

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: Проєкт застосунку «Менеджер паролів»

Виконав: студент 3-го курсу групи КБк-22-1
спеціальності 125 «Кібербезпека»



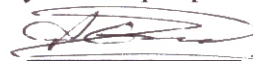
Тарас ГЛАДИШ

Керівник: доцент кафедри КІ та КБ, к.т.н.



Олексій ШЕЛУХА

Рецензент: завідувач кафедри КІ та КБ, к.т.н., доцент



Андрій ЄФІМЕНКО

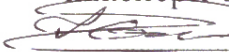
Житомир – 2025

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій ЄФІМЕНКО

3 березня 2025 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Гладиш Тарас В'ячеславович

Тема роботи: Проект застосунку «Менеджер паролів»

затверджена Наказом університету від 3 березня 2025 р. №80/с

Термін здачі студентом закінченої роботи 9 червня 2025 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

- Процеси захисту паролів

- Технології, методи та засоби захисту паролів, алгоритми шифрування та хешування.

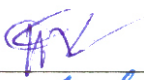
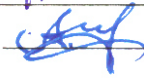
- Онлайн ресурси перевірки наявності паролів у переліку скомпрометованих.

Обладнання та програмне забезпечення:

1. середовище програмування Visual Studio Code
2. бібліотека для розробки мобільних застосунків React Native та Expo Go
3. операційні система Android та IOS
4. API сервісу HIBP

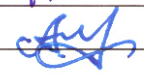
Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1-3	Пірог О.В.	03.03.2025	03.03.2025

Студент  Тарас ГЛАДИШ
Керівник  Олексій ШЕЛУХА

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	03.03.2025	
2	Аналіз отриманого завдання	05.03.2025	
3	Постановка задачі дослідження	10.03.2025	
4	Аналіз літературних джерел	28.04.2025	
5	Аналіз об'єкта дослідження	28.04.2025	
6	Проектування функціональних елементів застосунку	12.05.2025	
7	Реалізація функціональних блоків	26.05.2025	
8	Тестування розробленого застосунку	02.06.2025	
9	Оформлення пояснювальної записки	09.06.2025	

Студент  Тарас ГЛАДИШ
Керівник  Олексій ШЕЛУХА

АНОТАЦІЯ

Кваліфікаційна робота присвячена дослідженню методів та засобів безпечного зберігання, управління паролів користувача. В роботі виконано проєкт мобільного застосунку «Менеджер паролів» у якому користувач зможе безпечно зберігати, управляти та перевіряти свої паролі на скомпрометованість.

Пояснювальна записка до випускної кваліфікаційної роботи на тему «Проєкт застосунку «Менеджер паролів»» складається з трьох розділів, висновків, списку використаних джерел та 2 додатків обсягом 2 сторінки.

Загальний обсяг роботи займає 54 сторінки, містить 39 рисунок та 2 таблиці. Список використаних джерел містить 17 найменувань і займає 2 сторінки.

В першому розділі було розглянуто проблематику витоку даних користувачів, історія та ідея створення сервісу НІВР, проаналізовано існуючі аналоги та описано їх основні переваги та недоліки. Виконано постановку технічного завдання, яке формує необхідні вимоги для розроблюваного застосунку.

В другому розділі було проведено комплексний аналіз методів шифрування, з метою визначення найбільш ефективного та безпечного підходу для реалізації функціоналу захисту даних у мобільному застосунку. Окрему увагу приділено проектуванню алгоритмів функціонування застосунку.

В третьому розділі описано розробку проєкту та продемонстровано роботу застосунку. Проведено інтеграцію із сервісом НІВР.

Висновок містить в собі результати виконаної роботи.

У додатках до роботи наведено скріншоти роботи перевірки надійності та результати генерації пароля.

Ключові слова: МЕНЕДЖЕР ПАРОЛЕЙ, ШИФРУВАННЯ, БЕЗПЕЧНЕ ЗБЕРІГАННЯ ПАРОЛЕЙ, ЛОКАЛЬНЕ СХОВИЩЕ ПАРОЛЕЙ, МОБІЛЬНИЙ ДОДАТОК.

					ККІКБ.КР.Б – 125 – 3 – ПЗ					
Змн.	Арк.	№ докум.	Підпис	Дата	Проект застосунку "Менеджер паролів"			Літ.	Арк.	Аркушів
Розроб.		Гладиш Т.В.		19.06.25						
Керівник		Шелуха О.О.		19.06.25					4	54
Реценз.		Єфіменко А.А.		19.06.25				ФІКТ, Гр. КБк-22-1		
Н. контр.		Шелуха О.О.		19.06.25						
Затверд.		Єфіменко А.А.		19.06.25						

ANNOTATION

The qualification thesis is dedicated to the study of methods and tools for secure storage and management of user passwords. The project includes the development of a mobile application called "Password Manager", which allows users to securely store, manage, and check their passwords for compromise.

The explanatory note to the final qualification thesis titled "Password Manager Application Project" consists of three chapters, conclusions, a list of references, and two appendices with a total of 2 pages.

The total volume of the work is 54 pages, containing 39 figures and 2 tables. The list of references includes 17 sources and occupies 2 pages.

The first chapter examines the problem of user data breaches, the history and concept behind the HIBP service, and provides an analysis of existing analogs along with their main advantages and limitations. The technical specification is formulated, defining the requirements for the developed application.

The second chapter presents a comprehensive analysis of encryption methods to determine the most effective and secure approach for implementing data protection functionality in the mobile application. Particular attention is paid to designing the operational logic of the application.

The third chapter describes the development of the project and demonstrates the functionality of the application. Integration with the HIBP service has been implemented.

The conclusion summarizes the results of the completed work.

The appendices contain screenshots demonstrating the password strength check and password generation results.

Keywords: PASSWORD MANAGER, ENCRYPTION, SECURE PASSWORD STORAGE, LOCAL PASSWORD STORAGE, MOBILE APPLICATION.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ СУЧАСНОГО СТАНУ СФЕРИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ПАРОЛІВ	10
1.1 Кіберзагрози в контексті захисту облікових записів.	10
1.2 Аналіз ресурсу перевірки безпечності паролів.....	13
1.3 Аналіз існуючих засобів зберігання та управління паролями.	15
1.4 Постановка технічного завдання.....	19
Висновок до першого розділу.	20
РОЗДІЛ 2. ПРОЄКТУВАННЯ ФУНКЦІОНАЛЬНИХ ЕЛЕМЕНТІВ ЗАСТОСУНКУ «МЕНЕДЖЕР ПАРОЛІВ»	21
2.1 Аналіз методів шифрування.	21
2.2 Побудова алгоритмів функціонування модулів застосунку.....	25
2.3 Аналіз фреймворків для створення мобільних застосунків.	32
Висновок до другого розділу.....	35
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОЄКТУ.	36
3.1 Реалізація алгоритмів та функціональних блоків.....	36
3.2 Налаштування та перевірка функціоналу.	41
3.3 Реалізація функціоналу перевірки паролів на скомпрометованість.....	46
3.4 Аналіз отриманих результатів та подальших досліджень.....	49
Висновок до третього розділу.	50
ВИСНОВКИ	51
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	53
ДОДАТКИ	55

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AES – Advanced Encryption Standard

API – Application Programming Interface

HIBP – Have I Been Pwned

ДФА – двофакторна аутентифікація

МП – менеджер паролей

ОС – операційна система

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. У сучасному цифровому світі значна частина особистої та корпоративної інформації зберігається в онлайн-сервісах, доступ до яких забезпечується за допомогою паролів. Велика частина користувачів й досі використовує однакові паролі для всіх своїх акаунтів, зберігаючи їх у незашифрованому вигляді у нотатках телефону, скріншотах у галереї чи навіть на папірцях. Ненадійне зберігання та використання однакових паролів для різних ресурсів створює серйозні ризики витоку або втрати конфіденційних даних.

Отже, розробка надійного та функціонального менеджера паролів є актуальним завданням у контексті сучасних викликів цифрової безпеки. З огляду на зростання кількості онлайн-сервісів та ризиків, пов'язаних із витоком персональних даних, необхідність у засобах безпечного зберігання автентифікаційної інформації більше не є додатковою опцією, а виступає базовою вимогою для забезпечення конфіденційності та захисту даних користувача.

Метою кваліфікаційної роботи є розробка сучасного мобільного застосунку типу «Менеджер паролей», який дозволить користувачам зберігати конфіденційні дані у зашифрованому вигляді, забезпечуючи надійний захист за допомогою шифрування та інтуїтивно зрозумілий інтерфейс. В роботі шифрування буде забезпечуватись алгоритмом AES, а інтерфейс буде створено на базі бібліотеки React Native.

Визначена мета обумовлює наступні завдання:

1. Провести аналіз сучасного стану проблеми цифрової безпеки та визначити основні загрози, пов'язані з використанням ненадійних та однакових паролів. Проаналізувати наявні, аналогічні по функціоналу, рішення, визначити їх сильні й слабкі сторони.

2. Обрати оптимальні методи та технології для забезпечення безпеки паролів та розробити алгоритми роботи функціональних блоків мобільного застосунку, що буде об'єднувати весь необхідний функціонал.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Реалізувати проєкт мобільного застосунку «Менеджер паролів» що буде об'єднувати механізми шифрування даних, забезпечення локальної авторизації та перевірки збереженого пароля на скомпрометованість.

Об'єктом дослідження є процес безпечного зберігання та управління паролями користувача.

Предметом дослідження є методи та технології безпечного зберігання та шифрування паролів користувача, зокрема із використанням алгоритму AES, перевірки їх надійності та скомпрометованості, засобами HIBP API, методи забезпечення локальної авторизації та фреймворк React Native для створення сучасного, адаптивного мобільного інтерфейсу.

У процесі виконання кваліфікаційної роботи було розроблено проєкт мобільного застосунку «Менеджер паролей» – кросплатформного рішення, що дозволяє зберігати та шифрувати паролі користувачів локально на пристрої, без необхідності передавання даних у мережу.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ СУЧАСНОГО СТАНУ СФЕРИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ПАРОЛІВ

1.1 Кіберзагрози в контексті захисту облікових записів.

З розвитком цифрових технологій проблема захисту особистої інформації стає все більш гострою. На сьогоднішній день неможливо уявити життя без інтернету. В інтернеті ми спілкуємося з родиною, зберігаємо особисту інформацію, взаємодіємо з банківськими додатками і головною проблемою цих додатків є захист інформації та безпека. Головним інструментом в забезпеченні безпеки додатку є автентифікація.

Автентифікація – це процес перевірки справжності користувача за спроби доступу до системи чи сервісу, підтвердження того, що особа дійсно є тією, за кого себе видає [2]. Важливо розрізняти пов'язані процеси:

- ідентифікація – розпізнавання користувача в системі за унікальним ідентифікатором;
- автентифікація – підтвердження справжності наданих даних;
- авторизація – надання доступу до ресурсів системи після успішної автентифікації.

По сьогоднішній день для перевірки особистості користувача використовується логін та пароль і стає однією з цілей хакерів. Користувачі часто використовують повторювані паролі для своїх соцмереж, що значно полегшує доступ зловмисникам. Хакери, які регулярно зламують та зливають на своїх форумах бази даних великих компаній які зберігають інформацію про своїх користувачів. І ось декілька із найбільших зливів в історії: [3]

- Взлом PlayStation Network. В квітні 2011 року PlayStation Network зазнали потужної кібератаки. В результаті імена, адреси та інші особисті дані близько 77 млн людей, які мали облікові записи в компанії, були викрадені. Крім того, інформація також включала платіжні дані клієнтів та номери кредитних карток.

- Adobe Cyber Attack У 2013 році компанія Adobe стикнулася з атакою хакерів, яка призвела до масштабного витоку даних клієнтів. Спочатку вважалося,

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

що кібератака зламала дані 2,9 млн користувачів. Проте пізніше компанія заявила, що витік даних зачепив понад 38 млн.

- Витік даних в Yahoo. У 2014 році Yahoo зазнав однієї з найбільших кібератак року. Хакерам вдалося проникнути в мережу та скомпрометувати дані більш ніж 500 млн користувачів. Такий витік і досі вважають найбільшим у світі. В руках зловмисників опинилися імена, адреси електронної пошти, дати народження та телефонні номери.

- У 2012 році відбулася масштабна кібератака на LinkedIn. В результаті цієї атаки зловмисникам вдалося викрасти адреси електронних пошт, паролі та інші дані 117 млн користувачів. [4]

- Атака на Facebook. У 2021 році хакери отримали доступ до 533 млн записів користувачів із 106 країн. Дані включали в себе: facebook identities, повне ім'я користувача, дату народження, локацію та адресу електронної пошти.

- Найбільший витік паролей відбувся в 2024 році. На форумі хакерів, користувач з нікнеймом ObamaCare, виклав файл який містить в собі майже 10 млрд (а саме 9 948 575 739) унікальних паролей користувачів. Ці паролі були зібрані із 4 тисяч злитих баз даних та витоку який оприлюднили в 2021 році.

В результаті аналізу кібератак видно, що дані користувачів все частіше опиняються в руках зловмисників, що дозволяє отримати несанкціонований доступ до облікового запису жертви навіть без навичок хакінгу (рисунок 1.1).

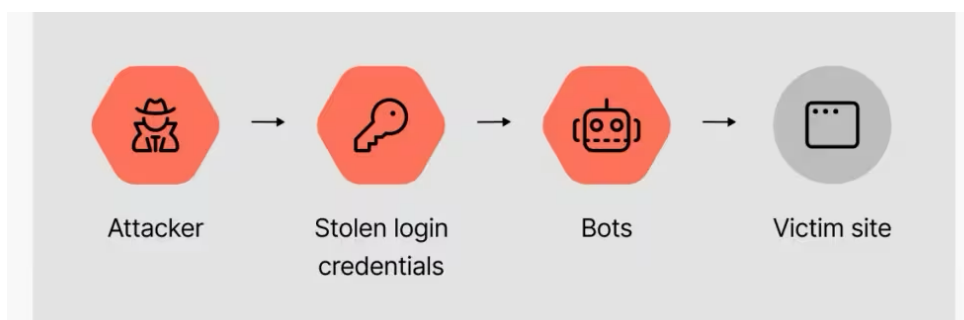


Рисунок 1.1 – Атака Credential stuffing

Тривожна тенденція витоку даних спонукала компанії шукати додаткові способи захисту акаунти користувачів від зламу шляхом підбору паролів. Один із

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

рішень, який допомагає захистити обліковий запис – двофакторна аутентифікація (далі ДФА). ДФА (2FA) — метод додаткового захисту вашого облікового запису. Перший «фактор» — це звичайний пароль, стандартний для будь-якого облікового запису. Другий «фактор» — код підтвердження, отриманий від окремої програми на мобільному пристрої чи комп'ютері [8]. Навіть при наявності логіну та пароля користувача, зломисник не отримає доступу без коду доступу або підтвердження із смартфона жертви зламу. Хоча багато відомих компаній дозволяють включити ДФА, такий метод автентифікації залишається опціональним і багато користувачів, не розуміючи ризику, ігнорують цей корисний інструмент.

Витік даних може статися за різних умов і з багатьох причин. Ось кілька найтипівіших ситуацій [17]:

- Неправильне поводження з даними. Досить часто витік стається не через складні хакерські атаки, а через звичайну необачність. Наприклад, людина може випадково надіслати конфіденційний файл не тій особі, залишити свій смартфон або ноутбук без нагляду в публічному місці, не заблокувавши його, або використовувати надто прості, повторювані паролі, які легко вгадати. Усі ці дії відкривають зломисникам шлях до особистої інформації без необхідності технічного втручання.

- Атака шкідливого програмного забезпечення. Трояни, кейлогери й інше шкідливе ПЗ активно використовуються для непомітного збору чутливої інформації: від натискань клавіш до знімків екрана та зчитування вмісту буфера обміну. Установлення таких програм часто відбувається без відома користувача – через заражені додатки, підроблені оновлення чи фішингові посилання.

- Витік на рівні організацій. Навіть великі, добре відомі компанії – банки, соціальні мережі, онлайн-магазини – не застраховані від витоків. Причинами можуть бути як цілеспрямовані атаки з боку хакерських угруповань, так і внутрішні помилки адміністраторів, відсутність шифрування або використання застарілих протоколів безпеки. Наслідки таких інцидентів можуть охоплювати мільйони акаунтів, ставлячи під загрозу дані великої кількості людей одночасно.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

- **Крадіжка пристрою.** У разі втрати чи крадіжки смартфона, планшета або ноутбука користувач ризикує втратити доступ до своїх облікових записів, банківських додатків, особистих фото та документів. Якщо на пристрої відсутній захист паролем або шифрування, зловмисники можуть отримати повний доступ до всіх збережених даних – і навіть продати їх третім сторонам.

1.2 Аналіз ресурсу перевірки безпечності паролів.

Have I Been Pwned (у перекладі – «Мене зламали?», далі HIBP) – це безкоштовний онлайн-сервіс, який допомагає людям перевірити, чи потрапили їх особисті дані до рук зловмисників. Основна функція HIBP – це перевірка наявності електронної пошти або пароля у масивних базах витоків, зібраних із відкритих і закритих джерел.

Унікальність сервісу полягає і в його безпеці – ніколи не потрібно передавати пароль у відкритому вигляді. Замість цього використовується алгоритм k-анонімності: клієнт надсилає на сервер лише перші п'ять символів SHA-1 хешу пароля, а далі зіставлення відбувається локально. Принцип роботи k-анонімності зображено на рисунку 1.2. Таким чином, навіть сам сервер не дізнається повного хешу, не кажучи вже про сам пароль.

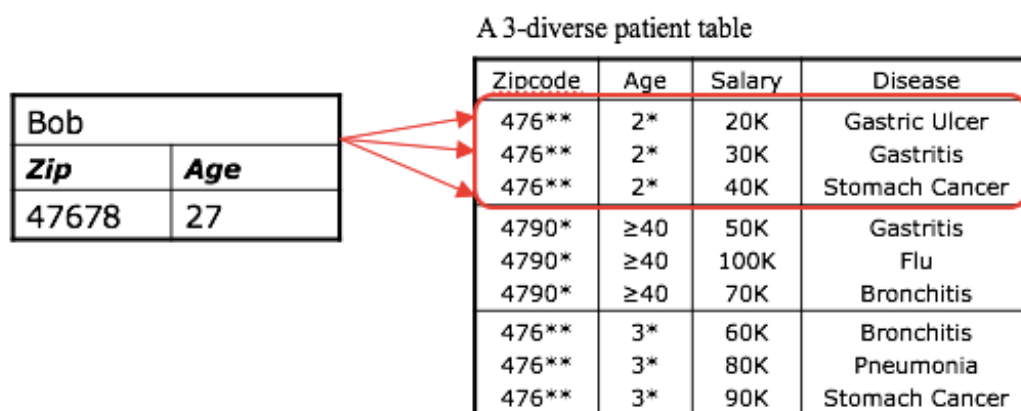


Рисунок 1.2 – Принцип роботи k-анонімності

Історія HIBP починається з однієї людини – Троя Ханта, австралійського спеціаліста з кібербезпеки, який у 2013 році працював у Microsoft і паралельно вів освітній блог про безпечну розробку. Його турбувало те, що з кожним новим витокком даних – чи то Adobe, LinkedIn, Dropbox чи MySpace – усе більше людей ставали жертвами, навіть не знаючи про це. Ті, хто мав технічну освіту, могли перевірити хеші чи знайти свою адресу у витокку. Але що робити звичайній людині?

У грудні 2013 року він вирішив створити публічний сервіс, який:

1. Дозволяв би будь-кому дізнатися про взлом їх акаунта.
2. Не вимагав технічних знань.
3. Був би настільки безпечним, щоб не став черговим джерелом витокку даних.

Так з'явився HIBP – спочатку як експериментальний сайт, де можна було ввести свою адресу електронної пошти й побачити, у яких витокках вона зустрічалась. Усе працювало на особистому ентузіазмі, без фінансування та реклами. Першою базою став масив зламу Adobe – понад 150 мільйонів записів. Виявилось, що попит на такий інструмент шалений – вже за перші тижні сайт отримав сотні тисяч переглядів.

Подальший розвиток був стрімким:

- У 2014 році Хант додав перевірку паролів;
- У 2017 році запустив Pwned Passwords API, де реалізував концепцію k-анонімності;
- У 2020 році, після декількох спроб комерціалізації, оголосив про відкриття коду частини HIBP;
- У 2021-2023 роках HIBP почали використовувати урядові організації, великі компанії та розробники по всьому світу.

Для взаємодії з сервісом можна використовувати офіційний сайт (рисунок 1.3), або API для інтеграції у власні проекти.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

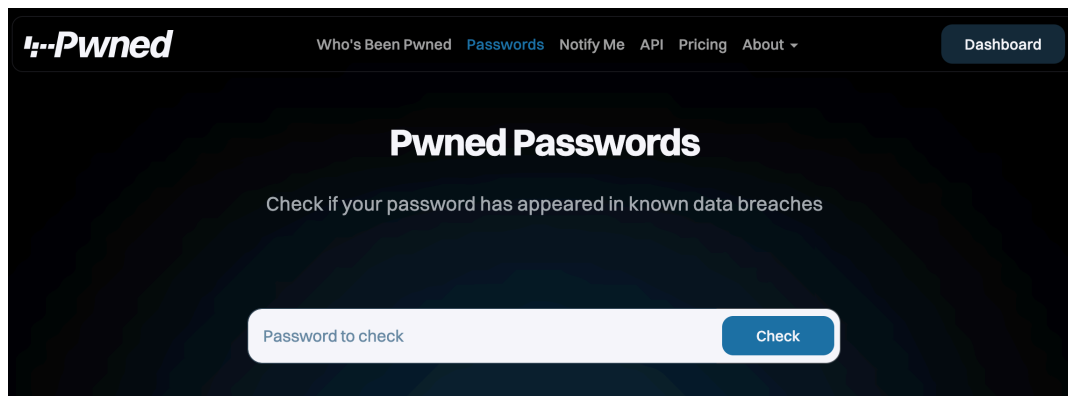


Рисунок 1.3 – Вигляд сайту для перевірки скомпрометованості пароля

НІВР – приклад того, як особиста ініціатива, помножена на технічну майстерність і етичну мотивацію, може змінити екосистему цифрової безпеки. І навіть сьогодні, коли сервіс працює в автоматичному режимі, Трой Хант продовжує особисто верифікувати великі бази даних і публічно ділитись аналізом кожного великого витоку.

Аналогічні сервіси для перевірки паролів мають обмеження для безкоштовної версії. Мають власні API, але для преміум версії. Ще одним суттєвим мінусом є надійність джерела. НІВР додає в загальну базу тільки перевірені та офіційні витoki, тоді як аналогічні сервіси використовують бази із даркнету або неофіціні.

1.3 Аналіз існуючих засобів зберігання та управління паролями.

Сучасний ринок пропонує великий список рішень які дозволяють користувачам безпечно зберігати свої паролі. Розглянемо найпопулярніші.

LastPass (рисунок 1.4) – це сервіс для управління паролями та їх безпечного зберігання [5]. Створений у 2008 році компанією Marvasol Inc. (згодом придбаний LogMeIn). Основні заходи безпеки: шифрування AES-256, двофакторна автентифікація, майстер-пароль. У 2022 році зазнав серйозного зламу: хакери викрали резервні копії зашифрованих паролів із хмарного сховища та частину незашифрованих метаданих. Це стало значним ударом по довірі до сервісу.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

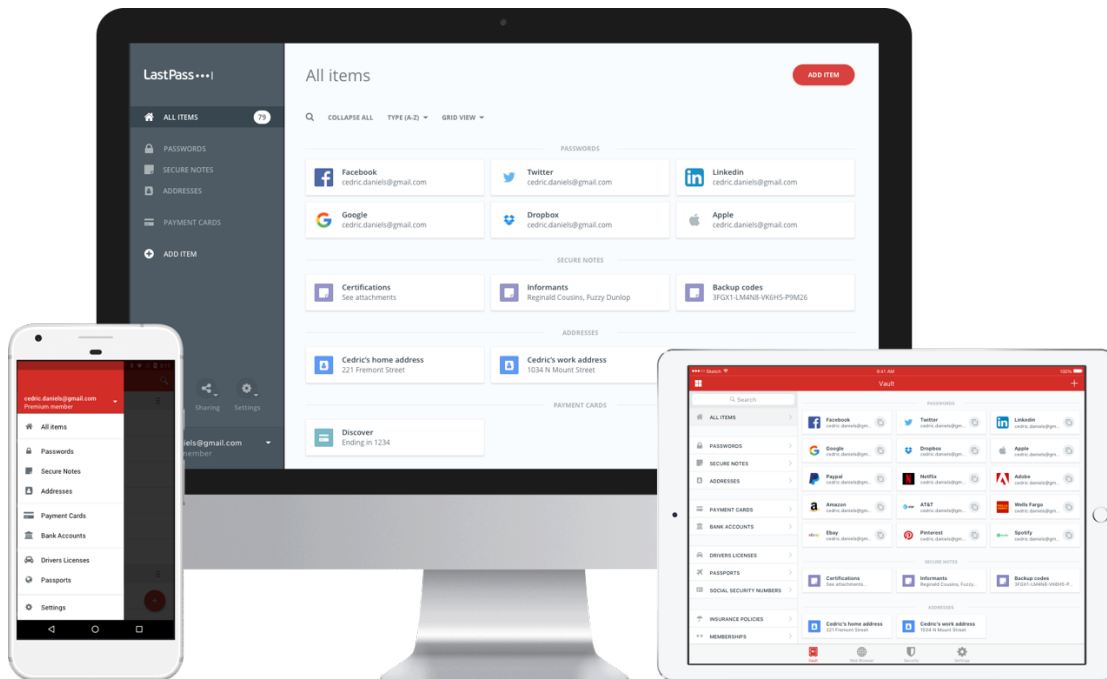


Рисунок 1.4 – Загальний вигляд LastPass на різних пристроях

1Password (рисунок 1.5) – програма для зберігання паролів та будь-яких інших даних користувача на локальному або віддаленому захищеному сервері [6]. Запущений у 2005 році канадською компанією AgileBits Inc. Серед заходів безпеки: використання Secret Key разом із майстер-паролем, AES-256, двофакторна автентифікація. Репутація додатку вважається однією з найкращих у галузі, з постійною увагою до оновлень та інновацій.

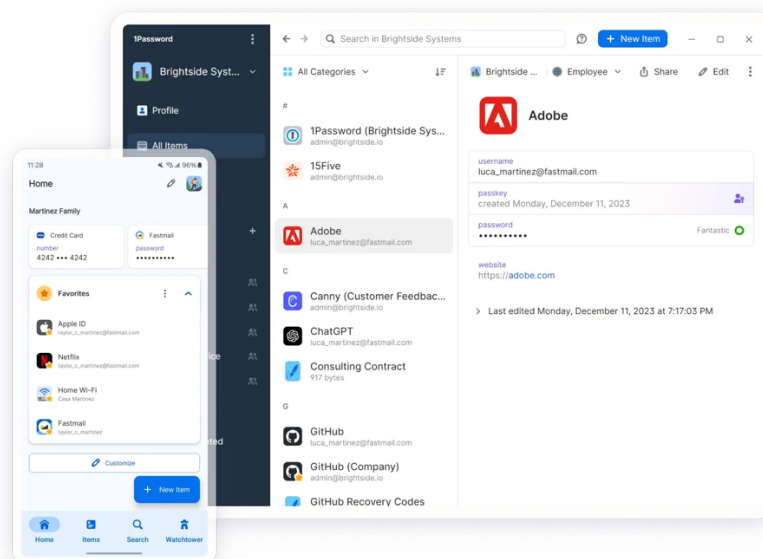


Рисунок 1.5 – Загальний вигляд 1Password

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Bitwarden (рисунок 1.6) – це популярний менеджер паролів з відкритим кодом, який допомагає зберігати, керувати та захищати дані. З'явився у 2016 році, розроблений компанією 8bit Solutions LLC. Захист забезпечується AES-256, PBKDF2 SHA-256, Argon2 та двофакторною автентифікацією. Bitwarden здобув популярність завдяки прозорості та спільноті, яка контролює якість коду.

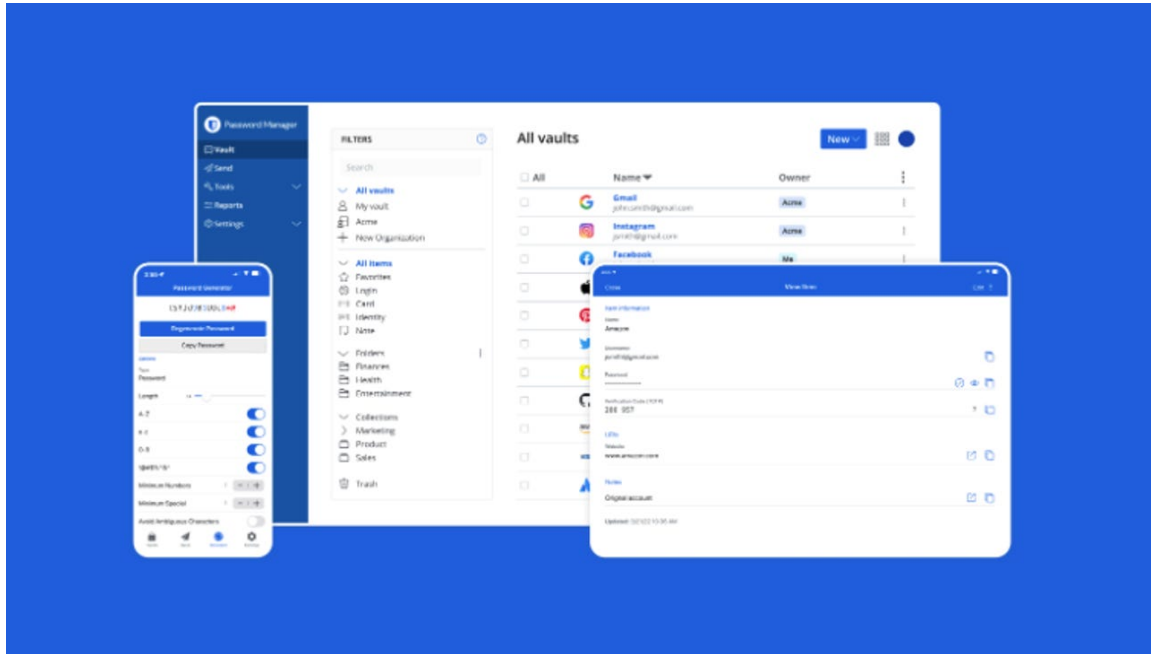


Рисунок 1.6 – Загальний вигляд Bitwarden

Dashlane (рисунок 1.7) – це зручний менеджер паролів, створений для покращення цифрової безпеки як для індивідуальних користувачів, так і для команд. Програма пропонує повний набір інструментів, що виходять за межі звичайного зберігання паролів: інтегрований VPN, аудит безпеки паролів та автоматичне заповнення [7]. Запущений у 2012 році однойменною компанією Dashlane Inc. Шифрування AES-256, zero-knowledge-архітектура. Dashlane вирізняється фокусом на UX/UI, проте має дорожчий платний план порівняно з конкурентами.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

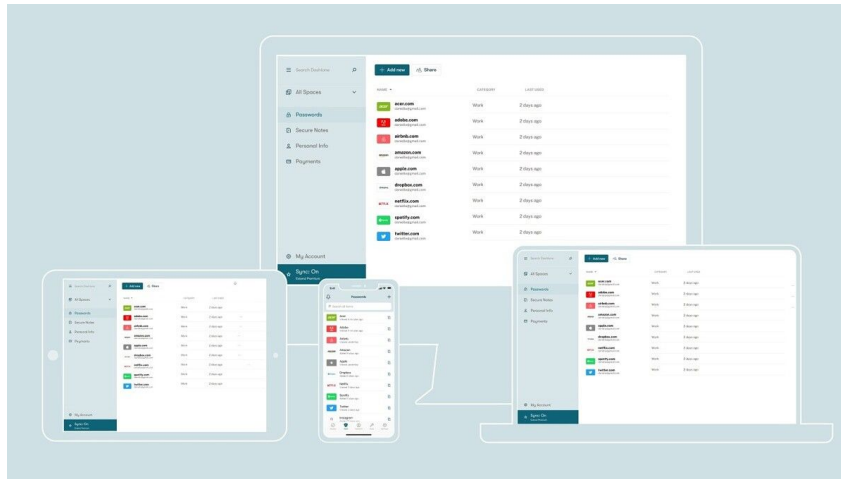


Рисунок 1.7 – Загальний вигляд Dashlane

Познайомившись з аналогами та розглянувши основні функції опишемо їх переваги та недоліки у форматі таблиці.

Таблиця 1.1

Головні переваги та недоліки існуючих МП

Назва	Переваги	Недоліки
LastPass	+ хмарне збереження, + простий інтерфейс, + автозаповнення	- витік даних у 2022 році, - обмеження у безкоштовному тарифі.
1Password	+ надійний захист, + інтеграція з Apple Watch, + сімейні тарифні плани	- висока ціна, - відсутність безкоштовної версії.
Bitwarden	+ відкритий код, + прозорість, + підтримка самостійного хостингу.	- не інтуїтивний інтерфейс
Dashlane	+ аналітика паролів, + зручний та гарний дизайн, + велика кількість інструментів та функцій	- висока ціна, навіть на фоні конкурентів, - обмеження у безкоштовному тарифі.

Проаналізувавши аналоги видно, що додатки допомагають вирішити проблему захисту паролів, але жоден з них не є ідеальним. Тому розробка власного рішення дозволяє не тільки врахувати кращі практики, але й адаптувати функціонал до потреб конкретного користувача, не залежати від сторонніх хмарних сервісів і створити локальне, незалежне та безпечне середовище.

1.4 Постановка технічного завдання.

Проблема злиття паролів є надзвичайно актуальною в умовах сучасної цифрової епохи. Існуючі рішення, як-от хмарні менеджери паролів, хоч і пропонують зручність та додаткові функції, проте несуть ризики, пов'язані з централізованим зберіганням даних та можливими зламами серверів. Більше того, користувачі не завжди довіряють третім сторонам, особливо після інцидентів типу зламу LastPass у 2022 році.

Основними недоліками існуючих рішень є:

- залежність від хмарних сервісів і постійного доступу до Інтернету;
- централізоване зберігання, що створює єдину точку вразливості;
- обмеження функціоналу додатку без підписки;

Також важливим аспектом під час розробки застосунку є вибір операційної системи (далі ОС), під яку буде розроблено програмний продукт. Це рішення безпосередньо впливає на охоплення цільової аудиторії, адже одним із ключових факторів є доступність платформи для звичайного користувача. В ідеалі – додаток повинен бути максимально зручним та зрозумілим для широкого кола людей, незалежно від їх технічної обізнаності.

Згідно з результатами опитування, проведеного у 2025 році серед користувачів Інтернету, 94,2% опитаних використовували смартфони для доступу до онлайн-сервісів, тоді як 61,5% обирали ноутбук або комп'ютер. Ці цифри красномовно свідчать про зміну цифрової поведінки: все більше людей обирають мобільні пристрої для виконання повсякденних задач, включно з керуванням особистими даними, зокрема паролями [9].

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

Світові тенденції лише підтверджують цю динаміку. Станом на 2025 рік кількість смартфонів у світі оцінюється приблизно в 7,2 мільярда одиниць, і хоча це не означає, що кожна людина на планеті має власний пристрій, така цифра є потужним індикатором масової доступності та популярності мобільних технологій. За останній рік ринок смартфонів зріс на 3,09%, що вкотре підкреслює його стабільне зростання [10].

Отже, метою кваліфікаційної роботи є розробка сучасного мобільного застосунку типу «Менеджер паролей», який дозволить користувачам зберігати конфіденційні дані у зашифрованому вигляді, забезпечуючи надійний захист за допомогою шифрування та інтуїтивно зрозумілий інтерфейс. В роботі шифрування буде забезпечуватись алгоритмом AES, а інтерфейс буде створено на базі бібліотеки React Native.

Визначена мета обумовлює наступні завдання:

1. Провести аналіз сучасного стану проблеми цифрової безпеки та визначити основні загрози, пов'язані з використанням ненадійних та однакових паролів. Проаналізувати наявні, аналогічні по функціоналу, рішення, визначити їх сильні й слабкі сторони.

2. Обрати оптимальні методи та технології для забезпечення безпеки паролів та розробити алгоритми роботи функціональних блоків мобільного застосунку, що буде об'єднувати весь необхідний функціонал.

3. Реалізувати проєкт мобільного застосунку «Менеджер паролів» що буде об'єднувати механізми шифрування даних, забезпечення локальної авторизації та перевірки збереженого пароля на скомпрометованість.

Висновок до першого розділу.

У першому розділі досліджено проблему витоку паролів яка залишається однією з найгостріших у сучасному цифровому світі. Розглянуто інструмент для перевірки паролів на скомпрометованість. Проведено аналіз аналогів МП, наведено переваги та недоліки до кожного з них. Визначено технічне завдання, яке формує необхідні вимоги до розроблюваного застосунку.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ФУНКЦІОНАЛЬНИХ ЕЛЕМЕНТІВ ЗАСТОСУНКУ «МЕНЕДЖЕР ПАРОЛІВ»

2.1 Аналіз методів шифрування.

З давніх часів людство прагнуло зберігати свої таємниці, захищаючи важливу інформацію від сторонніх очей. Ще задовго до появи комп'ютерів і смартфонів існувала потреба передавати повідомлення так, щоб їх зміст залишався зрозумілим лише адресатові. Одними з найвідоміших історичних методів приховування змісту були шифр Цезаря та шифр Скитала.

Шифр Цезаря – простий, але на той час ефективний метод зміщення літер алфавіту, який використовувався ще в Давньому Римі. Сам Юлій Цезар застосовував його для передачі стратегічно важливої інформації своїм легіонерам під час військових кампаній. Шифр Скитала, своєю чергою, належить до ще давнішого періоду – його використовували спартанці: для зашифрованого листа використовувалась дерев'яна паличка, на яку намотувалась смужка пергаменту, і тільки власник ідентичної скитали міг прочитати послання.

Хоча ці методи сьогодні здаються примітивними, саме вони стали відправною точкою на шляху розвитку криптографії, яка перетворилася з мистецтва приховування повідомлень у точну науку з математичною основою та глобальним значенням.

У XX столітті, особливо під час світових воєн, шифрування стало питанням виживання. Пристрої на зразок «Енігми» змінили уявлення про складність та обсяг захисту даних. Згодом, із появою комп'ютерів і стрімким розвитком цифрових технологій, виникла нова ера криптографії – симетричні та асиметричні алгоритми, хешування, цифрові підписи, протоколи безпечного обміну даними.

У сучасному світі, де конфіденційність стала не просто побажанням, а правом, шифрування – це вже не засіб, а необхідність. І хоча принципи залишилися схожими – зробити повідомлення незрозумілим для стороннього, – сучасні алгоритми працюють із незрівнянно вищим рівнем складності та ефективності.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

Щоб підібрати метод шифрування для нашого застосунку, потрібно проаналізувати які сучасні алгоритми існують, де вони використовуються та основні переваги і недоліки.

Перший метод шифрування який я розглянув був RSA. Він чудово підходить для захисту переданих по мережі даних. Його особливість у тому, що шифрування й дешифрування відбувається за допомогою двох різних ключів – публічного та приватного. Це ідеальний варіант для обміну даними між клієнтом і сервером, де важливо, щоб зломисник не міг розшифрувати повідомлення навіть у разі перехоплення. Утім, для мобільного середовища він занадто ресурсозатратний і повільний, особливо коли мова йде про швидку взаємодію із локальним сховищем.

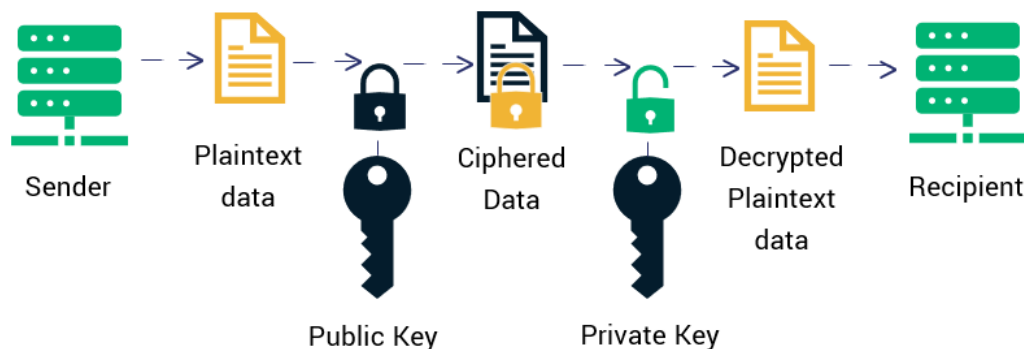


Рисунок 2.1 – Принцип роботи RSA

Blowfish часто використовують у старих файлових системах або для шифрування паролів у базах даних. Він досить швидкий і простий у реалізації, але морально застаріває, особливо на тлі нових вимог до безпеки. Twofish (рисунок 2.2), його наступник, теоретично більш надійний, і хоча його часто розглядають як альтернативу AES, на практиці він рідше використовується – здебільшого через обмежену підтримку в мобільних інструментах.

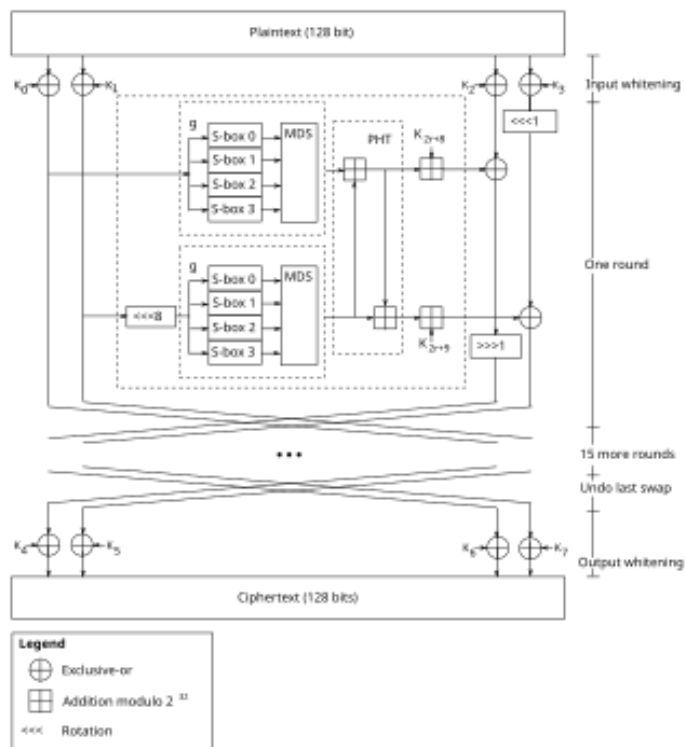


Рисунок 2.2 – Принцип роботи Twofish

Ще є DES (рисунок 2.3), який колись був стандартом, але сьогодні майже не застосовується через критичну вразливість і короткий розмір ключа. Його зараз можна вважати більше історичним прикладом, ніж практичним інструментом.

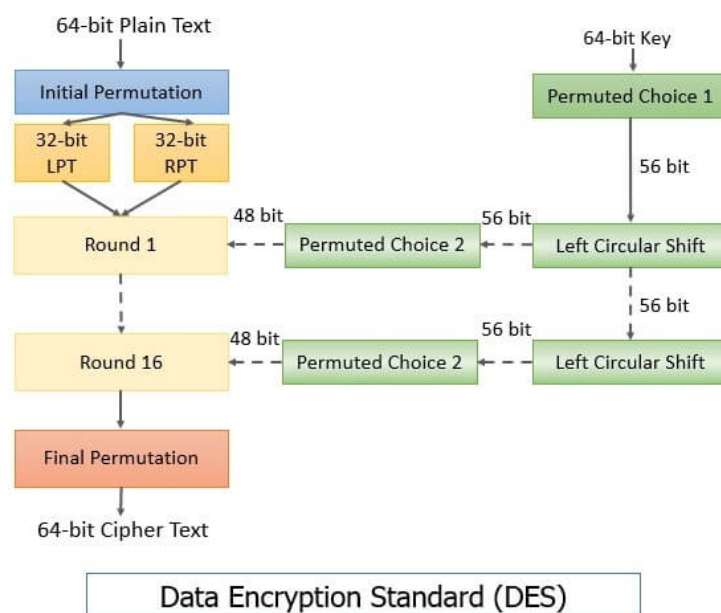


Рисунок 2.3 – Принцип роботи DES

На фоні всіх цих рішень особливо виділяється AES (рисунок 2.4). Цей алгоритм був розроблений як заміна DES, і з того часу став світовим стандартом у галузі локального шифрування, збереження конфіденційної інформації, а також захисту даних у державному та комерційному секторах. Його використовують у всьому: від захисту Wi-Fi (WPA2) до мобільних банківських додатків.

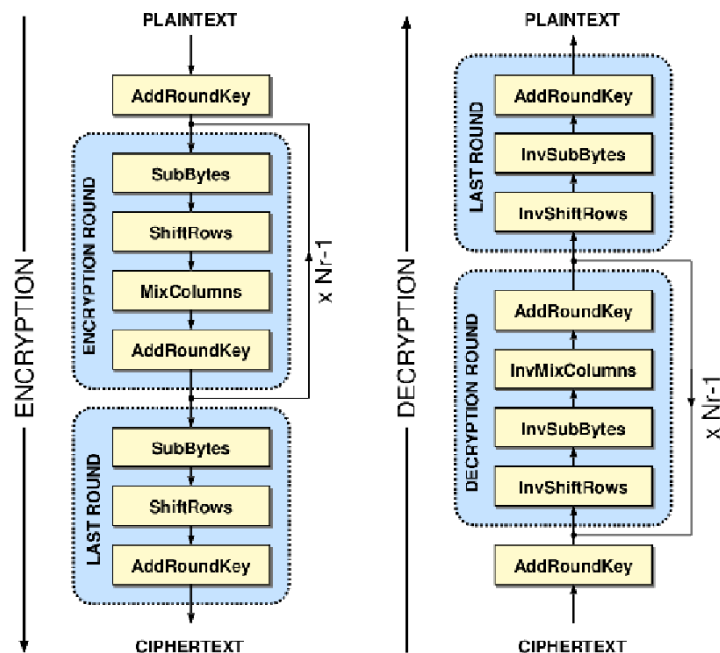


Рисунок 2.4 – Принцип роботи AES

Що робить AES дійсно унікальним – це його баланс між простотою реалізації та складністю зламу. У рамках цієї кваліфікаційної роботи AES було обрано не лише за рекомендацією світових стандартів, а й через його доведену стійкість проти відомих атак, ефективність на мобільних пристроях і високу швидкість [12]. Модель "state-матриці", чітка раундна структура і ідеально збалансоване використання алгебраїчних операцій дозволяють інтегрувати AES навіть у застосунки з мінімальним обчислювальним ресурсом, що добре підійде для інтеграції у мобільний застосунок.

Головні переваги та недоліки описаних методів шифрування

Назва	Переваги	Недоліки
AES	+ висока продуктивність; + надійна безпека, особливо AES-256; + широко підтримується на всіх платформах;	- необхідно безпечно зберігати ключ; - може бути складним для реалізації вручну без бібліотек.
RSA	+ асиметричність, + висока криптостійкість при довгих ключах	- високе споживання ресурсів; - непридатний для шифрування великих обсягів даних
Twofish	+ високий рівень безпеки; + підтримка ключів до 256 біт.	- повільніший за AES; - обмежена підтримка
DES	+ простий у реалізації; + низьке споживання ресурсів.	- застарілий алгоритм; - ключ (56 біт); - вразливий до brute-force атак

2.2 Побудова алгоритмів функціонування модулів застосунку.

Щоб мобільний застосунок працював стабільно та передбачувано, розробку необхідно починати не з написання коду, а з осмислення логіки його роботи. Важливо заздалегідь продумати, як реагуватиме система на дії користувача: від першого запуску до завершення сесії. Такий підхід не лише дозволяє уникнути критичних помилок на етапі розробки, а й суттєво полегшує подальше тестування та підтримку застосунку.

Першим сценарієм, який реалізується при запуску програми, є перевірка наявності зареєстрованого користувача. Система звертається до безпечного локального сховища пристрою, де зберігається хеш пароля. Якщо запис знайдено – користувача перенаправляє на екран авторизації. Якщо ж пароль відсутній, це означає, що користувач новий, і система пропонує пройти реєстрацію (рисунок 2.5).

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		25



Рисунок 2.5 – Алгоритм перевірки чи зареєстрований користувач

Реєстрація реалізована максимально просто й інтуїтивно. Користувач створює чотиризначний PIN-код, використовуючи лише цифри від 0 до 9, після чого повторно вводить його для підтвердження. У разі збігу паролі хешуються та зберігаються в захищеному середовищі пристрою, що дозволяє уникнути витоку даних навіть у разі компрометації файлової системи (рисунок 2.6).



Рисунок 2.6 – Алгоритм реєстрації

Процес входу до застосунку також заслуговує на увагу. Якщо користувач неодноразово вводить неправильний пароль, спрацьовує захист – після трьох невдалих спроб вхід блокується на 30 секунд. Це дозволяє зменшити ризик перебору PIN-кодів злоумисником (рисунок 2.7).

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

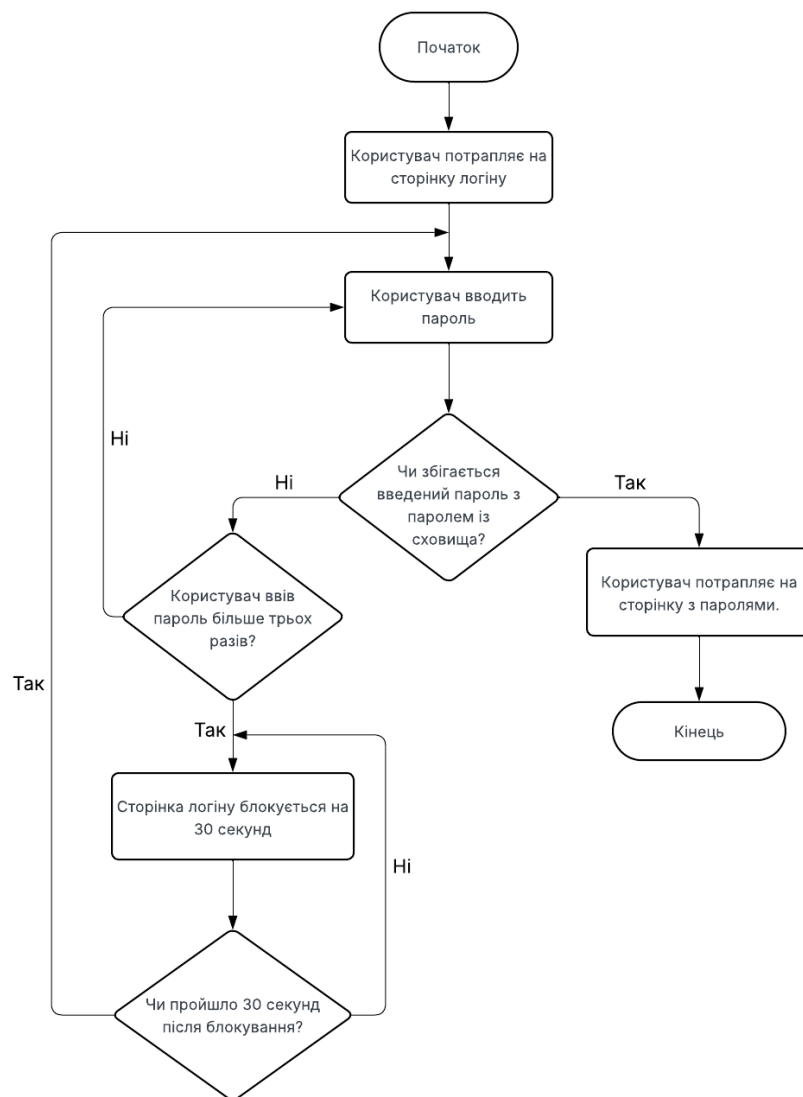


Рисунок 2.7 – Алгоритм логіну

Однією з інноваційних функцій застосунку є перевірка пароля на скомпрометованість за допомогою сервісу НІВР. Алгоритм реалізовано за принципом конфіденційної перевірки: пароль хешується з використанням алгоритму SHA-1, після чого поділяється на дві частини – prefix (перші 5 символів хешу) і suffix (усе інше). На сервер відправляється лише prefix, що дозволяє не розкривати весь хеш. У відповідь сервер повертає список suffix-хешів, які збіглися за префіксом. Якщо серед них виявиться і той, що відповідає нашому паролю – це свідчить про компрометацію, і користувача буде попереджено разом із вказанням кількості виявлень (рисунок 2.8).

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

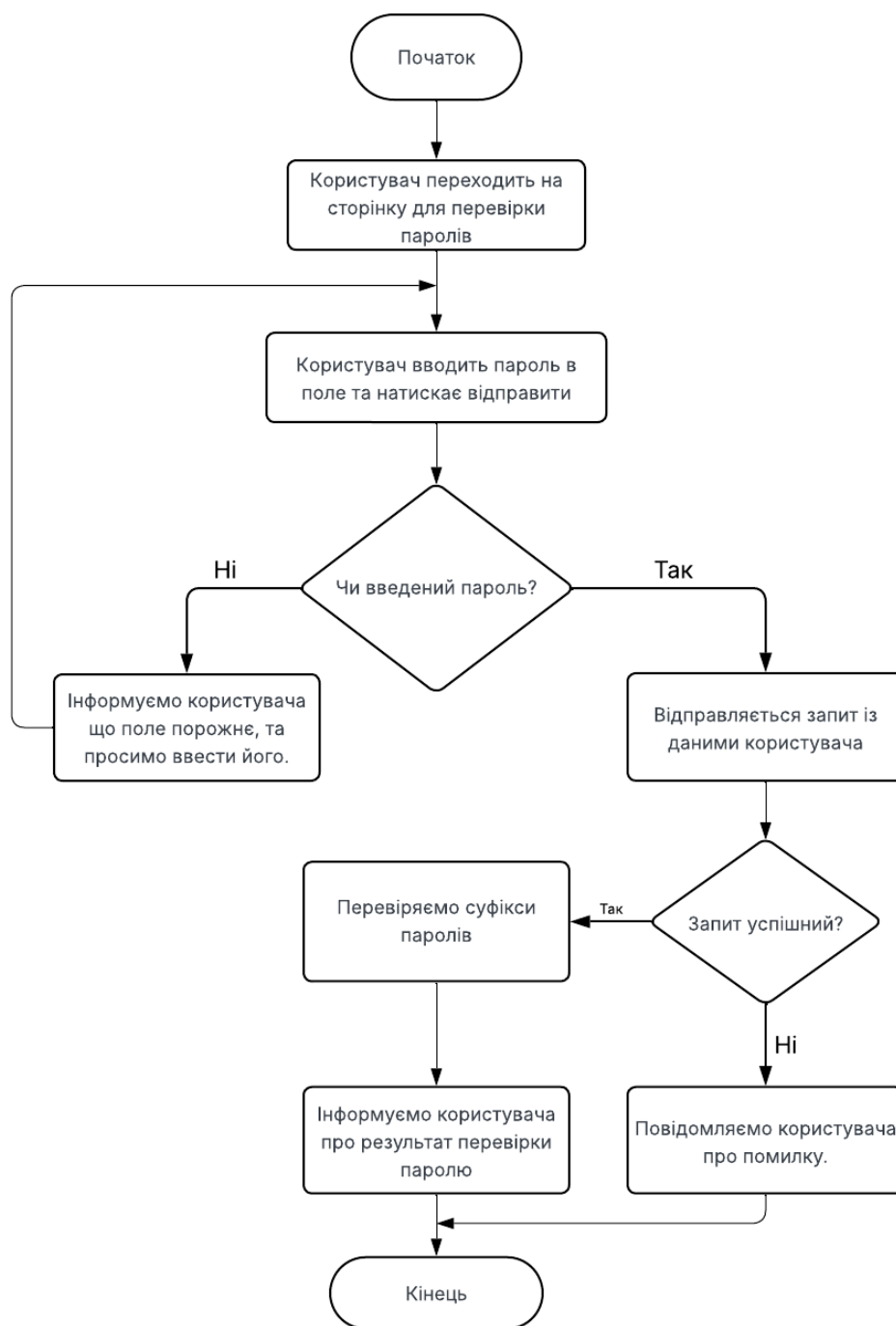


Рисунок 2.8 – Алгоритм перевірки паролів

Збереження паролів користувача в базі також відбувається з максимальною увагою до безпеки. Кожен новий запис спочатку шифрується методом AES, а як ключ використовується поєднання хешованого PIN-коду користувача та дати його реєстрації у мілісекундах. Таким чином, навіть при витоку бази паролі неможливо

дешифрувати без відповідного ключа. Після збереження шифрованих даних відбувається оновлення списку паролів у застосунку (рисунк 2.9).

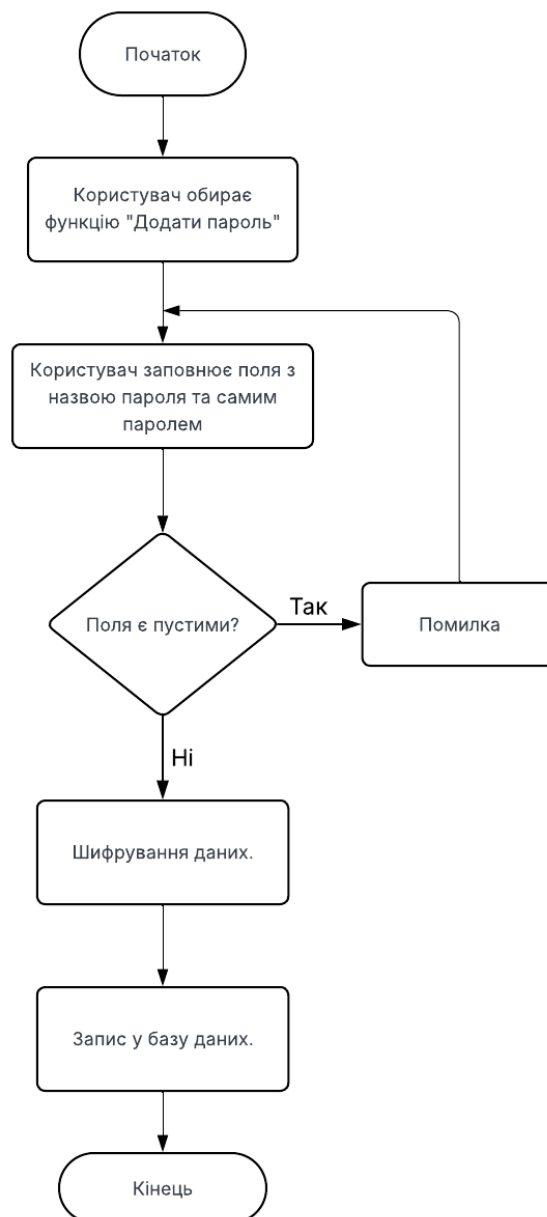


Рисунок 2.9 – Створення нового паролю в додатку

Функція редагування (рисунк 2.10) та видалення (рисунк 2.11) запису із додатку вже давно стали базовими для всіх програм. Такий підхід значно підвищує зручність для користувача. Для безпечної зміни даних додамо підтвердження дії через модальне вікно, що допомагає уникнути випадкових натискань і втрати важливої інформації

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

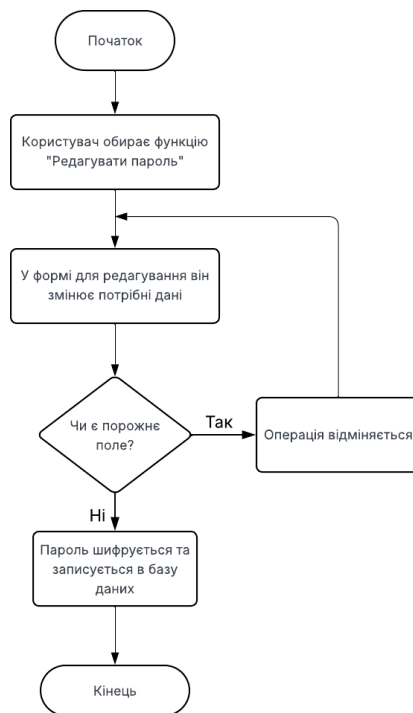


Рисунок 2.10 – Алгоритм редагування пароля

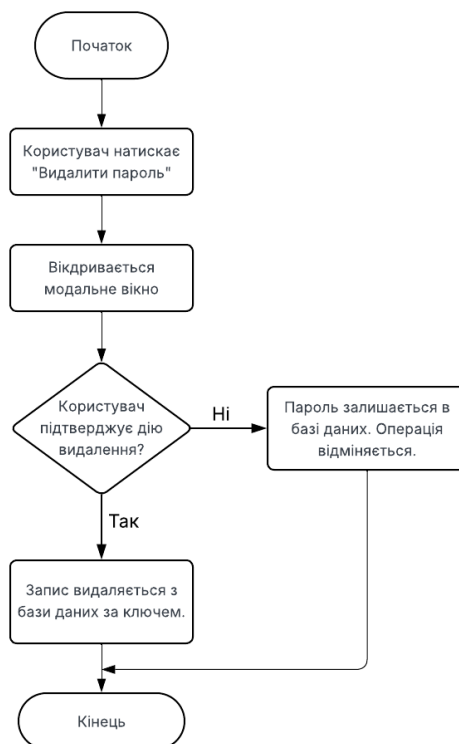


Рисунок 2.11 – Алгоритм видалення пароля

І нарешті, передбачено функцію повного видалення акаунту. Якщо користувач бажає повністю очистити свої дані, застосунок запропонує підтвердити цю дію в модальному вікні. У разі підтвердження видаляються всі дані з локальної бази, а також сам ключ доступу, після чого користувача буде автоматично перенаправлено на сторінку реєстрації (рисунк 2.12). Цей підхід гарантує, що видалення є свідомим і незворотним, і в той же час – простим і зрозумілим.



Рисунк 2.12 – Видалення локального облікового запису

2.3 Аналіз фреймворків для створення мобільних застосунків.

За останні два десятиліття індустрія смартфонів пройшла справжню технічну революцію. Те, що ще зовсім недавно здавалося розкішшю чи фантастикою з наукових фільмів, сьогодні стало звичним атрибутом повсякденного життя. У кишені майже кожної людини – пристрій, здатний на те, що донедавна вимагало

цілої кімнати обладнання. І разом із розвитком «заліза» зростав і софт – застосунки стали розумнішими, складнішими, інтерактивнішими.

Проте одна проблема довгий час залишалась перешкодою для розробників – розмаїття операційних систем. Дві головні ОС, Android і iOS, хоч і домінують на ринку, але мають абсолютно різні архітектури, набори інструментів, середовища розробки та мови програмування. Android живе на Java чи Kotlin, тоді як iOS вимагає знань Swift або Objective-C. Це означає, якщо хочеш додаток для всіх – пиши його двічі. Ідея звучить як виклик, особливо коли мова йде про обмежені ресурси або індивідуальну розробку.

Протягом багатьох років більшість застосунків створювалися саме нативно – тобто окремо для кожної платформи. Але ситуація почала змінюватися. На ринку з'явилося поняття, яке буквально перевернуло підхід до мобільного програмування – кросплатформова розробка. Вона принесла із собою не просто нові інструменти, а нову філософію: один код – багато платформ. Ідея, що колись здавалася утопією, стала реальністю.

Тепер компанії можуть створити додаток і запустити його одразу як для Android, так і для iOS, витрачаючи вдвічі менше часу, грошей і нервів. Кросплатформова розробка стала відповіддю на запит як бізнесу, так і користувачу. Для першим – це економія, для других – зручність і доступність.

У центрі цього руху – фреймворки, які стали фундаментом сучасної кросплатформенності. Серед найвідоміших – React Native, Flutter, Xamarin, Ionic, Cordova, кожна зі своєю історією, стилем і філософією [15].

React Native, створений командою Facebook, сьогодні є одним із найбільш гнучких і поширених рішень. Завдяки йому були розроблені такі гіганти, як Instagram, Facebook Ads, Discord. Його сила – у використанні JavaScript, однієї з найпопулярніших мов програмування, і доступі до нативних компонентів, що дозволяє створювати додатки, які виглядають і працюють як справжні — бо вони і є справжніми з точки зору системи.

Flutter – розробка Google – вирізняється своєю високою продуктивністю. Він компілюється безпосередньо у машинний код і використовує власний графічний

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

рушій, тому інтерфейси виглядають однаково на всіх пристроях. Завдяки Flutter працюють такі додатки, як Google Ads, Reflectly, Alibaba. Проте є один нюанс: Flutter використовує мову Dart, яка є не надто поширеною. Для новачка це означає додаткову криву навчання, а для команди – менше можливостей знайти готові рішення або плагіни, створені спільнотою.

Xamarin, підтримуваний Microsoft, орієнтований на .NET-розробників і дозволяє писати логіку C#, яка потім компілюється у нативний код. Деякі великі корпоративні продукти, зокрема UPS та Outlook, частково реалізовані за допомогою Xamarin. Використання C# є як перевагою так і недоліком. Поза екосистемою .NET, Xamarin виглядає дещо важким, а його інтерфейсні можливості менш гнучкі й відстають за швидкістю розвитку.

Cordova та Ionic – це веб-додатки, загорнуті в мобільну оболонку. Вони дозволяють створювати застосунки за допомогою HTML, CSS і JavaScript. Цей підхід використали такі сервіси, як Sworkit чи Pacifica. Це зручно, якщо потрібно швидко створити простий інтерфейс, але з точки зору продуктивності та нативного вигляду такі рішення часто поступаються.

Після аналізу доступних технологій особливу увагу було приділено фреймворку React Native. Його вибір зумовлений насамперед тим, що він дозволяє створювати повноцінні мобільні застосунки з єдиною кодовою базою, яка успішно компілюється під обидві ОС. Це дає змогу суттєво скоротити час розробки. Крім того, фреймворк має добре розвинену екосистему та велику спільноту розробників. Це дає можливість використовувати численні перевірені бібліотеки, зокрема для роботи з локальним сховищем і шифруванням, що суттєво полегшує реалізацію функціоналу та зменшує ризики, пов'язані з потенційними помилками у безпекових механізмах. Також на користь React Native вплинув наявний особистий досвід автора у використанні JavaScript та бібліотеки React. Це дозволило швидше перейти до реалізації основної логіки застосунку, не витрачаючи час на освоєння нової мови чи незнайомої платформи.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Висновок до другого розділу.

У другому розділі було проведено комплексний аналіз методів шифрування, з метою визначення найбільш ефективного та безпечного підходу для реалізації функціоналу захисту даних у мобільному застосунку. Окрему увагу приділено проектуванню алгоритмів функціонування застосунку. Їх створення надало змогу заздалегідь проаналізувати логіку взаємодії користувача з системою та уникнути типових помилок на етапі розробки й тестування.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОЄКТУ.

3.1 Реалізація алгоритмів та функціональних блоків.

Роботу над проєктом було розпочато з реалізації механізму перевірки чи зареєстрований користувач (рисунок 2.5) – адже безпека є основним принципом цього додатку. Головна мета на цьому етапі – правильно налаштувати логіку переходу між екранами в залежності від стану користувача. Передусім потрібно визначити, чи взагалі користувач вже створений. Якщо ні – він автоматично перенаправляється на екран реєстрації. У випадку, якщо користувач існує, але ще не пройшов автентифікацію, його чекає сторінка входу. Тільки після успішної автентифікації відкривається доступ до основного функціоналу.

```
17 const isAuth = useAuthStore((state) => state.isAuth); // перевірка чи пройшов користувач автентифікацію
18 const isRegister = SecureStore.getItem(STORE_PSWD_KEY as string); // перевіряємо чи наявний PIN-код в сховищі
19
20 if (!isRegister) return <Redirect href="/(auth)/register" />;
21
22 if (!isAuth) return <Redirect href="/(auth)/login" />;
23
```

Рисунок 3.1 – Перевірка чи зареєстрований користувач

Після цього переходимо до реалізації екрану реєстрації. Для обробки логіки використаємо хук `useEffect`, який `реагує` на зміни у змінних `password` та `confirmPassword`. Як тільки користувач вводить чотири символи – це є умовою завершення введення – перевіряємо, чи обидва введені значення збігаються. Якщо все в порядку, зберігаємо пароль у захищене сховище (`SecureStore`), відзначаємо, що користувач авторизований, і переводимо його на головну сторінку. У разі, якщо введені значення не збігаються, обнуляємо поля вводу і показуємо модальне вікно з повідомленням про помилку (рисунок 3.2).

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

```

16      useEffect(() => {
17          if (
18              password.length !== LEN_PASSWORD ||
19              confirmPassword.length !== LEN_PASSWORD
20          )
21              return;
22
23          if (!handleRegisterApp(password, confirmPassword)) {
24              setPassword("");
25              setConfirmPassword("");
26
27              return;
28          }
29
30          setAuth(true);
31
32          router.replace("/(tabs)");
33      }, [password, confirmPassword]);

```

Рисунок 3.2 – Процес реєстрації

На етапі запуску застосунку одним із перших завдань, яке необхідно виконати, є перевірка наявності локальної бази даних. У випадку, якщо база вже була створена раніше, застосунок безперешкодно отримує до неї доступ і переходить до основного функціоналу. В іншому випадку система самостійно ініціює створення нової бази даних з необхідними полями (рисунок 3.3).

```

31      const createDbIfNeeded = async (db: SQLiteDatabase) => {
32          await db.execAsync(
33              "CREATE TABLE IF NOT EXISTS password (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, password TEXT, passwordLength INTEGER);"
34          );
35
36          console.log("Creating db if needed");
37      };

```

Рисунок 3.3 – Функція для перевірки наявності бази даних

Для підтримки чистоти коду та спрощення подальшої роботи з базою даних винесемо основні операції у спеціальний хук. Такий підхід дозволяє централізувати всі запити, зменшити дублювання логіки та зробити код більш структурованим і підтримуваним. У цьому хуку реалізовано ключові функції, які забезпечують повний цикл взаємодії з локальною базою даних: отримання всіх записів, отримання одного запису за ключовим полем, редагування та видалення записів (рисунок 3.4).

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

```
const deletePassword = (id: number) => {
  db.runAsync("DELETE FROM password WHERE id = ?;", [id]);
};
```

Рисунок 3.4 – Видалення паролю з бази за id

Ще один важливий елемент є перевірка паролів на надійність. Для цього потрібно написати функцію, яка буде: перевіряти довжину пароля, наявність великих, малих літер, а також спецсимволи та цифри. Користувач може самостійно змінювати вимоги до паролю. Наприклад: вимкнути обов'язкову наявність спеціальних символів або змінити мінімальну довжину (рисунок 3.5).

```
14 export const validatePassword = (
15   password: string,
16   rules: PasswordRules = {
17     minLength: "8",
18     requireUppercase: true,
19     requireLowercase: true,
20     requireDigit: true,
21     requireSpecialChar: true,
22   }
23 ): PasswordValidationResult => {
24   const errors: string[] = [];
25
26   if (rules.minLength && password.length < Number(rules.minLength)) {
27     errors.push(`Пароль має містити щонайменше ${rules.minLength} символів.`);
28   }
29
30   if (rules.requireUppercase && ![A-Z]/.test(password)) {
31     errors.push("Пароль має містити хоча б одну велику літеру.");
32   }
33
34   if (rules.requireLowercase && ![a-z]/.test(password)) {
35     errors.push("Пароль має містити хоча б одну малу літеру.");
36   }
37
38   if (rules.requireDigit && ![0-9]/.test(password)) {
39     errors.push("Пароль має містити хоча б одну цифру.");
40   }
41
42   if (
43     rules.requireSpecialChar &&
44     !/[!@#$%^&*()_+{} \[\];'<>.,?/~\|~]/.test(password)
45   ) {
46     errors.push("Пароль має містити хоча б один спеціальний символ.");
47   }
48 }
```

Рисунок 3.5 – Перевірка пароля на надійність

У сучасному цифровому середовищі користувачам дедалі частіше пропонують створити обліковий запис навіть для другорядних сервісів, якими вони не користуватимуться регулярно. У такі моменти вигадати новий, складний і водночас унікальний пароль стає проблемою – саме тому багато хто вдається до повторного використання вже існуючих паролів. Це створює реальну загрозу

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

безпеці, адже компрометація одного облікового запису може відкрити доступ до всіх інших. Щоб цього уникнути, у застосунку реалізовано функцію генерації паролів, яка допомагає користувачу швидко створити надійний пароль відповідно до заданих критеріїв. Користувач сам визначає правила генерації – встановлює довжину, дозволяє або вимикає великі й малі літери, цифри, спеціальні символи (рисунок 3.6).

```

9  export const generatePassword = ({
10    minLength = "6",
11    requireUppercase = true,
12    requireLowercase = true,
13    requireDigit = true,
14    requireSpecialChar = true,
15  }: PasswordRules): string => {
16    let characterPool = "";
17    let requiredChars: string[] = [];
18
19    if (requireUppercase) {
20      characterPool += UPPERCASE;
21
22      requiredChars.push(UPPERCASE[Math.floor(Math.random() * UPPERCASE.length)]);
23    }
24    if (requireLowercase) {
25      characterPool += LOWERCASE;
26
27      requiredChars.push(LOWERCASE[Math.floor(Math.random() * LOWERCASE.length)]);
28    }
29    if (requireDigit) {
30      characterPool += NUMBERS;
31
32      requiredChars.push(NUMBERS[Math.floor(Math.random() * NUMBERS.length)]);
33    }
34    if (requireSpecialChar) {
35      characterPool += SYMBOLS;
36
37      requiredChars.push(SYMBOLS[Math.floor(Math.random() * SYMBOLS.length)]);
38    }

```

Рисунок 3.6 – Генерація пароля

У контексті застосунку, який працює з чутливою інформацією, зокрема з паролями, надзвичайно важливо забезпечити їхній захист не лише під час введення, а й у процесі зберігання. Звичайне збереження паролів у відкритому вигляді неприпустиме, оскільки навіть мінімальний витік даних може мати серйозні наслідки для безпеки користувача. Саме тому було реалізовано механізм шифрування перед тим, як паролі потрапляють до бази. Завдяки цьому навіть у випадку фізичного доступу до пристрою або до файлів бази, стороння особа не зможе отримати жодної корисної інформації (рисунок 3.7).

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

```

80 const createPassword = async (name: string, password: string) => {
81   if (!secretKey) {
82     return showErrorSecretKeyEmpty();
83   }
84
85   try {
86     const encryptPassword = CryptoJS.AES.encrypt(
87       password,
88       secretKey
89     ).toString();
90
91     await db.runAsync(
92       "INSERT INTO password (name, password, passwordLength) VALUES (?, ?, ?);",
93       [name, encryptPassword, password.length]
94     );
95
96     Toast.show({
97       text1: "Успішно!",
98       text2: "Пароль успішно збережено",
99       type: "success",
100    });
101  } catch (error) {
102    console.error(error);
103
104    Toast.show({
105      text1: "Помилка!",
106      text2: "Щось пішло не так :(\n Будь-ласка, спробуйте пізніше.",
107      type: "error",
108    });
109  }
110 };

```

Рисунок 3.7 – Функція зберігання зашифрованого паролю

Останню функцію яку ми розглянемо є блокування входу після трьох невдалих спроб на 30 секунд. Таймер зберігається після перезавантаження додатку. На рисунку 3.8 лише частина функції блокування в якій ми перевіряємо чи сходиться пароль, якщо так, перенаправляємо користувача на головну сторінку. В іншому випадку записуємо надалу спробу і якщо їх більше трьох – зберігаємо час невдалої спроби та блокуємо введення.

```

if (hashPassword === hashStorePassword) {
  await SecureStore.deleteItemAsync("ATTEMPTS");
  await SecureStore.deleteItemAsync("LAST_ATTEMPT");
  setAuth(true);
  return router.replace("/(tabs)");
}

Toast.show({ ...
});

setPassword("");

let attempts = await SecureStore.getItemAsync("ATTEMPTS");
const currentAttempts = attempts ? parseInt(attempts) + 1 : 1;

if (currentAttempts >= MAX_ATTEMPTS) {
  await SecureStore.setItemAsync("LAST_ATTEMPT", Date.now().toString());
  setIsLocked(true);
  setTimeLeft(LOCK_DURATION);
}

await SecureStore.setItemAsync("ATTEMPTS", currentAttempts.toString());
})();

```

Рисунок 3.8 – Функція блокування входу

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

3.2 Налаштування та перевірка функціоналу.

При першому запуску додатку нас зустрічає сторінка реєстрації. На цій сторінці користувач вводить пароль, який буде використовуватися при логіні в додаток. Також цей пароль використовується як частина ключа для шифрування паролів, які будуть додані в базу даних (рисунок 3.9).

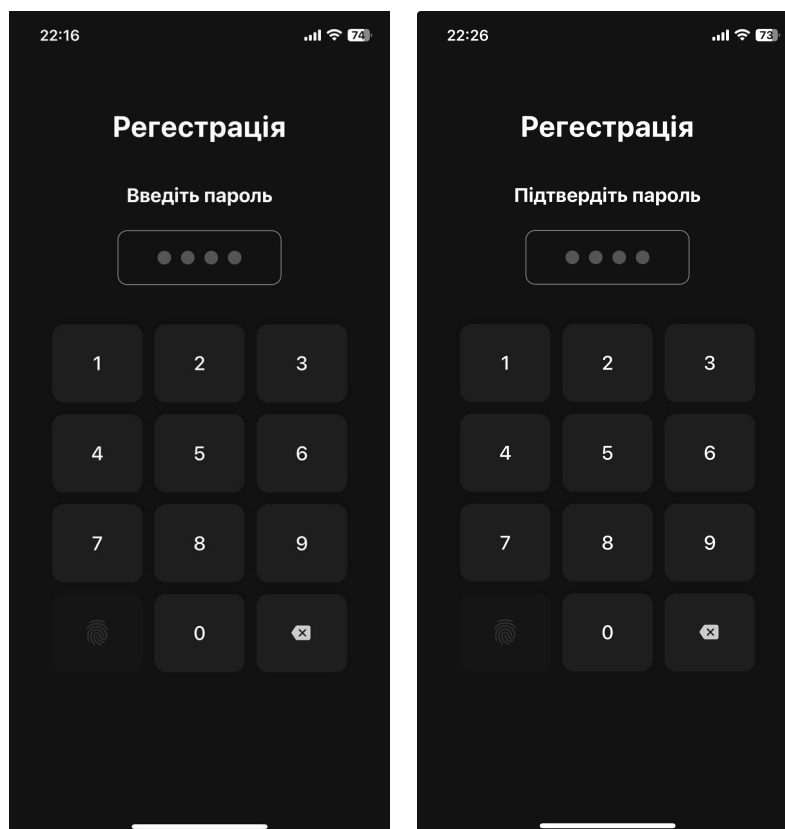


Рисунок 3.9 – Сторінка реєстрації

Після успішної реєстрації ми потрапляємо на головну сторінку. На цій сторінці користувач може додавати нові паролі, видаляти паролі та змінювати їх. Також в нижній частині екрану з'являється поле навігації. На елементу навігації присутні 3 сторінки: головна, перевірка паролів на компрометованість та налаштування (рисунок 3.10).

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

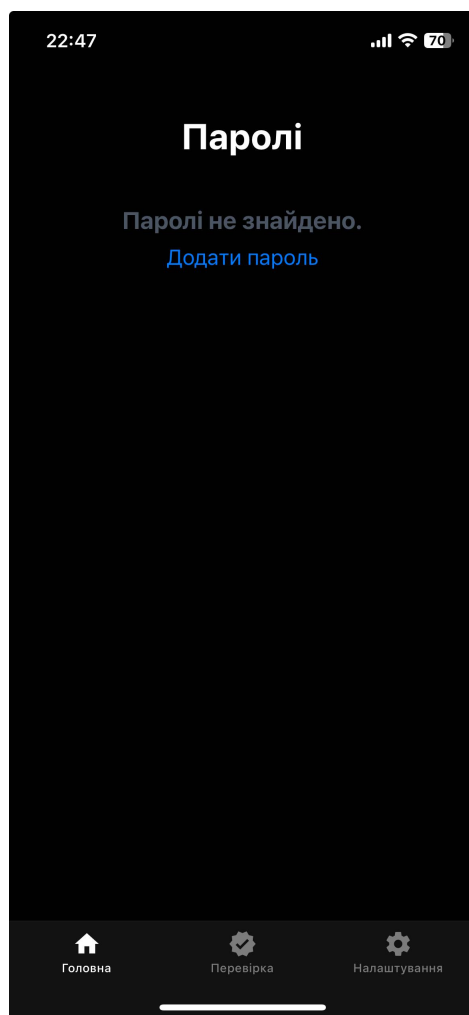


Рисунок 3.10 – Головна сторінка

Сторінка «Створити пароль» та «Змінити пароль» є одними з ключових в застосунку. На цій сторінці є поле для назви паролю, та для пароля. При введенні пароля він замінюється на крапки задля більшої безпеки. Для того щоб переглянути введений пароль потрібно натиснути на кнопку з оком. Також на сторінці є кнопка «Згенерувати пароль». При натисненні на цю кнопку буде створений новий пароль. Якщо в полі з паролем вже було якесь значення, воно автоматично заміниться. Також на сторінці будуть з'являтися помилки, які будуть свідчити про те, що пароль не задовільняє правила надійності. Їх завжди можна змінити на сторінці «Налаштування». Після заповнення всіх полів потрібно натиснути кнопку «Зберегти», щоб пароль зберігся в базі даних, або ж «Скасувати», щоб очистити поля та повернутися на головну, пароль при цьому не запишеться в базу. Сторінка «Зміна пароля» має аналогічний вигляд (рисунок 3.11).

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

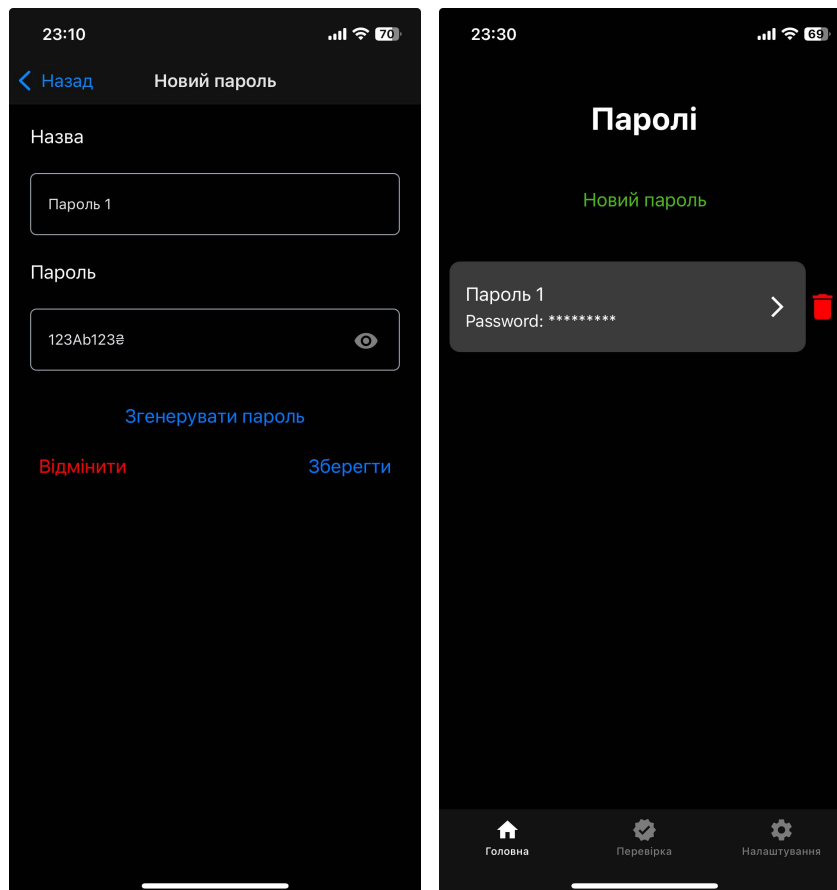


Рисунок 3.11 – Сторінка створення пароля. Та результат створення

Сторінка «Налаштування» у застосунку виконує роль швидкого доступу до ключових функцій, які впливають на безпеку та керування обліковим записом. Серед трьох доступних кнопок — «Надійність пароля», «Видалити акаунт» та «Вихід» — лише остання є інтуїтивно зрозумілою для більшості користувачів, тоді як перші дві можуть викликати додаткові запитання. Кнопка «Видалити акаунт» — це не просто вихід із профілю, а повне очищення локальної бази даних, видалення ключів доступу та повернення застосунку до стану «нового користувача». Такий функціонал вимагає обережності, тож для запобігання випадковим діям було реалізовано модальне вікно з коротким описом наслідків (рисунок 3.12).

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

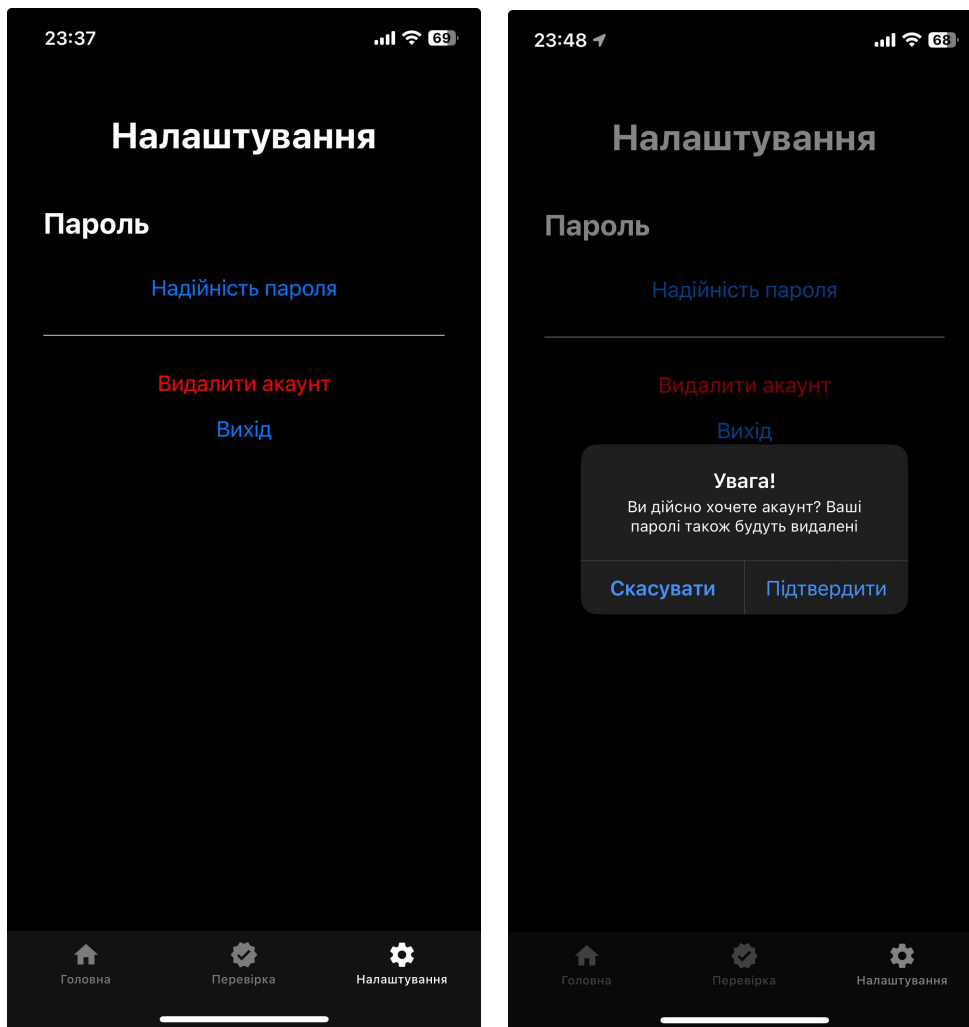


Рисунок 3.12 – Сторінка налаштування.

Та модальне вікно кнопки «Видалити акаунт»

Сторінка «Надійність пароля» у застосунку призначена для гнучкого налаштування правил, за якими будуть перевірятися паролі. Особливість полягає в тому, що ці самі правила застосовуються не лише до перевірки, а й до генерації паролів у вбудованому генераторі (рисунок 3.13).

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

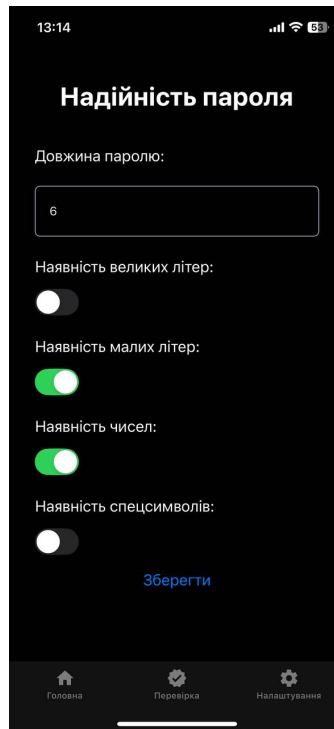


Рисунок 3.13 – Сторінка «Надійність пароля»

Важливо щоб поле з довжиною пароля не було пустим та було обрано хоча б одне правило (рисунок 3.14).

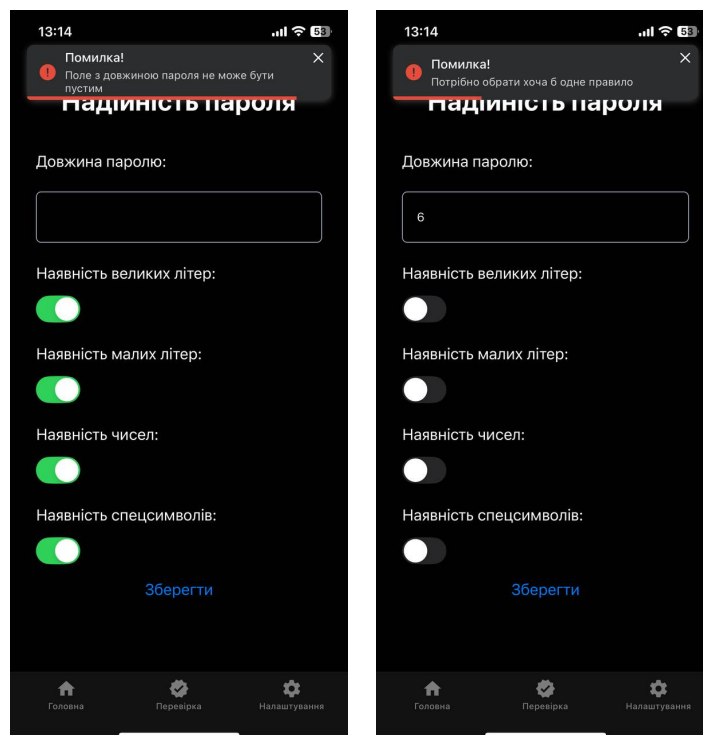


Рисунок 3.14 – Обробка помилок на сторінці «Надійність пароля»

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3 Реалізація функціоналу перевірки паролів на скомпрометованість.

Одним із важливих аспектів безпеки паролів є перевірка їх на наявність у відомих витоках даних. Для реалізації цієї функціональності в було інтегровано API сервісу НІВР, який надає відкритий доступ до бази даних скомпрометованих паролів. Це дозволяє користувачам перевірити, чи не був введений ними пароль раніше викрадений і оприлюднений унаслідок зламів різноманітних онлайн-сервісів (рисунок 3.15).

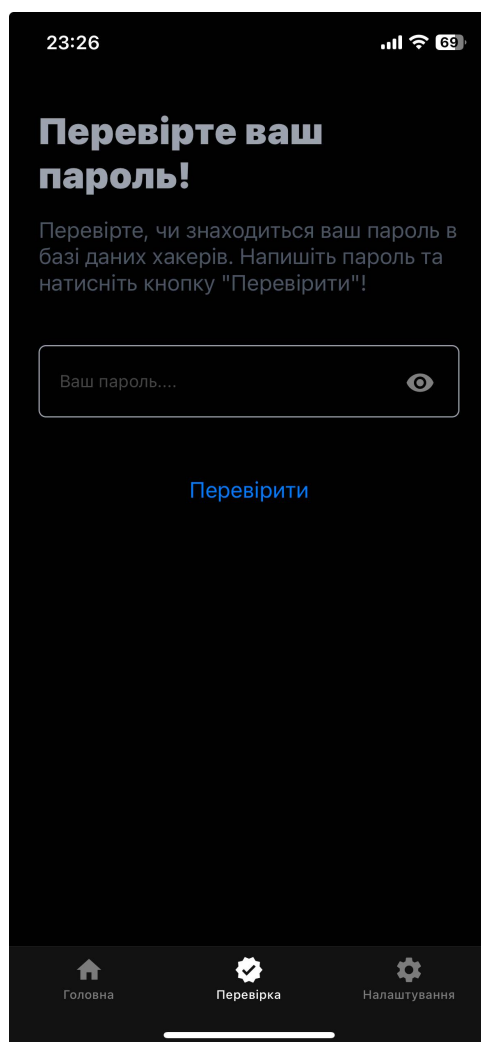


Рисунок 3.15 – Сторінка для перевірки паролів на скомпрометованість.

Особливістю даного API є застосування моделі конфіденційності, відомої як k-анонімність, що дозволяє перевіряти пароль, не передаючи його повністю на сторонній сервер. Це суттєво знижує ризики витоку чутливих даних навіть у

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

процесі перевірки. В процесі реалізації застосунку було розроблено функцію, яка хешує пароль за алгоритмом SHA-1, після чого розділяє отриманий хеш на дві частини: префікс із перших п'яти символів та суфікс – решта (рисунок 3.16).

```
const hash = SHA1(checkPassword).toString().toUpperCase();
const prefix = hash.slice(0, 5);
const suffix = hash.slice(5);

const response = await fetch(
  `https://api.pwnedpasswords.com/range/${prefix}`
);
```

Рисунок 3.16 – Хешування пароля та отримання префіксу та суфіксу

До API надсилається лише префікс, у відповідь сервер повертає список можливих відповідників, серед яких уже локально виконується пошук повного збігу суфіксу. Якщо такий збіг знаходиться, користувач отримує повідомлення з інформацією про кількість разів, коли даний пароль було виявлено у витоках (рисунок 3.17).

```
const dataLines = (await data).split("\r\n");

const match = dataLines.find((dataLine) => {
  const [dataSuffix] = dataLine.split(":");

  return dataSuffix === suffix;
});

if (match) {
  const [, count] = match.split(":");

  return Alert.alert(
    "Увага!",
    `Ваш пароль було злито ${count} разів. Рекомендуємо замінити його`
  );
}
```

Рисунок 3.17 – Локальний пошук збігу суфіксів

У процесі аналізу відповіді, що надходить від сервера НІВР, важливо враховувати характер самої бази даних, з якою працює застосунок. Одна й та сама комбінація символів, тобто пароль, може бути використана багатьма різними користувачами або ж одним користувачем для різних сервісів. Це звична практика, попри численні застереження фахівців з інформаційної безпеки. Як наслідок, при витоках з різних платформ той самий пароль може потрапляти в базу даних неодноразово. Саме тому сервіс НІВР не обмежується бінарною відповіддю, злитий чи не злитий пароль, а додатково вказує кількість виявлень відповідного хешу в базі даних. Цей підхід дозволяє користувачеві не лише дізнатись факт компрометації, а й оцінити ступінь поширеності пароля серед витоків, що дає змогу прийняти більш обґрунтоване рішення щодо його подальшого використання (рисунок 3.18).

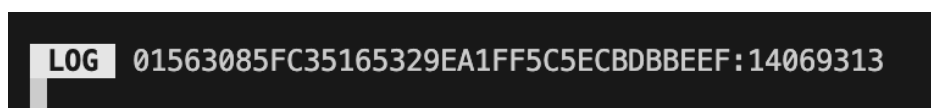


Рисунок 3.18 – Приклад відповіді серверу

На рисунку 3.18 зображений приклад відповіді серверу. 01563085FC35165329EA1FF5C5ECBDBBEEF – це суфікс SHA-1 хешу пароля, який був виявлений у базі даних. Число після двокрапки – 14069313 – вказує на кількість випадків, коли саме цей хеш зустрічався у злитих масивах даних. Після аналізу відповіді серверу, з'являється модальне вікно в якому є інформація про кількість збігів в базі (рисунок 3.19).

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

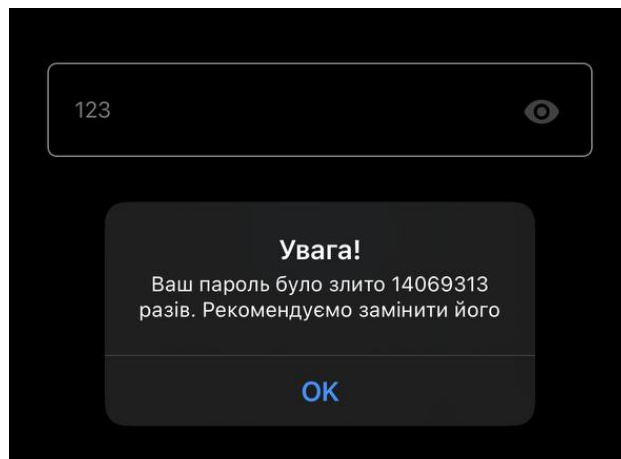


Рисунок 3.19 – Результат перевірки пароля «123»

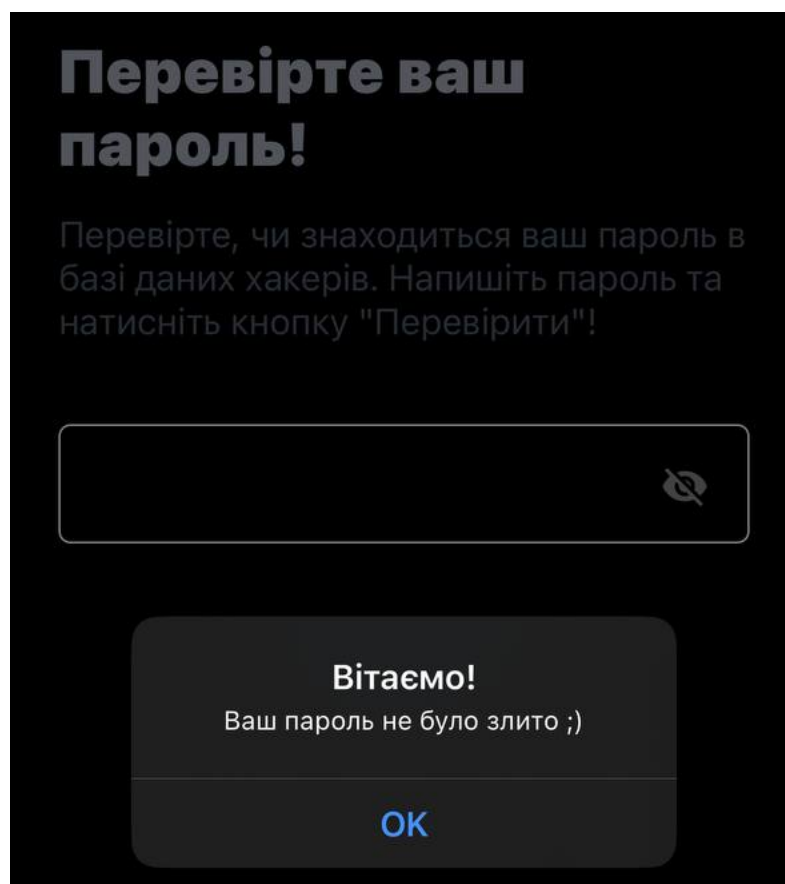


Рисунок 3.20 – Результат перевірки пароля якщо пароля не знайдено в базі

3.4 Аналіз отриманих результатів та подальших досліджень.

У результаті виконання кваліфікаційної роботи було розроблено повнофункціональний мобільний застосунок для керування паролями. Основний акцент було зроблено на безпеку, зручність користування та дотримання сучасних стандартів захисту даних.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

Розроблений застосунок реалізує механізм локального збереження даних що дає змогу повністю уникати використання зовнішніх баз даних. Це рішення особливо актуальне в контексті збереження чутливої інформації, оскільки користувачі повністю контролюють свої дані й не залежать від надійності сторонніх серверів. Збереження паролів реалізовано із застосуванням шифрування методом AES, що є одним із найнадійніших на сьогодні алгоритмів симетричної криптографії. Усі облікові дані шифруються ще до збереження, а ключ зберігається лише на пристрої користувача, що виключає можливість доступу до інформації сторонніми особами навіть у разі фізичного доступу до пристрою.

Інтеграція із сервісом НІВР відкрила новий рівень функціональності застосунку — він не лише зберігає й генерує паролі, а й активно інформує користувача про їх потенційну скомпрометованість. Таким чином, застосунок виконує не лише зберігаючу, але й профілактичну роль у захисті даних.

Особливу увагу було приділено реалізації функціоналу генерації паролів. Вбудований генератор дозволяє створювати надійні паролі із можливістю врахування кількох важливих критеріїв: наявність великих та малих літер, чисел, спеціальних символів, а також відповідної довжини.

В майбутньому можна додати функції експорту та імпорту паролів, щоб користувачі мали можливість легко, швидко та головне безпечно передавати свої дані іншим користувачам.

Висновок до третього розділу.

У третьому розділі було розроблено повнофункціональний застосунок з описом логіки та демонстрацією роботи додатку. Окрему увагу приділено інтеграції з API НІВР, яке дозволило реалізувати функцію перевірки паролів на скомпрометованість. В кінці розділу проаналізував отримані результати та можливий майбутній функціонал, який можна додати в застосунок.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Проблематика цифрової безпеки, зокрема загроза, пов'язана з використанням ненадійних та повторюваних паролів, виявилася особливо актуальною в умовах постійного зростання кількості кібератак і витоків конфіденційної інформації.

Кваліфікаційна робота присвячена дослідженню методів та засобів безпечного зберігання, управління паролів користувача. В ході роботи було розроблено проєкт мобільного застосунку «Менеджер паролів» у якому користувач зможе безпечно зберігати, управляти та перевіряти свої паролі на скомпрометованість.

У ході дослідження було здійснено ґрунтовний аналіз актуального стану проблеми безпеки паролів, визначено ключові загрози та чинники, що впливають на захист автентифікаційної інформації користувачів. Значну увагу приділено огляду та оцінці існуючих програмних рішень у сфері управління паролями – їхньої функціональності, переваг і обмежень. На основі проведеного аналізу було сформульовано технічне завдання, що окреслює вимоги до розроблюваного програмного забезпечення.

У результаті пошуку технічних підходів до реалізації системи зосереджено увагу на підборі ефективних методів шифрування та побудові логіки взаємодії користувача з мобільним застосунком. Проведено аналіз сучасних криптографічних алгоритмів з урахуванням їхньої ефективності в умовах обмежених ресурсів мобільного середовища, що дозволило обґрунтувати вибір оптимального варіанта для реалізації.

На завершальному етапі реалізовано повнофункціональний мобільний застосунок із підтримкою локальної автентифікації, генерації надійних паролів та шифрування збережених даних. До ключових функціональних можливостей також віднесено інтеграцію з сервісом API NIBP, що забезпечує перевірку паролів на предмет скомпрометованості. Проведено аналіз результатів реалізації та окреслено перспективи подальшого розвитку програмного продукту.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

У межах цієї кваліфікаційної роботи було реалізовано повнофункціональний мобільний застосунок для управління паролями, розроблений з урахуванням актуальних вимог до безпеки, автономності зберігання даних та зручності користування. Особливу увагу зосереджено на забезпеченні локального збереження облікової інформації без залучення сторонніх серверів, що дозволяє користувачеві повністю контролювати свої дані та мінімізувати ризики, пов'язані з витоками. Захист даних реалізовано за допомогою надійного алгоритму AES, при якому ключі зберігаються виключно на пристрої, а шифрування відбувається до моменту запису. Вбудований генератор паролів дозволяє формувати стійкі комбінації відповідно до обраних критеріїв, а інтеграція з сервісом НІВР забезпечує можливість перевірки паролів на компрометацію, що розширює функціональність застосунку і надає йому профілактичну роль у сфері кіберзахисту. Отримані результати демонструють доцільність такого підходу, а також визначають потенціал для подальшого розвитку – зокрема, впровадження функцій безпечного обміну даними через експорт та імпорт.

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. RockYou2024: 10 billion passwords leaked in the largest compilation of all time. *Cybernews*. URL: <https://cybernews.com/security/rockyou2024-largest-password-compilation-leak/> (дата звернення: 29.04.2025).

2. Автентифікація (аутентифікація): що це, методи та особливості. *Zen*. URL: <https://www.zen.com/ua/blog/personal-finance-uk/authentication-definition-methods-features/> (дата звернення: 29.04.2025).

3. 10 найпотужніших кібератак та витоків даних за всю історію. *Gigatrans*. URL: <https://gigatrans.ua/ua/news/10-naypotuzhn-shih-k-beratak-ta-vitok-v-danih-zavsyu-stor-yu> (дата звернення: 30.04.2025).

4. 2012 Linkedin Breach had 117 Million Emails and Passwords Stolen, Not 6.5M. *Trendmicro*. URL: <https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/2012-linkedin-breach-117-million-emails-and-passwords-stolen-not-6-5m> (дата звернення: 30.04.2025).

5. About LastPass. *Hellip*. URL: <https://hellip.com/ua/product/lastpass.html> (дата звернення: 01.05.2025).

6. Інструкція з налаштування 1Password. *Hackyourmom*. URL: <https://hackyourmom.com/pryvattist/instrukciya-z-nalashtuvannya-1password/> (дата звернення: 01.05.2025).

7. Dashlane – Надійний менеджер паролів і захист цифрової ідентичності. *Hellip*. URL: <https://hellip.com/ua/product/dashlane.html> (дата звернення: 02.05.2025).

8. Help:Two-factor authentication. *Wikipedia*. URL: https://en.wikipedia.org/wiki/Help:Two-factor_authentication (дата звернення: 02.05.2025).

9. Digital 2025: device trends for 2025. *Datareportal*. URL: <https://datareportal.com/reports/digital-2025-sub-section-device-trends> (дата звернення: 04.05.2025).

10. How Many People Own Smartphones? (2024-2029). *Explodingtopics*. URL: <https://explodingtopics.com/blog/smartphone-stats> (дата звернення: 04.05.2025).

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

11. Що таке AES-шифрування?. *Memory*. URL: [https://memory.net.ua/info/aes-shifruvannja?srsltid=AfmBOood8kmpU-h5j1-](https://memory.net.ua/info/aes-shifruvannja?srsltid=AfmBOood8kmpU-h5j1-uB4IZ9IWFjwNnHdrtSBNMCL365bHeWQ3GW46y)

[uB4IZ9IWFjwNnHdrtSBNMCL365bHeWQ3GW46y](https://memory.net.ua/info/aes-shifruvannja?srsltid=AfmBOood8kmpU-h5j1-uB4IZ9IWFjwNnHdrtSBNMCL365bHeWQ3GW46y) (дата звернення: 05.05.2025).

12. Twofish Encryption Algorithm. *Geeksforgeeks*. URL: <https://www.geeksforgeeks.org/twofish-encryption-algorithm/> (дата звернення: 06.05.2025).

13. Найкращі фреймворки для кросплатформної розробки. *Wezom*. URL: <https://wezom.com.ua/ua/blog/luchshie-frejmworki-dlja-krossplatformennoj-razrabotki-mobilnyh-prilozhenij> (дата звернення: 08.05.2025).

14. API HIBP. *Haveibeenpwned*. URL: <https://haveibeenpwned.com/API/v3> (дата звернення: 10.05.2025).

15. Expo SQLite. *Expo*. URL: <https://docs.expo.dev/versions/latest/sdk/sqlite/> (дата звернення: 12.05.2025).

16. Expo SecureStore. *Expo*. URL: <https://docs.expo.dev/versions/latest/sdk/securestore/> (дата звернення: 13.06.2025).

17. Що таке витоки даних, причини, наслідки та захист інформації. *Hackyourmom*. URL: <https://hackyourmom.com/osvita/shho-take-vytoky-danyh-prychyny-naslidky-ta-zahyst-informacziyi/> (дата звернення: 08.06.2025).

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

Валідація пароля

00:02

32

Назад

Змінити пароль

Назва

Пароль 1

Пароль

Пароль має містити щонайменше 4 символів.

Пароль має містити хоча б одну велику літеру.

Пароль має містити хоча б одну малу літеру.

Пароль має містити хоча б одну цифру.

Пароль має містити хоча б один спеціальний символ.

Згенерувати пароль

Відмінити

Зберегти

Генерація паролів

01:06
Камера

Назад Новий пароль

Назва

Генерація пароля

Пароль

S7D!2M

Згенерувати пароль

Відмінити Зберегти

Й ц у к е н г ш щ з х
ф і в а п р о л д ж є
я ч с м и т ь б ю
123 Пробіл

01:05
Камера

Надійність пароля

Довжина паролю:

6

Наявність великих літер:

Наявність малих літер:

Наявність чисел:

Наявність спецсимволів:

Зберегти

Головна Перевірка Налаштування

01:05
Камера

Назад Новий пароль

Назва

Генерація пароля

Пароль

s!NejЦ

Згенерувати пароль

Відмінити Зберегти

Й ц у к е н г ш щ з х
ф і в а п р о л д ж є
я ч с м и т ь б ю
123 Пробіл

01:04
Камера

Надійність пароля

Довжина паролю:

6

Наявність великих літер:

Наявність малих літер:

Наявність чисел:

Наявність спецсимволів:

Зберегти

Головна Перевірка Налаштування

		Гладиш Т.В.			ККІКБ.КР.Б – 125 – 3 – ПЗ	Арк.
		Шелуха О.О.				57
Змн.	Арк.	№ докум.	Підпис	Дата		