МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ
ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«___»_____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ»

Тема: «Автономна система збереження конфіденційних даних користувача»

Виконавець: Гліб ХИТРИК

Керівник: к.т.н., доцент Данііл ХОДАКОВ

Нормоконтролер: к.т.н., доцент Олена ТОЛСТІКОВА

КИЇВ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *кібербезпеки, комп'ютерної та програмної інженерії*
Кафедра *Комп'ютерних інформаційних технологій*
Спеціальність *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ:
завідувач кафедри КІТ
_____Аліна САВЧЕНКО
(підпис)
«____»_____2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
Хитрика Гліба Богдановича
(ПІБ випускника)

1. Тема роботи: «Автономна система збереження конфіденційних даних користувача» затверджена наказом ректора № 1774/ст від 28.09.2022р.
2. Термін виконання роботи: з 26 вересня 2022 року по 27 листопада 2022 року.
3. Вихідні дані до роботи: додаток для збереження конфіденційних даних на мові програмування Python, підтримуваний на декількох операційних системах.
4. Зміст пояснювальної записки: 1. Аналіз предметної області. 2. Огляд технологій. 3. Проектування та реалізація додатка 4. Застосування додатка.
5. Перелік обов'язкового ілюстративного матеріалу: 1. Діаграма варіантів використання. 2. Діаграма композитної структури 3. Інфологічна модель бази даних 4. Діаграма класів. 5. Тестування додатка.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Аналіз предметної області та огляд існуючих рішень. Написання 1 розділу, представлення керівнику.	26.09.2022-04.10.2022	
2.	Вибір та опис використаних технологій. Написання 2 розділу, представлення керівнику.	05.10.2022-12.10.2022	
3.	Проектування та розробка додатка за допомогою обраного стеку технологій. Написання 3 розділу, представлення керівнику.	13.10.2022-21.10.2022	
4.	Тестування системи та опис розвитку додатка. Написання 4 розділу, представлення керівнику.	22.10.2022-05.11.2022	
5.	Загальне редагування та проходження нормоконтролю.	06.11.2022-17.11.2022	
6.	Друк пояснювальної записки, перепліт пояснювальної записки.	18.11.2022-20.11.2022	
7.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації.	21.11.2022-22.11.2022	

7. Дата видачі завдання 26.09.2022р.

Керівник кваліфікаційної роботи _____ Даніїл ХОДАКОВ
(підпис керівника)

Завдання прийняв до виконання _____ Гліб ХИТРИК
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Автономна система збереження конфіденційних даних користувача» містить: 96 сторінок, 34 рисунки, 5 таблиць, 19 інформаційних джерел, 2 додатки.

Об’єкт дослідження – додаток у вигляді менеджера паролів, який зберігає конфіденційні дані.

Предмет дослідження – види персональних даних та методи їх збереження.

Мета кваліфікаційної роботи – розробка програми збереження конфіденційних даних, використовуючи сучасний стек технологій та алгоритм шифрування.

Методи дослідження – мова програмування Python, середовище розробки PyCharm, бібліотека інтерфейсів Tkinter, фреймворк Kivy, база даних SQLite, сервер Marin, операційні системи Windows та Android.

У першому розділі описаний аналіз предметної області та огляд існуючих рішень.

У другому розділі наданий опис використовуваних технологій.

У третьому розділі представлена розробка та проектування додатку.

У четвертому розділі проведене тестування працездатності системи.

Додатки: діаграма варіантів використання, лістинг програми.

Результати кваліфікаційної роботи рекомендується використовувати на персональних пристроях користувача, які підтримують різні операційні системи.

МЕНЕДЖЕР ПАРОЛІВ, АЛГОРИТМИ ШИФРУВАННЯ, КОНФІДЕНЦІЙНА ІНФОРМАЦІЯ, ФРЕЙМВОРК, ОПЕРАЦІЙНА СИСТЕМА, МОВА ПРОГРАМУВАННЯ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

NIST *(National Institute of Standards and Technology)*	–	Національний інститут стандартів і технології
VPN *(Virtual private network)*	–	Віртуальна приватна мережа
PIN *(Personal Identification Number)*	–	Аналог до паролю, який складається тільки з цифр
OTP (One Time Password)	–	Одноразовий пароль
MFA *(Multi-factor authentication)*	–	Багатофакторна автентифікація
AES *(Advanced Encryption Standard)*	–	Симетричний алгоритм блочного шифрування
IDE *(Integrated Development Environment)*	–	Інтегроване середовище розробки
GUI *(Graphical User Interface)*	–	Графічний інтерфейс користувача
NUI *(Natural User Interface)*	–	Природній інтерфейс користувача
SDK *(Software Development Kit)*	–	Набір засобів розробки під певну платформу
UML *(Unified Modeling Language)*	–	Уніфікована мова моделювання

ВСТУП

Прості способи підключення та прямий доступ до множини різноманітних даних роблять глобальну інформаційну мережу найбільш вимогливим інструментом, який регулярно використовується користувачем для створення, збереження, обміну, обробки та розповсюдження різноманітних матеріалів у цифровій формі [1]. Можливості інтернету задіяні для досягнення багатьох цілей, як, наприклад, отримання глибоких знань у певній галузі, вивчення та перегляд медіа ресурсів, застосування дистанційних форм комунікації, ведення конфіденційних фінансових операцій та виконання безлічі інших процесів, які значно підвищують швидкість досягнення кінцевого результату, знижуючи при цьому часові та ресурсні витрати.

На даний момент люди використовують безліч веб-сервісів, включаючи соціальні мережі, електронну пошту, операції в додатку банка, державні портали надання послуг. Очевидно, дані сервіси потребують автентифікації всіх користувачів, та найбільш поширеним способом є пароль та логін. Загальні положення по безпечному користуванню інтернету рекомендують для кожного мережевого ресурсу мати новий, неповторний та унікальний пароль. Оскільки кількість сайтів та сервісів, на яких необхідно кожного разу використовувати дані облікового запису для підтвердження своєї особистості, постійно збільшуються, таким чином запам’ятовування різних варіантів парольних фраз становить проблему. На додачу до цього, паролі повинні бути стійкими до злому: довгі, які містять різні регістри, з використанням цифр, символів та який повинен мінятися з періодичністю хоча б раз в півроку.

Відповідно до проведених досліджень, через цей фактор, з часом, користувач починає використовувати прості паролі або ті, які повторюються, а також ненадійні методи їх збереження у вигляді текстового файлу або браузера з автоматичним заповненням [2]. У цьому випадку рішенням становить один з популярних варіантів віддаленого зберігання конфіденційної інформації, а саме менеджери паролів. Вони можуть бути у вигляді програмних додатків, мережевих та веб-браузерних розширень.

Менеджер паролів початково зберігає всю інформацію користувача в зашифрованому вигляді. Для розшифрування та доступу до бази даних паролів та інших персональних даних зазвичай потрібен один головний пароль. Окрім цього, додаток містить генератор паролів, який допомагає формувати безпечні та унікальні паролі, що полегшує участь клієнта у придумуванні нових.

Актуальність роботи не вгасатиме через зростаючий фактор популярності менеджерів паролів. Також, варто відмітити, що це сховище надійних даних, які в разі чого зможуть захистити інформацію від атак зловмисників. Окрім того, будь-який доступ до одного з власних ресурсів, по факту, може бути гарантованим.

Об'єкт дослідження – додаток у вигляді менеджера паролів, який зберігає конфіденційні дані.

Предмет дослідження – види персональних даних та методи їх збереження.

Метою роботи кваліфікаційного проєкту полягає в розробці додатку, який буде більш ефективним, безпечним, надійним у порівнянні з аналогами, які доступні користувачам мережі інтернет. Дана ціль може бути досягнута за допомогою використання сучасних алгоритмів шифрування. Зрештою, виявлення недоліків у застосунках такого ж роду для правильної імплементації та продуктивнішої реалізації власного продукту.

Головне завдання – поєднання наявних рішень з використанням їх кращих та сильних сторін, забезпечивши перспективу розвитку додатку в майбутньому.

Наукова цінність полягає в детальному розгляді існуючих рішень для опису та реалізації кращого застосунку у сфері інформаційних технологій, використання якого є своєрідним регламентом для цифрової безпеки користувача.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Критичний фактор збереження конфіденційних даних

Кібербезпека стала однією з найбільших галузей комп'ютерної науки та індустрії технологій, що розвивається. Несправна безпека коштувала світовій економіці величезних збитків. Нерідко буває, що фінансові та інші види втрат пов'язані зі слабкою захищеністю паролів. Компанії та люди зазвичай не роблять достатньо для забезпечення суворих інструкцій щодо формування та зберігання паролів, як це рекомендує NIST (англ. National Institute of Standards and Technology) - Національний інститут стандартів і технології.

Дана нерозсудливість приводить до того, що зловмисники користуються нагодою та мають змогу вкрасти всю інформацію з пошти, соціальних мереж, банківських облікових записів і т.д. Часто бувають випадки, коли люди роблять помилки та піддаються навіюванню, особливо коли поспішають. З цієї причини кіберзлочинці користуються цими особливостями під час атак у психологічних трюках, також відомих як соціальна інженерія. Найвідомішим прикладом таких прийомів маніпулювання є фішинг. Зокрема, кіберзлочинці надсилають електронні листи нібито від відомих організацій, друзів чи колег жертви. Отриманий лист може виглядати правдоподібно і не викликати підозри, але містити шкідливе посилання або вкладений файл. Після завантаження він може інфікувати пристрій шкідливим програмним забезпеченням або відкрити сторінку, яка вимагає заповнення особистих даних. Однак для отримання іншої особистої інформації жертви шахраї можуть здійснювати телефонні дзвінки, видаючи себе за працівників банку або технічної підтримки. Цей метод шахрайства також називають "вішинг"

(голосовий фішинг).

Ще один досить популярний засіб крадіжки – за допомогою шкідливого програмного забезпечення. Як згадувалося вище, найчастіше дане ПЗ розповсюджується за допомогою фішингових листів, але також стати жертвою можна на етапі натискання реклами в Інтернеті або навіть простого відвідування шкідливого сайту [3]. В загальному існує багато видів шкідливих програм для крадіжки паролів, але найпоширеніші містять задачу зчитувати натискання клавіатури або створення скріншотів екрану жертви та надсилання їх зловмисникам.

Атака за словником – це метод підбору паролю з використанням часто вживаних слів особистості [4]. Наприклад, якщо людина захоплюється конкретним видом спорту та часто про нього згадує у соціальній мережі, підібрати пароль цієї людини не становитиме складнощів. Зазвичай, користувачі придумують пароль на основі двох факторів: хобі або близьке оточення, та пароль, що легко запам’ятовується. У даній атаці хакери використовують спеціально підібраний список слів, що містить повсякденні слова, які жертва може використовувати. Такий список складають на етапі збирання інформації щодо конкретної особи.

Атака перебором підбирає всі можливі популярні комбінації логін/пароль. Користувачі ще не перестали використовувати прості поширені паролі, тому цей спосіб вважається ефективним і нині. Довжина та складність паролю робить злом більш скрутним. Наприклад, перебір семизначного пароля займе набагато більше часу ніж чотиризначного. Даний спосіб часто використовується для злому сайтів. Моментами вище згадані атаки використовуються в комбінації. Хакери використовують атаку за словником для генерації фраз, а потім проводять атаку шляхом перебору останніх цифр. Замість того, щоб перевіряти кожен пароль, гібридна атака використовує набір облікових даних і створює та пробує незначні модифікації фраз по всьому списку, наприклад, зміна букв або цифр.

Атака з використанням райдужних таблиць відрізняються тим, що вони зламують не самі паролі, а їх хеш-функцію. Райдужна таблиця - це попередньо обчислена таблиця, яка кешує результат хеш-алгоритмів, які зазвичай використовуються для розшифрування хешу облікових даних. Ця атака також розглядається як зламування пароля в автономному режимі, оскільки хакер не повинен взаємодіяти зі сторінкою авторизації користувача або системою напряму. Як тільки хакер отримує доступ до хеш-паролів, переходить в автономний режим і перевіряє хеші за допомогою попередньо обчисленої хеш-таблиці. Цей спосіб часто використовується при зломі пароля Wi-Fi мереж та ОС.

Всі згадані вище атаки мотивують до створення надійних паролів, а також їх зберігання у відносно безпечних додатках, як в менеджерах паролів. Наступні пункти підказують, як краще захиститись від втрати персональних даних:

– створення надійних, унікальних даних для входу, а також уникнення їх повторного використання на різних сайтах;

– увімкнення двофакторної автентифікації для всіх облікових записів, де це можливо;

– встановлення менеджера паролів, який зберігатиме та генеруватиме надійні комбінації для входу;

– зміна облікових даних, дізнавшись про витік даних нещодавно;

– перехід лише на сайти, які використовують протокол HTTPS;

– ігнорування посилань та вкладень в листах від незнайомих осіб та організацій;

– завантаження програмних засобів лише з офіційних ресурсів;

– постійне оновлення операційної системи та програм;

– користування технологією VPN (англ. virtual private network) – віртуальна приватна мережа, під час підключення до загальнодоступного Wi-Fi.

1.2. Види персональних даних

Персональні дані – це будь-яка інформація, що відноситься до конкретної людини або суб'єкта персональних даних. ПІБ, мобільний телефон, email, адреса проживання, фотографія, паспортні дані - все це є прикладом персональних даних. Важливо, щоб дані належали до конкретної людини. Тобто, абстрактний номер телефону або назва поштової скриньки не є персональними даними, так як відсутній ідентифікатор до кого ці дані прив’язані. Для прикладу, якщо в базі даних компанії зберігається ПІБ клієнта, його мобільний номер та реквізити банківського рахунку, тоді це вже будуть персональними даними – однозначно зрозуміло, до якої особи вони належать.

В загальному є 4 види персональних даних, які зображені на рис. 1.1.

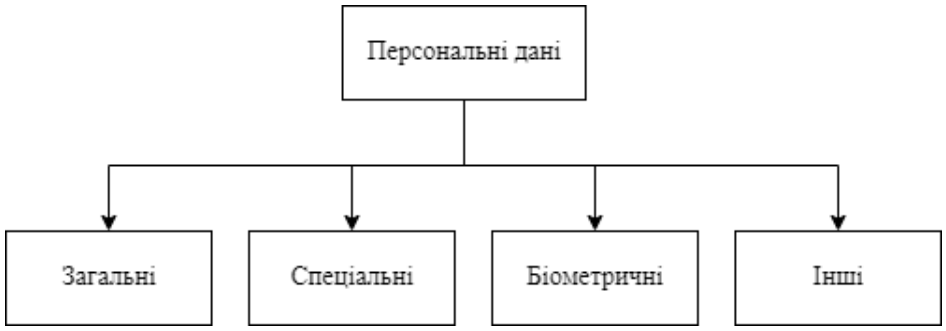


Рис. 1.1. Види персональних даних

До загальних персональних даних відносяться базові особисті дані: ПІБ, місце реєстрації, інформація про місце роботи, номер телефону, e-mail. Зазвичай ці дані вже відомі деяким оточуючим людям та вони можуть бути опублікованими в загальнодоступних джерелах. Наприклад, про місце роботи людини можуть знати його друзі у соціальних мережах або форумах.

Спеціальні персональні дані про людину можуть включати: расову та національну приналежність, політичні, філософські та релігійні погляди, інформація про судимості та подробиці інтимного життя. Їх суттєва різниця від загальних даних полягає в тому, що вони зазвичай знаходяться в закритому доступі. Тобто про них можна дізнатися лише в особистому контакті, або

зробивши офіційний запит до лікарні, поліції, суду чи державного органу. У більшості випадків, людина не зобов'язана повідомляти ці дані, так як це її особиста справа.

Біометричними даними називають фізіологічні чи біологічні особливості людини, які використовуються для ідентифікації особи [5]. До них можуть належати фотографії, відбитки пальців, група крові, генетична інформація. Однак вони не завжди є такими. Для прикладу, вони є біометричними, якщо вони зберігаються з метою ідентифікації людини. Наприклад, якщо на прохідній стоїть камера з розпізнаванням лиць, фотографії співробітників будуть біометричними даними - саме за ними визначається особистість людини. А якщо до особистої справи співробітника чи профілю клієнта прикріплено його фотографію – ці дані не біометричні. Вони не використовуються для ідентифікації, а просто доповнюють загальну інформацію профіля.

В останню категорію відносять все інше, що не належить до вищезгаданих. Тобто, це може бути належність до певної соціальної групи, як членство в спортивному клубі або корпоративні дані, які зберігаються в бухгалтерії: стажування, зарплата, кількість відпустки в рік. Найбільш тяжко інші дані відрізнити від спеціальних, але суттєва різниця полягає в тому, що спеціальні характеризують людина як особистість і людині може бути важливо, щоб про ці фактори інші не знали. У випадку з іншими даними – це додаткова інформація, яка може часто змінюватися.

Відповідно деякі дані люди мають бажання зберегти конфіденційними від чужих очей. Однак, зловмисники можуть провести несанкціонований доступ. Закон України про захист персональних даних встановлює обмеження, які намагаються запобігати цьому.

1.3. Пароль – спосіб ідентифікації користувача

Пароль – умовне слово, фраза або довільний набір знаків, який складається з літер нижнього та верхнього регістру, цифр, різноманітних символів та який вводиться для підтвердження особи або повноважень. Якщо місткість складається лише з цифр, то таку комбінацію можна ще назвати PIN-кодом (англ. Personal Identification Number). Пароль вважається найпопулярнішим метод автентифікації завдяки своїй економічності та простоті. Паролі використовуються як перша лінія захисту для майже всієї електронної інформації, мереж, серверів, пристроїв, облікових записів, баз даних тощо.

Рекомендацією до складанню пароля можна вказати поєднання слів з цифрами та спеціальними символами (*, #, $ і т.д.), використання малопоширених та неіснуючих слів [6]. Можна поєднувати всі аспекти одразу. Також, дотримуватися мінімальної довжини, а саме – 8 символів. Найкраще захищений пароль буде містити біля 16-18 елементів.

Окрім багаторазових паролів, які можуть бути скомпрометовані, існують інші методи для користувачів, які прагнуть більш безпечної альтернативи:

- динамічний пароль;
- біометричний пароль;
- BankID;
- ЕЦП – електронний цифровий підпис;
- двохфакторна автентифікація;
- автентифікація за допомогою унікального предмету.

Всі вищезгадані методи будуть детально описані в наступних підрозділах.

1.3.1. Статичний пароль

Це пароль у звичайному вигляді, який може багаторазово використовуватись для входу в обліковий запис додатку, соціальної мережі і тому подібне. Зазвичай, символи та їх послідовність придумується та запам’ятовується самим користувачем індивідуально, під час реєстрації акаунта. У любому випадку, користувач може змінити комбінацію символів, при цьому, частіше всього буде необхідно ввести старий пароль та новий, на який потрібно поміняти.

Статичний пароль не потрібно передавати стороннім третім лицям. Це може відноситися до будь-якої фінансової та державної установи.

Принцип створення надійного паролю було згадано вище, а також не потрібно забувати зберігати його у надійному місці, наприклад хороше захищеному менеджері паролів. Окрім того, ніколи не потрібно залишати свій ПК або мобільний пристрій без нагляду, так як це надає зловмисникам змоги вчинити протиправні дії або непомітно встановити шкідливе ПЗ, яке буде зчитувати та передавати дані без відома власника.

Можна зробити висновки, що підхід до придумування статичного паролю повинен бути розсудливим та чітко виваженим, адже від цього залежить безпека конфіденційних даних користувача. Також, єдиним істотним плюсом даного виду паролю є простота його реалізації. З іншого боку, відповідальність лежить на користувачу, який і відповідає за складність парольної фрази. Технічно складний пароль рекомендується міняти раз в півроку.

Очевидно, що багаторазові паролі програють одноразовим через менший захист та частоту їх використання.

1.3.1.1. Генерація пароля

Однією з найбільш часто згадуваних функцій менеджерів паролів є генерація паролів, тобто їхня здатність полегшити тягар користувача, пов'язаний із необхідністю придумувати новий пароль відповідної складності. Паролі, згенеровані таким інструментом є дуже безпечними, оскільки містять комбінацію випадково згенерованих символів ASCII.

Коли використовується програма для зберігання паролів та створюється новий пароль через генератор, зазвичай виходить щось на кшталт цього: e3Xj*7>&uOWU?, l7c5'=0XrPt5# і тому подібне.

Зауважимо, що для жодного з них немає попередньо визначеного шаблону, оскільки вони випадкові. Проте потрібно знати, що вони не настільки рандомні як може здатися на перший погляд.

В основі кожного генератора паролів лежить машина, яка називається генератором випадкових чисел. Існує три типи генераторів випадкових чисел:

– PRNG - генератори псевдовипадкових чисел;

– TRNG - справжні генератори випадкових чисел;

– CSPRNG - криптографічно захищені генератори псевдовипадкових чисел.

Випадковість насправді не є такою, тому що програміст може запрограмувати машину на генерування "випадкових" чисел, але машина знаходиться під владою свого програмування.

Генератор випадкових чисел – або в нашому випадку генератор випадкових паролів – це алгоритм, який на основі початкового числа або за допомогою постійного введення створює послідовність чисел або бітів.

Термін "навмання" можна визначити як "без чіткої мети, напряму, правила чи методу". З цієї точки зору алгоритм, розроблений людьми на основі певних правил для генератора випадкових чисел, не відповідає визначенню "випадкового". Отже, таким чином, ставлячи під сумнів його випадковість.

Іншими словами, комп’ютери створені для усунення випадковості за замовчуванням: слідувати правилам і покладатися на алгоритми під час обчислень. Детермінована природа обчислень змушує інші комп’ютери здогадуватися про те, що станеться на основі попередньої інформації. Випадковість є прямою протилежністю того, що роблять комп’ютери, ось чому детерміновані машини не можуть генерувати справді випадкові послідовності чисел.

Для захисту від хакерської атаки генератори паролів часто використовують криптографічні примітиви, такі як хеш-функції (SHA-1 або MD5) або блокові шифри (DES, Triple-DES, AES) для змішування вхідних даних або для маскування внутрішнього стану проти вихідних даних [7].

Деякі менеджери паролів використовують CSPRNG, який захищає користувача шляхом створення випадкових паролів і ключів шифрування та робить їх придатними для таких цілей.

Все-таки є спосіб генерувати машині по-справжньому рандомні рядки. Генератор справжніх випадкових чисел (TRNG) використовує фізичне як джерело для створення випадковості та введення її в комп’ютер. Це може бути зміна чиїхось рухів миші, радіоактивне джерело або атмосферний шум, який легко вловити за допомогою звичайного радіо. Використовуючи фізичне явище як джерело, процес генерування випадкових чисел передбачає виявлення невеликих, непередбачуваних змін у даних. Отже, згенерований рядок стає недетермінованим.

1.3.2. OTP – одноразовий пароль

OTP (англ. One Time Password) – одноразовий, а також відомий як динамічний пароль – це пароль, який практично завжди можна використовувати лише в лімітований період, а саме декілька секунд або хвилин після того, як пароль був сформований.

У деяких застосунках серія одноразових паролів заздалегідь визначена або навіть роздрукована, але в більшості сучасних програм генерується в режимі реального часу програмним або апаратним автентифікатором, яким володіє користувач. Автентифікатор, який є у користувача, ділиться криптографічним ключем з верифікатором, яким є програмне забезпечення, яке намагається перевірити особу користувача. Яким би способом він не був створений, кожен одноразовий пароль можна використовувати лише один раз. Верифікатор, який перевіряє пароль як засіб перевірки особи користувача, відхиляє повторне використання пароля.

У багатьох випадках використання автентифікатора OTP є лише одним з компонентів багатофакторної автентифікації. Об'єднавши одноразовий пароль з іншим фактором, таким як статичний пароль або біометричний підпис, інформація може бути надійнішою, ніж просто запам'ятований статичний пароль. Це пов'язано з тим, що зламаний пароль буде мало корисним для тих, хто намагається скомпрометувати обліковий запис або програму. При використанні статичних паролів хакер або шахрай, який отримав пароль користувача, матиме доступ до потенційно конфіденційної інформації, доки цей пароль не буде змінено. У ще гіршому сценарії той, хто скомпрометував цей обліковий запис, може змінити пароль до того, як його законний власник зможе змінити його та захистити свою інформацію.

Через використання поточного одноразового паролю можна захистити обліковий запис, тому що шахрай навіть захопивши даний пароль, не зможе його використати вдруге після власника. Користувач, який став жертвою фішингу або шкідливого програмного забезпечення, що перехоплює натискання клавіш, все одно буде захищений. Інформація залишиться у безпеці від звичайних методів крадіжки паролів.

Прикладами використання автентифікатора OTP є Google Authenticator та Steam Guard, які надають 6-ти та 5-ти значні коди відповідно, де є одна хвилина для входу у певний сервіс, наприклад на Facebook.

Єдиним крихітним недоліком є те, що потрібно кожен раз запам’ятовувати різну невелику кількість символів.

1.3.3. Біометричний пароль

Автентифікація - це спосіб перевірити безсумнівно, що особа є тим, ким вона себе видає. Біометрична автентифікація виконує цю перевірку шляхом перевірки відмінних біологічних або поведінкових характеристик [8]. Система автентифікації працює шляхом порівняння наданих даних із підтвердженою інформацією про користувача, що зберігається в базі даних. У біометричній автентифікації ця інформація визначається як фізичні чи поведінкові ознаки.

Наприклад, у системі розпізнавання обличчя різні риси обличчя обробляються та перетворюються на числові дані, які зберігаються в базі даних. Коли людина намагається увійти, система знову знімає її обличчя, витягує числові дані, а потім порівнює їх із тими, що зберігаються в базі даних. Інші типи біометричної автентифікації:

– сканування відбитків пальців;
– зіставлення ДНК;
– сканування сітківки ока;
– сканування вен;
– поведінкова біометрія.

Поведінкова біометрія перевіряє особу за допомогою аналізу фізичної та когнітивної поведінки користувача. Використовуються алгоритми машинного навчання для визначення шаблонів у поведінці та діяльності користувачів. Потім ці шаблони використовуються для визначення того, чи хтось є тим, за кого себе видає. Приклади:

– використання сенсорного екрану, а саме яку площу екрана вони займають;
– динаміка набору тексту та комбінації клавіш;
– діяльність миші.

Вся суть біометрії полягає в тому, що вона унікальна. Знаючи це, можна подумати, що біометричну автентифікацію неможливо зламати, що не є остаточною правдою. Як і будь-яка інша система, біометрична автентифікація не захищена від злому. Наприклад, сучасні алгоритми штучного інтелекту можна використовувати для створення відбитків пальців, які можуть обдурити сканери відбитків пальців.

Для підвищення захисту можна використовувати мультимодальний підхід, за якого різні біометричні дані перевіряються під час перевірки особи. Це значно ускладнює підробку зловмисникам. Наприклад, хакер може знайти фотографію людини в Інтернеті, яку він використовує, щоб успішно обдурити систему розпізнавання обличчя. Але якщо система вимагає від них надати додаткову інформацію, наприклад відео, на якому людина вимовляє свій пароль, вони навряд чи знайдуть його.

Біометрія є дуже необхідним покращенням порівняно з паролями. Паролі дуже легко зламати. Іноді все, що потрібно хакеру - це дата народження людини або її ім’я. З іншого боку, біометричні дані отримати набагато важче.

Шахраї не зможуть знайти біометричні дані людини, записані на наліпках або які автоматично заповнені в їхньому браузері. Таким чином зловмисникам стає набагато важче зламати біометричні системи без пароля, особливо ті, що використовують мультимодальну автентифікацію.

Основною причиною популярності та поширеності біометричної автентифікації є те, що користувачі вважають її набагато зручнішою. Не потрібно запам'ятовувати складний пароль або змінювати його щомісяця. Просто можна покласти палець на клавіатуру або подивитись на сканер очей.

Переваги біометричної автентифікації зображені на рис. 1.2.

Рис. 1.2. Переваги різних видів біометричної автентифікації

Так, біометрія, як правило, більш безпечна, але вона не надійна. Хакери можуть підробити біометричні дані, використовуючи різні методи, як-от завантаження чи друк фотографії людини, використання підробленого силіконового відбитка пальця або 3D-маски. Такі атаки відомі як презентаційні атаки.

Крім того, сканери відбитків пальців смартфонів часто покладаються на часткові збіги. Дослідники виявили, що можна створювати “основні відбитки”, які відповідають частинам багатьох людей і, таким чином, можуть надавати доступ до великої кількості облікових записів користувачів.

Крім того, що біометричні системи можуть бути зламаними, іноді вони можуть не розпізнавати дійсного користувача: хтось може носити нові окуляри, або голос користувача може звучати інакше, коли він хворий або щойно прокинувся.

Для бізнесу ще однією неприємною стороною біометричних даних є їх зберігання. Де б не зберігалися біометричні дані, вони повинні зберігатися надійно. Якщо біометричні дані зламано, шляху назад немає - людина не може змінити відбиток пальця чи райдужну оболонку ока.

Компанії, які вирішили зберігати біометричні дані співробітників або клієнтів, беруть на себе велику фінансову та етичну відповідальність. Це одна з причин розглянути можливість зберігання на пристрої: де біометричні дані

зберігаються на пристрої, який автентифікує користувача, наприклад на його смартфоні чи комп’ютері.

Хороші сторони біометрії все ще переважують погані та потворні аспекти настільки, що очікується, що дана технологія буде стрімко розвиватися надалі, все краще розробляючи свій захист.

1.3.4. ЕЦП

Електронний цифровий підпис - це саме те, що звучить як сучасна альтернатива підпису документів папером і ручкою. Він використовує передову математичну техніку для перевірки автентичності та цілісності цифрових повідомлень і документів. Це гарантує, що вміст повідомлення не буде змінено під час передачі, і допомагає нам подолати проблему видавання себе за іншу особу та підробки цифрових комунікацій. Цифрові підписи також надають додаткову інформацію, таку як походження повідомлення, статус і згода того, хто підписав.

Окрім цифрового підпису документів, вони також використовуються для фінансових транзакцій, постачальників послуг електронної пошти.

Використовуючи математичний алгоритм, постачальники рішень для цифрового підпису, наприклад як Zoho Sign, згенерують два ключі: відкритий та закритий. Коли відповідальний ставить цифровий підпис на документі, для документа створюється криптографічний хеш.

Цей криптографічний хеш потім шифрується за допомогою закритого ключа відправника, який зберігається в безпечній скриньці. Потім він додається до документа та надсилається одержувачам разом із відкритим ключем відправника. Даний етап процесу ілюстрований на рис. 1.3.

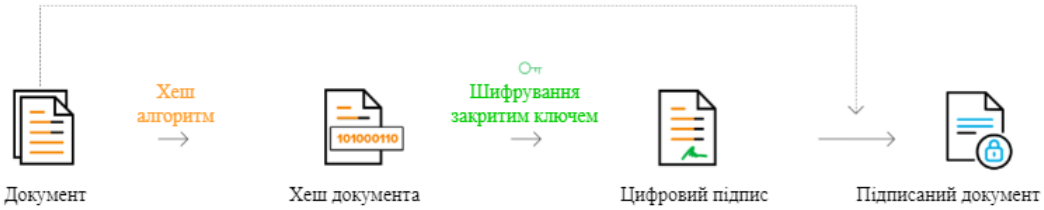


Рис. 1.3. Алгоритм підписання документа зі сторони відправника

Одержувач може розшифрувати зашифрований хеш за допомогою сертифіката відкритого ключа відправника. Криптографічний хеш знову генерується на стороні одержувача.

Обидва криптографічних хеші порівнюються для перевірки їх автентичності. Даний порядок дій відображено на рис. 1.4.

Рис. 1.4. Алгоритм перевірки документа зі сторони одержувача

Якщо хеші збігаються, документ не підроблено та вважається дійсним.

1.3.5. BankID

Основною метою створення BankID в Україні є забезпечення надійної та зручної ідентифікації користувачів для надання адміністративних та банківських послуг через Інтернет на спеціальних порталах.

BankID вирішує проблему ідентифікації користувачів через Інтернет, щоб надати, наприклад, довідку про заробітну плату. Сервіс спочатку повинен переконатися, що саме конкретна особа запитує інформацію. Якщо

громадянин обирає ідентифікацію через BankID, він вводить логін і пароль свого інтернет-банку. Надалі, він проходить другий етап авторизації за допомогою одноразового пароля з SMS, підтверджуючи свою особу.

BankID вирішує проблему ідентифікації користувачів через Інтернет: щоб надати, наприклад, довідку про заробітну плату, сервіс спочатку повинен переконатися, що саме конкретна особа запитує інформацію. Якщо громадянин обирає ідентифікацію через BankID, він вводить логін і пароль свого інтернет-банкінгу, проходить другий етап авторизації (наприклад, введення одноразового пароля з SMS) і таким чином підтверджує свою особу.

Для замовлення довідки, внесення даних до реєстру тощо, достатньо двох умов:

– бути клієнтом банку, підключеного до системи BankID;
– портал повинен бути підключений до однієї системи BankID.

Логіка системи базується на організації онлайн-запитів із сервісних порталів до банківської системи конкретного банку через єдиний шлюз, який є центральним хабом системи BankID НБУ, та передачі адресних даних у підписаному та зашифрованому вигляді. Усі запити проходять виключно через центральний вузол BankID НБУ і лише за натисканням кнопки клієнтом.

Дані, які банк може передавати на портал, такі:

– повне ім'я;
– дата народження;
– серія та номер паспорта;
– ідентифікаційний номер (ІПН);
– скан-копії паспорта та ІПН;
– адреса реєстрації;
– телефонний номер;
– адреса електронної пошти.

Ідентифікація через BankID – це практично те саме, що й особиста перевірка документів у банках. Це затверджено стандартами реєстрації та зберігання даних клієнтів і контролюється державою, а саме в особі

Національного банку України. Доступ до даних клієнта при використанні BankID захищений так само надійно, як і доступ до коштів через Інтернет-банк.

1.3.6. Багатофакторна автентифікація

MFA (англ. Multi-factor authentication) – це багатофакторна автентифікація, метод якої вимагає від користувача надати 2 або більше верифікаційних факторів для того, щоб отримати доступ до такого ресурсу як застосунок, онлайн акаунт або VPN [9]. MFA є основним компонентом надійної політики керування ідентифікацією та доступом. Замість стандартного прохання ввести пароль та логін, дана автентифікація просить ще надати зазвичай один, або більше факторів перевірки, що зменшує собою ймовірність вдалої кібератаки зі сторони. Наступний рис. 1.5 зображує порядок роботи MFA у простому вигляді:

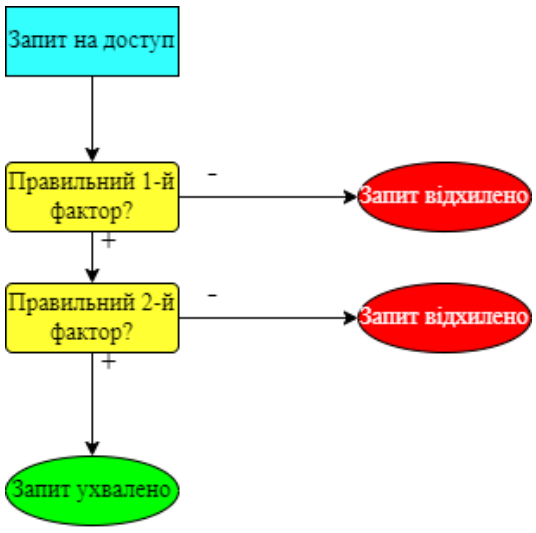


Рис. 1.5. Алгоритм роботи MFA

Основна перевага даної автентифікації це підсилення безпеки організації, вимагаючи від користувачів трішки більше ніж просто пароль з логіном. Важливо згадати, що імена користувачів та паролі вразливі до атак грубої сили, та можуть бути викрадені третіми особами. Примусове

використання MFA, наприклад відбитка вказівного пальця або фізичного апаратного ключа, означає підвищену впевненість у тому, що організація зможе підвищити безпеку від атак кіберзлочинців.

Найбільш частим випадком додаткової перевірочної інформації є вищезгаданий OTP – одноразовий пароль. Зазвичай, це 4-8 значні коди, які можна отримати за допомогою електронної пошти, SMS, або самого мобільного додатку. Новий код може генеруватися періодично або в той же час, коли надсилається запит на автентифікацію. Код генерується засновуючись на початкову значенні, яке призначається користувачеві під час першої реєстрації і деякого іншого фактура, яким може бути поступово збільшуваний інкремент або значення часу на поточний момент.

Більшість методологій автентифікації MFA базується на одному з трьох типів додаткової інформації:

– речі, які особа знає, наприклад пароль або PIN-код;

– речі, які особа має, наприклад картка або смартфон;

– речі, які становлять частину особи, тобто відбиток пальця або розпізнавання голосу.

Стало відомо, що багатофакторна автентифікація почала інтегрувати машинне навчання та штучний інтелект, тому її методи тепер стають більш комплексними, включаючи підтвердження на основі місцезнаходження. У такому випадку розглядається IP-адреса користувача та геолокація при можливості. Умовне можливе блокування доступу користувачу, якщо інформація про його місцезнаходження не є вказаною у білому списку. З іншого боку, її можна використовувати як додаткову форму автентифікації на додаток до інших факторів, таких одноразовий пароль для підтвердження ідентифікації даного користувача.

Кіберзлочинці витрачають своє життя на те, щоб викрасти цінну інформацію конкретної персони, і саме по цій причині ефективна та примусова стратегія MFA повинна бути першою лінією захисту від них. Присутній план

захисту даних заощадить час і гроші як особи, так і її організації в майбутньому.

1.3.7. Автентифікація за допомогою унікального предмету

Автентифікація на основі маркерів – це протокол, який дозволяє користувачам підтверджувати свою особу та отримувати натомість унікальний маркер доступу. Протягом терміну дії токена користувачі отримують доступ до веб-сайту чи програми, для якої було видано токен, замість того, щоб повторно вводити облікові дані кожного разу, коли вони повертаються до тієї самої веб-сторінки, програми чи будь-якого ресурсу, захищеного тим самим маркером.

Токени автентифікації працюють як проштампований квиток. Користувач зберігає доступ, доки маркер залишається дійсним. Коли користувач виходить із системи або закриває програму, маркер стає недійсним. Токени пропонують другий рівень безпеки, і адміністратори мають детальний контроль над кожною дією та транзакцією.

На даний момент існує три поширені типи маркерів автентифікації.

Контактний полягає в підключені певного пристрою до системи для доступу: диски, накопичувачі, ключі та інші фізичні елементи. Аналогія може бути проведена з використанням USB-пристрою або смарт-карти для входу в систему.

Безконтактний – це коли пристрій знаходиться достатньо близько до сервера, щоб з’єднатися з ним, але він не підключається. Так зване “магічне кільце” Microsoft може бути прикладом такого типу маркера.

Дистанційний – пристрій може спілкуватися із сервером на великій відстані, навіть якщо він взагалі ніколи не торкається іншого пристрою. Приклад може бути наведений на основі того, що згадувалося вище, а саме телефон для процесу двофакторної автентифікації, це і є використання цього типу маркера.

1.4. Методи та алгоритми шифрування даних

Шифрування даних – це досить поширена та ефективна практика захисту, а саме збереження інформації організації чи корпорації від сторонніх осіб. Нині існує декілька різних методів шифрування, які будуть розглядатися згодом.

У теперішній ситуації у світі, де зростає кількість зловмисників та їх кіберзлочинів, можна припустити, що існує так само багато методів захисту мережі, скільки і потенціальних її проникнень. Основне завдання полягає в тому, щоб визначити, які саме методи та алгоритми повинен застосовувати експерт з безпеки, щоб найкраще відповідати поточній ситуації в організації, враховуючи її політику.

Шифрування даних можна описати як спосіб захисту даних шляхом їх кодування таким чином, що їх може розшифрувати або отримати доступ до них лише та особа, яка має правильний ключ шифрування. Коли особа або організація отримує доступ до зашифрованих даних без дозволу, вони виглядають зашифрованими або нечитабельними. Тобто, це процес перетворення з читабельних даних на зашифрований формат. Це все потрібно для того, щоб сторонні особи не могли їх прочитати під час передачі. Звичайно, шифрування можна використовувати до файлів, документів, повідомлень і т.д.

Шифрування вважається основним будівельним блоком безпеки даних, який широко використовується великими організаціями, малими підприємствами та окремими споживачами. Майже все, що людина може спостерігати в Інтернеті, пройшло через певний рівень шифрування, будь то веб-сайти чи програми.

Дані, які потрібно зашифрувати, називають частіше всього відкритим текстом. Відкритий текст потрібно передати за допомогою певних алгоритмів шифрування, які в основному є математичними обчисленнями, які виконуються з необробленою інформацією. Існує кілька алгоритмів

шифрування, кожен з яких відрізняється своїм застосуванням та індексом безпеки. Окрім самого алгоритму, потрібно ще також ключ шифрування. За допомогою згаданого ключа та відповідного алгоритму шифрування, відкритий текст перетворюється на зашифрований фрагмент даних, також відомий як зашифрований текст.

Коли зашифроване формулювання дійшло до призначеного отримувача, ця людина може використати ключ дешифрування, для того, щоб шифрований текст перетворити у стандартний читабельний формат, тобто той же відкритий текст. Цей ключ розшифрування має постійно зберігатися в секреті та може бути не подібним на ключ, який експлуатується для шифрування повідомлення. Даний процес зображений на рис. 1.6:

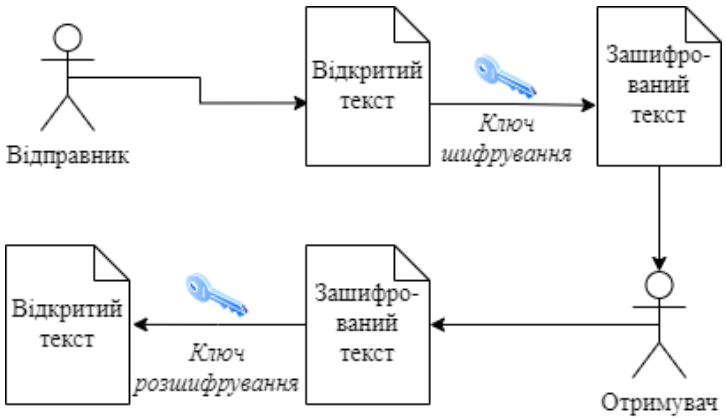


Рис. 1.6. Процес передачі та отримання зашифрованого повідомлення

Варто зазначити, що ключі виконують всю фактичну роботу з шифрування/дешифрування, залишаючи обом учасникам процесу більше часу.

Існує декілька доступних підходів стосовно шифрування даних. Більшість експертів по безпеці в Інтернеті розбивають шифрування на 3 методи, а саме:

– симетричне;
– асиметричне;
– хешування.

Вони, у свою чергу, поділяються на різні типи. Розглянемо кожен з методів окремо.

1.4.1. Симетричний метод шифрування

Також, може ще називатися криптографією із закритим ключем або алгоритмом секретного ключа. Цей метод потребує, щоб відправник та отримувач мали доступ до одного ключа. Таким чином, одержувач повинен мати ключ до того, як повідомлення буде розшифровано. Цей метод найкраще працює для закритих систем, які мають менший ризик вторгнення сторонніх осіб.

Істотною перевагою є те, що симетричне шифрування швидше, ніж асиметричне. Однак негативним є те, що ключ надійно зберігається та доступний лише програмному забезпеченню, яке має його використовувати.

Загальними плюсами можна вказати такі:

– виключна безпечність, якщо в даному шифруванні використовується такий безпечний алгоритм як AES (англ. Advanced Encryption Standard);

– швидкість, яка полягає в тому, що шифрування та дешифрування даних з симетричним ключем є досить простим, тому це забезпечує продуктивність читання та запису;

– вимагає низьких комп’ютерних ресурсів, так як наявний лише один ключ;

– мінімізація компромісу повідомлення, так як окремий ключ використовується для зв’язку лише з однією стороною, що запобігає широкому порушенню безпеки повідомлень.

Недоліками можна назвати:

– спільне використання ключа, так як його потрібно передати іншій стороні з ким проводиться комунікація, тому потрібен безпечний спосіб його доставки;

– якщо безпека буде скомпрометована у вигляді отримання симетричного ключа, то даним ключем можна декодувати все, що було ним зашифровано, тому, таким чином, обидві сторони розмови є вразливими;

– не можна гарантувати походження та автентичність повідомлення, оскільки як отримувач, так і відправник мають той самий ключ.

1.4.2. Асиметричний метод шифрування

Цей метод, також званий криптографією з відкритим ключем, використовує два ключі для процесу шифрування, а саме відкритий і закритий, які математично пов’язані між собою. Користувач використовує один ключ для шифрування, а інший – для дешифрування, хоча не має значення, який вибереться першим.

Як випливає з назви, відкритий ключ є у вільному доступі для будь-кого, тоді як закритий ключ залишається лише в тих учасників, яким він потрібен для розшифрування повідомлень. Обидва ключі – це просто великі числа, які не є ідентичними, але поєднані один з одним, і саме тут виникає “асиметрична” частина.

Переваги:

– наявна можливість автентифікувати повідомлення, оскільки шифрування з відкритим ключем дозволяє використовувати цифрові підписи, одержувачі повідомлень зможуть перевірити, чи вони справді надходять від певного відправника;

– зручність, яка полягає в розповсюдженні ключів для шифрування, коли кожен публікує свої відкриті ключі, а приватні ключі зберігаються в секреті;

– за допомогою цифрових підписів у шифруванні з відкритим ключем, одержувачі повідомлення можуть виявити, чи було повідомлення змінено під час передачі.

Недоліки:

– шифрування з відкритим ключем є повільнішим ніж симетричне шифрування, тому він не підходить для розшифрування масованої кількості інформації;

– відкриті ключі не є перевіреними, тобто ніхто достеменно не знає чи відкритий ключ належить саме тій особі, яку він визначає, тому потрібна перевірка стосовно того, що відкриті ключі належать відповідним особам;

– втрата закритого ключа є непоправною і тому повідомлення не зможуть розшифруватися;

– якщо закритий ключ ідентифікований зловмисником, всі повідомлення можуть бути скомпрометованими.

1.4.3. Хешування

Хешування генерує унікальний підпис фіксованої довжини для набору даних або повідомлення [10]. Кожне конкретне повідомлення має свій унікальний хеш, що дозволяє легко відстежувати незначні зміни в інформації. Дані, зашифровані за допомогою хешування, не можна розшифрувати або повернути до початкової форми. Тому хешування використовується лише як метод перевірки даних.

Багато фахівців з безпеки в Інтернеті навіть не вважають хешування фактичним методом шифрування, але сама межа, яка це відділяє, є достатньо розмитою для того, щоб класифікація була в силі. Суть хешування полягає в тому, що це ефективний спосіб перевірки чи ніхто не змінював інформацію.

Розглянувши методи шифрування даних, можна тепер приступити до загального огляду алгоритмів шифрування.

1.4.4. Алгоритми шифрування

Для перетворення даних у зашифрований текст використовуються різноманітні алгоритми шифрування. Експлуатуючи ключ шифрування,

алгоритм може змінювати дані в передбачуваній манері, у результаті чого вони виглядають випадковими або хаотичними, але їх можна перетворити назад у відкритий текст за допомогою ключа дешифрування.

Звичайно, на даний момент існує безліч різних алгоритмів шифрування, но в даній роботі будуть розглядатися п’ять, які є одні з найбільш поширених.

AES, також знаний як Advanced Encryption Standard - надійний стандартний алгоритм, який використовується урядом Сполучених Штатів, а також іншими організаціями та корпораціями. Він надзвичайно є ефективним у 128-бітній формі, AES також може використовувати 192 та 256-бітні ключі для дуже вимогливих цілей шифрування у разі збереження критично конфіденційної інформації. В загальному AES вважається невразливим практично до всіх атак, окрім перебором грубої сили. Незважаючи на це, багато експертів з безпеки в інтернеті вважають, що AES з часом вважатиметься основним стандартом для шифрування даних у приватних галузях та секторах.

Потрійний DES, який є наступником канонічного алгоритму Data Encryption Standard, який був створений для формування відповіді хакерам, яким вдалося з’ясувати як зламати DES. Це було симетричне шифрування, яке колись було найбільш поширеним алгоритмом в індустрії, хоча поступово він звільняє свої зайняті ніші. 3DES застосовує алгоритм DES тричі до кожного блоку даних і зазвичай використовується для шифрування паролів UNIX та PIN-кодів банкоматів.

RSA – названий на честь розробників Rivest-Shamir-Adleman, це асиметричний алгоритм і стандарт шифрування інформації, яка передається через Інтернет. Шифрування RSA є міцним і надійним, оскільки створює величезну купу тарабарщини, яка розчаровує потенційних хакерів, змушуючи їх витрачати багато часу та енергії на злом систем. Працює він на основі розкладання добутку двох великих простих чисел на множники. Лише користувач, який має інформацію стосовно того, які ці два числа, може успішно розшифрувати повідомлення. Цифрові підписи зазвичай

використовують RSA, але алгоритм сповільнюється під час шифрування великих обсягів даних.

Blowfish – ще один алгоритм, розроблений для заміни DES. Цей симетричний інструмент розбиває повідомлення на 64-розрядні блоки та шифрує їх індивідуально. Blowfish має репутацію швидкості, гнучкості та непорушності. Це загальнодоступне надбання, тож це робить його безкоштовним і ще більше додає йому привабливості. Blowfish зазвичай зустрічається на платформах електронної комерції, у захисті платежів і в інструментах керування паролями.

Twofish, навіть спираючись на назву, можна зрозуміти, що це є послідовник Blowfish. Являється симетричним шифруванням без потреби ліцензії, яке розшифровує 128-бітні блоки даних. Окрім цього, Twofish завжди шифрує дані в 16 циклів, незалежно від розміру ключа. Twofish ідеально підходить як для програмного, так і для апаратного середовища і вважається одним із найшвидших у своєму роді. Нині багато сучасних програмних рішень для шифрування файлів і папок використовують саме цей метод.

1.4.5. Майбутнє шифрування даних

Шифрування використовується в технічний продуктах та інструментах які є побутовими та купляються щодня. Воно й надалі залишатиметься основою безпеки для всього, від комп’ютерних ігор до телефонних дзвінків та відеочатів. Якщо інформацію можна надіслати або зберегти, значить швидше за все вона буде відповідно зашифрована, таке широке використання нині присутнє.

Зі зміною технологій змінюватимуться і типи шифрування, які розробляються та використовуються. Хакери стають все більш витонченими у своїх зусиллях та атаках, змушуючи професіоналів та експертів, які створюють ці безпечні інструменти з наміром випередити зловмисників.

Деякі зусилля зосереджені на збільшенні розмірів ключів для захисту від дешифрування методом грубої сили. Інші зусилля спрямовані на пошук нових криптографічних алгоритмів. Наприклад, Національний інститут стандартів і технологій оцінює алгоритм відкритого ключа наступного покоління, який має бути квантово безпечним. Проблема полягає в тому, що більшість квантово-безпечних алгоритмів неефективні в класичних комп'ютерних архітектурах. Для вирішення цієї проблеми, індустрія сфокусована на розробці прискорювачів для пришвидшення алгоритмів на платформах x86.

Окремим напрямком дослідження можна назвати гомоморфне шифрування. Дивовижна концепція полягає в тому, що вона дозволяє користувачам виконувати обчислення із зашифрованими даними без їх попереднього розшифровування. Таким чином, аналітик, який потребує цього, може зробити запит до бази даних, що містить секретну інформацію, без необхідності просити керівника з вищим рівнем пріоритету отримати доступ до даних або вимагати розсекречення даних.

Великою перевагою можна назвати те, що воно захищає дані в усіх станах. Тобто, у стані спокою, коли вони зберігається на жорсткому диску. Також, коли вони в стані руху, коли дані передаються через мережу та під час використання даних у пам'яті комп'ютера. На додачу, він є квантово безпечний, оскільки він базується на тій самій математиці, що й квантові обчислення. Недоліком ж є те, що гомоморфне шифрування поки погано та неефективно працює на традиційних комп'ютерах, оскільки воно не налаштоване на роботу з ними. Індустрія співпрацює над розробкою інструкцій для x86 систем, щоб ці нові криптосистеми працювали на хмарних швидкостях. Передбачається, що у ближній термін можна буде спостерігати практичне застосування даного шифрування.

Незважаючи на це, потрібно зауважити, що треба не тільки покладатися на найпопулярніші типи шифрування, а й постійно оновлювати та обслуговувати своє авторитетне програмне, так і апаратне забезпечення відповідно до вказівок виробників даного устаткування.

1.5. Методи збереження персональної інформації

Зберігання цифрових даних - це, по суті, запис цифрової інформації на носії, як правило, електронними засобами. Теперішні запам'ятовуючі пристрої або множинні сервіси дозволяють користувачу зберігати великий обсяг даних, а також вільно користуватися ними, надсилаючи та обмінюючись з іншими. Можна зберігати дані як постійно, так і тимчасово.

У нинішню цифрову еру в кожної людини є сукупність даних: фотографії, відео, документи, паролі, файли, архіви і так далі. Зазвичай, людина навіть несвідомо використовує декілька методів збереження персональної інформації, а саме у вигляді файлів на персональному комп'ютері чи ноутбуці, залитих даних у хмарне середовище, яке надає фіксований об'єм простору для їх зберігання, або зовнішні пристрої як диски, флешки, жорсткі диски і тому подібне.

Більшість сталих способів збереження інформації буде описано в наступних підрозділах.

1.5.1. Персональний комп'ютер

Якщо у людини наявний персональний комп'ютер, то ймовірно, там є дані та інформація, які зберігаються. Вони можуть бути як неструктурованими або ж навпаки, розфасованими по відповідним категоріям та папкам. Як згадувалося, це може бути будь-який вид, від важливих бізнес документів до звичайних сімейних фотографій. Єдина проблема цих даних полягає в тому, що при форс-мажорах або непередбачуваних обставин вони можуть бути втраченими або пошкодженими. Відповідно, вони не є довговічними і це становить питання, якими же засобами чи сервісами потрібно користуватися для того, щоб уникнути будь-яких цифрових втрат.

Розглядаючи саме ПК, можна виділити 2 способи альтернативного зберігання даних – резервне копіювання та архівування. Може здатися, що

вони достатньо схожі, хоча служать різним цілям. Основна відмінність полягає в тому, що резерв - це копія медіа даних, до яких людина регулярно отримує доступ. Це можуть бути навчальні або робочі файли. Резервні копії легко створювати та швидко отримувати до них доступ. Наприклад, в операційній системі Windows 10 можна відновити файли з резервної копії, створеної за допомогою опції резервного копіювання та відновлення.

Архіви призначені для медіа файлів, до яких людина не збирається регулярно звертатися. Це можуть бути старі фотографії, готові документи та будь-що інше, що потрібно зберегти, а не видалити. Доступ до даних можливий, але це займає трохи більше часу, для того щоб отримати їх.

Обидва способи роблять майже те саме, але архівні файли не повинні бути частиною звичайного розкладу резервного копіювання. При потребі створити резервну копію даних, потрібно її зробити лише для тих даних, які регулярно використовуються або над якими йде робота. Все інше можна оформити в архів.

1.5.2. Зовнішні пристрої

За весь час розвитку інформаційних технологій, методи збереження даних еволюціонували та трансформувалися плинним чином. Всі наступні пристрої підпадають під одну категорію, так як вони слугували додатковими зовнішніми засобами, де можна було зберігати інформацію. Деякими з них ще користуються в теперішній час, а деякі ще досі залишаються актуальними.

Hard Disk Drive та Solid-State Drive, або відповідно HDD та SSD, можна знайти майже в кожному настільному та портативному комп’ютері. Вони зберігають файли для операційної системи та програмного забезпечення, а також документи користувача, такі як фотографії, текстові файли, відео та аудіо. Жорсткий диск використовує магнітну пам’ять для запису та отримання цифрової інформації на один або кілька швидко обертових дисків. Твердотільний накопичувач в свою чергу використовує флеш-пам’ять для

зберігання даних та іноді використовується в таких пристроях, як ноутбуки та настільні комп’ютери, замість традиційного жорсткого диска. Переваги SSD над HDD включають більш високу швидкість читання/запису, безшумну роботу, більшу надійність і менше енергоспоживання.

DVD (англ. Digital Versatile Disk) та диск Blu-ray - це формати зберігання даних на цифрових оптичних дисках, які витіснили компакт-диски, головним чином через їх набагато більшу ємність. Наприклад, диск Blu-ray може зберігати 25 ГБ (гігабайт) даних на одношаровому диску та 50 ГБ на двошаровому відповідно.

Останніми в даному списку є USB флеш-накопичувачі. Флеш-пам'ять, як правило, більш ефективна та надійна ніж в оптичних носіях, оскільки вона швидша та має набагато більшу ємність. Флешки також довговічніші через відсутність рухомих частин.

1.5.3. Соціальні мережі та месенджери

Неочевидний, але реальний спосіб збереження даних. З часом, зростання популяризації різних соціальних мереж привела до висновку, що це свого роду портали з особистою інформацією, до яких зазвичай є доступ. Для прикладу, можна зберігати свої фотографії у соціальній мережі Instagram, а документи та робочі файли у категорії “збережені повідомлення” месенджеру Telegram.

Перевага, яка присутня в даному підході полягає в тому, що зайві файли не займають фізичного простору на відповідному пристрої або гаджеті. З іншого боку, доступ до них буде лише при наявному стабільному підключенню до інтернету. Окрім того, якість збереження фотографій або відео часто представлена у низькій роздільній здатності, що може не задовільнити потреби деяких користувачів.

1.5.4. Хмарні сховища

Гнучким, інноваційним та перспективним рішенням для збереження власної інформації є хмарне сховище. Воно представляє собою сукупність великої кількості серверів, які знаходяться в центрах обробки даних. Тобто, при завантаженні файлів у дану хмару, вони будуть зберігатися віддалено на серверах.

Таким чином, однією з переваг буде те, що не використовується дисковий простір ПК, що дає змогу ефективно розподіляти наявне місце на ньому. Надається значно більший обсяг простору в порівнянні з тими пристроями, які були згадані вище. На додачу, доступ до даних наявний з різноманітних девайсів, таких як мобільний телефон або ПК. Також, при втраті або поломки, наприклад, жорсткого диску або іншого зовнішнього пристрою, шанс відновити ці дані не становить високим. У хмарному сховищі немає таких ризиків: для даних створюються резервні копії, а власник має до них доступ будь-коли та з будь-якої точки, де єдина умова – підключення до Інтернету. Наявна функція миттєвої синхронізації, а також деякі хмари містять історію редакцій, для того щоб користувач міг повернутися до старішої версії завантаженого цифрового файлу.

На даний момент існує багато реалізованих хмарних сховищ, вибір якої залежить від потреб самого користувача. Внизу наведений список найбільш популярних та безпечних сховищ:

- Dropbox;
- Google Disc;
- Microsoft OneDrive;
- Mega;
- pCloud.

Недоліки, які присутні в тому чи іншому сховищі, можуть відрізнятися. Наприклад, в Microsoft OneDrive надають лише 5 Гб вільного простору, що не є великим об’ємом зважаючи на сучасні реалії. В pCloud лімітована швидкість

завантаження файлу. У деяких хмарах наявні лише платні підписки, але за дані кошти надається більш безпечний сервіс для завантажених даних.

1.5.5. Менеджери паролів

Ексклюзивні паролі на основі ентропії з точки зору безпеки є дійсним попитом багатьох онлайн-сервісів. Але паролі, які важко вгадати зловмиснику, також важко запам’ятати користувачам, з цієї причини частіше створюються слабкіші паролі, щоб уникнути когнітивного тягаря їх згадування. Відповідно до статистики останніх років, кількість паролів, які користувач повинен запам’ятати, зростає, а середній користувач Інтернету має 25, а то й більше різних облікових записів. Той факт, що різні сайти мають різні правила складності, часто пом’якшує точне повторення пароля. З іншого боку, користувачі часто використовують прості тактики, щоб обійти ці правила, як-от внесення незначних змін до популярного пароля. Наприклад, додавання цифри в кінці пароля, який використовується на іншому сайті.

Менеджери паролів намагаються вирішити цю проблему, а саме генеруючи та зберігаючи паролі на комп’ютері, а потім надавати їх користувачеві за потреби. Доступно багато менеджерів паролів; деякі вбудовані в браузери, інші пропонуються сторонніми розробниками, а є багато мережевих, наприклад 1password, де облікові дані зберігаються в хмарі та синхронізуються між пристроями користувача.

Коли справа доходить до створення пароля, людська пам’ять зосереджена на знайомих речах, які її оточують, а також на повторенні конкретної фрази. Всі ці фактори роблять людину відкритою до нападу. Середньостатистичний людський розум може запам’ятати біля 7 символів, якщо за ними не закріплена певна асоціація. Тому, досить важко запам’ятати величезний набір символів та цифр, який вважається безпечним для автентифікації на будь-якому інтернет-сервісі.

То що таке менеджер паролів? У найпростішому розумінні менеджер паролів - це інструмент, який зберігає облікові дані користувача (тобто логін користувача та пароль), щоб зменшити когнітивне навантаження, пов’язане із запам’ятовуванням багатьох унікальних облікових даних для входу. Сховище паролів – це, по суті, назва цієї колекції паролів. В ідеалі сховище має зберігатися в зашифрованому вигляді, при цьому ключ шифрування зазвичай генерується з мастер-пароля, який у свою чергу придумує користувач. За бажанням сховище паролів можна зберігати в режимі онлайн, що дозволяє синхронізувати кілька пристроїв. Більшість сучасних менеджерів паролів можуть допомагати користувачам створювати паролі, а саме пропонують їх генерацію на додаток до зберігання вибраних користувачем паролів. Довжина передбачуваного пароля, бажаний набір символів і будь-які конкретні параметри та пріоритети, які повинен мати пароль - усе це вхідні дані для генератора паролів. Генератор паролів генерує випадковим чином пароль, який відповідає наданим критеріям.

Багато менеджерів паролів додатково допомагають користувачам входити на веб-сайти шляхом автоматичного заповнення необхідного логіна користувача та пароля. Загальна робота менеджера паролів зображена на рис. 1.7.

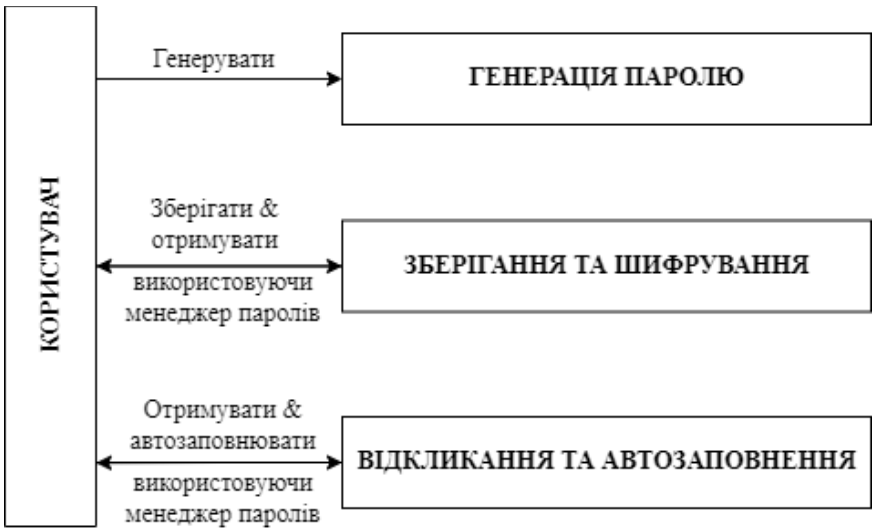


Рис. 1.7. Загальна робота менеджера паролів

Схильність до використання менеджерів паролів має змішаний зв’язок із довірою. Оцінка загрози втрати пароля людьми є більш важливим мотиватором прийняття менеджера паролів, ніж віра в саму технологію. Якщо вже порівнювати технологічні та нетехнологічні рішення, довіра, ймовірно, відіграє більш значну роль. Менеджери паролів використовуються, в свою чергу, через передбачувану вразливість і серйозність втрати пароля.

Менеджер паролів може надати ряд істотних переваг користувачу, якщо він правильно налаштований та використовується у доречних цілях:

1) Звільняє від зобов’язання запам’ятовування логінів та паролів та всієї іншої персональної інформації.

2) Легко встановити унікальний пароль для кожного веб-сайту та сервісу, запобігаючи його повторенню.

3) Створення паролів, які є стійкими до вгадування як онлайн, так і офлайн, є досить простим процесом.

Однак, потрібно підкреслити такий недолік, що менеджери паролів можуть стати єдиною точкою збою, через їх неналежну реалізацію, що може призвести до втрати всіх облікових даних користувача, а також їх використання в шкідливих цілях. Щоб захистити паролі, менеджери паролів повинні заповнювати облікові дані лише тоді, коли користувач явно дозволив операцію, тоді облікові дані зіставляються з веб-доменом або програмою, яку потрібно заповнити, і відповідна інформація видима лише для зіставленої програми чи веб-домену.

1.6. Огляд існуючих рішень

Використовуючи один із найкращих менеджерів паролів, можна достатньо просто, але істотно підвищити безпеку в Інтернеті. Це через те, що не потрібно запам’ятовувати десятки довгих складних паролів, або покладатися на такі, які можна скомпрометувати. Натомість буде лише один майстер-пароль, який може розблокувати дані для всіх облікових записів.

Варто зазначити, що жоден із найкращих менеджерів паролів не дозволяє відновити головний пароль, якщо користувач його не пам’ятає або загубив, тому слід записати його та зберігати в безпечному місці. Однак деякі з них дозволяють відновити доступ до облікового запису іншими способами з міркувань безпеки.

Одні з найкращих рішень також можуть, на основі критеріїв, швидко та легко генерувати надійні та складні паролі, щоб не придумувати їх самостійно. Більшість менеджерів можуть автоматично заповнювати форми входу замість користувача, як наприклад введення логіну номери кредитних карток. Звичайно, найкращі рішення, шифрують та надійно зберігають всі конфіденційні дані на серверах менеджера паролів. Також, часто є можливість увімкнення двофакторної автентифікації, що ускладнює злом зі сторони.

Найкращі менеджери паролів пропонують версії для Windows, Mac, Android та iOS, але деякі також підтримують операційну систему Linux. Є вірогідність синхронізувати свої паролі на необмеженій кількості пристроїв, але за це, можливо, доведеться заплатити. Безкоштовні менеджери паролів часто обмежують синхронізацію паролів одним або двома пристроями.

У деяких може поставати таке питання як: навіщо використовувати окремий менеджер паролів, коли більшість сучасних браузерів тепер можуть зберігати паролі? Це тому, що викрасти паролі з веб-браузерів неважко, а зловмисне ПЗ, яке може це робити, стає все більш поширеним.

Хоча існує багато справді погано реалізованих рішень, які є неефективними, надто складними та дорогими. У даній роботі будуть порівнюватися аналоги, які вважаються якісними на ринку. Аналіз буде проведений, зосереджуючись на взаємодії з користувачем, підтримці платформи, безпеці та загальній продуктивності.

1.6.1. 1Password

Він надзвичайно безпечний, багатофункціональний та достатньо інтуїтивний зрозумілий, з недорогими планами підписки як для окремих користувачів, так і для сімей. Дані користувача захищаються за допомогою незламного 256-бітного шифрування AES, який є тим самим типом шифрування, який використовується в банках та у воєнній галузі по всьому світу. Має політику нульового знання, що означає, що ніхто, окрім користувача, ніколи не зможе отримати доступ до сховища конфіденційних даних.

1Password також дозволяє створювати кілька сховищ, які роблять процес керування паролями та даними платіжних карток легшим, а також забезпечується покращений контроль під час обміну паролями - можна ввімкнути спільний доступ до одного сховища, але також зберігати елементи, якими не має бажання ділитися, у окреме сховище.

Відмінні функції безпеки, які присутні в 1Password:

– 2FA;
– сторожова вежа;
– режим подорожі;
– опція локального зберігання даних;
– картки конфіденційності.

Перша функція синхронізується з програмами для одноразового пароля, такими як Authy, USB-ключами, такими як YubiKey і Fido, і біометричними сканерами (обличчя, відбитків пальців і очей) для Windows, Android та iOS. 1Password також має вбудований 2FA-автентифікатор.

Сторожова вежа сканує різні кутки мережи та загальнодоступні бази даних на наявність злитих логінів та фінансової інформації. Таким чином, наявна перевірка сховища паролів на предмет безпеки та генерація паролів справді високої якості.

Наявний так званий “режим подорожі”. Приховує конфіденційні паролі від сховища, щоб нав’язлива перевірка на кордоні не могла отримати доступ до особистих даних.

Опція локального зберігання даних синхронізує комп’ютер із пристроєм Android або iOS через локальну бездротову мережу за допомогою сервера WLAN.

Суть карток конфіденційності полягає в тому, що надаються віртуальні платіжні картки для маскування фактичного номера дебетової картки під час здійснення онлайн-покупок.

На додачу, інтерфейс виконаний у світлій темі. Є достатньо прстим, зручним, що робить його дійсно хорошим вибором як для досвідчених користувачів, так і для початківців.

Єдині недоліки, які можна навести:

– мобільні версії є надто простими;
– відсутність безкоштовної версії.

1.6.2. Dashlane

Dashlane є надзвичайно безпечним, дуже простим у використанні та включає широкий діапазон видатних функцій, які не пропонують інші бренди. Dashlane надає:

– VPN (з необмеженими даними);
– моніторинг темної мережі;
– спільне використання пароля;
– аудит надійності пароля;
– аварійний доступ;
– безпечне зберігання файлів (1 Гб).

Усі функції зрозумілі та працюють належним чином. Функція аудиту - перевірка надійності пароля та сповіщення користувача, якщо будь-який

пароль є слабким, повторно використаним або зламаним. Можна також використовувати генератор паролів, щоб змінити свої на надзвичайно надійні.

На даний момент це єдиний менеджер в цьому списку з віртуальною приватною мережею, яка є безпечною, швидкою і де гарантований доступ до потокових сайтів.

Містить хороший безкоштовний план. Однак, в цьому випадку забезпечується зберігання лише 50 паролів на 1 пристрої, але зате присутні такі функції безпеки, які не пропонуються іншими брендами в пакетах преміум-класу. Містить також автоматичне заповнення та спільний доступ до паролів.

Dashlane також пропонує два плани найвищого рівня: Premium (для 1 користувача) і Premium Family (для 6 користувачів). Dashlane Premium трохи дорожчий, ніж в деяких конкурентів, але він має більше можливостей і функцій, ніж більшість менеджерів паролів. Потрібно пам’ятати таку деталь, що всі покупки Dashlane супроводжуються гарантією 30-денного повернення коштів.

Єдиний недолік, який можна вказати, це те, що як для керування паролями, це є достатньо недешевий варіант на ринку.

1.6.3. Keeper

Keeper – це інтуїтивно зрозумілий менеджер паролів з дуже високим рівнем захисту у вигляді 256-бітного шифрування AES. Присутня політика нульового знання та широкий спектр параметрів багатофакторної автентифікації, у тому числі сумісність з Google Authenticator. Окрім того, наявний вхід за обличчям та відбитком пальця на мобільних пристроях і розумних годинниках.

Неймовірно просто обмінюватися логінами з іншими користувачами, а також налаштовувати спільний доступ. Налаштування за замовчуванням для

спільного доступу до паролів “лише читання”, але є змога надавати іншим більше контролю над спільними паролями.

Keeper має також такі додаткові функції як:

– безпечний обмін повідомленнями (KeeperChat);
– зашифроване хмарне сховище (10 Гб);
– аудит безпеки паролів;
– моніторинг темної мережі.

KeeperChat – це зашифрований месенджер із широким набором опцій для безпечного надсилання та отримання повідомлень, включаючи відкликання повідомлень, самознищення та приватну галерею для зберігання фотографій і відео.

Keeper надає більший об’єм хмарного сховище, ніж конкуренти. Наприклад, 1Password та Dashlane, які згадані вище, включають 1 Гб хмарного сховища. В той час, Keeper надає 10 Гб хмарного сховища з можливістю оновлення до 100 Гб. У цьому плані в даного менеджера немає прямих конкурентів.

У Keeper Unlimited здобувається необмежена кількість паролів на необмеженій кількості пристроїв, обмін паролями та багатофакторна автентифікацію. А Keeper Family додає до 5 ліцензій і 10 Гб хмарного сховища. Можна протестувати всі преміум-функції за допомогою 30-денної безкоштовної пробної версії.

Незважаючи на всі плюси, які описані, є лише 2 недоліки присутні в даному програмному забезпеченні:

– окремі програми, які необхідні для біометрії робочого столу;
– можуть виникати глюки.

1.6.4. Bitwarden

Bitwarden – це менеджер паролів із відкритим вихідним кодом із чудовими функціями безпеки та дуже доступними планами. Дані користувачів

захищаються за допомогою галузевих стандартних функцій безпеки, таких як 256-бітове шифрування AES, політика нульового знання, широкий спектр опцій 2FA та такі розширені функції як-от локальне зберігання даних, аудит безпеки паролів і моніторинг витоку даних.

Присутній хороший та доступний безкоштовний план – він дозволяє необмежену кількість паролів для зберігання на необмеженій кількості пристроїв. Зазвичай провідні менеджери паролів обмежують користувача лише одним пристроєм і не надають змоги ділитися паролями в безкоштовних версіях.

Тим не менш, Bitwarden не такий простий у використанні, як інші аналоги. Наприклад, імпорт паролів з інших менеджерів паролів і обмін паролями є досить складними. Крім того, інтерфейс мобільних і настільних додатків Bitwarden не такий інтуїтивно зрозумілий і зручний, як у провідних конкурентів, таких як 1Password і Dashlane. Однак функції автоматичного збереження та заповнення працюють коректно та паролі синхронізуються між пристроями.

Bitwarden має дешеві тарифні плани, які підкріплені 30-денною гарантією повернення грошей.

Недоліки:

– немає особливостей як таких;
– не всі функції інтуїтивно зрозумілі у використанні.

1.6.5. RoboForm

RoboForm має масу функцій безпеки, пропонує доступні тарифні плани для окремих персон та сімей та містить найкращі можливості заповнення форм з усіх описаних менеджерів паролів. Він здатний автоматично заповнювати деякі з найскладніших веб-форм із ідеальною точністю лише за один клік.

Можливо створити кілька ідентифікаторів для веб-форм із 8 різними категоріями інформації, включаючи дані про паспорт, дебетову картку та

транспортний засіб. Форми від соціальних мереж до онлайн-форми бухгалтерського обліку можуть заповнитись без жодних помилок та пропущених полів.

RoboForm поставляється з:

– кілька варіантів 2FA;
– аудит безпеки паролів;
– моніторинг порушень даних;
– безпечний пароль і спільний доступ до нотаток;
– безпечне зберігання закладок;
– аварійний доступ.

Наявна змога легко та просто поділитися логінами з іншими користувачами, надати екстрений доступ довіреним контактам і перевірити своє сховище паролів на наявність слабких, повторюваних або іншим чином зламаних паролів. Містить інтеграцію з додатками 2FA як Google Authenticator.

Особливість даного менеджеру полягає в безпечному зберіганні закладок, яке дозволяє користувачам зберігати та синхронізувати закладки з браузера на будь-якому настільному пристрої, де встановлено RoboForm. Тобто, отримати доступ до всіх улюблених сайтів на будь-якому пристрої користувача.

Присутні недоліки:

– неінтуїтивна десктопна програма;
– реалізовані опції не настільки якісні, як в аналогів.

1.6.6. Аналіз аналогів

За допомогою детального огляду існуючих топових програмних рішень на ринку, можна підкреслити плюси та мінуси кожного з них. Для успішної розробки потрібно спиратися на сильні сторони конкурентів, реалізовуючи їх у застосунку, та покривати будь-які недоліки власними пропозиціями. Також,

не забувати експериментувати, так як це може стати свого роду інновацією у сфері інформаційних технологій. Порівняння вище описаних менеджерів паролів наведено в таблиці 1.1.

Таблиця 1.1.

Аналіз характеристик менеджерів паролів

Менеджер паролів	Вбудований генератор TOTP	Зашифроване сховище	Унікальні особливості
1Password	+	1 Гб	Кілька сховищ, сторожова вежа, режим подорожі, віртуальні платіжні картки
DashLane	+	1 Гб	VPN, аудит безпеки паролів, моніторинг темної мережі
Keeper	+	10 Гб	Зашифрований обмін повідомленнями, безпечне зберігання, моніторинг темної мережі
Bitwarden	+	1 Гб	Відкритий код, вбудований 2FA, доступна ціна

Продовження таблиці 1.1.

RoboForm	+	-	Багато шаблонів для заповнення форм, безпечний обмін нотатками

Порівняння цінової політики тих же менеджерів наведене в таблиці 1.2.

Таблиця 1.2.

Цінова політика менеджерів паролів

Менеджер паролів	Безкоштовний план	Випробувальний період	Гарантія повернення грошей
1Password	-	14 днів	-
DashLane	1 девайс, 50 паролів	30 днів	30 днів
Keeper	1 девайс, необмежена кількість паролів	30 днів	-
Bitwarden	Необмежена кількість девайсів, необмежена кількість паролів	-	30 днів
RoboForm	1 девайс, необмежена кількість паролів	30 днів	30 днів

Аналіз характеристик дав зрозуміти, що в кожного рішення є свої особливості і достатньо складно виділити умовну першість серед них. Зважаючи на дані подані в таблицях, можна визначити декілька параметрів, які необхідно буде реалізувати у власному додатку: об’єм більший за 1 Гб зашифрованого сховища, вбудований VPN, аудит безпеки паролів, зашифрований обмін повідомленнями та нотатками, декілька сховищ та вбудована 2FA. У кожного аналога є свої переваги, які будуть взяті для розробки додатку з глибинним функціоналом.

Стосовно цінової політики, тут найкраще показав себе RoboForm, так як тут присутній випробувальний період та надається гарантія повернення грошей, так само як і в DashLane. Однак під час безкоштовного плану в RoboForm надається необмежена кількість паролів для зберігання на 1 девайсі, коли в DashLane є фіксована кількість в числі 50.

Цінова політика розглядалася в цій роботі через урахування перспектив, у тому випадку, якщо розроблений застосунок буде виходити на широкий ринок програмних продуктів в майбутньому.

ВИСНОВКИ ДО РОЗДІЛУ 1

Проведений аналіз предметної області дозволив розглянути такі поняття як види персональних даних, пароль та його форми, методи та алгоритми шифрування даних, методи збереження інформації. Теоретичний масив даних надав орієнтир та вектор майбутньої розробки, дозволив частково сформувати формат майбутнього додатку.

Огляд існуючих рішень представив переваги та недоліки наявних додатків на ринку, на які можна опиратися та які потрібно уникати під час створення власного рішення. Також, був розглянутий аспект ринкової політики кожного з них, що дозволяє створити власний фінансовий шлях після повної технічної розробки продукту.

РОЗДІЛ 2

ОГЛЯД ТЕХНОЛОГІЙ

2.1. Python – мова програмування

Python – це інтерпретована, інтерактивна, багатоцільова та об'єктно-орієнтована мова програмування, яка створена Гвідо Ван Россумом наприкінці 1980-х років [11]. Дана мова програмування являється тією, яку розуміють як люди, так і комп'ютери. Вона є досить потужною, а також вирізняється використанням достатньо простого синтаксису, що робить її ідеальною для вивчення новачками.

Також, це гнучка мова, яка дозволяє писати прості програми та створювати складні програмні рішення. Python широко застосовується для веб та інтернет-розробок, для математичних та наукових обчислень даних, а також у сфері розробки ігор та графіки. Youtube, Dropbox, Instagram, Pinterest – популярні веб-сайти, які використовують дану мову програмування. Python є чудовим вибором завдяки зрозумілій природі та широким можливостям.

Деякі з особливостей Python, через які користувачі починають вивчати дану мову:

– популярність, а саме одна з найбільш поширених та розвинених мов програмування, яку використовують такі компанії-розробники програмного забезпечення як Netflix, Spotify, Quora, Google тощо;

– інтерпретація, що означає, що програми передаються прямо до інтерпретатора, який безпосередньо їх виконує на відміну від компілятора, де вихідний код спочатку перетворюється у машинний перед запуском;

– вихідний код, який є відкритим, що дає змогу вільно використовувати та розповсюджувати його навіть у комерційних цілях;

– портативність, яка дозволяє коду, написаному на одній платформі, працювати на будь-якій іншій, якщо встановлено інтерпретатор Python;

– простота, яка полягає в тому, що використовується менше ключових слів порівняно з такими мовами як C++ або Java.

Завантаження певної версії можна виконати з офіційного ресурсу Python Software Foundation.

Після встановлення можна виконати перевірку за допомогою інтерактивної оболонки Python. Наприклад, у Windows потрібно відкрити термінал та ввести “python” для Python 2.7 або “py -3” для Python 3.0.

В загальному Python складається з двох видів помилок, а саме синтаксичні та винятки або помилки під час виконання програми.

Існує декілька підходів для ефективного способу запуску та виконання коду на Python: shell, IDLE, редактор коду.

Python Shell корисна для простих однорядкових операторів. Очікуються вхідні команди від користувача та повертається результат виконання. Це найменш потужний серед трьох способів. Наявний один істотний недолік. Код, написаний у даній оболонці не є постійним, що означає неможливість використовувати код повторно.

IDLE (англ. Integrated Development and Learning Environment) – схожий на оболонку та містить як вікно оболонки, так і вікно редактора. Можна створювати та зберігати код, оскільки дозволяється його багаторазове використання. Однак, він все ще стоїть на другому місці за потужністю.

Редактор коду є найпотужнішим серед усіх. Це текстовий редактор, який корисний для редагування вихідних кодів комп’ютерних програм. Може бути окремою програмою або діяти як IDE (англ. Integrated Development Environment) – інтегроване середовище розробки. Зазвичай виділяється простотою використання, нумерацією рядків, автоматичним відступом, підсвічуванням синтаксису та можливістю додавання додаткових функцій.

Гарне та коректне кодування залежить від його способу, а саме правильному підтриманню стилю. Також, логіка є найважливішою частиною.

Найефективніше її використання – логічно розбити проблему на різні частини, а потім вирішувати її одну за одною. Останнє, але не менш важливе, постійне вдосконалення концепцій, які виникають на шляху розробки програмного забезпечення.

2.2. PyCharm – середовище розробки

PyCharm – це гібридна платформа, яка розроблена JetBrains як IDE для Python [12]. Вона зазвичай використовується для розробки додатків Python. Деякі масштабні організації використовують дану платформу, такі як: Twitter, Facebook, Amazon. Підтримуються дві версії, а саме v2.x та v3.x Можливий запуск на декількох операційних системах як Windows, Linux, macOS. Також, платформа містить модулі та пакети, які допомагають програмістам розробляти програмне забезпечення за допомогою Python за менший час і з мінімальними зусиллями. Крім того, її також можна налаштувати відповідно до вимог розробників.

PyCharm містить свої суттєві особливості. Інтелектуальний редактор коду допомагає писати високоякісні коди, складається з кольорових схем для ключових слів, класів і функцій. Це, в свою чергу, допомагає підвищити читабельність та розуміння коду. Звісно, це допомагає легко виявляти синтаксичні помилки. Забезпечується функція автозаповнення та інструкція щодо завершення коду.

Код навігації допомагає розробникам редагувати та вдосконалювати код із меншими зусиллями та часом. За допомогою навігації по коду розробник може легко перейти до функції, класу або файлу. Програміст може швидко знайти елемент, символ або змінну у вихідному коді. Крім того, використовуючи режим лінзи, розробник може ретельно перевірити та налагодити весь вихідний код.

Рефакторинг дозволяє вносити ефективні та швидкі зміни як до локальних, так і до глобальних змін. Надається змога розробникам

покращувати внутрішню структуру без зміни зовнішньої продуктивності коду. Це також допомагає розділяти більш розширені класи та функції за допомогою методу екстракту.

Присутня підтримка багатьох веб-технологій таких як HTML, CSS і JavaScript. Розробники мають можливість редагувати в реальному часу за допомогою цієї IDE. Водночас можна попередньо переглянути створену або оновлену веб-сторінку. Розробники можуть стежити за змінами безпосередньо у веб-браузері. PyCharm також підтримує AnglularJS і NodeJS для розробки веб-додатків.

Наявна підтримка різноманітних веб-фреймворків. Для прикладу, можна розглядати один з найвідоміших як Django. Забезпечується автозаповнення та пропозиції щодо параметрів даного фреймворку. Це допомагає в налагодженні коду. Також підтримуються інші популярні веб-фреймворки як web2py та Pyramid.

Забезпечуються такі наукові бібліотеки як Matplotlib, NumPy, Anaconda. Ці наукові бібліотеки допомагають створювати проекти у сфері машинного навчання. Складається з інтерактивних графіків, які допомагають розробникам зрозуміти та проаналізувати дані. Є можливість інтегруватися з різними інструментами, такими як IPython, Pytest. Ця інтеграція допомагає створювати інноваційні унікальні рішення.

На даний момент на ринку присутні дві версії – Community та Professional. Перша версія є безкоштовною та має відкрити й вихідний код, друга ж є платною. Після скачування можна налаштовувати інтегроване середовище розробки під власні потреби.

Існує понад 500 компаній, які використовують PyCharm для розробки додатків: Gorgias, Hivemind, Sainsburys, Lyft і т.д.

Однак, як і в будь-якому продукті, існують плюси та мінуси. До переваг можна віднести:

– простота інсталяції та експлуатації;
– наявно багато корисних плагінів;

– функції автозаповнення та розфарбовування;

– перегляд вихідного коду в один клік;

– швидша розробка програмного забезпечення;

– функція виділення помилок;

– велика спільнота розробників, за допомогою якої можна вирішувати запити та проблеми.

Мінусів досить менше, серед них:

– Professional версія є істотно дорогою;

– не вважається хорошим варіантом для початківців через функцію автозаповнення;

– можуть створитися проблеми під час лагодження таких інструментів як venv.

2.3. Tkinter – графічна бібліотека інтерфейсів

Tkinter – це кросплатформна структура GUI, яка вбудована в стандартну бібліотеку Python. Тобто, той самий код можна використовувати в будь-якій операційній системі, наприклад Linux, macOS і Windows. Tkinter надає об'єктно-орієнтований інтерфейс для набору інструментів Tk GUI.

Порівняно з іншими доступними варіантами, Tkinter легкий і простий у використанні. Тому це найкращий вибір для швидкого створення додатків, які можуть працювати між платформами та не потребують сучасного вигляду.

Розробка десктопних програм за допомогою цього засобу не є складним завданням. Пусте вікно верхнього рівня можна створити за допомогою наступних кроків:

1) Імпортування модуля Tkinter.

2) Створення головного вікна програми.

3) Додавання до вікна таких віджетів як мітки, рамки, кнопки тощо.

4) Викликання головного циклу подій, щоб дії могли відбуватися на екрані комп’ютера.

Tkinter надає різноманітні елементи керування, такі як кнопки, мітки та текстові поля, що використовуються у GUI. Ці елементи керування зазвичай називаються віджетами. У таблиці 2.1 наведені основні з них та їх короткий опис.

Таблиця 2.1.

Основні віджети Tkinter

Віджет	Опис
Button	Використовується для відображення кнопок в додатку
Canvas	Використовується для малювання форм, таких як: лінії, овали, полігональні фігури та прямокутники
Checkbutton	Використовується для відображення декількох параметрів у вигляді галочок. Користувач може вибрати кілька варіантів одночасно
Entry	Використовується для відображення однорядкового текстового поля для прийняття значень від користувача
Frame	Використовується як віджет-контейнер для організації інших віджетів
Label	Використовується для однорядкового підпису для інших віджетів. Може містити зображення
ListBox	Використовується для надання списку опцій користувачу

Продовження таблиці 2.1.

Menu	Використовується для надання різноманітних команд користувачу
Message	Використовується для відображення багаторядкових текстових полів для прийняття значень від користувача
Radiobutton	Використовується для відображення кількох параметрів у вигляді перемикачів. Користувач може вибрати лише один варіант за раз
Scale	Використовується для надання слайдеру, за допомогою якого можна регулювати масштаб
Scrollbar	Використовується для надання користувачу смуги прокручування вгору та вниз
Text	Використовується для відображення тексту в декількох рядках

Усі дані віджети мають доступ до певних методів керування геометрією, метою яких є їх організація у всій батьківській області віджетів. Tkinter надає наступні класи:

– метод pack() – цей менеджер геометрії організовує віджети в блоки перед тим, як розмістити їх у батьківському віджеті;

– метод grid() – цей менеджер геометрії організовує віджети у вигляді таблиці в батьківському віджеті;

– метод place() – цей менеджер геометрії впорядковує віджети, розміщуючи їх у певній позиції в батьківському віджеті.

2.4. Kivy – фреймворк

Нині багато розробників працюють над мобільними та веб-додатками. Python не має вбудованих можливостей розробки, але існують пакети, які можна використовувати для створення мобільних програм як-от Kivy, PyQt, Toga.

Усі ці бібліотеки є основними сутностями в мобільному просторі Python. Однак, є деякі переваги при експлуатації саме фреймворку Kivy. Це не тільки полягатиме в тому, що додаток буде виглядати однаково на всіх платформах, а також відпадає необхідність повторно компілювати код після кожної зміни. Більше того, можливо використовувати зрозумілий синтаксис Python для розробки власного додатку.

Kivy вперше було випущено на початку 2011 року. Дану структуру Python можна розгорнути на Windows, Mac, Linux і Raspberry Pi [13]. Крім звичайного введення з клавіатури та миші, підтримуються мультисенсорні події. Kivy навіть підтримує GPU-прискорення своєї графіки, оскільки вони побудовані з використанням OpenGL ES2. Проект використовує ліцензію MIT, тому можливо використовувати цю бібліотеку для безкоштовного та комерційного програмного забезпечення.

Коли програміст створює програму за допомогою Kivy, він створює NUI (англ. Natural User Interface) – природній інтерфейс користувача. Ідея полягає в тому, що користувач може легко та просто навчитися користуватися програмним забезпеченням без будь-яких підказок та інструкцій.

Kivy не намагається використовувати власні елементи керування чи віджети. Усі віджети намальовані власноруч зі сторони користувачів. Це означає, що програми будуть виглядати ідентично на всіх платформах. Однак, це також позначає, що відчуття його використання будуть відрізнятися в залежності від операційної системи через її функціонал, який відрізняється серед аналогів. Це може бути перевагою чи недоліком, залежно від ситуації та аудиторії.

2.5. SQLite – база даних

SQLite – це бібліотека програмного забезпечення, яка впроваджує систему керування реляційною базою даних [14]. Написана мовою програмування С. В основному позиціонується не як окрема програма, а як бібліотека, яку розробники вбудовують у програми. Це найпоширеніший механізм баз даних, оскільки він використовується кількома найкращими веб-браузерами, операційними системами, мобільними телефонами та іншими вбудованими системами [15].

Частина “Lite” у назві позначає легкість з точки зору налаштування, адміністрування бази даних та необхідних ресурсів. Якщо всі функції ввімкнено, розмір бібліотеки може бути менше 750 КБ, залежно від цільової платформи та налаштувань оптимізації компілятора. Існує компроміс між використанням пам’яті та швидкістю. SQLite зазвичай працює швидше, чим більше пам’яті йому наділено. Тим не менш, продуктивність зазвичай досить хороша навіть у середовищах з низьким обсягом пам'яті. Починаючи з версії 3.33.0, максимальний підтримуваний розмір бази даних становить 281 ТБ.

SQLite має такі помітні особливості: самодостатність, без серверу, без конфігурації, транзакційність.

Зазвичай для керування типовими реляційними системами базами даних як-от MySQL чи PostgreSQL потрібен окремий серверний процес. Програми, які хочуть отримати доступ до серверу бази даних, використовують протокол TCP/IP для надсилання та отримання запитів. Це називається архітектурою клієнт/сервер. Наступний рис. 2.1 ілюструє дану архітектуру:

Рис. 2.1. Клієнт-серверна архітектура у відношенні реляційних баз даних

У випадку SQLite архітектура має інший вигляд. Вона на потребує серверу. База даних SQLite інтегрована з програмою, яка отримує доступ до бази даних. Програми взаємодіють із базою даних SQLite, читаючи та записуючи безпосередньо з файлів бази даних, що зберігаються на диску. Рис. 2.2 ілюструє даний вид взаємодії:

Рис. 2.2. Безсерверна архітектура SQLite

SQLite є самодостатнім, тобто таким, який вимагає мінімальної підтримки з боку операційної системи чи зовнішньої бібліотеки. Це робить SQLite придатним для використання в будь-якому середовищі, особливо у вбудованих девайсах як iPhone, телефони на базі Android, ігрові консолі тощо. Для розробки програми, яка використовує SQLite, потрібно перекинути файл з вихідним кодом sqlite3.c та його заголовний файл sqlite.h у власний проект та скомпілювати з його кодом програми.

З попереднього рисунку можна зробити висновок, що через безсерверну архітектуру не потрібно “встановлювати” SQLite перед його використанням. Немає серверних процесів, які потрібно налаштовувати, запускати чи припиняти. Окрім того, не використовуються ніякі файли конфігурації.

Усі транзакції повністю відповідають властивостям ACID. Це означає, що усі запити та зміни є атомарними, послідовними, ізольованими та довговічними. Іншими словами, усі зміни в рамках транзакції відбуваються повністю, або частково, навіть якщо виникає форс-мажор у вигляді збою програми, живлення або операційної системи.

SQLite використовує динамічні типи для таблиць, що надає змогу використовувати будь-яке значення в будь-якому стовпці, незалежно від типу даних.

Одиничне з’єднання з базою даних дозволяє отримати доступ до кількох файлів бази даних одночасно. Це приносить багато корисних функцій, як-от об’єднання таблиць у різних базах даних або копіювання даних між базами даних за допомогою однієї команди.

2.6. Windows – операційна система

Windows – це графічна операційна система, яка розроблена компанією Microsoft [16]. Надається дозвіл користувачам переглядати та зберігати файли, запускати програмне забезпечення, грати у відеоігри, підключатися до мережі Інтернет.

На сьогоднішній момент існує декілька версій даної операційної системи, у кваліфікаційній роботі буде використовуватися Windows 10 – одна з найбільш сучасних наявних версій. Вона була представлена 29 липня 2015 року. Випущена з деякими новими функціями, такими як: Microsoft Edge, Cortana, Action Center. Також, пропонуються різні видання даної версії, адаптовані під різноманітні типи користування:

- Home;
- Pro;
- Education;
- Enterprise;
- Mobile.

В даній роботі буде використовуватися саме Education версія, так як вона призначена для потреб учнів, студентів та адміністраторів навчальних комплексів.

Windows достатньо легко використовувати в порівнянні з іншими ОС, так як забезпечується простий інтерфейс користувача. При цьому час її установлення може займати більше часу. Також, за останні роки Windows покращила свою надійність та функції безпеки. Такі міри впроваджені оскільки Windows має величезну базу користувачів та вони та їх дані можуть

легко стати мішенню для зловмисних програмістів. Windows дозволяє використовувати командний рядок, але не в такій мірі, наскільки це надає така система як Linux. Потрібно пам'ятати, що Microsoft Windows без ліцензії не дозволяє змінювати програмне забезпечення, так як тоді відсутній доступ до вихідного коду. Користувачам надається онлайнова та інтегрована довідкова система, а також велика кількість інформативних книг, для того, щоб допомагати людям усіх рівнів кваліфікації. Врешті-решт, регулярно скачуються обновлення, які своєю наявністю надають представлення про те, що кібербезпека даної системи тільки підвищується з часом.

2.7. Android – мобільна операційна система

У даній роботі програма буде підтримувати часткову роботу на декількох операційних системах, включаючи Android. Android – це програмний пакет і операційна система на основі Linux для мобільних пристроїв, таких як планшетні комп’ютери та смартфони [17]. Розроблена початково компанією Google, а пізніше OHA – Open Handset Alliance. Мова Java в основному використовується для написання коду, хоча можуть експлуатуватися ще й інші мови програмування. Загальна мета проекту Android є створення успішного реального продукту, який покращить мобільний досвід для кінцевих користувачів.

Нижче наведено важливі функції даної системи:

– відкритий код;

– кожен може налаштувати платформу під власні потреби та вимоги;

– велике різноманіття мобільних додатків, а також девайсів, які можуть вибрати споживачі;

– надається багато цікавих функцій, як-от відомості про погоду, початковий екран, живі канали RSS.

Окрім того, забезпечується підтримка служб обміну повідомленнями SMS, веб-браузер, сховище SQLite, підключення по Bluetooth, Wi-Fi тощо.

Додаток в даній роботі є лише одним з масивної величини всіх додатків на ринку. Найпопулярніші категорії:

– розваги;
– спілкування;
– музика та аудіо;
– медіа та відео;
– подорожі;
– продуктивність;
– персоналізація.

У поточній роботі буде використовуватись версія Android 12 або Android Snow Cone – дванадцята основна версія мобільної операційної системи, яка остаточно була випущена 4 жовтня 2021 року, що позначає її актуальність. Були реалізовані новації, які позитивно вплинули на оптимізацію системи. На даний момент, це третя по популярності версія Android на смартфонах.

Покращився користувацький інтерфейс, а саме відбулося оновлення матеріального дизайну операційної системи під назвою “Material You”, яка має більше видів кнопок та кількості анімацій. Додатково була опрацьована платформа, на якій підвищили продуктивність таких служб як WindowManager, PackageManager. Більш комплексна стала конфіденційність, а саме функції машинного навчання на рівні ОС стали ізольованими. Тепер програмам можна обмежити доступ лише до “приблизних” даних про місцезнаходження, а не до “точних". Останнє, але не менш важливе, до перемикачів швидких налаштувань додані елементи керування для вимкнення камери та мікрофона.

SDK (англ. Software Development Kit) – це набір засобів розробки програмного забезпечення в одному інстальованому пакеті. SDK на Android надає набір інструментів, необхідних для створення програм на платформі і надає гарантію плавного проходження процесу. Незалежно від мови програмування потрібен SDK, для того, щоб програму можна було запустити на будь-якому пристрої Android. Нині, Android SDK постачається разом із

Android Studio, інтегрованим середовищем розробки, де багато інструментів тепер краще керовані та доступні.

2.8. GitHub – веб-сервіс

GitHub – це служба розміщення репозиторіїв Git, яка надає графічний веб-інтерфейс.

Система управління версіями – програмне забезпечення для полегшеної роботи зі змінною інформацією. Система управління версіями дозволяє зберігати декілька версій одного і того ж документа, при необхідності повертатися до попередніх версій, визначати ким і коли було зроблено зміну у файлі та багато іншого.

Такі системи найбільш широко використовуються при розробці ПЗ для зберігання вихідних кодів програми, яка розробляється. Також, вони можуть з успіхом застосовуватися і в інших областях, в яких ведеться робота з великою кількістю електронних документів, які безперервно змінюються.

Основна ідея – кожен розробник має власний репозиторій – місце, де зберігаються і підтримуються будь-які дані, найчастіше у вигляді файлів, куди складаються версії файлів і відбувається синхронізація між розробниками (рис. 2.3).

Рис. 2.3. Процес роботи системи контролю версій

Наведемо основні команди для роботи в системі управління версіями Git у таблиці 2.2.

Таблиця 2.2.

Основні команди Git

Назва команди	Значення команди
git clone	Клонування репозиторію на локальний комп’ютер
git push	Відправка змін у віддалений репозиторій
git pull	Отримання змін з віддаленого репозиторію
git commit	Запис змін в локальний репозиторій, створення нової версії
git add	Добавлення файлу в локальний репозиторій

Завантаження власного проекту на GitHub можна спостерігати на рис. 2.4:

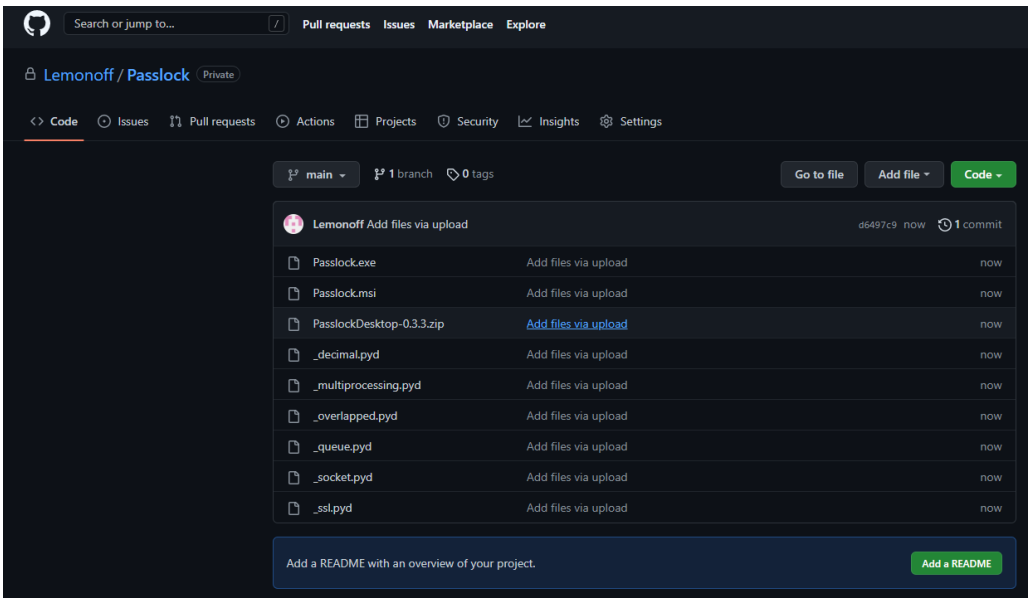


Рис. 2.4. Веб-сайт для розміщення git-репозиторіїв

GitHub стає все більш популярним програмним ресурсом, який використовується для обміну кодом. Це сайт соціальної мережі для програмістів, який багато компаній і організацій використовують для полегшення управління проектами та співпраці.

Завантажені проекти можна завантажувати, а також використовувати у своїх цілях для покращення власного коду програми, так і розробки нового на основі цього ж. Користувач може налаштувати політику приватності та загружати файли у приватний репозиторій, який не буде нікому доступний окрім самого власника проекту.

ВИСНОВКИ ДО РОЗДІЛУ 2

Вибір таких технологій як мова програмування, середовище розробки, графічні бібліотеки інтерфейсів, фреймворки, бази даних та підтримувані операційні системи є достатньо вагомим кроком для визначення сфери, де та як буде розроблятися програмне забезпечення. Наявні існуючі технології можуть полегшити майбутню розробку та сприяти створенню інноваційних рішень в IT-сфері.

Перед початком проектування та реалізації потрібно досконало вивчити та вміти використовувати весь інструментарій, який буде задіяний протягом розробки самого додатку. Це допоможе розробнику бути більш ефективним та сприяє витраті часу у мінімальному масштабі.

РОЗДІЛ 3

ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ДОДАТКА

3.1. Діаграма композитної структури

Діаграма композитної структури є частиною UML – уніфікованої мови моделювання [18]. Вона показує структуру класу через низку взаємодій між елементами на діаграмі. Розробники програмного забезпечення використовують ці діаграми для зображення взаємодії між сутностями та всередині них.

Випадки використання композитної діаграми:

– показ внутрішніх елементів класу;

– виявлення динамічного зв’язку між різними класами та елементами, що дає перевагу над діаграмами класу;

Відповідну діаграму внутрішньої структури даної кваліфікаційної роботи можна спостерігати на рис. 3.1:

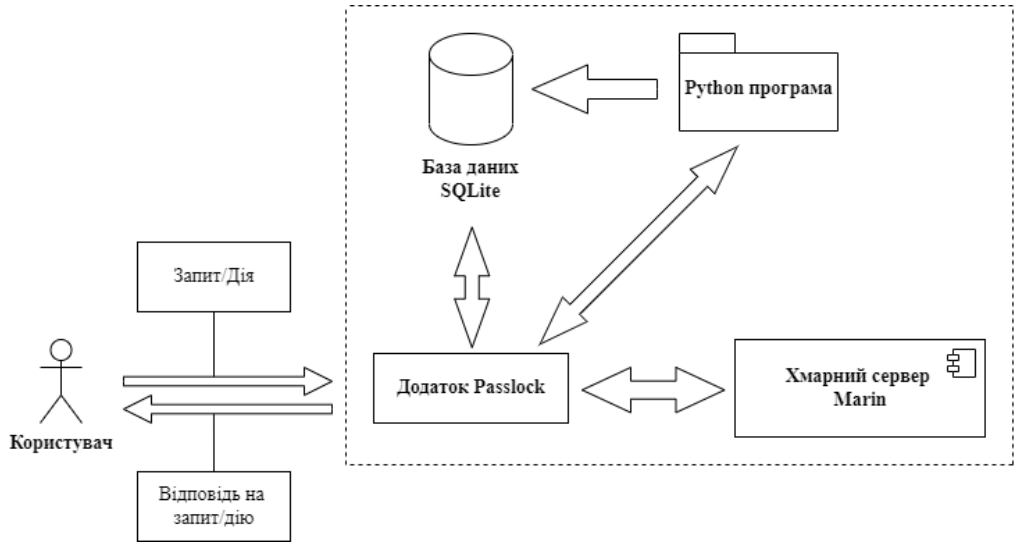


Рис. 3.1. Діаграма композитної структури

Система працює таким чином. Користувач початково взаємодіє з графічною оболонкою додатку Passlock. Будь-яка його дія, запит обробляється програмним, а згодом і машинним кодом, який написаний на мові програмування Python. При наданні конфіденційних даних, всі вони записуються у фізичну базу даних під назвою SQLite, яка також підключена програмно до додатку. Окрім того, всі дані можуть бути збережені у резервній копії у вигляду файлу, або записаними на хмарний сервер під назвою Marin. В будь-який момент часу, дану синхронізацію можна використати у цілях отримання конфіденційних даних у разі втрати або видалення їх через додаток. Це можливо буде зробити як через десктоп, так і через мобільний девайс, що позначає актуальність та узгодженість даної діаграми з різними типами девайсів.

3.2. Діаграма варіантів використання

Для моделювання системи найважливішим аспектом є фіксація динамічної поведінки. Динамічна поведінка означає поведінку системи під час її роботи. В UML доступно п’ять діаграм для моделювання динамічної природи, і діаграма варіантів використання є однією з них. Відповідно, в такому випадку повинні існувати деякі внутрішні та зовнішні фактори для здійснення взаємодії. Ці внутрішні та зовнішні агенти відомі як актори. Діаграма варіантів використання складається з акторів, варіантів використання та їхніх зв’язків.

В загальному використовується для зібрання вимог до системи, включаючи внутрішні та зовнішні впливи. Ідентифікуються учасники та варіанти використання для аналізу системи, а саме її функціональних можливостей.

Коротко можна навести, що цілі даної діаграми такі:

– збір вимог до системи;
– огляд системи ззовні;

– визначити зовнішні та внутрішні фактори, які впливають на систему;
– представити взаємодію між акторами.

Діаграму варіантів використання програми, яка описується в даній роботі можна спостерігати в Додатку А.

Наявні 3 види акторів: незареєстрований користувач, зареєстрований користувач та адміністратор. У даному випадку можна спостерігати відношення узагальнення, коли предком є саме адміністратор, а його нащадками відповідні користувачі. Нащадки успадковують властивості та атрибути предків, а також можуть наділятися своїми власними властивостями поведінки.

Для початку, незареєстрованому користувачу доступна тільки одна асоціація, а саме реєстрація власного акаунта. Без цього йому не доступний ніякий функціонал додатку.

В адміністратора функціонал власний полягає у взаємодії з БД, а саме видалення/модифікування/добавлення даних. Також, дана роль може розглядати деякі конкретні запити від рядових користувачів, а саме: прийняття запиту на доступ до бази даних, та підвищення користувача до ролі адміністратора. Перше ініціюється самим користувачем, а другий випадок може бути введений в дію тільки через бажання самого адміністратора. Інші його функції полягають у введенні/видаленні/модифікуванні вже наявного функціоналу.

Тепер розглянемо взаємодію звичайного зареєстрованого користувача з системою. Очевидно, що за допомогою своїх даних як пошта та пароль, він/вона може ввійти у додаток та користуватися його функціоналом. Є 2 способи створити пароль: записати його вручну, або використати комплексний генератор паролів, який можна конфігурувати під свої потреби. При ручному способі, одразу надається поле для записування назви або тегу, до чого прикріплюється сам пароль. Існує альтернатива, а саме запис паролю вже по існуючим категоріям в додатку. Любий з видів запису можна додати в обрані для швидкого доступу. Присутній пошук тегу, куди під’єднується

такий варіант зв’язку як розширення. Дане відношення показує, що прецедент використання розширює базову послідовність дій і вставляється власна послідовність, яка виконується при певних умовах. Тобто через пошук тегу паролю, можна як видалити сам тег з паролем, так і оновити його. Можливе просте копіювання самого паролю. Дана поведінка вже залежить від намірів самого користувача. Стосовно резервного копіювання конфіденційних даних є також 2 способи: ввімкнення синхронізації та резервне копіювання у файл, який буде зберігатися на диску комп’ютера. У першому випадку можна спостерігати таке відношення як включення, де незалежний варіант використання завжди буде включати в себе поведінку залежного варіанта використання. Ввімкнення синхронізації по стандарту включає в себе функції як копіювання даних у хмару, так і відновлення їх через неї. З критично важливих особливостей, користувач може заблокувати додаток, тобто тоді прийдеться ввести знову тільки пароль, для того, щоб обмежити доступ до потенційних зловмисників у реальному часі.

Останній функціонал має більш візуальний напрямок, а саме налаштування розміру шрифту та зміни на денний/нічний режим, при цьому маючи можливість вибрати кольорову гамму на свій вибір. Додаток повинен не тільки виділятися своїм функціоналом, а й гнучкістю до налаштування візуалу під смаки користувачів, що може у свою чергу охопити більшу аудиторію.

3.3. Інфологічна модель бази даних

Концептуальний дизайн – це побудова семантичної моделі, тобто інформаційної моделі найвищого рівня абстракції. Дана модель створюється без націлювання на конкретну СУБД та модель даних. Такі терміни як “семантична модель”, “інфологічна модель”, “концептуальна модель” є синонімами. Крім того, у цьому контексті слова “концептуальна модель бази даних” і “концептуальна модель предметної області” можуть вживатися

однаково, оскільки така модель є одночасно і образом реальності, і проектованою базою даних для цієї реальності.

Найчастіше концептуальна модель бази даних включає:

– опис інформаційних об'єктів або понять предметної області та зв’язків між ними;

– опис обмежень цілісності, тобто вимог до дійсних значень даних і зв’язків між ними.

Вид і зміст концептуальної моделі бази даних визначається обраним для цього формальним апаратом. Частіше всього використовуються графічні ER-діаграми, в даному випадку – нотація Чена.

Основні елементи:

– сутності;

– атрибути сутностей;

– зв’язок між сутностями.

Проаналізовані та сформовані сутності, їх атрибути наведені в таблиці 3.1:

Таблиця 3.1.

Сутності та атрибути концептуальної моделі

Сутність	Атрибути
Користувач	Логін, пароль, статус
Менеджер паролів	Назва, алгоритм шифрування, версія, опис
Конфіденційні дані	Назва, категорія, статистика

Присутні зв’язки між користувачем та конфіденційними даними: створювати, модифікувати, видаляти, резервно копіювати.

Присутні зв’язки між конфіденційними даними та менеджером паролів: зберігати дані, шифрувати дані, синхронізувати дані.

Сформована інфологічна модель представлена на рис. 3.2.

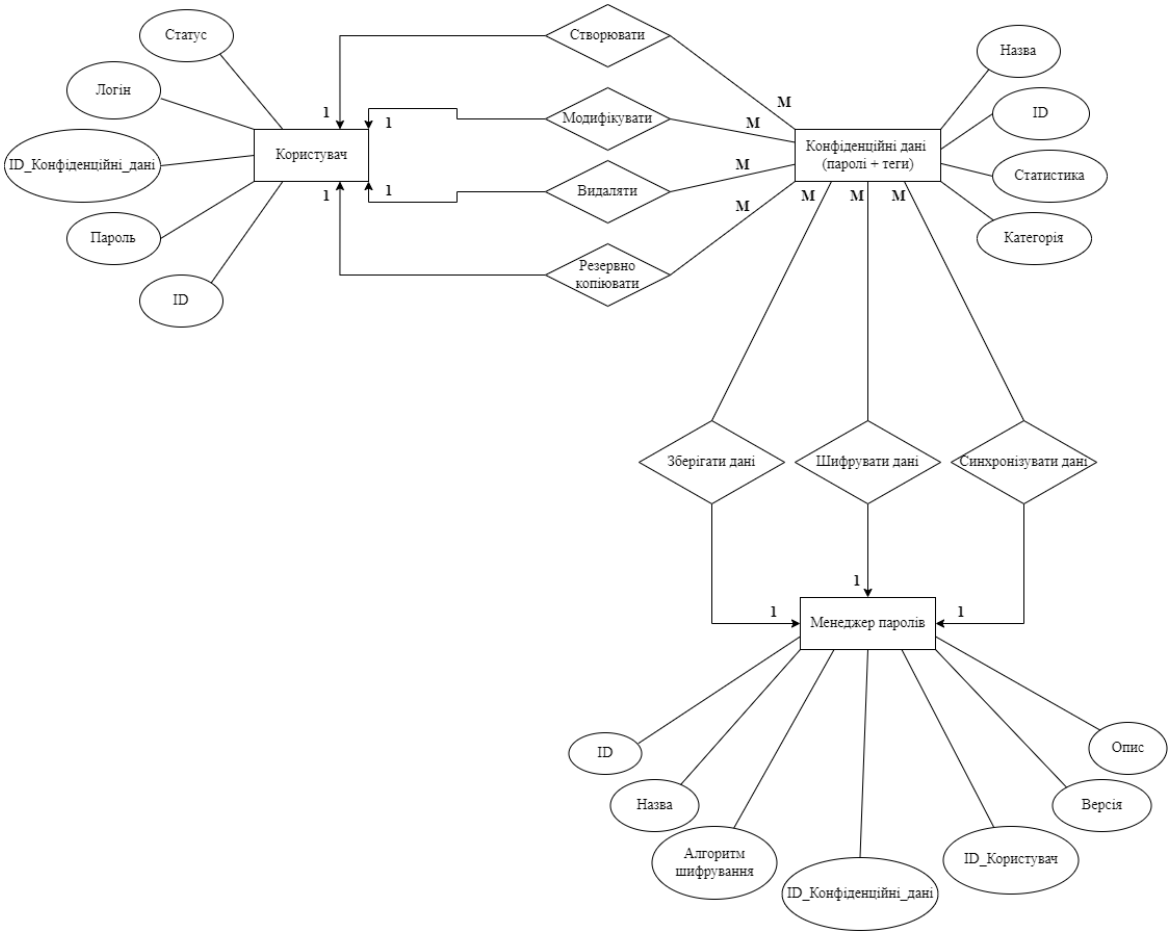


Рис. 3.2. Інфологічна модель бази даних

Присутній зв’язок 1:M, що означає виконання процесу одним екземпляром сутності над декількома екземплярами іншої сутності. Наприклад, один користувач може зберігати у великої кількості свої конфіденційні дані. В цьому ж випадку, множина конфіденційних даних можуть належати одному користувачу.

Обмеження цілісності не порушено так як:

– для кожної сутності вказаний власний ідентифікатор;

– збереження в БД всієї необхідної інформації без її зайвого дублювання;

– модель логічно не протиречить в рамках зв’язків між сутностями та її екземплярами.

3.4. Діаграма класів

Діаграма класів зображує статичний вигляд програми [19]. Вона представляє типи об’єктів, які знаходяться в системі і зв’язки між ними. Клас, в свою чергу, складається з цих об’єктів, а також вони можуть успадковуватися від інших класів. Діаграма класів використовується для візуалізації, опису, документування різних аспектів системи, а також для створення виконуваного програмного коду. Показуються класи, атрибути, функції та зв’язки, щоб дати загальне уявлення про програмну систему.

Основна мета – подача статичного вигляду програми. Це єдина діаграма, яка широко використовується для побудови, і її можна відобразити за допомогою об’єктно-орієнтованих мов. Це одна з найпопулярніших діаграм UML.

Призначення діаграми класів:

– аналізування та проектування статичного вигляду програми;

– опис основних обов’язків системи;

– формування основи для створення діаграм компонентів та розгортання;

– включення прямого та зворотнього інжинірингу.

Переваги діаграми класів:

– може представляти модель об'єкта для складних систем;

– скорочується час обслуговування, надаючи огляд того, як програма структурована перед кодуванням;

– надається загальна схема програми для кращого розуміння;

– являє собою детальну діаграму з виділенням потрібного коду, який потрібно запрограмувати;

– корисно для зацікавлених сторін і розробників.

Відповідна діаграма класів додатку Passlock наведена на рис. 3.3:

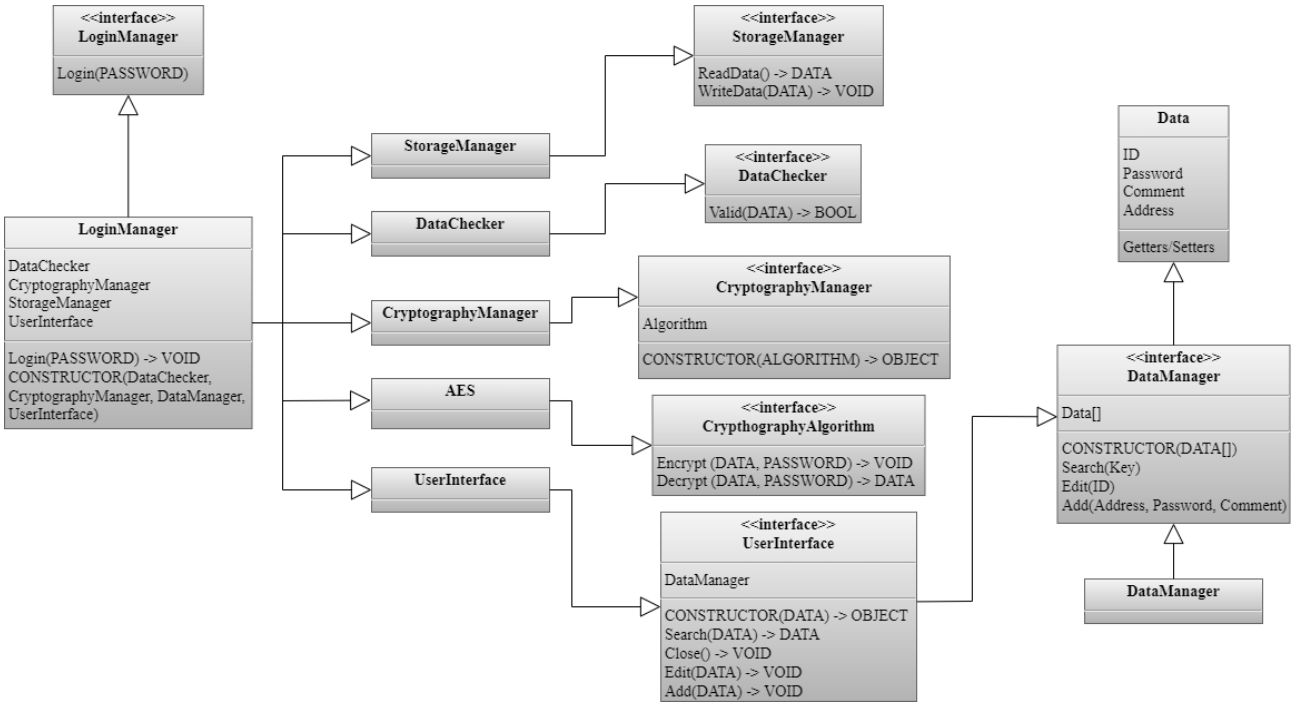


Рис. 3.3. Діаграма класів

LoginManager – цей клас відповідатиме за автентифікацію користувача, оскільки дані на жорсткому диску будуть зашифровані, це вдасться, лише якщо дані будуть успішно розшифровані, перший рядок файлу даних буде DECRYPTED, так що програма може зрозуміти, що дані успішно розшифровано, і дозволити користувачеві перейти до інтерфейсу користувача.

Залежності:

– DataChecker;
– CryptographyManager;
– StorageManager;
– UserInterface.

Інтерфейс користувача буде відповідати за редагування, пошук, перегляд і додавання даних, а саме облікових записів. Наявна залежність – це DataManager. Відповідальність цього класу полягає в управлінні даними, він міститиме дерево або хеш-таблицю даних. Повинен мати можливість пошуку, редагування та додавання нових даних до структури даних. Залежність DataManager – Data. Цей клас містить дані, які розділені на поля – ID,

обліковий запис, коментар, пароль. Він повинен забезпечити належну інкапсуляцію.

DataChecker – відповідальність цього класу полягає в перевірці того, що перший рядок даного файлу DECRYPTED, інакше кажучи, щоб перевірити, чи дані успішно розшифровані та зашифровані (перед закриттям програми файл повинен бути зашифрований).

StorageManager – цей клас відповідає за читання і запис даних з диску.

CryptographyManager – цей клас відповідає за зберігання алгоритму криптографії [20]. Залежність – CryptographyAlgorithm. Це інтерфейс для різних алгоритмів криптографії, кожен клас повинен реалізовувати методи для шифрування та дешифрування.

ВИСНОВКИ ДО РОЗДІЛУ 3

Правильне початкове проектування – це заклад успішної роботи додатку в майбутньому. Діаграма композитної структури надає поняття, які елементи задіяні в середовищі загалом, та які між ними взаємозв’язки. Діаграма варіантів використання, у свою чергу, більше надає поняття про види акторів та їх можливості, які будуть вже безпосередньо використовувати систему та яким чином. Інфологічна модель надає опис інформаційних об’єктів та понять предметної області, а також зв’язків між ними.

Діаграма класів вже відноситься більше до реалізації, а саме представляє статичний вигляд самої програми. Розробка класів та інтерфейсів дозволить коректно визначити обов’язки кожного з них та запобіжить порушенню основних принципів SOLID об’єктно-орієнтованого програмування.

РОЗДІЛ 4
ЗАСТОСУВАННЯ ДОДАТКА

4.1. Тестування системи

Для початку потрібно встановити додаток на ОС Windows 10 – через файл Passlock.msi, який містить інсталяційний процес. На даному етапі закладено декілька можливостей у вигляді:

– модифікування, де можна змінити як саме будуть встановлюватися функція та програма в цілому;

– ремонт, де виправляються помилки при останній установці, наприклад, виправлення багу зникнення або пошкодження файлів, ярликів;

– видалення програми з комп'ютера.

Також, для Android системи існує файл з розширенням apk.

При запуску додатку користувача зустрічає головне меню, рис. 4.1:

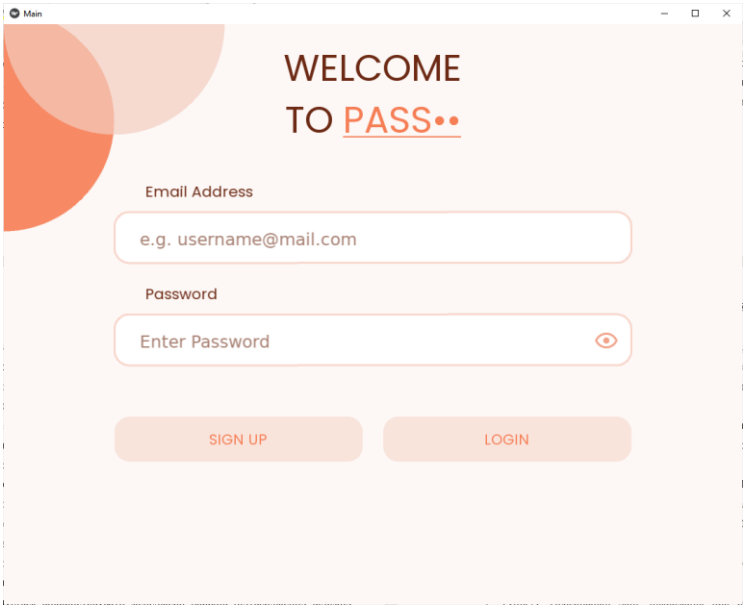


Рис. 4.1. Головне меню додатку Passlock

Дане вікно дозволяю користувачу зареєструватися (кнопка “Sign Up”), або ввійти у вже існуючий акаунт. Потрібно ввести пошту, пароль та натиснути кнопку “Login”. Пароль може бути як прихований від посторонніх очей, так і навпаки. Важливо відмітити, що потрібно ввести саме адресу електронної пошти, так як закладена перевірка на правильність введення даних у поле і тому при помилці внизу видається сповіщення, яке зображене на рис. 4.2:

Рис. 4.2. Видача помилки при неправильному введенні електронної пошти

Також є ліміт, згідно якому, пароль до даного акаунту повинен бути не менше 6-ти символів, в іншому випадку видасть попередження, яке зображене на рис. 4.3:

Рис. 4.3. Попередження про створення слабкого паролю

Якщо користувач вже є у системі, але замість входу, він виконує реєстрацію з тією самою поштою буде виводитися наступне повідомлення як на рис. 4.4:

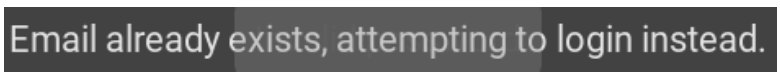


Рис. 4.4. Попередження у випадку реєстрації користувача, якщо пошта вже зареєстрована в системі

Якщо користувач намагається ввійти в акаунт багато разів безуспішно, буде надане таке повідомлення як на рис. 4.5:

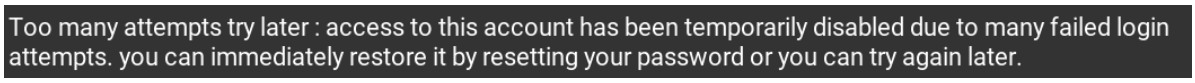


Рис. 4.5. Помилка після багатьох невдалих спроб автентифікації

Після успішної реєстрації акаунту, на пошту, яка була вказана, прийде наступне повідомлення (рис. 4.6):

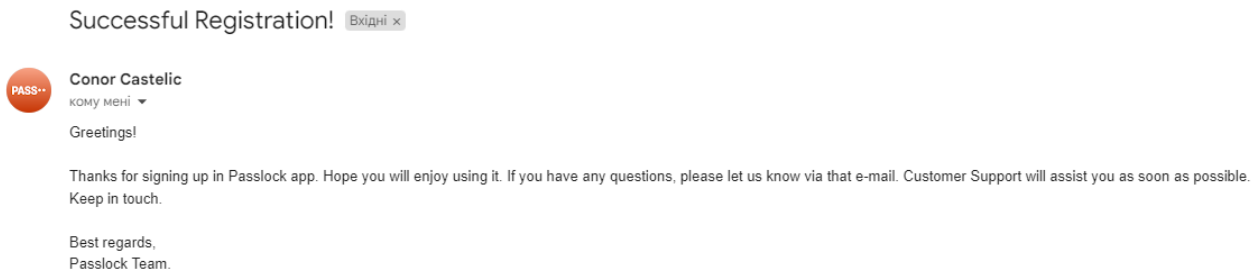


Рис. 4.6. Повідомлення на пошту від додатку користувачу при успішній реєстрації

При входженні в систему абоненту представляється таке вікно як на рис. 4.7:

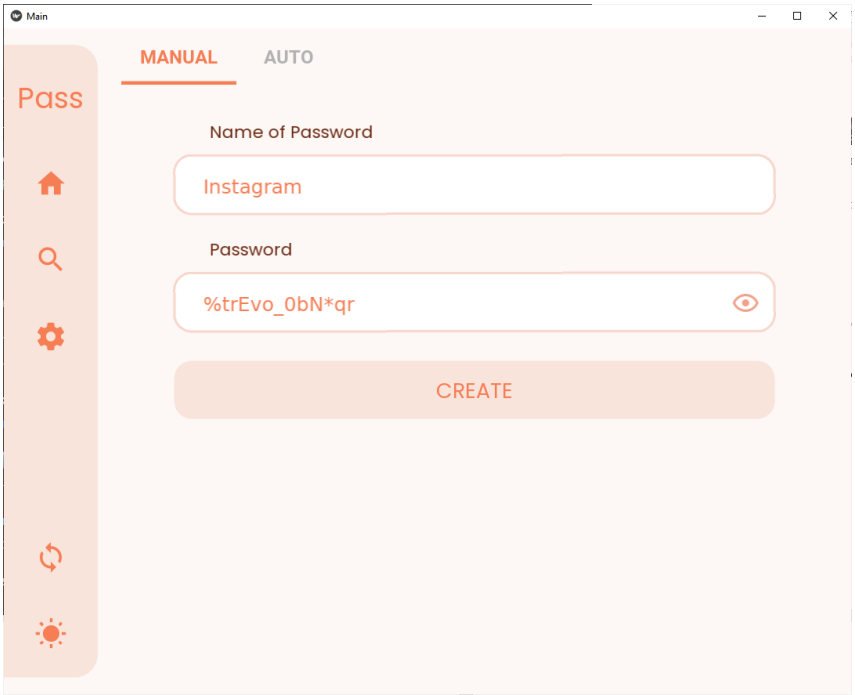


Рис. 4.7. Вікно програми, де надається інтерфейс та функціонал ручного запису паролю

Присутні 2 способи створення паролю: вручну та за допомогою генератора. На рисунку вище вже введені дані, а саме назва соціальної мережі та пароль від акаунта. При натисненні кнопки “Create” видасть повідомлення з контекстом, що пароль успішно записався. Якщо ж дані повторно записати, система сповістить, що даний пароль вже існує в БД.

При перехід на автоматичну генерацію паролю можна спостерігати вікно як на рис. 4.8:

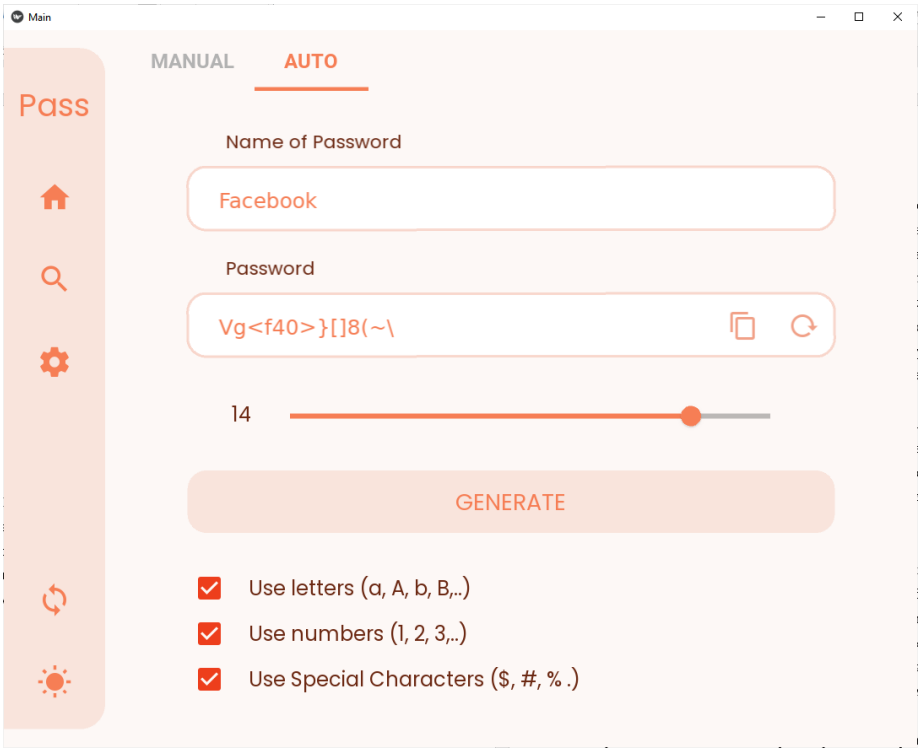


Рис. 4.8. Автоматичний генератор паролю

У даному генераторі абонент вводить назву категорії та вибирає налаштування, як буде генеруватися сам пароль. Можна вибрати кількість символів як буде присутня за допомогою слайдера. Також є чекбокси на використання літер, цифр та символів. Звісно, при одних і тих самих налаштуваннях, можна проводити нову генерацію поки парольна фраза не підійде користувачу. Можна скопіювати даний пароль про всяк випадок, так як велика ймовірність, що він не буде читабельним, зате буде безпечним і складним для зламування. Після натиснення кнопки “Generate” тег та пароль записується у БД.

Можливе також не тільки власне введення тегу, а й вибір серед системних опцій, тобто наданий список категорій куди саме буде записуватися тег та пароль як на рис. 4.9:

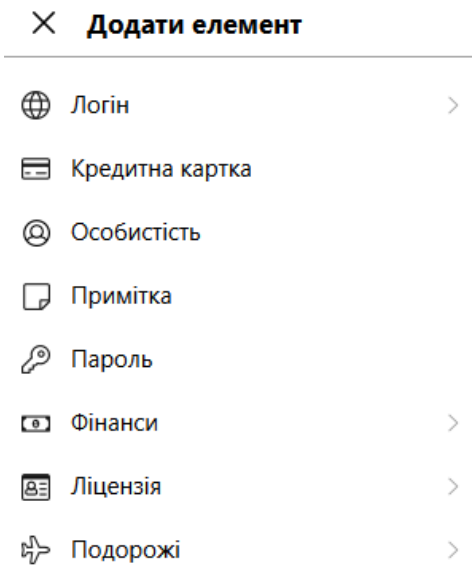


Рис. 4.9. Список категорій куди можуть записатись конфіденційні дані користувача

При натисненні знаку пошуку в лівому вертикальному меню, користувачу представляється вікно як на рис. 4.10:

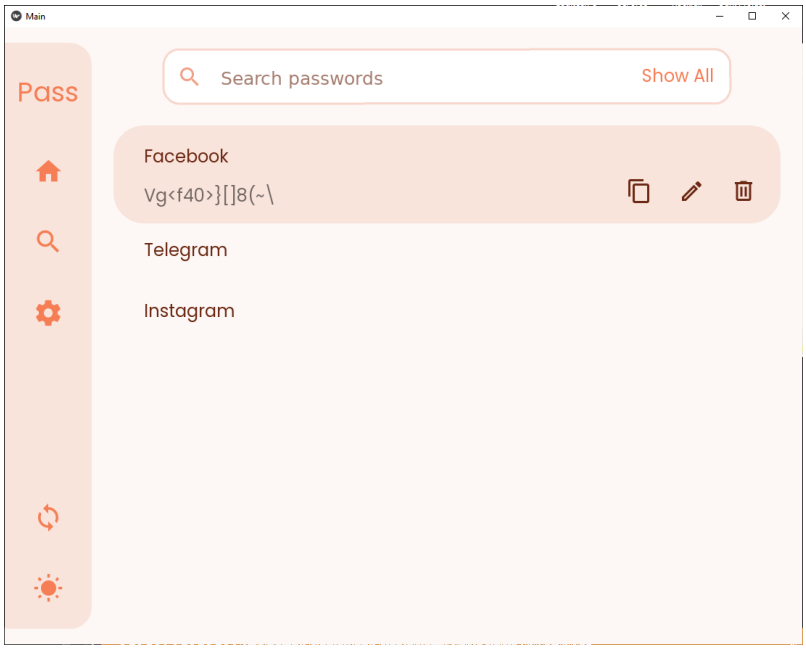


Рис. 4.10. Меню пошуку

Можна використати пошук по назві тегу, або натиснути кнопку “Show All” – тоді висвітляться всі записані користувачем дані в БД. На вищенаведеному рисунку спостерігається, що є декілька опцій, а саме: копіювання паролю, зміна даних та видалення їх. При модифікуванні виникає таке вікно як на рис. 4.11:

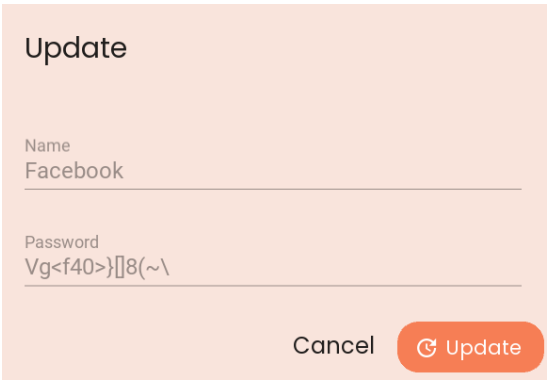


Рис. 4.11. Вікно зміни даних

При випадковому видаленні даних є можливість швидко натиснути кнопку “UNDO” в нижньому сповіщенні від програми, для того, щоб запобігти цьому.

При натисненні шестерні, користувач потрапляє до налаштувань та контактних даних (рис. 4.12):

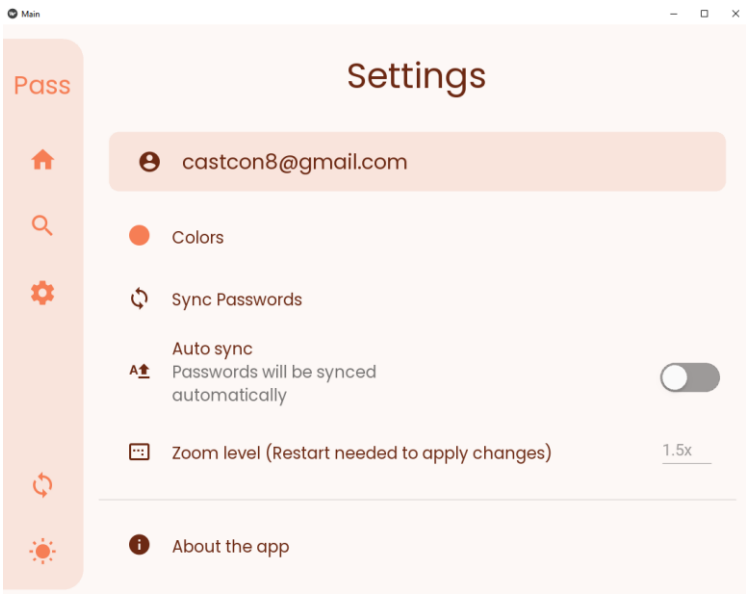


Рис. 4.12. Вікно налаштувань

Тут відображається пошта користувача, кольорова гамма, яку можна налаштовувати. Для прикладу змінення кольорового спектру додатку на рис. 4.13:

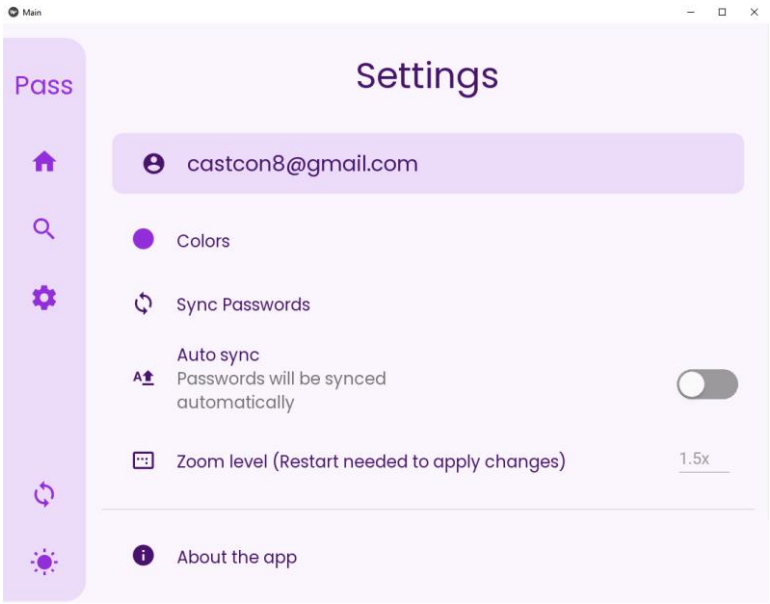


Рис. 4.13. Альтернативна кольорова гамма додатку

Також доступна ще версія у синьому відтінку. Окрім цього, на лівій панелі знизу користувач може вибирати денний або нічний режим роботи програми. Нічний режим представлений на рис. 4.14:

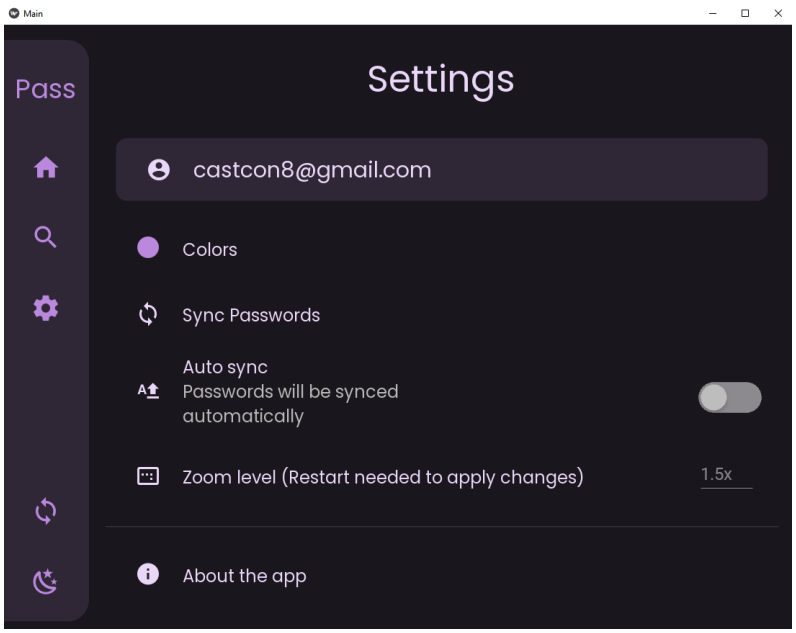


Рис. 4.14. Нічний режим роботи програми

Останнє, що може мінятися з точки візуального дизайну – це розмір масштабу самого додатку. На даний момент цей параметр стоїть у шкалі 1.5x. При оновленні значенні масштабу потрібно перезапустити програму для виконання зміни.

Можна дізнатися про контактні дані натиснувши “About the app”, зображено на рис. 4.15:

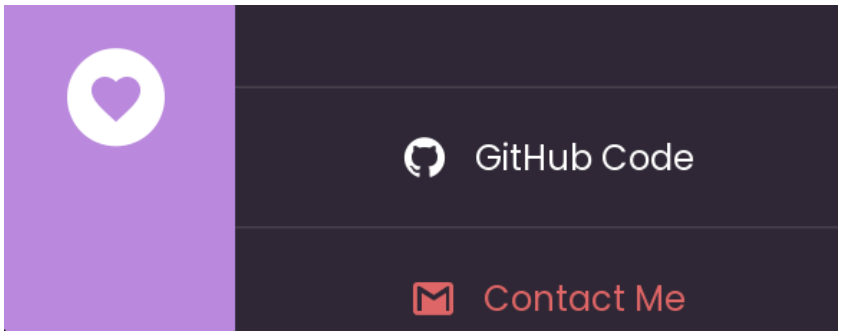


Рис. 4.15. Контактні дані при натисненні кнопки "About the app"

Можна потрапити на GitHub, де залитий даний проєкт (див. рис. 2.4). Крім того, навести контакт з поштою, яка зв’язується по e-mail при реєстрації в системі нового користувача.

У додатку присутні 2 види резервного копіювання: файл та хмарний сервіс. Натиснувши на стрілки в лівому вертикальному меню, можна зберегти файл з розширенням .passlock зі всіма конфіденційними даними на диск комп’ютера. При втраті акаунту, але з наявністю цього файлу, можна відновити всі паролі та теги користувача. При натисненні “Sync Passwords” виникає таке меню як на рис. 4.16:

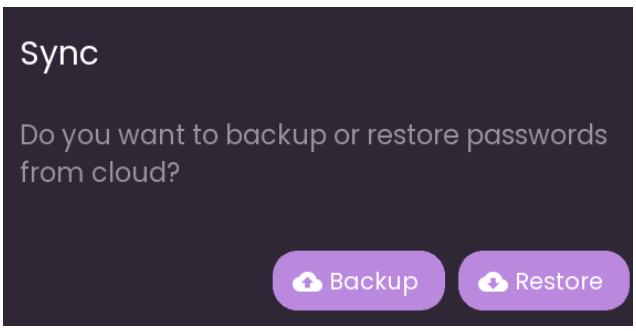


Рис. 4.16. Меню синхронізації з хмарним сервером

У даній роботі використовується хмарний сервер Marin від платформи OpenStack. Наявна можливість як завантажити дані у хмару, так і з неї. Доступна опція автоматичної синхронізації, для того, щоб користувач не зважав кожен раз на те, що йому потрібно зберегти нові оновлення в хмару, це зробиться автоматично на фоні.

При не остаточному виході, відсутності користувача в додатку протягом 5-ти хвилин та більше, система буде просити ввести тільки пароль від акаунта як на рис. 4.17:

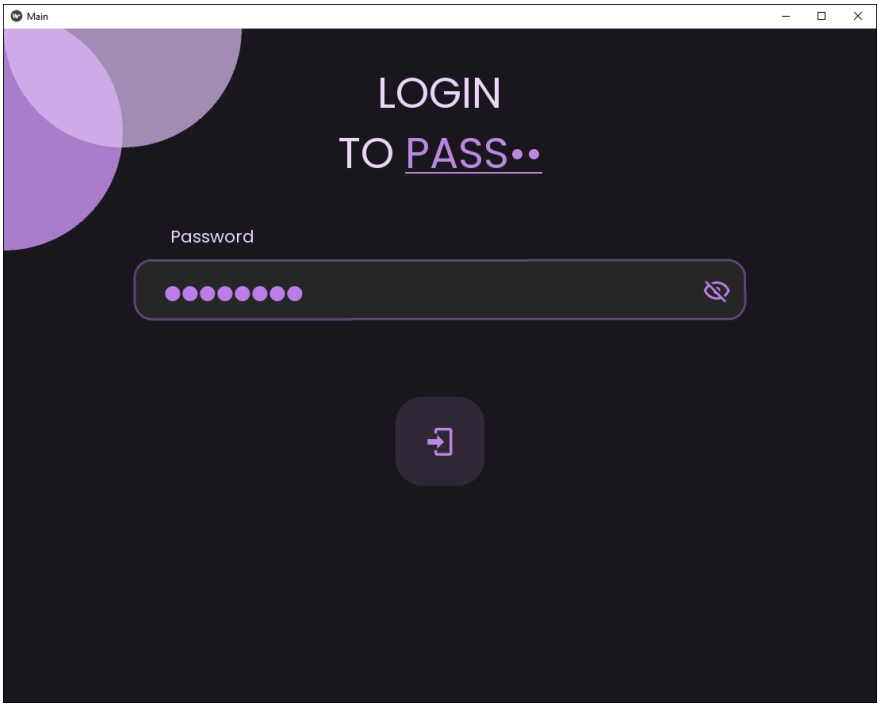


Рис. 4.17. Форма для введення тільки паролю

Проведемо тест-кейс, де всі дані записані на даному акаунті будуть збережені в хмару вручну, та потрібно їх відновити на цьому ж акаунті, тільки на іншому новому девайсі – на телефоні.

Входимо в додаток на телефоні, рис. 4.18:

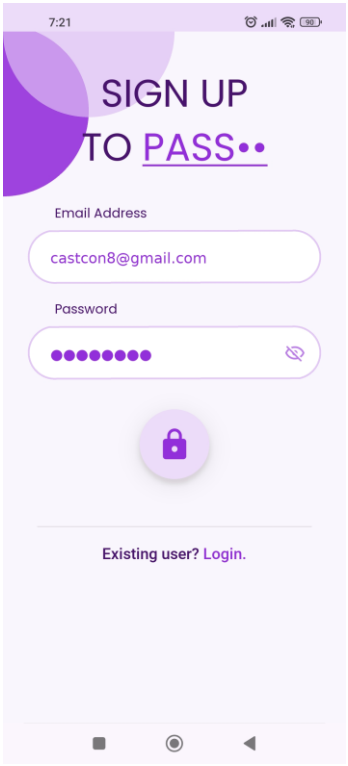


Рис. 4.18. Вхід в акаунт на мобільному девайсі

Після натиснення іконки стрілок та процесу завантаження, всі конфіденційні дані користувача підтягнулися на мобільну версію додатку, як зображено на рис. 4.19:

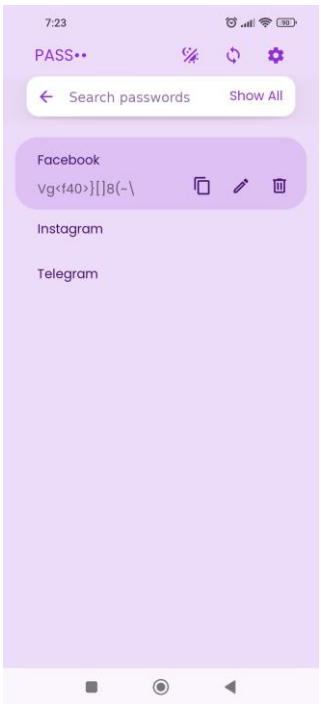


Рис. 4.19. Успішне відновлення конфіденційних даних користувача на мобільній версії додатку

Спостерігаємо, що всі теги та відповідні паролі правильно підтягнулися в акаунт. Відповідно, хмарне резервне копіювання працює належним чином.

4.2. Перспективи розвитку

Кожен застосунок, додаток повинен покращуватися та розвиватися з часом. Постійне оновлення продукту, як з програмної, так і з зовнішньої сторони представляє собою його працездатність та підтримування розробниками.

Варто відмітити, що будь-які нововведення у сфері захисту даних можуть як покращити безпеку, так і її спростувати. У додатку Passlock використовується алгоритм шифрування AES зі 128-бітним ключем. На даний момент він є досить ефективним, але у провідних менеджерах паролів як Bitwarden, Keeper, 1Password вже використовується 256-бітна форма даного шифрування. Так як потенційні загрози з часом тільки виростають, потрібно покращувати кібербезпеку для збереження критичної конфіденційної інформації.

Для розширення використання додатку потрібно реалізувати його інтеграцію у веб-браузери. Для початку містити функцію авторизування веб-браузерів за допомогою цифрових підписів. Тобто, з’єднання буде надане, тільки коли цифрові підписи будуть звіреними. Окрім того, додаток робитиме запит автентифікації кожного разу, коли веб-браузер встановлює нове з’єднання з розширенням. Врешті-решт, містити функцію автозаповнення даних, яка може вмикатися та вимикатися.

Потрібно надавати користувачем не просто фіксоване хмарне сховище, як в даному випадку, але й вибір серед списку сховищ, де можуть міститися OneDrive, Google Drive, Dropbox, NextCloud тощо. Звичайно, надавати об’єм безкоштовного простору мінімум від 1 Гб. Це надасть можливість зручно користуватися додатком тим абонентам, в кого всі інші дані записані в одному з цих сховищ.

Важлива планова новація стосовно всіх даних, які записані у додатку Passlock – синхронізація за допомогою сервера через Wi-Fi. Тобто, відбувається синхронізація сховища на комп'ютері з іншими пристроями через локальну мережу. Можна буде це оформити через ручний спосіб, або через QR-код, який потрібно буде відсканувати з додатку на іншому пристрої.

Очевидно, що для охоплення більшої аудиторії, додаток повинен підтримуватися на різних операційних системах. На даний момент, реалізація продукту є тільки на Windows та Android. З часом, можливе поширення і на інші ОС, такі як: iOS, Linux, macOS.

Стосовно менш глобальних особливостей розробки, можливо добавити двохфакторну автентифікацію, для більш захищеного входу в систему. В той же час, при відсутності користувача в додатку більше 5-ти хвилин, система просить ввести тільки пароль для входу. Тут можна застосувати додатковий аналог, а саме використання умовного PIN-коду для того, щоб вхід, який складатиметься тільки з цифр, був більш простим процесом ідентифікації користувача.

Якщо у додатку знаходиться великий масив конфіденційної інформації, є причина ввести статистику самим додатком на кількість ідентичних паролів, слабких паролів або в загальному дані, в яких вийшов термін (для прикладу – реквізити банківської карти). Один з можливих варіантів реалізації статистики зображений на рис. 4.20:

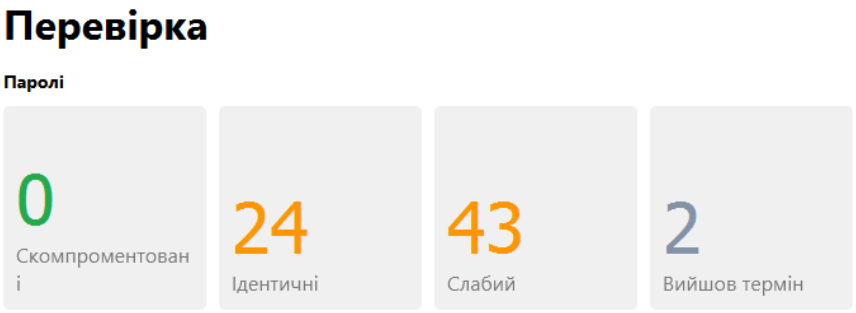


Рис. 4.20. Проектний графічний концепт статистики паролів та інших даних у додатку

Останнє, але не менш важливе, підтримка додатком різноманітних мов. Люди з різних країн та культур повинні мати можливість використовувати Passlock на рідній мові, а не тільки наявну англійську версію. Таким чином, популярність та поширеність додатку значно виросте в IT-сфері та й по всьому світу загалом.

ВИСНОВКИ ДО РОЗДІЛУ 4

У даному розділі відбулося детальне тестування системи, а саме перевірка її функціональності та коректного відображення графічного інтерфейсу. Загальна робота програмного забезпечення є успішною, підтримуються глобальні особливості як хмарне резервне копіювання через девайси різних типів та операційних систем.

Звичайно, має місце існувати постійне вдосконалення технічного продукту. Можливо імплементувати як локальні, так і глобальні рішення, деякі з них є свого роду інновацією у сфері кібербезпеки. Після їх реалізації, продукт може цілком виходити на ринок та змагатися з конкурентами за лідируючі позиції у ніші автономних систем збереження конфіденційних даних користувачів.

ВИСНОВКИ

Безліч різноманітних користувальницьких процесів виконуються з обов'язковим застосуванням персональних комп'ютерних пристроїв. Велика їх частина поступово переміщується в область віддаленої взаємодії, основним середовищем якої є глобальна інформаційна мережа Інтернет.

Можливості глобальної мережі повноцінно розкриває спеціалізоване програмне забезпечення, яке окрім безперешкодного доступу до ресурсів мережі пропонує безліч різноманітних корисних та зручних функціональних можливостей, у тому числі й способи дистанційного зберігання користувацьких конфіденційних паролів.

Звертаючи увагу на сучасні реалії, не багато користувачів здатні запам'ятати велику кількість паролів для входу на веб-сайти та сервіси, особливо якщо вони є технічно складними. Щоб зберегти всі персональні та конфіденційні дані в недоступному та віддаленому сховищу від злому, користувачам рекомендується використовувати менеджери паролів підтверджених відомих розробників.

Через об'ємний спектр вже готових реалізацій та рішень, досить важко придумати щось абсолютне нове та оригінальне. Якщо неможливо ввірватися у галузь з інновацією, потрібно розглянути вже існуючі варіанти, відокремивши головні переваги та недоліки. При власній розробці потрібно максимально покращити вже наявні плюси, виявити та ліквідувати присутні мінуси. Таким чином відбувається так звана еволюція, коли на основі попередніх напрацювань вдається реалізувати більш досконалий продукт в порівнянні з попередніми аналогами.

Не менш важливим є ретельний підхід до вибору мови програмування, середовища розробки, технологій та комплексних програмних рішень. Саме ці фактори визначають як зручність користування, так і компактність фінального продукту. Врешті-решт, накопичується практичний досвід, який можна буде використовувати в подальшому майбутньому.

Одними з найсуттєвіших аспектів є планування та детальний опис процесів роботи, функціональної складової для кращого розуміння проекту та точних поставлених цілей. Відповідне угрупування даних буде тільки сприяти коректності видання результатів. Чим далі продовжується вивчення та поглиблення, тим більше виникає нюансів, труднощів та перешкод.

З оптимальним підходом та аналізом причинно-наслідкового ланцюга, досить просто виявити умовні проломи як на логічному, так і на практичному рівнях. Модифікація впорядковування всіх наявних даних дозволить виявляти наступаючі проблеми завчасно, що надає само по собі часу та ресурсів, щоб підготуватися до них.

Через цю обґрунтовану причину проводяться різноманітні тести продукту заради виявлення будь-якої неточності, яку можна швидко помітити та виправити. Чим різноманітніші перевірки розробленого проекту, тим більше шансу пізнати багатогранність та різносторонність поглядів на розроблену працюючу програму.

На даний момент, людство недооцінює це програмне забезпечення. Но це не означає, що потреба в ньому відпала. Спроби злому та крадіжка конфіденційних даних, як звичайних людей, так і масштабних корпорацій зростають з кожним днем і в майбутньому будуть тільки збільшуватись. Компанії роблять усе можливе, щоб захистити сервери від атак зловмисників, але вразливе місце знаходиться саме на стороні самого користувача, який використовує прості та банальні паролі, які достатньо легко скомпрометувати. Тому, для того, щоб полегшити участь у запам’ятовуванні складних парольних фраз, рішенням є вищезгадані менеджери паролів.

Дана робота підкреслює важливість поширення обізнаності про це істотно важливе програмне забезпечення, щоб майбутнє особистої цифрової безпеки покращувалося.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Зубарський Д.О. Менеджери паролів / Погляд у майбутнє приладобудування : Збірник праць XIV Науково-практична конференція студентів, аспірантів та молодих вчених / 18-19 травня 2021 р. – К.:ПБФ, КПІ ім. Ігоря Сікорського. – 2021. - С. 38-41.

2. Качинський А. Б. Безпека, загрози і ризик: наукові концепції та математичні методи. Київ, 2004. 472 с.

3. Наскільки безпечними є сучасні менеджери паролів? [Електронний ресурс] – Режим доступу: https://hetmanrecovery.com/ru/recovery_news/how-safe-are-modern-password-managers.htm#plan_6 – Назва з екрану. – Дата звернення: 03.09.2022 р.

4. Why should we use password managers? [Електронний ресурс] – Режим доступу: https://www.cylab.cmu.edu/news/2022/01/27-password-managers. – Назва з екрану. – Дата звернення: 03.09.2022 р.

5. Тунік А. Персональні дані, інформація про особу, конфіденційна інформація про особу: аспекти співвідношення. Підприємництво, господарство і право. 2012. 5 (197). С. 50–54.

6. Кавун С. В. Інформаційна безпека : підручник. Харків : ХНЕУ, 2009. 368 с.

7. Яремчук Ю. Є. Криптографічні методи та засоби шифрування інформації на основі рекурентних послідовностей : Монографія. Вінниця : Кн.-Вега, 2002. 136 с.

8. Перевозчикова О. Л. Інформаційні системи і структури даних : навч. посіб. для студентів ВНЗ. Київ : Києво-Могил. акад., 2007. 288 с.

9. Single-factor, Two-factor, and Multi-factor Authentication [Електронний ресурс] – Режим доступу: https://www.pingidentity.com/resources – Назва з екрану. – Дата звернення: 08.09.2022 р.

10. What Is Data Encryption: Types, Algorithms, Techniques and Methods [Електронний доступ] – Режим доступу: https://www.simplilearn.com/data-encryption-methods-article – Назва з екрану. – Дата звернення: 19.09.2022 р.

11. Шапошникова С.Л. Основи програмування на Python : підручник. Початковий курс. - версія 2. - 2011. - 44 с.

12. Kornilovska N. V., Vyshemyrska S. V., Kolmykov M. O. DEVELOPMENT OF PYTHON ELECTRONIC MESSAGE INFORMATION PROTECTION SYSTEM USING THE PYCHARM WORKING AREA. Вісник Херсонського національного технічного університету. 2021. Т. 76, № 1. С. 106–112.

13. Roberto Ulloa. Kivy: Interactive Applications in Python / Roberto Ulloa. – Birmingham: Packt Publishing, 2013. – 122 с.

14. Філіпова Л. Я. Автоматизовані бази даних у світовому інформаційному просторі. Вісник Харківської державної академії культури. 2009. Вип. 1. С. 144–152.

15. Федулова С. О. Інформаційна безпека та захист інтелектуальної власності на бази даних. Економічний вісник ДВНЗ "Український державний хіміко-технологічний університет". 2016. № 2 (4). С. 189–193.

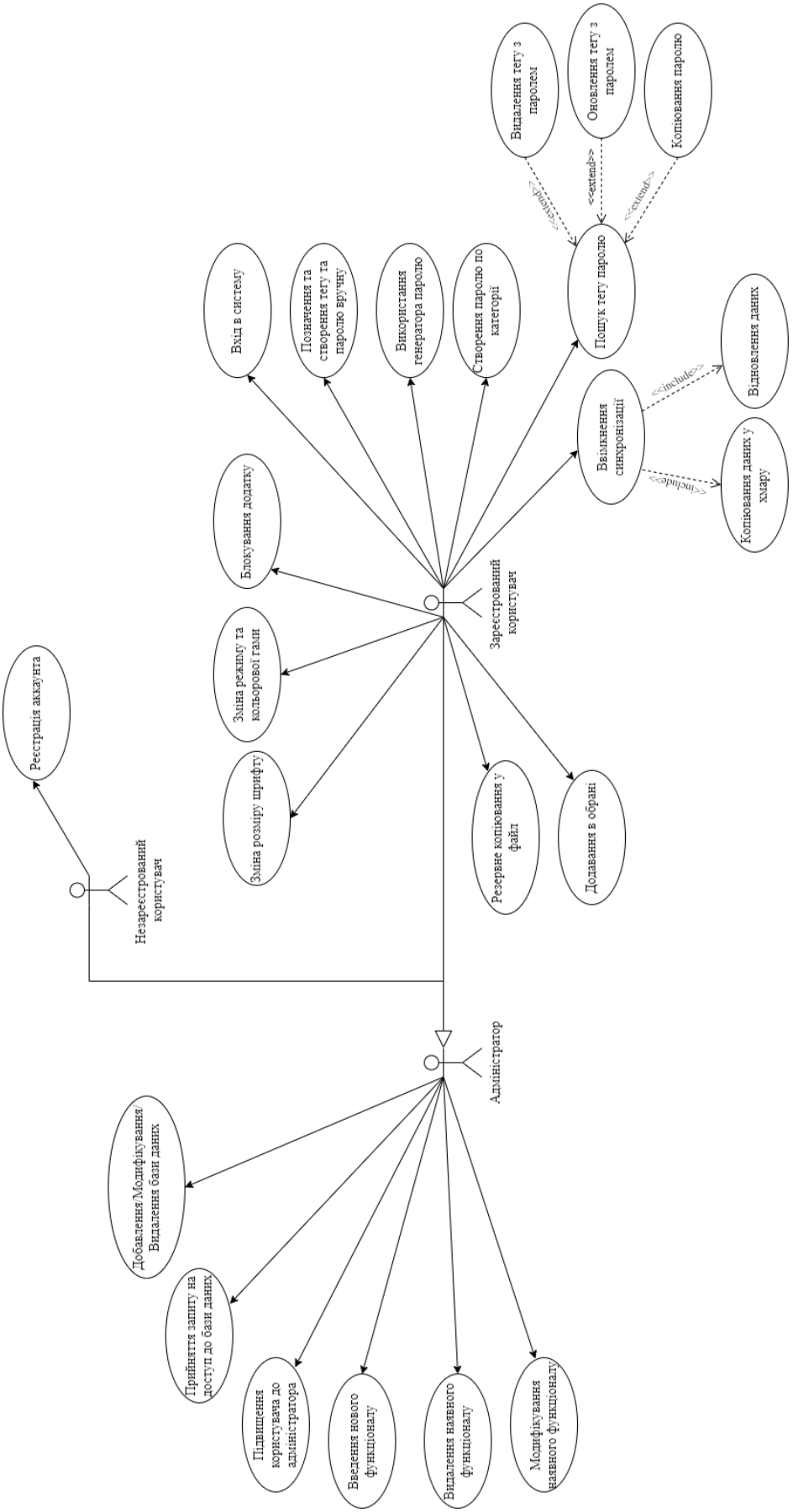
16. Трихлєб А. С., Демішонкова С. А. Інформаційна комп'ютерна система контролю та управління : Thesis. - 2020. - 57 с.

17. Android Operating System (OS): Definition and How It Works [Електронний ресурс] – Режим доступу: https://www.investopedia.com/android-operating-system.asp – Назва з екрану. – Дата звернення: 15.10.2022 р.

18. Гуляєв К. Д. Уніфікована система адресації мереж зв’язку : автореф. дис. канд. техн. наук. Одеса, 2010. 20 с.

19. What is Class Diagram? [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/ – Назва з екрану. – Дата звернення: 04.11.2022 р.

ДОДАТОК А. ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



ДОДАТОК Б. ЛІСТИНГ ПРОГРАМИ

```
import hashlib
from libs.utils import *
from kivy.network.urlrequest import UrlRequest
from kivy.logger import Logger
from Crypto import Random
from Crypto.Cipher import AES
from base64 import urlsafe_b64encode, urlsafe_b64decode
from kivymd.app import MDApp

import os
os.chdir(rf"{os.path.dirname(os.path.realpath(__file__))}")

from libs.utils import *
os.environ["KIVY_METRICS_DENSITY"] = get_scaling()

import threading
from colorsys import rgb_to_hls, hls_to_rgb

from libs.screens.root import Root
from libs.firebase import Firebase

from kivy.core.clipboard import Clipboard
from kivy.animation import Animation
from kivy.core.window import Window
from kivy.clock import Clock
from kivy.properties import (
    BooleanProperty,
    ColorProperty,
    get_color_from_hex,
    StringProperty,
)
from kivy.metrics import Metrics
from kivymd.toast import toast
from kivymd.app import MDApp
from kivy.core.text import LabelBase
from kivy import platform


if platform == 'linux':
    import subprocess
    output = subprocess.Popen(
        'xrandr | grep "\*" | cut -d" " -f4',
        shell = True,
        stdout = subprocess.PIPE).communicate()[0]
    screenx = int(output.replace(b'\n', b'').split(b'x')[0])
    screeny = int(output.replace(b'\n', b'').split(b'x')[1])
elif platform == 'win':
    from win32api import GetSystemMetrics
    screenx = GetSystemMetrics(0)
    screeny = GetSystemMetrics(1)
else:
    screenx = 0
    screeny = 0


fonts_path = f"./fonts/"
fonts = [
    {
    "name": "Poppins",
        "fn_regular": fonts_path + "Poppins-Regular.ttf",
        "fn_bold": fonts_path + "Poppins-Bold.ttf",
    },
    {
```

```
    "name": "Poppins-Bold",
        "fn_regular": fonts_path + "Poppins-Bold.ttf",
    },
    {
    "name": "BigCircleFont",
        "fn_regular": fonts_path + "DejavuSans.ttf",
    },
]

for font in fonts:
    LabelBase.register(**font)
dpi = Metrics.dpi
if dpi < 80:
    size_x, size_y = 770, 590
    Window.minimum_height = size_x
    Window.minimum_width = 500
else:
    size_x, size_y = 1100, 850
    Window.minimum_height = size_y
    Window.minimum_width = 700
Window.left = (screenx - size_x) / 2
Window.top = (screeny - size_y) / 2
Window.size = (size_x, size_y)

# TODO: Add support for chrome os and linux
# TODO: Make updated .exe file that fixes password not deleting

font_file = f"./fonts/Poppins-Regular.ttf"


class MainApp(MDApp):
    dark_mode = BooleanProperty(False)
    extra_security = BooleanProperty(False)
    text_color = ColorProperty()
    primary_accent = ColorProperty()
    primary_palette = StringProperty()
    bg_color = ColorProperty()
    entered_app = BooleanProperty(False)
    email = StringProperty("DemoMail")

    password_changed = False
    system_dark_mode = False
    auto_sync = False
    backup_failure = False

    encryption_class = None
    update_dialog = None
    exit_dialog = None
    sync_widget = None
    anim_sync = None

    passwords = { }
encrypted_keys = { }
screen_history = []

    path_to_live_ui = "backup_design.kv"

    def __init__(self):
        super().__init__()
        from libs.save_config import SaveConfig

        self.save_config = SaveConfig(
            "auto_sync",
            "dark_mode",
            "system_dark_mode",
            "backup_failure",
            "primary_palette",
```

```
            "ui_scaling"
        )
        self.theme_cls.font_styles.update(
            {
    "H1": [font_file, 96, False, -1.5],
                "H2": [font_file, 60, False, -0.5],
                "H3": [font_file, 48, False, 0],
                "H4": [font_file, 34, False, 0.25],
                "H5": [font_file, 24, False, 0],
                "H6": [font_file, 20, False, 0.15],
                "Button": [font_file, 14, True, 1.25],
                "Subtitle1": [font_file, 16, False, 0.15],
                "Body1": [font_file, 16, False, 0.5],
                "Body2": [font_file, 14, False, 0.25],
            }
        )
        self.primary_palette = get_primary_palette()
        self.signup = False if os.path.exists("./data/user_id.txt") else True
        self.auto_sync = check_auto_sync()
        self.ui_scaling = get_scaling()
        Window.on_minimize = lambda: self.backup_on_pause()
        self.firebase = Firebase()
        self.set_dark_mode()
        threading.Thread(target = self.set_user_mail, daemon = True).start()

    def build(self):
        self.root = Root()
        from libs.modules import CardTextField, Toolbar, List
        self.theme_cls.material_style = "M3"
        self.root.load_screen("SignupScreen" if self.signup else "LoginScreen")
        if not self.signup:
            self.root.LoginScreen.ids.password.focus = True

    def on_primary_palette(self, instance, value):
        self.theme_cls.primary_palette = value
        self.set_theme_colors()

    def set_theme_colors(self):
        self.text_color = (
            self.generate_color(lightness = 0.25)
            if not self.dark_mode
            else self.generate_color(lightness = 0.91)
        )
        self.light_color = self.generate_color()
        self.bg_color_light = self.generate_color(lightness = 0.98)
        self.bg_color_light_hex = self.generate_color(lightness = 0.98, return_hex = True)
        self.bg_color_dark = self.generate_color(darkness = 0.1)
        self.bg_color_dark_hex = self.generate_color(darkness = 0.1, return_hex = True)
        self.bg_color = self.bg_color_dark if self.dark_mode else self.bg_color_light
        self.dark_color = self.generate_color(darkness = 0.18)  # 262626
        self.login_circle_light = self.generate_color(lightness = 0.85)
        self.primary_accent = self.dark_color if self.dark_mode else self.light_color
        self.light_hex = self.generate_color(return_hex = True)
        self.dark_hex = self.generate_color(darkness = 0.18, return_hex = True)


def set_user_mail(self, * args):
        self.email = get_email()

    def backup(self, sync_widget):
        def backup_success():
            toast("Backup Successful")
            self.backup_failure = False
            sync_widget.stop()
            self.password_changed = False

        def backup_failure(*args):
```

```
            print(*args)
            self.backup_failure = True
            toast("Couldn't backup :(, Check your internet connection")
            sync_widget.stop()

        sync_widget.icon = "cloud-upload"
        sync_widget.text = "Backing up.."
        sync_widget.start()
        self.firebase.backup_success = lambda * args: backup_success()
        self.firebase.backup_failure = lambda * args: backup_failure(*args)
        self.firebase.backup()

    def restore(self, sync_widget, user_id= None, decrypt= True):
        def restore_success(req, result):
            sync_widget.stop()
            if result is not None:
                from libs.utils import write_passwords

                write_passwords(result)
                if decrypt:
                    self.passwords = self.encryption_class.load_decrypted()
                toast("Restored successfully")
            else:
                toast("No passwords to restore.")

        def restore_failure(req, result):
            sync_widget.stop()
            toast("Restore Failed")

        sync_widget.icon = "cloud-download"
        sync_widget.text = "Restoring.."
        sync_widget.start()
        self.firebase.restore_success = lambda req, result: restore_success(req, result)
        self.firebase.restore_failure = lambda req, result: restore_failure(req, result)
        if user_id:
            self.firebase.restore(user_id)
        else:
            self.firebase.restore()

    def set_dark_mode(self):
        self.dark_mode = is_dark_mode()
        Clock.schedule_once(
            lambda x: exec("self.entered_app = True", { "self": self}), 1
        )

    def show_toast_copied(self, item):
        toast("Item copied")
        Clipboard.copy(item)

    def animate_signup(self, instance):

        """Animation to be shown when user enters the signup screen"""
        if instance:
            Animation(pos_hint ={ "top": 0.95}, opacity = 1, d = 0.5, t = "out_back").start(
                instance
            )

    def generate_color(
            self,
            hex_color= False,
            color= None,
            return_hex= False,
            lightness= 0.92,
            darkness= 0,
            saturation= None,
    ):
        """
```

```
    :param hex_color:  Instead of passing color as list hexadecimal value can be passed.
    :param color: Takes color like [.5,.5,.5, 1] as Parameter
    :param return_hex: Boolean value if set true the function will return hexadecimal value.
    :param lightness: Value from 0-1. If set to 1 it will return white and 0 will return original color.
    :return:
    """
    if hex_color:
        color = get_color_from_hex(hex_color)
    elif not color:
        color = self.theme_cls.primary_color[:-1]

    h, l, s = rgb_to_hls(*color)
    l = lightness if not darkness else darkness
    if saturation is None:
        s = 0.7 if not darkness else 0.15
    else:
        s = saturation
    color = list(hls_to_rgb(h, l, s))

    if not return_hex:
return color + [1]
    else:
        r, g, b = color
        _hex = (
            hex(round(r * 255))[2:]
            + hex(round(g * 255))[2:]
            + hex(round(b * 255))[2:]
        )
        return _hex

def back_button(self, home_screen= False, * args):
    if not home_screen:
self.screen_history.pop()
    else:
        self.screen_history = ["HomeScreen"]
    self.root.transition.mode = "pop"
    self.root.transition.direction = "right"
    self.root.current = self.screen_history[-1]

def on_dark_mode(self, instance, mode):
    if self.entered_app:
        current_screen = self.root.current
        if current_screen == "HomeScreen":
            primary_color = Animation(
                primary_accent = self.dark_color
                if self.dark_mode
                else self.light_color,
                bg_color = self.bg_color_dark
                if self.dark_mode
                else self.bg_color_light,
                duration = 0.3,
            )
            primary_color.start(self)
            primary_color.on_complete = self.set_theme_style
    else:
        self.set_theme_style()

def set_theme_style(self, * args):
    print("theme_style set")
    self.text_color = (
        self.generate_color(lightness = 0.25)  # get_color_from_hex("611c05")
        if not self.dark_mode
        else self.generate_color(lightness = 0.91)  # get_color_from_hex("fde9e2")
    )
    self.bg_color = self.bg_color_dark if self.dark_mode else self.bg_color_light
    self.primary_accent = self.dark_color if self.dark_mode else self.light_color
    if self.dark_mode:
```

```
            self.theme_cls.theme_style = "Dark"
            self.theme_cls.primary_hue = "300"
        else:
            self.theme_cls.theme_style = "Light"
            self.theme_cls.primary_hue = "500"

    def toggle_mode(self, * args):
        self.dark_mode = not self.dark_mode

    def backup_on_pause(self):
        def success():
            toast("Backed up!")
            self.backup_failure = False

        def failure(*args):
            toast("Some Error occured couldn't backup!")
            self.backup_failure = True

        if self.auto_sync and self.password_changed:
self.firebase.backup()
            self.firebase.backup_success = lambda * args: success()
            self.firebase.backup_failure = lambda * args: failure(*args)
            self.password_changed = False

    def on_stop(self):
        self.save_config.save_settings()


if __name__ == "__main__":
    MainApp().run()


app = MDApp.get_running_app()


class Encryption :
    BLOCK_SIZE = 16

    def __init__(self, master_password: str) -> None:
self.key = hashlib.sha256(master_password.encode()).digest()

    def encrypt(self, plain_text: str) -> str:
plain_text = self.pad(plain_text)
        iv = Random.new().read(self.BLOCK_SIZE)  # Initialization vector
        aes = AES.new (self.key, AES.MODE_CBC, iv)
        encrypted_text = aes.encrypt(plain_text.encode())
        return urlsafe_b64encode(iv + encrypted_text).decode("utf-8")

    def decrypt(self, encrypted_text: str) -> str:
encrypted_text = urlsafe_b64decode(encrypted_text)
        iv = encrypted_text[: self.BLOCK_SIZE]
        aes = AES.new (self.key, AES.MODE_CBC, iv)
        plain_text = aes.decrypt(encrypted_text[self.BLOCK_SIZE :]).decode("utf-8")
        return self.unpad(plain_text)

    def pad(self, plain_text: str) -> str:
number_of_bytes_to_pad = self.BLOCK_SIZE - len(plain_text) % self.BLOCK_SIZE
        letter = chr(number_of_bytes_to_pad)
        padding_str = number_of_bytes_to_pad * letter
        padded_plain_text = plain_text + padding_str
        return padded_plain_text

    def unpad(self, plain_text: str) -> str:
return plain_text[: -ord(plain_text[-1:])]

    def load_decrypted(self) -> dict:
decrypted_pass = { }
```

```
encrypted_pass = load_passwords()
        for name in encrypted_pass:
            decrypted_key = self.decrypt(name)
            app.encrypted_keys[decrypted_key] = name
            decrypted_pass[decrypted_key] = self.decrypt(encrypted_pass[name])
        return decrypted_pass

    def add(self, name: str, password: str) -> bool:
        """
        Add a new password to the dictionary.
        """
        data = load_passwords()
        if name not in app.encrypted_keys:
            encrypted_name = self.encrypt(name)
            app.encrypted_keys[name] = encrypted_name
            data[encrypted_name] = self.encrypt(password)
            write_passwords(data)
            app.password_changed = True
            return True
        else:
            return False

    def delete(self, name: str) -> None:
data = load_passwords()
        del data[name]
        del app.encrypted_keys[self.decrypt(name)]
write_passwords(data)
        app.password_changed = True

    def update(self, name: "encrypt(str)", password: str) -> None:
data = load_passwords()
        password = self.encrypt(password)
        data[name] = password
        app.encrypted_keys[self.decrypt(name)] = name
        write_passwords(data)
        app.password_changed = True

    def find_key(
        self, dictionary: dict, text: str
    ) -> List[Tuple[Tuple[str, str], int]]:
        """
        Finding Algorithm, uses basic search along with fuzzy search (self-made).
        """
        weighted_pass = { }


def sub_string_search(text: str, name: str, password: str):
            def get_max_val(text: str, name: str, priority: float):
                """
                Weights password based on length of searched text and no. of matching letters in the passwords.
                """
                return max(len(text) / len(name), max_value) + priority

            name_wo_space = name.replace(" ", "")
            text_wo_space = text.replace(" ", "")
            case_insensitive_text, case_insensitive_name = text.lower(), name.lower()
            max_value = 0
            if text == name_wo_space:
                return True
            else:
                if case_insensitive_name.startswith(case_insensitive_text):
                    max_value = get_max_val(
                        case_insensitive_text, case_insensitive_name, 0.15
                    )
                else:
                    if (
                        text in name
```

```
            ):  # Just checks if letteres searched are substring of the key in dictionary.
                max_value = get_max_val(text, name, priority = 0)

            if (
                case_insensitive_text in case_insensitive_name
            ):  # Weights passwords without checking case.
                max_value = get_max_val(
                    text = case_insensitive_text,
                    name = case_insensitive_name,
                    priority = 0,
                )

            if (
                text in name_wo_space
            ):  # Removes spaces from keys then checks whether text is found.
                max_value = get_max_val(
                    text, name = name_wo_space, priority = -0.01
                )

            if (
                text_wo_space in name_wo_space
            ):  # Removes spaces both searched text and key and then weights password.
                max_value = get_max_val(
                    text = text_wo_space, name = name_wo_space, priority = -0.02
                )

        if max_value > 0.1:  # If no letters match don't add.
            weighted_pass[(name, password)] = max_value

        return False

def fuzzy_search(text: str, name: str, password: str):
    def get_max_val(search_text: str, name: str, priority: float):
        matching_letter = 0
        min_length = min(len(search_text), len(name))
        max_length = max(len(search_text), len(name))
        for index in range(min_length):
            if search_text[index] == name[index]:
                matching_letter += 1
        return (matching_letter / max_length) + priority

    def get_max_ordered_value(search_text: str, name: str, priority: float):
        """
        Checks for matching subsequences found in the searched text.
        Like "abcef" in "abcdef" and vice-versa.
        """
        matching_letter = 0
        min_length = min(len(search_text), len(name))
        max_length = max(len(search_text), len(name))
        search_index = 0
        for index in range(min_length):
            if search_text[search_index] == name[index]:
                matching_letter += 1
                search_index += 1
            else:
                matching_letter -= 0.05

        max_matching = matching_letter
        key_index = matching_letter = 0
        for index in range(min_length):
            if search_text[index] == name[key_index]:
                matching_letter += 1
                key_index += 1
            else:
                matching_letter -= 0.05

        max_matching = max(matching_letter, max_matching)
```

```
            return (max_matching / max_length) + priority

        max_val = max(
            get_max_val(text, name, -0.04), get_max_ordered_value(text, name, -0.07)
        )

        lower_text = text.lower()
        lower_name = name.lower()
        max_val = max(
            get_max_val(lower_text, lower_name, -0.05),
            get_max_ordered_value(lower_text, lower_name, -0.08),
            max_val,
        )

        nospace_text = text.replace(" ", "")
        nospace_name = name.replace(" ", "")
        max_val = max(
            get_max_val(nospace_text, nospace_name, -0.06),
            get_max_ordered_value(nospace_text, nospace_name, -0.09),
            max_val,
        )
        if max_val > 0.1:
            weighted_pass[(name, password)] = max_val

    def sort_keys(key_dict: dict):
        key_arr = list(key_dict.items())
        key_arr.sort(
            key = lambda x: x[1], reverse = True
        )  # Sorts the list based on the weighted values.
        return key_arr

    if (
        text in dictionary
        or text.lower() in dictionary
        or text.upper() in dictionary
    ):
        return [((text, dictionary[text]), 1)]
    else:
        for name, password in dictionary.items():

            # If text equals password without space then it returns that, else checks all passwords.
            if sub_string_search(text, name, password):
                return [((name, password), 1)]

        if not weighted_pass:
for name, password in dictionary.items():

                # Uses approximate search to find the password.
    fuzzy_search(text, name, password)


result = sort_keys(weighted_pass)
    return result


class Firebase :
    SIGNUP_URL = (
        "https://www.googleapis.com/identitytoolkit/v3/relyingparty/signupNewUser?key="
    )
    LOGIN_URL = (
        "https://www.googleapis.com/identitytoolkit/v3/relyingparty/verifyPassword?key="
    )
    DELETE_URL = (
        "https://www.googleapis.com/identitytoolkit/v3/relyingparty/deleteAccount?key="
    )

    WEB_API_KEY = "AIzaSyB0-Xy5Ar7JL_Rlg1PeO4CLKkaVbjM71TQ"  # Use your API key, for your own project.
```

```
    DATABASE_URL = "https://paock-9978a-default-rtdb.asia-southeast1.firebasedatabase.app"  # Use your database URL, for your own project.

    def signup_success(self, req, result):
        """
        Implement this method to handle the result of the successful signup request.
        """
        Logger.warn("Firebase: Signup success not implemented")

    def signup_failure(self, req, result):
        """
        Implement this method to handle the result of the signup failure request.
        """
        Logger.warn("Firebase: Signup failure not implemented")

    def signup(self, name, password):
        """
        Used to signup the user.
        """
        url = self.SIGNUP_URL + self.WEB_API_KEY
        data = json.dumps(
            { "email": name, "password": password, "returnSecureToken": True}
        )
        UrlRequest(
            url,
            req_body = data,
            on_success = self.signup_success,
            on_failure = self.signup_failure,
            timeout = 15,
        )

    def login_success(self, req, result):
        """
        Implement this method to handle the result of the successful login request.
        """
        Logger.warn("Firebase: Login success not implemented")

    def login_failure(self, req, result):
        """
        Implement this method to handle the result of the login failure request.
        """
        Logger.warn("Firebase: Login failure not implemented")

    def login(self, name, password):
        """
        Used to login the user.
        """

        url = self.LOGIN_URL + self.WEB_API_KEY
        data = json.dumps(
            { "email": name, "password": password, "returnSecureToken": True}
        )
        UrlRequest(
            url,
            req_body = data,
            on_success = self.login_success,
            on_failure = self.login_failure,
            timeout = 10,
        )

    def backup_success(self, req, result):
        """
        Override this method to handle the result of the successful backup request.
        """
        Logger.warn(f"Firebase: Backup success not implemented, {result}")
```

```
    def backup_failure(self, req, result):
        """
        Override this method to handle the result of the failure in backup request.
        """
        Logger.warn(f"Firebase: Backup failure not implemented, {result}")

    def backup(self):
        """
        Used to backup the user's passwords.
        """
        result = load_passwords()
        _json = { }
_json[get_uid()] = result
        UrlRequest(
            self.DATABASE_URL + "/.json",
            req_body = json.dumps(_json),
            req_headers ={ "Content-type": "application/json", "Accept": "text/plain"},
            on_success = self.backup_success,
            on_failure = self.backup_failure,
            on_error = self.backup_failure,
            method = "PATCH",
        )

    def restore_success(self, req, result):
        """
        Override this method to handle the result of the successful restore request.
        """
        Logger.warn(f"Firebase: Restore success not implemented, {result}")

    def restore_failure(self, req, result):
        """
        Override this method to handle the result of the failure in restore request.
        """
        Logger.warn(f"Firebase: Restore failure not implemented, {result}")

    def restore(self, user_id= None):
        """
        Used to restore the user's passwords.
        """
        if user_id is None:
            user_id = get_uid()

        UrlRequest(
            f"{self.DATABASE_URL}/{user_id}.json",
            on_success = self.restore_success,
            on_failure = self.restore_failure,
            on_error = self.restore_failure,
        )

__version__ = "1.0.0.dev0"
"""KivyMD version."""

release = False
kivy.require("2.0.0")


try:
    from kivymd._version import __date__, __hash__, __short_hash__
except ImportError:
__hash__ = __short_hash__ = __date__ = ""

path = os.path.dirname(__file__)
"""Path to KivyMD package directory."""

images_path = os.path.join(path, f"images{os.sep}")
"""Path to images directory."""
```

```
_log_message = (
    "KivyMD:"
    + (" Release" if release else "")
    +f" {__version__}"
    + (f", git-{__short_hash__}" if __short_hash__ else "")
    +(f", {__date__}" if __date__ else "")
    +f' (installed at "{__file__}")'
)
Logger.info(_log_message)

import kivymd.factory_registers  # NOQA
import kivymd.font_definitions  # NOQA

if "KIVY_DOC_INCLUDE" in os.environ:
    dp = lambda x: x  # NOQA: F811

DEVICE_IOS = platform == "ios" or platform == "macosx"
if platform != "android" and platform != "ios":
    DEVICE_TYPE = "desktop"
elif Window.width >= dp(600) and Window.height >= dp(600):
    DEVICE_TYPE = "tablet"
else:
    DEVICE_TYPE = "mobile"
```