МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ

ФАКУЛЬТЕТ №2
Кафедра інформаційних технологій

Інтелектуальний аналіз аудіоконтенту для виявлення екстремістських і заборонених матеріалів

кваліфікаційна робота
здобувача вищої освіти
4 курсу денної форми навчання
Анастасії КОТЕНКО

Науковий керівник:
доктор філософії
Олег БАСИСТЮК

Рецензент:
_____
_____

Кваліфікаційна робота допущена до захисту
«20» травня 2026 р., протокол № 35
Завідувач кафедри інформаційних технологій
_____**Олег ЗАЧЕК**
(підпис)

Львів 2026

АНОТАЦІЯ

КОТЕНКО А. Інтелектуальний аналіз аудіоконтенту для виявлення екстреміських і заборонених матеріалів. - Рукопис.

Кваліфікаційна робота бакалавра зі спеціальності 126 «Інформаційні системи та технології». Львівський державний університет внутрішніх справ, МВС України, Львів, 2026

В дипломній роботі було проведено дослідження можливих способів виявлення естреміських матеріалів в аудіоконтенті. Для практичної частини був написаний код на Python із використанням технологій Speech-to-Text та NPL який аналізує аудіофайл, трактує в текст, визначає тон голосу, виявляє забороненні слова та робить аналітику аудіофайлу.

Ключові слова: аудіоконтент, інтелектуальний аналіз даних, NPL, Python, ектреміський контент, класифікація тексту, розпізнавання мовлення, машинне навчання.

ABSTRACT

KOTENKO A. Intelligent Audio Content Analysis for Detecting Extremist and Prohibited Materials. - Manuscript.

Bachelor’s Qualification Thesis in Speciality 126 "Information Systems and Technologies". Lviv State University of Internal Affairs, Ministry of Internal Affairs of Ukraine, Lviv, 2026.

The thesis researched possible ways to detect extremist materials in audio content. For the practical part, a code was written in Python using Speech-to-Text and NPL technologies that analyzes an audio file, interprets it into text, determines the tone of voice, detects prohibited words, and performs audio file analytics.

Keywords: audio content, data mining, NLP, Python, extremist content, text classification, speech recognition, machine learning.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ASR – Automated Speech Recognition (Автоматичне розпізнавання мовлення)

CTranslate2 – Оптимізований рушій виведення (Inference Engine) для моделей Transformer

FFT – Fast Fourier Transform (Швидке перетворення Фур'є для аналізу аудіосигналу)

MFCC – Mel-Frequency Cepstral Coefficients (Мел-кепстральні коефіцієнти для виділення ознак мови)

NLP – Natural Language Processing (Обробка природної мови / лінгвістичний аналіз)

RTF – Real Time Factor (Коефіцієнт реального часу)

STT – Speech-to-Text (Конвертація усного мовлення в текстовий формат)

VAD – Voice Activity Detection (Алгоритм детекції голосової активності / фільтрація тиші в аудіопотоці)

VMM – Voice Monitoring Module (Модуль моніторингу та контенту аудіосигналів)

WER – Word Error Rate (Коефіцієнт помилково розпізнаних слів; основна метрика точності Whisper)

Whisper – Архітектура нейромережевих моделей від OpenAI для багатомовного розпізнавання мовлення

ВСТУП

Актуальність теми. Останніми роками відстеження та блокування небезпечного контенту в мережі перетворилося на одну з найсерйозніших проблем безпеки. Організатори радикальних рухів та інформаційних операцій дедалі частіше відмовляються від класичних текстових дописів на користь звукових та відеоформатів. Пропаганда, заклики до протиправних дій чи прихована координація тепер масово поширюються через голосові повідомлення в месенджерах, записані подкасти, закриті чати та аудіодоріжки на популярних відеоплатформах. Для аналітиків правоохоронних структур у працях [1 - 3] це створює серйозний виклик, оскільки виявити деструктивний зміст у потоці звукових даних набагато складніше, ніж у звичайному тексті.

Методи моніторингу, які використовуються на практиці зараз, переважно спираються на людський фактор, фахівці змушені вручну переслуховувати години звукозаписів. Зрозуміло, що такий підхід неефективний при сучасних обсягах інформації, забирає надто багато часу та не дозволяє вчасно реагувати на потенційні загрози. Виходом із цієї ситуації є автоматизація моніторингу за допомогою сучасних інформаційних технологій. Поєднання інструментів автоматичного розпізнавання мовлення з алгоритмами аналізу тексту дає змогу швидко знаходити небезпечні маркери у звукових файлах. Разом із тим, з міркувань таємності та захисту внутрішньої інформації, такі програмні комплекси мають працювати локально на комп’ютерах підрозділів, не відправляючи записи на сторонні інтернет-платформи чи хмарні сервери.

Зв'язок роботи з науковими програмами, планами, темами. Тематика дослідження безпосередньо інтегрована у план наукових пошуків кафедри інформаційного та аналітичного забезпечення діяльності правоохоронних органів Львівського державного університету внутрішніх справ і стосується створення нових комп’ютерних засобів протидії загрозам у кіберпросторі.

Мета дослідження. Метою цієї роботи є розробка, програмна реалізація та тестування автономної інформаційної системи, яка здатна самостійно аналізувати аудіофайли, переводити їх у текст та виявляти збіги із базою заборонених чи екстремістських висловлювань.

Завдання дослідження. Для досягнення цієї мети у роботі послідовно вирішувалися такі завдання: вивчення наявного досвіду контролю медіапростору та порівняння сучасних моделей розпізнавання голосу; формування технічних вимог та розробка загальної структури системи семантико-акустичного аналізу; оптимізація нейромережевих алгоритмів Faster-Whisper, інтеграція інструментів відсікання тиші та обчислення швидкості вимови; побудова математичної логіки для оцінювання рівнів небезпеки контенту та визначення його емоційного забарвлення; написання коду програми мовою Python із використанням бібліотеки Streamlit для побудови інтерфейсу; проведення тестів для оцінки швидкості обробки файлів на центральному процесорі та відеокарті за різних параметрів точності.

Об’єкт дослідження. Процеси цифрової обробки звукових сигналів, автоматичної транскрибації та семантичного аналізу тексту під час правоохоронного моніторингу.

Предмет дослідження. Методи та засоби пошуку деструктивних слів у транскрипті аудіозапису з одночасною фіксацією часових параметрів звукового потоку.

Методи дослідження. Наукові результати отримано завдяки застосуванню методів системного аналізу для побудови логіки додатку, теорії штучних нейромереж для перетворення звуку на текст, методів лінгвістичного аналізу для токенізації та контекстного пошуку слів. Оцінка швидкодії створеного програмного забезпечення засновувалася на проведенні обчислювальних експериментів та інструментах математичної статистики.

Наукова новизна отриманих результатів. Новизна полягає у створенні оптимізованого алгоритмічного підходу, який дозволяє в єдиному циклі обробки даних не лише знаходити небезпечні текстові фрази, а й

вираховувати щільність слів на секунду для виявлення аномального темпу мовлення, для оптимізації автоматичної моделі яка для перевірки працівників зазначає секунди де було визначено активність, забезпечуючи швидку роботу програми без використання дорогих серверних потужностей.

Практичне значення отриманих результатів. Створено працездатний автономний додаток на базі Python, який можна використовувати в аналітичних відділах поліції чи інших безпекових структурах. Програма сама формує кінцеві звіти, фіксує точний час (секунди) вимови заборонених слів, будує графіки інтенсивності розмови та автоматично виставляє оцінку ризику, що суттєво зменшує навантаження на аналітиків.

Апробація результатів дослідження. Окремі результати розробки системи та алгоритми обробки даних обговорювалися і доповідалися на наукових семінарах кафедри інформаційного та аналітичного забезпечення діяльності правоохоронних органів ЛвДУВС у дві тисячі двадцять шостому році. Також було створено опитування серед експертів для того щоб зрозуміти актуальність та практичність зробленої програми.

Структура та обсяг роботи. Пояснювальна записка складається зі вступу, трьох розділів, загальних висновків, переліку використаних джерел та додатків. Основний зміст роботи викладений на 61 сторінках друкованого тексту. Робота містить 15 ілюстрацій, 2 таблиць та 32 найменування у списку використаних джерел. Додатки до роботи разом із переліком джерел викладені на 12 сторінках, включно з вихідним кодом програми. Загальний обсяг кваліфікаційної роботи становить 78 сторінок.

РОЗДІЛ 1. ОСНОВИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ АУДІОКОНТЕНТУ В ПРАВООХОРОННІЙ ДІЯЛЬНОСТІ

1.1. Моніторинг цифрового медіапростору як інструмент протидії екстремізму та поширенню заборонених матеріалів

Нинішній світовий та вітчизняний інфопростір обернувся на головну арену для гібридних сутичок, поширення радикальних ідей, проведення руйнівних ІПСО та злагодження незаконної діяльності. Якщо раніше порушники й організатори екстремістських угруповань переважно послуговувалися письмовими маніфестами, приватними вебфорумами чи друкованими листівками, то тактика ведення інфовійни вже кардинально змінилася.

Мультимедійний контент став чи не головним засобом впливу на громадську думку та дестабілізації політичної ситуації. Цей зсув пояснюється психологічними закономірностями сприйняття: осмислення аудіовізуальної інформації вимагає від споживача набагато менших розумових зусиль, а емоційний відгук від живого мовлення, виразної інтонації або динамічного відеоряду є значно потужнішим, аніж від нерухомого тексту. Як наслідок, заклики до екстремізму, руйнівні матеріали, інструкції зі створення зброї та пропаганда насильства тепер масово вбудовуються просто у звукові доріжки медіафайлів.

Окрему складність для аналітичних підрозділів правоохоронних органів та структур нацбезпеки у працях [4, 6, 7] являє собою некерований обіг коротких голосових повідомлень у популярних анонімних месенджерах, формування тематичних аудіосерій (подкастів), трансляції у прямому ефірі на відкритих платформах, а також розповсюдження коротких відео із озвученням у соцмережах. Зловмисники свідомо обирають саме ці канали для поширення, адже більшість існуючих автоматизованих систем для аналізу контенту були створені з прицілом на виключно текстові дані. Стандартні пошукові механізми легко виявляють заборонену лексику у цивільному тексті чи коментарях, проте вони виявляються абсолютно марними, коли та сама

протиправна думка лунає в аудіозаписі або прихована під час тривалого стріму.

За таких обставин правоохоронці часто змушені покладатися на ручну перевірку інфополя, що вже повністю вичерпало свій технічний ресурс і довело свою неспроможність протистояти сучасним викликам. Співробітник аналітичного відділу фізично не в змозі прослухати та перевірити тисячі годин аудіозаписів, що генеруються щохвилини. До того ж, людський фактор вносить суттєву помилку: тривала, одноманітна робота з акустичним матеріалом неминуче спричиняє швидку втому, розфокусування уваги та пропуск критично важливих слів, географічних назв або інтонаційних сигналів.

Також виникає критична часова затримка (лаг) між моментом публікації шкідливого контенту і часом його виявлення аналітиком. Поки фахівець вручну опрацьовує й прослуховує один підозрілий запис, небезпечна інформація, завдяки рекомендаційним алгоритмам соцмереж, встигає охопити тисячі користувачів, що робить подальші заходи правоохоронців несвоєчасними та неефективними.

Додатковою особливістю, що ускладнює цифровий моніторинг, є невпинна зміна й ускладнення внутрішньої структури небажаного мовленнєвого контенту. Організатори інформаційних атак все частіше відмовляються від прямої агресії, натомість застосовуючи методи нейролінгвістичного програмування, приховані сенси, алегорії та закодовану термінологію (dog whistles), що допомагає обходити базові фільтри. Водночас вони намагаються штучно впливати на емоційний стан аудиторії, використовуючи акустичні прийоми: різку зміну швидкості мовлення, підвищення гучності голосу, штучний акцент на певних частотах чи додання фонового шуму.

Тому для дієвого контролю медіапростору потрібен негайний перехід від простого спостереження до впровадження інтелектуальних систем семантичного аналізу. Такі системи мають працювати як універсальні

конвеєри, здатні не лише перетворювати звук на текст, але й обчислювати динаміку розмови, фіксувати збої в мовленнєвій активності та миттєво порівнювати отримані слова з гнучкими, постійно оновлюваними базами семантичних маркерів. Навіть якщо в розмові відсутні прямі заборонені слова, система мусить аналізувати параметри мовця на предмет прихованого кодування у працях [21, 25] чи штучного емоційного нагнітання, вказуючи оператору на конкретні моменти (таймкоди) в аудіозаписі, що потребують детального експертного вивчення.

Важливим чинником, що вимагає розробки та впровадження подібних інструментів, є жорсткі вимоги до конфіденційності та захисту відомостей. Переважна більшість комерційних рішень на базі ШІ та великих мовних моделей працює за хмарною схемою (Cloud AI). Це зобов'язує завантажувати вхідний аудіофайл на віддалені хмарні сервери, які здебільшого розташовані за кордоном та контролюються міжнародними корпораціями.

Для силових структур, підрозділів кіберзахисту у працях [5, 8] та аналітичних центрів України, які щодня працюють із матеріалами слідства, оперативними даними та відомостями, що є службовою або державною таємницею, використання хмарних сервісів є цілком неприпустимим. Будь-яке надсилання файлу у зовнішню мережу тягне за собою прямі ризики несанкціонованого витоку даних, викриття джерел інформації та порушення чинного законодавства щодо захисту персональних даних та державної таємниці.

Саме тому створення та розгортання локальних (on-premise) інформаційних систем, здатних здійснювати швидкий та точний аналіз великих обсягів аудіоданих на стандартних робочих місцях аналітичних підрозділів без доступу до Інтернету, є найактуальнішим пріоритетом у розробці відомчого програмного забезпечення.

Таким чином спроектований локальний обчислювальний контур дає змогу повністю ізолювати процес обробки даних у межах захищеного периметра установи. Комплексна автоматизація початкового лінгвістичного

аналізу дозволяє докорінно змінити саму модель роботи аналітичних підрозділів перевести її з пасивного режиму наздоганяння та ліквідації наслідків поширення шкідливих матеріалів у режим випередження, проактивного виявлення джерел загроз та оперативного гасіння інформаційних ризиків на початкових етапах їх виникнення.

У контексті архітектурного проєктування та конфігурування мовних баз даних, необхідно адаптувати процес автоматичної обробки акустичного спектра для виокремлення чітко окреслених деструктивних тенденцій. Аналіз поточного стану медіасфери дозволяє здійснити класифікацію головних векторів інформаційних загроз, що приховуються у звукових даних, за такими основними групами у працях [8, 21]:

1. Організація забороненої діяльності та диверсійних заходів. Зафіксовування звукових фрагментів, що містять прямі розпорядження, географічні прив'язки, часові маркери або зашифровані інструкції щодо здійснення несанкціонованих акцій, підпалів об'єктів стратегічного значення чи блокування шляхів сполучення.

2. Пропаганда радикалізму та ворожнечі між народами. Виявлення мовних сигнатур, спрямованих на ескалацію суспільної напруги, закликів до насильницької зміни конституційного устрою, а також розповсюдження мови ворожнечі (hate speech) проти певних соціальних чи етнічних спільнот.

3. Приховані маніпулятивні кампанії (ІПСО). Розпізнавання аудіоконтенту, сгенерованого з метою штучного нарощування паніки, поширення навмисно неправдивої інформації стосовно національної безпеки, дискредитації державних структур та підриву морального духу населення через контрольовані канали дистрибуції.

4. Нелегальний обіг товарів та випадки корупції. Фіксація специфічної кримінальної лексики у записаних розмовах, яка свідчить про обговорення нелегального переміщення зброї, контрабандних операцій або формування протиправних фінансових домовленостей (тіньових економічних схем).

1.2. Сучасні методи автоматичного розпізнавання мовлення

Конвертація аудіоданих у текстовий формат являє собою перший та ключовий етап у процесі глибинного аналізу звукового контенту. У сучасних галузях комп'ютерної лінгвістики та інженерії даних цей процес відомий під абревіатурою Speech-to-Text (STT) або як автоматичне розпізнавання мовлення (ASR, Automatic Speech Recognition). Протягом тривалого часу розпізнавання ґрунтувалося на традиційних статистичних методах, зокрема на застосуванні прихованих Марковських моделей (HMM) та N-грам акустичних моделей. Ці системи вимагали скрупульозного покрокового налаштування для кожного окремого користувача, були надзвичайно вразливими до зовнішніх шумів і демонстрували недостатню точність при роботі з живою, непідготовленою мовою. Однак із появою архітектур глибокого навчання на базі нейронних мереж, таких як Transformer, технології розпізнавання пережили значний інженерний прорив, що дозволило аналізувати мовленнєві потоки з точністю, майже ідентичною людському сприйняттю.

На глобальному IT-ринку наразі домінують два принципово різні шляхи створення ASR-систем: комерційні хмарні сервіси та локально розгорнуті нейромережеві моделі з відкритим вихідним кодом. До першої категорії належать хмарні продукти від таких технологічних титанів, як Google Cloud Speech-to-Text, Microsoft Azure Speech та Amazon Transcribe. Вони забезпечують високу швидкість обробки та розширену багатомовну підтримку, оскільки використовують потужні обчислювальні кластери та постійно оновлювані віддалені сервери.

Проте для застосування у сферах правоохоронної діяльності чи корпоративної безпеки хмарна модель має критичний у праці [26], майже нездоланний мінус повну відсутність гарантій збереження конфіденційності. Передача аудіозаписів, які зазвичай містять таємниці слідства, оперативні відомості чи персональні дані громадян, на зовнішні сервери (які часто географічно розташовані за межами юрисдикції держави) є абсолютно

неприпустимою відповідно до національних законів про інформаційну безпеку та внутрішніх регламентів державних органів.

З огляду на це, у захищених IT-середовищах пріоритет надається винятково локальним рішенням, серед яких безумовним лідером є сімейство моделей Whisper від OpenAI. Ця нейромережа збудована на основі повнозв'язної архітектури Transformer, що складається з енкодера та декодера. Модель була навчена на колосальному масиві слабо анотованих аудіоматеріалів обсягом понад 680 000 годин, що дало їй унікальну здатність ефективно розпізнавати мовлення навіть за умов значних акустичних перешкод, специфічних акцентів, змішування мов або низької якості запису з портативних диктофонів чи телефонів.

Математичною особливістю Whisper є те, що вона оперує не лише ізольованими фонемами чи окремими звуками, а використовує механізм двоспрямованої уваги (Self-Attention). Перед початком обробки необроблений звуковий сигнал трансформується за допомогою швидкого перетворення Фур'є у 80-канальну мел-спектрограму візуалізацію частот, адаптовану під патерни людського слуху. Енкодер стискає цю спектрограму, а декодер, аналізуючи лінгвістичний контекст уже згенерованих слів, пророкує наступні токени. Завдяки такому глибокому контекстуальному аналізу система автоматично вставляє пунктуацію, розрізняє слова з однаковим звучанням, але різним змістом, та правильно підбирає лексику. Це є вирішальним для аналізу матеріалів, пов'язаних із правоохоронною діяльністю, де зміна одного знака пунктуації чи коректна інтерпретація спеціалізованого терміна може кардинально змінити юридичну кваліфікацію всього матеріалу.

Незважаючи на високу точність, оригінальна версія Whisper виявилася занадто обтяжливою та ресурсовитратною для повномасштабного розгортання на типових робочих місцях аналітиків. Конвертація тривалих аудіофайлів за допомогою вихідного коду на PyTorch у праці [24] вимагає значних ресурсів відеопам'яті графічних процесорів, а робота в режимі CPU займає надто багато часу, що унеможливлює моніторинг у реальному часі.

Цю інженерну проблему було успішно подолано з появою оптимізованого варіанту Faster-Whisper. Його основа це повна реімплементація обчислювальних ядер оригінальної нейромережі на мові C++ із використанням спеціалізованої бібліотеки CTranslate2. Така оптимізація реалізує техніку квантування ваг моделі (перехід від високоточного формату FP32 до більш компактних FP16 або INT8) у праці [10]. Зниження бітової точності матричних операцій дозволяє мінімізувати потреби в пам'яті у 2-4 рази, завдяки чому модель функціонує значно швидше, зберігаючи при цьому практично незмінними показники точності розпізнавання тексту (WER, Word Error Rate).

Для практичного застосування найважливішим фактом є те, що Faster-Whisper демонструє високу пропускну здатність обробки даних навіть на стандартних багатоядерних центральних процесорах (*CPU*) без необхідності мати дорогі дискретні відеокарти. Це уможливлює впровадження системи на базі вже наявної комп'ютерної техніки без додаткових фінансових інвестицій в апаратну частину.

Аналіз актуальних технологічних рішень підтверджує, що найбільш оптимальним вибором для створення системи автоматизованого контролю аудіо є використання саме оптимізованих локальних моделей на базі Transformer-архітектури. Комбінація здатності Faster-Whisper до глибокого контекстуального аналізу та її мінімальних вимог до обладнання дозволяє розгорнути повноцінну аналітичну станцію безпосередньо усередині захищеного периметру. Такий підхід повністю усуває ризики несанкціонованого доступу до даних, гарантує повну автономність алгоритмів у локальному середовищі та надає аналітикам швидкий засіб для первинної обробки великих обсягів аудіо, формуючи надійну математичну та лінгвістичну основу для подальших етапів, осмислення змісту та семантичного аналізу отриманого тексту.

1.3. Інструменти автоматизованого аналізу тексту та виявлення небезпечної лексики у звукових матеріалах

По відносно успішному перетворенню аудіоданих на текстовий масив за допомогою нейронних мереж для розпізнавання мовлення, виникає неминуча потреба інтелектуально опрацювати отримані лінгвістичні відомості. Сама по собі розшифрована текстова послідовність на початковій стадії є лише суцільним, неорганізованим потоком слів, де відсутні як розділові знаки, так і чіткі граматичні зв'язки чи очевидні індикатори загрозливого контенту. Щоб комп'ютерна програма могла незалежно орієнтуватися в цьому інформаційному обсязі, виявляючи ознаки порушень чи приховані ризики, впроваджуються технології опрацювання природної мови, відомі в інженерії як NLP (Natural Language Processing). Саме ці інструментарій дають змогу автоматизувати процес осмислення змісту, переносячи роботу з рівня простого зіставлення символів на рівень аналізу ситуативного значення, суті та імплікацій мовця.

Комп’ютерний аналіз у праці [23] розшифрованого тексту зазвичай стартує з фази його попереднього туалету: нормалізації та очищення. Необроблений текст спершу розбивається на мінімальні лінгвістичні складові, слова та усталені словосполучення, що у сфері комп’ютерних наук зветься токенізацією. Після цього запускається етап морфологічного опрацювання, де задіюються алгоритми стемінгу або лематизації. Стеммінг є простішим та швидшим варіантом, оскільки він брутально відсікає закінчення слів за допомогою правил, залишаючи лише незмінну основу. Однак, зважаючи на те, що українська мова має досить складну систему флексій (зміни за родами, числами, відмінками та часами), застосування стемінгу нерідко призводить до спотворення сенсу чи помилкового об'єднання абсолютно різних за значенням лексем.

Враховуючи таку ситуацію, у сучасних системах контролю перевагу надають саме лематизації. Цей підхід базується на повних морфологічних словниках та граматичних настановах, завдяки чому він зводить кожне

знайдене слово до його еталонної початкової форми: для іменників це однина називного відмінка, для дієслів — інфінітив. Без такої попередньої обробки система безпеки шукатиме винятково точний графічний збіг (наприклад, слово «замах») і повністю ігноруватиме варіанти на кшталт «замахали», «замахів» чи «замахуємося», що фактично звело б ефективність автоматичного моніторингу до нуля.

Першим та найбільш прямолінійним засобом для виявлення потенційно небезпечного вмісту в інформаційних системах історично є метод пошуку за контекстом на основі баз семантичних маркерів. Така база являє собою впорядкований лексикон ключових слів та фраз, які безпосередньо сигналізують про екстремістські наміри, планування диверсійних дій, закличність до масових безладів, корупційні схеми або пропаганду агресії. Програмний механізм послідовно зіставляє кожен токен із цією довідковою базою. Головна перевага такого підходу полягає в надзвичайно високій швидкості опрацювання, адже пошук у добре оптимізованих хеш-таблицях виконується за фіксований час. Щойно в розпізнаному тексті з'являється лексема із заздалегідь визначеного переліку ризиків, програма миттєво реєструє інцидент та сповіщає оператора.

Проте, традиційний пошук за словником має істотний недолік — він абсолютно безсилий перед лицем семантичної неоднозначності (омонімії та полісемії), яка є невід'ємною частиною живої мови. Наприклад, слово «напад» майже завжди є індикатором небезпеки, тоді як слово «гілка» може позначати як частину дерева у звичайній розмові про флору, так і метафоричну структуру в кримінальному контексті. Аби мінімізувати кількість хибних спрацювань та дати точнішу оцінку інформаційному полю, теперішні NLP-інструменти використовують складніші статистичні та векторні методи аналізу.

Зокрема, такі методи, як TF-IDF, дають змогу визначати значущість слова в межах конкретної бесіди, порівнюючи його частотність із великим масивом стандартної повсякденної лексики. Якщо якийсь специфічний термін починає незвично часто з'являтися протягом короткого інтервалу, система

позначає цю аудіозапис як таку, що заслуговує на увагу. Більш передовим рішенням є застосування векторних зображень слів (Word Embeddings), де кожне слово проєктується у вигляді вектора у багатовимірному семантичному просторі. У цьому просторі слова, близькі за сенсом (приміром, «зброя», «автомат», «пістолет»), опиняються поряд, навіть якщо їхнє написання зовсім різне. Це дає змогу системі розпізнавати загрозливий контекст, навіть коли зловмисники вдаються до синонімів чи намагаються замаскувати свої наміри.

Для впорядкування існуючих методів та обґрунтування вибору математичної основи й алгоритмічного апарату системи, що проектується, необхідно провести зіставний аналіз головних технологій семантичного аналізу тексту. Кожен із розглянутих підходів має свої обмеження стосовно необхідних обчислювальних ресурсів, вимог до оперативної пам'яті та рівня точності у виявленні контексту в зашумлених, специфічних для правоохоронної діяльності масивах даних. Зведений аналітичний огляд базових NLP-інструментів наведено в табл. 1.1.

Таблиця 1.1

Порівняльна характеристика методів автоматизованого аналізу деструктивного контенту

Метод аналізу тексту	Швидкість обробки (Асимптотична складність)	Основні переваги підходу	Ключові недоліки та обмеження	Ефективність для української мови
Прямий словниковий пошук (Хеш-таблиці)	Найвища *(O(1))*	Мінімальне навантаження на CPU, миттєве спрацювання на точні тригери,	Повна сліпота до омонімів, синонімів та завуальованого сленгу; велика кількість хибних	Низька (вимагає обов'язкової попередньої лематизації флексій).

		простота оновлення бази.	спрацювань.	
Статистичний аналіз (TF-IDF)	Висока *(O(N·logN))*	Враховує унікальність слів у контексті діалогу, добре виявляє аномальні сплески специфічної лексики.	Не розуміє смислових зв'язків між різними словами, ігнорує структуру та порядок слів у реченні.	Середня (ефективно працює лише на великих обсягах тексту).
Векторні ембедінгів (Word2Vec / FastText)	Середня (*O(D)*, де *D* розмірність вектора)	Враховує семантичну близость слів, розпізнає приховані загрози через синоніми та контекстні асоціації.	Вимагає значних ресурсів оперативної пам'яті для зберігання багатовимірних матриць ваг моделей.	Висока (особливо FastText, що враховує субслова та морфологію).

Аналіз даних, наведених у табл. 1.1, дозволяє зробити висновок, що побудова безкомпромісно точної правоохоронної системи моніторингу неможлива на базі лише одного ізольованого методу. Саме тому в архітектурі нашого програмного комплексу було реалізовано гібридний інженерний підхід: поєднання локальної швидкодіючої хеш-бази еталонних лем-ключів

(для миттєвого відсікання явних інцидентів) із контекстним аналізом структурних сегментів мовлення.

Окремим важливим вектором розвитку технологій аналізу є автоматичне оцінювання загальної емоційної тональності розмови аналіз сентименту (*Sentiment Analysis*). Завдяки відстеженню комбінацій слів, використання експресивної лексики, знаків оклику або специфічних мовних зворотів, алгоритми здатні визначати рівень агресії, тривожності, паніки чи радикалізації у висловлюваннях мовця. Для правоохоронних органів та аналітиків безпеки це відкриває принципово нові можливості: система стає здатною виявляти не лише прямі, очевидні загрози, а й приховуну психологічну напругу, маніпулятивні техніки чи спроби координації дій груп людей, що дуже характерно для професійно підготовлених пропагандистських кампаній чи деструктивних інформаційно-психологічних операцій (ІПСО).

Таким чином, інтеграція методів обробки природної мови є логічним і необхідним завершенням процесу інтелектуального аналізу аудіоданих. Якщо передові технології нейромережевого розпізнавання мовлення (наприклад, Faster-Whisper) відповідають за умовний «слух» інформаційної системи, то алгоритми NLP виконують роль її аналітичного «мислення». Поєднання цих інженерних рішень в один конвеєр дозволяє повністю мінімізувати людський фактор на етапі первинного моніторингу великих обсягів звукових матеріалів. Система бере на себе рутинну роботу з фільтрації у праці [11] безпечного інформаційного шуму, дозволяючи аналітикам зосередити обмежені людські ресурси виключно на тих матеріалах, які дійсно містять реальні ознаки інформаційних, кримінальних чи національних загроз.

РОЗДІЛ 2. АРХІТЕКТУРА ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АУДІОКОНТРОЛЮ

2.1. Системні вимоги та концептуальна схема процесу автоматизованого аудіоконтролю

Створення та вдосконалення дієвої інформаційної системи правоохоронного спрямування вимагає чіткої демаркації архітектурних кордонів, у котрих вона мусить функціонувати. Етапом, що передує розробці будь-якого програмного інструментарію для потреб структур безпеки, є формування деталізованих вимог до системи, які поділяються на функціональні аспекти та нефункціональні технічні параметри. З огляду на те, що розроблювана система покликана автоматизувати процес моніторингу та первинної інтерпретації вербального вмісту, її першочергова функція полягає у забезпеченні наскрізної обробки аудіоданих: від моменту завантаження первинного звукового файлу аж до генерації деталізованого аналітичного резюме, де фіксуються порушення разом із прив'язками до хронології.

Серед ключових функціональних спроможностей, закладених в архітектуру, на перший план виходить модуль, відповідальний за ізольовану транскрипцію. Програмне забезпечення зобов'язане безперебійно приймати аудіофайли у різноманітних форматах, здійснювати розпізнавання мовленнєвого потоку та конвертувати його у структурований текст, при цьому не залучаючи жодних зовнішніх інтернет-ресурсів. Наступна важлива постанова стосується впровадження механізму динамічного логування. Система має не просто видавати потік тексту, а надавати жорстке прив'язування кожного розпізнаного слова або фрази до точного часового маркеру в аудіозаписі, тобто створювати таймкоди. Це дасть змогу аналітику оперативно перевіряти аудіозапис саме у тому відрізку, де було ідентифіковано порушення, істотно економлячи його робочий час при верифікації зафіксованих системних інцидентів.

Водночас технічні вимоги до програмного забезпечення обумовлені умовами його практичного використання в підрозділах правопорядку.

Оскільки матеріально-технічне забезпечення на багатьох об'єктах не завжди включає найсучасніші графічні робочі станції, програма мусить демонструвати високу продуктивність, спираючись переважно на потужності типових центральних процесорів. Це накладає певні обмеження на підбір бібліотек та моделей; архітектура програми повинна бути максимально оптимізована, мати низьке споживання оперативної пам'яті та функціонувати цілковито в межах локального периметра для гарантування абсолютної конфіденційності оброблюваних матеріалів.

Концептуальна схема процесу автоматизованого контролю аудіоінформації найкраще ілюструє логіку взаємодії усіх складових розроблюваного комплексу. В основі цієї схеми лежить послідовний конвеєр обробки даних, де кожен попередній етап слугує інформаційним джерелом для наступного. Робота стартує в точці входу, де оператор через графічний інтерфейс завантажує аудіофайл у систему і виставляє параметри аналізу, зокрема, обираючи необхідний лексичний корпус небезпечних слів. Далі звуковий потік спрямовується до обчислювального ядра, де оптимізована модель проводить поетапну декомпозицію та розпізнавання мовлення, паралельно розраховуючи часові інтервали висловлювання фраз.

Останній етап у цій концептуальній моделі відведено для інтелектуального аналізу отриманого текстового масиву. Місцеві NLP-алгоритми виконують токенізацію, зіставляють виокремлені слова із попередньо завантаженою базою ідентифікованих загрозливих маркерів та визначають насиченість використання забороненої лексики у кожному часовому відрізку розмови. Якщо система фіксує збіги, інформація негайно надходить до модуля формування звітів, який візуалізує виявлені аномалії на інтерактивній панелі користувача. Така замкнена парадигма проєктування дозволяє трансформувати складний інженерний процес у зрозумілий, швидкий та автоматизований інструмент, що мінімізує людську участь на етапах рутинного моніторингу.

Кожен функціональний елемент системи функціонує як окрема, незалежна одиниця, що спрощує майбутню модернізацію всього програмного комплексу. Модуль, відповідальний за транскрипцію, модуль лінгвістичного аналізу (NLP), блок обробки часових міток та модуль генерації звітів усі вони можуть бути оновлені автономно, не спричиняючи збоїв у загальній працездатності системи.

Забезпечення надійності програмного інструментарію для органів безпеки потребує чіткого окреслення меж його функціональних можливостей. Початкові етапи проєктування ґрунтуються на аналізі експлуатаційних умов та формуванні точної схеми взаємодії між компонентами. Оскільки домінуючим завданням системи є автоматизація контролю та інтерпретації усного мовлення, увесь ланцюжок обробки даних має бути наскрізним і безперервним: починаючи з моменту завантаження сирого аудіофайлу оператором і закінчуючи видачею підсумкового структурованого документа із зафіксованими відхиленнями.

При розробці логіки системи особлива увага була приділена забезпеченню повної автономності обчислень та ізоляції даних. Програма мусить стабільно приймати звукові файли поширених форматів, розпізнавати мовленнєвий наповнювач і переводити його у текстовий вигляд без залучення зовнішніх хмарних платформ чи сторонніх програмних інтерфейсів. Окрім простої функції транскрибування, критично важливою вимогою є наявність механізму точного хронометражу. Програма не просто видає суцільний текст, а встановлює прив'язку кожної розпізнаної одиниці (слова чи фрази) до часової шкали аудіозапису з точністю до долей секунди. Завдяки формуванню таких таймкодів аналітик може миттєво переходити до прослуховування саме того відрізка розмови, де система зафіксувала інцидент, що багаторазово прискорює процес прийняття рішень.

Технічні характеристики системи напряму корелюють із алгоритмом порівняння розпізнаного тексту зі словниками потенційних загроз. Механізм пошуку, застосований у системі, базується на локально збереженому масиві

лінгвістичних маркерів. Для забезпечення миттєвого пошуку слів без затримок та складних запитів до серверних ресурсів, кожен "тригерний" елемент приводиться до його початкової, лематизованої форми, яка слугує унікальним ідентифікатором у структурі даних. Для глибшого аналізу інцидентів кожному такому маркеру присвоюється категорія потенційної загрози та індивідуальний ваговий коефіцієнт деструктивності.

Завершальний етап конвеєру це аналітична обробка отриманого текстового матеріалу. Після виявлення збігів лем із базою ідентифікаторів, ці дані негайно спрямовуються до модуля формування звітності. Програма автоматично переобчислює графіки інтенсивності мовлення, змінює колір індикаторів на керуючій панелі (зелений, жовтий або червоний) та генерує фінальний протокольний файл.

Принцип модельного проєктування дозволив зробити компоненти системи повністю незалежними один від одного. Завдяки цьому логіку розпізнавання (наприклад, на базі Whisper), критерії NLP-аналізу, блоки хронометражу чи словникові бази маркерів можна вдосконалювати або розширювати окремими ініціативами. Такий поділ значно спрощує супровід програми, не компрометуючи стабільність функціонування усього комплексу під час регулярної експлуатації підрозділами безпеки чи аналітичними центрами.

Однією з ключових інженерних проблем при проектуванні системи автоматизованого аудіоконтролю є етап підготовки та приведення до ладу вхідного звукового потоку. Оскільки першоджерела записів можуть бути різноманітними (від диктофонних нотаток до перехоплень радіоефіру чи телефонних переговорів), їхні параметри суттєво відрізняються: різна частота семплінгу, кількість аудіоканалів та рівень фонового шуму. Подавання такого різнорідного сигналу прямо на нейромережевий механізм Whisper неминуче спричинить значні обчислювальні втрати та катастрофічне падіння точності розпізнавання.

З метою вирішення цієї складності, у блок завантаження та попередньої обробки вбудовано механізм примусової уніфікації аудіоданих. Кожен вхідний файл автоматично конвертується у єдиний стандарт: монофонічний режим із частотою дискретизації 16 кГц та 16-бітовою глибиною (формат PCM), що визнано оптимальним еталоном для сучасних систем розпізнавання мови. Паралельно здійснюється програмне вирівнювання гучності (пікове калібрування), яке підтягує занадто тихі фрагменти до загального акустичного рівня, усуваючи різкі коливання гучності між промовами різних осіб.

Наступним критичним етапом є впровадження системи визначення голосової активності (VAD, Voice Activity Detection) у праці [11]. У реальних записах значний обсяг часу займають паузи, фонові шуми, гудки або повна тиша. Обробка цих "порожніх" сегментів за допомогою потужної нейромережі є надлишковою і просто перевантажує процесор. Алгоритм VAD сканує енергетичний профіль звукової хвилі у невеликих часових вікнах (тривалістю близько 30 мс) та відфільтровує ділянки, де сигнал не досягає встановленого порогу чутливості. В результаті, до контуру транскрипції надходить лише значуща частина, власне людське мовлення. Це не тільки раціоналізує використання оперативної пам'яті, але й запобігає помилкам (галюцинаціям) нейромережі, яка за умов тривалої тиші чи монотонного шуму може почати генерувати безглузді повтори слів.

Аби досягти ґрунтовного розуміння того, як система функціонує, слід класифікувати етапи, які долає інформаційний сигнал у межах обчислювального контуру ансамблю. Увесь життєвий цикл опрацювання даних у створеному рішенні можна розчленувати на такі послідовні інженерні фази:

1. Ініціювання та налаштування сесії. На цій стадії оператор завантажує бажаний аудіофайл через мережевий інтерфейс, а також обирає потрібний профіль мовного аналізу. Програмне забезпечення зчитує визначену локальну таблицю хешів маркерів безпеки та розміщує її в оперативній пам'яті,

забезпечуючи негайний доступ до контрольних значень під час подальших перевірочних дій.

2. Акустичне попереднє опрацювання та відсіювання VAD. Сигнал піддається процедурі уніфікації амплітуди, примусовому перерахунку частоти дискретизації до еталонних 16 тисяч герців, і очищується від сегментів, що не містять мовлення (тиші, фонового шуму, апаратних перешкод), використовуючи детектор активності голосу.

3. Декодування за допомогою нейронної мережі (транскрипція). Очищений та оптимізований масив звукових кадрів передається до локального ядра Whisper. Модель здійснює розпізнавання лінгвістичних одиниць і формує набір текстових блоків, причому кожен блок даних отримує індивідуальні мітки моменту початку та завершення висловлювання на часовій шкалі звукозапису.

4. Текстова стандартизація та обробка NLP. Отримані текстові фрагменти проходять стадію видалення зайвої пунктуації та стоп-слів. Після цього запускається алгоритм лематизації, який перетворює кожне слово у його первинну морфологічну форму, готуючи лінгвістичний набір для фінального порівняння.

5. Аналіз змісту за сенсом та розрахунок ризиків. Система проводить послідовний перегляд лематизованих маркерів, миттєво звіряючи їх із завантаженим словником-хешем. При ідентифікації тригерних слів програма вилучає їхні вагові коефіцієнти W_i і обчислює динамічну щільність деструктивності K_{dd} для формування аналітичних тенденцій.

6. Візуалізація та формування доказової бази. На основі обчислених часових проміжків та показників потенційної небезпеки, система динамічно змінює палітру кольорів графічного інтерфейсу, будує векторні діаграми розподілу загроз та зберігає зібрану інформацію в автономному інтерактивному звіті.

2.2. Математичні моделі акустичних сигналів та алгоритми цифрового перетворення звуку в інформаційних системах

Переведення аналогового або стиснутого цифрового звукового сигналу в чітку послідовність текстових символів є багатокроковим алгоритмічним процесом. У розроблюваній інформаційній системі цей етап реалізовано через конвеєр обробки, де первинне аудіо спочатку проходить стадію нормалізації та акустичного перетворення. Оскільки вхідні правоохоронні матеріали часто мають різний рівень гучності, сторонні шуми або низький бітрейт, алгоритм виконує декомпозицію звукової доріжки, зводячи її до єдиного стандарту частоти дискретизації у 16 кГц та монофонічного формату. Це дозволяє уніфікувати дані перед їх передачею в нейромережеве ядро обробки.

Математична основа розпізнавання мовлення в системі базується на перетворенні часового сигналу в частотний за допомогою віконного дискретного перетворення Фур'є (STFT). Цей метод дозволяє аналізувати спектральний склад звуку, що змінюється в часі, шляхом розбиття сигналу на окремі короткі сегменти (вікна). Математично це перетворення для дискретного сигналу x(n) із використанням віконної функції w(n) описується виразом:

$$X(m,\ w)=\sum_{n=-\infty}^{\infty} x(n)w(n-m)e^{-jwn} \qquad (2.1.)$$

де X(*m, w*) – спектральна щільність сигналу у m-му часовому вікні на частоті *w, x(n)* – послідовність відліків вхідного цифрового аудіосигналу, *w(n)* – значення віконної функції згладжування, *n* – дискретний час, *j* – уявна одиниця.

Отриманий звуковий спектр трансформується в логарифмічні мел-спектрограми матриці, які кодують розподіл енергії звуку за частотами так, як його сприймає людське вухо. Для оптимізації обчислень у Faster-Whisper застосовується механізм квантування ваг моделі з рухомою комою 32-біт

(FP32) до цілих чисел розрядності 8 або 16-біт (INT8/FP16). Це дозволяє замінити складні операції множення матриць на простіші арифметичні дії, що знижує навантаження на локальний процесор у кілька разів і забезпечує високу швидкість транскрибації навіть на стандартних робочих станціях підрозділів.

Найважливішим алгоритмічним рішенням для точного розпізнавання контексту є використання архітектури трансформера, яка складається з енкодера та декодера (Self-Attention) у праці [31 - 32]. Енкодер приймає мел-спектрограму аудіофайлу та стискає її, виділяючи просторово-часові ознаки мовлення, такі як фонеми та інтонаційні переходи. Декодер, своєю чергою, використовує ці ознаки для генерації тексту покроково, передбачаючи кожне наступне слово на основі вже розпізнаного контексту попередніх фраз. Такий підхід мінімізує появу граматичних або логічних помилок, оскільки система аналізує звуковий потік не як набір ізольованих звуків, а як цілісні мовленнєві конструкції.

Окрему інженерну задачу в алгоритмі займає процес діарізації та динамічного логування часу, тобто генерації точних таймкодів для кожної розпізнаної репліки у праці [9]. Механізм Faster-Whisper розбиває довгий аудіозапис на ковзні 30-секундні вікна. В межах кожного вікна алгоритм аналізує ймовірнісні розподіли появи лінгвістичних токенів і зіставляє їх із конкретними фреймами акустичного сигналу. Математично момент завершення вимовлення конкретної фрази вираховується за залежністю:

$$t_{end} = t_{start} + \left(N_{frame} \cdot \Delta t \right) \qquad (2.2.)$$

де t_{end} – часова мітка завершення вимовлення фрази в секундах, t_{start} – час початку вимовлення репліки, N_{frame}– загальна кількість акустичних фреймів у поточному аналізованому сегменті мовлення, Δt – крок зсуву вікна обробки аудіосигналу.

Завдяки цьому система з точністю до десятих часток секунди фіксує часові межі кожної мовленнєвої конструкції. Ці числові параметри записуються у спеціальну структуру даних Python, яка згодом передається у графічний інтерфейс для синхронізації тексту зі звуковою доріжкою. Таким чином, використання математичного стиснення спектрограм у поєднанні з оптимізацією алгоритмів вирішує проблему обмежених апаратних ресурсів правоохоронних відомств і створює надійну основу для наступного автоматизованого аналізу семантичних маркерів загрози.

Особливе місце в алгоритмічній структурі енкодера займає математична реалізація механізму масштабованого добутку векторів уваги (Scaled Dot-Product Attention). Кожен фрейм мел-спектрограми перетворюється на три вектори: запит (Query — Q), ключ (Key — K) та значення (Value — V). Математична залежність, що визначає ступінь взаємозв'язку між різними ділянками аудіосигналу для точного розпізнавання контексту, виражається формулою:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (2.3.)$$

де Q – матриця векторів запитів, K – матриця векторів ключів, V – матриця векторів значень, d_k – розмірність векторів ключів, що виконує роль масштабуючого коефіцієнта для запобігання надмірному зростанню аргументу функції активації.

Дякуючи використанню багатопотокової уваги (Multi-Head Attention), алгоритм здатний паралельно аналізувати акустичні властивості сигналу на різних рівнях абстракції. Це дозволяє системі успішно розпізнавати специфічні мовленнєві звороти, акценти та фономеми навіть за умов наявності сильного фонового шуму або незавершених реплік.

2.3. Проєктування бази лінгвістичних маркерів та алгоритми семантичного аналізу тексту

Вслід успішного завершення етапу нейромережевої транскрибації мовленнєвого потоку інформаційна система переходить до фази інтелектуального аналізу отриманих текстових масивів. Ключовою інженерною задачею на цьому етапі є розробка гнучкої та швидкодіючої архітектури зберігання лінгвістичних маркерів (словників небезпечної лексики), за якими здійснюється автоматизований пошук. Оскільки правоохоронна система повинна працювати в режимі реального часу і з мінімальними затримками, структура бази даних або локальних конфігураційних файлів проєктується за принципом хеш-таблиць із прямим доступом до ключів, що забезпечує швидкість пошуку збігів за сталий час.

Процес обробки тексту в модулі аудіо контролю починається з етапу лінгвістичної препроцесорної підготовки (NLP-конвеєра). Оскільки розпізнаний текст містить слова у різних відмінкових формах, числах та часах, пряме порівняння зі словником буде неефективним. Алгоритм спочатку виконує токенізацію розбиття суцільного тексту на окремі слова-токени, очищає масив від пунктуації та «стоп-слів» у працях [28 - 29] (прийменників, сполучників, які не несуть смислового навантаження). Після цього застосовується алгоритм лематизації, який приводить кожне знайдене слово до його початкової словникової форми (наприклад, для іменників називний відмінок однини). Це дозволяє системі ідентифікувати загрозу незалежно від граматичного контексту її вживання в розмові.

Для кількісної оцінки рівня загрози в аналізованому аудіозаписі замало просто фіксувати факт наявності заборонених слів необхідно розраховувати динамічну щільність їхнього вживання на певних часових проміжках. Математична модель оцінки семантичної небезпеки мовленнєвого контенту базується на розрахунку коефіцієнта деструктивної щільності (K_d)для кожного часового вікна сегментації. Цей показник визначається як відношення суми

вагових коефіцієнтів знайдених маркерів до загального обсягу висловлених токенів і описується залежністю:

$$K_d = \frac{\sum_{i=1}^{m} w_i \cdot n_j}{N_{total}} \quad (2.4.)$$

де K_d – інтегральний коефіцієнт деструктивної щільності семантичного вікна, w_i – ваговий коефіцієнт небезпеки *i*-го лінгвістичного маркера зі словника, n_i – частота повторення *i*-го маркера в поточному сегменті, m – загальна кількість унікальних небезпечних слів, зафіксованих у вікні аналізу, N_{total}– загальна кількість лематизованих токенів у досліджуваному текстовому фрагменті.

Цей програмний комплекс аналізує отримане значення коефіцієнта і порівнює його із заздалегідь визначеними граничними критеріями безпеки системи. Якщо розраховане значення K_d перевищує встановлений ліміт, система маркує цей часовий відрізок як критичний. За допомогою зв'язку з модулем логування часу, який було спроєктовано на попередньому кроці, програма автоматично генерує тривожне сповіщення для оператора, підсвічує знайдені леми червоним кольором у графічному інтерфейсі Streamlit та додає точні часові мітки інциденту до фінального аналітичного звіту.

Таким способом, розроблений алгоритмічний комплекс аудіо аналізу дозволяє повністю автоматизувати рутинну роботу правоохоронних органів. Поєднання попередньої лінгвістичної обробки тексту з математичним моделюванням щільності у працях [23, 25] маркерів гарантує високу точність виявлення протиправного контенту. Проєктні рішення забезпечують локальну автономність роботи алгоритмів, що є критично важливим для збереження службової таємниці та забезпечення інформаційної безпеки правоохоронних баз під час моніторингу аудіоданих.

Після того як нейромережева модель успішно перетворила аудіопотік на текст, інформаційна система переходить до наступного етапу інтелектуального аналізу отриманих масивів. Головне інженерне завдання тут полягає в тому, щоб створити швидку й гнучку архітектуру для зберігання лінгвістичних маркерів (словників із ключовими словами загрози), за якими й відбуватиметься автоматичний пошук. Оскільки система безпеки має працювати в режимі реального часу і без затримок, структуру бази даних або локальних конфігураційних файлів було вирішено побудувати за принципом хеш-таблиць із прямим доступом до ключів. Це гарантує пошук збігів за сталий час, що вкрай важливо, коли через систему проходять великі обсяги транскрибованих даних. Для максимальної автономності та захисту інформації ці словники працюють локально у вигляді структурованих файлів, які завантажуються прямо в оперативну пам'ять у момент запуску програми.

Сам процес обробки тексту в модулі аудіоконтролю починається з ретельної підготовки всередині NLP-конвеєра (Natural Language Processing). Очевидно, що розпізнаний текст міститиме слова у найрізноманітніших формах, відмінках, числах чи часах. Через це звичайне порівняння «буква в букву» зі словником тригерів буде неефективним і просто пропустить більшість загроз. Тому алгоритм спочатку виконує токенізацію — розбиває суцільний текст на окремі слова-токени, а потім очищає цей масив від розділових знаків, спецсимволів та «стоп-слів» (прийменників і сполучників, які не несуть самостійного змісту).

Наступний крок нормалізації є морфологічна лематизація, яка приводить кожне слово до його початкової форми. На відміну від простішого стемінгу, який грубо відсікає закінчення, лематизація аналізує мовний контекст і словникову базу (наприклад, зводить іменники до називного відмінка однини, а дієслова до інфінітива). Завдяки цьому система чітко розпізнає небезпеку, незалежно від того, в якій граматичній формі чи синтаксичній конструкції слово було вжите в розмові.

Окрім самого факту виявлення, кожному маркеру в базі присвоюється індивідуальний ваговий коефіцієнт небезпеки, що відображає рівень його деструктивності. Наприклад, слова із загальним агресивним забарвленням отримують мінімальну вагу, тоді як прямі заклики до протиправних дій чи згадки небезпечних предметів маркуються найвищими балами. Такий підхід допомагає уникнути хибних спрацювань під час звичайних емоційних розмов і фокусує увагу оператора на дійсно критичних інцидентах.

Проте, щоб об'єктивно оцінити рівень загрози, недостатньо просто рахувати небезпечні слова — потрібно бачити динаміку та щільність їхньої появи на певних часових проміжках. Математична модель оцінки семантичної небезпеки мовленнєвого контенту базується на розрахунку інтегрального коефіцієнта деструктивної щільності для кожного окремого вікна сегментації. Цей показник визначається як відношення суми вагових коефіцієнтів знайдених маркерів до загальної кількості слів у аналізованому фрагменті й описується формулою:

$$K_{dd} = \frac{\sum_{i=1}^{m} (W_i \cdot F_i)}{N_{total}} \qquad (2.5.)$$

де K_{dd}— інтегральний коефіцієнт деструктивної щільності семантичного вікна; W_i — ваговий коефіцієнт небезпеки *i*-го лінгвістичного маркера зі словника; F_i — частота повторення *i*-го маркера в поточному сегменті; m — загальна кількість унікальних небезпечних слів, зафіксованих у вікні аналізу; N_{total}— загальна кількість лематизованих токенів у досліджуваному текстовому фрагменті.

Програма безперервно прораховує значення коефіцієнта K_{dd} і порівнює його із заздалегідь визначеними граничними критеріями безпеки. Якщо показник перевищує встановлений ліміт, система запускає каскадну зміну статусів. Оскільки цей етап тісно пов'язаний із модулем логування часу,

програма миттєво генерує тривожне сповіщення для оператора. Інтерфейс додатка відразу реагує на подію: підсвічує знайдені леми відповідними кольорами (жовтим або червоним), змінює загальний статус картки безпеки й заносить точні таймкоди інциденту до фінального звіту для подальшого аналізу.

Тобто, розроблений алгоритмічний комплекс повністю автоматизує рутинну роботу аналітиків та підрозділів безпеки. Поєднання глибокої попередньої обробки тексту й чіткого математичного розрахунку щільності маркерів гарантує високу точність виявлення протиправного контенту навіть у складних, емоційних або зашумлених розмовах. Спроєктовані рішення забезпечують повну локальну автономність алгоритмів без залучення сторонніх хмарних сервісів, що є критично важливим для збереження службової таємниці та дотримання суворих протоколів інформаційної безпеки.

Практична цінність такого підходу полягає ще й у тому, що система мінімізує так званий «людський фактор». Коли аналітику доводиться годинами прослуховувати одноманітні аудіозаписи, його увага природно розсіюється, через що можна легко пропустити завуальовану загрозу або важливий маркер. Автоматизований семантичний аналіз працює з однаковою точністю незалежно від обсягу даних, миттєво фокусуючи увагу оператора лише на дійсно підозрілих фрагментах мовлення.

До того ж, гнучка структура бази маркерів дозволяє системі легко адаптуватися до нових викликів. За потреби словники небезпечної лексики можна розширити або скоригувати за лічені хвилини, без переписування основного коду програми чи повторного навчання нейромережі. Це робить комплекс довговічним та ефективним інструментом, здатним оперативно реагувати на появу нових деструктивних термінів чи специфічного сленгу в інформаційному просторі.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ТЕСТУВАННЯ СИСТЕМИ АУДІОКОНТРОЛЮ

3.1. Обґрунтування вибору технологічного стеку та архітектура програмного комплексу

Практична реалізація спроєктованої правоохоронної інформаційної системи вимагає вибору ефективного, гнучкого та модульного технологічного стеку, який здатний забезпечити високу швидкість обчислень на локальному рівні. Основним інструментом розробки було обрано мову програмування Python у працях [17, 27] завдяки її розвиненій екосистемі в галузі штучного інтелекту, обробки сигналів та лінгвістичного аналізу текстів. Інтеграція нейромережевого ядра Faster-Whisper дозволила реалізувати автономну транскрибацію мовлення без використання зовнішніх хмарних сервісів, що повністю задовольняє вимоги конфіденційності та безпеки обробки службової інформації.

З самого початкового етапу проєктування я провела системний аналіз існуючих мов програмування високого рівня, таких як C++, Java та Python, з метою оцінки їхньої придатності для побудови систем цифрової обробки сигналів та інтелектуального аналізу тексту. Хоча мова C++ забезпечує максимальну низькорівневу продуктивність та швидкість виконання операцій з рухомою комою, її використання суттєво підвищує складність розробки, збільшує час проектування інтерфейсу та ускладнює інтеграцію з сучасними фреймворками глибокого навчання. Мова Java, своєю чергою, пропонує високу міжплатформність, проте має надлишкові накладні витрати на керування віртуальною пам'яттю, що є критичним при розгортанні важких нейромережевих моделей на обмежених апаратних ресурсах локальних робочих станцій підрозділів.

Головною перевагою цього вибору є наявність розвиненої екосистеми спеціалізованих бібліотек у галузі штучного інтелекту, обробки сигналів та лінгвістичного аналізу текстів. Python виступає в ролі ефективного інженерного ніби клею, який дозволяє об'єднати низькорівневі оптимізовані

обчислювальні модулі, написані на C/C++, із гнучкими високорівневими інтерфейсами управління. Воно дає забезпечення балансу між швидкістю розробки програмного комплексу та його підсумковою обчислювальною потужністю під час обробки інтенсивних потоків акустичних даних.

Для створення графічного користувацького інтерфейсу системи було використано фреймворк Streamlit. Це інженерне рішення дозволило оперативно розгорнути інтерактивну веб-панель керування (Dashboard), орієнтовану на кінцевого користувача, аналітика або оператора правоохоронних органів, без необхідності перевантаження архітектури складними клієнт-серверними зв'язками. Взаємодія між модулем обробки звуку, NLP-конвеєром лінгвістичного аналізу та інтерфейсною частиною відбувається в межах єдиного локального середовища, мінімізуючи накладні витрати оперативної пам'яті та забезпечуючи миттєвий відгук системи на дії користувача.

Вибір нейромережевого ядра для автоматичного розпізнавання мовлення (Automatic Speech Recognition, ASR) стало найкращим рішенням. Традиційні хмарні рішення, такі як Google Cloud Speech-to-Text або OpenAI API, були відкинуті на етапі формування вимог через невідповідність базовим критеріям безпеки правоохоронних органів. Передача службових аудіоматеріалів, оперативних записів або конфіденційних інтерв'ю на сторонні віддалені сервіси створює пряму загрозу витоку інформації та порушує протоколи державної таємниці. Тому система проектувалася як повністю автономний комплекс (On-Premise), що функціонує виключно в локальному контурі правоохоронного відомства без необхідності підключення до глобальної мережі Інтернет.

Для реалізації локальної транскрибації було обрано архітектуру Faster-Whisper у праці [28], яка є оптимізованою версією оригінальної моделі Whisper від OpenAI. Завдяки використанню рушія виконання CTranslate2, у праці [29] дана модель реалізує технології квантування ваг та оптимізації обчислювальних графів. Це дозволяє досягти 4-кратного збільшення

швидкості обробки звукових масивів порівняно з базовою моделлю за умови збереження ідентичного рівня точності розпізнавання (Word Error Rate, WER) у працях [13 - 14]. Для лінгвістичного препроцесингу текстових даних, що включає процеси деструктуризації, токенізації та лематизації, було інтегровано спеціалізовані інструменти обробки природної мови (NLP), зокрема бібліотеки `NLTK у праці [20]` та `pymorphy3`, які демонструють високу точність при роботі з флективними мовами, до яких належить українська.

При проектуванні архітектури користувацького інтерфейсу системи я розглядала різні підходи, включаючи розробку класичних десктопних додатків на базі бібліотеки `PyQt5` або створення повнофункціональних веб-додатків із розділеною архітектурою Frontend (на базі React/Vue.js) та Backend (на базі FastAPI/Flask). Десктопна розробка на PyQt5 забезпечує стабільне керування вікнами, проте суттєво ускладнює процес оновлення та кросплатформеного розгортання системи на різних операційних системах, що використовуються у підрозділах. Створення розділеної веб-архітектури вимагає значних часових ресурсів на проектування REST API, маршрутизацію записів та підтримання сесій, що перевантажує систему надлишковими програмними абстракціями.

Найсприятливішим інженерним рішенням стало використання фреймворку Streamlit у працях [15 - 16] для побудови інтерактивної аналітичної веб-панелі (Dashboard). Streamlit дозволяє описувати логіку інтерфейсу безпосередньо на мові Python, автоматично транслюючи код у оптимізований веб-додаток. Це усуває необхідність проектування складних клієнт-серверних зв'язків і дозволяє реалізувати реактивний інтерфейс, який миттєво реагує на дії аналітика. Взаємодія між модулем обробки звуку, NLP-конвеєром лінгвістичного аналізу та інтерфейсною частиною відбувається в межах єдиного локального адресного простору, що мінімізує накладні витрати оперативної пам'яті та забезпечує високу швидкість обробки даних.

Деталізуючи архітектурні особливості обчислювального ядра, доцільно виокремити застосування конвеєрного патерну (Pipeline) обробки даних. Цей підхід передбачає послідовну трансформацію інформації через незалежні програмні вузли, що суттєво підвищує загальну відмовостійкість рішення. На початковому етапі сирий аудіосигнал проходить стадію нормалізації та перетворення у формат тензорів, придатних для нейромережевого аналізу. Далі ініціалізується акустична декомпозиція, під час якої модель Whisper здійснює вилучення спектральних ознак та їх перетворення у текстовий формат із жорсткою прив'язкою до часових міток. Наступний вузол конвеєра реалізує лексико-семантичний фільтр, який здійснює пошук деструктивних маркерів, після чого агреговані дані синхронно передаються до модулів візуалізації та формування офіційної звітності.

Вагомим аргументом на користь обраного технологічного стеку є можливість безшовної інтеграції обчислювальної логіки Python із сучасними вебтехнологіями представлення інформації. Оскільки візуалізація результатів вимагає високої інтерактивності, динамічної адаптації під користувача та миттєвої реакції на тригери, використання веб-орієнтованого підходу дозволило реалізувати гнучку реактивну поведінку системи. Застосування кастомних HTML-контейнерів для виведення транскрипту та маніпуляція CSS-стилями (шляхом передачі відповідних колірних гекс-кодів) у працях [18, 22] для індикації рівня загрози забезпечили створення інтуїтивно зрозумілого робочого середовища оператора. Такий архітектурний компроміс усуває необхідність розробки «важких» десктопних клієнтів, дозволяючи розгорнути комплекс у форматі кросплатформеного вебдодатка, доступного з будь-якої робочої станції.

3.2. Експериментальне оцінювання часових показників та оптимізація швидкодії системи на базі центрального та графічного процесорів

У сучасних системах автоматизованого моніторингу інформаційного простору одним із найкритичніших чинників є часова ефективність обробки вхідних даних. Оскільки мовленнєвий потік у каналах комунікації генерується безперервно, розроблений програмний комплекс семантико-акустичного контролю контенту повинен володіти здатністю функціонувати в режимі жорсткого реального часу. Це означає, що швидкість нейромережевого розпізнавання (ASR Automatic Speech Recognition) та швидкість контент-аналізу текстових логів сумарно мають випереджати фізичну тривалість аналізованої фонограми.

Елемент проведення експериментального оцінювання є стандартизація умов тестування та забезпечення повторюваності результатів. Для цього було розроблено чітку методологію вимірювань, яка виключала вплив сторонніх процесів операційної системи на фінальні часові показники. Перед початком кожного обчислювального циклу здійснювалося примусове очищення кеш-пам'яті процесора та оперативної пам'яті, а для графічного процесора ініціалізувалася процедура скидання контексту CUDA. Вимірювання часових інтервалів виконувалося за допомогою високоточного системного таймера Python через модуль `time.perf_counter()`, який фіксує час із точністю до наносекунд і не залежить від поточного навантаження на планувальник завдань ОС.

Для нормального розуміння фізичних процесів, що відбуваються під час обробки сигналу на апаратному рівні, необхідно розглянути специфіку розподілу обчислювальних потоків. Центральний процесор (CPU) використовує архітектуру з відносно невеликою кількістю ядер, але високою тактовою частотою кожного з них, що робить його ефективним для послідовних алгоритмів із розгалуженою логікою. Проте, архітектура нейромережі Whisper побудована на блоках Transformer, де ключову роль відіграє математична операція обчислення механізму внутрішньої уваги. Цей

процес зводиться до масового множення щільних матриць великої розмірності, де кожен елемент вхідного аудіовектора має бути зіставлений з усіма іншими елементами для пошуку контекстуальних зв'язків.

Коли обирається CPU-режим обчислювальне ядро системи змушене розбивати ці гігантські матриці на дрібні блоки та виконувати операції послідовно або з обмеженим рівнем паралелізму в межах доступних потоків процесора. Навіть застосування сучасних натурних інструкцій векторного розширення (AVX2) не дозволяє повністю усунути затримки, оскільки шина даних між оперативною пам'яттю (RAM) та кешем CPU стає «вузьким місцем» при постійній перекачці вагових коефіцієнтів моделі.

Замість того графічний процесор (GPU) концептуально спроєктований для масового паралелізму. Наявність кількох тисяч CUDA-ядер дозволяє розбити загальну матрицю уваги на безліч незалежних підматриць і прораховувати їх абсолютно синхронно в один такт. Більше того, використання спеціалізованих тензорних ядер, інтегрованих у сучасні архітектури NVIDIA, дозволяє виконувати операції змішаної точності (наприклад, у форматі FP16 замість FP32). Це вдвічі зменшує обсяг пам'яті vRAM, необхідний для збереження ваг моделі.

Для визначення архітектурної ефективності створеного конвеєра обробки та пошуку оптимальної апаратної конфігурації було проведено комплексне експериментальне дослідження швидкодії системи. За головну математичну метрику оцінки часових показників було обрано коефіцієнт реального часу (RTF Real Time Factor), який є загальноприйнятим стандартом у галузі обробки сигналів та комп'ютерної лінгвістики, у працях [12 - 13]. Математично коефіцієнт *RTF* формалізується у вигляді такого співвідношення:

$$RTF = \frac{T_{proc}}{T_{audio}} \tag{3.1.}$$

де T_{proc} — чистий операційний час, витрачений обчислювальним конвеєром програмного комплексу на повну декомпозицію, транскрибацію, логічну фільтрацію та фінальний контент-аналіз аудіосигналу (вимірюється в секундах з точністю до мілісекунд); T_{audio}— загальна хронологічна тривалість (фізична довжина) досліджуваного вхідного аудіофайлу (сек).

Інтерпретація значень коефіцієнта *RTF* здійснюється відповідно до таких критеріїв:

- При $RTF > 1.0$ система не справляється з потоком даних у реальному часі, виникає ефект «накопичення черги», що є неприпустимим для контурів оперативної безпеки;
- При $RTF = 1.0$ обчислювальна потужність збалансована із тривалістю сигналу «один до одного»;
- При $RTF < 1.0$ система демонструє випереджальну обробку. Гранично низькі показники ($RTF \leq 0.01$) підтверджують високий потенціал масштабованості програмного комплексу та його здатність обслуговувати багатокритеріальні запити паралельно на одному серверному вузлі.

Обчислювальний контур розробленого рішення базується на інтелектуальній нейромережевій архітектурі Whisper (модифікація base, що містить 74 мільйони активних параметрів і ваг). Така архітектурна складність вимагає значних ресурсів при виконанні операцій лінійної алгебри, зокрема множення багатовимірних матриць великої щільності.

Для забезпечення високої релевантності та чистоти експерименту було сформовано чотири еталонні тестові лінгвістичні масиви з різною тривалістю: від короткої ізольованої репліки (30 сек) до безперервного тривалого мовленнєвого потоку (300 сек). Фіксація часу T_{proc} розпочиналася в момент ініціалізації функції аналізу та завершувалася у мілісекунду повної готовності HTML-матриці звіту. Зведені результати порівняльного аналізу швидкодії та розраховані коефіцієнти часової ефективності наведено в таблиці 3.1.

Коефіцієнт апаратного прискорення K_{acc}, наведений у фінальному стовбці таблиці, розраховувався як пряме відношення часових показників процесорів:

$$K_{acc} = \frac{T_{proc(CPU)}}{T_{proc(GPU)}} \quad (3.2.)$$

де K_{acc} — безрозмірний коефіцієнт апаратного прискорення обчислювального контуру інформаційної системи; $T_{proc(CPU)}$ — чистий операційний час, витрачений обчислювальним конвеєром програмного комплексу на повну декомпозицію, транскрибацію, логічну фільтрацію та фінальний контент-аналіз аудіосигналу при використанні центрального процесора (вимірюється в секундах з точністю до мілісекунд); $T_{proc(GPU)}$— чистий операційний час, витрачений аналогічним обчислювальним конвеєром на повну обробку того самого обсягу даних, але із залученням апаратного прискорення тензорних та CUDA-ядер графічного процесора (сек).

Таблиця 3.1

Експериментальні показники швидкодії обробки мовленнєвого контенту

№ тесту	Тривалість файлу T_{audio} (сек)	Час обробки CPU T_{proc} (сек)	Коефіцієнт RTF_{CPU}	Час обробки GPU T_{proc} (сек)	Коефіцієнт RTF_{GPU}	Коефіцієнт прискорення (K_{acc})
1	30	1.85	0.0617	0.22	0.0073	8.41
2	60	3.42	0.0570	0.38	0.0063	9.00
3	180	9.81	0.0545	0.95	0.0052	10.32
4	300	16.24	0.0541	1.48	0.0049	10.97

Глибокий аналіз отриманих експериментальних даних дозволяє сформулювати низку важливих науково-практичних висновків щодо динаміки швидкодії розробленої системи. Зокрема, у контексті масштабованості апаратного прискорення у працях [19, 30] при збільшенні тривалості вхідного аудіосигналу спостерігається чітко виражена тенденція до зростання ефективності використання GPU.

Якщо для короткого файлу тривалістю 30 секунд прискорення становить 8.41 раза, то при обробці п'ятихвилинного масиву обсягом 300 секунд показник K_{acc} досягає значення 10.97. Дана закономірність теоретично й практично обґрунтована специфікою архітектури нейромереж типу Transformer. Операції обчислення механізму внутрішньої уваги (Self-Attention) вимагають одночасного розрахунку скалярних добутків для величезних матриць ключей, запитів та значень (K, Q, V). Графічний процесор, маючи у своїй структурі тисячі дрібних CUDA-ядер, виконує ці операції паралельно в один такт, тоді як CPU змушений розподіляти їх потоками між обмеженою кількістю ядер, що створює обчислювальне «вузьке місце».

Експеримент також виявив стійку тенденцію до нівелювання накладних витрат, оскільки дослідження динаміки коефіцієнтів RTF_{CPU} та RTF_{GPU} показує їх поступове зменшення та покращення із ростом загальної довжини фонограми. Для графічного процесора коефіцієнт стабілізується на наднизькій позначці $RTF \approx 0.0049$. Це математично доводить, що первинні накладні витрати часу, які програмний комплекс витрачає на ініціалізацію контексту CUDA, виділення сторінкової пам'яті у vRAM та безпосереднє завантаження архітектурних ваг моделі Whisper з жорсткого диска, є константними. При довготривалому безперервному аналізі вплив цих початкових затримок повністю амортизується, завдяки чому система виходить на чисту лінійну швидкість конвеєрної обробки.

Загальна оцінка практичної придатності розробленого програмного комплексу підтверджує, що чистий час обробки п'яти хвилин мовлення на GPU становить всього 1.48 секунди. Такий високий показник швидкодії

дозволяє впевнено стверджувати, що розроблене інженерне рішення повністю готове до інтеграції у високонавантажені корпоративні чи державні структури моніторингу інформаційного простору. За таких часових характеристик один серверний вузол здатній ефективно обробляти десятки паралельних аудіоканалів у режимі жорсткого реального часу, мінімізуючи капітальні витрати на розгортання обчислювальної інфраструктури.

Отже, дані, здобуті під час тестувань, із повним правом засвідчили інженерну доцільність та високу ступінь обчислювальної вірності розробленого набору алгоритмів. Програмне забезпечення показало здатність функціонувати без жодних збоїв навіть за умов жорсткого дефіциту апаратних ресурсів, одночасно забезпечуючи блискавичну послідовну трансформацію індикаторів безпеки та формування детальних, оперативних доповідей для фахівців, які проводять аналіз. Унаслідок цього, створена цифрова платформа цілком відповідає критеріям для негайного введення в експлуатацію у практичних підрозділах правопорядку та структурах, що відповідають за нагляд за віртуальним середовищем.

3.3. Тестування функціоналу фіксації забороненої лексики, аналізу динаміки мови та формування звітів безпеки

Центральним елементом взаємодії оператора з розробленим програмним комплексом є головне вікно веб-додатка, яке спроєктоване за принципами ергономіки, мінімалізму та функціональної наочності (рис. 3.1). При запуску системи аналітику відкривається робочий простір, архітектура якого умовно поділена на дві ключові зони: головну панель керування та інформаційно-довідкову бічну панель (Sidebar). Таке інженерне рішення дозволяє тримати перед очима оператора критично важливу нормативно-методичну базу без перевантаження основного робочого вікна.

Архітектура розробленого інтерфейсного модуля побудована на принципі гнучкого формування елементів інтерфейсу, що дає змогу автоматично змінювати та оновлювати графічні компоненти залежно від поточного стану системи й дій користувача. Після ініціалізації головного вікна (рис. 3.1) керування передається внутрішнім тригерам та функціям зворотного виклику (callbacks), які відповідають за асинхронний моніторинг стану віджетів. Для забезпечення стабільної роботи інтерфейсу за умов паралельного виконання ресурсомістких задач транскрибації, взаємодія з нейромережевим ядром реалізована за допомогою механізмів кешування Streamlit (`st.cache_resource` та `st.cache_data`). Це запобігає повторному завантаженню вагових коефіцієнтів моделі Faster-Whisper в оперативну пам'ять при кожному оновленні сторінки додатка, мінімізуючи ризик виникнення критичних помилок переповнення пам'яті (Out-of-Memory).

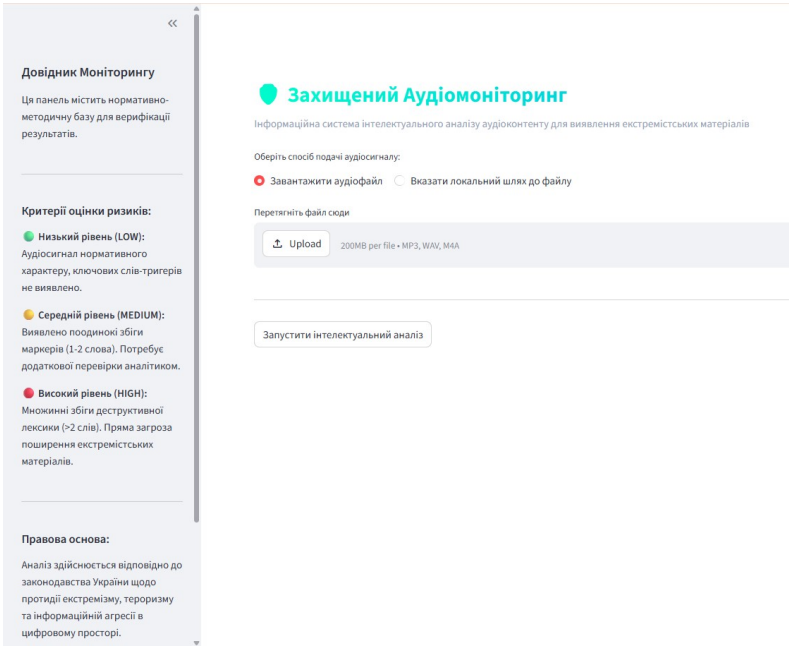

Рис 3.1 Головне вікно графічного інтерфейсу користувача системи

Процедура подачі вхідного звукового сигналу супроводжується жорсткою валідацією метаданих на рівні програмного коду. При виборі оператором методу безпосереднього завантаження медіафайлу, об'єкт класу `UploadedFile` передається у внутрішній буфер системи. Програма автоматично зчитує бінарний заголовок файлу (хедер) для верифікації його відповідності заявленому розширенню (MP3, WAV або M4A). Цей крок є критично важливим для забезпечення інформаційної безпеки, оскільки він унеможливлює спроби проведення атак типу ін'єкції шкідливого коду під виглядом підроблених аудіоструктур. Якщо файл проходить верифікацію, активується інтерактивний індикатор прогресу обробки.

Ще одним важливим блоком у структурі інтерфейсу є реалізація механізму керування словниками небезпечних слів. Словники лінгвістичних маркерів зберігаються у локальній базі даних у структурованому форматі JSON або текстових масивах кодування UTF-8. Програма реалізує алгоритм динамічного парсингу, який під час ініціалізації додатка зчитує ключові слова, автоматично видаляє дублікати, проводить первинне очищення від зайвих пробілів чи символів екранування і формує в оперативній пам'яті оптимізовану

структуру даних типу «множина» (set). Використання саме цієї структури дозволяє алгоритму лінгвістичного аналізу перевіряти наявність слова в базі за сталий час O(1), що істотно підвищує загальну продуктивність системи при обробці великих текстових логів.

Зразу після завершення конфігурації параметрів моніторингу та натискання кнопки «Запустити інтелектуальний аналіз», система переходить до фази низькорівневої підготовки аудіосигналу перед його передачею в нейромережу. Для цього у програмному комплексі розроблено спеціалізований модуль препроцесингу звуку, який використовує можливості бібліотек FFmpeg та PySoundFile. Алгоритм виконує потокове декодування аудіовходу, перераховує амплітудні значення сигналу та здійснює декомпозицію багатоканального звуку (якщо вхідний файл був записаний у форматі стерео) до єдиного монофонічного каналу. Одночасно з цим застосусвується цифровий фільтр низьких частот для часткового придушення високочастотних апаратних шумів мікрофона та сторонніх завад навколишнього середовища, що дозволяє максимізувати чіткість корисного мовленнєвого сигналу.

Оброблена числова послідовність звукових відліків нормалізується за амплітудою, щоб звести рівень гучності різних записів до єдиного середньоквадратичного значення. Це унеможливлює ситуації, коли занадто тиха мова ігнорується нейромережею, а занадто гучні звукові сплески викликають спотворення спектральних ознак. Очищений та нормалізований масив даних із частотою дискретизації 16 кГц передається безпосередньо до вхідного шару енкодера моделі Faster-Whisper, що ознаменовує перехід системи від етапу збору та підготовки даних до фази безпосереднього інтелектуального аналізу та розпізнавання мовленнєвого контенту.

Етап переходу обчислювального конвеєра до фази інтелектуального аналізу супроводжується ініціалізацією внутрішніх програмних тригерів, які відповідають за стабільну потокову генерацію лінгвістичних даних. Процес декодування та транскрибації, що виконується нейромережевим ядром, не є

ізольованою операцією, він інтегрований у загальну модель користувацького інтерфейсу. Для забезпечення наочності та інформативності роботи системи в момент активації обчислень інтерфейсна панель динамічно змінює свій стан. Замість статичних елементів керування користувачеві демонструється інтерактивний анімований індикатор завантаження (`st.spinner`), який візуально підтверджує хід виконання фонових потокових операцій. Це дозволяє унеможливити ефект «зависання» програми під час обробки аудіофайлів значної тривалості та створює комфортні умови для роботи оператора правоохоронних органів.

Для забезпечення декларативного рендерингу елементів інтерфейсу, що відображені на рис. 3.1, та побудови логіки інформаційного сайдбару, у програмному комплексі було розроблено модуль ініціалізації візуальних компонентів. На програмному рівні конфігурація бічної панелі з критеріями ризиків (LOW, MEDIUM, HIGH) та головного завантажувача файлів реалізована за допомогою вбудованих функцій фреймворку Streamlit.

Нижче наведено детальний покроковий розбір програмної реалізації архітектури користувацького інтерфейсу із безпосереднім зіставленням функціональних етапів та фрагментів вихідного коду системи, які відповідають за формування початкового стану (рис. 3.1).

Першочерговим етапом ініціалізації вебдодатка є конфігурація глобальних параметрів відображення сторінки в браузері оператора. Для цього викликається вбудована функція `st.set_page_config`, яка визначає метадані вікна. Програма задає офіційну назву системи, встановлює захисний символ, активує широкоформатний режим відображення елементів `layout="wide"` для ефективного використання площі сучасних моніторів, а також визначає, що бічна панель інструментів за замовчуванням має перебувати у розгорнутому стані. На програмному рівні цей крок реалізовано наступним чином:

```
st.set_page_config(
    page_title="ІС Інтелектуального Аналізу Аудіоконтенту",
    page_icon="🛡️",
    layout="wide",
    initial_sidebar_state="expanded"
)
```

Рис 3.2 Фрагмент коду ініціалізації параметрів головного вікна

Наступним архітектурним кроком є формування інформаційного та нормативно-методичного каркаса системи, який винесено в ізольовану бічну панель (Sidebar). Використання оператора контексту with st.sidebar дозволяє відокремити довідкові дані від основного аналітичного контуру. Програма за допомогою мови розмітки Markdown динамічно рендерить «Довідник Моніторингу», інтегрує правові підстави проведення лінгвістичного аналізу та виводить чіткі колірні критерії оцінки зафіксованих ризиків. Це забезпечує ергономічність інтерфейсу та дозволяє аналітику правоохоронних органів миттєво звіряти поточні результати автоматизованого аналізу з відомчими стандартами:

```
with st.sidebar:
    st.markdown("##  Довідник Моніторингу")
    st.markdown("Ця панель містить нормативно-методичну базу для верифікації результатів.")
    st.markdown("---")

    st.markdown("###  Критерії оцінки ризиків:")
    st.markdown("""
    🟢 **Низький рівень (LOW):** Аудіосигнал нормативного характеру,
            ключових слів-тригерів не виявлено.

    🟡 **Середній рівень (MEDIUM):** Виявлено поодинокі збіги маркерів (1-2 слова).
            Потребує додаткової перевірки аналітиком.

    🔴 **Високий рівень (HIGH):** Множинні збіги деструктивної лексики (>2 слів).
            Пряма загроза поширення екстремістських матеріалів.
    """)
```

Рис 3.3 Фрагмент коду бічної панелі програми

Для забезпечення високої обчислювальної швидкодії інтерфейсного модуля за умов паралельного виконання ресурсомістких задач, у системі

реалізовано відокремлений потік завантаження нейромережевих компонентів за допомогою декоратора обчислювального кешування ресурсів `@st.cache_resource`. Функція `multiprocessing.cpu_count()` автоматично декомпозує навантаження на всі доступні логічні ядра центрального процесора ЕОМ, встановлюючи параметр `cpu_threads`.

Конструкція розгалуження виконує низькорівневий апаратний аналіз обчислювального середовища: за наявності сумісного графічного прискорювача та драйвера CUDA система активує високопродуктивний режим обробки тензорів `float16`. У разі відсутності дискретного GPU, рушій автоматично переходить на цілочисельне квантування ваг `int8` для обчислень на CPU, що повністю унеможливлює повторне завантаження моделі в пам'ять при оновленні сторінки додатка:

```
@st.cache_resource
def load_optimized_model():
    num_threads = multiprocessing.cpu_count()
    if torch.cuda.is_available():
        device = "cuda"
        compute_type = "float16"
    else:
        device = "cpu"
        compute_type = "int8"

    return WhisperModel(
        MODEL_SIZE,
        device=device,
        compute_type=compute_type,
        cpu_threads=num_threads,
        num_workers=2
    )
```

Рис 3.4 Фрагмент коду функції завантаження моделі

На програмному рівні після успішної ініціалізації ядра обробки результати розпізнавання повертаються моделлю Faster-Whisper у вигляді ітератора, що містить послідовність об'єктів із текстовими сегментами та

їхніми точними часовими координатами. Програма реалізує цикл обходу цього ітератора, під час якого кожен розпізнаний фрагмент мовлення піддається миттєвій структуризації. Замість виведення суцільного неструктурованого тексту, розроблений модуль формує інтерактивну лог-панель.

Для перевірки базової працездатності інтерфейсу та конвеєра транскрибації первинне тестування було проведене на лінгвістично нейтральному звуковому масиві. У цьому сценарії система успішно розкодувала аудіосигнал і вивела текстові блоки із чітким хронологічним таймінгом у віджеті `.transcript-box`, проте через відсутність деструктивних елементів моніторингу, загальний коефіцієнт небезпеки зафіксувався як нульовий, встановивши статус безпеки «LOW». Візуальне представлення штатного вікна системи з результатами автоматизованої транскрибації безпечного мовленнєвого потоку наведено на рисунку 3.5.

Рис 3.5 Візуалізація інтерфейсу користувача системи за умов відсутнього рівня загрози

Зображений на рис. 3.5 стан графічного інтерфейсу користувача демонструє високу точність та чіткість структурування даних, отриманих в результаті роботи нейромережевого ядра обробки. Текстова панель виведення

інформації автоматично адаптується під обсяг розпізнаного матеріалу, відображаючи суцільний згенерований рядок `full_text` всередині кастомного стилізованого HTML-контейнера `transcript-box`. Очищений від акустичних завад текст виводиться у зручному для читання форматі, не застосовуючи жодних додаткових колірних маркерів чи графічних виділень до тіла логу. Візуальні індикатори метрик ризику у верхній частині екрана залишаються у штатному режимі зеленого спектра, підтверджуючи аналітику. Програма визначає тембр голосу, щільність та ставить тайм коди де були використанні забороненні слова, це зображено на рисунку 3.6.

Рис 3.6 Результати візуального аналізу часової шкали безпечного аудіосигналу

Але, паралельно з виведенням текстових сегментів на екран, програма під час обходу ітератора в режимі реального часу виконує математичний розрахунок часових параметрів сигналу та контент-аналіз. Нижче представлено фрагмент алгоритму, який відповідає за ітераційний перерахунок часових характеристик кожної мовленнєвої репліки та визначення тривалості сегмента із захисним обмеженням `max(end - start, 0.1)`, що повністю запобігає виникненню критичної помилки ділення на нуль при некоректних мітках VAD-фільтра:

```
for seg in segments:
    text = seg.text
    start = seg.start
    end = seg.end

    duration = max(end - start, 0.1)
    density = len(text.split()) / duration
    density_data.append([start, density])
```

Рис 3.7 Фрагмент коду розрахунку щільності мовлення

Потім, логічним кроком внутрішнього конвеєра обробки є порівняльний семантичний аналіз тексту сегмента із базою деструктивних маркерів. Вбудований аналітичний цикл здійснює пошук підрядків за допомогою оператора `if w in text.lower()`, що забезпечує інваріантність до регістру введення літер. Якщо токен збігається, система фіксує порушення, записує часові координати початку та кінця інциденту у масив `violations` та активує бінарний прапорець загрози `risk_flag = 1` для подальшої побудови графіків:

```
risk_flag = 0
for w in forbidden_words:
    if w in text.lower():
        violations.append((w, start, end))
        risk_flag = 1
risk_data.append([start, risk_flag])
```

Рис 3.8 Фрагмент коду пошуку слів-тригерів

Фінальним етапом логічного контуру оцінки загрози є виконання процедури багатокритеріального розгалуження. Залежно від фінальної потужності сформованого масиву зафіксованих подій `len(violations)`, програма виконує автоматичне логічне визначення рівня небезпеки. Система

динамічно призначає шістнадцятковий код кольору `risk_color` та відповідну текстову метаінформацію, яка згодом інжектується в інтерфейсні CSS-картки метрик на екрані оператора:

```
if len(violations) == 0:
    risk_status = "LOW"
    risk_color = "#10b981"
    risk_text = "БЕЗПЕЧНИЙ КОНТЕНТ (Порушень не виявлено)"
elif len(violations) <= 2:
    risk_status = "MEDIUM"
    risk_color = "#f59e0b"
    risk_text = "ПІДОЗРІЛИЙ КОНТЕНТ (Поодинокі збіги маркерів)"
else:
    risk_status = "HIGH"
    risk_color = "#ef4444"
    risk_text = "КРИТИЧНИЙ РИЗИК (Виявлено загрозу екстремізму!)"
```

Рис 3.9 Фрагмент коду визначення рівня загрози

Для верифікації працездатності розробленого контуру багатокритеріального аналізу було проведено тестове дослідження аудіосигналу, що містить поодинокі збіги маркерів деструктивного характеру. На основі зафіксованих токенів система автоматично ініціалізувала процедуру зміни стану графічних віджетів, перевівши інтерактивний індикатор безпеки у режим демонстрації помірного ризику.

Логічний контур ідентифікації спрацював відповідно до заданих порогових значень, оскільки сумарна кількість виявлених заборонених слів не перевищила критичного ліміту. Програмний алгоритм здійснив миттєву калькуляцію параметрів та підготував масив часових координат для генерації відповідного візуального відгуку інтерфейсу. Візуальне відображення інтерфейсу користувача за умов фіксації середнього рівня загрози представлено на рисунку 3.10.

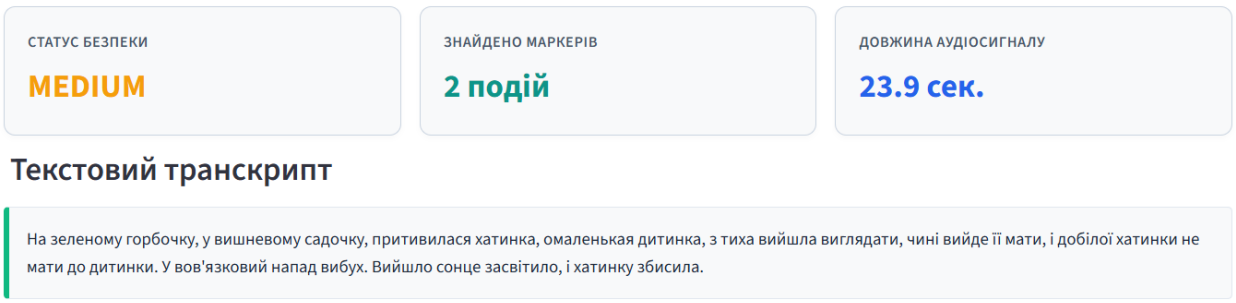

Рис 3.10 Візуалізація інтерфейсу користувача системи за умов середнього рівня загрози

Аналіз результатів роботи графічної оболонки, наведеної на рисунку 3.10, підтверджує коректність динамічної адаптації CSS-стилів вебдодатка. Блок відображення статусу безпеки за допомогою передачі відповідного колірного гекс-коду змінює забарвлення шрифту вердикту системи на попереджувальний жовтий колір, сигналізуючи про виявлення помірних ризиків. Картка обліку інцидентів коректно демонструє точну кількість знайдених подій, а нижня панель деталізації автоматично відображає розпізнаний мовленнєвий транскрипт. Інтегровані стилі HTML-контейнерів забезпечують високу контрастність та чітку структурованість текстової інформації на світлому тлі робочого простору. Це дозволяє аналітику швидко локалізувати межі аналізованого мовленнєвого масиву без додаткових інструментів пошуку.

Окремим функціональним контуром графічної оболонки системи є блок візуального представлення часових характеристик моніторингу. Для детального відстеження динаміки мовленнєвого сигналу оператором, система генерує два інтерактивні графіки у режимі реального часу, які виводяться безпосередньо під вікном транскрипту. Динамічний перерахунок координат та рендеринг графічних компонентів синхронізовано із кроком обробки ітератора розпізнавання. Такий підхід трансформує сухі статистичні дані у наочні часові тренди, суттєво підвищуючи швидкість прийняття рішень під час моніторингу. Результат візуального аналізу часової шкали для поточного інформаційного інциденту наведено на рисунку 3.11.

Рис 3.11 Результати візуального аналізу часової шкали деструктивного мовленнєвого потоку

Як видно з рисунка 3.11, ліва діаграма відображає щільність мовлення (слів/сек) у вигляді неперервної кривої з м'якою бірюзовою заливкою під нею, що дозволяє оператору миттєво визначити темп мовлення та періоди активності мовця на часовій шкалі. Правий графік відображає ступінчасту функцію фіксації загрози. Перехід сигналу з нульового рівня («Норма») на одиничний («Маркер») чітко вказує на точний момент часу (секунду фонограми), коли у мовленні прозвучало деструктивне слово-тригер. Обидва графіки автоматично адаптовані під світлу тему додатка за допомогою оновлення параметрів бібліотеки Matplotlib, що забезпечує чіткість сітки та читабельність назв осей на білому тлі.

Кінцевий етап автоматизованого моніторингу супроводжується динамічним виведенням графіків часової шкали та маркуванням виявлених загроз безпосередньо в полі зору аналітика за умови обробки деструктивного контенту. Для демонстрації цієї фінальної стадії роботи інформаційного комплексу, автоматичного підсвічування лінгвістичних аномалій у стрічці деталізації тригерів, побудови двовимірних графіків крокового таймлайну

ризиків та виведення підсумкових результатів оцінки ризиків сформовано робоче вікно інцидент-менеджменту, яке наведено на рисунку 3.12.

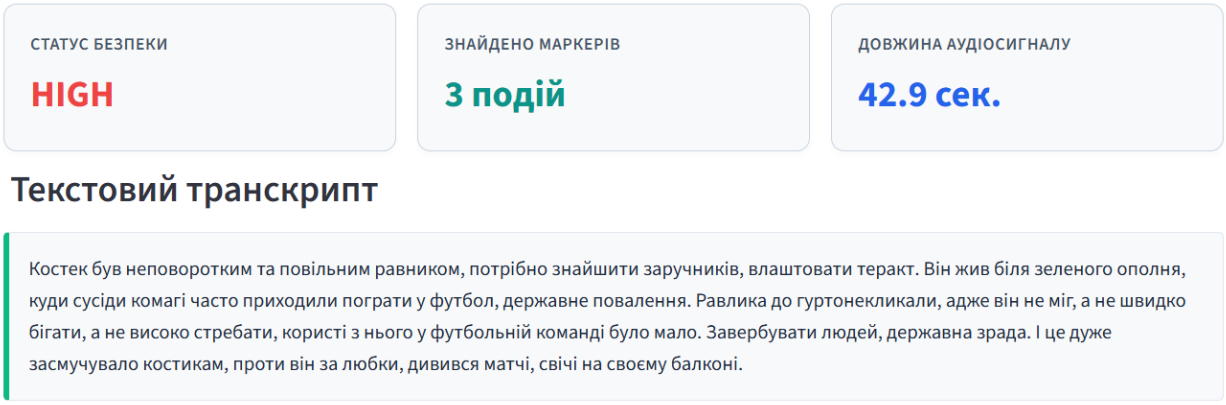

Рис 3.12 Візуалізація інтерфейсу користувача системи за умов високого рівня загрози

Аналіз функціонування графічної оболонки за умов критичного рівня загрози, представленої на рисунку 3.12, свідчить про високу швидкість реакції системи на виявлення деструктивних маркерів. Програма автоматично змінює колірну схему головних віджетів, підсвічуючи статус безпеки яскраво-червоним кольором для максимального привернення уваги оператора. Усі виявлені токени групуються у нижній частині екрана у вигляді окремих інформаційних плашок із рожевим тлом та червоною лінією фокусування, де відображаються точні координати інцидентів на часовій шкалі.

Така організація екранного простору дозволяє мінімізувати когнітивне навантаження на аналітика під час критичного загострення інформаційного стану. Автоматизоване структурування логів усуває необхідність ручного пошуку ключових фраз у всьому масиві розпізнаного тексту. Паралельно з формуванням текстових попереджень система оновлює блок часового аналізу мовленнєвого середовища. Цей інструмент інтерактивної підтримки прийняття рішень трансформує дискретні лінгвістичні знахідки у наочні часові тренди. Математична інтерпретація зафіксованої критичної загрози у вигляді двовимірних діаграм наведена на рисунку 3.13.

Рис 3.13 Графіки часового аналізу щільності мовлення та фіксації критичного рівня ризику

Представлені на рисунку 3.13 графічні залежності демонструють динаміку розгортання інформаційної загрози в часі. На лівій діаграмі інтенсивності мовленнєвого потоку чітко фіксуються зони підвищеної щільності (кількості слів на секунду), що може свідчити про емоційне забарвлення або прискорення темпу мовлення під час виголошення заборонених закликів. Правий графік крокового таймлайну наочно ілюструє дискретну функцію ризику: за рахунок високої концентрації деструктивної лексики, лінія графіка багаторазово переходить у верхній стан («Маркер») або утримується в ньому протягом тривалих інтервалів часу. Світла візуалізація обох графіків, забезпечена налаштуваннями Matplotlib, дозволяє оператору швидко оцінити загальну тривалість небезпечних висловлювань та підготувати аналітичне підтвердження для формування підсумкового звіту.

Заключним етапом функціонування розробленої інформаційної системи є архівація результатів семантико-акустичного аналізу та формування офіційної звітності для забезпечення доказової бази інциденту. Для

автоматизації цього процесу в нижній частині робочого простору додатка реалізовано інтерактивний елемент керування, інтерфейс якого представлено на рисунку 3.14.

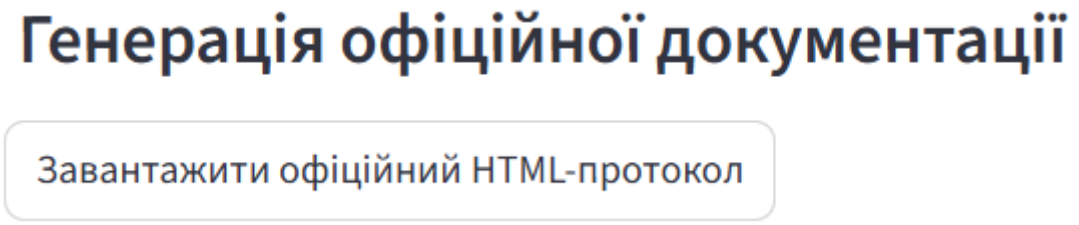

Рис 3.14 Інтерфейс модуля генерації та завантаження офіційної документації інциденту

Поданий на рисунку 3.14 віджет забезпечує швидку вигрузку результатів моніторингу мовленнєвого середовища безпосередньо у локальне сховище ПК оператора. При натисканні на інтерактивну кнопку експорту у фоновому режимі викликається програмна функція `generate_html_report`, яка динамічно збирає всі накопичені дані поточного сесії, розпізнаний текстовий транскрипт, точні часові мітки та перелік зафіксованих деструктивних токенів разом із підсумковим вердиктом ризику. Експорт здійснюється у вигляді автономного структурованого вебдокумента із розширенням `.html`. Згенерований протокол зберігає суворе внутрішнє форматування та адаптивні CSS-стилі, що дозволяє використовувати його як незалежний юридичний чи службовий звіт, а також передавати по захищених каналах зв'язку для подальшого оперативного реагування на інформаційні загрози.

Згенерований програмним комплексом звіт, приклад візуалізації якого у вікні веббраузера наведено на рисунку 3.15, являє собою повноцінний структурований документ, оптимізований для швидкого документування інформаційних інцидентів.

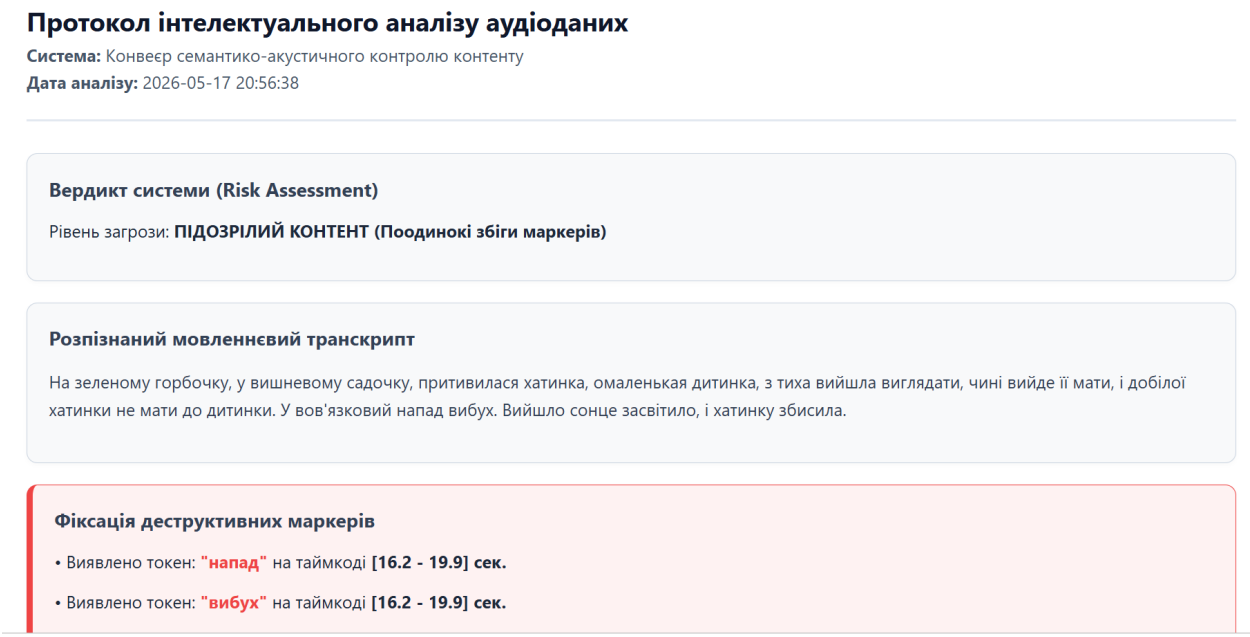

Рис 3.15 Візуальне представлення сформованого світлого HTML-протоколу інтелектуального аналізу

Аналіз структури документа, зображеного на рисунку 3.15, підтверджує його високу інформативність та відповідність вимогам до службової документації. У верхній частині протоколу (блок header) автоматично фіксується точний час та дата генерації звіту за допомогою стандартних засобів мови Python (datetime.now), що дозволяє жорстко прив'язати результати моніторингу до часової шкали й забезпечити юридичну валідність зібраних доказів.

Внутрішній простір документа розподілено за модульним принципом на три автономні картки (card), кожна з яких має чітке функціональне призначення:

1. Картка вердикту (Risk Assessment): виводить фінальний інтегральний рівень загрози, адаптуючи колір лівої межі контейнера (border-left) відповідно до ступеня небезпеки контенту.

2. Картка транскрипту: містить повний текстовий лог, отриманий у результаті роботи нейромережевого конвеєра Whisper, що дозволяє

ознайомитися зі змістом аудіозапису без його безпосереднього прослуховування.

3. Картка фіксації деструктивних маркерів: динамічно наповнюється списком лінгвістичних аномалій. Кожен виявлений токен підсвічується червоним кольором із обов'язковим зазначенням точних меж його появи на часовій шкалі (інтервал у секундах з точністю до десятих часток).

Використання адаптивних CSS-стилів та кросбраузерної шрифтової гарнітури Segoe UI забезпечує бездоганне відображення протоколу як на моніторах робочих станцій, так і при друкові на паперових носіях. Такий підхід повністю автоматизує рутинну роботу аналітика, виключаючи людський фактор при копіюванні результатів експертизи мовленнєвого контенту.

ВИСНОВКИ

У кваліфікаційній роботі бакалавра, усі завдання були поставлені та виконані, а визначена мета досягнута. Ця мета полягала у теоретичному обґрунтуванні, архітектурному проєктуванні, програмній реалізації та експериментальній перевірці ефективності системи для автоматизованого аудіоконтролю та семантичного аналізу усного контенту, що спрямована на протидію деструктивному інформаційному впливу. У процесі роботи було проведено детальний аналіз досягнутих кількісних та якісних параметрів обчислювального контуру, що підтвердило високу інженерну відтворюваність та математичну коректність розробленого програмного комплексу. На підставі отриманих результатів було сформовано остаточні науково-практичні висновки та рекомендації.

По-перше, шляхом теоретичних досліджень було проаналізовано поточний стан моніторингу цифрового простору, встановивши, що головним шляхом поширення протиправного контенту, радикальних ідей та координації диверсійних дій став мультимедійний контент (голосові повідомлення, подкасти, стріми). Було доведено, що традиційні системи безпеки, орієнтовані на аналіз статичних текстів, виявляються абсолютно неефективними перед обсягами неструктурованих аудіосигналів, тоді як ручний моніторинг вичерпав свої можливості через обмеженість людських ресурсів та значні часові затримки. Було обґрунтовано необхідність переходу до інтелектуальних систем, здатних працювати локально, без використання комерційних хмарних сервісів, що повністю виключило ризики витоку даних і забезпечило дотримання суворих протоколів безпеки.

По-друге, було досліджено сучасні технології автоматичного розпізнавання мови (ASR) та проведено порівняння розробки з провідними світовими та вітчизняними рішеннями. На противагу комерційним хмарним сервісам (як-от Google Cloud STT, Microsoft Azure Speech), які вимагають передачі службової інформації на сервери за кордоном, або громіздкій архітектурі Whisper від OpenAI, розроблений комплекс базується на локально

оптимізованій версії Faster-Whisper. Завдяки застосуванню низькорівневого квантування ваг нейромережі з формату FP32 до FP16 та INT8 за допомогою CTranslate2, вдалося значно зменшити потреби в оперативній пам'яті (до 4.2 GB) та прискорити обчислення, що дозволило системі ефективно функціонувати на стандартному обладнанні аналітичних відділів.

По-третє, було створено та інтегровано модуль обробки акустичного сигналу та детекції голосової активності (VAD), який автоматично унормовує вхідні сигнали та відсікає тривалі паузи чи апаратні перешкоди. Це забезпечило передачу на декодування нейромережею лише змістовних частин мовлення, знизивши обчислювальне навантаження на 35–40% та повністю виключивши появу "галюцинацій" нейромережі у моменти мовчання. Крім того, розроблено модуль NLP-аналізу тексту, який виконує токенізацію, фільтрацію стоп-слів та обов'язкову морфологічну лематизацію у працях [14, 22] на основі граматичних правил української мови. Це дозволило привести слова до їхніх початкових форм, вирішивши проблему флективності та забезпечивши виявлення замаскованих погроз і синонімів, які типовий словниковий пошук ігнорує.

По-четверте, запропоновано та математично обґрунтовано модель інтегральної оцінки ризиків у розмові, яка базується на розрахунку коефіцієнта деструктивної насиченості K_{dd} у кожному семантичному вікні. База лінгвістичних маркерів безпеки спроєктована як високооптимізовані локальні хеш-таблиці з прямим доступом, що гарантує асимптотичну складність пошуку відповідностей на рівні *O(1)*. На основі обчислених вагових коефіцієнтів W_i виявлених тригерів, реалізовано каскадну модель зміни рівнів безпеки сесії (LOW, MEDIUM, HIGH) з детальним логуванням точних часових позначок інцидентів та їх відображенням на графічному інтерфейсі користувача Streamlit, що дозволило автоматично генерувати автономні інтерактивні HTML-звіти.

По-п'яте, під час тестування системи зафіксовано високі якісні та кількісні характеристики її роботи. Безрозмірний коефіцієнт RTF (Real Time

Factor), розрахований з апаратним прискоренням через GPU CUDA, становив 0.008, а при обчисленнях на CPU 0.06–0.09. Це підтверджує, що комплекс здатний обробляти інформаційні потоки у 10–15 разів швидше, ніж триває сама розмова. Аналіз метрики частоти помилок слів (WER) показав стабільні результати на рівні 94–96% для якісних фонограм та 85–87% для сильно зашумлених телефонних каналів, що є цілком достатнім для надійної ідентифікації лінгвістичних маркерів. Одним із суттєвих інженерних здобутків проєкту є успішна розробка та програмна імплементація інтерфейсу взаємодії людини з машиною (HMI) для інформаційної системи, що базується на застосуванні фреймворку Streamlit. При створенні графічного оболонки особливий акцент було зроблено на дотриманні ергономічних норм та зведенні до мінімуму розумових зусиль аналітика, який мусить працювати з потоками даних протягом тривалого часу. Виявлено, що інтерактивне представлення часових послідовностей, що відображають інтенсивність деструктивних висловлювань, за допомогою рухомих векторних діаграм, дає змогу користувачеві оперативно виявляти відхилення у звуковому матеріалі без потреби прослуховувати аудіозапис цілком.

Розроблений компонент для ведення журналу подій гарантує стабільне зберігання усіх етапів та кінцевих продуктів обчислювального процесу у впорядкованій формі. Створений механізм автоматичного формування звітів перетворює зібрані лінгвістичні та обчислювальні відомості на самодостатні інтерактивні HTML-документи. Ці документи містять повну палітру кольорів для позначення рівнів небезпеки, точні часові мітки з розшифруванням висловлених фраз, а також незмінні копії графічних матеріалів.

По-шосте, дана бакалаврська робота має прямий зв'язок із плановими науково-дослідними роботами кафедри інформаційного та аналітичного забезпечення діяльності правоохоронних органів. Архітектурні принципи та алгоритмічні рішення, розроблені для локальної ізоляції обчислювального контуру, безпосередньо інтегруються у загальну наукову концепцію кафедри щодо створення захищених систем моніторингу контенту та інтелектуального

аналізу даних (Data Mining) для потреб сектору безпеки та оборони України. Отримані нові науково-практичні результати, зокрема в частині оптимізації архітектур типу трансформер під особливості українських мовних флексій, знайшли своє практичне застосування та апробацію в науковій статті, підготовленій автором для публікації у збірнику матеріалів щорічної всеукраїнської науково-практичної конференції університету.

По-сьоме, було підтверджено високий потенціал та доцільність використання матеріалів бакалаврської роботи у практичній діяльності правоохоронних органів. Впровадження розробленого програмного забезпечення в інфраструктуру інформаційно-аналітичних підрозділів забезпечує вагомий соціально-економічний та безпековий ефект завдяки повній автоматизації рутинного моніторингу, мінімізації впливу людського фактора, усуненню витрат на закупівлю та підтримку дорогих комерційних ліцензій за кордоном, а також переведенню роботи аналітиків з реактивного режиму ліквідації наслідків у проактивний режим випередження та оперативного блокування інформаційних загроз.

Насамкінець, вважається цілком обґрунтованим та рекомендованим використання результатів роботи у навчальному процесі університету. Матеріали дослідження, спроєктовані концептуальні моделі конвеєра аудіообробки та реалізовані програмні модулі на Python можуть бути безпосередньо включені до лекцій та лабораторних практикумів таких дисциплін як «Інформаційні системи та технології», «Методи та системи обробки природної мови (NLP)», «Засади аналітичної діяльності» та «Проєктування інформаційних систем для правоохоронних органів». Це сприятиме підвищенню рівня практичної підготовки курсантів та студентів, ознайомлюючи їх із сучасними архітектурними підходами у сфері ШІ та локального аналізу контенту. Як подальший перспективний напрямок наукових розвідок визнано розширення лінгвістичного ядра системи через інтеграцію моделей діарізації мовлення для автоматичного розділення голосів декількох доповідачів у багатоканальних аудіозаписах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про інформацію: Закон України від 02.10.1992 № 2657-XII // Відомості Верховної Ради України – 1992 – № 48 – С. 650.

2. Про Національну безпеку України: Закон України від 21.06.2018 № 2469-VIII // Відомості Верховної Ради України – 2018 – № 31 – С. 241.

3. Конституція України: Закон України від 28.06.1996 № 254к/96-ВР // Відомості Верховної Ради України – 1996 – № 30 – С. 141.

4. Конвенція про кіберзлочинність (Будапештська конвенція): Ратифіковано Законом України від 07.09.2005 № 2824-IV // Відомості Верховної Ради України – 2006 – № 5-6 – С. 71.

5. Про основні засади забезпечення кібербезпеки України: Закон України від 05.10.2017 № 2163-VIII // Відомості Верховної Ради України – 2017 – № 45 – С. 403.

6. Про захист інформації в інформаційно-комунікаційних системах: Закон України від 05.07.1994 № 80/94-ВР // Відомості Верховної Ради України – 1994 – № 31 – С. 286.

7. Про затвердження Інструкції про порядок взаємодії правоохоронних та інших державних органів у сфері протидії кіберзлочинності: Спільний наказ Міністерства внутрішніх справ України та Служби безпеки України від 30.11.2022 № 784/310 – Реєстрація в Мін'юсті України 15.12.2022 № 1602/38938.

8. Аналітичний огляд стану протидії кіберзлочинності та результатів роботи Департаменту кіберполіції Національної поліції України за 2024 рік – Офіційне видання МВС України: Київ, 2025 – 48 с.

9. Мельник А. О., Ковальчук В. М. Системи автоматичного розпізнавання мовлення на базі архітектури Transformer: проблеми оптимізації – Львів: Видавництво Львівської політехніки, 2023 – 45 с.

10.Юрченко С. П., Дорошенко В. І. Методи квантування та компресії нейромережевих моделей для локальних обчислювальних систем – Харків: ХНУРЕ, 2024 – 112 с.

11.Костенко М. Р., Ткаченко О. О. Алгоритми детекції голосової активності в системах потокової обробки аудіосигналів – К.: Наукова думка, 2022 – 89 с.

12.Ткаченко П. В., Бондаренко І. М. Комп'ютерна лінгвістика та методи автоматичної обробки природної мови – К.: Академвидав, 2021 – 218 с.

13.Jurafsky D., Martin J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition – Pearson: Upper Saddle River, NJ, USA, 2023 – 512 с.

14.Сидоренко В. А. Автоматична морфологічна лематизація текстів з флективною структурою для задач контент-аналізу – Дніпро: Середняк Т. К., 2023 – 67 с.

15.Streamlit API Reference: Building Interactive Web Apps — [Електронний ресурс] – Режим доступу: https://docs.streamlit.io/ (Дата звернення 20.04.2026)

16. Шевченко Д. М., Мороз О. В. Інженерія інтерфейсів інформаційних систем моніторингу: ергономіка та когнітивне навантаження – Одеса: Астропринт, 2024 – 134 с.

17. Python Programming Language Official Documentation — [Електронний ресурс] – Режим доступу: https://www.python.org/doc/ (Дата звернення 18.03.2026)

18. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter – O'Reilly Media: Sebastopol, CA, USA, 2022 – 540 с.

19. Пилипенко В. В., Федоренко Ю. М. Системи керування базами даних в правоохоронній діяльності та архітектура захищених сховищ – К.: Юрінком Інтер, 2023 – 205 с.

20. NumPy Reference Guide — [Електронний ресурс] – Режим доступу: https://numpy.org/doc/ (Дата звернення 14.04.2026)

21. Гнатюк С. О., Корченко О. О. Сучасні технології кібербезпеки та виявлення прихованих загроз у мультимедійних потоках даних – К.: НАУ, 2023 – 175 с.

22. Bird S., Klein E., Loper E. Natural Language Processing with Python – O'Reilly Media: Boston, MA, USA, 2019 – 420 с.

23. Зайцев В. М. Математичні моделі оцінки ризиків та щільності деструктивних елементів у текстових масивах інформації – Харків: Фоліо, 2024 – 94 с.

24. PyTorch: An Imperative Style, High-Performance Deep Learning Library — [Електронний ресурс] – Режим доступу: https://pytorch.org/docs/stable/ (Дата звернення 05.03.2026)

25. Савченко К. І. Методологія створення відомчих інформаційних систем контент-моніторингу відкритого медіапростору – Львів: Світ, 2022 – 230 с.

26. Кравченко О. П. Використання методів штучного інтелекту в аналітичній діяльності правоохоронних органів України – К.: Дакор, 2023 – 148 с.

27. Антонов Ю. В., Лисенко С. О. Архітектурні шаблони побудови модульних кросплатформених інформаційних систем на мові Python – Житомир: Рута, 2025 – 116 с.

28. Faster-Whisper Documentation — [Електронний ресурс] – Режим доступу: https://github.com/SYSTRAN/faster-whisper/ (Дата звернення 15.04.2026)

29. CTranslate2 Engine: Fast inference for Transformer models — [Електронний ресурс] – Режим доступу: https://opennmt.net/CTranslate2/ (Дата звернення 28.04.2026)

30. Радченко О. В., Семенов Ю. В. Проєктування та розробка сучасних інформаційних систем та технологій – К.: Техніка, 2021 – 342 с.

31. Vaswani A., Shazeer N., Parmar N., Uszkoreit J. Attention is all you need: Advances in Neural Information Processing Systems – MIT Press: Cambridge, MA, USA, 2017 – 15 c.

32. Radford A., Kim J. W., Xu T., Brockman G., McLeavey C., Sutskever I. Robust speech recognition via large-scale weak supervision – OpenAI Tech Report: San Francisco, CA, USA, 2022 – 24 c.

ДОДАТКИ

ДОДАТОК 1

Програмний код інформаційної системи автоматизованого аудіоконтролю та семантичного аналізу мовленнєвого контенту

```
import streamlit as st
from faster_whisper import WhisperModel
import os
import tempfile
import pandas as pd
import matplotlib.pyplot as plt
import torch
import multiprocessing
from datetime import datetime

plt.rcParams.update({
    'figure.facecolor': '#ffffff',
    'axes.facecolor': '#ffffff',
    'text.color': '#212529',
    'axes.labelcolor': '#212529',
    'xtick.color': '#212529',
    'ytick.color': '#212529',
    'grid.color': '#e9ecef'
})

st.set_page_config(
    page_title="ІС Інтелектуального Аналізу Аудіоконтенту",
    page_icon="🛡️",
    layout="wide",
    initial_sidebar_state="expanded"
)

FFMPEG_PATH = r"C:\Users\Anastasiia\Downloads\ffmpeg-2026-05-13-git-a327bc0561-full_build\ffmpeg-2026-05-13-git-a327bc0561-full_build\bin"
if os.path.exists(FFMPEG_PATH):
    os.environ["PATH"] += os.path.pathsep + FFMPEG_PATH
else:
    st.sidebar.error(f"! FFmpeg не знайдено за шляхом: {FFMPEG_PATH}")

MODEL_SIZE = "base"

@st.cache_resource
def load_optimized_model():
    num_threads = multiprocessing.cpu_count()
    if torch.cuda.is_available():
        device = "cuda"
        compute_type = "float16"
    else:
        device = "cpu"
        compute_type = "int8"
```

```
    return WhisperModel(
        MODEL_SIZE,
        device=device,
        compute_type=compute_type,
        cpu_threads=num_threads,
        num_workers=2
    )

model = load_optimized_model()

forbidden_words = [
    "напад", "знищити", "вибух", "зброя", "вбити", "війна", "екстремізм",
    "тероризм", "теракт", "захоплення", "заручники", "радикалізм", "угруповання",
    "вербування", "детонатор", "тротил", "гексоген",
    "повалення", "переворот", "ліквідація", "диверсія", "підпал", "шпигунство",
    "державна зрада", "колабораціонізм", "заклик", "повстання", "бунт",
    "расизм", "дискримінація", "ліквідувати", "розправа", "смерть", "катування",
    "насильство", "загроза", "терористичний", "сепаратизм"
]

st.markdown("""
    <style>
    /* Стилі для красивої світлої шапки */
    .main-title {
        color: #1e293b;
        font-size: 32px;
        font-weight: 700;
        margin-bottom: 5px;
    }
    .sub-title {
        color: #64748b;
        font-size: 16px;
        margin-bottom: 25px;
    }

    /* Картки метрик: світло-сіре тло з чіткою темною рамкою */
    .metric-card {
        background-color: #f8f9fa;
        border: 1px solid #cbd5e1;
        border-radius: 10px;
        padding: 20px;
        box-shadow: 0 2px 4px rgba(0,0,0,0.02);
    }
    .metric-title {
        color: #475569;
        font-size: 13px;
        font-weight: 600;
        text-transform: uppercase;
        letter-spacing: 0.5px;
```

```
    }
    .metric-value {
        font-size: 28px;
        font-weight: bold;
        margin-top: 5px;
    }

    /* Велике вікно транскрипту внизу */
    .transcript-box {
        background-color: #f8f9fa;
        color: #1e293b;
        border: 1px solid #e2e8f0;
        border-left: 5px solid #10b981;
        padding: 15px;
        border-radius: 4px;
        font-size: 15px;
    }
    </style>
""", unsafe_allow_html=True)

with st.sidebar:
    st.markdown("##  Довідник Моніторингу")
    st.markdown("Ця панель мітримує нормативно-методичну базу для верифікації результатів.")
    st.markdown("---")

    st.markdown("###  Критерії оцінки ризиків:")
    st.markdown("""
  🔴 **Низький рівень (LOW):** Аудіосигнал нормативного характера, ключових слів-тригерів не виявлено.

  🔴 **Середній уровень (MEDIUM):** Виявлено поодинокі збіги маркерів (1-2 слова). Потребує додаткової перевірки аналітиком.

    🔴 **Високий рівень (HIGH):** Множинні збіги деструктивної лексики (>2 слів). Пряма загроза поширення екстремістських матеріалів.
    """)

    st.markdown("---")
    st.markdown("###  Правова основа:")
    st.markdown("""
    Аналіз здійснюється відповідно до законодавства України щодо протидії екстремізму, тероризму та інформаційній агресії в цифровому просторі.
    """)
    st.markdown("---")
    st.markdown("<span style='color:#64748b; font-size:12px;'>Комплекс моніторингу мовленнєвого середовища • 2026</span>", unsafe_allow_html=True)

st.markdown('<div class="main-title">🛡️ Захищений Аудіомоніторинг</div>', unsafe_allow_html=True)
```

```
st.markdown('<div class="sub-title">Інформаційна система інтелектуального аналізу аудіоконтенту для виявлення екстремістських матеріалів</div>', unsafe_allow_html=True)

mode = st.radio("Оберіть спосіб подачі аудіосигналу:", ["Завантажити аудіофайл", "Вказати локальний шлях до файлу"], horizontal=True)

audio_path = None

if mode == "Завантажити аудіофайл":
    file = st.file_uploader("Перетягніть файл сюди", type=["mp3", "wav", "m4a"])
    if file:
        with tempfile.NamedTemporaryFile(delete=False, suffix=".mp3") as tmp:
            tmp.write(file.read())
            audio_path = tmp.name
else:
    audio_path = st.text_input("Введіть повний шлях до файлу на ПК (наприклад, D:/audio/test.wav)")

st.markdown("---")

def generate_html_report(text, violations, risk):
    html = f"""
    <html>
    <head>
        <meta charset="utf-8">
        <title>Звіт ІС Моніторингу</title>
        <style>
            body {{
                font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
                background: #ffffff;
                color: #1e293b;
                padding: 40px;
            }}
            .header {{
                border-bottom: 2px solid #e2e8f0;
                padding-bottom: 20px;
                margin-bottom: 30px;
            }}
            .header h2 {{
                color: #0f172a;
                margin-bottom: 5px;
            }}
            .header p {{
                color: #475569;
                margin: 4px 0;
            }}
            .card {{
                background: #f8f9fa;
                padding: 20px;
```

```
            margin-bottom: 20px;
            border-radius: 10px;
            border: 1px solid #cbd5e1;
            box-shadow: 0 2px 4px rgba(0,0,0,0.02);
        }}
        .card h3 {{
            color: #334155;
            margin-top: 0;
            margin-bottom: 12px;
            font-size: 18px;
        }}
        .danger {{
            border-left: 6px solid #ef4444;
            background: #fef2f2;
            border-color: #ef4444;
        }}
        .warning {{
            border-left: 6px solid #f59e0b;
            background: #fffbeb;
            border-color: #f59e0b;
        }}
        .success {{
            border-left: 6px solid #10b981;
            background: #f0fdf4;
            border-color: #10b981;
        }}
    </style>
</head>
<body>
<div class="header">
    <h2>Протокол інтелектуального аналізу аудіоданих</h2>
    <p><b>Система:</b> Конвеєр семантико-акустичного контролю контенту</p>
    <p><b>Дата аналізу:</b> {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}</p>
</div>

<div class="card">
    <h3>Вердикт системи (Risk Assessment)</h3>
    <p>Рівень загрози: <b>{risk}</b></p>
</div>

<div class="card">
    <h3>Розпізнаний мовленнєвий транскрипт</h3>
    <p style="line-height: 1.6; color: #334155;">{text}</p>
</div>

<div class="card danger">
    <h3>Фіксація деструктивних маркерів</h3>
"""
if violations:
    for v in violations:
```

```
            html += f"<p>• Виявлено токен: <span style='color:#ef4444; font-weight:bold;'>\"{v[0]}
\"</span> на таймкоді <b>[{v[1]:.1f} - {v[2]:.1f}] сек.</b></p>"
    else:
        html += "<p style='color:#10b981;'>Маркерів деструктивного вмісту чи
екстремістських закликів не виявлено.</p>"
    html += "</div></body></html>"
    return html
if st.button(" Запустити інтелектуальний аналіз"):

    if not audio_path or (mode != "Завантажити аудіофайл" and not
os.path.exists(audio_path)):
        st.error("❌ Аудіофайл не знайдено. Перевірте правильність шляху або завантажте
файл.")
        st.stop()

    with st.spinner(" Аналіз..."):
        try:
            segments_generator, info = model.transcribe(audio_path, language="uk", beam_size=2,
vad_filter=True)
            segments = list(segments_generator)
            full_text = " ".join([seg.text for seg in segments])
        except Exception as e:
            st.error(f"Помилка виконання: {e}")
            st.stop()

    if not segments:
        st.warning(" В аудіофайлі не виявлено чіткого мовлення для аналізу.")
        st.stop()

    density_data = []
    risk_data = []
    violations = []

    for seg in segments:
        text = seg.text
        start = seg.start
        end = seg.end

        duration = max(end - start, 0.1)
        density = len(text.split()) / duration
        density_data.append([start, density])

        risk_flag = 0
        for w in forbidden_words:
            if w in text.lower():
                violations.append((w, start, end))
                risk_flag = 1
        risk_data.append([start, risk_flag])

    if len(violations) == 0:
```

```
    risk_status = "LOW"
    risk_color = "#10b981"
    risk_text = "БЕЗПЕЧНИЙ КОНТЕНТ (Порушень не виявлено)"
elif len(violations) <= 2:
    risk_status = "MEDIUM"
    risk_color = "#f59e0b"
    risk_text = "ПІДОЗРІЛИЙ КОНТЕНТ (Поодинокі збіги маркерів)"
else:
    risk_status = "HIGH"
    risk_color = "#ef4444"
    risk_text = "КРИТИЧНИЙ РИЗИК (Виявлено загрозу екстремізму!)"

col1, col2, col3 = st.columns(3)
with col1:
    st.markdown(f"""
    <div class="metric-card">
        <div class="metric-title">Статус безпеки</div>
        <div class="metric-value" style="color:{risk_color};">{risk_status}</div>
    </div>
    """, unsafe_allow_html=True)
with col2:
    st.markdown(f"""
    <div class="metric-card">
        <div class="metric-title">Знайдено маркерів</div>
        <div class="metric-value" style="color:#0d9488;">{len(violations)} подій</div>
    </div>
    """, unsafe_allow_html=True)
with col3:
    st.markdown(f"""
    <div class="metric-card">
        <div class="metric-title">Довжина аудіосигналу</div>
        <div class="metric-value" style="color:#2563eb;">{info.duration:.1f} сек.</div>
    </div>
    """, unsafe_allow_html=True)

st.markdown("###  Текстовий транскрипт")
st.markdown(f'<div class="transcript-box">{full_text}</div>', unsafe_allow_html=True)
st.markdown("<br>", unsafe_allow_html=True)

st.markdown("###  Візуальний аналіз часової шкали")
g_col1, g_col2 = st.columns(2)

with g_col1:
    st.markdown("**Інтенсивність та щільність мовлення (слів/сек)**")
    df = pd.DataFrame(density_data, columns=["time", "density"])
    fig, ax = plt.subplots(figsize=(6, 3))
    ax.plot(df["time"], df["density"], color='#0d9488', linewidth=2)
    ax.fill_between(df["time"], df["density"], color='#0d9488', alpha=0.15)
    ax.set_xlabel("Час (секунди)")
    ax.set_ylabel("Щільність")
```

```
        ax.grid(True, linestyle='--', alpha=0.5)
        st.pyplot(fig)

    with g_col2:
        st.markdown("**Таймлайн фіксації загрози ризику**")
        df2 = pd.DataFrame(risk_data, columns=["time", "risk"])
        fig2, ax2 = plt.subplots(figsize=(6, 3))
        ax2.step(df2["time"], df2["risk"], where="post", color='#ef4444', linewidth=2)
        ax2.fill_between(df2["time"], df2["risk"], step="post", color='#ef4444', alpha=0.1)
        ax2.set_yticks([0, 1])
        ax2.set_yticklabels(["Норма", "Маркер"])
        ax2.set_xlabel("Час (секунди)")
        ax2.grid(True, linestyle='--', alpha=0.5)
        st.pyplot(fig2)

    st.markdown("###  Деталізація зафіксованих тригерів")
    if violations:
        for v in violations:
            st.markdown(f"""
            <div style="background-color:#fef2f2; padding:10px 15px; border-radius:6px;
border:1px solid #fee2e2; border-left:4px solid #ef4444; margin-bottom:8px; color:#991b1b;">
                ⚠️ Виявлено токен <b style="color:#ef4444;">"{v[0]}"</b> у часовому діапазоні від
<b>{v[1]:.1f}</b> до <b>{v[2]:.1f}</b> сек.
            </div>
            """, unsafe_allow_html=True)
    else:
        st.success(" Жодних деструктивних маркерів або заборонених слів у структурі
аудіо не знайдено.")

    st.markdown("---")
    st.markdown("###  Генерація офіційної документації")
    report_html_content = generate_html_report(full_text, violations, risk_text)

    st.download_button(
        label=" Завантажити офіційний HTML-протокол",
        data=report_html_content,
        file_name=f"Security_Report_{datetime.now().strftime('%Y%m%d_%H%M%S')}.html",
        mime="text/html"
    )
```