Полтавський університет економіки і торгівлі

Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп’ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___»_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

РОЗРОБКА УТИЛІТИ «МЕНЕДЖЕР ПАРОЛІВ» З ВИКОРИСТАННЯМ АЛГОРИТМІВ ШИФРУВАННЯ

зі спеціальності 122 Комп’ютерні науки
освітня програма «Комп’ютерні науки»
ступеня бакалавра

Виконавець роботи Шупта Владислав Валерійович
_____«____»_____ 202_ р.
(підпис)

Науковий керівник к. ф.-м. н., доцент, Ольховська Олена Володимирівна
_____«____»_____ 202_ р.
(підпис)

Рецензент _____

ПОЛТАВА 2026

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
cellClick_	Це подія в таблицях DataGridView, яка спрацьовує щоразу, коли користувач натискає на будь-яку клітинку. Вона дозволяє миттєво визначити, який рядок або стовпець обрано, отримати дані з цієї клітинки чи всього рядка та виконати потрібні дії — наприклад, заповнити текстові поля для редагування, вибрати запис для оновлення або видалення, або ж запускати додаткову логіку програми.
Convert.ToInt32	Це метод у C#, який перетворює значення будь-якого сумісного типу у ціле число типу int.
SelectedIndex	Це властивість елементів керування яка повертає або задає номер обраного елемента у списку. Якщо нічого не вибрано — значення буде -1.
ExecuteNonQuery()	це метод команди SQL у C# (SqlCommand), який виконує запити без повернення результатів таблиці.

ВСТУП

Проектування утиліти із використанням алгоритмів шифрування є важливим і актуальним завданням сучасної інформаційної безпеки, оскільки у цифровому середовищі щодня зростає обсяг конфіденційних даних, які потребують надійного захисту від несанкціонованого доступу. Користувач взаємодіє з десятками сервісів, кожен із яких вимагає створення унікального й складного пароля, а їх запам’ятовування стає практично неможливим без допоміжних інструментів.

Саме тому постає потреба у створенні зручної, безпечної та автономної утиліти, здатної шифрувати й зберігати облікові записи, забезпечуючи користувачу швидкий доступ до них і водночас зберігаючи високий рівень захисту. Менеджер паролів повинен гарантувати конфіденційність, цілісність і доступність даних, використовуючи криптографічні алгоритми, що відповідають сучасним стандартам безпеки. Крім того, актуальність створення такої програми зумовлена необхідністю протидії кіберзагрозам, таким як крадіжка особистих даних, фішингові атаки та спроби несанкціонованого отримання доступу до облікових записів.

Розробка менеджера паролів передбачає аналіз вимог користувача, вибір оптимальних методів шифрування, створення зручного інтерфейсу та забезпечення надійної взаємодії з локальними чи хмарними сховищами. Таким чином, проєкт має комплексний характер і охоплює питання безпеки, програмної архітектури та зручності використання, що робить його важливою частиною сучасних технологій захисту інформації. [1]

Мета кваліфікаційної роботи – розробка програмної утиліти «Менеджер паролів», яка забезпечує безпечне зберігання, шифрування та керування даними користувача, використовуючи криптографічні алгоритми.

Об’єкт розробки – програмне забезпечення для керування паролями користувача.

Предмет розробки – методи і засоби шифрування, архітектура програмної утиліти, алгоритми обробки та зберігання конфіденційних даних, а також механізми захисту інформації в менеджері паролів

Методи дослідження – аналіз літературних джерел і стандартів безпеки для визначення найбільш надійних криптографічних алгоритмів. Порівняльний аналіз існуючих менеджерів паролів для визначення їхніх сильних та слабких сторін. Методи об’єктно-орієнтованого програмування для побудови архітектури утиліти. Алгоритмічне моделювання та розробка програмної логіки, що забезпечує шифрування, дешифрування та зберігання даних. Тестування програмного забезпечення для перевірки надійності, стабільності й працездатності утиліти. [2]

Актуальність розробки – зумовлена зростанням кількості кіберзагроз, широким використанням онлайн-сервісів і необхідністю надійного захисту конфіденційної інформації. Користувачам доводиться створювати та запам’ятовувати десятки складних паролів, що нерідко призводить до використання слабких або повторюваних комбінацій. Менеджер паролів, який застосовує сучасні алгоритми шифрування, дає можливість зберігати всі дані в одному безпечному місці, мінімізуючи ризики крадіжки інформації, атак перебору та несанкціонованого доступу. Саме тому розробка такого інструмента є важливою та практично необхідною.

Крім того, використання менеджера паролів значно підвищує зручність роботи користувача, оскільки дозволяє автоматично заповнювати облікові дані, швидко генерувати складні паролі та синхронізувати їх між різними пристроями. Це зменшує навантаження на пам’ять людини та сприяє формуванню більш відповідального підходу до кібербезпеки, розробка ефективного менеджера паролів не лише відповідає актуальним викликам інформаційної безпеки, а й сприяє підвищенню загального рівня захищеності користувачів у цифровому середовищі.

У першому розділі сформульовано постановку задачі, що полягає у створенні утиліти “Менеджер паролів”. У другому розділі здійснено огляд менеджера паролів #1 Password Manager для безпечного зберігання паролів, а також проаналізовано аналогічний застосунок Passwarden. У третьому розділі детально описано типи шифрування і алгоритми, наведено UML-діаграми функціонування застосунку і його програмної частини. У четвертому розділі висвітлено процес розробки утиліти “Менеджер паролів” та подано опис її роботи. Обсяг пояснювальної записки: 47 стор., в т.ч. основна частина – 43 стор., джерела – 20 назв.

1. ПОСТАНОВКА ЗАДАЧІ

Основною метою даної роботи є розробка та програмна реалізація утиліти «Менеджер паролів», що забезпечуватиме безпечне зберігання, шифрування та керування обліковими даними користувача.

Для досягнення мети проектування утиліти необхідно виконати такі завдання:

1. Аналіз вимог до утиліти – визначити функціональні й нефункціональні вимоги, оглянути схожі програмні продукти.
2. Проектування архітектури системи – окреслити основні модулі утиліти, їх взаємодію, структуру сховища даних та принципи шифрування.
3. Розробка детальних графічних діаграм роботи утиліти, описати логіку виконання основних операцій: збереження, шифрування та відображення паролів.
4. Розробка модуля управління записами користувача, створити структуру даних, алгоритми додавання, редагування та видалення записів.
5. Розробка криптографічного модуля, опис алгоритмів шифрування й дешифрування.
6. Розробка модуля захисту утиліти, від можливого несанкціонованого доступу.
7. Створення інформаційного вікна, щоб надати користувачу коротку інформацію про утиліту.
8. Проектування інтерфейсу користувача, створення макетів основних вікон та реалізація інтуїтивної взаємодії з утилітою.
9. Детальне тестування працездатності утиліти.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Огляд менеджера паролів #1 Password Manager

Розглянемо детально менеджера паролів “#1 Password Manager”. [3]

В першу чергу, потрібно встановити утиліту. Необхідно знайти програму за назвою #1 Password Manager або перейти за посиланням у браузері, потім натиснути отримати або встановити. Далі лише дочекатися завершення установки. Після цього програма буде доступна у меню «Пуск».

Також утиліту можна завантажити з Microsoft Store. (рис. 2.1).

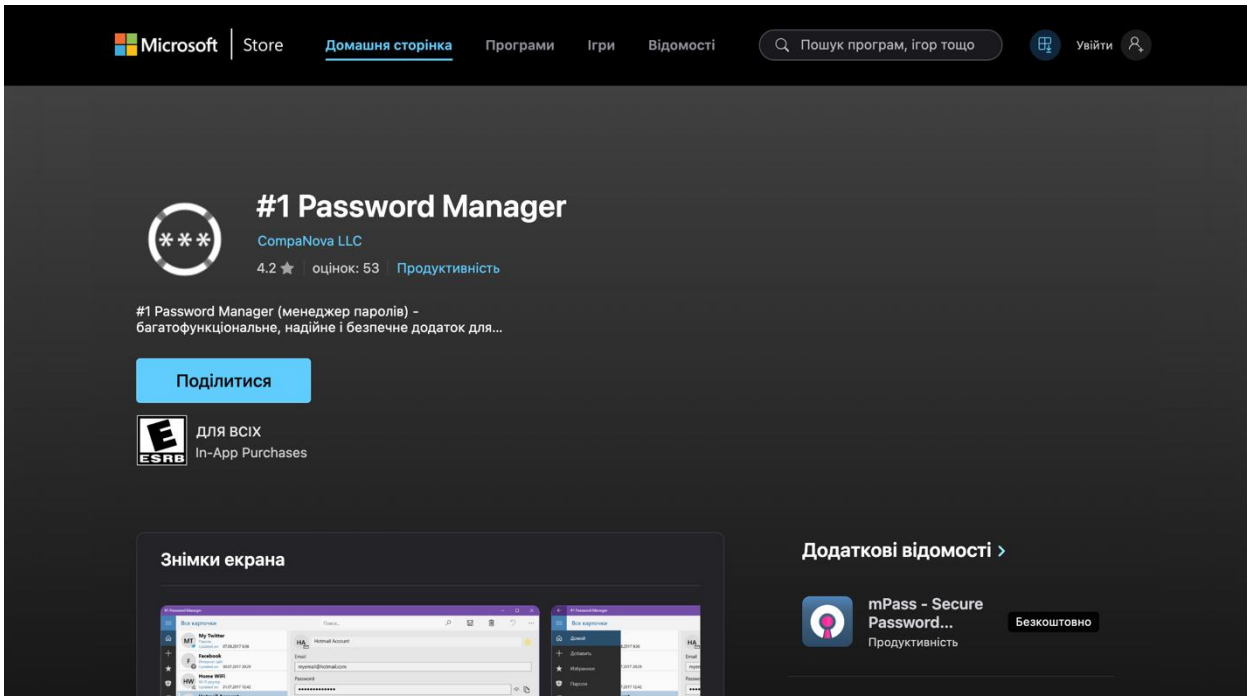


Рисунок 2.1 – утиліта менеджера паролів

Щоб додати запис потрібно натиснути кнопку «Додати новий запис» або «+» і заповнити поля, пароль (можна створити за допомогою вбудованого генератора паролів). (рис. 2.2). При збереженні — дані будуть зашифровані і збережені в локальному сховищі. (рис. 2.3).

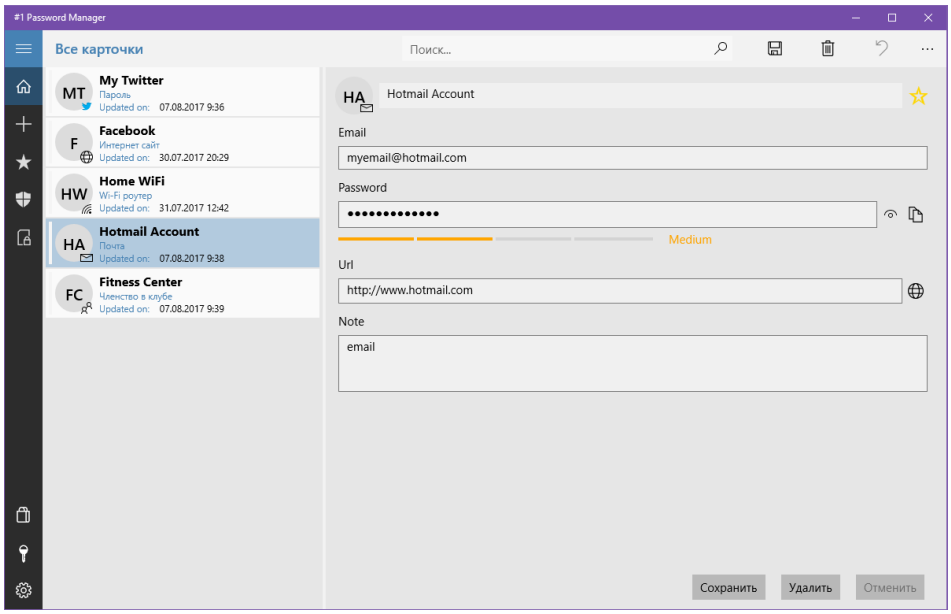


Рисунок 2.2 – додавання запису

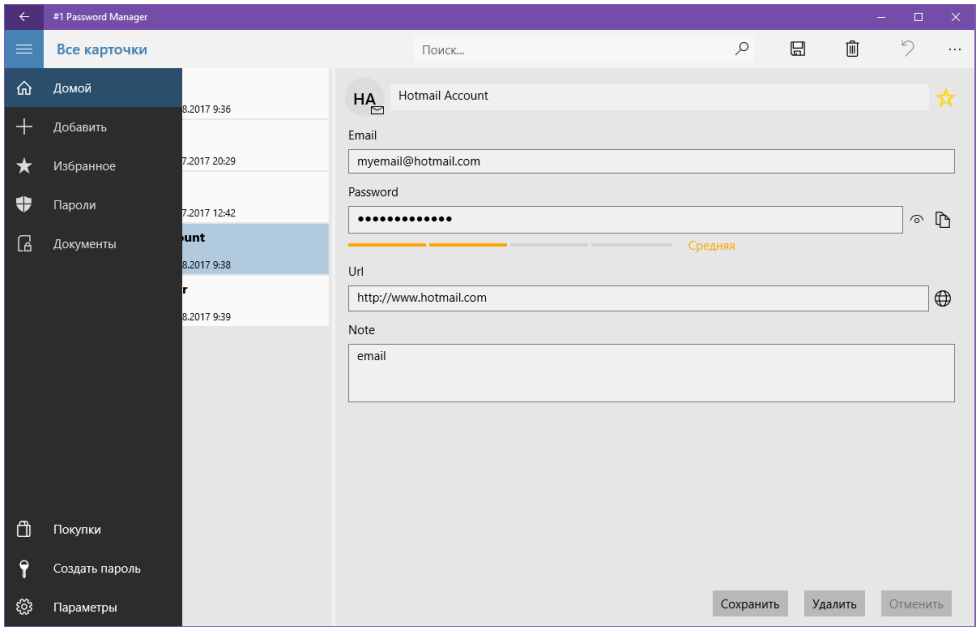


Рисунок 2.3 – процес зберігання запису

Також утиліта підтримує зручний вибір типу збереженого запису для кращої інформативності. (рис. 2.4).

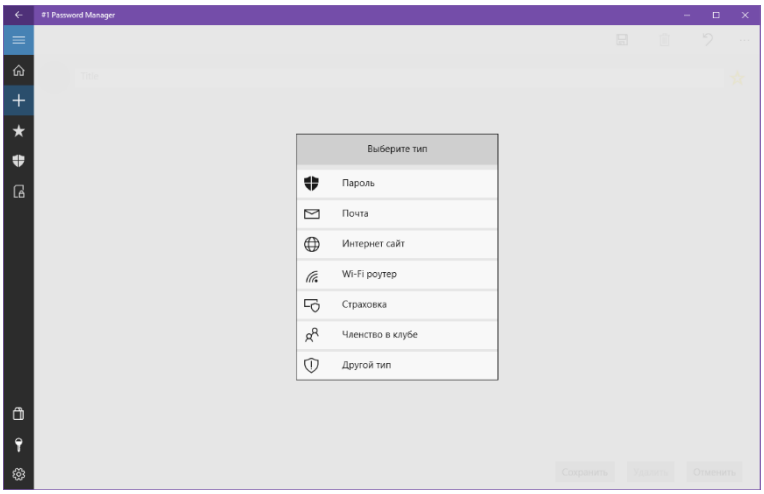


Рисунок 2.4 – тип збереженого запису

При створенні паролю можна використовувати також спеціальні символи. (рис. 2.5).

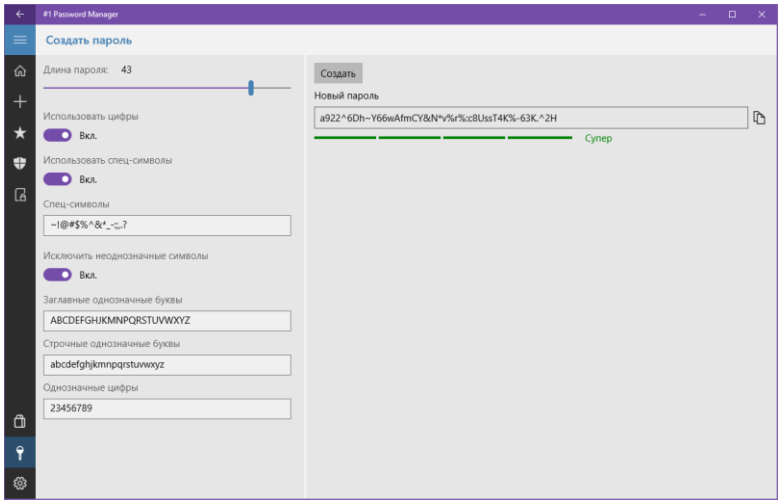


Рисунок 2.5 – підтримка спеціальних символів

За потреби є зворотній зв’язок з розробниками утиліти. (рис. 2.6).

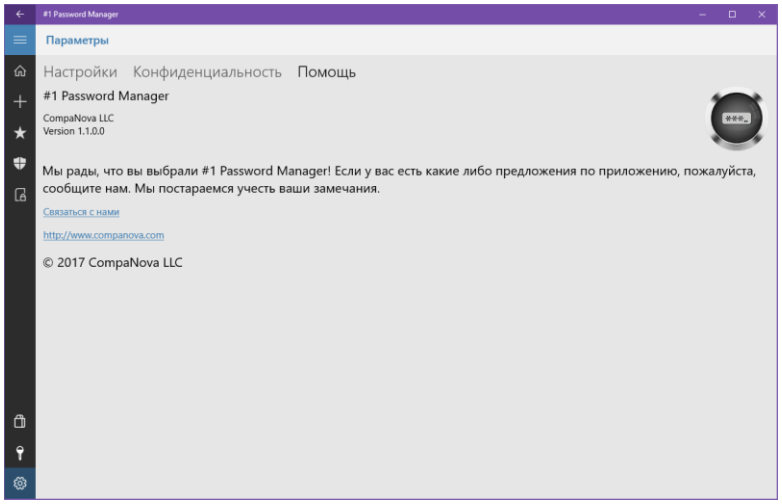


Рисунок 2.6 – зворотній зв’язок

Генератор паролів допомагає створювати надійні унікальні паролі. Пошук записів, швидкий доступ до потрібного облікового запису за назвою сервісу або логіном. Експорт/імпорт даних дозволяє переносити паролі між пристроями або робити резервні копії. (рис. 2.7).

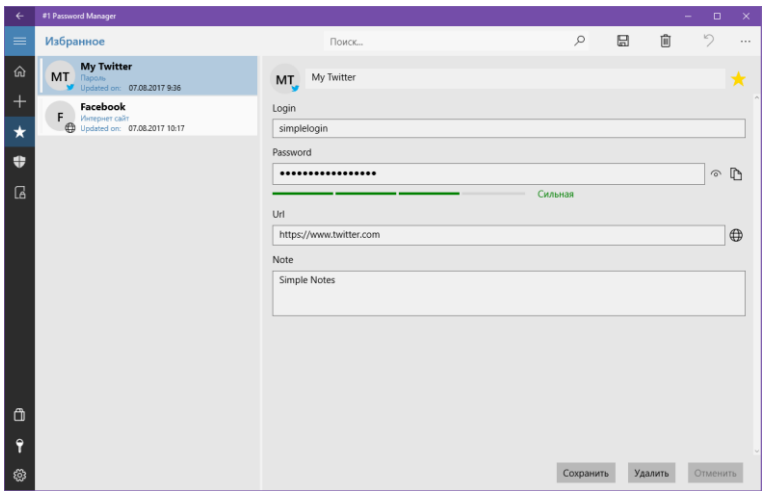


Рисунок 2.7 – огляд додаткових функцій

Переваги:

1. Сильне шифрування даних, #1 Password Manager використовує шифрування за алгоритмом AES‑ 256. Це один з сучасних стандартів, які забезпечують надійний захист від несанкціонованого доступу.
2. Зберігання даних лише на пристрої
3. Зручність і простота у користуванні — інтерфейс програми описаний як інтуїтивно зрозумілий, що підходить для користувачів без поглиблених технічних знань.
4. Можливість централізовано зберігати багато облікових записів — програма дає змогу зберігати логіни, паролі та інші приватні дані в одному місці, що позбавляє потреби пам’ятати їх усіх окремо. Це дуже зручно при великій кількості різних акаунтів.
5. Зменшення ризику забути пароль або втратити доступ — замість паперових записів, вся інформація зашифрована і зберігається в захищеному форматі, що підвищує безпеку та організованість.

Недоліки:

1. Мало публічної технічної інформації про алгоритми та внутрішню безпеку — у загальному описі є лише загальні фрази про “найкращі методи та передові стандарти безпеки”, без чіткого переліку реалізованих алгоритмів чи їх конфігурацій. Це ускладнює незалежну оцінку рівня захищеності.
2. Якщо master‑ пароль буде слабким або скомпрометованим, тоді всі збережені дані можуть опинитися під загрозою.
3. Менша прозорість реалізації безпеки, ніж у відомих менеджерів — на відміну від популярних менеджерів, які відкрито публікують технічні деталі, аудити, способи шифрування тощо.

2.2. Огляд менеджера паролів Passwarden

Менеджер паролів Passwarden забезпечує надійне зберігання та зручне керування паролями завдяки поєднанню сучасних технологій безпеки та продуманого функціоналу. Щодо керування паролями, Passwarden дозволяє централізовано організовувати всі облікові дані в одному місці. Користувач може додавати нові записи вручну або імпортувати їх із браузерів чи інших менеджерів паролів . (рис. 2.8).

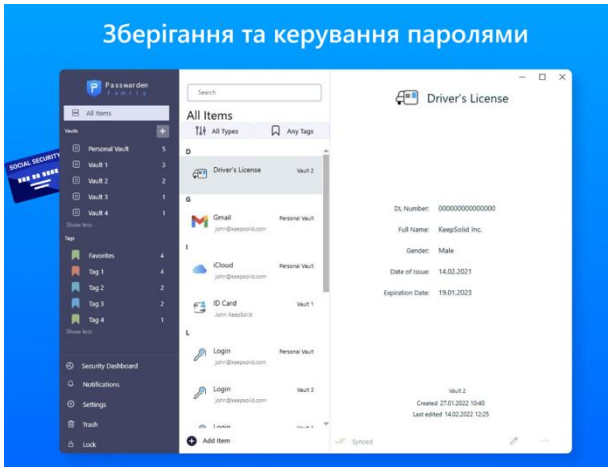


Рисунок 2.8 – Головний екран менеджера

Менеджер паролів Passwarden забезпечує ефективний захист від витоків даних завдяки використанню сучасних технологій шифрування та багаторівневих механізмів безпеки. (рис. 2.9). Менеджер паролів Passwarden підтримує безпечний обмін логінами та паролями. (рис. 2.10)

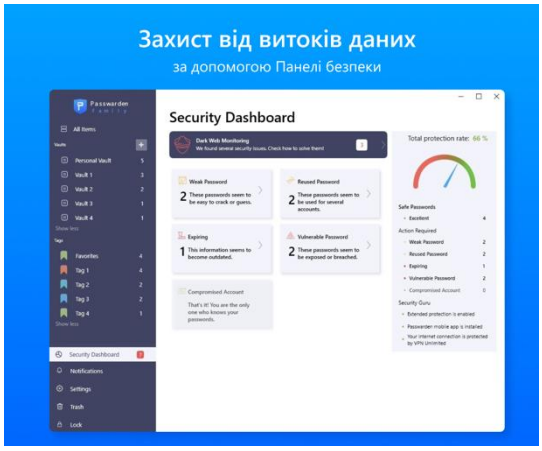


Рисунок 2.9 – Процес захисту від витоків даних

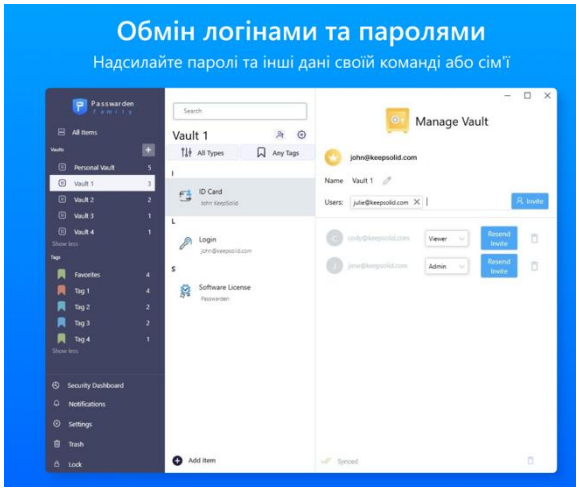


Рисунок 2.10 – Обмін логінами та паролями

Менеджер паролів Passwarden забезпечує синхронізацію та спільне використання даних, що робить його зручним для користування на кількох пристроях та у команді чи родині. (рис. 2.11).

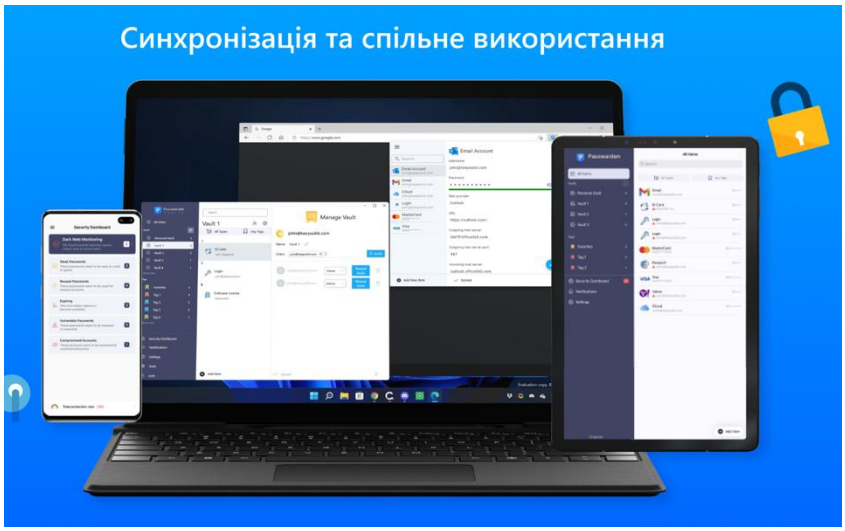


Рисунок 2.11 – Синхронізація

Менеджер паролів Passwarden дозволяє створювати надійні та складні паролі, що значно підвищує безпеку облікових записів користувача. (рис. 2.12).

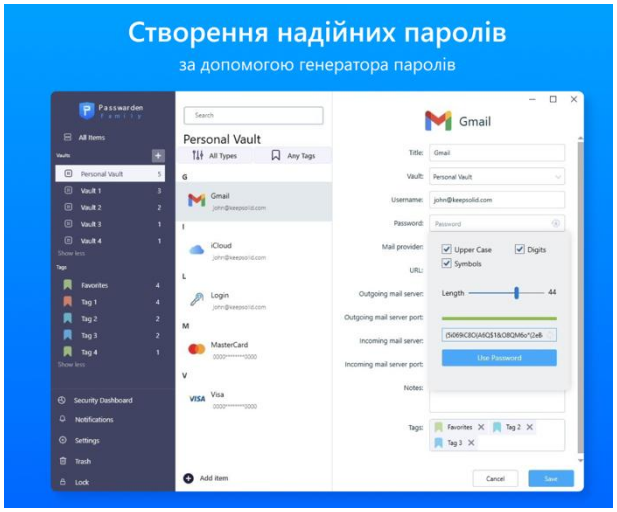


Рисунок 2.12 – Створення надійних паролів

Пакет безпеки MonoDefense це комплекс рішень для максимального захисту даних і свободи в інтернеті, створений компанією KeepSolid. Він поєднує в собі кілька інструментів, які разом працюють над тим, щоб ти міг безпечно серфити, зберігати конфіденційну інформацію та залишатися анонімним у мережі. (рис. 2.13).

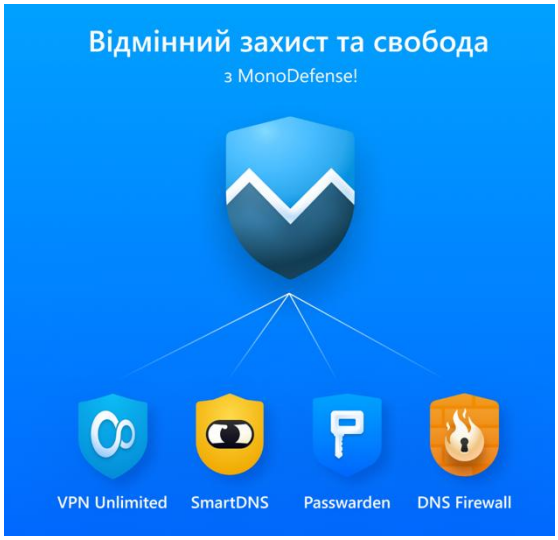


Рисунок 2.13 – Пакет безпеки MonoDefense

Менеджер паролів Passwarden має низку суттєвих переваг, які роблять його зручним і безпечним інструментом для зберігання конфіденційних даних. Насамперед, він забезпечує високий рівень безпеки завдяки використанню сучасного шифрування, що гарантує захист паролів навіть у разі витоку даних. Користувачеві достатньо запам’ятати лише один головний пароль, що значно спрощує керування великою кількістю облікових записів. Додатковою перевагою є синхронізація між різними пристроями, завдяки чому доступ до даних можливий у будь-який час і з будь-якого місця. Інтерфейс програми є інтуїтивно зрозумілим, тому навіть нові користувачі можуть швидко освоїти її функціонал. Також варто відзначити наявність додаткових засобів захисту, таких як двофакторна автентифікація та спеціальний режим безпеки, який дозволяє приховати справжні дані у разі примусу.

Водночас Passwarden має і певні недоліки. Зокрема, іноді можуть виникати проблеми з функцією автозаповнення, яка не завжди працює коректно, через що частину даних доводиться вводити вручну. Ще одним мінусом є обмежені можливості експорту даних, що ускладнює перехід до інших менеджерів паролів. Крім того, деякі функції доступні лише у платній версії, що може бути недоліком для користувачів, які шукають повністю безкоштовне рішення. Також обмін паролями можливий лише між користувачами цього ж сервісу, що обмежує гнучкість у використанні.

Отже, Passwarden є надійним і функціональним менеджером паролів, який добре підходить для щоденного використання та забезпечує високий рівень захисту даних. Незважаючи на окремі недоліки, пов’язані з функціональністю та вартістю, він залишається ефективним інструментом для підвищення безпеки користувача в інтернеті та зручного керування обліковими записами.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Типи шифрування і алгоритми

Шифрування — це процес перетворення інформації у формат, непридатний для прочитання, щоб доступ до неї мали лише користувачі з відповідними правами. Така трансформація здійснюється за допомогою криптографічних ключів у поєднанні з певними математичними алгоритмами. У цьому матеріалі розглянемо два ключові види шифрування — симетричне та асиметричне, а також п’ять найпоширеніших алгоритмів, що застосовуються на практиці.

Симетричне шифрування. Як випливає з назви, у цьому методі для шифрування та дешифрування інформації застосовується один і той самий криптографічний ключ. Використання спільного ключа для обох операцій значно спрощує весь процес. Розгляньмо принцип роботи симетричного шифрування на простому прикладі:

У Києві мешкають двоє добрих друзів — Антон і Аліса. Через певні обставини Аліса змушена залишити місто. Єдиний спосіб підтримувати зв’язок між ними — листування поштою. Однак виникає проблема: обоє хвилюються, що хтось може перехопити й прочитати їхні повідомлення.

Щоб убезпечити своє спілкування, вони домовляються кодувати текст так, щоб кожна літера замінювалася на літеру, що стоїть на сім позицій далі в алфавіті. Наприклад, замість «Apple» вони писатимуть «hwwsl» (A → H, P → W, L → S, E → L). Для зворотного перетворення потрібно просто зрушити кожну літеру назад на сім позицій. Такий метод шифрування використовувався ще в античні часи римським полководцем Гаєм Юлієм Цезарем і відомий під назвою «шифр Цезаря».

Чому симетричне шифрування є вдалим вибором. Головною перевагою симетричного шифрування є його простота: один і той самий ключ застосовується як для зашифрування даних, так і для їх

розшифрування. Коли необхідно захистити великі обсяги інформації, цей метод працює особливо ефективно. Як результат, алгоритми симетричного шифрування:

1. Працюють помітно швидше, ніж асиметричні алгоритми (до яких ми ще повернемося);
2. Вимагають менше ресурсів для обчислень;
3. Не впливають на швидкість інтернет-з’єднання.

Три популярні алгоритми симетричного шифрування. «Шифр Цезаря» ґрунтується на простому принципі заміни символів, тож розкрити його нескладно, якщо зрозуміти логіку перетворення. Натомість сучасні системи шифрування використовують надзвичайно складні математичні операції, які зробили їх практично недоступними для зламу. Сьогодні існують сотні симетричних алгоритмів! Найвідоміші серед них — AES, RC4, DES, 3DES, RC5, RC6 тощо. Розгляньмо три найбільш поширені з них.

1. DES — алгоритм симетричного шифрування

DES (Data Encryption Standard), представлений у 1976 році, вважається одним з найстаріших симетричних алгоритмів. Він був створений компанією IBM для захисту секретних урядових даних і в 1977 році офіційно затверджений для використання федеральними установами США. Алгоритм DES також застосовувався у ранніх версіях TLS (1.0 і 1.1), що забезпечують безпечну передачу даних у мережі.

DES працює з 64-бітними блоками відкритого тексту, які розділяються на два 32-бітні сегменти. Кожен із них проходить через послідовність із 16 раундів різноманітних операцій — розширення, перестановок, заміни та інших криптографічних перетворень. У результаті цих раундів формується 64-бітний блок зашифрованого тексту. У 2005 році DES офіційно визнали застарілим і замінили на більш надійний алгоритм AES. Основна причина — надто короткий ключ, який робив DES

вразливим до перебору. У сучасному протоколі TLS 1.2 цей метод шифрування вже не використовується.

2. 3DES — алгоритм симетричного шифрування

3DES, також відомий як TDEA (Triple Data Encryption Algorithm), є вдосконаленою версією класичного DES. Його створили наприкінці 1990-х років, щоб усунути слабкі сторони оригінального алгоритму. Як видно з назви, 3DES виконує три послідовні цикли DES над кожним блоком даних, що значно ускладнює процес злому порівняно з DES. Завдяки підвищеній стійкості TDEA став широко застосовуватися у платіжних системах та інших фінансових технологіях. Крім того, він був інтегрований у такі криптографічні протоколи, як TLS, SSH, IPsec та OpenVPN.

Проте, як і будь-який алгоритм, 3DES з часом став вразливим. Дослідники Картікеян Бхаварган і Гаетан Леурент виявили уразливість Sweet32, що поставило під сумнів подальшу безпечність 3DES. Це відкриття спонукало спеціалістів з кібербезпеки переглянути актуальність алгоритму, а Національний інститут стандартів і технологій США (NIST) у 2019 році офіційно оголосив про його поступове виведення з використання.

Відповідно до рекомендацій NIST, застосування 3DES у нових системах має бути повністю припинено після 2023 року. Варто додати, що найновіший стандарт TLS 1.3 також відмовився від використання цього алгоритму.

3. AES — алгоритм симетричного шифрування

AES (Advanced Encryption Standard), також відомий як Rijndael, є одним із найпоширеніших і найнадійніших алгоритмів симетричного шифрування. Він був створений як заміна DES і в 2001 році затверджений NIST як новий стандарт захисту даних. AES являє собою сімейство блокових шифрів, що відрізняються розміром ключа та блоків.

Принцип роботи AES ґрунтується на операціях підстановки та перестановки. Спочатку відкритий текст розбивається на блоки, після чого

кожен блок проходить через послідовність криптографічних перетворень із використанням ключа. Цей процес включає такі операції, як зсув рядків, змішування стовпців та додавання раундового ключа. Кількість раундів залежить від довжини ключа: для 128-бітного ключа виконується 10 раундів, для 192-бітного — 12, а для 256-бітного — 14. Важливо, що останній раунд відрізняється тим, що не містить етапу змішування стовпців.

AES вважається надзвичайно стійким та ефективним, тому широко використовується у сучасних протоколах безпеки, хмарних сервісах, мобільних додатках і системах захисту даних.

У підсумку AES є одночасно безпечним, швидким і універсальним. Він працює значно швидше, ніж DES, а можливість використовувати ключі різної довжини є його ключовою перевагою: що довший ключ, то важче його зламати.

Сьогодні AES — найпоширеніший алгоритм симетричного шифрування. Його застосовують у численних технологіях та сервісах, зокрема:

1. бездротових системах безпеки;
2. апаратному шифруванні та захисті файлів;
3. протоколах SSL / TLS, які забезпечують безпеку вебсайтів;
4. Wi-Fi-з’єднаннях;
5. мобільних застосунках;
6. VPN-технологіях тощо.

Чимало державних установ США також використовують AES для захисту конфіденційних даних, що підтверджує його надійність і довіру до цього алгоритму. [4]

3.2. Покроковий алгоритм роботи коду утиліти “Менеджер паролів”

Алгоритм шифрування починається з того, що збережені у програмі рядки aesKey та aesIV перетворюються у масиви байтів за допомогою кодування UTF-8. Це робиться для того, щоб ключ та ініціалізаційний вектор можна було встановити у параметри AES-алгоритму. Далі створюється об’єкт AES через Aes.Create(), після чого в нього встановлюється ключ, IV, режим роботи CBC та заповнення блоків PKCS7. Після цього через метод CreateEncryptor() створюється шифрувальник, який здатний перетворювати дані. Наступним кроком початковий текст, який потрібно зашифрувати, також переводиться у байти UTF-8. Потім ці байти передаються до encryptor.TransformFinalBlock(), який виконує фактичне шифрування та повертає масив зашифрованих байтів. Отриманий результат перетворюється у Base64-рядок, щоб його можна було зручно зберігати або передавати. На цьому процес шифрування завершується, і метод повертає готовий Base64-шифротекст.

EncryptAES — шифрування

1. Отримати рядок aesKey ключ і рядок aesIV ініціалізаційний вектор.

2. Перетворити aesKey у масив байтів: keyBytes = Encoding.UTF8.GetBytes aesKey.

3. Перетворити aesIV у масив байтів: ivBytes = Encoding.UTF8.GetBytes aesIV.

4. Викликати Aes.Create() і зберегти об’єкт AES.

5. Встановити aes.Key = keyBytes.

6. Встановити aes.IV = ivBytes.

7. Налаштувати режим шифрування: aes.Mode = CipherMode.CBC.

8. Налаштувати заповнення блоків: aes.Padding = PaddingMode.PKCS7.

9. Створити шифрувальник: using (var encryptor = aes.CreateEncryptor()).

10. Перетворити вхідний відкритий текст plainText у байти: plainBytes = Encoding.UTF8.GetBytes plainText.

11. Викликати шифрування над байтами: encryptedBytes = encryptor.TransformFinalBlock(plainBytes, 0, plainBytes.Length).

12. Перетворити зашифровані байти в Base64-рядок: Convert.ToBase64String encryptedBytes.

13. Повернути цей Base64-рядок як результат методу.

Алгоритм розшифрування працює у зворотному порядку. Спочатку той самий ключ і IV знову перетворюються у масиви байтів UTF-8, ensuring що параметри AES збігаються з тими, що використовувалися під час шифрування. Потім вхідний шифротекст, який зберігається у вигляді Base64, декодується у масив байтів. Створюється новий об’єкт AES, у якого встановлюється той самий ключ, IV, режим CBC і Padding PKCS7. Далі створюється дешифрувальник через CreateDecryptor(). Шифровані байти передаються до decryptor.TransformFinalBlock(), який повертає початковий масив незашифрованих байтів. Останнім кроком ці байти конвертуються назад у текст UTF-8, після чого метод повертає оригінальний розшифрований рядок.

DecryptAES — дешифрування

1. Отримати ті самі рядки aesKey і aesIV, що використовувалися для шифрування.

2. Перетворити aesKey у масив байтів: keyBytes = Encoding.UTF8.GetBytes aesKey.

3. Перетворити aesIV у масив байтів: ivBytes = Encoding.UTF8.GetBytes aesIV.

4. Перетворити вхідний шифротекст cipherText з Base64 у байти: cipherBytes = Convert.FromBase64String cipherText.

5. Викликати Aes.Create() і зберегти об’єкт AES.

6. Встановити aes.Key = keyBytes.

7. Встановити aes.IV = ivBytes.

8. Налаштувати режим: aes.Mode = CipherMode.CBC.

9. Налаштувати заповнення: aes.Padding = PaddingMode.PKCS7.

10. Створити дешифрувальник: using var decryptor = aes.CreateDecryptor().

11. Викликати дешифрування над байтами: decryptedBytes = decryptor.TransformFinalBlock(cipherBytes, 0, cipherBytes.Length).

12. Перетворити отримані байти назад у текст UTF-8: Encoding.UTF8.GetString decryptedBytes.

13. Повернути розшифрований рядок як результат методу.

Розглянутий код реалізує надійну та структуровану систему шифрування і дешифрування даних за стандартом AES у режимі CBC із використанням заповнення PKCS7. Обидві функції — EncryptAES та

DecryptAES — виконують коректну послідовність операцій: перетворюють ключ і IV у байти, налаштовують параметри AES, створюють відповідні перетворювачі та опрацьовують текстові дані, забезпечуючи захист інформації від несанкціонованого доступу. Шифротекст формується у форматі Base64, що дозволяє зручно зберігати його у базах даних або передавати через мережу. Така реалізація демонструє правильне застосування симетричної криптографії та забезпечує безпечне зберігання конфіденційних даних у програмі. Якщо ключ і IV мають відповідну довжину та зберігаються захищено, алгоритм гарантує високий рівень криптографічної безпеки.

3.3. Загальна UML діаграма роботи утиліти

UML діаграма роботи утиліти “Менеджер паролів” (рис. 3.1)

Рисунок 3.1 – UML діаграма

3.4. UML діаграма роботи алгоритму шифрування

UML діаграма роботи алгоритму шифрування утиліти (рис. 3.2)

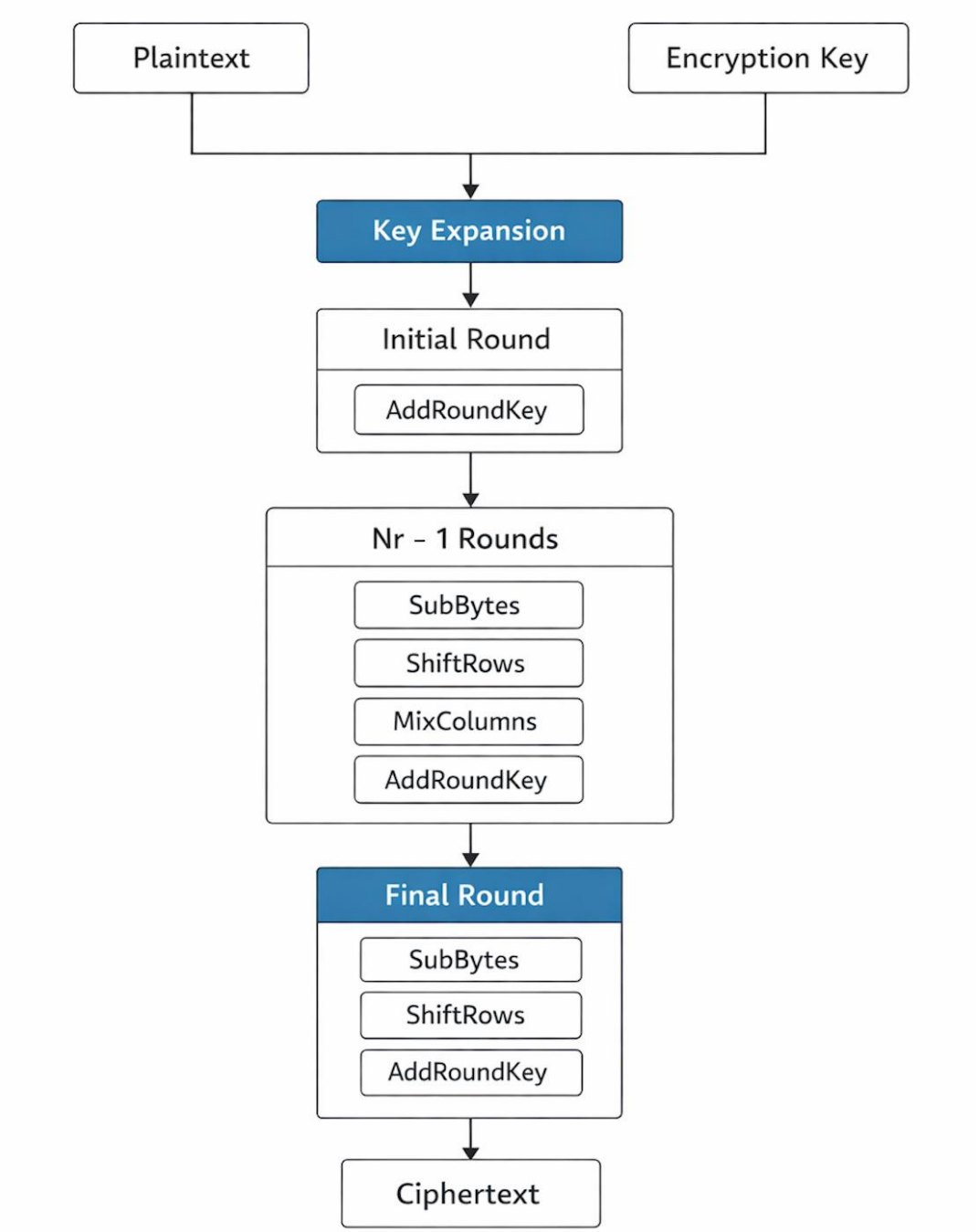


Рисунок 3.2 – UML діаграма роботи алгоритму шифрування утиліти

4. ПРАКТИЧНА ЧАСТИНА

4.1. Обґрунтування вибору програмних засобів

Для розробки утиліти «Менеджер паролів» був обраний сучасний високорівневий мова програмування C# разом із платформою .NET, що забезпечує широкий спектр функціональних можливостей для створення безпечних, стабільних і зручних у користуванні програмних продуктів. Однією з ключових переваг C# є його повна інтеграція з операційною системою Windows, що дозволяє легко реалізовувати графічний інтерфейс користувача через Windows Forms, WPF або UWP, а також забезпечує доступ до апаратних ресурсів і системних сервісів. Для менеджера паролів це особливо важливо, оскільки інтерфейс повинен бути інтуїтивно зрозумілим, швидким і зручним, а також підтримувати надійні механізми аутентифікації та шифрування даних користувача.

C# надає розробнику багатий набір стандартних бібліотек, що значно спрощує роботу з файлами, базами даних, шифруванням та криптографією. Використання класів із простору System.Security.Cryptography дозволяє реалізувати надійне шифрування паролів і конфіденційної інформації за сучасними алгоритмами, такими як AES або RSA, що відповідає сучасним вимогам інформаційної безпеки. Крім того, C# підтримує роботу з різними типами сховищ даних — від локальних файлів у зашифрованому вигляді до підключення до баз даних SQL, що робить утиліту гнучкою у використанні та масштабованою для подальшого розвитку.

Мова C# має виражену об’єктно-орієнтовану природу, що дозволяє розробляти додаток модульно, із чіткою структурою класів та методів. Це сприяє підвищенню надійності та підтримуваності коду, а також дозволяє легко додавати нові функціональні можливості, наприклад генератор надійних паролів, функцію резервного копіювання даних, автоматичне автозаповнення форм у браузері або синхронізацію між пристроями.

Об’єктно-орієнтований підхід дозволяє чітко відокремити логіку шифрування, зберігання даних і роботу інтерфейсу користувача, що значно спрощує процес тестування та відлагодження програми.

C# має активну спільноту та великий набір готових компонентів, що прискорює розробку та дозволяє інтегрувати сучасні бібліотеки для підвищення безпеки й зручності користування. Використання Visual Studio як інтегрованого середовища розробки надає широкі можливості для автоматизованого тестування, відлагодження та аналізу коду, а також підтримку систем контролю версій, що робить розробку більш організованою і професійною.

Крім технічних переваг, вибір C# обґрунтований і практично: він дозволяє створити додаток, який буде легко встановлюватися і запускатися на будь-яких версіях Windows без додаткових налаштувань, що особливо важливо для кінцевого користувача менеджера паролів. Крім того, C# забезпечує високу продуктивність, стабільність виконання і можливість масштабування програми в майбутньому, наприклад для підтримки багатокористувацьких функцій, інтеграції з хмарними сховищами чи мобільними пристроями.

Таким чином, вибір C# і платформи .NET для розробки утиліти «Менеджер паролів» є оптимальним, оскільки ця мова поєднує високу продуктивність, надійність, простоту реалізації сучасного інтерфейсу, підтримку об’єктно-орієнтованого програмування та потужні криптографічні можливості, що дозволяє створити безпечний, функціональний та зручний у використанні програмний продукт, який відповідатиме сучасним вимогам користувачів і стандартам інформаційної безпеки.

4.2. Опис програмної реалізації утиліти

Утиліту менеджера паролів з шифруванням написано на мові програмування C# [9, 10], в зручному середовищі розробки Visual Studio. Необхідно створити новий проект. (рис. 4.1)

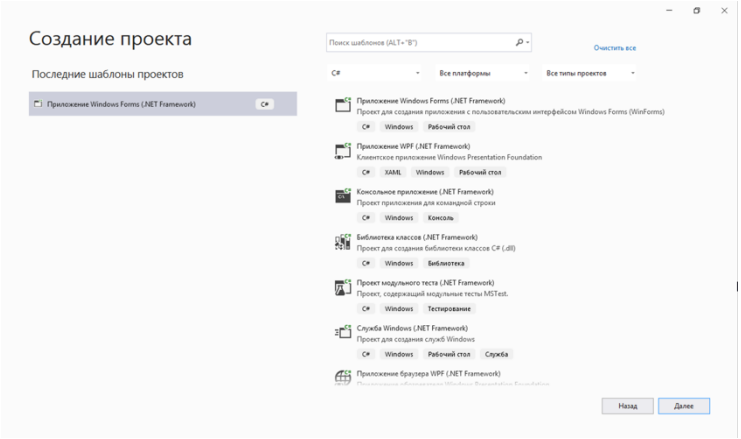


Рисунок 4.1 – новий проект.

Спочатку розробляється стартова форма утиліти менеджера паролів з алгоритмами шифрування. (рис. 4.2)

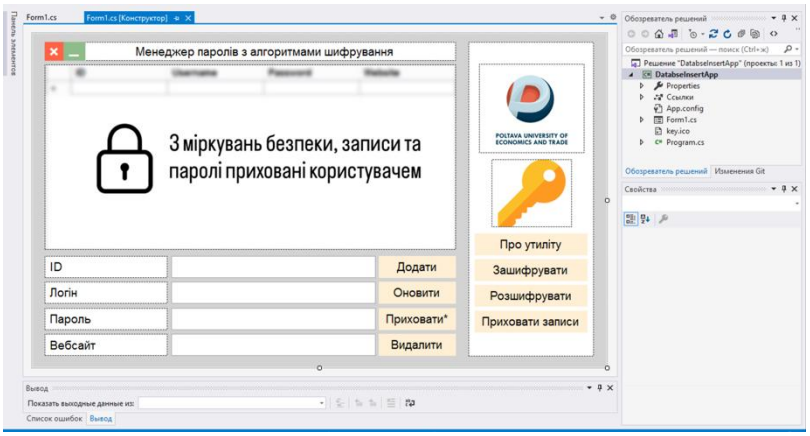


Рисунок 4.2 – стартова форма утиліти менеджера паролів

Після розміщення головних елементів на формі утиліти, необхідно прописати код шифрування алгоритмом AES. (рис. 4.3)

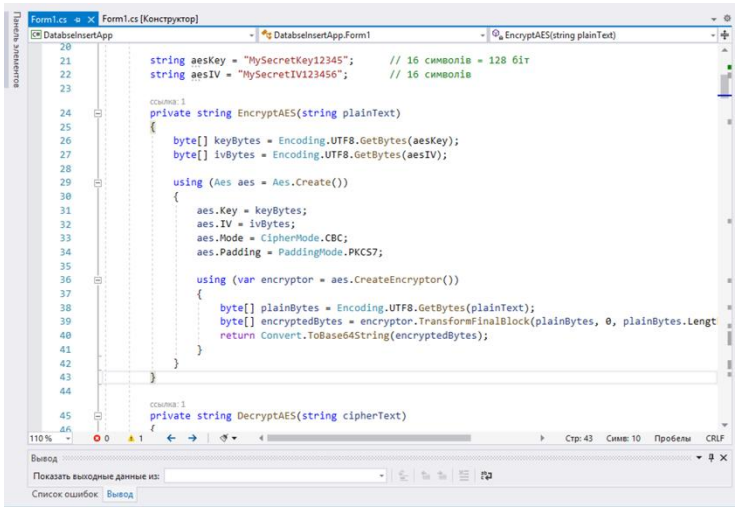


Рисунок 4.3 – написання коду шифрування AES.

Для того щоб зашифровані дані користувача зберігалися, необхідно створити підключення до локальної бази даних SQL. (рис. 4.4)

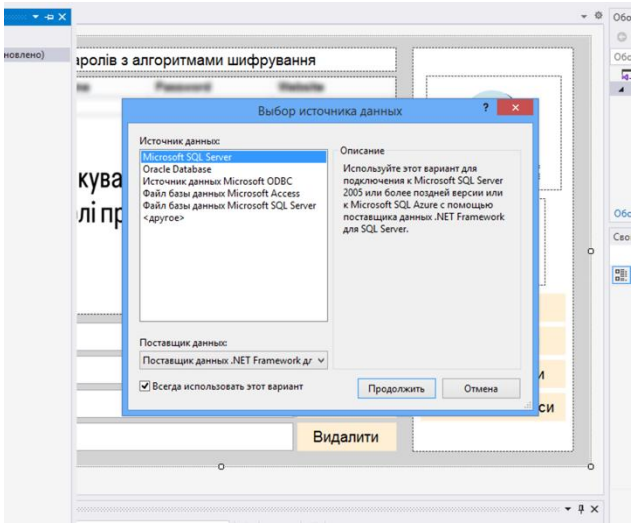


Рисунок 4.4 – підключення до локальної бази даних SQL.

Для утиліти необхідно створити іконку, для конвертування фото у формат іконки було обрано сервіс cloud convert. (рис. 4.5)

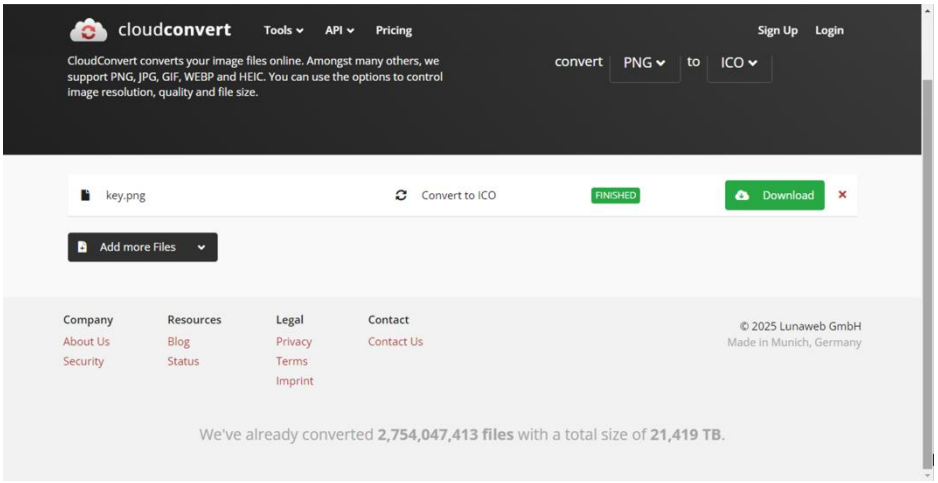


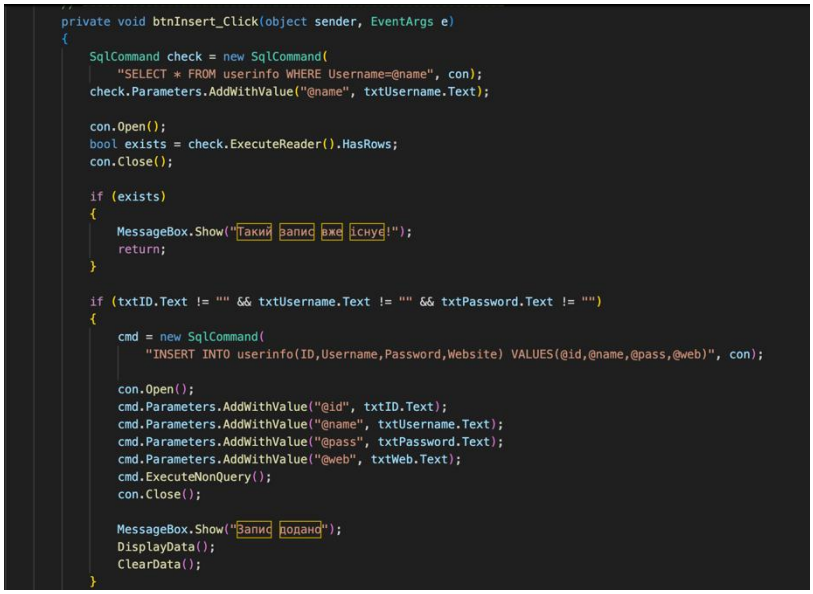
Рисунок 4.5 – сервіс створення іконки.

Для збереження даних у локальній базі даних, необхідно прописати запити і логіку роботи коду при запуску, щоб утиліта автоматично створила файл і таблицю SQL (рис. 4.6)

```
148     private void EnsureTablesExist()
149     {
150         string conStr = $@"
151         Data Source=(LocalDB)\MSSQLLocalDB;
152         AttachDbFilename={dbPath};
153         Integrated Security=True";
154 
155         using (SqlConnection db = new SqlConnection(conStr))
156         {
157             db.Open();
158 
159             string createTable = @"
160             IF NOT EXISTS (SELECT * FROM sysobjects WHERE name='userinfo')
161             CREATE TABLE userinfo(
162                 ID INT PRIMARY KEY,
163                 Username NVARCHAR(255),
164                 Password NVARCHAR(255),
165                 Website NVARCHAR(255)
166             )";
167 
168             new SqlCommand(createTable, db).ExecuteNonQuery();
169         }
170     }
```

Рисунок 4.6 – запити на створення таблиці SQL.

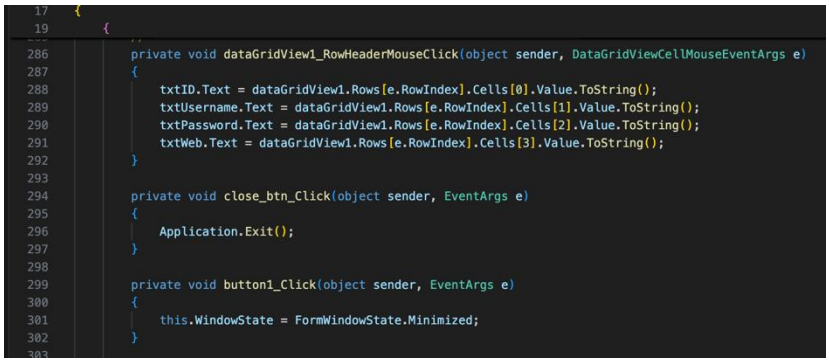
Після створення файлу та таблиці, необхідно прописати головні кнопки утиліти на додавання, оновлення, видалення записів з бази даних. (рис. 4.7)



```
private void btnInsert_Click(object sender, EventArgs e)
{
    SqlCommand check = new SqlCommand(
        "SELECT * FROM userinfo WHERE Username=@name", con);
    check.Parameters.AddWithValue("@name", txtUsername.Text);

    con.Open();
    bool exists = check.ExecuteReader().HasRows;
    con.Close();

    if (exists)
    {
        MessageBox.Show("Такий запис вже існує!");
        return;
    }

    if (txtID.Text != "" && txtUsername.Text != "" && txtPassword.Text != "")
    {
        cmd = new SqlCommand(
            "INSERT INTO userinfo(ID,Username,Password,Website) VALUES(@id,@name,@pass,@web)", con);

        con.Open();
        cmd.Parameters.AddWithValue("@id", txtID.Text);
        cmd.Parameters.AddWithValue("@name", txtUsername.Text);
        cmd.Parameters.AddWithValue("@pass", txtPassword.Text);
        cmd.Parameters.AddWithValue("@web", txtWeb.Text);
        cmd.ExecuteNonQuery();
        con.Close();

        MessageBox.Show("Запис доданo");
        DisplayData();
        ClearData();
    }
```

Рисунок 4.7 – прописування логіки головних кнопок утиліти.

Щоб записи з бази даних підтягувались та відображались правильно на головній формі, додатково потрібно вивести їх у dataGridView. (рис. 4.8)



```
{
    {
        private void dataGridView1_RowHeaderMouseClick(object sender, DataGridViewCellMouseEventArgs e)
        {
            txtID.Text = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
            txtUsername.Text = dataGridView1.Rows[e.RowIndex].Cells[1].Value.ToString();
            txtPassword.Text = dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString();
            txtWeb.Text = dataGridView1.Rows[e.RowIndex].Cells[3].Value.ToString();
        }

        private void close_btn_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Minimized;
        }
```

Рисунок 4.8 – код відображення записів на головній формі.

При першому запуску, утиліта створює файл бази даних у тому ж місці де розміщений сам .exe утиліти у папці Data. (рис. 4.9)

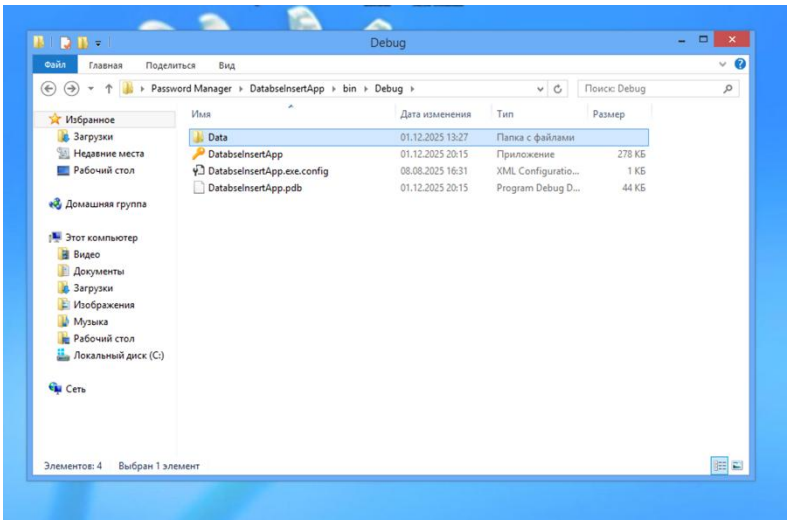


Рисунок 4.9 – шлях збереження бази даних.

Наступним кроком, необхідно створити інформаційне вікно для утиліти. (рис. 4.10)

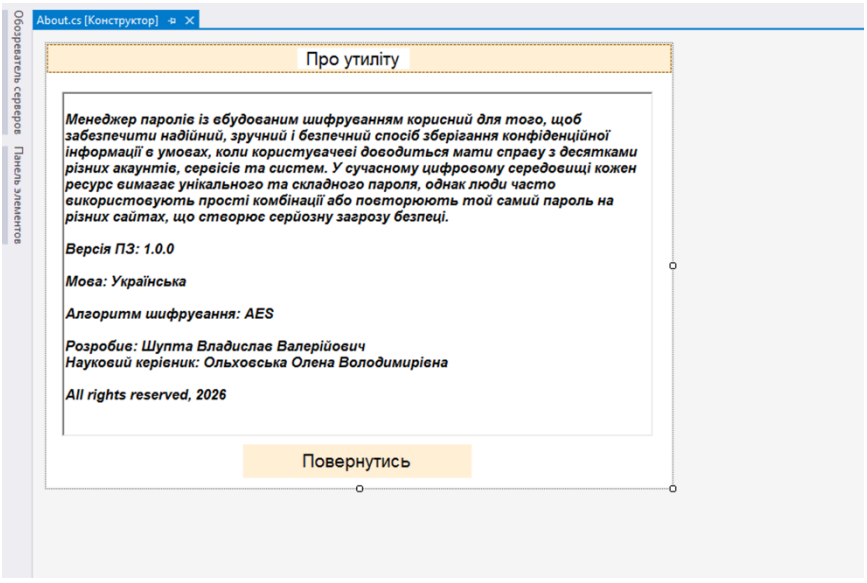
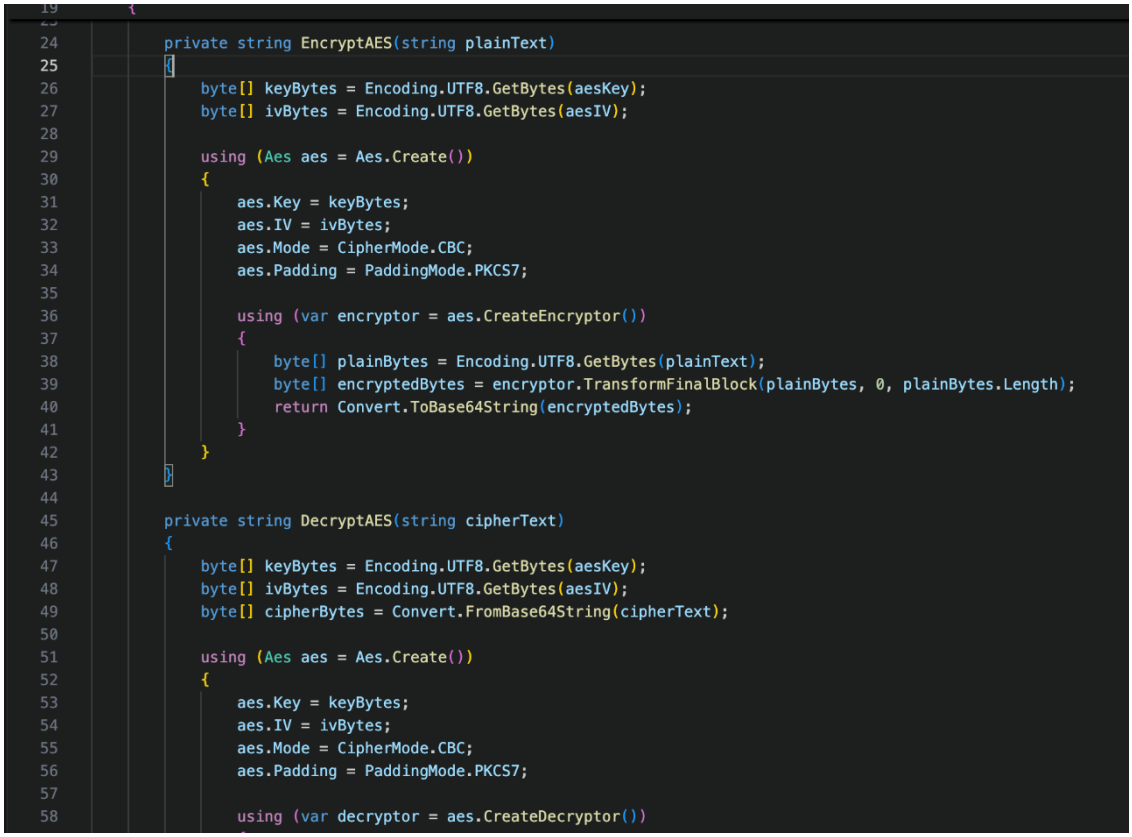


Рисунок 4.10 – інформаційне вікно утиліти.

Не менш значущим етапом створення програмного продукту є реалізація механізмів шифрування й дешифрування, а також детальна обробка та тестування коду і логіки роботи всієї утиліти. (рис. 4.11)



```
    {
        private string EncryptAES(string plainText)
        {
            byte[] keyBytes = Encoding.UTF8.GetBytes(aesKey);
            byte[] ivBytes = Encoding.UTF8.GetBytes(aesIV);

            using (Aes aes = Aes.Create())
            {
                aes.Key = keyBytes;
                aes.IV = ivBytes;
                aes.Mode = CipherMode.CBC;
                aes.Padding = PaddingMode.PKCS7;

                using (var encryptor = aes.CreateEncryptor())
                {
                    byte[] plainBytes = Encoding.UTF8.GetBytes(plainText);
                    byte[] encryptedBytes = encryptor.TransformFinalBlock(plainBytes, 0, plainBytes.Length);
                    return Convert.ToBase64String(encryptedBytes);
                }
            }
        }

        private string DecryptAES(string cipherText)
        {
            byte[] keyBytes = Encoding.UTF8.GetBytes(aesKey);
            byte[] ivBytes = Encoding.UTF8.GetBytes(aesIV);
            byte[] cipherBytes = Convert.FromBase64String(cipherText);

            using (Aes aes = Aes.Create())
            {
                aes.Key = keyBytes;
                aes.IV = ivBytes;
                aes.Mode = CipherMode.CBC;
                aes.Padding = PaddingMode.PKCS7;

                using (var decryptor = aes.CreateDecryptor())
```

Рисунок 4.11 – код шифрування та дешифрування.

Код утиліти «Менеджер паролів» було написано за допомогою середовища розробки Microsoft Visual Studio 2019 із використанням мови програмування C# та технології Windows Forms. Для графічного проєктування інтерфейсу, автоматичної генерації коду елементів форми та зручної прив’язки подій застосовувалася Visual Studio 2022, що значно спростило розробку користувацького інтерфейсу.

4.3. Опис роботи утиліти “Менеджер паролів”

Після запуску утиліти менеджера паролів, користувача зустрічає екран з паролем у мінімалістичному дизайні. Перший раз необхідно придумати пароль та увести, програма його занесе до бази даних для повторного використання. (рис. 4.12)

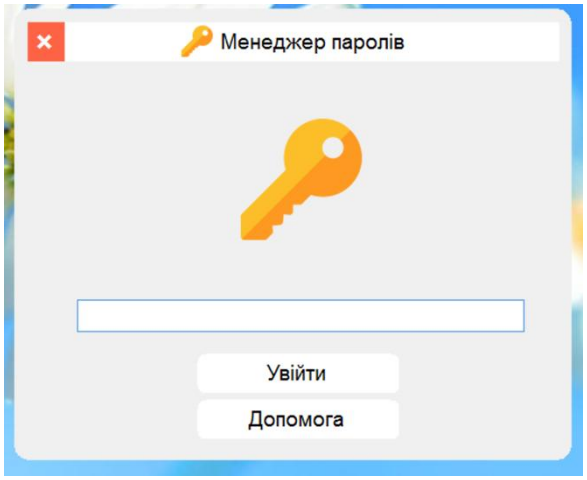


Рисунок 4.12 – екран з паролем.

Якщо користувач натисне кнопку допомоги, йому виведеться вікно з додатковою інформацією про пароль входу утиліти. (рис. 4.13)

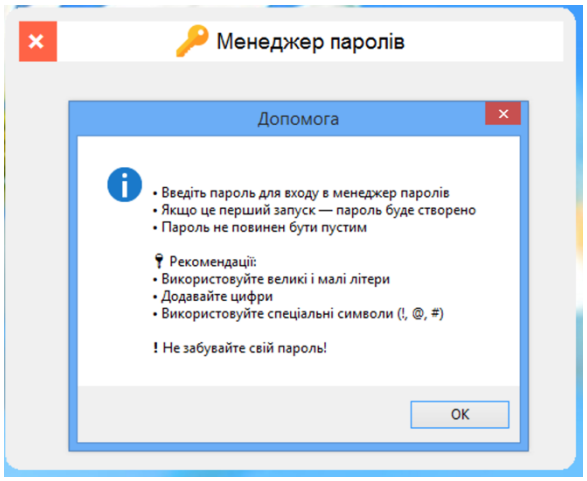


Рисунок 4.13 – вікно допомоги.

В утиліті у вікні введення паролю додатково оброблено помилку пустого поля, якщо користувач не увів нічого і натисне кнопку входу – висвітиться помилка. (рис. 4.14)

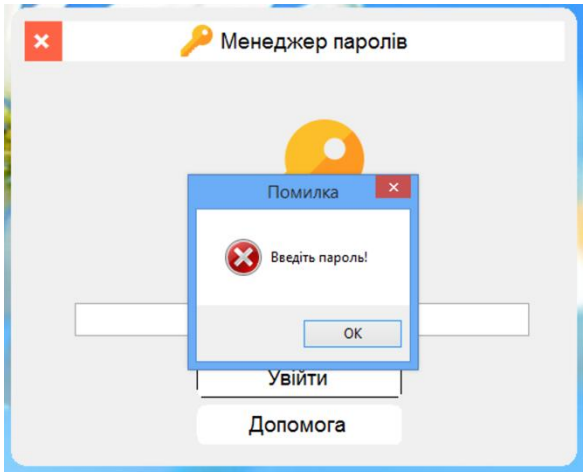


Рисунок 4.14 – обробка помилки пустого поля.

Також оброблено помилку коли пароль не збігається з тим паролем, який збережено в базі даних. (рис. 4.15)

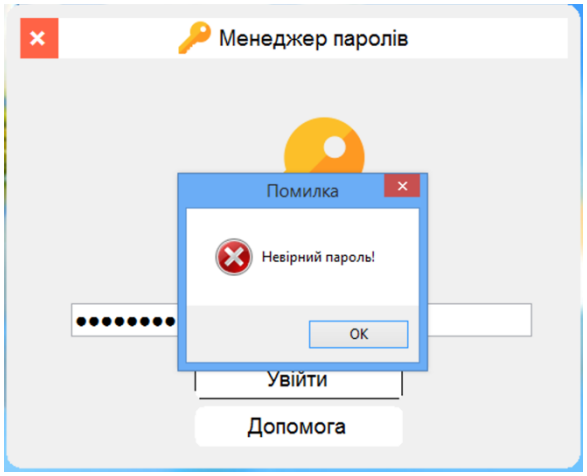


Рисунок 4.15 – обробка помилки неправильного пароля.

Після успішного входу до утиліти менеджера паролів, користувача зустрічає головний екран з обліковими записами та кнопками керування з полями для введення. У правій частині утиліти розміщено кнопки за допомогою яких можна шифрувати та дешифрувати паролі облікових записів. (рис. 4.16)

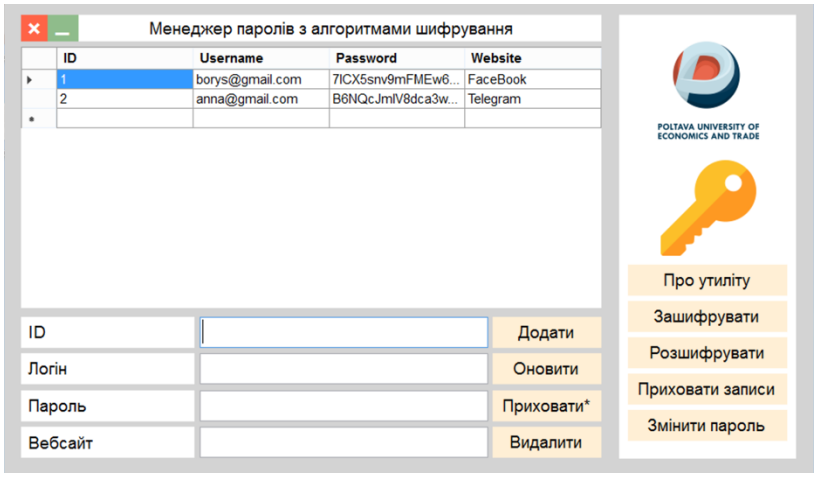


Рисунок 4.16 – головне вікно утиліти.

Якщо користувач натисне кнопку “Про утиліту” у правій частині вікна, він отримає довідкову інформацію. (рис. 4.17)

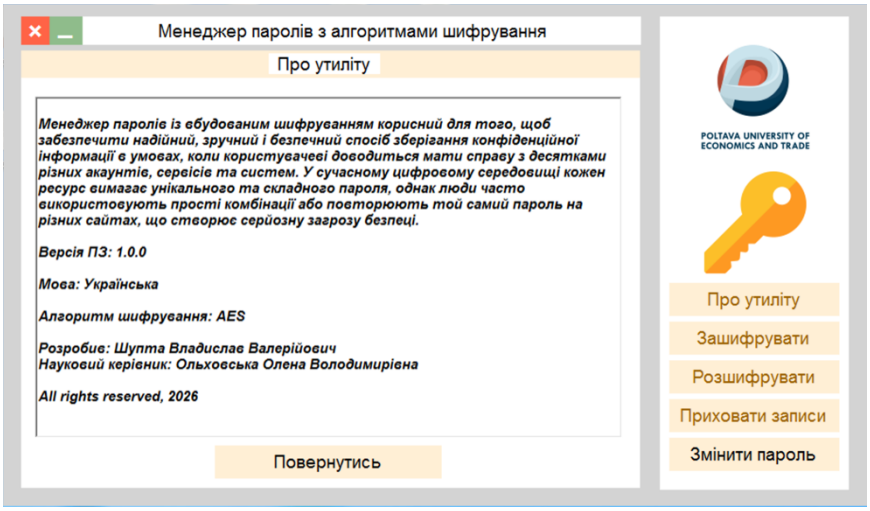


Рисунок 4.17 – довідкова інформація.

Щоб додати обліковий запис до менеджера, необхідно спочатку заповнити всі поля утиліти. (рис. 4.18)

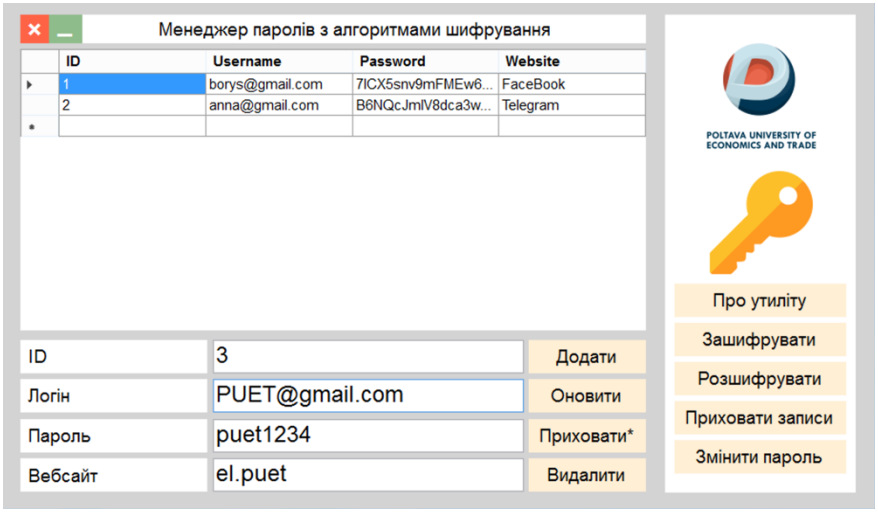


Рисунок 4.18 – заповнення полів з паролем.

Якщо не заповнити якесь поле, утиліта додатково виведе вікно попередження про необхідність заповнення всіх полів. (рис. 4.19)

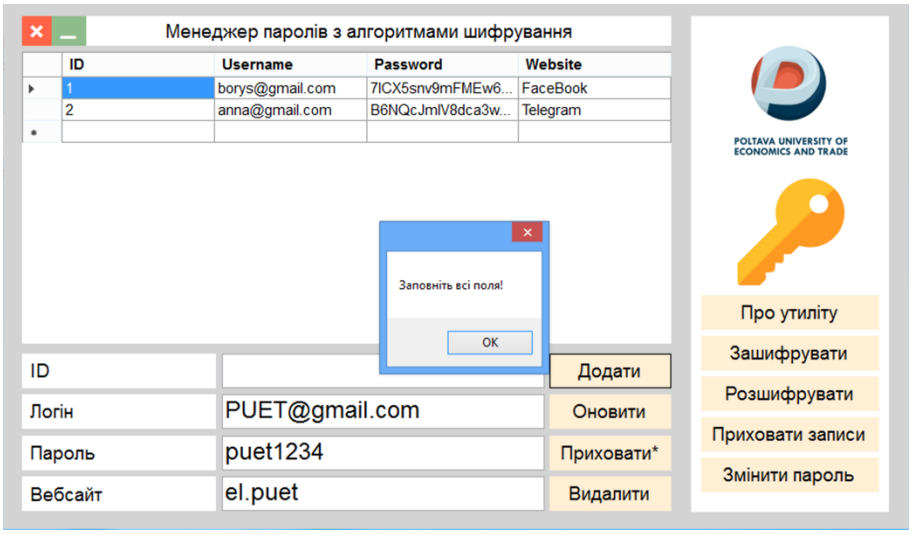


Рисунок 4.19 – вікно попередження.

Якщо всі поля заповнені, запис буде успішно додано до бази даних утиліти менеджера паролів. (рис. 4.20)

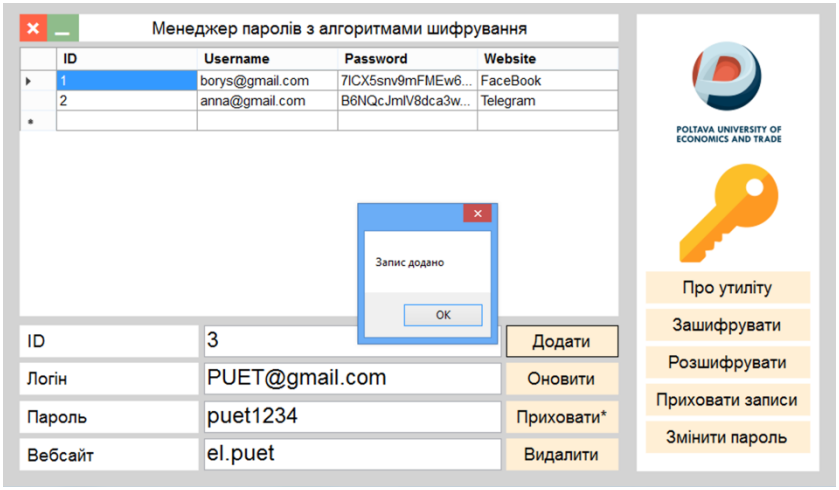


Рисунок 4.20 – вікно доданого запису.

Якщо натиснути кнопку “Приховати*”, приховається поле введення пароль і логін для додаткової безпеки. (рис. 4.21)

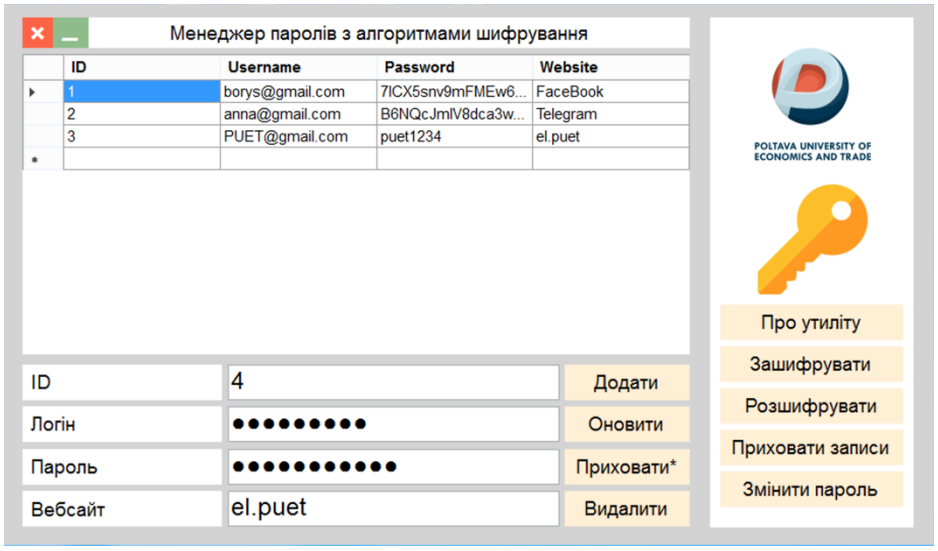


Рисунок 4.21 – функція приховання паролю і логіну.

Якщо користувачу необхідно оновити якийсь запис – є можливість, вибравши запис за допомогою стрілочки з ліва, виділиться увесь запис, потім у полях для введення вставиться інформація цього запису, користувач додасть зміни і натисне кнопку “Оновити”, додатково виведеться вікно. (рис. 4.22)

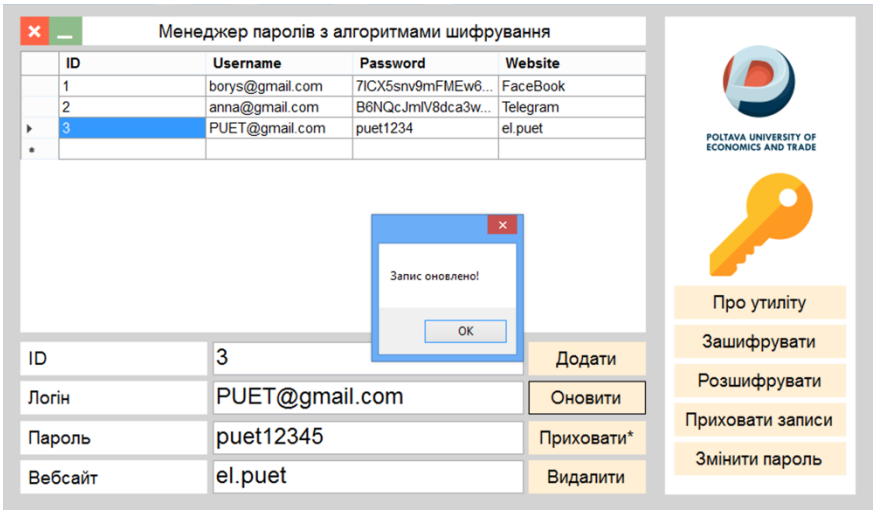


Рисунок 4.22 – оновлення запису.

Якщо потрібно видалити запис, спочатку необхідно за допомогою стрілочки виділити запис і потім натиснути кнопку “Видалити”, утиліта додатково проінформує користувача вікном. (рис. 4.23)

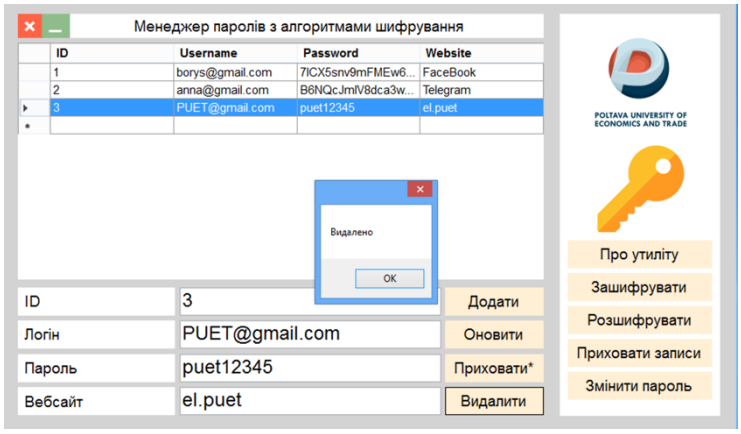


Рисунок 4.23 – видалення запису.

Якщо потрібно приховати список з записами та паролями, в утиліті також є така опція, користувачу необхідно натиснути кнопку “приховати записи”. (рис. 4.24)

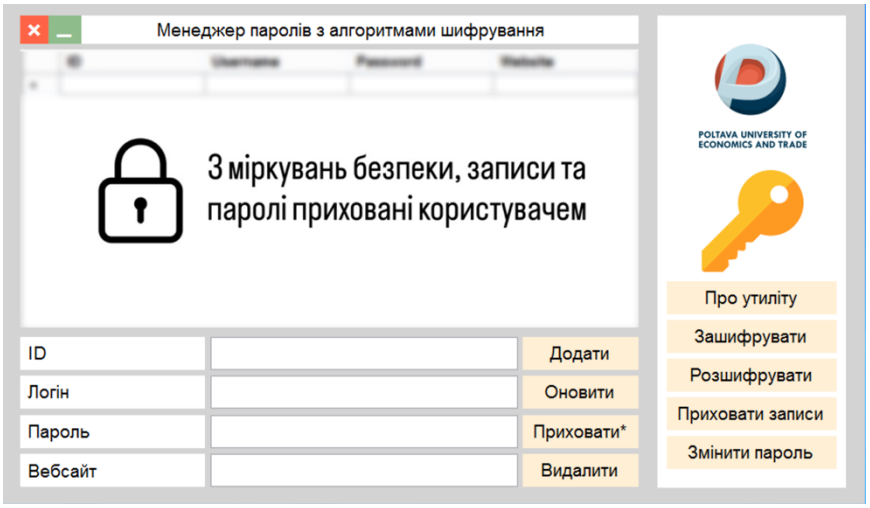


Рисунок 4.24 – приховання записів.

Шифрувати пароль потрібно перед внесенням до бази даних. Спочатку користувач вписує дані, потім натискає кнопку “Зашифрувати”, далі натиснути кнопку “Додати” і запис з зашифрованим паролем алгоритмом AES буде додано до списку. (рис. 4.25)

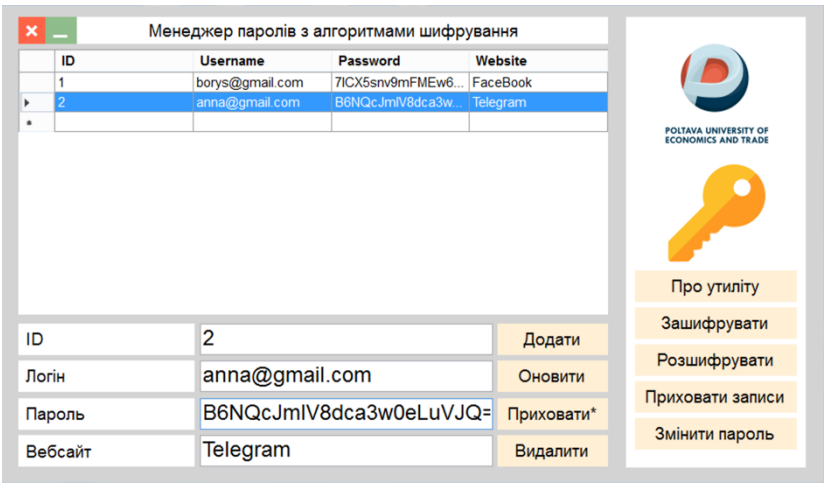


Рисунок 4.25 – шифрування записів.

Для того щоб розшифрувати запис, користувачу потрібно вибрати потрібний запис стрілочкою, потім натиснути кнопку “Розшифрувати”, пароль облікового запису буде розшифровано алгоритмом AES. (рис. 4.26)

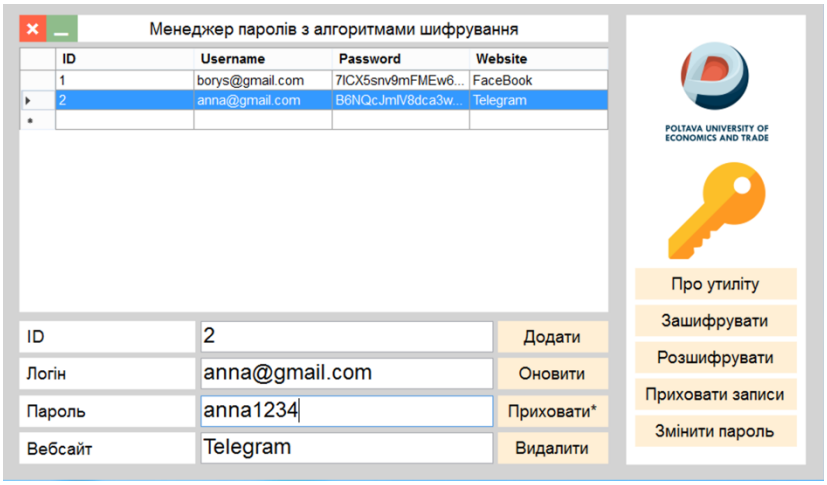


Рисунок 4.26 – розшифрування записів.

У підсумку процес розробки утиліти «Менеджер паролів» включає кілька ключових етапів:

1. Аналіз вимог та проєктування – визначення функціоналу програми, вимог до безпеки та структури збереження даних.
2. Розробка графічного інтерфейсу – створення зручного користувацького інтерфейсу за допомогою Windows Forms у середовищі Visual Studio, автоматична генерація елементів форми та налаштування прив’язки подій.
3. Реалізація алгоритмів шифрування – інтеграція криптографічних методів (AES, RSA) для безпечного зберігання та обробки паролів.
4. Робота з базою даних – організація збереження зашифрованих даних у локальній базі (SQL Server LocalDB), забезпечення функцій додавання, редагування, видалення та відображення записів.

5. Тестування та відлагодження – перевірка коректності роботи шифрування, функцій керування паролями та стабільності програми.
6. Оптимізація та підготовка до використання – покращення продуктивності, зручності користування та безпеки.

Таким чином, процес розробки поєднує технічну реалізацію, безпеку даних та зручний інтерфейс, що забезпечує ефективне та надійне керування паролями для користувача.

ВИСНОВКИ

Отже, у ході дослідження та розробки утиліти «Менеджер паролів» було встановлено, що сучасні потреби користувачів у безпечному збереженні конфіденційних даних вимагають застосування надійних криптографічних алгоритмів. Розробка програмного засобу, який забезпечує зберігання паролів у зашифрованому вигляді, дозволяє значно підвищити рівень захисту інформації, запобігти несанкціонованому доступу та мінімізувати ризики витоку або компрометації даних.

Використання алгоритму AES (Advanced Encryption Standard) як основного методу шифрування підтвердило його високу ефективність завдяки поєднанню швидкодії, криптостійкості та гнучкості у виборі довжини ключа. Це забезпечує надійний захист даних навіть у випадку спроб несанкціонованого втручання або атак.

У процесі розробки також було враховано аспекти зручності користування програмою, що є важливим фактором для широкого впровадження подібних рішень. Інтуїтивно зрозумілий інтерфейс, автоматизація процесів збереження та обробки паролів сприяють підвищенню ефективності використання утиліти.

Отримані результати свідчать про те, що впровадження утиліти «Менеджер паролів» є доцільним і ефективним рішенням для організації безпечного зберігання паролів та інших конфіденційних даних у повсякденній діяльності користувачів.

Загалом, проведена робота демонструє практичну значущість застосування сучасних методів шифрування у програмній інженерії та підкреслює важливість комплексного підходу до розробки програмних продуктів, який поєднує безпеку, надійність, продуктивність і зручність використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ольховська О.В., Черненко О. О. методичні рекомендації до виконання кваліфікаційної роботи для студентів спеціальності 122 Комп’ютерні науки освітня програма «Комп’ютерні науки» ступеня бакалавра / О. О. Черненко., Ольховська О.В. – Полтава : ПУЕТ, 2025. – 58 с.
2. Об'єктно-орієнтоване програмування [Електронний ресурс]. – Режим доступу: https://career.softserveinc.com/uk-ua/stories/what-is-object-oriented-programming-oop-explaining-four-major-principles/
3. Огляд програмного забезпечення #1 Password Manager [Електронний ресурс]. – Режим доступу: https://apps.microsoft.com/detail/9n0cqdt7zwqv?hl=uk-UA&gl=US
4. Огляд програмного забезпечення Passwarden [Електронний ресурс]. – Режим доступу: https://apps.microsoft.com/detail/9wzdnmw8?hl=uk-UA&gl=UA
5. Шифрування: типи і алгоритми [Електронний ресурс]. – Режим доступу: https://hostpro.ua/wiki/ua/security/encryption-types-algorithms/ -
6. Розробка програмного забезпечення [Електронний ресурс]. – Режим доступу: https://studfile.net/preview/5118185
7. Загальні відомості про AES-шифрування [Електронний ресурс]. – Режим доступу: https://techvizor.info/zagalni-vidomosti-pro-aes-shyfruvannya/
8. Мова програмування C# [Електронний ресурс]. – Режим доступу: https://abitap.com/category/c/#google_vignette
9. C# Підручник [Електронний ресурс]. – Режим доступу: https://www.devopsschool.com/blog/what-is-c#-net-and-use-cases-of-c#-net/
10. Створення програми в Visual Studio 2022 [Електронний ресурс]. – Режим доступу: https://learn.ztu.edu.ua/mod/page/view.php?id=9976

11. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: ДСТУ 7.1-2006. – [Чинний від 2007-07-01]. – К. : Держспоживстандарт України, 2007. – 47 с.
12. Microsoft Visual Studio [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio
13. development with Visual Studio [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/?view=vs-2022
14. Get Started [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/cs/cs_getstarted.php
15. Visual Studio [Електронний ресурс]. – Режим доступу: https://www.halvorsen.blog/documents/programming/csharp/csharp.php
16. Створення застосунків в Visual Studio 2019 [Електронний ресурс]. – Режим доступу: https://learn.ztu.edu.ua/mod/page/view.php?id=9976
17. Microsoft Visual Studio [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio
18. Що таке AES-шифрування? [Електронний ресурс]. – Режим доступу: https://memory.net.ua/info/aes-shifruvannja?srsltid=AfmBOoqf42vORj0GC6MMpo9u40gHSdTLKCHdF1PYpWCvZkKhID_x-K8w
19. Об'єктно-орієнтоване програмування (Advanced Encryption Standard instructions) [Електронний ресурс]. – Режим доступу: https://vseosvita.ua/test/elementy-teorii-obiektno-oriientovanoho.
20. Стандарт AES NIST: Стандарт шифрування [Електронний ресурс]. – Режим доступу: https://lazarusalliance.com/uk/what-is-advanced-encryption-standard-aes-and-how-is-it-related-to-nist/

ДОДАТОК А. КОД ПРОГРАМИ

```
namespace DatabseInsertApp
{
   partial class Form1
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         System.Windows.Forms.DataGridViewCellStyle dataGridViewCellStyle5 = new
System.Windows.Forms.DataGridViewCellStyle();
         System.Windows.Forms.DataGridViewCellStyle dataGridViewCellStyle6 = new
System.Windows.Forms.DataGridViewCellStyle();
         System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Form1));
         this.txtID = new System.Windows.Forms.TextBox();
         this.txtUsername = new System.Windows.Forms.TextBox();
         this.txtPassword = new System.Windows.Forms.TextBox();
         this.btnInsert = new System.Windows.Forms.Button();
         this.btnUpdate = new System.Windows.Forms.Button();
         this.btnDelete = new System.Windows.Forms.Button();
         this.dataGridView1 = new System.Windows.Forms.DataGridView();
         this.panel1 = new System.Windows.Forms.Panel();
         this.hideData_btn = new System.Windows.Forms.Button();
         this.decrypt_btn = new System.Windows.Forms.Button();
         this.encrypt_btn = new System.Windows.Forms.Button();
         this.About_btn = new System.Windows.Forms.Button();
         this.pictureBox2 = new System.Windows.Forms.PictureBox();
```

```
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.txtWeb = new System.Windows.Forms.TextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.panel2 = new System.Windows.Forms.Panel();
            this.panel3 = new System.Windows.Forms.Panel();
            this.label5 = new System.Windows.Forms.Label();
            this.panel4 = new System.Windows.Forms.Panel();
            this.label2 = new System.Windows.Forms.Label();
            this.panel5 = new System.Windows.Forms.Panel();
            this.label3 = new System.Windows.Forms.Label();
            this.panel6 = new System.Windows.Forms.Panel();
            this.button1 = new System.Windows.Forms.Button();
            this.close_btn = new System.Windows.Forms.Button();
            this.label4 = new System.Windows.Forms.Label();
            this.hidePass_btn = new System.Windows.Forms.Button();
            this.pictureBox3 = new System.Windows.Forms.PictureBox();
            this.pictureBox4 = new System.Windows.Forms.PictureBox();
            ((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).BeginInit();
            this.panel1.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox2)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
            this.panel2.SuspendLayout();
            this.panel3.SuspendLayout();
            this.panel4.SuspendLayout();
            this.panel5.SuspendLayout();
            this.panel6.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox3)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox4)).BeginInit();
            this.SuspendLayout();
            //
            // txtID
            //
            this.txtID.Font = new System.Drawing.Font("Arial", 18F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.txtID.Location = new System.Drawing.Point(221, 354);
            this.txtID.Name = "txtID";
            this.txtID.Size = new System.Drawing.Size(326, 35);
            this.txtID.TabIndex = 3;
            //
            // txtUsername
            //
            this.txtUsername.Font = new System.Drawing.Font("Arial", 18F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.txtUsername.Location = new System.Drawing.Point(221, 395);
            this.txtUsername.Name = "txtUsername";
            this.txtUsername.Size = new System.Drawing.Size(326, 35);
            this.txtUsername.TabIndex = 4;
            this.txtUsername.UseSystemPasswordChar = true;
            //
            // txtPassword
            //
```

```
            this.txtPassword.Font = new System.Drawing.Font("Arial", 18F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.txtPassword.Location = new System.Drawing.Point(221, 437);
            this.txtPassword.Name = "txtPassword";
            this.txtPassword.Size = new System.Drawing.Size(326, 35);
            this.txtPassword.TabIndex = 5;
            this.txtPassword.UseSystemPasswordChar = true;
            //
            // btnInsert
            //
            this.btnInsert.BackColor = System.Drawing.Color.PapayaWhip;
            this.btnInsert.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnInsert.FlatAppearance.BorderSize = 0;
            this.btnInsert.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.btnInsert.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.btnInsert.ForeColor = System.Drawing.Color.Black;
            this.btnInsert.Location = new System.Drawing.Point(552, 354);
            this.btnInsert.Name = "btnInsert";
            this.btnInsert.Size = new System.Drawing.Size(123, 35);
            this.btnInsert.TabIndex = 6;
            this.btnInsert.Text = "Додати";
            this.btnInsert.UseVisualStyleBackColor = false;
            this.btnInsert.Click += new System.EventHandler(this.btnInsert_Click);
            //
            // btnUpdate
            //
            this.btnUpdate.BackColor = System.Drawing.Color.PapayaWhip;
            this.btnUpdate.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnUpdate.FlatAppearance.BorderSize = 0;
            this.btnUpdate.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.btnUpdate.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.btnUpdate.ForeColor = System.Drawing.Color.Black;
            this.btnUpdate.Location = new System.Drawing.Point(552, 395);
            this.btnUpdate.Name = "btnUpdate";
            this.btnUpdate.Size = new System.Drawing.Size(123, 35);
            this.btnUpdate.TabIndex = 7;
            this.btnUpdate.Text = "Оновити";
            this.btnUpdate.UseVisualStyleBackColor = false;
            this.btnUpdate.Click += new System.EventHandler(this.btnUpdate_Click);
            //
            // btnDelete
            //
            this.btnDelete.BackColor = System.Drawing.Color.PapayaWhip;
            this.btnDelete.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnDelete.FlatAppearance.BorderSize = 0;
            this.btnDelete.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.btnDelete.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.btnDelete.ForeColor = System.Drawing.Color.Black;
            this.btnDelete.Location = new System.Drawing.Point(552, 478);
```

```
            this.btnDelete.Name = "btnDelete";
            this.btnDelete.Size = new System.Drawing.Size(123, 35);
            this.btnDelete.TabIndex = 8;
            this.btnDelete.Text = "Видалити";
            this.btnDelete.UseVisualStyleBackColor = false;
            this.btnDelete.Click += new System.EventHandler(this.btnDelete_Click);
            //
            // dataGridView1
            //
            this.dataGridView1.AutoSizeColumnsMode =
System.Windows.Forms.DataGridViewAutoSizeColumnsMode.Fill;
            this.dataGridView1.BackgroundColor = System.Drawing.SystemColors.Window;
            this.dataGridView1.BorderStyle = System.Windows.Forms.BorderStyle.None;
            dataGridViewCellStyle5.Alignment =
System.Windows.Forms.DataGridViewContentAlignment.MiddleLeft;
            dataGridViewCellStyle5.BackColor = System.Drawing.SystemColors.Control;
            dataGridViewCellStyle5.Font = new System.Drawing.Font("Arial", 11.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            dataGridViewCellStyle5.ForeColor = System.Drawing.SystemColors.WindowText;
            dataGridViewCellStyle5.SelectionBackColor =
System.Drawing.SystemColors.Highlight;
            dataGridViewCellStyle5.SelectionForeColor =
System.Drawing.SystemColors.HighlightText;
            dataGridViewCellStyle5.WrapMode =
System.Windows.Forms.DataGridViewTriState.True;
            this.dataGridView1.ColumnHeadersDefaultCellStyle = dataGridViewCellStyle5;
            this.dataGridView1.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            dataGridViewCellStyle6.Alignment =
System.Windows.Forms.DataGridViewContentAlignment.MiddleLeft;
            dataGridViewCellStyle6.BackColor = System.Drawing.SystemColors.Window;
            dataGridViewCellStyle6.Font = new System.Drawing.Font("Arial", 11.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            dataGridViewCellStyle6.ForeColor = System.Drawing.SystemColors.ControlText;
            dataGridViewCellStyle6.SelectionBackColor =
System.Drawing.SystemColors.Highlight;
            dataGridViewCellStyle6.SelectionForeColor =
System.Drawing.SystemColors.HighlightText;
            dataGridViewCellStyle6.WrapMode =
System.Windows.Forms.DataGridViewTriState.False;
            this.dataGridView1.DefaultCellStyle = dataGridViewCellStyle6;
            this.dataGridView1.EditMode =
System.Windows.Forms.DataGridViewEditMode.EditProgrammatically;
            this.dataGridView1.Location = new System.Drawing.Point(19, 49);
            this.dataGridView1.Name = "dataGridView1";
            this.dataGridView1.Size = new System.Drawing.Size(656, 295);
            this.dataGridView1.TabIndex = 9;
            this.dataGridView1.RowHeaderMouseClick += new
System.Windows.Forms.DataGridViewCellMouseEventHandler(this.dataGridView1_RowHe
aderMouseClick);
            //
            // panel1
```

```
            //
            this.panel1.BackColor = System.Drawing.Color.White;
            this.panel1.Controls.Add(this.hideData_btn);
            this.panel1.Controls.Add(this.decrypt_btn);
            this.panel1.Controls.Add(this.encrypt_btn);
            this.panel1.Controls.Add(this.About_btn);
            this.panel1.Controls.Add(this.pictureBox2);
            this.panel1.Controls.Add(this.pictureBox1);
            this.panel1.Location = new System.Drawing.Point(695, 13);
            this.panel1.Name = "panel1";
            this.panel1.Size = new System.Drawing.Size(200, 501);
            this.panel1.TabIndex = 10;
            //
            // hideData_btn
            //
            this.hideData_btn.BackColor = System.Drawing.Color.PapayaWhip;
            this.hideData_btn.Cursor = System.Windows.Forms.Cursors.Hand;
            this.hideData_btn.FlatAppearance.BorderSize = 0;
            this.hideData_btn.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.hideData_btn.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.hideData_btn.ForeColor = System.Drawing.Color.Black;
            this.hideData_btn.Location = new System.Drawing.Point(10, 428);
            this.hideData_btn.Name = "hideData_btn";
            this.hideData_btn.Size = new System.Drawing.Size(180, 35);
            this.hideData_btn.TabIndex = 20;
            this.hideData_btn.Text = "Приховати записи";
            this.hideData_btn.UseVisualStyleBackColor = false;
            this.hideData_btn.Click += new System.EventHandler(this.hideData_btn_Click);
            //
            // decrypt_btn
            //
            this.decrypt_btn.BackColor = System.Drawing.Color.PapayaWhip;
            this.decrypt_btn.Cursor = System.Windows.Forms.Cursors.Hand;
            this.decrypt_btn.FlatAppearance.BorderSize = 0;
            this.decrypt_btn.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.decrypt_btn.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.decrypt_btn.ForeColor = System.Drawing.Color.Black;
            this.decrypt_btn.Location = new System.Drawing.Point(10, 387);
            this.decrypt_btn.Name = "decrypt_btn";
            this.decrypt_btn.Size = new System.Drawing.Size(180, 35);
            this.decrypt_btn.TabIndex = 19;
            this.decrypt_btn.Text = "Розшифрувати";
            this.decrypt_btn.UseVisualStyleBackColor = false;
            this.decrypt_btn.Click += new System.EventHandler(this.decrypt_btn_Click);
            //
            // encrypt_btn
            //
            this.encrypt_btn.BackColor = System.Drawing.Color.PapayaWhip;
            this.encrypt_btn.Cursor = System.Windows.Forms.Cursors.Hand;
            this.encrypt_btn.FlatAppearance.BorderSize = 0;
```

```
            this.encrypt_btn.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.encrypt_btn.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.encrypt_btn.ForeColor = System.Drawing.Color.Black;
            this.encrypt_btn.Location = new System.Drawing.Point(10, 346);
            this.encrypt_btn.Name = "encrypt_btn";
            this.encrypt_btn.Size = new System.Drawing.Size(180, 35);
            this.encrypt_btn.TabIndex = 18;
            this.encrypt_btn.Text = "Зашифрувати";
            this.encrypt_btn.UseVisualStyleBackColor = false;
            this.encrypt_btn.Click += new System.EventHandler(this.encrypt_btn_Click);
            //
            // About_btn
            //
            this.About_btn.BackColor = System.Drawing.Color.PapayaWhip;
            this.About_btn.Cursor = System.Windows.Forms.Cursors.Hand;
            this.About_btn.FlatAppearance.BorderSize = 0;
            this.About_btn.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.About_btn.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.About_btn.ForeColor = System.Drawing.Color.Black;
            this.About_btn.Location = new System.Drawing.Point(10, 305);
            this.About_btn.Name = "About_btn";
            this.About_btn.Size = new System.Drawing.Size(180, 35);
            this.About_btn.TabIndex = 17;
            this.About_btn.Text = "Про утиліту";
            this.About_btn.UseVisualStyleBackColor = false;
            this.About_btn.Click += new System.EventHandler(this.About_btn_Click);
            //
            // pictureBox2
            //
            this.pictureBox2.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox2.Image")));
            this.pictureBox2.Location = new System.Drawing.Point(37, 187);
            this.pictureBox2.Name = "pictureBox2";
            this.pictureBox2.Size = new System.Drawing.Size(128, 108);
            this.pictureBox2.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;
            this.pictureBox2.TabIndex = 1;
            this.pictureBox2.TabStop = false;
            //
            // pictureBox1
            //
            this.pictureBox1.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
            this.pictureBox1.Location = new System.Drawing.Point(17, 35);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(168, 142);
            this.pictureBox1.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox1.TabIndex = 0;
            this.pictureBox1.TabStop = false;
            //
```

```
            // txtWeb
            //
            this.txtWeb.Font = new System.Drawing.Font("Arial", 18F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.txtWeb.Location = new System.Drawing.Point(221, 478);
            this.txtWeb.Name = "txtWeb";
            this.txtWeb.Size = new System.Drawing.Size(326, 35);
            this.txtWeb.TabIndex = 11;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.BackColor = System.Drawing.Color.White;
            this.label1.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label1.Location = new System.Drawing.Point(4, 6);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(30, 22);
            this.label1.TabIndex = 13;
            this.label1.Text = "ID";
            //
            // panel2
            //
            this.panel2.BackColor = System.Drawing.Color.White;
            this.panel2.Controls.Add(this.label1);
            this.panel2.Location = new System.Drawing.Point(19, 354);
            this.panel2.Name = "panel2";
            this.panel2.Size = new System.Drawing.Size(196, 35);
            this.panel2.TabIndex = 11;
            //
            // panel3
            //
            this.panel3.BackColor = System.Drawing.Color.White;
            this.panel3.Controls.Add(this.label5);
            this.panel3.Location = new System.Drawing.Point(19, 395);
            this.panel3.Name = "panel3";
            this.panel3.Size = new System.Drawing.Size(196, 35);
            this.panel3.TabIndex = 14;
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.BackColor = System.Drawing.Color.White;
            this.label5.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label5.Location = new System.Drawing.Point(4, 6);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(54, 22);
            this.label5.TabIndex = 13;
            this.label5.Text = "Логін";
            //
            // panel4
```

```
            //
            this.panel4.BackColor = System.Drawing.Color.White;
            this.panel4.Controls.Add(this.label2);
            this.panel4.Location = new System.Drawing.Point(19, 437);
            this.panel4.Name = "panel4";
            this.panel4.Size = new System.Drawing.Size(196, 35);
            this.panel4.TabIndex = 15;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.BackColor = System.Drawing.Color.White;
            this.label2.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label2.Location = new System.Drawing.Point(4, 6);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(76, 22);
            this.label2.TabIndex = 13;
            this.label2.Text = "Пароль";
            //
            // panel5
            //
            this.panel5.BackColor = System.Drawing.Color.White;
            this.panel5.Controls.Add(this.label3);
            this.panel5.Location = new System.Drawing.Point(19, 478);
            this.panel5.Name = "panel5";
            this.panel5.Size = new System.Drawing.Size(196, 35);
            this.panel5.TabIndex = 15;
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.BackColor = System.Drawing.Color.White;
            this.label3.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label3.Location = new System.Drawing.Point(4, 7);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(83, 22);
            this.label3.TabIndex = 13;
            this.label3.Text = "Вебсайт";
            //
            // panel6
            //
            this.panel6.BackColor = System.Drawing.Color.White;
            this.panel6.Controls.Add(this.button1);
            this.panel6.Controls.Add(this.close_btn);
            this.panel6.Controls.Add(this.label4);
            this.panel6.Location = new System.Drawing.Point(19, 13);
            this.panel6.Name = "panel6";
            this.panel6.Size = new System.Drawing.Size(656, 30);
            this.panel6.TabIndex = 16;
            //
```

```
            // button1
            //
            this.button1.BackColor = System.Drawing.Color.DarkSeaGreen;
            this.button1.Cursor = System.Windows.Forms.Cursors.Hand;
            this.button1.FlatAppearance.BorderSize = 0;
            this.button1.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.button1.Font = new System.Drawing.Font("Arial", 20.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.button1.ForeColor = System.Drawing.Color.White;
            this.button1.Location = new System.Drawing.Point(30, -13);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(35, 45);
            this.button1.TabIndex = 18;
            this.button1.Text = "_";
            this.button1.UseVisualStyleBackColor = false;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // close_btn
            //
            this.close_btn.BackColor = System.Drawing.Color.Tomato;
            this.close_btn.Cursor = System.Windows.Forms.Cursors.Hand;
            this.close_btn.FlatAppearance.BorderSize = 0;
            this.close_btn.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.close_btn.Font = new System.Drawing.Font("Arial", 20.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.close_btn.ForeColor = System.Drawing.Color.White;
            this.close_btn.Location = new System.Drawing.Point(0, -4);
            this.close_btn.Name = "close_btn";
            this.close_btn.Size = new System.Drawing.Size(33, 37);
            this.close_btn.TabIndex = 17;
            this.close_btn.Text = "×";
            this.close_btn.UseVisualStyleBackColor = false;
            this.close_btn.Click += new System.EventHandler(this.close_btn_Click);
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.BackColor = System.Drawing.Color.White;
            this.label4.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label4.Location = new System.Drawing.Point(141, 4);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(421, 22);
            this.label4.TabIndex = 14;
            this.label4.Text = "Менеджер паролів з алгоритмами шифрування";
            //
            // hidePass_btn
            //
            this.hidePass_btn.BackColor = System.Drawing.Color.PapayaWhip;
            this.hidePass_btn.Cursor = System.Windows.Forms.Cursors.Hand;
            this.hidePass_btn.FlatAppearance.BorderSize = 0;
            this.hidePass_btn.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
```

```
            this.hidePass_btn.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.hidePass_btn.ForeColor = System.Drawing.Color.Black;
            this.hidePass_btn.Location = new System.Drawing.Point(552, 437);
            this.hidePass_btn.Name = "hidePass_btn";
            this.hidePass_btn.Size = new System.Drawing.Size(123, 35);
            this.hidePass_btn.TabIndex = 18;
            this.hidePass_btn.Text = "Приховати*";
            this.hidePass_btn.UseVisualStyleBackColor = false;
            this.hidePass_btn.Click += new System.EventHandler(this.hidePass_btn_Click);
            //
            // pictureBox3
            //
            this.pictureBox3.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox3.Image")));
            this.pictureBox3.Location = new System.Drawing.Point(19, 49);
            this.pictureBox3.Name = "pictureBox3";
            this.pictureBox3.Size = new System.Drawing.Size(656, 295);
            this.pictureBox3.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox3.TabIndex = 19;
            this.pictureBox3.TabStop = false;
            this.pictureBox3.Visible = false;
            //
            // pictureBox4
            //
            this.pictureBox4.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox4.Image")));
            this.pictureBox4.Location = new System.Drawing.Point(19, 49);
            this.pictureBox4.Name = "pictureBox4";
            this.pictureBox4.Size = new System.Drawing.Size(656, 465);
            this.pictureBox4.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox4.TabIndex = 20;
            this.pictureBox4.TabStop = false;
            this.pictureBox4.Click += new System.EventHandler(this.pictureBox4_Click);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.Color.LightGray;
            this.ClientSize = new System.Drawing.Size(915, 531);
            this.Controls.Add(this.pictureBox4);
            this.Controls.Add(this.pictureBox3);
            this.Controls.Add(this.hidePass_btn);
            this.Controls.Add(this.panel6);
            this.Controls.Add(this.panel5);
            this.Controls.Add(this.panel4);
            this.Controls.Add(this.panel3);
            this.Controls.Add(this.panel2);
            this.Controls.Add(this.txtWeb);
```

```
    this.Controls.Add(this.panel1);
    this.Controls.Add(this.dataGridView1);
    this.Controls.Add(this.btnDelete);
    this.Controls.Add(this.btnUpdate);
    this.Controls.Add(this.btnInsert);
    this.Controls.Add(this.txtPassword);
    this.Controls.Add(this.txtUsername);
    this.Controls.Add(this.txtID);
    this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
    this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
    this.MaximizeBox = false;
    this.Name = "Form1";
    this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
    this.Load += new System.EventHandler(this.Form1_Load);
    ((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).EndInit();
    this.panel1.ResumeLayout(false);
    ((System.ComponentModel.ISupportInitialize)(this.pictureBox2)).EndInit();
    ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
    this.panel2.ResumeLayout(false);
    this.panel2.PerformLayout();
    this.panel3.ResumeLayout(false);
    this.panel3.PerformLayout();
    this.panel4.ResumeLayout(false);
    this.panel4.PerformLayout();
    this.panel5.ResumeLayout(false);
    this.panel5.PerformLayout();
    this.panel6.ResumeLayout(false);
    this.panel6.PerformLayout();
    ((System.ComponentModel.ISupportInitialize)(this.pictureBox3)).EndInit();
    ((System.ComponentModel.ISupportInitialize)(this.pictureBox4)).EndInit();
    this.ResumeLayout(false);
    this.PerformLayout();

}

#endregion
private System.Windows.Forms.TextBox txtID;
private System.Windows.Forms.TextBox txtUsername;
private System.Windows.Forms.TextBox txtPassword;
private System.Windows.Forms.Button btnInsert;
private System.Windows.Forms.Button btnUpdate;
private System.Windows.Forms.Button btnDelete;
private System.Windows.Forms.DataGridView dataGridView1;
private System.Windows.Forms.Panel panel1;
private System.Windows.Forms.TextBox txtWeb;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Panel panel2;
private System.Windows.Forms.Panel panel3;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.Panel panel4;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Panel panel5;
```

```
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Panel panel6;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Button close_btn;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.PictureBox pictureBox2;
        private System.Windows.Forms.Button About_btn;
        private System.Windows.Forms.Button hidePass_btn;
        private System.Windows.Forms.Button encrypt_btn;
        private System.Windows.Forms.Button decrypt_btn;
        private System.Windows.Forms.Button hideData_btn;
        private System.Windows.Forms.PictureBox pictureBox3;
        private System.Windows.Forms.PictureBox pictureBox4;
    }
}
```