МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

"До захисту допущений"
Зав. кафедри прикладної математики
д.т.н., професор Мартинюк П.М.
«___» _____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ТЕМУ:
«МЕТОДИ ТА ЗАСОБИ КОМП'ЮТЕРНО-ТЕХНІЧНОЇ ЕКСПЕРТИЗИ»

Виконав: Заремба Андрій Миколайович

студент навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки

групи ІПЗ-41

_____

підпис

Керівник: старший викладач, Турбал Маріанна Юріївна

_____

підпис

Рівне – 2023

ЗМІСТ

РЕФЕРАТ

Об'єкт дослідження в даному дослідженні є методи та інструменти комп’ютерно-технічної експертизи

Предмет дослідження є комп’ютерно-технічна експертиза.

Метою даного дослідження є аналіз сфери комп'ютерно-технічної експертизи шляхом дослідження методів та інструментів збору, аналізу та збереження цифрових доказів, а також шляхом розробки власного програмного забезпечення для проведення базової комп’ютерно-технічної експертизи.

Актуальність дослідження на тему «Методи та засоби комп’ютерно-технічної експертизи» полягає у вирішальній ролі такої експертизи в сучасних цифрових дослідженнях. Зі швидким розвитком технологій комп'ютерні системи стали невід'ємною частиною різних аспектів нашого життя, включаючи спілкування, фінанси, бізнес та розваги. Однак ця залежність від цифрової інфраструктури також призвела до збільшення кіберзлочинів, таких як хакерство, витік даних, шахрайство та крадіжка інтелектуальної власності. Комп’ютерно-технічна експертиза, або ж комп’ютерна криміналістика – це науковий процес збору, аналізу та збереження електронних доказів для розслідування та запобігання кіберзлочинам. Вона передбачає застосування різних методів та інструментів для відновлення та вивчення цифрової інформації з комп'ютерів, пристроїв зберігання даних, мереж та цифрових носіїв. Результати комп'ютерних криміналістичних розслідувань можуть бути використані як докази в судових процесах, допомагаючи виявляти та переслідувати кіберзлочинців. Дослідження методів та засобів комп'ютерної криміналістики має вирішальне значення з кількох причин. По-перше, кіберзлочинність продовжує розвиватися і ставати все більш досконалою, що вимагає розробки передових методів ефективного розслідування та боротьби з ними. По-друге, зростаюче використання шифрування, засобів анонімізації та хмарного сховища створює значні проблеми для комп'ютерних криміналістів. Тому важливо розробити нові методології та інструменти, які зможуть подолати ці перешкоди та своєчасно та точно відновити цифрові докази. По-третє, поширення цифрових

пристроїв і величезний обсяг даних, що генеруються щодня, створюють величезні проблеми для слідчих. Дослідження необхідні для розробки ефективних і масштабованих методів для обробки великомасштабного збору, зберігання, аналізу та візуалізації даних. Це включає досягнення у відновленні даних, мережевій криміналістиці, криміналістиці пам'яті та криміналістиці мобільних пристроїв. Крім того, правовий аспект, що стосується кіберзлочинів та цифрових доказів, постійно розвивається. Експертам слід бути в курсі нових правових розробок і забезпечити, щоб методи та інструменти комп'ютерної криміналістики відповідали юридичним вимогам, проблемам конфіденційності та етичним стандартам. Сюди входять питання, пов'язані з конфіденційністю, ланцюгом зберігання, допустимістю доказів та цифровими правами.

Ключові слова: комп'ютерна криміналістика, комп’ютерно-технічна експертиза, цифрові докази, кіберзлочинність, аналіз даних, програма.

ВСТУП

Актуальність дослідження на тему «Методи та засоби комп’ютерно-технічної експертизи» полягає у вирішальній ролі такої експертизи в сучасних цифрових дослідженнях. Зі швидким розвитком технологій комп'ютерні системи стали невід'ємною частиною різних аспектів нашого життя, включаючи спілкування, фінанси, бізнес та розваги. Однак ця залежність від цифрової інфраструктури також призвела до збільшення кіберзлочинів, таких як хакерство, витік даних, шахрайство та крадіжка інтелектуальної власності.

Комп’ютерно-технічна експертиза, або ж комп’ютерна криміналістика – це науковий процес збору, аналізу та збереження електронних доказів для розслідування та запобігання кіберзлочинам. Вона передбачає застосування різних методів та інструментів для відновлення та вивчення цифрової інформації з комп'ютерів, пристроїв зберігання даних, мереж та цифрових носіїв. Результати комп'ютерних криміналістичних розслідувань можуть бути використані як докази в судових процесах, допомагаючи виявляти та переслідувати кіберзлочинців. Дослідження методів та засобів комп'ютерної криміналістики має вирішальне значення з кількох причин. По-перше, кіберзлочинність продовжує розвиватися і ставати все більш досконалою, що вимагає розробки передових методів ефективного розслідування та боротьби з ними. По-друге, зростаюче використання шифрування, засобів анонімізації та хмарного сховища створює значні проблеми для комп'ютерних криміналістів. Тому важливо розробити нові методології та інструменти, які зможуть подолати ці перешкоди та своєчасно та точно відновити цифрові докази. По-третє, поширення цифрових пристроїв і величезний обсяг даних, що генеруються щодня, створюють величезні проблеми для слідчих. Дослідження необхідні для розробки ефективних і масштабованих методів для обробки великомасштабного збору, зберігання, аналізу та візуалізації даних. Це включає досягнення у відновленні даних, мережевій криміналістиці, криміналістиці пам'яті та криміналістиці мобільних пристроїв. Крім того, правовий аспект, що стосується кіберзлочинів

та цифрових доказів, постійно розвивається. Експертам слід бути в курсі нових правових розробок і забезпечити, щоб методи та інструменти комп'ютерної криміналістики відповідали юридичним вимогам, проблемам конфіденційності та етичним стандартам. Сюди входять питання, пов'язані з конфіденційністю, ланцюгом зберігання, допустимістю доказів та цифровими правами.

Метою даного дослідження є аналіз сфери комп'ютерно-технічної експертизи шляхом дослідження методів та інструментів збору, аналізу та збереження цифрових доказів, а також шляхом розробки власного програмного забезпечення для проведення базової комп’ютерно-технічної експертизи. ***Об'єктом дослідження*** в даному дослідженні є методи та інструменти комп’ютерно-технічної експертизи ***Предметом дослідження*** є комп’ютерно-технічна експертиза. Для досягнення мети нам слід виконати наступні завдання:

- Проаналізувати теоретичні засади комп’ютерно-технічної експертизи;
- Дослідити методи та інструменти комп’ютерно-технічної експертизи;
- Здійснити власного практичну розробку програмного забезпечення для проведення базової комп’ютерно-технічної експертизи.

Для того, щоб результати цього дослідження були об’єктивними та комплексними нами було використано кількісні та якісні **методи дослідження**. Їх можна поділити на теоретичні – метод аналізу, метод синтезу, метод порівняння, індуктивні та дедуктивні методи, метод критичного аналізу, описовий метод, а також емпіричні методи – метод спостереження, методи моделювання, дослідницькі методи статистики та репрезентативний метод.

Дослідження використовувати різні методи для досягнення своїх цілей. Нижче наведено опис використовуваних методів, включаючи аналіз, синтез, порівняльний метод, індуктивний та дедуктивний методи, метод критичного аналізу та описовий метод. ***Метод аналізу*** передбачає розбиття складних явищ або понять на більш дрібні компоненти, щоб зрозуміти їх структуру, відносини та характеристики. У контексті комп’ютерно-технічної експертизи метод аналізу може бути використаний для вивчення теоретичних основ, існуючих

методологій та інструментів комп'ютерної криміналістики, виявлення їх сильних сторін, обмежень та областей вдосконалення. ***Метод синтезу*** передбачає об'єднання окремих елементів або ідей для створення нового цілого. У комп'ютерних криміналістичних дослідженнях метод синтезу може бути застосований для розробки нових методологій або програмних засобів, які інтегрують існуючі знання та методи. Це передбачає синтез інформації з різних джерел і синтез різних підходів для створення інноваційних рішень. ***Порівняльний метод*** передбачає порівняння та протиставлення різних явищ, випадків або підходів для виявлення подібностей, відмінностей, закономірностей та тенденцій. У дослідженнях комп'ютерно-технічної експертизи порівняльний метод може бути використаний для оцінки та порівняння різних методологій, інструментів або методів для їх ефективності, ефективності та застосовності в конкретних контекстах. ***Індуктивний метод*** передбачає міркування від конкретних спостережень або прикладів до більш широких узагальнень або теорій. Він передбачає збір та аналіз даних для виявлення закономірностей та формулювання гіпотез. Навпаки, дедуктивний метод передбачає міркування від загальних принципів або теорій до конкретних висновків. У комп'ютерних криміналістичних дослідженнях можуть використовуватися як індуктивні, так і дедуктивні методи. Наприклад, індуктивні методи можуть бути використані для отримання висновків з емпіричних даних, тоді як дедуктивні методи можуть бути використані для перевірки та підтвердження існуючих теорій або гіпотез. ***Метод критичного аналізу*** передбачає систематичну та сувору оцінку, опитування та вивчення ідей, теорій чи доказів. Вона передбачає виявлення сильних і слабких сторін, невідповідностей і упереджень в існуючих знаннях або аргументах. У дослідженнях комп'ютерної криміналістики метод критичного аналізу може бути застосований для оцінки валідності та надійності методологій, інструментів або результатів досліджень. ***Описовий метод*** включає опис, документування та аналіз явищ або ситуацій, як вони природно відбуваються. У дослідженнях комп'ютерно-технічної експертизи описовий метод може бути використаний для

вивчення реальних випадків, дослідження використання конкретних інструментів або методологій або документування найкращих практик у цій галузі. Цей метод забезпечує всебічне розуміння предмета і служить основою для подальшого аналізу і дослідження. В цьому досліджені автором можуть використовуватися ці методи окремо або в комбінації, залежно від конкретних питань дослідження, цілей та наявних даних. Кожен метод сприяє всебічному і ретельному дослідженню предмета.

Теоретичне значення: дослідження має значну теоретичну цінність у кількох напрямках. По-перше, це сприяє існуючому масиву знань, досліджуючи та аналізуючи теоретичні основи комп'ютерно-технічної експертизи. Це дослідження заглиблюється в концепції, принципи та методології, які лежать в основі галузі, забезпечуючи всебічне розуміння предмета. По-друге, дослідження вивчає та оцінює існуючі методи та інструменти, що використовуються в комп'ютерних криміналістичних розслідуваннях. Аналізуючи їх сильні сторони, обмеження та ефективність, дослідження сприяє розробці більш надійної та надійної основи для проведення комп'ютерної криміналістики. Це допомагає встановити найкращі практики та рекомендації для слідчих, підвищуючи загальну якість судових експертиз.

Практична значимість цього дослідження проявляється в його впливі на комп'ютерно-технічної експертизи. Досліджуючи та розробляючи вдосконалені методи та інструменти, дослідження підвищує ефективність, точність та результативність збору, аналізу та збереження цифрових доказів. Це безпосередньо приносить користь правоохоронним органам, фахівцям з кібербезпеки та практикуючим юристам. Крім того, розробка власного програмного забезпечення для проведення базових комп'ютерних експертиз має практичні наслідки для слідчих. Таке програмне забезпечення може впорядкувати та автоматизувати певні аспекти криміналістичного процесу, скоротивши час та зусилля, необхідні для аналізу та формування достовірних звітів. Це сприяє швидшому прийняттю рішень, полегшує своєчасне реагування на кіберінциденти та збільшує шанси на успішне судове переслідування.

Наукова новизна: дослідження приносить наукову новизну в галузь кількома шляхами. По-перше, він досліджує нові виклики та досягнення у сфері кіберзлочинності, цифрових технологій та криміналістичних методологій. Розглядаючи ці нові аспекти, дослідження пропонує нові ідеї та перспективи, рухаючи галузь вперед. Крім того, розробка власного програмного забезпечення, спеціально розробленого для базових комп'ютерних судових експертиз, впроваджує інноваційні рішення в цій галузі. Це програмне забезпечення, адаптоване для задоволення унікальних потреб та викликів цифрових розслідувань, являє собою новий внесок, який може підвищити загальну ефективність та результативність комп'ютерної криміналістики. Крім того, всебічний аналіз дослідження та критична оцінка існуючих методів та інструментів сприяють науковій літературі. Виявляючи прогалини, слабкі місця та області для вдосконалення, дослідження надихає на подальші дослідження та досягнення в галузі комп'ютерної криміналістики, сприяючи безперервному циклу наукових інновацій. Таким чином, дослідження має теоретичне значення шляхом просування теоретичних основ комп’ютерно-технічної експертизи та оцінки існуючих методологій. Його практичне значення полягає в підвищенні ефективності та точності аналізу цифрових доказів, допомагаючи слідчим у реальних справах. Наукова новизна демонструється через дослідження нових проблем, розробку невільного програмного забезпечення та критичну оцінку існуючих методів, що рухає область комп'ютерної криміналістики вперед.

Мета, предмет та завдання сформували ***структуру роботи.*** Робота складається із вступу, трьох розділів, списку використаних джерел та літератури. Обсяг роботи становить 91 сторінок, також опрацьовано та введено в обсяг курсової роботи 29 позицій використаних джерел та літератури, 1 додаток.

РОЗДІЛ І ТЕОРЕТИЧНИЙ АНАЛІЗ МЕТОДІВ ТА ІНСТРУМЕНТІВ КОМП'ЮТЕРНО-ТЕХНІЧНОЇ ЕКСПЕРТИЗИ

1.1. Аналіз правової основи проведення комп'ютерно-технічної експертизи

Комп'ютерно-технічна експертиза є формою інженерно-технічної експертизи, яка включає дослідження технічних характеристик цифрового обладнання (комп'ютери, зберігачі інформації, мобільні телефони і т. д.) та програмного забезпечення з метою отримання фактичних даних, що стосуються злочину або предмета цивільного позову. Ця експертиза допомагає виявити ключові ознаки злочину (інциденту) та створити повну доказову базу шляхом аналізу інформації. Комп'ютерно-технічна експертиза проводиться в рамках кримінальних і цивільних справ. В кримінальних справах експертиза може бути призначена слідчим або судом та здійснюється конкретним експертом або експертною установою. Результатом експертизи є висновок експерта, який служить доказом у справі. У справах експертиза може бути призначена судом, замовлена стороною або проведена за ініціативою нотаріуса. [25]

Об'єктами комп'ютерно-технічної експертизи є апаратне забезпечення (системні блоки комп'ютерів, комплектуючі, сервери, ноутбуки, жорсткі диски, флеш-накопичувачі, модеми, маршрутизатори і т. д.) та програмне забезпечення (комп'ютерні програми, бази даних і т. д.). Призначення комп'ютерно-технічної експертизи базується на процесуальних підставах. Суд призначає експертизу ухвалою, згідно ст. 104 ЦКУ, в якій вказуються підстави для проведення експертизи, питання, на які експерт повинен відповісти, особа (або особи), які здійснюють експертизу, перелік матеріалів, що підлягають дослідженню, та інші дані, необхідні для проведення експертизи. Ухвала про призначення експертизи направляється особам, яким доручено проведення експертизи, а також учасникам справи. Об'єкти та матеріали, що підлягають дослідженню, передаються особі, якій доручено проведення експертизи (ведучому експерту або експертній установі). Експертне дослідження проводиться на підставі договору з експертом або експертною установою, укладеного на письмову заяву

замовника (юридичної або фізичної особи). У договорі зазначаються реквізити замовника, перелік питань, які підлягають вирішенню та об'єкти, що надаються для дослідження. Що ж до комп’ютерно-технічної експертизи, її проведення регламентується низку різних за змістом нормативно-правовими актами: Цивільний процесуальний кодекс України; Кримінальний процесуальний кодекс України; Закон України "Про судову експертизу"; Інструкція про призначення та проведення судових експертиз та експертних досліджень, затверджена наказом Міністерства юстиції України від 08 жовтня 1998 року № 53/5; Науково-методичні рекомендації з питань підготовки та призначення судових експертиз та експертних досліджень, затверджена наказом Міністерства юстиції України від 08 жовтня 1998 року № 53/5. [26] [27] [28] [29].

Види комп'ютерно-технічної експертизи:

- Програмно-комп'ютерна експертиза: цей тип експертизи зосереджений на вивченні та аналізі комп'ютерного програмного забезпечення. Вона включає дослідження функціональності, структури та поведінки комп'ютерних програм, а також виявлення будь-яких потенційних вразливостей або порушень безпеки. Експертиза зазвичай використовується у випадках, пов'язаних з порушенням авторських прав на програмне забезпечення, суперечками щодо розробки програмного забезпечення або підозрою на шкідливу діяльність із програмним забезпеченням.
- Інформаційно-комп'ютерна експертиза передбачає вивчення цифрової інформації, що зберігається на різних комп'ютерних системах і пристроях, та включає пошук, аналіз та інтерпретацію даних для визначення їх релевантності та значущості для конкретного випадку. Інформаційно-комп'ютерні експертизи часто використовується в цифровій криміналістиці, розслідуваннях кіберзлочинів та справах, пов'язаних із витоком даних або крадіжкою інтелектуальної власності.
- Апаратно-комп'ютерна експертиза зосереджена на вивченні компонентів та пристроїв комп'ютерного обладнання та включає оцінку технічних

характеристик, конфігурації, функціональності та фізичного стану апаратного обладнання, такого як комп'ютери, сервери, пристрої зберігання даних та периферійні пристрої. Експертиза зазвичай застосовується у випадках, пов'язаних з несправностями обладнання, пошкодженням обладнання або спорами про відповідальність за якість продукції.

- Комп'ютерно-мережева експертиза стосується вивчення комп'ютерних мереж та питань, пов'язаних з мережами. Вона включає в себе аналіз мережевих архітектур, конфігурацій, протоколів і заходів безпеки для оцінки їх цілісності, продуктивності і вразливостей. Цей досвід часто використовується у випадках, пов'язаних із вторгненнями в мережу, порушеннями даних, суперечками щодо мережевої інфраструктури або оцінкою ефективності мережі.

Відповідно до п. 13.1 Інструкції про призначення і проведення судових експертиз та експертних досліджень завданнями комп'ютерно-технічної експертизи є:

- Визначення робочого стану комп'ютерно-технічних пристроїв: це передбачає оцінку робочого стану та функціональності комп'ютерної техніки, периферійних пристроїв та інших супутніх пристроїв. Експерт вивчає фізичні компоненти, з'єднання та конфігурації, щоб визначити, чи знаходяться вони в належному робочому стані.
- Встановлення обставин, пов'язаних з використанням комп'ютерно-технічних пристроїв, інформації та програмного забезпечення: експерт досліджує та аналізує діяльність, дії чи події, пов'язані з використанням комп'ютерних систем, цифрової інформації та програмного забезпечення. Вони перевіряють файли журналів, системні записи, дії користувачів та інші відповідні дані, щоб зрозуміти контекст і часову шкалу подій.
- Ідентифікація інформації та програмного забезпечення, що зберігається на комп'ютерних носіях: експерт проводить ретельну перевірку пристроїв зберігання даних комп'ютера, таких як жорсткі диски, твердотільні накопичувачі, USB-накопичувачі або інші носії. Вони отримують та

аналізують дані, що зберігаються на цих пристроях, включаючи файли, папки, бази даних та встановлення програмного забезпечення, щоб витягти відповідну інформацію чи докази.

- Перевірка відповідності програмних продуктів конкретним версіям або вимогам до розробки: експерт оцінює програмне забезпечення та продукти, щоб визначити їх відповідність певним версіям або специфікаціям розробки. Це передбачає вивчення коду, структури, функцій та функціональності програмного забезпечення, щоб підтвердити його відповідність заданим стандартам або вимогам.

Ці завдання мають важливе значення в комп'ютерно-технічній експертизі та сприяють всебічному аналізу та розумінню цифрових систем, інформації та програмного забезпечення в різних юридичних та слідчих контекстах. Інструкція про призначення і проведення судових експертиз та експертних досліджень містить орієнтовний перелік питань, які можуть бути поставлені експерту при проведенні комп'ютерно-технічної експертизи. Хоча цей перелік не є вичерпним, важливо зазначити, що відповідно до частини 6 статті 103 Цивільного процесуального кодексу питання, поставлені експерту, та його висновки не повинні виходити за межі спеціальних знань експерта. При необхідності може бути залучений фахівець, який забезпечить правильну постановку питань. Типові питання, які вирішуються комп’ютерно-технічною експертизою є:

- Чи міститься на даному носії інформація стосовно (зазначити, яка інформація цікавить) і у якому вигляді?
- Чи містить носій досліджуваного комп'ютера інформацію про певні (зазначити, які саме) дії користувача?
- Чи піддавався досліджуваний накопичувач певним процедурам з метою знищення інформації?
- Чи могла бути створена зазначена інформація на цьому комп'ютері чи вона перенесена з іншого носія?

- Яким чином інформація (зазначити, яка саме) перенесена до досліджуваного комп'ютера (носія)?
- Яка технологія та хронологія створення електронного документа (зазначити електронний документ та певний зміст)?
- Які атрибути (час друку, редагування, створення, видалення тощо) файлів, що містять інформацію стосовно... (зазначити зміст)?
- Чи містить накопичувач інформації досліджуваного комп'ютера певне (зазначити, яке саме - встановлене, не встановлене) програмне забезпечення?
- Які функціональні несправності мають дане комп'ютерне обладнання або його окремі складові та пристрої і як ці несправності впливають на роботу обладнання в цілому?
- Чи можливо виконання певних дій за допомогою даного програмного продукту?
- Чи можливе вирішення певного завдання за допомогою даного програмного продукту?
- Чи реалізовані у даному програмному продукті (програмному коді) функції, передбачені технічним завданням на його розробку?

Строк проведення експертизи визначається керівником експертної установи (або заступником керівника, чи керівником структурного підрозділу) і не повинен перевищувати 90 календарних днів. У разі значного навантаження на експерта (за наявності у нього більше десяти супутніх експертиз, включаючи комісії) більші розумні строки встановлюються за письмовою домовленістю з органом (особою), який (яка) призначив експертизу (залучив експерта), після завчасного вивчення експертом наданих матеріалів. Термін вивчення експертом матеріалів не повинен перевищувати п'ятнадцяти робочих днів. У разі відмови органу(особи), що призначає, погодитися із запропонованими розумними строками проведення експертизи, матеріали справи повертаються з пропозицією про призначення експертизи іншим суб'єктам судово-експертної діяльності,

визначеним статтею 7 Закону України «Про судову експертизу». У разі невиконання запитів експерта про надання додаткових матеріалів, оплати експертизи не здійснюється протягом 45 календарних днів з дня звернення відповідно до чинного законодавства, не забезпечується прибуття експерта, відсутній безперешкодний доступ до об'єкта експертизи або не забезпечуються належні умови праці (перешкоджання сторонам, які беруть участь у справі, в огляді об'єкта), Матеріали справи повертаються органу (особі), що призначає, з поясненням обґрунтованих причин неможливості проведення експертизи. Строк проведення експертизи починається з робочого дня, наступного за днем отримання матеріалів експертною установою, і закінчується в день підготовки висновку експерта (повідомлення про неможливість надати висновок). Якщо закінчення встановленого терміну проведення експертизи припадає на неробочий день, датою закінчення вважається наступний робочий день. Строки проведення експертизи не включають час, необхідний для виконання запитів експерта, усунення недоліків, допущених органом (особою), що призначає(ла), який (яка) призначив експертизу (залучив експерта).

Отож, законодавство України передбачає чітке та якісне регулювання проведення комп'ютерно-технічної експертизи, забезпечення дотримання процедур, стандартів та вимог. Воно встановлює рамки для кваліфікації експертів, процедур призначення, неупередженості та незалежності. Законодавство підкреслює точність, встановлює стандарти оцінювання. Це підвищує достовірність та достовірність експертних висновків, сприяючи ефективному правосуддю та довірі до судової системи.

1.2. Загальна характеристика висновку експерта та його ролі при проведенні комп'ютерно-технічної експертизи

Висновок експерта служить засобом доказування, а його зміст і форма мають істотне значення для встановлення його доказового значення. Проміжні факти, виявлені під час експертизи, становлять фактичну основу для висновків експерта, але не можуть бути визнані доказами самі по собі. Оцінка висновку експерта та його місце в системі доказів визначається судом, керуючись правилами оцінки, встановленими процесуальним законом. Правова природа висновку експерта відрізняє його від письмового доказу, так як експерт надає професійну оцінку знову отриманих фактичних даних. Висновок експерта може бути як прямим, так і непрямим доказом, в залежності від інформації, яку він містить. В цілому висновок експерта є рівним серед інших засобів доказування і не має особливої юридичної сили. Його правова природа визначається специфікою формування інформації та дотриманням цивільних процесуальних вимог. Наведені вище факти демонструють, що загальні та всеосяжні ознаки можуть бути застосовані до фактичних даних, які вважаються зафіксованими експертами. Однак фактичні дані володіють унікальними характеристиками, які відрізняють їх в рамках системи доказування і висновок експерта в рамках засобів доказування. Таким чином, висновок експерта можна визначити як засіб доказування, де спеціалізована інформація, отримана конкретними особами, юридично оформлена у процесуальній формі, визначеній законом Хоча в наданому визначенні відсутня конкретна категоризація ознак для висновку експерта як засобу доказування, важливо зрозуміти конкретний прояв цих ознак, щоб зрозуміти суть висновку експерта. Висновок експерта служить спеціалізованою формою закріплення інформації, маючи на увазі, що процес її отримання повинен відповідати певному процесуальному порядку. Відмінні ознаки висновку експерта можуть бути виявлені через внутрішнє уточнення інформації і способу її формулювання. Ці аспекти є переважно унікальними і випливають із загального визначення судової експертизи, яке передбачає

дослідження конкретних фактичних обставин на підставі рішення суду. Особа, яка проводить іспит, відома як експерт, володіє спеціалізованими знаннями в таких галузях, як наука, мистецтво, техніка або ремесла. Результатом їх роботи є новий засіб доказування, відомий як експертний висновок. Тому при дослідженні висновку експерта важливо наголошувати як на його змісті, який містить обґрунтований висновок, отриманий на основі проведених досліджень та професійної оцінки фактів, так і на його формі як офіційного документа. Обидва компоненти мають однакове значення при визначенні доказового значення висновку експерта. Якщо суд визнає інформацію, що міститься у висновку експерта, недостовірною, вона не може розглядатися як доказ. Аналогічно, навіть якщо висновок експерта має обґрунтований та об'єктивний зміст, він не може бути визнаний засобом доказування у разі невідповідності необхідної форми, наприклад, неповних пояснень, відсутності письмового висновку або відсутності підпису експерта. Крім вищезазначених моментів, аналіз поняття висновку експерта тягне за собою розгляд його як доказу, отриманого в результаті призначеної судом судової експертизи, проведеної в установленому процесуальному порядку. Невідповідність цим вимогам робить висновок експерта неадекватним як засіб доказування. [23]

Щодо доказового значення висновку експерта в рамках процесуальної науки немає єдиної думки щодо того, чи спирається воно виключно на висновки або включає проміжні факти, встановлені під час експертизи. Багато хто стверджує, що цінність судово-медичних доказів включає в себе як висновок експерта, так і проміжні факти. Важливо визнати, що думка експерта, як самостійний засіб доказування, володіє певними характеристиками. Експерт письмово повідомляє суд про фактичні дані, встановлені в процесі проведення експертизи, використовуючи свої спеціальні знання, виходячи із завдання, сформульованого в рішенні про призначення експертизи. Інші способи доказування, визначені законом, не включають судження про факти як істотний елемент. Наприклад, показання свідків надають інформацію про спостережувані

факти, але їм не вистачає оціночної точки зору експерта. Тому саме висновок експерта має доказове значення, оскільки включає оцінки фактичних даних.

Отже, проміжні факти служать фактичною основою для висновку експерта. Однак ці факти, окремо від висновків експерта щодо них, не можуть бути визнані доказами. Суд може тільки вивчити і оцінити висновок експерта в цілому. Відхилення від загальних висновків не можуть бути використані судом як доказ, оскільки їх тлумачення вимагає спеціальних знань, які можуть стирати межі між процесуальною роллю суду та функцією експерта. Оскільки експерт отримує різну інформацію в процесі дослідження, не вся вона може вважатися судовим доказом. Для вирішення цього питання слід користуватися логіко-юридичним критерієм, що закладений у дефініції частини першої ст. 57 ЦПК України, тобто здатністю інформації відображати властивості конкретних обставин, що мають певне юридичне значення (а саме значення доказового факту). Проміжні факти такою здатністю не наділені, вони відображають природу експертизи як дослідження.

Судова експертиза відіграє вирішальну роль у визначенні цінності кримінальної справи шляхом проведення ретельного та неупередженого розслідування. Судові експертизи можна класифікувати за різними критеріями. Ось деякі класифікації: Рівні судової експертизи: виділяють чотири рівні - клас, рід, вид і різновид. Клас представляє спільноту знань та об'єктів, досліджених у цій галузі (наприклад, судово-медична експертиза, судово-медична та психофізіологічна експертиза, сільськогосподарська експертиза). Тип відображає предмет, об'єкт і методи дослідження в рамках судових експертиз (наприклад, почерк, ідентифікація автора, технічна експертиза документів, трасологічні). Вид включає специфічні елементи типу, що розрізняють специфіку його предмета та методи дослідження (наприклад, дослідження деталей документа, дослідження матеріалів документа). Підтипи - це додаткові підрозділи всередині типу, зосереджені на конкретних завданнях і методах дослідження (наприклад, вивчення відбитків печаток, експертиза документів, отриманих за допомогою копіювального обладнання). Природа предмета або

галузі знань: судові експертизи можна класифікувати за такими категоріями, як судові експертизи, медичні та психофізіологічні експертизи, інженерні та транспортні експертизи. Різновиди, засновані на розподілі: судові експертизи можуть бути традиційними (наприклад, дактилоскопічні, почеркознавчі, судово-балістичні) або нетрадиційними (наприклад, експертиза матеріалів, речовин і виробів, експертиза ґрунту, фоноскопічна, фонографічна, графологічна). Сутність завдань і методів дослідження: судові експертизи можна класифікувати як ідентифікаційні (встановлення індивідуальної ідентичності), класифікаційні (визначення групової приналежності, виду, роду), діагностичні (оцінка властивостей і можливостей об'єкта), ситуаційні (розгляд конкретних ситуацій і умов сцени) або змішані (поєднання декількох підходів). Порядок проведення експертизи: експертизи можуть бути первинними (огляд вперше) або повторними (проводитися, коли початковий висновок ставиться під сумнів або суперечить іншим матеріалам справи). Обсяг досліджень: судові експертизи можна класифікувати як базові (охоплюють основні питання, що потребують експертного аналізу) або додаткові (розглядають питання, не включені до первинної експертизи або доповнюють і уточнюють її висновки). Склад експертів: експертизи можуть проводитися одним експертом або комісією експертів з однієї галузі знань. Комісійні експертизи використовуються для вирішення складних питань або коли існують різні думки. Досягаються спільні висновки або готуються окремі висновки, коли згоди досягти не вдається.

1.3. Аналіз методів та інструментів комп'ютерно-технічної експертизи

Комп'ютерно-технічна експертиза або ж комп'ютерна криміналістика відіграє вирішальну роль у забезпеченні цілісності цифрових доказів у системі цивільного та кримінального правосуддя. З повсюдним використанням комп'ютерів та цифрових пристроїв у різних аспектах життя значення цифрових доказів та криміналістичного процесу, що використовується для їх збору, збереження та розслідування, зросло у вирішенні юридичних справ та вирішенні кримінальних справ. Більшість людей рідко стикаються з великою інформацією, яку збирають сучасні пристрої. Наприклад, автомобільні комп'ютери безшумно записують дані про гальмування, перемикання передач і зміни швидкості без відома водія. Однак ця інформація може бути життєво важливою в судових розслідуваннях, і комп'ютерна криміналістика часто допомагає у виявленні та захисті таких даних. Цифрові докази виявляються цінними не тільки для вирішення кіберзлочинів, таких як крадіжка даних, зломи мережі та незаконна діяльність в Інтернеті, але й для розкриття злочинів фізичного світу, таких як крадіжки зі зломом, напади, нещасні випадки та вбивства. Підприємства використовують комплексне управління даними, управління даними та стратегії мережевої безпеки для захисту конфіденційної інформації. Збереження добре керованих та безпечних даних полегшує криміналістичний процес у разі розслідування. Підприємства також покладаються на комп'ютерну криміналістику для відстеження інформації, пов'язаної з системними або мережевими компрометаціями, що дозволяє ідентифікувати та переслідувати кіберзлочинців. Крім того, експерти та процедури цифрової криміналістики допомагають підприємствам у відновленні даних після системних або мережевих збоїв, спричинених стихійними лихами або іншими катастрофами. Оскільки суспільство стає все більш залежним від цифрових технологій для повсякденних функцій, поширеність кіберзлочинності продовжує зростати. Отже, фахівці з комп'ютерної криміналістики більше не є єдиним авторитетом у цій галузі.

Правоохоронні органи, такі як у Великобританії, впроваджують комп'ютерну криміналістику для ефективної боротьби зі зростаючим рівнем кіберзлочинності.

Існують різні види комп'ютерних криміналістичних розслідувань, кожен з яких зосереджений на певному аспекті інформаційних технологій. Деякі з основних типів включають:

- Криміналістика баз даних: вивчення даних та пов'язаних з ними метаданих, що зберігаються в базах даних.
- Криміналістика електронної пошти: відновлення та аналіз повідомлень електронної пошти та іншої інформації, присутньої на платформах електронної пошти, наприклад, розкладів та контактів.
- Криміналістика шкідливих програм: аналіз коду для виявлення потенційного шкідливого програмного забезпечення та аналізу їх корисного навантаження. Такі програми можуть включати трояни, програми-вимагачі або різні типи вірусів.
- Криміналістика пам'яті: збір інформації, що зберігається в оперативній пам'яті (ОЗП) комп'ютера та кеші.
- Мобільна криміналістика: вивчення мобільних пристроїв для вилучення та аналізу інформації, яку вони містять, включаючи контакти, вхідні та вихідні текстові повідомлення, зображення та відеофайли.
- Мережева криміналістика: пошук доказів шляхом моніторингу мережевого трафіку за допомогою таких інструментів, як брандмауери або системи виявлення вторгнень.

Кожен тип комп'ютерної криміналістики включає спеціалізовані методи та інструменти для вилучення, аналізу та інтерпретації цифрових доказів, пов'язаних з цією конкретною областю розслідування. Ця практика має вирішальне значення для виявлення та подання доказів у судових процесах. [9]

Комп'ютерна криміналістика дотримується встановлених процедур, які можуть відрізнятися залежно від контексту розслідування, залученого пристрою та шуканої інформації. Як правило, процес включає три основні етапи:

- Збір даних: інформація, що зберігається в електронному вигляді, збирається, забезпечуючи її цілісність. Зазвичай це включає фізичну ізоляцію пристрою, щоб запобігти випадковому забрудненню або несанкціонованому втручанню. Експерти створюють криміналістичне зображення, цифрову копію зберігання даних пристрою, надійно зберігаючи при цьому оригінальний пристрій. Розслідування ведеться за копією. У деяких випадках загальнодоступна інформація, така як публікації в соціальних мережах або записи публічних транзакцій, також може бути використана для криміналістичних цілей.
- Аналіз: слідчі вивчають цифрові копії носіїв у контрольованому середовищі, щоб отримати відповідну інформацію для справи. Різні інструменти, такі як програмне забезпечення для розсічення жорсткого диска та аналізатори мережевих протоколів, використовуються для допомоги в цьому процесі. Такі методи, як тремтіння миші, можуть бути використані для запобігання переходу комп'ютера в сплячий режим і втрати нестабільних даних пам'яті.
- Презентація: експерти представляють свої висновки в суді, де судді або присяжні оцінюють докази, щоб визначити результат справи. У ситуаціях, пов'язаних із відновленням даних, експерти представляють інформацію, яку вони змогли отримати з скомпрометованих систем.

Комп'ютерні криміналістичні розслідування часто включають використання декількох інструментів і методів для перевірки отриманих результатів. Наприклад, дослідник «Лабораторії Касперського» розробив криміналістичний інструмент з відкритим вихідним кодом, який дозволяє віддалено збирати докази шкідливого програмного забезпечення без шкоди для цілісності системи. [8]

Що ж стосується методів, експерти використовують різні методи та власні криміналістичні програми для вивчення інформації пристрою. Вони шукають приховані папки і незайнятий дисковий простір для копій видалених, зашифрованих або пошкоджених файлів. Будь-які докази, знайдені на цифровій

копії, ретельно документуються у звіті про пошук та перевіряються оригінальним пристроєм при підготовці до судового розгляду, який включає відкриття, відкладення або фактичний судовий процес. Комп'ютерні криміналістичні розслідування використовують комбінацію методів та експертних знань. Деякі поширені методи включають наступне:

- ***Зворотна стеганографія.*** Стеганографія є поширеною тактикою, яка використовується для приховування даних всередині будь-якого типу цифрового файлу, повідомлення або потоку даних. Комп'ютерні криміналісти скасовують спробу стеганографії, аналізуючи хешування даних, які містить відповідний файл. Якщо кіберзлочинець приховує важливу інформацію всередині зображення або іншого цифрового файлу, це може виглядати однаково до і після для непідготовленого ока, але основний хеш або рядок даних, який представляє зображення, зміниться.
- ***Стохастична криміналістика***. Тут дослідники аналізують і реконструюють цифрову активність без використання цифрових артефактів. Артефакти - це ненавмисні зміни даних, які відбуваються в результаті цифрових процесів. Артефакти включають підказки, пов'язані з цифровим злочином, такі як зміна атрибутів файлів під час крадіжки даних. Стохастична криміналістика часто використовується в розслідуваннях витоку даних, де зловмисник вважається інсайдером, який може не залишити після себе цифрових артефактів.
- ***Крос-драйвовий аналіз***. Цей метод співвідносить і перехресно посилається на інформацію, знайдену на декількох комп'ютерних дисках, для пошуку, аналізу та збереження інформації, що має відношення до розслідування. Події, які викликають підозру, порівнюються з інформацією про інші диски, щоб знайти схожість і надати контекст, також відоме як виявлення аномалій.
- ***Живий аналіз.*** За допомогою цього методу комп'ютер аналізується зсередини ОС під час роботи комп'ютера або пристрою, використовуючи системні інструменти на комп'ютері. При аналізі розглядаються

енергонезалежні дані, які часто зберігаються в кеші або оперативній пам'яті. Багато інструментів, що використовуються для вилучення нестабільних даних, вимагають, щоб комп'ютер знаходився в криміналістичній лабораторії для підтримки легітимності ланцюжка доказів.

- ***Відновлення видалених файлів.*** Цей метод передбачає пошук в комп'ютерній системі і пам'яті фрагментів файлів, які були частково видалені в одному місці, але залишають сліди в іншому місці на машині. Це іноді називають різьбленням по файлах або різьбленням по даних.

Комп'ютерна криміналістика використовується як доказ правоохоронними органами та в кримінальному та цивільному праві з 1980-х років. Деякі помітні випадки включають наступне: Крадіжка комерційної таємниці Apple. Інженер на ім'я Сяолан Чжан з підрозділу автономних автомобілів Apple оголосив про свою відставку і сказав, що повернеться до Китаю, щоб піклуватися про свою літню матір. Він сказав своєму менеджеру, що планує працювати у виробника електронних автомобілів у Китаї, що викликало підозри. Згідно з заявою Федерального бюро розслідувань (ФБР), команда безпеки Apple переглянула діяльність Чжана в мережі компанії і виявила, що за кілька днів до своєї відставки він завантажував комерційну таємницю з конфіденційних баз даних компанії, до яких мав доступ. Йому було пред'явлено звинувачення від ФБР у 2018 році.

В одному з найбільш часто згадуваних скандалів з бухгалтерським шахрайством Enron, американська компанія з енергетики, сировинних товарів і послуг, неправдиво повідомила про мільярди доларів доходу, перш ніж збанкрутувати в 2001 році, завдавши фінансової шкоди багатьом співробітникам та іншим людям, які інвестували в компанію. Комп'ютерні криміналісти вивчили терабайти даних, щоб зрозуміти складну схему шахрайства. Скандал став значним фактором прийняття Закону Сарбейнса-Окслі 2002 року, який встановив нові вимоги до бухгалтерського обліку для публічних компаній. Компанія оголосила про банкрутство в 2001 році. Крадіжка комерційної таємниці Google. Ентоні Скотт Левандовські, колишній керівник Uber і Google, був звинувачений у 33 пунктах звинувачення в крадіжці комерційної таємниці в

2019 році. З 2009 по 2016 рік Лєвандовскі працював у програмі самокерованих автомобілів Google, де завантажував тисячі файлів, пов'язаних із програмою, із захищеного паролем корпоративного сервера. Він пішов з Google і створив Otto, компанію з виробництва безпілотних вантажівок, яку Uber купив у 2016 році, повідомляє The New York Times. Лєвандовскі визнав себе винним за одним пунктом звинувачення в крадіжці комерційної таємниці і був засуджений до 18 місяців тюремного ув'язнення і 851 499 доларів штрафів і реституції. Лєвандовскі отримав президентське помилування в січні 2021 року. Ларрі Томас. Томас застрелив Ріто Ламаса-Хуареса в 2016 році Пізніше Томас був засуджений за допомогою сотень постів у Facebook, які він зробив під фальшивим ім'ям Slaughtaboi Larro. Один з постів містив фотографію, на якій він одягнений у браслет, який був знайдений на місці злочину. Майкл Джексон. Слідчі використовували метадані та медичні документи з iPhone лікаря Майкла Джексона, які показали, що лікар, Конрад Мюррей, прописав смертельну кількість ліків Джексону, який помер у 2009 році. Мікайла Манн. Манн втопила свою новонароджену дитину у ванні своєї кімнати гуртожитку Манчестерського університету в 2016 році. Слідчі знайшли пошукові запити на її комп'ютері, що містять фразу «аборт вдома», яка використовувалася для її засудження.

Комп'ютерна криміналістика передбачає дослідження та аналіз цифрових пристроїв та електронних доказів для юридичних цілей. Процес включає отримання криміналістичного зображення носія інформації, збереження його цілісності, відновлення та реконструкцію даних, аналіз отриманих даних на предмет відповідних доказів, проведення аналізу часової шкали та зв'язків, злом паролів, якщо це необхідно, підготовку вичерпних звітів та надання експертних свідчень у суді. Ця сфера постійно адаптується до досягнень у галузі технологій для ефективного вилучення та аналізу електронних доказів, зберігаючи їх цілісність та допустимість у судочинстві.

Висновки до розділу І

Аналіз правової основи комп'ютерного криміналістичного аналізу має вирішальне значення для забезпечення того, щоб процес проводився в межах чинних законів та нормативних актів. Що ж до комп’ютерно-технічної експертизи, її проведення регламентується низку різних за змістом нормативно-правовими актами: Цивільний процесуальний кодекс України; Кримінальний процесуальний кодекс України; Закон України "Про судову експертизу"; Інструкція про призначення та проведення судових експертиз та експертних досліджень, затверджена наказом Міністерства юстиції України від 08 жовтня 1998 року № 53/5; Науково-методичні рекомендації з питань підготовки та призначення судових експертиз та експертних досліджень, затверджена наказом Міністерства юстиції України від 08 жовтня 1998 року № 53/5.

Комп’ютерно-технічна експертиза передбачає розслідування та аналіз цифрових пристроїв та електронних доказів для збору інформації для юридичних цілей. Він охоплює цілий ряд методів та інструментів, що використовуються для вилучення, збереження та аналізу даних з комп'ютерів, ноутбуків, мобільних пристроїв та інших цифрових носіїв інформації. Існує безліч методів та інструментів, які зазвичай використовуються в комп’ютерній експертизі.

Першим кроком у комп'ютерному криміналістичному аналізі є отримання криміналістичного зображення або покрокової копії досліджуваного носія інформації. Це гарантує, що вихідні дані залишаються незмінними, поки дослідники працюють зі скопійованими даними. Після отримання криміналістичного зображення дуже важливо зберегти його цілісність та запобігти будь-яким змінам або підробкам. Це передбачає надійне зберігання зображення та створення ланцюга зберігання для підтримки допустимості доказів у суді. Комп'ютерні криміналісти-аналітики використовують спеціалізовані інструменти та методи для відновлення видалених, зашифрованих або пошкоджених файлів та відновлення їх до початкового стану. Цей процес включає в себе пошук прихованих папок, аналіз незайнятого дискового простору і відновлення фрагментів даних. Далі проводиться аналіз. На цьому етапі слідчі

вивчають отримані дані, щоб виявити докази, що мають відношення до справи. Вони можуть використовувати різні інструменти та методи для пошуку ключових слів, типів файлів та конкретних моделей поведінки. Різьблення по даних часто використовується для вилучення файлів з незайнятого простору або фрагментованих областей носіїв інформації. Експерти створюють хронологічну хронологічну шкалу подій на основі часових позначок та метаданих, пов'язаних із файлами та системними діями. Цей аналіз допомагає встановити послідовність дій і виявити будь-які аномалії або підозрілі дії. Інструменти аналізу посилань використовуються для візуалізації та аналізу відносин між сутностями, такими як фізичні особи, IP-адреси, веб-сайти та файли. Це допомагає визначити зв'язки, асоціації та закономірності, які можуть мати вирішальне значення для розуміння взаємозв'язків між різними елементами справи. Якщо зустрічаються зашифровані файли або захищені паролем пристрої, комп'ютерні криміналісти можуть використовувати методи злому паролів, щоб отримати доступ до даних. Це передбачає використання спеціалізованого програмного забезпечення та методів для спроби отримати або обійти паролі. Протягом усього процесу аналізу ведеться детальна документація для запису зроблених кроків, висновків та висновків. Складається комплексний висновок, в якому чітко і організовано викладаються докази, придатні для представлення в суді. Для забезпечення цілісності та допустимості доказів можуть бути залучені комп'ютерні судово-медичні експерти для надання свідчень у суді. Вони надають експертні висновки, пояснюють використовувані методи та представляють висновки, щоб допомогти судді та присяжним зрозуміти технічні аспекти справи. Важливо відзначити, що комп’ютерно-технічна експертиза є динамічною галуззю, і використовувані методи та інструменти можуть змінюватися залежно від конкретного випадку та характеру технології, що розвивається. Судові експерти постійно адаптуються до нових викликів та розробок у цифрових пристроях та сховищах даних для ефективного дослідження та аналізу електронних доказів.

РОЗДІЛ ІІ. ЗАГАЛЬНІ ЗАСАДИ СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОМП'ЮТЕРНО-ТЕХНІЧНОЇ ЕКСПЕРТИЗИ

2.1. Необхідність створення власного програмного забезпечення для проведення комп'ютерно-технічної експертизи

Необхідність створення програмного забезпечення для комп'ютерної криміналістики виникає через зростаючу залежність від цифрових пристроїв та необхідність розслідування та аналізу цифрових доказів у різних правових та слідчих контекстах. Ось кілька ключових причин для розробки спеціальної програми для комп'ютерної криміналістики: з поширенням цифрових пристроїв та кількості згенерованих даних зростає попит на спеціалізовані інструменти для ефективного аналізу та інтерпретації цифрових доказів. Спеціальна програма може забезпечити необхідні функції та функціональні можливості для вирішення складних завдань, пов'язаних з комп'ютерною криміналістикою. Автоматизація та ефективність: ручна перевірка цифрових доказів може зайняти багато часу та призвести до помилок. Розробка програмного забезпечення дозволяє автоматизувати повторювані завдання, оптимізувати процес розслідування та підвищити загальну ефективність. Автоматизовані процеси можуть допомогти обробляти великі обсяги даних і ефективніше виконувати такі завдання, як відновлення даних, пошук за ключовими словами та аналіз метаданих. Стандартизація та послідовність: комп'ютерна криміналістика передбачає дотримання встановлених процедур та керівних принципів для забезпечення цілісності та допустимості доказів у судовому процесі. Спеціальна програмна програма може забезпечити стандартизацію, включаючи заздалегідь визначені криміналістичні протоколи, забезпечуючи послідовні та надійні результати для різних розслідувань та експертів. Розширені можливості аналізу: криміналістичне програмне забезпечення може запропонувати розширені можливості аналізу для виявлення прихованих або видалених даних, виявлення закономірностей та встановлення взаємозв'язків між цифровими артефактами.

Програмне забезпечення може полегшити співпрацю, дозволяючи безпечний обмін доказами, централізоване управління справами та спілкування в режимі реального часу. Крім того, програмне забезпечення може створювати вичерпні звіти з детальними висновками, результатами аналізу та підтверджуючими доказами, забезпечуючи чітку та стислу документацію для юридичних цілей. Цифрові пристрої та технології продовжують швидко розвиватися, представляючи нові виклики для комп'ютерної криміналістики. Розробляючи спеціальну програму, стає легше адаптувати та включати нові технології, формати файлів та криміналістичні методи, щоб бути в курсі мінливого ландшафту цифрових розслідувань. Загалом, створення спеціального програмного забезпечення для комп'ютерної криміналістики допомагає вирішувати унікальні проблеми та вимоги аналізу цифрових доказів, забезпечуючи ефективне та результативне розслідування, зберігаючи цілісність та допустимість доказів у судовому процесі.

Створення власного програмного забезпечення для комп'ютерної криміналістики пропонує для мене різні переваги, адаптовані до конкретних потреб. Його важливість підкреслюють наступні причини:

- Налаштування: налаштування функції та функціональні можливості програмного забезпечення відповідно до конкретних вимог, на відміну від готових рішень. Ця настройка дозволяє зосередитися на найважливіших аспектах комп'ютерної криміналістики.
- Спеціалізований фокус: власне програмне забезпечення дає можливість зосередитись на конкретних областях або типах досліджень, що стосуються моєї роботи, підвищуючи ефективність та досвід у цих конкретних областях.
- Інтеграція з існуючими інструментами: власне програмне забезпечення інтегрується з конкретними інструментами або робочими процесами, що використовуються в криміналістичних експертизах, забезпечуючи плавний та ефективний процес розслідування. Автоматизація передачі та аналізу даних між різними програмними компонентами, заощаджуючи час та зусилля.

- Запатентовані методи або алгоритми: включають та оптимізують власні методи або алгоритми, які забезпечують унікальні переваги в комп'ютерній експертизі, що дає змогу забезпечити ефективне використання спеціалізованих методів для точного та глибокого аналізу.
- Контроль та безпека: можливість мати повний контроль над безпекою та конфіденційністю конфіденційних даних, що беруть участь у комп'ютерних експертизах через впровадження надійних заходів безпеки, протоколів та контролю доступу з урахуванням конкретних вимог, забезпечуючи конфіденційність та цілісність даних протягом усього процесу розслідування.
- Професійний розвиток: створення власної програми дасть можливість отримати цінний досвід навчання та сприяти професійному розвитку та поглибити розуміння технічних аспектів комп'ютерної криміналістики, покращити навички програмування та розвинути експертизу в розробці програмного забезпечення. Цей досвід може створити нові можливості та підвищити довіру до комп'ютерної криміналістики.

Створення власного програмного забезпечення для комп'ютерної криміналістики вимагає значних зусиль та досвіду. Однак він забезпечує індивідуальне рішення, яке точно відповідає конкретним потребам, підвищує ефективність та розширює можливості комп’ютерно-технічної експертизи.

2.2. Основні характеристики та структура власного програмного забезпечення для проведення комп’ютерно-технічної експертизи

Під час розробки власного програмного забезпечення я запланував його структуру та функціонал для успішної комп’ютерно-технічної експертизи. Отож моє програмне забезпечення передбачає набір таких функцій:

- Отримати IP-адресу: функція дозволяє отримати IP-адресу пристрою, підключеного до мережі. IP-адреса – це унікальний ідентифікатор, присвоєний пристрою в мережі, який забезпечує зв'язок між пристроями.
- Отримати MAC (контроль доступу до медіа) Ethernet: функція отримує MAC-адресу інтерфейсу Ethernet пристрою. MAC-адреса - це унікальний ідентифікатор, присвоєний мережевій інтерфейсній карті (NIC) пристрою. Він використовується для зв'язку по локальній мережі.
- Витягти метадані з фотографій: функція дозволяє витягувати метадані з фотографій. Метадані включають таку інформацію, як марка та модель камери, дата та час зйомки, GPS-координати (за наявності) та інші технічні відомості про фотографію.
- Отримати загальну інформацію про ПК: функція надає нам загальну інформацію про комп'ютер, як версія операційної системи, тип процесора та швидкість, обсяг встановленої оперативної пам'яті, дисковий простір та інші технічні характеристики обладнання та програмного забезпечення.
- Геодані точок доступу Wi-Fi: функція отримує дані геолокації, пов'язані з точками доступу Wi-Fi. Якщо на пристрої ввімкнено Wi-Fi, він сканує найближчі точки доступу та збирає таку інформацію, як ім'я мережі (SSID), рівень сигналу та GPS-координати точки доступу (за наявності).
- Запис звуку з мікрофона: функція дозволяє захоплювати звук за допомогою мікрофона пристрою. Це дозволяє записувати звуки, голосові нотатки або будь-який інший аудіовхід, який можна обробити або зберегти для подальшого використання.

- Зберегти дані браузера: функція дозволяє зберегти всі дані з веб-браузера. Може включати закладки, історію веб-перегляду, файли cookie, збережені паролі та іншу інформацію, пов'язану з браузером. Збереження даних браузера дозволяє відновити його пізніше або перенести на інший пристрій.
- Отримати серійні номери ПК: функція отримує серійні номери апаратних компонентів комп'ютера, таких як материнська плата, жорсткий диск або відеокарта. Серійні номери – це унікальні ідентифікатори, присвоєні окремим компонентам, які можуть бути корисними для ідентифікації та керування обладнанням у системі.

Програмний код є скриптом Python, який пропонує інтерфейс командного рядка на основі меню (CLI) з різними функціональними можливостями. Давайте розберемо код і опишемо процес створення. Код починається з імпорту необхідних модулів і пакетів. Функція визначена для відображення параметрів меню користувачеві – print_menu() Потім код визначає кілька функцій, які відповідають параметрам меню: get_external_ip(): надсилає запит до загальнодоступного API, щоб отримати зовнішню IP-адресу користувача та відображає її. Get_mac(): отримує та відображає MAC-адресу користувача (керування доступом до медіа). Get_metadata(): обробляє метадані з фотографій у каталозі "фотографії" за допомогою бібліотеки. Він витягує дату, місце розташування (широту та довготу) та створює файл GeoJSON з назвою "Geo.json" з метаданими.exifread. Get_main_system(): надає системну інформацію, таку як операційна система, ім'я користувача, версія системи, машина, інформація про графічний процесор, відомості про пам'ять, технологічний процес та інформація про диск. Інформація відображається за допомогою бібліотек and.psutilGPUtil. Get_geo_wifi(): отримує доступні мережі Wi-Fi та інформацію про них (SSID, BSSID, канал, потужність сигналу та тип шифрування). Потім він використовує API геолокації для отримання географічних даних на основі BSSID (MAC-адреси) кожної мережі. Get_record_voice(): запис звуку з мікрофона протягом указаного часу та

збереження його як файл WAV. Запис повторюється щохвилини до натискання клавіші "esc". Get_save_cookies(): копіює файли cookie з браузерів Google Chrome, Mozilla Firefox і Microsoft Edge і зберігає їх у каталозі з назвою "SaveCookies". Нарешті, код входить у нескінченний цикл, щоб відобразити меню та обробити введені користувачем дані на основі вибраного параметра. Цей код є лише набором функцій і не включає основну точку входу програми (). Щоб використовувати цей код, зазвичай потрібно додати цю точку входу та викликати функцію для запуску програми.if __name__ == "__main__":print_menu()

Процес створення включав визначення бажаних функціональних можливостей, дослідження та вибір відповідних бібліотек та API, а потім впровадження коду для досягнення намічених цілей. Код демонструє різні методи програмування на Python, такі як робота із зовнішніми API, читання метаданих з файлів, взаємодія з системною інформацією, захоплення аудіо та копіювання файлів. Python – мова програмування, відома своєю читабельністю і простотою. Він має чистий синтаксис з акцентом на читабельність коду, що полегшує його розуміння та обслуговування.

Python легко вивчати, що робить його популярним вибором для початківців. Він поставляється з великою стандартною бібліотекою, яка надає готові модулі для різних завдань, зменшуючи потребу в зовнішніх залежностях. Python є кросплатформним, що означає, що код, написаний на Python, може працювати на різних операційних системах без змін. Він динамічно типізований, що дозволяє гнучко і швидше розвиватися. Python підтримує об'єктно-орієнтоване програмування, дозволяючи створювати багаторазовий і модульний код. Це інтерпретована мова, що означає, що код виконується рядок за рядком без компіляції. Python має активну спільноту, яка розробила численні сторонні бібліотеки та фреймворки, розширюючи свої можливості. Він зазвичай використовується для сценаріїв та завдань автоматизації та пропонує масштабованість та гнучкість як для невеликих сценаріїв, так і для великомасштабних додатків. Саме тому я обрав цю мову для створення ПО.

При написанні коду я виконав наступну послідовність:

- Імпорт необхідних модулів: програма починається з імпорту необхідних модулів, таких як requests, uuid, os, json, datetime, exifread, platform, GPUtil, psutil, subprocess, tabulate, wifi, sounddevice, soundfile, sqlite3, shutil, wmi, signal і keyboard. Ці модулі забезпечують різні функціональні можливості, необхідні для різних завдань.
- Друк меню: функція визначена для відображення параметрів меню користувачеві. Кожен варіант відповідає певному завданню, яке може виконати програма.print_menu()
- Реалізація функцій завдання: окремі функції визначаються для кожного завдання, згаданого в меню. ці функції включають отримання зовнішньої IP-адреси, отримання MAC-адреси, вилучення метаданих з фотографій, отримання загальної інформації про комп'ютер, збір даних геолокації точок доступу Wi-Fi, запис звуку з мікрофона та збереження файлів cookie.
- Основний цикл програми: функція містить цикл, який безперервно відображає меню та пропонує користувачеві вибір. На основі введених користувачем даних викликається відповідна функція завдання. Якщо користувач вибирає варіант «0», цикл розривається, і програма завершується .main()
- Обробка введення даних: в основному циклі вибір користувача отримується за допомогою функції. Потім вибір порівнюється з кожним варіантом за допомогою операторів if-elif-else. Якщо користувач вводить недійсний вибір, відображається відповідне повідомлення .input()

В цілому, ця програма дотримується модульного підходу, при цьому кожне завдання реалізується як окрема функція. Основний цикл гарантує, що програма продовжує працювати, доки користувач не вирішить вийти.

2.3. Інструкція з охорони праці при роботі на персональному комп’ютером

Дотримання рекомендацій з охорони праці під час роботи на комп'ютері має першорядне значення. Ці принципи забезпечують безпеку людей на робочому місці. Дотримуючись інструкцій з охорони праці, потенційні небезпеки та ризики, пов'язані з використанням комп'ютера, можуть бути зведені до мінімуму або усунені. Дотримання рекомендацій з охорони праці допомагає запобігти фізичним та ергономічним проблемам, таким як травми напруги (RSI), напруга очей та порушення опорно-рухового апарату. Правильне налаштування робочого місця, включаючи ергономічні крісла, регульовані столи та правильну поставу, може значно знизити ризик виникнення цих проблем зі здоров'ям. Інструкції з охорони праці також підкреслюють важливість регулярних перерв, підтримки гарної постави та практики вправ для очей, щоб зменшити навантаження на очі та тіло. Належні умови освітлення та належна вентиляція є важливими факторами, які сприяють здоровому та комфортному робочому середовищу. Крім того, керівні принципи з охорони праці стосуються електробезпеки, сприяючи використанню мережевих фільтрів, належному управлінню кабелями та регулярному обслуговуванню обладнання для запобігання електричним небезпекам, таким як ураження електричним струмом або пожежа. Таким чином, дотримання інструкцій з охорони праці під час роботи на комп'ютері має вирішальне значення для підтримки безпечного та здорового робочого середовища, запобігання фізичним та ергономічним травмам та забезпечення загального благополуччя. Важливо визначити пріоритетність цих керівних принципів для підвищення продуктивності, мінімізації ризиків для здоров'я та сприяння позитивному робочому середовищу.

Інструкція з охорони праці та техніки безпеки при роботі на персональному комп'ютері (ПК) охоплює кілька важливих аспектів. Інструкція поширюється на всі підрозділи підприємства, де ведеться робота з ПК. Він був розроблений відповідно до нормативних актів, що регулюють охорону праці та техніку безпеки. Співробітники, які користуються ПК, проходять інструктаж перед

початком роботи і отримують повторний інструктаж кожні півроку, при цьому результати заносяться в журнал. Користувачі несуть відповідальність за свою особисту безпеку і здоров'я, а також безпеку оточуючих. Вони повинні дотримуватися правил внутрішнього трудового розпорядку, уникати виконання інструкцій, що суперечать правилам техніки безпеки, бути обізнаними в наданні домедичної допомоги та використанні первинних засобів пожежогасіння. Основними факторами, які можуть вплинути на користувачів, є статична електрика, розподіл яскравості в полі зору, ураження електричним струмом, напруга зору, втома від уваги і тривалі періоди статичних навантажень. Робоче місце повинно мати достатнє природне та штучне освітлення, із застереженнями, щоб запобігти прямому освітленню екрана природним світлом та мінімізувати світлові відблиски від клавіатури та екрана. Основне обладнання включає ПК або ноутбук, монітор, клавіатуру, маніпулятор, настільний комп'ютер і стілець. Розміщення цих елементів повинно враховувати робочу позу користувача, вимоги до простору, видимість і здатність комфортно працювати. Додаткові вимоги безпеки включають огляд робочого місця та забезпечення його впорядкованості, перевірку загального стану обладнання та електропроводки, регулювання крісла та спинки, належне підключення необхідного обладнання, регулювання яскравості та контрастності, повідомлення про несправності керівникам, а також перерви для зменшення напруги та втоми. Також окреслюються заходи безпеки при надзвичайних ситуаціях і нещасних випадках, таких як короткі замикання або пожежі. Загалом, ця вичерпна інструкція підкреслює важливість підтримки безпечного робочого середовища під час використання персонального комп'ютера, забезпечення належної ергономіки, мінімізації ризиків та сприяння добробуту людей на робочому місці. [24]

Висновки до розділу ІІ

Виходячи з поданої в розділі І інформації, очевидно, що існує потреба щодо розробки спеціалізованого програмного забезпечення, спеціально призначеного для проведення комп'ютерно-технічної експертизи. Це вказує на важливість наявності спеціальних інструментів і систем, які можуть ефективно оцінювати та оцінювати проблеми, пов'язані з комп'ютером, такі як апаратні несправності, вразливості програмного забезпечення або порушення даних. Розробка власного програмного забезпечення дозволяє налаштувати та адаптувати їх до конкретних потреб та вимог комп'ютерно-технічних експертів, дозволяючи їм проводити всебічні та точні оцінки. Щодо характеристик та структури власного програмного забезпечення для проведення комп'ютерно-технічної експертизи, мною було розроблено характеристики та загальну структуру програми, які повинні бути включені у власне програмне забезпечення, призначене для комп'ютерно-технічної експертизи. Програмне забезпечення володіє спеціальними можливостями та функціональними можливостями, які полегшують вивчення та аналіз комп'ютерних систем, включаючи такі аспекти, як дізнатися IP адресу; дізнатися MAC Ethernet; отримати метадані з фото; загальна інформація про ПК; геодата точок Wi-Fi; записати звук з мікрофона; зберегти всі дані з браузера; отримати серійні номера ПК; Слід зазначити, що структура програмного забезпечення розроблена логічним та організованим чином, забезпечуючи ефективну навігацію, управління даними та функції звітності. Слід також зазначити про важливість надання інструкцій щодо забезпечення охорони праці та техніки безпеки при роботі на персональних комп'ютерах. Інструкції охоплюють різні аспекти, пов'язані зі створенням безпечного робочого середовища, включаючи ергономічне налаштування робочого місця, підтримання відповідних умов освітлення, уникнення потенційних небезпек та дотримання протоколів безпеки під час надзвичайних ситуацій. Ці інструкції мають вирішальне значення для

зниження ризиків для здоров'я та сприяння загальному добробуту людей, які активно працюють з персональними комп'ютерами.

РОЗДІЛ ІІІ РІВЕНЬ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ, РОЗРОБОК І ПРОПОЗИЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Мова програмування

У рамках цієї кваліфікаційної роботи метою є розробка програми з широким спектром функціональних можливостей. Обраною мовою для цього проекту є Python. Python - це високорівнева інтерпретована мова програмування з динамічною семантикою. Вбудовані структури даних і динамічні функції роблять її добре придатною для швидкої розробки додатків, а також як скриптову мову або мову-склейку для з'єднання компонентів. Простота і зрозумілість мови сприяють зниженню витрат на підтримку програм. Python сприяє модульності та повторному використанню коду завдяки підтримці модулів та пакетів. Інтерпретатор Python та велика стандартна бібліотека є у вільному доступі у вигляді вихідних кодів або двійкових файлів для основних платформ.

Однією з причин, чому програмістів приваблює Python, є його здатність підвищувати продуктивність. Відсутність етапу компіляції забезпечує швидкий цикл редагування-тестування-налагодження. Налагодження програм на Python є простим, оскільки помилки обробляються за допомогою винятків, а не викликають помилки сегментації. Коли виникає помилка, інтерпретатор згенерує виняток і надасть трасування стеку, якщо вона не була перехоплена програмою. Самоаналітична сила Python проявляється в його відладчику на рівні вихідного коду, який дозволяє перевіряти змінні, обчислювати вирази, встановлювати точки зупинки та переходити по коду. Крім того, додавання операторів виводу до вихідного коду часто є ефективною технікою налагодження завдяки швидкому циклу редагування-тестування-налагодження мови. [20] У контексті мого дослідження я провів порівняння між Python та кількома іншими інтерпретованими мовами, а саме Java, JavaScript, Perl, Tcl, Smalltalk, C++, Common Lisp та Scheme. Це порівняння фокусується на

специфічних для кожної мови аспектах і має на меті надати уявлення про придатність Python для мого конкретного проекту.

1. **Java.** Програми на Python, як правило, швидше розробляються порівняно з програмами на Java. Код на Python часто в 3-5 разів коротший за еквівалентний код на Java завдяки високорівневим типам даних та динамічній типізації Python. Java вимагає явного оголошення змінних і не має синтаксичної підтримки поліморфних типів списків і словників, які є в Python. Python добре підходить як мова «склеювання», тоді як Java більше підходить як мова низькорівневої реалізації. Python і Java можна використовувати разом, комбінуючи Java-компоненти в Python-додатки або використовуючи Python для створення прототипів перед реалізацією на Java.
2. **JavaScript.** «Об'єктно-орієнтована» підмножина Python схожа на JavaScript, але Python підтримує більший обсяг програмування та краще повторне використання коду завдяки справжньому об'єктно-орієнтованому програмуванню з класами та успадкуванням.
3. **Perl.** Python і Perl мають схожі можливості, але різну філософію. Perl наголошує на підтримці загальних прикладних завдань, тоді як Python зосереджується на методологіях програмування, проектуванні структур даних та об'єктно-орієнтованому програмуванні. Нотація Python націлена на читабельність та зручність супроводу. Хоча Python можна порівняти з Perl у своїй початковій області застосування, вона має ширшу сферу застосування.
4. **Tcl.** Python і Tcl можна використовувати як мови розширення додатків, але відсутність структур даних і менша швидкість виконання Tcl порівняно з Python роблять її менш придатною для написання великих програм. Можливість розробки додатків повністю на Python (без використання компонентів C або C++) забезпечує швидший процес розробки. Інструментарій Tk з Tcl був прийнятий Python як стандартна бібліотека компонентів графічного інтерфейсу.
5. **Smalltalk.** Синтаксис Python є більш поширеним у порівнянні з Smalltalk, що полегшує навчання програмістів. Обидві мови мають динамічну типізацію та

зв'язування, і все в Python є об'єктом. Python розрізняє вбудовані типи об'єктів від визначених користувачем класів і наразі не дозволяє успадкування від вбудованих типів. Типи колекційних даних Smalltalk є більш досконалими, в той час як бібліотека Python перевершує їх у функціональності, пов'язаній з Інтернетом та веб-технологіями.

6. **C++.** Код на Python часто в 3-5 разів коротший, ніж еквівалентний код на Java, і в 5-10 разів коротший, ніж еквівалентний код на C++. Python ефективна як клейка мова для об'єднання компонентів, написаних на C++.
7. **Common Lisp та Scheme.** Хоча Python поділяє динамічну семантику з Common Lisp та Scheme, відмінності в синтаксисі роблять пряме порівняння суб'єктивним. Інтроспективні можливості Python подібні до Lisp, і програми на Python можуть динамічно конструювати та виконувати фрагменти програм. Звичайний Лісп є більш широким, а світ Схеми фрагментований, тоді як Python пропонує єдину, безкоштовну та компактну реалізацію.

Отже, лаконічний синтаксис Python, динамічна типізація, підтримка об'єктно-орієнтованого програмування, великі бібліотеки та можливість інтеграції з іншими мовами роблять її кращим вибором для проекту порівняно з вищезгаданими мовами. Її читабельність і зручність підтримки сприяють підвищенню продуктивності, а наявність широкого спектру бібліотек та інструментів полегшує процес розробки. [2]

3.2. Програмне забезпечення для створення програми.

Після ретельного вивчення та оцінки, я вирішив обрати PyCharm як інтегроване середовище розробки (IDE), якому я надаю перевагу. Розроблений компанією JetBrains, PyCharm вважається найпопулярнішим IDE для Python і широко використовується такими великими компаніями, як Twitter, Facebook, Amazon та Pinterest. PyCharm пропонує сумісність з багатьма операційними системами, включаючи Windows, Linux та macOS, що робить його універсальним вибором. Однією з його помітних переваг є включення модулів і пакетів, які значно допомагають розробникам у програмуванні на Python, дозволяючи швидше та ефективніше розробляти програмне забезпечення.

Крім того, PyCharm забезпечує гнучкість у налаштуванні IDE відповідно до конкретних вимог мого проекту. PyCharm має цілу низку переваг. Інтелектуальний редактор коду значно полегшує написання якісного коду, а реалізація різних колірних кодів для ключових слів, класів та функцій покращує читабельність коду та виявлення помилок. Функція автозаповнення особливо корисна для прискорення процесу кодування. IDE також пропонує зручні засоби навігації кодом, що полегшують редагування та вдосконалення коду.

Це дозволяє легко переходити до функцій, класів або файлів, а також просто знаходити певні елементи, символи або змінні у вихідному коді. Режим лінзи ще більше спрощує перевірку та налагодження вихідного коду. PyCharm включає можливості рефакторингу, що дозволяє швидко та ефективно змінювати як локальні, так і глобальні змінні, не впливаючи на зовнішню продуктивність коду. Широка підтримка розробки веб-додатків на Python, що охоплює такі популярні веб-технології, як HTML, CSS, JavaScript, AngularJS і NodeJS, ще більше зміцнила моє рішення. IDE також легко інтегрується з основними веб-фреймворками Python, такими як Django, web2py та Pyramid, пропонуючи автозавершення, підказки параметрів та інструменти для налагодження. Враховуючи, що мій проект зосереджений на Data Science, сумісність PyCharm з основними науковими бібліотеками Python, такими як

Matplotlib, NumPy та Anaconda, виявилася безцінною. Інтеграція з такими інструментами, як Django, IPython та Pytest, полегшила пошук інноваційних рішень. Крім того, інтерактивні графіки в PyCharm допомогли зрозуміти та ефективно проаналізувати дані. PyCharm вирізнявся не лише простотою встановлення та використання, але й великою колекцією плагінів та ярликів для підвищення продуктивності. Функції автозавершення та розфарбовування коду були особливо привабливими для мене, оскільки вони покращують загальний процес розробки. Наявність великої спільноти розробників Python є цінним ресурсом для вирішення проблем та підтримки. Вважаю, що широкі можливості та підтримка PyCharm роблять його ідеальним вибором для мого проекту. [13]

3.3. Початок написання коду. Імпорт необхідних бібліотек та модулів

Щоб розпочати свій проект, я здійснив імпорт основних бібліотек та модулів, необхідних для моєї програми на Python. Завдяки імпорту необхідних бібліотек та модулів, я отримав доступ до широкого спектру готових функцій та інструментів, які допоможуть досягти бажаної функціональності.

```
import requests
import uuid
import os
import json
from datetime import datetime
import exifread
import platform
import GPUtil
import psutil
import datetime
import os
import subprocess as sp
from tabulate import tabulate
from psutil._common import bytes2human
import wifi
import requests
import sounddevice as sd
import soundfile as sf
import datetime
import time
import sqlite3
import os
import shutil
import wmi

```

Рис. 3.1. Перелік бібліотек, необхідних для розробки програми

Requests - це HTTP-бібліотека, яка пропонує просте і елегантне рішення для надсилання HTTP/1.1 запитів. За допомогою Requests можна легко надсилати

запити без необхідності вручну обробляти рядки запитів або кодувати дані PUT і POST у формі. Бібліотека надає зручний метод json для роботи з даними JSON. Як один з найпопулярніших пакетів Python, Requests може похвалитися вражаючою кількістю щотижневих завантажень, яка наразі перевищує 30 мільйонів. Йому широко довіряють і покладаються на нього понад мільйон репозиторіїв на GitHub. Requests добре обладнаний для створення надійних і надійних додатків, які взаємодіють через HTTP. Він пропонує такі функції, як Keep-Alive і пул з'єднань, підтримка міжнародних доменів і URL-адрес, управління сеансами зі збереженням файлів cookie, перевірка TLS/SSL, подібна до веб-браузерів, методи автентифікації, включаючи Basic і Digest, обробка файлів cookie, подібна до словників, автоматичне розпакування і декодування вмісту, завантаження файлів, що складаються з кількох частин, підтримка SOCKS-проксі, тайм-аути з'єднання, потокове завантаження, автоматичне врахування .netrc-файлів, а також підтримка шматкових HTTP-запитів. [15]

Uuid – це модуль, який пропонує незмінні об'єкти UUID через клас UUID, а також кілька функцій для генерації різних версій UUID. Функції uuid1(), uuid3(), uuid4() і uuid5() відповідають версіям 1, 3, 4 і 5 UUID відповідно, як визначено в RFC 4122. Для отримання унікального ідентифікатора рекомендується використовувати uuid1() або uuid4(). Важливо зазначити, що uuid1() потенційно може порушувати конфіденційність, оскільки включає мережеву адресу комп'ютера у створений UUID. З іншого боку, uuid4() генерує випадковий UUID. Поведінка uuid1() залежить від використовуваної платформи, оскільки вона може повертати або не повертати «безпечний» UUID. Безпечний UUID - це UUID, згенерований за допомогою методів синхронізації, які гарантують, що жоден процес не може отримати той самий UUID. [19]

Os – це модуль, який пропонує крос-платформне рішення для використання специфічних функцій операційної системи. У випадку, якщо метою є читання або запис файлів, можна скористатися функцією open(). Для обробки шляхів доступний модуль os.path, а якщо існує необхідність обробити рядки з декількох файлів, переданих як аргументи командного рядка, можна

скористатися модулем fileinput. Розширення, специфічні для певної операційної системи, можна отримати за допомогою модуля os, але їх використання може погіршити переносимість. Всі функції, які приймають шлях або імена файлів, можуть обробляти як байти, так і рядкові об'єкти в якості аргументів і повертати об'єкт того ж типу, що і вхідні дані. Однак на різних платформах існують певні обмеження та варіації. Наприклад, на VxWorks не підтримуються деякі функції, такі як os.popen, os.fork, os.execv та os.spawnp. На платформах WebAssembly wasm32-emscripten та wasm32-wasi деякі частини модуля os можуть бути недоступні або працювати по-іншому. Важливо зазначити, що будь-які невірні або недоступні імена файлів, шляхи або аргументи правильного типу, які не приймаються операційною системою, викличуть помилку OSError або її підкласи.. Атрибут os.name містить назву імпортованого модуля, залежного від операційної системи. Наразі зареєстровано такі імена: 'posix', 'nt' і 'java'. Для детальнішої перевірки ідентифікації системи ви можете звернутися до модуля платформи. Функція os.uname() надає інформацію про версію системи. Загалом модуль os надає спосіб узгодженої роботи з функціональністю, залежною від операційної системи, але важливо пам'ятати про варіації та обмеження, характерні для конкретної платформи. [10]

JSON (JavaScript Object Notation) - це полегшений формат, який використовується для обміну даними, як визначено в RFC 7159 та ECMA-404. Він заснований на синтаксисі об'єктних літералів JavaScript, хоча і не є точною підмножиною JavaScript. Важливо проявляти обережність при розборі даних JSON з ненадійних джерел, оскільки зловмисний рядок JSON може потенційно споживати значні ресурси процесора і пам'яті. Рекомендується обмежувати розмір даних, що аналізуються. Json надає API, подібний до модулів marshal і pickle у стандартній бібліотеці, що полегшує його користувачам цих модулів. [7]

Модуль datetime надає класи для маніпулювання датами та часом. Хоча він підтримує арифметику дати і часу, основна увага при реалізації приділяється ефективному вилученню атрибутів для форматування та маніпуляцій з вихідними даними. Об'єкти дати і часу можна класифікувати як «обізнані» або

«наївні» залежно від того, чи містять вони інформацію про часовий пояс. Обізнаний об'єкт, знаючи відповідні коригування часу, такі як часові пояси та перехід на літній час, може визначити своє положення відносно інших обізнаних об'єктів. Він представляє конкретний момент часу без двозначності. З іншого боку, наївний об'єкт не має достатньої інформації, щоб визначити свою позицію відносно інших об'єктів дати/часу. Чи представляє він всесвітній координований час (UTC), місцевий час або інший часовий пояс, залишається на розсуд програми. З наївними об'єктами легко працювати, але вони можуть не враховувати певні аспекти реальності. Для програм, які потребують усвідомлених об'єктів, об'єкти datetime та time мають додатковий інформаційний атрибут часового поясу, який називається tzinfo. Цей атрибут можна встановити для екземпляра підкласу абстрактного класу tzinfo. Ці об'єкти tzinfo містять такі дані, як зміщення UTC, назву часового поясу та інформацію про перехід на літній час. Модуль datetime надає один конкретний клас tzinfo, який називається timezone. Він може представляти прості часові пояси з фіксованим зміщенням від UTC, такі як власне UTC або північноамериканські часові пояси EST та EDT. Підтримка більш детальних часових поясів залишається на розсуд програми. Правила, що регулюють коригування часу в усьому світі, часто визначаються політикою, а не раціональністю, вони часто змінюються, і не існує універсального стандарту, окрім UTC. [3]

EXIFread - це зручний модуль Python, призначений для вилучення метаданих Exif з файлів цифрових зображень. Він підтримує різні формати файлів, включаючи TIFF, JPEG, PNG, Webp та HEIC. Модуль сумісний з версіями Python від 3.5 до 3.10 і більше не підтримує Python 2, починаючи з версії 3.0.0, через синтаксичні помилки, пов'язані з підказкою типів. Він пропонує опції обробки, які можна використовувати як в режимі консолі, так і в скрипті. [4]

Platform – це вбудований у Python модуль, який дозволяє збирати вичерпну інформацію про систему, на якій виконується ваша програма. Ця інформація включає дані про пристрій, його операційну систему, вузол, версію ОС, версію Python тощо. Модуль platform особливо корисний при перевірці

сумісності програми з встановленою версією Python на конкретній системі або при перевірці відповідності апаратних специфікацій вимогам програми. Цей модуль доступний у бібліотеці Python і не потребує встановлення через pip. [11]

GPUtil - це модуль Python, який використовує команду nvidia-smi для отримання інформації про стан графічних процесорів NVIDIA. Скануючи комп'ютер, GPUtil визначає всі наявні графічні процесори, оцінює їхню доступність і генерує відсортований список доступних на даний момент графічних процесорів. Доступність кожного GPU визначається на основі таких факторів, як використання пам'яті та робоче навантаження. Хоча модуль розроблено з урахуванням вибору GPU для глибокого навчання, він не обмежується якоюсь конкретною задачею або бібліотекою. Його можна використовувати в різних сценаріях, де визначення доступних графічних 5процесорів буде корисним. [5]

Psutil - це бібліотека Python, яка надає послідовний і незалежний від платформи спосіб отримання інформації про активні процеси та використання системи, включаючи процесор, пам'ять, диски, мережу і датчики. Вона слугує цінним інструментом для таких завдань, як моніторинг системи, профілювання, управління ресурсами та контроль процесів. Пропонуючи функціональність, подібну до різних інструментів командного рядка UNIX, таких як ps, top, lsof, netstat та інших, psutil спрощує отримання системних даних і дозволяє ефективно аналізувати та керувати запущеними процесами. [12]

Subprocess – це модуль в Python, який надає зручний спосіб виконання зовнішніх програм або команд у вашому Python-коді. Він дозволяє запускати нові програми, передавати їм дані та отримувати їхні результати. Цей модуль слугує засобом для надання інструкцій комп'ютеру за допомогою Python замість прямого введення команд у командний рядок. Це спрощує автоматизацію завдань і полегшує інтеграцію між Python та іншими програмами. За допомогою модуля підпроцесів у фахівця з’являється можливість виконувати команди командного інтерпретатора, такі як «ls» або «ping», і фіксувати отримані результати у вашому Python-коді. Він також дозволяє запускати інші скрипти

Python або виконувані файли, включаючи .exe-файли у Windows. Крім того, subprocess пропонує можливості для перенаправлення вхідних і вихідних потоків, надаючи контроль над даними, що надсилаються до виконуваного процесу та отримуються від нього. Однією з найцінніших особливостей модуля підпроцесу є його здатність обробляти вхідні, вихідні дані та помилки, згенеровані дочірнім процесом, безпосередньо у коді Python. Це дозволяє використовувати вихідні дані підпроцесу як змінні у своїх скриптах, підвищуючи потужність та універсальність використання підпроцесів. [1]

Tabulate – це бібліотека, яка пропонує зручне рішення для візуально привабливого представлення табличних даних у Python.

Основними цілями цієї бібліотеки є

- Спрощення друку невеликих таблиць, оскільки за допомогою одного виклику функції бібліотека автоматично форматує таблицю на основі даних, мінімізуючи необхідність ручного форматування.
- Створення табличних даних для полегшеної розмітки простого тексту завдяки підтримці кількох вихідних форматів, придатних для подальшого редагування або перетворення, що робить її придатною для створення табличних даних у вигляді простого тексту.
- Представлення змішаних текстових і числових даних у зручному для читання вигляді. Бібліотека самостійно вирівнює стовпці, забезпечує настроюване форматування чисел і вирівнює числові дані на основі десяткових крапок.

Основна функція, яку надає бібліотека, називається «табулювання». Вона приймає різні типи табличних даних як перший аргумент, наприклад, список списків, ітерабельний список змінних, список або ітерабельний список словників (з ключами як стовпчиками), словник змінних (з ключами як стовпчиками), двовимірний масив NumPy або масиви записів NumPy (з іменами стовпчиків). На виході функція генерує добре відформатовану таблицю простого тексту. [6]

Wifi - це інструмент командного рядка, який спрощує процес підключення до мереж WiFi, діючи як обгортка для iwlist та /etc/network/interfaces. Він пропонує простіший спосіб керування з'єднаннями WiFi безпосередньо з командного рядка. Крім того, команда wifi реалізована у вигляді бібліотеки Python, що дозволяє використовувати її програмно у скриптах Python. [14]

Sounddevice згідно з документацією, пропонує прив'язку до бібліотеки PortAudio, а також зручні функції для обробки аудіосигналів, що зберігаються у масивах NumPy. Для відтворення WAV-файлів необхідно встановити numpy та soundfile, які дозволяють відкривати WAV-файли як масиви NumPy. Після встановлення python-sounddevice, numpy та soundfile стає можливим читання WAV-файлу як масиву NumPy та його подальше відтворення. [21]

SoundFile – це звукова бібліотека, яка використовує libsndfile, CFFI та NumPy. За допомогою SoundFile користувач може читати і записувати звукові файли. Основну функціональність для читання і запису файлів надає libsndfile, кросплатформенна бібліотека з відкритим вихідним кодом (LGPL), призначена для роботи з різними форматами семплів звукових файлів. Вона сумісна з Windows, OS X, Unix та іншими платформами. SoundFile використовує CFFI, інтерфейс зовнішніх функцій, для доступу до можливостей libsndfile з Python. CFFI підтримує CPython 2.6+, Python 3.x та PyPy 2.0+. У SoundFile звукові дані представлені за допомогою масивів NumPy. [17]

SQLite - це бібліотека C, яка пропонує компактне та ефективне рішення для роботи з дисковими базами даних без необхідності використання окремого серверного процесу. Вона дозволяє користувачам взаємодіяти з базою даних за допомогою модифікованої версії мови запитів SQL. SQLite зазвичай використовується для внутрішнього зберігання даних у різних додатках. Крім того, він може слугувати корисним інструментом для створення прототипів додатків перед міграцією до більших баз даних, таких як PostgreSQL або Oracle. Модуль sqlite3, розроблений Герхардом Херінгом (Gerhard Häring), надає інтерфейс SQL, який відповідає специфікаціям, викладеним у стандарті DB-API

2.0, як описано в PEP 249. Для безперешкодної інтеграції потрібна мінімальна версія SQLite 3.7.15. [18]

Модуль Shutil - це частина стандартних утиліт Python, яка надає зручні високорівневі операції для роботи з файлами. Він пропонує такі функції, як копіювання, створення та видалення файлів і каталогів, роблячи завдання керування файлами більш автоматизованими. [16]

Модуль Python WMI - це оптимізований рівень, побудований на розширенні pywin32, що спрощує процес використання WMI API в Python. Він написаний на чистій мові Python і був ретельно протестований з версіями Python від 2.5 до 3.4. Він розроблений для сумісності з найновішими версіями pywin32. WMI, скорочення від Windows Management Instrumentation, є реалізацією Microsoft Web-Based Enterprise Management (WBEM), яка має на меті забезпечити стандартизовану модель (CIM) для доступу до різної інформації про комп'ютерні системи. [21] [22] Імпорт необхідних бібліотек та модулів став початковим кроком при роботі безпосередньо над проектом та заклав основу для нарощування існуючих можливостей Python та використання функціональних можливостей, що надаються зовнішніми бібліотеками. Імпорт забезпечив мені надійну відправну точку для ефективної реалізації необхідних функцій та можливостей у моєму проекті.

3.4. Створення меню програми та написання основних функцій

У процесі розробки моєї програми я почав працювати над функцією меню, щоб надати користувачам зручний інтерфейс для доступу до різних функціональних можливостей. Функція меню, позначена як «print_menu», була розроблена для представлення структурованого та організованого списку опцій. Її завдання полягало у виведенні на екран заголовку меню із заголовком «Меню», за яким йшов пронумерований список завдань, які могла виконати програма.

Ці завдання включали отримання IP-адреси, отримання MAC-адреси Ethernet, вилучення метаданих з фотографій, отримання загальної інформації про комп'ютер, отримання геоданих точок Wi-Fi, запис звуку з мікрофона, збереження даних браузера, отримання серійних номерів комп'ютерів і, нарешті, можливість вийти з програми. Метою функції меню є покращення досвіду користувача, пропонуючи чіткий і стислий огляд доступних опцій, надаючи можливість користувачам легко орієнтуватися в програмі.

```
26  def print_menu():
27      print("==== Меню ====")
28      print("1. Дізнатися IP адресу")
29      print("2. Дізнатися MAC Ethernet")
30      print("3. Отримати метадані з фото")
31      print("4. Загальна інформація про ПК")
32      print("5. Геодата точок Wi-Fi")
33      print("6. Записати звук з мікрофона")
34      print("7. Зберегти всі дані з браузера")
35      print("8. Отримати серійні номера ПК")
36      print("0. Вийти з програми ")
37      print("================")
```

Рис. 3.2. Функція меню print_menu() та її структурні елементи

Після розробки функції меню я перейшов до послідовної реалізації конкретних функцій для кожного завдання. Першим завданням було отримання IP-адреси. Для цього я створив функцію з назвою «get_external_ip». У цій функції я використовував бібліотеку запитів, щоб відправити HTTP GET-запит на кінцеву точку «https://ifconfig.co/ip», яка надає зовнішню IP-адресу. Отримана відповідь була оброблена, і IP-адреса була витягнута шляхом видалення всіх початкових і кінцевих пробілів. Зрештою, отримана IP-адреса відображається

для користувача із супровідним повідомленням «Ваша зовнішня IP-адреса:», за яким слідує фактична IP-адреса. Ця функція слугувала для отримання та представлення зовнішньої IP-адреси, що сприяло підвищенню загальної функціональності та корисності програми.

```
38
39  def get_external_ip():
40      response = requests.get('https://ifconfig.co/ip')
41      ip = response.text.strip()
42      print("Ваша зовнішня IP-адреса:", ip)
```

Рис. 3.3. Функція отримання IP-адреси get_external_ip() та її структурні елементи

Продовжуючи процес розробки, наступним завданням було отримання MAC-адреси Ethernet. Для цього я створив функцію під назвою «get_mac». В рамках цієї функції я спочатку відобразив повідомлення «Ваш MAC-address це : «Потім я використав модуль uuid для отримання MAC-адреси, виконавши побітові операції і відформатувавши отримані значення.

Отриману MAC-адресу представлено у вигляді послідовності шістнадцяткових чисел, розділених двокрапкою, за допомогою функції з'єднання та розуміння списку. Діапазон, що використовувався при розбитті, забезпечив обробку MAC-адрес групами по два елементи. Крім того, отриманий список був перевернутий, щоб забезпечити правильний порядок елементів. У результаті MAC-адреса виводилася на консоль. Ця функція полегшує пошук і відображення MAC-адреси Ethernet, що однозначно сприяє розширенню функціональних можливостей програми.

```
44  def get_mac():
45      print ("Ваш MAC address це : ", end="")
46      print (':'.join(['{:02x}'.format((uuid.getnode() >> elements) & 0xff)
47      for elements in range(0,2*6,2)][::-1]))
```

Рис. 3.4. Функція отримання MAC-адреси get_mac() та її структурні елементи

Наступним кроком стало отримання метаданих з фотографій. Для цього я реалізував функцію з назвою «get_metadata». У цій функції я ініціалізував порожній список «GEOJSON_LIST» для зберігання інформації про метадані. Крім того, я визначив словник з назвою «STATUS», який містив два ключі, «success» і «fail», обидва з яких спочатку були встановлені на нуль.

Метою цієї функції було зібрати метадані з фотографії, такі як інформація про геолокацію, і зберегти їх у списку «GEOJSON_LIST». Словник «STATUS» відстежував кількість успішних і неуспішних спроб збору метаданих. Виконуючи цю функцію, програма обробляє фотографію і намагається витягти відповідні метадані. Витягнуті метадані додаються до списку «GEOJSON_LIST» відповідно. Словник «STATUS» оновлюватиметься на основі успіху або невдачі процесу вилучення метаданих. Реалізація функції «get_metadata» покращила загальну функціональність програми, надавши можливість отримувати та відстежувати метадані з фотографій.

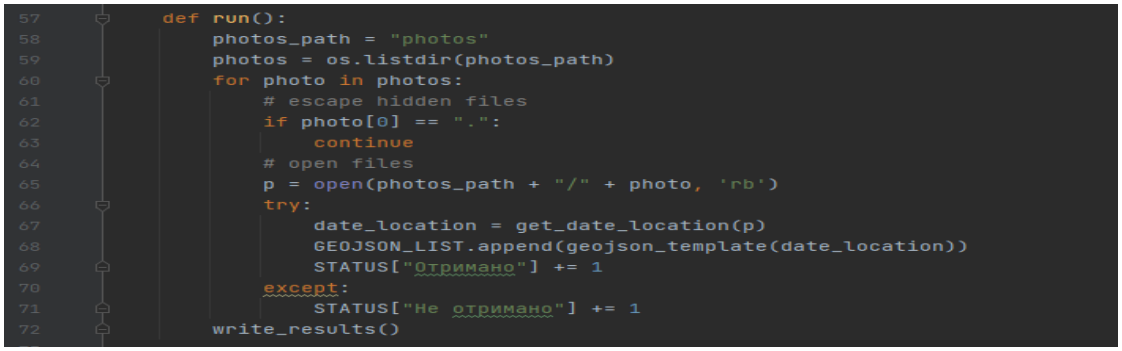
```
49  def get_metadata():
50      GEOJSON_LIST = []
51
52      STATUS = {
53          "success": 0,
54          "fail": 0
55      }
```

Рис. 3.5. Функція отримання мета-даних з фотографій get_metadata() та її структурні елементи

Не менш важливим кроком стала реалізації функції під назвою «run» для виконання певного завдання. У межах цієї функції програма зосереджується на обробці кількох фотографій, що зберігаються у визначеному каталозі, позначеному як «photos_path». Функція використовує модуль «os» для отримання списку фотографій, присутніх у вказаному каталозі.

Щоб забезпечити пропуск прихованих файлів під час обробки, функція включала умовну перевірку для виключення файлів, що починаються з символу крапки («.»). Ці приховані файли ігноруються, що дозволяє програмі обробляти лише видимі фотографії. Для кожної знайденої дійсної фотографії функція відкриває файл у двійковому режимі за допомогою функції «open». Після цього програма намагається витягнути з фотографій інформацію про дату та місцезнаходження, викликаючи функцію «get_date_location». У разі успіху, отримана інформація використовувалася для створення шаблону GeoJSON за допомогою функції «geojson_template». Потім цей шаблон GeoJSON додавався до файлу «GEOJSON_LIST» для подальшої обробки.

Під час обробки фотографій програма підтримує механізм відстеження статусу за допомогою словника «STATUS». Словник містить два ключі, «Отримано» (отримано) і «Не отримано» (не отримано), які відповідно інкрементуються залежно від успішності або неуспішності вилучення інформації про дату і місцезнаходження з кожної фотографії. Після обробки всіх фотографій функція викликає «write_results» для збереження зібраних даних, включаючи шаблони GeoJSON та інформацію про стан, для подальшого аналізу або зберігання. Реалізація функції «run» дає змогу систематично обробляти декілька фотографій, витягувати відповідну інформацію та генерувати шаблони GeoJSON, відстежуючи кількість успішних та невдалих спроб за допомогою словника «STATUS».



```
def run():
    photos_path = "photos"
    photos = os.listdir(photos_path)
    for photo in photos:
        # escape hidden files
        if photo[0] == ".":
            continue
        # open files
        p = open(photos_path + "/" + photo, 'rb')
        try:
            date_location = get_date_location(p)
            GEOJSON_LIST.append(geojson_template(date_location))
            STATUS["Отримано"] += 1
        except:
            STATUS["Не отримано"] += 1
    write_results()
```

Рис. 3.6. Функція run(), яка забезпечує обробку декількох фотографій з подальшим витягуванням відповідної інформації та генеруванням шаблонів GeoJSON та її структурні елементи

Ще одним завданням в рамках проекту була реалізація функції «get_date_location». Мета функції полягала в тому, щоб витягти інформацію про дату і місцезнаходження з метаданих EXIF заданого файлу. Для цього функція використовує модуль «exifread» для обробки файлу і створення словника EXIF, який зберігався у змінній «exif_dict». Отримавши доступ до певних тегів у цьому словнику, можна отримати відповідні метадані. Одним з найважливіших тегів є «EXIF DateTimeOriginal», який містить початкове значення часу файлу. Щоб відформатувати це значення у потрібний формат часу, функція використовує функцію «datetime.strptime», в результаті чого відформатований час зберігається у змінній «export_datetime». Крім того, функція витягує інформацію про широту і довготу з тегів, пов'язаних з GPS, включаючи «GPS GPSLatitudeRef», «GPS

GPSLatitude», «GPS GPSLongitudeRef» і «GPS GPSLongitude». Потім ці значення обробляються функцією «convert_location», яка перетворює їх у змістовний формат місцезнаходження. Отримана інформація про місцезнаходження зберігається у змінній «location». Зрештою, функція повертає словник, що містить витягнуту інформацію про дату та місцезнаходження. Значення дати було доступне за допомогою ключа «datetime», а інформація про місцезнаходження - за допомогою ключа «location». Реалізувавши функцію «get_date_location», я зміг успішно витягти необхідні метадані з EXIF-інформації кожного файлу. Ця функція виявилася безцінною для створення шаблонів GeoJSON і полегшення подальшої обробки в рамках проекту.

```
74  def get_date_location(file):
75      exif_dict = exifread.process_file(file)
76
77      exif_datetime = exif_dict['EXIF DateTimeOriginal'].printable
78      export_datetime = str(datetime.strptime(exif_datetime, '%Y:%m:%d %H:%M:%S'))
79
80      lon_ref = exif_dict["GPS GPSLongitudeRef"].printable
81      lon = exif_dict["GPS GPSLongitude"].printable
82      lat_ref = exif_dict["GPS GPSLatitudeRef"].printable
83      lat = exif_dict["GPS GPSLatitude"].printable
84
85      location = convert_location(lon_ref, lon, lat_ref, lat)
86
87      return {
88          "datetime": export_datetime,
89          "location": location
90      }
```

Рис. 3.7. Функція get_data_location(file) та її структурні елементи

Наступним завданням була реалізація функції «convert_location». Її мета полягає у перетворенні координат широти та довготи зі специфічного формату в більш змістовне представлення. Для цього функція отримала чотири параметри:

- «lon_ref» для посилання на довготу;
- «longitude» для значення довготи;
- «lat_ref» для посилання на широту;
- «latitude» для значення широти.

Ці параметри було надано у вигляді рядків. Функція починається з обробки значення довготи. Рядок обробляється, щоб видалити непотрібні символи та розділити на окремі компоненти. Градуси, хвилини, секунди та напрямок вилучаються та перетворюються у значення з плаваючою комою. Після обробки та розділу ці значення використовуються для обчислення остаточної координати довготи з урахуванням відповідних перетворень. Якщо орієнтир довготи вказує

напрямок, відмінний від «E» (Схід), координата довготи множиться на -1, щоб відобразити правильну орієнтацію. Аналогічні кроки виконуються і для значення широти. Рядок проходить обробку з виділенням компонентів, а на основі градусів, хвилин, секунд і напрямку обчислюється остаточна координата широти. Якщо посилання на широту вказує напрямок, відмінний від «N» (північ), координата широти множиться на -1. Результатом функції є повернення списку, що містить перетворені координати широти та довготи.

Реалізувавши функцію «convert_location», мені вдалось успішно перетворити надані значення широти і довготи в більш змістовне представлення. Цей процес перетворення мав вирішальне значення для точної інтерпретації та використання географічної інформації в рамках проекту.

```
92    def convert_location(lon_ref, longitude, lat_ref, latitude):
93        # Longitude
94        lon = longitude[1:-1].replace(" ", "").replace("/", ",").split(",")
95        lon = float(lon[0]) + float(lon[1]) / 60 + float(lon[2]) / float(lon[3]) / 3600
96        if lon_ref != "E":
97            lon = lon * (-1)
98
99        # Latitude
100       lat = latitude[1:-1].replace(" ", "").replace("/", ",").split(",")
101       lat = float(lat[0]) + float(lat[1]) / 60 + float(lat[2]) / float(lat[3]) / 3600
102       if lat_ref != "N":
103           lat = lat * (-1)
104
105       # return [lat, lon, 20]
106       return [lat, lon]
```

Рис. 3.8. Функція convert_location (lon_ref, longitude, lat_ref, latitude) та її структурні елементи

Крім того, в рамках проекту я реалізував функцію «geojson_template». Ця функція відіграє ключову роль у створенні шаблонів GeoJSON на основі витягнутої інформації про дату та місцезнаходження. Функція отримала словник «date_location», який містить дані про дату та місцезнаходження, отримані з функції «get_date_location». Використовуючи ці дані, функція створює шаблон GeoJSON. Шаблон GeoJSON складається з трьох основних компонентів:

- ключ «type» був встановлений у значення «Feature», щоб вказати, що об'єкт GeoJSON представляє об'єкт;
- ключ «geometry» було визначено за допомогою власного вкладеного словника. Цей вкладений словник визначає «type» як «Point» для представлення геометрії точки і надає «coordinates», використовуючи інформацію про місцезнаходження, витягнуту з «дата_місцезнаходження».

- ключ «properties», який представляє додаткові властивості, пов'язані з об'єктом GeoJSON. У цьому випадку властивість «datetime» було встановлено у відповідне значення з «date_location['datetime']».

Структурувавши шаблон GeoJSON таким чином, функція дозволяє програмі послідовно відформатувати та впорядкувати витягнуту інформацію про дату та місцезнаходження. Цей стандартизований формат був важливим для подальшої обробки та аналізу геопросторових даних в рамках проекту.

```
108    def geojson_template(date_location):
109        return {
110            "type": "Feature",
111            "geometry": {
112                "type": "Point",
113                "coordinates": date_location["location"]
114            },
115            "properties": {
116                "datetime": date_location["datetime"],
117            },
118
119        }
```

Рис. 3.9. Функція geojson_template(date_location) та її структурні елементи

Крім того, в рамках проекту я реалізував функцію «write_results». Ця функція відіграла вирішальну роль в обробці результатів обробки метаданих та створенні вихідного файлу, що містить дані у форматі GeoJSON. Після виклику функція спочатку відображає підсумок оброблених метаданих. Цей підсумок включає у себе кількість успішно оброблених локацій, доступ до яких здійснюється через «STATUS['success']», та кількість невдалих спроб, доступ до яких здійснюється через «STATUS['fail']». Після цього функція відкриває файл з назвою «Geo.json» у режимі запису, створюючи новий файл або перезаписуючи існуючий. Потім використовується функція «json.dumps» для перетворення «GEOJSON_LIST» у рядок у форматі JSON. Це рядкове представлення даних GeoJSON записується до відкритого файлу за допомогою методу «file.write».

За наслідками двох попередніх кроків, послідовно друкується повідомлення з підтвердженням, що файл, який містить метадані, успішно збережено. Для завершення загального робочого процесу функція викликає функцію «run», яка відповідає за виконання основної логіки програми і координує завдання, пов'язані з отриманням і обробкою потрібної інформації.

Завдяки включенню в програму функції «write_results», витягнуті метадані ефективно зберігаються в структурованому форматі, що уможливило подальший аналіз і використання геопросторових даних в рамках проекту.

```
121  def write_results():
122      # print(GEOJSON_LIST)
123      print("Успішно оброблені метадані з {} locations, {} не успішно".format(STATUS["success"], STATUS["fail"]))
124      file = open("Geo.json", "w")
125      file.write(json.dumps(GEOJSON_LIST, sort_keys=True, indent=4, separators=(', ', ': ')))
126      print('Файл з метаданими збережено!')
127
128  run()
```

Рис. 3.10. Функція write_results() та її структурні елементи

Ще одним завданням в рамках програми була реалізація функції «get_main_system», яка мала на меті отримати та зібрати необхідну інформацію про основні компоненти системи. Спочатку функція використовує метод «platform.uname()» для доступу до детальної інформації про систему, включаючи операційну систему, мережевий вузол, версію випуску, архітектуру машини та тип процесора. Ця інформація зберігається у змінній «uname».

Крім того, функція отримує поточну дату і час за допомогою методу «datetime.datetime.now()», зберігаючи їх у змінній «date». Усередині функції визначаються змінні «Mb» і «Gb» для представлення розмірів мегабайт і гігабайт, відповідно, у байтах. Крім того, було визначено допоміжну функцію з назвою «Round» для перетворення числових значень в округлений десятковий формат. Ця функція ділить вхідне значення на відповідний множник у байтах (1024 * 1024 * 1024) і повертає округлений результат з двома десятковими знаками. Включення функції «get_main_system» дозволяє програмі отримати важливу інформацію про систему, таку як відомості про операційну систему, дату і час, а також розмір пам'яті. Ця інформація є цінною для створення вичерпних звітів або аналізу характеристик системи в рамках проекту.

```
130  def get_main_system():
131      uname = platform.uname()
132      date = datetime.datetime.now()
133
134      Mb = 1024 * 1024
135      Gb = Mb * 1024
136      def Round(nmup):
137          nmup / (1024 * 1024 * 1024)
138          return round(nmup, 2)
```

Рис. 3.11. Функція та її структурні елементи

3.5. Загальна інформація про ПК

Окремої уваги заслуговує масштабний розділ програми «Загальна інформація про комп'ютер» у якому було реалізовано різні елементи для отримання та представлення основних відомостей про комп'ютерну систему.

3. Інформація про систему - надає вичерпні дані про операційну систему, мережевий вузол, версію, архітектуру комп'ютера та тип процесора. Цю інформацію було отримано за допомогою методу «platform.uname()», що дає уявлення про загальну конфігурацію системи.
4. Інформація про графічний процесор - відображає для користувача відповідні деталі про графічний процесор (GPU), встановлений в системі. Сюди входить така інформація, як модель графічного процесора, версія драйвера, об'єм пам'яті та інші специфічні для GPU параметри. Отримавши доступ до цієї інформації та представивши її, користувач може отримати уявлення про можливості свого графічного процесора
5. Інформація про CPU - відіграє важливу роль у наданні інформації користувачу про центральний процесор (CPU) системи. У межах цієї програми зазначений розділ витягує та представляю таку інформацію, як модель, архітектура, тактова частота та кількість фізичних і логічних ядер. Цей елемент дозволяє користувачам оцінити обчислювальну потужність і можливості центрального процесора.
6. Інформація про оперативну пам'ять - має на меті отримати детальну інформацію про оперативну пам'ять (ОЗП) системи. У цьому розділі надано інформацію про загальний обсяг встановленої пам'яті, а також про доступну та використану пам'ять. Цей елемент дозволяє користувачам відстежувати та оцінювати використання пам'яті своїх систем.
7. Інформація про накопичувачі - містить дані про накопичувачі системи (тип накопичувача, ємність, файлова система та вільний простір). Завдяки цій інформації користувачі можуть отримати уявлення про конфігурацію своїх сховищ і відстежувати використання місця на дисках.

Загалом, розділ «Загальна інформація про комп'ютер» програми охоплював багато елементів, які надавали всебічний огляд різних аспектів комп'ютерної системи. Отримавши доступ до системної інформації, графічного та центрального процесорів, оперативної пам'яті та дисків, користувачі могли отримати цінну інформацію про конфігурацію, продуктивність та зберігання даних своїх комп'ютерів.

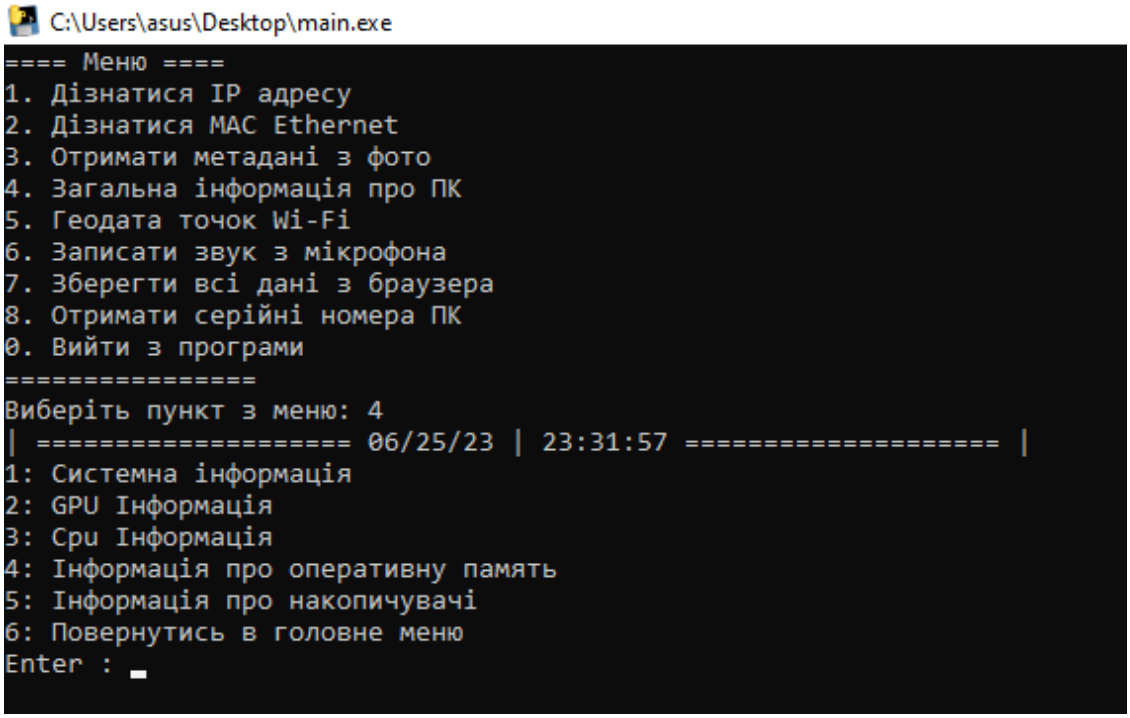


Рис. 3.12. Інтерфейс програми та перелік функцій програми у розділі «Загальна інформація про ПК»

У розділі "Загальна інформація про комп'ютер" програми одним з ключових елементів є компонент "Інформація про систему". Цей елемент зосереджений на наданні користувачам важливих відомостей про їхню комп'ютерну систему. Після запуску програма виводить на екран відформатований результат, що містить різноманітну системну інформацію. Елемент "Інформація про систему" починається з відображення рядка з рівних знаків для створення візуального поділу. Потім він переходить до представлення таких деталей, як назва системи, версія випуску, імена користувачів, версія та архітектура машини. Ця інформація отримується за допомогою методу "platform.uname()" для доступу до базової інформації про систему.

Для покращення користувацького досвіду програма також включає паузу, реалізовану за допомогою функції "input()", що дозволяє користувачам переглянути відображену інформацію перед тим, як продовжити роботу. Після натискання користувачем будь-якої клавіші екран очищується за допомогою команди "sp.call('cls', shell=True)", забезпечуючи чистий інтерфейс для подальшого пошуку інформації. Завдяки включенню в програму елемента "Системна інформація" користувачі отримують доступ до життєво важливих даних про свою комп'ютерну систему. Ця інформація може допомогти в системному адмініструванні, усуненні несправностей і розумінні базової архітектури та специфікацій комп'ютера.

```
"""                                                    System Informations


def system_informations():
    print("=" * 20, "System Information", "=" * 20)
    print(f"System    : {uname.system} {uname.release}")
    print(f"Users Name: {psutil.users()}")
    print(f"Version   : {uname.version}")
    print(f"Machine   : {uname.machine}")
    input('\nPress any key to continue.. ')
    sp.call('cls', shell=True)
```

Рис. 3.13. Функція system_informations() та її структурні елементи

У розділі "Загальна інформація про ПК" програми ще одним важливим елементом є компонент "Інформація про графічний процесор". Цей компонент зосереджений на наданні користувачам детальної інформації про їхні графічні процесори (GPU). Під час виконання програма збирає і представляє різні відомості, пов'язані з GPU, у добре відформатованому вигляді.

Елемент "Інформація про GPU" починається з використання функції "GPUtil.getGPUs()" для отримання інформації про доступні GPU у системі. Вона перебирає кожен графічний процесор і витягує відповідні дані, такі як ідентифікатор графічного процесора, назву, відсоток завантаження, вільну пам'ять, використану пам'ять, загальну пам'ять, температуру, версію драйвера та UUID. Ці дані зберігаються у вигляді списку, що полегшує їх форматування та подальше відображення. Для покращення читабельності програма використовує функцію "tabulate" з бібліотеки "tabulate" для створення гарно відформатованої

таблиці. Заголовки таблиці відповідають різним атрибутам графічного процесора, зокрема ідентифікатору, назві, завантаженню, пам'яті (вільна, використана, загальна), температурі, версії драйвера та UUID. Список відомостей про графічний процесор передається до функції "tabulate" разом із заданим форматом таблиці ("fancy_grid"), в результаті чого створюється візуально привабливе табличне представлення інформації про графічний процесор. Для того, щоб користувачі могли переглянути відображену інформацію про графічний процесор, у програмі передбачено паузу, реалізовану за допомогою функції "input()". Як тільки користувач натискає будь-яку клавішу, екран очищується за допомогою команди "sp.call('cls', shell=True)", забезпечуючи чистий інтерфейс для подальших операцій. Завдяки включенню елемента "Інформація про графічний процесор" програма надає користувачам цінну інформацію про їхні графічні процесори, зокрема про завантаження, використання пам'яті, температуру та інші важливі деталі. Ця інформація може бути корисною для моніторингу продуктивності графічного процесора, виявлення потенційних проблем та оптимізації графічно інтенсивних програм і робочих процесів.

```
def Gpu_informations():
    gpus = GPUtil.getGPUs()
    list_gpus = []
    for gpu in gpus:
        gpu_id = gpu.id
        gpu_name = gpu.name
        gpu_load = f'{round(((gpu.load) * 100), 2)}%'
        gpu_free_M = f'{gpu.memoryFree}MB'
        gpu_Use_M = f'{gpu.memoryUsed}MB'
        gpu_Total_M = f'{gpu.memoryTotal}MB'
        gpu_temperature = f'{gpu.temperature}°C'
        gpu_version = f'{gpu.driver}'
        gpu_uuid = gpu.uuid
        list_gpus.append(
            (gpu_id, gpu_name, gpu_load, gpu_free_M, gpu_Use_M, gpu_Total_M, gpu_temperature, gpu_version, gpu_uuid))
    print("=" * 50, "GPU Інформація", "=" * 50)
    print(tabulate(list_gpus, headers=(
    "id", "name", "load", "MemoryFree", "MemoryUsed", "MemoryTotal", "Temperature", "Version", "Uuid"),
                   tablefmt="fancy_grid"))
    input("\nНажміть кнопку для продовження.. ")
    sp.call('cls', shell=True)
```

Рис. 3.14. Функція GPU_informations() та її структурні елементи

Доволі важливим компонентом є модуль "Інформація про пам'ять". Цей компонент надає користувачам інформацію про використання пам'яті системи та пов'язані з цим деталі. Після запуску функція "Інформація про пам'ять" збирає

інформацію про віртуальну пам'ять системи за допомогою функції "psutil.virtual_memory()". Отримані дані про пам'ять зберігаються у вигляді списку, що полегшує їх форматування та представлення.

Для покращення читабельності інформації, що виводиться на екран, програма використовує функцію "tabulate" з бібліотеки "tabulate". Ця функція створює добре структуровану таблицю, яка містить важливі атрибути пам'яті, такі як швидкість пам'яті, загальний обсяг пам'яті, використання пам'яті (як в абсолютному значенні, так і у відсотках), а також доступну пам'ять. Список, що містить дані про пам'ять, передається до функції "tabulate" разом із заданим форматом таблиці ("fancy_grid"), в результаті чого створюється візуально привабливе представлення інформації про пам'ять.

Для того, щоб користувачі могли переглянути відображену інформацію про пам'ять, у програмі передбачено паузу за допомогою функції "input()". Це дозволяє користувачеві не поспішати з вивченням інформації перед тим, як продовжити роботу. Після натискання користувачем будь-якої клавіші екран очищується за допомогою команди "sp.call('cls', shell=True)", забезпечуючи чистий інтерфейс для подальшої взаємодії.

Завдяки включенню модуля "Інформація про пам'ять", програма надає користувачам всебічний огляд використання пам'яті системи, включаючи загальний обсяг пам'яті, поточне використання та доступну пам'ять. Ця інформація може бути корисною для моніторингу продуктивності системи, виявлення проблем, пов'язаних з пам'яттю, та оптимізації розподілу ресурсів для підвищення ефективності та стабільності роботи.

```
184    def memory_informations():
185        print("=" * 25, "Інформація про оперативну память", "=" * 25)
186        mem = psutil.virtual_memory()
187        list_memory = []
188
189        memory_total = f"{Round(mem.total / Gb)} GB"
190        memory_use = f"{Round(mem.used / Mb)} MB | {mem.percent}%"
191        memory_free = f"{Round(mem.free / Mb)} MB | {round(((mem.total - mem.used) / mem.total) * 100, 2)}%"
192        memory_speed = 121
193        list_memory.append((memory_speed, memory_total, memory_use, memory_free))
194
195        print(tabulate(list_memory, headers=(" Name", "Total", "       Use", "      Free"), tablefmt="fancy_grid"))
196        input("\nНажміть кнопку для продовження..")
197        sp.call('cls', shell=True)
```

Рис. 3.15. Функція memory_informations() та її структурні елементи

Окремої уваги заслуговує функція "Інформація про процеси". Цей компонент надає користувачам можливість здійснити огляд процесів, запущених у системі. Під час виконання функція "Process" використовує бібліотеку "psutil" для отримання інформації про всі процеси, запущені у системі. Потім функція друкує отримані дані про процес, дозволяючи користувачам спостерігати за різними процесами і пов'язаною з ними інформацією.

Для покращення користувацького досвіду у програмі передбачено паузу за допомогою функції "input()". Це дозволяє користувачеві переглянути відображену інформацію про процес у власному темпі, перш ніж продовжити роботу. Як тільки користувач натискає будь-яку клавішу, екран очищається за допомогою команди "sp.call('cls', shell=True)", забезпечуючи чистий інтерфейс для подальшої взаємодії. Завдяки включенню модуля "Інформація про процеси", програма надає користувачам інформацію про активні процеси в системі. Це може бути корисно для моніторингу системних ресурсів, виявлення ресурсоємних процесів та усунення проблем з продуктивністю. Розуміння запущених процесів може сприяти управлінню та оптимізації системи.

```
"""                                                    Process Informations


def Process():
    print("=" * 35, "Всі процеси ", "=" * 35)
    psutil.test()
    input("\nНажміть кнопку для продовження.. ")
    sp.call('cls', shell=True)

```

Рис. 3.16. Функція Process() та її структурні елементи

Модуль "Інформація про накопичувачі" надає користувачам детальну інформацію про дискові накопичувачі або пристрої зберігання даних, підключені до системи. Під час виконання функція "disk" використовує бібліотеку "psutil" для збору інформації про кожен розділ диска або пристрій зберігання даних. Для кожного розділу функція отримує ключові показники, такі як назва пристрою, загальний обсяг пам'яті, використаний простір, вільний простір, відсоток використання і тип файлової системи. Для покращення читабельності, функція перетворює значення ємності з байт у формат, зручний для читання людиною, за

допомогою функції "bytes2human". Зібрана інформація потім представляється в табличному форматі за допомогою функції "tabulate", що дозволяє користувачам легко зрозуміти деталі, пов'язані з диском. Як і в інших модулях, у програмі передбачена пауза після відображення інформації про диск, що надає користувачам можливість переглянути інформацію у власному темпі. Як тільки користувач натискає будь-яку клавішу, екран очищується за допомогою команди "sp.call('cls', shell=True)", забезпечуючи чистий інтерфейс для подальшої взаємодії. Завдяки включенню модуля "Інформація про накопичувачі", програма дозволяє користувачам отримувати інформацію про їхні дискові накопичувачі або пристрої зберігання даних, включаючи їхню ємність, використання та тип файлової системи. Ця інформація необхідна для керування дисковим простором, моніторингу використання дисків та прийняття обґрунтованих рішень щодо керування та оптимізації сховища.

```
211     ""
212     """                                                    Disk Informations
213
214
215     def disk():
216         print("="*20, "Інформація про дискові накопичувачі", datetime.datetime.now().strftime('%x'), "="*20)
217         list_total = []
218         for part in psutil.disk_partitions(all=True):
219             if os.name == 'nt':
220                 if 'cdrom' in part.opts or part.fstype == '':
221                     continue
222             usage = psutil.disk_usage(part.mountpoint)
223             list_total.append((part.device, bytes2human(usage.total), bytes2human(usage.used), bytes2human(usage.free),
224             f"{usage.percent} %", part.fstype))
225
226         print(tabulate(list_total, headers=("Device", "Total", "Used", "Free", "Use ", "Type"), tablefmt="fancy_grid"))
227         input("\nНажміть кнопку для продовження.. ")
228         sp.call('cls', shell=True)
```

Рис. 3.17. Функція disk() та її структурні елементи

На додаток до окремих функцій для збору конкретної інформації про ПК, програма включає центральну функцію управління, яка називається "go". Ця функція діє як меню, за допомогою якого користувачі можуть вибрати потрібну категорію інформації, яку вони хочуть отримати. Після виконання, функція "go" відображає відформатований заголовок з поточною датою і часом. Потім користувачеві пропонується ввести число, що відповідає бажаній категорії інформації. Доступні варіанти включають системну інформацію, інформацію

про графічний процесор, інформацію про центральний процесор, інформацію про пам'ять, інформацію про диск і можливість повернутися до головного меню.

На основі введених користувачем даних функція виконує відповідну функцію для отримання та відображення запитуваної інформації. Наприклад, вибір опції "1" запускає виконання функції "system_informations", опція "2" запускає функцію "Gpu_informations" і так далі. Якщо користувач вводить невірний або нерозпізнаний ввід, з'являється повідомлення про помилку, що вказує на необхідність повторного введення правильного варіанту. Щоб забезпечити безперебійну роботу користувача, програма продовжує працювати в циклі до нескінченності, дозволяючи користувачам отримувати доступ до декількох категорій інформації послідовно без необхідності перезапускати програму. Завдяки функції "go" та пов'язаному з нею циклу меню, програма дозволяє користувачам зручно переміщатися та отримувати доступ до різних розділів, що полегшує ефективний пошук та аналіз системних деталей.

```
def go():
    print("|", "=" * 20, date.strftime("%x | %X"), "=" * 20, "|")
    x = input(
        "1: Системна інформація\n2: GPU Інформація\n3: Cpu Інформація\n4: Інформація про оперативну память\n5:
        "Інформація про накопичувачі "
        "\n6: Вернутись в головне меню\nEnter : ")
    if x == "1":
        system_informations()
    elif x == "2":
        Gpu_informations()
    elif x == "3":
        memory_informations()
    elif x == "4":
        Process()
    elif x == "5":
        disk()
    elif x == "6":
        main()
    else:
        print("Error Renter")

while True:
    go()
```

Рис. 3.18. Функція go() та її структурні елементи

3.6. Розробка спеціальних функцій для програми

В процесі розробки програми були створені таокж і спеціальні функції, зокрема, для вирішення задачі отримання геолокаційної інформації за допомогою Wi-Fi мереж. Ця функціональність досягається за допомогою функції "get_geo_wifi", яка складається з двох вкладених функцій: "get_available_wifi_networks" та "get_geo".

Перша функція, "get_available_wifi_networks", відповідає за отримання списку доступних Wi-Fi мереж. Вона ініціалізує порожній список під назвою "wifi_networks" для зберігання інформації про кожну мережу. Використовуючи метод "wifi.Cell.all" з відповідною назвою інтерфейсу (у цьому випадку "wlp0s20f3"), вона знаходить мережі і перебирає їх.

Для кожної мережі витягуються відповідні дані, такі як SSID, BSSID, канал, рівень сигналу і тип шифрування, які додаються до списку "wifi_networks". Нарешті, функція повертає завершений список.

```
256  def get_geo_wifi():
257      def get_available_wifi_networks():
258          wifi_networks = []
259
260          # Отримання списку доступних Wi-Fi мереж
261          networks = wifi.Cell.all('wlp0s20f3')  # Замініть 'wlan0' на інтерфейс Wi-Fi вашої системи
262
263          for network in networks:
264              wifi_networks.append({
265                  'ssid': network.ssid,
266                  'bssid': network.address,
267                  'channel': network.channel,
268                  'signal_strength': network.signal,
269                  'encryption_type': network.encryption_type
270              })
271
272          return wifi_networks
273
274      # Отримання списку доступних Wi-Fi мереж
275      available_networks = get_available_wifi_networks()
276
277      # Виведення списку доступних Wi-Fi мереж
278      for network in available_networks:
279          print(f"SSID: {network['ssid']}")
280          print(f"BSSID: {network['bssid']}")
281          print(f"Канал: {network['channel']}")
282          print(f"Рівень сигналу: {network['signal_strength']}")
283          print(f"Тип шифрування: {network['encryption_type']}")
284          print()
```

Рис. 3.19. Функція get_available_wifi_networks () та її структурні елементи

Після отримання доступних мереж Wi-Fi викликається друга функція "get_geo". Ця функція має на меті отримати інформацію про геолокацію для кожної мережі Wi-Fi у списку "available_networks". Вона ітераційно перебирає мережі, витягуючи BSSID (мережеву адресу) і створюючи URL-адресу за допомогою Mylnikov Geolocation API. Ця URL-адреса включає BSSID як

параметр і вказує бажаний формат і версію даних. Потім робиться GET-запит до API, використовуючи сконструйовану URL-адресу.

Відповідь від API у форматі JSON містить інформацію про результат геолокації та дані. Змінна "get_status" використовується для перевірки коду статусу відповіді, при цьому значення 404 означає, що геолокацію не вдалося знайти, а значення 200 означає успішне отримання геолокації. Якщо статус 404, буде надруковано повідомлення про неможливість знайти геолокацію. З іншого боку, якщо статус дорівнює 200, дані геолокації витягуються і роздруковуються. Для виконання цієї функції функція "get_geo" викликається всередині функції "get_geo_wifi". Це запускає отримання інформації про геолокацію для кожної доступної мережі Wi-Fi. Отримані дані потім роздруковуються, надаючи такі деталі, як BSSID та інформацію про геолокацію для кожної мережі.

```
286     def get_geo():
287         for bssid_all in available_networks:
288             bssid = f"{bssid_all['bssid']}"
289             print(bssid)
290             url = f"https://api.mylnikov.org/geolocation/wifi?bssid={bssid}&data=open&v=1.1"
291             print(url)
292             get_geo = requests.get(url=url)
293             result_json = get_geo.json()
294             get_status = result_json['result']
295             get_data = result_json['data']
296             if get_status == 404:
297                 print('Нажаль найти геопозицію не вдалось')
298             elif get_status == 200:
299                 print(f'{get_data}')
300     get_geo()
```

Рис. 3.20. Функція get_geo() та її структурні елементи

Завдяки цим спеціалізованим функціям програма може отримувати і представляти інформацію про геолокацію на основі доступних мереж Wi-Fi. Ця функціональність дозволяє користувачам отримувати інформацію про географічне розташування, пов'язане з кожною мережею Wi-Fi, пропонуючи потенційні застосування в таких сферах, як аналіз мереж, геопросторове картографування та послуги на основі визначення місцезнаходження.

Під час створення програми я розробив також і спеціальну функцію для виконання завдання запису аудіо. Основна функція, get_record_voice, включає в себе кілька кроків для полегшення процесу запису.

1. Встановлюються параметри запису. Змінна duration визначає тривалість кожного запису в секундах, а змінна sample_rate - частоту дискретизації, тобто кількість відліків, захоплених за секунду. Це значення залежить від використовуваного аудіопристрою.
2. Функція працює в нескінченному циклі while, що дозволяє здійснювати безперервний запис. Під час кожної ітерації циклу відбуваються наступні дії:

- За допомогою функції datetime.datetime.now() отримуються поточні дата/час.
- Отримана мітка часу форматується у вигляді рядка з використанням формату %Y-%m-%d_%H-%M-%S, створюючи ідентифікатор для файлу запису.
- Запис звуку ініціюється викликом функції sd.rec з бібліотеки sounddevice. Ця функція приймає два аргументи: загальну кількість семплів для запису, що обчислюється множенням тривалості на частоту дискретизації, та задану частоту дискретизації.
- Записаний звук зберігається у змінній audio.
- Програма чекає на завершення запису за допомогою функції sd.wait(). Це гарантує, що програма зупинить виконання до завершення процесу запису.
- Записаний звук зберігається у WAV-файл за допомогою функції sf.write з бібліотеки soundfile. Ця функція приймає три аргументи: ім'я файлу, який потрібно зберегти, записані аудіодані та частоту дискретизації, з якою було записано аудіо.
- Виводиться повідомлення із зазначенням імені файлу, куди було збережено аудіо, що підтверджує успішний запис.
- Перед початком наступного запису вводиться пауза в 1 хвилину, реалізована за допомогою функції time.sleep. Ця затримка дозволяє контролювати інтервал між послідовними записами.

Реалізувавши функцію get_record_voice, програма може безперервно записувати аудіо, генеруючи унікальні імена файлів для кожного запису на основі поточної дати і часу. Ця функціональність пропонує потенційні

застосування в різних сферах, включаючи збір аудіоданих, дослідження та розробку систем на основі голосу.

Слід зазначити, що икористання функції get_record_voice для аудіозапису завжди має відбуватися за явної згоди осіб, які беруть участь у ньому. Повага до приватного життя та дотримання етичних норм є важливими при роботі з аудіозаписами. Отримання згоди є фундаментальним етичним принципом захисту прав і приватного життя людей. Дуже важливо проінформувати особу, яку записують, та отримати її згоду, переконавшись, що вона повністю усвідомлює мету, тривалість та потенційне використання запису. Дотримуючись практики, що ґрунтується на згоді, користувачам слід підтримувати етичні стандарти та ставити на перше місце приватність і автономію людей.

```
304  def get_record_voice():
305      # Параметри запису
306      duration = 10  # Час запису одного голосового зразка
307      sample_rate = 44100  # Частота запису
308
309      stop_recording = False  # Флаг для зупинки запису
310
311      def stop(e):
312          nonlocal stop_recording
313          stop_recording = True
314
315      keyboard.on_release_key("esc", stop)  # Встановлення обробника на натискання клавіші ESC
316
317      while not stop_recording:
318          # Генерація імені файлу з датою та часом
319          current_time = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
320          filename = f"recording_{current_time}.wav"
321
322          # Запис аудіо
323          print(f"Записую у файл: {filename}")
324          audio = sd.rec(int(duration * sample_rate), samplerate=sample_rate, channels=2)
325          sd.wait()  # Очікування завершення запису
326          sf.write(filename, audio, sample_rate)  # Збереження записаного аудіо в файл
327
328          print(f"Зберіг у файл: {filename}")
329
330          # Пауза на 1 хвилину перед наступним записом
331          time.sleep(60)
332
333      keyboard.unhook_all()  # Звільнення обробника клавіші ESC
```

Рис. 3.21. Функція get_record_voice() та її структурні елементи

Справді вагомою, на думку, автора функцією в контексті розробки програмного забезпечення є спеціальна функція, відома як get_save_cookies, для вирішення конкретної задачі - вилучення та збереження файлів cookie з різних веб-браузерів. Ця функція виконує ряд кроків для ефективного виконання цього завдання. Для зберігання видобутих файлів cookie створюється тека з назвою

"SaveCookies". Функція os.makedirs використовується для створення теки, якщо вона ще не існує. Функція витягує файли cookie з веб-браузерів Google Chrome, Mozilla Firefox і Microsoft Edge.

- Для Google Chrome розташування файлу cookie вказується за допомогою змінної chrome_cookie_file, яка вказує на файл cookie за замовчуванням у каталозі даних користувача Chrome. Потім файл копіюється в папку "SaveCookies" з назвою "chrome_cookies".

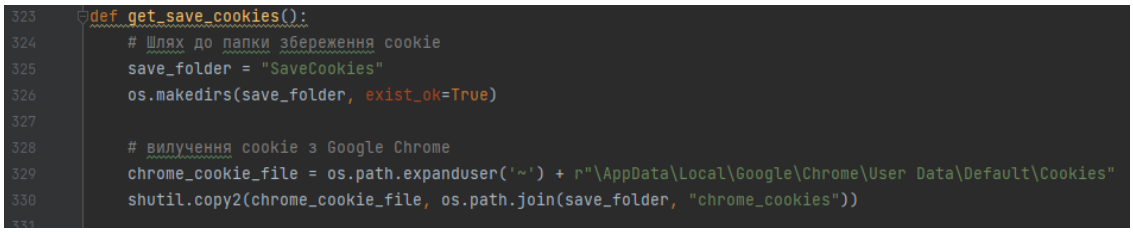


Рис. 3.22. Функція get_save_cookies() та її елемент для вилучення cookie з Google Chrome

- Для Mozilla Firefox розташування файлу cookie вказується за допомогою змінної firefox_cookie_file. Шлях до файлу містить заповнювач <профіль>, який потрібно замінити на відповідну назву профілю Firefox. Як і в Chrome, файл копіюється в папку "SaveCookies" з назвою "firefox_cookies.sqlite".

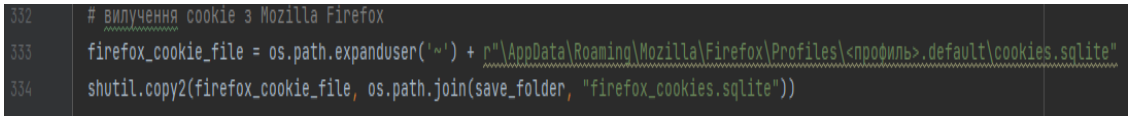


Рис. 3.23. Функція get_save_cookies() та її елемент для вилучення cookie з Mozilla Firefox

- Для Microsoft Edge розташування файлу cookie задається за допомогою змінної edge_cookie_file. Файл копіюється в папку "SaveCookies" з ім'ям "edge_cookies".

Рис. 3.24. Функція get_save_cookies() та її елемент для вилучення cookie з Microsoft Edge

Після цього витягнуті файли cookie витягуються з відповідного файлу cookie кожного браузера.

- Для Google Chrome встановлюється з'єднання з файлом "chrome_cookies" за допомогою sqlite3.connect. Створюється курсор і виконується SQL-запит для отримання імені та значення файлу cookie з таблиці "cookies". Отримані файли cookie зберігаються у змінній chrome_cookies.

```
340    # вилучення cookie з Google Chrome
341    chrome_conn = sqlite3.connect(os.path.join(save_folder, "chrome_cookies"))
342    chrome_cursor = chrome_conn.cursor()
343    chrome_cursor.execute("SELECT name, value FROM cookies")
344    chrome_cookies = chrome_cursor.fetchall()
345    chrome_conn.close()
```

Рис. 3.25. Функція get_save_cookies() та її елемент для з'єднання з файлом «chrome_cookies»

- Для Mozilla Firefox виконується аналогічний процес. Встановлюється з'єднання з файлом "firefox_cookies.sqlite", створюється курсор і виконується SQL-запит для отримання імені та значення файлу cookie з таблиці "moz_cookies". Отримані файли cookie зберігаються у змінній firefox_cookies.

```
347    # вилучення cookie з Mozilla Firefox
348    firefox_conn = sqlite3.connect(os.path.join(save_folder, "firefox_cookies.sqlite"))
349    firefox_cursor = firefox_conn.cursor()
350    firefox_cursor.execute("SELECT name, value FROM moz_cookies")
351    firefox_cookies = firefox_cursor.fetchall()
352    firefox_conn.close()
```

Рис. 3.26. Функція get_save_cookies() та її елемент для з'єднання з файлом «firefox_cookies.sqlite»

- Для Microsoft Edge цей процес відбувається так само, як і для Google Chrome. Встановлюється з'єднання з файлом "edge_cookies", створюється курсор і виконується SQL-запит для отримання імені та значення файлу cookie з таблиці "cookies". Отримані файли cookie зберігаються у змінній edge_cookies.

```
354    # вилучення cookie з Microsoft Edge
355    edge_conn = sqlite3.connect(os.path.join(save_folder, "edge_cookies"))
356    edge_cursor = edge_conn.cursor()
357    edge_cursor.execute("SELECT name, value FROM cookies")
358    edge_cookies = edge_cursor.fetchall()
359    edge_conn.close()
```

Рис. 3.27. Функція get_save_cookies() та її елемент для з'єднання з файлом «edge_cookies»

Витягнуті файли cookie роздруковуються і зберігаються в окремих текстових файлах для кожного браузера.

- Файли cookie з Google Chrome перебираються, і ім'я та значення кожного файлу друкуються. Крім того, інформація про файли cookie додається до файлу "chrome_cookies.txt" у папці "SaveCookies".

```
# Виведення та збереження cookie
for cookie in chrome_cookies:
    print(f"Chrome Cookie - Name: {cookie[0]}, Value: {cookie[1]}")
    with open(os.path.join(save_folder, "chrome_cookies.txt"), "a") as f:
        f.write(f"Name: {cookie[0]}, Value: {cookie[1]}\n")
```

Рис. 3.28. Функція get_save_cookies() та її елемент для виведення та збереження cookie з Google Chrome

- Той самий процес повторюється для файлів cookie, витягнутих з Mozilla Firefox та Microsoft Edge. Інформація про файли cookie роздруковується і додається до відповідних текстових файлів "firefox_cookies.txt" і "edge_cookies.txt".

```
for cookie in firefox_cookies:
    print(f"Firefox Cookie - Name: {cookie[0]}, Value: {cookie[1]}")
    with open(os.path.join(save_folder, "firefox_cookies.txt"), "a") as f:
        f.write(f"Name: {cookie[0]}, Value: {cookie[1]}\n")

for cookie in edge_cookies:
    print(f"Edge Cookie - Name: {cookie[0]}, Value: {cookie[1]}")
    with open(os.path.join(save_folder, "edge_cookies.txt"), "a") as f:
        f.write(f"Name: {cookie[0]}, Value: {cookie[1]}\n")
```

Рис. 3.29. Функція get_save_cookies() та її елемент для виведення та збереження cookie з Mozilla Firefox та Microsoft Edge відповідно

Таким чином, реалізувавши функцію get_save_cookies, програма може витягувати файли cookie з популярних веб-браузерів і зберігати їх для подальшого аналізу або використання. Ця функціональність надає можливості для таких завдань, як веб-скрепінг, управління сесіями та автоматизація процесів, що базуються на браузері. Спеціальна функція get_ram_serial_number була створена для отримання серійного номера оперативної пам'яті (RAM) комп'ютера. Ця функціональність була досягнута завдяки використанню модуля wmi, який надає доступ до системної інформації.

У межах функції створюється екземпляр класу WMI, що дозволяє взаємодіяти з інструментарієм керування Windows. Доступ до класу Win32_PhysicalMemory здійснюється через об'єкт c, який представляє з'єднання WMI. Перебираючи доступні модулі пам'яті за допомогою циклу for, функція отримує серійний номер, пов'язаний з кожним модулем. Для демонстрації використання функції get_ram_serial_number викликаються додаткові спеціальні функції для отримання серійних номерів інших апаратних компонентів. Функції

get_motherboard_serial_number і get_hard_drive_serial_number викликаються для отримання серійних номерів материнської плати і жорсткого диска відповідно.

Нарешті, отримані серійні номери роздруковуються для кожного компонента. Серійний номер материнської плати виводиться з повідомленням "Серійний номер материнської плати: ", серійний номер жорсткого диска - "Серійний номер жорсткого диска:", а серійний номер оперативної пам'яті - "Серійний номер оперативної пам'яті:". Це дозволяє користувачам легко ідентифікувати та диференціювати ці апаратні компоненти на основі їх унікальних серійних номерів. Завдяки включенню цих спеціалізованих функцій в програму, користувачі можуть збирати важливу інформацію про обладнання, таку як серійні номери материнської плати, жорсткого диска та оперативної пам'яті. Ця функціональність виявляється корисною в різних сценаріях, включаючи діагностику системи, ідентифікацію обладнання та завдання технічного обслуговування.

```
def get_all_serios():
    def get_motherboard_serial_number():
        c = wmi.WMI()
        for board in c.Win32_BaseBoard():
            return board.SerialNumber

    def get_hard_drive_serial_number():
        c = wmi.WMI()
        for drive in c.Win32_DiskDrive():
            return drive.SerialNumber

    def get_ram_serial_number():
        c = wmi.WMI()
        for module in c.Win32_PhysicalMemory():
            return module.SerialNumber

    # Отримання серійного номера материноської плати
    motherboard_serial = get_motherboard_serial_number()
    print("Серійний номер материнської плати:", motherboard_serial)

    # Отримання серійного номера жорсткого диску
    hard_drive_serial = get_hard_drive_serial_number()
    print("Серійний номер жорсткого диска:", hard_drive_serial)

    # Отримання серійного номера оперативної памяті
    ram_serial = get_ram_serial_number()
    print("Серійний номер оперативної памяті:", ram_serial)
```

Рис. 3.30. Функція get_ram_serial_number() та її структурні елементи.

3.7. Оформлення головного меню

На завершальному етапі процесу розробки було реалізовано комплексну головну функцію для організації виконання різних завдань на основі даних, введених користувачем. Ця функція формує ядро програми і забезпечує інтерфейс, керований меню, для взаємодії користувачів з різними функціональними можливостями. Основна функція визначена за допомогою нескінченного циклу while, який гарантує, що програма продовжує працювати, доки користувач не вирішить вийти. У кожній ітерації циклу викликається функція print_menu, яка виводить на екран доступні користувачеві опції. Потім користувачеві пропонується зробити свій вибір.

Наступний блок коду використовує декілька операторів if-elif-else для управління виконанням різних завдань на основі вибору користувача. Кожен оператор if відповідає певному вибору з меню, і відповідний код виконується відповідно. Наприклад, якщо користувач вводить "1" як свій вибір, виконується блок коду під першим оператором if. Цей код викликає функцію get_external_ip, яка отримує зовнішню IP-адресу.

```
418  def main():
419      while True:
420          print_menu()
421          choice = input("Виберіть пункт з меню: ")
422
423          if choice == "1":
424              get_external_ip()
425              input("Натисніть Enter, щоб повернутись в головне меню...")
426              if os.name == 'nt':  # Для Windows
427                  os.system('cls')
428              else:  # Для Unix-подібних систем (Linux, macOS)
429                  os.system('clear')
```

Рис. 3.31 Елемент функції main(), щодо обрання пункту 1 – Дізнатись IP-адресу

Після виконання завдання користувачеві пропонується натиснути клавішу Enter для повернення до головного меню. Крім того, залежно від операційної системи (Windows або Unix), екран консолі очищується за допомогою відповідної команди (cls для Windows і clear для Unix-систем).

```
C:\Users\asus\Desktop\main.exe
==== Меню ====
1. Дізнатися IP адресу
2. Дізнатися MAC Ethernet
3. Отримати метадані з фото
4. Загальна інформація про ПК
5. Геодата точок Wi-Fi
6. Записати звук з мікрофона
7. Зберегти всі дані з браузера
8. Отримати серійні номера ПК
0. Вийти з програми
================
Виберіть пункт з меню: 1
Ваша зовнішня IP-адреса: 77.123.53.81
Натисніть Enter, щоб повернутись в головне меню...
```

Рис. 3.32 Результати після обрання у розділі меню в програмі пункту 1

Аналогічно, інші оператори if відповідають решті пунктів меню, а пов'язані з ними блоки коду виконують відповідні завдання. Ці завдання включають отримання MAC-адреси (get_mac), отримання інформації про метадані (get_metadata), отримання основної інформації про систему (get_main_system), отримання інформації про геолокацію на основі Wi-Fi (get_geo_wifi), запис голосу (get_record_voice), збереження файлів cookie з різних веб-браузерів (get_save_cookies) та отримання серійних номерів різних апаратних компонентів (get_all_serios).

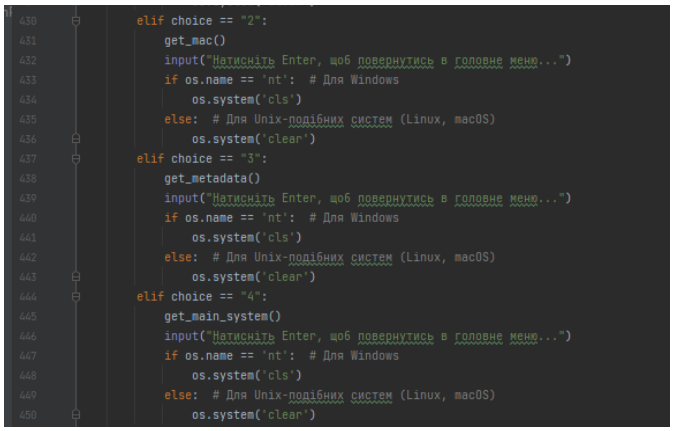


Рис. 3.31 Елемент функції main(), щодо обрання пунктів 2-4 через конструкцію з оператором «elif»

Після виконання кожного завдання користувачеві пропонується натиснути клавішу Enter для повернення до головного меню, а екран консолі відповідно очищується. Якщо користувач вводить "0", програма виводить повідомлення про намір вийти, а оператор break завершує цикл while, таким чином завершуючи роботу програми. Блок if __name__ == '__main__': гарантує, що головна функція буде виконана тільки в тому випадку, якщо скрипт запускається безпосередньо, а не імпортується як модуль. Загалом, ця добре структурована головна функція забезпечує зручний інтерфейс, що дозволяє користувачам легко вибирати і виконувати різні завдання. Включення чітких інструкцій, функцій для конкретних завдань і навігації за допомогою меню підвищує зручність використання програми та полегшує ефективну взаємодію з різними функціональними можливостями.

Висновки до розділу ІІІ

У рамках кваліфікаційної роботи мною було здійснено розробку програми на мові Python з широким спектром функціональних можливостей. Python, відомий своєю високорівневою інтерпретованою природою, динамічною семантикою та великими бібліотеками, є чудовим вибором для цього проекту. Під час роботи над програмою ця мова програмування забезпечувала швидку розробку, підтримку коду та сумісність з іншими мовами. Було проведено порівняння між Python та різними іншими інтерпретованими мовами, яке показало переваги Python з точки зору швидкості розробки, високорівневих типів даних, підтримки об'єктно-орієнтованого програмування, повторного використання коду та читабельності коду. Лаконічний синтаксис Python та великі бібліотеки зробили її оптимальним вибором для цього проекту.

IDE PyCharm було обрано як найкраще середовище розробки для проекту завдяки його крос-платформенній сумісності, всебічній підтримці Python, функціям, що підвищують продуктивність, та високій репутації в індустрії. Інтелектуальний редактор коду, інтерфейс, що налаштовується, та безшовна інтеграція з популярними фреймворками та бібліотеками зробили його ідеальним інструментом для розробки додатків на Python.

У процесі розробки мною було імпортовано кілька бібліотек і модулів Python для використання існуючих можливостей. Серед них такі бібліотеки/модулі, як Requests, UUID, os, JSON, ExifRead, Platform, GPUtil, psutil, subprocess, Tabulate, wifi, sounddevice, SoundFile, SQLite, shutil та Python WMI. Ці бібліотеки/модулі надають широкий спектр функцій та інструментів для ефективного досягнення бажаної функціональності.

Основною функцією програми є створення зручного інтерфейсу за допомогою структурованого меню. Користувачі програми можуть легко переміщатися і вибирати різні завдання, такі як отримання IP-адрес, MAC-адрес, вилучення метаданих, системної інформації, геолокація Wi-Fi, аудіозапис, пошук файлів cookie та серійних номерів обладнання. Для кожного завдання було призначено окрему функцію, яка використовувала можливості

імпортованих бібліотек/модулів. Програма надає вичерпну інформацію про комп'ютерну систему, включаючи відомості про операційну систему, графічний процесор, центральний процесор, оперативну пам'ять та інформацію про диски. Функція централізованого керування дозволяє користувачам легко переміщатися між різними розділами.

Спеціалізовані функції, такі як геолокація Wi-Fi та аудіозапис, розширили можливості програми для аналізу мережі, геопросторового картографування, збору аудіоданих, досліджень та розробки голосових систем.

Програма надає користувачам цінну інформацію про їхні комп'ютерні системи, дозволяючи їм приймати обґрунтовані рішення, усувати несправності та оптимізувати продуктивність системи. Однак конфіденційність та етичні міркування, особливо для таких функцій, як аудіозапис, повинні зберігатися за умови чіткої згоди та дотримання етичних стандартів.

Отже, розроблена програма продемонструвала потужність та гнучкість Python для створення універсального додатку з широким спектром функціональних можливостей. Використовуючи можливості Python, програма може бути корисною в різних сферах, включаючи системний аналіз, мережевий аналіз, геопросторове картографування, збір аудіоданих та дослідження. Зручний інтерфейс та ефективне виконання завдань роблять її цінним інструментом для користувачів, які шукають вичерпну інформацію про свої комп'ютерні системи.

ВИСНОВКИ

Метою нашого дослідження було здійснити аналіз сфери комп'ютерно-технічної експертизи шляхом дослідження методів та інструментів збору, аналізу та збереження цифрових доказів, а також шляхом розробки власного програмного забезпечення для проведення базової комп’ютерно-технічної експертизи. Для досягнення нашої мети мені слід було виконали низку завдань.

Перш за все нам слід було проаналізувати теоретичні засади комп’ютерно-технічної експертизи, а також дослідити методи та інструменти комп’ютерно-технічної експертизи. Як результат мною було написано Розділ І. Теоретичний аналіз методів та засобів комп'ютерно-технічної експертизи. Цей розділ присвячений вивченню правової бази та нормативно-правових актів, пов'язаних з проведенням комп'ютерно-технічної експертизи. Він досліджує закони, політику та керівні принципи, які регулюють процес комп'ютерної криміналістики та цифрових розслідувань, забезпечуючи дотримання законодавчих вимог та збереження цілісності доказів. У цьому розділі наведено загальний огляд експертного звіту в контексті комп'ютерно-технічної експертизи. Він пояснює мету, структуру та значення експертного висновку для документування висновків, представлення висновків та супроводу судового розгляду у справах, пов'язаних з комп'ютерними інцидентами. У цьому також розділі аналізуються різні методи та засоби, що використовуються в комп'ютерно-технічній експертизі. Він досліджує різні методи розслідування, методи збору даних, інструменти криміналістичного аналізу та інші технології, що використовуються для вивчення цифрових доказів для виявлення потенційних кіберзлочинів або технічних проблем. Висновки, зроблені наприкінці Розділу І, узагальнюють основні висновки та висновки, отримані в результаті теоретичного аналізу методів та інструментальних засобів комп'ютерно-технічної експертизи. Ці висновки можуть підкреслити важливість дотримання законодавства, роль експертних висновків та важливість

використання відповідних методів та інструментів при проведенні ефективних комп'ютерно-технічних розслідувань.

Також мною було здійснено власну практичну розробку програмного забезпечення для проведення базової комп'ютерно-технічної експертизи. Мною було створено власне програмне забезпечення, загальні аспекти якого описані у Розділі II. Загальні принципи створення програмного забезпечення для комп'ютерно-технічної експертизи У розділі наголошується на необхідності створення спеціалізованого програмного забезпечення з урахуванням специфічних вимог комп'ютерно-технічної експертизи. У ньому розглядаються переваги наявності програмного забезпечення, здатного ефективно підтримувати та впорядковувати процес проведення комп'ютерно-технічних досліджень. У розділі окреслено ознаки та загальну структуру, які повинні бути у власному програмному забезпеченні комп'ютерно-технічної експертизи. У цьому розділі обговорюються такі функції програмного забезпечення, як системний аналіз, управління даними, звітність та параметри налаштування, які підвищують ефективність та результативність процесу розслідування. Він охоплює такі аспекти, як ергономіка, правильне налаштування робочого місця, умови освітлення та протоколи безпеки, спрямовані на мінімізацію потенційних ризиків для здоров'я та сприяння безпечному робочому середовищу.У висновках узагальнено основні висновки, пов'язані із загальними принципами створення програмного забезпечення для комп'ютерно-технічної експертизи. Вони включають в себе підкреслення необхідності власного програмного забезпечення, виділення ключових характеристик і вимог до структури, а також важливості принципів з охорони праці та техніки безпеки.

У Розділі III: Впровадження, розробка та пропозиція програмного забезпечення розглядається вибір мови програмування для реалізації програмного забезпечення комп'ютерно-технічної експертизи. Він досліджує різні мови програмування та їх придатність для розробки бажаних функціональних можливостей. Також аналізуються програмні засоби розробки програм. Це включає вивчення різних варіантів програмного забезпечення,

доступних для створення програмного забезпечення комп'ютерно-технічної експертизи, та оцінку їх функцій та можливостей. Цей розділ охоплює початкові етапи написання коду програмного забезпечення для комп'ютерно-технічної експертизи. Він пояснює, як імпортувати необхідні бібліотеки та модулі, а також надає огляд процесу кодування. Розділ також присвячений створенню меню програм та реалізації основних функцій програмного забезпечення комп'ютерно-технічної експертизи. У ньому обговорюється дизайн і структура меню, а також розробка основних функцій для підтримки процесу розслідування. Мною також була представлена загальна інформація про персональні комп'ютери, що включає відомості про апаратні компоненти, операційні системи та іншу відповідну інформацію, необхідну для проведення комп'ютерно-технічної експертизи. У цьому розділі досліджується розвиток спеціалізованих функцій у складі програмного забезпечення комп'ютерно-технічної експертизи. У ньому обговорюється створення додаткових функцій, які розширюють можливості програмного забезпечення та підвищують ефективність процесу розслідування. Заключна частина присвячена форматуванню головного меню програмного забезпечення комп'ютерно-технічної експертизи, що охоплює візуальний дизайн та макет, забезпечуючи зручний інтерфейс для доступу до різних функцій.

Отож, можемо підвести висновок нашого дослідження, ми успішно проаналізували сферу комп'ютерно-технічної експертизи шляхом дослідження методів та інструментів збору, аналізу та збереження цифрових доказів. Додатково розроблено власне програмне забезпечення для проведення базової комп'ютерно-технічної експертизи. Щоб досягти поставленої мети, нам довелося виконати ряд завдань. Проводячи це дослідження, ми отримали цінну інформацію про сферу комп'ютерно-технічної експертизи та її важливість. Аналіз методів та інструментів дав нам всебічне розуміння процесів, пов'язаних зі збором та аналізом цифрових доказів, які мають вирішальне значення в сучасних криміналістичних розслідуваннях. Крім того, розробка власного програмного забезпечення дозволила нам налаштувати інструменти відповідно до конкретних потреб комп'ютерно-технічної експертизи.

Значимість цієї теми не можна недооцінювати. Зі швидким розвитком технологій і зростаючою залежністю від цифрових систем потреба в ефективній комп'ютерно-технічній експертизі стала першорядною. Кіберзлочини, витоки даних та інші цифрові інциденти становлять значні загрози, а здатність точно розслідувати та аналізувати цифрові докази має вирішальне значення для забезпечення справедливості, безпеки та захисту осіб та організацій. Успішно виконуючи наші цілі та проводячи це дослідження, ми сприяли просуванню галузі комп'ютерно-технічної експертизи. Наш аналіз та розробка власного програмного забезпечення служать цінними ресурсами для професіоналів, які займаються цифровими розслідуваннями та судовими експертизами. Ми сподіваємося, що це дослідження сприятиме вдосконаленню методів та інструментів у цій галузі, що в кінцевому підсумку призведе до більш ефективних та результативних процесів комп'ютерно-технічної експертизи.

Крім того, наше дослідження підкреслює необхідність постійного прогресу в галузі комп'ютерно-технічної експертизи. Оскільки технології продовжують розвиватися, виникають нові виклики та складності, що вимагають від професіоналів у цій галузі постійно адаптувати та вдосконалювати свої методи та інструменти. Розробка власного програмного забезпечення демонструє важливість інновацій та кастомізації для задоволення конкретних вимог розслідування. Крім того, наші висновки підкреслюють критичну роль цифрових доказів у сучасних юридичних та слідчих процесах. Аналіз та збереження цифрових доказів відіграють ключову роль у визначенні істини, встановленні злочинців та наданні доказів у суді. Надійність і точність комп'ютерно-технічної експертизи безпосередньо впливають на результати судових справ, тому важливо використовувати надійні методи та інструменти. На завершення, розширюючи знання та можливості комп'ютерно-технічної експертизи, ми прагнемо сприяти постійному вдосконаленню слідчих процесів, захисту цифрових активів та просуванню правосуддя в цифрову епоху.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. Ali M. An Introduction to Python Subprocess: Basics and Examples. Learn Data Science and AI Online | DataCamp. URL: https://www.datacamp.com/tutorial/python-subprocess (дата звернення: 19.06.2023).
2. Comparing Python to Other Languages. Python.org. URL: https://www.python.org/doc/essays/comparisons/ (дата звернення: 19.06.2023).
3. datetime – Basic date and time types. Python documentation. URL: https://docs.python.org/3/library/datetime.html (дата звернення: 19.06.2023).
4. ExifRead. PyPI. URL: https://pypi.org/project/ExifRead/ (дата звернення: 19.06.2023).
5. GitHub - anderskm/gputil: A Python module for getting the GPU status from NVIDA GPUs using nvidia-smi programmically in Python. GitHub. URL: https://github.com/anderskm/gputil (дата звернення: 19.06.2023).
6. GitHub - gregbanks/python-tabulate: fork of https://bitbucket.org/astanin/python-tabulate. GitHub. URL: https://github.com/gregbanks/python-tabulate (дата звернення: 19.06.2023).
7. json - JSON encoder and decoder. Python documentation. URL: https://docs.python.org/3/library/json.html (дата звернення: 19.06.2023).
8. Khosiyat M. Modern possibilities of computer technical examination in the investigation of crimes in the field of information technologies. International journal of advanced research. 2019. Т. 7, № 12. С. 1029–1032. URL: https://doi.org/10.21474/ijar01/10249 (дата звернення: 07.06.2023).
9. Lutkevich B. What is computer forensics (cyber forensics)?. Security. URL: https://www.techtarget.com/searchsecurity/definition/computer-forensics (дата звернення: 31.05.2023).
10. os – Miscellaneous operating system interfaces. Python documentation. URL: https://docs.python.org/3/library/os.html (дата звернення: 19.06.2023).

11. Platform Module in Python - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/platform-module-in-python/ (дата звернення: 19.06.2023).
12. psutil documentation – psutil 5.9.5 documentation. psutil documentation – psutil 5.9.5 documentation. URL: https://psutil.readthedocs.io/en/latest/ (дата звернення: 19.06.2023).
13. PyCharm: all about the most popular Python IDE. Data Science Courses | DataScientest. URL: https://datascientest.com/en/pycharm-all-about-the-most-popular-python-ide (дата звернення: 19.06.2023).
14. Real Python. Playing and Recording Sound in Python – Real Python. Python Tutorials – Real Python. URL: https://realpython.com/playing-and-recording-sound-python/ (дата звернення: 19.06.2023).
15. requests. PyPI. URL: https://pypi.org/project/requests/ (дата звернення: 19.06.2023).
16. Satyam K. Shutil Module in Python - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/shutil-module-in-python/ (дата звернення: 19.06.2023).
17. SoundFile – PySoundFile 0.10.3post1-1-g0394588 documentation. SoundFile – PySoundFile 0.10.3post1-1-g0394588 documentation. URL: https://pysoundfile.readthedocs.io/en/latest/ (дата звернення: 19.06.2023).
18. sqlite3 – 2.0 interface for SQLite databases. Python documentation. URL: https://docs.python.org/3/library/sqlite3.html (дата звернення: 19.06.2023).
19. uuid – UUID objects according to RFC 4122. Python documentation. URL: https://docs.python.org/3/library/uuid.html (дата звернення: 19.06.2023).
20. What is Python? Executive Summary. Python.org. URL: https://www.python.org/doc/essays/blurb/ (дата звернення: 19.06.2023).
21. wifi. PyPI. URL: https://pypi.org/project/wifi/ (дата звернення: 19.06.2023).
22. WMI. PyPI. URL: https://pypi.org/project/WMI/ (дата звернення: 19.06.2023).
23. Буряк Л. С. Парова природа висновку експерта як засобу доказування в цивільному процесі. Криміналістичний вісник • № 1 (21), 2014. 6 с.

24. Інструкція з охорони праці при роботі на персональному комп’ютері (зразок). Календарь бухгалтера. URL: https://services.uteka.ua/ua/publication/zrazky-34-trudovi-vidnosyny-ta-oplata-pratsi-138-instrukciya-po-oxrane-truda-pri-rabote-na-personalnom-kompyutere-obrazec (дата звернення: 27.05.2023).
25. Комп'ютерно-технічна експертиза – WikiLegalAid. Платформа правових консультацій - WikiLegalAid. URL: https://wiki.legalaid.gov.ua/index.php/Комп'ютерно-технічна_експертиза (дата звернення: 27.05.2023).
26. Кримінальний процесуальний кодекс України : Кодекс України від 13.04.2012 р. № 4651-VI : станом на 28 квіт. 2023 р. URL: https://zakon.rada.gov.ua/laws/show/4651-17#Text (дата звернення: 27.05.2023).
27. Про затвердження Інструкції про призначення та проведення судових експертиз та експертних досліджень та Науково-методичних рекомендацій з питань підготовки та призначення судових експертиз та експертних досліджень : Наказ М-ва юстиції України від 08.10.1998 р. № 53/5 : станом на 7 берез. 2023 р. URL: https://zakon.rada.gov.ua/laws/show/z0705-98#Text (дата звернення: 27.05.2023).
28. Про судову експертизу : Закон України від 25.02.1994 р. № 4038-XII : станом на 1 січ. 2023 р. URL: https://zakon.rada.gov.ua/laws/show/4038-12#Text (дата звернення: 27.05.2023).
29. Цивільний процесуальний кодекс України : Кодекс України від 18.03.2004 р. № 1618-IV : станом на 18 черв. 2023 р. URL: https://zakon.rada.gov.ua/laws/show/1618-15#Text (дата звернення: 27.05.2023).

ДОДАТКИ

Додаток А. Повна версія коду до програми

```
import requests
import uuid
import os
import json
from datetime import datetime
import exifread
import platform
import GPUtil
import psutil
import datetime
import os
import subprocess as sp
from tabulate import tabulate
from psutil._common import bytes2human
import wifi
import requests
import sounddevice as sd
import soundfile as sf
import datetime
import time
import sqlite3
import os
import shutil
import wmi
import signal
import keyboard


def print_menu():
    print("==== Меню ====")
    print("1. Дізнатися IP адресу")
    print("2. Дізнатися MAC Ethernet")
    print("3. Отримати метадані з фото")
    print("4. Загальна інформація про ПК")
    print("5. Геодата точок Wi-Fi")
    print("6. Записати звук з мікрофона")
    print("7. Зберегти всі дані з браузера")
    print("8. Отримати серійні номера ПК")
    print("0. Вийти з програми ")
    print("================")

def get_external_ip():
    response = requests.get('https://ifconfig.co/ip')
    ip = response.text.strip()
    print("Ваша зовнішня IP-адреса:", ip)

def get_mac():
    print ("Ваш MAC address це : ", end="")
    print (':'.join(['{:02x}'.format((uuid.getnode() >> elements) & 0xff)
    for elements in range(0,2*6,2)][::-1]))

def get_metadata():
    GEOJSON_LIST = []

    STATUS = {
        "success": 0,
        "fail": 0
```

```
    }

    def run():
        photos_path = "photos"
        photos = os.listdir(photos_path)
        for photo in photos:
            # escape hidden files
            if photo[0] == ".":
                continue
            # open files
            p = open(photos_path + "/" + photo, 'rb')
            try:
                date_location = get_date_location(p)
                GEOJSON_LIST.append(geojson_template(date_location))
                STATUS["Отримано"] += 1
            except:
                STATUS["Не отримано"] += 1
        write_results()

    def get_date_location(file):
        exif_dict = exifread.process_file(file)

        exif_datetime = exif_dict['EXIF DateTimeOriginal'].printable
        export_datetime = str(datetime.strptime(exif_datetime, '%Y:%m:%d
%H:%M:%S'))

        lon_ref = exif_dict["GPS GPSLongitudeRef"].printable
        lon = exif_dict["GPS GPSLongitude"].printable
        lat_ref = exif_dict["GPS GPSLatitudeRef"].printable
        lat = exif_dict["GPS GPSLatitude"].printable

        location = convert_location(lon_ref, lon, lat_ref, lat)

        return {
            "datetime": export_datetime,
            "location": location
        }

    def convert_location(lon_ref, longitude, lat_ref, latitude):
        # Longitude
        lon = longitude[1:-1].replace(" ", "").replace("/", ",").split(",")
        lon = float(lon[0]) + float(lon[1]) / 60 + float(lon[2]) / float(lon[3])
/ 3600
        if lon_ref != "E":
            lon = lon * (-1)

        # Latitude
        lat = latitude[1:-1].replace(" ", "").replace("/", ",").split(",")
        lat = float(lat[0]) + float(lat[1]) / 60 + float(lat[2]) / float(lat[3])
/ 3600
        if lat_ref != "N":
            lat = lat * (-1)

        # return [lat, lon, 20]
        return [lat, lon]

    def geojson_template(date_location):
        return {
            "type": "Feature",
            "geometry": {
                "type": "Point",
                "coordinates": date_location["location"]
            },
```

```
            "properties": {
                "datetime": date_location["datetime"],
            },

        }

    def write_results():
        # print(GEOJSON_LIST)
        print("Успішно оброблені метадані з {} locations, {} не
успішно".format(STATUS["success"], STATUS["fail"]))
        file = open("Geo.json", "w")
        file.write(json.dumps(GEOJSON_LIST, sort_keys=True, indent=4,
separators=(', ', ': ')))
        print('Файл з метаданими збережено!')

    run()

def get_main_system():
    uname = platform.uname()
    date = datetime.datetime.now()

    Mb = 1024 * 1024
    Gb = Mb * 1024
    def Round(nmup):
        nmup / (1024 * 1024 * 1024)
        return round(nmup, 2)

    """       System Informations
"""
    def system_informations():
        print("=" * 20, "System Information", "=" * 20)
        print(f"System    : {uname.system} {uname.release}")
        print(f"Users Name: {psutil.users()}")
        print(f"Version   : {uname.version}")
        print(f"Machine   : {uname.machine}")
        input('\nPress any key to continue.. ')
        sp.call('cls', shell=True)
    ""
    """    Gpu Informations
"""
    def Gpu_informations():
        gpus = GPUtil.getGPUs()
        list_gpus = []
        for gpu in gpus:
            gpu_id = gpu.id
            gpu_name = gpu.name
            gpu_load = f'{round(((gpu.load) * 100), 2)}%'
            gpu_free_M = f'{gpu.memoryFree}MB'
            gpu_Use_M = f'{gpu.memoryUsed}MB'
            gpu_Total_M = f'{gpu.memoryTotal}MB'
            gpu_temperature = f'{gpu.temperature}°C'
            gpu_version = f'{gpu.driver}'
            gpu_uuid = gpu.uuid
            list_gpus.append(
                (gpu_id, gpu_name, gpu_load, gpu_free_M, gpu_Use_M, gpu_Total_M,
gpu_temperature, gpu_version, gpu_uuid))
        print("=" * 50, "GPU Інформація", "=" * 50)
        print(tabulate(list_gpus, headers=(
        " id", "name", "load", "MemoryFree", "MemoryUsed", "MemoryTotal",
"Temperature", "Version", "Uuid"),
                       tablefmt="fancy_grid"))
        input("\nНажміть кнопку для продовження.. ")
        sp.call('cls', shell=True)
```

```
    ""
    """  Memory Informations
"""
    def memory_informations():
        print("=" * 25, "Інформація про оперативну память", "=" * 25)
        mem = psutil.virtual_memory()
        list_memory = []

        memory_total = f"{Round(mem.total / Gb)} GB"
        memory_use = f"{Round(mem.used / Mb)} MB | {mem.percent}%"
        memory_free = f"{Round(mem.free / Mb)} MB | {round(((mem.total -
mem.used) / mem.total) * 100, 2)}%"
        memory_speed = 121
        list_memory.append((memory_speed, memory_total, memory_use,
memory_free))

        print(tabulate(list_memory, headers=(" Name", "Total", "       Use", "
Free"), tablefmt="fancy_grid"))
        input("\nНатисніть кнопку для продовження..")
        sp.call('cls', shell=True)
    ""
    """  Process Informations
"""
    def Process():
        print("=" * 35, "Всі процеси ", "=" * 35)
        psutil.test()
        input("\nНатисніть кнопку для продовження.. ")
        sp.call('cls', shell=True)
    ""
    """   Disk Informations
"""
    def disk():
        print("="*20, "Інформація про дискові накопичувачі",
datetime.datetime.now().strftime('%x'), "="*20)
        list_total = []
        for part in psutil.disk_partitions(all=True):
            if os.name == 'nt':
                if 'cdrom' in part.opts or part.fstype == '':
                    continue
            usage = psutil.disk_usage(part.mountpoint)
            list_total.append((part.device, bytes2human(usage.total),
bytes2human(usage.used), bytes2human(usage.free), f"{usage.percent} %",
part.fstype))

        print(tabulate(list_total, headers=("Device", "Total", "Used", "Free",
"Use ", "Type"), tablefmt="fancy_grid"))
        input("\nНатисніть кнопку для продовження.. ")
        sp.call('cls', shell=True)
    ""


    def go():
        print("|", "=" * 20, date.strftime("%x | %X"), "=" * 20, "|")
        x = input(
            "1: Системна інформація\n2: GPU Інформація\n3: Cpu Інформація\n4:
Інформація про оперативну память\n5: Інформація про накопичувачі "
            "\n6: Повернутись в головне меню\nEnter : ")
        if x == "1":
            system_informations()
        elif x == "2":
            Gpu_informations()
        elif x == "3":
```

```
            memory_informations()
        elif x == "4":
            Process()
        elif x == "5":
            disk()
        elif x == "6":
            main()
        else:
            print("Error Renter")


    while True:
        go()

def get_geo_wifi():
    def get_available_wifi_networks():
        wifi_networks = []

        # Отримання списку доступних Wi-Fi мереж
        networks = wifi.Cell.all('wlp0s20f3')  # Замініть 'wlan0' на інтерфейс
Wi-Fi вашої системи

        for network in networks:
            wifi_networks.append({
                'ssid': network.ssid,
                'bssid': network.address,
                'channel': network.channel,
                'signal_strength': network.signal,
                'encryption_type': network.encryption_type
            })

        return wifi_networks

    # Отримання списку доступних Wi-Fi мереж
    available_networks = get_available_wifi_networks()

    # Виведення списку доступних Wi-Fi мереж
    for network in available_networks:
        print(f"SSID: {network['ssid']}")
        print(f"BSSID: {network['bssid']}")
        print(f"Канал: {network['channel']}")
        print(f"Рівень сигналу: {network['signal_strength']}")
        print(f"Тип шифрування: {network['encryption_type']}")
        print()

    def get_geo():
        for bssid_all in available_networks:
            bssid = f"{bssid_all['bssid']}"
            print(bssid)
            url =
f"https://api.mylnikov.org/geolocation/wifi?bssid={bssid}&data=open&v=1.1"
            print(url)
            get_geo = requests.get(url=url)
            result_json = get_geo.json()
            get_status = result_json['result']
            get_data = result_json['data']
            if get_status == 404:
                print('Нажаль найти геопозицію не вдалось')
            elif get_status == 200:
                print(f'{get_data}')
    get_geo()

def get_record_voice():
```

```
    # Параметри запису
    duration = 10  # Час запису одного голосового зразка
    sample_rate = 44100  # Частота запису

    stop_recording = False  # Флаг для зупинки запису

    def stop(e):
        nonlocal stop_recording
        stop_recording = True

    keyboard.on_release_key("esc", stop)  # Встановлення обробника на натискання
клавіші ESC

    while not stop_recording:
        # Генерація імені файлу з датою та часом
        current_time = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
        filename = f"recording_{current_time}.wav"

        # Запис аудіо
        print(f"Записую у файл: {filename}")
        audio = sd.rec(int(duration * sample_rate), samplerate=sample_rate,
channels=2)
        sd.wait()  # Очікування завершення запису
        sf.write(filename, audio, sample_rate)  # Збереження записаного аудіо в
файл

        print(f"Зберіг у файл: {filename}")

        # Пауза на 1 хвилину перед наступним записом
        time.sleep(60)

    keyboard.unhook_all()  # Звільнення обробника клавіші ESC


def get_save_cookies():
    # Директорія для збереження куків
    save_folder = "SaveCookies"
    os.makedirs(save_folder, exist_ok=True)

    # Отримання кукі з Google Chrome
    chrome_cookie_file = os.path.expanduser('~') +
r"\AppData\Local\Google\Chrome\User Data\Default\Cookies"
    shutil.copy2(chrome_cookie_file, os.path.join(save_folder,
"chrome_cookies"))

    # Отримання кукі з Mozilla Firefox
    firefox_cookie_file = os.path.expanduser('~') +
r"\AppData\Roaming\Mozilla\Firefox\Profiles\<профиль>.default\cookies.sqlite"
    shutil.copy2(firefox_cookie_file, os.path.join(save_folder,
"firefox_cookies.sqlite"))

    # Отримання кукі з Microsoft Edge
    edge_cookie_file = os.path.expanduser('~') +
r"\AppData\Local\Microsoft\Edge\User Data\Default\Cookies"
    shutil.copy2(edge_cookie_file, os.path.join(save_folder, "edge_cookies"))

    # Отримання кукі з Google Chrome
    chrome_conn = sqlite3.connect(os.path.join(save_folder, "chrome_cookies"))
    chrome_cursor = chrome_conn.cursor()
    chrome_cursor.execute("SELECT name, value FROM cookies")
    chrome_cookies = chrome_cursor.fetchall()
    chrome_conn.close()
```

```
    # Отримання кукі з Mozilla Firefox
    firefox_conn = sqlite3.connect(os.path.join(save_folder,
"firefox_cookies.sqlite"))
    firefox_cursor = firefox_conn.cursor()
    firefox_cursor.execute("SELECT name, value FROM moz_cookies")
    firefox_cookies = firefox_cursor.fetchall()
    firefox_conn.close()

    # Отримання кукі з Microsoft Edge
    edge_conn = sqlite3.connect(os.path.join(save_folder, "edge_cookies"))
    edge_cursor = edge_conn.cursor()
    edge_cursor.execute("SELECT name, value FROM cookies")
    edge_cookies = edge_cursor.fetchall()
    edge_conn.close()

    # Вивід та збереження кукі
    for cookie in chrome_cookies:
        print(f"Chrome Cookie - Name: {cookie[0]}, Value: {cookie[1]}")
        with open(os.path.join(save_folder, "chrome_cookies.txt"), "a") as f:
            f.write(f"Name: {cookie[0]}, Value: {cookie[1]}\n")

    for cookie in firefox_cookies:
        print(f"Firefox Cookie - Name: {cookie[0]}, Value: {cookie[1]}")
        with open(os.path.join(save_folder, "firefox_cookies.txt"), "a") as f:
            f.write(f"Name: {cookie[0]}, Value: {cookie[1]}\n")

    for cookie in edge_cookies:
        print(f"Edge Cookie - Name: {cookie[0]}, Value: {cookie[1]}")
        with open(os.path.join(save_folder, "edge_cookies.txt"), "a") as f:
            f.write(f"Name: {cookie[0]}, Value: {cookie[1]}\n")

def get_all_serios():
    def get_motherboard_serial_number():
        c = wmi.WMI()
        for board in c.Win32_BaseBoard():
            return board.SerialNumber

    def get_hard_drive_serial_number():
        c = wmi.WMI()
        for drive in c.Win32_DiskDrive():
            return drive.SerialNumber

    def get_ram_serial_number():
        c = wmi.WMI()
        for module in c.Win32_PhysicalMemory():
            return module.SerialNumber

    # Отримання серійного номера материноської плати
    motherboard_serial = get_motherboard_serial_number()
    print("Серійний номер материнської плати:", motherboard_serial)

    # Отримання серійного номера жорсткого диску
    hard_drive_serial = get_hard_drive_serial_number()
    print("Серійний номер жорсткого диска:", hard_drive_serial)

    # Отримання серійного номера оперативної памяті
    ram_serial = get_ram_serial_number()
    print("Серійний номер оперативної памяті:", ram_serial)

def main():
    while True:
        print_menu()
        choice = input("Виберіть пункт з меню: ")
```

```
if choice == "1":
    get_external_ip()
    input("Натисніть Enter, щоб повернутись в головне меню...")
    if os.name == 'nt':  # Для Windows
        os.system('cls')
    else:  # Для Unix-подібних систем (Linux, macOS)
        os.system('clear')
elif choice == "2":
    get_mac()
    input("Натисніть Enter, щоб повернутись в головне меню...")
    if os.name == 'nt':  # Для Windows
        os.system('cls')
    else:  # Для Unix-подібних систем (Linux, macOS)
        os.system('clear')
elif choice == "3":
    get_metadata()
    input("Натисніть Enter, щоб повернутись в головне меню...")
    if os.name == 'nt':  # Для Windows
        os.system('cls')
    else:  # Для Unix-подібних систем (Linux, macOS)
        os.system('clear')
elif choice == "4":
    get_main_system()
    input("Натисніть Enter, щоб повернутись в головне меню...")
    if os.name == 'nt':  # Для Windows
        os.system('cls')
    else:  # Для Unix-подібних систем (Linux, macOS)
        os.system('clear')
elif choice == "5":
    get_geo_wifi()
    input("Натисніть Enter, щоб повернутись в головне меню...")
    if os.name == 'nt':  # Для Windows
        os.system('cls')
    else:  # Для Unix-подібних систем (Linux, macOS)
        os.system('clear')
elif choice == "6":
    get_record_voice()
    input("Натисніть Enter, щоб повернутись в головне меню...")
    if os.name == 'nt':  # Для Windows
        os.system('cls')
    else:  # Для Unix-подібних систем (Linux, macOS)
        os.system('clear')
elif choice == "7":
    get_save_cookies()
    input("Натисніть Enter, щоб повернутись в головне меню...")
    if os.name == 'nt':  # Для Windows
        os.system('cls')
    else:  # Для Unix-подібних систем (Linux, macOS)
        os.system('clear')
elif choice == "8":
    get_all_serios()
    input("Натисніть Enter, щоб повернутись в головне меню...")
    if os.name == 'nt':  # Для Windows
        os.system('cls')
    else:  # Для Unix-подібних систем (Linux, macOS)
        os.system('clear')
elif choice == "0":
    print("Вийти з программи")
    exit()
else:
    print("Не правильний выбор. Спробуйте знову.")
```

```
if __name__ == '__main__':
    main()
```