Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»
Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка
до кваліфікаційної роботи
бакалавра

на тему: «Розробка застосунку для збору внутрішньої інформації Chrome браузеру»
за освітньою програмою 12 Інженерія програмного забезпечення
зі спеціальності: 121 Інженерія програмного забезпечення

Виконав: студент групи ПЗ1912: _____ / Максим КУРОЧКА /
(підпис)

Керівник: доцент _____ / Олександра ГОРБОВА /
(підпис)

Нормоконтролер: доцент _____ / Світлана ВОЛКОВА /
(підпис)

Засвідчую, що у цій роботі немає
запозичень з праць інших авторів
без відповідних посилань.
Студент

Дніпро – 2023 рік

Український державний університет науки і технологій

Факультет “Комп’ютерні технології і системи”
Кафедра “Комп’ютерні інформаційні технології”
Спеціальність “Інженерія програмного забезпечення”

«ДО ЗАХИСТУ»
Завідувач кафедри
_____ Вадим ГОРЯЧКІН
(підпис) (ПІБ)
2023 р._____ «_____»

ЗАВДАННЯ
до дипломної роботи на здобуття ОС «бакалавр»

студента групи ПЗ1912 Курочка Максим Віталійович
1 Тема дипломної роботи: Розробка застосунку для збору внутрішньої інформації Chrome браузеру
затверджена наказом по університету від «07» грудня 2020 р. № 77.
2 Термін подання студентом закінченої роботи 19.06.2023
3 Вихідні дані до дипломної роботи: відображена інформація у графічному інтерфейсі користувача, експортовані файли у csv форматі з даними історії пошука та логінів, експортовані кеш файли.
4 Зміст пояснювальної записки (перелік питань до розробки): вступ, теоретична частина, збір та аналіз вимог до програмного забезпечення, зовнішнє і внутрішнє проектування, розробка програми, тестування та налагодження, висновки, список використаних джерел.
5 Перелік демонстраційного матеріалу: презентація, відео-демонстрація роботи програми.

Вихідними даними виступають відображена інформація, експортовані файли csv формату та кеш-файли.

КАЛЕНДАРНИЙ ПЛАН

№ пор.	Зміст роботи (розділу)	Термін виконання розділів роботи	Примітка
1	Вступ	20.04.2023	
2	Аналіз сучасного стану вирішення обраної задачі та програмно-апаратного забезпечення	24.04.2023	
3	Збір вимог до програм, опис бізнес-процесів та розробка прототипів, постановка задачі	05.05.2023	30%
4	Зовнішнє та внутрішнє проектування	10.05.2023	
5	Розробка програмного забезпечення	18.05.2023	60%
6	Тестування та налагодження програмного забезпечення	27.05.2023	
7	Оформлення пояснювальної записки	11.06.2023	
8	Розробка демонстраційних матеріалів	15.06.2023	100%

Дата видачі завдання «21» вересня 2023р.

Керівник дипломної роботи _____ Олександра ГОРБОВА

Завдання прийняв до виконання _____ Максим КУРОЧКА

РЕФЕРАТ

Пояснювальна записка складається з 9 розділів:

- вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 2 сторінки;
- теоретична частина – в даному розділі описується інформації стосовно розташування та формату збереження даних історії пошуків, логінів та кешу; Складається з 12 сторінок;
- збір та аналіз вимог до програмного забезпечення – у цьому розділи описуються аналоги програми, результати анкетування зацікавлених сторін для формування найбільш повних вимог до програмного забезпечення. Складається з 11 сторінок;
- зовнішнє і внутрішнє проектування – у цьому розділі наведено формалізація задачі, огляд вхідних і вихідних даних, розробка фізичного проекту, приводиться опис об’єктно-орієнтованого проектування, ескізи форми та проектування інтерфейсу користувача, аналіз проекту, проектування динаміки системи, вибір мови програмування. Складається з 16 сторінок;
- розробка програми – у цьому розділі наведено розбиття програми за допомогою метода покрокової деталізації, алгоритми функцій та їх специфікації. Складається з 14 сторінок;
- тестування та налагодження – включає в себе результати тестування методами «білої» та «чорної» скриньки. Також наведено аналіз помилок, що зустрів під час тестування програми, опис їх вплив на систему та вирішення проблеми. Складається з 11 сторінок;
- висновки. Складається з 1 сторінки;
- додатки – містить технічне завдання і робочий проект;
- список літератури – включає в себе бібліографічний список використаної літератури. Складає 1 сторінка;

Кількість таблиць: 18 таблиць.

Кількість рисунків: 34 рисунків.

Ключові слова: історія пошуку, логін, кеш, шифрування, майстер ключ, пошук, ключове слово, сортування, запит.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

AES-256-GCM – Advanced Encryption Standard in Galios/Counter Mode

API – Application Programming Interface

DPAPI – Data Protection Application Programming Interface

DLL – Dynamic Link Library

CSV – comma-separated values

JSON – JavaScript Object Notation

SQL – Structured Query Language

UI – User Interface

UML – Unified Modeling Language

XML – Extensible Markup Language

ПЗ – Програмне Забезпечення

ВСТУП

Актуальність теми. Браузери вже давно стали частиною повсякденного життя, вони завжди використовуються для пошуку якоїсь інформації в Інтернеті. Браузери також можна використовувати і для навігації по файловій системі операційної системи.

Браузери зберігають дуже багато приватної інформації: пошукові запити, ім’я користувача, паролі, закладки, данні кредитової карти, зображення, відео та інше. Закладки та пошукові запити (історія браузера) дають уяву про інтереси користувача. Тобто, браузери мають доступ до приватного життя користувачів і можуть дати доволі точно оцінку людині, як особі та про її інтереси. Таким чином, вони грають дуже велику роль в криміналістиці із-за об’єму даних, що вони зберігають. Але не завжди є змога переглянути дані браузера через сам браузер та є такі дані, що взагалі не відображаються навіть через сам браузер. Наприклад, є ситуація: потрібно якось переглянути дані браузера, що були вже видаленні з самого комп’ютера, але нам вдалось відновити їх за допомогою спеціальної утілити, але переглянути їх немає змоги. Саме тоді і прийде на допомогу ця програма, яка з легкістю зможе надати доступ до інформації, що зберігається в відновлених файлах.

Об’єктом дослідження є Google Chrome браузер, що зберігає локально деяку дуже приватну інформацію користувача.

Предметом дослідження є внутрішні дані Google Chrome браузеру, що зберігаються локально на комп’ютері

Мета роботи. Ознайомитись з видами інформації, що Google Chrome зберігає на комп’ютері локально, розробити програму для парсингу внутрішніх даних Google Chrome браузеру, надати змогу пошуку серед переглянутої інформації.

Експлуатаційне призначення. З допомогою розробленої програми можна отримати конфіденційні дані самого популярного веб-браузеру у світі

– Google Chrome. Можна вилучити історію пошуків, збережені паролі, збережені відео та фото дані. Ці дані можуть бути використані як у криміналістиці, так і повсякденному житті, бо бувають випадки коли потрібно якось витягти дані браузера, але з будь-яких причин немає змоги переглянути їх через сам браузер.

РОЗДІЛ 1. ТЕОРЕТИЧНА ЧАСТИНА

1.1 Особливості та характеристики даних, що зберігаються локально

Коли йде мова про внутрішні дані Google Chrome браузера, що зберігаються локально, то це питання про історію пошукових запитів, закладок, перелік збережених файлів, даних кешу, збережених паролі тощо

Ці дані являють собою файли, що зберігаються в певних папках операційної системи та мають певний формат.

Кожний браузер зберігає свої файли в відповідних для нього місцях і для кожного браузера ці місця різні, але здебільшого файли мають схожий формат даних.

Перелік найпоширеніших даних, що браузери зберігають локально:

- історія пошукових запитів;
- дані для автозаповнення − це дані, які браузер пропонує на основі того, що користувач найчастіше шукає;
- закладки;
- розширення та доповнення;
- кеш – це дані, браузер створює при навігації на веб-сайтах з багатьох причин, наприклад, щоб прискорити завантаження веб-сайтів;
- збережені паролі та логіни;
- фавікони − це маленькі значки, які можна знайти на вкладках, URL-адресах, закладках тощо;
- дані форми – це всі дані що вводиться всередині форм, вони часто зберігається браузером

1.2 Історія пошукових запитів

Файл, що зберігає історію пошукових запитів є SQLite базу даних. Chrome не додає розширення до своїх файлів; але бази даних SQLite мають заголовок файлу «SQLite format 3», що робить їх легкими для розпізнавання. База даних з історією пошуку має такий шлях: “C:\Users\<username>\ AppData\Local\Google\Chrome\User Data\Default\History”.

Дана база даних містить 18 різних таблиць, але серед них нас цікавить таблиця з назвою “urls”. У табл. 1.1 наведено інформацію по полям даних у таблиці з бази даних.

Таблиця 1.1 – Структура urls таблиці

Назва	Тип даних	Сенс
id	INTEGER	Первинний ключ таблиці. Це використовується в багатьох інших таблицях для посилання на цю таблицю.
url	LONGVARCHAR	Зберігає унікальну відвідану URL-адресу.
title	LONGVARCHAR	Зберігає назву сторінки URL.
visit_count	INTEGER	Зберігає загальну кількість відвідувань для цієї URL-адреси.
typed_count	INTEGER	Зберігає кількість разів, коли URL-адресу було введено вручну.
last_visit_time	INTEGER	Зберігає час останнього відвідування URL-адреси. Це зберігається у власній варіації часу Google.
hidden	INTEGER	Вказує, чи буде URL-адреса відображатися функцією автозаповнення. Значення 1 збереже його прихованим, а 0 відобразить.

З вищенаведених даних потрібно брати url, title та last_visit_time поля з таблиці urls. З першим та другим полями немає ніяких проблем, бо ці поля вже без якоїсь обробки дають важливу інформацію, а ось з останнім параметром є проблеми. last_visit_time поле відформатована як кількість мікросекунд із 1 січня 1601 року.

1.3 Паролі

Файл, що зберігаю логіни та паролі є SQLite – база даних. Chrome не додає розширення до своїх файлів; але, на щастя, бази даних SQLite мають заголовок файлу «SQLite format 3», що робить їх легкими для розпізнавання. База даних з історією пошуку має такий шлях: “C:\Users\<username>\AppData\Local\Google\Chrome\User Data\Default\Login Data”.

Дана база даних містить 9 різних таблиць, але серед них найцікавіша інформація розташована в таблиці з назвою “logins”. У табл. 1.2 наведено частину корисної інформації по полям даних у таблиці з бази даних.

Таблиця 1.2 – Частина структури logins таблиці

Назва	Тип даних	Сенс
origin_url	VARCHAR	Домен, для входу.
action_url	VARCHAR	Домен (або веб-сторінка, пов’язана з функцією входу в цей домен).
username_element	VARCHAR	Значення, що залежить від домену та залежить від того, як домен назвав поле «введення користувача» у своєму коді. Тут можна побачити різні значення, такі як «ім’я користувача», «електронна адреса» або «ідентифікатор».
username_value	VARCHAR	Значення, що зберігає ім'я користувача, який намагався увійти до свого облікового запису, буде збережено. Різні імена користувачів в одному домені створюватимуть окремі записи.

Продовження таблиці 1.2

password_element	VARCHAR	Значення залежить від домену та залежить від того, як домен назвав поле «введення пароля» у своєму коді. Таким чином, можна побачити різні значення, такі як «пароль», або «пас», або «сеанс[пароль]».
password_value	BLOB	Значення, для збереження паролю користувача, який намагався увійти в свій обліковий запис. Пароль зберігається не у вигляді простого тексту, а у форматі blob.
signon_realm	VARCHAR	Домен (або веб-сторінка, пов'язана з функцією входу в цей домен).
date_created	INTEGER	Точна позначка часу (також у форматі Google Chrome).
blacklisted_by_user	INTEGER	Правда чи хибність (1 або 0). Звичайно, якщо користувач вибере варіант 1, це значення буде 0.
id	INTEGER	Ідентифікатор конкретного запису в таблиці «логіни». Кожен збільшує ідентифікатор на одиницю (також починаючи з одиниці).
date_last_used	INTEGER	Останній раз, коли користувач намагався автоматично ввійти (як запам'ятовуються його облікові дані) з цим іменем користувача до свого облікового запису.

З вищенаведених даних потрібно брати origin_url, username_value та password_value поля з таблиці logins. З першим та другим полями немає ніяких проблем, бо ці поля не потрібно додатково оброблювати, а з останнім параметром є проблеми: password_value зберігається як blob блок.

Для отримання вже обробленого результату пароля, що буде рядком, потрібно виділити два різних метода розшифрування пароля, ці методи залежать від версії Google Chrome.

Якщо версія веб-браузера менша за 80, то використовується метод розшифрування за допомогою DPAPI інтсрументів, а в іншому випадку потрібно AES-256-GCM.

1.3.1 Розшифрування паролів у старих версіях Google Chrome

Google прагне використовувати системні методи та функції для всього, у тому числі й шифрування, саме тому для шифрування паролів вони використовували DPAPI функцію CryptProtectData, а для розшифровки CryptUnprotectData.

Обрано DPAPI тому що зашифрований пароль не може бути розшифрований іншим користувачем або іншим комп'ютером. Отже, відновлення пароля Chrome має виконуватися на тому ж комп'ютері, що й той самий користувач.

Шифрування персональних даних користувача в DPAPI відбувається у три етапи:

1. програма викликає функцію CryptProtectData, вказуючи вихідний матеріал для захисту та необов'язковий параметр ентропії. Якщо останній не заданий, будь-яка інша програма в контексті поточного користувача може виконати зворотне розшифрування, отримавши фізичний доступ до об'єкта DPAPI;
2. функція CryptProtectData відноситься до сімейства CryptoAPI і знаходиться в Crypt32.dll, звідки запит на подальшу обробку через захищений канал RPC перенаправляється в системний процес Local

Security Authority (LSA), і переходить зміна контексту поточного користувача на системний. Усі подальші маніпуляції проводять у контексті локальної системи;

3. у LSA відбувається сам процес шифрування даних і відправлення їх назад каналом RPC в Crypt32.dll, звідки вони благополучно повертаються у вихідний додаток.

CryptProtectData і CryptUnprotectData насправді є простими функціями-обгортками. Основна логіка роботи DPAPI відбувається в LSA і прихована від очей сторонніх спостерігачів.

Процес шифрування довгий. Спочатку з пароля користувача за допомогою криптографічного стандарту PBKDB2 виходить ключ шифрування Майстер Ключа. Цей ключ шифрування використовується для розшифрування Майстер Ключа. Якщо розшифровка пройшла невдало, пароль користувача розшифровує перший хеш історії паролів CREDHIST, з якого в свою чергу знову виходить ключ шифрування Майстер Ключа і робиться чергова спроба розшифровки. історії.

Щоб не запитувати пароль щоразу при виклику CryptProtectData/ CryptUnprotectData, Windows кешує його, зберігаючи в надрах LSA протягом усього профілю користувача.

Отже, Майстер Ключ користувача знаходиться у спеціальному файлі-контейнері, в якому можуть зберігатися додаткові резервні копії Майстер Ключа. Резервна копія Майстер Ключа використовується, якщо основна копія з тієї чи іншої причини не була розшифрована. Наприклад, після примусового скидання пароля. Майстер Ключ є критичним елементом у криптографії DPAPI, тому його довжина становить 512 біт, і за його шифруванні застосовуються перевірені алгоритми.

Для шифрування Майстер Ключа застосовується SHA1 хеш. Після того, як Майстер Ключ успішно декодовано, його вихідний 512-бітний матеріал бере участь у отриманні симетричного ключа шифрування об'єкта DPAPI. Для цього Майстер Ключ "перемішується" з випадковими даними (які

потім зберігатимуться в DPAPI об'єкті) і ентропією, якщо вона задана. Таким чином, досягається потрійна унікальність, якщо можна так сказати, ключа шифрування для кожного об'єкта DPAPI.

В кінці відбувається безпосередньо шифрування вихідних даних. У Windows 7 для шифрування використовується алгоритм AES-256. На всіх етапах цього процесу використовується «сіль», тобто набір випадкових даних, що уніфікує ключі шифрування і запобігає таким чином потенційної атаки перебором за допомогою райдужних таблиць. Після того, як дані зашифровані, вони групуються в єдину структуру разом з рештою службової інформації і відправляються додатку у вигляді об'єкта DPAPI.

1.3.2 Розшифрування паролів у останніх версіях Google Chrome

У 80-ій версії Google Chrome з метою посилення рівня безпеки компания змінила алгоритм шифрування даних із DPAPI на AES-256-GCM.

Для розшифрування паролю у нових версіях Google Chrome нам потрібно виконати наступні три кроки:

1. знайти та взяти ключ шифрування;
2. взяти зашифровані паролі;
3. розшифрувати збережені паролі.

Крок №1. Ключ шифрування зберігається у JSON файлі, який можна знайти за наступним шляхом: “C:\Users\<username>\AppData\Local\Google\Chrome\User Data\Local State”. Це текстовий файл, хоча у нього і не відображаються його тип.

Крок №2. Ми вже знаємо, що паролі знаходяться у “ C:\Users\<username>\AppData\Local\Google\Chrome\UserData\Default\LoginData” у таблиці з назвою “logins”.

У отриманому значенні зашифрованого пароля міститься два важливих значення, котрі потрібно також витягнути: вектор ініціалізації та зашифрований пароль.

Вектор ініціалізації зберігається між символами від 4 до 20, тоді як зашифрований пароль зберігається між символами від 21 до N-16 (тобто загальна кількість символів мінус 16).

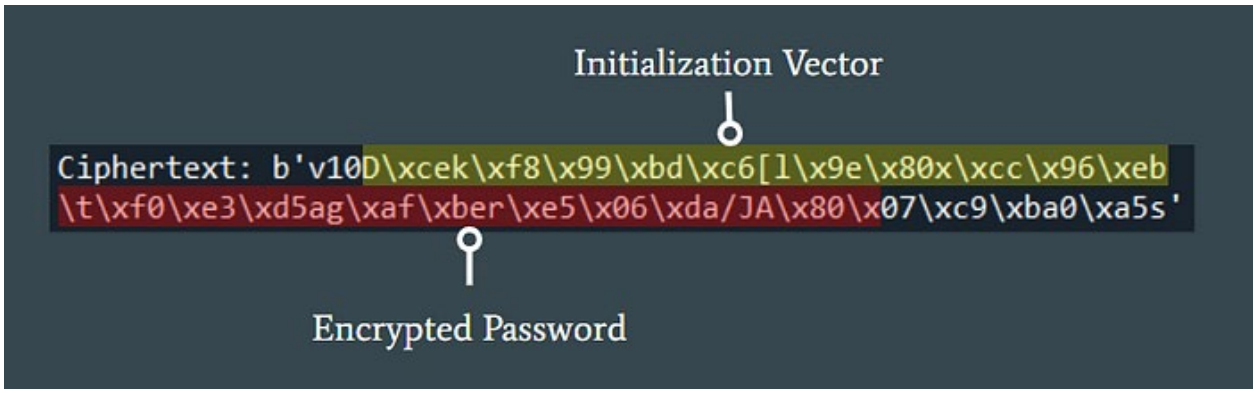

Рисунок 1.1 – Приклад розташування вектора ініціалізації
та зашифрованого пароля [2]

На рис.1.1 показано приклад зашифрованого пароля за допомогою AES-256-GCM алгоритму. Перші три байти – підпис (зазвичай «v10»), його слід пропустити. Наступні 12 байт – вектор ініціалізації. Інші байти це зашифрований пароль разом з тегом, тег займає в даному виді алгоритму16 байт, але не варто його видаляти, бо він має вагому роль для алгоритму.

Крок №3. AES-256-GCM є симетричним ключовим алгоритмом, який Chrome використовує для шифрування та дешифрування паролів.

AES використовує ключ шифрування та виконує складні математичні операції (наприклад, додавання, віднімання, змішування та зсув) для шифрування пароля. Цей параметр ми отримали в першому кроці.

Щоб підвищити його безпеку, додається вектор ініціалізації (випадковий рядок даних), що ускладняє процес дешифрування пароля. Цей параметр ми отримали в другому кроці.

Далі нам потрібно передати ключ шифрування, вектор ініціалізації та зашифрований пароль як аргументи та отримати результат у вигляді розшифрованого пароля.

1.4 Кеш

Усі файли, які зберігають дисковий кеш Google Chrome, знаходяться в одній папці (як ви вже здогадалися, вона називається кеш), і кожен файл у цій папці вважається частиною кешу і будь-який файл може бути видалений у певний момент часу.

Google Chrome використовує принаймні п’ять файлів: один файл індексу та чотири файли даних. Якщо будь-який із цих файлів відсутній або пошкоджений, весь набір файлів створюється заново.

Файл індексу містить основну хеш-таблицю, яка використовується для пошуку записів у кеші, а файли даних містять різноманітні цікаві дані, від бухгалтерської інформації до фактичних HTTP-заголовків і даних даного запиту. Ці файли даних також відомі як блочні файли, оскільки їхній формат оптимізований для зберігання інформації в «блоках» фіксованого розміру.

Коли розмір фрагмента даних перевищує 16 КБ, він більше не зберігатиметься в одному з стандартних блок-файлів. У цьому випадку він зберігатиметься в «окремому файлі», який не має спеціальних заголовків і містить лише ті дані, які ми хочемо зберегти. Ім’я окремого файлу має форму f_xxx, де xxx − це лише шістнадцяткове число, яке ідентифікує файл.

1.4.1 Індексний файл

Індексний файл – це хеш-таблиця. Файл відображається в пам’яті, щоб забезпечити швидкий перехід між хешем назви ресурсу (ключ) і адресою кешу, у якому зберігається ресурс. Молодші біти хешу використовуються для індексації таблиці, а вміст таблиці є адресою першого збереженого ресурсу з такими самими молодшими бітами хешу.

Однією з речей, яку необхідно перевірити при роботі з файлами кешу диска (індексом і кожним блочним файлом), є те, що магічне число в заголовку відповідає очікуваному значенню та що номер версії правильний. Версія має мажорну і мінорну частини, і очікувана поведінка полягає в тому, що будь-яка

зміна в мажорному номері означає, що формат тепер несумісний зі старішими форматами.

1.4.2 Блочні файли

Файли блоків мають назву data_n, де n — десятковий номер файлу.

Заголовок файлу має розмір у 8КБ та відображається в пам’яті, щоб забезпечити ефективне створення та видалення елементів файлу. Основна частина заголовка насправді є бітовим зображенням, яке ідентифікує використані блоки у файлі. Таким чином, максимальна кількість блоків, які можна зберегти в одному файлі, становить трохи менше 64 КБ.

Коли у файлі недостатньо вільних блоків для зберігання додаткових даних, файл збільшується на 1024 блоки, доки не буде досягнуто максимальної кількості блоків. У цей момент створюється новий блок-файл такого ж типу, і два файли зв’язуються разом.

Тип блочного файлу – це просто розмір блоків, які файл зберігає, тому всі файли, які зберігають блоки однакового розміру, пов’язані разом. Майте на увазі, що навіть якщо існує кілька блокових файлів, з’єднаних разом, адреса кешу вказує безпосередньо на файл, який зберігає певний запис. Ланцюжок використовується лише під час пошуку місця для виділення нового запису.

Щоб спростити розподіл дискового простору, можна зберігати лише записи, які використовують від одного до чотирьох фактичних блоків. Якщо загальний розмір запису більший, необхідно використовувати інший тип блок-файлу.

Ще одне спрощення алгоритму розподілу полягає в тому, що частина даних не перетинає межу вирівнювання чотирьох блоків

У заголовку є кілька полів, які допомагають у процесі виділення місця для нового запису. Порожнє поле зберігає лічильники доступного простору для кожного типу блоку, а підказки зберігають останнє скановане розташування для кожного типу блоку. У цьому контексті тип блоку — це кількість блоків, запитаних для розподілу.

Важливо розуміти, що після виділення запису його розмір не можна збільшити. Єдиний спосіб збільшити вже збережений запис — прочитати його, потім видалити з файлу та створити новий запис потрібного розміру.

1.4.3 Кеш адреса

Кожна частина даних, що зберігається в дисковому кеші, має задану «кеш-адресу». Адреса кешу — це просто 32-розрядне число, яке точно описує, де насправді знаходяться дані. Структура кеш адреси наведено на рис. 1.2.

Структура **CacheAddr**

0	0 0 0	1 1	1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1
1	Тип блока	Резерв	Розмір в блоках + 1	Номер data_N файла			Зміщення до початку інформації. Це значення помножене на розмур блоку та додане до 0x2000		

- Зустрічаються в файле **index** почаниючи зі зміщення 0x0170

Типи блоків			
0	0	0	separate file on disk
0	0	1	rankings block file
0	1	0	256 byte block file
0	1	1	1k byte block file
1	0	0	4k byte block file
1	0	1	external files block file
1	1	0	active entries block file
1	1	1	evicted entries block file

Рисунок 1.2 – Структура CacheAddr

Структура адреси кешу в основному повідомляє, чи зберігаються необхідні дані в блочному файлі чи як окремий файл, а також номер файлу (блоковий файл чи інший). Якщо дані є частиною блок-файлу, адреса кешу також містить номер першого блоку з даними, кількість використаних блоків і тип блок-файлу.

1.4.4 Приклад структури зберігання кешу

На рис. 1.3 наведено дисковий кеш із 7 файлами на диску: один індексний файл, п’ять блокових файлів і один окремий файл.

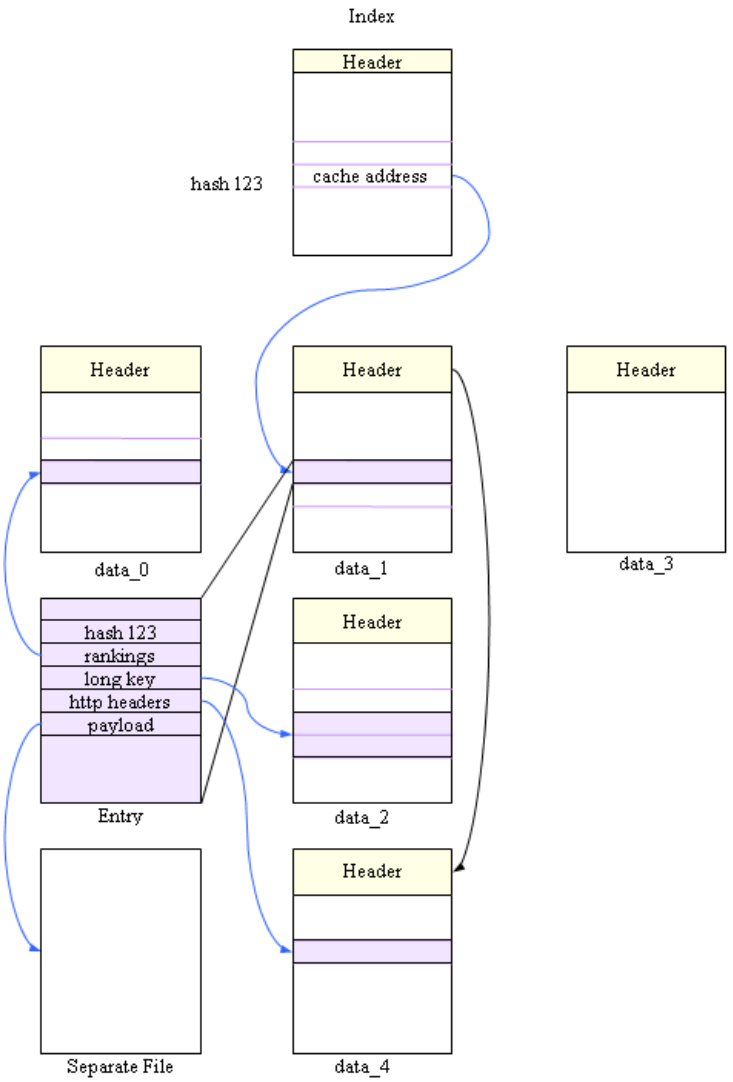

Рисунок 1.3 – Приклад структури зберігання кешу[6]

Файли data_1 і data_4 об’єднані разом, тому вони зберігають блоки однакового розміру (256 байт), тоді як data_2 зберігає блоки по 1 КБ, а data_3 зберігає блоки по 4 КБ.

Зображений запис має свій ключ і враховуючи те, що він використовує два блоки, він повинен мати від одного до двох кілобайтів. Цей запис також має два потоки даних: один для HTTP-заголовків (менше 256 байт), а інший для фактичного корисного навантаження (більше 16 КБ, тому він міститься у спеціальному файлі). Усі сині стрілки вказують на те, що адреса кешу використовується для пошуку іншого фрагмента даних.

РОЗДІЛ 2. ЗБІР ТА АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Збір вимог до програмного забезпечення

Після збору вимог за допомогою анкетування було проведено узагальнення отриманих даних. Розглянувши кожну з відповідей, було з'ясовано, що протиріччя відсутні, тобто кожна з них не мала подвійного змісту, цілі авторів співпадали та було відсутнє входження однієї і тієї ж системної функції з різними описами до вимог.

Результати анкетування. Перші два питання дають нам розуміння про те, хто саме користується схожими програмними продуктами. Як видно з рис. 2.1 здебільшого це чоловіки з 18 до 25 років. Це дає нам зрозуміти, що схожими програмними додатками користуються люди, що вже мають досвід роботи з небагато бюджетними програмами. Як результат, ми можемо створити мінімалістичний UI, що буде зрозумілим та функціональним.

Рисунок 2.1 – Статистика перших двох питань з анкети

Наступні два питання стосуються того для якого веб-браузера та яку саме програму використовували анкетовані люди для витягування даних. Як видно з рис. 2.2 найпопулярнішим веб-браузером серед зазначених є Google Chrome. Тобто програма буде покривати велику кількість запитів серед користувачів що хочуть отримати збережені дані. Також, серед аналогів найпопулярнішім є BrowsinHistoryView від NirSoft та Browser History Viewer від foxton.

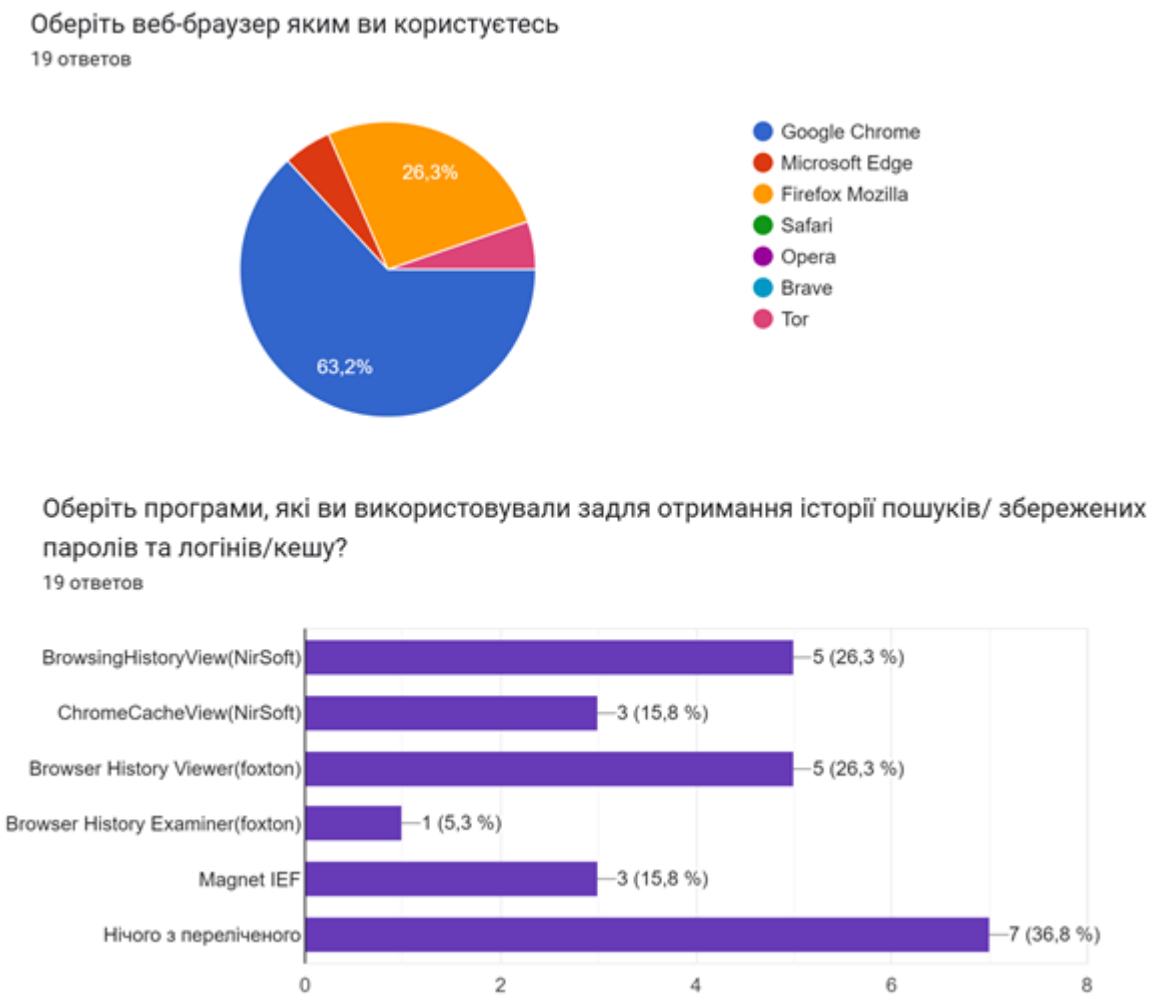

Рисунок 2.2 – Статистика третього та четвертого питання з анкети

З рис. 2.3 можна зробити висновок, що найприорітетнішими даними для відображення − історія пошуку, паролі та кеш. Також, важливо додати можливість пошуку серед результатів, що відображається на UI.

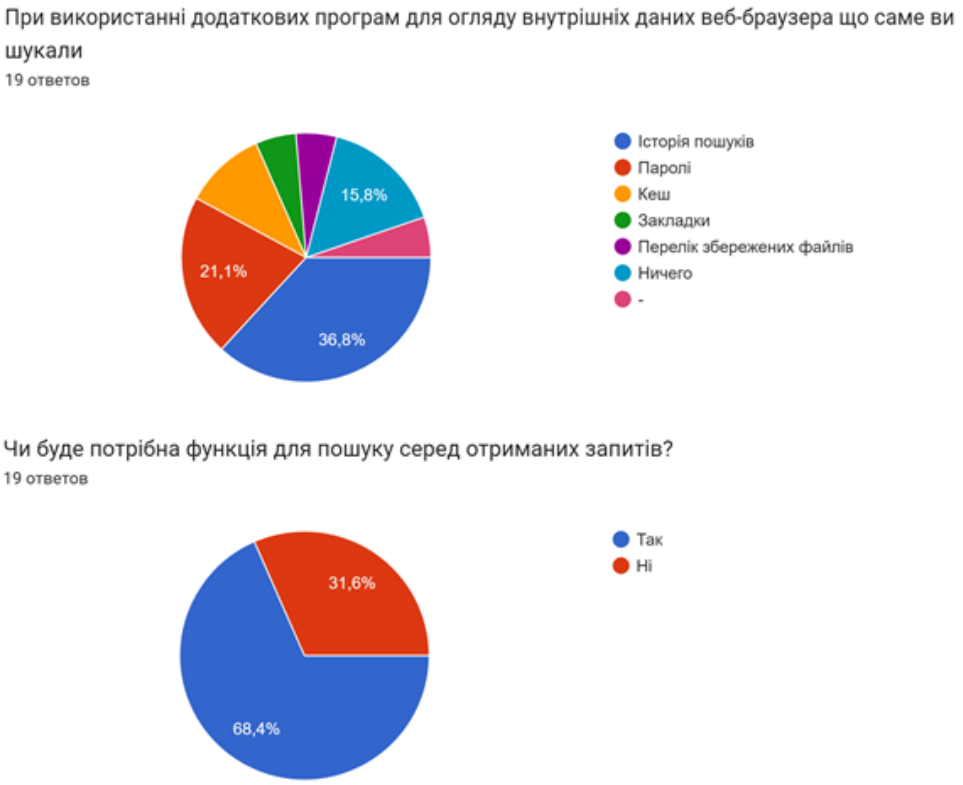

Рисунок 2.3 – Статистика п'ятого та шостого питань з анкети

З Рис 2.4 виходить що також потрібно додати сортування за алфавітом, це сильно спростить пошук серед отриманих результатів, бо додасть більше розуміння в структуру результату. Також, потрібно додати функцію для експорту історії пошуку, логінів та паролів і кешу. Це може бути дуже корисно, якщо потрібно створити якийсь звіт.

За результатом анкетування були здобуті наступні вимоги:

– створити програму схожу до BrowsingHistoryView від NirSoft або/та Browsing History Viewer від foxton;
– додати можливість пошуку серед отриманих результатів;
– додати можливість сортування за алфавитом серед отриманих результатів;
– додати можливість експорту історії пошуків, збережених логінів та паролів та кешу.

Рисунок 2.4 – Статистика останніх питань з анкети

2.2 Прототипування

Виходячи з отриманих вимог було розроблено прототип UI частини програми, цей прототип представлен на рис. 2.5.

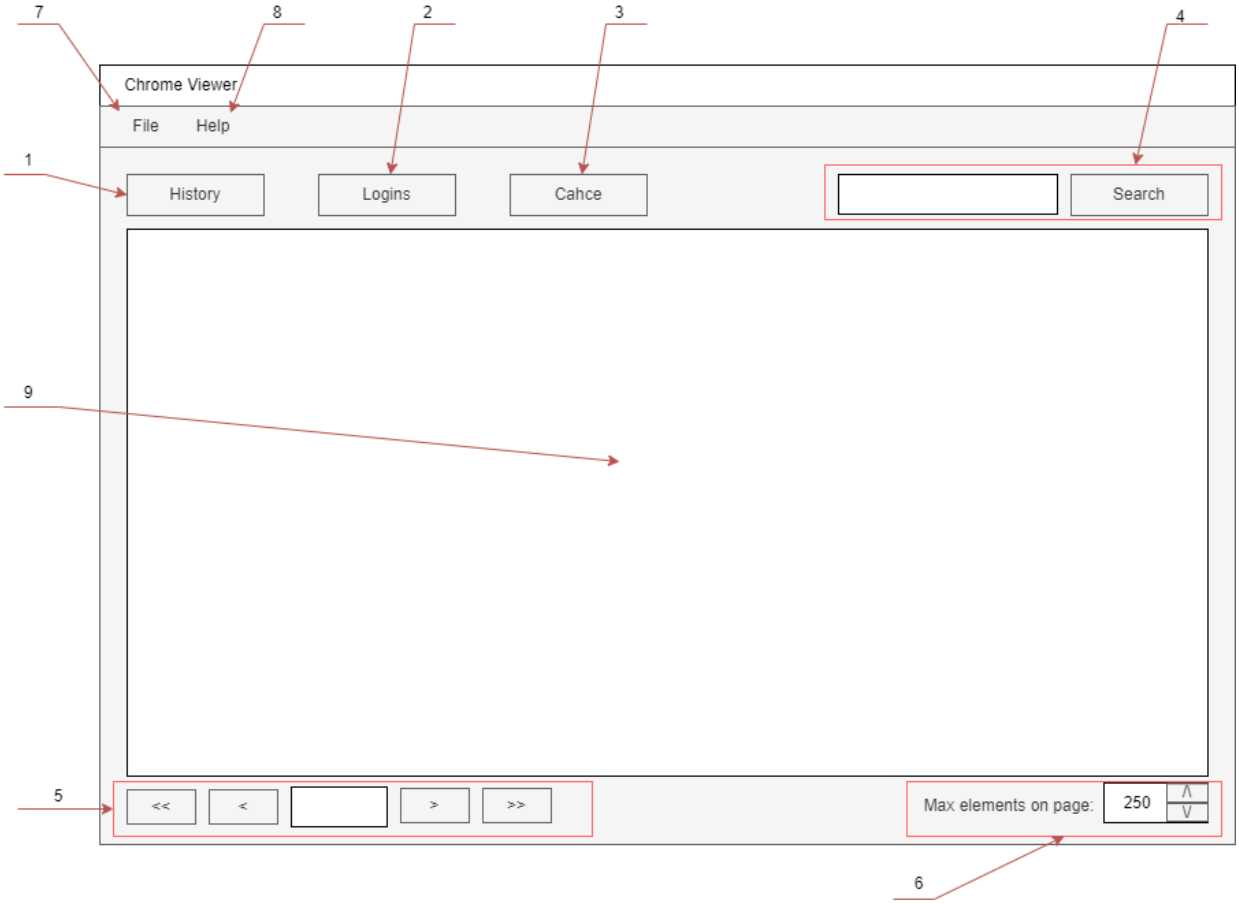

Рисунок 2.5 – Прототип UI

Елементи 1, 2 та 3 відповідають за зміну стану відображення інформації у полі 9. Як зрозуміло з назви, елементи(стани) 1, 2 та 3 відповідають за відображення історії пошуків, збережених логінів і паролей та кешу відповідно. Кожний з них має свій вплив на стан елементу 9: при 1 та 2 станах буде відображуватися 3 стовпчики за допомогою DataGridView елементу управління або його аналогу, при 3 стані повинно відображатись 5 стовпчиків.

Елементи UI, що об'єднанні в поле під номером 4 відповідають за пошук введеного тексту в TextBox елемент управління серед отриманих результатів. Після натискання Button елементу управління буде оновлено елемент 9 та відображено все, що відповідає введеному тексту серед доступної інформації.

Елементи UI, що об'єднані в поле під номером 5 відповідають за можливість пролистування відображеної інформації. Існує два типа Button

елементів управління: перший – переміщює відображену інформацію на першу/ останню сторінку, другий – переміщює на одну сторінку вперед\назад.

Елементи UI, що об'єднані в поле під номером 6 відповідають за зміну максимального значення кількості одночасно відображених рядків на одній сторінці.

Елементи 7 та 8 є частиною MenuStrip чий ого аналогу. Елемент 8 повинен відображувати додаткову інформацію про програму та інструкцію по використанню програму. Елемент 7 містить в собі інші піделементи, що відповідають за експортування відображених даних.

Як вже було сказано, елемент 9 відповідає за відображення інформації у вигляді таблиці. За допомогою натискання на головні стовпчики можна активувати сортування усіх результатів за алфавитом.

2.3 Аналіз, опис та дослідження бізнес-процесів

Нехай є завдання розробки бізнес-процесу відображення даних про історію пошукі/логінів та паролей/кешу. Результатом цього процесу має бути забезпечення користувача необхідною для нього інформацією.

Тоді бізнес-процес виконується таким чином:

1. Менеджер отримує інформацію про дані для відображення.
2. У програмі створюється SQL запит та обробляється.
3. Результат виконання минулого пункту відображається.
4. Якщо додатково обирається якийсь режим сортування чи пошук за ключовим словом, то користувач робить відповідні дії і пункти 2 та 3 повторюються. Якщо нічого не обирається(тобто режим відображення стандартний), то повертаємо потрібну інформацію.

Вважатимемо бізнес-процес завершеним, оскільки покупець зараз або після надходжень інформації отримає те, що він хотів.

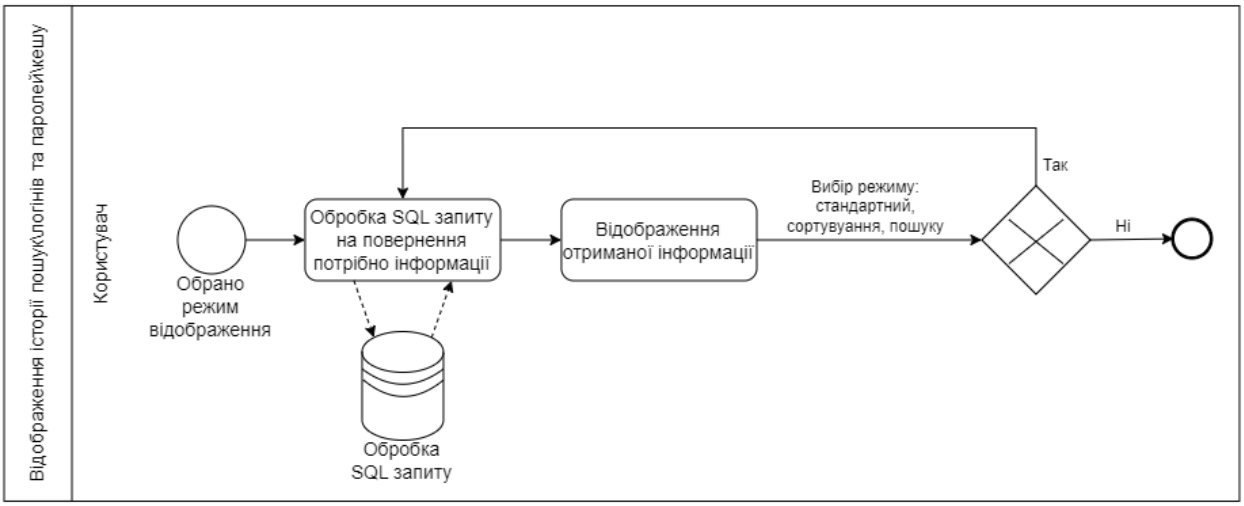

Рисунок 2.6 – Бізнес-процес для потреб програми

2.4 Опис аналогів

BrowsingHistoryView (див. рис. 2.7) від NirSoft – це утиліта, яка читає дані історії таких веб-браузерів: Mozilla Firefox, Google Chrome, Internet Explorer, Microsoft Edge, Opera. Утиліта дозволяє переглядати історію перегляду всіх профілів користувачів у запущеній системі, а також отримувати історію перегляду із зовнішнього жорсткого диска. Таблиця історії веб-перегляду містить таку інформацію: URL-адресу, назву сторінки, час відвідування, кількість відвідувань, веб-браузер і профіль користувача. Також, можна експортувати історію веб-перегляду як csv, html, та xml файл з інтерфейсу користувача або з командного рядка. Серед функціоналу є можливість сортувати дані, шукати інформацію за ключовими словами, обирати можливість відображення для певних браузерів та фільтрація запитів за певний період часу.

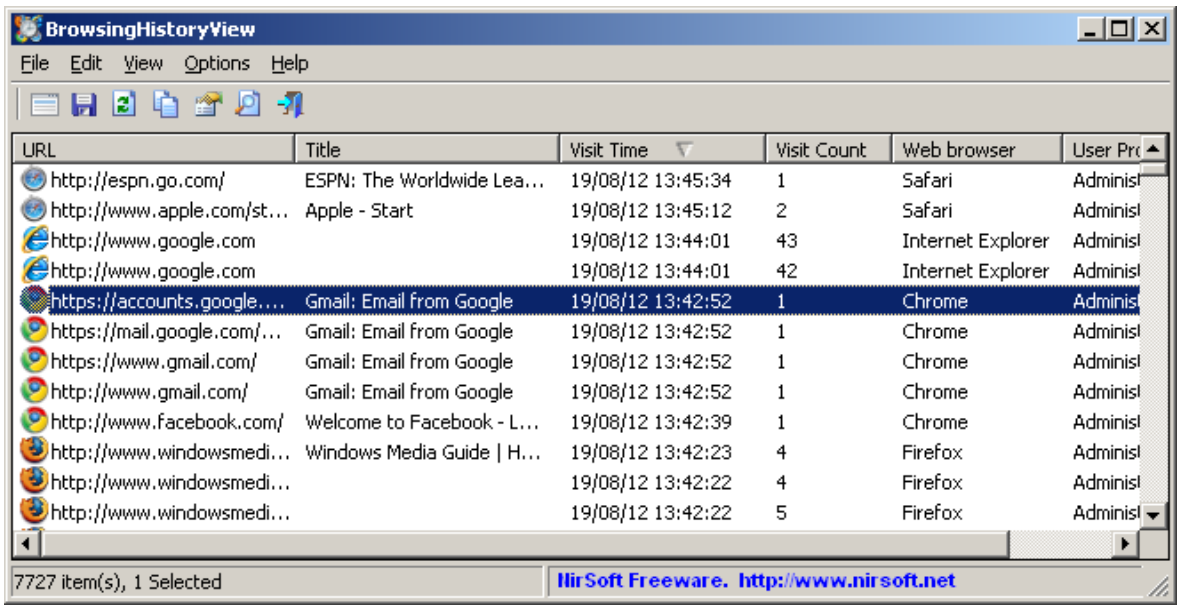

Рисунок 2.7 – Вигляд графіного інтерфейсу BrowsingHistoryView

ChromeCacheView (див. рис. 2.8) від NirSoft – це невелика утиліта, яка читає папку кешу веб-браузера Google Chrome і відображає список усіх файлів, які наразі зберігаються в кеші. Для кожного файлу кешу відображається така інформація: URL-адреса, тип вмісту, розмір файлу, час останнього доступу, термін дії, ім’я сервера, відповідь сервера тощо. Коли ви вибираєте елемент кешу зображення (gif, png, jpg) або текстовий елемент кешу (html, css, json, javascript), ви можете переглядати вміст файлу кешу. Можна легко вибрати один або кілька елементів зі списку кешу, а потім витягнути файли в іншу папку або скопіювати список URL-адрес у буфер обміну.

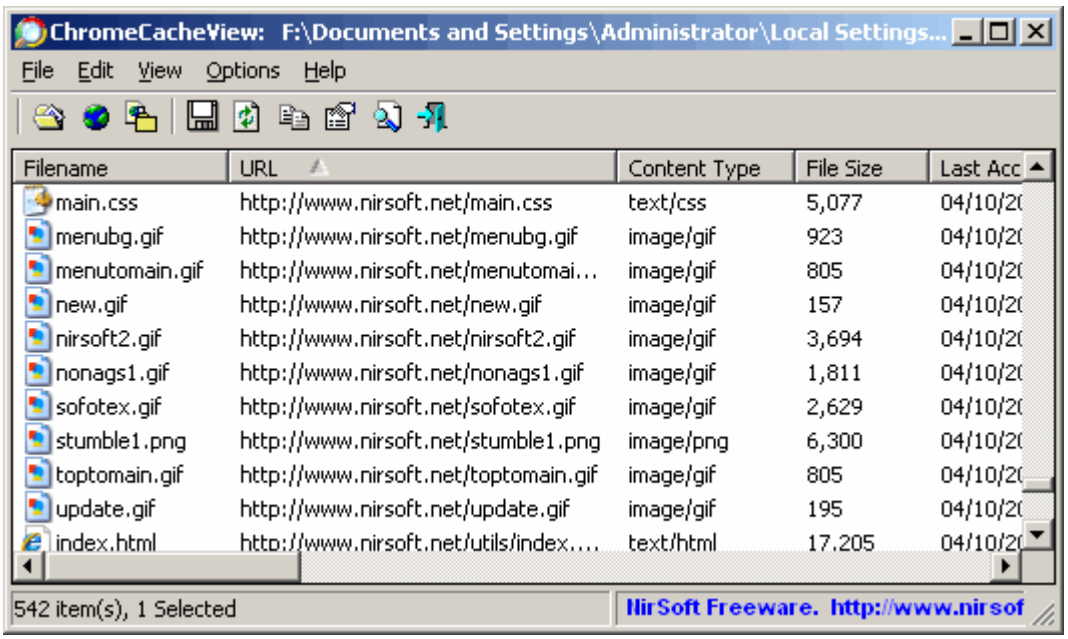

Рисунок 2.8 – Вигляд графіного інтерфейсу ChromeCacheView

Browser History Viewer (див. рис. 2.9) від foxton – це інструмент криміналістичного програмного забезпечення для отримання та перегляду історії Інтернету з таких веб-браузерів: Google Chrome, Microsoft Edge, Mozilla Firefox та Internet Explorer. Серед доступних функцій є перегляд історії пошуків, кешованих даних, пошук за ключовими словами, фільтрація результатів за окремий проміжок часу, фільтрація результатів за браузерами та можливість перегляду закешованих даних прямо через цю програму.

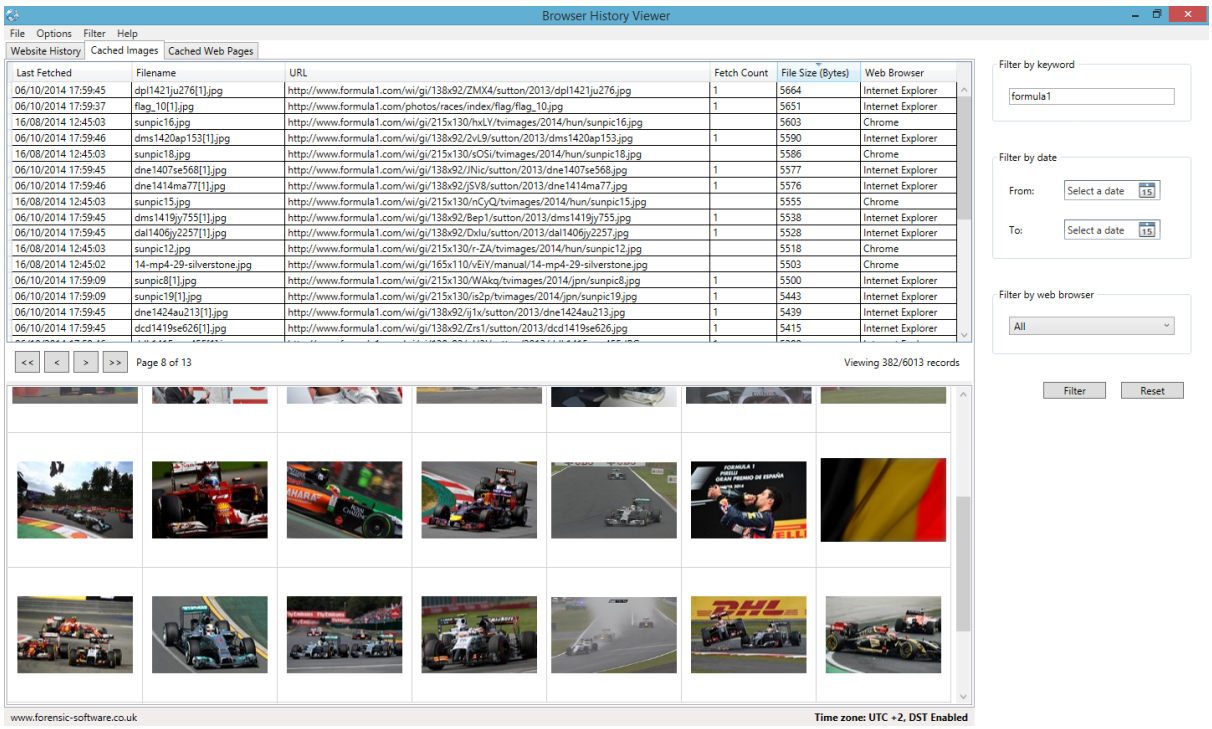

Рисунок 2.9 – Вигляд графіного інтерфейсу Browser History Viewer

Browser History Examiner (див. рис. 2.10) від foxton – це інструмент криміналістичного програмного забезпечення для збору, аналізу та звітування про історію Інтернету з основних веб-браузерів. Взагалі, це покращений та поглиблений Browser History Viewer, але платний. На відміну від свого безкоштовного аналога додано такий функціонал: перегляд всіх можливих збережених даних веб-браузерів(закладки, перелік завантажених файлів, логіни, фавікони, електроні адреси і т.д.), можливість створення звітів з отриманої інформації, відновлення видаленої історії пошуків і т.д.

Magnet AXIOM Examine – це інструмент криміналістичного програмного забезпечення для збору, аналізу та звітування всієї операційної системи. Ця програма точно такий же функціонал пов’язаний з веб-баузерами, що і попередній розглянутий кандидат, але вся різниця в тому, як саме виглядає UI частині (див. рис 2.11) програм.

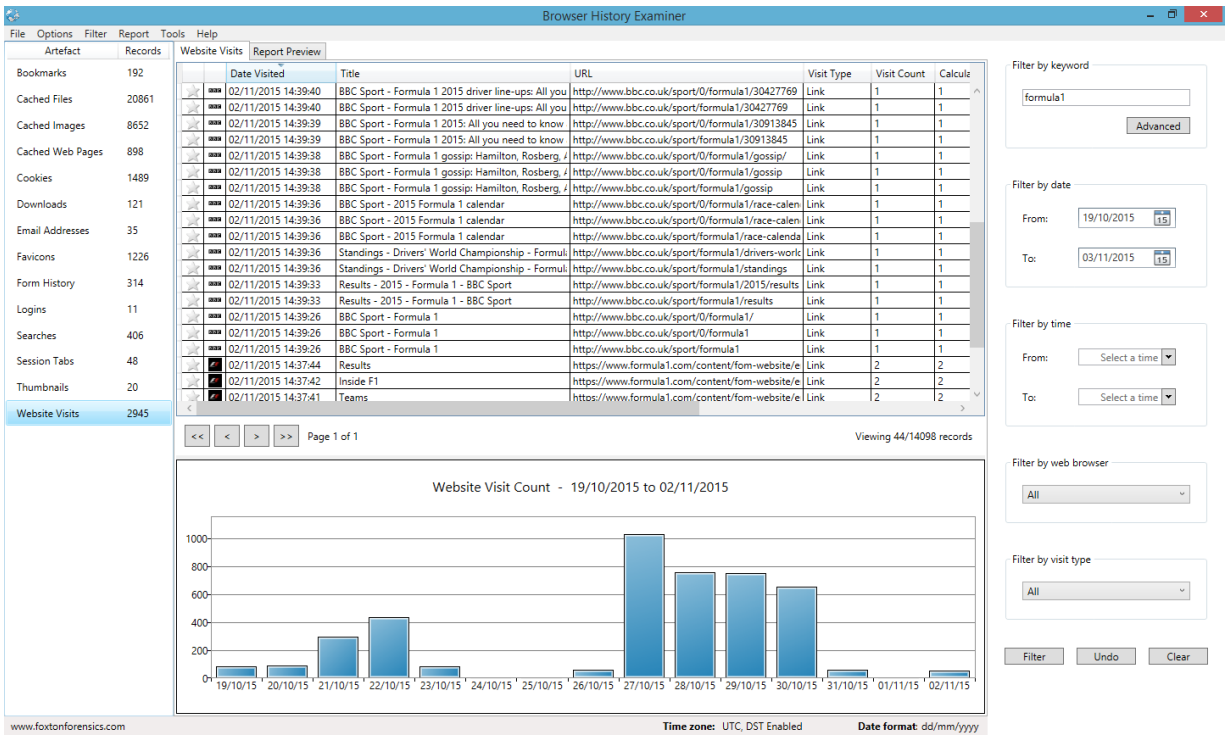

Рисунок 2.10 – Вигляд графіного інтерфейсу Browser History Examiner

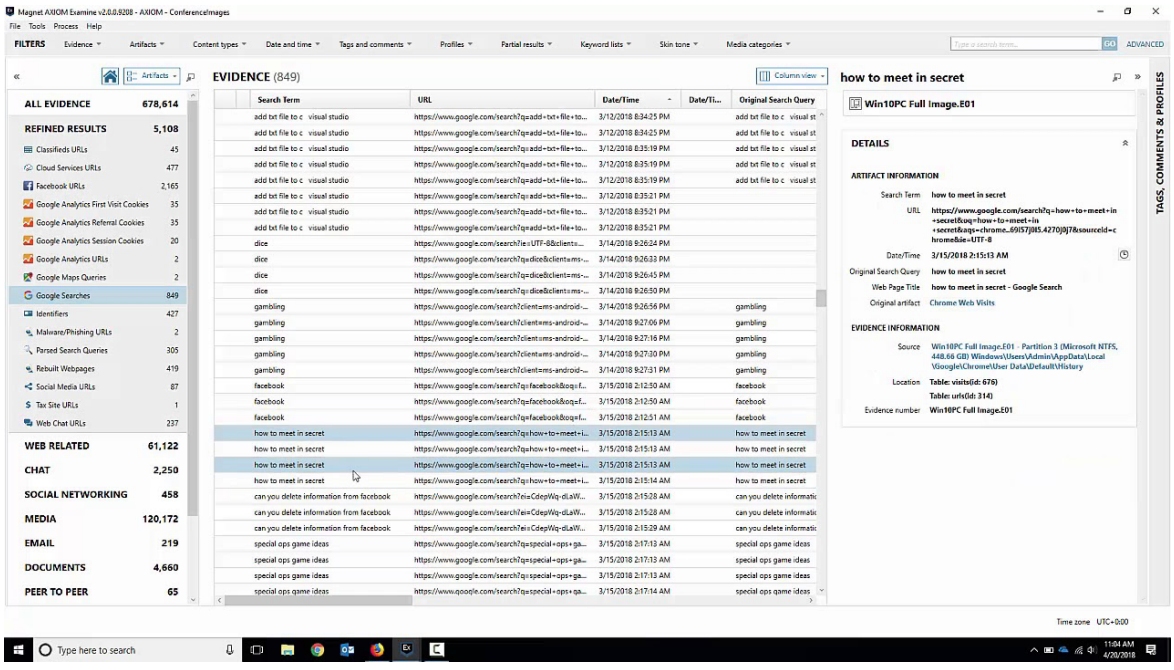

Рисунок 2.11 – Вигляд графіного інтерфейсу Magnet AXIOM Examine

РОЗДІЛ 3. ЗОВНІШНЄ І ВНУТРІШНЄ ПРОЕКТУВАННЯ

3.1 Зовнішнє проєктування

Функціональне призначення: обробка та відображення даних історії пошуків, збережених паролів та логінів і кешу Google Chrome веб-браузеру.

Експлуатаційне призначення: зменшення часових витрат на пошук та отримання інформації(історія пошуків, збережені паролі та логіни, кеш) з збережених даних Google Chrome веб-браузеру, який може бути або не доступним, або видаленим з комп’ютера.

Функціональні вимоги:

1. читання та відображення даних історії пошуків, збережених паролів та логінів і кешу Google Chrome веб-браузеру;
2. можливість експорту історії пошуків та збережених логінів і паролів у csv;
3. можливість експорту кешу;
4. можливість пошуку за ключовим словом серед отриманих результатів;
5. можливість сортування отриманих результатів за алфавитом.

Вхідними дані:

- дві бази даних(Login Data та History) Google Chrome;
- кеш-файли;
- ключове слово для пошуку;
- стовпчик для сортування за алфавитом;
- максимальна кількість рядків на одній сторінці результатів.

Вихідними даними виступають відображена інформація, експортовані файли csv формату та кеш-файли.

Опис зовнішнього інформаційного середовища. Бази даних Login Data та History повинні бути створені та заповнені Google Chrome веб-браузером. Стовпчик для сортування обирається за допомогою натискання лівої кнопки миші на будь-який існуючий стовпчик. Пошук за ключовим словом

починається після вводу довільної інформації в поле та натискання на кнопку. Результати експорту даних подається в текстовому форматі у вигляді текстового файлу, графічному форматі – на формі як таблиця з даними.

Для функціонування програми необхідно наявність на комп’ютері структури файлів Google Chrome або сам встановлений веб-браузер, а також операційна система повинна бути 64-бітною.

Виконаємо специфікацію функціональних вимог у вигляді прецедентів (рис. 3.1).

Рисунок 3.1 – Діграмам прецедентів

3.2 Внутрішнє проєктування

3.2.1 Проєктування архітектури системи

У розділі розглянуто проєктування для системи читання даних Google Chrome веб-браузера, функціонал якої описано в попередньому підрозділі та побудова діаграми класів.

Етапи проєктування:

Етап 1. Для моделювання словника системи проводиться аналіз зовнішніх специфікацій системи. Ідентифікування сутностей: читач історії пошуків, читач збережених паролів та логінів, читач кешу, модуль взаємодії з базами даних, дешифратор і модуль взаємодії з файлами. Додатково зазначається інтерфейс користувача у вигляді екранної форми.

Ідентифіковані обов’язки:

– читач історії пошуків – багаторазове виконання отримання кількості різних записів та отримання самих записів;

– читач збережених паролів та логінів – багаторазове виконання отримання кількості різних записів та отримання самих записів;

– читач кешу – багаторазове виконання отримання кількості різних записів та отримання самих записів;

– модуль взаємодії з базами даних – багаторазове виконання взаємодії з базою даних на основі sql-запросів;

– модуль взаємодії з файлами – багаторазове виконання взаємодії з файловою системою;

– модуль взаємодії з кешом – багаторазове виконання взаємодії з кеш даними(отримання кеш рядків, експортування кешу);

– дешифратор – багаторазове виконання дешифруючих дії з обраним рядком;

– форма – введення параметрів відображення, запуск та перегляд результатів, подання команди на збереження результатів.

Атрибути та операції, необхідні для виконання обов’язків кожної сутності-класу наведемо у табл. 3.1.

Таблиця 3.1 – Сутності, атрибути та методи

Сутність	Атрибути	Методи
Читач		отримати рядки з таблиці бази даних отримати кількість рядків

Продовження таблиці 3.1

Читач історії пошуків	dbname – ім'я бази даних; tablename – ім'я таблиці з бази даних; listOfColumns – перелік імен стовпців таблиці.	отримати рядки з таблиці бази даних отримати кількість рядків
Читач збережених паролів та логінів	dbname – ім'я бази даних; tablename – ім'я таблиці з бази даних; listOfColumns – перелік імен стовпців таблиці.	отримати рядки з таблиці бази даних отримати кількість рядків
Читач кешу	cahceDirPath – шлях до діректорії з кешом	отримати рядки з таблиці бази даних отримати кількість рядків експортувати
Модуль взаємодії з базами даних		отримати рядки з таблиці бази даних виконати sql запит отримати кількість рядків в таблиці
Модуль взаємодії з файлами		копіювати файл видалити файл експортувати отримати шлях до appdata\local
Модуль взаємодії з кешем		отримати рядки з кеш даними отримати кількість рядків експортувати

Продовження таблиці 3.1

Дешифратор		отримати мастер ключ дешифрувати з допомогою DPAPI дешифрувати за допомогою AES-256-GCM
Форма	поля для введення вхідних даних поле виведення рядків даних кнопки для експорту даних	запустити експеримент побудувати графіки зберегти в файл

Етап 2. Моделювання розподілу обов’язків у системі

Множини класів, які працюють спільно для досягнення деякої поведінки:

– читач історії пошуків, модуль взаємодії з базами даних – отримання рядків з бази даних історії пошуків;

– читач збережених паролів та логінів, модуль взаємодії з базами даних – отримання рядків з бази даних збережених паролів та логінів;

– читач збережених паролів та логінів, дешифратор – отримання розшифрованих паролів;

– читач кешу, модуль взаємодії з кешом – отримання кеш-даних, експортування кеш-файлів;

– форма, модуль взаємодії з файлами – копіювання баз даних історії пошуків і паролів та логінів;

– форма, читач – виведення даних історії пошуку/збережених паролів та логіні/кешу (дані можуть бути стандартними, сортованими чи результатом пошуку за ключовим словом), також є можливість експорту даних;

Етап 3. Моделювання непрограмних сутностей для даної системи виконувати не потрібно, оскільки особливості апаратної частини не використовується в роботі системи, всі інші сутності вже представлено у вигляді класів.

Етапи 4-7. Моделювання примітивних типів, простих залежностей, наслідування та структурних зв’язків.

Визначення примітивних типів виконаємо на етапі побудови діаграми класів. Результат моделювання різних видів зв’язків подано у табл. 3.2.

Таблиця 3.2 – Моделювання залежностей

Клас, який зв'язується	Клас, з яким зв’язуються	Тип зв’язку
Читач	Читач історії пошуків	Реалізація
	Читач збережених паролів та логінів	Реалізація
	Читач кешу	Реалізація
Форма	Читач	Агрегація
	Модуль взаємодії з файлами	Асоціація
Читач історії пошуків	Модуль взаємодії з базами даних	Асоціація
Читач збережених паролів та логінів	Модуль взаємодії з базами даних	Асоціація
	Дешифратор	Асоціація
Читач кешу	Модуль взаємодії з кешом	Асоціація

Представимо заключний результат моделювання у вигляді діаграми класів (рис. 3.2). На діаграмі наразі відсутня більша частина конструкторів, сетерів та гетерів.

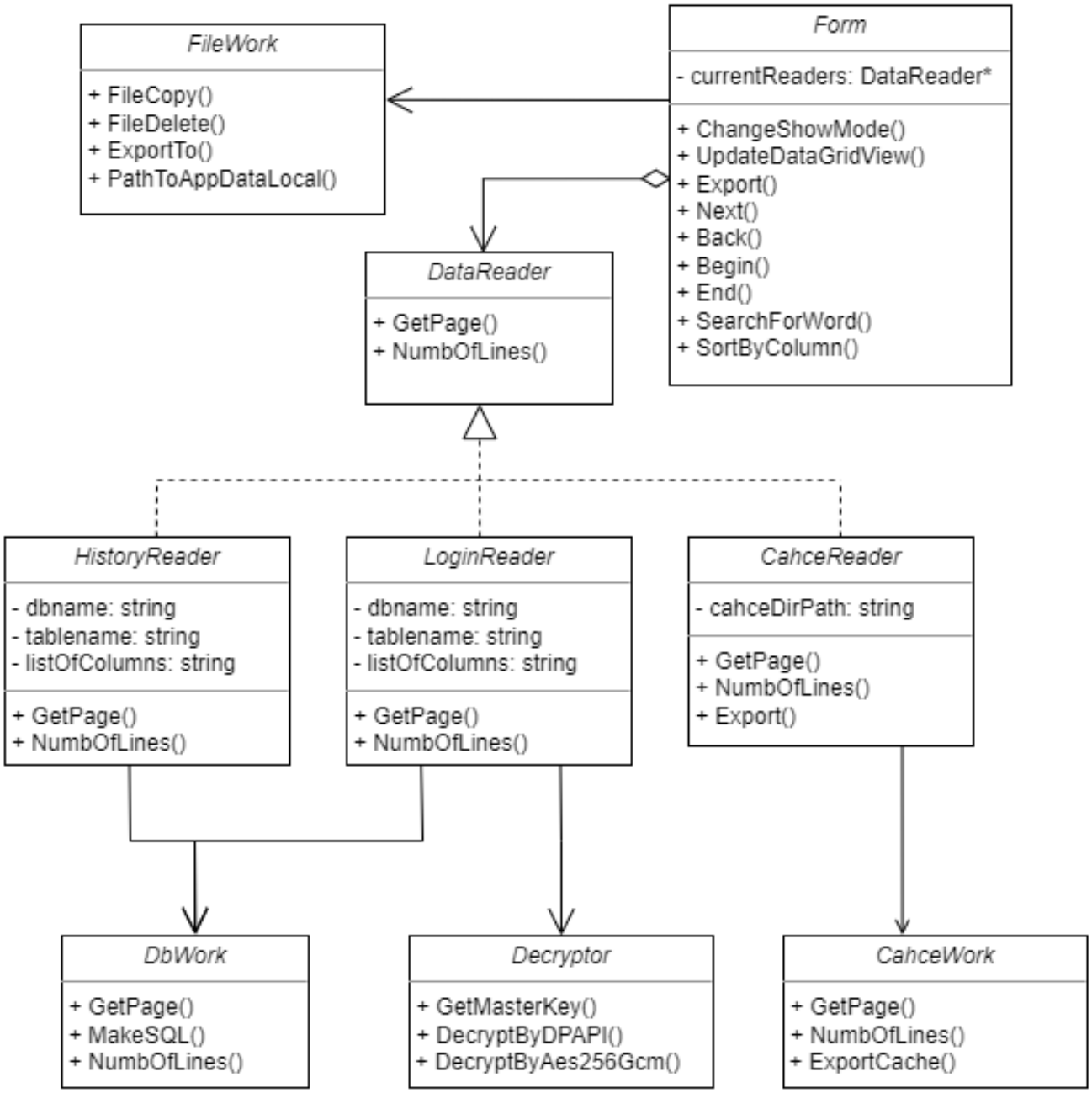

Рисунок 3.2 – Діаграма класів

У табл. 3.3 наведено CRC-картки для всіх наявих класів.

Таблиця 3.3 – Загальна CRC-картки для всіх класів

DataReader	
Базовий клас	Похідні класи (нащадки)
-	HistoryReader, LoginReader, CahceReader
Обов’язки	Зв’язки
Поліморфізм	Form
HistoryReader	
Базовий клас	Похідні класи (нащадки)
DataReader	-
Обов’язки	Зв’язки
Отримання рядків історії пошуків	DbWork
LoginReader	
Базовий клас	Похідні класи (нащадки)
DataReader	-
Обов’язки	Зв’язки
Отримання рядків збережених паролів та логінів	DbWork та Decryptor
CahceReader	
Базовий клас	Похідні класи (нащадки)
DataReader	-
Обов’язки	Зв’язки
Отримання рядків кешу	-
DbWork	
Базовий клас	Похідні класи (нащадки)
-	-
Обов’язки	Зв’язки
Відповідає за взаємодію з базами даних за допомогою sql-запитів	HistoryReader та LoginReader

Продовження таблиці 3.3

Decryptor	
Базовий клас	Похідні класи (нащадки)
-	-
Обов’язки	Зв’язки
Дешифрування паролів за допомогою DPAPI та AES-256-GCM	LoginReader
Form	
Базовий клас	Похідні класи (нащадки)
-	-
Обов’язки	Зв’язки
Відображення даних у виді таблиці, введення вхідних даних, можливість почати експорт даних	DataReader та FileWork
FileWork	
Базовий клас	Похідні класи (нащадки)
-	-
Обов’язки	Зв’язки
Видалення та копіювання файлу, експорт даних	Form
CacheWork	
Базовий клас	Базовий клас
-	-
Обов’язки	Обов’язки
Обробка всієї інформації пов'язаної з кешем	CacheReader

3.2.2 Проєктування інтерфейсу користувача

Наведемо сценарій діалогу для системи описаної в п. 3.1 (рис. 3.3). В якості засобу формалізації використаємо діаграму станів. На діаграмі стани,

позначені кругом та кругом в колі, є початковим та заключним станами відповідно. Вони позначають початок та завершення роботи з програмою. Вихід з програми можливий з будь-якого стану, проте, зважаючи на функціональне призначення програми, перехід у заключний стан на діаграмі представлено в одному екземплярі

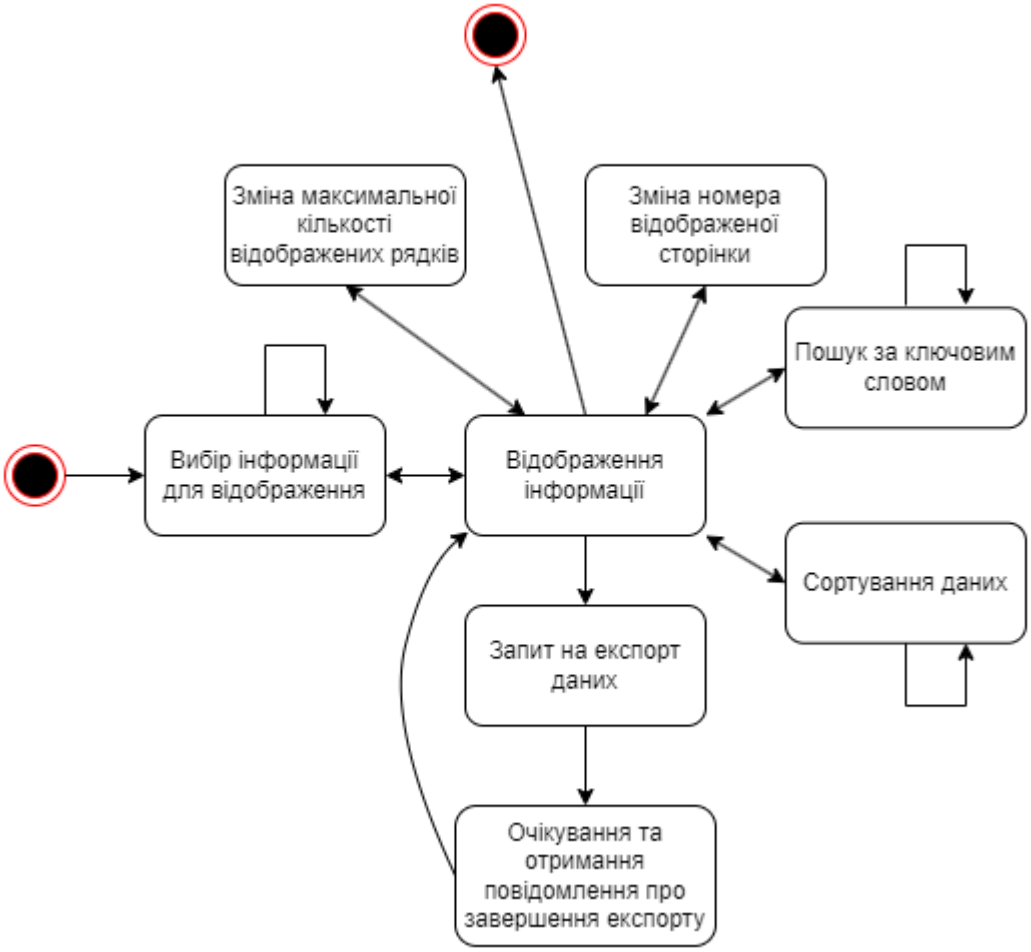

Рисунок 3.3 – Діаграма станів користувача програми

Для обраної системи не буде зручно використовувати структура діалогу на основі екранної форми, оскільки користувач повинен мати змогу для переключення між інформацію для відображення, зміни номера сторінки для відображення, зміни максимальної кількості рядків, пошуку та сортування. Всі перелічені можливості повинні бути доступні в будь-який час. Саме цьому найкращий варіант, що виконує вищенаведені вимоги був Browser History Viewer, але програми від NirSoft мають дуже мінімалістичний та зрозумілий інтерфейс. Тому виходячи з вимог та аналогів наведених у п. 3.2

було розроблено інтерфейс за допомогою Visual Studio та заповнено його власноруч тестовими даними, щоб мати уяву про майбутній вигляд програми. На рис. 3.4 зображено приклад виведення даних історії пошука.

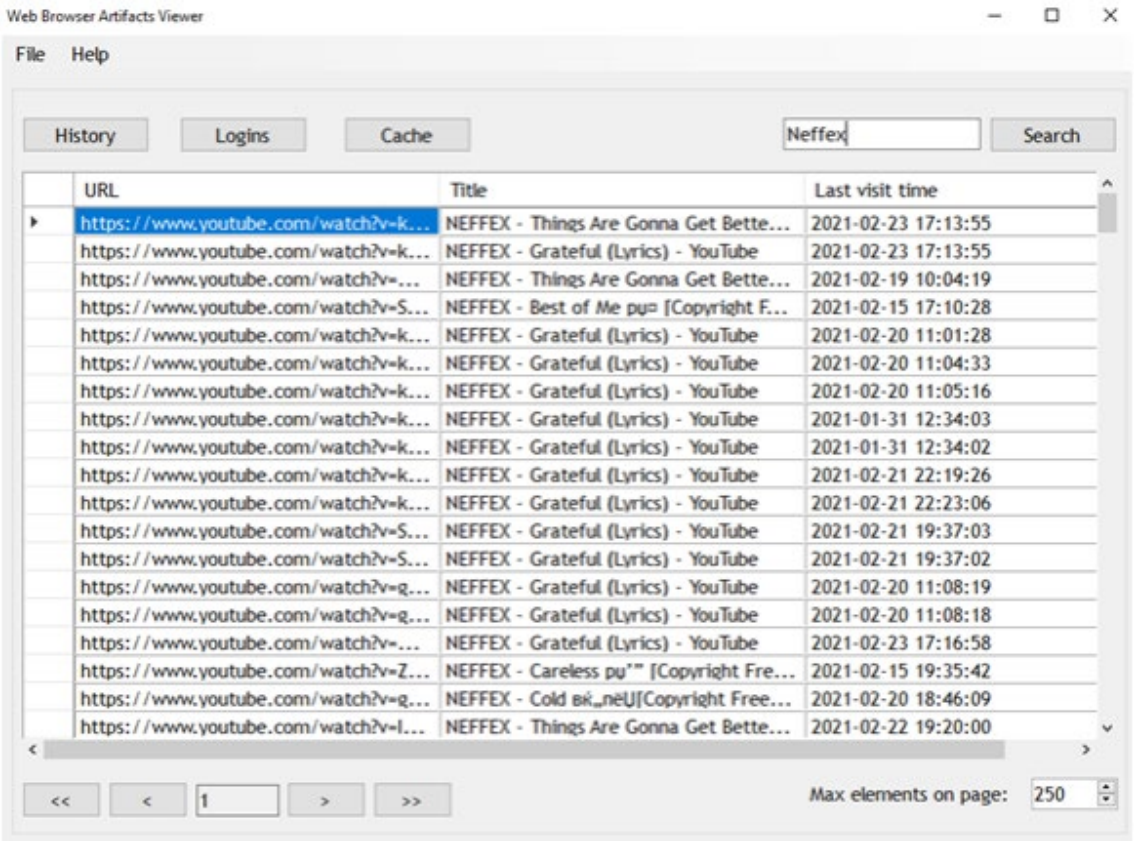

Рисунок 3.4 – Приклад виводу даних історії пошуку

Нижче наведено текстовий варіант поведінки програми на дії користувача.

При натисканні на кнопки History, Logins чи Cache буде оновлюватись DataGridView та виводитись відповідна інформація. Якщо ж потрібних нам даних не існує(наприклад, для вкладок History та Logins це бази даних, що зберігають в собі відповідну інформацію), то програма повинна надати нам про це інформацію.

Максимальна кількість рядків одночасно відображених не може бути менше за 50 та більше за 1000. Це обмеження самого елементу управління – NumericUpDown.

Для навігації по сторінкам можна використовувати лише кнопки. Елемент управління, що відповідає за відображення номера сторінки,

знаходиться в ReadOnly стані, тобто ми не можемо нічого ввести в цей TextBox. Найменший номер сторінки дорівнює 1, а найбільший являє собою межу значення int.

Для пошуку необхідно ввести будь-що у TextBox елемент та натиснути на кнопку Search, щоб розпочати пошук. Якщо TextBox пустий, то ми просто нічого не шукаємо і відбувається простий вивід інформації. Якщо таблиця пуста, то пошук просто не відбудеться.

Для сортування потрібно натиснути на будь-який стопчик з назвою. Всього реалізовано два типи сортування: від ‘A’ до ‘Z’ та від ‘Z’ до ‘A’. Для зміни типу сортування потрібно повторно натиснути на обраний стовпчик. Якщо таблиця пуста, то сортування просто не відбудеться.

Для меню було використано MenuStrip елемент управління. На рис. 3.5 наведено структуру меню.

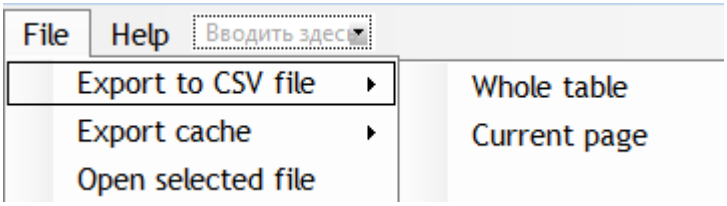

Рисунок 3.5 – Структура меню

Меню для експорту історії пошуків\паролів та логіні\кешу у нас є два вибору: екпортувати лише відображені дані чи експортувати всю таблицю. Для кешу експортується не таблиця, а самі файли. Також, саме для кешу використовується останній пункт меню, за допомогою цього можна одразу відкрити кешований файл через програму.

3.3.3 Проєктування динаміки системи

Наведемо приклад діаграми (рис. 3.6) послідовності для прецеденту “Відобразити логіни та паролі”, який реалізує багаторазове виконання дешифрування для визначення реального значення пароля. На представленій діаграмі додатково використано позначка циклу, що передбачає багаторазове виконання методів, які в нього включено.

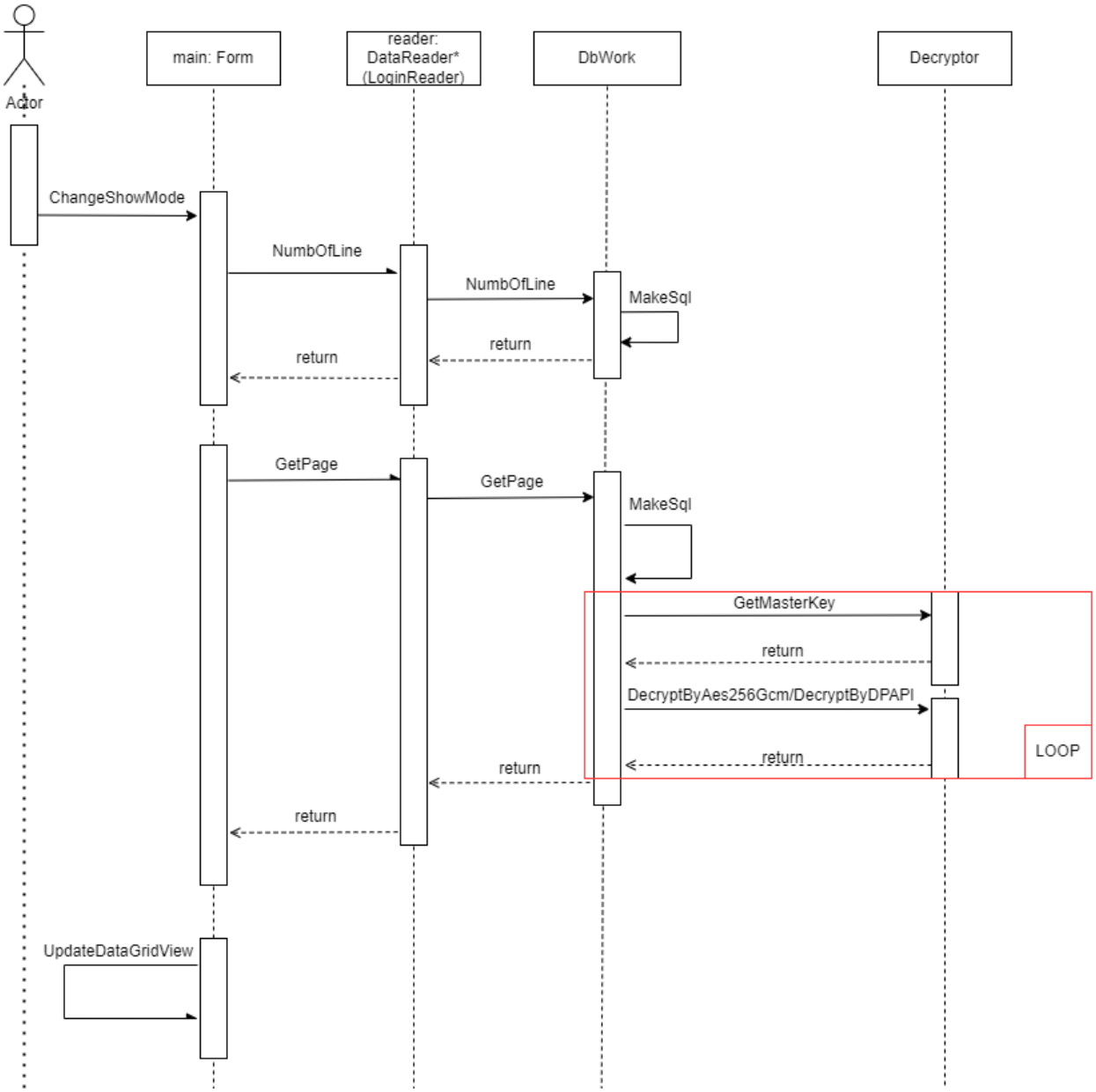

Рисунок 3.6 – Діаграма послідовностей

Розглянемо приклад побудови діаграми діяльності. На діаграмі представимо дії для реалізації прецеденту побудови графіка(рис 3.7). Ініціатором прецеденту − сценарію дій − є користувач як зовнішній об’єкт системи. Оскільки логіни можуть бути дешифровані двома шляхами, то діаграма містить позначку ромба, що вказує на варіативність дій.

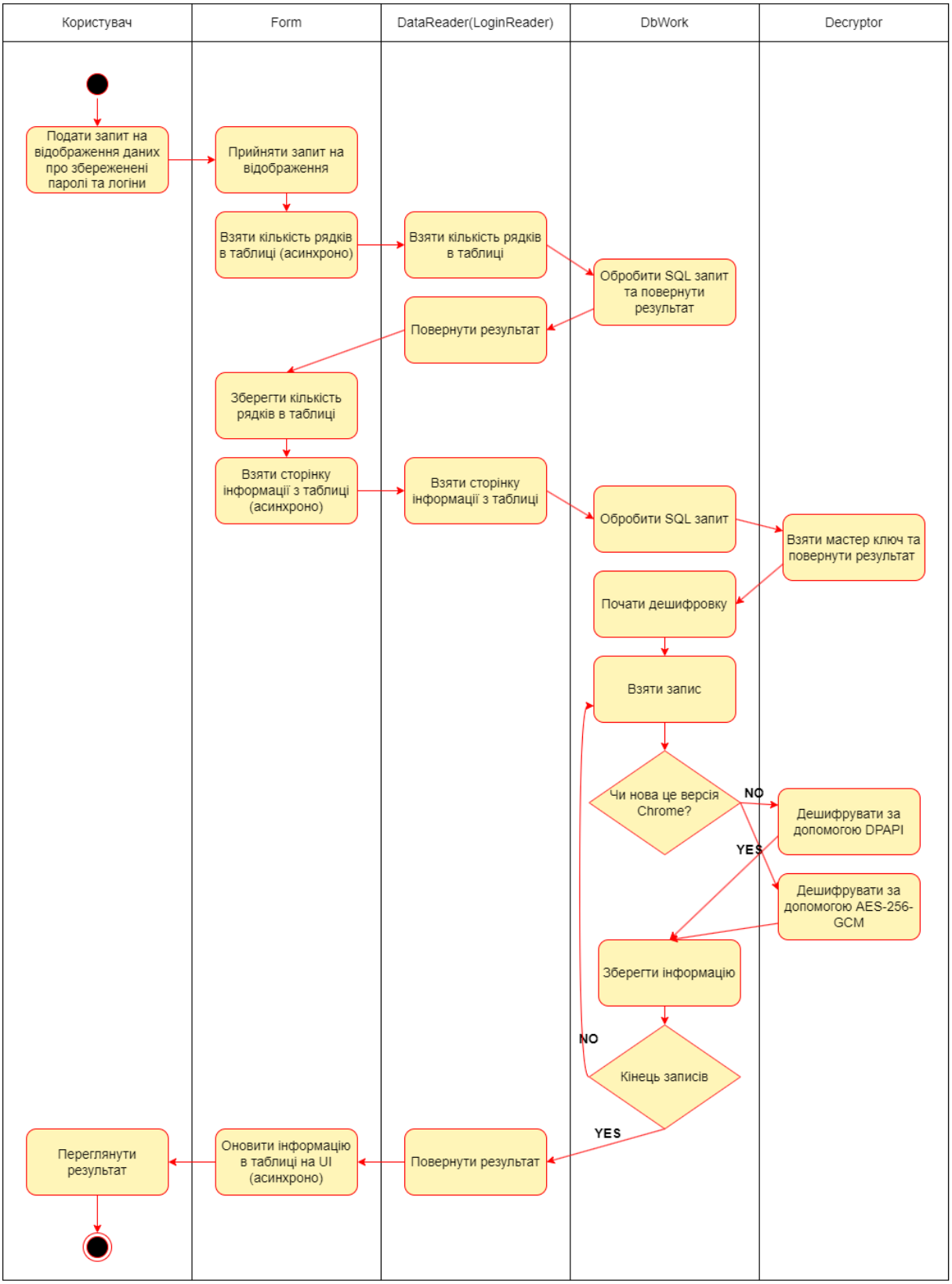

Рисунок 3.7 – Діаграма діяльності

Діаграма станів застосовується для моделювання внутрішніх та зовнішніх об’єктів системи. Так до зовнішніх можна віднести користувача.

Діграма станів на рис. 3.8 ілюструє стани користувача системи для отримання даних збережених логінів та паролів. Як приклад моделювання внутрішнього об’єкта було розглянуто стани екземпляру класу DbWork.

Інформація з результату з SQL запиту розділяється та використовується повторно - цим пояснюється повернення до стана взяття паролю з запиту. Оскільки процес обробки всіх паролів в результаті SQL запиті − процес циклічний, то відповідний стан обміну елементів має на діаграмі дугу-петлю.

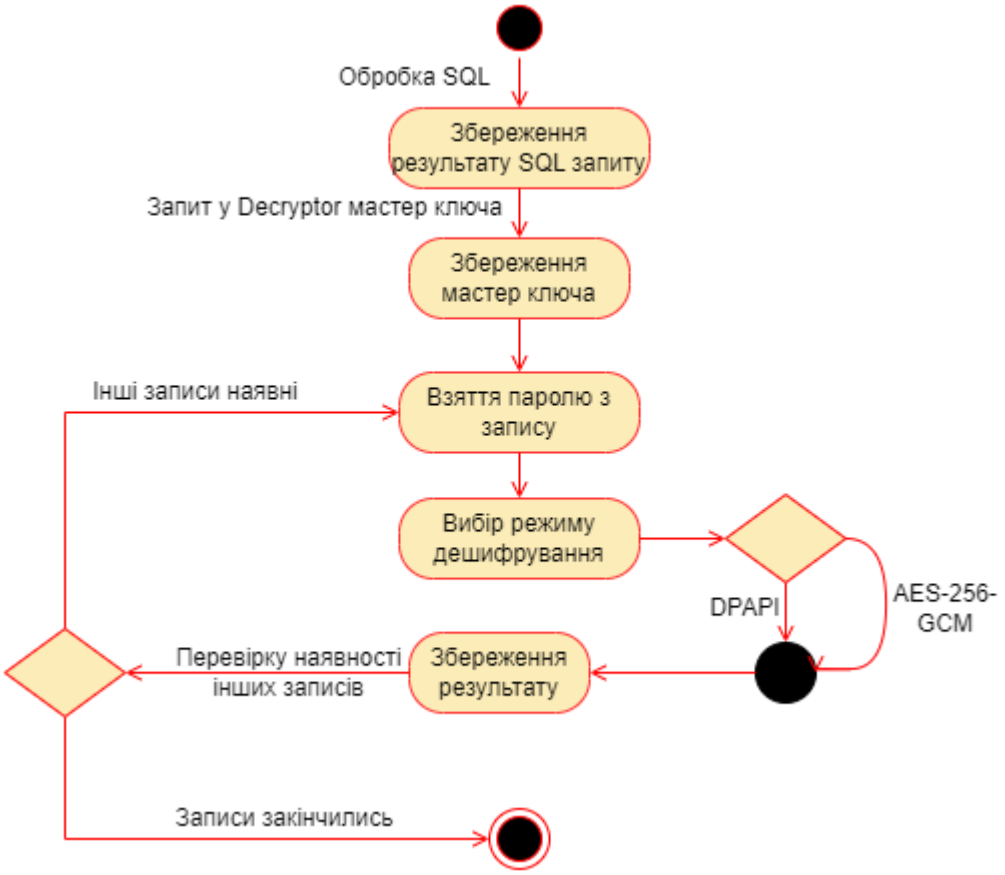

Рисунок 3.8 – Діаграма станів

3.3.4 Проєктування системи на фізичному рівні

Розглянемо приклад діаграми артефактів для нащой системи(рис. 3.9). Передбачаються, що система буде реалізована мовами C++ та C#. Всі назви компонентів збігаються з відповідними назвами класів, наприклад, компонент Form реалізовує створення форми. Компонент libmagic.dll є артефактом процесу виконання - бібліотека, яка реалізує логіку взаємодії з кешом.

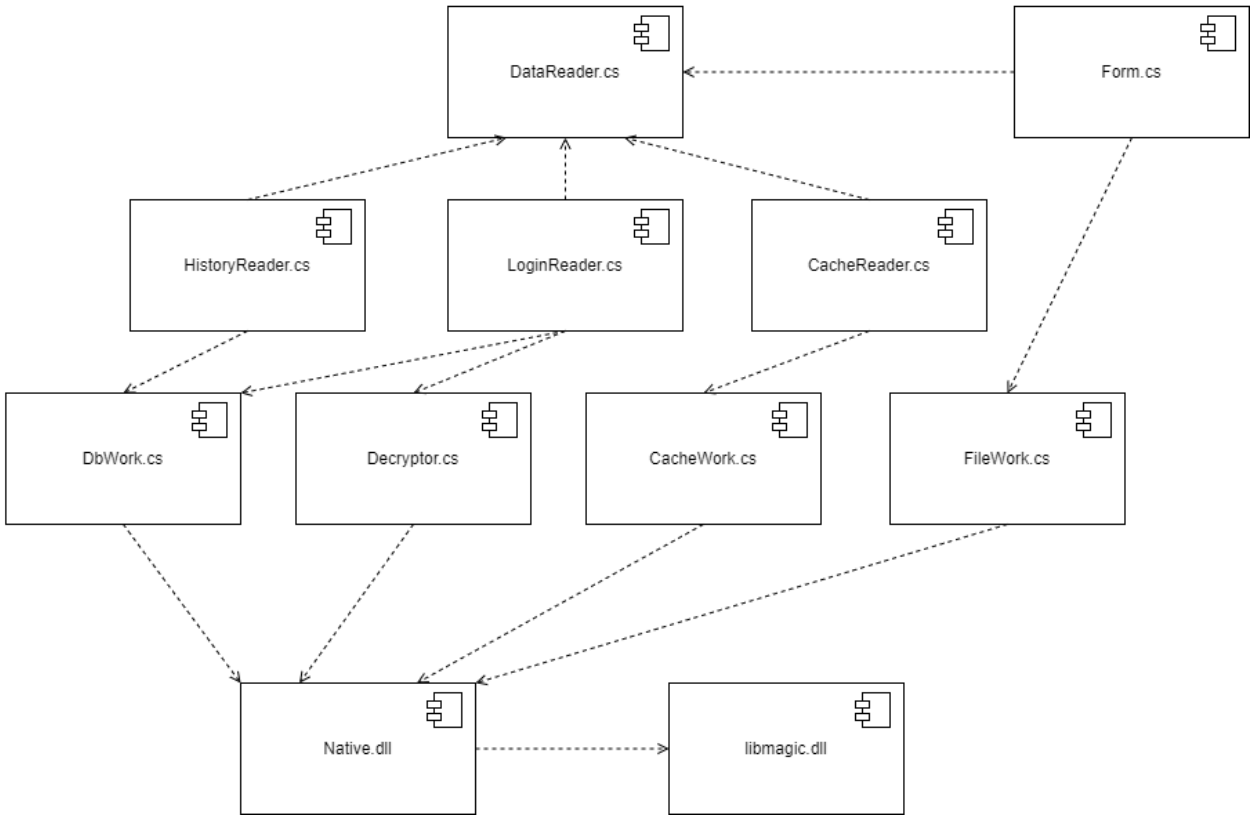

Рисунок 3.9 – Діаграма артефактів (компонентів)

Артефактом розміщення для даної системи буде obj–файл, який можна створити засобами IDE на основі cs-файлів. cs-компоненти, наведені на діаграмі, є артефактами робочих продуктів. Щодо стереотипів, то до Executable віднесемо obj–файл; Library присутні в розмірі двох штук; File - всі cs-файли; Document відсутні.

Для системи визначення обчсилювальної складності алгоритму, яка проєктувалася в попердніх пунктах, діграма розгортання не буде інформативною, оскільки система не є розподіленою. Зовнішній вигляд діаграми в даному випадку зводиться до одного вузла-процессора, що позначається кубом.

РОЗДІЛ 4. РОЗРОБКА ПРОГРАМИ

4.1 Розбиття програми на підзадачі за допомогою методу покрокової деталізації

Перед застосуванням методу покрокової деталізації розглядається задача. Існує декілька різних типів інформації для відображення, але всі ці типи мають єдину особливість – інформація відображається в таблиці. Таблиця є моделлю відображеної інформації, де кожний стовпчик – тип відображеної інформації, а рядок – це послідовність типів. Також, існує декілька форматів відображення: стандартний, просортована таблиця за обраним стовпцем та рядки з наявність ключового слова. Необхідно знайти та відобразати інформацію у заданому форматі.

Представимо декомпозицію задачі за методом покрокової деталізації. Перепишемо подану задачу у вигляді специфікації вхідних та вихідних даних.

Вхід: тип інформації для відображення, кількість рядків на сторінці, номер сторінки відображення, режим сортування, ключове слово для пошуку, бази даних з потрібною інформацією та кеш-файли.

Вихід: таблиця з результуючою інформацією та експортовані файли.

Розіб’ємо задачу на під задачі

1. вибір типу інформації для відображення;
   1.1. обробка інформації історії пошуків;
      1.1.1. звернення до бази даних з історією пошуку;
   1.2. обробка збережених паролів та логінів;
      1.2.1. звернення до бази даних з збереженими паролями та логінами;
      1.2.2. дешифрування паролів;
   1.3. обробка кешу;
2. зміна формату відображення результатів;
   2.1. вибір стовпчику для сортування та режиму;
   2.2. зміна кількості максимального значення одночасно відображених рядків у таблиці;

2.3. введення ключового слова для пошуку;

2.4. зміна сторінки для відображення ключової інформації;

3. оновлення таблиці з результуючою інформацією;

4. експортування даних.

Кожна з під задач є значно простішою, ніж задача в цілому.

Представимо цю задачу деревом (рис. 4.1) та таблицею каскадів (табл. 4.1). У каскадному представлені зробимо на один рівень деталізації більше, щоб наблизитися до безпосередньої реалізації.

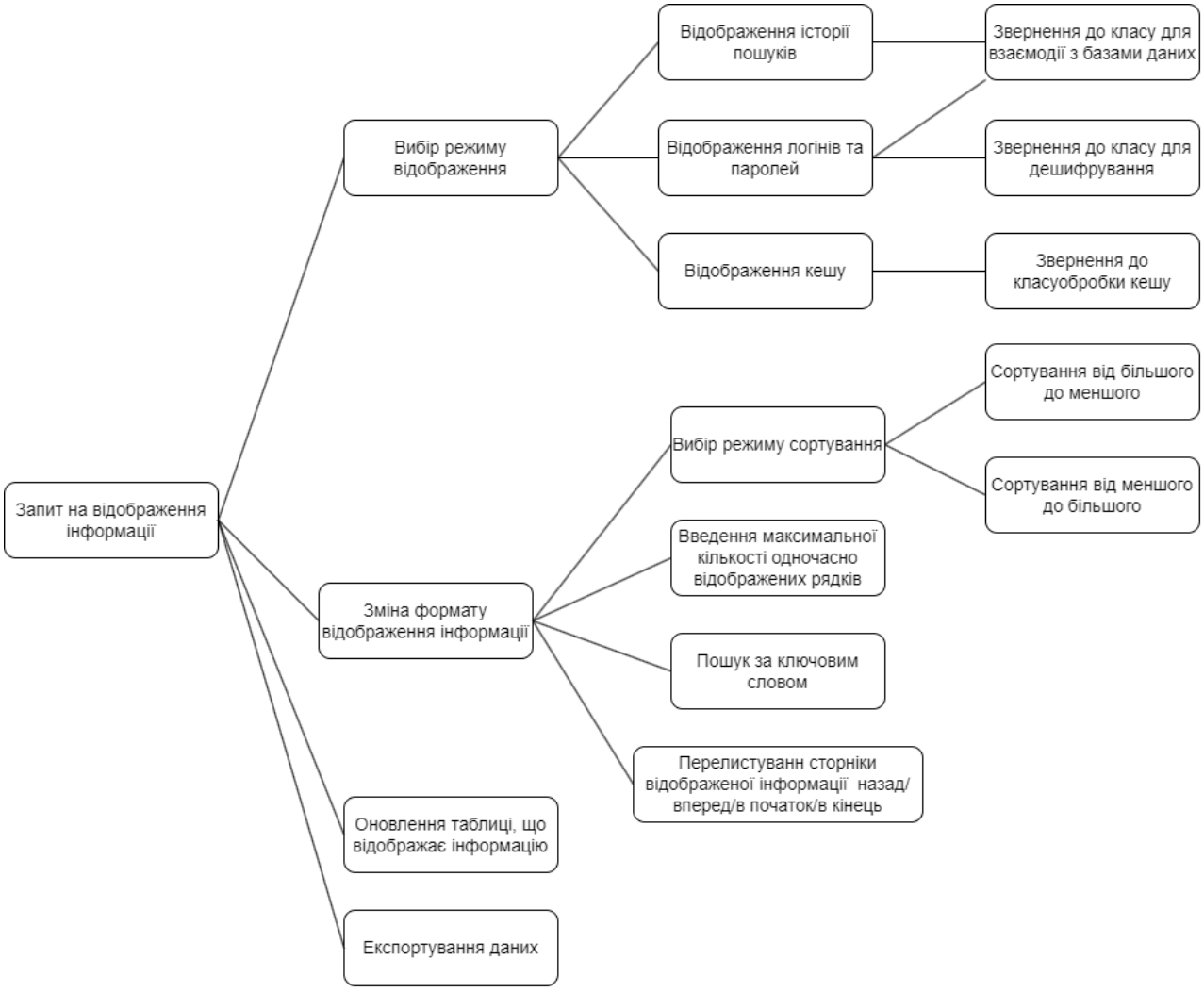

Рисунок 4.1 – Дерево покрокової деталізації

Таблиця 4.1 – Каскадне представлення декомпозиції

Задача	Підзадачі			
Пошук та відображення інформації у заданому форматі	Вибір режиму відображення	Відображення історії пошуків	Звернення до класу для взаємодії з базами даних	Створити клас-обгортку над SQLite API та створити функції для експорту, що всередені використ. клас-обгортку
		Відображення логінів та паролей		
			Звернення до класу для дешифрування	Створити функції для експорту дешифрув. за допомогою DPAPI або AES-256-GCM
		Відображення кешу	Звернення до класу для взаємодії з кешом	Створити функції для експорту взаємодії з кешом

Продовження таблиці 4.1

	Зміна формату відображення інформації	Вибір режиму сортування	Від більшого до меншого
			Від меншого до більшого
		Введення максимальної кількості одночасно відображених рядків	
		Пошук за ключовим словом	
		Перелистування сторінок відображеної інформації	
	Оновлення таблиці, що відображає інформацію		
	Експортування даних		

Для вирішення вищеописаної задачі напишемо використовуючи C++ та C# мови програмування з графічним інтерфейсом користувача. Весь функціонал, що стосується взаємодії з базами даних, дешифруванням даних та кешем винесемо до окремого DLL модуля, що буде експортувати потрібний нам функціонал. На мові програмування C# буде створений графічний модуль, що буде імпортувати в себе DLL модуль.

4.2 Графічне представлення алгоритмів DLL модуля

У рис. 4.2 наведено алгоритм по побудові SQL запиту на основі вхідних даних у формі діаграми Насі-Шнейдермана. Вхідні дані алгоритму: ім'я таблиці, перелік назв стовпців через спеціальний символ, вихідний рядок, розмір вихідного рядку, режим сортування, номер стовпця за яким сортуємо, ключове слово для пошуку.

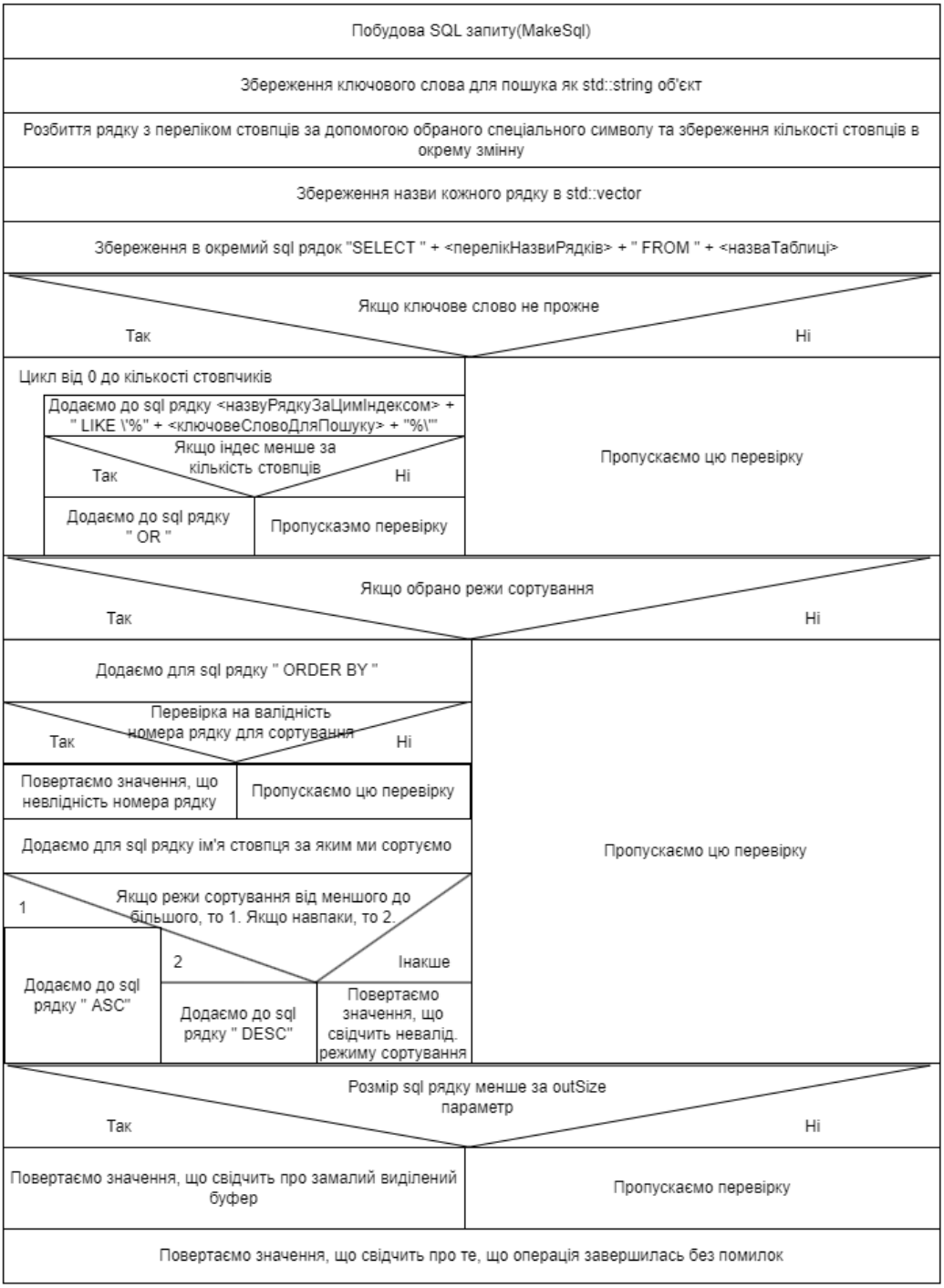

Рисунок 4.2 – Алгоритм для побудови SQL запиту

Наступним буде алгоритм для підрахунку кількості рядків в таблиці, його зображено на рис.4.3. Вхідними даними алгоритму виступає: шлях до бази даних, ім’я таблиці, перелік стовпців через спеціальний символ та ключове слово для пошуку.

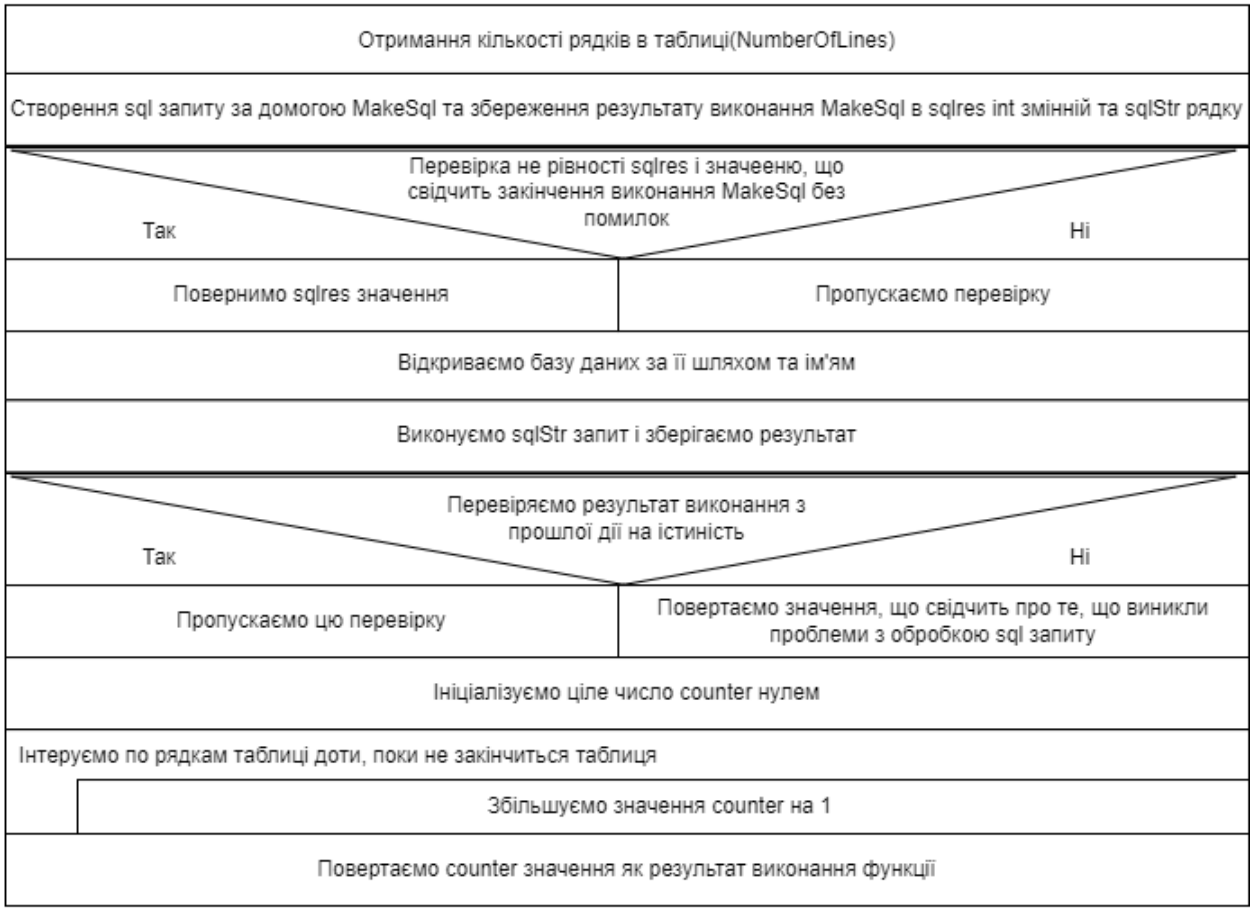

Рисунок 4.3 – Алгоритм для підрахунку кількості рядків в таблиці

На рис. 4.4 наведено алгоритм для отримання інформації з таблиці, що зберігає історії пошукі. Вхідні дані алгоритму: шлях до бази даних; ім'я таблиці; перелік стовпців таблиці через спеціальний символ; рядок з результуючою інформацією; розмір попереднього рядка; номер рядку таблиці з якого почнем записувати інформацію; кількість рядків, що збережемо; режим сортування; стовпчик за яким буде проводитись сортування та ключове слово для пошуку.

Алгоритм для отримання інформації з таблиці, що зберігає логіни та паролі наводитись не буду, тому що алгоритм повністю повторяє логіку викладену на рис. 4.4, але під час запису інформації з кожного рядку та стовпця ми перевіряємо назву стовпця, отримаємо збережене значення, перевіряємо розмір отриманого рядку та дешифруємо за один з двох алгоритмів. Якщо отримане значення більше за 16 символів, то використовуємо DPAPI алгоритм, інакше – AES.

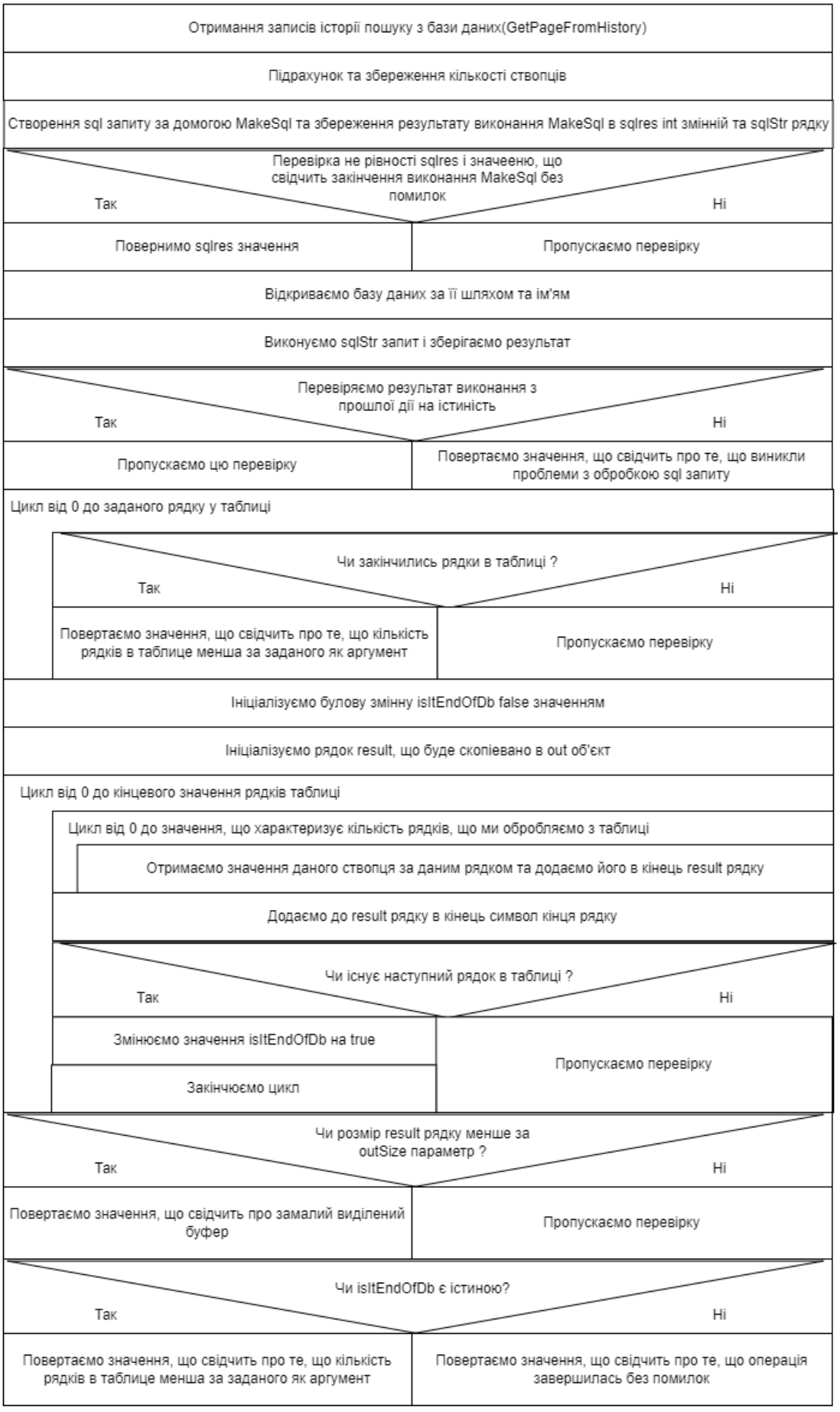

Рисунок 4.3 – Алгоритм для отримання історії пошуку

4.3 Специфікації до експортованих функцій

1. FileCopy – це функція, що відповідає за копіювання файлу за заданим шляхом до іншої директорії. Функція є невеличкою обгорткою над CopyFile, тому наводити алгоритм немає сенсу.
    a. Вхідні дані:
        i. oldFilename – шлях до файлу, що буде копіюватись;
        ii. newFilename – шлях до місця розташування нового файлу;
        iii. isFileInAppDataLocal – флаг, що свідчить про розташування oldFilename файлу десь у “AppData\Local” директорії; якщо флаг дорівнює true, то oldFilename повинно ігнорувати свою частину, що дорівнює шляху до “AppData\Local” директорії, інакше шлях повинен буди повним.
    b. Вихідні дані: ціле значення, що характеризує статус виконання функції; якщо значення дорівнює 0, то функція закінчилась з помилкою, інакше – функція закінчила виконання без проблем.
2. FileDelete – це функція, що відповідає за видаленням файлу за заданим шляхом до іншої директорії. Функція є невеличкою обгорткою над DeleteFile, тому наводити алгоритм немає сенсу.
    a. Вхідні дані:
        i. filename – шлях до файлу, що буде видалятись;
        ii. isFileInAppDataLocal – флаг, що свідчить про розташування filename файлу десь у “AppData\Local” директорії; якщо флаг дорівнює true, то oldFilename повинно ігнорувати свою частину, що дорівнює шляху до “AppData\Local” директорії, інакше шлях повинен буди повним.
    b. Вихідні дані: ціле значення, що характеризує статус виконання функції; якщо значення дорівнює 0, то функція закінчилась з помилкою, інакше – функція закінчила виконання без проблем.

3. ExportTo – це функція, що відповідає за створення нового файлу з заданим текстом або запис цього тексту до існуючого файлу. Функція є невеличкою обгорткою над std::fstream об’єктом.
    a. Вхідні дані:
        i. filename – шлях до файлу;
        ii. text – текст, що буде записаний до файлу.
        iii. isClear – флаг, що свідчить про те, чи буде очищен файл перед доданням тексту, чи ні.
    b. Вихідні дані: булеве значення, що характеризує статус виконання функції; якщо значення дорівнює true, то функція закінчилась без помилок, інакше – функція закінчила виконання з якоюсь проблемою.
4. PathToAppDataLocal – це функція, що відповідає надання шляху до “AppData\Local” директорії. Функція є невеличкою обгорткою над WinAPI SHGetFolderPath функцією.
    a. Вхідні дані: відсутні.
    b. Вихідні дані: шлях до “AppData\Local” директорії.
5. DecryptByDPADPI – це функція, що відповідає за дешифрування даних за допомогою DPAPI функції. Функція є невеличкою обгорткою над DPAPI, тому наводити алгоритм немає сенсу.
    a. Вхідні дані: data – вказівник на початок масиву, що зберігає в собі зашифровані дані, масив має фіксований розмір в 1024 елементів.
    b. Вихідні дані: вказівник на початок масиву дешифрованих даних.
6. GetMasterKey – це функція, що відповідає за отримання значення мастер ключа, котре підлягає base64 обробці та дешифруванню за допомогою DPAPI. Функція є невеличкою обгорткою, тому наводити алгоритм немає сенсу.
    a. Вхідні дані: jsonName – шлях до JSON файлу, що містить в собі мастер ключ.

b. Вихідні дані: вказівник на початок рядку, що містить в собі мастер ключ.

7. DecryptByAes256Gcm – це функція, що відповідає за дешифрування даних за допомогою AES-256-GCM алгоритму. Функція є невеличкою обгорткою над cryptopp об’єктами, тому наводити алгоритм немає сенсу.
    a. Вхідні дані:
        i. bytebuff – вказівник на початок масиву, що збергіає в собі зашифровані дані;
        ii. bytekey – вказівник на початок масиву, що збергіає в собі мастер ключ.
    b. Вихідні дані: вказівник на початок рядку, що містить розшифрований текст.
8. MakeSQL – це функція, що відповідає за побудову SQL запиту за переданими даними.
    a. Вхідні дані:
        i. tablename – ім’я таблиці у базі даних;
        ii. listOfColumns – рядок, що містить перелік стовпців з бази даних через кому;
        iii. outSize – розмір рядку, через який повертається результат побудований SQL запит;
        iv. modeOfSorting – тип сортування;
        v. columnBySorting – номер стовпеця за яким проходить сортування;
        vi. searchWord – рядок, що містить ключове слово для пошуку.
    b. Вихідні дані:
        i. out – рядок, що повинен містити SQL запит;
        ii. ціле значення, що характеризує статус виконання функції; якщо значення дорівнює 0, то функція закінчилась з помилкою, інакше – функція закінчила виконання без проблем.

9. GetPageFromLogin – це функція, що відповідає за повернення кінцевої інформації з бази даних логінів.
    a. Вхідні дані:
        i. dbname – шлях до бази даних;
        ii. tablename – ім’я таблиці у базі даних;
        iii. listOfColumns – рядок, що містить перелік стовпців з бази даних через кому;
        iv. jsonName – шлях до JSON файлу, що містить в собі мастер ключ;
        v. outSize – розмір рядку, через який повертається результат побудований SQL запит;
        vi. startLine – номер рядку, починаючи з якого беремо інформацію;
        vii. numberOfLines – загальня кількість рядків в таблиці;
        viii. modeOfSorting – тип сортування;
        ix. columnBySorting – номер стовпеця за яким проходить сортування;
        x. searchWord – рядок, що містить ключове слово для пошуку.
    b. Вихідні дані:
        i. out – рядок, що повинен містити дані з таблиці логінів;
        ii. ціле значення, що характеризує статус виконання функції; якщо значення дорівнює 0, то функція закінчилась з помилкою, інакше – функція закінчила виконання без проблем.
10. GetPageFromHistory– це функція, що відповідає за повернення кінцевої інформації з бази даних історії пошуку.
    a. Вхідні дані:
        i. dbname – шлях до бази даних;
        ii. tablename – ім’я таблиці у базі даних;

iii. listOfColumns – рядок, що містить перелік стовпців з бази даних через кому;
iv. outSize – розмір рядку, через який повертається результат побудований SQL запит;
v. startLine – номер рядку, починаючи з якого беремо інформацію;
vi. numberOfLines – загальня кількість рядків в таблиці;
vii. modeOfSorting – тип сортування;
viii. columnBySorting – номер стовпеця за яким проходить сортування;
ix. searchWord – рядок, що містить ключове слово для пошуку.

b. Вихідні дані:
i. out – рядок, що повинен містити дані з таблиці історії пошуку;
ii. ціле значення, що характеризує статус виконання функції; якщо значення дорівнює 0, то функція закінчилась з помилкою, інакше – функція закінчила виконання без проблем.

11. NumbOfLines – це функція, що відповідає за отримання кількості рядків в зазначеній таблиці заданої бази даних.
a. Вхідні дані:
i. dbname – шлях до бази даних;
ii. tablename – ім’я таблиці у базі даних;
iii. listOfColumns – рядок, що містить перелік стовпців з бази даних через кому;
iv. searchWord – рядок, що містить ключове слово для пошуку.

b. Вихідні дані: ціле значення, що характеризує статус виконання функції; якщо значення дорівнює 0, то функція закінчилась з помилкою, інакше – функція закінчила виконання без проблем.

12. GetPageFromCahce – це функція, що відповідає за повернення кінцевої інформації з кеш-файлів.
    a. Вхідні дані:
        i. cahceDirPath – шлях до директорії з кеш-файлами;
        ii. outSize – розмір рядку, через який повертається результат побудований SQL запит;
        iii. startLine – номер рядку, починаючи з якого беремо інформацію;
        iv. numberOfLines – загальня кількість рядків в таблиці;
        v. modeOfSorting – тип сортування;
        vi. columnBySorting – номер стовпеця за яким проходить сортування;
        vii. searchWord – рядок, що містить ключове слово для пошуку.
    b. Вихідні дані:
        i. outBuffer – рядок, що повинен містити дані з кеш-файлів;
        ii. ціле значення, що характеризує статус виконання функції; якщо значення дорівнює 0, то функція закінчилась з помилкою, інакше – функція закінчила виконання без проблем.
13. NumbOfLines – це функція, що відповідає за отримання кількості рядків інформації про весь кеш.
    a. Вхідні дані:
        i. cahceDirPath – шлях до директорії з кеш-файлами;
        ii. searchWord – рядок, що містить ключове слово для пошуку.
    b. Вихідні дані: ціле значення, що характеризує статус виконання функції; якщо значення дорівнює 0, то функція закінчилась з помилкою, інакше – функція закінчила виконання без проблем.
14. ExportCache – це функція, що відповідає за експортування всіх кеш-файлів, що підпадають під вимоги, які представлені в вигляді параметрів даної функції.

a. Вхідні дані:
    i. cahceDirPath – шлях до директорії з кеш-файлами;
    ii. sort_column – номер стовпеця за яким проходить сортування;
    iii. ascending – флаг, що характеризує тип сортування;
    iv. entries_offset – зміщення, починаючи з якого беремо інформацію про інші кеш-файли;
    v. entries_number – кількість файлів про які беремо інформацію;
    vi. filter – рядок, що являє собою фільтр;
    vii. export_dir – шлях до директорії куди буде експортуватись кеш-файли;
    viii. save_md5 – флаг, що характеризує потрібно експортувати md5 хеш кеш-файлів.
b. Вихідні дані: відсутні.

15. SaveFileToTemp – це функція, що відповідає за експортування одного окремого файлу.
    a. Вхідні дані:
        i. cahceDirPath – шлях до директорії з кеш-файлами;
        ii. cahce_addr – ціле число, що характеризує адрес заданого кеш-файлу;
        iii. out_path – шлях до експортованого кеш-файлу;
        iv. out_path_size – розмір рядку, що містить інформацію про кеш-файли.
    b. Вихідні дані: відсутні.

РОЗДІЛ 5. ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

5.1 Тестування методом білої скриньки

Як метод білої скриньки було обрано метод покриття умов, що передбачає створення такого набору тестів, за яким кожна умова виконається та не виконається хоча б один раз. Результати тестування оформлено у вигляді таблиці, де по вертикалі номери умови, а по горизонталі − тести.

Нижче наведено таблиці з результатами тестування за методом покриття умов всіх експортованих функції, специфікації функцій можна переглянути в п. 4.3.

Таблиця 5.1 – Тестування FileCopy функції

Номери умови	Умови	Тести	
		“C:\\1.txt”, “D:\\2.txt”, false	“\\1.txt”, “D:\\2.txt”, true
1	Чи розташован файл в “AppData\Local”	-	+

Таблиця 5.2 – Тестування FileDelete функції

Номери умови	Умови	Тести	
		“C:\\1.txt”, false	“\\1.txt”, true
1	Чи розташован файл в “AppData\Local”	-	+

Таблиця 5.3 – Тестування ExportTo функції

Номери умови	Умови	Тести	
		“C:\\1.txt”, “”, false	“C:\\1.txt”, “t\n”, true
1	Чи потрібно очистити файл	-	+

Продовження таблиці 5.3

2	Чи потрібно не очищювати файл	+	-
3	Чи файл не валідний	-	-
4	Чи індекс менше кількості рядків та розмір рядка більше за 0	-	+
5	Чи знайден символ кінця рядку	-	+
6	Чи не знайден символ кінця рядку	-	+

Таблиця 5.4 – Тестування PathToAppDataLocal функції

Номери умови	Умови	Тести
		Нічого
1	Чи вдало отримали шлях до “AppData\Local”	-
2	Чи не вдалось отримали шлях до “AppData\Local”	+

Таблиця 5.5 – Тестування DecryptByDPAPI функції

Номери умови	Умови	Тести	
		“smthg”	<decryptedText>
1	Чи індекс менше за 1024	+	+
2	Чи вдалось розшифрувати	-	+
3	Чи індекс менше за розмір отриманого буферу	-	+
4	Чи індекс менше за розмір розшифрованого буферу	-	+

Таблиця 5.6 – Тестування GetMasterKey функції

Номери умови	Умови	Тести	
		<nonExistentFile>	<filename>
1	Чи при читанні файлу дійшли ми до його кінця	-	+
2	Чи знайдено “encrypted_key”: в даному рядку	-	+
3	Чи індекс менше за розмір ключа	-	+

Таблиця 5.7 – Тестування DecryptByAes256Gcm функції

Номери умови	Умови	Тести	
		“”, “”	<decryptedText>, <masterKey>
1	Чи індекс менше за 32	-	+
2	Чи індекс менше за розмір буфера	-	+

Таблиця 5.8 – Тестування GetPageFromLogin функції

Номери умови	Умови	Тести					
		“…”, “logins”, “password_value”, “…”, 1000, 0, 2, 0, 0, “”	““…”, “logins”, “id”, “…”, 1000, 0, 2, 0, -1, “”	“”, “”, “id”, “”, 0, 0, 0, 0, 0, “”	““…”, “logins”, “id”, “…”, 1000, 0, 10, 0, -1, “””	“…”, “logins”, “id”, “…”, 1, 0, 2, 0, 0, “”	“…”, “logins”, “id”, “…”, 1000, 2, 5, 0, 0, “”
1	Чи є поле з зашифрованими даними	+	-	-	-	+	-
2	Чи вдало було створено SQL запит	-	+	-	-	-	-
3	Чи вдалось виконати SQL запит	-	-	+	-	-	-
4	Чи індекс менше за номер початковий рядок	+	-	-	-	+	+
5	Чи не вдалось перейти на наступний рядок	-	-	-	+	-	-

Продовження таблиці 5.8

6	Чи індекс менше за кількість рядоків	+	-	-	-	+	+
7	Чи індекс менше за кількість стовпців	+	-	-	-	+	+
8	Чи індекс дорівнює кількості стовпців	+	-	-	-	+	+
9	Чи розмір отриманої ячейки більше за 16	+	-	-	-	+	+
10	Чи розмір отриманої ячейки менше за 16	-	-	-	-	-	-
11	Чи не вдалось перейти на наступни рядок	-	-	-	-	-	+
12	Чи не замалий розмір буфер	-	-	-	-	+	-
13	Чи дійшли до кінця файлу	-	-	-	-	-	+

Таблиця 5.9 – Тестування GetPageFromHistory функції

Номери умови	Умови	Тести					
		“…”, “urls”, “id”, “…”, 1000, 0, 2, 0, 0, “”	““…”, “urls”, “id”, “…”, 1000, 0, 2, 0, -1, “”	“”, “”, “id”, “”, 0, 0, 0, 0, 0, “”	““…”, “urls”, “id”, “…”, 1000, 0, 10, 0, -1, “””	“…”, “urls”, “id”, “…”, 1, 0, 2, 0, 0, “”	“…”, “urls”, “id”, “…”, 1000, 2, 5, 0, 0, “”
1	Чи вдало було створено SQL запит	+	-	-	-	+	-
2	Чи вдалось виконати SQL запит	-	+	-	-	-	-
3	Чи індекс менше за номер початковий рядок	-	-	+	-	-	-
4	Чи не вдалось перейти на наступний рядок	+	-	-	-	+	+
5	Чи індекс менще за кількість рядків	-	-	-	+	-	-

Продовження таблиці 5.9

6	Чи індекс менше за кількість стовпців	+	-	-	-	+	+
7	Чи не вдалось перейти на наступний рядок	+	-	-	-	+	+
9	Чи вдалось перейти на наступний рядок	-	-	-	-	-	+
10	Чи не замалий розмір буфер	-	-	-	-	+	-
11	Чи дійшли до кінця файлу	-	-	-	-	-	+

Таблиця 5.10 – Тестування NumbOfLines функції

Номери умови	Умови	Тести		
		“…”, “urls”, “”, “”	“”, “urls”, “id”, “”	“…”, “urls”, “id”, “asd”
1	Чи вдало було створено SQL запит	+	-	-
2	Чи вдалось виконати SQL запит	-	+	-
3	Чи вдалось перейти на наступний рядок	-	-	+

Таблиця 5.11 – Тестування GetPageFromCahce функції

Номери умови	Умови	Тести			
		“”, 0, 0, 1, 0, 0, “”	“…”, 1000, 0, 1, 0, 0, “”	“…”, 0, 0, 1, 0, 0, “”	“…”, 1000, <lastRecord>, 1000, 0, 0, “”
1	Чи дійшли до кінця контейнеру	-	+	+	+
2	Чи не замалий розмір буфер	-	-	+	+
3	Чи дійшли до кінця файлу	-	-	-	+

5.2 Тестування програми методом чорної скриньки

Тестування методом чорної скриньки виконано на основі unit-тестів. Unit-тести були написані для всіх експортованих функцій, окрім функцій для взаємодії з кешем, бо не було знайдено нормального шляху для створення тестових файлів, що зберігали би інформацію про кеш файли.

Для FileCopy функції було створено 4 unit-тесту: коли копіюємо неіснуючий файл, коли копіюємо існуючий файл, коли копіюємо неіснуючий файл до “AppData\Local” директорії та коли копіюємо існуючий файл “AppData\Local” директорії.

Для FileDelete функції було створено 4 unit-тесту: коли видаляємо неіснуючий файл, коли видаляємо існуючий файл, коли видаляємо неіснуючий файл з “AppData\Local” директорії та коли видаляємо існуючий файл з “AppData\Local” директорії.

Для ExportTo функції було створено 2 unit-тесту: коли експортуємо текст у існуючий та неіснуючі файли.

Для DecryptDPAPI функції було створено 2 unit-тесту: коли дешифруємо зашифрований текст та коли дешифруємо будь-який випадковий рядок.

Для MakeSQL функції було створено 4 unit-тесту: коли створюємо рядок без помилок та коли отримаємо кожну з можливих помилок.

Для NumbOfLines функції було створено 2 unit-тесту: коли не існує база даних\таблиця та коли все виконється правильно.

Для GetPageFromLogin\GetPageFromHistory функцій було створено 4 unit-тесту: коли читаємо всі рядки з таблиці, коли читаємо не всі рядки, коли передаємо замалий буфер та коли намагаємось отримати інформацію з неіснуючої таблиці/бази даних. Дві функції були об’єднані для тестування тому, що вони мають одне й ту ж саму логіку для взаємодії з базами даних.

Наведені вище unit-тести покривають усі можливі варіанти виконання функцій. На рис. 5.1 наведено результат виконнаня unit-тестів.

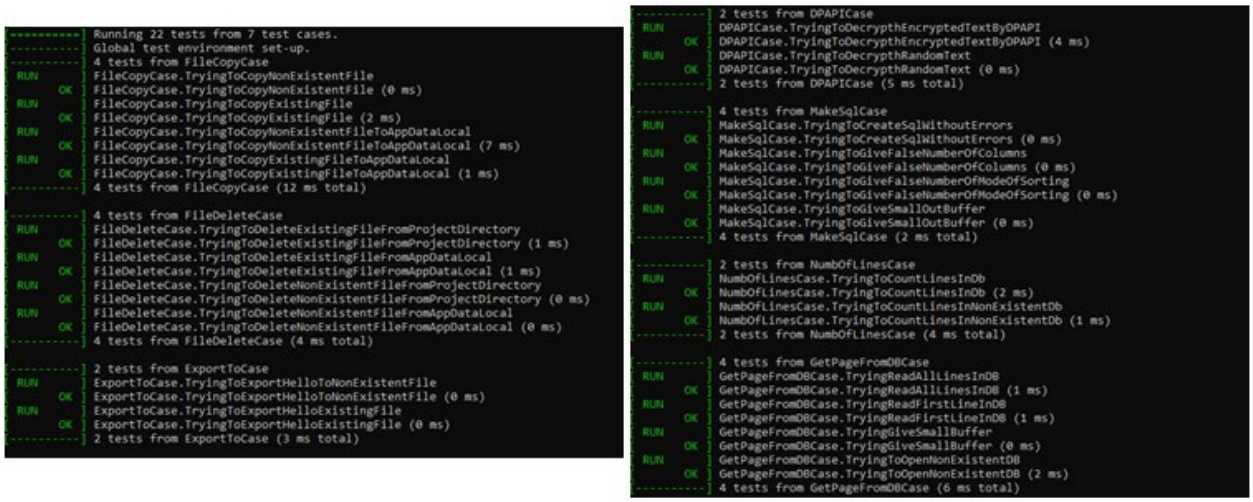

Рисунок 5.1 – Результат виконання unit-тестів

5.3 Налагодження програми

Результати налагодження програми наведено у табл. 5.12. Опис помилки має визначення у чому полягає розбіжність тестованої програмної сутності зі специфікацією. Опис умов містить набори вхідних даних, за яких сталася помилка. Способи усунення включають одну або декілька гіпотез з приводу можливих дій для усунення помилки.

Таблиця 5.12 – Протокол налагодження програми

Опис помилки	Опис ситуації	Способи усунення	Дії, що були застосовані для усунення
Функція PathToAppDataLocal повертала порожній рядок	Ця функція не приймає ніяких параметрів	Використання SHGetFolderPath Win API функції	Опис дії для усунення наведений в попередньому стовпці
MakeSQL повертала порожній SQL запит в NumbOfLines та GetPageFrom***	Буфер, що передається в функцію як out параметр має замалий розмір	Додати логіку для обчислення приблизного розміру SQL запиту	Опис дії для усунення наведений в попередньому стовпці
GetMasterKey не повертало розшифроване значення мастер ключа	DecryptByDPAPI не може розшифрувати мастер ключ рядок	Вивчити вхідні дані в DecryptByDPAPI, та змінити їх для правильного розшифрування	Після base64 декодування слід видалити «DPAPI» символи у початку мастер ключа
Помилка при копіюванні С рядків	strcpy не може нормально скомпілюватись	Або додати макрос для дозволу роботи незахищених функцій, або використати аналог strcpy_s	Використав аналог – strcpy_s

Продовження таблиці 5.12

При зміні відображеної інформації на кеш програма видавала помилку	Програма не могла обробити кеш інформацію	Ручним відлагодженням програми знайти місце самої помилки	Виявилось, що Google Chrome трохи змінив місце розташування кешу, тому були додані зміни до рядку зі шляхом розташування кешу
При експортуванні кешу за заданим шляхом відбувалась помилка	При експортуванні кешу за заданим шляхом при наявності схожого шляху відбувались помилки	Додати логіку для обробки ситуації при наявності обраного шляху	Додав функцію з створення директорії за обраним шляхом тільки з “(2/3/…)” рядком в кінці
При експортуванні MD5 хешу кеш-файлів за заданим шляхом відбувалась помилка	При підрахунку MD5 хешу для великих кеш-файлів відбувалась помилка	Вивчення функції, що відповідає за підрахунок кешу	Розбив логіку для читання файлу за чанками

ВИСНОВКИ

Результатом виконання дипломного проекту стала розробка програмного застосунку для обробки та відображення локально збережених даних Google Chrome веб-браузера. Програмний продукт розроблений об’єктно- орієнтованими мовами програмування – C++ та C#(використовуючи .NET framework). На C++ розроблено DLL, що експортує потрібну нам логіка, а на C# розроблено графічний інтерфейс для програми. Середовища розробки було обрано програмне забезпечення Visual Studio 2017.

Було проведено анкетування користувачів, дослідження існуючих програмних аналогів, проаналізовані основні недоліки, запропоновані програмні та інтерфейсні рішення для покращення роботи.

Розроблене програмне забезпечення надає можливість експортувати будь-яку відображену інформацію: історію пошуків, збережені логіни або збережений кеш.

Необхідними можливостями, які забезпечує додаток, є:

- відображення даних історії пошуків, збережених логінів і кешу;
- експорту історії пошуків та збережених логінів у форматі *.csv;
- можливість експорту кешу;
- можливість пошуку за ключовим словом серед отриманих результатів;
- можливість сортування отриманих результатів.

В ході роботи було поглиблено знання та навички використання об'єктно-орієнтованих мов. Створення додатку надало можливість пройти всі етапи розробки і опрацювати різні аспекти: візуальна та функціональна складові і тд.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. How to Recover Saved Passwords in Google Chrome [Virtual Resource] / Xen Armor. – Edit: 2019 – 10 August – Access Mode: URL: https://xenarmor.com/how-to-recover-saved-passwords-google-chrome/
2. How to crack Chrome password with Python? [Virtual Resource] / Yicong. – Edit: 2021 – 5 January – Access Mode: URL: https://ohyicong.medium.com/how-to-hack-chrome-password-with-python-1bedc167be3d
3. Web Browsers Forensics [Virtual Resource] / Nasreddine Bencherchali. – Edit: 2019 – 20 September – Access Mode: URL: https://nasbench.medium.com/web-browsers-forensics-7e99940c579a
4. Browser History Examiner – User Guide [Virtual Resource] / foxton. – Edit: 2023 – 11 June – Access Mode: URL: https://www.foxtonforensics.com/browser-history-examiner/chrome-history-location
5. Has the user logged into this account, or not? [Virtual Resource] / atropos4n6. – Edit: 2020 – 15 September – Access Mode: URL: https://atropos4n6.com/windows/chrome-login-data-forensics/
6. Disk Cache [Virtual Resource] / chromium. – Edit: 2023 – 10 June – Access Mode: URL: https://www.chromium.org/developers/design-documents/network-stack/disk-cache/

ДОДАТКИ

ДОДАТОК А

Native.dll

FileType.hpp

```
#pragma once


// Provides information about what type of file is stored in the cache
enum class FileType
{
    /* Separate file on disk */
    EXTERNAL      = 0,
    /* Rankings block file */
    RANKINGS      = 1,
    /* 256 byte block file */
    BLOCK_256     = 2,
    /* 1k byte block file */
    BLOCK_1K      = 3,
    /* 4k byte block file */
    BLOCK_4K      = 4,
    /* External files block file */
    BLOCK_FILES   = 5,
    /* Active entries block file */
    BLOCK_ENTRIES = 6,
    /* Evicted entries block file */
    BLOCK_EVICTED = 7
};
```

utils.hpp

```
#pragma once

#include <string>
#include <fstream>
#include <memory>
#include <stdexcept>
#include <stdlib.h>
#include <shlobj_core.h>
#include <wchar.h>
#include "direct.h"

struct GLOBALS
{
    static const std::wstring APP_NAME;
};

namespace Utils
{
    std::wstring CombinePath(const std::wstring& a,
                             const std::wstring& b);

    std::wstring CombinePath(const std::wstring& a,
                             const std::wstring& b,
                             const std::wstring& c);

    std::wstring CombinePath(const std::wstring& a,
                             const std::wstring& b,
                             const std::wstring& c,
```

```
                              const std::wstring& d);

    int FileCopy(const std::wstring& oldFilename,
                 const std::wstring& newFilename);

    void CreateDir(const std::wstring& path);

    std::wstring GetTempDirPath();

    std::wstring CreateTempCopy(const std::wstring& from);

    long GetFileSize(const std::wstring& filename);

    bool IsFileExists(const std::wstring& name);

    std::wstring CreateNoDupPath(const std::wstring& dir,
                                 const std::wstring& name,
                                 const std::wstring& ext);
};
```

utils.cpp

```
#include "utils.hpp"

const std::wstring GLOBALS::APP_NAME = L"WebBrowserArtifactViewer";

std::wstring Utils::CombinePath(const std::wstring& a,
    const std::wstring& b)
{
    // avoid if 'a' already have backslash e.g. "dir\subdir\"
    if (a.size() != 0 &&
        a[a.size() - 1] == L'\\')
        return a + b;

    return a + L'\\' + b;
}

std::wstring Utils::CombinePath(const std::wstring& a,
    const std::wstring& b,
    const std::wstring& c)
{
    return CombinePath(a, CombinePath(b, c));
}

std::wstring Utils::CombinePath(const std::wstring& a,
    const std::wstring& b,
    const std::wstring& c,
    const std::wstring& d)
{
    return CombinePath(CombinePath(a, b), CombinePath(c, d));
}


int Utils::FileCopy(const std::wstring& oldFilename,
    const std::wstring& newFilename)
{
    return CopyFileW(oldFilename.c_str(), newFilename.c_str(), FALSE);
}

void Utils::CreateDir(const std::wstring& path)
{
    auto _ = _wmkdir(path.c_str());
}

std::wstring Utils::GetTempDirPath()
```

```
{
    DWORD dwRetVal = 0;
    TCHAR lpTempPathBuffer[MAX_PATH];
    std::wstring w;
    dwRetVal = GetTempPath(MAX_PATH,          // length of the buffer
        lpTempPathBuffer); // buffer for path
    std::wstring wstr(&lpTempPathBuffer[0]);
    return wstr;
}

std::wstring Utils::CreateTempCopy(const std::wstring& from)
{
    auto temp_dir = GetTempDirPath();
    auto subdir = CombinePath(temp_dir, GLOBALS::APP_NAME);

    CreateDir(subdir);

    wchar_t drive[_MAX_DRIVE] = { 0 };
    wchar_t dir[_MAX_DIR] = { 0 };
    wchar_t fname[_MAX_FNAME] = { 0 };
    wchar_t ext[_MAX_EXT] = { 0 };
    _wsplitpath_s(from.c_str(), drive, dir, fname, ext);
    std::wstring filename(fname);
    std::wstring extension(ext);

    auto result_path = CombinePath(subdir, filename + extension);

    FileCopy(from, result_path);

    return result_path;
}

long Utils::GetFileSize(const std::wstring& filename)
{
    struct _stat stat_buf;
    int rc = _wstat(filename.c_str(), &stat_buf);
    return rc == 0 ? stat_buf.st_size : -1;
}

bool Utils::IsFileExists(const std::wstring& name)
{
    std::ifstream f(name.c_str());
    return f.good();
}

std::wstring Utils::CreateNoDupPath(const std::wstring& dir, const std::wstring& name,
const std::wstring& ext)
{
    std::wstring result = Utils::CombinePath(dir, name + ext);

    int counter = 2;
    while (Utils::IsFileExists(result))
    {
        if (ext != L"")
        {
            result = Utils::CombinePath(
                dir,
                name +
                L"(" + std::to_wstring(counter++) + L")" +
                ext);
        }
        else
        {
            int last_dot_pos = name.find_last_of('.');
```

```
            std::wstring n, e;

            if (last_dot_pos == -1)
            {
                n = name.substr(0, name.size());
                e = L"";
            }
            else
            {
                n = name.substr(0, last_dot_pos);
                e = name.substr(last_dot_pos, name.size());
            }

            result = Utils::CombinePath(
                dir,
                n +
                L"(" + std::to_wstring(counter++) + L")" +
                e);
        }
    }

    return result;
}
```

CacheAddr.hpp

```
#pragma once

#include <cstdint>
#include <string>
#include <sstream>
#include <iomanip>
#include <ios>

#include "FIleType.hpp"

// Defines a storage address for a cache record
class CacheAddr
{
public:
    CacheAddr();
    CacheAddr(uint32_t address);

    // Returns whole addr value
    uint32_t GetAddress() const;

    // Returns true if initialized bit set to 1
    bool IsValid() const;

    // Returns file type
    FileType GetFileType() const;

    // Returns separated file name. Name valid if FileType = EXTERNAL only. Name example:
'f_000f74'
    std::wstring GetFileName() const;

    // Returns number of contiguous blocks taken in data_n file
    int GetNumberOfBlocks() const;

    // Returns number of data_n file
    int GetDataFileNumber() const;

    // Returns number of the block in data_n file, where data starts
    int GetStartBlockOffset() const;
```

```
private:
    uint32_t value_;

    static const uint32_t kInitializedMask    = 0x80000000;
    static const uint32_t kInitializedOffset  = 31;
    static const uint32_t kFileTypeMask       = 0x70000000;
    static const uint32_t kFileTypeOffset     = 28;
    static const uint32_t kReservedBitsMask   = 0x0c000000;
    static const uint32_t kNumBlocksMask      = 0x03000000;
    static const uint32_t kNumBlocksOffset    = 24;
    static const uint32_t kFileSelectorMask   = 0x00ff0000;
    static const uint32_t kFileSelectorOffset = 16;
    static const uint32_t kStartBlockMask     = 0x0000FFFF;
    static const uint32_t kFileNameMask       = 0x0FFFFFFF;
};
```

CacheAddr.cpp

```
#include "CacheAddr.hpp"

CacheAddr::CacheAddr() :
        value_(0){}

CacheAddr::CacheAddr(uint32_t address) :
        value_(address){}

uint32_t CacheAddr::GetAddress() const
{
    return value_;
}

bool CacheAddr::IsValid() const
{
    return (value_ & kInitializedMask) >> kInitializedOffset;
}

FileType CacheAddr::GetFileType() const
{
    return FileType((value_ & kFileTypeMask) >> kFileTypeOffset);
}

int CacheAddr::GetNumberOfBlocks() const
{
    return ((value_ & kNumBlocksMask) >> kNumBlocksOffset) + 1;
}

int CacheAddr::GetDataFileNumber() const
{
    return (value_ & kFileSelectorMask) >> kFileSelectorOffset;
}

int CacheAddr::GetStartBlockOffset() const
{
    return value_ & kStartBlockMask;

}

std::wstring CacheAddr::GetFileName() const
{
    int fileNumber = value_ & kFileNameMask;
    std::stringstream stream;
    stream << "f_"
        << std::setfill('0') << std::setw(6) // crop to last 6 bytes
        << std::hex << fileNumber;
```

```
    auto str = stream.str();
    return std::wstring(str.begin(), str.end());
}
```

exceptions.hpp

```
#pragma once
#include <stdexcept>
#include <string>

#include "Cache/CacheAddr.hpp"

struct file_not_found_error : public std::runtime_error
{
    file_not_found_error(const std::string& file_path);
    file_not_found_error(const std::string& file_path, int code);
};

struct invalid_index_file : public std::runtime_error
{
    invalid_index_file();
};

struct no_data_entry : public std::runtime_error
{
    no_data_entry(int file_num, int block_num, int block_size);
};

struct index_wrong_address : public std::runtime_error
{
    index_wrong_address(const CacheAddr& address);
};

struct file_not_block_type : public std::runtime_error
{
    file_not_block_type();
};

struct dll_fail_load_dll : public std::runtime_error
{
    dll_fail_load_dll(const std::string& dll_name, unsigned long err_code);
};

struct dll_fail_load_magic : public std::runtime_error
{
    dll_fail_load_magic();
};

struct dll_fail_load_magic_db : public std::runtime_error
{
    dll_fail_load_magic_db(const std::string& magic_error);
};

struct dll_fail_run_magic_file : public std::runtime_error
{
    dll_fail_run_magic_file();
};

struct skip_entry_exception : public std::runtime_error
{
    skip_entry_exception(const std::string& message);
};

struct no_url : public skip_entry_exception
```

```
{
    no_url();
};

struct no_external_file : public skip_entry_exception
{
    no_external_file();
};

struct invalid_data_address : public skip_entry_exception
{
    // no data address found
    invalid_data_address();
};

struct addr_wrong_type : public std::runtime_error
{
    // file address points to wrong file type
    addr_wrong_type();
};

struct md5_failed : public std::runtime_error
{
    md5_failed(const std::string& message, int status);
};
```

CahceEntry.hpp

```
#pragma once

#include <string>

struct CacheEntry
{
    /* Column 0: Cache addr value */
    uint32_t address_index;

    uint32_t address_data;

    /* Column 1: File url */
    std::wstring url;

    /* Column 2: File name */
    std::wstring file_name;

    /* Column 3: File extention / mime type */
    std::wstring extension;

    /* Column 4: File size */
    size_t size;

    CacheEntry() = default;

    CacheEntry(uint32_t address_index,
               uint32_t address_data,
               const std::wstring& url,
               const std::wstring& file_name,
               const std::wstring& extension,
               size_t size) :
        address_index(address_index),
        address_data(address_data),
        url(url),
        file_name(file_name),
        extension(extension),
        size(size) {}
```

```
bool ContainsFilter(const std::wstring& filter)
{
    auto name = file_name + extension;

    return
        name.find(filter) != std::string::npos ||
        url.find(filter)  != std::string::npos;
}

/* Sort by index */
static bool CompareBy_0(const CacheEntry& a,
                        const CacheEntry& b)
{
    return a.address_index < b.address_index;
}

/* Sort by url */
static bool CompareBy_1(const CacheEntry& a,
                        const CacheEntry& b)
{
    return a.url < b.url;
}

/* Sort by name */
static bool CompareBy_2(const CacheEntry& a,
                        const CacheEntry& b)
{
    return a.file_name < b.file_name;
}

/* Sort by type */
static bool CompareBy_3(const CacheEntry& a,
                        const CacheEntry& b)
{
    return a.extension < b.extension;
}

/* Sort by size */
static bool CompareBy_4(const CacheEntry& a,
                        const CacheEntry& b)
{
    return a.size < b.size;
}

/* Sort by index descending */
static bool CompareBy_0_descending(const CacheEntry& a,
                        const CacheEntry& b)
{
    return a.address_index > b.address_index;
}

/* Sort by url descending */
static bool CompareBy_1_descending(const CacheEntry& a,
                        const CacheEntry& b)
{
    return a.url > b.url;
}

/* Sort by name descending */
static bool CompareBy_2_descending(const CacheEntry& a,
                        const CacheEntry& b)
{
    return a.file_name > b.file_name;
```

```
    }

    /* Sort by type descending */
    static bool CompareBy_3_descending(const CacheEntry& a,
                                       const CacheEntry& b)
    {
        return a.extension > b.extension;
    }

    /* Sort by size descending */
    static bool CompareBy_4_descending(const CacheEntry& a,
                                       const CacheEntry& b)
    {
        return a.size > b.size;
    }
};
```

IIndexReader.hpp

```
#pragma once

#include <fstream>
#include <vector>

#include "../CacheAddr.hpp"

struct IIndexReader
{
       virtual ~IIndexReader() {};
       virtual void Open(const std::wstring& cache_dir_path) = 0;
       virtual CacheAddr Next() = 0;
       virtual void Skip(int n) = 0;
protected:
       virtual  std::wstring CopyIndexFile(const std::wstring& cache_dir_path) = 0;
};
```

IndexReader.hpp

```
#pragma once

#include "IIndexReader.hpp"

#include "../CacheAddr.hpp"
#include "../../exceptions.hpp"
#include "../../utils.hpp"

class IndexReader : public IIndexReader
{
public:
    IndexReader() = default;

    // Opens index file for reading and prepares to read addresses
    void Open(const std::wstring& cache_dir_path) override;

    // Reads next valid address. Returns 0 address if no more addresses left.
    CacheAddr Next() override;

    // Skips n addresses
    void Skip(int n) override;

private:
    std::wstring CopyIndexFile(const std::wstring& cache_dir_path) override;

    // Size of index file header
```

```
    const int HEADER_SIZE = 0x0170;

    // Size of index file data
    const int DATA_SIZE   = 0x080000;

    // Total file size
    const int INDEX_FILE_SIZE = HEADER_SIZE + DATA_SIZE;

    // Size of buffer of values
    const size_t BUFFER_SIZE = DATA_SIZE / sizeof(uint32_t);

    const std::wstring PREFIX_INDEX = L"index";


    // Buffer of addr values
    std::vector<uint32_t> buffer_;

    // Current buffer position
    std::vector<uint32_t>::const_iterator position_citer_;
};
```

IndexReader.cpp

```
#include "IndexReader.hpp"

using  Utils::CombinePath;
using  Utils::CreateTempCopy;
using  Utils::GetTempDirPath;
using  Utils::CreateDir;
using  Utils::FileCopy;

void IndexReader::Open(const std::wstring &cache_dir_path)
{
    auto index_path = CopyIndexFile(cache_dir_path);

    // Check file existance and size
    auto size = Utils::GetFileSize(index_path);


    // Read buffer
    std::fstream file(index_path, std::ios_base::in | std::ios_base::binary);

    if (file.fail())
        throw file_not_found_error("index");

    file.ignore(HEADER_SIZE);

    size -= HEADER_SIZE;

    buffer_ = std::vector<uint32_t>(size);
    file.read(reinterpret_cast<char*>(buffer_.data()), size);

    position_citer_ = buffer_.cbegin();
}

CacheAddr IndexReader::Next()
{
    uint32_t value;

    while (position_citer_ != buffer_.cend())
    {
        CacheAddr addr(*position_citer_++);
        if (addr.IsValid())
            return addr;
```

```
    }
    return CacheAddr();
}

void IndexReader::Skip(int n)
{
    for (int i = 0; i < n; i++)
        Next();
}

std::wstring IndexReader::CopyIndexFile(const std::wstring& cache_dir_path)
{
    auto path_from = CombinePath(cache_dir_path, PREFIX_INDEX);
    auto temp_dir = GetTempDirPath();
    auto dir = CombinePath(temp_dir, GLOBALS::APP_NAME);
    CreateDir(dir);

    auto path_to = CombinePath(dir, PREFIX_INDEX);
    FileCopy(path_from.c_str(), path_to);
    return path_to;
}
```

IExtensionReader.hpp

```
#pragma once

#include <string>

struct IExtensionReader
{
	virtual ~IExtensionReader() {};
	virtual std::wstring GetFileExtension(const std::wstring& path) = 0;
};
```

ExtensionReader.hpp

```
#pragma once

#include "IExtensionReader.hpp"

#include <windows.h>
#include <stdio.h>
#include <errno.h>
#include <fcntl.h>
#include <time.h>
#include <sys/types.h>
#ifndef WIN32
#include <sys/param.h>
#endif
#include <sys/stat.h>
#include <stdarg.h>
#include <comdef.h>
#include "../../../exceptions.hpp"
#include "../../../utils.hpp"

class ExtensionReader : public IExtensionReader
{
public:
	ExtensionReader();

	~ExtensionReader() override;

	std::wstring GetFileExtension(const std::wstring& path) override;

private:
```

```
        static const std::wstring DLL_FILE_NAME;

        static const int MAXDESC = 64;    /* max len of text description/MIME type */
        static const int MAXMIME = 80;    /* max len of text MIME type */
        static const int MAXstring = 128;    /* max len of "string" types */
        static const int MAGICNO = 0xF11E041C;
        static const int VERSIONNO = 16;
        static const int FILE_MAGICSIZE = 376;
        static const int FILE_GUID_SIZE = sizeof("XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX");

        static const int MAGIC_SETS = 2;

        static const int MAGIC_NONE =                          0x0000000; /* No flags */
        static const int MAGIC_DEBUG =                         0x0000001; /* Turn on
debugging */
        static const int MAGIC_SYMLINK =               0x0000002; /* Follow symlinks */
        static const int MAGIC_COMPRESS =              0x0000004; /* Check inside
compressed files */
        static const int MAGIC_DEVICES =               0x0000008;/* Look at the contents
of devices */
        static const int MAGIC_MIME_TYPE =             0x0000010; /* Return the MIME type
*/
        static const int MAGIC_CONTINUE =              0x0000020; /* Return all matches
*/
        static const int MAGIC_CHECK =                         0x0000040; /* Print
warnings to stderr */
        static const int MAGIC_PRESERVE_ATIME =        0x0000080; /* Restore access time
on exit */
        static const int MAGIC_RAW =                           0x0000100; /* Don't convert
unprintable chars */
        static const int MAGIC_ERROR =                         0x0000200; /* Handle ENOENT
etc as real errors */
        static const int MAGIC_MIME_ENCODING =         0x0000400; /* Return the MIME
encoding */
        static const int MAGIC_MIME =                          (MAGIC_MIME_TYPE |
MAGIC_MIME_ENCODING);
        static const int MAGIC_APPLE =                         0x0000800; /* Return the
Apple creator/type */
        static const int MAGIC_EXTENSION =             0x1000000; /* Return a /-separated
list of * extensions */
        static const int MAGIC_COMPRESS_TRANSP = 0x2000000; /* Check inside compressed
files * but not report compression */
        static const int MAGIC_NODESC =                        (MAGIC_EXTENSION |
MAGIC_MIME | MAGIC_APPLE);

        static const int MAGIC_NO_CHECK_COMPRESS =     0x0001000; /* Don't check for
compressed files */
        static const int MAGIC_NO_CHECK_TAR =          0x0002000;/* Don't check for tar
files */
        static const int MAGIC_NO_CHECK_SOFT =         0x0004000; /* Don't check magic
entries */
        static const int MAGIC_NO_CHECK_APPTYPE =0x0008000; /* Don't check application
type */
        static const int MAGIC_NO_CHECK_ELF =          0x0010000;/* Don't check for elf
details */
        static const int MAGIC_NO_CHECK_TEXT =         0x0020000; /* Don't check for text
files */
        static const int MAGIC_NO_CHECK_CDF =          0x0040000; /* Don't check for cdf
files */
        static const int MAGIC_NO_CHECK_CSV =          0x0080000;/* Don't check for CSV
files */
        static const int MAGIC_NO_CHECK_TOKENS = 0x0100000; /* Don't check tokens */
        static const int MAGIC_NO_CHECK_ENCODING =     0x0200000; /* Don't check text
encodings */
```

```
	static const int MAGIC_NO_CHECK_JSON =		0x0400000;/* Don't check for JSON
files */

	static const int STDIN_FILENO = 0;

	struct buffer
	{
		int fd;
		struct stat st;
		const void* fbuf;
		size_t flen;
		off_t eoff;
		void* ebuf;
		size_t elen;
	};

	union VALUETYPE
	{
		uint8_t b;
		uint16_t h;
		uint32_t l;
		uint64_t q;
		uint8_t hs[2];	/* 2 bytes of a fixed-endian "short" */
		uint8_t hl[4];	/* 4 bytes of a fixed-endian "long" */
		uint8_t hq[8];	/* 8 bytes of a fixed-endian "quad" */
		char s[MAXstring];	/* the search string or regex pattern */
		unsigned char us[MAXstring];
		uint64_t guid[2];
		float f;
		double d;
	};

	struct magic {
		/* Word 1 */
		uint16_t cont_level;	/* level of ">" */
		uint8_t flag;

		uint8_t factor;

		/* Word 2 */
		uint8_t reln;		/* relation (0=eq, '>'=gt, etc) */
		uint8_t vallen;		/* length of string value, if any */
		uint8_t type;		/* comparison type (FILE_*) */
		uint8_t in_type;	/* type of indirection */

		/* Word 3 */
		uint8_t in_op;		/* operator for indirection */
		uint8_t mask_op;	/* operator for mask */
		uint8_t dummy;
		uint8_t factor_op;


		/* Word 4 */
		int32_t offset;		/* offset to magic number */
		/* Word 5 */
		int32_t in_offset;	/* offset from indirection */
		/* Word 6 */
		uint32_t lineno;	/* line number in magic file */
		/* Word 7,8 */
		union
		{
			uint64_t _mask;	/* for use with numeric and date types */
			struct
			{
```

```
                        uint32_t _count;    /* repeat/line count */
                        uint32_t _flags;    /* modifier flags */
                } _s;       /* for use with string types */
        } _u;

        /* Words 9-24 */
        union VALUETYPE value;    /* either number or string */
        /* Words 25-40 */
        char desc[MAXDESC];    /* description */
        /* Words 41-60 */
        char mimetype[MAXMIME]; /* MIME type */
        /* Words 61-62 */
        char apple[8];        /* APPLE CREATOR/TYPE */
        /* Words 63-78 */
        char ext[64];        /* Popular extensions */
};

/* list of magic entries */
struct mlist
{
        struct magic* magic;        /* array of magic entries */
        uint32_t nmagic;        /* number of entries in array */
        void* map;            /* internal resources used by entry */
        struct mlist* next, * prev;
};

struct level_info
{
        int32_t off;
        int got_match;
};

struct cont
{
        size_t len;
        struct level_info* li;
};

struct magic_set
{
        struct mlist* mlist[MAGIC_SETS];    /* list of regular entries */
        struct cont c;
        struct out
        {
                char* buf;        /* Accumulation buffer */
                size_t blen;        /* Length of buffer */
                char* pbuf;        /* Printable buffer */
        } o;
        uint32_t offset;            /* a copy of m->offset while we */
                                    /* are working on the magic entry */
        uint32_t eoffset;        /* offset from end of file */
        int error;
        int flags;            /* Control magic tests. */
        int event_flags;        /* Note things that happened. */
        const char* file;
        size_t line;            /* current magic line number */
        unsigned short mode; /*mode_t mode;    */        /* copy of current stat
mode */

        /* data for searches */
        struct
        {
                const char* s;        /* start of search in original source */
                size_t s_len;        /* length of search region */
```

```
                    size_t offset;        /* starting offset in source: XXX - should
this be off_t? */
                    size_t rm_len;        /* match length */
                } search;

                /* FIXME: Make the string dynamically allocated so that e.g.
                   strings matched in files can be longer than MAXstring */
                union VALUETYPE ms_value;    /* either number or string */
                uint16_t indir_max;
                uint16_t name_max;
                uint16_t elf_shnum_max;
                uint16_t elf_phnum_max;
                uint16_t elf_notes_max;
                uint16_t regex_max;
                size_t bytes_max;        /* number of bytes to read from file */
                size_t encoding_max;        /* bytes to look for encoding */
        };

        static const int ordinal_close = 4;
        static const int ordinal_error = 8;
        static const int ordinal_file = 9;
        static const int ordinal_load = 13;
        static const int ordinal_open = 15;
        static const int ordinal_version = 18;

        //int magic_version(void)
        typedef int(CALLBACK* MAGIC_VERSION)();
        MAGIC_VERSION magic_version;    // Function pointer

        //const char * magic_file(struct magic_set* ms, const char* inname)
        typedef const char* (CALLBACK* MAGIC_FILE)(struct magic_set* ms, const char*
inname);
        MAGIC_FILE magic_file;    // Function pointer

        //struct magic_set* //magic_open(int flags)
        typedef struct magic_set* (CALLBACK* MAGIC_OPEN)(int flags);
        MAGIC_OPEN magic_open;

        // int magic_load(struct magic_set* ms, const char* magicfile)
        typedef int(CALLBACK* MAGIC_LOAD)(struct magic_set* ms, const char* magicfile);
        MAGIC_LOAD magic_load;

        // void magic_close(struct magic_set* ms)
        typedef void(CALLBACK* MAGIC_CLOSE)(struct magic_set* ms);
        MAGIC_CLOSE magic_close;

        // const char * magic_error(struct magic_set* ms)
        typedef const char* (CALLBACK* MAGIC_ERROR_)(struct magic_set* ms);
        MAGIC_ERROR_ magic_error;

        // takes first ext from extensions string divided with '/'
        std::wstring SplitExtension(const std::wstring& ext_list);

        struct magic_set* ms;
};
```

ExtensionReader.cpp

```
#include "ExtensionReader.hpp"

const std::wstring ExtensionReader::DLL_FILE_NAME = L"libmagic.dll";

ExtensionReader::ExtensionReader()
{
        std::wstring tmp = DLL_FILE_NAME;
```

```
	HINSTANCE hGetProcIDDLL = LoadLibrary(tmp.c_str());

	if (!hGetProcIDDLL)
		throw dll_fail_load_dll(std::string(DLL_FILE_NAME.begin(),
DLL_FILE_NAME.end()), GetLastError());

	magic_version = (MAGIC_VERSION) GetProcAddress(hGetProcIDDLL,
MAKEINTRESOURCEA(ordinal_version));
	magic_file =    (MAGIC_FILE)    GetProcAddress(hGetProcIDDLL,
MAKEINTRESOURCEA(ordinal_file));
	magic_open =    (MAGIC_OPEN)    GetProcAddress(hGetProcIDDLL,
MAKEINTRESOURCEA(ordinal_open));
	magic_load =    (MAGIC_LOAD)    GetProcAddress(hGetProcIDDLL,
MAKEINTRESOURCEA(ordinal_load));
	magic_close =   (MAGIC_CLOSE)   GetProcAddress(hGetProcIDDLL,
MAKEINTRESOURCEA(ordinal_close));
	magic_error =   (MAGIC_ERROR_)  GetProcAddress(hGetProcIDDLL,
MAKEINTRESOURCEA(ordinal_error));

	ms = magic_open(MAGIC_CHECK | MAGIC_EXTENSION);

	if (ms == NULL)
		throw dll_fail_load_magic();

	if (magic_load(ms, NULL) != 0)
	{
		auto error = magic_error(ms);
		magic_close(ms);
		throw dll_fail_load_magic_db(error);
	}
}

ExtensionReader::~ExtensionReader()
{
	magic_close(ms);
}

std::wstring ExtensionReader::GetFileExtension(const std::wstring& path)
{
	auto result_p = magic_file(ms, /*wchar_t* to char* conversion*/
_bstr_t(&path[0]));

	if (!result_p)
		throw dll_fail_run_magic_file();

	std::string sresult = result_p;
	std::wstring wresult(sresult.begin(), sresult.end());
	return SplitExtension(wresult);
}

std::wstring ExtensionReader::SplitExtension(const std::wstring& ext_list)
{
	return ext_list.substr(0, ext_list.find(L'/'));
}
```

CacheReader.hpp

```
#pragma once

#include <map>
#include <vector>
#include <string>
#include <fstream>
#include <memory>
#include <algorithm>
```

```cpp
#include <functional>

#include "../CacheEntry.hpp"
#include "IndexReader.hpp"
#include "../../utils.hpp"
#include "ExtensionReader/ExtensionReader.hpp"


class CacheReader
{
public:
    CacheReader();

    CacheReader(std::unique_ptr<IExtensionReader>&& extension_reader,
                std::unique_ptr<IIndexReader>&& index_reader);

    // Reads  all cache entries sorted by column
    //
    // columns:
    // ID | URL | NAME | TYPE | SIZE
    std::vector<CacheEntry> ReadEntries(const std::wstring& cache_dir_path,
                                        uint32_t            sort_column,
                                        bool                ascending,
                                        uint32_t            entries_offset,
                                        uint32_t            entries_number,
                                        const std::wstring& filter);

    std::vector<CacheEntry> ReadEntries(const std::wstring& cache_dir_path,
                                        const std::vector<CacheAddr>& addresses);

    // Saves file that has address to block 256, 1k and 4k
    // new name is address value - it is always unique
    std::wstring SaveTypeBlock(uint32_t size,
                               const CacheAddr& address,
                               const std::wstring& savedir = L"");


    uint32_t NumberOfEnries(const std::wstring& cache_dir_path,
                            const std::wstring& filter);


    // Saves file from address to temp and returns its path
    std::wstring SaveFileToTemp(
        const std::wstring& cache_dir_path,
        const CacheAddr&    address);
private:

    // Reads cache entry info from data_n block
    // address_index and address_data is data for exporting
    // and should not be seen in GUI
    CacheEntry ReadEntry(const CacheAddr& address);

    // Returns size of block in bytes.
    int GetBlockSize(FileType filetype) const;

    // Saves file from address to temp and returns its path
    std::wstring SaveFileToTemp(uint32_t size, const CacheAddr& address);

    // Saves file that has address to external file
    std::wstring SaveTypeExternal(const CacheAddr& address);

    // Returns substring from last slash to first non-path symbol after it
    std::wstring GetNameFromURL(const std::wstring& url) const;
```

```
    // Just from '.' till end. Renurns empty string if there is no '.'
    std::wstring GetExtensionFromName(const std::wstring& name) const;

    // Saves file to temp and reads extension by it's content
    std::wstring GetExtensionFromFile(uint32_t size, const CacheAddr& addr);

    std::wstring GetNameWithoutExtension(const std::wstring& name) const;

    std::wstring CropUrl(const std::wstring& url) const;

    // Checks if extension contains prohibited symbols
    bool IsExtensionValid(const std::wstring& name) const;

    // opens block file for reading
    void OpenBlockFile(int number);

    // selects needed comparator
    std::function<bool(const CacheEntry&, const CacheEntry&)> GetComparator(uint32_t
column, bool ascending);

    // Max file name without path
    const int MAX_NAME_LENGTH = 64;

    // size of the block file header
    const int FILE_HEADER_SIZE = 0x2000;

    // size of the data entry header
    const int ENTRY_HEADER_SIZE = 0x60;

    // offset from data entry to url size value
    const int ENTRY_URL_SIZE_OFFSET = 0x20;

    // offset to arrays of int[4] sizes of data
    const int ENTRY_DATA_SIZE_OFFSET = 0x28;

    // offset to arrays of int[4] addresses of data
    const int ENTRY_DATA_ADDR_OFFSET = 0x38;

    // offset to the beginning of url
    const int ENTRY_URL_OFFSET = ENTRY_HEADER_SIZE;

    // block file data_n prefix
    const std::wstring PREFIX_BLOCK = L"data_";

    // external file prefix
    const std::wstring PREFIX_FILE = L"f_";

    const std::wstring FORBIDDEN_PATH_SYMBOLS = L"/\\:*?\"<>|";

    // key - file data_n
    // value - path to copy
    std::map<int, std::fstream> data_n_copies_;

    // path to cache dir
    std::wstring cache_dir_path_;

    // Reader for reading file extension if it not defined in url file name
    std::unique_ptr<IExtensionReader> extension_reader_;

    // Reader for index files
    std::unique_ptr<IIndexReader> index_reader_;
};
```

CacheReader.cpp

```cpp
#include "CacheReader.hpp"

using  Utils::CombinePath;
using  Utils::CreateTempCopy;
using  Utils::GetTempDirPath;

CacheReader::CacheReader()
{
	extension_reader_ = std::unique_ptr<ExtensionReader>(new ExtensionReader());
	index_reader_ = std::unique_ptr<IIndexReader>(new IndexReader());
}

CacheReader::CacheReader(std::unique_ptr<IExtensionReader>&& extension_reader,
                                        std::unique_ptr<IIndexReader>&& index_reader) :
	extension_reader_(std::move(extension_reader)),
	index_reader_(std::move(index_reader))
{}


std::vector<CacheEntry> CacheReader::ReadEntries(
	const std::wstring& cache_dir_path,
	uint32_t            sort_column,
	bool                ascending,
	uint32_t            entries_offset,
	uint32_t            entries_number,
	const std::wstring& filter)
{
	cache_dir_path_ = cache_dir_path;

	std::vector<CacheEntry> result;

	index_reader_->Open(cache_dir_path);

	while (true)
	{
		auto address = index_reader_->Next();

		if (address.GetAddress() == 0) break;

		// index file entries always address to data_n file
		// data_n file stores link and possible redirection to other files

		if (!(address.GetFileType() == FileType::BLOCK_256 ||
			address.GetFileType() == FileType::BLOCK_1K ||
			address.GetFileType() == FileType::BLOCK_4K))
		{
			throw index_wrong_address(address);
		}

		try
		{
			auto entry = ReadEntry(address);
			if (entry.ContainsFilter(filter))
			{
				result.push_back(entry);
			}
		}
		// skip empty entries
		catch (const no_url& ex)
		{
			continue;
```

```
                }
                catch (const invalid_data_address& ex)
                {
                        continue;
                }
        }

        // if there is no such elements return empty vector
        if (entries_offset >= result.size())
        {
                return std::vector<CacheEntry>();
        }

        // If there is some elemets, but not all, crop
        if (size_t(entries_offset) + entries_number >= result.size())
        {
                entries_number = result.size() - entries_offset;
        }

        // Select needed column comparator
        auto comparator = GetComparator(sort_column, ascending);

        // Sort by needed column
        std::sort(result.begin(), result.end(), comparator);

        // if boundaries not defined return all entries
        if (entries_offset == -1 &&
                entries_number == -1)
        {
                return result;
        }


        // Return part of sorted vector
        auto from = result.begin() + entries_offset;
        auto to   = from + entries_number;
        return std::vector<CacheEntry>(from, to);
}

std::vector<CacheEntry> CacheReader::ReadEntries(
        const std::wstring& cache_dir_path,
        const std::vector<CacheAddr>& addresses)
{
        cache_dir_path_ = cache_dir_path;

        std::vector<CacheEntry> result;

        for (auto& i : addresses)
        {
                CacheAddr address(i);
                try
                {
                        auto entry = ReadEntry(address);
                        result.push_back(entry);
                }
                catch (const skip_entry_exception& ex)
                {
                        continue;
                }
        }
        return result;
}
```

```
CacheEntry CacheReader::ReadEntry(const CacheAddr& address)
{
	auto blockfile_num = address.GetDataFileNumber();

	OpenBlockFile(blockfile_num);

	// jump to entry
	int block_size = GetBlockSize(address.GetFileType());
	int offset = FILE_HEADER_SIZE + block_size * address.GetStartBlockOffset();
	data_n_copies_[blockfile_num].seekg(offset, data_n_copies_[blockfile_num].beg);


	// copy buffer
	size_t buffer_size = static_cast<long long>(block_size) *
address.GetNumberOfBlocks();
	std::vector<BYTE> entry_buffer(buffer_size);
	data_n_copies_[blockfile_num].read(reinterpret_cast<char*>(entry_buffer.data()),
buffer_size);


	// read url size
	uint32_t url_size;
	memcpy(&url_size, &entry_buffer.data()[ENTRY_URL_SIZE_OFFSET], sizeof(uint32_t));

	// check size
	if (url_size == 0 ||
		url_size > buffer_size - ENTRY_URL_OFFSET)
		throw no_url();

	// read url
	auto from = entry_buffer.begin() + ENTRY_URL_OFFSET;
	auto to = from + url_size;
	std::wstring url(from, to);

	if (url.empty())
		throw no_url();

	url = CropUrl(url);

	// Remove semicolon from url
	std::replace(url.begin(), url.end(), ';', ':');

	// read data addresses and sizes
	const int addr_entries = 4;
	uint32_t sizes[addr_entries];
	uint32_t temp[addr_entries];
	CacheAddr addresses[addr_entries];

	memcpy(&sizes, &entry_buffer.data()[ENTRY_DATA_SIZE_OFFSET], sizeof(uint32_t) *
addr_entries);
	memcpy(&temp, &entry_buffer.data()[ENTRY_DATA_ADDR_OFFSET], sizeof(uint32_t) *
addr_entries);
	for (int i = 0; i < addr_entries; i++)
		addresses[i] = CacheAddr(temp[i]);


	// check if second value is valid
	if (!addresses[1].IsValid())
		throw invalid_data_address();

	if (addresses[1].GetFileType() == FileType::EXTERNAL)
	{
		auto path = CombinePath(cache_dir_path_, addresses[1].GetFileName());
		if (!Utils::IsFileExists(path))
```

```
                {
                        //ignore non existant files
                        throw no_external_file();
                }
        }

        auto name = GetNameFromURL(url);
        auto ext  = GetExtensionFromName(name);
        if (ext.empty())
        {
                ext = GetExtensionFromFile(sizes[1], addresses[1]);
        }
        else
        {
                name = GetNameWithoutExtension(name);
        }

        return CacheEntry(
                address.GetAddress(),
                addresses[1].GetAddress(),
                url,
                name,
                ext,
                sizes[1]);
}


int CacheReader::GetBlockSize(FileType filetype) const
{
        switch (filetype)
        {
        case FileType::BLOCK_256:
                return 256;

        case FileType::BLOCK_1K:
                return 1024;

        case FileType::BLOCK_4K:
                return 4096;

        case FileType::EXTERNAL:
        case FileType::RANKINGS:
        case FileType::BLOCK_FILES:
        case FileType::BLOCK_ENTRIES:
        case FileType::BLOCK_EVICTED:
        default:
                throw file_not_block_type();
        }
}

std::wstring CacheReader::SaveFileToTemp(uint32_t size, const CacheAddr& address)
{
        switch (address.GetFileType())
        {
        case FileType::EXTERNAL:
                //return SaveTypeExternal(address);
                return CombinePath(cache_dir_path_, address.GetFileName());

        case FileType::BLOCK_256:
        case FileType::BLOCK_1K:
        case FileType::BLOCK_4K:
                return SaveTypeBlock(size, address);

        default:
```

```
                throw addr_wrong_type();
        }


        return std::wstring();
}

std::wstring CacheReader::SaveTypeExternal(const CacheAddr& address)
{
        auto path_from = CombinePath(cache_dir_path_, address.GetFileName());
        return CreateTempCopy(path_from);
}

std::wstring CacheReader::SaveTypeBlock(uint32_t            size,
                                                                const CacheAddr&
address,
                                                                const std::wstring&
name)
{
        auto blockfile_num = address.GetDataFileNumber();

        OpenBlockFile(blockfile_num);

        // jump to entry
        int block_size = GetBlockSize(address.GetFileType());
        int offset = FILE_HEADER_SIZE + block_size * address.GetStartBlockOffset();
        data_n_copies_[blockfile_num].seekg(offset, data_n_copies_[blockfile_num].beg);

        // copy block to new file
        std::vector<BYTE> entry_buffer(size);
        data_n_copies_[blockfile_num].read(reinterpret_cast<char*>(entry_buffer.data()),
size);


        std::wstring path = name;
        if (name == L"")
        {
                auto dir = CombinePath(GetTempDirPath(), GLOBALS::APP_NAME);
                path = CombinePath(dir, std::to_wstring(address.GetAddress()));
        }

        std::fstream temp_file(path, std::ios_base::out | std::ios_base::binary);

        temp_file.write(reinterpret_cast<char*>(entry_buffer.data()), size);

        return path;
}


uint32_t CacheReader::NumberOfEnries(const std::wstring& cache_dir_path, const
std::wstring& filter)
{
        cache_dir_path_ = cache_dir_path;
        index_reader_->Open(cache_dir_path);

        int count = 0;

        while (true)
        {
                auto address = index_reader_->Next();

                if (address.GetAddress() == 0) break;

                // index file entries always address to data_n file
```

```
                // data_n file stores link and possible redirection to other files

                if (!(address.GetFileType() == FileType::BLOCK_256 ||
                        address.GetFileType() == FileType::BLOCK_1K ||
                        address.GetFileType() == FileType::BLOCK_4K))
                {
                        throw index_wrong_address(address);
                }

                try
                {
                        auto entry = ReadEntry(address);

                        auto name = entry.file_name + entry.extension;

                        if (name.find(filter)      != std::string::npos ||
                                entry.url.find(filter) != std::string::npos)
                        {
                                count++;
                        }
                }
                catch (const skip_entry_exception& ex)
                {
                        continue;
                }
        }
        return count;
}

std::wstring CacheReader::SaveFileToTemp(
        const std::wstring& cache_dir_path,
        const CacheAddr&    address)
{
        cache_dir_path_ = cache_dir_path;
        auto entry = ReadEntry(address);
        return SaveFileToTemp(entry.size, entry.address_data);
}

std::wstring CacheReader::GetNameFromURL(const std::wstring& url) const
{
        auto pos_slash = url.find_last_of(L'/') + 1;
        auto sub = url.substr(pos_slash, url.size() - 1);


        auto pos_forbidden = sub.find_first_of(FORBIDDEN_PATH_SYMBOLS);
        if (pos_forbidden >= sub.size())
                pos_forbidden = sub.size();

        auto name = sub.substr(0, pos_forbidden);

        if (name.size() > MAX_NAME_LENGTH)
                name = name.substr(name.size() - MAX_NAME_LENGTH, name.size());

        return name;
}

std::wstring CacheReader::GetExtensionFromName(const std::wstring& name) const
{
        auto pos_dot = name.find_last_of(L'.');
        if (pos_dot == 0 || pos_dot == std::wstring::npos)
                return std::wstring();

        return name.substr(pos_dot, name.size());
}
```

```
std::wstring CacheReader::GetExtensionFromFile(uint32_t size, const CacheAddr& addr)
{
	auto path = SaveFileToTemp(size, addr);
	if (!Utils::IsFileExists(path))
	{
		throw no_url();
	}

	auto ext = extension_reader_->GetFileExtension(path);

	if (IsExtensionValid(ext))
	{
		return L"." + ext;
	}
	else
	{
		return std::wstring();
	}
}

std::wstring CacheReader::GetNameWithoutExtension(const std::wstring& name) const
{
	auto pos_dot = name.find_last_of(L'.');
	if (pos_dot == 0 || pos_dot == std::wstring::npos)
		return std::wstring();

	return name.substr(0, pos_dot);
}

std::wstring CacheReader::CropUrl(const std::wstring& url) const
{
	auto beg = url.find_last_of(L' ') + 1;
	auto end = url.size();

	if (beg == std::wstring::npos ||
		end == std::wstring::npos)
	{
		return url;
	}

	return url.substr(beg, end);
}

bool CacheReader::IsExtensionValid(const std::wstring& name) const
{
	return name.find_first_of(FORBIDDEN_PATH_SYMBOLS) == std::wstring::npos;
}

void CacheReader::OpenBlockFile(int number)
{
	// Create copy if not exists
	auto blockfile_name = PREFIX_BLOCK + std::to_wstring(number);
	if (data_n_copies_.find(number) == data_n_copies_.end())
	{
		auto path = CreateTempCopy(CombinePath(cache_dir_path_, blockfile_name));
		data_n_copies_[number].open(path.c_str(), std::ios_base::in |
std::ios_base::binary);
	}
}

std::function<bool(const CacheEntry&, const CacheEntry&)>
CacheReader::GetComparator(uint32_t column, bool ascending)
{
```

```
	auto comparator = CacheEntry::CompareBy_0;
	if (ascending)
	{
		comparator =
			column == 0 ? CacheEntry::CompareBy_0 :
			column == 1 ? CacheEntry::CompareBy_1 :
			column == 2 ? CacheEntry::CompareBy_2 :
			column == 3 ? CacheEntry::CompareBy_3 :
			column == 4 ? CacheEntry::CompareBy_4 :
			CacheEntry::CompareBy_0;
	}
	else
	{
		comparator =
			column == 0 ? CacheEntry::CompareBy_0_descending :
			column == 1 ? CacheEntry::CompareBy_1_descending :
			column == 2 ? CacheEntry::CompareBy_2_descending :
			column == 3 ? CacheEntry::CompareBy_3_descending :
			column == 4 ? CacheEntry::CompareBy_4_descending :
			CacheEntry::CompareBy_0_descending;
	}
	return std::function<bool(const CacheEntry&, const CacheEntry&)>(comparator);
}
```

Imd5.hpp

```
#pragma once

#include <string>

struct IMD5
{
	virtual ~IMD5() {}
	virtual std::string HashFile(const std::wstring& path) = 0;
};
```

md5.hpp

```
#pragma once

#include "Imd5.hpp"

#include <stdio.h>
#include <windows.h>
#include <Wincrypt.h>

#include <string>
#include <sstream>

#include "../../exceptions.hpp"


#define BUFSIZE 1024
#define MD5LEN  16



class MD5 : public IMD5
{
public:
	MD5();
	~MD5();
	std::string HashFile(const std::wstring& path) override;
};
```

```
#include "md5.hpp"

MD5::MD5() { }

MD5::~MD5() { }

std::string MD5::HashFile(const std::wstring& path)
{
    DWORD dwStatus = 0;
    BOOL bResult = FALSE;
    HCRYPTPROV hProv = 0;
    HCRYPTHASH hHash = 0;
    HANDLE hFile = NULL;
    BYTE rgbFile[BUFSIZE];
    DWORD cbRead = 0;
    BYTE rgbHash[MD5LEN];
    DWORD cbHash = 0;
    CHAR rgbDigits[] = "0123456789abcdef";
    LPCWSTR filename = path.c_str();
    // Logic to check usage goes here.

    hFile = CreateFile(filename,
                       GENERIC_READ,
                       FILE_SHARE_READ,
                       NULL,
                       OPEN_EXISTING,
                       FILE_FLAG_SEQUENTIAL_SCAN,
                       NULL);

    if (INVALID_HANDLE_VALUE == hFile)
    {
        dwStatus = GetLastError();
        throw file_not_found_error("", dwStatus);
    }

    // Get handle to the crypto provider
    if (!CryptAcquireContext(&hProv,
                             NULL,
                             NULL,
                             PROV_RSA_FULL,
                             CRYPT_VERIFYCONTEXT))
    {
        dwStatus = GetLastError();
        CloseHandle(hFile);
        throw md5_failed("CryptAcquireContext failed: ", dwStatus);
    }

    if (!CryptCreateHash(hProv, CALG_MD5, 0, 0, &hHash))
    {
        dwStatus = GetLastError();
        CloseHandle(hFile);
        CryptReleaseContext(hProv, 0);

        throw md5_failed("CryptCreateHash failed: ", dwStatus);
    }

    while (bResult = ReadFile(hFile, rgbFile, BUFSIZE,
                              &cbRead, NULL))
    {
        if (0 == cbRead)
        {
            break;
```

```
        }

        if (!CryptHashData(hHash, rgbFile, cbRead, 0))
        {
            dwStatus = GetLastError();
            CryptReleaseContext(hProv, 0);
            CryptDestroyHash(hHash);
            CloseHandle(hFile);

            throw md5_failed("CryptHashData failed: ", dwStatus);
        }
    }

    if (!bResult)
    {
        dwStatus = GetLastError();
        CryptReleaseContext(hProv, 0);
        CryptDestroyHash(hHash);
        CloseHandle(hFile);
        throw md5_failed("ReadFile failed: ", dwStatus);
    }

    cbHash = MD5LEN;

    std::stringstream result_hash;

    if (CryptGetHashParam(hHash, HP_HASHVAL, rgbHash, &cbHash, 0))
    {
        result_hash << std::hex;
        for (DWORD i = 0; i < cbHash; i++)
        {
            result_hash << rgbDigits[rgbHash[i] >> 4] << rgbDigits[rgbHash[i] & 0xf];
        }
    }
    else
    {
        dwStatus = GetLastError();
        CryptDestroyHash(hHash);
        CryptReleaseContext(hProv, 0);
        CloseHandle(hFile);

        throw md5_failed("CryptGetHashParam failed: ", dwStatus);
    }

    CryptDestroyHash(hHash);
    CryptReleaseContext(hProv, 0);
    CloseHandle(hFile);

    return result_hash.str();
}
```

CacheExporter.hpp

```
#pragma once

#include <vector>
#include <memory>

#include "../CacheAddr.hpp"
#include "../md5/md5.hpp"
#include "../CacheReader/CacheReader.hpp"

class CacheExporter
{
public:
```

```
        CacheExporter();

        CacheExporter(std::unique_ptr<IMD5>&& md5_hasher);

        void Export(const std::wstring& cache_dir_path,
                            uint32_t            sort_column,
                            bool                ascending,
                            uint32_t            entries_offset,
                            uint32_t            entries_number,
                            const std::wstring& filter,
                            const std::wstring& export_dir,
                            bool                save_md5);

        std::wstring ExportSingleFile(
               const std::wstring& cache_dir_path,
               uint32_t            cache_addr);

private:

        // Saves entry to file on disk to given dir
        std::wstring SaveEntry(const CacheEntry& cache_entry,
                               const std::wstring& cache_dir);

        // Saves filetype block file on disk to path
        // Wrap for CacheReader SaveTypeBlock
        std::wstring SaveTypeBlock(uint32_t size,
                                   const CacheAddr& address,
                                   const std::wstring& name);

        // Directory for exporting files
        std::wstring export_dir_;

        // hash get hash of exported file
        std::unique_ptr<IMD5> md5_hasher_;
};
```

CacheExporter.cpp

```
#include "CacheExporter.hpp"

CacheExporter::CacheExporter()
{
        md5_hasher_ = std::unique_ptr<IMD5>(new MD5());
}

CacheExporter::CacheExporter(std::unique_ptr<IMD5> && md5_hasher) :
        md5_hasher_(std::move(md5_hasher)) { }

void CacheExporter::Export(const std::wstring& cache_dir_path,
                           uint32_t            sort_column,
                           bool                ascending,
                           uint32_t            entries_offset,
                           uint32_t            entries_number,
                           const std::wstring& filter,
                           const std::wstring& export_dir,
                           bool                save_md5)
{
        export_dir_ = export_dir;

        std::vector<CacheEntry> entries;
        {
               CacheReader cache_reader;
               entries = cache_reader.ReadEntries(cache_dir_path,
                                                  sort_column,
```

```
                                                                    ascending,
                                                                    entries_offset,
                                                                    entries_number,
                                                                    filter);
        }

        for (auto& i : entries)
        {
                auto path = SaveEntry(i, export_dir);
                if (save_md5 && Utils::IsFileExists(path))
                {
                        auto hash = md5_hasher_->HashFile(path);
                        std::ofstream file(path + L".md5.txt");
                        file << hash;
                }
        }
}

std::wstring CacheExporter::ExportSingleFile(
        const std::wstring& cache_dir_path,
        uint32_t            cache_addr)
{
        CacheReader reader;
        return reader.SaveFileToTemp(cache_dir_path, cache_addr);
}


std::wstring CacheExporter::SaveEntry(const CacheEntry&   cache_entry,
                                                                const std::wstring&
export_dir)
{
        CacheAddr address(cache_entry.address_data);

        auto name = cache_entry.file_name;
        if (name == std::wstring())
                name = std::to_wstring(cache_entry.address_index);

        // Create name with (2) if it already exist in export dir
        auto unique_name = Utils::CreateNoDupPath(export_dir,
                                                                                name,
                                                        cache_entry.extension);

        switch (address.GetFileType())
        {
        case FileType::EXTERNAL:
                // Just copy from cache dir
                Utils::FileCopy(Utils::CombinePath(export_dir, address.GetFileName()),
                                                unique_name);
                return unique_name;

        case FileType::BLOCK_256:
        case FileType::BLOCK_1K:
        case FileType::BLOCK_4K:
                // extract to unique_name path
                return SaveTypeBlock(cache_entry.size, address, unique_name);
        default:
                throw addr_wrong_type();
        }
        return unique_name;
}

std::wstring CacheExporter::SaveTypeBlock(uint32_t size,
                                                const CacheAddr& address,
                                                const std::wstring& name)
```

```
{
	CacheReader reader;
	return reader.SaveTypeBlock(size, address, name);
}
```

cachework.hpp

```
#pragma once

#include <iostream>

#include "CacheReader/CacheReader.hpp"

#ifdef CACHEWORK_EXPORTS
#define CACHEWORK_API __declspec(dllimport)
#else
#define CACHEWORK_API __declspec(dllexport)
#endif

enum CacheWorkStatus
{
	SUCC = 0,
	UNHANDLED_EXSEPTION = -1,
	FILE_NOT_FOUND = -2,

	END_OF_FILE = 1,
	TOO_SMALL_BUFFER = 4,
};


extern "C"
CACHEWORK_API int GetPageFromCache(
	const wchar_t* cacheDirPath,
	wchar_t*       outBuffer,
	int            outSize,
	int                    startLine,
	int            numbOfLines,
	int            modeOfSorting,
	int            columnBySorting,
	const wchar_t* searchWord);


extern "C"
CACHEWORK_API int NumbOfLinesCache(
	const wchar_t* cacheDirPath,
	const wchar_t*    searchWord);

extern "C"
CACHEWORK_API void ExportCache(
	const wchar_t*      cache_dir_path,
	int                 sort_column,
	int                 ascending,
	int                 entries_offset,
	int                 entries_number,
	const wchar_t*      filter,
	const wchar_t*      export_dir,
	int                 save_md5);


extern "C"
CACHEWORK_API void SaveFileToTemp(
	const wchar_t* cache_dir_path,
	int            cache_addr,
	wchar_t*       out_path,
	int            out_path_size
```

```
);
```

cachework.cpp

```
#include "cachework.hpp"
#include "CacheExporter/CacheExporter.hpp"
#include <algorithm>
#include <sstream>
#include <iterator>

CACHEWORK_API int GetPageFromCache(
	const wchar_t* cacheDirPath,
	wchar_t*       outBuffer,
	int            outSize,
	int                    startLine,
	int            numbOfLines,
	int            modeOfSorting,
	int            columnBySorting,
	const wchar_t* searchWord)
{
	std::vector<CacheEntry> entries;
	try
	{
		CacheReader reader;

		/*auto str = std::string(cacheDirPath);
		auto wstr = Utils::s2ws(str);*/
		entries = reader.ReadEntries(
			cacheDirPath,
			columnBySorting,
			modeOfSorting,
			startLine,
			numbOfLines,
			searchWord);
	}
	catch (const file_not_found_error& ex)
	{
		return CacheWorkStatus::FILE_NOT_FOUND;
	}
	catch (const std::runtime_error& ex)
	{
		return CacheWorkStatus::UNHANDLED_EXSEPTION;
	}
	catch (const std::exception&)
	{
		return CacheWorkStatus::UNHANDLED_EXSEPTION;
	}
	catch (...)
	{
		return CacheWorkStatus::UNHANDLED_EXSEPTION;
	}

	std::wostringstream oresult;
	const std::wstring delimeter = L"; ";

	for (auto& i : entries)
	{
		std::vector<std::wstring> elems
		{
			std::to_wstring(i.address_index),
			i.url,
			i.file_name,
			i.extension,
			std::to_wstring(i.size),
```

```
            };

            std::copy(elems.begin(), elems.end(), std::ostream_iterator<std::wstring,
wchar_t>(oresult, delimeter.c_str()));
            oresult << "\n";
        }

        auto result = oresult.str();

        if (result.size() + 1 > outSize)
            return CacheWorkStatus::TOO_SMALL_BUFFER;

        wcscpy_s(outBuffer, outSize, result.c_str());
        outBuffer[result.size()] = '\0';

        if (entries.size() != numbOfLines)
            return CacheWorkStatus::END_OF_FILE;

        return CacheWorkStatus::SUCC;
}

CACHEWORK_API int NumbOfLinesCache(
        const wchar_t* cacheDirPath,
        const wchar_t* searchWord)
{
        CacheReader reader;
        return reader.NumberOfEnries(cacheDirPath, searchWord);
}

CACHEWORK_API void ExportCache(
        const wchar_t*      cache_dir_path,
        int                 sort_column,
        int                 ascending,
        int                 entries_offset,
        int                 entries_number,
        const wchar_t*      filter,
        const wchar_t*      export_dir,
        int                 save_md5)
{
        CacheExporter exporter;
        exporter.Export(
            cache_dir_path,
            sort_column,
            ascending,
            entries_offset,
            entries_number,
            filter,
            export_dir,
            save_md5);
}

CACHEWORK_API void SaveFileToTemp(
        const wchar_t* cache_dir_path,
        int            cache_addr,
        wchar_t*       out_path,
        int            out_path_size)
{
        CacheExporter exporter;
        auto path = exporter.ExportSingleFile(cache_dir_path, cache_addr);
        wcscpy_s(out_path, out_path_size, path.c_str());
        out_path[path.size()] = '\0';
}
```

filework.h

```
#pragma once

#include <fstream>
#include <memory>
#include <string>
#include <cstring>

#include <shlobj_core.h>
#include <wchar.h>

#include "framework.h"

#ifdef FILEWORK_EXPORTS
#define FILEWORK_API __declspec(dllimport)
#else
#define FILEWORK_API __declspec(dllexport)
#endif

//Copies an existing file to a new file.
//Result: If the function succeeds, the return value is nonzero. If the function fails,
the return value is zero.
extern "C"
FILEWORK_API int FileCopy(const char* oldFilename,     //The name of existing file.
       const char* newFilename,                                    //The name of new
file.
       bool isFileInAppDataLocal);                                 //If it is true,
function add path to dir 'AppData/Local' to the beginning of the oldFilename.

//Deletes an existing file.
//Result: If the function succeeds, the return value is nonzero. If the function fails,
the return value is zero.
extern "C"
FILEWORK_API int FileDelete(const char* filename,      //The name of the file to be
deleted.
       bool isFileInAppDataLocal);                                 //If it is true,
function add path to dir 'AppData/Local' to the beginning of the filename.

//Write text to the file.
//Result: If the function succeeds, the return value is true. If the function fails, the
return value is false.
extern "C"
FILEWORK_API bool ExportTo(const char* filename,//The name of the file into which we
write text. If this file is non existing, it will be created.
       const char* text,                                                   //The text
for export to the file.
       bool isClear);                                                              //If
it is true, the file will be cleared before writing.

//Get the path to the 'AppData/Local' directory(including them).
//Result: return Unicode string that is the path to the 'AppData/Local', else returns
empty string.
extern "C"
FILEWORK_API wchar_t* PathToAppDataLocal();
```

filework.cpp

```
#include "filework.h"

FILEWORK_API int FileCopy(const char* oldFilename,
       const char* newFilename,
       bool isFileInAppDataLocal)
{
```

```
	std::wstring wpath;
	std::wstring wfilename;
	std::string str;

	if (isFileInAppDataLocal)
		wpath = std::wstring(PathToAppDataLocal());

	str = std::string(oldFilename);
	wpath.append(str.begin(), str.end());

	str = std::string(newFilename);
	wfilename.append(str.begin(), str.end());

	return CopyFile(wpath.c_str(), wfilename.c_str(), FALSE);
}

FILEWORK_API int FileDelete(const char* filename,
	bool isFileInAppDataLocal)
{
	std::string flnm = std::string(filename);
	std::wstring wpath;

	if (isFileInAppDataLocal)
		wpath = std::wstring(PathToAppDataLocal());

	wpath.append(flnm.begin(), flnm.end());

	return DeleteFile(wpath.c_str());
}

FILEWORK_API bool ExportTo(const char* filename,
	const char* text,
	bool isClear)
{
	auto customDeleter = [](std::fstream* ptr) {
		ptr->close();
		delete ptr;
	};

	std::unique_ptr<std::fstream, decltype(customDeleter)> fout(new std::fstream,
customDeleter);
	if (isClear)
		fout->open(filename, std::ios::out);
	else
		fout->open(filename, std::ios::app);

	if (!fout->is_open())
		return false;

	std::string tmp, str = std::string(text);
	size_t k, nw, cnt = count(str.begin(), str.end(), '\n');

	for (size_t i = 0; i <= cnt && str.size(); i++)
	{
		nw = str.find('\n');
		if (nw == std::string::npos)
			k = 0, nw = str.size();
		else
			k = 1;

		tmp = str.substr(0, nw);
		(*fout) << tmp << '\n';
		str.erase(str.begin(), str.begin() + nw + k);
	}
```

```
	return true;
}

FILEWORK_API wchar_t* PathToAppDataLocal()
{
	int bufsize;
	wchar_t* buffer;
	wchar_t wpath[MAX_PATH];

	SHGetFolderPath(NULL, CSIDL_LOCAL_APPDATA, NULL, 0, (LPWSTR)wpath);

	if (SHGetFolderPath(NULL, CSIDL_LOCAL_APPDATA, NULL, 0, (LPWSTR)wpath) != S_OK)
	{
		bufsize = 2;
		buffer = (wchar_t*)CoTaskMemAlloc(bufsize * sizeof(wchar_t));
		buffer[0] = '\0';
	}
	else
	{
		bufsize = wcslen(wpath) + 1;
		buffer = (wchar_t*)CoTaskMemAlloc(bufsize * sizeof(wchar_t));
		wcscpy_s(buffer, bufsize, wpath);
	}

	return buffer;
}
```

decryptor.h

```
#pragma once

#include <fstream>
#include <memory>
#include <string>
#include <cstring>

#include <shlobj_core.h>

//for exceptions
#include "cryptopp\cryptlib.h"

//for base64 encoder and decoder
#include "cryptopp\base64.h"

//converter input into cipher
#include "cryptopp\filters.h"

//add AES gcm
#include "cryptopp\aes.h"
#include "cryptopp\gcm.h"

#include "framework.h"

#pragma comment(lib, "crypt32")

#ifdef DECRYPTOR_EXPORTS
#define DECRYPTOR_API __declspec(dllimport)
#else
#define DECRYPTOR_API __declspec(dllexport)
#endif

#define TAG_SIZE 16

//Decrypth string of bytes with DPAPI function CryptUnprotectData.
```

```
//Result: If the function succeeds, the return nonempty string. If the function fails,
the return empty string.
extern "C"
DECRYPTOR_API unsigned char* DecryptByDPAPI(const unsigned char* data);     //Encrypted
string

//Get and decrypth secret key with DPAPI function CryptUnprotectData.
//Result: If the function succeeds, the return nonempty string. If the function fails,
the return empty string.
extern "C"
DECRYPTOR_API unsigned char* GetMasterKey(const char* jsonName);            //The name of
JSON file

//Decrypth string of bytes with AES-256-GCM alghoritm of decrypting.
//Result: If the function succeeds, the return nonempty string. If the function fails,
the return empty string.
extern "C"
DECRYPTOR_API char* DecryptByAes256Gcm(unsigned char* bytebuff,
	//Encrypted string
	unsigned char* bytekey);
			//Decrypted secret key
```

decryptor.cpp

```
#include "decryptor.h"

DECRYPTOR_API unsigned char* DecryptByDPAPI(const unsigned char* data)
{
	DATA_BLOB in;
	DATA_BLOB out;

	BYTE trick[1024];
	for (int i = 0; i < 1024; i++)
		trick[i] = data[i];

	in.pbData = trick;
	in.cbData = 1024;
	std::vector<BYTE> decrypted;

	if (CryptUnprotectData(&in, NULL, NULL, NULL, NULL, 0, &out))
		for (size_t i = 0; i < out.cbData; i++)
			decrypted.push_back(out.pbData[i]);
	LocalFree(out.pbData);

	unsigned char* result = (unsigned char*)CoTaskMemAlloc((decrypted.size() + 1) *
sizeof(unsigned char));
	for (size_t i = 0; i < decrypted.size(); i++)
		result[i] = decrypted[i];
	result[decrypted.size()] = '\0';

	return result;
}

DECRYPTOR_API unsigned char* GetMasterKey(const char* jsonName)
{
	auto customDeleter = [](std::fstream* ptr) {
		ptr->close();
		delete ptr;
	};

	std::unique_ptr<std::fstream, decltype(customDeleter)> json(new std::fstream,
customDeleter);
	std::string str, key;
	int pos;
```

```
	//Take master key from \"Local State.json\" file
	json->open(jsonName, std::ios::in);
	while (getline(*json, str)) {
		pos = str.find("\"encrypted_key\":\"");
		if (pos != std::string::npos) {
			key = str.substr(pos + 17, 356);
			break;
		}
	}

	//Decode master key by base64decode
	std::string decoded_key;
	CryptoPP::Base64Decoder decoder;
	decoder.Attach(new CryptoPP::StringSink(decoded_key));
	decoder.Put(reinterpret_cast<const unsigned char*>(key.c_str()), key.length());
	decoder.MessageEnd();

	//Delete DPAPI start of decrypted master key
	std::string bytes_of_decoded_key;
	for (size_t i = 5; i < decoded_key.size(); i++)
		bytes_of_decoded_key.push_back(decoded_key[i]);

	return DecryptByDPAPI((unsigned char*)bytes_of_decoded_key.c_str());
}

DECRYPTOR_API char* DecryptByAes256Gcm(unsigned char* bytebuff,
	unsigned char* bytekey)
{
	std::string buff, decrypted, encrypted;
	std::vector<BYTE> key;

	buff = std::string((char*)bytebuff);

	try {
		for (size_t i = 0; i < 32; i++)
			key.push_back(bytekey[i]);

		//Take initalization vector
		std::vector<BYTE> iv(buff.begin() + 3, buff.begin() + 15);

		//Take encrypted text with tag without initalization vector
		for (size_t i = 15; i < buff.size(); i++)
			encrypted.push_back(buff[i]);

		//Decrypt by AES-256-GCM
		CryptoPP::GCM<CryptoPP::AES>::Decryption d;
		d.SetKeyWithIV(&key[0], key.size(), &iv[0], iv.size());
		CryptoPP::StringSource s(encrypted,
			true,
			new CryptoPP::AuthenticatedDecryptionFilter(d,
				new CryptoPP::StringSink(decrypted),
				CryptoPP::HashVerificationFilter::HASH_AT_END,
				TAG_SIZE) //streamtransformationfilter
		);//stringsource

		char* result;
		int resultSize = decrypted.size() + 1;
		result = (char*)CoTaskMemAlloc(resultSize * sizeof(char));
		strcpy_s(result, resultSize, decrypted.c_str());

		return result;
	}
	catch (...) {
```

```
	}

	return (char*)"-";
}
```

WrapperDB.h

```
#pragma once

#include <string>
#include "sqlite/sqlite3.h"

class WrapperDB
{
private:
	sqlite3* db;                                                    //Each open SQLite
database is represented by a pointer to an instance of the opaque structure named
"sqlite3".
	sqlite3_stmt* stmt;                                             //SQL statement that
has been compiled into binary form and is ready to be evaluated.

public:
	//Empty ctor.
	WrapperDB();

	//Open database with name dbname.
	WrapperDB(const std::string& dbname);           //The name of database

	//Open database with name dbname and initalize SQL statement by SQL inquery.
	WrapperDB(const std::string& dbname,            //The name of database
		const std::string& sql);                        //SQL inquery

	//Clear SQL statement and close database.
	~WrapperDB();

	//Get SQL statement.
	sqlite3_stmt* GetStmt();

	//Close opened database and open new database with name dbname.
	//Result: If the function succeeds, the return value is nonzero. If the function
fails, the return value is zero.
	int SetDb(const std::string& dbname);           //The name of database

	//
	//Result: If the function succeeds, the return value is nonzero. If the function
fails, the return value is zero.
	int SetStmt(const std::string& sql);            //SQL inquery

	//Close opened database and clear SQL statement.
	//Result: If the function succeeds, the return value is nonzero. If the function
fails, the return value is zero.
	int Clear();

	//Reset SQL statement.
	//Result: If the function succeeds, the return value is nonzero. If the function
fails, the return value is zero.
	int StmtPrepare(const std::string& sql); //SQL inquery

	//Go to next line of database.
	//Result: If the function succeeds, the return value is nonzero. If the function
fails, the return value is zero.
	//If the returned value is -1, it is end of database. If the returned value is 1,
it is NOT end of database.
	int StmtNextLine();
};
```

WrapperDB.cpp

```cpp
#include "WrapperDB.h"

WrapperDB::WrapperDB() {}

WrapperDB::WrapperDB(const std::string& dbname)
{
	sqlite3_open(dbname.c_str(), &db);
}

WrapperDB::WrapperDB(const std::string& dbname,
	const std::string& sql)
{
	sqlite3_open(dbname.c_str(), &db);
	sqlite3_prepare_v2(db, sql.c_str(), -1, &stmt, NULL);
}

WrapperDB::~WrapperDB()
{
	sqlite3_finalize(stmt);
	sqlite3_close(db);
}

sqlite3_stmt* WrapperDB::GetStmt()
{
	return stmt;
}

int WrapperDB::SetDb(const std::string& dbname)
{
	Clear();
	sqlite3_open(dbname.c_str(), &db);
	return db ? 1 : 0;
}

int WrapperDB::SetStmt(const std::string& sql)
{
	sqlite3_prepare_v2(db, sql.c_str(), -1, &stmt, NULL);
	return stmt ? 1 : 0;
}

int WrapperDB::Clear()
{
	sqlite3_finalize(stmt);
	return sqlite3_close(db) == SQLITE_BUSY ? 0 : 1;
}

int WrapperDB::StmtPrepare(const std::string& sql)
{
	sqlite3_prepare_v2(db, sql.c_str(), -1, &stmt, NULL);
	return stmt ? 1 : 0;
}

int WrapperDB::StmtNextLine()
{
	if (!stmt || !db)
		return 0;
	return sqlite3_step(stmt) == SQLITE_DONE ? -1 : 1;
}
```

```
#pragma once

#include <algorithm>
#include <string>
#include <cstring>

#include <wchar.h>

#include "framework.h"
#include "WrapperDB.h"

#ifdef DBWORK_EXPORTS
#define DBWORK_API __declspec(dllimport)
#else
#define DBWORK_API __declspec(dllexport)
#endif

#include "decryptor.h"

//Modes of sorting
#define STANDART_MODE 0
#define MODE_FROM_SMALLEST_TO_BIGGEST 1
#define MODE_FROM_BIGGEST_TO_SMALLEST 2

//Results of NumbOfLines
#define SUCCESS 0
#define CANT_OPEN_DB -1
#define CANT_CLOSE_DB -2
#define DB_IS_EMPTY_OR_IS_NOT_FIND -3
#define SQL_BUFFER_IS_TOO_SMALL -4

//Results of MakeSQL
#define SQL_CREATED 0
#define UNIDENTIFIED_COLUMN -5
#define UNIDENTIFIED_MODE_OF_SORTING -6
#define SQL_OUT_BUFFER_IS_TOO_SMALL -7

//Results of GetPages
#define LINES_IN_DATABASE_IS_OVER 0
#define LINES_IN_DATABASE_IS_NOT_OVER -8
#define CANNOT_OPEN_DATABASE -9
#define CANNOT_CLOSE_DATABASE -10
#define OUT_BUFFER_IS_TOO_SMALL -11
#define DB_OR_IS_EMPTY_OR_IS_NOT_FIND -12

//Size of password_value
#define SIZE_OF_PASSWORD_VALUE 14

//Return SQL inquery.
//Result: If the function succeeds, the return value is zero. If the function fails, the
return value is nonzero.
//Errors:     1) UNIDENTIFIED_COLUMN is value -5. If you set index of column that is non
existing.
//                  2) UNIDENTIFIED_MODE_OF_SORTING is value -6. If you set index of
mode of sorting that is non existing.
//                  3) SQL_OUT_BUFFER_IS_TOO_SMALL is value -7. If you set out buffer
that has size smaller, than created sql inquery.
extern "C"
DBWORK_API int MakeSQL(const char* tablename,                //The name of table in
database
       const char* listOfColumns,                                          //The list of
column`s names in selected table
```

```
	char* out,
	//The out buffer
	int outSize,                                                    //The
size of the out buffer
	int modeOfSorting,                                              //The
sorting`s mode
	int columnBySorting,                                        //The index
of column for sorting
	const char* searchWord);                                    //The word
that is searched in selected columns of database

//Returns decrypted lines from logins database.
//Result: If the function succeeds, the return value is 0 or 1. If the function fails,
the return value is 2, or 3, or 4, or 5.
//Errors:     1) CANNOT_OPEN_DATABASE is value 2. If database is busy.
//                  2) CANNOT_CLOSE_DATABASE is value 3. If database is busy.
//                  3) OUT_BUFFER_IS_TOO_SMALL is value 4. If you set out buffer that
has size smaller, than readed result.
//                  4) DB_OR_IS_EMPTY_OR_IS_NOT_FIND is value 5. If database is empty,
or you set false data.
extern "C"
DBWORK_API int GetPageFromLogin(const char* dbname,          //The name of database
	const char* tablename,                                          //The
name of table in database
	const char* listOfColumns,                                  //The list of
column`s names in selected table
	const char* jsonName,                                           //The
name of JSON file, where is located secret key
	char* out,
	//The out buffer
	int outSize,                                                    //The
size of the out buffer
	int startLine,
	//The number of line from we started to read database
	int numbOfLines,                                                //The
number of lines for reading
	int modeOfSorting,                                              //The
sorting`s mode
	int columnBySorting,                                        //The index
of column for sorting
	const char* searchWord);                                    //The word
that is searched in selected columns of database

//Returns lines from history database.
//Result: If the function succeeds, the return value is 0 or 1. If the function fails,
the return value is 2, or 3, or 4, or 5.
//Errors:     1) CANNOT_OPEN_DATABASE is value 2. If database is busy.
//                  2) CANNOT_CLOSE_DATABASE is value 3. If database is busy.
//                  3) OUT_BUFFER_IS_TOO_SMALL is value 4. If you set out buffer that
has size smaller, than readed result.
//                  4) DB_OR_IS_EMPTY_OR_IS_NOT_FIND is value 5. If database is empty,
or you set false data.
extern "C"
DBWORK_API int GetPageFromHistory(const char* dbname,  //The name of database
	const char* tablename,                                          //The
name of table in database
	const char* listOfColumns,                                  //The list of
column`s names in selected table
	char* out,
	//The out buffer
	int outSize,                                                    //The
size of the out buffer
	int startLine,
	//The number of line from we started to read database
```

```
	int numbOfLines,                                                    //The
number of lines for reading
	int modeOfSorting,                                                  //The
sorting`s mode
	int columnBySorting,                                                //The index
of column for sorting
	const char* searchWord);                                            //The word
that is searched in selected columns of database

//Count number of lines in database.
//Result: If the function succeeds, the return value is zero. If the function fails, the
return value is nonzero.
//Errors:     1) UNIDENTIFIED_COLUMN is value -5. If you set index of column that is non
existing.
//                  2) UNIDENTIFIED_MODE_OF_SORTING is value -6. If you set index of
mode of sorting that is non existing.
//                  3) SQL_OUT_BUFFER_IS_TOO_SMALL is value -7. If you set out buffer
that has size smaller, than created sql inquery.
//                  4) DB_OR_IS_EMPTY_OR_IS_NOT_FIND is value 5. If database is empty,
or you set false data.
extern "C"
DBWORK_API int NumbOfLines(const char* dbname,                //The name of database
	const char* tablename,                                              //The
name of table in database
	const char* listOfColumns,                                          //The list of
column`s names in selected table
	const char* searchWord);                                            //The word
that is searched in selected columns of database
```

dbwork.cpp

```
#include "dbwork.h"

DBWORK_API int MakeSQL(const char* tablename,
	const char* listOfColumns,
	char* out,
	int outSize,
	int modeOfSorting,
	int columnBySorting,
	const char* searchWord)
{
	std::string tmp, sql, search;
	search = std::string(searchWord);

	//count number of columns
	tmp = std::string(listOfColumns);
	int numbOfColumns = count(tmp.begin(), tmp.end(), ';');

	//get names of columns if some non standart modes is activated
	std::vector<std::string> names;
	if (modeOfSorting != STANDART_MODE || !search.empty())
	{
		int nw, i;
		std::string res;

		tmp = std::string(listOfColumns);
		for (i = 0; i <= numbOfColumns; i++)
		{
			nw = tmp.find(';');
			if (nw == std::string::npos)
			{
				names.push_back(tmp);
				break;
			}
```

```
                names.push_back(tmp.substr(0, nw));
                tmp.erase(tmp.begin(), tmp.begin() + nw + 1);
        }
    }

    //replace all ';' on ','
    tmp = std::string(listOfColumns);
    for (int i = 0; i < (int)tmp.size(); i++)
        if (tmp[i] == ';')
            tmp[i] = ',';

    sql = "SELECT " + tmp + " FROM " + std::string(tablename);

    //search words in fields of table
    if (!search.empty())
    {
        sql += " WHERE ";
        for (int i = 0; i <= numbOfColumns; i++)
        {
            sql += names[i] + " LIKE \'%" + search + "%\'";
            if (i + 1 <= numbOfColumns)
                sql += " OR ";
        }
    }

    //sort table by column
    if (modeOfSorting != STANDART_MODE)
    {
        sql += " ORDER BY ";

        //check by range
        if (columnBySorting > numbOfColumns || columnBySorting < 0)
            return UNIDENTIFIED_COLUMN;

        sql += names[columnBySorting];

        //choose mode of sorting
        if (modeOfSorting == MODE_FROM_SMALLEST_TO_BIGGEST)
            sql += " ASC";
        else if (modeOfSorting == MODE_FROM_BIGGEST_TO_SMALLEST)
            sql += " DESC";
        else
            return UNIDENTIFIED_MODE_OF_SORTING;
    }

    //clear
    names.clear();
    tmp.clear();
    search.clear();

    //return result
    if ((int)sql.size() + 1 > outSize)
        return SQL_OUT_BUFFER_IS_TOO_SMALL;

    strcpy_s(out, outSize, sql.c_str());
    out[sql.size() + 1] = '\0';
    return SQL_CREATED;
}

DBWORK_API int GetPageFromLogin(const char* dbname,
    const char* tablename,
    const char* listOfColumns,
    const char* jsonName,
```

```
	char* out,
	int outSize,
	int startLine,
	int numbOfLines,
	int modeOfSorting,
	int columnBySorting,
	const char* searchWord)
{
	WrapperDB db;

	//get number of column that called password_value
	std::string tmp = std::string(listOfColumns);
	int numbOfPasswordValue = tmp.find("password_value");
	if (numbOfPasswordValue != std::string::npos) {
		tmp = tmp.substr(0, numbOfPasswordValue + 14);
		numbOfPasswordValue = count(tmp.begin(), tmp.end(), ';');
	}

	//count number of columns
	tmp = std::string(listOfColumns);
	int numbOfColumns = count(tmp.begin(), tmp.end(), ';');

	//prepare sql
	std::string sql(3 * (strlen(listOfColumns) + strlen(searchWord) + 10) + 250,
'\0');
	int sqlres = MakeSQL(tablename, listOfColumns, (char*)sql.c_str(), sql.size(),
modeOfSorting, columnBySorting, searchWord);
	if (sqlres != SQL_CREATED)
		return sqlres;

	//open our database
	db.SetDb(dbname);

	//read data from database, write to file and show it
	if (!db.SetStmt(sql))
		return DB_OR_IS_EMPTY_OR_IS_NOT_FIND;
	sql.clear();

	//go to our line
	for (int i = 0; i <= startLine; i++)
		if (db.StmtNextLine() == -1)
			return LINES_IN_DATABASE_IS_OVER;

	std::string str, result;
	bool isItEndOfDB = false;
	unsigned char* bytes, *bytekey = GetMasterKey(jsonName);

	for (int i = 0; i < numbOfLines; i++)
	{
		for (int j = 0; j <= numbOfColumns; j++) {
			if (j == numbOfColumns) {
				tmp = std::string((char*)sqlite3_column_text(db.GetStmt(),
j));
				bytes = (unsigned char*)sqlite3_column_text(db.GetStmt(), j);

				if (tmp.size() > 16)
					result += std::string(DecryptByAes256Gcm(bytes,
bytekey));
				else
					result += std::string((char*)DecryptByDPAPI(bytes));

				result += "; ";
				continue;
			}
```

```
                result += (char*)sqlite3_column_text(db.GetStmt(), j);
                result += "; ";
            }
            result += '\n';

            if (db.StmtNextLine() == -1)
            {
                isItEndOfDB = true;
                break;
            }
        }

        if ((int)result.size() + 1 > outSize)
            return OUT_BUFFER_IS_TOO_SMALL;

        strcpy_s(out, outSize, result.c_str());
        out[result.size() + 1] = '\0';

        if (isItEndOfDB)
            return LINES_IN_DATABASE_IS_OVER;

        return LINES_IN_DATABASE_IS_NOT_OVER;
}

DBWORK_API int GetPageFromHistory(const char* dbname,
        const char* tablename,
        const char* listOfColumns,
        char* out,
        int outSize,
        int startLine,
        int numbOfLines,
        int modeOfSorting,
        int columnBySorting,
        const char* searchWord)
{
        WrapperDB db;

        //count number of columns
        std::string tmp = std::string(listOfColumns);
        int numbOfColumns = count(tmp.begin(), tmp.end(), ';');

        //prepare sql
        std::string sql(3 * (strlen(listOfColumns) + strlen(searchWord) + 10) + 250,
'\0');
        int sqlres = MakeSQL(tablename, listOfColumns, (char*)sql.c_str(), sql.size(),
modeOfSorting, columnBySorting, searchWord);
        if (sqlres != SQL_CREATED)
            return sqlres;

        //open our database
        db.SetDb(dbname);

        //read data from database, write to file and show it
        if (!db.SetStmt(sql))
            return DB_OR_IS_EMPTY_OR_IS_NOT_FIND;
        sql.clear();

        //go to our line
        for (int i = 0; i <= startLine; i++)
            if (db.StmtNextLine() == -1)
                return LINES_IN_DATABASE_IS_OVER;

        std::string result;
        bool isItEndOfDB = false;
```

```
	//output result
	for (int i = 0; i < numbOfLines; i++)
	{
		for (int j = 0; j <= numbOfColumns; j++)
		{
			result += (char*)sqlite3_column_text(db.GetStmt(), j);
			result += "; ";
		}
		result += '\n';

		if (db.StmtNextLine() == -1)
		{
			isItEndOfDB = true;
			break;
		}
	}

	if ((int)result.size() + 1 > outSize)
		return OUT_BUFFER_IS_TOO_SMALL;

	strcpy_s(out, outSize, result.c_str());
	out[result.size() + 1] = '\0';

	if (isItEndOfDB)
		return LINES_IN_DATABASE_IS_OVER;

	return LINES_IN_DATABASE_IS_NOT_OVER;
}

DBWORK_API int NumbOfLines(const char* dbname,
	const char* tablename,
	const char* listOfColumns,
	const char* searchWord)
{
	WrapperDB db;
	std::string sql;

	//prepare sql
	sql.resize(3 * (strlen(listOfColumns) + strlen(searchWord) + 10) + 250);
	int sqlres = MakeSQL(tablename, listOfColumns, (char*)sql.c_str(), sql.size(),
STANDART_MODE, 0, searchWord);
	if (sqlres != SQL_CREATED)
		return sqlres;

	//open our database
	db.SetDb(dbname);

	//read data from database, write to file and show it
	if (!db.SetStmt(sql))
		return DB_OR_IS_EMPTY_OR_IS_NOT_FIND;
	sql.clear();

	//go to our line
	int counter = 0;
	while (db.StmtNextLine() != -1)
		counter++;

	return counter;
}
```

framework.h

```
#pragma once

#define WIN32_LEAN_AND_MEAN             // Исключите редко используемые компоненты из
заголовков Windows
// Файлы заголовков Windows
#include <windows.h>
```

dllmain.cpp

```
// dllmain.cpp : Определяет точку входа для приложения DLL.
#include "filework.h"
#include "decryptor.h"
#include "dbwork.h"

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

UnitTests

pch.h

```
//
// pch.h
//

#pragma once

#include "gtest/gtest.h"

#include <vector>
#include <string>
```

pch.cpp

```
//
// pch.cpp
//

#include "pch.h"
```

test.cpp

```
#include "pch.h"
#include "WrapperDB.h"

#include "filework.h"
#include "decryptor.h"
#include "dbwork.h"

void EncryptStringUsingDPAPI(const std::string& str, std::vector<BYTE>& outBuff)
{
```

```
	DATA_BLOB DataIn;
	DATA_BLOB DataOut;
	BYTE* pbDataInput = (BYTE*)str.c_str();
	DWORD cbDataInput = strlen((char*)pbDataInput) + 1;

	DataIn.pbData = pbDataInput;
	DataIn.cbData = cbDataInput;

	CryptProtectData(&DataIn, NULL, NULL, NULL, NULL, 0, &DataOut);

	outBuff.clear();
	outBuff.resize(1024);
	for (int i = 0; i < DataOut.cbData; i++)
		outBuff[i] = DataOut.pbData[i];
	outBuff[DataOut.cbData] = '\0';

	LocalFree(DataOut.pbData);
}

//-------------------------------------------------------- FileCopy ------------------------
------------------------------
TEST(FileCopyCase, TryingToCopyNonExistentFile)
{
	EXPECT_FALSE(FileCopy("xxx.txt", "xxxx.txt", false));
}

TEST(FileCopyCase, TryingToCopyExistingFile)
{
	auto res = CreateFile(L"1.txt", GENERIC_WRITE, FILE_SHARE_READ, NULL, CREATE_NEW,
FILE_ATTRIBUTE_NORMAL, NULL);
	EXPECT_TRUE(res != INVALID_HANDLE_VALUE);
	EXPECT_TRUE(CloseHandle(res));

	EXPECT_TRUE(FileCopy("1.txt", "2.txt", false));
}

TEST(FileCopyCase, TryingToCopyNonExistentFileToAppDataLocal)
{
	EXPECT_FALSE(FileCopy("xxx.txt", "xxxx.txt", true));
}

TEST(FileCopyCase, TryingToCopyExistingFileToAppDataLocal)
{
	std::wstring wpath = PathToAppDataLocal();
	std::string spath(wpath.begin(), wpath.end());
	spath += "\\3.txt";

	auto res = CreateFile(L"1.txt", GENERIC_WRITE, FILE_SHARE_READ, NULL,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
	EXPECT_TRUE(res != INVALID_HANDLE_VALUE);
	EXPECT_TRUE(CloseHandle(res));

	EXPECT_TRUE(FileCopy("1.txt", (char*)spath.c_str(), false));
}

//-------------------------------------------------------- FileDelete ----------------------
------------------------------
TEST(FileDeleteCase, TryingToDeleteExistingFileFromProjectDirectory)
{
	auto res = CreateFile(L"4.txt", GENERIC_WRITE, FILE_SHARE_READ, NULL, CREATE_NEW,
FILE_ATTRIBUTE_NORMAL, NULL);
	EXPECT_TRUE(res != INVALID_HANDLE_VALUE);
	EXPECT_TRUE(CloseHandle(res));
```

```
	EXPECT_TRUE(FileDelete("4.txt", false));
}

TEST(FileDeleteCase, TryingToDeleteExistingFileFromAppDataLocal)
{
	std::wstring wpath = PathToAppDataLocal();
	wpath += L"\\3.txt";

	auto res = CreateFile(wpath.c_str(), GENERIC_WRITE, FILE_SHARE_READ, NULL,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
	EXPECT_TRUE(res != INVALID_HANDLE_VALUE);
	EXPECT_TRUE(CloseHandle(res));

	EXPECT_TRUE(FileDelete("\\3.txt", true));
}

TEST(FileDeleteCase, TryingToDeleteNonExistentFileFromProjectDirectory)
{
	EXPECT_FALSE(FileDelete("xxxx.txt", false));
}

TEST(FileDeleteCase, TryingToDeleteNonExistentFileFromAppDataLocal)
{
	EXPECT_FALSE(FileDelete("xxxx.txt", true));
}

//------------------------------------------------------ ExportTo ------------------------
------------------------------
TEST(ExportToCase, TryingToExportHelloToNonExistentFile)
{
	FileDelete("exportTo.txt", false);

	EXPECT_TRUE(ExportTo("exportTo.txt", "Hello", true));
}

TEST(ExportToCase, TryingToExportHelloExistingFile)
{
	auto res = CreateFile(L"exportTo.txt", GENERIC_WRITE, FILE_SHARE_READ, NULL,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
	EXPECT_TRUE(res != INVALID_HANDLE_VALUE);
	EXPECT_TRUE(CloseHandle(res));

	EXPECT_TRUE(ExportTo("exportTo.txt", "Hello", true));
}

//------------------------------------------------- DecryptByDPAPI ---------------------
------------------------------
TEST(DPAPICase, TryingToDecrypthEncryptedTextByDPAPI)
{
	std::string word("Hello");

	std::vector<BYTE> encryptedBuff;
	EncryptStringUsingDPAPI(word, encryptedBuff);

	std::string result =
reinterpret_cast<char*>(DecryptByDPAPI(encryptedBuff.data()));
	EXPECT_STREQ(result.c_str(), word.c_str());
}

TEST(DPAPICase, TryingToDecrypthRandomText)
{
	std::string word("Hello");

	std::vector<BYTE> encryptedBuff;
```

```
	for (auto& symbol : word)
	{
		encryptedBuff.emplace_back(symbol);
	}
	encryptedBuff.resize(1024);

	std::string result =
reinterpret_cast<char*>(DecryptByDPAPI(encryptedBuff.data()));
	EXPECT_STREQ(result.c_str(), "");
}

//------------------------------------------------------ MakeSQL ------------------------
------------------------------
TEST(MakeSqlCase, TryingToCreateSqlWithoutErrors)
{
	std::string str;
	str.resize(150);

	auto res = MakeSQL("urls", "url", (char*)str.c_str(), str.size(), 0, 0, "");
	EXPECT_EQ(res, SQL_CREATED);
}

TEST(MakeSqlCase, TryingToGiveFalseNumberOfColumns)
{
	std::string str;
	str.resize(150);

	auto res = MakeSQL("urls", "url", (char*)str.c_str(), str.size(), 1, -1, "");
	EXPECT_EQ(res, UNIDENTIFIED_COLUMN);
}

TEST(MakeSqlCase, TryingToGiveFalseNumberOfModeOfSorting)
{
	std::string str;
	str.resize(150);

	auto res = MakeSQL("urls", "url", (char*)str.c_str(), str.size(), -1, 0, "");
	EXPECT_EQ(res, UNIDENTIFIED_MODE_OF_SORTING);
}

TEST(MakeSqlCase, TryingToGiveSmallOutBuffer)
{
	std::string str;

	auto res = MakeSQL("urls", "url", (char*)str.c_str(), str.size(), 0, 0, "");
	EXPECT_EQ(res, SQL_OUT_BUFFER_IS_TOO_SMALL);
}

//------------------------------------------------------ NumbOfLines ------------------------
------------------------------
TEST(NumbOfLinesCase, TryingToCountLinesInDb)
{
	auto res = NumbOfLines("hstr", "urls", "url", "");
	EXPECT_EQ(res, 3);
}

TEST(NumbOfLinesCase, TryingToCountLinesInNonExistentDb)
{
	auto res = NumbOfLines("xxxx", "xxx", "x", "");
	EXPECT_EQ(res, DB_OR_IS_EMPTY_OR_IS_NOT_FIND);

	res = NumbOfLines("hstr", "xxx", "url", "");
	EXPECT_EQ(res, DB_OR_IS_EMPTY_OR_IS_NOT_FIND);
}
```

```
//------------------------------------------------- GetPageFromDB --------------------
-----------------------------
TEST(GetPageFromDBCase, TryingReadAllLinesInDB)
{
	std::string str;
	str.resize(5000);

	auto res = GetPageFromHistory("hstr", "urls", "url, title, visit_count",
(char*)str.c_str(), str.size(), 0, 5, 0, 1, "");
	EXPECT_EQ(res, LINES_IN_DATABASE_IS_OVER);
}

TEST(GetPageFromDBCase, TryingReadFirstLineInDB)
{
	std::string str;
	str.resize(1000);

	auto res = GetPageFromHistory("hstr", "urls", "url, title, visit_count",
(char*)str.c_str(), str.size(), 0, 1, 0, 1, "");
	EXPECT_EQ(res, LINES_IN_DATABASE_IS_NOT_OVER);
}

TEST(GetPageFromDBCase, TryingGiveSmallBuffer)
{
	std::string str;

	auto res = GetPageFromHistory("hstr", "urls", "url, title, visit_count",
(char*)str.c_str(), str.size(), 0, 5, 0, 1, "");
	EXPECT_EQ(res, OUT_BUFFER_IS_TOO_SMALL);
}

TEST(GetPageFromDBCase, TryingToOpenNonExistentDB)
{
	std::string str;
	str.resize(1000);

	FileDelete("tmp", false);

	auto res = GetPageFromHistory("tmp", "urls", "url, title, visit_count",
(char*)str.c_str(), str.size(), 0, 5, 0, 1, "");
	EXPECT_EQ(res, DB_OR_IS_EMPTY_OR_IS_NOT_FIND);
}

int main(int argc, char** argv)
{
	::testing::InitGoogleTest(&argc, argv);
	return RUN_ALL_TESTS();
}
```

GUI.exe

GeneralInfo.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GUI
{
    public struct GeneralInfo
    {
```

```
        public int lentghOfOneLine;
        public int selectedPage;
        public int numbOfLines;
        public int numbOfLists;
        public int modeOfSorting;
        public int columnForSorting;
        public DataReader selectedReader;
        public String searchWord;
    }
}
```

FileWork.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;

namespace GUI
{
    static public class FileWork
    {
        private const string PathToDLL = "Native.dll";

        //-----------------------------------------------PathToAppDataLocal----------------
------------------
        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl,
            CharSet = CharSet.Unicode)]
        static public extern string PathToAppDataLocal();

        //-----------------------------------------------FileCopy--------------------------
------------------
        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl,
            CharSet = CharSet.Ansi)]
        static public extern int FileCopy(String oldFilename,
            String newFilename,
            bool isFileInAppDataLocal);

        //-----------------------------------------------FileDelete------------------------
------------------
        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl,
            CharSet = CharSet.Ansi)]
        static public extern int FileDelete(String filename,
            bool isFileInAppDataLocal);

        //-----------------------------------------------ExportTo--------------------------
------------------
        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl,
            CharSet = CharSet.Ansi)]
        static public extern bool ExportTo(String filename,
            String text,
            bool isClear);
    }
}
```

Decryptor.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;
```

```
namespace GUI
{
    static public class Decryptor
    {
        private const string PathToDLL = "Native.dll";

        //-----------------------------------------------GetMasterKey----------------------
------------------
        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl,
            CharSet = CharSet.Ansi)]
        static public extern string GetMasterKey(String jsonName);

        //-----------------------------------------------DecryptByDPAPI--------------------
------------------
        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl,
            CharSet = CharSet.Ansi)]
        static public extern string DecryptByDPAPI(String data);

        //-----------------------------------------------DecryptByAes256Gcm----------------
------------------
        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl,
            CharSet = CharSet.Ansi)]
        static public extern string DecryptByAes256Gcm(String bytebuff,
            String bytekey);
    }
}
```

DbWork.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;

namespace GUI
{
    static public class DbWork
    {
        private const string PathToDLL = "Native.dll";

        //-----------------------------------------GetPageFromLogin--------------------
------------------
        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        static public extern int GetPageFromLogin(String dbname,
            String tablename,
            String listOfColumns,
            String jsonName,
            StringBuilder outBuff,
            int outSize,
            int startLine,
            int numbOfLines,
            int modeOfSorting,
            int columnBySorting,
            String searchWord);

        //-----------------------------------------GetPageFromHistory------------------
------------------
        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        static public extern int GetPageFromHistory(String dbname,
            String tablename,
            String listOfColumns,
```

```
            StringBuilder outBuff,
            int outSize,
            int startLine,
            int numbOfLines,
            int modeOfSorting,
            int columnBySorting,
            String searchWord);

        //-------------------------------------------------MakeSQL----------------------
-------------------
        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        static public extern int MakeSQL(String tablename,
            String listOfColumns,
            StringBuilder outBuff,
            int outSize,
            int modeOfSorting,
            int columnBySorting,
            String searchWord);

        //-------------------------------------------------NumbOfLines--------------------
-------------------
        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        static public extern int NumbOfLines(String dbname,
            String tablename,
            String listOfColumns,
            String searchWord);
    }
}
```

CacheWork.cs

```
using System;
using System.Runtime.InteropServices;
using System.Text;

namespace GUI
{
    class CacheWork
    {
        private const string PathToDLL = "Native.dll";

        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Unicode)]
        static public extern int GetPageFromCache(
            String        cacheDirPath,
            StringBuilder outBuff,
            int           outSize,
            int           startLine,
            int           numbOfLines,
            int           modeOfSorting,
            int           columnBySorting,
            String        searchWord);


        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Unicode)]
        static public extern int NumbOfLinesCache(
            String cacheDirPath,
            String searchWord);

        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Unicode)]
```

```
        static public extern void ExportCache(
            String cacheDirPath,
            int    sortColumn,
            int    ascending,
            int    entriesOffset,
            int    entriesNumber,
            String filter,
            String exportDir,
            int    saveMD5);

        [DllImport(PathToDLL, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Unicode)]
        static public extern void SaveFileToTemp(
            String        cache_dir_path,
            int           cache_addr,
            StringBuilder out_path,
               int           out_path_size);
    }
}
```

DataReader.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GUI
{
    public interface DataReader
    {
        int GetPage(int startline,
            int numboflines,
            StringBuilder result,
            int result_size,
            int modeofsorting,
            int columnforsorting,
            String searchword);

        int NumbOfLines(String searchWord);
    }
}
```

CacheReader.cs

```
using System;
using System.IO;
using System.Text;

namespace GUI
{
    class CacheReader : DataReader
    {
        private readonly string CacheDirPath_;

        public CacheReader()
        {
            CacheDirPath_ = Path.Combine(

Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData),
                    @"Google\Chrome\User Data\Default\Cache\Cache_Data");
        }

        public int GetPage(
           int startline,
```

```
   int numboflines,
   StringBuilder result,
   int result_size,
   int modeofsorting,
   int columnforsorting,
   String searchword)
{
    return CacheWork.GetPageFromCache(
        CacheDirPath_,
        result,
        result_size,
        startline,
        numboflines,
        ReinterpetModeSorting(modeofsorting),
        columnforsorting,
        searchword);
}

public int NumbOfLines(String searchWord)
{
    return CacheWork.NumbOfLinesCache(
        CacheDirPath_,
        searchWord);
}

public void ExportCache(
    int    sortColumn,
    bool   ascending,
    int    entriesOffset,
    int    entriesNumber,
    string filter,
    string exportDir,
    bool   saveMD5)
{
    CacheWork.ExportCache(
        CacheDirPath_,
        sortColumn,
        ascending ? 1 : 0,
        entriesOffset,
        entriesNumber,
        filter,
        exportDir,
        saveMD5 ? 1 : 0);
}

public string SaveToTemp(
    int cacheAddr)
{
    const int pathSize = 256;
    StringBuilder builder = new StringBuilder(pathSize);
    CacheWork.SaveFileToTemp(
        CacheDirPath_,
        cacheAddr,
        builder,
        pathSize);
    return builder.ToString();
}

/// <summary>
/// Reinterprets mode of sornting to match false - ascending, true - descending
/// </summary>
/// <param name="mode">Old mode</param>
/// <returns>New mode</returns>
private int ReinterpetModeSorting(int mode)
```

```
        {
            if (mode == 0 || mode == 1)
            {
                return 0;
            }
            else
            {
                return 1;
            }
        }

    }
}
```

LoginReader.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GUI
{
    class LoginReader: DataReader
    {
        //constatnts
        static private string LoginPath = "\\Google\\Chrome\\User Data\\Default\\Login
Data";
        static private string JsonPath = "\\Google\\Chrome\\User Data\\Local State";
        static public int UrlColumn = 0;
        static public int UserNameColumn = 1;
        static public int PasswordColumn = 2;

        //fields
        private string dbname;
        private string tablename;
        private string listofcolumns;
        private string json;
        private string key;

        //methods
        public LoginReader()
        {
            dbname = "login";
            tablename = "logins";
            json = "local";
            listofcolumns = "origin_url; username_value; password_value";

            FileWork.FileCopy(LoginPath, dbname, true);
            FileWork.FileCopy(JsonPath, json, true);

            key = Decryptor.GetMasterKey(json);
        }

        ~LoginReader()
        {
            FileWork.FileDelete(dbname, false);
            FileWork.FileDelete(json, false);
        }

        public int GetPage(int startline,
            int numboflines,
            StringBuilder result,
            int result_size,
```

```
            int modeofsorting,
            int columnforsorting,
            String searchword)
        {
            return DbWork.GetPageFromLogin(dbname,
                tablename,
                listofcolumns,
                json,
                result,
                result_size,
                startline,
                numboflines,
                modeofsorting,
                columnforsorting,
                searchword);
        }

        public int NumbOfLines(String searchWord)
        {
            return DbWork.NumbOfLines(dbname,
                tablename,
                listofcolumns,
                searchWord);
        }
    }
}
```

HistoryReader.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GUI
{
    class HistoryReader: DataReader
    {
        //constatnts
        static private string HistoryPath = "\\Google\\Chrome\\User
Data\\Default\\History";
        static public int UrlColumn = 0;
        static public int TitleColumn = 1;
        static public int LastVisitTimeColumn = 2 ;

        //fields
        private string dbname;
        private string tablename;
        private string listofcolumns;

        //methods
        public HistoryReader()
        {
            dbname = "history";
            tablename = "urls";
            listofcolumns = "url; title; datetime(last_visit_time / 1000000 -
11644473600, 'unixepoch', 'localtime')";

            FileWork.FileCopy(HistoryPath, dbname, true);
        }

        ~HistoryReader()
        {
            FileWork.FileDelete(dbname, false);
```

```
        }

        public int GetPage(int startline,
            int numboflines,
            StringBuilder result,
            int result_size,
            int modeofsorting,
            int columnforsorting,
            String searchword)
        {
            return DbWork.GetPageFromHistory(dbname,
                tablename,
                listofcolumns,
                result,
                result_size,
                startline,
                numboflines,
                modeofsorting,
                columnforsorting,
                searchword);
        }

        public int NumbOfLines(String searchWord)
        {
            return DbWork.NumbOfLines(dbname,
                tablename,
                listofcolumns,
                searchWord);
        }
    }
}
```

MainForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using Microsoft.WindowsAPICodePack.Dialogs;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Diagnostics;
using System.IO;
using System.Reflection;

namespace GUI
{
    public partial class MainForm : Form
    {
        GeneralInfo ginfo;

        private bool isAscending = true;

        public MainForm()
        {
            InitializeComponent();
            ginfo = new GeneralInfo();
        }

        void ChangeParam(GeneralInfo g)
        {
            ginfo.selectedPage = 0;
```

```
            ginfo.columnForSorting = g.columnForSorting;
            ginfo.modeOfSorting = g.modeOfSorting;
            ginfo.searchWord = g.searchWord;

            if (ginfo.selectedReader == null)
                return;

            float tmp = (float)ginfo.selectedReader.NumbOfLines(ginfo.searchWord) /
ginfo.numbOfLines;
            if (tmp > (int)tmp)
                ginfo.numbOfLists = (int)tmp + 1;
            else
                ginfo.numbOfLists = (int)tmp;
        }

        async Task UpdateStructAsync()
        {
            ginfo.numbOfLines = (int)numericUpElementsOnList.Value;
            ginfo.modeOfSorting = 0;
            ginfo.columnForSorting = 0;
            ginfo.selectedPage = 0;
            ginfo.searchWord = "";
            ginfo.numbOfLists = await GetNumberOfPagesAsync(ginfo);
        }

        bool IsDateInvalid(DataReader dr, string date)
        {
            return dr.GetType() == typeof(HistoryReader) &&
                date.Length != 0 &&
                date[0] == '1';
        }

        async Task<int> UpdateDataGridViewAsync()
        {
            StringBuilder request = new StringBuilder(ginfo.lentghOfOneLine *
ginfo.numbOfLines);

            textBoxSelectedPage.Text = (ginfo.selectedPage + 1).ToString();

            int startline = ginfo.selectedPage * ginfo.numbOfLines;

            var result = await Task.Run(() =>
            {
                return ginfo.selectedReader.GetPage(startline,
                ginfo.numbOfLines,
                request,
                request.Capacity,
                ginfo.modeOfSorting,
                ginfo.columnForSorting,
                ginfo.searchWord);
            });


            //LINES_IN_DATABASE_IS_OVER || LINES_IN_DATABASE_IS_NOT_OVER
            if ((result == 0 || result == 1) &&
                (request.Length == 0 || request[0] != '\0' || request[1] != '\0'))
            {
                dataGridView.Rows.Clear();

                while (request.Length != 0)
                {
                    object[] row = new object[dataGridView.ColumnCount];

                    for (int i = 0; i < dataGridView.ColumnCount; i++)
```

```
                    {
                        row[i] = TakeColumn(request);
                        if (i == 2 && IsDateInvalid(ginfo.selectedReader,
row[i].ToString()))
                            row[i] = "";
                    }

                    request.Remove(0, 1);

                    dataGridView.Rows.Add(row);
                }
            }

            return result;
        }

        string TakeColumn(StringBuilder str)
        {
            string column = "";
            //char[] column = new char[1];
            int semicolPos = str.ToString().IndexOf(';');

            if (semicolPos < 0)
                return column;

            //column = new char[tmp + 1];

            //str.CopyTo(0, column, 0, tmp);
            column = str.ToString(0, semicolPos);

            str.Remove(0, semicolPos + 2);

            return column;
        }

        async Task ExportTableAsync(int start_page, int end_page, int step)
        {
            if (ginfo.selectedReader == null)
                return;

            if (saveFileDialog.ShowDialog() != DialogResult.OK)
            {
                MessageBox.Show("Table was NOT exported", "Export", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                return;
            }

            string title = "", filename = saveFileDialog.FileName;
            StringBuilder str = new StringBuilder(ginfo.lentghOfOneLine *
ginfo.numbOfLines);

            var tableType = ginfo.selectedReader.GetType();

            if (tableType == typeof(HistoryReader))
                title = @"URL;Title;Last visit time";

            if (tableType == typeof(LoginReader))
                title = @"Origin URL;Username;Password";

            if (tableType == typeof(CacheReader))
                title = @"ID;URL;Name;Type;Size";

            await Task.Run(() =>
            {
```

```
                FileWork.ExportTo(filename, title, true);
                for (int i = start_page; i < end_page; i++)
                {
                    if (ginfo.selectedReader.GetPage(i * step,
                        ginfo.numbOfLines,
                        str,
                        str.Capacity,
                        ginfo.modeOfSorting,
                        ginfo.columnForSorting,
                        ginfo.searchWord) == 0)
                    {
                        FileWork.ExportTo(filename, str.ToString(), false);
                        break;
                    }
                    FileWork.ExportTo(filename, str.ToString(), false);
                }
            });


            MessageBox.Show("Exported succsessfuly", "Export", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }

        async Task ExportCacheAsync(int startPage, int endPage, int step)
        {
            if (ginfo.selectedReader == null)
            {
                return;
            }

            var reader = ginfo.selectedReader as CacheReader;

            var tableType = ginfo.selectedReader.GetType();
            if (tableType != typeof(CacheReader))
            {
                return;
            }

            var dialog = new CommonOpenFileDialog();
            dialog.IsFolderPicker = true;
            if (dialog.ShowDialog() != CommonFileDialogResult.Ok)
            {
                return;
            }

            var exportDir = Path.Combine(dialog.FileName, "Exported Cache");
            Directory.CreateDirectory(exportDir);

            var saveMD5 = MessageBox.Show(
                "Save MD5 hash?",
                "",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Question) == DialogResult.Yes;

            await Task.Run(() =>
            {
                for (int i = startPage; i < endPage; i++)
                {
                    reader.ExportCache(
                        ginfo.columnForSorting,
                        ginfo.modeOfSorting == 0 ? false : true,
                        i * step,
                        ginfo.numbOfLines,
                        ginfo.searchWord,
```

```
                exportDir,
                saveMD5);
        }
    });
}

private void MainForm_Load(object sender, EventArgs e)
{
    ginfo.lentghOfOneLine = 1000;
    ginfo.numbOfLines = (int)numericUpElementsOnList.Value;
    ginfo.selectedPage = 0;
    ginfo.numbOfLists = 0;
    ginfo.modeOfSorting = 0;
    ginfo.columnForSorting = 0;
    ginfo.searchWord = "";

    textBoxSelectedPage.Text = "1";
}

private async void buttonHistory_ClickAsync(object sender, EventArgs e)
{
    ginfo.selectedReader = new HistoryReader();
    await UpdateStructAsync();

    dataGridView.ColumnCount = 3;
    dataGridView.Columns[0].Name = "URL";
    dataGridView.Columns[1].Name = "Title";
    dataGridView.Columns[2].Name = "Last visit time";

    textBoxSelectedPage.Text = (ginfo.selectedPage + 1).ToString();
    await UpdateDataGridViewAsync();
    JustifyDataGrid();
}

private async void buttonLogins_ClickAsync(object sender, EventArgs e)
{
    ginfo.selectedReader = new LoginReader();
    await UpdateStructAsync();

    dataGridView.ColumnCount = 3;
    dataGridView.Columns[0].Name = "Origin URL";
    dataGridView.Columns[1].Name = "Username";
    dataGridView.Columns[2].Name = "Password";

    textBoxSelectedPage.Text = (ginfo.selectedPage + 1).ToString();
    await UpdateDataGridViewAsync();
    JustifyDataGrid();
}

private async void buttonCache_ClickAsync(object sender, EventArgs e)
{
    ginfo.selectedReader = new CacheReader();
    await UpdateStructAsync();

    dataGridView.ColumnCount = 5;
    dataGridView.Columns[0].Name = "ID";
    dataGridView.Columns[1].Name = "URL";
    dataGridView.Columns[2].Name = "Name";
    dataGridView.Columns[3].Name = "Type";
    dataGridView.Columns[4].Name = "Size";

    textBoxSelectedPage.Text = (ginfo.selectedPage + 1).ToString();
    await UpdateDataGridViewAsync();
    JustifyDataGrid();
```

```
        }

        private async void buttonNext_ClickAsync(object sender, EventArgs e)
        {
            if (ginfo.selectedPage + 1 >= ginfo.numbOfLists)
                return;

            ginfo.selectedPage++;
            textBoxSelectedPage.Text = (ginfo.selectedPage + 1).ToString();
            await UpdateDataGridViewAsync();
        }

        private async void buttonPrevious_ClickAsync(object sender, EventArgs e)
        {
            if (ginfo.selectedPage - 1 < 0)
                return;

            ginfo.selectedPage--;
            textBoxSelectedPage.Text = (ginfo.selectedPage + 1).ToString();
            await UpdateDataGridViewAsync();
        }

        private async void buttonToTheBegin_ClickAsync(object sender, EventArgs e)
        {
            if (ginfo.selectedReader == null)
                return;

            ginfo.selectedPage = 0;
            textBoxSelectedPage.Text = (ginfo.selectedPage + 1).ToString();
            await UpdateDataGridViewAsync();
        }

        private async void buttonToTheEnd_ClickAsync(object sender, EventArgs e)
        {
            if (ginfo.selectedReader == null)
                return;

            if (ginfo.numbOfLists == 0)
                ginfo.selectedPage = ginfo.numbOfLists;
            else
                ginfo.selectedPage = ginfo.numbOfLists - 1;
            textBoxSelectedPage.Text = (ginfo.selectedPage + 1).ToString();
            await UpdateDataGridViewAsync();
        }

        private async void ExportAll_ClickAsync(object sender, EventArgs e)
        {
            await ExportTableAsync(0, ginfo.numbOfLists, ginfo.numbOfLines);
        }

        private async void ExportSelectedPage_ClickAsync(object sender, EventArgs e)
        {
            await ExportTableAsync(ginfo.selectedPage, ginfo.selectedPage + 1,
ginfo.numbOfLines);
        }

        private async void ExportAllCache_ClickAsync(object sender, EventArgs e)
        {
            await ExportCacheAsync(0, ginfo.numbOfLists, ginfo.numbOfLines);
        }

        private async void ExportSelectedPageCache_ClickAsync(object sender, EventArgs e)
        {
```

```
            await ExportCacheAsync(ginfo.selectedPage, ginfo.selectedPage + 1,
ginfo.numbOfLines);
        }

        private async void dataGridView_ColumnHeaderMouseClickAsync(object sender,
DataGridViewCellMouseEventArgs e)
        {
            if (isAscending)
            {
                ginfo.modeOfSorting = 1;
            }
            else
            {
                ginfo.modeOfSorting = 2;
            }
            ginfo.columnForSorting = e.ColumnIndex;
            isAscending = !isAscending;

            if (ginfo.selectedReader != null)
            {
                await UpdateDataGridViewAsync();
            }
        }

        private async void buttonSearch_ClickAsync(object sender, EventArgs e)
        {
            ginfo.searchWord = textBoxSearch.Text;
            if (ginfo.selectedReader != null)
            {
                ginfo.selectedPage = 0;
                ginfo.numbOfLists = await GetNumberOfPagesAsync(ginfo);
                await UpdateDataGridViewAsync();
            }
        }

        private async Task<int> GetNumberOfPagesAsync(GeneralInfo info)
        {
            var totalLines = await Task.Run(() =>
            {
                return info.selectedReader.NumbOfLines(info.searchWord);
            });
            var maxLinesPerPage = info.numbOfLines;
            var pages = (float)totalLines / maxLinesPerPage;
            return (int)Math.Ceiling(pages);
        }

        private void textBoxSearch_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                buttonSearch_ClickAsync(null, null);
            }
        }

        private void JustifyDataGrid()
        {
            for (int i = 0; i < dataGridView.Columns.Count; i++)
            {
                dataGridView.Columns[i].Width = dataGridView.Width /
dataGridView.Columns.Count;
            }
        }

        private void Help_Click(object sender, EventArgs e)
```

```
        {
            const string docName = "Web browser artifacts viewer.pdf";
            Extract("GUI.res", "", docName);
            Process.Start(docName);
        }

        public void Extract(string namespaceName, string internalFilePath, string
resName)
        {
            Assembly assembly = Assembly.GetCallingAssembly();
            using (Stream s = assembly.GetManifestResourceStream(namespaceName + "." +
(internalFilePath == "" ? "" : internalFilePath + ".") + resName))
            using (BinaryReader binaryReader = new BinaryReader(s))
            using (FileStream fileStream = new FileStream(resName,
FileMode.OpenOrCreate))
            using (BinaryWriter binaryWriter = new BinaryWriter(fileStream))
                binaryWriter.Write(binaryReader.ReadBytes((int)s.Length));
        }

        private void openSelectedFileToolStripMenuItem_Click(object sender, EventArgs e)
        {
            var tableType = ginfo.selectedReader.GetType();
            if (tableType != typeof(CacheReader))
            {
                return;
            }
            Console.WriteLine("pass1");
            var index = dataGridView.CurrentCell.RowIndex;
            if (index < 0)
            {
                return;
            }
            Console.WriteLine("pass2");

            var addr = Convert.ToUInt32(dataGridView.Rows[index].Cells[0].Value);
            Console.WriteLine("pass3");

            var reader = ginfo.selectedReader as CacheReader;
            Console.WriteLine("pass4");
            var path = reader.SaveToTemp((int)addr);
            Process.Start(path);
        }
    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GUI
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
```

```
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```