МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра Інформаційних технологій

Кваліфікаційна робота магістра

на тему: Розробка алгоритмів для оптимізації генерації та аналізу паролів і ключів з використанням методів та засобів ШІ

Виконала студентка групи МІС-22
спеціальності 122 Комп'ютерні науки
Бойко Віталій Вікторович

Керівник к.т.н., доцент
Фразе-Фразенко Олексій Олексійович

Рецензент Начальник ІОЦ ОНЕУ
к.т.н., Домаскін Олег Михайлович

Одеса 2023

АНОТАЦІЯ

на магістерську кваліфікаційну роботу
«Розробка алгоритмів для оптимізації генерації та аналізу паролів і ключів з використанням методів та засобів ШІ»
студента Бойко Віталія Вікторовича

Актуальність теми магістерської кваліфікаційної роботи полягає в постійному зростанні кількості кіберзагроз та необхідності забезпечення високого рівня кібербезпеки. Розробка ефективних алгоритмів для оптимізації генерації та аналізу паролів і ключів з використанням методів та засобів шифрування інформації є важливим етапом у зміцненні кіберзахисту. Застосування сучасних підходів до розробки алгоритмів, спрямованих на підвищення стійкості та складності парольних систем, є критичним для запобігання несанкціонованому доступу та забезпечення конфіденційності інформації у віртуальному середовищі.

Мета роботи – розробка та оптимізація алгоритмів для генерації та аналізу паролів і ключів з використанням методів та засобів ШІ.

Об'єкт дослідження – сучасні засоби та методи розробки алгоритмів для генерації та аналізу паролів і ключів з використанням ШІ. Дослідження охоплює аспекти, пов'язані з криптографічними принципами, алгоритмами генерації, а також методами аналізу надійності та стійкості паролів.

Предмет дослідження – технології розробки алгоритмів для генерації та аналізу паролів і ключів, які використовуються для забезпечення інформаційної безпеки. Особлива увага буде приділена оптимізації цих алгоритмів з урахуванням сучасних вимог до безпеки та ефективності їхнього використання.

Методи дослідження – аналіз сучасних програмних продуктів та технологій в області криптографії та безпеки інформації. Для визначення ефективності розроблених алгоритмів буде використано тестування.

В роботі було проведено аналіз та розробку ефективних алгоритмів для оптимізації генерації та аналізу паролів і ключів з використанням методів та

засобів ШІ. Було створено мобільний додаток на платформі Android для реалізації розроблених алгоритмів. Розроблений мобільний додаток використовує сучасні методи шифрування та засоби криптографії для забезпечення надійного зберігання паролів. Також користувачі цього мобільного додатку можуть легко генерувати складні паролі та зберігати їх у додатку, забезпечуючи високий рівень безпеки своїх конфіденційних даних.

Магістерська кваліфікаційна робота містить 76 сторінок, 40 рисунків та 16 джерел.

Ключові слова: ШИФРУВАННЯ, ГЕНЕРАЦІЯ, ПАРОЛЬ, ANDROID, МОБІЛЬНИЙ ДОДАТОК, JAVA, ANDROID STUDIO.

SUMMARY

for a master's degree

"Development of algorithms for optimising the generation and analysis of passwords and keys using AI methods and tools"

of student Vitalii Viktorovych Boiko

The relevance of the topic of the master's thesis lies in the constant increase in the number of cyber threats and the need to ensure a high level of cybersecurity. The development of effective algorithms to optimise the generation and analysis of passwords and keys using information encryption methods and tools is an important step in strengthening cyber defence. The use of modern approaches to the development of algorithms aimed at increasing the stability and complexity of password systems is critical to prevent unauthorised access and ensure the confidentiality of information in the virtual environment.

The purpose of the study is to develop and optimise algorithms for generating and analysing passwords and keys using AI methods and tools.

The object of research is modern tools and methods for developing algorithms for generating and analysing passwords and keys using AI. The research covers aspects related to cryptographic principles, generation algorithms, and methods for analysing password strength and stability.

The subject of the study is the technology of developing algorithms for generating and analysing passwords and keys used to ensure information security. Particular attention will be paid to the optimisation of these algorithms with regard to modern requirements for security and efficiency of their use.

Research methods – analysis of modern software products and technologies in the field of cryptography and information security. Testing will be used to determine the effectiveness of the developed algorithms.

The work analyses and develops effective algorithms for optimising the generation and analysis of passwords and keys using AI methods and tools. A mobile

application on the Android platform has been created to visualise the developed algorithms. The developed mobile application uses modern encryption methods and cryptography tools to ensure secure password storage. Also, users of this mobile application can easily generate complex passwords and store them in the application, ensuring a high level of security for their confidential data.

Master's qualification work contains 76 pages, 40 figures and 16 sources.

Keywords: ENCRYPTION, GENERATION, PASSWORD, ANDROID, MOBILE APPLICATION, JAVA, ANDROID STUDIO.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних.

ОС – операційна система.

ПК — персональний комп'ютер.

ШІ – шифрування інформації.

AES (Advanced Encryption Standard) – симетричний блочний алгоритм шифрування.

APK (Android Package Kit) – формат архівних виконуваних файлів-додатків для Android і низки інших операційних систем, заснованих на Android.

CPU (Central Processing Unit) – функціональна частина комп'ютера, що призначена для інтерпретації команд.

CSV (Comma-Separated Values) – текстовий формат, призначений для подання табличних даних.

DES (Data Encryption Standard) – симетричний блочний алгоритм шифрування.

DSA (Digital Signature Algorithm) – асиметричний алгоритм шифрування.

ECDSA (Elliptic Curve Digital Signature Algorithm) – асиметричний алгоритм шифрування.

GCM (Galois/Counter Mode) – широко застосовуваний режим роботи симетричних блокових шифрів, що має високу ефективність і продуктивність.

GPS (Global Positioning System) – супутникова система навігації, що забезпечує вимірювання відстані, часу і визначає місце розташування у всесвітній системі координат.

HMAC (Hash-based Message Authentication Code) – механізм перевірки цілісності інформації, що передається або зберігається в ненадійному середовищі.

IDE (Integrated Development Environment) – комплекс програмних засобів, що використовується програмістами для розробки програмного забезпечення.

IDEA (International Data Encryption Algorithm) – симетричний блочний алгоритм шифрування.

IBM (International Business Machines) – американська компанія, один із найбільших у світі виробників і постачальників апаратного та програмного забезпечення, а також IT-сервісів і консалтингових послуг.

ISO/IEC 27001 – міжнародний стандарт з інформаційної безпеки.

JVM (Java Virtual Machine) – віртуальна машина для виконання байт-коду Java.

KDF (Key Derivation Function) – функція, що формує один або кілька секретних ключів на основі секретного значення (головний ключ, пароль або парольна фраза) за допомогою псевдовипадкової функції.

MITM (Man in the middle) – вид атаки в криптографії та комп'ютерній безпеці, коли зловмисник таємно ретранслює і, за необхідності, змінює зв'язок між двома сторонами, які вважають, що вони безпосередньо спілкуються одна з одною.

MMS (Multimedia Messaging Service) – система передавання мультимедійних повідомлень (зображень, мелодій, відео) у мережах стільникового зв'язку.

NIST (National Institute of Standards and Technology) – національний інститут стандартів і технологій.

PBKDF2 (Password-Based Key Derivation Function 2) – стандарт формування ключа на основі пароля.

PDF (Portable Document Format) – міжплатформний відкритий формат електронних документів.

RC4 (Rivest cipher 4) – симетричний потоковий алгоритм шифрування.

RSA (Rivest, Shamir та Adleman) – асиметричний алгоритм шифрування.

RSA Security (Rivest, Shamir та Adleman Security) – корпорація, яка спеціалізується на виробництві продуктів пов'язаних з інформаційною безпекою.

SEAL (Software-optimized Encryption Algorithm) – симетричний потоковий алгоритм шифрування.

SHA (Secure Hash Algorithm) – сімейство криптографічних хеш-функцій, здатних приймати повідомлення довільної довжини й обчислювати унікальний хеш-код фіксованої довжини.

SMS (Short Message Service) – технологія приймання та передавання коротких текстових повідомлень за допомогою мобільного телефону.

SOC-2 (Service Organization Control 2) – стандарт безпеки даних.

UML (Unified Modeling Language) – уніфікована мова моделювання.

3DES (Triple Data Encryption Standard) – симетричний блочний алгоритм шифрування.

ВСТУП

Розробка алгоритмів для оптимізації генерації та аналізу паролів і ключів з використанням методів і засобів ШІ набуває особливої актуальності в контексті зростання обсягу цифрової інформації та збільшення загроз кібербезпеці. У сучасному цифровому світі, де безпека особистої інформації та конфіденційності є важливими аспектами, виробництво ефективних алгоритмів стає необхідністю.

Підходи до розроблення алгоритмів для оптимізації генерації та аналізу паролів і ключів у контексті застосування методів і засобів ШІ стають об'єктом серйозного вивчення та дослідження. Широка доступність обчислювальних ресурсів і високий рівень кіберзагроз вимагають розроблення технологій, спрямованих на забезпечення високого рівня безпеки особистої інформації та захист від небажаних вторгнень.

Метою даної кваліфікаційної роботи є розробка ефективних алгоритмів, що забезпечать генерацію та аналіз паролів і ключів. Розглядатимуться сучасні методи та засоби криптографічного захисту з огляду на специфіку їх використання в мобільних додатках.

Для досягнення поставленої мети в роботі необхідно вирішити наступні завдання:

- виконати аналіз предметної області, аналіз подібних мобільних додатків для генерації та аналізу паролів;
- обґрунтувати вибір програмних засобів та технологій для створення мобільного додатку;
- розробити мобільний додаток, що міститиме наступні функціональні можливості: генерація паролів, ефективний аналіз паролів, захищене зберігання паролів у додатку;
- перевірити працездатність додатка.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Методи шифрування

Шифрування або криптографія – це оборотне перетворення інформації, спрямоване на те, щоб приховати її від сторонніх осіб і надати доступ авторизованим користувачам. Шифрування забезпечує 3 компоненти захисту інформації:

- конфіденційність. Шифрування приховує інформацію від неавторизованих користувачів під час передачі або зберігання.
- чесність. Шифрування використовується для запобігання зміні інформації під час передачі або зберігання.
- ідентифікованість. Шифрування допомагає автентифікувати джерело інформації та не дає змоги відправнику інформації заперечувати, що він дійсно був відправником даних.

Шифрування використовує математичні алгоритми та ключі. Алгоритм – це набір математичних операцій, необхідних для виконання певного процесу шифрування, а ключі – це рядки тексту і цифр, які використовуються для шифрування і дешифрування даних.

Існує два основних типи шифрування – симетричне й асиметричне, які розрізняються за типом ключів, що використовуються для шифрування і дешифрування.

Симетричне шифрування – найстаріший метод шифрування, відомий людству. За майже всю історію криптографії, що налічує близько 4000 років, це був єдиний метод шифрування інформації.

Симетричне шифрування, також зване шифруванням із закритим ключем, – це коли дані шифруються та дешифруються відправником і одержувачем з використанням одного й того самого секретного ключа. Це означає, що ключ має передаватися безпечно, щоб тільки одержувач міг отримати до нього доступ.

Ось як працює процес захисту інформації за допомогою симетричного шифрування:

– відправник (або одержувач) обирає алгоритм шифрування, генерує ключ, інформує одержувача (або відправника, залежно від обставин) про обраний алгоритм і надсилає ключ захищеним каналом зв'язку.
– відправник шифрує повідомлення за допомогою ключа і відправляє зашифроване повідомлення одержувачу.
– одержувач отримує зашифроване повідомлення і розшифровує його за допомогою того самого ключа.

Існує два основних типи симетричних шифрів: блокові та потокові.

При блоковому шифруванні інформація ділиться на блоки фіксованої довжини (наприклад, 64 або 128 біт). Потім ці блоки шифруються один за одним. Ключ застосовується до кожного блоку у встановленому порядку. Зазвичай це передбачає кілька циклів змішування і заміни. Блоковий шифр є важливим компонентом багатьох криптографічних протоколів і широко використовується для захисту даних, що передаються мережею.

Кожен вихідний символ перетворюється на зашифрований у потоковому шифрі, залежно від використовуваного ключа і його місця розташування у вихідному тексті. Потокові шифри мають вищу швидкість шифрування, ніж блокові шифри, але вони також мають більше вразливостей.

Симетричних шифрів досить багато. Ось приклади найвідоміших алгоритмів симетричного шифрування.

Блокові шифри:

– DES – це алгоритм шифрування, розроблений IBM і схвалений урядом США 1977 року як офіційний стандарт. Розмір блоку DES становить 64 біти. Нині вважається застарілим і невикористаним.
– 3DES був створений 1978 року на основі алгоритму DES для усунення головного недоліку останнього: невеликої довжини ключа (56 біт), який можна зламати перебором. Швидкість 3DES утричі нижча, ніж у

DES, але криптографічна безпека набагато вища. Алгоритм 3DES заснований на DES, тому для його реалізації можна використовувати програми, створені для DES. Він все ще використовується, особливо в індустрії електронних платежів, але поступово замінюється більш новими алгоритмами.

– AES – це алгоритм шифрування з розміром блоку 128 біт і ключем 128/192/256 біт. AES був розроблений у 2001 році як заміна DES. Наразі його вважають одним із найефективніших і найбезпечніших симетричних шифрів і тому широко використовують.
– IDEA – це алгоритм, розроблений 1991 року швейцарською компанією Ascom. Він використовує 128-бітний ключ і 64-бітний розмір блоку. Хоча зараз він також вважається застарілим, його все ще використовують.

Потокові шифри:

– RC4 – алгоритм, розроблений 1987 року американською компанією RSA Security. Він став популярним завдяки простоті апаратної та програмної реалізації та високій швидкості роботи алгоритмів. Нині його вважають застарілим і недостатньо безпечним, але все ще використовують.
– SEAL розробила IBM 1993 року. Алгоритм оптимізований і рекомендований для 32-бітних процесорів. Це один із найшвидших шифрувальників, який вважається дуже безпечним [1].

Найбільш помітною перевагою симетричного шифрування є його простота, оскільки він використовує один ключ як для шифрування, так і для дешифрування. Таким чином, симетричні алгоритми шифрування значно швидші за асиметричні та потребують меншої обчислювальної потужності.

Водночас той факт, що для шифрування і дешифрування використовується один і той самий ключ, є основною вразливістю систем симетричного шифрування. Необхідність передати ключ іншій стороні є вразливістю безпеки, бо, якщо він потрапить у чужі руки, інформація буде розшифрована.

Відповідно, особливу увагу слід приділяти можливим способам перехоплення ключа та підвищенню безпеки передачі.

Асиметричне шифрування – це відносно нова криптографічна система, що з'явилася в 1970-х роках. Його основна мета – усунути вразливість симетричного шифрування, тобто використання одного ключа.

Асиметричне шифрування, також зване шифруванням із відкритим ключем, – це криптографічна система, що використовує два ключі. Відкритий ключ може бути переданий незахищеним каналом і використовується для шифрування повідомлення. Для розшифрування повідомлення використовується закритий ключ, відомий тільки одержувачу.

Пара ключів математично пов'язана один з одним, тому ви можете обчислити відкритий ключ, знаючи приватний, але не навпаки.

Ось як працює асиметричне шифрування:

– одержувач обирає алгоритм шифрування і генерує пару відкритого і закритого ключів.
– одержувач передає відкритий ключ відправнику.
– відправник шифрує повідомлення за допомогою відкритого ключа і надсилає зашифроване повідомлення одержувачу.
– одержувач отримує зашифроване повідомлення і розшифровує його, використовуючи свій закритий ключ.

Приклади відомих алгоритмів асиметричного шифрування:

– RSA, найстаріший алгоритм асиметричного шифрування, було опубліковано 1977 року, його назвали на честь його творців, американських учених із Массачусетського технологічного інституту Рона Рівеста, Аді Шаміра та Леонарда Адлемана. Це відносно повільний алгоритм, який часто використовують у гібридних системах шифрування в поєднанні із симетричними алгоритмами.
– DSA був створений 1991 року Національним інститутом стандартів і технологій NIST у США. Використовується для автентифікації циф-

рового підпису. Електронний підпис створюється за допомогою закритого ключа в цьому алгоритмі, але може бути перевірений за допомогою відкритого ключа. Це означає, що тільки власник підпису може створити підпис, але будь-хто може перевірити його справжність.

– ECDSA – це алгоритм із відкритим ключем для створення цифрового підпису. Це варіант DSA, в якому використовується криптографія на основі еліптичних кривих. ECDSA використовується в мережі Біткойн для підписання транзакцій.
– Діффі-Геллман був опублікован 1976 року американськими криптографами Вітфілдом Діффі та Мартіном Геллманом. Це криптографічний протокол, який дає змогу двом або більше сторонам отримати спільний закритий ключ, використовуючи незахищений канал зв'язку. Ключ використовується для шифрування іншої частини обміну з використанням симетричних алгоритмів шифрування. Схема розподілу ключів захищеними каналами, запропонована Діффі та Геллманом, стала важливим проривом у криптографії, оскільки зняла головну проблему класичної криптографії – розподіл ключів.

Найбільш очевидною перевагою шифрування з асиметричним ключем є його безпека, оскільки закритий ключ не потрібно нікому передавати. Це значно спрощує управління ключами у більших мережах.

Однак у цього методу шифрування є й недоліки. Одним із прикладів є вища складність, нижча швидкість і вищий попит на обчислювальні ресурси. Крім того, попри високу безпеку асиметричного шифрування, він, як і раніше, вразливий до атаки "людина посередині" (MITM), під час якої зловмисник перехоплює відкритий ключ, відправлений одержувачем відправнику. Потім зловмисник створює свою власну пару ключів і маскується під одержувача, відправляючи хибний відкритий ключ відправнику, який, на його думку, є відкритим ключем, відправленим одержувачем. Зловмисник перехоплює зашифровані повідомлення від відправника до одержувача, розшифровує їх своїм закритим ключем, повторно шифрує їх відкритим ключем одержувача і

відправляє повідомлення одержувачу. У такий спосіб жоден з учасників не здогадується, що третя особа перехоплює повідомлення або замінює його неправдивим. Це підкреслює необхідність аутентифікації з відкритим ключем.

Основна відмінність між симетричним і асиметричним шифруванням полягає у використанні одного ключа порівняно з парою ключів. Решта відмінностей між цими методами є лише наслідком цієї основної відмінності.

Симетричні алгоритми гарні для передачі великих обсягів зашифрованих даних. Крім того, щоб організувати двосторонній обмін даними з використанням асиметричного алгоритму, обидві сторони повинні знати відкритий і закритий ключі, або має бути дві пари ключів. Крім того, структурні особливості симетричних алгоритмів значно спрощують їхню зміну, ніж асиметричні.

З іншого боку, асиметричні алгоритми значно повільніші. Однак вони підвищують безпеку даних, усуваючи можливість перехоплення зловмисником закритого ключа. Попри це, він залишається вразливим для атак "зловмисник посередині" [2].

1.2 Аналіз мобільних додатків аналогів для генерації та аналізу паролів

З огляду на актуальність теми безпеки паролів та конфіденційної інформації в сучасному світі, існує значна кількість додатків, які надають можливість генерації, аналізу та надійного зберігання паролів. Найпопулярнішими з них для платформи Android є SafeInCloud, Dashlane, Bitwarden, My Passwords Manager, Keeper, Kaspersky, PasswordSafe.

Менеджер паролів SafeInCloud (рис. 1.1) дає змогу зберігати логіни, паролі та іншу особисту інформацію в безпеці в зашифрованій базі даних. Можна синхронізувати дані з іншим телефоном, планшетом або ПК через акаунт у хмарі.

Усі можливості та переваги цього додатка перераховані нижче:

– простий у використанні;

– чорна тема;

– стійке шифрування (256-bit Advanced Encryption Standard);

– синхронізація з хмарою (Google Диск, Dropbox, OneDrive);

– вхід за відбитком пальця, обличчям, сітківкою;

– автозаповнення в додатках;

– автозаповнення в Google Chrome;

– інтеграція з браузером;

– аналіз стійкості паролів;

– генератор паролів;

– автоматичний імпорт даних.

Єдиним недоліком цього додатку є відсутність безкоштовної версії та необхідність покупки додатка за 14,49$ [3].

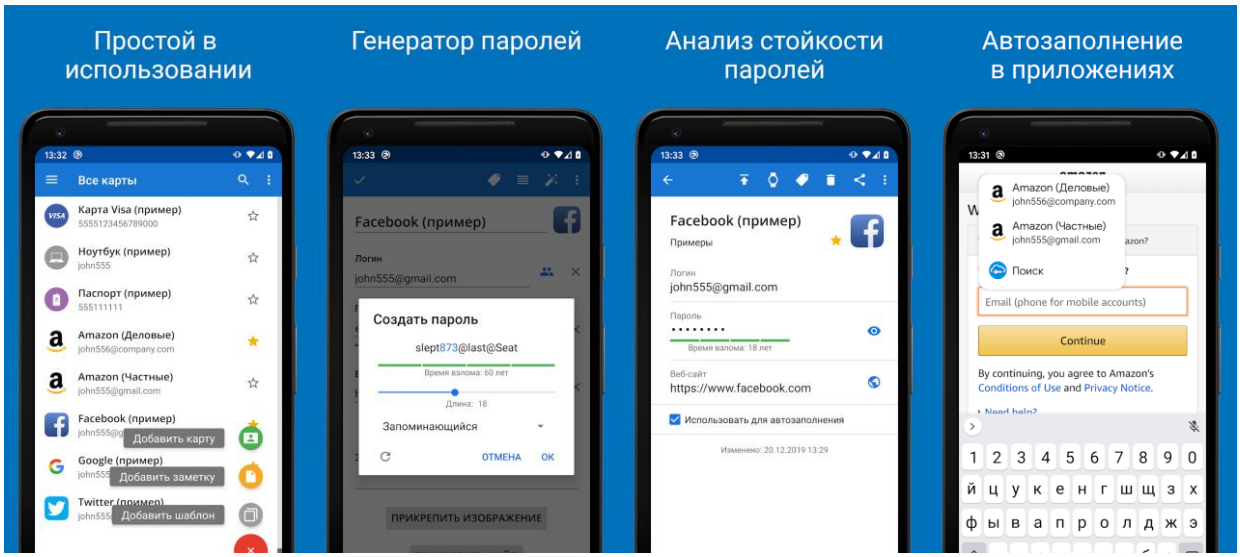

Рисунок 1.1 – Менеджер паролів SafeInCloud

Менеджер паролів Bitwarden (рис. 1.2) – простий і безпечний спосіб створювати та зберігати всі логіни та паролі й легко їх синхронізувати між усіма пристроями користувача.

Bitwarden зберігає всі логіни та паролі в зашифрованому сховищі, яке синхронізується між усіма пристроями користувача. До того, як дані покинуть пристрій, вони будуть зашифровані і тільки потім відправлені. Дані зашифровані за допомогою алгоритму AES-256 і PBKDF2 SHA-256.

Bitwarden – це програмне забезпечення з відкритим на 100% вихідним кодом. Вихідний код Bitwarden розміщено на GitHub, і кожен може вільно переглядати, перевіряти і робити внесок у код [4].

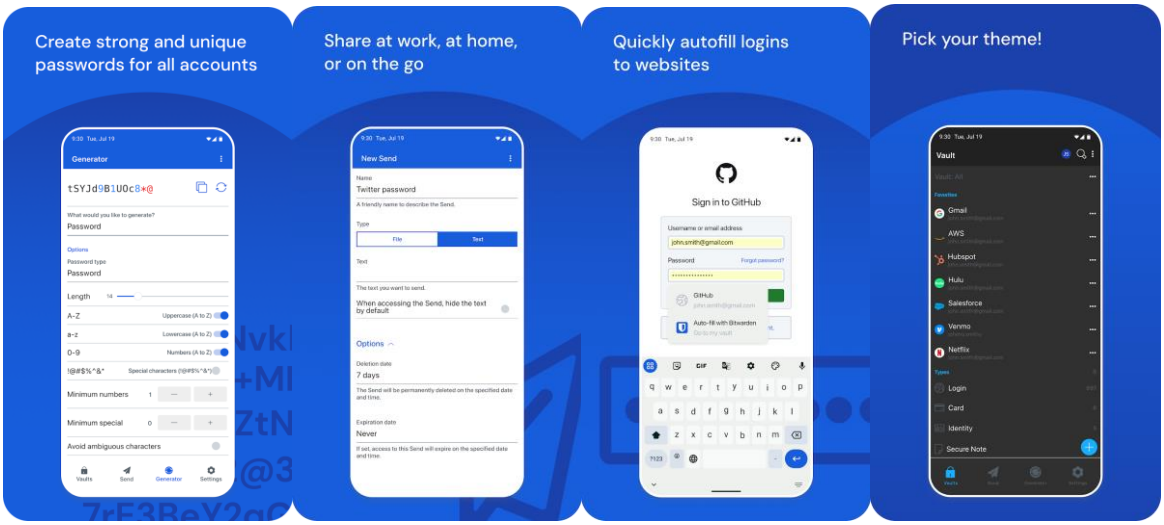

Рисунок 1.2 – Менеджер паролів Bitwarden

Dashlane (рис. 1.3) – це провідний менеджер паролів, який настільки ж простий у використанні, наскільки і безпечний. Додаток допомагає захищати паролі, платежі та особисту інформацію й отримувати доступ до цих даних у будь-якому місці, де вони потрібні, і все це з найкращою безпекою.

Збереження пароля відбувається тільки у сховищі паролів. Інтуїтивно зрозумілий додаток Dashlane дає змогу легко зробити це, тому не потрібно запам'ятовувати свої паролі. Зашифроване сховище – найбільш безпечне (і зручне) місце для зберігання паролів.

Сховище легко синхронізується на всіх пристроях користувача, тому користувач завжди може отримати доступ, згенерувати або безпечно поділитися

паролем. З Dashlane можна організувати своє сховище, щоб ефективно знаходити та фільтрувати паролі.

Такі функції, як автозаповнення, спрощують введення паролів і платіжної інформації в Інтернеті, а Dark Web Monitoring від Dashlane уважно стежить за глибинами Інтернету, щоб попередити користувача про що-небудь підозріле.

Dashlane використовує архітектуру з нульовим розголошенням, тому ніхто, навіть Dashlane, не може отримати доступ до інформації користувача. Код додатка Dashlane є загальнодоступним для Android і iOS, щоб кожен міг перевірити код і зрозуміти, як створювався Dashlane.

Повний захист: на відміну від деяких інших менеджерів паролів, Dashlane шифрує всі особисті дані, а не тільки паролі, за допомогою найнадійнішого доступного шифрування.

Недоліком додатка є те, що він не повністю безкоштовний і для доступу до деяких функцій додатка потрібно заплатити від 3,49$ [5].

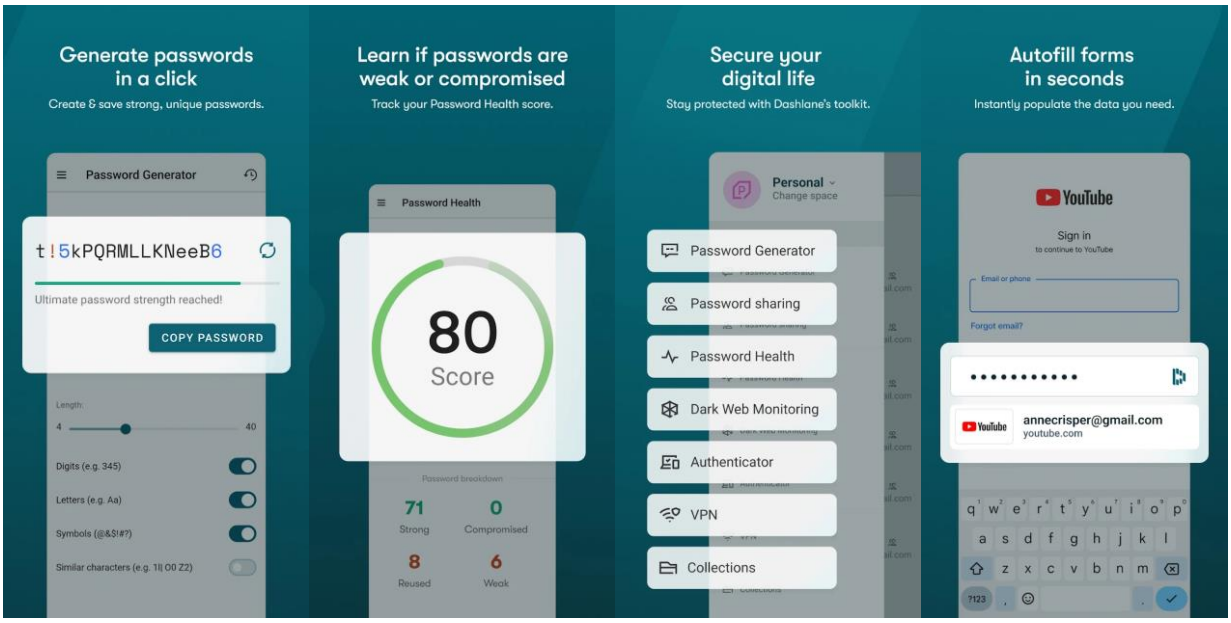

Рисунок 1.3 – Менеджер паролів Dashlane

Менеджер паролів My Password Manager (рис. 1.4) допомагає зберігати логіни, паролі та іншу особисту інформацію в надійній зашифрованій базі даних. Єдине, що потрібно зробити, це запам'ятати майстер-пароль, який використовується як ключ шифрування.

Додаток на 100 % безпечний, оскільки у нього немає доступу до Інтернету.

Безкоштовні функції додатка:

– надійне шифрування даних із використанням AES-256 біт;
– резервне копіювання та відновлення даних;
– автоматичне резервне копіювання в локальне сховище;
– немає доступу до Інтернету;
– вбудований генератор паролів;
– автоматичний вихід під час вимкнення екрана;
– підтримка декількох вікон;
– необмежена кількість записів.

Платні функції:

– архівні записи;
– біометрична аутентифікація (відбиток пальця тощо);
– автоматичне очищення буфера обміну;
– користувальницькі поля;
– експорт та імпорт CSV-файлів;
– експорт у pdf і друк;
– прикріплені зображення;
– історія паролів;
– самознищення;
– вибір теми;
– необмежена кількість ярликів;
– масові дії введення (призначення ярлика тощо);

Необов'язкова версія PRO доступна з однією покупкою в застосунку, яка розблокує всі додаткові функції.

Дані завжди шифруються з використанням 256-бітного розширеного стандарту шифрування (AES), що був прийнятий урядом США та використовується в усьому світі.

Якщо потрібен новий надійний пароль, можна просто створити його за допомогою вбудованого генератора паролів.

Якщо у користувача кілька пристроїв, можна легко обмінюватися паролями з усіма його пристроями. Треба створити резервну копію паролів на одному пристрої та перенести її на інший пристрій, де її можна буде відновити за допомогою того ж майстер-пароля.

Недоліком цього додатку є те, щоб отримати доступ до всіх його функцій треба заплатити 8,99$ [6].

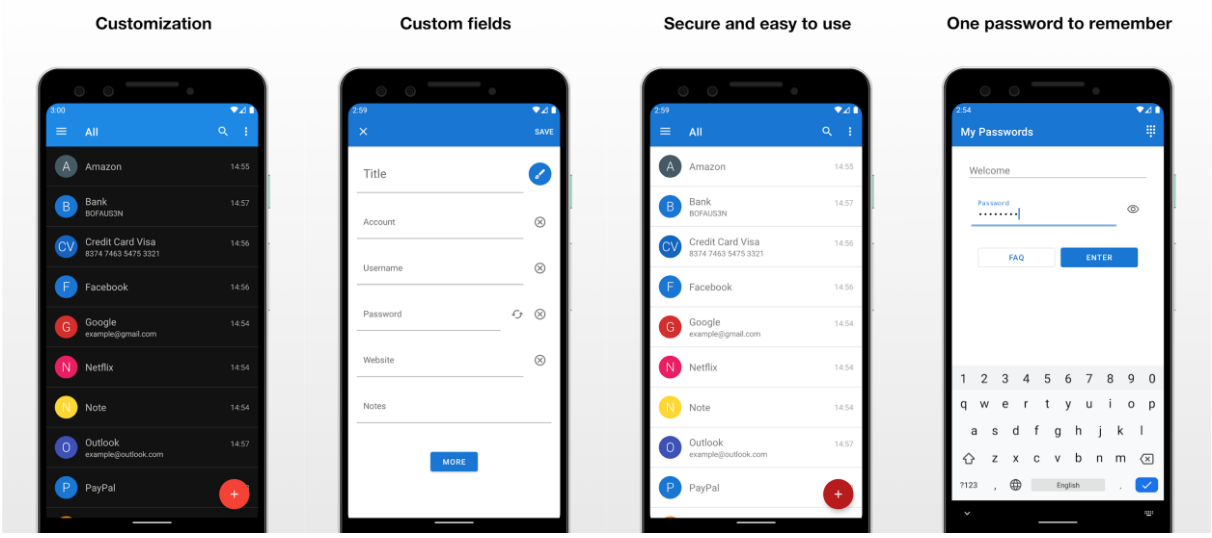

Рисунок 1.4 – Менеджер паролів My Password Manager

Менеджер паролів Keeper (рис. 1.5) максимально підвищує безпеку паролів і захищає персональні дані. Keeper – визнаний лідер у галузі кібербезпеки, що захищає паролі мільйонів людей і тисяч компаній по всьому світу.

За допомогою додатку Keeper можна автоматично генерувати надійні паролі, зберігати їх у безпечному цифровому сховищі, отримувати до них доступ з будь-якого пристрою, ділитися паролями й автоматично вводити їх на

всіх сайтах і в додатках. Потужне шифрування Keeper захищає паролі та конфіденційну інформацію від витоків даних, програм-вимагачів та інших кібератак.

Менеджер паролів Keeper дає змогу безпечно зберігати необмежену кількість паролів, конфіденційні файли, платіжні картки та багато іншого в зашифрованому цифровому сховищі. Доступ до сховища паролів можна отримати на необмеженій кількості мобільних пристроїв, планшетів і комп'ютерів. Безпеку можна підвищити увімкнувши захист за відбитками пальців з метою миттєвого та безпечного доступу.

Keeper дає змогу забезпечити безпеку паролів, відстежуючи DarkNet на предмет зламаних облікових записів і паролів за допомогою BreachWatch. Якщо користувач став жертвою витоку наданих даних, йому негайно прийде повідомлення, щоб він міг швидко вжити заходів для захисту своїх онлайн-акаунтів.

Keeper є одним з найбезпечніших у світі менеджер паролів:

- запатентована архітектура безпеки Keeper з нульовим розголошенням гарантує, що сховище Keeper і всі дані в ньому повністю зашифровані та доступні тільки користувачу додатка;
- сумісність із безліччю методів двофакторної аутентифікації, включно з Google Authenticator, Microsoft Authenticator, Duo, RSA, YubiKey та іншими;
- використовує 256-бітове шифрування AES, техніку еліптичної криптографії та метод PBKDF2;
- єдиний у галузі менеджер паролів, сертифікований за стандартами SOC-2 та ISO/IEC 27001;
- сумісний з усіма браузерами.

Недоліком Keeper є те що для доступу до деяких функцій додатка треба заплатити від 9,99$ [7].

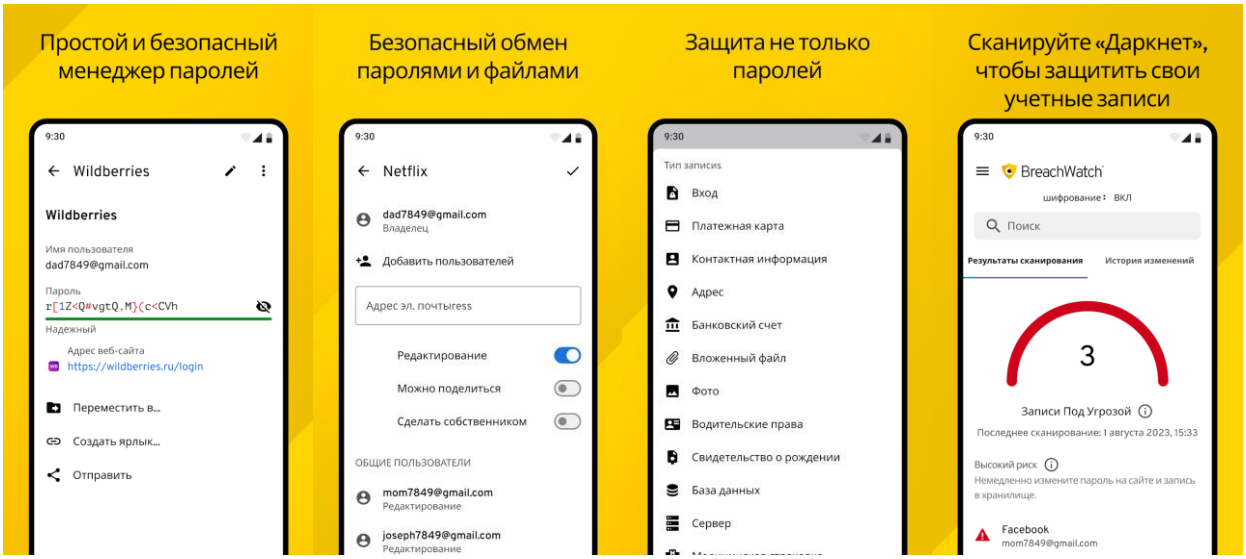

Рисунок 1.5 – Менеджер паролів Keeper

Менеджер паролів Kaspersky (рис. 1.6) забезпечує безпечне зберігання паролів, адрес, даних банківських карт, особистих нотаток і зображень конфіденційних документів. Паспорт, водійські права та інша конфіденційна інформація зберігаються в надійному місці та синхронізовані на всіх пристроях користувача – для більш швидкого доступу до вебсайтів, додатків і важливої інформації.

Основні функції:

– безпечне зберігання паролів і даних;
– єдиний майстер-пароль для доступу до сховища;
– синхронізація на всіх пристроях;
– автозаповнення паролів для економії часу;
– надійний генератор паролів;
– зручний пошук і систематизація документів;
– безпечне зберігання документів і фото;
– безпечне зберігання даних банківських карт.

Недоліком додатка є те що функціонал безкоштовної версії дає змогу зберігати не більше 5 активних записів. Платна версія коштує від 4,99$ [8].

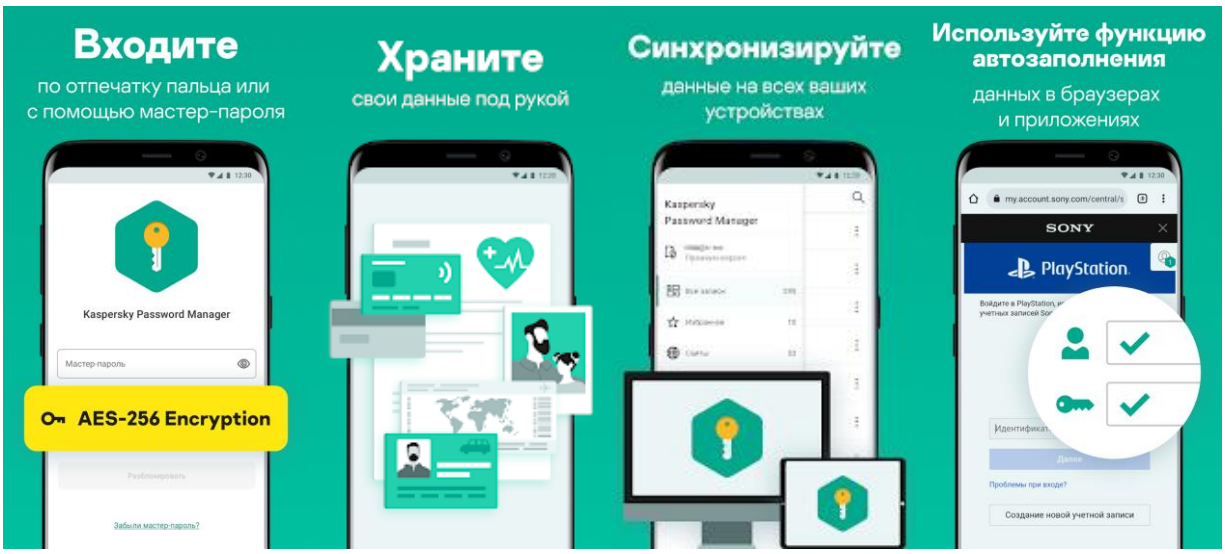

Рисунок 1.6 – Менеджер паролів Kaspersky

Менеджер паролів PasswordSafe (рис. 1.7) є надійною програмою для зберігання паролів яка кодує базу даних з паролями і керує нею завдяки кодуванню Advanced Encryption (AES) 256 bit. Користувачу необхідно запам'ятати лише один майстер-пароль.

PasswordSafe може згенерувати пароль і відправити його в оперативну пам'ять.

Безкоштовні функції:

– надійне збереження паролів і даних авторизації;
– доступність до БД одним майстер-паролем;
– генератор паролів;
– збереження і відновлення БД;
– можливість налаштування інтерфейсу;
– автоматичне збереження даних;
– CSV – імпорт/експорт;
– індикатор сили пароля.

Платні функції:

– прикріплення зображень;
– можливість реєстрації відбитком пальця;

– можливість імпорту/експорту в/з Excel;
– експорту у pdf;
– автоматичне блокування програми після закінчення часу та/або блокування монітора;
– великий вибір дизайну;
– режим самознищення.

Користувачу додатка потрібно запам’ятати лише один пароль і він отримає доступ до всіх інших. Інтуїтивний дизайн допомагає легко керувати даними. Використання категорій, щоб організувати записи, дає змогу спростити організацію та пошук певної інформації. Безпеку гарантує надійний 256-бітний розширений стандарт шифрування.

Немає причин боятися витоку, атаки серверів або чогось подібного, оскільки в застосунку немає можливості підключення до Інтернету. Проте можна створити резервну копію даних і легко їх відновити.

До недоліку додатка можна віднести необхідність платити 15,99$ для доступу до всіх його функцій [9].

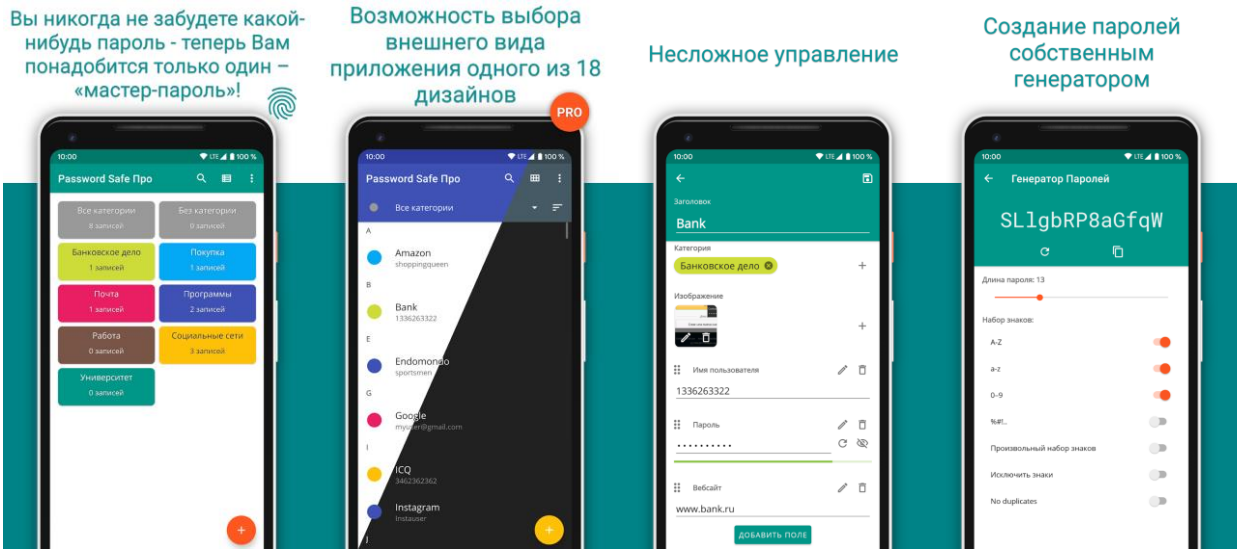

Рисунок 1.7 – Менеджер паролів PasswordSafe

2 ОБГРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ

Для реалізації теми "Розробка алгоритмів для оптимізації генерації та аналізу паролів і ключів з використанням методів та засобів ШІ" було вибрано створення Android-додатка. Цей вибір обумовлений необхідністю забезпечення безпечної та ефективної взаємодії з паролями на мобільних пристроях. Операційна система Android надає широкі можливості для впровадження функціоналу ШІ, таких як шифрування даних, управління доступом, автентифікація користувача та інші. Даний вибір сприяє підвищенню надійності та забезпеченню конфіденційності відповідно до сучасних стандартів інформаційної безпеки, роблячи Android-додаток оптимальним рішенням для використання функціоналу ШІ в контексті генерації та аналізу паролів і ключів.

2.1 Опис операційної системи Android

Існує декілька найбільш розповсюджених операційних систем для смартфонів, таких як Android, iOS, Windows Phone, KaiOS. Для розробки мобільного додатку було обрано операційну систему Android з наступних причин:

- Android – операційна система з відкритим вихідним кодом, що забезпечує гнучкість та можливість налаштувань;
- велика поширеність Android, станом на листопад 2023 року відсоток користувачів смартфонів з ОС Android становить близько 70% серед всіх ОС (рис. 2.1) [10];
- Android пропонує простий доступ до розробки, роблячи процес створення додатків доступним для широкого кола розробників
- ОС Android є абсолютно безкоштовною для розробки;
- велика кількість доступних інструментів та ресурсів для розробників на платформі Android сприяє швидкому старту та ефективній розробці додатків, що полегшує процес створення та оптимізації програмного забезпечення для смартфонів.

Рисунок 2.1 – Відсоток пристроїв на різних ОС станом на листопад 2023 року

Android – це операційна система з відкритим вихідним кодом, створена для мобільних пристроїв на основі модифікованого ядра Linux. Цю ОС розробив консорціум Open Handset Alliance, що складається з великих технологічних компаній за організаційної ролі Google. Вихідний код ОС представлений як частина проекту Android Open Source Project з ліцензією Apache. Випущений на ринок у 2007 році Android незабаром став найпопулярнішою операційною системою в історії, завдяки своїй відкритій моделі розробки і зручному інтерфейсу. Станом на листопад 2023 року останньою версією Android є Android 14.

Проєкт Android з'явився у 2003 році з метою розробки інтелектуальних мобільних пристроїв. Починався він з розробки ОС для цифрових фотокамер, але незабаром акцент змістився на мобільні телефони через їхню велику поширеність на ринку. У 2005 році проєкт придбав Google і в якості основи для цієї ОС було обрано ядро Linux внаслідок його гнучкості та можливості оновлення.

З метою розробки платформи з відкритим вихідним кодом для мобільних пристроїв у 2007 році Google сформувала Open Handset Alliance з кількома виробниками обладнання та операторами бездротового зв'язку. У той час кожен виробник випускав мобільні телефони на базі власної платформи, з обмеженими можливостями для сторонніх додатків. Альянс заявив, що відкрита платформа забезпечить тісну співпрацю між виробниками і розробниками, щоб прискорити виробництво недорогих інноваційних продуктів і додатків.

Платформа Android була презентована в 2007 році і вийшла на ринок наступного року. Спочатку їй заважав обмежений набір функцій і невелика база користувачів порівняно з конкурентами Symbian і Windows. Однак можливість оновлення стала найбільшою перевагою цієї ОС, оскільки кожне оновлення давало нові функції та поліпшену продуктивність. Через "солодкість, яку вони приносять у наше життя", перші версії були названі на честь десертів, за алфавітним порядком, як-от Cupcake, Jellybean та KitKat. Однак з 2019 року нові версії ОС отримують номери, що починаються з Android 10. Ліцензія з відкритим вихідним кодом також допомогла збільшити популярність цієї ОС серед виробників мобільних пристроїв, оскільки вони можуть тепер модифікувати ОС під свої вимоги, не впливаючи при цьому на розробку додатків.

Але найголовніша особливість у тому, що Android – це більше, ніж просто операційна система. Він багато в чому зрівняв мобільні пристрої з персональними комп'ютерами, дозволивши розробникам писати застосунки незалежно від апаратної платформи пристрою. Це призвело до створення глобальної платформи для застосунків і зміцнило позиції Android, як передової мобільної платформи, і в 2011 році він став найбільш продаваною операційною системою для смартфонів і для планшетів у 2013 році. Сьогодні на Android працює безліч електронних пристроїв, включно зі смарт-камерами, годинниками, медіаплеєрами та багато іншого [11].

Android надає користувачам широкі можливості для управління своїм смартфоном. Android забезпечує доступ до безлічі додатків, ігор і сервісів, які можна завантажити з інтернету через Google Play. Android також має інтеграцію з різними сервісами Google, такими як Gmail, Google Maps, Google Drive та іншими. Операційна система дає змогу персоналізувати домашній екран, налаштовувати сповіщення, керувати налаштуваннями приватності та багато іншого.

Основні функції операційної системи Android включають:

– керування додатками: дає змогу встановити, видалити та оновити додатки з магазину додатків.

- робота з мультимедіа: відтворення відео та аудіо, перегляд зображень і запис мультимедіафайлів.
- зв'язок: доступ до інтернету, надсилання та отримання повідомлень (SMS, MMS), виконання голосових і відеодзвінків.
- навігація: використання GPS та інших локаційних сервісів для визначення місця розташування.
- налаштування та персоналізація: можливість налаштування різних параметрів операційної системи та інтерфейсу, встановлення та зміна звуків і тем.
- безпека: захист даних і пристрою від шкідливого програмного забезпечення та несанкціонованого доступу, включно з використанням паролів, шифрування та блокування.

Операційна система Android завжди оновлюється і розвивається, щоб запропонувати більш досконалі функції і поліпшені можливості з кожною новою версією.

Основні особливості та переваги Android:

- відкритий вихідний код: Android заснований на ядрі Linux і є повністю відкритим вихідним кодом. Це означає, що розробники можуть вільно вивчати і змінювати код операційної системи, що сприяє появі безлічі версій Android, які можна налаштувати, і різноманітності застосунків.
- безліч застосунків: Google Play Store, офіційний магазин застосунків для Android, пропонує величезне розмаїття програмних застосунків, які можна завантажити і встановити на смартфон. Користувачі можуть вибирати з мільйонів застосунків, включно з іграми, соціальними мережами, мультимедіа-плеєрами та багатьма іншими.
- інтеграція з сервісами Google: Android має глибоку інтеграцію з сервісами Google, такими як Gmail, Google Maps, Google Drive та іншими. Це дає змогу користувачам зручно використовувати різні сервіси та синхронізувати дані між пристроями.

– велика спільнота розробників і служба підтримки: Завдяки своїй популярності, Android має величезну спільноту розробників, які створюють нові додатки та працюють над покращенням операційної системи. Крім того, Google надає підтримку та різні інструменти для розробників, що сприяє появі якісних додатків.

Загалом, Android пропонує широкі можливості для користувачів смартфонів, забезпечуючи гнучкість, налаштованість і доступ до багатого набору додатків. Завдяки цим перевагам, Android залишається однією з найбільш відомих і потрібних операційних систем для мобільних пристроїв [12].

2.2 Середовище розробки Android Studio

Для створення додатка на ОС Android існують різні середовища розробки, такі як Eclipse, Android Studio, IntelliJ IDEA і інші. Для виконання даної магістерської роботи було обрано Android Studio від компанії Google. Вибір зумовлений наявністю широкого спектру технологій та засобів розробки, зручністю інтерфейсу та високою ефективністю програми.

Android Studio – це інтегроване середовище розробки (IDE), яке призначене для створення додатків для операційної системи Android. Це середовище розробки розроблено компанією Google і є офіційним інструментом для розробки Android-додатків.

Android Studio є відкритою платформою, що означає, що вона доступна для всіх розробників безкоштовно. Це дає можливість розробникам з усього світу використовувати та робити свій внесок у розвиток середовища розробки. Велика та активна спільнота розробників також сприяє популярності Android Studio.

Інтеграція з Android SDK дозволяє розробникам легко створювати, налагоджувати та тестувати додатки на різних пристроях Android. Ця інтеграція також відкриває доступ до різноманітних інструментів і бібліотек, що сприяють прискоренню розробки.

Серед потужних інструментів розробки, які пропонує Android Studio, варто відзначити автозаповнення коду, інтегрований редактор макетів і систему збирання проєктів. Ці інструменти спрощують та полегшують розробку високоякісних додатків.

Розробники можуть вибирати мову програмування за своїми уподобаннями, оскільки Android Studio підтримує кілька мов, включаючи Java, Kotlin і C++. Зокрема, мова Kotlin отримала значну популярність серед розробників Android-додатків.

Активна підтримка та регулярні оновлення від Google гарантують надійність та актуальність середовища розробки. Це робить Android Studio найкращим вибором для розробників, які цінують доступ до нових функцій, виправлень помилок і поліпшень у своєму інструменті.

Загалом, Android Studio користується великою популярністю серед розробників завдяки своїй функціональності, безоплатності, відкритості, підтримці кількох мов програмування та зручним інструментам розробки, які допомагають створювати високоякісні додатки для Android [13].

Єдиним недоліком Android Studio є те, що, незважаючи на вбудований Android-емулятор у самому середовищі розробки, можуть виникнути труднощі при тестуванні новорозробленого додатка. Це зумовлено необхідністю в потужному апаратному забезпеченні ПК для ефективного запуску емулятора, що може ускладнювати процес тестування на менш продуктивних комп'ютерах.

2.3 Вибір мови програмування

У інтегрованому середовищі Android Studio є можливість вибрати між мовою програмування Java та Kotlin.

Java є об'єктно-орієнтованою мовою програмування, яку почали використовувати в 1995 році. Java було розроблено компанією Sun Microsystems, яку пізніше придбала Oracle. Якщо потрібно запустити Java програму в браузері,

потрібно використовувати Java-аплети, які вбудовані як плагін, що не рекомендується. Таким чином, Java в основному використовується для автономних додатків або back-end розробки. Java був створений Джеймсом Гослінгом, і його основною реалізацією був OpenJDK. Коли справа доходить до розробки додатків для Android, більшість розробників обирають Java, оскільки сам Android написаний на Java.

Kotlin є сучасною мовою програмування, розробленою програмістами з JetBrains. Уперше вона з'явилася у 2011 році, а офіційний реліз відбувся у 2016 році. Kotlin являє собою мову з відкритим вихідним кодом. Вона також є мовою програмування зі статичною типізацією, такою як Java, C ++, яка заснована на JVM. Kotlin можна скомпілювати як JavaScript, Android і Native, а також для створення і запуску коду на iOS.

Kotlin повністю сумісний з Java. Переключитися з Java на Kotlin дуже просто, потрібно просто встановити плагін. Під час виступу на Google I/O було оголошено, що Kotlin стане офіційно підтримуваною мовою для розробки додатків на Android.

Обидва варіанти популярні на ринку. Ось основні відмінності між Java і Kotlin:

- у Kotlin є підтримка smart cast, яка ідентифікує незмінні типи та виконує неявне перетворення компілятором, тоді як у Java нам потрібно ідентифікувати та виконувати перетворення;
- Kotlin підтримує виведення типу. Це означає, що не потрібно явно вказувати тип даних змінної. У Java потрібно вказувати тип явно;
- у Kotlin немає перевірки винятків. Це є недоліком, оскільки призводить до коду, схильного до помилок. У Java є підтримка перевірки винятків, за допомогою якої можна виконувати обробку помилок;
- час компіляції Java на 15-20% швидший, ніж час компіляції Kotlin. Однак, з точки зору інкрементальної компіляції збірки, Kotlin також буде займати той самий час компіляції, що і Java;

– у Kotlin не можна призначати null-значення змінним або значенням, що повертаються. Якщо дуже потрібно, можна оголосити змінну зі спеціальним синтаксисом, тоді як у Java можна призначати null значення. Однак під час спроби отримати доступ до об'єктів, що вказують на null-значення, викликається виняток;
– Kotlin і Java є взаємозамінними. Можна викликати код Kotlin у Java і код Java у Kotlin. Таким чином, у проєкті можна розміщувати поруч класи Java і Kotlin і компілювати без будь-яких проблем. Після компіляції не вдасться знайти, який клас написаний на Java, а який на Kotlin.

Детальне порівняння мов програмування Java і Kotlin представлено у табл. 1 [14].

Таблиця 1 – Таблиця порівняння Java та Kotlin

Підстава для порівняння	Java	Kotlin
Null Safe	У Java NullPointerExceptions може сильно ускладнити життя розробника. Він дозволяє присвоювати значення null будь-якій змінній. Замість отримання доступу до посилання на об'єкт з null-значенням – піднімається виняток null покажчика, який потрібно виправляти.	У Kotlin за замовчуванням усі типи змінних є non-nullable (тобто ми не можемо присвоїти null-значення жодному типу змінних/об'єктів). При спробі присвоїти або повернути null значення Kotlin не дозволить продовження компіляції коду. Якщо потрібно, щоб змінна мала null-значення, її можна оголосити таким чином: val num:Int?=null

Продовження таблиці 1

Підстава для порівняння	Java	Kotlin
Функції розширення	У Java, якщо потрібно розширити функціональність наявного класу, потрібно створити новий клас, який успадковує батьківський клас. У Java немає функцій розширення.	Kotlin надає розробникам можливість розширити функціональність наявного класу. Ми можемо створити розширену функцію за допомогою додавання префікса з назвою класу до нової функції.
Підтримка підпрограм	У Java під час ініціалізації тривалої I/O мережі або інтенсивних операцій CPU відповідний потік буде заблоковано. В Android за замовчуванням один потік. Java надає можливість фонового створення і запуску декількох потоків, однак управління ними є складним завданням.	У Kotlin можна створити кілька потоків для запуску тривалих інтенсивних операцій. До того ж у ньому є підтримка підпрограм, яка дає змогу відкласти виконання в певний момент без блокування потоків під час виконання тривалих інтенсивних операцій.
Неперевірені винятки	Java має підтримку перевірки винятків, яка дає змогу розробникам оголошувати та фіксувати винятки, що зрештою призводить до ясного коду з хорошою обробкою помилок.	У Kotlin немає перевірки винятків. Розробникам не потрібно оголошувати або фіксувати винятки, у чому є свої переваги і недоліки.

Продовження таблиці 1

Підстава для порівняння	Java	Kotlin
Класи даних	Припустимо, потрібен клас, який має містити тільки дані і нічого більше. У Java для цього потрібно визначити конструктори, змінні для зберігання даних, методи get() і set(), функції hashcode(), toString() і equals().	У Kotlin, якщо потрібні класи для вмісту даних, можна оголосити клас через ключове слово "data" у визначенні клас. Потім компілятор зробить усю решту роботи на кшталт створення конструкторів, методів get() і set() для різних полів.
Розумні перетворення – Smart Cast	У Java потрібно перевіряти тип змінних і робити перетворення відповідно до операції.	У Kotlin розумне перетворення, або smart cast, займається перевіркою за допомогою ключового слова "is-checks", яке перевіряє наявність незмінюваних значень і здійснює неявне перетворення.
Виведення типу	У Java потрібно вказувати тип кожної змінної під час її оголошення.	У Kotlin не потрібно вказувати тип кожної змінної, він буде призначатися на підставі операції, з якою ця змінна буде використовуватися. Якщо потрібно самостійно вказати потрібний тип, це також можна зробити.

Продовження таблиці 1

Основа порівняння	Java	Kotlin
Функціональне програмування	До версії Java 8 у Java не було підтримки функціонального програмування. Під час розроблення Android додатків підтримується тільки підмножина функціоналу Java 8.	Kotlin являє собою мікс процедурної та функціональної мови програмування, який складається з таких корисних методів, як лямбда, перевантаження оператора, функції вищого порядку тощо.

З урахуванням всіх відмінностей між Java та Kotlin, для створення додатка було обрано мову програмування Java. Це рішення прийнято через значний обсяг статей з прикладами додатків для Android, які були розроблені саме цією мовою програмування.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

3.1 UML діаграма проекту

UML – уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною процесу розробки програмного забезпечення [15].

На рис. 3.1 наведено діаграму класів проекту з використанням UML, на якій відображені взаємозв'язки між класами.

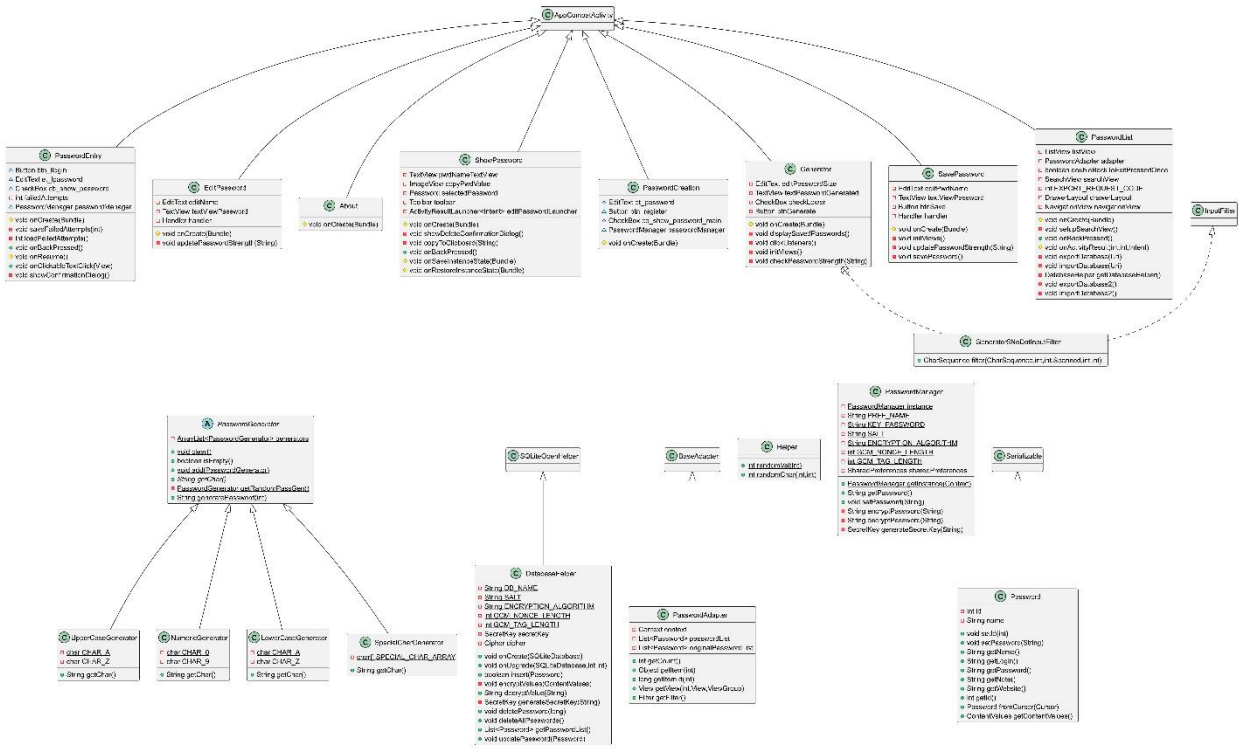

Рисунок 3.1 – UML діаграма класів

UML був створений для визначення, візуалізації, проєктування й документування програмних систем. UML не є мовою програмування, але на підставі UML-моделей можлива генерація коду.

3.2 Проектування зовнішнього вигляду та елементів керування

Створення дизайну додатка є одним з ключових етапів у процесі розробки, оскільки він визначає спосіб, яким користувач взаємодіє з продуктом. Дизайн впливає на перше враження від додатка, забезпечуючи зручність використання, естетичний вигляд і відповідність функціональним потребам. Вірно розроблений дизайн сприяє позитивному досвіду користувача, підвищує конкурентоспроможність продукту і сприяє його успішному прийняттю.

Для мобільного додатка було створено 9 макетів:

– екран створення майстер-пароля;
– екран введення майстер-пароля;
– головний екран зі списком паролів;
– екран для додавання та збереження інформації про пароль;
– екран з інформацією про пароль;
– екран для редагування інформації про пароль;
– бокова панель, яка служить в якості меню;
– екран для генерації паролів;
– екран з інформацією про додаток.

Для додатку було створено дві теми: чорна та біла (рис. 3.2). Ці теми були створені з метою надання користувачам вибору інтерфейсу відповідно до їхніх власних вподобань та потреб. Макети додатку представлені в чорній темі.

Макет екрану створення майстер-пароля зображено на рис. 3.3. На цьому екрані користувачу потрібно створити майстер-пароль, завдяки якому він зможе заходити у додаток, після створення йому потрібно натиснути кнопку "Продовжити" для продовження.

Після натискання кнопки "Продовжити" користувач переходить на екран введення майстер-пароля (рис. 3.4). На цьому екрані користувачу потрібно ввести створений майстер-пароль та натиснути кнопку "Увійти" для входу у додаток.

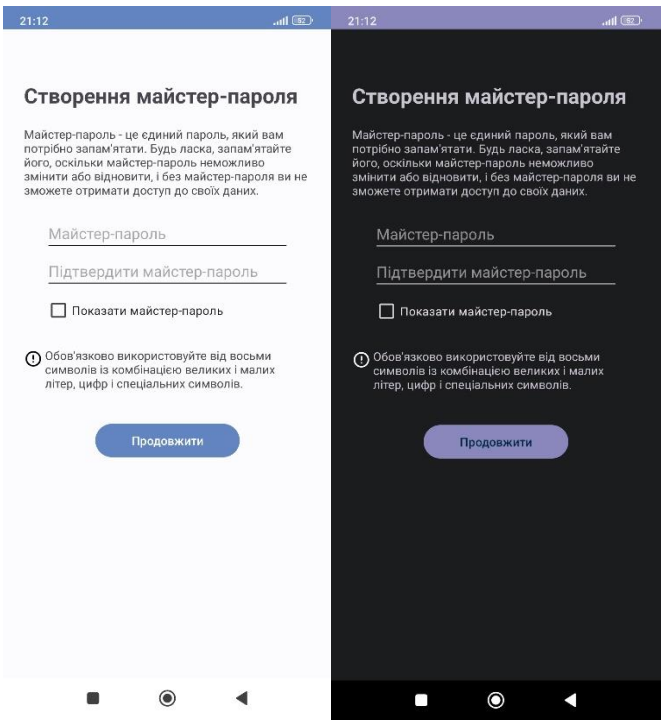

Рисунок 3.2 – Чорна та біла теми додатка

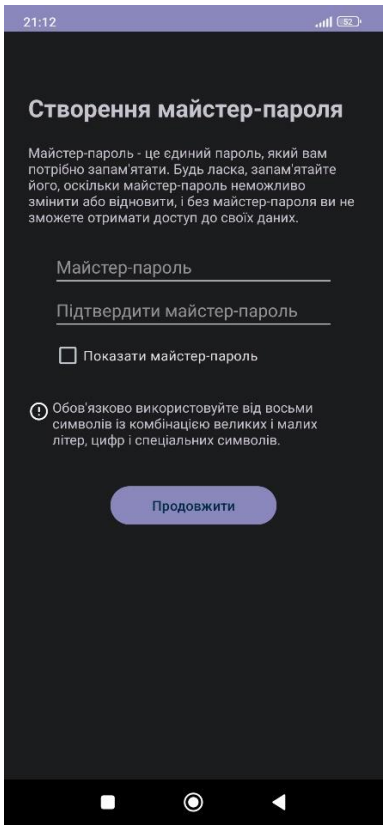

Рисунок 3.3 – Екран створення майстер-пароля

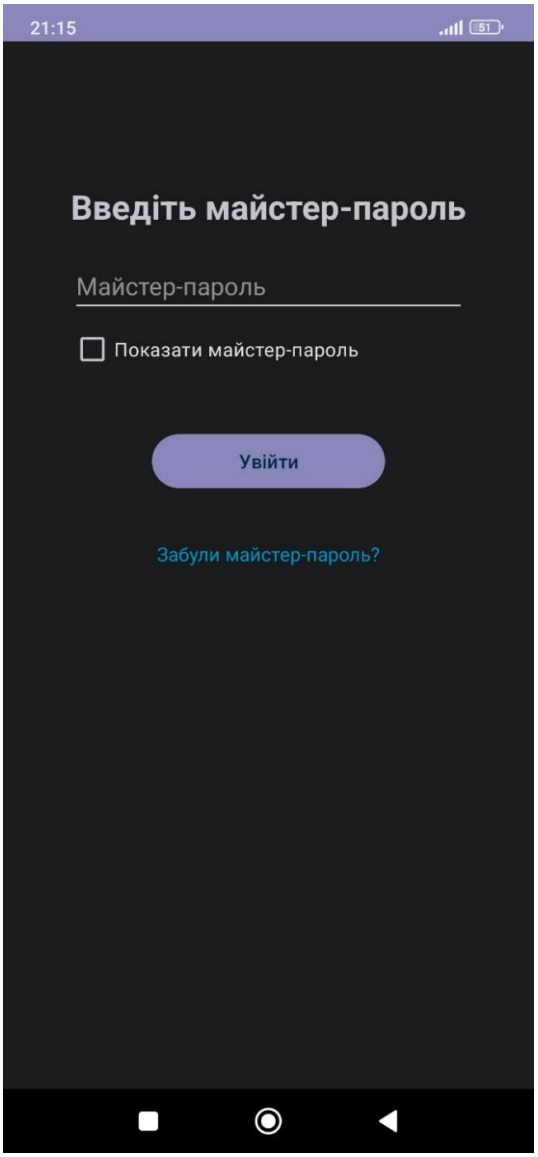

Рисунок 3.4 – Екран введення майстер-пароля

Після введення майстер-пароля користувач переходить на головний екран додатка в якому знаходиться список його паролів (рис. 3.5).

При натисканні на іконку лупи на цьому екрані користувач може знайти потрібний йому пароль.

При натисканні на кнопку з іконкою "+" користувач перейде на екран для додавання та збереження інформації про пароль, в якому він може ввести інформацію про пароль та зберегти її натиснувши на кнопку "Зберегти" (рис. 3.6).

При натискані на пароль зі списку відкривається екран з інформацією про цей пароль (рис. 3.7).

Знаходячись на цьому екрані користувач може скопіювати потрібну інформацію про пароль натиснувши на одну з іконок копіювання, видалити пароль натиснувши на іконку кошика для сміття або натиснути на іконку олівця та перейти на екран із редагуванням інформації про пароль (рис. 3.8)

Знаходячись на екрані із редагуванням інформації про пароль користувач може змінити інформацію про пароль і натиснути кнопку "Зберегти зміни" щоб зберегти внесені зміни.

Рисунок 3.5 – Головний екран зі списком паролів

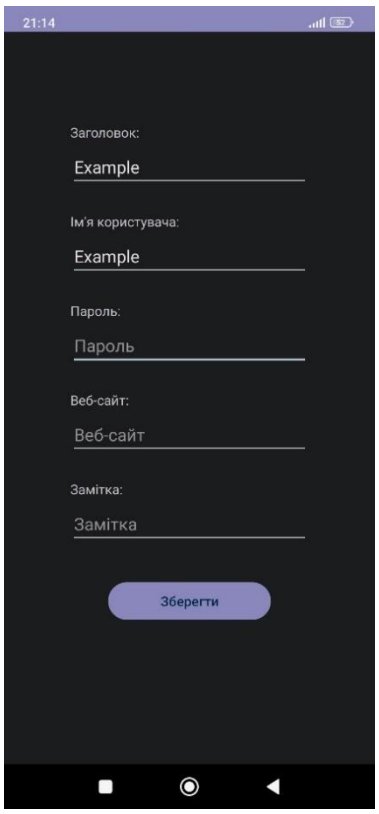

Рисунок 3.6 – Екран для додавання та збереження інформації про пароль

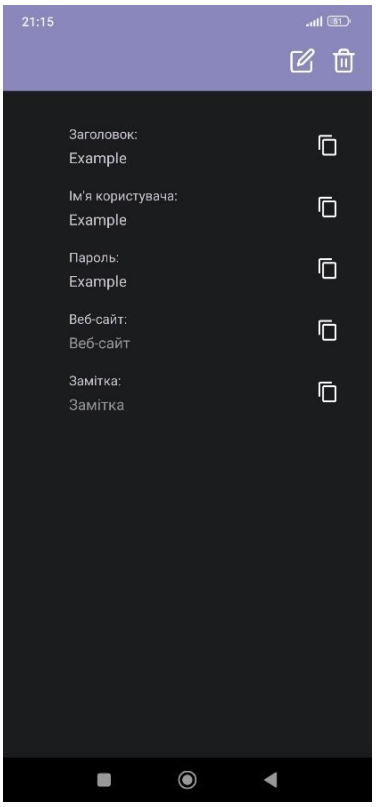

Рисунок 3.7 – Екран з інформацією про пароль

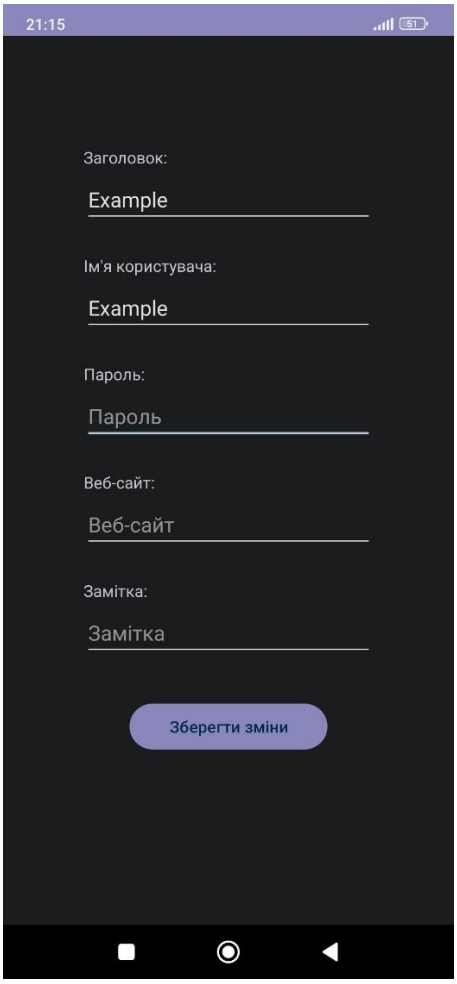

Рисунок 3.8 – Екран для редагування інформації про пароль

При натискані на іконку меню на головному екрані додатку відкривається бокова панель, яка служить в якості меню, в якому користувач може вибрати один з чотирьох пунктів (рис. 3.9).

При натисканні на пункт "Згенерувати пароль" користувач переходить на екран для генерації паролів (рис. 3.10).

На екрані для генерації паролів користувач може вибрати вміст та розмір пароля та згенерувати його натиснувши на кнопку "Згенерувати". Після генерації користувачу стане доступна кнопка "Копіювати" для копіювання пароля та кнопка "Зберегти" яка відкриває екран для додавання та збереження інформації про пароль в який автоматично внесено згенерований пароль.

При натисканні на пункт "Експорт паролів" користувач переходить до теки завантаження на своєму смартфоні, де він може експортувати свої паролі у вигляді файлу з зашифрованими паролями в базі даних.

При натисканні на пункт "Імпорт паролів" користувач переходить до теки завантаження на своєму смартфоні, де він може обрати раніше створений файл з зашифрованими паролями в базі даних та імпортувати дані цієї БД в додаток.

При натисканні на пункт "Про додаток" користувачу переходить до екрану з інформацією про додаток (рис. 3.11).

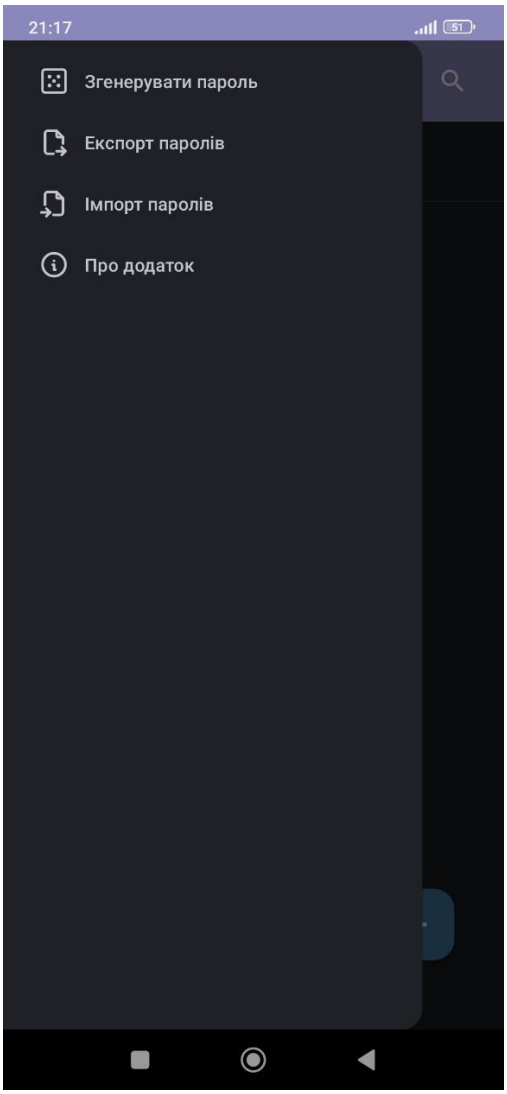

Рисунок 3.9 – Меню додатка

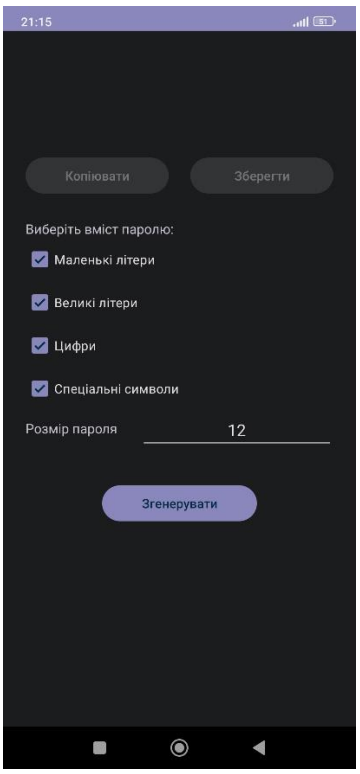

Рисунок 3.10 – Екран для генерації паролів

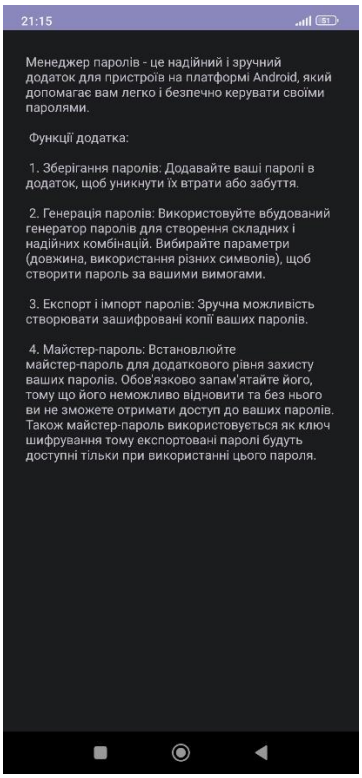

Рисунок 3.11 – Екран з інформацією про додаток

3.3 Реалізація користувальницького інтерфейсу та алгоритмів для оптимізації генерації та аналізу паролів і ключів

Спочатку був створений новий проект у Android Studio. Цей проект розрахований на операційну систему Android версії 6.0 і вище. Було здійснено зміни у файлі colors.xml, який відповідає за визначення кольорів у Android-додатку.

3.2.1 Розробка екрана створення майстер-пароля

Створено новий клас PasswordCreation та файл макету для екрану activity_password_creation.xml.

Для екрана була створена кнопка з текстом "Продовжити". При натисканні на цю кнопку, користувач переходить на екран для введення майстер-пароля. Але щоб перейти на цей екран введений користувачем пароль має пройти низку перевірок.

Перевірка на порожні поля:

```
if (password.equals("") || confirm_password.equals("")) {
    Toast.makeText(getApplicationContext(), "Потрібно заповнити поля",
Toast.LENGTH_SHORT).show();
```

Якщо одне з полів (пароль або його підтвердження) порожнє, виводиться повідомлення про те, що всі поля повинні бути заповнені.

Перевірка на довжину паролю:

```
} else if (password.length() < 8) {
                Toast.makeText(getApplicationContext(), "Майстер-пароль
повинен містити мінімум 8 символів", Toast.LENGTH_SHORT).show();
```

Якщо довжина паролю менше 8 символів, виводиться повідомлення про те, що майстер-пароль повинен бути не менше 8 символів.

Перевірка на наявність маленьких літер:

```
} else if (!password.matches(".*[a-za-яґєії].*")) {
    Toast.makeText(getApplicationContext(), "Потрібна хоча б одна мала
літера", Toast.LENGTH_SHORT).show();
```

Якщо в паролі відсутні малі літери, виводиться повідомлення про те, що майстер-пароль повинен містити хоча б одну малу літеру.

Перевірка на наявність великих літер:

```
} else if (!password.matches(".*[A-ZA-ЯҐЄІЇ].*")) {
    Toast.makeText(getApplicationContext(), "Потрібна хоча б одна
велика літера", Toast.LENGTH_SHORT).show();
```

Якщо в паролі відсутні великі літери, виводиться повідомлення про те, що майстер-пароль повинен містити хоча б одну велику літеру.

Перевірка на наявність цифр:

```
} else if (!password.matches(".*[0-9].*")) {
    Toast.makeText(getApplicationContext(), "Потрібна хоча б одна
цифра", Toast.LENGTH_SHORT).show();
```

Якщо в паролі відсутні цифри, виводиться повідомлення про те, що майстер-пароль повинен містити хоча б одну цифру.

Перевірка на наявність спеціальних символів:

```
} else if (!password.matches(".*[!@#$%^&*()_+\\-
=\\[\\]{};':\"\\\\|,.<>/?].*")) {
    Toast.makeText(getApplicationContext(), "Потрібен хоча б один
спеціальний символ", Toast.LENGTH_SHORT).show();
```

Якщо в паролі відсутні спеціальні символи, виводиться повідомлення про те, що майстер-пароль повинен містити хоча б один спеціальний символ.

Перевірка на співпадіння паролів:

```
} else if (!password.equals(confirm_password)) {
    Toast.makeText(getApplicationContext(), "Майстер-паролі не співпадають", Toast.LENGTH_SHORT).show();
```

Якщо паролі не співпадають, виводиться повідомлення про те, що майстер-паролі не співпадають.

Ці перевірки допомагають користувачеві створити міцний майстер-пароль, враховуючи різні аспекти безпеки.

Робота цих перевірок відображена на рис. 3.12

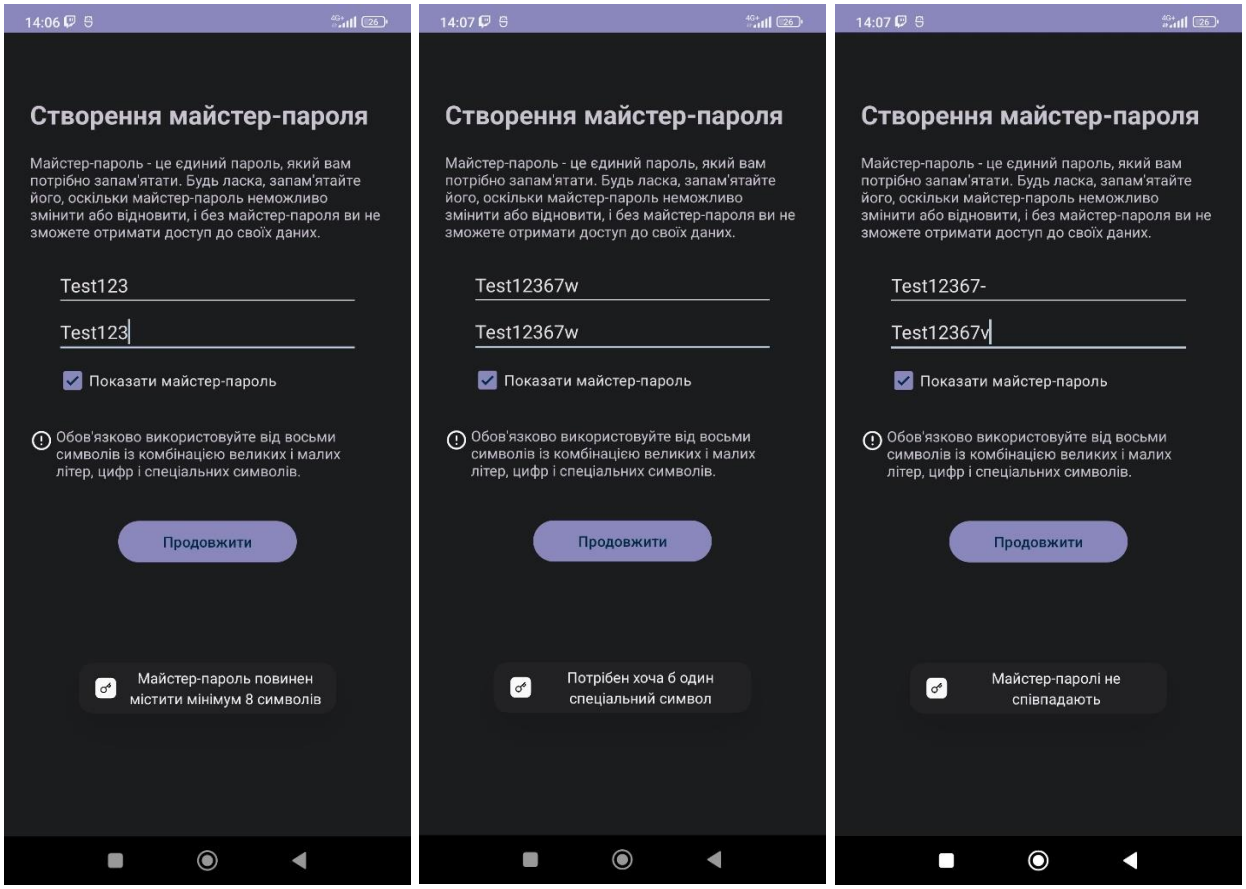

Рисунок 3.12 – Перевірка пароля

3.2.2 Розробка екрана введення майстер-пароля

Створено новий клас PasswordEntry та файл макету для екрану activity_password_entry.xml.

Для екрана була створена кнопка з текстом "Увійти". При натисканні на цю кнопку, користувач переходить на головний екран зі списком паролів. Щоб перейти на цей екран введений користувачем пароль має співпадати зі створеним раніше майстер-паролем. Якщо користувач зробить більше 10 невдалих спроб введення майстер-пароля то всі дані додатка включно з майстер-паролем будуть видалені. Ось фрагмент коду перевірки кількості невдалих спроб:

```
if (failedAttempts >= 3 && failedAttempts < 10) {
    Toast.makeText(getApplicationContext(), "Невірний майстер-пароль.
Залишилося " + (10 - failedAttempts) + " спроб",
Toast.LENGTH_SHORT).show();
}
if (failedAttempts == 10) {
    DatabaseHelper databaseHelper = new
DatabaseHelper(PasswordEntry.this, password);
    databaseHelper.deleteAllPasswords();
    passwordManager.setPassword("");
    Toast.makeText(getApplicationContext(), "Усі дані видалено",
Toast.LENGTH_SHORT).show();
    Intent mainIntent = new Intent(PasswordEntry.this,
PasswordCreation.class);
    startActivity(mainIntent);
    finish();
} else if (failedAttempts < 3) {
    Toast.makeText(getApplicationContext(), "Невірний майстер-пароль",
Toast.LENGTH_SHORT).show();
}
```

На рис. 3.13 показана робота перевірки кількості невдалих спроб.

На екрані введення майстер-пароля також присутній напис "Забули майстер-пароль?" При натисканні на цей напис з'явиться діалогове вікно (рис. 3.14), де користувачу буде запропоновано видалити майстер-пароль та всі дані додатка. Якщо він обере "ВИДАЛИТИ" у цьому діалоговому вікні, з'явиться ще одне діалогове вікно (рис. 3.15) для підтвердження видалення всіх даних. При натисканні на "ТАК" у цьому діалоговому вікні всі дані додатка будуть безповоротно видалені.

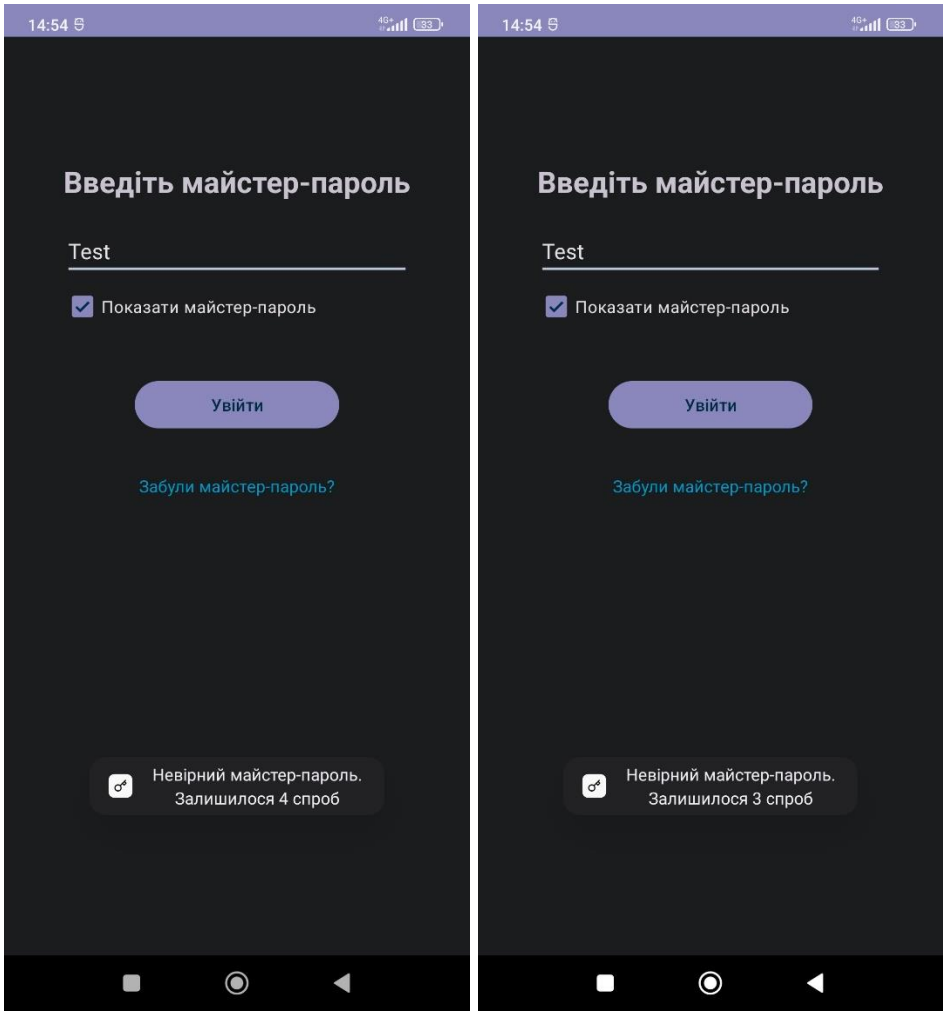

Рисунок 3.13 – Перевірка кількості невдалих спроб

Рисунок 3.14 – Вікно видалення всіх даних

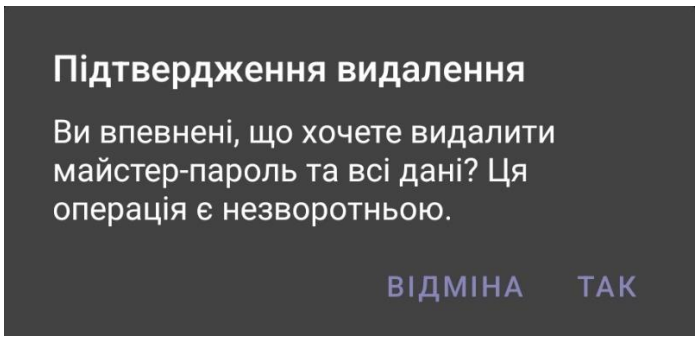

Рисунок 3.15 – Діалогове вікно підтвердження видалення всіх даних

3.2.3 Створення бази даних для збереження паролів

Для безпечного збереження паролів потрібно створити базу даних для зберігання паролів у зашифрованому вигляді.

Створення БД розпочинається зі створення конструктора DatabaseHelper який:

– ініціалізує змінні, такі як ім'я БД (DB_NAME), сіль для генерації ключа (SALT), алгоритм шифрування (ENCRYPTION_ALGORITHM), довжину ініціалізаційного вектора (GCM_NONCE_LENGTH), та довжину тега GCM (GCM_TAG_LENGTH);
– генерує секретний ключ (secretKey) з використанням паролю користувача та солі за допомогою алгоритму PBKDF2WithHmacSHA256.

Код конструктора:

```
public DatabaseHelper(@Nullable Context context, String password) {
    super(context, DB_NAME, null, 1);
    try {
        secretKey = generateSecretKey(password);
        cipher = Cipher.getInstance(ENCRYPTION_ALGORITHM);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Алгоритм PBKDF2 з HMAC-SHA256 (Hash-based Message Authentication Code with Secure Hash Algorithm 256-bit) є криптографічно стійким методом отримання ключів з паролів. PBKDF2 є функцією виведення ключа (KDF), спеціально призначеною для отримання ключа з великого об'єму даних, такого як пароль. Однією з особливостей PBKDF2 є можливість вказати кількість ітерацій, що робить його більш витратним для атак з використанням перебору. Вхідний пароль та сіль обробляються разом, і цей процес повторюється задану кількість разів для ускладнення обчислень.

HMAC є криптографічним протоколом, який використовує хеш-функцію для створення коду аутентифікації. SHA256 є однією з версій безпечної хеш-функції SHA-2, повертаючи 256-бітний хеш. У використанні разом, PBKDF2 використовує HMAC-SHA256 для внутрішніх ітерацій. Вхідний пароль та сіль об'єднуються, і HMAC-SHA256 використовується для отримання інтермедіатного результату. Цей процес повторюється задану кількість разів, і кожен проміжний результат додається до кінцевого ключа.

Алгоритм PBKDF2WithHmacSHA256 є важливим інструментом для посилення безпеки паролів, оскільки він ускладнює процес отримання ключа з пароля. Це робить атаки з використанням перебору менш ефективними, допомагаючи зберігати паролі у вигляді хешів та утруднюючи їх відновлення навіть у випадку проникнення атак.

Після створення конструктора йде створення метода onCreate(SQLiteDatabase sqLiteDatabase) який створює таблицю "PASSWORDS" у базі даних з колонками для id, імені (name), паролю (password), логіну (login), веб-сайту (website) та примітки (note).

Код методу:

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    sqLiteDatabase.execSQL(
            "CREATE TABLE PASSWORDS" +
                    "(id INTEGER PRIMARY KEY, name TEXT, password TEXT,
login TEXT, website TEXT ,note TEXT);");
}
```

Створення методу onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) який викликається, якщо версія БД змінилася. Видаляє поточну таблицю "PASSWORDS" та викликає метод onCreate для створення нової версії таблиці.

Код методу:

```
@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
    sqLiteDatabase.execSQL("DROP TABLE IF EXISTS PASSWORDS");
    onCreate(sqLiteDatabase);
}
```

Створення методу insert(Password password) який:

– вставляє новий запис у таблицю "PASSWORDS" у базі даних;

– зашифровує значення полів пароля перед вставкою в базу даних.

Код методу:

```
public boolean insert(Password password) {
    SQLiteDatabase db = getWritableDatabase();
    ContentValues values = password.getContentValues();
    encryptValues(values);
    return db.insert("PASSWORDS", null, values) != -1;
}
```

Створення методу encryptValues(ContentValues values) який:

– зашифровує значення полів за допомогою AES/GCM/NoPadding;

– для кожного ключа у ContentValues генерує випадковий ініціалізаційний вектор (iv), зашифровує значення та замінює його на Base64-кодоване зашифроване значення.

Код методу:

```
private void encryptValues(ContentValues values) {
    try {
        for (String key : values.keySet()) {
            String fieldValue = values.getAsString(key);
            byte[] iv = new byte[GCM_NONCE_LENGTH];
```

```
            cipher.init(Cipher.ENCRYPT_MODE, secretKey, new
GCMParameterSpec(GCM_TAG_LENGTH, iv));
            byte[] encryptedValue =
cipher.doFinal(fieldValue.getBytes());
            values.put(key, Base64.encodeToString(encryptedValue,
Base64.DEFAULT));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

AES/GCM/NoPadding – це конкретний режим роботи для алгоритму шифрування AES з інтегрованою аутентифікацією галузевого коду (GCM) та без доповнення (NoPadding). Цей режим забезпечує як конфіденційність, так і аутентифікацію даних. AES є симетричним алгоритмом блочного шифрування, призначеним для захисту конфіденційності даних та вважається операційно ефективним і безпечним.

GCM є режимом блочного шифрування, який використовується для забезпечення конфіденційності і аутентифікації даних. Він використовує аутентифікаційний тег (галузевий код) для перевірки цілісності даних і захисту від атак, таких як зміна або вставка даних.

Режим NoPadding вказує, що довжина даних, які шифруються, повинна бути кратною розміру блоку алгоритму (у випадку AES - 128 біт або 16 байт). Якщо довжина даних не є кратною розміру блоку, додаються додаткові байти доповнення, але в режимі NoPadding це не використовується.

Отже, AES/GCM/NoPadding використовує алгоритм AES для шифрування даних у режимі GCM, надаючи конфіденційність та аутентифікацію, і не використовує додаткового доповнення. Цей режим є ефективним та безпечним для захисту інформації.

Створення методу decryptValue(String encryptedValue) який розшифровує передане зашифроване значення, використовуючи той самий алгоритм та ключ, що і при шифруванні.

Код методу:

```
public String decryptValue(String encryptedValue) {
    try {
        byte[] iv = new byte[GCM_NONCE_LENGTH];
        cipher.init(Cipher.DECRYPT_MODE, secretKey, new
GCMParameterSpec(GCM_TAG_LENGTH, iv));
        byte[] decodedValue = Base64.decode(encryptedValue,
Base64.DEFAULT);
        byte[] decryptedValue = cipher.doFinal(decodedValue);
        return new String(decryptedValue);
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

Створення методу generateSecretKey(String password) який генерує секретний ключ з використанням алгоритму PBKDF2WithHmacSHA256.

Код методу:

```
private SecretKey generateSecretKey(String password) {
    try {
        SecretKeyFactory factory = SecretKeyFactory.get-
Instance("PBKDF2WithHmacSHA256");
        KeySpec spec = new PBEKeySpec(password.toCharArray(), SALT.get-
Bytes(), 10000, 256);
        SecretKey tmp = factory.generateSecret(spec);
        return new SecretKeySpec(tmp.getEncoded(), "AES");
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

Створення методів deletePassword(long id) та deleteAllPasswords() які видаляють відповідно один конкретний пароль та всі паролі з БД.

Код методу deletePassword(long id):

```
public void deletePassword(long id) {
    SQLiteDatabase db = getWritableDatabase();
    String[] whereArgs = {String.valueOf(id)};
    db.delete("PASSWORDS", "id=?", whereArgs);
```

```
    db.close();
}
```

Код методу deleteAllPasswords():

```
public void deleteAllPasswords() {
    SQLiteDatabase db = getWritableDatabase();
    db.execSQL("DELETE FROM PASSWORDS");
    db.close();
}
```

Створення методу getPasswordList() який отримує список паролів з БД та розшифровує їх значення, використовуючи метод decryptValue.

Код методу:

```
public List<Password> getPasswordList() {
    List<Password> passwordList = new ArrayList<>();
    SQLiteDatabase db = getReadableDatabase();
    Cursor cursor = db.rawQuery("SELECT * FROM PASSWORDS", null);
    cursor.moveToFirst();
    while (!cursor.isAfterLast()) {
        Password pwd = Password.fromCursor(cursor);
        String decryptedName = decryptValue(pwd.getName());
        String decryptedPassword = decryptValue(pwd.getPassword());
        String decryptedLogin = decryptValue(pwd.getLogin());
        String decryptedWebsite = decryptValue(pwd.getWebsite());
        String decryptedNote = decryptValue(pwd.getNote());
        Password decryptedPwd = new Password(decryptedName, decrypt-
edLogin, decryptedPassword, decryptedWebsite, decryptedNote);
        decryptedPwd.setId(pwd.getId());
        passwordList.add(decryptedPwd);
        cursor.moveToNext();
    }
    cursor.close();
    return passwordList;
}
```

Створення методу updatePassword(Password password) який оновлює існуючий запис у таблиці "PASSWORDS" у базі даних, зашифровуючи значення перед оновленням.

Код методу:

```
public void updatePassword(Password password) {
        SQLiteDatabase db = getWritableDatabase();
        ContentValues values = password.getContentValues();
        encryptValues(values);
        String[] whereArgs = {String.valueOf(password.getId())};
        db.update("PASSWORDS", values, "id=?", whereArgs);
}
```

3.2.4 Розробка головного екрана зі списком паролів

Створено новий клас PasswordList та файл макету для екрану activity_password_list.xml та файл для елементів списку list_item.xml.

На цьому екрані знаходиться список доданих користувачем паролів.

Зверху екрана присутня іконка лупи для пошуку необхідних паролів по заголовку пароля (рис. 3.16)

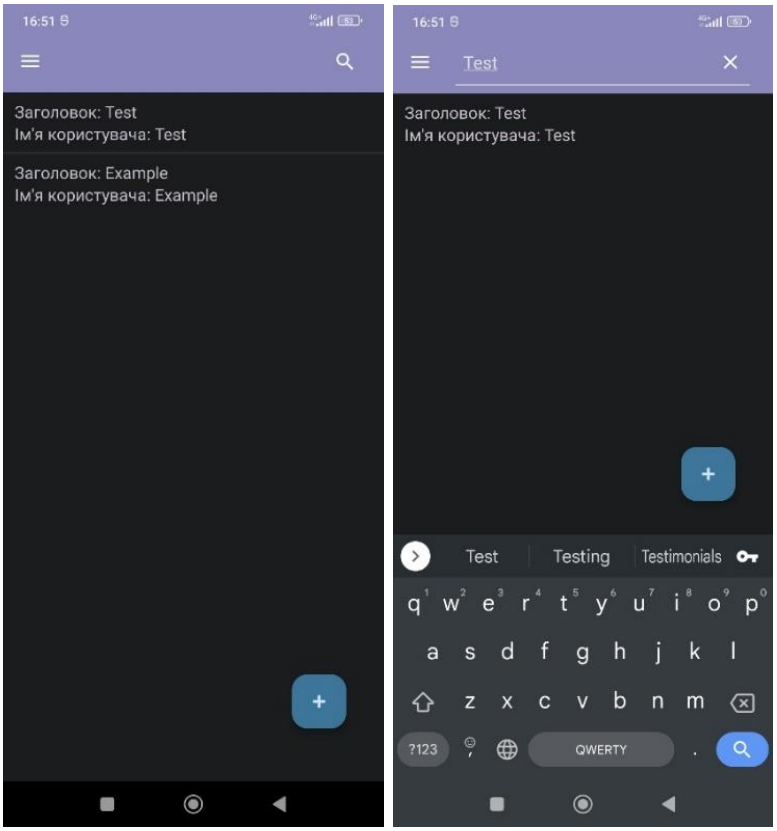

Рисунок 3.16 – Пошук пароля

Також на цьому екрані присутня кнопка з іконкою "+" при натисканні на яку користувач перейде на екран для додавання та збереження інформації про пароль.

3.2.5 Розробка екрана для додавання та збереження інформації про пароль і екрана для редагування інформації про пароль

Створено нові класи SavePassword та EditPassword і файли макету для екранів activity_save_password.xml та activity_edit_password.xml.

У екрані для додавання та збереження інформації про пароль користувач може внести інформацію про пароль та зберегти її натиснувши на кнопку "Зберегти". Макет цього екрана зображено на рис. 3.6.

У екрані для редагування інформації про пароль користувач може редагувати інформацію про пароль. Макет цього екрана зображено на рис. 3.8.

У ці класи додано оцінювання надійності пароля за допомогою zxcvbn. (рис. 3.17)

zxcvbn – це оцінювач надійності паролів, створений на основі зломщиків паролів. Завдяки зіставленню шаблонів і консервативній оцінці він розпізнає і оцінює 30 тис. поширених паролів, поширені імена та прізвища, популярні англійські слова з Вікіпедії та американського телебачення і кіно, а також інші поширені шаблони, такі як дати, повтори (aaa), послідовності (abcd), клавіатурні шаблони (qwertyuiop) і l33t speak.

zxcvbn можна використовувати як алгоритмічну альтернативу політиці складання паролів – він безпечніший, гнучкіший і зручніший у використанні.

Безпечніший: політики складання паролів часто не спрацьовують в обох напрямках, дозволяючи слабкі паролі (P@ssword1) і забороняючи надійні паролі.

Гнучкіший: zxcvbn дає змогу використовувати безліч стилів паролів за умови, що він виявляє достатню складність – парольні фрази оцінюються високо, якщо вони містять достатньо рідкісних слів, клавіатурні комбінації

оцінюються на основі довжини, а великі літери додають складності, коли вони непередбачувані.

Зручніший у використанні: zxcvbn розроблен для створення простих інтерфейсів без правил, які дають миттєвий зворотній зв'язок. На додаток до оцінки складності, zxcvbn включає мінімальний, цілеспрямований словесний зворотній зв'язок, який може допомогти користувачам створювати паролі, які складно вгадати [16].

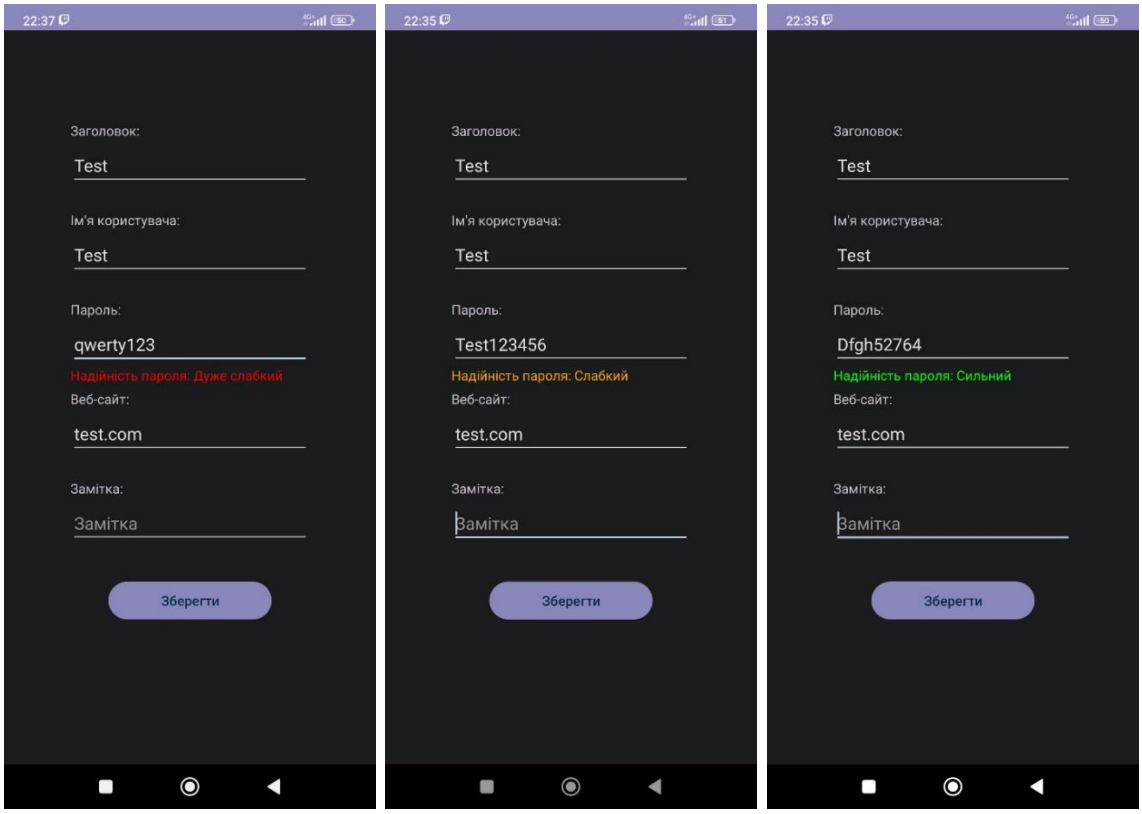

Рисунок 3.17 – Оцінювання надійності пароля за допомогою zxcvbn

3.2.6 Розробка екрана з інформацією про пароль

Створено новий клас ShowPassword та файл макету для екрану activity_show_password.xml.

У цьому екрані користувач може скопіювати інформацію про пароль, натиснувши на одну з іконок копіювання, натиснути на іконку олівця та перейти

на екран із редагуванням інформації про пароль або видалити пароль натиснувши на іконку кошику для сміття. Макет екрана з інформацією про пароль зображено на рис. 3.7.

При натискані на іконку для сміття з'являється діалогове вікно для підтвердження видалення пароля і якщо користувач натисне на "ТАК" пароль буде видалено (рис. 3.18).

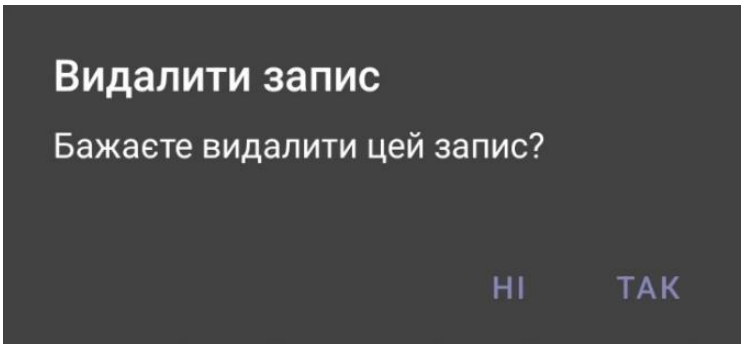

Рисунок 3.18 – Підтвердження видалення пароля

3.2.7 Розробка бокової панелі, яка служить в якості меню

Створено файл макету для меню navigation_menu.xml.

У цьому меню користувач може вибрати один з чотирьох пунктів. Макет меню зображено на рис. 3.9.

При натисканні на пункт "Згенерувати пароль" користувач перейде на екран для генерації паролів.

При натисканні на пункт "Експорт паролів" користувач експортує свої паролі

При натисканні на пункт "Імпорт паролів" імпортує паролі. Приклад експортування та імпортування зображено на рис. 3.19 – 3.23.

При натисканні на напис "Про додаток" користувачу переходить до екрану з інформацією про додаток.

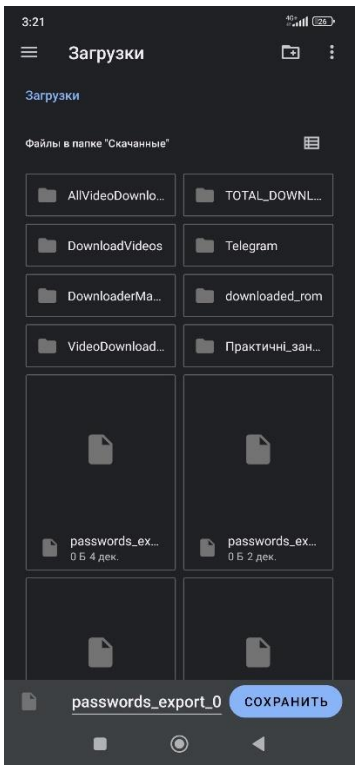

Рисунок 3.19 – Перехід до теки завантаження на смартфоні та експортування паролів у вигляді файлу з зашифрованими паролями в базі даних.

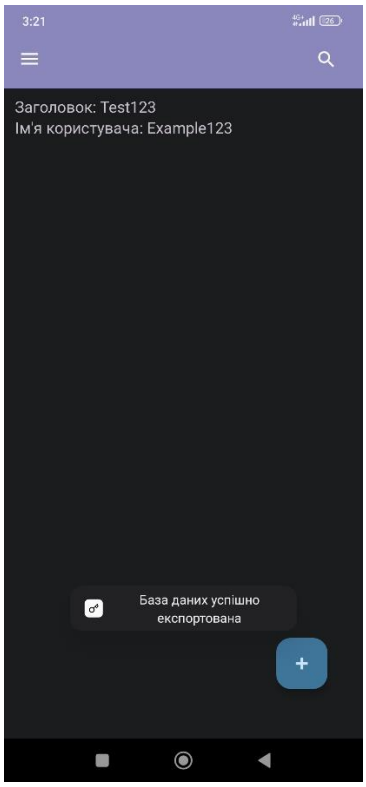

Рисунок 3.20 – Повідомлення про успішне експортування БД

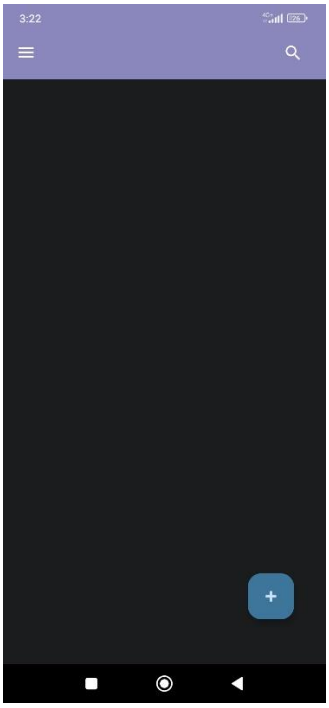

Рисунок 3.21 – Видалення пароля для перевірки імпортування БД

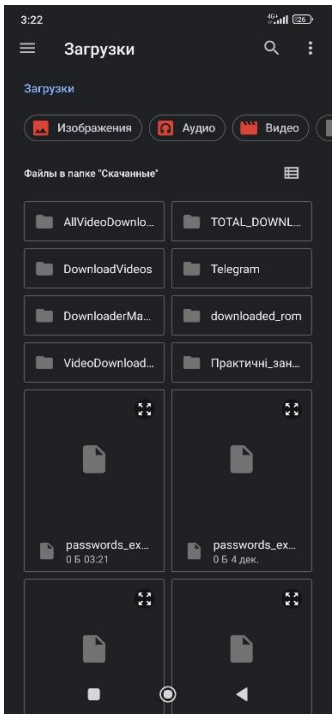

Рисунок 3.22 – Перехід до теки завантаження на смартфоні та обрання раніше створеного файлу з зашифрованими паролями в базі даних.

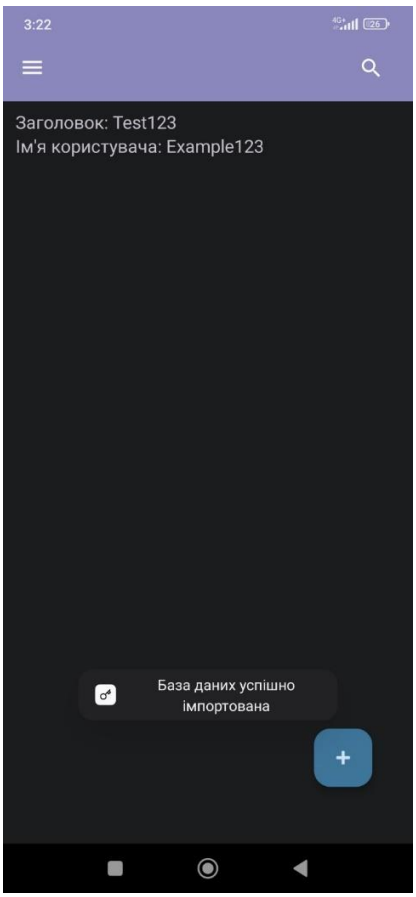

Рисунок 3.23 – Успішне імпортування БД

3.2.8 Розробка екрана для генерації паролів

Створено новий клас Generator та файл макета для екрану generator.xml.

У цьому екрані користувач може вибрати вміст та розмір пароля та згенерувати його натиснувши на кнопку "Згенерувати". Після генерації користувачу буде доступна кнопка "Копіювати" для копіювання пароля та кнопка "Зберегти" яка відкриває екран для додавання та збереження інформації про пароль в який автоматично внесено згенерований пароль. Також у цьому екрані присутнє оцінювання надійності пароля за допомогою zxcvbn. Макет екрана для генерації паролів зображено на рис. 3.10.

Для генерації паролів окрім класу Generator створено клас GeneratorHelper і абстрактний клас PasswordGenerator та підкласи PasswordGenerator:

UpperCaseGenerator, SpecialCharGenerator, LowerCaseGenerator, NumericGenerator.

Основний клас Generator відповідає за взаємодію з користувачем та обробку подій. Користувач може вибирати розмір пароля та типи символів, такі як маленькі літери, великі літери, спеціальні символи та цифри. Після натискання кнопки генерації викликається метод generatePassword класу PasswordGenerator, який вирішує, які типи символів використовувати.

Клас PasswordGenerator є абстрактним і використовується для комбінування різних генераторів символів. Він має статичний масив генераторів, де кожен генератор відповідає за певний тип символів. Метод generatePassword генерує пароль заданого розміру, використовуючи випадковий вибір генератора для кожного символу.

Генератори символів, такі як UpperCaseGenerator, SpecialCharGenerator, LowerCaseGenerator та NumericGenerator, реалізують метод getChar(), що генерує один символ свого типу. Наприклад, UpperCaseGenerator генерує випадковий символ від "A" до "Z", а SpecialCharGenerator – випадковий спеціальний символ.

Клас GeneratorHelper містить методи для отримання випадкових значень, використовуючи SecureRandom, що забезпечує безпечність генерації випадкових чисел.

Ця система дозволяє створювати паролі різної довжини з різними типами символів, забезпечуючи різнорівневу надійність паролів.

В результаті інтерфейс екрану генерації паролів надає користувачеві широкий вибір параметрів, а додаткові опції, такі як оцінювання надійності пароля та можливість збереження та копіювання згенерованих паролів, роблять процес керування та використання паролів максимально зручним. Такий підхід забезпечує високий рівень безпеки та відповідає сучасним вимогам до захисту конфіденційної інформації.

Генерація паролів зображена на рис. 3.24.

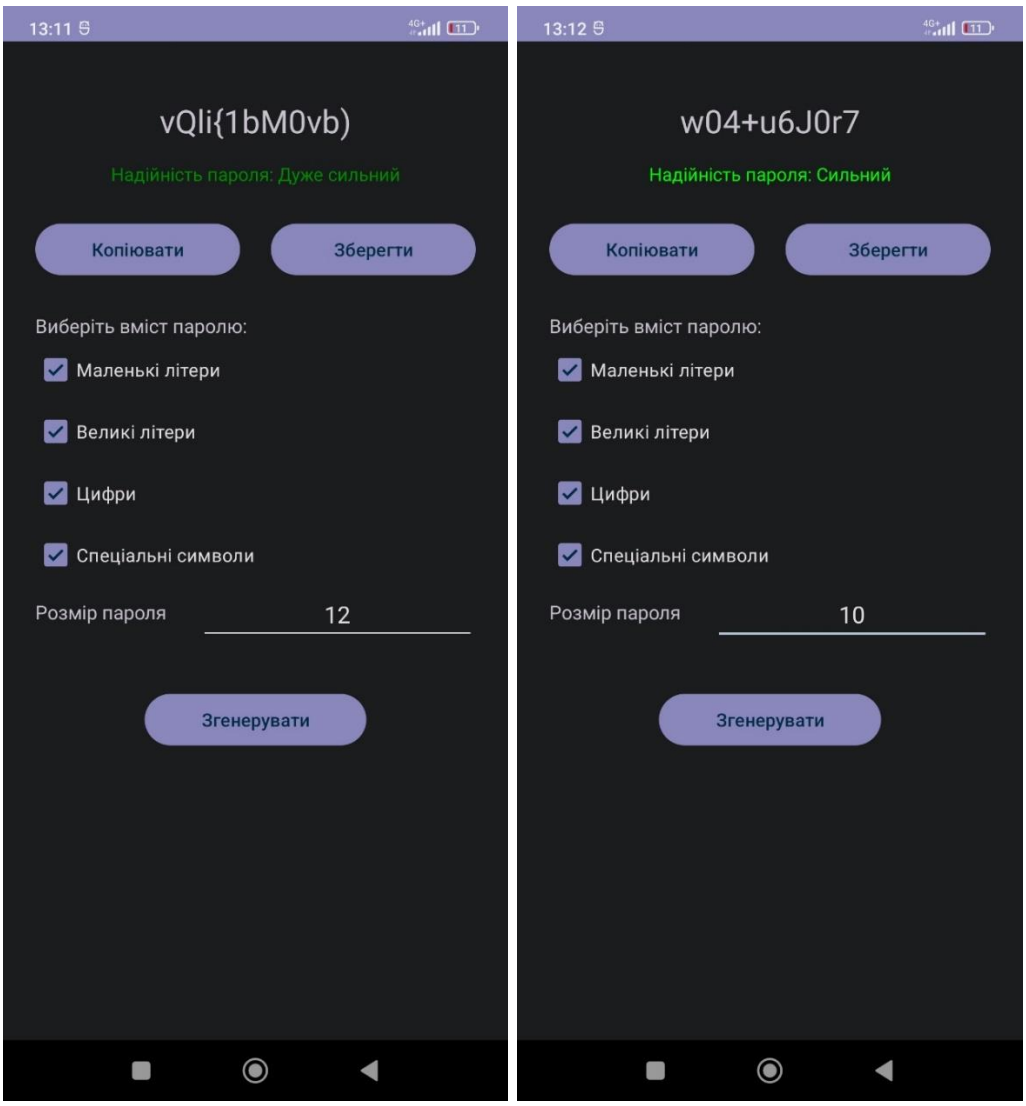

Рисунок 3.24 – Генерація паролів

4 ТЕСТУВАННЯ РОБОТИ ДОДАТКА

4.1 Інсталяція та системні вимоги

Додаток встановлюється на пристрій користувача за допомогою установки файлу "apk". Оскільки програма розроблена для операційної системи Android, необхідно, щоб користувач мав цю операційну систему на своєму пристрої. Для правильної роботи додатка потрібна версія операційної системи не нижче 6.0.

4.2 Функціонал додатка

Після встановлення додатка та запуску додатка потрібно створити майстер-пароль. Створення майстер-пароля зображено на рис. 4.1.

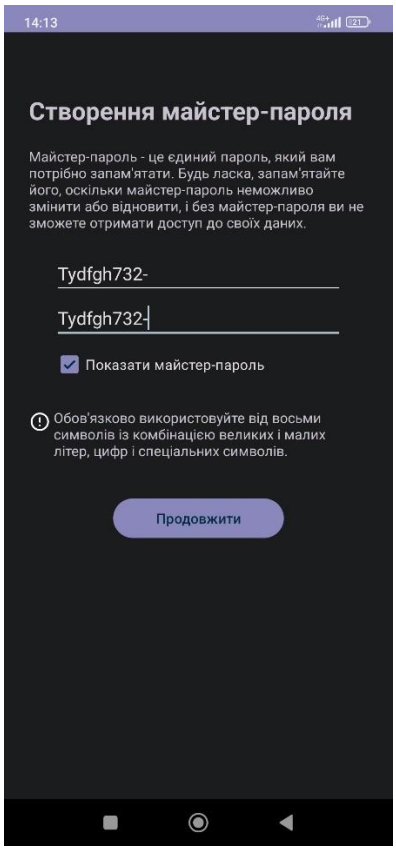

Рисунок 4.1 – Створення майстер-пароля

Після створення майстер пароля потрібно натиснути кнопку "Продовжити" після чого відкривається екран входу в додаток де потрібно ввести створений майстер-пароль (рис. 4.2).

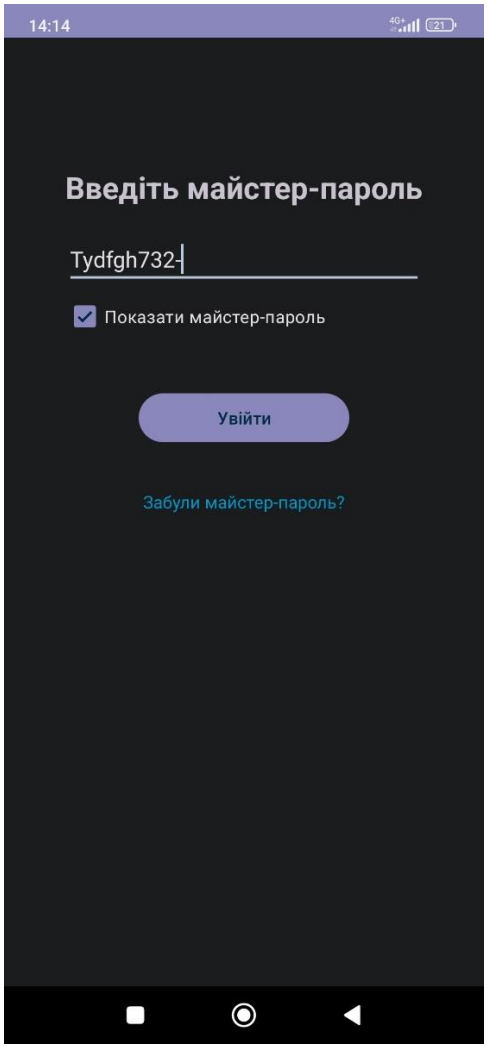

Рисунок 4.2 – Введення майстер-пароля

Після введення майстер пароля відкривається головний екран зі списком паролів (рис. 4.3). Оскільки жодного пароля поки введено не було список паролів порожній. Можна додати пароль власноруч натиснувши на кнопку з іконкою "+" чи згенерувати його за допомогою генератора паролів.

При натисканні іконки меню відкривається бокова панель, яка служить в якості меню додатка (рис. 4.4).

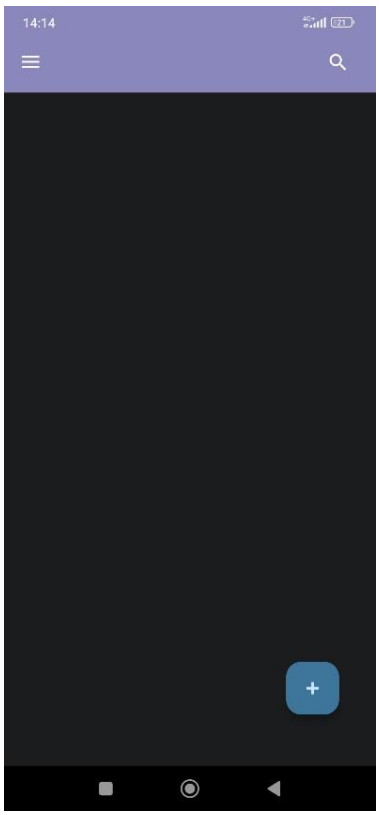

Рисунок 4.3 – Головний екран додатка

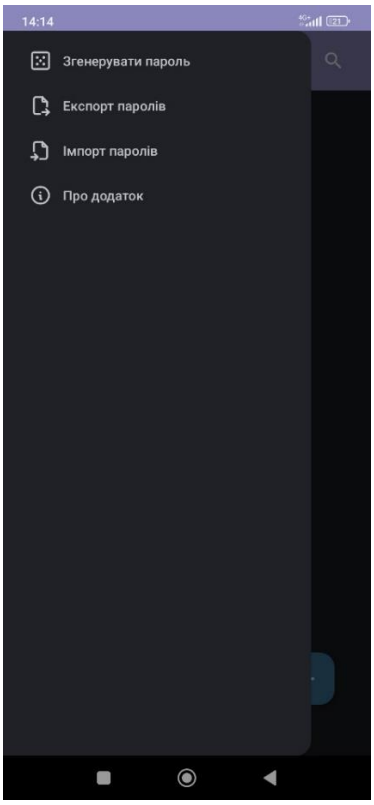

Рисунок 4.4 – Меню додатка

При натисканні на пункт меню "Згенерувати пароль" відкривається екран для генерації паролів (рис. 4.5).

На цьому екрані можна натиснути на кнопку "Згенерувати" і згенерується пароль.

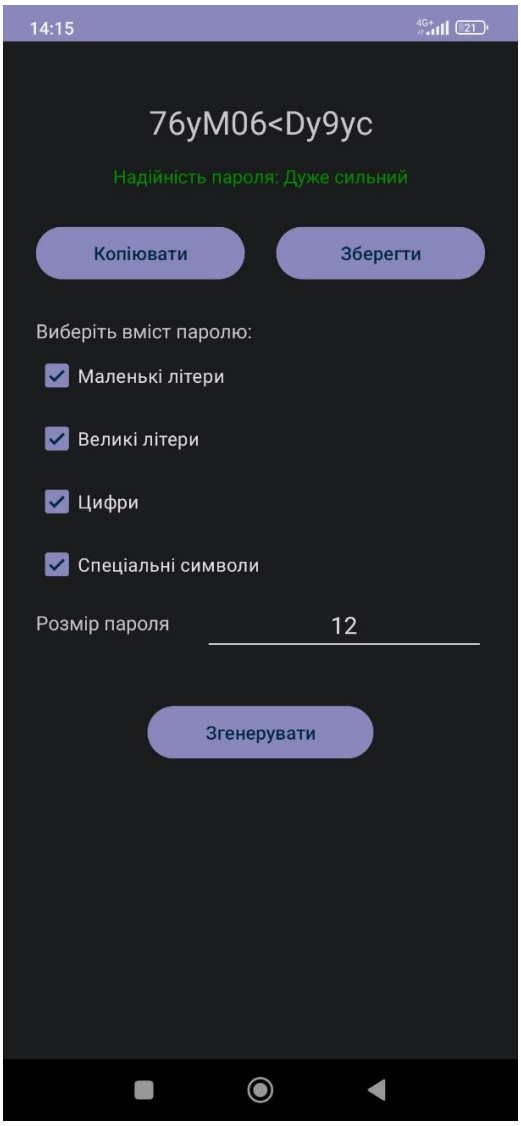

Рисунок 4.5 – Екран для генерації паролів

Після генерації пароля його можна зберегти в додатку натиснувши на кнопку "Зберегти". Після натискання на цю кнопку з'являється екран збереження пароля (рис. 4.6).

На екрані збереження пароля вводиться вся потрібна інформація про пароль та зберігається у додатку при натисканні на кнопку "Зберегти"

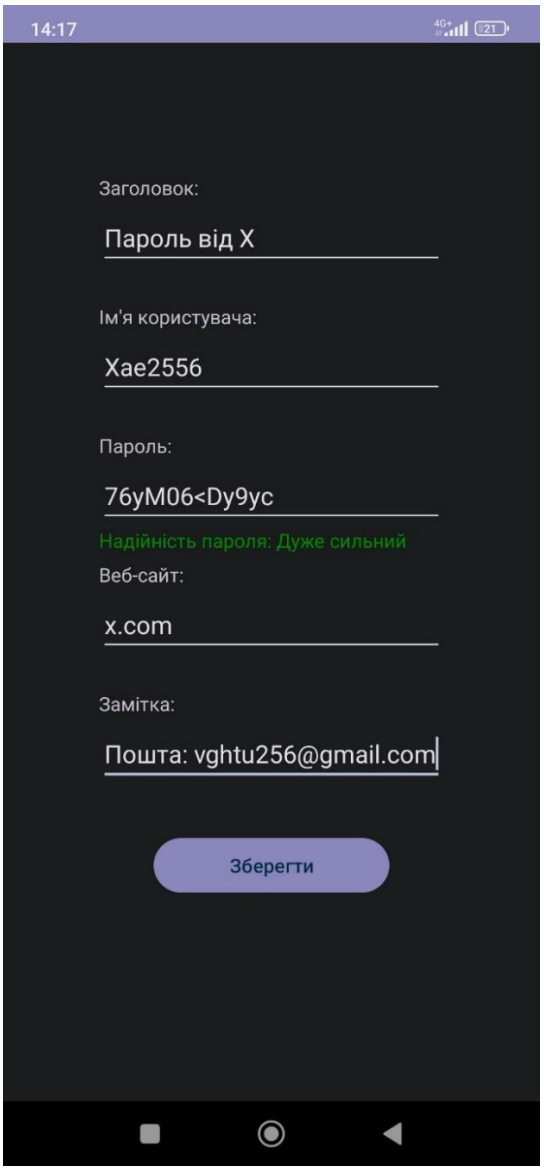

Рисунок 4.6 – Екран збереження пароля

Після натискання кнопки "Зберегти" відкривається головний екран додатка зі збереженим паролем (рис. 4.7).

При натисканні на збережений пароль відкривається інформація про цей пароль. (рис. 4.8).

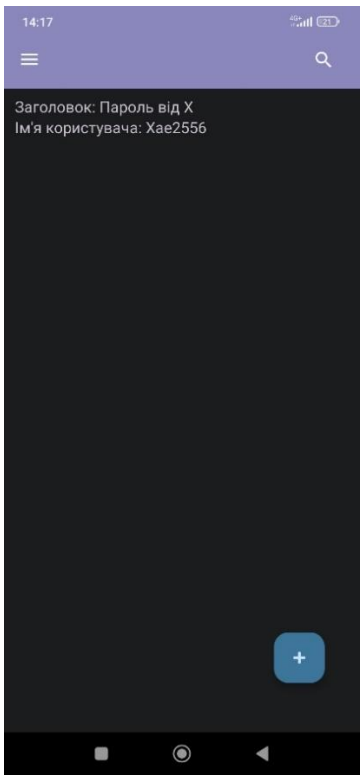

Рисунок 4.7 – Головний екран додатка зі збереженим паролем

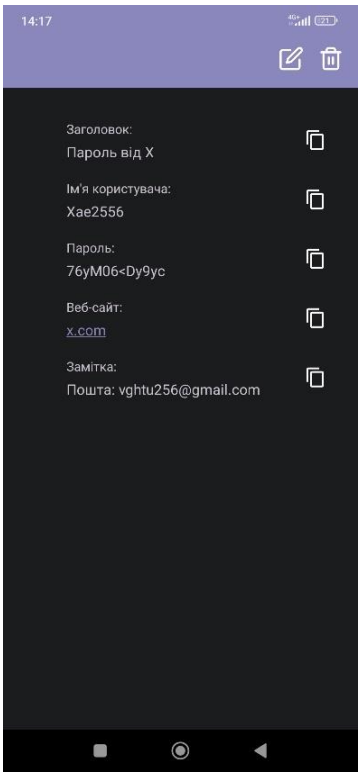

Рисунок 4.8 – Інформація про пароль

ВИСНОВКИ

Актуальність розробки алгоритмів для оптимізації генерації та аналізу паролів і ключів за допомогою методів і засобів шифрування інформації (ШІ) визначається необхідністю забезпечення високого рівня кібербезпеки в сучасному інформаційному середовищі. Зростання кількості та складності кіберзагроз, а також розширення спектру атак на різноманітні аспекти цифрової безпеки створюють великі виклики для захисту конфіденційної інформації та забезпечення надійності доступу до систем.

Використання ефективних алгоритмів генерації паролів є ключовим елементом захисту інформації. Розвиток нових методів для створення надійних і важкопіддатливих до атак парольних комбінацій дозволяє ускладнити завдання зловмисникам при спробах незаконного доступу.

Результатом виконання магістерської роботі є розроблений мобільний додаток, що містить такі функціональні можливості як: генерація паролів, ефективний аналіз паролів, захищене зберігання паролів у додатку.

Надалі є можливості до розвитку мобільного додатка, зокрема, можна додати інші мови до інтерфейсу та впровадити автозаповнення паролів у браузері та додатках.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Симметричное и асимметричное шифрование с StormGain. URL: https://stormgainclub.com/ru/simmetricnoe-i-asimmetricnoe-sifrovanie-s-stormgain-000689 (дата звернення 02.12.2023).
2. Симметричное и асимметричное шифрование. URL: https://cryptonews.net/ru/news/other/3374079/ (дата звернення 02.12.2023).
3. Password Manager SafeInCloud. URL: https://play.google.com/store/apps/details?id=com.safeincloud&hl=en (дата звернення 04.12.2023).
4. Bitwarden Password Manager. URL: https://play.google.com/store/apps/details?id=com.x8bit.bitwarden&hl=en (дата звернення 04.12.2023).
5. Dashlane - Password Manager. URL: https://play.google.com/store/apps/details?id=com.dashlane&hl=en (дата звернення 04.12.2023).
6. My Password Manager. URL: https://play.google.com/store/apps/details?id=com.er.mo.apps.mypasswords&hl=en (дата звернення 04.12.2023).
7. Keeper Password Manager. URL: https://play.google.com/store/apps/details?id=com.callpod.android_apps.keeper&hl=en (дата звернення 04.12.2023).
8. Kaspersky Password Manager. URL: https://play.google.com/store/apps/details?id=com.kaspersky.passwordmanager&hl=en (дата звернення 04.12.2023).
9. Password Safe and Manager. URL: https://play.google.com/store/apps/details?id=com.reneph.passwordsafe&hl=en (дата звернення 04.12.2023).
10. Mobile Operating System Market Share Worldwide. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (дата звернення 05.12.2023).
11. Операционная система Android. URL: https://medium.com/nuances-of-programming/операционная-система-android-826fb74c5af9 (дата звернення 05.12.2023).

12. Операционная система Android для смартфона: что это. URL: https://uchet-jkh.ru/i/operacionnaya-sistema-android-dlya-smartfona-cto-eto/ (дата звернення 05.12.2023).
13. Что такое Android Studio: основы использования и возможности. URL: https://uchet-jkh.ru/i/cto-takoe-android-studio-osnovy-ispolzovaniya-i-vozmoznosti/ (дата звернення 06.12.2023).
14. Java vs Kotlin. URL: https://www.educba.com/java-vs-kotlin/ (дата звернення 06.12.2023).
15. Unified Modeling Language. URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language (дата звернення 06.12.2023).
16. dropbox/zxcvbn. URL: https://github.com/dropbox/zxcvbn (дата звернення 07.12.2023).