену здатність до виявлення та виправлення помилок.

Порівнюючи ці засоби з контрольною сумою, можна зробити висновок, що геш-функції мають так само гарний рівень виявлення помилок, але працюють повільніше, а коди, що виявляють та виправляють помилки працюють не так точно та виправляють помилки, що може допомогти зловмиснику розшифрувати паролі.

Серед алгоритмів обчислення контрольної суми варто виділити CRC (циклічний надлишковий код) [23]. До його переваг можна віднести такі:

– проста структура, яку легко реалізувати, як в програмному, так і в апаратному забезпеченні;
– швидке виконання, яке потребує менше обчислювальних ресурсів, ніж інші алгоритми;
– ефективне використання пам’яті, оскільки для обчислення CRC не потрібно зберігати весь вміст даних, що перевіряються, обробляти дані можна блоками, або пакетами;

– висока здатність виявляти помилки, що дозволяє виявити такі помилки, як помилки бітів або шум при передачі даних, саме тому даний алгоритм використовується в багатьох системах, як Ethernet, Bluetooth, Wi-Fi, ZIP-архівування [23].

Саме тому для реалізації алгоритму захисту цілісності буде використано алгоритм обчислення контрольної суми CRC.

2.5 Структура модуля взаємодії з користувачем

В даному модулі користувач, який пройшов процес автентифікації, отримує доступ до розшифрованих даних, що сформовані в таблицю з записами. Серед функцій доступних користувачу є такі:

– додавання нових записів;
– видалення створених записів;
– редагування створених записів;
– генерування паролю, який може бути використаний для нового запису;
– копіювання логіну та паролю з певного запису.

На рисунку 2.4 зображено структуру модуля взаємодії з користувачем.

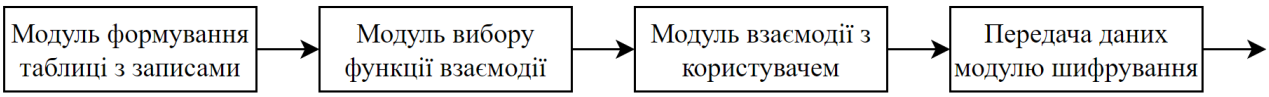


Рисунок 2.4 – Структура модуля взаємодії з користувачем

Для створення структури модуля взаємодії з користувачем, було виділено його основні процеси, для кожного з яких було створено окремий модуль (рис. 2.4):

– модуль формування таблиці з записами, який необхідний для того, щоб користувач міг взаємодіяти зі записами через графічний інтерфейс, після формування таблиці користувач отримує доступ до наступного модулю;

– модуль вибору функції взаємодії, так як користувачу необхідно буде обрати функцію для взаємодії з таблицею з записами, після обрання функції, її код буде переданий до наступного модуля;
– модуль взаємодії з користувачем, який отримавши код функції, що була обрана, виконає відповідну дію;
– передача даних модулю шифрування, для забезпечення конфіденційності даних користувача.

Таким чином було розроблено структуру модуля взаємодії з користувачем.

2.6 Висновки з розділу

У даному розділі було розроблено узагальнену архітектуру засобу, що дозволило на її основі розробити структури основних модулів засобу, які будуть використані для розробки алгоритмів в наступному розділі. Також було проведено обґрунтування вибору криптографічних алгоритмів та алгоритму захисту цілісності, наведено їх переваги та особливості реалізації. В наступному розділі варто розробити алгоритм роботи засобу на основі розроблених структурних схем.

3 АЛГОРИТМИ РОБОТИ ЗАСОБУ ЗАХИЩЕНОГО ЗБЕРІГАННЯ ПАРОЛІВ

3.1 Узагальнений алгоритм роботи засобу

Для того, щоб розробити застосунок захищеного зберігання паролів, необхідним є розробка його узагальненого алгоритму та алгоритмів роботи модулів. В даному розділі буде розроблено узагальнений алгоритм роботи засобу та її основних модулів на основі структурних схем, розроблених в попередньому розділі.

Узагальнений алгоритм роботи засобу зберігання паролів починається з того, що користувачу необхідно обрати одну з опцій: відкрити файл в якому збережені паролі, якщо він вже користувався застосунком, або створити новий файл для збереження паролів. Коли користувач обрав опцію сформується таблиця з записами, якщо користувач створив новий файл, то таблиця буде пустою. Після формування таблиці користувач переходить до етапу взаємодії з таблицею, в якому може створювати нові записи та редагувати, або видаляти створені. Також йому доступні функції генерації паролів, копіювання логінів та паролів в буфер обміну для їх використання та блокування робочого простору, якщо він відходить від робочого місця. Як тільки користувач бажає завершити роботу з застосунком, він може вибрати дві опції: зберегти зміни у файлі, або ж закрити програму відмінивши їх. Якщо користувач зберігає зміни, то перш за все відбувається їх шифрування, після чого підраховується їх контрольна сума та відбувається запис зашифрованих даних у файл. Після чого користувач може завершити роботу з програмою.

На рисунку 3.1 зображено узагальнений алгоритм роботи засобу зберігання паролів.

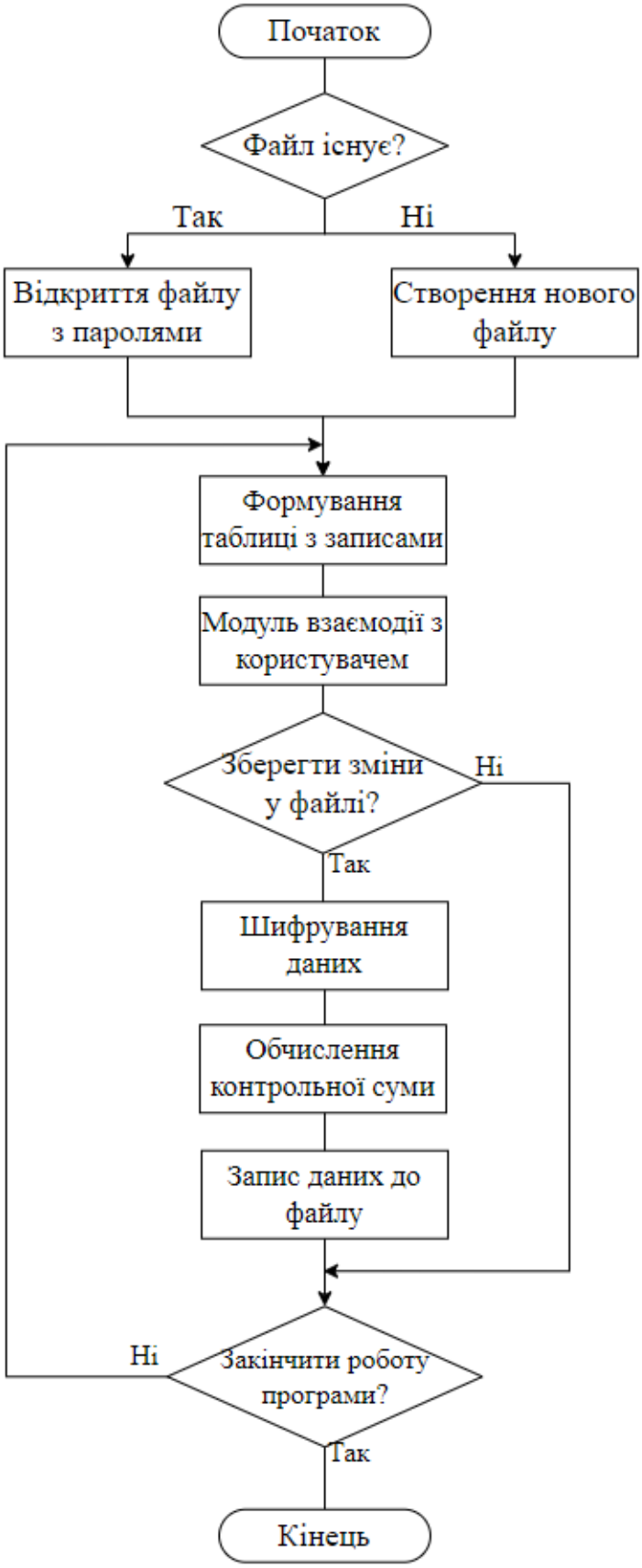


Рисунок 3.1 – Узагальнений алгоритм роботи засобу зберігання паролів

Після розробки узагальненого алгоритму, варто розробити алгоритми основних модулів засобу.

3.2 Алгоритм захисту конфіденційності

В даному підрозділі необхідно розробити алгоритм захисту конфіденційності. Для захисту конфіденційності обрано двофакторну автентифікацію: на основі паролю та носія на якому був створений файл.

На рисунку 3.2 зображено алгоритм захисту конфіденційності.

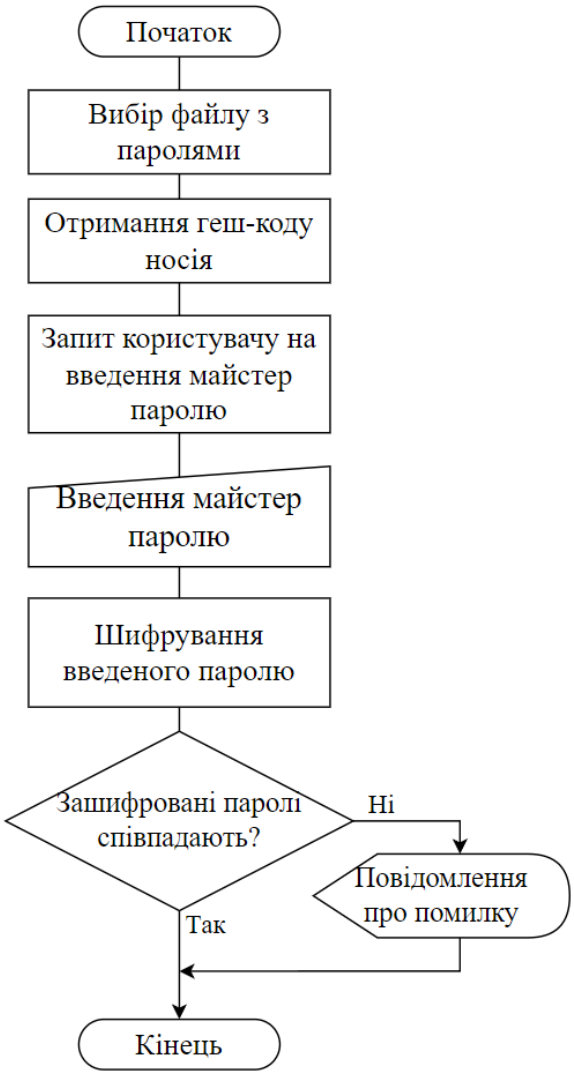


Рисунок 3.2 – Алгоритм захисту конфіденційності

Алгоритм захисту конфіденційності починається з того, що користувачу необхідно вибрати файл, в якому збережені його паролі. Після чого відбувається отримання геш-коду носія, на якому відкривався даний файл, та запит користувачу на введення паролю. Наступним кроком, відбувається шифрування введеного паролю та перевірка даного паролю з тим, що використовувався при

створенні файлу. Якщо введений пароль не відповідає створеному, або файл був відкритий не на тому носії, то виведеться повідомлення про помилку.

Таким чином було розроблено алгоритм захисту конфіденційності.

3.3 Алгоритм роботи модуля криптографічного захисту

В даному підрозділі необхідно розробити алгоритм роботи модуля криптографічного захисту. Як вже було обґрунтовано в попередньому розділі необхідним є використання шифрування RSA для забезпечення захисту паролів.

На рисунку 3.3 зображено алгоритм роботи модуля криптографічного захисту.

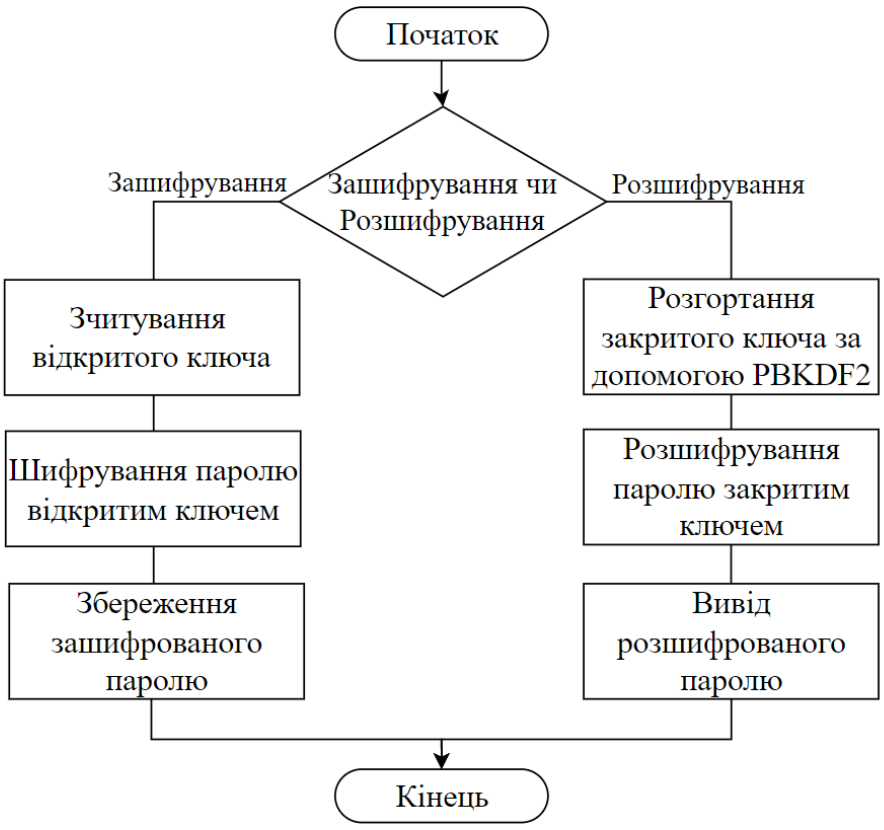


Рисунок 3.3 – Алгоритм роботи модуля криптографічного захисту

Для початку необхідно обрати необхідну опцію: зашифрування паролю або розшифрування паролю. При зашифруванні виконується зчитування відкритого ключа. Після чого відбувається шифрування паролю та збереження його у файл. При розшифруванні виконується розгортання ключа за допомогою алгоритму PBKDF2. Після чого відбувається розшифрування паролю та його вивід, який

залежить від паролю введеного користувачем при автентифікації, якщо користувач ввів не правильний пароль, то ключ на основі PBKDF2 розгорнеться також не правильний і замість паролю, користувач отримає нечитабельний набір символів.

Таким чином було розроблено алгоритм роботи модуля криптографічного захисту.

3.4 Алгоритм захисту цілісності

В даному підрозділі необхідно розробити алгоритм захисту цілісності. Як вже було обґрунтовано в попередньому розділі необхідним є алгоритму CRC для підрахування контрольної суми.

На рисунках 3.4-3.5 зображено алгоритм роботи модуля криптографічного захисту.

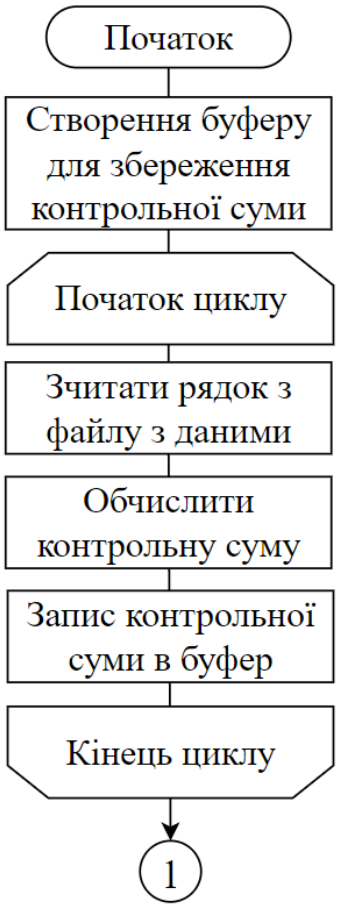


Рисунок 3.4 – Алгоритм захисту цілісності ч. 1

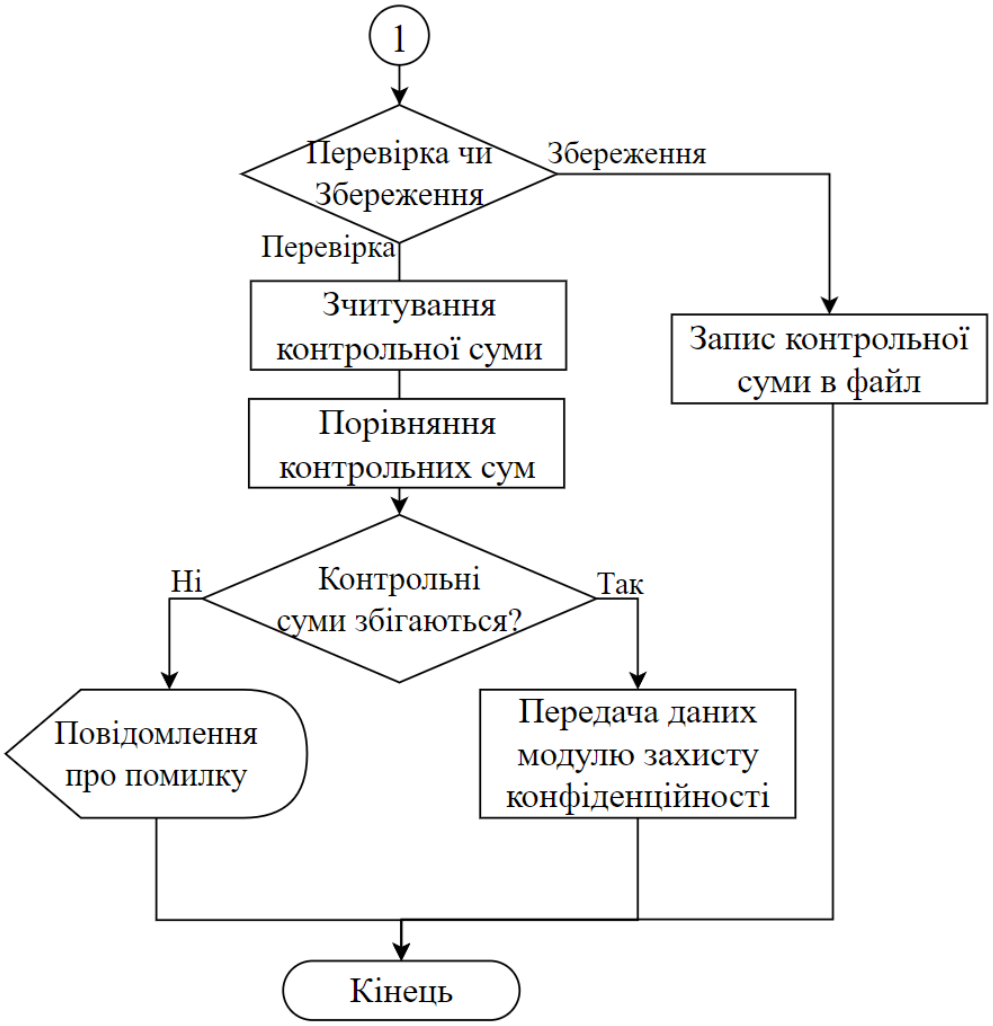


Рисунок 3.5 – Алгоритм захисту цілісності ч. 2

Для початку необхідним є створення буферу для збереження контрольної суми, після чого починається цикл, в якому відбувається порядкове зчитування даних, обчислення контрольної суми цих даних та запис їх у буфер. По завершенню виконання циклу, необхідно обрати дана сума обчислювалась для збереження, чи для порівняння з контрольною сумою, яка обчислювалась під час збереження файлу. Якщо контрольна сума обчислювалась для збереження, то вона просто записується у файл. Якщо вона обчислювалась для порівняння, то відбувається зчитування суми, яка обчислювалась під час збереження файлу, та порівняння цих контрольних сум. Якщо контрольні суми збігаються, то дані передаються модулю захисту конфіденційності, де буде відбуватись

автентифікація користувача. Якщо вони не збігаються, то виводиться повідомлення про помилку.

Таким чином було розроблено алгоритм захисту цілісності.

3.5 Алгоритми модуля взаємодії з користувачем

В даному підрозділі необхідно розробити алгоритми роботи модуля взаємодії з користувачем. Даний модуль має складатися з декількох алгоритмів, які відповідають функціям взаємодії користувача з засобом.

На рисунку 3.6 зображено алгоритм роботи функції додавання нових записів.

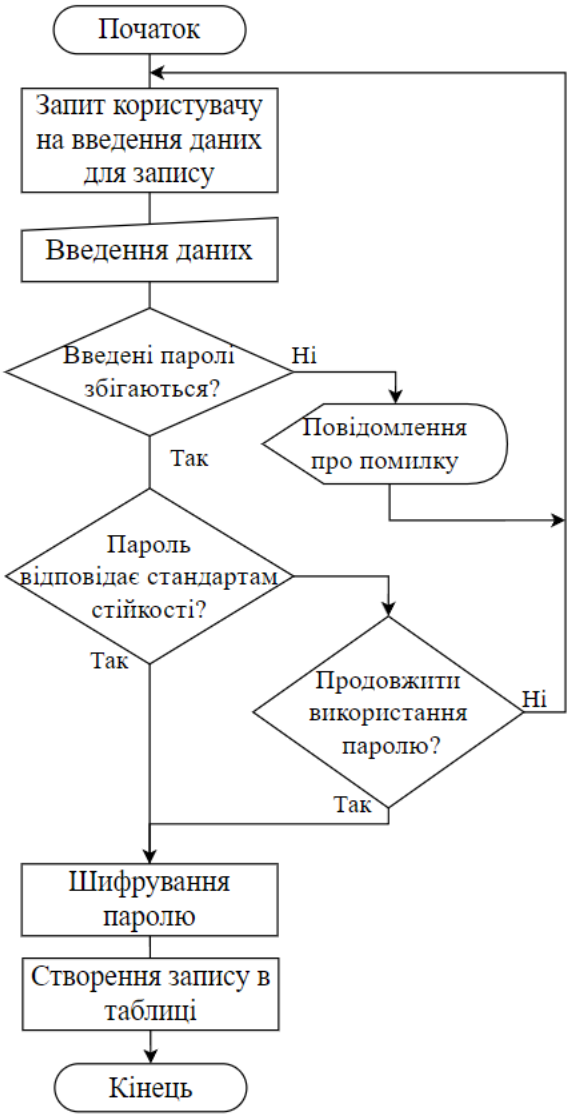


Рисунок 3.6 – Алгоритм роботи функції додавання нових записів

Для початку виконується запит користувачу на введення даних. Якщо паролі, які ввів не збігаються, то користувачу виводиться повідомлення про помилку після чого виконується новий запит на введення даних. Якщо паролі збігаються, то далі йде перевірка паролю на відповідність стандартам стійкості. Якщо пароль її не проходить, то користувачу надається дві опції: продовжити використання нестійкого паролю, чи ввести його заново. Якщо користувач обирає ввести пароль заново, то засіб знову відправляє йому запит на введення. Якщо він обирає продовжити використання паролю, то як і в випадку коли пароль пройшов перевірку, він шифрується, після чого створюється запис в таблиці.

Таким чином було розроблено алгоритм роботи функції додавання нових записів.

Наступним алгоритмом буде розроблено алгоритм роботи функції редагування створених раніше записів (рис. 3.7).

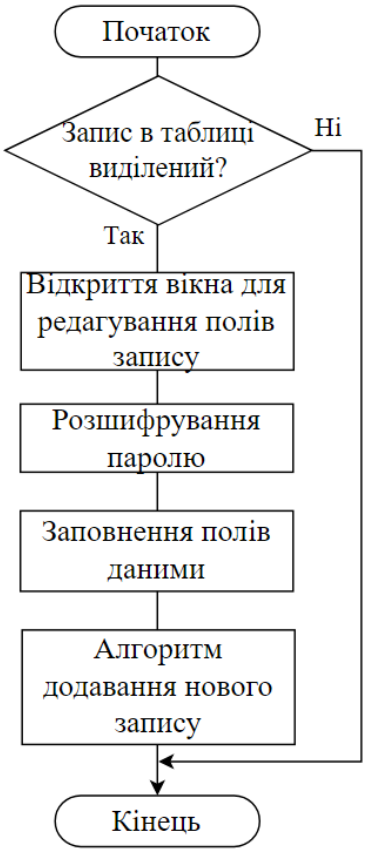


Рисунок 3.7 – Алгоритм роботи функції редагування створених раніше записів

Для початку відбувається перевірка на те, чи виділений запис у таблиці. Якщо він не виділений, то редагування не можливе. Якщо він виділений, то відкривається вікно з полями для запису даних. Після чого відбувається процес розшифрування паролю даного запису та заповнення всіх полів. Далі алгоритм ідентичний до алгоритму додавання нового запису.

Таким чином було розроблено алгоритм роботи функції редагування створених раніше записів.

На рисунку 3.8 зображено алгоритм роботи функції видалення записів.

Рисунок 3.8 – Алгоритм роботи функції видалення записів

Для початку відбувається перевірка на те, чи виділений запис у таблиці. Якщо він не виділений, то видалення не можливе. Якщо він виділений, то відкривається вікно з запитанням: «Ви впевнені, що хочете видалити виділений запис?» з варіантами «так» або «ні». Якщо користувач відмовляється від

видалення, то процес закінчується та повертає користувача до вікна в таблицею. Якщо він підтверджує видалення, то засіб визначає номер рядка в даного запису в таблиці та видаляє запис зі списку по цьому номеру. Після чого відбувається оновлення таблиці з записами.

На рисунку 3.9 зображено алгоритм роботи функції генерування паролів.

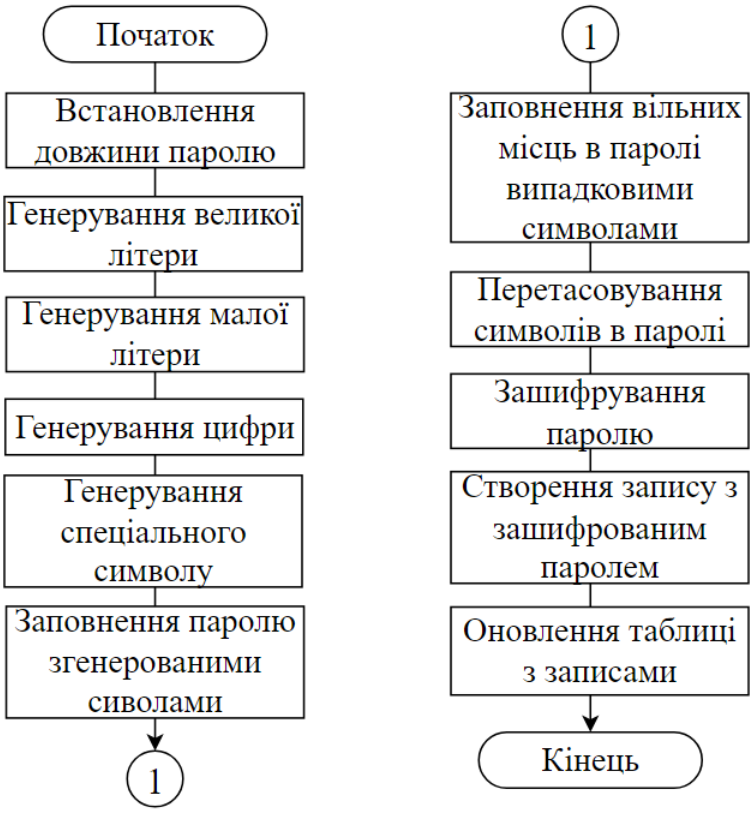


Рисунок 3.9 – Алгоритм роботи функції генерування паролів

На початку роботи алгоритму необхідно встановити довжину пароля, що буде згенеровано. Потім відбувається генерування великої та малої літери, цифри та спеціального символу. Далі згенеровані паролі формують першу частину паролю. Для формування другої частини паролю та заповнення всіх вільних місць генеруються випадкові символи. Після чого дві частини паролю об’єднуються та відбувається перетасування символів місцями. Наступним кроком відбувається зашифрування паролю, створення запису з зашифрованим паролем та оновлення таблиці з записами.

Таким чином було розроблено алгоритм роботи функції генерування паролів.

Наступним алгоритмом буде розроблено алгоритм роботи функції копіювання логіну та паролю з певного запису (рис. 3.10).

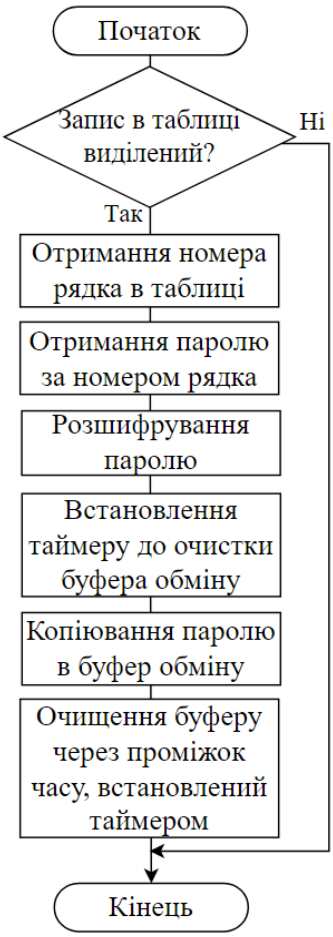


Рисунок 3.10 – Алгоритм роботи функції копіювання паролю

На початку роботи алгоритму необхідно перевірити чи виділений запис в таблиці, якщо ні, то копіювання неможливе. Якщо запис виділений, то відбувається отримання номера рядка запису в таблиці, після чого отримується пароль зі списку за номером рядка. Далі відбувається розшифрування паролю та встановлення таймеру до очищення буфера обміну. Потім пароль копіюється в буфер обміну та запускається таймер. Коли час закінчиться буфер обміну буде очищений. Алгоритм функції копіювання логіну аналогічний алгоритму копіювання паролю, за винятком процесу розшифрування. Так, як в засобі шифруються тільки паролі.

3.6 Висновки з розділу

У даному розділі було розроблено узагальнений алгоритм роботи засобу та алгоритми роботи основних модулів, до яких належать: алгоритм захисту конфіденційності, алгоритм захисту цілісності, алгоритм роботи модуля криптографічного захисту та алгоритми функцій додавання нових записів, видалення та редагування вже створених записів, генерування паролю, який може бути використаний для нового запису та копіювання логіну та паролю з певного запису. Дані алгоритми були розроблені на основі структурних схем, розроблених в попередньому розділі й будуть використані для розробки засобу захищеного зберігання паролів у наступному розділі.

4 ТЕСТУВАННЯ ЗАСОБУ ЗАХИЩЕНОГО ЗБЕРІГАННЯ ПАРОЛІВ

4.1 Обґрунтування засобів розробки

Перед початком розробки та тестування засобу необхідно обґрунтувати вибір інструментів для розробки застосунку.

Засіб захищеного зберігання паролів повинен бути сумісним з Windows, Linux та MacOS, саме тому необхідним є використання мови програмування для розробки кросплатформених застосунків. Серед даних мов програмування варто виділити Java, C# та Python.

Java [26] є мовою високого рівня, серед її основних переваг варто виділити кросплатформеність, про яку йшла мова вище, у Java дана функція дозволяє програмам, написаним на цій мові, працювати на системах, як Windows, Linux та MacOS, без необхідності переписування коду. Це досягається за допомогою віртуальної машини – JVM, яка виконує програми написані на Java. Також Java використовує об’єктно-орієнтований підхід, що дозволяє моделювати реальні об’єкти за допомогою класів та об’єктів. Даний підхід полегшує реалізацію складних програм. Ще однією з переваг Java є вбудована система безпеки, яка має функції: перевірки байт-коду, менеджер безпеки, який можна використовувати для контролю доступу до файлів, мережевих підключень та інших потенційно небезпечних операцій, та механізм завантаження класів, що забезпечує безпеку, перевіряючи цілісність класів перед їх завантаженням. Це запобігає неавторизованій модифікації класів, гарантуючи, що виконується лише перевірений код. Автоматичне збирання сміття є перевагою Java. Даною функцією керує JVM. Коли існує об’єкт, який не відноситься до жодного класу, JVM автоматично видаляє його з програми, тому не потрібно писати додатковий код. Java підтримує багатопоточність. За допомогою цього можна запускати більше одного потоку одночасно. Вони використовують спільну пам’ять для підвищення ефективності та продуктивності програми. Потоки виконуються незалежно один від одного. Java є розподіленою мовою програмування, так як

вона має механізм для обміну даними та програмами між декількома комп’ютерами, що підвищує продуктивність і ефективність системи. Також Java підтримує розширюваність, що дозволяє розширювати функціональність програми за допомогою створення власних класів, інтерфейсів та бібліотек. Java має широкий набір бібліотек, фреймворків та середовищ розробки, найбільш популярними є IntelliJ IDEA, Eclipse та NetBeans.

C# [27] є мовою високого рівня. Вона вважається гарним вибором для розробки десктопних програм для системи Windows та навіть розробки ігор, оскільки ігровий рушій Unity побудований на C#. Дана мова програмування також є об’єктно-орієнтованою. C# розроблений, як частина платформи .NET, що забезпечує кросплатформеність та забезпечує розробників різними середовищами виконання, такими як Common Language Runtime, що використовується для запуску та компіляції десктопних та веб-додатків та підтримує системи Windows, Linux, MacOS, Android та iOS, та Mono Runtime, який використовується для додатків, що мають критичні вимоги до продуктивності, наприклад ігор або мобільних програм. Інтероперабельність є ще однією перевагою .NET. Дана функція дозволяє C# взаємодіяти з програмами, що написані на сумісних мовах, таких як C++, F#, Visual Basic та Windows PowerShell. У C# також є збирач сміття.

Python [28] – це мова високого рівня, яка також є об’єктно-орієнтованою. Python не пов’язаний з якоюсь певною галуззю. Дана мова використовується у багатьох сферах, таких як веб-розробка, аналіз даних, машинне навчання, DevOps і системне адміністрування, автоматизоване тестування та прототипування програмного забезпечення. На рисунку 4.1 зображено результати опитування розробників у 2022 році [29]. Python не залежить від платформи, розробник має можливість запускати той самий вихідний код у різних операційних системах, будь то Windows, MacOS або Linux. Портативність досягається за рахунок байт-коду та віртуальної машини – PVM, які служать посередниками між розробником і фактичним процесором, що виконує програму. Також, Python може бути об’єднаний з іншими мовами програмування

за допомогою розширень Cython для мови C, Gython для мови Go, Jython для мови Java та IronPython для .NET. Python є лідером в глибинному навчанні завдяки колекції бібліотек, таких як TensorFlow, Keras та PyTorch.

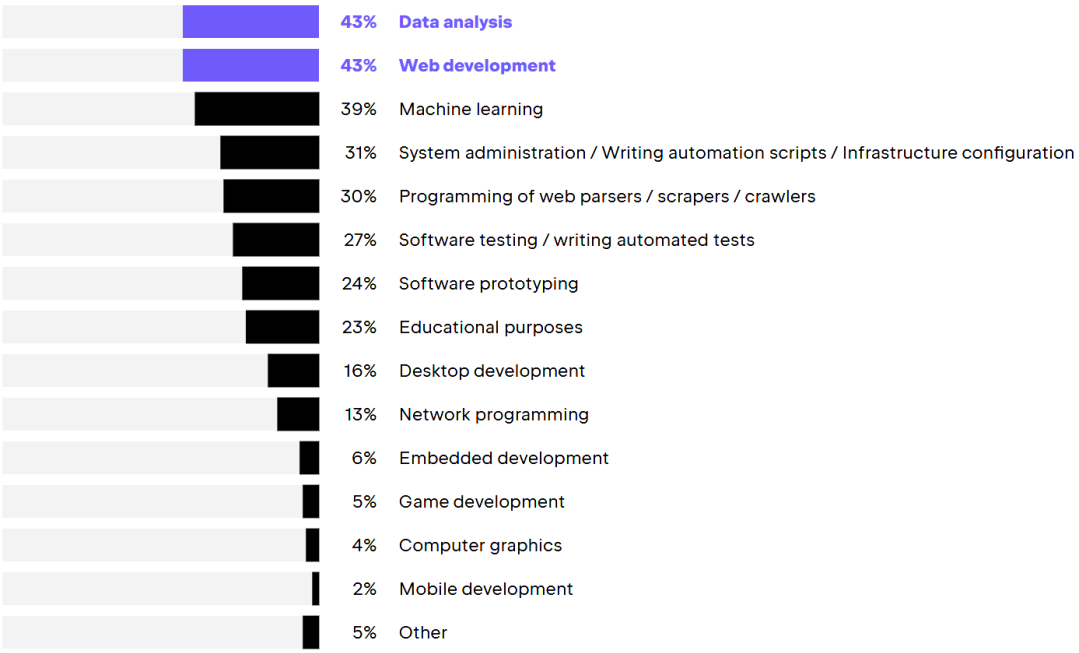


Рисунок 4.1 – Результати опитування розробників, щодо цілей використання Python

Python має велику кількість фреймворків для тестування, що покращують процес перевірки якості коду, серед них варто виділити: PyUnit, pytest, Locust, Splinter та Hypothesis [28]. Як вже зазначалось раніше Python використовують для DevOps, так як багато скріптів були написані на Python. Серед інструментів DevOps, що написані на Python можна виділити: SCons та PyBuilder, Ansible та Buildbot.

Для розробки засобу захищеного зберігання паролів варто використати мову програмування Java. Якщо порівнювати Java з Python, то Python компілює байт-код для запуску на віртуальній машині під час виконання, тоді як Java робить усі перетворення заздалегідь. Python динамічно типізується, а Java дотримується статичної типізації, коли тип присвоюється значенню раз і назавжди. Python перевершує Java за швидкістю розробки, оскільки дана мова має чіткий синтаксис і потребує менше рядків коду. У свою чергу, Java є більш

швидкою та ефективною, коли справа доходить до виконання коду. Вона використовує механізм компіляції точно перед виконанням (JIT-компіляція), який підвищує продуктивність програми. Що буде перевагою Java, так як в застосунку будуть шифруватись паролі методом RSA, який має низьку швидкість виконання. До того ж, Python має широкий спектр використання, який не є потрібним для розробки даного засобу.

Якщо порівнювати Java з C#, то перевагами Java є кросплатформеність, яка дозволяє програмам працювати на системах Windows, Linux та MacOS, що відповідає технічним вимогам. C# є мовою для розробки програм для системи Windows, а кросплатформеність у ній досягається тільки інтеграцією .NET. Також до переваг Java варто віднести вбудовану систему безпеки, переваги якої наведені вище.

Для кращого зображення порівняння варто створити порівняльну таблицю цих мов програмування (табл. 4.1).

Таблиця 4.1 – Результати порівняльного аналізу мов програмування

	Java	C#	Python
Кросплатформеність	+	Тільки з підключенням .NET	+
Мова високого рівня	+	+	+
Об’єктно-орієнтованість	+	+	+
Вбудована система безпеки	У Java має більший функціонал	+	+
Збирач сміття	+	+	+
Багатопоточність	+	+	+
Статична типізація	+	+	-
Простий синтаксис	-	-	+
JIT-компіляція	+	+	-
Вузьке направлення, що відповідає цілям даного засобу	+	-	-

Саме тому для розробки засобу захищеного зберігання паролів обрано мову програмування Java.

Після вибору мови програмування необхідним є вибір середовища розробки (IDE). Найбільш популярними є IntelliJ IDEA, Eclipse та NetBeans. Необхідно розглянути їх переваги.

IntelliJ IDEA [30] вважається однією з найкращих IDE для розробки Java [32]. Кожен аспект IntelliJ IDEA розроблено для підвищення продуктивності розробника. Дане середовище забезпечує розумне завершення коду. Базове завершення коду пропонує назви класів, методів і ключових слів у межах видимості. Однак, інтелектуальне завершення коду пропонує лише ті типи, які очікуються в цьому конкретному контексті. Також IntelliJ IDEA має ефективний та зручний дизайн, що робить розробку більш продуктивною. IntelliJ IDEA також розуміє інші мови, такі як HTML, Javascript, SQL, Kotlin тощо, і надає допомогу в інтелектуальному кодуванні для них. Дане середовище пропонує безліч плагінів, які можна інтегрувати в редактор коду. Також можна використовувати плагіни інших розробників. IntelliJ IDEA пропонує чудову підтримку контролю версій (підтримка Git), що значно полегшує зберігання та обмін кодом.

Eclipse [32] – це безкоштовна Java IDE із відкритим кодом. Те, що робить Eclipse унікальним, це його хмарна версія. Eclipse IDE випускається у двох версіях: десктопна та хмарна. Хмарна версія Eclipse дозволяє розробникам кодувати у веб-браузері та зберігати свій код у хмарі. Окрім хмарної підтримки, це також дозволяє розробникам створювати власні функції. Розробники можуть використовувати середовище розробки плагінів для створення власних функцій, і це робить Eclipse відмінною від інших IDE. Дане середовище розробки має великий набір плагінів, що дозволяє розробникам налаштовувати функціональні можливості для розробки програм. Однак, воно може працювати повільно після встановлення кількох плагінів. Керування пам’яттю в Eclipse не дуже добре. Існує ризик зіткнутися з деякими збоями під час роботи з кількома робочими областями.

Apache NetBeans [33] – ще одна IDE для мови програмування Java. Це IDE з відкритим кодом. Дане середовище розробки забезпечує функцію виділення вихідного коду синтаксично та семантично. NetBeans також дозволяє легко рефакторити код за допомогою ряду зручних і потужних інструментів. NetBeans підтримує розробку всіх типів програм Java, включаючи JavaFX, Java SE, Java ME тощо. Воно містить усі модулі, необхідні для розробки Java, за одне завантаження, що дозволяє користувачеві негайно розпочати роботу над будь-яким проектом Java. Ще однією з переваг NetBeans є те, що на ньому легше запускати серверні програми порівняно з іншими IDE. Дане середовище можна легко інтегрувати з серверами веб-додатків, такими як Tomcat і GlassFish. А також добре працює з Git та іншими системами контролю версій. Однак, процес компіляції та виконання програм відбувається повільніше, ніж в інших IDE.

Для кращого зображення порівняння варто створити порівняльну таблицю цих середовищ розробки (табл. 4.2).

Таблиця 4.2 – Результати порівняльного аналізу середовищ розробки

	IntelliJ IDEA	Eclipse	Apache NetBeans
Розумне завершення коду	Найкраще серед всіх IDE	+	+
Зручний дизайн	+	+	+
Підтримка Git	Найкраща серед всіх IDE	+	+
Безкоштовність	З обмеженим функціоналом	+	+
Хмарна версія	-	+	-
Підтримка плагінів	+	+	+
Висока швидкодія	+	-	-

Для розробки засобу варто обрати IntelliJ IDEA, так як дане середовище є зручним у використанні та не має проблем з ефективністю виконання, як у інших IDE.

4.2 Основні семантичні одиниці програмного коду

Так як застосунок має графічний інтерфейс, то необхідним було створення класів для кожного вікна програми. Окрім цих класів потрібно було створити класи для реалізації шифрування, підрахунку контрольної суми та генерації й перевірки стійкості паролю. А також створення класу для формування об’єкту з даними. Розглянемо ці класи, їх методи та змінні.

Першим розглянемо клас PasswordData.java, який використовується для формування об’єкту з даними. Списком даних об’єктів обробляються записи в застосунку.

В даному класі створено поля з даними: рядкові title, userName, url та notes, і масив байтів password, в якому зберігається пароль. Також в цьому класі створено конструктор та гетери й сетери для всіх полів з даними.

Наступним доцільно проаналізувати клас PasswordCheck.java, в якому відбувається перевірка паролю на стійкість.

В даному класі створено 5 булевих змінних яким було встановлено значення «false». Кожна з них відповідає за те, чи відповідає пароль параметрам стійкості. Для перевірки наявності в паролі великої та малої літери й цифр використовуються змінні типу Pattern та Matcher. Змінна типу Pattern встановлює патерн на перевірку символів, а Matcher перевіряє символи з паролю на збіг з встановленим патерном, після чого результат перевірки записується в булеву змінну. Для перевірки на наявність спеціальних символів у паролі було створено змінну CHARACTERS рядкового типу, в якій перераховано всі спеціальні символи, що доступні для створення паролю. Після перевірки великих та малих літер і цифр, запускається цикл, який перевіряє чи є в паролі хоча б один спеціальний символ зі змінної CHARACTERS. Коли знаходиться перший символ, робота циклу переривається та булева змінна отримує значення «true». Наступною йде перевірка паролю на довжину, якщо його довжина принаймні 8 символів, то булева змінна змінює своє значення на «true». Після проведення всіх

перевірок булеві змінні перемножаються та отримане значення повертається як відповідь.

Наступним буде проаналізовано клас PasswordGenerator.java, в якому відбувається генерування паролю.

Для початку у класі генерування паролів створюється змінна CHARACTERS рядкового типу, в якій перераховано всі символи, що можуть використовуватись для створення паролю, а також змінна PASSWORD_LENGTH цілочисельного типу, якій присвоєно значення 10, що відповідає за довжину паролю. У класі є метод PassGen, що має тип списку об’єктів типу PasswordData, саме в такому вигляді записи зберігаються у програмі. При виклику даного метода серед його параметрів необхідно вказати список, який наразі зберігає дані з записами та файл, який містить в собі відкритий ключ для зашифровування паролю, який буде згенеровано. Після виклику методу створюється змінна рядкового типу, якій буде присвоєно результат виконання методу GeneratePass, який одразу ж викликається. Після виклику даного методу створюється список елементів символьного типу passwordCharacters, довжиною PASSWORD_LENGTH, з яких буде складатись пароль. Після чого за допомогою методу SecureRandom виконується почергове генерування великої літери, малої літери, цифри та спеціального символу зі змінної CHARACTERS. Згенеровані символи додаються в список passwordCharacters, після чого відбувається генерування шести випадкових символів зі змінної CHARACTERS, які одразу додаються до списку. Враховуючи те, що при генерації перші чотири символи завжди мають визначений тип, було прийнято рішення перетасувати поряд символів за допомогою методу Collections.shuffle. Після даної дії за допомогою методу StringBuilder символи збираються у рядок та повертаються до методу PassGen. При поверненні до даного методу рядку password присвоюється значення згенерованого паролю. Потім створюється змінна типу масиву байтів encPass, в яку буде записано зашифрований пароль. Після чого відбувається зашифровування паролю методом encrypt з класу RSAEncDec. Для цього в метод передається пароль, що

збережений у змінній password та файл, в якому зберігається публічний ключ. Коли пароль зашифровано він записується у змінну encPass, після чого створюється список з даними того ж типу, що передавався в метод PassGen, але розміром на 1 більше, щоб вмістити запис зі зашифрованим паролем. Потім дані зі старого списку переносяться в новий та додається новий запис із зашифрованим паролем. Після чого даний метод повертає новий список з записами.

Наступним проведемо аналіз класу CRC.java, в якому відбувається обрахування контрольної суми та збереження її у файл.

У даному класі існує два методи calculateChecksum типу long для підрахування контрольної суми та метод saveChecksumToFile для збереження контрольної суми для подальшої перевірки перед автентифікацією. При виклику методу calculateChecksum йому необхідно передати рядок у якому збережено шлях до файла, контрольну суму якого треба перевірити. Після чого створюється змінна crc для використання методів з класу CRC32. Далі створюється масив байтів buffer, за допомогою якого буде виконуватись поблочне зчитування даних з файлу, так як алгоритм CRC дозволяє зчитувати дані і підраховувати контрольну суму блоками, а також змінна цілочисельного типу bytesRead, в яку будуть зберігатись зчитані дані. Далі починається робота масиву, який поблоково зчитує дані та записує їх у зміну bytesRead. Коли блок даних зчитано викликається метод update з класу CRC32, який виконує підрахування контрольної суми. Цикл виконується до тих пір поки в файлі є дані, які можна зчитати. По завершенню циклу метод повертає змінну типу long.

Враховуючи те, що це застосунок з графічним інтерфейсом і взаємодія користувача з програмою відбувається за допомогою пунктів меню у головному вікні програми, варто розглянути клас головного вікна програми та прослуховувачів, що відповідають за пункти меню. MainWindow наслідується від класу JDialog та має в собі оголошені поля з основними елементами головного вікна та 4 методи: MainWindow, який використовується для виклику й створення головного вікна, menuBarCreation для формування пунктів меню,

table для формування та оновлення таблиці з даними та actionMenuCreation, в якому створюються прослуховувачі для елементів меню.

Розглянемо метод menuBarCreation. Даний метод, як і методи table та actionMenuCreation, викликається методом MainWindow. У цьому методі першим кроком створюється об’єкт JMenuBar з ім’ям menuBar. Саме до цього об’єкту будуть прикріплятися інші елементи меню. Потім створюються об’єкти JMenu: file, entry та tools, які потім додаються до об’єкта menuBar. Далі ініціалізуються об’єкти JMenuItem, що були оголошені в класі MainWindow, які потім прикріпляються до відповідних об’єктів file, entry або tools. Коли об’єкти JMenuItem прикріпляються деякі з них блокуються методом setEnabled(false), для того щоб користувач не мав до них доступу, поки не виконаються умови, такі як відкриття або створення нового файлу з записами, або виділення комірки таблиці. Після чого об’єкт menuBar додається до головного вікна командою setJMenuBar(menuBar).

Наступним розлгянемо метод table. Перш за все, у даному методі створюється об’єкт DefaultWorkSheetTableModel з ім’ям tableModel, з класу DefaultWorkSheetTableModel, який наслідується від класу AbstractTableModel і формує модель таблиці з записами, які передаються списком об’єктів типу PasswordData. Далі створюється об’єкт ListSelectionModel з ім’ям selectionModel, який отримує модель вибору виділення об’єкту table1, який був оголошений в класі MainWindow. Потім для даної моделі встановлюється прослуховувач, який перевіряє таблицю на наявність виділеної комірки. Коли комірка виділяється об’єкти меню copyUserNameItem, copyPasswordItem, editEntryItem та deleteEntryItem становляться доступними для вибору, так як тільки при виборі комірки відкривається можливість копіювання логіну та паролю з запису та редагування й видалення запису. Після чого доступними становяться елементи addEntryItem, saveFileItem, lockItem та generatePasswordItem, так як якщо таблиця сформована, то отже до неї можна додавати нові записи, зберігати її у файл, блокувати робочий простір та генерувати паролі. Далі для об’єкту table1

встановлюється модель таблиці tableModel та встановлюється мінімальна ширина одного зі стовбців таблиці на ширину 150 пікселів.

Метод actionMenuCreation містить в собі прослуховувачі для кожного об’єкта меню.

Коли обирається елемент меню «Копіювати пароль», першим кроком створюється змінна selectedRow цілочисельного типу якій присвоюється значення рядка в таблиці, який на даний момент є виділеним. Потім створюється масив байтів bytePass, в який записується значення зашифрованого паролю зі списку з записами. Далі оголошується змінна рядкового типу password і викликається метод decrypt() з класу RSAEncDec, в який передається зашифрований пароль та файл в якому зберігається закритий ключ. Після розшифрування паролю він повертається в рядковому виді та записується в змінну password. Далі створюється об’єкт StringSelection з ім’ям stringSelection в який передається рядок password, який містить пароль. Далі створюється об’єкт Clipboard з ім’ям clipboard. Клас Clipboard використовується для роботи з буфером обміну. Потім clipboard отримує доступ до буфера обміну системи за допомогою методу getSystemClipboard після отримання доступу до інструментів системи за допомогою методу getDefaultToolkit. Далі у буфер обміну встановлюється значення з stringSelection за допомогою метода setContents. Після встановлення буферу, створюється об’єкт Timer, якому присвоюється значення 10000 мілісекунд, що еквівалентно 10 секундам. Цього часу має бути достатньо для введення паролю. Після закінчення роботи таймеру буфер обміну очищується встановленням в нього об’єкту StringSelection з пустим значенням.

Прослуховувач для об’єкту copyUserNameItem має ідентичний код, за винятком розшифрування, так як поле логіну не шифрується.

Коли обирається елемент меню «Відкрити файл», першим кроком ініціалізується об’єкт JFileChooser з ім’ям file, який був оголошений у класі MainWindow. Даний об’єкт використовується для відкриття вінка для вибору файлу з метою подальшої роботи з ним. Далі створюється об’єкт FileNameExtensionFilter з ім’ям filter, якому передається значення “.json” для

того, щоб користувач міг обрати файли тільки з таким типом, саме в такому файлі зберігаються дані з записами. Потім цей фільтр встановлюється об’єкту file за допомогою методу setFileFilter, і також за допомогою методу setAcceptAllFileFilterUsed, в якому передається значення false, фільтр, який дозволяє відображати всі файли не буде відображатись. Таким чином користувач зможе обрати тільки файли типу json. Далі створюється змінна цілочисельного типу n, якій присвоюється значення методу showDialog, який виконується для об’єкту file та відкриває вікно для вибору файлу. Якщо файл обрано, то змінній n присвоюється значення 0, яке відповідає значенню JFileChooser.APPROVE_OPTION. Після цього створюється об’єкт File з ім’ям chosenFile, якому передається файл, що був обраний, методом getSelecterFile. Далі створюється змінна рядкового типу checksumFilePath, якій присвоюється значення шляху обраного файлу зі зміною типу файлу з json на checksum. Даний шлях повинен відповідати файлу в якому зберігається контрольна сума файлу з даними, вона потрібна для перевірки цілісності файлу. Далі створюється змінна типу long з ім’ям savedCheckSum, в яку буде записано контрольну суму з файлу. Потім створюється об’єкт BufferedReader з ім’ям reader, якому передається файл збережений по шляху checksumFilePath. Потім створюється рядкова змінна line в яку записується інформація з файлу. Після чого інформація з line записується в savedCheckSum. Наступним кроком створюється нова змінна newCheckSum типу long, в яку записується результат підрахування контрольної суми файлу, який був обраний для відкриття. Для цього виконується метод calculateChecksum з класу CRC, в який передається шлях файлу, який було обрано. Далі відбувається порівняння цих контрольних сум і якщо вони збігаються, то відкривається вікно для проходження автентифікації, якщо ж ні, то користувач отримує повідомлення про помилку під час перевірки контрольної суми. Якщо вибір файлу був відмінений, то користувач отримає повідомлення про те, що вибір файлу був відмінений.

Коли обирається елемент меню «Зберегти файл», першим кроком створюється об’єкт Gson з ім’ям gson. Gson [34] – це бібліотека у Java для роботи

з файлами типу json. Далі використовується GsonBuilder для налаштування серіалізації json, який викликає setPrettyPrinting для зручного способу читання інформації та create, який передає налаштування об’єкту gson. Далі створюється об’єкт Type з ім'ям listType, якому передається тип даних, які будуть серіалізовані, в даному випадку це список об’єктів типу PasswordData. Наступним кроком створюється об’єкт FileWriter з ім’ям writer, який буде використовуватись для запису даних у файл selectedFile. Далі відбувається процес серіалізації списку об’єктів passwordData типу listType у об’єкт writer, цей процес відбувається за допомогою метода toJson, який викликається з об’єкту gson. Після запису даних у файл, створюється змінна типу long з ім’ям checksum, якій присвоюється значення результату підрахування контрольної суми файлу, в який записались дані, за допомогою методу calculateChecksum з класу CRC. Після чого відбувається процес запису контрольної суми у файл за допомогою методу saveChecksumToFile з того ж класу.

Таким чином у даному підрозділі було описано роботу основних класів та методів засобу захищеного зберігання паролів та семантичних одиниць коду. Наступним кроком необхідно провести тестування даного засобу.

4.3 Блокове тестування засобу

Першим необхідно провести блокове тестування (unit testing) [36] засобу. Так як розробка застосунку виконувалась у середовищі IntelliJ IDEA, то для блокового тестування можна використати пакет JUnit.

Враховуючи те, що у програмі інформація про записи користувача передається в об’єктом PasswordData, то першим необхідно протестувати конструктор, гетери та сетери даного класу. Для цього було створено клас PasswordDataTest.java, в якому було створено два тести, перед виконанням кожного тесту за допомогою конструктора створюється об’єкт типу PasswordData, якому передаються дані відповідних конструктору типів. У першому тесті перевіряються гетери і за допомогою методу assertEquals з класу

Assertions, в який передаються очікуваний результат та реальний, який передають гетери, проходить перевірка гетерів для кожного поля об’єкта PasswordData. У другому ж тесті полям об’єкту PasswordData, за допомогою сетерів, присвоюються нові значення, а потім так само, як і в попередньому тесті, за допомогою метода assertEquals, відбувається перевірка результатів. Результати тестування цього класу зображені на рисунку 4.2.

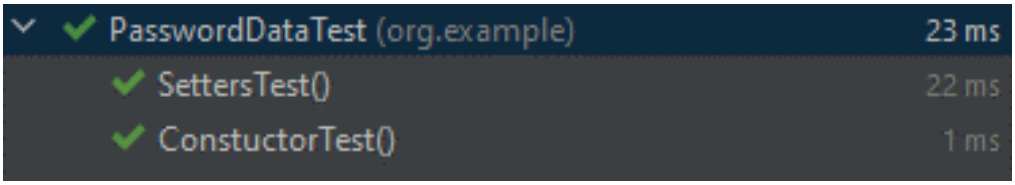


Рисунок 4.2 – Результати тестування класу PasswordDataTest.java

Наступним буде проведено тестування класу, який перевіряє введені користувачем паролі на стійкість. Для перевірки паролю на стійкість до нього висуваються такі вимоги:

– пароль повинен містити хоча б одну велику літеру;
– пароль повинен містити хоча б одну маленьку літеру;
– пароль повинен містити хоча б одну цифру;
– пароль повинен містити хоча б один спеціальний символ;
– пароль повинен бути довжиною принаймні 8 символів.

Для цього розроблено клас PasswordCheckTest.java, у якому створено два тести, які задовольняють умовам стійкого паролю, та сім тестів не задовольняють. Серед семи тестів, які не задовольняють умовам є паролі, які містять в собі тільки маленькі літери, тільки великі літери, суміш великих та маленьких літер, тільки цифри, суміш цифр та літер, тільки спеціальні символи та пароль, який не містить жодного символа. Результати тестування цього класу зображені на рисунку 4.3.

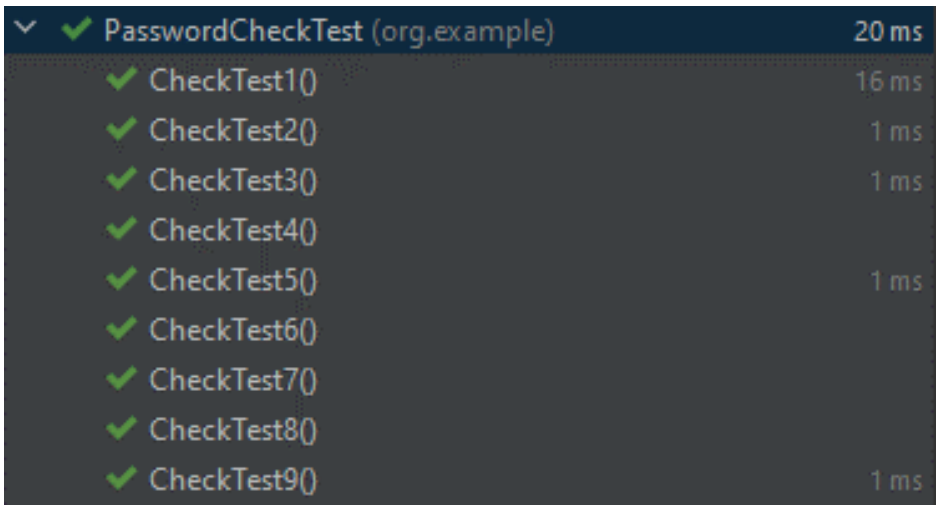


Рисунок 4.3 – Результати тестування класу PasswordCheckTest.java

Наступним буде проведено тестування класу, який забезпечує цілісність файлів, обчисленням контрольної суми. Для цього розроблено клас CRCTest.java, у якому створено два тести. Перший тест обчислює контрольну суму та за допомогою раніше згаданого методу assertEquals перевіряє її з тою, яка повинна бути. Другий же тест перевіряє метод saveChecksumToFile, який зберігає контрольну суму в файл. Для цього створюється змінна типу long, в яку записується значення обчисленої контрольної суми, а потім вона записується у файл. Після чого дані з файлу зчитуються в нову змінну та за допомогою методу assertEquals порівнюються. Результати тестування цього класу зображені на рисунку 4.4.

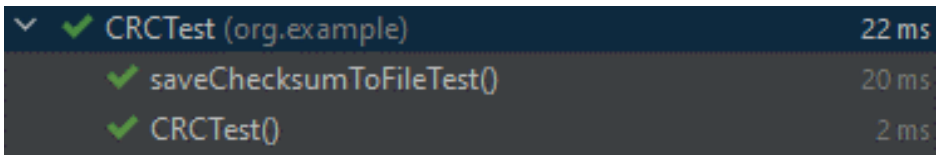


Рисунок 4.4 – Результати тестування класу CRCTest.java

В даному підрозділі було проведено блокове тестування основних класів та методів програми, які відповідають за перевірку паролів, забезпечення цілісності файлу з даними та клас, який використовується, як об’єкт для обробки даних в програмі. Увесь код блокового тестування наведено у додатку Б. Наступним необхідно провести інтеграційне тестування засобу.

4.4 Інтеграційне тестування засобу

Після проведення блокового тестування варто провести інтеграційне [36]. Так як користувач взаємодіє з програмою через головне вікно, яке зв’язує всі класи та методи, то варто тестувати саме його. Запустимо код, який дозволить перевірити програму в режимі тестування (код наведено у додатку Б).

Для початку роботи програми користувачу потрібно пройти етап реєстрації, якщо він бажає створити новий файл з записами, або пройти етап автентифікації, якщо він бажає відкрити раніше створений файл. Для реєстрації користувачу необхідно обрати файл типу .json, в якому він бажає зберігти дані. Після чого йому необхідно створити пароль та підтвердити його повторним введенням (рис. 4.5).

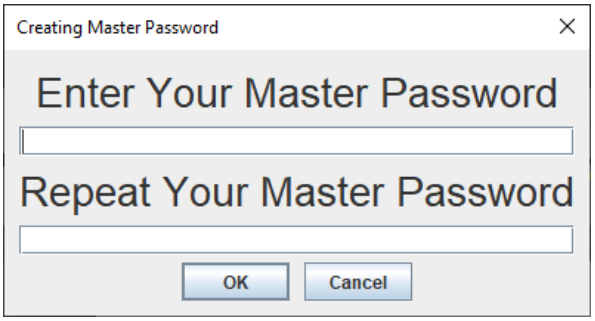


Рисунок 4.5 – Результат виводу вікна для автентифікації

Після введення паролю він буде переданий модулю перевірки стійкості паролю. Якщо рівень стійкості паролю буде низьким, то користувач отримає відповідне повідомлення з рекомендаціями до створення стійкого паролю (рис. 4.6), і зможе або ввести його повторно, або на свій страх та ризик використовувати цей.

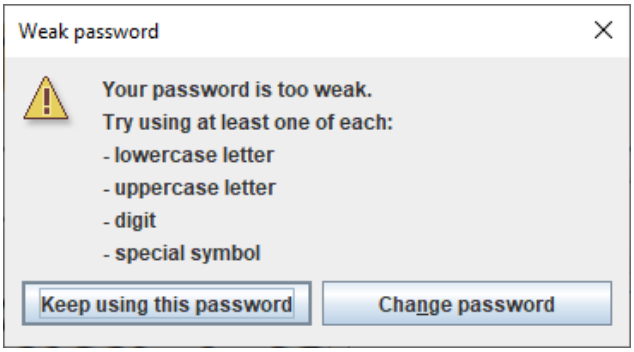


Рисунок 4.6 – Результат виводу вікна після перевірки паролю на стійкість

Після чого користувач повернеться до головного вікна програми, в якому отримає повідомлення про успішне створення файлу (рис. 4.7) та зможе створювати та зберігати записи.

Рисунок 4.7 – Результат про успішну реєстрацію та сворення файлу

Для автентифікації користувачу необхідно обрати файл, який він хоче відкрити. Який далі передається модулю перевірки цілісності, де виконується перевірка контрольної суми для того, щоб перевірити чи був файл з даними модифікований, якщо так, то користувач отримає відповідне повідомлення, якщо ні, то користувачеві буде запропоновано ввести пароль (рис. 4.8), який буде передано модулю перевірки паролю.

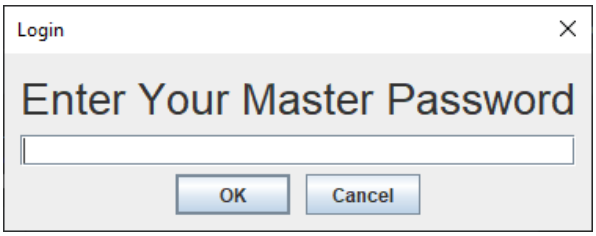


Рисунок 4.8 – Результат про успішну реєстрацію та створення файлу

Після перевірки паролю перевіряється носій на якому було відкрито файл, якщо його відкрито на носію відмінному від того, на якому він був створений, то доступ до паролів користувач не отримає. При успішній автентифікації, так само як і при реєстрації, користувач отримує доступ до головного вікна програми в якому може взаємодіяти з таблицею з записами.

Тепер необхідно провести тестування модулю додавання запису. При додаванні нового запису користувачу відкривається діалогове вікно з полями для вводу даних (рис. 4.9), вони не є обов’язковими для заповнення, їх можна залишити пустими, а потім редагувати за допомогою відповідної функції.

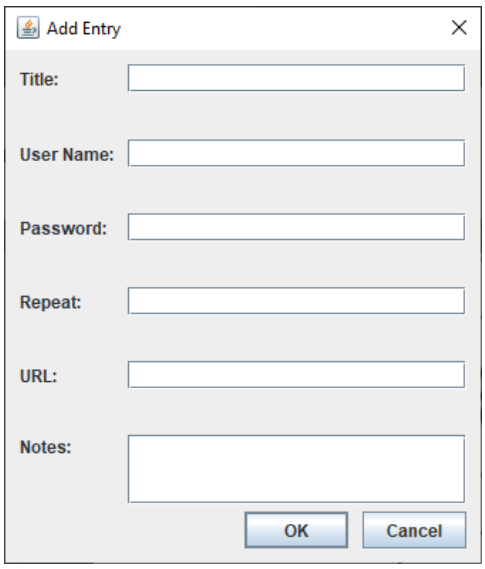


Рисунок 4.9 – Результат відкриття вікна для додавання нового запису

Перед збереженням запису введений в поле пароль буде передано модулю перевірки стійкості й так само, як при реєстрації буде виведено відповідний результат (рис. 4.6). Після проведення перевірки паролю, запис буде додано до таблиці у головному вікні програми (рис. 4.10).

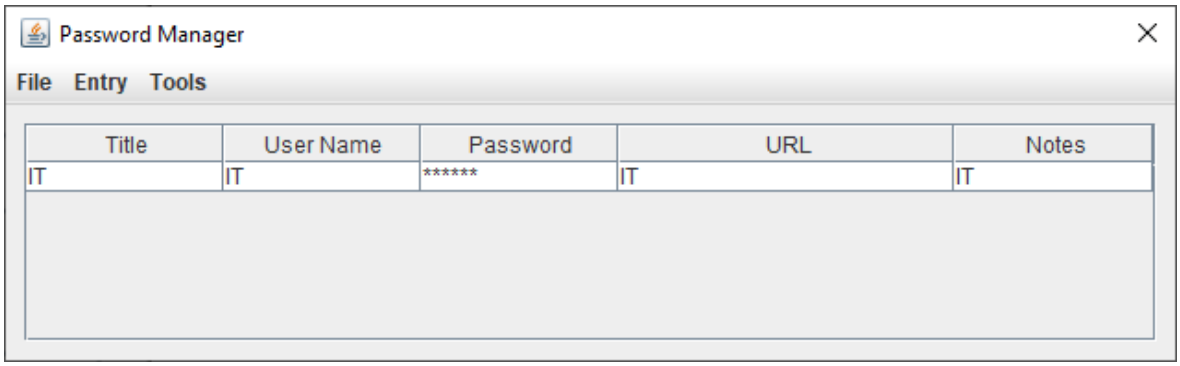


Рисунок 4.10 – Результат додавання нового запису до таблиці

Наступним необхідно провести тестування функції редагування паролю. Дана функція, так само як і функції копіювання логіну або паролю та видалення запису, буде активною тільки після того, як буде обрано запис. При взаємодії з даною функцією дані обраного запису будуть передані модулю редагування записів. Після чого користувачеві відкриється вікно аналогічне вінку додавання записів, але з полями заповненими даними обраного запису, де користувач зможе їх редагувати (рис. 4.11).

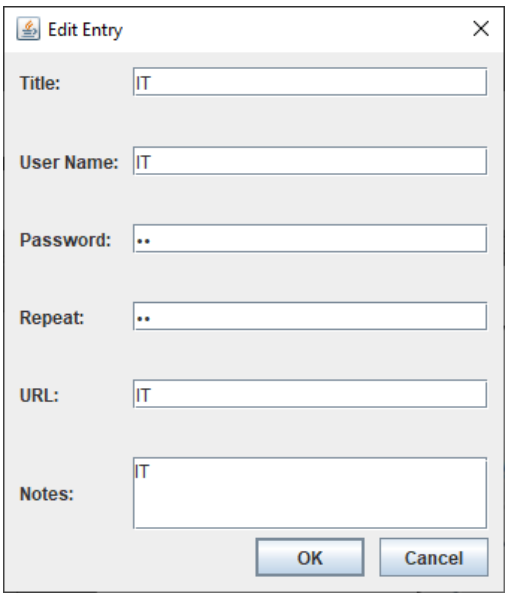


Рисунок 4.11 – Результат відкриття вікна для редагування запису

Далі необхідно провести тестування функції копіювання логіну та паролю. Після вибору запису та взаємодії з функцією копіювання логіну (рис. 4.12), дані про запис будуть передані модулю копіювання логіну.

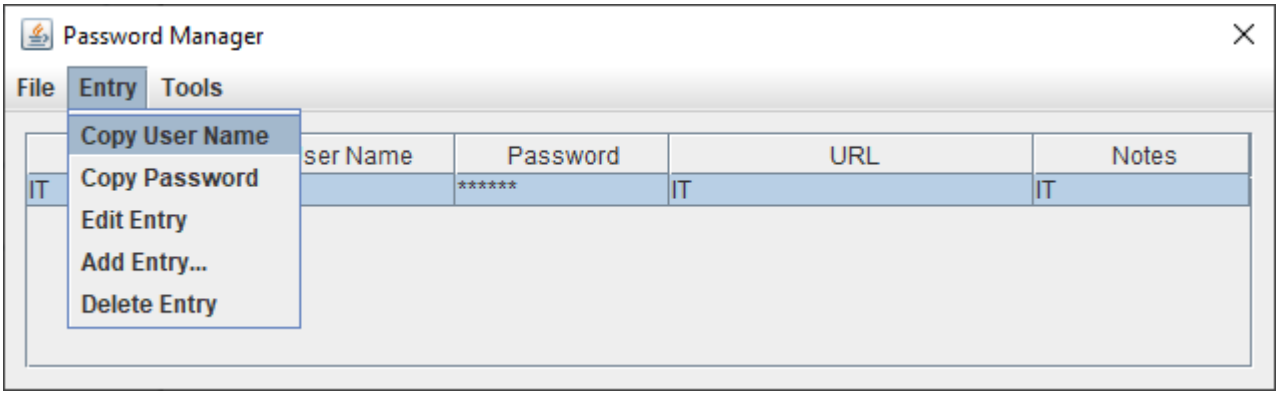


Рисунок 4.12 – Зображення елементу меню для копіювання логіну

Після чого логін буде скопійовано до буферу обміну та запуститься десятисекундний таймер під час якого користувач матиме змогу вставити логін, за допомогою комбінації клавіш “ctrl+V”, або контекстному меню, що викликається на праву клавішу миші та функції “Paste” (рис. 4.13).

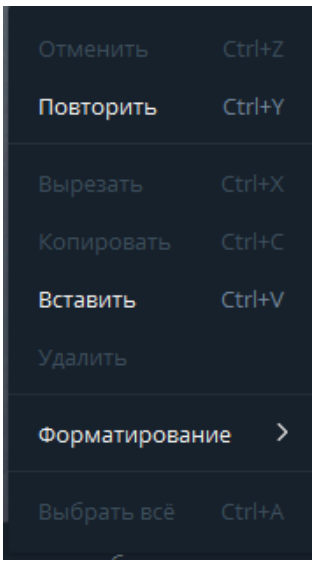


Рисунок 4.13 – Зображення можливості копіювання логіну через контекстне меню

Після закінчення десяти секунд буфер обміну очиститься та копіювання буде неможливе. Функція копіювання паролю працює аналогічним чином.

Наступним необхідно провести тестування функції видалення записів. Після вибору запису та взаємодії з функцією видалення, користувачеві виведеться повідомлення, щодо того чи впевнений він, що бажає видалити даний запис (рис. 4.14).

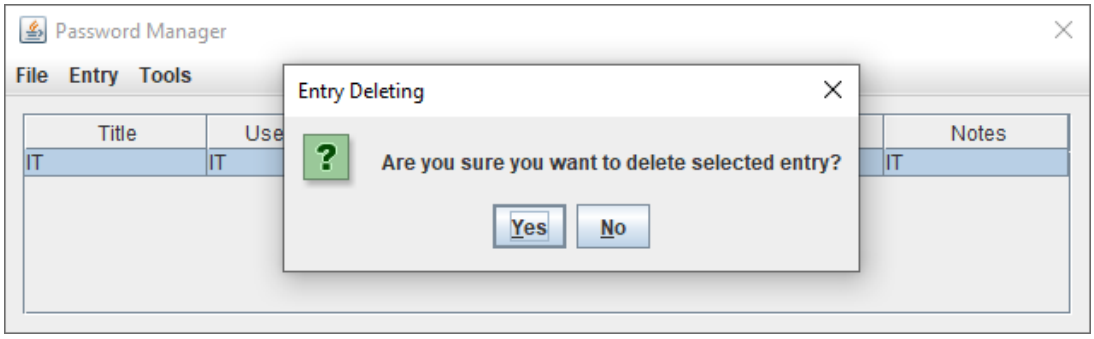


Рисунок 4.14 – Результат відкриття вікна для підтвердження видалення запису

Якщо користувач підтверджує видалення, то обраний запис буде передано модулю видалення, після чого він зникне з таблиці (рис. 4.15).

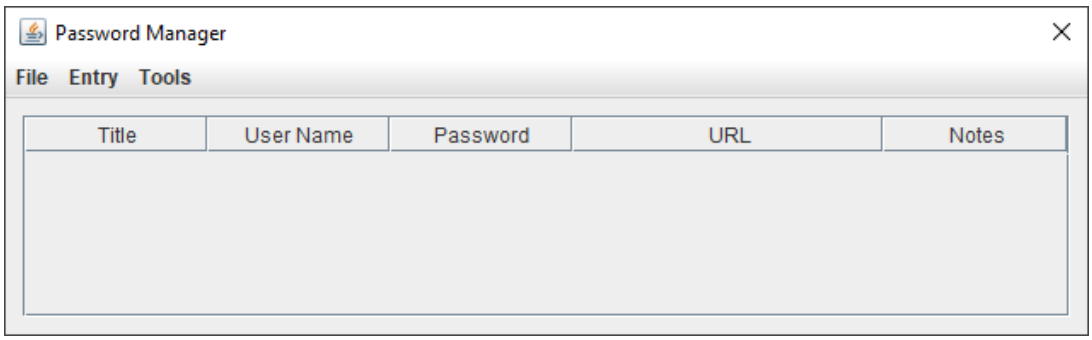


Рисунок 4.15 – Результат видалення запису

Наступним необхідно провести тестування функції генерування паролю. При взаємодії з функцією генерування паролю (рис. 4.16) буде згенеровано пароль, який буде доданий до таблиці у виді запису з пустими полями, окрім поля зі згенерованим паролем (рис. 4.17). Потім даний запис можна буде редагувати або видалити.

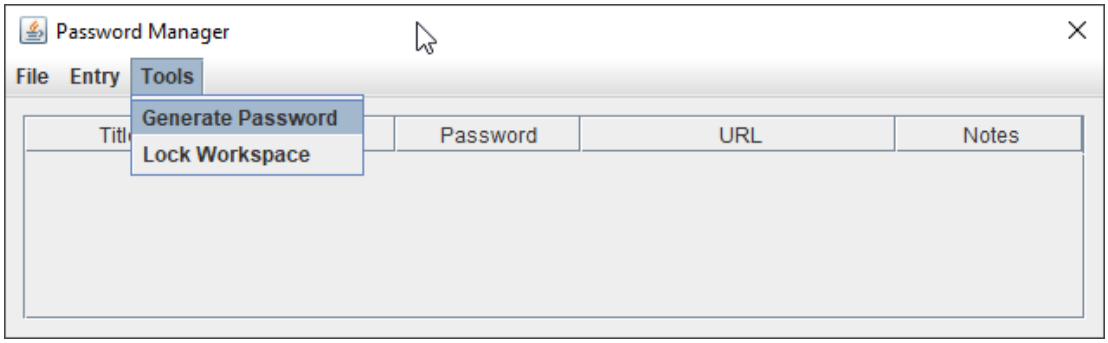


Рисунок 4.16 – Зображення елементу меню для генерування паролю

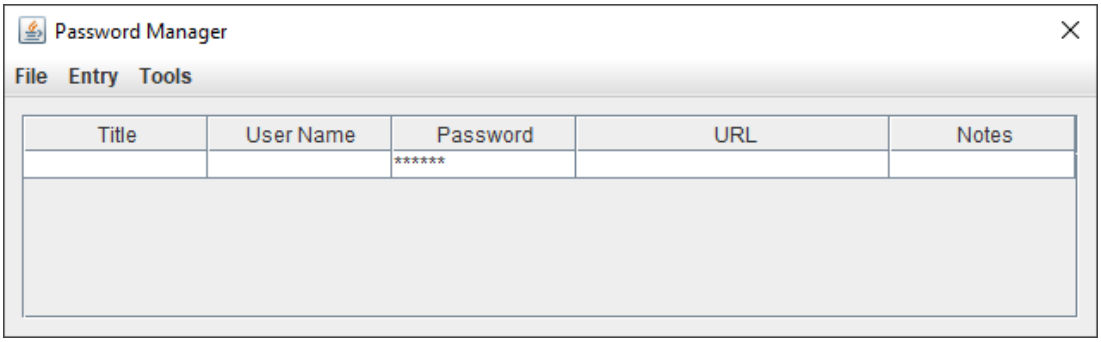


Рисунок 4.17 – Результат генерування паролю і додавання його у таблицю

Далі необхідно провести тестування функції збереження даних. Коли користувач взаємодіє з даною функцією, то дані з таблиці з записами передаються модулю збереження даних, де вони записуються у відповідний файл. Після чого підраховується контрольна сума даного файлу для перевірки,

коли файл буде відриватись знову. Після цього користувач отримує повідомлення про успішне збереження (рис. 4.18).

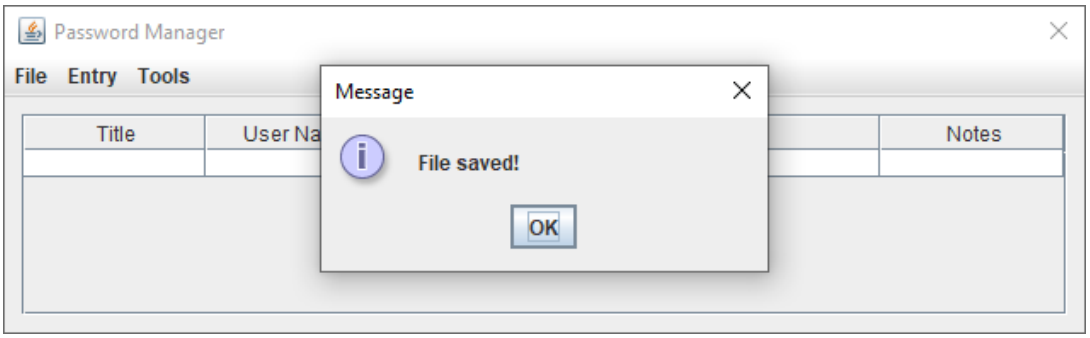


Рисунок 4.18 – Зображення повідомлення про успішне збереження файлу

Наступним необхідно провести тестування функції блокування робочого простору. Коли користувач взаємодіє з даною функцією, то таблиця повністю очищається та відкривається вікно автентифікації (рис. 4.19). Далі користувачеві для отримання доступу до таблиці з записами треба знову пройти процес автентифікації, або відмовитись та закрити засіб.

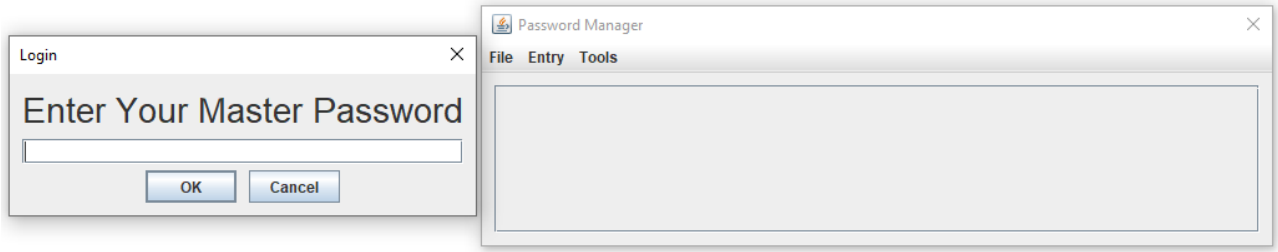


Рисунок 4.19 – Результат блокування робочого простору

На рисунку 4.20 зображено загальне покриття коду блоковими та інтеграційними тестами.

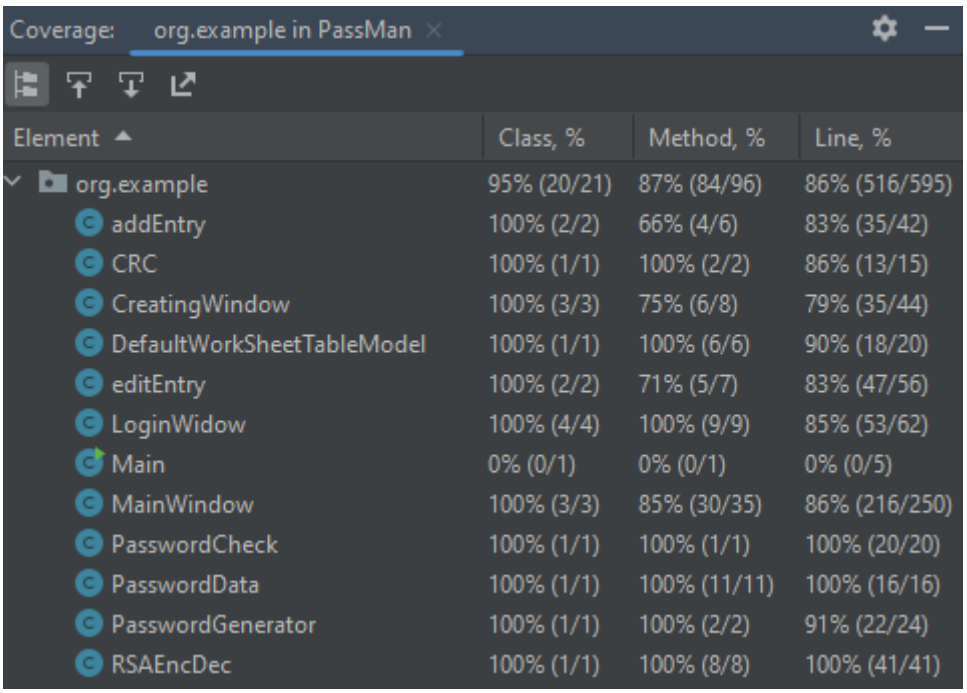


Рисунок 4.20 – Результати покриття коду тестами

Таким чином було проведено модульне та інтеграційне тестування. В наступному підрозділі буде описано висновки щодо роботи проведеної в даному розділі.

4.5 Висновки з розділу

У даному розділі було проведено обґрунтування засобів для розробки програмного забезпечення, що дозволило підвищити ефективність розробки засобу захищеного зберігання паролів. Після чого було проведено аналіз основних семантичних одиниць коду, що дозволило краще зрозуміти роботу програмного засобу. Далі було написано тести для проведення блокового тестування основних методів таким чином, щоб не зачепити роботу інших, що дозволило знайти та виправити помилки допущені при реалізації методів. Після проведення блокового тестування, було проведено інтеграційне тестування для перевірки коректної роботи всієї системи та взаємодії її модулів. Це дозволило виявити помилки при передачі даних між модулями та виправити їх.

ВИСНОВКИ

На основі проведеного аналізу атак на засоби захищеного зберігання паролів було визначено, що наразі актуально розробляти засоби захищеного зберігання паролів, оскільки вони захищають від помилок користувачів, а також це дозволяє використовувати захищені паролі без необхідності їх запам’ятовування. Далі було проведено порівняльний аналіз відомих засобів захисту паролів, що дозволило зробити висновки щодо особливостей їх реалізації та спільних недоліків. Проведені аналізи дозволили виконати постановку задачі. Було розроблено архітектуру засобу, що дозволило керувати складністю процесу дипломного проєктування. Розроблено структури основних модулів засобу, які пізніше були використані при розробці алгоритмів засобу.

Проведений аналіз криптографічних алгоритмів та алгоритмів захисту цілісності дозволив навести переваги й особливості алгоритмів, що будуть використовуватись при розробці засобу.

Було розроблено узагальнений алгоритм системи засобу та алгоритми основних модулів, що дозволило спростити та структурувати процес реалізації програмного засобу та створити зв’язки між алгоритмами.

Проведений порівняльний аналіз мов програмування та середовищ розробки допоміг навести переваги й особливості засобів, що були обраними, як найбільш ефективні для розробки засобу захищеного зберігання паролів.

Аналіз основних семантичних одиниць коду дозволив краще пояснити роботу програмного засобу, його модулів та основних функцій. Розроблені блокові тести допомогли провести ефективне тестування окремих методів програмного засобу, що дозволило виявити та виправити помилки допущені при розробці.

Виконане інтеграційне тестування дозволило перевірити коректність роботи всієї системи та взаємодії її модулів, що допомогло виявити та виправити помилки при передачі даних між модулями та виправити їх. Проведене блокове та інтеграційне тестування дозволило покрити 86% коду тестами.

Розроблений засіб захищеного зберігання паролів може бути використаний як для особистого користування, так і для робочого на підприємстві. Враховуючи те, що файл створений для збереження даних з паролями прив’язується до носія, то можна створювати файл на жорсткому диску та зберігати резервну копію файлу на флеш-носії у сейфі, щоб відновити доступ до файлу, якщо він буде несанкціоновано модифікований. Якщо файл буде несанкціоновано скопійовано з жорсткого диску, то через прив’язку до носія зловмисник не зможе отримати доступ до паролів.

Також розроблений засіб можна використовувати для збереження не тільки паролів, а й інших факторів автентифікації (PIN-кодів, JWT токенів тощо).

У розробленому засобі для забезпечення захисту від шпигунських програм, що роблять знімки екрану, вивід паролю маскується під послідовність символів «*». Для того, щоб зменшити ризик несанкціонованого копіювання логіну і паролю з буферу обміну, реалізовано його очищення через 10 секунд після копіювання логіну або паролю. Цього часу має бути достатньо для копіювання даних у форму автентифікації.

Розроблений засіб може бути удосконалений розробкою функції збереження часу створення або останньої модифікації паролю для нагадування користувачу, що пароль використовується давно та його час замінити. Також можна розробити візуальне відображення таймеру до очистки буферу обміну. Ще однією важливою функцією для забезпечення захисту даних буде реалізація шифрування інших полів з даними (тому що наразі шифруються тільки паролі), наприклад симетричним шифруванням, так як, воно працює швидше асиметричного. Також можна виконати реалізацію альтернативних потоків даних, в які будуть записуватись файли з контрольною сумою та відкритим ключем для шифрування даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Клименко В. О.. Аналіз засобів захищеного зберігання паролів. Доповідь LII Науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії. 2023. Секція захисту інформації. 3 с. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view/17386 (дата звернення 16.05.2023).

2. Клименко В. О.. Контроль над громадянами в епоху цифрових технологій. Доповідь XLIX Науково-технічної конференції Інституту соціально-гуманітарних наук. 2020. Інститут соціально-гуманітарних наук. 2 с. URL: https://d.conf.vntu.edu.ua/index.php/all-hum/all-hum-2020/paper/view/10351/8690 (дата звернення: 16.05.2023).

3. Баришев Ю. В., Чайкін М. М., Кохан О. В.. Метод та засіб підвищення стійкості зрозумілих користувачам текстових паролів. Наукові праці ВНТУ. 2022. 8 с. URL: https://praci.vntu.edu.ua/index.php/praci/article/view/655 (дата звернення: 16.05.2023)

4. Shannon Riley. Password Security : What Users Know and What They Actually Do. Usability News. 2006. Vol. 8, Issue 1. P. 5. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.597.5846&rep=rep1&type=pdf (accessed: 16.05.2023).

5. Create and use strong passwords - Microsoft Support. Microsoft Support. URL: https://support.microsoft.com/en-us/windows/create-and-use-strong-passwords-c5cebb49-8c53-4f5e-2bc4-fe357ca048eb (accessed: 17.05.2023).

6. Grimes R. Password Managers Can Be Hacked Lots of Ways and Yes, You Should Still Use Them. KnowBe4 Security Awareness Training Blog. URL: https://blog.knowbe4.com/password-managers-can-be-hacked (accessed: 18.05.2023).

7. Kost E. What is a Keylogger? Hackers Could Be Stealing Your Passwords | UpGuard. Third-Party Risk and Attack Surface Management Software | UpGuard. URL: https://www.upguard.com/blog/what-is-a-keylogger (accessed: 18.05.2023).

8. Brute Force vs. Dictionary Attack: What's the Difference? – Rublon. Rublon. URL: https://rublon.com/blog/brute-force-dictionary-attack-difference/ (accessed: 18.05.2023).

9. Phishing attacks. Imperva. URL: https://www.imperva.com/learn/application-security/phishing-attack-scam/ (accessed: 19.05.2023).

10. Vijayan J. Convincing, Malicious Google Ads Look to Lift Password Manager Logins. Dark Reading. URL: https://www.darkreading.com/threat-intelligence/convincing-malicious-google-ads-password-managers (accessed: 19.05.2023).

11. Toubba K. Security Incident December 2022 Update - LastPass. The LastPass Blog. URL: https://blog.lastpass.com/2022/12/notice-of-recent-security-incident/ (accessed: 19.05.2023).

12. NordPass – NordPass features. URL: https://nordpass.com/features/ (accessed: 22.05.2023).

13. RoboForm – RoboForm features. URL: https://www.roboform.com/en/key-features (accessed: 22.05.2023).

14. 1Password – 1Password pricing. URL: https://1password.com/teams/pricing/ (date of access: 22.05.2023).

15. Keeper – Keeper Unlimited Free Trial & Keeper Free Version. URL: https://www.keepersecurity.com/free-trial-vs-free-version.html (accessed: 22.05.2023).

16. Dashlane – Trusted Personal Password Manager. URL: https://www.dashlane.com/personal-password-manager (date of access: 22.05.2023).

17. LastPass – Why LastPass. URL: https://www.lastpass.com/why-lastpass (accessed: 22.05.2023).

18. Bitwarden – Bitwarden pricing. URL: https://bitwarden.com/pricing/business/ (date of access: 22.05.2023).

19. Enpass – Enpass pricing. URL: https://www.enpass.io/pricing/ (accessed: 22.05.2023).

20. Miller B. 8 Pros and Cons of Asymmetric Encryption. Green Garage. URL: https://greengarageblog.org/8-pros-and-cons-of-asymmetric-encryption (accessed: 29.05.2023).

21. US4405829A - Cryptographic communications system and method - Google Patents. URL: https://patents.google.com/patent/US4405829 (accessed: 05.06.2023).

22. RSA Security Releases RSA Encryption Algorithm into Public Domain - RSA, The Security Division of EMC. Wayback Machine. URL: https://web.archive.org/web/20071120112201/http://www.rsa.com/press_release.aspx?id=261 (accessed: 05.06.2023).

23. Koopman P., Driscoll K., Hall B. Selection of Cyclic Redundancy Code and Checksum Algorithms to Ensure Critical Data Integrity. Springfield, Virginia : National Technical Information Services, 2015. 111 p. URL: https://www.tc.faa.gov/its/worldpac/techrpt/tc14-49.pdf (accessed: 06.06.2023).

24. Paar C., Pelzl J. Understanding Cryptography. Berlin, Heidelberg : Springer, 2009. 372 p. URL: http://euro.ecom.cmu.edu/resources/elibrary/epay/Hash.pdf (date of access: 07.06.2023).

25. Katz A., Dash S. Error correcting codes | Brilliant Math & Science Wiki. Brilliant | Learn interactively. URL: https://brilliant.org/wiki/error-correcting-codes/ (date of access: 07.06.2023).

26. Pros and Cons of Java | Advantages and Disadvantages of Java - DataFlair. URL: https://data-flair.training/blogs/pros-and-cons-of-java/ (accessed: 12.06.2023).

27. The Good and the Bad of C# Programming. AltexSoft. URL: https://www.altexsoft.com/blog/c-sharp-pros-and-cons/ (accessed: 12.06.2023).

28. The Good and the Bad of Python Programming Language. AltexSoft. URL: https://www.altexsoft.com/blog/python-pros-and-cons/ (accessed: 12.06.2023).

29. Python Programming - The State of Developer Ecosystem in 2022 Infographic. JetBrains: Developer Tools for Professionals and Teams. URL: https://www.jetbrains.com/lp/devecosystem-2022/python/ (accessed: 12.06.2023).

30. IntelliJ IDEA – the Leading Java and Kotlin IDE. JetBrains. URL: https://www.jetbrains.com/idea/ (accessed: 13.06.2023).

31. Most Popular Java IDEs [2022] | JRebel by Perforce. JRebel by Perforce. URL: https://www.jrebel.com/blog/best-java-ide (accessed: 13.06.2023).

32. Eclipse Downloads | The Eclipse Foundation. The Community for Open Innovation and Collaboration | The Eclipse Foundation. URL: https://www.eclipse.org/downloads/ (accessed: 13.06.2023).

33. Welcome to Apache NetBeans. Welcome to Apache NetBeans. URL: https://netbeans.apache.org/ (accessed: 13.06.2023).

34. GitHub - google/gson: A Java serialization/deserialization library to convert Java Objects into JSON and back. GitHub. URL: https://github.com/google/gson/ (accessed: 15.06.2023).

35. Hamilton T. Unit Testing Tutorial – What is, Types & Test Example. Guru99. URL: https://www.guru99.com/unit-testing-guide.html (accessed: 16.06.2023).

36. Hamilton T. Integration Testing: What is, Types with Example. Guru99. URL: https://www.guru99.com/integration-testing.html (accessed: 16.06.2023).

ДОДАТКИ

Додаток А. Код основних функціональних можливостей засобу

Код файлу MainWindow.java

```
package org.example;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.reflect.TypeToken;

import javax.swing.*;
import javax.swing.table.TableColumnModel;
import javax.swing.filechooser.FileNameExtensionFilter;
import java.awt.*;
import java.awt.datatransfer.*;
import java.awt.event.*;
import java.io.*;
import java.util.List;

public class MainWindow extends JDialog {
    private JPanel contentPane;
    private JTable table1;
    public JMenuItem addEntryItem;
    public JMenuItem newFileItem;
    public JMenuItem openFileItem;
    public JMenuItem saveFileItem;
    public JMenuItem lockItem;
    public JMenuItem exitItem;
    public JMenuItem copyUserNameItem;
    public JMenuItem copyPasswordItem;
    public JMenuItem editEntryItem;
    public JMenuItem deleteEntryItem;
    public JMenuItem generatePasswordItem;
    public JFileChooser file;
    private List<PasswordData> passwordData;
    private File selectedFile;


    public MainWindow(List<PasswordData> passwordDataTransferable, boolean flag, File file) {
        selectedFile = file;
        passwordData = passwordDataTransferable;
        setContentPane(contentPane);
        setModal(true);
        setTitle("Password Manager");
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        setPreferredSize(new Dimension(650, 200));
        setLocationRelativeTo(null);
        menuBarCreation();
        actionMenuCreation();
        if (flag) table();
    }
```

```
    public void menuBarCreation(){
        JMenuBar menuBar = new JMenuBar();

        JMenu file = new JMenu("File");
        JMenu entry = new JMenu("Entry");
        JMenu tools = new JMenu("Tools");

        menuBar.add(file);
        menuBar.add(entry);
        menuBar.add(tools);

        newFileItem = new JMenuItem("New File...");
        openFileItem = new JMenuItem("Open File...");
        saveFileItem = new JMenuItem("Save File");
        lockItem = new JMenuItem("Lock Workspace");
        exitItem = new JMenuItem("Exit");
        copyUserNameItem = new JMenuItem("Copy User Name");
        copyPasswordItem = new JMenuItem("Copy Password");
        editEntryItem = new JMenuItem("Edit Entry");
        addEntryItem = new JMenuItem("Add Entry...");
        deleteEntryItem = new JMenuItem("Delete Entry");
        generatePasswordItem = new JMenuItem("Generate Password");

        file.add(newFileItem);
        file.add(openFileItem);
        file.add(saveFileItem);
        saveFileItem.setEnabled(false);
        file.add(exitItem);
        entry.add(copyUserNameItem);
        copyUserNameItem.setEnabled(false);
        entry.add(copyPasswordItem);
        copyPasswordItem.setEnabled(false);
        entry.add(editEntryItem);
        editEntryItem.setEnabled(false);
        entry.add(addEntryItem);
        addEntryItem.setEnabled(false);
        entry.add(deleteEntryItem);
        deleteEntryItem.setEnabled(false);
        tools.add(generatePasswordItem);
        generatePasswordItem.setEnabled(false);
        tools.add(lockItem);
        lockItem.setEnabled(false);

        setJMenuBar(menuBar);
    }

    private void table() {
        DefaultWorkSheetTableModel tableModel = new
DefaultWorkSheetTableModel(passwordData);

        ListSelectionModel selectionModel = table1.getSelectionModel();
```

```
    selectionModel.addListSelectionListener(e -> {
        if (!e.getValueIsAdjusting()) {
            boolean cellSelected = table1.getSelectedRowCount() > 0;
            copyUserNameItem.setEnabled(cellSelected);
            copyPasswordItem.setEnabled(cellSelected);
            editEntryItem.setEnabled(cellSelected);
            deleteEntryItem.setEnabled(cellSelected);
        }
    });

    addEntryItem.setEnabled(true);
    saveFileItem.setEnabled(true);
    lockItem.setEnabled(true);
    generatePasswordItem.setEnabled(true);

    table1.setModel(tableModel);
    TableColumnModel columnModel = table1.getColumnModel();
    columnModel.getColumn(3).setMinWidth(150);
}

private void actionMenuCreation(){
    addEntryItem.addActionListener(e -> {
        addEntry addEntry1 = new addEntry(passwordData, selectedFile);
        addEntry1.pack();
        addEntry1.setVisible(true);
        savingToFile();
    });

    copyUserNameItem.addActionListener(e -> {
        int selectedRow = table1.getSelectedRow();
        String userName = passwordData.get(selectedRow).getUserName();

        StringSelection stringSelection = new StringSelection(userName);
        Clipboard clipboard =Toolkit.getDefaultToolkit().getSystemClipboard();
        clipboard.setContents(stringSelection, (clipboard1, contents) -> { });

        Timer timer = new Timer(10000, e13 -> {
            if (clipboard.isDataFlavorAvailable(DataFlavor.stringFlavor)) {
                clipboard.setContents(new StringSelection(""), null);
            }
        });
        timer.setRepeats(false);
        timer.start();
    });

    copyPasswordItem.addActionListener(e -> {
        int selectedRow = table1.getSelectedRow();
        byte [] bytePass = passwordData.get(selectedRow).getPassword();
        String password;
        try {
            password = RSAEncDec.decrypt(bytePass, selectedFile);
        } catch (Exception ex) {
```

```
            throw new RuntimeException(ex);
        }

        StringSelection stringSelection = new StringSelection(password);
        Clipboard clipboard =Toolkit.getDefaultToolkit().getSystemClipboard();
        clipboard.setContents(stringSelection, (clipboard12, contents) -> { });

        Timer timer = new Timer(10000, e12 -> {
            if (clipboard.isDataFlavorAvailable(DataFlavor.stringFlavor)) {
                clipboard.setContents(new StringSelection(""), null);
            }
        });
        timer.setRepeats(false);
        timer.start();
    });

    editEntryItem.addActionListener(e -> {
        int selectedRow = table1.getSelectedRow();
        editEntry editEntry1 = new editEntry(passwordData,
                selectedRow, selectedFile);
        editEntry1.pack();
        editEntry1.setVisible(true);
        savingToFile();
    });

    deleteEntryItem.addActionListener(e -> {
        UIManager.put("OptionPane.yesButtonText", "Yes");
        UIManager.put("OptionPane.noButtonText", "No");
        int check = JOptionPane.showConfirmDialog(null, "Are you sure you want to" +
                        " delete selected entry?", "Entry Deleting",
                JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);

        if (check == JOptionPane.YES_OPTION) {
            int selectedRow = table1.getSelectedRow();
            passwordData.remove(selectedRow);
            table();
        }
        savingToFile();
    });

    generatePasswordItem.addActionListener(e -> {
        PasswordGenerator passwordGenerator = new PasswordGenerator();
        passwordData = passwordGenerator.PassGen(passwordData, selectedFile);
        table();
        savingToFile();
    });

    openFileItem.addActionListener(e -> {
        file = new JFileChooser("C:\\");
        FileNameExtensionFilter filter = new FileNameExtensionFilter(".json", "json");
        file.setFileFilter(filter);
        file.setAcceptAllFileFilterUsed(false);
```

```
            int n = file.showDialog(null, "Choose");
            if (n == JFileChooser.APPROVE_OPTION) {
                File chosenFile = file.getSelectedFile();
                String checksumFilePath = chosenFile.getAbsolutePath().replace(".json", "") +
".checksum";
                long savedCheckSum = 0L;
                try (BufferedReader reader = new BufferedReader(new FileReader(checksumFilePath)))
{
                    String line = reader.readLine();
                    savedCheckSum = Long.parseLong(line);
                } catch (IOException e2) {
                    JOptionPane.showMessageDialog(this, "Error! File was modified!");
                }

                long newCheckSum = CRC.calculateChecksum(chosenFile.getAbsolutePath());
                if (savedCheckSum == newCheckSum){
                    LoginWidow loginWidow = new LoginWidow(file.getSelectedFile());
                    loginWidow.pack();
                    loginWidow.setVisible(true);
                } else JOptionPane.showMessageDialog(this, "Error! File was modified!");

            } else JOptionPane.showMessageDialog(this, "File opening canceled.");
        });

        newFileItem.addActionListener(e -> {
            file = new JFileChooser("D:\\");
            FileNameExtensionFilter filter = new FileNameExtensionFilter(".json","json");
            file.setFileFilter(filter);
            file.setAcceptAllFileFilterUsed(false);
            int n = file.showDialog(null, "Create");
            if (n == JFileChooser.APPROVE_OPTION)
            {
                selectedFile = file.getSelectedFile();
                try {
                    if (selectedFile.createNewFile()) {
                        CreatingWindow creatingWindow = new CreatingWindow(selectedFile);
                        creatingWindow.pack();
                        creatingWindow.setVisible(true);

                        if (selectedFile.exists()) {
                            JOptionPane.showMessageDialog(this, "File created successfully.");
                        }else JOptionPane.showMessageDialog(this, "File creating canceled.");
                    } else {
                        JOptionPane.showMessageDialog(this, "File already exists.");
                    }
                } catch (IOException e2) {
                    JOptionPane.showMessageDialog(this,
                            "Error CreatingWindow file: " + e2.getMessage());
                }
            }
            table();
        });
```

```
        saveFileItem.addActionListener(e -> {
            savingToFile();
            JOptionPane.showMessageDialog(this, "File saved!");
        });

        lockItem.addActionListener(e -> {
            dispose();
            LoginWidow loginWidow = new LoginWidow(selectedFile);
            loginWidow.pack();
            loginWidow.setVisible(true);
        });

        exitItem.addActionListener(e -> {
            savingToFile();
            UIManager.put("OptionPane.yesButtonText", "Yes");
            UIManager.put("OptionPane.noButtonText", "No");
            int n = JOptionPane.showOptionDialog(null, "Do you want to exit?", "Exit",
                    JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, null,
null);
            if (n == JOptionPane.YES_OPTION) System.exit(0);
        });

        addWindowListener(new WindowListener() {
            @Override
            public void windowOpened(WindowEvent e) { }

            @Override
            public void windowClosing(WindowEvent e) {
                UIManager.put("OptionPane.yesButtonText", "Yes");
                UIManager.put("OptionPane.noButtonText", "No");
                int n = JOptionPane.showOptionDialog(e.getWindow(),
                        "Do you want to exit?", "Exit",
                        JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,
                        null, null, null);
                if (n == JOptionPane.YES_OPTION)
                {
                    e.getWindow().setVisible(false);
                    System.exit(0);
                }
            }

            @Override
            public void windowClosed(WindowEvent e) { }

            @Override
            public void windowIconified(WindowEvent e) { }

            @Override
            public void windowDeiconified(WindowEvent e) { }

            @Override
```

```
            public void windowActivated(WindowEvent e) { }

            @Override
            public void windowDeactivated(WindowEvent e) { }
        });
    }

    private void savingToFile() {
        Gson gson = new GsonBuilder().setPrettyPrinting().create();
        java.lang.reflect.Type listType = new TypeToken<List<PasswordData>>() {}.getType();

        try (FileWriter writer = new FileWriter(selectedFile)) {
            gson.toJson(passwordData, listType, writer);
        } catch (IOException e2) {
            e2.printStackTrace();
        }
        long checksum = CRC.calculateChecksum(selectedFile.getAbsolutePath());
        CRC.saveChecksumToFile(selectedFile.getAbsolutePath(), checksum);
    }
}
```

Код класу PasswordData.java

```
package org.example;

public class PasswordData {
    private String title;
    private String userName;
    private byte[] password;
    private String url;
    private String notes;
    public PasswordData(String title, String userName, byte[] password, String url, String notes) {
        this.title = title;
        this.userName = userName;
        this.password = password;
        this.url = url;
        this.notes = notes;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getUserName() {
        return userName;
    }
    public void setUserName(String userName) {
        this.userName = userName;
    }
    public byte[] getPassword() { return password; }
    public void setPassword(byte[] password) { this.password = password; }
```

```
    public String getUrl() {
        return url;
    }
    public void setUrl(String url) {
        this.url = url;
    }
    public String getNotes() {
        return notes;
    }
    public void setNotes(String notes) {
        this.notes = notes;
    }
}
```

Код класу PasswordCheck.java

```
package org.example;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class PasswordCheck {
    private static final String CHARACTERS = "!@#$%^&*(){}:;'\"\\|.<>/?-_=+*`~";

    public static boolean Check(String password){
        boolean containsLowerCase = false;
        boolean containsUpperCase = false;
        boolean containsNumber = false;
        boolean containsSpecialSymbol = false;
        boolean passwordLength = false;

        Pattern lowercasePattern = Pattern.compile("[a-z]");
        Matcher lowercaseMatcher = lowercasePattern.matcher(password);
        containsLowerCase = lowercaseMatcher.find();

        Pattern uppercasePattern = Pattern.compile("[A-Z]");
        Matcher uppercaseMatcher = uppercasePattern.matcher(password);
        containsUpperCase = uppercaseMatcher.find();

        Pattern numberPattern = Pattern.compile("[0-9]");
        Matcher numberMatcher = numberPattern.matcher(password);
        containsNumber = numberMatcher.find();

        for (int i = 0; i < password.length(); i++) {
            if (CHARACTERS.contains(String.valueOf(password.charAt(i)))) {
                containsSpecialSymbol = true;
                break;
            }
        }
        if (password.length() > 7) passwordLength = true;
```

```
        return (containsLowerCase && containsUpperCase && containsNumber &&
containsSpecialSymbol && passwordLength);
    }
}
```

Код класу PasswordGenerator.java

```
package org.example;

import java.io.File;
import java.security.SecureRandom;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class PasswordGenerator {
    private static final String CHARACTERS =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz" +
            "0123456789!@#$%^&*(){}:;'\"\\|.<>/?-_=+*`~";
    private static final int PASSWORD_LENGTH = 10;

    public List<PasswordData> PassGen(List<PasswordData> passwordDataTransferable, File file)
{
        String password = GeneratePass();
        byte[] encPass;
        try {
            encPass = RSAEncDec.encrypt(password, file);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
        List<PasswordData> passwordData = new ArrayList<>(passwordDataTransferable.size() + 1);
        passwordData = passwordDataTransferable;
        passwordData.add(passwordData.size(), new PasswordData("", "", encPass, "", ""));
        return passwordData;
    }

    private String GeneratePass(){
        List<Character> passwordChars = new ArrayList<>(PASSWORD_LENGTH);
        SecureRandom random = new SecureRandom();
        passwordChars.add((char) (random.nextInt(26) + 'A'));
        passwordChars.add((char) (random.nextInt(26) + 'a'));
        passwordChars.add(Character.forDigit(random.nextInt(10), 10));
        passwordChars.add(CHARACTERS.charAt(random.nextInt(CHARACTERS.length() - 30) +
62));
        for (int i = 4; i < PASSWORD_LENGTH; i++) {
            passwordChars.add(CHARACTERS.charAt(random.nextInt(CHARACTERS.length()))); }
        Collections.shuffle(passwordChars, random);
        StringBuilder sb = new StringBuilder(PASSWORD_LENGTH);
        for (Character c : passwordChars) { sb.append(c); }
        return sb.toString();
    }
}
```

Код класу RSAEncDec.java

```
package org.example;

import java.io.*;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.security.*;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Arrays;
import javax.crypto.Cipher;

public class RSAEncDec {
   public static KeyPair generateRSAKeys() throws NoSuchAlgorithmException {
      KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
      keyPairGenerator.initialize(2048); // Key size
      return keyPairGenerator.generateKeyPair();
   }

   public static byte[] encryptMaster(String data, File file) throws Exception {
      KeyPair keyPair = generateRSAKeys();
      PublicKey publicKey = keyPair.getPublic();
      PrivateKey privateKey = keyPair.getPrivate();

      savePublicKey(publicKey, file.getAbsolutePath().replace(".json", "") + "_public_key.pem");
      savePrivateKey(privateKey, file.getAbsolutePath().replace(".json", "") + "_private_key.pem");

      Cipher cipher = Cipher.getInstance("RSA");
      cipher.init(Cipher.ENCRYPT_MODE, publicKey);
      return cipher.doFinal(data.getBytes());
   }

   public static byte[] encrypt(String data, File file) throws Exception {
      PublicKey publicKey = loadPublicKey(file.getAbsolutePath().replace(".json", "")
            + "_public_key.pem");
      Cipher cipher = Cipher.getInstance("RSA");
      cipher.init(Cipher.ENCRYPT_MODE, publicKey);
      return cipher.doFinal(data.getBytes());
   }

   public static String decrypt(byte[] encryptedData, File file) throws Exception {
      PrivateKey privateKey = loadPrivateKey(file.getAbsolutePath().replace(".json", "")
            + "_private_key.pem");

      Cipher cipher = Cipher.getInstance("RSA");
      cipher.init(Cipher.DECRYPT_MODE, privateKey);
      byte[] decryptedBytes = cipher.doFinal(encryptedData);
      String encPass = new String(decryptedBytes);
      encPass = Arrays.toString(encPass.toCharArray());
      encPass = encPass.replaceAll("[\\[\\],\\s]", "");
```

```
        return encPass;
    }

    public static void savePublicKey(PublicKey publicKey, String fileName) throws IOException {
        X509EncodedKeySpec keySpec = new X509EncodedKeySpec(publicKey.getEncoded());
        try (FileOutputStream fos = new FileOutputStream(fileName)) {
            fos.write(keySpec.getEncoded());
        }
    }

    public static PublicKey loadPublicKey(String fileName) throws Exception {
        Path path = Paths.get(fileName);
        byte[] bytes = Files.readAllBytes(path);
        X509EncodedKeySpec keySpec = new X509EncodedKeySpec(bytes);
        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        return keyFactory.generatePublic(keySpec);
    }

    public static void savePrivateKey(PrivateKey privateKey, String fileName) throws IOException
{
        PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(privateKey.getEncoded());
        try (FileOutputStream fos = new FileOutputStream(fileName)) {
            fos.write(keySpec.getEncoded());
        }
    }

    public static PrivateKey loadPrivateKey(String fileName) throws Exception {
        Path path = Paths.get(fileName);
        byte[] bytes = Files.readAllBytes(path);
        PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(bytes);
        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        return keyFactory.generatePrivate(keySpec);
    }
}
```

Код класу CRC.java

```
package org.example;

import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.util.zip.CRC32;

public class CRC {

    public static long calculateChecksum(String filePath) {
        CRC32 crc = new CRC32();
        try (FileInputStream fis = new FileInputStream(filePath)) {
            byte[] buffer = new byte[8192];
            int bytesRead;
            while ((bytesRead = fis.read(buffer)) != -1) {
```

```
                crc.update(buffer, 0, bytesRead);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return crc.getValue();
    }

    public static void saveChecksumToFile(String filePath, long checksum) {
        String checksumFilePath = filePath.replace(".json", "") + ".checksum";
        try (FileWriter writer = new FileWriter(checksumFilePath)) {
            writer.write(Long.toString(checksum));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Додаток Б. Коди блочного та інтеграційного тестувань

Код класу PasswordDataTest.java

```
package org.example;

import org.junit.jupiter.api.*;

public class PasswordDataTest {
    PasswordData passwordData;
    String title = "title";
    String userName = "userName";
    byte[] password = {12, 23, 45, 32};
    String url = "url";
    String notes = "notes";

    @BeforeEach
    void setUp(){
        passwordData = new PasswordData(title, userName, password, url, notes);
    }

    @Test
    void ConstuctorTest() {
        Assertions.assertEquals(title, passwordData.getTitle());
        Assertions.assertEquals(userName, passwordData.getUserName());
        Assertions.assertEquals(password, passwordData.getPassword());
        Assertions.assertEquals(url, passwordData.getUrl());
        Assertions.assertEquals(notes, passwordData.getNotes());
    }

    @Test
    void SettersTest() {
        passwordData.setTitle("new_title");
        passwordData.setUserName("new_userName");
        password = new byte[]{16, 25, 93, 54, 76};
        passwordData.setPassword(password);
        passwordData.setUrl("new_url");
        passwordData.setNotes("new_notes");

        Assertions.assertEquals("new_title", passwordData.getTitle());
        Assertions.assertEquals("new_userName", passwordData.getUserName());
        Assertions.assertEquals(password, passwordData.getPassword());
        Assertions.assertEquals("new_url", passwordData.getUrl());
        Assertions.assertEquals("new_notes", passwordData.getNotes());
    }
}
```

Код класу CRCTest.java

```
package org.example;

import org.junit.jupiter.api.*;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class CRCTest {
    File file = new File("D:\\PasswordTest\\123.json");
    long newCheckSum;

    @Test
    void CRCTest() {
        String checksumFilePath = file.getAbsolutePath().replace(".json", "") + ".checksum";
        long savedCheckSum = 0L;
        try (BufferedReader reader = new BufferedReader(new FileReader(checksumFilePath))) {
            String line = reader.readLine();
            savedCheckSum = Long.parseLong(line);
        } catch (IOException e) {
            e.printStackTrace();
        }

        newCheckSum = CRC.calculateChecksum(file.getAbsolutePath());
        Assertions.assertEquals(savedCheckSum, newCheckSum, "must be true");
    }

    @Test
    void saveChecksumToFileTest() {
        newCheckSum = CRC.calculateChecksum(file.getAbsolutePath());
        CRC.saveChecksumToFile(file.getAbsolutePath(), newCheckSum);

        long savedCheckSum = 0L;
        try (BufferedReader reader = new BufferedReader(new
FileReader("D:\\PasswordTest\\123.checksum"))) {
            String line = reader.readLine();
            savedCheckSum = Long.parseLong(line);
        } catch (IOException e2) {
            e2.printStackTrace();
        }

        Assertions.assertEquals( newCheckSum,savedCheckSum, "must be true");
    }
}
```

Код класу PasswordCheckTest.java

```
package org.example;

import org.junit.jupiter.api.*;

public class PasswordCheckTest {
    String password;

    @Test
    void CheckTest1() {
        password = "p@55VV0rD";
        Assertions.assertTrue(PasswordCheck.Check(password), "must be true");
    }

    @Test
    void CheckTest2() {
        password = "password";
        Assertions.assertFalse(PasswordCheck.Check(password), "must be false");
    }

    @Test
    void CheckTest3() {
        password = "password";
        Assertions.assertFalse(PasswordCheck.Check(password), "must be false");
    }

    @Test
    void CheckTest4() {
        password = "Klymenko";
        Assertions.assertFalse(PasswordCheck.Check(password), "must be false");
    }

    @Test
    void CheckTest5() {
        password = "12345678";
        Assertions.assertFalse(PasswordCheck.Check(password), "must be false");
    }

    @Test
    void CheckTest6() {
        password = "";
        Assertions.assertFalse(PasswordCheck.Check(password), "must be false");
    }

    @Test
    void CheckTest7() {
        password = "qwe123";
        Assertions.assertFalse(PasswordCheck.Check(password), "must be false");
    }
```

```
    @Test
    void CheckTest8() {
        password = "!@#$%^&*()";
        Assertions.assertFalse(PasswordCheck.Check(password), "must be false");
    }

    @Test
    void CheckTest9() {
        password = "5pY@n45Ch_d";
        Assertions.assertTrue(PasswordCheck.Check(password), "must be true");
    }
}
```

Код класу MainWindowTest.java

```
package org.example;

import javafx.stage.Stage;
import org.junit.jupiter.api.*;
import org.testfx.framework.junit5.ApplicationTest;

import java.util.ArrayList;
import java.util.List;

public class MainWindowTest extends ApplicationTest {
    MainWindow mainWindow;
    List<PasswordData> passwordData = new ArrayList<PasswordData>();

    @Override
    public void start(Stage stage) {
        mainWindow = new MainWindow(passwordData, false, null);
        mainWindow.pack();
        mainWindow.setVisible(true);
    }

    @Test
    public void testMainWindow() {
    }
}
```

Додаток В

ПРОТОКОЛ ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Засіб захищеного зберігання паролів
Автор роботи: Клименко Володимир Олександрович
Тип роботи: бакалаврська дипломна робота .
Підрозділ кафедра захисту інформації ФІТКІ .

Показники звіту подібності Unicheck

Оригінальність – 95,7%. Схожість – 4,3%.

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Каплун В. А.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Володимир КЛИМЕНКО
(підпис) (прізвище, ініціали)

Керівник роботи _____ Юрій БАРИШЕВ
(підпис) (прізвище, ініціали)

ІЛЮСТРАТИВНА ЧАСТИНА

ЗАСІБ ЗАХИЩЕНОГО ЗБЕРІГАННЯ ПАРОЛІВ

Виконав: студент 4 курсу групи 1БС-19б
спеціальності 125 Кібербезпека
_____Володимир КЛИМЕНКО
16 червня 2023 р.

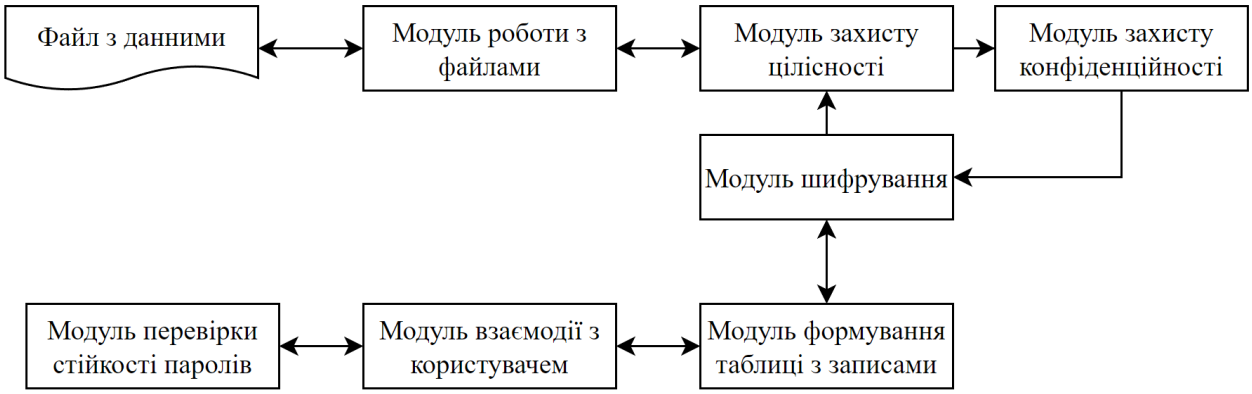
Керівник: к.т.н., доцент, доцент каф. ЗІ
_____Юрій БАРИШЕВ
16 червня 2023 р.

РЕЗУЛЬТАТИ ПОРІВНЯЛЬНОГО АНАЛІЗУ ЗАСОБІВ ЗАХИСТУ ПАРОЛІВ

	RoboForm	1Password	Keeper	LastPass
Обмеження для безкоштовної версії	Відсутня можливість резервного копіювання в хмару	Тільки платна	Тільки для мобільного пристрою	Синхронізація між пристроями одного типу
Багатофакторна автентифікація	В платній версії	В платній версії	+	+
Біометричний вхід	+	В платній версії	+	+
Заповнення форм	+	В платній версії	+	+
Генератор паролів	+	В платній версії	+	+
Аварійний доступ	В платній версії	В платній версії	В платній версії	В платній версії
Надійність паролю	+	В платній версії	В платній версії	+
Платформи	Windows, Mac, iOS, Android, Linux	Windows, Mac, iOS, Android. Linux	Windows, Mac, iOS, Android, Linux	Windows, Mac, iOS, Android, Linux
Браузери	Chrome, Firefox, Safari, Edge, IE	Chrome, Firefox, Safari, Edge, Brave	Chrome, Firefox, Safari, Edge, IE, Opera	Chrome, Firefox, Safari, Edge, Opera

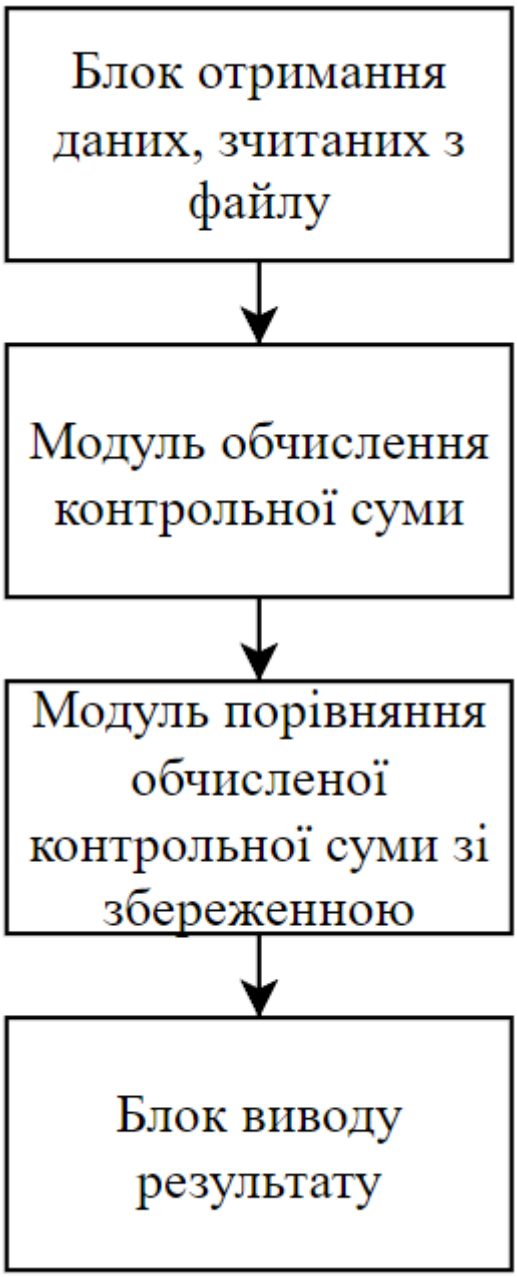
	Bitwarden	Enpass	NordPass	Dashlane
Обмеження для безкоштовної версії	Відсутній TOTP автентифікатор та можливість шифрування файлів	25 паролів для мобільної версії	Для одного пристрою	Для одного пристрою
Багатофакторна автентифікація	+	+	+	+
Біометричний вхід	+	+	+	+
Заповнення форм	+	+	+	+
Генератор паролів	+	+	+	+
Аварійний доступ	В платній версії	–	В платній версії	–
Надійність паролю	+	+	В платній версії	+
Платформи	Windows, Mac, iOS, Android, Linux	Windows, Mac, iOS, Android, Linux	Windows, Mac, iOS, Android, Linux	Windows, Mac, iOS, Android, Linux
Браузери	Chrome, Firefox, Safari, Edge, Brave, Opera, Vivaldi, Tor	Chrome, Firefox, Safari, Edge, Opera, Vivaldi	Chrome, Firefox, Safari, Edge, Brave, Opera	Chrome, Firefox, Safari, Edge, IE

УЗАГАЛЬНЕНА АРХІТЕКТУРА ЗАСОБУ ЗАХИЩЕНОГО ЗБЕРІГАННЯ ПАРОЛІВ

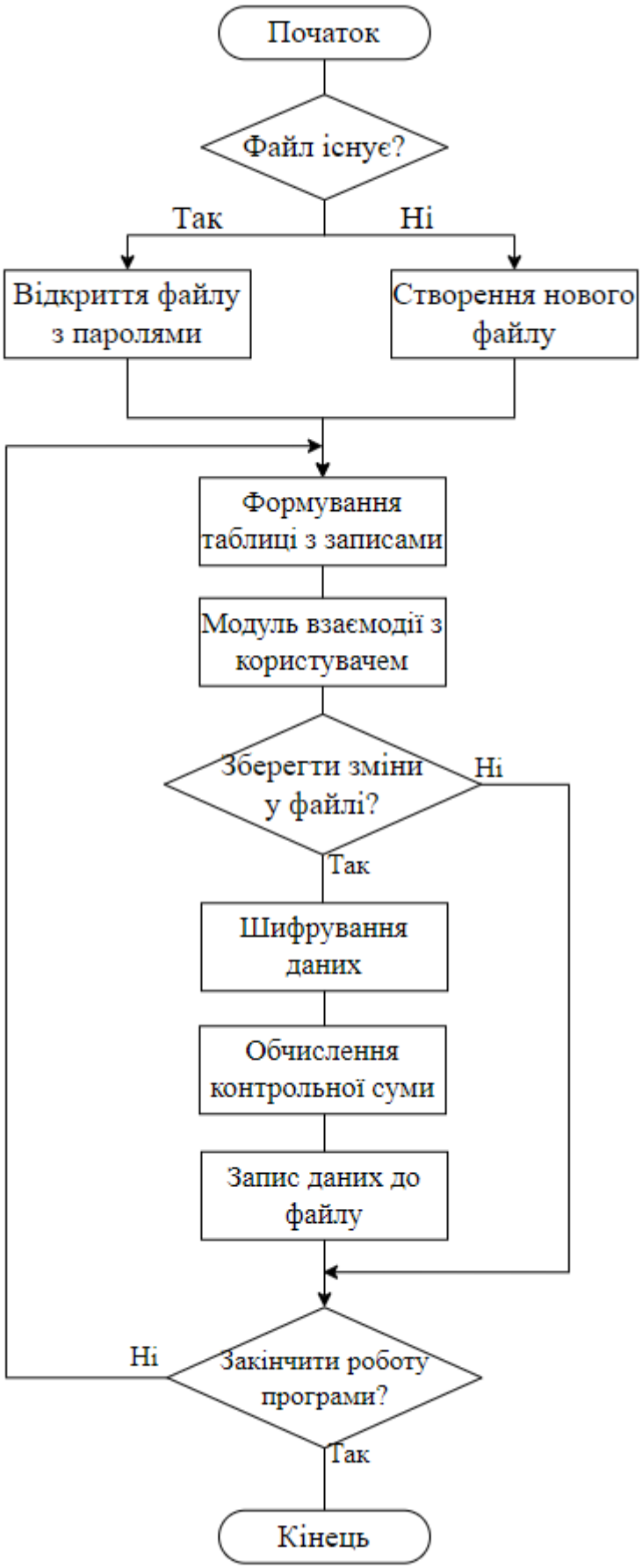


СТРУКТУРА МОДУЛЯ ЗАХИСТУ КОНФІДЕНЦІЙНОСТІ

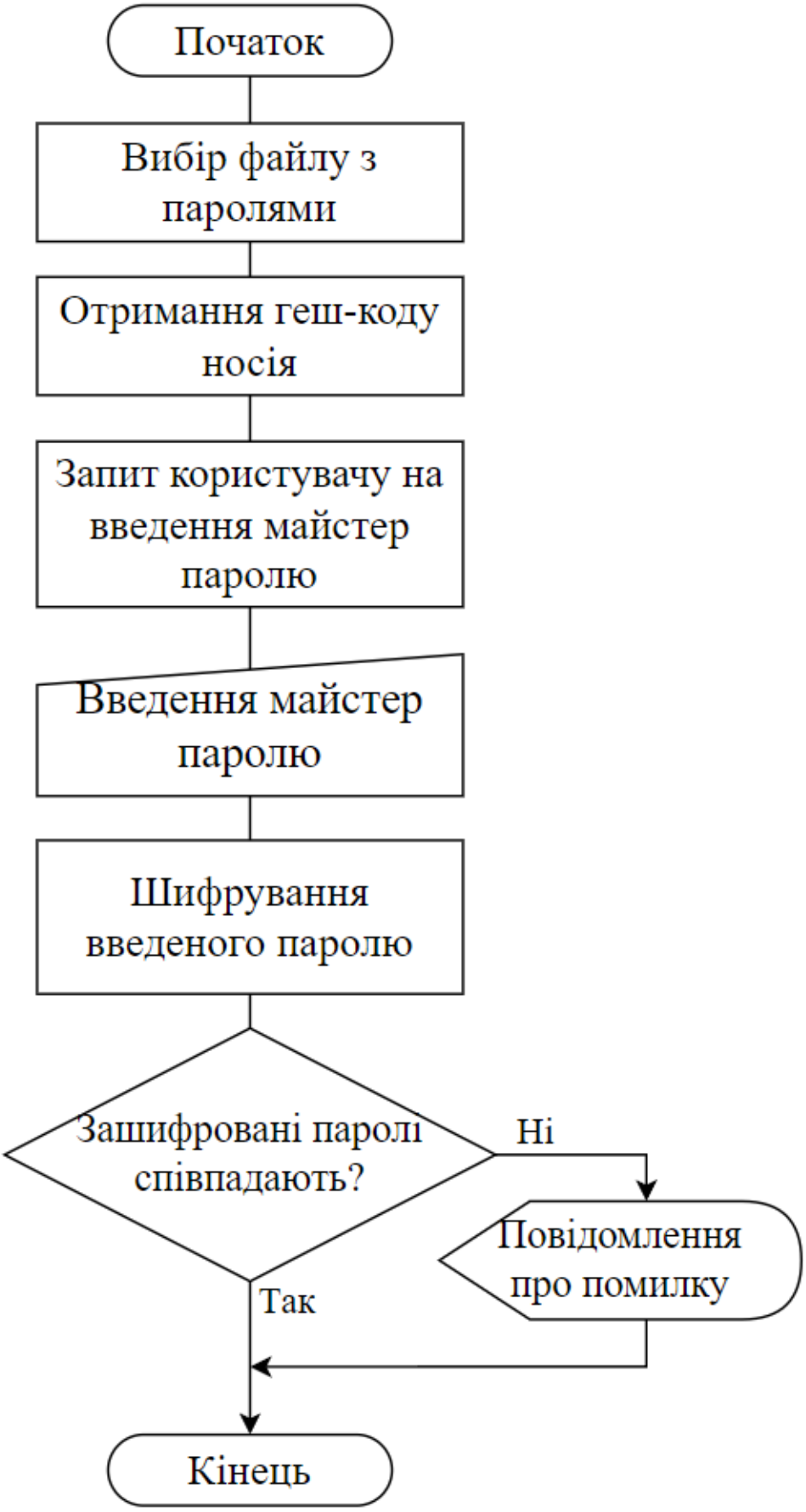
СТРУКТУРА МОДУЛЯ ЗАХИСТУ ЦІЛІСНОСТІ



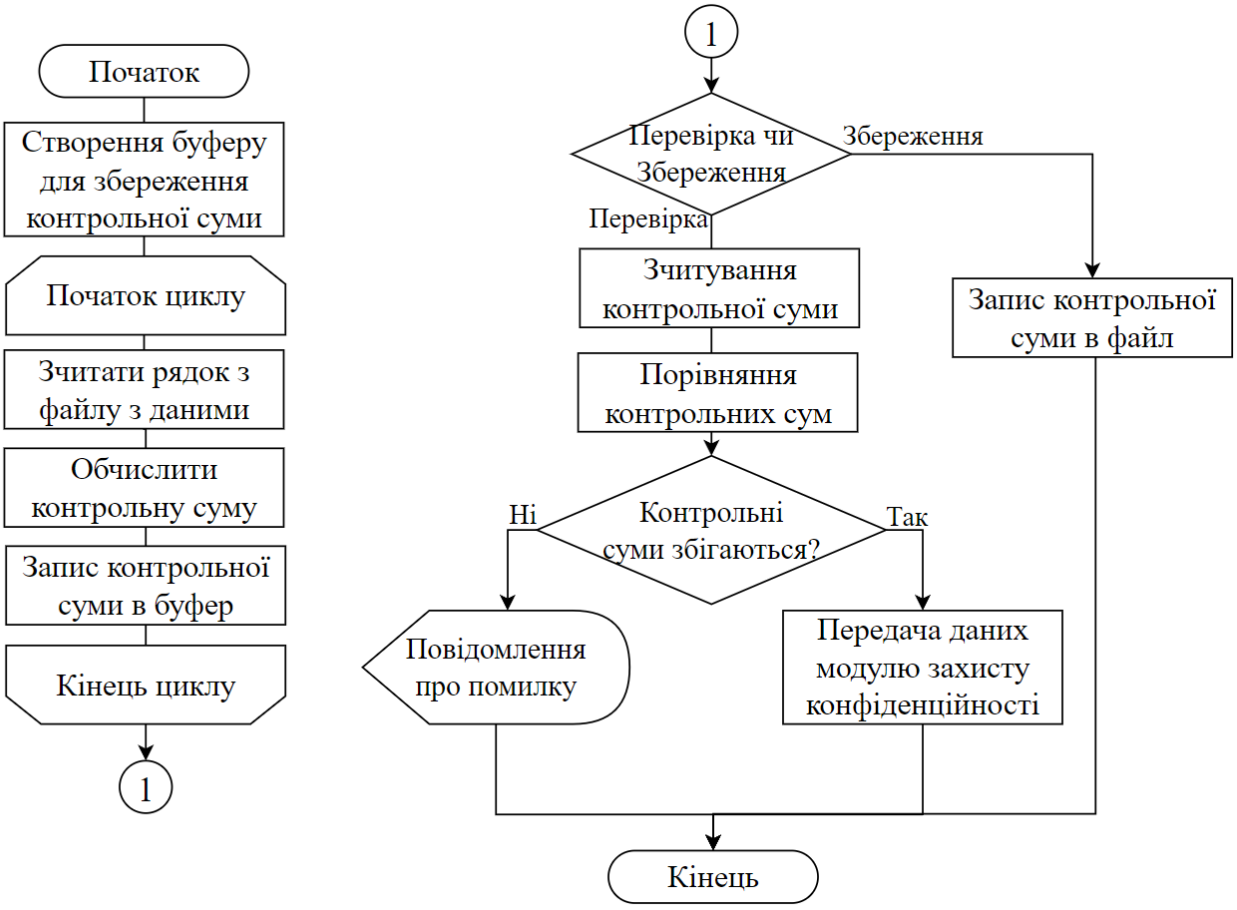
УЗАГАЛЬНЕНИЙ АЛГОРИТМ РОБОТИ ЗАСОБУ



АЛГОРИТМ ЗАХИСТУ КОНФІДЕНЦІЙНОСТІ



АЛГОРИТМ ЗАХИСТУ ЦІЛІСНОСТІ ПАРОЛІВ



ПОРІВНЯЛЬНИЙ АНАЛІЗ ЗАСОБІВ РОЗРОБКИ

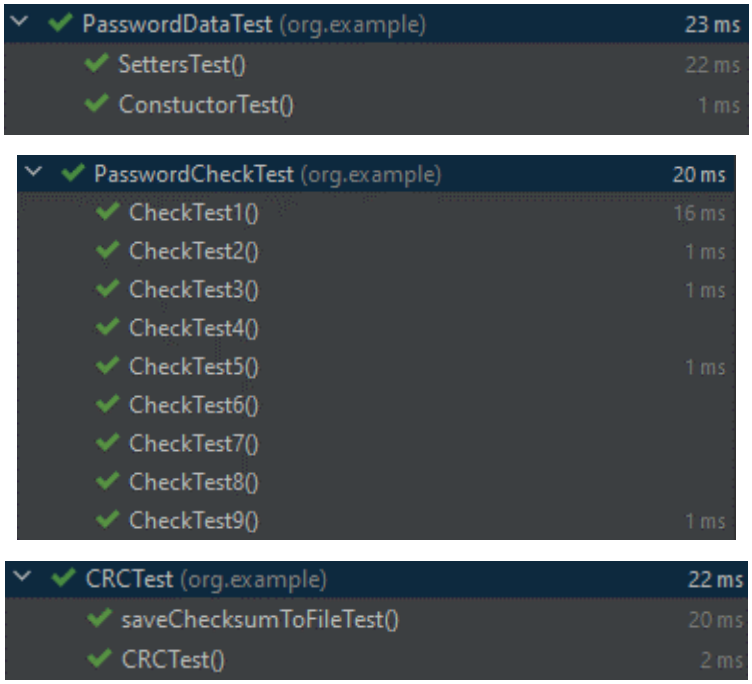
Результати порівняльного аналізу мов програмування

	Java	C#	Python
Кросплатформеність	+	Тільки з підключенням .NET	+
Мова високого рівня	+	+	+
Об’єктно-орієнтованість	+	+	+
Вбудована система безпеки	У Java має більший функціонал	+	+
Збирач сміття	+	+	+
Багатопоточність	+	+	+
Статична типізація	+	+	-
Простий синтаксис	-	-	+
JIT-компіляція	+	+	-
Вузьке направлення, що відповідає цілям даного засобу	+	-	-

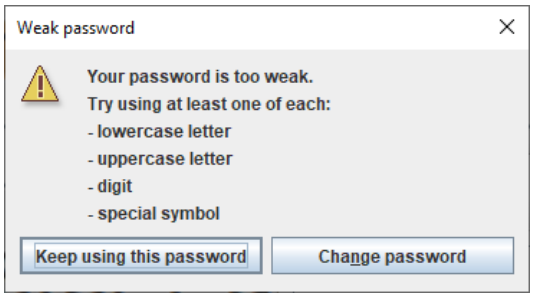
Результати порівняльного аналізу середовищ розробки

	IntelliJ IDEA	Eclipse	Apache NetBeans
Розумне завершення коду	Найкраще серед всіх IDE	+	+
Зручний дизайн	+	+	+
Підтримка Git	Найкраща серед всіх IDE	+	+
Безкоштовність	З обмеженим функціоналом	+	+
Хмарна версія	-	+	-
Підтримка плагінів	+	+	+
Висока швидкодія	+	-	-

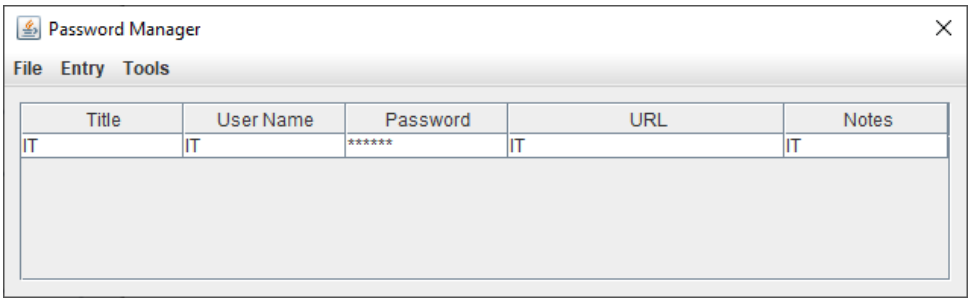
РЕЗУЛЬТАТИ БЛОКОВОГО ТЕСТУВАННЯ



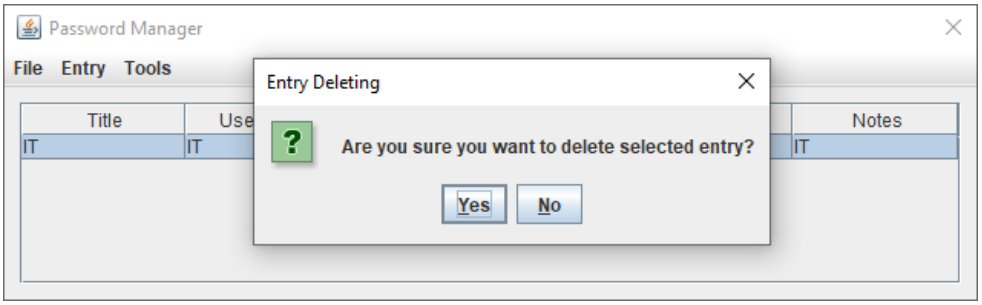
РЕЗУЛЬТАТИ ІНТЕГРАЦІЙНОГО ТЕСТУВАННЯ



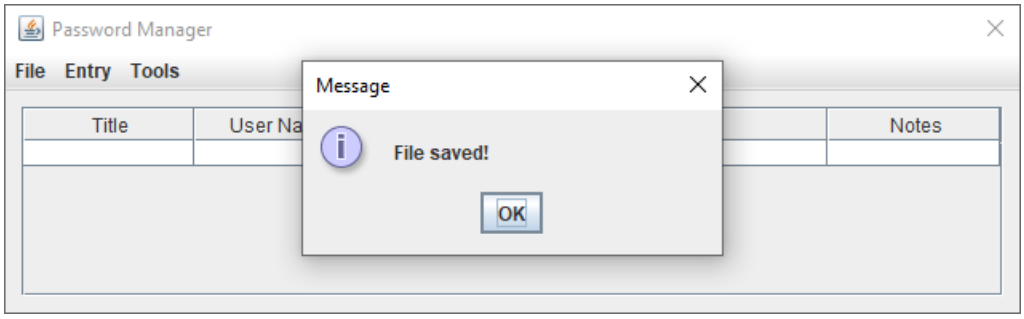
Результат виводу вікна після перевірки паролю на стійкість



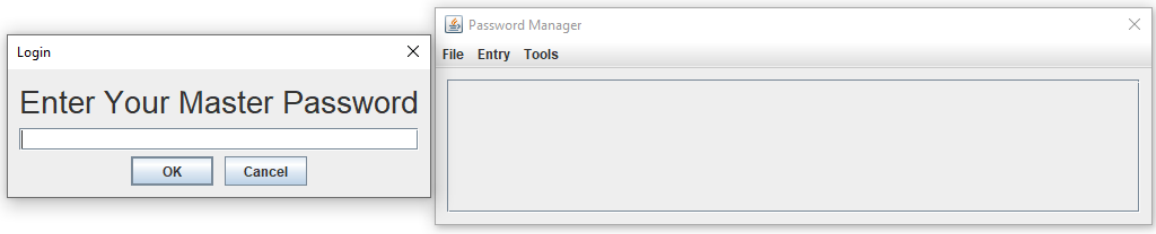
Результат додавання нового запису до таблиці



Результат відкриття вікна для підтвердження видалення запису



Зображення повідомлення про успішне збереження файлу



Результат блокування робочого простору