Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

Допущено до захисту
Завідувач кафедри комп’ютерних наук,
доктор технічних наук, професор
Андрій БОНДАРЧУК
« 01 » червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»
Спеціальність 122 Комп’ютерні науки
Освітня програма 122.00.01 Інформатика

АВТОМАТИЗОВАНА СИСТЕМА ІНТЕЛЕКТУАЛЬНОЇ МОДЕРАЦІЇ ВИЯВЛЕННЯ СПАМУ ТА ЗАБОРОНЕНОГО КОНТЕНТУ В СОЦІАЛЬНИХ МЕРЕЖАХ

Виконав
студент групи Інб-2-22-4.0д
Димерлій Денис Сергійович
_____

Науковий керівник
к.т.н.., ст. досл.
Багацький О.В.
_____

Київ – 2026

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

«Затверджую»
Завідувач кафедри комп'ютерних наук
доктор технічних наук, професор
Андрій БОНДАРЧУК
«31» жовтня 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

"Автоматизована система інтелектуальної модерації виявлення спаму та забороненого контенту в соціальних мережах**"**

Виконавець – студент групи ІНб-2-22-4.0д,
спеціальності 122 Комп'ютерні науки,
освітньої програми 122.00.01 Інформатика
Димерлій Денис Сергійович

1. Вихідні дані: наукові публікації, документація програмних засобів і сервісів, нормативні та методичні матеріали за напрямом дослідження.
2. Основні завдання: Дослідити теоретичні основи автоматизованої модерації цифрового контенту та визначити роль методів штучного інтелекту в аналізі текстових і аудіоповідомлень. Проаналізувати сучасні підходи, моделі та сервіси, що застосовуються для виявлення токсичності, спаму, шахрайства, нецензурної лексики та емоційного тону. Реалізувати застосунок для аналізу текстових та аудіоповідомлень, інтегрувати модулі оцінювання ризику, забезпечити збереження історії перевірок і формування підсумкових звітів.
3. Пояснювальна записка: Обсяг - до 45 стор. формату А4 комп'ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.
4. Графічні матеріали: презентація, структурна схема системи, схема взаємодії модулів, діаграма бази даних, UML-діаграми, інтерфейси вебзастосунку.
5. Додатки: фрагменти програмного коду, додаткові схеми та ілюстрації.
6. Строк подання роботи на кафедру: «01» червня 2026р.

Науковий керівник	Виконавець:
к.т.н.., ст. досл.	
_____ Багацький О.В.	_____ Димерлій Д.С.
_____ «31» жовтня 2025 р	_____«31» жовтня 2025 р

Календарний план

№	Етап виконання роботи	Термін виконання	Примітка
1	Формування теми дипломної роботи бакалавра та її затвердження	31.10.2025	**Виконано**
2	Аналіз області дослідження, огляд існуючих технологій і рішень	01.11.2025 - 20.11.2025	**Виконано**
3	Складання змісту, календарного плану та літературних джерел	21.11.2025 - 20.12.2025	**Виконано**
4	Написання вступу та реферату пояснювальної записки	21.12.2025 - 20.01.2026	**Виконано**
5	Написання першого розділу пояснювальної записки	21.01.2026 - 15.02.2026	**Виконано**
6	Проєктування архітектури системи та структури бази даних	16.02.2026 - 28.02.2026	**Виконано**
7	Розробка backend-модулів аналізу текстових та аудіоповідомлень	01.03.2026 - 10.03.2026	**Виконано**
8	Інтеграція AI-сервісів, модуля агрегування результатів та вебінтерфейсу	11.03.2026 - 15.03.2026	**Виконано**
9	Тестування системи, аналіз результатів, написання другого, третього розділів та висновків	16.03.2026 - 15.04.2026	**Виконано**
10	Подання роботи на перевірку, внесення правок від керівника і проходження антиплагіату	29.04.2026 - 15.05.2026	**Виконано**
11	Захист кваліфікаційної роботи	15.06.2026	

«Затверджую»

Науковий керівник **Індивідуальний план склав**

к.т.н.., ст. досл., Багацький О. В. Димерлій Д.С.

_____ _____

РЕФЕРАТ

Кваліфікаційна робота: 34 с., 6 рис., 3 табл., 36 джерел, 2 додатки.

Актуальність: у зв’язку зі стрімким зростанням обсягів цифрового контенту виникає потреба в ефективних засобах його швидкої та надійної модерації. Поширення повідомлень у соціальних мережах, чатах, сервісах підтримки та інших інформаційних системах ускладнює ручний контроль вмісту, тому особливого значення набуває використання методів штучного інтелекту для автоматизованого аналізу й оцінювання ризику.

Об’єкт дослідження: процес автоматизованої модерації користувацького цифрового контенту в сучасних інформаційних системах.

Предмет дослідження: методи обробки природної мови, сервіси штучного інтелекту, алгоритми оцінювання ризику та програмні засоби реалізації системи модерації.

Мета роботи: розробити інтелектуальну систему автоматизованої модерації цифрового контенту, здатну аналізувати текстові та аудіоповідомлення, формувати оцінку ризику та надавати користувачеві результат перевірки.

Завдання:

1. дослідити сучасні підходи до модерації цифрового контенту та визначити роль методів штучного інтелекту в автоматизованому аналізі повідомлень;
2. проаналізувати види небажаного контенту та методи їх виявлення;
3. обґрунтувати архітектуру та технологічні засоби реалізації системи;
4. реалізувати модулі аналізу текстових та аудіоповідомлень;
5. забезпечити формування підсумкової оцінки ризику, збереження результатів перевірки та формування звітів.

Основні результати: у процесі виконання роботи було розроблено функціональний прототип системи автоматизованої модерації цифрового контенту на основі FastAPI, PostgreSQL, SQLAlchemy, Jinja2, Docker і зовнішніх AI-сервісів.

Практичне значення дослідження: полягатиме у впровадженні розробленої системи у вебплатформах, сервісах підтримки користувачів, чатах і маркетплейсах для автоматизованого контролю безпечності цифрового контенту.

Ключові слова: автоматизована модерація, штучний інтелект, цифровий контент, обробка природної мови, токсичність, спам, шахрайств

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ASR – автоматичне розпізнавання мовлення.

CSS – каскадні таблиці стилів.

Docker – платформа контейнеризації програмного забезпечення.

FastAPI – вебфреймворк для розроблення серверних застосунків мовою Python.

Jinja2 – шаблонізатор для формування HTML-сторінок.

NLP – обробка природної мови.

PostgreSQL – реляційна система керування базами даних.

SQLAlchemy – програмна бібліотека для роботи з базами даних у Python.

URL – уніфікований локатор ресурсу.

UVicorn – ASGI-сервер для запуску застосунків FastAPI.

X-Ray – механізм пояснення результатів модерації через відображення фрагментів тексту, що вплинули на підсумкове рішення.

ЗМІСТ

ВСТУП

Актуальність теми дипломної роботи визначається стрімким зростанням обсягів цифрового контенту, який щосекунди створюється та поширюється в соціальних мережах, месенджерах, форумах, новинних порталах і корпоративних інформаційних системах. Разом із корисною інформацією в цифровому середовищі активно поширюються токсичні повідомлення, спам, шахрайські звернення, фішингові посилання, нецензурна лексика та інші форми небажаного контенту. Наявність таких матеріалів негативно впливає на якість комунікації, безпеку користувачів, репутацію платформ та ефективність адміністрування онлайн-сервісів.

Традиційна ручна модерація контенту, хоча і залишається актуальною в окремих випадках, уже не може забезпечити необхідну швидкість обробки повідомлень у великих інформаційних системах. Оператор-модератор обмежений у часі, а також може приймати суб’єктивні рішення залежно від контексту, навантаження та власного досвіду.

У сучасних умовах підвищену ефективність демонструють гібридні системи модерації, які не обмежуються одним алгоритмом чи окремою моделлю машинного навчання, а поєднують декілька незалежних джерел оцінювання. Такий підхід дає змогу враховувати різні аспекти ризику: токсичність, спам-активність, шахрайські патерни, емоційне забарвлення та вживання нецензурної лексики.

Об’єктом дослідження є процес автоматизованої модерації користувацького цифрового контенту в інформаційних системах.

Предметом дослідження є методи обробки природної мови, сервіси штучного інтелекту, алгоритми агрегування результатів та програмні засоби реалізації веборієнтованої системи модерації контенту.

Метою дипломної роботи є розроблення інтелектуальної вебсистеми автоматизованої модерації контенту, яка забезпечує аналіз текстових та аудіоповідомлень, інтегрує декілька незалежних сервісів штучного інтелекту, формує підсумкову оцінку ризику та надає користувачу детальний результат перевірки.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

- Проаналізувати сучасний стан автоматизованої модерації цифрового контенту.
- Дослідити основні види небажаного контенту та методи його виявлення.
- Спроєктувати структуру програмного застосунку, базу даних і логіку обробки користувацьких запитів.
- Реалізувати вебзастосунок для аналізу текстового та аудіоконтенту.
- Реалізувати механізм агрегування результатів незалежних джерел аналізу.
- Провести тестування системи та визначити напрями її подальшого розвитку.

Методи дослідження включають аналіз предметної області, структурне та модульне проєктування, методи обробки природної мови, інтеграцію зовнішніх API, засоби контейнеризації та практичне тестування працездатності програмної системи.

Практичне значення роботи полягає у створенні функціонального програмного прототипу, який може бути використаний як основа для модерації повідомлень у соціальних сервісах, системах підтримки користувачів, корпоративних платформах, освітніх ресурсах та інших вебзастосунках, де необхідний попередній контроль якості вхідного контенту.

РОЗДІЛ 1

АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ

1.1 Характеристика проблеми модерації цифрового контенту

Сучасне цифрове середовище характеризується безперервним створенням і поширенням користувацького контенту. Повідомлення, коментарі, публікації, звернення до служб підтримки, голосові повідомлення та інші форми даних надходять до інформаційних систем у режимі реального часу, що істотно ускладнює їх оперативну перевірку. За таких умов повністю ручна модерація втрачає ефективність, оскільки людський ресурс не здатний забезпечити одночасно високу швидкість реакції, стабільність прийняття рішень і повноту охоплення всього потоку повідомлень [1], [2].

Наявність немодерованого або несвоєчасно обробленого небажаного контенту створює для цифрових платформ низку ризиків. По-перше, знижується рівень довіри користувачів до середовища комунікації та самої платформи. По-друге, погіршується якість інформаційного обміну через поширення образ, агресії, нав’язливої реклами або шахрайських повідомлень. По-третє, порушення внутрішніх правил спільноти чи вимог зовнішнього регулювання може мати правові, репутаційні та фінансові наслідки для власників сервісу. Таким чином, модерація контенту є не лише технічною задачею, а й важливою складовою інформаційної безпеки та управління цифровою платформою [1], [2].

Складність цієї проблеми зумовлена її міждисциплінарним характером. З технічного погляду необхідно аналізувати великі обсяги різнорідних даних, часто в умовах обмеженого часу відповіді. З організаційного погляду важливо зменшити навантаження на модераторів і передавати людині лише ті випадки, які справді потребують додаткової перевірки. З етичного погляду система має

зберігати баланс між свободою висловлювання та захистом користувачів від шкідливих або маніпулятивних повідомлень. Із безпекового погляду особливо важливим є раннє виявлення шахрайських, фішингових або соціально небезпечних звернень до моменту їх масового поширення [2-4].

Додатковою проблемою є постійна зміна форми цифрового контенту. Користувачі застосовують сленг, скорочення, навмисні орфографічні викривлення, транслітерацію, змішані мови, емодзі, скорочені URL-адреси та інші прийоми, що дозволяють маскувати небажаний зміст і обходити прості фільтри. Унаслідок цього система модерації, побудована лише на статичних списках заборонених слів або жорстких правилах, демонструє обмежену ефективність. Практично цінним стає підхід, здатний враховувати не тільки окремі лексичні ознаки, а й контекст висловлювання, емоційне забарвлення, ймовірний намір автора та сукупність незалежних індикаторів ризику [5-8].

1.2 Основні категорії небажаного контенту

Для побудови прикладної системи автоматизованої модерації доцільно формалізувати основні категорії небажаного контенту, які мають найбільше практичне значення для вебплатформ, чатів і сервісів обробки користувацьких звернень. Таке структурування потрібне не лише для теоретичного опису предметної області, а й для подальшого вибору відповідних інструментів аналізу, джерел ознак та логіки агрегування результатів [2].

Першою важливою категорією є токсичний контент. До нього належать образи, приниження, агресивні висловлювання, мова ненависті, погрози, психологічний тиск і різні форми конфліктної комунікації. Наявність подібних повідомлень негативно впливає на якість взаємодії між користувачами, формує ворожу атмосферу та може призводити до ескалації конфліктів у цифровому середовищі. Саме тому виявлення токсичності є однією з базових функцій сучасних систем модерації [4-6].

Другою категорією є спам. У загальному випадку спамовий контент характеризується нав’язливим рекламним або маніпулятивним характером, повторюваністю повідомлень, штучним привернення уваги, надмірним використанням закликів до дії, підозрілих посилань або гіперболізованих маркетингових формулювань. Хоча спам не завжди є токсичним, він суттєво знижує якість користувацького досвіду, перевантажує канали комунікації та нерідко слугує оболонкою для шахрайських сценаріїв [9].

Окрему підвищено ризикову категорію становлять шахрайські та фішингові повідомлення. Їхньою метою зазвичай є виманювання коштів, банківських реквізитів, паролів, кодів підтвердження, персональних даних або інших конфіденційних відомостей. Такі повідомлення можуть імітувати офіційні звернення, експлуатувати фактор терміновості, використовувати психологічний тиск або посилання на підроблені ресурси. У контексті безпеки саме ця категорія становить найбільшу загрозу, оскільки її наслідком може бути не лише порушення правил спільноти, а й реальна матеріальна шкода для користувачів [3].

Наступною категорією є нецензурна та ненормативна лексика. Її наявність не завжди тотожна високій токсичності, однак часто виступає важливим індикатором неприйнятного стилю комунікації. Для прикладної системи модерації виявлення такої лексики є корисним як для самостійної сигналізації про порушення, так і як додатковий фактор у загальній оцінці ризику повідомлення [5], [6].

Ще однією значущою категорією є контент із вираженим негативним емоційним забарвленням. Аналіз емоційного тону не замінює семантичної класифікації, проте доповнює її, дозволяючи виявляти повідомлення з високим рівнем роздратування, ворожості, зневаги або емоційного тиску. Такий сигнал є корисним у випадках, коли текст ще не містить явних образ чи заборонених слів, але вже демонструє ознаки потенційно конфліктної поведінки [10].

Суттєвою особливістю перелічених категорій є їх часткове перетинання. Одне й те саме повідомлення може одночасно містити ознаки кількох типів небажаного контенту: наприклад, спамове повідомлення може бути шахрайським, а токсичний коментар може супроводжуватися ненормативною лексикою та сильним негативним емоційним фоном. Саме ця багатовимірність предметної області зумовлює потребу в гібридному підході до модерації, у межах якого різні модулі аналізують контент з кількох незалежних сторін [2], [4].

1.3 Аналіз підходів до автоматизованої модерації

На сучасному етапі розвитку систем аналізу тексту можна виокремити кілька базових підходів до автоматизованої модерації контенту, які відрізняються принципами роботи, вимогами до ресурсів, гнучкістю та точністю результатів.

Найпростішим підходом є системи на основі правил, тобто механізми, у яких рішення про модерацію приймається за наперед заданими списками ключових слів, шаблонами, регулярними виразами та формальними умовами. Їх перевага полягає в прозорості реалізації, легкості контролю та мінімальних вимогах до обчислювальних ресурсів. Проте ефективність таких систем суттєво знижується в ситуаціях, коли небажаний зміст передається не прямо, а через контекст, евфемізми, навмисні викривлення написання чи мовні варіації. Тому підхід на основі правил доцільно використовувати переважно як допоміжний або фільтраційний рівень, але не як єдиний механізм прийняття рішення.

Наступну групу становлять класичні методи машинного навчання, зокрема наївний баєсівський класифікатор, логістична регресія, метод опорних векторів та інші моделі, що працюють на попередньо підготовлених ознаках. Такі підходи можуть демонструвати прийнятні результати на добре розмічених наборах даних і є порівняно легкими для навчання та інтерпретації. Водночас їхня якість значною мірою залежить від якості ознак, а здатність враховувати складний

мовний контекст, приховані семантичні зв’язки або змішані комунікативні наміри користувача залишається обмеженою.

Суттєво вищий рівень якості забезпечують сучасні нейромережеві трансформерні моделі, які здатні враховувати контекст повідомлення, взаємозв’язок між словами та приховані семантичні патерни [7], [8]. Саме трансформерні архітектури сьогодні лежать в основі багатьох високоточних систем класифікації тексту, визначення токсичності, виявлення спаму, аналізу тональності та інших задач обробки природної мови. Їхнім недоліком є вища складність упровадження, значні вимоги до обчислювальних ресурсів при локальному розгортанні та необхідність постійної підтримки моделей.

Для задач багатомовної класифікації доцільним є використання моделей сімейства XLM-RoBERTa, побудованих на основі підходів до масштабного крослінгвального представлення тексту [11], [12]. Це особливо важливо для систем модерації, які працюють із повідомленнями різними мовами або з мовно змішаним контентом, оскільки дозволяє зберігати прийнятну якість аналізу без потреби створювати окрему модель для кожної мови.

Окрему практично важливу категорію становлять зовнішні AI-сервіси та inference API, які надають доступ до готових моделей без необхідності самостійного навчання та локального обслуговування інфраструктури [13], [14]. Такий підхід особливо зручний для прикладних проєктів, де важливо швидко інтегрувати кілька незалежних інтелектуальних функцій, зосередившись на архітектурі системи, логіці обробки даних та інтерфейсі користувача. Недоліками є залежність від сторонніх постачальників, мережеві затримки, зовнішні обмеження за квотами та потенційна зміна поведінки сервісів.

З огляду на переваги й обмеження кожного з наведених підходів найбільш доцільною для прикладної системи модерації є гібридна архітектура. У такій архітектурі фінальний висновок формується не одним джерелом, а на основі сукупності кількох незалежних модулів аналізу. Це дозволяє компенсувати

слабкі сторони окремих моделей, підвищити стійкість до помилок та отримати більш збалансоване рішення щодо рівня ризику повідомлення.

У межах даної дипломної роботи саме гібридний підхід є найбільш раціональним компромісом між якістю результатів, складністю реалізації та прикладною цінністю. Повноцінне локальне навчання набору спеціалізованих моделей вимагало б значних обчислювальних ресурсів, часу на підготовку датасетів та окремої експериментальної інфраструктури. Натомість використання доступних моделей через API у поєднанні з власною логікою агрегування результатів дозволяє зосередити увагу на створенні завершеної програмної системи модерації, придатної до практичного використання.

1.4 Аналіз існуючих систем та сервісів автоматизованої модерації контенту

Для обґрунтування архітектури розроблюваної системи доцільно проаналізувати не окремі технічні компоненти, а вже існуючі сервіси та платформи, які можуть розглядатися як аналоги або часткові прототипи систем автоматизованої модерації контенту. Такий підхід дає змогу оцінити їх функціональні можливості, сильні сторони, обмеження та визначити, які саме рішення доцільно врахувати під час проєктування власної системи.

Як критерії порівняння було обрано такі характеристики: типи підтримуваного контенту, категорії виявлення небажаного вмісту, можливість інтеграції через API, наявність пояснюваності результатів, підтримка багатомовності, а також придатність до побудови прикладної вебсистеми модерації.

Серед сучасних рішень, близьких до тематики роботи, доцільно виділити Perspective API, OpenAI Moderation API, Hive Moderation та Azure AI Content Safety. Зазначені сервіси орієнтовані на виявлення небажаного або ризикового

контенту, однак відрізняються за функціональністю, глибиною аналізу та сферою застосування.

Таблиця 1.1

Аналіз та порівняння існуючих систем і сервісів модерації контенту

Сервіс	Типи Контенту	Основні можливості	Переваги	Обмеження
Perspective API	текст	оцінка токсичності, агресії, образливості	проста інтеграція, зручність для аналізу коментарів	орієнтація переважно на текстову токсичність
OpenAI Moderation API	текст, зображення	виявлення небезпечного контенту за категоріями harm/safety	офіційний API, швидка інтеграція, сучасні моделі безпеки	не охоплює повноцінно задачі спаму, шахрайства та прикладної бізнес-логіки модерації
Hive Moderation	текст, зображення, відео, аудіо	модерація мультимодального контенту, правила, moderation dashboard	широка функціональність, орієнтація на trust & safety	складніше та важче рішення для інтеграції невеликих проєктів
Azure AI Content Safety	текст, зображення	виявлення harmful content, severity-рівні, custom categories	корпоративна інтеграція, масштабованість, підтримка кастомних категорій	частина можливостей і мовна підтримка залежать від конкретних моделей та сценаріїв

Проведене порівняння показує, що жоден із розглянутих сервісів не покриває повністю всі вимоги, поставлені до системи, що розробляється в межах дипломної роботи. Одні рішення спеціалізуються переважно на токсичності тексту, інші орієнтовані на загальну перевірку небезпечного контенту, а більш комплексні платформи є надлишковими або менш гнучкими для побудови власного прикладного вебзастосунку. Крім того, для поставленої задачі важливими є не лише окремі класифікації, а й можливість агрегування

результатів із кількох джерел, підтримка аналізу аудіоповідомлень, збереження історії перевірок, формування PDF-звітів та пояснення отриманого результату.

Отже, аналіз існуючих аналогів підтверджує доцільність побудови гібридної системи автоматизованої модерації контенту, у якій поєднуються декілька незалежних сервісів аналізу. Саме такий підхід дає змогу підвищити повноту оцінювання, зменшити залежність від одного джерела класифікації та реалізувати прикладну систему, адаптовану до потреб вебплатформ, чатів і сервісів підтримки користувачів.

РОЗДІЛ 2

ПРОЄКТУВАННЯ СИСТЕМИ АВТОМАТИЗОВАНОЇ МОДЕРАЦІЇ КОНТЕНТУ

2.1 Загальна архітектурна концепція

Розроблена система є веборієнтованим застосунком, що реалізує клієнт-серверну модель взаємодії. Користувач працює через браузер, у якому вводить текстове повідомлення, завантажує аудіофайл або ініціює запис аудіо з мікрофона. Серверна частина приймає запит, виконує попередню валідацію даних, запускає набір незалежних модулів аналізу, агрегує отримані результати, зберігає їх у базі даних та повертає користувачеві сторінку з підсумковим висновком [15-19].

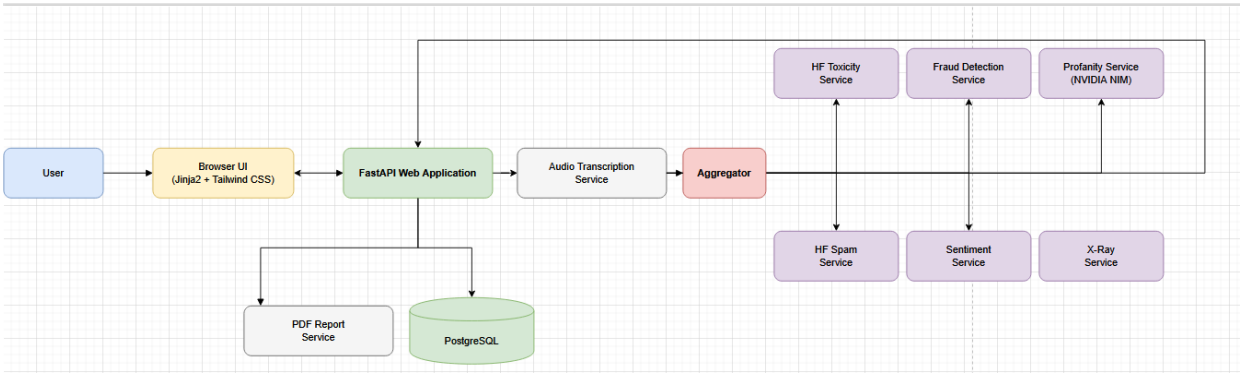

Рисунок 2.1. Загальна архітектура системи

Архітектурно система складається з таких підсистем:

- Підсистема взаємодії з користувачем.
- Підсистема серверної логіки.
- Підсистема інтеграції з зовнішніми AI-сервісами.
- Підсистема агрегування результатів.
- Підсистема зберігання історії перевірок.
- Підсистема формування звітів.

Підсистема взаємодії з користувачем забезпечує роботу з вебінтерфейсом, приймання текстових даних, завантаження аудіофайлів, відображення результатів перевірки, перегляд історії модерації та формування запиту на завантаження PDF-звіту. Підсистема серверної логіки координує маршрутизацію, валідацію вхідних даних, запуск модулів аналізу та підготовку відповіді користувачеві. Підсистема інтеграції з зовнішніми AI-сервісами відповідає за взаємодію з Hugging Face Inference API та NVIDIA NIM, тобто фактично реалізує зовнішній інтелектуальний рівень системи [13], [14], [18].

Підсистема агрегування результатів виконує нормалізацію часткових оцінок, узгодження форматів відповідей окремих джерел аналізу та формування інтегрального показника ризику. Підсистема зберігання історії перевірок забезпечує запис результатів до бази даних, а підсистема формування звітів відповідає за генерацію структурованого PDF-документа з підсумками аналізу [20-22].

Подібний поділ є важливим не лише з погляду теорії проєктування, а й з практичної точки зору. Чітке розмежування відповідальності між підсистемами зменшує зв’язність коду, спрощує тестування, полегшує супровід і дозволяє поступово розширювати функціональність без повного перегляду загальної архітектури [15], [16], [21]. Наприклад, у майбутньому до системи можна додати окремий модуль аналізу зображень, нове джерело AI-оцінювання або адміністративний інтерфейс без зміни базової логіки обробки запитів та збереження результатів.

2.2 Структура програмного проєкту

Структура репозиторію побудована за модульним принципом. Кожен каталог і файл мають чітке функціональне призначення, що забезпечує зрозумілу організацію коду та полегшує подальший розвиток системи.

1. app/main.py містить маршрути, логіку запуску застосунку та базову координацію HTTP-запитів.
2. app/config.py реалізує централізоване керування конфігурацією через змінні середовища.
3. app/database.py відповідає за підключення до PostgreSQL, ініціалізацію асинхронних сесій та налаштування доступу до БД.
4. app/models/moderation.py описує ORM-модель таблиці записів модерації.
5. app/services/ містить модулі окремих інтелектуальних перевірок, мережеву взаємодію із зовнішніми API та агрегатор результатів.
6. app/i18n.py забезпечує багатомовну підтримку інтерфейсу.
7. templates/ містить HTML-шаблони сторінок, сформовані за допомогою Jinja2.
8. static/style.css містить стилі оформлення інтерфейсу та елементи візуалізації X-Ray-підсвічування.
9. Dockerfile і docker-compose.yml забезпечують контейнеризоване розгортання системи.
10. Файли конфігурації середовища містять параметри підключення до зовнішніх сервісів, СУБД та технічні налаштування застосунку.

Така структура є логічною та придатною для подальшого командного розвитку. Вона дозволяє відокремити предметну логіку від інтерфейсної частини, ізолювати модулі інтеграції із зовнішніми сервісами, а також спростити навігацію в коді. Для дипломного проєкту це особливо важливо, оскільки читач повинен мати можливість швидко співвіднести описані в записці функціональні модулі з конкретними файлами реалізації [15], [17].

Крім того, модульна структура позитивно впливає на тестованість системи. Оскільки окремі сервіси аналізу, доступ до бази даних, механізми конфігурації та формування інтерфейсу розділені між собою, це спрощує локальну перевірку

окремих частин застосунку, пошук помилок та поетапне вдосконалення програмної реалізації.

2.3 Обґрунтування технологічного стеку та взаємодії компонентів

Вибір технологій здійснювався з урахуванням прикладного характеру дипломної роботи, потреби в інтеграції AI-моделей, підтримки асинхронної обробки та зручності розгортання.

Основою серверної частини обрано FastAPI, оскільки цей фреймворк забезпечує високу продуктивність, підтримку асинхронного програмування та зручну роботу з маршрутами, формами і файлами [15]. Для запуску застосунку використовується Uvicorn [19]. Асинхронна логіка системи базується на asyncio, а для взаємодії із зовнішніми AI-сервісами застосовується HTTPX [18], [23].

Для зберігання історії перевірок використовується PostgreSQL, а доступ до даних реалізовано через SQLAlchemy з асинхронною підтримкою [20], [21]. На нижчому рівні з’єднання з базою даних забезпечує asyncpg [24]. Керування конфігурацією реалізовано через pydantic-settings, що дозволяє зберігати чутливі параметри в змінних середовища [25].

Інтелектуальна частина системи спирається на Hugging Face Inference API, який використовується для аналізу токсичності, спаму, шахрайства, транскрипції аудіо та X-Ray аналізу [12], [13], [26], [27]. Для задач модерації нецензурної лексики додатково використовується NVIDIA NIM [14]. Аналіз емоційного тону реалізовано за допомогою NLTK VADER [10], [28].

Інтерфейс побудовано з використанням Jinja2 і Tailwind CSS [16], [29], а підтримка завантаження файлів забезпечується через python-multipart [30]. Формування PDF-звітів виконується засобами ReportLab [22]. Для контейнеризованого розгортання використано Docker і Docker Compose [17].

Для наочного подання логіки функціонування системи автоматизованої модерації контенту також використано UML-діаграми, які відображають

взаємодію користувача із системою, послідовність виконання основних дій та загальний алгоритм обробки вхідних даних.

Актором системи є користувач, який взаємодіє з вебзастосунком через графічний інтерфейс. До основних прецедентів використання належать відкриття головної сторінки, введення тексту для аналізу, завантаження аудіофайлу, запис аудіо з мікрофона, запуск аналізу, перегляд результату модерації, перегляд історії перевірок, завантаження PDF-звіту та перемикання мови інтерфейсу.

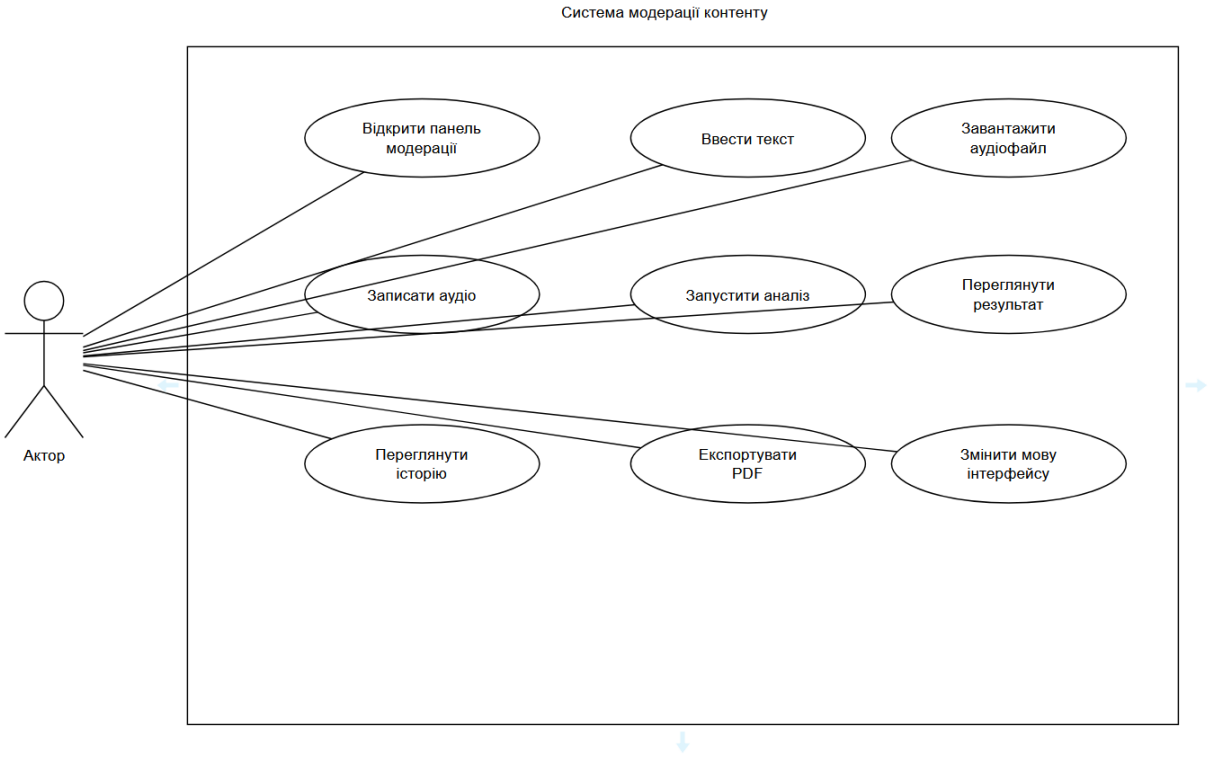

Рисунок 2.2. Use Case діаграма

Для деталізації внутрішньої логіки роботи системи доцільно також подати діаграму активності. Вона відображає послідовність етапів від моменту отримання вхідних даних до формування підсумкового результату модерації. Зокрема, на такій діаграмі можуть бути показані етапи введення або завантаження даних, перевірки коректності вхідної інформації, транскрипції аудіо, запуску модулів аналізу, агрегування часткових результатів, збереження

запису в базі даних та відображення результату користувачеві. Діаграму активності наведено на рисунку 2.3.

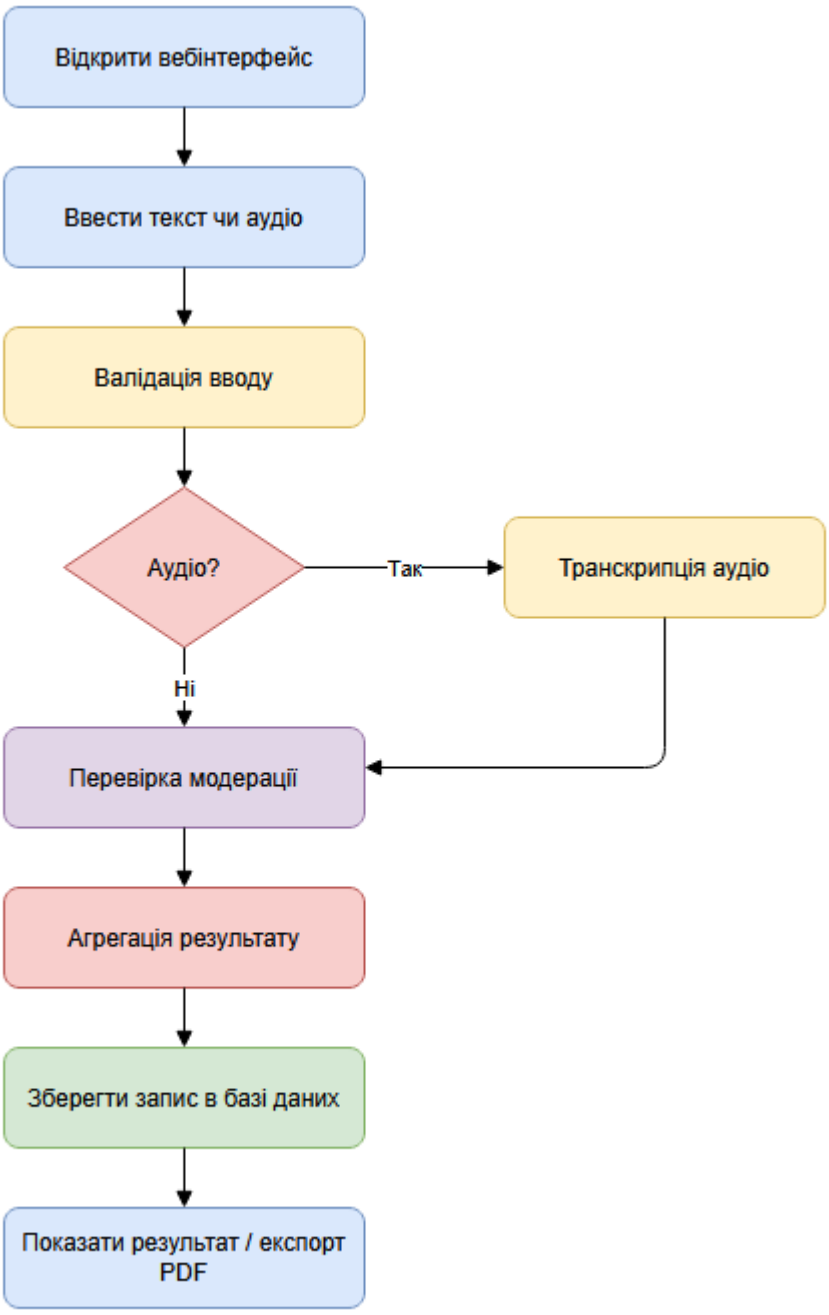

Рисунок 2.3. Процес модерації контенту

2.4 Проєктування функціональних модулів

Функціонально система складається з модулів введення тексту, завантаження аудіофайлу, транскрипції аудіо, визначення токсичності, спаму, шахрайства, нецензурної лексики, емоційного тону, X-Ray візуалізації, агрегування результатів, збереження даних і формування PDF-звіту.

Модулі введення та завантаження відповідають за приймання і первинну валідацію вхідних даних. Якщо користувач надсилає аудіо, система виконує його транскрипцію у текстову форму [13], [27], [31]. Отриманий текст передається до незалежних модулів аналізу, які оцінюють токсичність, наявність спаму, ознаки шахрайства, нецензурну лексику та емоційне забарвлення повідомлення [9], [10], [12], [14], [26], [28].

Центральним елементом є модуль агрегування, який узгоджує часткові результати, нормалізує їх і формує інтегральну оцінку ризику. На основі цієї оцінки система визначає фінальний вердикт щодо повідомлення. Додатково модуль X-Ray забезпечує пояснюваність результату, а модулі збереження та звітності відповідають за запис даних у базу та формування PDF-документа [13], [20-22].

Така модульна побудова забезпечує гнучкість, зрозумілість архітектури та можливість заміни окремих компонентів без зміни всієї системи.

2.5 Проєктування структури бази даних, конфігурації та розгортання

Основною сутністю системи є таблиця з результатами модерації. У проєкті вона представлена ORM-моделлю ModerationRecord. Запис містить як основні поля результату, так і допоміжні атрибути для деталізації перевірки.

Оскільки поточна версія системи зберігає історію в межах однієї ключової сутності, структура є компактною та зручною для демонстраційного або прототипного використання [20], [21].

Таблиця 2.1

Структура таблиці бази даних

Поля	Тип даних	Обмеження	Призначення
id	UUID	PRIMARY KEY, default=uuid4	Унікальний ідентифікатор запису
text	Text	nullable	Текст повідомлення, яке передано на аналіз.
input_type	String(10)	NOT NULL, default='text'	Тип вхідних даних: текст, аудіо або інший підтримуваний формат.
audio_language	String(10)	nullable	Мова, визначена під час розпізнавання аудіо.
spam_score	Float	nullable	Імовірність належності повідомлення до спаму.
toxicity_score	Float	nullable	Оцінка токсичності за шкалою ризику від 0 до 1.
spam_details	JSON	nullable	Деталізована відповідь сервісу виявлення спаму.
final_score	Float	NOT NULL	Інтегральна оцінка ризику, обчислена на основі окремих показників.
status	String(20)	NOT NULL	Фінальний вердикт модерації: APPROVED, FLAGGED або REJECTED.

З погляду кваліфікаційної роботи це також демонструє дотримання сучасних практик інженерії програмного забезпечення. Наявність контейнеризованої конфігурації підвищує переносимість рішення, спрощує перевірку проєкту на іншому обладнанні та зменшує ймовірність помилок, пов’язаних із локальними налаштуваннями середовища.

РОЗДІЛ 3

ПРОГРАМНА РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА ПЕРСПЕКТИВИ РОЗВИТКУ СИСТЕМИ

3.1 Реалізація серверної частини

Серверна частина системи реалізована у файлі app/main.py. У ньому створюється екземпляр застосунку FastAPI, налаштовується життєвий цикл програми, підключаються шаблони та статичні ресурси, а також описуються HTTP-маршрути для взаємодії з користувачем.

На етапі запуску система:

1. Ініціалізує структуру бази даних.
2. Налаштовує журналювання.
3. Завантажує необхідні мовні ресурси.
4. Підключає HTML-шаблони й статичні файли.

Такий підхід забезпечує централізований контроль над серверною логікою та зручне розширення застосунку в майбутньому [15], [16].

Окремої уваги заслуговує використання механізму `lifespan`, який дозволяє підготувати середовище ще до початку обробки запитів. Завдяки цьому база даних ініціалізується на старті застосунку, а необхідні мовні ресурси для аналізу тону автоматично перевіряються та, за потреби, завантажуються. Така організація покращує стабільність роботи системи та робить її запуск менш залежним від ручних дій користувача [15],[25],[28].

3.2 Реалізація маршрутів вебзастосунку

У застосунку реалізовано набір маршрутів, які покривають основні сценарії використання системи:

1. GET / – відкриття головної сторінки модерації.

2. POST /moderate – аналіз текстового контенту.

3. POST /moderate-audio – аналіз аудіоконтенту через транскрипцію.

4. GET /result/{record_id} – перегляд детального результату.

5. GET /result/{record_id}/pdf – формування PDF-звіту.

6. GET /history – перегляд історії перевірок.

7. GET /api/records – отримання історії у форматі JSON.

8. GET /health – технічна перевірка працездатності сервісу.

Наявність одночасно HTML-інтерфейсу та JSON-маршруту є корисною архітектурною рисою. Вона означає, що система вже на поточному етапі частково готова до інтеграції з іншими клієнтами, наприклад окремою адміністративною панеллю, ботом чи зовнішньою системою аналітики [15],[16]. Таким чином, навіть дипломний прототип уже закладає основу для сервісно-орієнтованого розвитку.

3.3 Реалізація модуля транскрипції аудіо

Модуль audio_service.py призначений для аналізу голосових повідомлень. Його завданням є прийняття аудіофайлу, перевірка MIME-типу та розміру, надсилання файлу до моделі Whisper через Hugging Face API та повернення текстової транскрипції [13],[27],[31].

У межах цього модуля реалізовано:

1. Перевірку підтримуваних аудіоформатів.
2. Обмеження максимального розміру файлу.
3. Захист від порожніх аудіозаписів.
4. Формування запиту до зовнішнього сервісу.
5. Отримання тексту та мови аудіозапису.

Підтримка аудіовходу суттєво розширює практичну цінність системи. У реальних умовах токсичний або шахрайський контент може надходити не лише у вигляді текстових повідомлень, а й у формі голосових записів. Саме тому

реалізація транскрипції робить систему більш універсальною та демонструє інтеграцію між кількома етапами AI-обробки: розпізнавання мовлення та подальший аналіз змісту [13], [27], [31].

3.4 Реалізація модулів аналізу контенту

Модуль hf_service.py використовує готову модель класифікації спаму, hf_toxicity_service.py визначає рівень токсичності повідомлення, fraud_service.py застосовує zero-shot підхід для перевірки тексту на відповідність сценаріям шахрайської поведінки, profanity_service.py інтегрує NVIDIA NIM для контекстно чутливого аналізу небажаної лексики, а sentiment_service.py поєднує VADER і додаткові евристики для української мови [10-12],[14],[26]. У сукупності ці модулі формують багатошаровий механізм аналізу, де кожен сервіс спеціалізується на своєму аспекті змісту. Такий розподіл дозволяє уникнути надмірної залежності від будь-якої окремої моделі та підвищує загальну якість прийняття рішення [14].

3.5 Інтерфейс користувача, історія перевірок та формування звітів

Інтерфейс розробленої системи орієнтований на просту та зрозумілу взаємодію користувача з функціями модерації контенту. Вебзастосунок надає можливість виконувати аналіз текстових повідомлень, завантажувати або записувати аудіо, переглядати результати перевірки, зберігати історію звернень і формувати підсумкові звіти. На рисунку 3.1 наведено головну сторінку системи, яка містить основні елементи для введення вхідних даних та запуску аналізу.

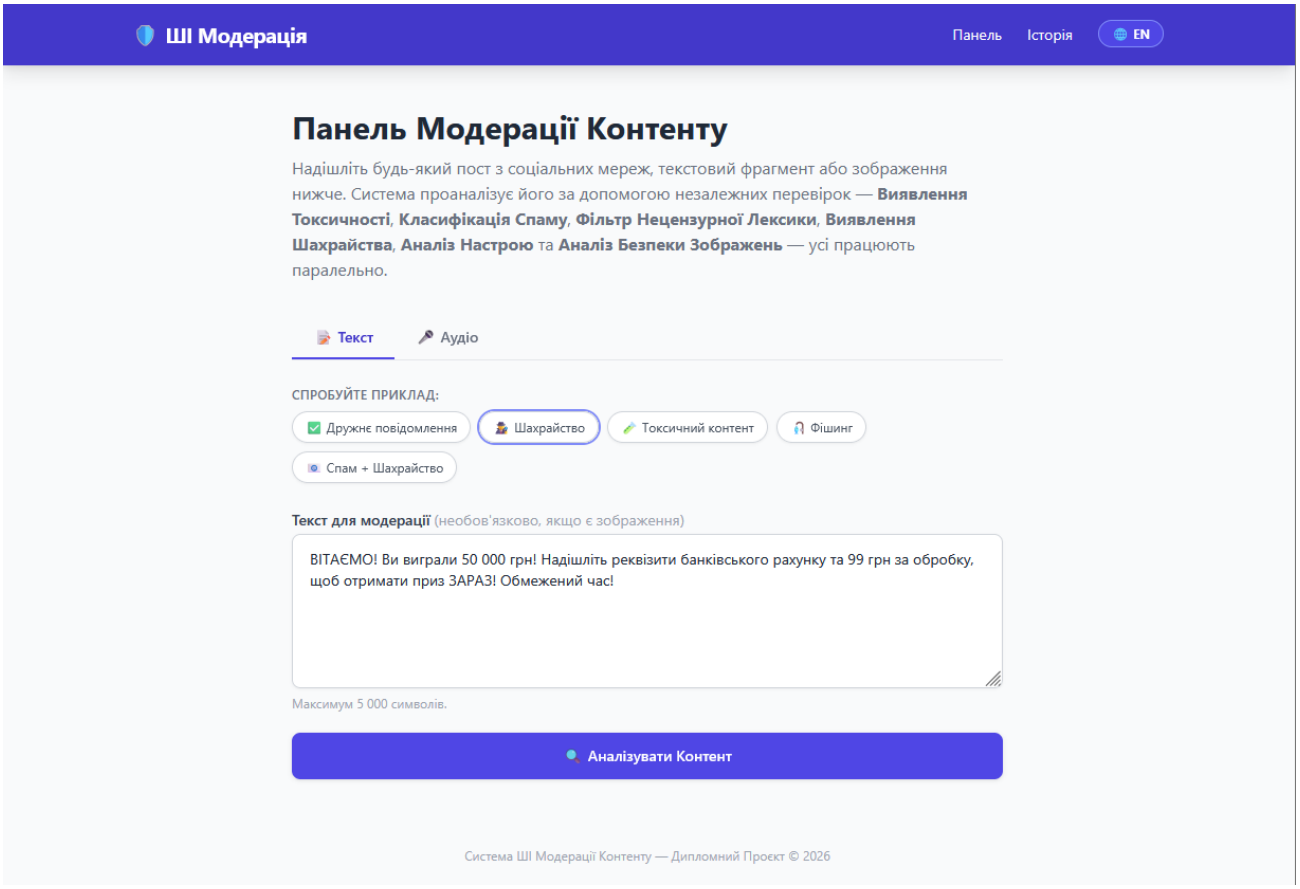

Рисунок 3.1. Головна сторінка системи модерації контенту

Ключовим модулем системи є aggregator.py, який координує виклики всіх незалежних перевірок. У ньому реалізовано асинхронний запуск сервісів через `asyncio.gather`, перерозподіл ваг у разі недоступності окремих джерел, обчислення фінального бала та формування статусу [23].

У поточній реалізації використовуються такі ваги:

1. Toxicity – 0.25
2. Spam – 0.20
3. Profanity – 0.20
4. Fraud – 0.25
5. Sentiment – 0.10

Система також застосовує критичні пороги. Якщо окреме джерело фіксує дуже високий ризик, остаточний статус може бути підвищений незалежно від середнього значення. Це дозволяє уникнути ситуації, коли небезпечне

повідомлення помилково отримує м’який вердикт лише через низькі бали інших модулів.

Інженерна цінність такого механізму полягає в тому, що він відображає логіку реальної модерації. У багатьох практичних сценаріях один сильний індикатор небезпеки має більшу вагу, ніж декілька слабких позитивних сигналів.

Після виконання аналізу користувач отримує результат, що містить підсумковий статус перевірки, загальний рівень ризику, а також деталізацію за окремими критеріями. Такий підхід дає змогу не лише побачити фінальний висновок системи, а й зрозуміти, які саме модулі вплинули на отримане рішення (див. рисунок 3.2).

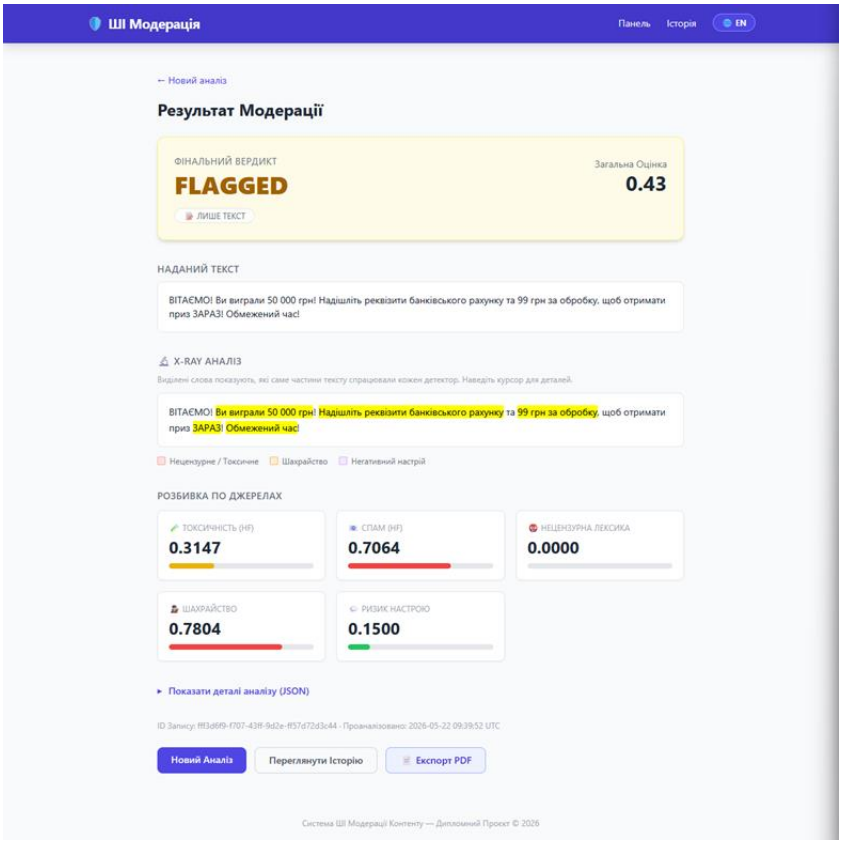

Рисунок 3.2. Сторінка результату аналізу повідомлення

Для забезпечення можливості повторного перегляду вже виконаних перевірок у системі реалізовано окрему сторінку історії. Вона дозволяє зберігати результати попередніх аналізів, що є зручним як для демонстрації роботи

системи, так і для практичного використання в середовищах, де необхідний облік модераційних рішень. На рисунку 3.3 наведено сторінку історії перевірок.

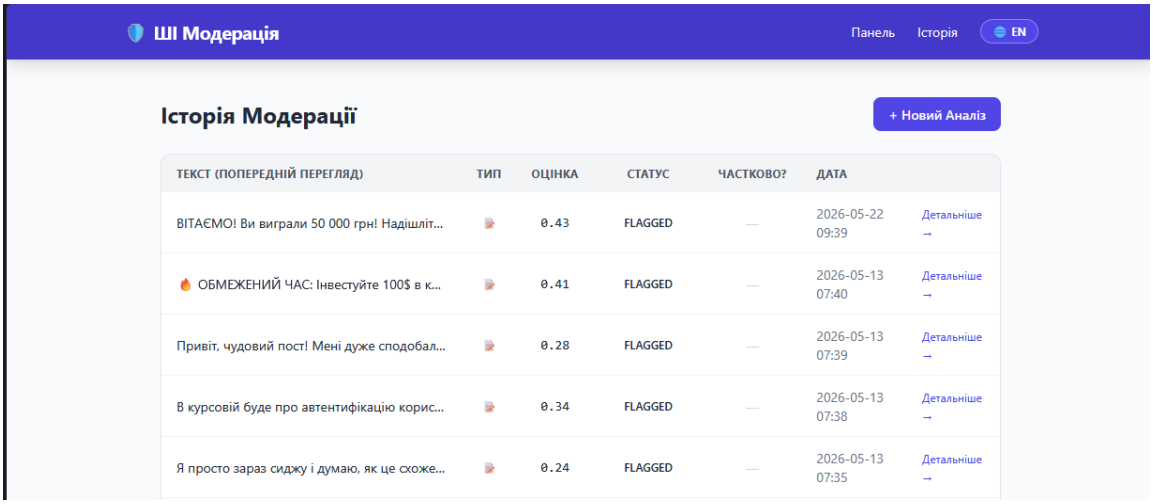

Рисунок 3.3. Сторінка історії перевірок

3.6 Тестування системи

Для перевірки працездатності розробленої системи було проведено функціональне та інтеграційне тестування. Метою тестування була оцінка коректності роботи основних функцій вебзастосунку, перевірка логіки обробки текстових і аудіоповідомлень, а також аналіз стійкості системи в умовах часткової недоступності зовнішніх сервісів.

Функціональне тестування охоплювало перевірку окремих сценаріїв використання системи. Зокрема, було перевірено аналіз нейтральних повідомлень, агресивного та образливого контенту, спам-повідомлень, шахрайських і фішингових звернень, україномовного токсичного тексту, а також аудіоповідомлень після їх транскрипції у текстовий формат. Окремо було перевірено формування PDF-звіту та збереження результатів у сторінці історії перевірок.

Під час тестування нейтральних повідомлень система повинна була формувати статус APPROVED, що підтверджувало відсутність виражених

ризикових ознак. Для повідомлень із токсичним, агресивним або нецензурним змістом очікуваним результатом був статус REJECTED, тоді як для рекламних або нав’язливих повідомлень система зазвичай формувала статус FLAGGED або REJECTED залежно від рівня виявленого ризику. У випадку повідомлень із фішинговими або шахрайськими ознаками система також повинна була визначати підвищений рівень небезпеки та відображати відповідний результат модерації.

Окрему увагу було приділено тестуванню аудіоаналізу. У цьому сценарії перевірялася коректність завантаження або запису аудіофайлу, виконання транскрипції та подальшого аналізу отриманого тексту засобами модераційних модулів. Такий підхід дав змогу переконатися, що система забезпечує єдину логіку оцінювання як для текстових, так і для аудіоповідомлень.

Результати функціонального тестування показали, що система коректно виконує основні сценарії модерації та формує підсумкові статуси APPROVED, FLAGGED і REJECTED відповідно до змісту вхідних даних. Також було підтверджено працездатність механізму збереження історії перевірок, формування PDF-звітів і відображення пояснювальної інформації щодо результату аналізу.

Інтеграційне тестування було спрямоване на перевірку узгодженої роботи всіх основних компонентів системи: FastAPI-застосунку, зовнішніх AI-сервісів, модуля агрегування, бази даних PostgreSQL та користувацького інтерфейсу. Особливу увагу приділено сценаріям, у яких один або декілька зовнішніх сервісів тимчасово недоступні. У таких випадках система не завершує роботу аварійно, а формує частковий результат, використовуючи доступні модулі аналізу та коригуючи підсумкову оцінку ризику.

Кількісне узагальнення результатів функціонального тестування наведено в таблиці 3.1.

Таблиця 3.1

Кількісні результати функціонального тестування

Група перевірок	**Кількість сценаріїв**	**Успішно**	**Неуспішно**	**Успішність %**	**Кількісний результат**
Текстовий аналіз	6	6	0	100	1 APPROVED, 2 FLAGGED, 3 REJECTED
Аудіоаналіз	2	2	0	100	1 APPROVED, 1 REJECTED
Звіти та історія	2	2	0	100	PDF-звіт сформовано; історію відображено
Відмовостійкість і валідація	2	2	0	100	Частковий результат і повідомлення про помилку сформовано
Усього	12	12	0	100	8 модераційних і 4 сервісні сценарії

Інтеграційне тестування було спрямоване на перевірку узгодженої роботи всіх основних компонентів системи: FastAPI-застосунку, зовнішніх AI-сервісів, модуля агрегування, бази даних PostgreSQL та користувацького інтерфейсу. Особливу увагу приділено сценаріям, у яких один або декілька зовнішніх сервісів тимчасово недоступні. У таких випадках система не завершує роботу аварійно, а формує частковий результат, використовуючи доступні модулі аналізу та коригуючи підсумкову оцінку ризику.

Таким чином, проведене тестування підтвердило працездатність розробленої системи, коректність реалізованої логіки аналізу цифрового контенту та її придатність до подальшого розвитку й практичного використання.

3.7 Переваги, недоліки та перспективи розвитку системи

До основних переваг системи належать:

1. багатофакторний аналіз контенту;
2. підтримка тексту та аудіо;
3. модульна архітектура;
4. пояснюваність результатів;
5. підтримка української та англійської мов;
6. наявність історії та PDF-звітів;
7. просте контейнеризоване розгортання.

До обмежень системи належать:

1. залежність від зовнішніх API;
2. змінна затримка відповіді сторонніх сервісів;
3. потреба в токенах доступу;
4. відсутність локального навчання власних моделей;
5. обмеження прототипної архітектури порівняно з промисловими рішеннями.

Попри зазначені обмеження, система є цілісною та достатньо функціональною для рівня дипломної роботи. Вона демонструє не тільки знання окремих фреймворків чи бібліотек, а й розуміння повного циклу побудови прикладного інтелектуального вебсервісу [17], [22], [25].

Подальший розвиток системи може здійснюватися в кількох напрямах:

1. додавання повноцінного аналізу зображень;
2. підтримка відеоконтенту;
3. навчання власних моделей на тематичних датасетах;
4. побудова адміністративної панелі модератора;
5. реалізація ролей користувачів і журналу аудиту;
6. розширення переліку мов;

7. додавання аналітичних дашбордів і статистики;
8. інтеграція із зовнішніми платформами через API.

Окремим перспективним напрямом є розроблення адаптивної системи налаштування ваг агрегатора. У такому випадку вагові коефіцієнти могли б коригуватися відповідно до типу платформи, мови контенту, сценарію використання або статистики помилок попередніх перевірок. Це перетворило б систему з фіксованого правила-орієнтованого рішення на більш гнучкий інструмент підтримки модерації [2] [4]. Це зробило б систему більш придатною для подальшого розвитку та практичного використання.

ВИСНОВКИ

У дипломній роботі було розглянуто актуальну науково-прикладну задачу створення інтелектуальної системи автоматизованої модерації цифрового контенту. На основі аналізу предметної області встановлено, що ефективна модерація сучасного інформаційного потоку потребує використання не одного окремого класифікатора, а комплексу взаємодоповнювальних методів і сервісів штучного інтелекту. Такий підхід дозволяє точніше виявляти різні типи небажаного контенту та формувати більш обґрунтований підсумковий результат перевірки.

У процесі виконання роботи:

1. досліджено види небажаного контенту та методи його виявлення;

2. обґрунтовано архітектуру веборієнтованої системи модерації;

3. реалізовано застосунок на основі FastAPI, PostgreSQL, SQLAlchemy, Jinja2 та Docker;

4. інтегровано модулі аналізу токсичності, спаму, шахрайства, нецензурної лексики та емоційного тону;

5. реалізовано підтримку транскрипції аудіоповідомлень;

6. побудовано механізм агрегування результатів у єдиний інтегральний показник;

7. забезпечено історію перевірок, X-Ray пояснюваність та генерацію PDF-звітів.

Реалізована система дозволяє виконувати комплексну перевірку текстового та аудіоконтенту, зберігати результати аналізу й надавати користувачеві пояснення щодо сформованої оцінки ризику. Завдяки цьому застосунок є не лише інструментом автоматичної класифікації, а й засобом підтримки прийняття рішень у процесі модерації.

Отже, поставлену мету дипломної роботи досягнуто. Розроблена система може бути використана як працездатний прототип для подальшого розширення, адаптації до реальних сценаріїв використання та інтеграції в інформаційні платформи, що потребують контролю якості та безпечності контенту.

Перспективність розробки полягає у можливості її масштабування для реальних вебплатформ, сервісів підтримки користувачів, освітніх середовищ і корпоративних інформаційних систем, де потрібен швидкий попередній аналіз вхідного контенту. Подальший розвиток системи може бути пов’язаний із розширенням набору AI-моделей, підтримкою більшої кількості мов, додаванням аналізу зображень і відеоконтенту та впровадженням точніших метрик оцінювання ризику. Крім того, перспективним напрямом є удосконалення механізмів пояснюваності результатів і налаштування правил модерації відповідно до потреб конкретної платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gillespie T. Custodians of the Internet: Platforms, Content Moderation, and the Hidden Decisions That Shape Social Media. New Haven : Yale University Press, 2018. URL: https://yalebooks.yale.edu/book/9780300261431/custodians-of-the-internet/ (дата звернення: 04.02.2026).

2. Gorwa R., Binns R., Katzenbach C. Algorithmic content moderation: Technical and political challenges in the automation of platform governance // Big Data & Society. 2020. Vol. 7, No. 1. DOI: 10.1177/2053951719897945. URL: https://doi.org/10.1177/2053951719897945 (дата звернення: 04.02.2026).

3. Dhamija R., Tygar J. D., Hearst M. Why Phishing Works // Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2006. P. 581-590. DOI: 10.1145/1124772.1124861. URL: https://doi.org/10.1145/1124772.1124861 (дата звернення: 04.02.2026).

4. Borkan D., Dixon L., Sorensen J., Thain N., Vasserman L. Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification // Companion Proceedings of The 2019 World Wide Web Conference. 2019. P. 491-500. DOI: 10.1145/3308560.3317593. URL: https://doi.org/10.1145/3308560.3317593 (дата звернення: 04.02.2026).

5. Wulczyn E., Thain N., Dixon L. Ex Machina: Personal Attacks Seen at Scale // Proceedings of the 26th International Conference on World Wide Web. 2017. P. 1391-1399. URL: https://research.google/pubs/ex-machina-personal-attacks-seen-at-scale/ (дата звернення: 05.02.2026).

6. Davidson T., Warmsley D., Macy M., Weber I. Automated Hate Speech Detection and the Problem of Offensive Language // Proceedings of the International AAAI Conference on Web and Social Media. 2017. Vol. 11, No. 1. P. 512-515. DOI: https://doi.org/10.1609/icwsm.v11i1.14955 (дата звернення: 05.02.2026).

7. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. Attention Is All You Need. 2017. URL: https://arxiv.org/abs/1706.03762 (дата звернення: 05.02.2026).

8. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of NAACL-HLT. 2019. P. 4171-4186. URL: https://aclanthology.org/N19-1423/ (дата звернення: 05.02.2026).

9. Almeida T. A., Hidalgo J. M. G., Yamakami A. Contributions to the Study of SMS Spam Filtering: New Collection and Results // Proceedings of the 11th ACM Symposium on Document Engineering. 2011. P. 259-262. DOI: 10.1609/icwsm.v11i1.14955. URL: https://doi.org/10.1145/2034691.2034742 (дата звернення: 21.02.2026).

10. Hutto C. J., Gilbert E. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text // Proceedings of the International AAAI Conference on Web and Social Media. 2014. Vol. 8, No. 1. P. 216-225. DOI: 10.1609/icwsm.v8i1.14550. URL: https://doi.org/10.1609/icwsm.v8i1.14550 (дата звернення: 21.02.2026).

11. Conneau A., Khandelwal K., Goyal N., Chaudhary V., Wenzek G., Guzman F., Grave E., Ott M., Zettlemoyer L., Stoyanov V. Unsupervised Cross-lingual Representation Learning at Scale. 2019. URL: https://arxiv.org/abs/1911.02116 (дата звернення: 01.03.2026).

12. joeddav/xlm-roberta-large-xnli : model card / Hugging Face. URL: https://huggingface.co/joeddav/xlm-roberta-large-xnli (дата звернення: 01.03.2026).

13. Hugging Face. Inference Providers Documentation. URL: https://huggingface.co/docs/inference-providers (дата звернення: 01.03.2026).

14. NVIDIA NIM for Large Language Models Documentation / NVIDIA. URL: https://docs.nvidia.com/nim/large-language-models/latest/ (дата звернення: 16.03.2026).

15. FastAPI : documentation. URL: https://fastapi.tiangolo.com/ (дата звернення: 16.03.2026).

16. Jinja Documentation. URL: https://jinja.palletsprojects.com/ (дата звернення: 16.03.2026).

17. Docker Compose / Docker Docs. URL: https://docs.docker.com/compose/ (дата звернення: 21.03.2026).

18. HTTPX Documentation. URL: https://www.python-httpx.org/ (дата звернення: 21.03.2026).

19. Uvicorn. Documentation. URL: https://www.uvicorn.org/ (дата звернення: 21.03.2026).

20. PostgreSQL Documentation / PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/current/ (дата звернення: 30.03.2026).

21. SQLAlchemy. Asynchronous I/O (asyncio) : documentation. URL: https://docs.sqlalchemy.org/20/orm/extensions/asyncio.html (дата звернення: 30.03.2026).

22. ReportLab PDF Library User Guide / ReportLab. URL: https://www.reportlab.com/docs/reportlab-userguide.pdf (дата звернення: 30.03.2026).

23. Python Standard Library. asyncio – Asynchronous I/O. URL: https://docs.python.org/3/library/asyncio.html (дата звернення: 30.03.2026).

24. asyncpg Documentation. URL: https://magicstack.github.io/asyncpg/current/ (дата звернення: 30.03.2026).

25. Pydantic Settings Documentation / Pydantic. URL: https://docs.pydantic.dev/latest/concepts/pydantic_settings/ (дата звернення: 11.04.2026).

26. mrm8488/bert-tiny-finetuned-sms-spam-detection : model card / Hugging Face. URL: https://huggingface.co/mrm8488/bert-tiny-finetuned-sms-spam-detection (дата звернення: 11.04.2026).

27. openai/whisper-large-v3-turbo : model card / Hugging Face. URL: https://huggingface.co/openai/whisper-large-v3-turbo (дата звернення: 11.04.2026).

28. NLTK : nltk.sentiment.vader module documentation. URL: https://www.nltk.org/api/nltk.sentiment.vader.html (дата звернення: 11.04.2026).

29. Tailwind CSS Documentation. URL: https://tailwindcss.com/docs/installation (дата звернення: 11.04.2026).

30. python-multipart project page. URL: https://pypi.org/project/python-multipart/ (дата звернення: 11.04.2026).

31. Radford A., Kim J. W., Xu T., Brockman G., McLeavey C., Sutskever I. Robust Speech Recognition via Large-Scale Weak Supervision. 2022. URL: https://arxiv.org/abs/2212.04356 (дата звернення: 11.04.2026).

32. Pillow Documentation. URL: https://pillow.readthedocs.io/ (дата звернення: 11.04.2026).

33. Коршун, Н. В., Бондарчук, А. П., Складанний, П. М., Соколов, В. Ю., & Крижанівська, Т. М. (2026). Моделювання та реалізація атак соціальної інженерії. Телекомунікаційні та інформаційні технології, (1), 130-139.

34. Залива, В. В., Бондарчук, А. П., & Золотухіна, О. А. (2019). Фільтрація спаму електронної пошти за допомогою машинного навчання. Зв'язок, (6), 61-66.

35. Бондарчук, А. П., & Хлиста, І. А. (2023). Вирішення проблеми часткового оновлення вмісту веб-сторінки шляхом оптимізації параметрів SPA. Інформаційні технології і автоматизація–2023, 200.

36. Машкіна, І. В., Абрамов, В. О., Носенко, Т. І., Іваніченко, Є. В., & Яскевич, В. О. (2024). Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп'ютерні науки для здобувачів вищої освіти денної форми навчання.

ДОДАТОК А

Структура програмного проєкту

```
app/
├── main.py   – модуль застосунку FastAPI, керування життєвим циклом
├── config.py          – модуль конфігурації та параметрів середовища
├── database.py        – модуль асинхронної взаємодії з базою даних
├── schemas.py         – моделі вхідних і вихідних даних
├── models/
│   └── moderation.py    – модель запису результатів модерації
└── services/
    ├── hf_service.py          – модуль визначення спаму
    ├── hf_toxicity_service.py   – модуль визначення токсичності
    ├── hf_xray_service.py       – модуль пояснення результатів перевірки
    ├── nvidia_nim_service.py    – модуль взаємодії із сервісом NVIDIA NIM
    ├── profanity_service.py     – модуль визначення нецензурної лексики
    └── aggregator.py          – модуль обчислення підсумкової оцінки та
формування рішення

templates/             – шаблони сторінок користувацького інтерфейсу
├── base.html
├── dashboard.html
├── result.html
├── history.html
├── error.html

static/                – статичні ресурси застосунку
Dockerfile             – файл складання контейнера застосунку
docker-compose.yml     – файл опису спільного запуску застосунку та бази
даних
requirements.txt       – перелік програмних залежностей
.env.example           – приклад файлу змінних середовища
README.md              – файл опису проєкту та інструкції з запуску
```

ДОДАТОК Б

Таблиця тестових сценаріїв

№	Категорія	Вхід	Очікуваний результат	Факт
1	Текст	Нейтральний текст	APPROVED	APPROVED
2	Токсичність	Образливий текст	REJECTED	REJECTED
3	Спам	Спам-ознаки	FLAGGED/REJECTED	FLAGGED
4	Фішинг	Фішинговий текст	REJECTED	REJECTED
5	Нецензурна лексика	Нецензурний текст	REJECTED	REJECTED
6	Тон	Негативний тон	FLAGGED/REJECTED	FLAGGED
7	Аудіо	Аудіо без порушень	APPROVED	APPROVED
8	Аудіо	Токсичне аудіо	REJECTED	REJECTED
9	Звіт	PDF-звіт	PDF сформовано	Звіт сформовано
10	Історія	Перегляд історії	Історія коректна	Дані відображено
11	Стійкість	AI-сервіс недоступний	Частковий результат	Частковий результат
12	Валідація	Порожнє поле	Помилка введення	Помилку показано