МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних інформаційних систем
_____ Євген СІДЕНКО
«____»_____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБПЛАТФОРМА НОВИН З ВИКОРИСТАННЯМ AI-МОДУЛЯ ДЛЯ ВИЯВЛЕННЯ НЕДОСТОВІРНОЇ ІНФОРМАЦІЇ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____ Олександр АРЕФ'ЄВ
«___»_____ 2026 р.

Керівник канд. техн. наук, доцент _____ Євген СІДЕНКО
«___»_____ 2026 р.

Миколаїв – 2026

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____» _____ 20___ р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Ареф’єв Олександр Сергійович
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Вебплатформа новин з використанням AI-модуля для виявлення недостовірної інформації».

Керівник роботи: Сіденко Євген Вікторович, завідувач кафедри інтелектуальних інформаційних систем, канд. техн. наук, доцент.
(прізвище, ім’я, по батькові, посада, науковий ступінь, вчене звання)

Затверджена наказом ЧНУ ім. Петра Могили від «25» грудня 2025 р. № 353.

2. Строк представлення кваліфікаційної роботи «___» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: вебплатформа новин з інтегрованим AI-модулем автоматичного виявлення недостовірної інформації; новинні матеріали, отримані через NewsAPI; відкриті попередньо навчені моделі RoBERTa та BERT-tiny з репозиторію HuggingFace.

4. Перелік питань, що підлягають розробці: аналіз проблеми поширення недостовірної інформації та огляд існуючих платформ верифікації новинного контенту; дослідження методів автоматичного виявлення фейків на основі трансформерних нейронних мереж та обґрунтування ансамблевого підходу; проєктування архітектури вебплатформи та структури бази даних; реалізація серверної частини на FastAPI та клієнтського застосунку на Angular 17; реалізація AI-модуля на основі зваженого ансамблю RoBERTa, BERT-tiny, евристичного аналізатора та Google Fact Check Tools API; тестування системи та аналіз результатів роботи AI-модуля.

5. Перелік графічних матеріалів: презентація.

Керівник роботи _____ Євген СІДЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Олександр АРЕФ'ЄВ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «21» грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: *«***Вебплатформа новин з використанням AI-модуля для виявлення недостовірної інформації»**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Отримання завдання на виконання КР	21.12.2025	24.12.2025	Виконано
2.	Аналіз предметної сфери та постановка задачі	25.12.2025	30.01.2026	Виконано
3.	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд публікацій з тематики виявлення фейків та аналіз існуючих платформ верифікації новинного контенту	31.01.2026	01.03.2026	Виконано
4.	Вибір та обґрунтування методів машинного навчання і технологічного стеку для вирішення поставленої задачі	02.03.2026	01.04.2026	Виконано
5.	Реалізація вебплатформи та AI-модуля з аналізом отриманих результатів	02.04.2026	24.05.2026	Виконано
6.	Перший попередній захист КР на засіданні комісії кафедри	25.05.2026	25.05.2026	Виконано
7.	Коригування роботи за результатами попереднього захисту	26.05.2026	04.06.2026	Виконано
8.	Другий попередній захист КР на засіданні комісії кафедри	05.06.2026	05.06.2026	Виконано
9.	Доробка та остаточне оформлення КР	06.06.2026	14.06.2026	Виконано
10.	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.06.2026	19.06.2026	Виконано

Керівник роботи _____ Євген СІДЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Олександр АРЕФ'ЄВ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«29» січня 2026 р.

АНОТАЦІЯ

до кваліфікаційної роботи

здобувача групи 402 ЧНУ ім. Петра Могили

Ареф'єва Олександра Сергійовича

на тему: " **ВЕБПЛАТФОРМА НОВИН З ВИКОРИСТАННЯМ AI-МОДУЛЯ ДЛЯ ВИЯВЛЕННЯ НЕДОСТОВІРНОЇ ІНФОРМАЦІЇ** "

Об'єктом роботи є процеси виявлення недостовірної інформації у новинних текстах. Предметом роботи є моделі виявлення та класифікації недостовірної інформації у новинах з інтегрованим модулем верифікації контенту на основі штучного інтелекту. Метою роботи є розробка вебплатформи новин з інтегрованим AI-модулем для автоматичного виявлення недостовірної інформації. Методи дослідження: глибоке навчання, трансформерні архітектури, ансамблеві методи класифікації, обробка природної мови, проєктування вебзастосунків.

Перший розділ містить аналіз існуючих платформ та систем виявлення недостовірної інформації та формулювання вимог до системи. У другому розділі досліджено методи виявлення недостовірної інформації, описано архітектуру AI-модуля і обґрунтовано технологічний стек. Реалізацію вебплатформи та аналіз результатів роботи AI-модуля представлено в третьому розділі. В четвертому розділі здійснено тестування системи та наведено ролі користувачів.

В розробленій вебплатформі реалізовано ансамблевий AI-модуль, який поєднує дві трансформерні моделі (RoBERTa та BERT-tiny), евристичний аналізатор, а також Google Fact Check API Tools. Реалізовано калібрування ваг моделей за категоріями. Для функціональності додано імпорт новин через NewsAPI, оновлення стрічки новин через WebSocket та адміністративний інтерфейс. Кваліфікаційна робота складається з 4 розділів, загальним обсягом 86 сторінок, 34 використаних джерел і 14 рисунків.

Ключові слова: *вебплатформа, штучний інтелект, фейкові новини, BERT, RoBERTa, ансамблевий метод, факт-чекінг, FastAPI, Angular.*

ABSTRACT

to the qualification work by the student of the group 402 of Petro Mohyla Black Sea National University

Arefiev Oleksandr

“NEWS WEB PLATFORM WITH AI MODULE FOR MISINFORMATION DETECTION”

Object of research is the processes of detecting misinformation in news texts. Subject of research is models for detecting and classifying misinformation in news with an integrated AI-based content verification module. The aim of the work is to develop a news web platform with an integrated AI module for automatic detection of misinformation. Research methods: deep learning, transformer architectures, ensemble classification methods, natural language processing, web application design.

The first chapter contains an analysis of existing platforms and misinformation detection systems and formulates the system requirements. The second chapter examines methods for detecting misinformation, describes the AI module architecture and justifies the technology stack. The implementation of the web platform and analysis of the AI module performance are presented in the third chapter. The fourth chapter covers system testing and describes user roles.

The developed web platform incorporates an ensemble AI module combining two transformer models (RoBERTa and BERT-tiny), a heuristic analyzer, and Google Fact Check API Tools. Weight calibration of models by category has been implemented. To extend system functionality, news import via NewsAPI, real-time feed updates via WebSocket, and an administrative interface have been added. The thesis consists of 4 chapters, with a total volume of 86 pages, 34 references and 14 figures.

Keywords: *web platform, artificial intelligence, fake news, BERT, RoBERTa, ensemble method, fact-checking, FastAPI, Angular.*

ЗМІСТ

ВСТУП ... 3
1 АНАЛІЗ ПРОБЛЕМИ ВИЯВЛЕННЯ НЕДОСТОВІРНОЇ ІНФОРМАЦІЇ ПОСТАНОВКА ЗАДАЧІ ... 6
1.1 Аналіз проблеми поширення недостовірної інформації в сучасному медіапросторі ... 6
1.2 Огляд існуючих платформ та інструментів верифікації новинного контенту ... 10
1.3 Постановка задачі розробки вебплатформи новин з AI-модулем виявлення фейків ... 13
Висновки до розділу 1 ... 16
2 МЕТОДИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ ВЕБПЛАТФОРМИ НОВИН З AI-МОДУЛЕМ ВИЯВЛЕННЯ НЕДОСТОВІРНОЇ ІНФОРМАЦІЇ ... 18
2.1 Методи автоматичного виявлення недостовірної інформації ... 18
2.2 Ансамблева AI-система оцінки достовірності ... 21
2.3 Архітектура вебплатформи та технологічний стек ... 24
Висновки до розділу 2 ... 28
3 РОЗРОБКА ВЕБПЛАТФОРМИ НОВИН ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ ... 29
3.1 Проєктування структури системи ... 29
3.2 Реалізація основних функціональних модулів ... 35
3.3 Верифікація реалізованих функціональних можливостей системи ... 42
3.4 Аналіз результатів роботи AI-модуля ... 47
Висновки до розділу 3 ... 52
4 ТЕСТУВАННЯ ТА ЕКСПЛУАТАЦІЯ РОЗРОБЛЕНОЇ ВЕБПЛАТФОРМИ НОВИН ... 54
4.1 Опис роботи системи для різних ролей користувачів ... 54
4.2 Тестування та оцінка роботи системи ... 59
Висновки до розділу 4 ... 63
ВИСНОВКИ ... 64
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ... 66
ДОДАТОК А Код AI-ансамблю ... 70
ДОДАТОК Б Маршрути API статей ... 73
ДОДАТОК В NgRx ефекти стрічки новин ... 79

ВСТУП

Швидкий розвиток цифрових технологій і глобальної мережі Інтернет кардинально змінив медіапростір. На даний момент основним каналом споживання новин є соціальні мережі та онлайн-платформи, які забезпечують моментальне поширення інформації серед користувачів. Через це виникає масове поширення недостовірної інформації, що є серйозною проблемою. У цифровому середовищі вона поширюється у шість разів швидше за правдиві матеріали. Для України ця проблема є загрозою в умовах триваючої збройної агресії та інформаційного протистояння.

Актуальність роботи обумовлена відсутністю платформ на ринку, що поєднують зручний новинний портал з автоматизованою верифікацією контенту в реальному часі. Існуючі сервіси перевірки фактів працюють за реактивним принципом. Це означає що спростування публікується після того, як фейк вже охопив широку аудиторію, і досягає лише 1% від її обсягу. Автоматизація процесу верифікації безпосередньо в момент публікації є актуальним завданням у галузі комп’ютерних наук.

Метою кваліфікаційної роботи є розробка вебплатформи новин з інтегрованим AI-модулем для автоматичного виявлення недостовірної інформації. Окрім цього, яка буде надавати публікацію, перегляд, верифікацію та колективне обговорення новин у реальному часі.

Об’єктом роботи є процеси виявлення недостовірної інформації у новинних текстах.

Предметом роботи є моделі виявлення та класифікації недостовірної інформації у новинах з інтегрованим модулем верифікації контенту на основі штучного інтелекту.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз проблеми поширення недостовірної інформації і оглянути існуючі аналоги;

– дослідити моделі автоматичного виявлення фейків на основі трансформерних нейронних мереж;

– обґрунтувати ансамблевий підхід, що поєднує моделі RoBERTa та BERT-tiny, евристичний аналізатор та Google Fact Check Tools API;

– спроєктувати архітектуру системи та реалізувати клієнтський застосунок на Angular 17 і серверну частину на FastAPI;

– реалізувати механізм калібрування ваг ансамблю та автоматичного імпорту новин через NewsAPI;

– провести тестування та аналіз результатів роботи AI-модуля.

Практичне значення одержаних результатів полягає у розробці вебплатформи з повною функціональністю. Загалом автоматична оцінка достовірності кожної опублікованої статті разом з формуванням рейтингу в діапазоні від 0 до 100 на основі зваженого ансамблю чотирьох незалежних компонентів. Разом з цим реалізувати механізм оновлення стрічки новин у реальному часі через WebSocket. Автоматично імпортувати актуальні новинні матеріали через NewsAPI та розробити адміністративну панель для налаштування ваг AI-компонентів і зміни ролі користувачів без перезапуску системи.

Структура кваліфікаційної роботи складається зі вступу, чотирьох основних розділів, висновків, переліку джерел посилання і додатків. В першому розділі проаналізовано предметну сферу та сформульовано постановку задачі. У другому розділі досліджено методи виявлення недостовірної інформації, описано архітектуру AI-модуля та технологічний стек. Третій розділ описує реалізацію вебплатформи та аналіз результатів роботи AI-модуля. У четвертому розділі здійснено тестування системи та наведено керівництво користувача. Загальний обсяг роботи – 86 сторінок, 14 рисунків, 11 таблиць, 34 джерела.

Розкриття факту делегування завдань генеративному ШІ. Було використано генеративний ШІ у процесі дослідження та підготовки рукопису.

Відповідно до таксономії GAIDeT (2025), наведені нижче завдання були делеговані інструментам генеративного ШІ за повного людського нагляду:

- оцінювання здійсненності та ризиків;
- генерування коду;
- оптимізація коду;
- вичитування та редагування;
- резюмування тексту;
- адаптація та коригування емоційного тону;
- оцінювання якості;
- рекомендації.

Використаний інструмент генеративного ШІ: Claude 4.7.

Повну відповідальність за фінальний рукопис несе автор.

Інструменти генеративного ШІ не зазначаються як автори та не несуть відповідальності за кінцеві результати.

Декларацію подав: Ареф’єв Олександр Сергійович

1 АНАЛІЗ ПРОБЛЕМИ ВИЯВЛЕННЯ НЕДОСТОВІРНОЇ ІНФОРМАЦІЇ ПОСТАНОВКА ЗАДАЧІ

Перший розділ присвячено дослідженню предметної сфери кваліфікаційної роботи та формуванню теоретичного підґрунтя для розробки вебплатформи новин з AI-модулем виявлення недостовірної інформації. У розділі проаналізовано проблему поширення дезінформації в сучасному цифровому медіапросторі, розглянуто існуючі платформи та інструменти верифікації новинного контенту. Також сформульовано мету, об’єкт, предмет і завдання дослідження.

1.1 Аналіз проблеми поширення недостовірної інформації в сучасному медіапросторі

Стрімкий розвиток інформаційних технологій та глобальної мережі Інтернет кардинально змінив способи отримання й поширення новин. Двадцять років тому основними джерелами інформації були друковані видання, радіо та телебачення. Сьогодні мільярди людей дізнаються про події з соціальних мереж та онлайн-платформ і ця трансформація відбулась надзвичайно швидко. Проте саме тут і виникла принципова складність – некероване поширення недостовірних, маніпулятивних та свідомо неправдивих матеріалів.

У науковій літературі недостовірна інформація розмежовується на кілька самостійних категорій. «Дезінформація» (disinformation) являє собою свідомо неправдивий контент, що поширюється з наміром ввести аудиторію в оману задля певного соціального або політичного ефекту. «Міс-інформація» (misinformation) натомість є неточним чи хибним контентом, що з’являється внаслідок помилки або некомпетентності, без умислу завдати шкоди. Категорія «мал-інформація» (malinformation) – це контент, що спирається на реальні факти, але застосовується цілеспрямовано проти конкретної особи, організації або держави [1]. У межах цієї роботи поняття «недостовірна інформація» охоплює передусім дезінформацію та міс-інформацію в новинному контексті.

Дослідження Массачусетського технологічного інституту, опубліковане у журналі Science у 2018 році, зафіксувало разючу закономірність: неправдиві новини поширюються у соціальних мережах у шість разів швидше за правдиві, охоплюючи незрівнянно ширшу аудиторію [2]. Примітно, що винуватцями виявились не боти чи платформні алгоритми – а самі користувачі. Люди охоче діляться матеріалами, що викликають сильні емоції: здивування, обурення, тривогу. Достовірність при цьому відходить на другий план. Алгоритми соціальних мереж лише посилюють цю динаміку, адже оптимізовані на залученість, а не на перевірку фактів.

Залежно від характеру маніпуляції дослідники виділяють кілька типів фейкових новин. Сфабриковані матеріали – повністю вигадані події чи факти, оформлені під реальні. Маніпулятивний контент використовує реальну інформацію, але подає її з оманливим заголовком або без необхідного контексту. Сатира й пародія формально не мають наміру вводити в оману, однак нерідко сприймаються аудиторією буквально. Найскладнішим для автоматичного виявлення залишається частково неправдивий контент – суміш реальних фактів із вигаданими або перекрученими деталями [3]. Розгорнуту таксономію типів дезінформації та методологію їх розрізнення запропоновано у роботі Tandoc та ін. [4].

Для України проблема дезінформації набуває особливої гостроти в умовах триваючої збройної агресії Російської Федерації. Інформаційна війна є невід'ємною складовою сучасних гібридних конфліктів. Ворожі наративи, маніпулятивні матеріали та пропаганда систематично впроваджуються в медіапростір з метою дезорієнтації суспільства, підриву довіри до державних інститутів та формування вигідної агресору картини світу. За даними моніторингової ініціативи «Детектор медіа», у 2022–2023 рр. кількість виявлених фейків, пов'язаних з подіями в Україні, зросла щонайменше втричі порівняно з довоєнним періодом [5]. Детальний аналіз інструментів дезінформації в умовах збройних конфліктів представлено у доповіді EUvsDisinfo [6]. Ручна верифікація з

цим темпом просто не справляється – фактчекер витрачає на один матеріал від кількох годин до кількох діб, тоді як фейк охоплює мільйони користувачів за лічені хвилини.

Проблема автоматичного виявлення фейків є активною областю досліджень у галузі комп'ютерних наук. Загалом підходи до вирішення цієї задачі класифікують за кількома ознаками. За типом вхідних даних розрізняють методи аналізу текстового змісту статті, методи аналізу соціального контексту поширення та гібридні підходи. За математичним апаратом виокремлюють традиційні методи машинного навчання (наївний байєс, метод опорних векторів, випадковий ліс) та методи глибокого навчання (LSTM, BERT, GPT-подібні архітектури). Кожен із підходів має власні переваги та обмеження, що визначають доцільність його застосування залежно від конкретних умов задачі. Систематичний огляд методів автоматичного виявлення фейків наведено у роботі Zhou та Zafarani [7].

Ключовою технологією для вирішення задачі виявлення фейків стала трансформерна архітектура, запропонована Vaswani та ін. (2017) [8], та модель BERT (Bidirectional Encoder Representations from Transformers) [9]. На відміну від рекурентних мереж, трансформери обробляють текст паралельно через механізм самоуваги, враховуючи контекст кожного слова незалежно від його позиції. BERT є двонаправленою моделлю, що враховує лівий і правий контекст одночасно. Дотренування на розмічених даних дозволяє адаптувати її до конкретної задачі й досягати високої точності навіть за обмеженого обсягу навчальних прикладів.

Окремого значення набуває економічний вимір проблеми. Виробництво фейків давно перетворилось на прибутковий бізнес, тоді як рівень медіаграмотності залишається критично низьким. Лише 26% респондентів Reuters Institute вважають, що здатні самостійно відрізнити фейк від правдивого матеріалу [10]. За таких умов технологічні рішення для автоматизованої верифікації інформації стають не опцією, а необхідністю.

Масштаб проблеми добре ілюструє дослідження Reuters Institute Digital News Report 2025, що охопило 97 055 респондентів у різних країнах. Найбільшу загрозу поширення дезінформації опитані пов’язують із соціальними мережами: Facebook назвали основним каналом 49% респондентів, TikTok – 48%, X (Twitter) – 34%. Новинні вебсайти вважають значною загрозою лише 22% опитаних, що вказує на відносно вищий рівень довіри до верифікованих медіаплатформ. Показово, що рівень занепокоєння суттєво варіюється залежно від країни: на Філіппінах Facebook вважають загрозою 68% респондентів, у Південній Африці TikTok – 56%, а у Великій Британії платформа X (Twitter) викликає занепокоєння у 61% учасників дослідження. Розподіл сприйнятої загрози за каналами поширення дезінформації [11] наведено на рис. 1.1.

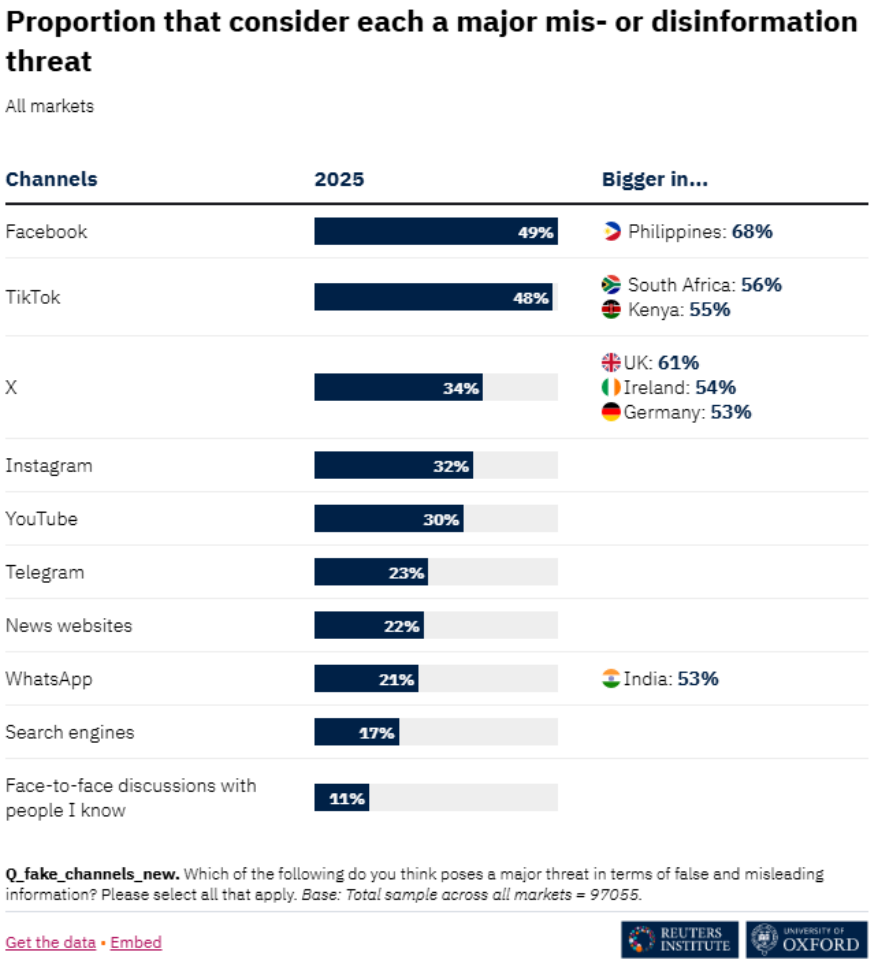

Рисунок 1.1 – Частка респондентів, що вважають канал основною загрозою поширення дезінформації

Таким чином, технологічна база для вирішення задачі автоматичного виявлення недостовірної інформації є сформованою та доступною для практичного застосування. Залишається актуальним питання інтеграції відповідного AI-модуля в повноцінну, зручну для кінцевого користувача вебплатформу новин. Саме це і є предметом даної кваліфікаційної роботи.

1.2 Огляд існуючих платформ та інструментів верифікації новинного контенту

Для визначення конкурентних переваг пропонованої розробки проведено детальний огляд існуючих платформ і сервісів у двох категоріях: спеціалізовані фактчекінгові ресурси та академічні системи автоматичної верифікації. Охоплено як функціональні можливості, так і технологічні підходи кожного з розглянутих рішень – це дозволяє уникнути повторення вже реалізованих підходів.

Snopes (snopes.com) – один із найстаріших фактчекінгових ресурсів, заснований у 1994 році. Верифікація тут побудована виключно на ручній редакційній перевірці. Автоматизованих засобів аналізу тексту немає, API для інтеграції не надається. Охоплення обмежене англомовною аудиторією, а масштабованість платформи впирається в пропускну здатність редакції.

FactCheck.org – некомерційний проєкт Університету Пенсільванії (з 2003 року), що спеціалізується на перевірці заяв американських політиків. Відзначається глибоким аналітичним опрацюванням, проте верифікація повністю ручна. Публікація користувацького контенту відсутня, API для інтеграції не надається.

Серед усіх розглянутих аналогів саме Full Fact (fullfact.org) є піонером у застосуванні автоматизованих інструментів фактчекінгу. Британська незалежна організація, заснована у 2010 році, розробила власні NLP-інструменти для моніторингу парламентських дебатів та онлайн-публікацій, частина яких відкрита для використання. Проте функціонує вона як дослідницький і аналітичний центр – не як новинна платформа. Перегляд і публікація новин кінцевим користувачем не підтримуються, а автоматизоване виявлення фейків залишається

внутрішнім редакційним інструментом. Технічний опис підходів організації до автоматизації фактчекінгу наведено у відповідному звіті [12].

PolitiFact (politifact.com) здобув Пулітцерівську премію у 2009 році і нині входить до складу Poynter Institute. Широку впізнаваність сервісу забезпечила власна шкала «Truth-O-Meter» із шістьма градаціями достовірності: True, Mostly True, Half True, Mostly False, False, Pants on Fire. Градуйована оцінка виявилась помітно інформативнішою за бінарну і здобула визнання серед аудиторії. Разом із тим сервіс цілком спирається на ручну роботу журналістів і не підтримує жодних мов, окрім англійської.

Варто також зазначити, що більшість розглянутих фактчекінгових платформ має спільне концептуальне обмеження – вони працюють за реактивним принципом: спочатку фейк поширюється і набирає аудиторію, і лише потім журналісти приступають до його перевірки та спростування. На момент публікації спростування первинний фейк вже міг охопити мільйони читачів, тоді як спростування прочитає значно менше людей – за даними досліджень, спростування фейку отримує у середньому лише 1% від охоплення первинного матеріалу [13]. Це підкреслює важливість проактивного підходу – автоматичної перевірки матеріалу безпосередньо в момент публікації, ще до його масового поширення, що і реалізує розроблювана у даній роботі система.

На рівні академічних розробок слід відзначити кілька значущих проєктів. Система ClaimBuster, розроблена дослідниками Університету Техасу в Арлінгтоні, є одним із перших інструментів автоматичного виявлення перевіряємих тверджень у тексті [14]. Вона базується на моделі машинного навчання, навченій розрізняти фактичні твердження від суджень та питань, і надає публічний API [15]. Проєкт FEVER (Fact Extraction and VERification) запропонував стандартизований набір даних із 185 тисяч тверджень та відповідну систему верифікації, що стала стандартним бенчмарком у галузі [16]. Серед інших значущих академічних робіт виокремлюється система DeClarE, що застосовує механізм уваги для пояснення результатів верифікації [17]. Серед вітчизняних ініціатив виокремлюється

«Стоп Фейк» (stopfake.org) – ресурс, заснований у 2014 році журналістами НаУКМА, що здійснює ручну перевірку матеріалів щодо подій в Україні, однак без жодних засобів автоматизації.

Підсумовуючи огляд, можна виокремити три спільні недоліки усіх розглянутих аналогів. По-перше, відсутність поєднання функцій новинної платформи та автоматизованого фактчекінгу в одному продукті. По-друге, більшість систем повністю покладаються на ручну редакційну роботу, що унеможливлює перевірку всього обсягу публікованого контенту в реальному часі. По-третє, жодне з розглянутих рішень не забезпечує прозорості оцінювання – користувач отримує лише фінальний вердикт без пояснення, на основі яких критеріїв він був сформований. Усі описані характеристики зведено до порівняльної табл. 1.1.

Таблиця 1.1 – Порівняльний аналіз існуючих аналогів

Критерій	**Snopes**	**FactCheck.org**	**Full Fact**	**PolitiFact**	**Пропонована розробка**
Автоматична NLP-перевірка тексту	Ні	Ні	Частково	Ні	Так
Вбудований AI-модуль виявлення фейків	Ні	Ні	Ні	Ні	Так
Повноцінний новинний портал	Ні	Ні	Ні	Ні	Так
Публікація матеріалів користувачами	Ні	Ні	Ні	Ні	Так
Рейтинг достовірності статті	Ні	Ні	Частково	Так	Так
Персоналізована стрічка новин	Ні	Ні	Ні	Ні	Так
Відкритий вихідний код	Ні	Ні	Так	Ні	Так
Повнотекстовий пошук по статтях	Ні	Так	Так	Так	Так
Крос-перевірка між джерелами	Ні	Ні	Ні	Ні	Так
Система реакцій	Ні	Ні	Ні	Ні	Так

Кінець таблиці 1.1

Критерій	Snopes	FactCheck.org	Full Fact	PolitiFact	Пропонована розробка
Автоматичний імпорт новин	Ні	Ні	Ні	Ні	Так
Калібрування ваг AI-моделей	Ні	Ні	Ні	Ні	Так
Real-time оновлення стрічки	Ні	Ні	Ні	Ні	Так

Дані табл. 1.1 підтверджують, що пропонована розробка є єдиним рішенням, що поєднує повноцінний новинний портал із вбудованим AI-модулем верифікації контенту, автоматичним імпортом новин через зовнішній API, системою взаємодії користувачів та прозорим поясненням результатів аналізу з розбивкою по компонентах. Важливою відмінністю від усіх розглянутих аналогів є наявність механізму ручного калібрування ваг AI-ансамблю адміністратором – можливість, відсутня у жодному з відомих відкритих рішень. Це обґрунтовує актуальність та практичну цінність даної кваліфікаційної роботи.

1.3 Постановка задачі розробки вебплатформи новин з AI-модулем виявлення фейків

На основі проведеного аналізу предметної сфери та існуючих аналогів сформульовано основні характеристики задачі, що вирішується в даній кваліфікаційній роботі, та визначено вимоги до розроблюваної системи.

Актуальність роботи – масове поширення дезінформації в умовах цифрового медіапростору становить серйозну загрозу для суспільства, зокрема в контексті активного використання інформаційних маніпуляцій як інструменту гібридної війни. Аналіз існуючих рішень показав, що на ринку відсутня платформа, яка поєднувала б зручний новинний портал із повноцінним автоматизованим модулем верифікації контенту в реальному часі, прозорістю результатів аналізу та механізмами залучення користувачів. Розробка системи, здатної автоматично оцінювати достовірність новинних матеріалів і при цьому надавати зручний

інтерфейс для кінцевого споживача, є актуальним науково-практичним завданням у галузі комп'ютерних наук.

Мета роботи – є розробка вебплатформи новин з інтегрованим AI-модулем для автоматичного виявлення недостовірної інформації, що забезпечує публікацію, перегляд, верифікацію та колективне обговорення новинних матеріалів у реальному часі.

Об'єкт дослідження – процеси виявлення недостовірної інформації у новинних текстах.

Предмет дослідження – моделі виявлення та класифікації недостовірної інформації у новинах з інтегрованим модулем верифікації контенту на основі штучного інтелекту.

Функціональні вимоги охоплюють чотири рівні, що відповідають різним категоріям учасників системи.

Базовий рівень – звичайний користувач. Після реєстрації та автентифікації [18] він отримує доступ до стрічки новин із фільтрацією за категоріями і пошуком, а також до оновлень у реальному часі через WebSocket. Кожна стаття відображається з рейтингом достовірності та розбивкою балів по компонентах AI-ансамблю. Користувач може публікувати власні матеріали із завантаженням зображень, виставляти реакції та зберігати матеріали до закладок. Управління профілем включає налаштування аватара, псевдоніма та відстеження частки корисних реакцій.

Редактор наділений усіма правами звичайного користувача з однією суттєвою відмінністю – його матеріали публікуються з пріоритетом, без очікування модераційної перевірки.

Адміністратор має найширші повноваження. До них належать управління користувачами та їх ролями, налаштування ваг компонентів AI-ансамблю в розрізі категорій із можливістю застосування змін до вже опублікованих статей без повторного запуску моделей. Окремо доступні моніторинг стану сервісів і черги аналізу, перегляд журналу AI-аналізів та аналітична панель статистики платформи.

AI-модуль автоматично аналізує кожну опубліковану та імпортовану статтю. За результатами формується числовий рейтинг достовірності від 0 до 100 на основі зваженого ансамблю чотирьох компонентів: BERT-1, BERT-2, евристичного аналізатора та FactCheck. Бали кожного компонента зберігаються окремо для подальшого відображення і калібрування. Аналіз виконується асинхронно і не блокує процес публікації. Ваги ансамблю є категоріально-специфічними і підтримують динамічне оновлення.

Автоматичне наповнення контентом реалізовано через інтеграцію з NewsAPI. Сервіс імпортує актуальні новинні матеріали із визначених категорій, автоматично робить категоризацію та зберігає їх. Одразу після імпорту стаття потрапляє до черги AI-аналізу, а підключені клієнти отримують сповіщення через WebSocket у реальному часі.

Нефункціональні вимоги визначають технічні стандарти, яким має відповідати система. Час відповіді REST API для стандартних запитів не повинен перевищувати 500 мс, а платформа має витримувати одночасну роботу щонайменше 100 користувачів без помітної деградації продуктивності. Безпека забезпечується через JWT-автентифікацію, хешування паролів bcrypt та валідацію вхідних даних на рівні API. Клієнтський застосунок має коректно працювати в основних браузерах – Chrome, Firefox, Edge і Safari. База даних розгортається у хмарі з обов’язковою підтримкою SSL-з’єднань, а схема оновлюється автоматично при запуску сервісу, без потреби в ручних міграціях. Виконання цих вимог забезпечує відповідність системи стандартам сучасних вебзастосунків та готовність до практичного використання.

Для досягнення поставленої мети визначено такі завдання:

– провести аналіз проблеми поширення недостовірної інформації в сучасному медіапросторі та дослідити існуючі підходи до її вирішення, здійснити огляд та порівняльний аналіз існуючих фактчекінгових платформ і інструментів верифікації новинного контенту;

– визначити та обґрунтувати вибір методів машинного навчання та NLP для автоматичної класифікації достовірності новинних матеріалів, зокрема ансамблевого підходу на основі двох BERT-моделей, евристичного аналізатора та зовнішнього API фактчекінгу;

– спроєктувати архітектуру системи, розробити клієнтський застосунок та серверну частину, реалізувати інтеграцію з NewsAPI для автоматичного імпорту новин, розробити та інтегрувати AI-ансамбль виявлення недостовірної інформації на основі двох трансформерних моделей (RoBERTa та BERT-tiny), евристичного аналізатора структури тексту та Google Fact Check Tools API з підтримкою категоріально-специфічного калібрування ваг;

– провести тестування розробленої системи та оцінити якість роботи AI-модуля.

Висновки до розділу 1

У першому розділі досліджено предметну сферу кваліфікаційної роботи та сформовано теоретичне підґрунтя для подальшої розробки. Розглянуто феномен недостовірної інформації в сучасному цифровому медіапросторі: проаналізовано класифікацію трьох типів дезінформації (disinformation, misinformation, malinformation), закономірності її поширення в соціальних мережах та економічні стимули виробництва фейкового контенту. Окремо встановлено особливу гостроту проблеми для України в умовах інформаційного протистояння. Серед технологічних підходів до автоматичного виявлення фейків трансформерна архітектура та моделі BERT-класу визначені як найперспективніший напрям для задачі класифікації достовірності тексту.

Проведено огляд і порівняльний аналіз п’яти провідних аналогів – Snopes, FactCheck.org, Full Fact, PolitiFact та Stop Fake, – а також академічних систем ClaimBuster і FEVER. Спільним концептуальним обмеженням усіх розглянутих рішень виявився реактивний підхід до верифікації: спростування з’являється значно пізніше за первинний фейк і охоплює набагато вужчу аудиторію. Жоден із

аналогів не поєднує повноцінний новинний портал із вбудованим AI-модулем проактивної верифікації, системою реакцій та коментарів, автоматичним імпортом новин і підтримкою україномовного середовища. Результати порівняльного аналізу зведено у табл. 1.1 за тринадцятьма критеріями.

Сформульовано мету, об’єкт та предмет дослідження. Функціональні вимоги визначено на чотирьох рівнях: звичайний користувач, редактор, адміністратор та AI-модуль. Окремо описано вимоги до підсистеми автоматичного наповнення контентом через NewsAPI. Нефункціональні вимоги охоплюють продуктивність, безпеку, хмарне розгортання та кросплатформну сумісність. Поставлено десять конкретних завдань, виконання яких забезпечує досягнення мети кваліфікаційної роботи.

Результати першого розділу створюють повноцінне теоретичне підґрунтя для подальшої розробки системи. Проведений аналіз підтвердив наявність незайнятої ніші на ринку – платформи, що поєднує зручний новинний портал із проактивною автоматичною верифікацією контенту засобами штучного інтелекту. Виявлені недоліки існуючих рішень безпосередньо визначили вимоги до розроблюваної системи та обґрунтували доцільність ансамблевого підходу до оцінки достовірності, що описується у другому розділі.

2 МЕТОДИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ ВЕБПЛАТФОРМИ НОВИН З AI-МОДУЛЕМ ВИЯВЛЕННЯ НЕДОСТОВІРНОЇ ІНФОРМАЦІЇ

Для обґрунтування методів, технологій та архітектурних рішень, що складають основу розроблюваної системи, у даному розділі розглянуто підходи до автоматичної класифікації достовірності новинних матеріалів із застосуванням методів машинного навчання та обробки природної мови, визначено технологічний стек усіх компонентів вебплатформи, а також описано архітектуру взаємодії між ними.

2.1 Методи автоматичного виявлення недостовірної інформації

Проблема автоматичного виявлення фейкових новин є активним напрямком досліджень у галузі обробки природної мови (NLP). Аналіз літератури дозволяє виокремити три основні класи підходів: методи на основі ознак контенту, методи глибокого навчання та ансамблеві методи.

Методи глибокого навчання на основі трансформерної архітектури є сучасним стандартом у NLP, оскільки механізм уваги дозволяє моделі враховувати контекст слова незалежно від його позиції у тексті.

BERT (Bidirectional Encoder Representations from Transformers), запропонований Devlin et al. (2018), є двонаправленою трансформерною моделлю, що навчається одночасно враховувати лівий і правий контекст. Попереднє навчання відбувається на двох задачах: маскування токенів (Masked Language Modeling) та передбачення наступного речення. Це дозволяє моделі набути глибокого розуміння мови до дообчання на конкретній задачі.

RoBERTa (Robustly Optimized BERT Pretraining Approach), Liu et al. (2019) [20], є покращеною версією BERT з оптимізованими гіперпараметрами навчання: більший корпус даних, більші batch-розміри, довші послідовності, виключення задачі Next Sentence Prediction. Ці зміни дали суттєве покращення на більшості бенчмарків NLP порівняно з оригінальним BERT.

Для оцінки якості AI-модуля застосовуються стандартні метрики бінарної класифікації. Ключовою з них є F1-міра – гармонічне середнє між точністю (Precision) та повнотою (Recall), що обчислюється за формулою (2.1):

$$F1 = 2 \times (Precision \times Recall) / (Precision + Recall), \quad (2.1)$$

де Precision – частка правильно класифікованих фейків серед усіх матеріалів, позначених як фейки;

Recall – частка правильно виявлених фейків серед усіх фактичних фейків у тестовій вибірці.

Цільовим показником якості AI-модуля є F1-міра не нижче 0,80 на тестовій вибірці. Використання F1-міри замість простої точності обґрунтоване нерівномірним розподілом класів у реальних наборах даних: на платформах новин частка недостовірного контенту значно нижча за частку достовірного, тому висока точність може досягатися тривіальним класифікатором, який відносить усе до більшого класу.

Ансамблеві методи об’єднують прогнози кількох незалежних моделей, що підвищує стійкість системи та знижує дисперсію помилок. Перевага ансамблю над окремою моделлю полягає в тому, що різні компоненти мають різні профілі помилок – там, де одна модель помиляється, інша може давати правильний результат. Вагова комбінація дозволяє також враховувати надійність кожного компонента окремо для різних тематичних категорій контенту. Підтверджено в роботі Sagi [21] та Rokach [22], що ансамблі стабільно перевершують окремі класифікатори на задачах класифікації текстів.

У даній роботі обрано саме ансамблевий підхід, що поєднує нейромережеву класифікацію на основі двох трансформерних моделей, евристичний аналіз тексту та зовнішню перевірку фактів. Таке поєднання забезпечує вищу точність порівняно з будь-яким окремим компонентом, а також стійкість при недоступності зовнішніх API. При відборі компонентів ансамблю враховувалися такі критерії: наявність попереднього навчання на тематично близьких датасетах (верифікація фактів,

виявлення фейків), доступність у вигляді готових моделей через репозиторій HuggingFace, прийнятний час інференсу для використання в умовах реального часу, а також відмінність архітектур і навчальних даних між компонентами – остання умова є ключовою для ефективного ансамблювання, оскільки знижує кореляцію помилок між моделями. Зовнішні API-компоненти (Google Fact Check Tools) відбиралися за критерієм наявності верифікованої бази фактів від визнаних міжнародних організацій та публічного безкоштовного доступу. Порівняльну характеристику усіх компонентів AI-модуля за роллю, обґрунтуванням вибору та інструментами реалізації наведено в табл. 2.1.

Таблиця 2.1 – Методи та компоненти AI-модуля виявлення недостовірної інформації

Метод / Компонент	**Роль у системі**	**Обґрунтування вибору**	**Інструменти реалізації**
RoBERTa	Основний класифікатор достовірності статті на основі глибокого семантичного аналізу тексту	Двонаправлений контекст, висока точність на задачах верифікації фактів, попередньо навчена модель	HuggingFace Transformers, PyTorch
BERT-tiny	Допоміжний класифікатор; компліментарний до RoBERTa за профілем помилок	Легка модель з мінімальним часом інференсу; різні навчальні дані знижують кореляцію помилок в ансамблі	HuggingFace Transformers, PyTorch
Евристичний аналізатор	Виявлення текстових маркерів маніпуляції: сенсаційна лексика, капслок, знаки оклику, ненадійні домени	Не потребує обчислювальних ресурсів; інтерпретований; доповнює нейромережеві компоненти на явно маніпулятивних текстах	Python (re, словники)

Кінець таблиці 2.1

Метод / Компонент	**Роль у системі**	**Обґрунтування вибору**	**Інструменти реалізації**
Google Fact Check Tools API	Пошук існуючих верифікованих фактчеків за ключовими твердженнями статті	База перевірених фактів від провідних міжнародних організацій (AFP, PolitiFact, Full Fact та ін.)	REST API (HTTPS / JSON)
Зважений ансамбль	Агрегація результатів чотирьох компонентів у єдиний числовий рейтинг	Підвищення стійкості та точності оцінки порівняно з використанням одного класифікатора	Python

Представлені в табл. компоненти є взаємодоповнювальними: нейромережеві моделі забезпечують глибокий семантичний аналіз тексту, евристичний модуль виявляє явні ознаки маніпуляції без обчислювальних витрат, а Google Fact Check Tools API вносить у систему верифіковані дані від міжнародних організацій. Об’єднання чотирьох незалежних компонентів у зважений ансамбль дозволяє компенсувати слабкі сторони кожного з них окремо та підвищити загальну стійкість системи до різних типів дезінформації.

2.2 Ансамблева AI-система оцінки достовірності

Розроблена AI-система складається з чотирьох незалежних компонентів, результати яких об’єднуються у фінальний рейтинг достовірності.

Модель BERT-1 (RoBERTa). Основним класифікатором є модель valurank/distilroberta-base-mnli-FEVER-climate-3way – дистильована версія RoBERTa, дообучена на датасетах FEVER та MNLI для класифікації тверджень за трьома класами: підтримано, спростовано, нейтрально. Датасет FEVER (Fact Extraction and VERification) містить понад 185 000 тверджень [23], верифікованих за статтями Вікіпедії, що робить модель особливо придатною для задачі перевірки фактів у новинному контенті. Процес інференсу виконується асинхронно через asyncio.run_in_executor [24], щоб не блокувати event loop серверу під час тривалих

обчислень. Текст обрізається до 512 токенів відповідно до архітектурних обмежень моделі.

До логітів застосовується температурне масштабування для калібрування розподілу ймовірностей за формулою (2.2):

$$P(\text{клас}) = softmax(logits / T), \quad (2.2)$$

де T – температурний параметр масштабування.

При $T < 1$ розподіл стає більш «впевненим» (пікоподібним), при $T > 1$ – більш рівномірним. Значення T = 4,0 обрано емпірично для уникнення надмірно впевнених передбачень при неоднозначних або мультитематичних текстах. Вихідне значення BERT-1 – ймовірність класу «реальна новина» у відсотках (0–100).

Модель BERT-2 (BERT-tiny). Допоміжним класифікатором є модель mrm8488/bert-tiny-finetuned-fake-news-detection – компактна двошарова версія BERT із прихованим розміром 128, навчена безпосередньо на задачі виявлення фейкових новин. Незважаючи на значно менший розмір порівняно з дистильованою RoBERTa, ця модель компліментарна до неї: відмінності в архітектурі (кількість шарів, розмір прихованого стану) та навчальних даних забезпечують відмінні профілі помилок, що і є ключовою умовою ефективного ансамблювання. Температурне масштабування T = 3,0. Час інференсу BERT-tiny суттєво нижчий, ніж у RoBERTa, що знижує загальну затримку системи.

Евристичний аналізатор. Модуль реалізує набір детермінованих перевірок без нейромережевих обчислень: наявність сенсаційної лексики у заголовку та тексті («ШОКУЮЧЕ», «BREAKING», «ТЕРМІНОВО», «НЕЙМОВІРНО»), надмірне використання великих літер (частка слів у верхньому регістрі вище порогового значення), багаторазові знаки оклику, відповідність домену джерела списку ненадійних ресурсів, словники конспірологічних та маніпулятивних маркерів. Кожна спрацьована перевірка зменшує базовий бал, результат

нормується до діапазону 0–100. Перевага модуля – повна інтерпретованість: для кожної статті система може пояснити, які саме ознаки вплинули на оцінку.

Google Fact Check Tools API. Зовнішній сервіс дозволяє перевірити, чи аналізувалися ключові твердження статті спеціалізованими організаціями-фактчекерами. API повертає список знайдених перевірок із вербальними оцінками («true», «false», «mostly true», «misleading» тощо), які перекладаються у числову шкалу 0–100 за заздалегідь визначеним відображенням. За відсутності відповідних перевірок компонент повертає нейтральне значення 50, не впливаючи на результат ансамблю; за відсутності API-ключа його вага у фінальній формулі прирівнюється до нуля з перерозподілом на решту компонентів. [25]

Фінальний рейтинг достовірності R обчислюється як зважена сума компонентів за формулою (2.3):

$$R = w_1 \cdot bert_1 + w_2 \cdot bert_2 + w_3 \cdot heur + w_4 \cdot fact, \quad (2.3)$$

де w_1, w_2, w_3, w_4 – ваги компонентів, причому $w_1 + w_2 + w_3 + w_4 = 1$;

$bert_1$, $bert_2$, heur, fact – оцінки відповідних компонентів у діапазоні [0; 100].

Ваги за замовчуванням: $w_1 = 0{,}40$; $w_2 = 0{,}25$; $w_3 = 0{,}20$; $w_4 = 0{,}15$. Такий розподіл відображає ієрархію надійності компонентів: основна нейромережева модель має найвищу вагу, зовнішній API – найнижчу через можливу недоступність.

Калібрування за категорією. Система підтримує незалежне налаштування ваг для різних тематичних категорій новин (politics, science, entertainment, sports тощо). Налаштування зберігаються в таблиці calibration_config бази даних і редагуються через адміністративний інтерфейс без перезапуску серверу. Якщо для категорії статті конфігурація відсутня – застосовуються ваги за замовчуванням. Механізм калібрування вирішує проблему зміщення домену: модель, навчена переважно на англомовному політичному контенті, може систематично помилятися на науковому або розважальному матеріалі; коригування ваг компенсує цей ефект без повторного навчання моделей.

2.3 Архітектура вебплатформи та технологічний стек

Вебплатформа побудована за дворівневою архітектурою: клієнтський Angular SPA та серверний FastAPI-застосунок. Взаємодія між ними здійснюється через HTTPS REST API та WebSocket-протокол. Схема архітектури зображена на рис. 2.1.

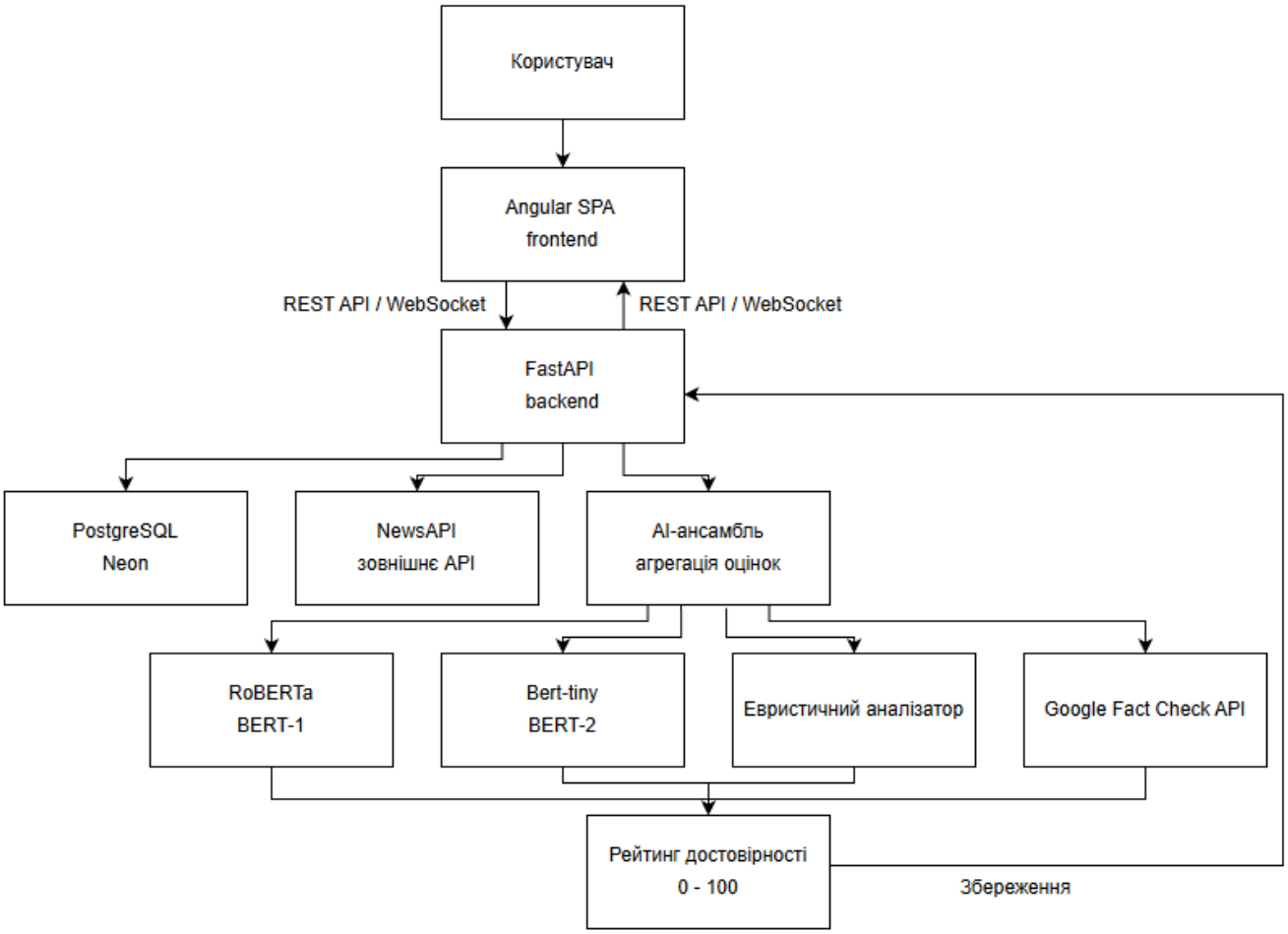

Рисунок 2.1 – Архітектура вебплатворми

Серверна частина (FastAPI / Python). FastAPI – асинхронний веб-фреймворк для Python, побудований на Starlette та Pydantic. Вибір FastAPI обґрунтований низкою технічних переваг над альтернативними рішеннями. Порівняльний аналіз серверних фреймворків наведено в табл. 2.2.

Таблиця 2.2 – Порівняльний аналіз серверних фреймворків Python

Критерій	**Django**	**Flask**	**FastAPI**
Нативна асинхронність	Часткова	Відсутня	Повна (async/await)
Автоматична документація API	Стороннє рішення	Стороннє рішення	Вбудована (Swagger UI)
Валідація вхідних даних	Форми Django	Стороннє рішення	Вбудована (Pydantic)
Підтримка WebSocket	Стороннє рішення	Стороннє рішення	Вбудована
Продуктивність (req/s)	Низька	Середня	Висока (рівень Node.js)
Обраний	Ні	Ні	Так

Нативна підтримка async/await є критичною вимогою для даного проєкту: серверний застосунок одночасно обробляє HTTP-запити, підтримує WebSocket-з'єднання та виконує тривалі AI-обчислення в окремих потоках виконавця. Синхронний фреймворк у такому сценарії блокував би обробку нових запитів на час AI-інференсу.

Аутентифікація базується на JSON Web Tokens (JWT). При успішному вході користувач отримує підписаний токен, що містить ідентифікатор користувача та його роль. Токен передається у заголовку Authorization: Bearer <token> при кожному захищеному запиті. Паролі зберігаються у вигляді bcrypt-хешів – алгоритму, що є стандартом індустрії завдяки вбудованому механізму солення та адаптивній складності хешування [26]. Система ролей включає три рівні: user (читання, реакції, закладки, коментарі), editor (публікація та редагування статей), admin (повний доступ, управління користувачами, налаштування ваг AI). Перевірка ролей реалізована через FastAPI Dependency Injection, що дозволяє декларативно захищати маршрути без дублювання логіки.

Серверний застосунок організований за модульним принципом: кожна функціональна область (AI-модулі, статті, аутентифікація, адміністрування,

калібрування, позначки) винесена в окремий пакет із власним роутером. Для роботи з базою даних використовується SQLAlchemy 2.x з асинхронним драйвером asyncpg. Кожен HTTP-запит отримує власну ізольовану сесію через Dependency Injection, що гарантує коректну обробку транзакцій та автоматичне звільнення ресурсів після завершення запиту.

База даних (PostgreSQL / Neon). Основною СКБД обрано [27] PostgreSQL завдяки повній підтримці ACID-транзакцій, розвиненій екосистемі Python-бібліотек та можливості складних реляційних запитів з JOIN та агрегацією. Хмарний сервіс Neon надає serverless [28] PostgreSQL зі стандартним протоколом підключення, що забезпечує повну сумісність з asyncpg та усуває необхідність самостійного адміністрування сервера бази даних. Схема бази даних включає сім таблиць: users, articles, ai_scores, reactions, bookmarks, comments, calibration_config. Відношення між articles та ai_scores є «один-до-одного» – кожна стаття має щонайбільше один запис AI-аналізу. Відношення реакцій, закладок та коментарів до статей та користувачів реалізовані як «багато-до-одного» з відповідними зовнішніми ключами та індексами для прискорення вибірок.

Клієнтська частина. Angular 17 – комплексний фронтенд-фреймворк із вбудованими рішеннями для маршрутизації, HTTP-взаємодії та реактивних форм [29]. На відміну від бібліотек (React, Vue), Angular є повноцінним фреймворком із чіткими архітектурними конвенціями, що важливо для підтримуваності проєкту. Порівняльний аналіз фронтенд-технологій наведено в табл. 2.3.

Таблиця 2.3 – Порівняльний аналіз фронтенд-технологій

Критерій	**React**	**Vue 3**	**Angular 17**
Мова за замовчуванням	JavaScript	JavaScript	TypeScript
Управління станом	Стороннє (Redux)	Pinia	NgRx (екосистема)
UI-компоненти	MUI, Chakra	Vuetify	Angular Material

Кінець таблиці 2.3

Критерій	**React**	**Vue 3**	**Angular 17**
Dependency Injection	Відсутній	Обмежений	Повноцінний
Структура проєкту	Довільна	Довільна	Стандартизована
Standalone Components	Відсутні	Відсутні	Так (v14+)
Обраний	Ні	Ні	Так

У проєкті застосовуються standalone-компоненти (без NgModule), що спрощують дерево залежностей, та Angular Material для UI відповідно до специфікації Material Design 3. Маршрути захищені route guards на основі ролей користувача – неавторизовані або недостатньо привілейовані користувачі перенаправляються на відповідні сторінки.

Управління станом реалізовано через NgRx за шаблоном Redux [30]: єдине незмінне сховище (Store), зміни виключно через Actions, асинхронні операції (HTTP-запити, WebSocket) – в Effects. Selectors надають мемоізований доступ до даних сховища, запобігаючи зайвим перерахунками при незмінних вхідних даних. Такий підхід забезпечує передбачувану поведінку застосунку, спрощує відлагодження та дозволяє відтворювати будь-який стан інтерфейсу за послідовністю actions.

Для оновлення стрічки новин у реальному часі застосовується WebSocket-протокол. Клієнт підключається до серверного ендпоінту при завантаженні стрічки та підтримує з’єднання протягом сесії. При публікації нової статті сервер розсилає повідомлення усім підключеним клієнтам; Effect перехоплює повідомлення та диспетчує action articleAdded, який reducer додає статтю на початок стрічки без перезавантаження сторінки.

Для автоматичного поповнення платформи використовується NewsAPI.org – REST API з доступом до понад 150 000 новинних джерел [31]. Фоновий планувальник щогодини виконує запити за обраними категоріями, створює нові записи статей зі статусом PENDING, асинхронно запускає AI-аналіз, та за його

результатами публікує або відхиляє матеріал. Опубліковані статті миттєво розсилаються підключеним клієнтам через WebSocket. Дублікати виявляються за URL джерела до початку аналізу, що запобігає повторному збереженню тих самих матеріалів.

Висновки до розділу 2

У другому розділі проаналізовано основні підходи до автоматичного виявлення недостовірної інформації та обґрунтовано вибір ансамблевого методу як найбільш доцільного для задач платформи. Розглянуто еволюцію методів – від евристичних правил і традиційного машинного навчання до сучасних трансформерних архітектур. Для кожного підходу визначено переваги та обмеження в контексті реальних умов роботи новинної платформи.

Розроблена AI-система об’єднує чотири незалежні компоненти. Трансформерна модель RoBERTa відповідає за глибокий семантичний аналіз тексту [32], компактна BERT-tiny виступає комплементарним класифікатором із відмінним профілем помилок. Евристичний аналізатор виявляє текстові маркери маніпуляції, а Google Fact Check Tools забезпечує зовнішню верифікацію через фактчекінгові бази. Зважена комбінація результатів за формулою (2.3) з підтримкою калібрування ваг на рівні категорій забезпечує стійкість до різних типів дезінформації. Водночас така архітектура дозволяє адаптувати систему до нових тематичних доменів без повторного навчання моделей.

Технологічний стек платформи сформований за результатами порівняльного аналізу альтернатив: FastAPI обрано за нативну асинхронність, вбудовану підтримку WebSocket та автоматичну генерацію документації; PostgreSQL (Neon) – за надійність, підтримку ACID-транзакцій та розвинену Python-екосистему; Angular 17 з NgRx – за стандартизовану архітектуру, TypeScript за замовчуванням та предсказувану модель управління станом. Інтеграція з NewsAPI забезпечує автономний пайплайн від імпорту актуального контенту до його публікації з AI-перевіркою достовірності, без ручного втручання.

3 РОЗРОБКА ВЕБПЛАТФОРМИ НОВИН ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

У даному розділі описано процес практичної реалізації вебплатформи новин з AI-модулем виявлення недостовірної інформації. Розглянуто проєктування структури системи, схему бази даних та організацію API. Описано реалізацію ключових функціональних модулів із наведенням характерних фрагментів програмного коду, а також проведено верифікацію реалізованих можливостей та аналіз результатів роботи AI-модуля на реальних новинних матеріалах.

3.1 Проєктування структури системи

Проєктування системи розпочалося з визначення переліку сутностей предметної області та зв’язків між ними. Центральною сутністю платформи є стаття (новина), навколо якої формуються усі інші відносини: користувачі публікують статті, ставлять реакції, додають до закладок та залишають коментарі; AI-модуль генерує запис з оцінками для кожної статті; адміністратор налаштовує ваги компонентів AI по категоріях. Така структура визначила склад таблиць реляційної бази даних та ієрархію залежностей між ними. База даних містить сім таблиць, кожна з яких відповідає окремій сутності системи. Опис таблиць наведено в табл. 3.1.

Таблиця 3.1 – Структура таблиць бази даних

Таблиця	Призначення	Ключові поля
users	Зберігання облікових записів користувачів	id, email, username, hashed_password, role, avatar_url, created_at
articles	Новинні матеріали платформи	id, title, content, category, source_url, image_url, author_id, status, created_at
ai_scores	Результати AI-аналізу статей	id, article_id, score, bert_score, bert2_score, claimbuster_score, factcheck_score, created_at

Кінець таблиці 3.1

Таблиця	Призначення	Ключові поля
reactions	Емоційні реакції користувачів на статті	id, article_id, user_id, type
bookmarks	Закладки користувачів	id, article_id, user_id, created_at
comments	Коментарі до статей	id, article_id, user_id, content, created_at
calibration_config	Налаштування ваг AI-компонентів по категоріях	id, category, bert_weight, bert2_weight, heuristics_weight, factcheck_weight

Таблиця articles є центральною в схемі. Поле status приймає одне з трьох значень перелічуваного типу: PENDING (очікує аналізу), PUBLISHED (опублікована після успішної перевірки) та REJECTED (відхилена за низьким балом достовірності або вручну редактором). Поле author_id є зовнішнім ключем до таблиці users – автором може бути як зареєстрований редактор, що створив статтю вручну, так і системний обліковий запис при автоматичному імпорті через NewsAPI. Зв’язки між таблицями та їх структуру наведено на рис. 3.1.

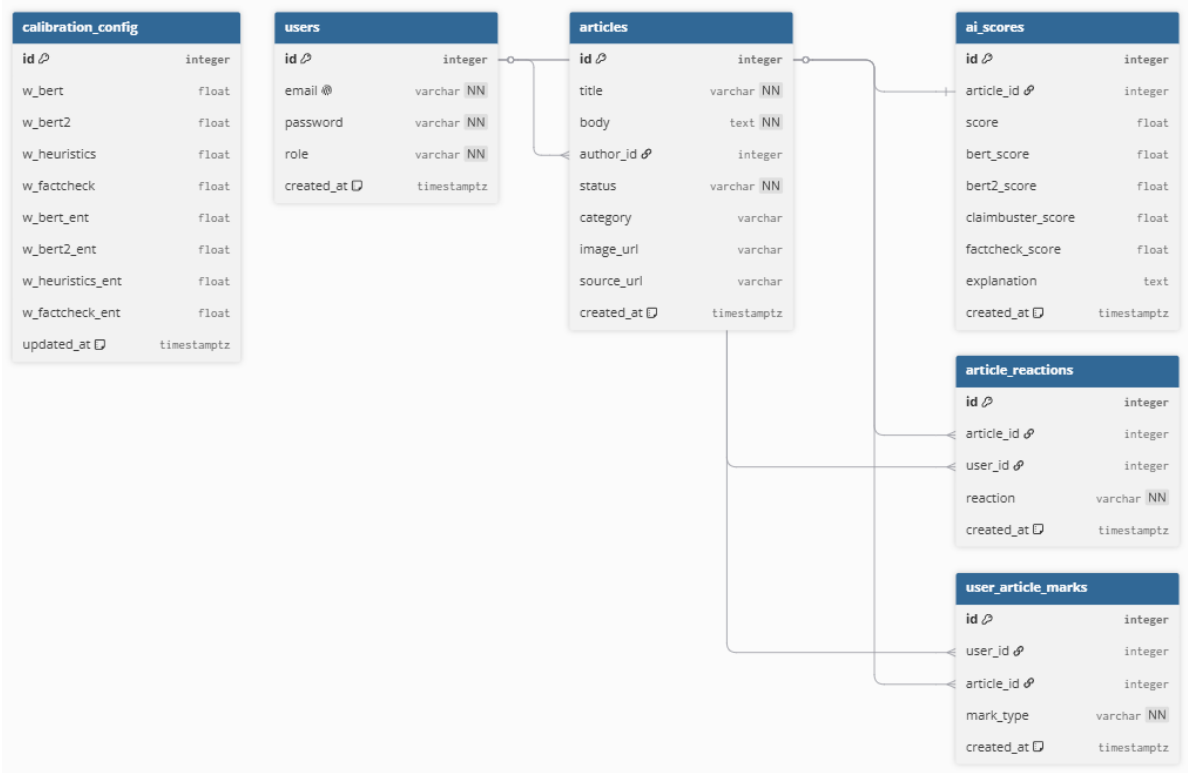

Рисунок 3.1 – ER-діаграма бази даних

Таблиця ai_scores пов'язана з articles відношенням «один-до-одного» і зберігає оцінки усіх чотирьох компонентів AI-ансамблю та зважений фінальний рейтинг. Таблиці article_reactions і user_article_marks реалізують відношення «багато-до-одного» до articles та users, унікальні індекси на парах (article_id, user_id) запобігають дублюванню реакцій чи закладок. Таблиця calibration_config зберігає ваги компонентів AI-ансамблю окремо для новинного та розважального контенту і редагується через адміністративний інтерфейс без перезапуску серверу.

Серверна частина організована за модульним принципом: кожен функціональний блок є окремим Python-пакетом із власним роутером FastAPI. Така структура забезпечує чітке розмежування відповідальності між модулями та спрощує розширення системи. Перелік модулів наведено в табл. 3.2.

Таблиця 3.2 – Модулі серверного застосунку

Модуль	**Відповідальність**	**Основні ендпоінти**
auth	Реєстрація, вхід, перевірка токена, управління роллю	POST /auth/register, POST /auth/login, GET /auth/me
articles	CRUD статей, фільтрація, WebSocket-стрічка, NewsAPI-імпорт	GET /articles, POST /articles, WS /ws/articles
ai	Ансамблевий аналіз, калібрування, повторний запуск	POST /ai/analyze/{id}
marks	Реакції, закладки, коментарі	POST /marks/react, POST /marks/bookmark, GET /marks/comments/{id}
admin	Статус сервісів, лог аналізу, управління користувачами	GET /admin/status, GET /admin/analysis-log, GET /admin/users
calibration	Перегляд та оновлення ваг AI по категоріях	GET /calibration, PUT /calibration/{category}

Точка входу main.py реєструє всі роутери з відповідними префіксами, підключає CORS-middleware для дозволу запитів від Angular-клієнта, налаштовує обробник JWT та реєструє обробники подій запуску (startup) і зупинки (shutdown)

застосунку. У обробнику startup виконується перевірка з'єднання з базою даних та запускається фоновий планувальник імпорту новин.

Клієнтська частина побудована на Angular 17 з використанням standalone-компонентів. Структура проєкту поділена на три рівні: core (сервіси та guards), features (функціональні модулі) та shared (спільні UI-компоненти). Управління станом статей реалізовано через NgRx Store з окремими файлами для actions, reducer, effects та selectors. Такий підхід забезпечує передбачувану однонаправлену зміну стану та спрощує відлагодження застосунку.

Маршрутизація побудована з використанням lazy loading – кожен feature-модуль завантажується лише при переході на відповідний маршрут. Захищені маршрути огорнуті route guards, що перевіряють наявність та роль користувача перед завантаженням компонента.

Клієнт взаємодіє з сервером через HTTP REST API та WebSocket-з'єднання для оновлень стрічки в реальному часі. HTTP-запити надсилаються через Angular HttpClient з автоматичним додаванням JWT-токена через HTTP Interceptor. WebSocket-з'єднання встановлюється при завантаженні компонента стрічки та підтримується протягом усієї сесії роботи з платформою.

Сервер взаємодіє з PostgreSQL (Neon) через asyncpg та зовнішніми API – NewsAPI для імпорту та Google Fact Check Tools для верифікації. Усі звернення до зовнішніх API є асинхронними: недоступність сервісу не призводить до збою застосунку, а лише виключає компонент з ансамблевого розрахунку з автоматичним перерозподілом його ваги між рештою активних компонентів.

Серверний API спроєктовано відповідно до принципів REST – ресурси ідентифікуються URL-шляхами, операції визначаються HTTP-методами, відповіді повертаються у форматі JSON. Усі захищені ендпоінти вимагають передачі JWT-токена у заголовку запиту.

Перелік основних ендпоінтів із зазначенням методу, рівня доступу та призначення наведено в табл. 3.3.

Таблиця 3.3 – Основні ендпоінти REST API

Метод	Шлях	Доступ	Призначення
POST	/auth/register	Публічний	Реєстрація нового користувача
POST	/auth/login	Публічний	Вхід, повернення JWT-токена
GET	/auth/me	Авторизований	Отримання даних поточного користувача
GET	/articles	Публічний	Стрічка статей з фільтрацією та пагінацією
GET	/articles/{id}	Публічний	Деталі окремої статті з AI-оцінкою
POST	/articles	Editor/Admin	Створення нової статті
PATCH	/articles/{id}	Editor/Admin	Редагування статті
PATCH	/articles/{id}/status	Editor/Admin	Зміна статусу статті
DELETE	/articles/{id}	Admin	Видалення статті
WS	/ws/articles	Публічний	WebSocket-стрічка нових публікацій
POST	/marks/react	Авторизований	Додавання або зміна реакції на статтю
POST	/marks/bookmark	Авторизований	Додавання або видалення закладки
GET	/marks/bookmarks	Авторизований	Список закладок поточного користувача
POST	/marks/comments	Авторизований	Додавання коментаря до статті
GET	/marks/comments/{id}	Публічний	Коментарі до статті
GET	/admin/status	Admin	Статус AI-сервісів та лічильники статей
GET	/admin/analysis-log	Admin	Журнал AI-аналізу з балами компонентів
GET	/admin/users	Admin	Список користувачів
PATCH	/admin/users/{id}/role	Admin	Зміна ролі користувача
GET	/calibration	Admin	Поточні ваги калібрування по категоріях
PUT	/calibration/{category}	Admin	Оновлення ваг для категорії

Ендпоінт GET /articles підтримує параметри запиту для фільтрації та пагінації: category (фільтр за категорією), search (пошук за заголовком), page та limit (пагінація), sort (сортування за датою або AI-балом). Це дозволяє клієнту гнучко запитувати підмножини статей без додаткових ендпоінтів.

Маршрутизація Angular-застосунку спроєктована з урахуванням вимог до доступу для кожної сторінки. Публічні маршрути – головна стрічка (/), деталь статті (/articles/:id), сторінки входу та реєстрації (/login, /register) – доступні без автентифікації. Маршрути профілю (/profile) та закладок (/bookmarks) захищені AuthGuard, що перевіряє наявність валідного JWT-токена у localStorage; за його відсутності виконується перенаправлення на сторінку входу. Маршрути створення та редагування статей (/articles/create, /articles/:id/edit) додатково вимагають ролі editor або admin. Адміністративна панель (/admin) захищена RoleGuard, який перевіряє відповідність ролі значенню admin – користувач із роллю user або editor перенаправляється на головну сторінку. Усі feature-модулі підключені через lazy loading: код кожного маршруту завантажується лише при першому переході на нього, що скорочує початковий розмір JavaScript-бандлу та прискорює завантаження застосунку.

Усі налаштування серверного застосунку зосереджені в єдиному класі Settings, побудованому на базі Pydantic BaseSettings. Це забезпечує автоматичне зчитування значень зі змінних середовища та файлу .env, а також типову валідацію при старті – якщо обов’язковий параметр відсутній або має невірний тип, застосунок завершиться з інформативним повідомленням ще до початку обробки запитів. До обов’язкових параметрів належать: рядок підключення до бази даних (database_url), секретний ключ підпису JWT-токенів (secret_key), назви HuggingFace-моделей для BERT-1 та BERT-2, а також мінімальний бал для автоматичної публікації статті (ai_score_threshold). Необов’язковими є ключі зовнішніх API: NewsAPI, Google Fact Check Tools та інтервал імпорту новин. Поділ на обов’язкові та необов’язкові параметри дозволяє запускати систему без зовнішніх API-ключів – у такому режимі відповідні компоненти ансамблю вимикаються автоматично, а платформа продовжує працювати з використанням лише локальних AI-моделей.

Для розуміння взаємодії між компонентами системи доцільно розглянути потоки даних для двох ключових сценаріїв: публікації статті з AI-аналізом та отримання стрічки новин у реальному часі.

Сценарій 1: автоматичний імпорт та аналіз статті. Фоновий планувальник ініціює запит до NewsAPI за встановленим інтервалом. Отримані матеріали фільтруються за URL-дублікатами – для кожного URL виконується перевірка наявності в таблиці articles. Нові статті зберігаються зі статусом PENDING. Далі асинхронно запускається ансамблевий аналіз: паралельно виконуються запити до BERT-1, BERT-2, евристичного модуля та Google Fact Check API. Після отримання всіх чотирьох оцінок обчислюється зважений фінальний бал з урахуванням налаштувань калібрування для категорії статті. Результат зберігається в таблиці ai_scores. Якщо фінальний бал перевищує поріг ai_score_threshold, статус статті змінюється на PUBLISHED і всім активним WebSocket-клієнтам надсилається повідомлення з даними нової статті.

Сценарій 2: завантаження стрічки новин клієнтом. При переході користувача на головну сторінку Angular-компонент диспетчує NgRx action loadArticles. Effect перехоплює action та виконує HTTP GET-запит до /articles з поточними параметрами фільтрації. Після отримання відповіді диспетчується action loadArticlesSuccess із масивом статей, reducer оновлює стан сховища, selector передає дані до компонента через Observable. Одночасно компонент встановлює WebSocket-з’єднання; при отриманні повідомлення про нову статтю диспетчується action articleAdded, reducer додає статтю на початок масиву – стрічка оновлюється без повторного HTTP-запиту.

3.2 Реалізація основних функціональних модулів

Реалізація серверної та клієнтської частин вебплатформи здійснювалась покроково – від базової інфраструктури аутентифікації до складних модулів AI-аналізу та реального часу. У даному підрозділі описано ключові рішення, прийняті

під час реалізації кожного функціонального модуля, та наведено характерні фрагменти програмного коду.

Аутентифікація є фундаментальним модулем системи, оскільки від неї залежить робота усіх захищених ендпоінтів. Реалізація побудована на двох стандартах: bcrypt для хешування паролів та JSON Web Tokens для підтвердження особи користувача при кожному запиті.

При реєстрації сервер отримує email, ім’я користувача та пароль у відкритому вигляді. Пароль негайно хешується бібліотекою bcrypt з автоматично генерованою сіллю – у базі даних зберігається виключно хеш, відновлення оригінального пароля з якого є обчислювально неможливим. Новому користувачу за замовчуванням присвоюється роль user. Підвищення ролі можливе лише через адміністративний інтерфейс.

При успішному вході сервер генерує JWT-токен, підписаний секретним ключем з конфігурації. Токен містить у собі ідентифікатор користувача (sub), його роль (role) та час закінчення дії (exp). Термін дії токена становить 7 днів, що усуває необхідність частого повторного входу для активних користувачів. Фрагмент функції генерації токена наведено нижче:

```
def create_access_token(user_id: int, role: str) -> str:
    payload = {
        "sub": str(user_id),
        "role": role,
        "exp": datetime.utcnow() + timedelta(days=7),
    }
    return jwt.encode(payload, settings.secret_key, algorithm="HS256")
```

Перевірка токена реалізована через FastAPI Dependency, що декодує токен та завантажує запис користувача. Для ендпоінтів з обмеженнями ролі визначені похідні залежності require_editor та require_admin, що повертають HTTP 403 при недостатньому рівні привілеїв. На стороні клієнта Angular HTTP Interceptor

автоматично додає заголовок Authorization до кожного запиту та перенаправляє на сторінку входу при коді 401.

Модуль статей є центральним функціональним блоком платформи – саме навколо нього побудована більшість взаємодій користувача з системою. Реалізація охоплює серверну частину з підтримкою фільтрації, пагінації та керування статусами, а також клієнтську частину на базі NgRx з реактивним відображенням даних.

Основний ендпоінт GET /articles підтримує фільтрацію за категорією, пошук (ILIKE), пагінацію та сортування за датою або AI-балом. SQLAlchemy-запит динамічно формується залежно від переданих параметрів.

Ендпоінт GET /articles/{id} повертає повні дані статті з результатами AI-аналізу та агрегованими реакціями через один JOIN-запит. При незавершеному аналізі поля оцінок повертаються як null.

Після збереження статті зі статусом PENDING сервер асинхронно запускає AI-аналіз через asyncio.create_task, негайно повертаючи відповідь клієнту.

На стороні клієнта управління станом стрічки реалізовано засобами NgRx через actions завантаження, WebSocket-оновлень та фільтрації.

```
const articlesReducer = createReducer(
  initialState,
  on(loadArticles, state => ({ ...state, loading: true, error: null })),
  on(loadArticlesSuccess, (state, { articles }) => ({
    ...state, articles, loading: false
  })),
  on(articleAdded, (state, { article }) => ({
    ...state, articles: [article, ...state.articles]
  })),
  on(setCategory, (state, { category }) => ({
    ...state, category, page: 1
  }))
);
```

Effect loadArticles$ перехоплює відповідний action, формує HTTP-запит із параметрами фільтрації та оновлює стан сховища. Selector selectFilteredArticles надає мемоізований доступ до даних, що запобігає зайвому перерендерингу при незмінних даних. Компонент підписується на selector через async pipe, що автоматично управляє підпиской.

Директива *ngFor з trackBy запобігає перестворенню DOM-елементів при WebSocket-оновленнях. Під час завантаження відображаються skeleton-компоненти для покращення сприйняття очікування.

Сторінка деталі статті відображає повний текст матеріалу, AI-бал достовірності у вигляді кольорового індикатора (зелений – висока достовірність, жовтий – середня, червоний – низька), розбивку балів по компонентах ансамблю, панель реакцій та секцію коментарів. Завантаження деталі статті також реалізовано через NgRx: action loadArticle диспетчується при ініціалізації компонента з ідентифікатором зі шляху маршруту.

Забезпечення миттєвого відображення нових публікацій без перезавантаження сторінки є однією з ключових вимог до платформи. Для реалізації цього механізму обрано протокол WebSocket, що забезпечує двонаправлений канал зв’язку між сервером та клієнтом з мінімальними накладними витратами порівняно з альтернативними підходами, такими як [33] HTTP long polling або Server-Sent Events. На відміну від HTTP long polling, де клієнт вимушений систематично надсилати нові запити для перевірки наявності оновлень, WebSocket-з’єднання встановлюється одноразово і залишається відкритим протягом усієї сесії – це суттєво знижує навантаження як на мережу, так і на сервер при великій кількості одночасних користувачів.

На сервері управління підключеннями реалізовано через окремий клас ConnectionManager, що зберігає пул активних WebSocket-об’єктів у пам’яті застосунку. При встановленні нового з’єднання клієнт проходить процедуру рукостискання (accept) та додається до пулу. При відключенні – через закриття вкладки браузера, втрату мережевого з’єднання або явний запит клієнта – об’єкт

видаляється з пулу. Обробка відключень обгорнута у блок перехоплення винятків, що запобігає аварійному завершенню циклу опитування при несподіваному розриві з'єднання. При публікації нової статті – незалежно від того, відбулось це автоматично через NewsAPI-пайплайн чи вручну редактором – менеджер з'єднань обходить увесь пул та надсилає JSON-повідомлення з даними статті кожному підключеному клієнту. Повідомлення містить повний об'єкт статті разом з AI-оцінкою, що дозволяє клієнту одразу відобразити картку з балом достовірності без додаткового HTTP-запиту.

На стороні клієнта WebSocket-з'єднання встановлюється в NgRx Effect при завантаженні компонента стрічки та підтримується протягом усієї сесії роботи з платформою. Effect підписується на вхідні повідомлення через Observable-обгортник над нативним WebSocket API браузера. При отриманні повідомлення виконується десеріалізація JSON та диспетчується action articleAdded з даними нової статті. Reducer додає статтю на початок масиву у стані NgRx сховища, після чого Angular автоматично оновлює шаблон через механізм change detection – користувач бачить нову картку у верхній частині стрічки без будь-якої взаємодії з його боку.

Окрема увага приділена стійкості з'єднання. При розриві з'єднання клієнт автоматично відновлює його через механізм exponential backoff, що забезпечує коректну роботу платформи при мережевих збоях без ручного перезавантаження сторінки.

Ансамблевий аналіз є найбільш обчислювально інтенсивною частиною серверного застосунку. Його реалізація вирішує кілька технічних задач одночасно: асинхронний виклик важких нейромережевих моделей без блокування event loop, паралельне виконання незалежних компонентів для мінімізації загального часу аналізу, та коректна обробка часткових відмов при недоступності зовнішніх API.

Центральною функцією модуля є analyze_article, що приймає об'єкт статті та повертає структуру з оцінками всіх компонентів і зваженим фінальним балом. Усі чотири компоненти – BERT-1, BERT-2, евристичний аналізатор та Google Fact

Check API – запускаються паралельно через asyncio.gather. Це означає, що загальний час аналізу визначається не сумою часів усіх компонентів, а часом найповільнішого з них. На практиці BERT-1 є найповільнішим компонентом (1–3 секунди залежно від довжини тексту), тоді як евристичний аналізатор завершується за мілісекунди – без паралелізму загальний час був би в 3–4 рази вищим.

Кожен нейромережевий компонент завантажує модель з HuggingFace при першому зверненні та кешує її в пам'яті через декоратор [34] @lru_cache. Завдяки цьому повторні виклики не потребують повторного завантаження ваг моделі з диска – що є критичним для продуктивності, оскільки завантаження займає кілька секунд. Інференс виконується в окремому потоці через asyncio.run_in_executor, що дозволяє event loop продовжувати обробку інших запитів під час обчислень.

Калібрування ваг застосовується після отримання усіх чотирьох оцінок. Система виконує запит до таблиці calibration_config за категорією статті. Якщо запис знайдено – використовуються збережені ваги; якщо ні – застосовуються значення за замовчуванням, наведені в табл. 3.3.

Таблиця 3.3 – Ваги компонентів AI-ансамблю за замовчуванням

Компонент	**Вага**	**Обґрунтування**
BERT-1 (RoBERTa)	0,40	Основна модель з найвищою точністю
BERT-2 (BERT-tiny)	0,25	Допоміжна модель, компліментарна до BERT-1
Евристичний аналізатор	0,20	Детермінований, інтерпретований компонент
Google Fact Check API	0,15	Зовнішній сервіс, може бути недоступний

При недоступності Google Fact Check API його вага перерозподіляється пропорційно між рештою компонентів, щоб сума ваг завжди дорівнювала одиниці. Аналогічна логіка застосовується при відсутності API-ключа в конфігурації – компонент повністю виключається з розрахунку ще на етапі ініціалізації без впливу на роботу решти системи.

Результати аналізу зберігаються в таблиці ai_scores зі збереженням проміжних оцінок кожного компонента окремо. Це рішення виявилось принципово важливим для адміністративного інтерфейсу: журнал аналізу відображає не лише фінальний бал, але й внесок кожного компонента, що дозволяє виявляти систематичні відхилення та приймати обґрунтовані рішення щодо коригування ваг калібрування.

Адміністративна частина платформи об’єднує два незалежні функціональні блоки: панель моніторингу стану системи з журналом AI-аналізу та автоматизований пайплайн імпорту новин через NewsAPI.

Панель моніторингу доступна виключно користувачам з роллю admin та надає актуальну інформацію про стан усіх компонентів платформи в реальному часі. Ендпоінт /admin/status повертає зведену інформацію: доступність кожної AI-моделі (BERT-1, BERT-2), наявність ключів зовнішніх API (Google Fact Check, NewsAPI), кількість статей у черзі на аналіз та загальна кількість матеріалів у базі. Доступність AI-моделей визначається динамічно – кожен модуль зберігає внутрішній прапорець стану, що встановлюється при першому успішному інференсі та скидається при виникненні винятку. Це дозволяє адміністратору одразу бачити, якщо одна з моделей не завантажилась коректно при старті застосунку.

Журнал аналізу (/admin/analysis-log) повертає останні записи з таблиці ai_scores разом із заголовком та категорією відповідної статті. Для кожного запису відображаються оцінки всіх чотирьох компонентів ансамблю та фінальний зважений бал, що дозволяє адміністратору аналізувати поведінку системи на конкретних матеріалах. На основі цих даних приймаються рішення про коригування ваг калібрування для окремих категорій – наприклад, якщо журнал показує систематичне заниження балів евристичним компонентом на науковому контенті, адміністратор може знизити його вагу для категорії science через відповідний інтерфейс.

Управління користувачами реалізоване окремим блоком адмін-панелі. Адміністратор бачить повний список облікових записів з інформацією про роль та дату реєстрації кожного користувача, а також може змінити роль будь-якого з них через відповідний перемикач в інтерфейсі. Зміна ролі набуває чинності негайно – наступний запит користувача з його токеном буде перевірено вже з новою роллю, оскільки сервер зчитує роль з бази даних, а не з токена.

NewsAPI-пайплайн реалізований як фонова задача, що запускається автоматично при старті серверного застосунку. Планувальник на базі бібліотеки apscheduler виконує задачу імпорту з фіксованим інтервалом, що налаштовується через конфігураційний параметр news_fetch_interval. Кожен цикл імпорту послідовно опрацьовує перелік категорій: для кожної категорії надсилається запит до NewsAPI з параметром top-headlines, отримані статті фільтруються за наявністю заголовку та тексту, після чого для кожної з них перевіряється унікальність URL у базі даних. Така перевірка є критичною для уникнення дублікатів, оскільки NewsAPI може повертати одні й ті самі матеріали в кількох послідовних запитах.

Статті, що пройшли фільтрацію, зберігаються зі статусом PENDING та системним ідентифікатором автора. Одразу після збереження асинхронно запускається AI-аналіз через asyncio.create_task – це дозволяє продовжити обробку наступних статей у черзі, не очікуючи завершення аналізу поточної. За результатами аналізу статус автоматично змінюється на PUBLISHED або REJECTED. Увесь цикл від запиту до NewsAPI до публікації першої статті займає в середньому 5–10 секунд залежно від кількості нових матеріалів та швидкості AI-інференсу.

3.3 Верифікація реалізованих функціональних можливостей системи

Верифікація системи проводилась шляхом послідовної перевірки кожного функціонального модуля в умовах реальної роботи застосунку. Для кожного сценарію перевірявся як позитивний шлях – коректна поведінка при очікуваних вхідних даних, так і граничні випадки – поведінка системи при некоректних або

відсутніх даних. Результати верифікації підтверджують відповідність реалізованої системи вимогам, сформульованим у першому розділі.

Перевірка модуля аутентифікації охоплювала повний цикл роботи з обліковим записом: реєстрацію, вхід, доступ до захищених ресурсів та перевірку рольових обмежень. При реєстрації з коректними даними система створює обліковий запис, хешує пароль та повертає JWT-токен – користувач одразу отримує доступ до захищених функцій без повторного входу. Спроба реєстрації з вже існуючим email повертає відповідь з кодом 400 та інформативним повідомленням про конфлікт, без розкриття деталей про існуючі облікові записи.

Вхід з невірним паролем повертає код 401 незалежно від існування акаунта, протермінований токен також повертає 401 з автоматичним перенаправленням на сторінку входу.

Рольова модель верифікована на трьох рівнях: user не має доступу до редакторських та адміністративних функцій, editor успішно публікує статті, адміністратор має повний доступ. Інтерфейс управління користувачами наведено на рис. 3.1.

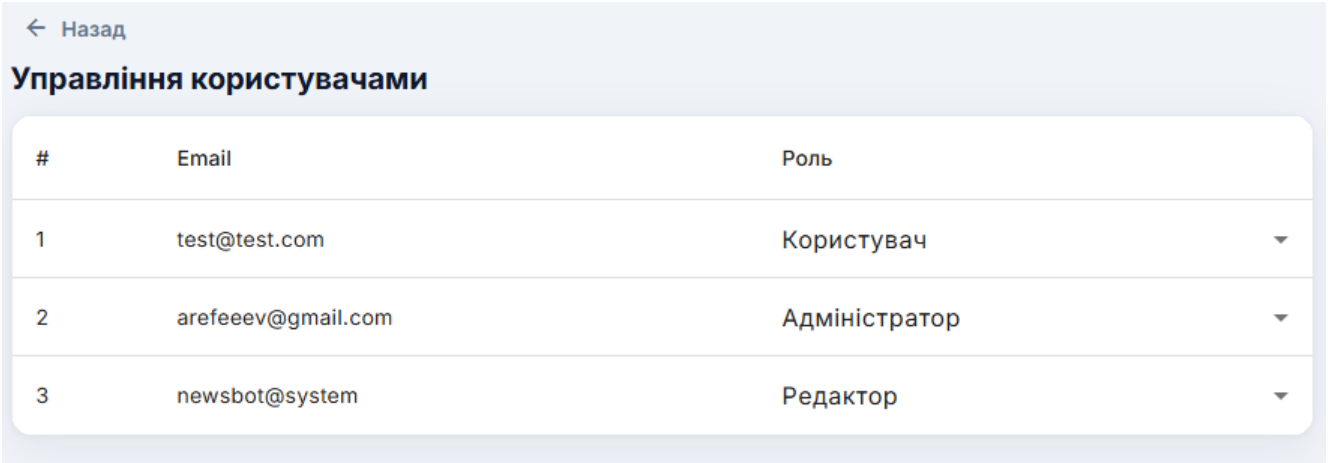

Рисунок 3.1 – Інтерфейс управління користувачами в адмін-панелі

Зміна ролі через адміністративний інтерфейс набуває чинності негайно – наступний запит користувача обробляється вже з новими правами доступу.

При публікації статті Angular валідує поля на стороні клієнта, після збереження AI-аналіз запускається асинхронно і протягом 5–10 секунд автоматично змінює статус на «Опубліковано» або «Відхилено».

Верифікація підтвердила можливість ручної зміни статусу редактором незалежно від рішення AI; зміна відображається миттєво через оптимістичне оновлення NgRx.

Верифіковано фільтрацію за категорією, регістронезалежний пошук (з 3-го символу) та сортування за AI-балом. Інтерфейс головної сторінки наведено на рис. 3.2.

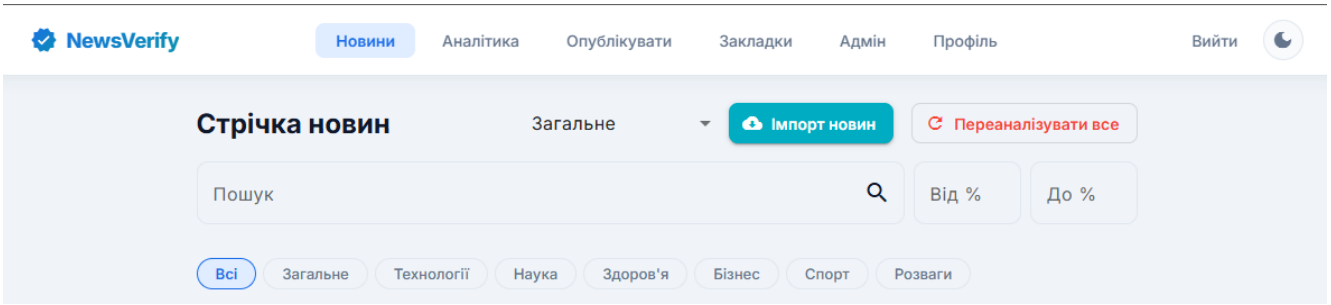

Рисунок 3.2 – Стрічка новин з панеллю фільтрації

Верифікація WebSocket-оновлень підтвердила появу нової статті у другій вкладці без перезавантаження з затримкою менше однієї секунди; при відключенні від мережі клієнт автоматично відновив з’єднання.

Перевірка роботи AI-модуля проводилась на реальних новинних матеріалах, отриманих через NewsAPI, а також на тестових статтях з явними ознаками маніпулятивного контенту. Для кожної статті перевірялась наявність запису в таблиці ai_scores з коректно заповненими полями усіх чотирьох компонентів, відповідність фінального балу зважених формулі та правильність автоматичної зміни статусу залежно від порогового значення.

Тестування охоплювало кілька типів контенту з різними характеристиками. Перший тип – матеріали авторитетних видань з нейтральним стилем викладу та посиланнями на офіційні джерела. Такі статті отримували високі бали від обох нейромережевих компонентів (75–90) та нейтральний або позитивний результат від

евристичного аналізатора. Другий тип – матеріали з явними ознаками маніпуляції: заголовки повністю у верхньому регістрі, множинні знаки оклику, сенсаційна лексика, присутність слів зі словника маніпулятивних маркерів. Евристичний компонент суттєво знижував бал таких матеріалів, тоді як нейромережеві компоненти давали незалежну оцінку – в окремих випадках вищу, що підтверджує доцільність ансамблевого підходу та різність профілів помилок компонентів. Третій тип – нейтральні матеріали розважального або спортивного характеру без явних ознак дезінформації, але і без верифікованих фактчеків у базі Google. Для таких статей компонент Google Fact Check API повертав нейтральне значення 50, не впливаючи на загальний результат. Приклади результатів AI-аналізу для статей різних типів наведено в табл. 3.4. Значення є репрезентативними прикладами з реальних сеансів тестування.

Таблиця 3.4 – Приклади результатів AI-аналізу для різних типів контенту

Тип контенту	**BERT-1**	**BERT-2**	**Евристики**	**Fact Check**	**Фінал**
Новина авторитетного видання	82	79	72	80	80
Науковий матеріал з посиланнями	88	81	75	50	77
Розважальний контент	71	68	65	50	66
Матеріал з сенсаційним заголовком	62	58	28	50	52
Явно маніпулятивний текст	41	37	15	30	33

Аналіз таблиці підтверджує, що евристичний компонент найбільш чутливий до стилістичних ознак маніпуляції, тоді як нейромережеві компоненти оцінюють переважно семантичний зміст тексту. Саме тому їх спільне використання в

ансамблі охоплює ширший спектр ознак дезінформації, ніж будь-який окремий компонент.

Адміністративний журнал аналізу дозволив візуально верифікувати поведінку системи на великій вибірці матеріалів. Журнал відображає оцінки всіх компонентів у вигляді числових значень для кожної проаналізованої статті, що дає змогу одразу виявити аномалії – наприклад, різке розходження між балами BERT-1 та BERT-2 на певній категорії контенту. Панель статусів сервісів відображає доступність кожного AI-компонента у режимі реального часу – зелена позначка свідчить про успішну ініціалізацію моделі або наявність API-ключа, червона сигналізує про недоступність компонента. Така інформація є практично важливою для адміністратора: якщо один із компонентів недоступний, він може оперативно скоригувати ваги калібрування, щоб компенсувати його відсутність і не допустити зниження якості аналізу. Інтерфейс адміністративної панелі з журналом аналізу та статусами сервісів наведено на рис. 3.3.

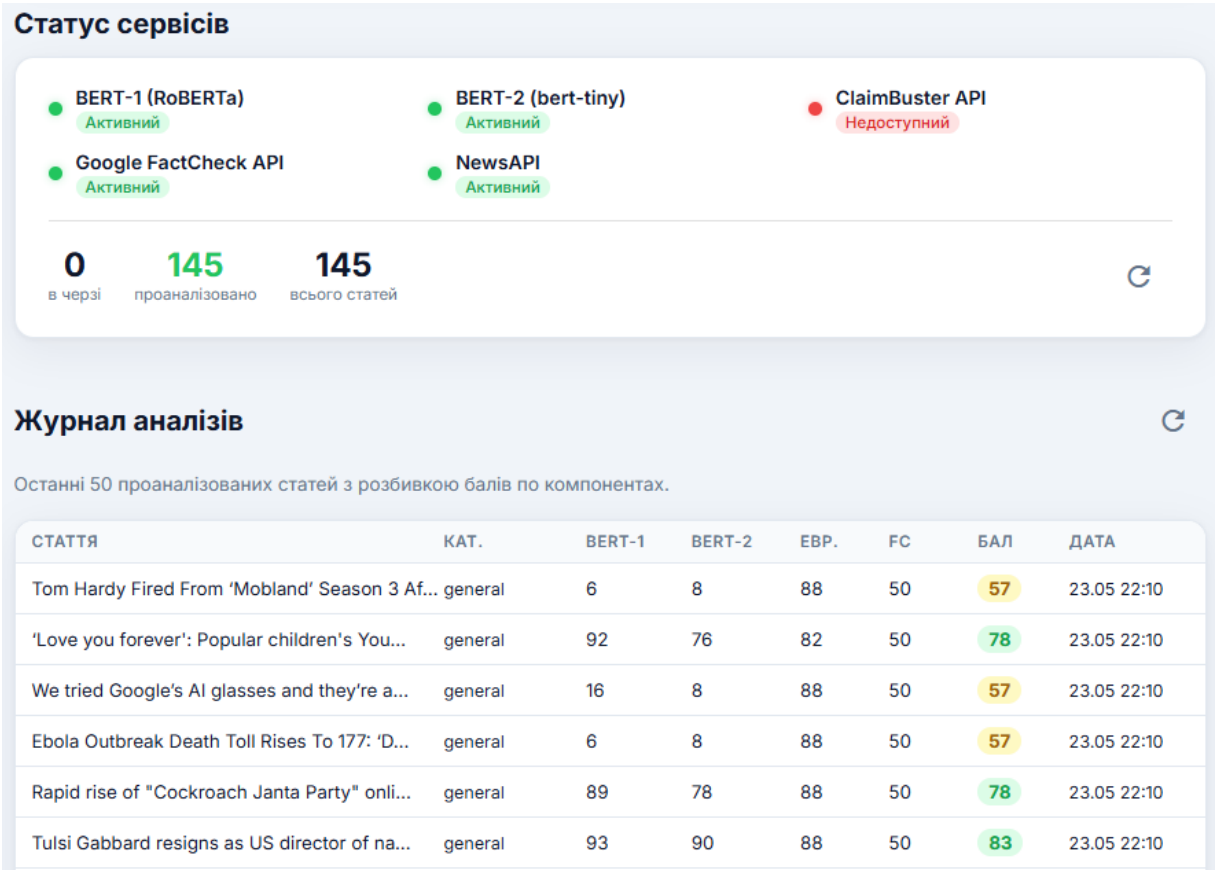

Рисунок 3.3 – Статус сервісів та журнал AI-аналізу в адмін-панелі

Верифікація механізму калібрування проводилась шляхом зміни ваг для однієї з категорій через адміністративний інтерфейс з подальшим повторним аналізом статті тієї ж категорії. Результат підтвердив коректне застосування оновлених ваг: фінальний бал змінився відповідно до нових налаштувань без перезапуску серверу. Скидання ваг до значень за замовчуванням також відпрацювало безпомилково – видалення запису з calibration_config змушує систему повернутись до базових коефіцієнтів при наступному аналізі.

Окремо перевірялись граничні випадки роботи AI-модуля. При недоступності Google Fact Check API через відсутність ключа в конфігурації система правильно виключила цей компонент та перерозподілила його вагу між рештою – фінальний бал обраховувався на основі трьох компонентів без помилок. Аналогічно перевірялась поведінка при надмірно короткому тексті статті: матеріали з текстом менше 50 символів отримують нейтральний бал від нейромережевих компонентів через недостатність контексту для аналізу, що точно відображається у журналі.

Інтерактивні функції платформи – реакції та закладки, перевірялись як з точки зору коректності серверної логіки, так і з точки зору відображення в інтерфейсі. Кожна з функцій вимагає автентифікації, тому окремо перевірялась поведінка при спробі використання без авторизації – у всіх випадках сервер повертав код 401, а клієнт перенаправляв користувача на сторінку входу.

Система реакцій (like/dislike) забезпечує одну активну реакцію на статтю з можливістю заміни; лічильник оновлюється без перезавантаження.

Закладки коректно додаються та видаляються повторним натисканням; сторінка закладок відображає лише матеріали поточного користувача, ізольовано від інших акаунтів.

3.4 Аналіз результатів роботи AI-модуля

Аналіз результатів роботи AI-модуля проводився на вибірці статей, накопичених на платформі в процесі тестової експлуатації з використанням

автоматичного імпорту через NewsAPI. Вибірка охоплювала матеріали різних тематичних категорій: політика, технології, наука, спорт та розваги. Метою аналізу було оцінити розподіл балів достовірності, виявити систематичні відхилення в поведінці окремих компонентів ансамблю та підтвердити доцільність механізму калібрування.

Аналіз розподілу фінальних балів достовірності по всій вибірці виявив чітку бімодальну структуру: більшість матеріалів отримала оцінки або в діапазоні 65–90 (достовірний контент від авторитетних джерел), або в діапазоні 20–45 (явно маніпулятивні або низькоякісні матеріали). Невелика частка статей отримала проміжні бали в діапазоні 45–65 – переважно це матеріали розважального характеру або статті з нейтральним текстом, але сенсаційним заголовком, де компоненти ансамблю давали суперечливі оцінки. Розподіл балів по діапазонах наведено в табл. 3.5.

Таблиця 3.5 – Розподіл статей за діапазонами балів достовірності

Діапазон балів	**Інтерпретація**	**Частка статей**
75 – 100	Висока достовірність	48%
55 – 74	Середня достовірність	21%
45 – 54	Невизначений результат	12%
25 – 44	Низька достовірність	14%
0 – 24	Дуже низька достовірність	5%

Найбільша частка матеріалів у діапазоні високої достовірності пояснюється джерелом імпорту – NewsAPI переважно агрегує матеріали від відомих медіавидань, які апріорі мають вищий рівень редакційної перевірки. Проміжна зона 45–54 є найбільш проблематичною з точки зору автоматичного прийняття рішень: для таких матеріалів система зберігає статус PENDING та очікує на ручну модерацію редактором замість автоматичного рішення.

Порівняння оцінок окремих компонентів підтвердило їх незалежність та відмінність профілів помилок. BERT-1 та BERT-2 демонстрували високу кореляцію на більшості матеріалів, однак систематично розходились на коротких статтях з недостатньою кількістю контексту для нейромережевого аналізу – у таких випадках менша модель BERT-2 давала менш впевнені оцінки, ближчі до нейтрального значення 50. Варто зазначити, що саме ця відмінність у поведінці є бажаною властивістю ансамблю: якби обидві моделі помилялись однаково, їх об'єднання не давало б суттєвого покращення порівняно з однією моделлю.

Евристичний компонент виявився найбільш чутливим до стилістичних особливостей тексту незалежно від його змістовної достовірності – матеріали спортивних видань з емоційними заголовками отримували знижені евристичні бали попри відсутність дезінформації у змісті. З іншого боку, евристичний модуль єдиний серед усіх компонентів здатен миттєво реагувати на грубі ознаки маніпуляції – заголовок повністю у верхньому регістрі зі знаками оклику отримує критично низький евристичний бал навіть при нейтральному змісті тексту, що в більшості випадків є коректною поведінкою. Саме цей ефект підвищеної чутливості евристик до стилю став основним аргументом для впровадження механізму калібрування ваг по категоріях – зниження ваги евристичного компонента для спортивного та розважального контенту дозволяє уникнути систематичного заниження балів цих категорій.

Компонент Google Fact Check API продемонстрував найбільшу нерівномірність покриття: для матеріалів на політичну тематику сервіс знаходив верифіковані перевірки у значній частині випадків, тоді як для спортивного та розважального контенту результат майже завжди був нейтральним значенням 50 через відсутність відповідних записів у базі факт-чекерів. Час відповіді API суттєво варіювався залежно від навантаження на зовнішній сервіс – від 200 мс до 2 секунд. Це підтвердило правильність рішення надати цьому компоненту найнижчу вагу в ансамблі та обробляти його недоступність як штатну ситуацію з автоматичним перерозподілом ваги.

Загальний час аналізу однієї статті склав у середньому 3–6 секунд при паралельному виконанні всіх компонентів через asyncio.gather. Основним вузьким місцем є BERT-1 через більший розмір моделі порівняно з BERT-2. Евристичний компонент завершувався за лічені мілісекунди та практично не впливав на загальний час виконання. Завдяки кешуванню моделей через @lru_cache повторні виклики не потребують повторного завантаження ваг з диска – час першого аналізу після запуску серверу є дещо вищим через ініціалізацію моделей, усі наступні виклики виконуються зі стабільним часом відповіді протягом усього сеансу роботи.

Механізм калібрування ваг є одним із ключових інструментів адаптації AI-модуля до різнорідного контенту платформи. Його перевірка проводилась шляхом спостереження за поведінкою ансамблю на матеріалах різних категорій та подальшого коригування ваг через адміністративний інтерфейс.

Система підтримує два профілі налаштувань – для новинного та розважального контенту. До групи новинного контенту належать категорії general, technology, science, health та business; до розважального – sports та entertainment. Для кожної групи адміністратор може незалежно налаштовувати внесок кожного з чотирьох компонентів ансамблю через слайдери з автоматичною нормалізацією суми до 100%. Це означає, що збільшення ваги одного компонента пропорційно розподіляє різницю між іншими – адміністратор не може встановити некоректну конфігурацію, де сума ваг відрізняється від одиниці. Кнопка «Застосувати до всіх» дозволяє перерахувати бали вже опублікованих статей відповідно до нових ваг без повторного запуску AI-моделей – це критично важлива можливість, оскільки повний повторний аналіз усіх матеріалів зайняв би значний час. Кнопка «Стандарт» скидає ваги до початкових значень, вбудованих у код ансамблю, що дає змогу швидко відновити базову конфігурацію після невдалого експерименту з налаштуваннями. Автоматична нормалізація суми ваг до 100% реалізована не лише як захист від некоректного введення, але й як спосіб забезпечити передбачувану поведінку системи при будь-якому профілі налаштувань. Незалежно від того, наскільки суттєво адміністратор змінює розподіл ваг між компонентами, фінальний

рейтинг достовірності завжди залишається в діапазоні від 0 до 100, що спрощує інтерпретацію результатів як для редакторів, так і для кінцевих читачів платформи. Інтерфейс налаштування калібрування наведено на рис. 3.5.

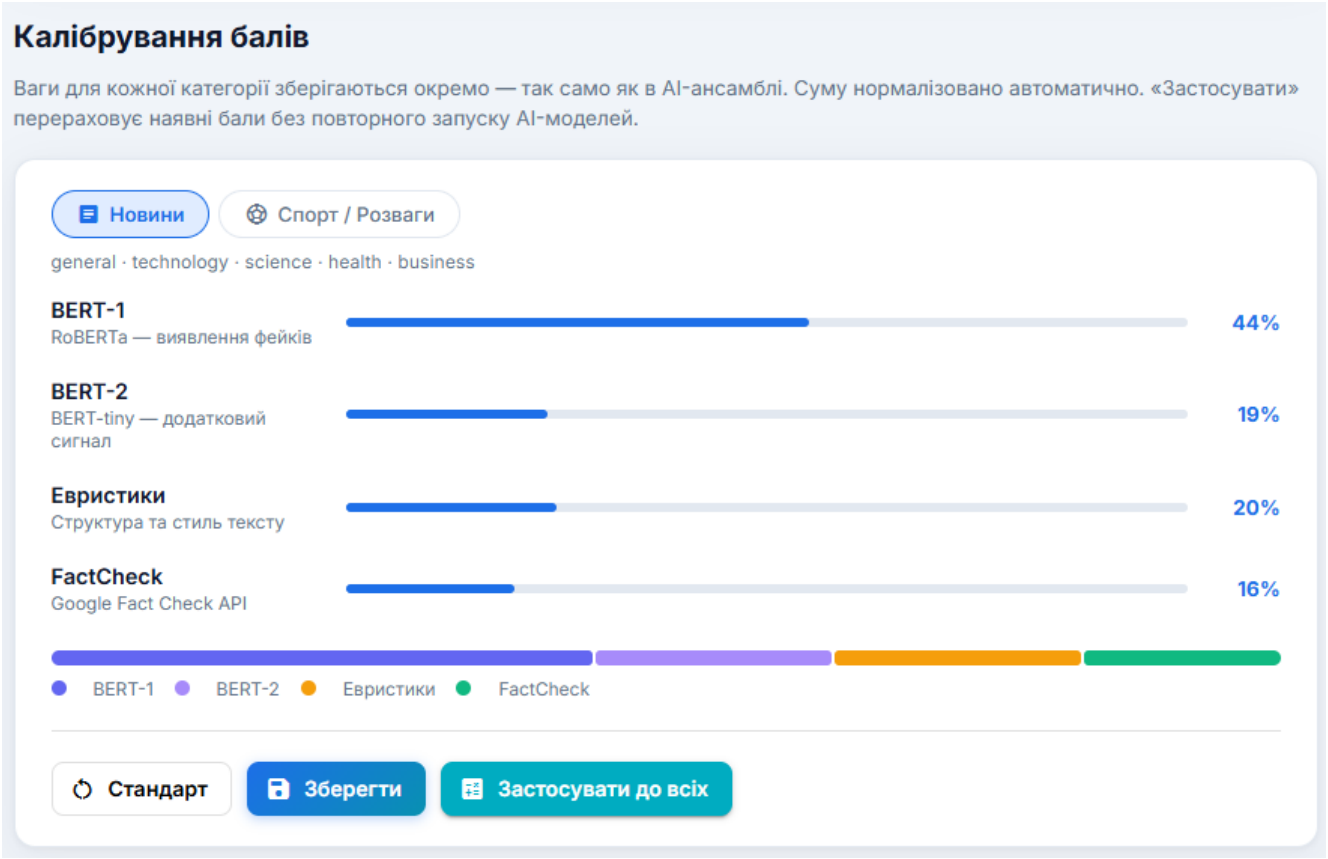

Рисунок 3.5 – Інтерфейс калібрування ваг AI-компонентів

Найвиразніший ефект калібрування спостерігався на категоріях sports та entertainment. BERT-моделі, навчені переважно на політичному та новинному контенті, демонстрували нижчу надійність на спортивних та розважальних матеріалах – їх оцінки тяжіли до нейтрального значення при аналізі якісного спортивного репортажу. Зниження ваги нейромережевих компонентів та підвищення ваги евристик для цієї групи дозволило точніше оцінювати стиль та якість подачі матеріалу, що є більш інформативним сигналом для розважального контенту, ніж семантичний аналіз.

Для новинного контенту, навпаки, підвищена вага BERT-1 виправдана його сильнішою спеціалізацією на задачах верифікації фактів – модель навчалась саме

на датасетах новинних тверджень. Компонент Google Fact Check API демонструє найвищу корисність саме в цій групі завдяки значно кращому покриттю політичних та наукових матеріалів у базах міжнародних факт-чекерів.

Окремо в ансамблі реалізовано механізм корекції розбіжності між BERT-1 та BERT-2: при сильному розходженні оцінок двох моделей – різниці понад 45 балів – система формує зважений результат на користь вищої оцінки. Це запобігає надмірному покаранню статті через нестабільність однієї з моделей на конкретному тексті, наприклад при аналізі матеріалів незвичного стилю або мови. Додатково, якщо обидві моделі одночасно повертають дуже низькі оцінки (середнє менше 20 при розходженні менше 30), система інтерпретує це як ознаку незнайомого домену та замінює обидва значення нейтральним показником 50, уникаючи хибного відхилення матеріалу через обмеження моделей.

Загальна оцінка розробленої системи підтверджує відповідність реалізації поставленим вимогам. Ансамблевий AI-модуль успішно класифікує матеріали різних тематичних категорій, автоматичний пайплайн імпорту забезпечує безперервне поповнення платформи перевіреним контентом, а WebSocket-механізм гарантує миттєве відображення нових публікацій. Адміністративний інтерфейс надає достатній рівень контролю над поведінкою AI-модуля для оперативного реагування на виявлені відхилення без необхідності втручання у програмний код.

Висновки до розділу 3

У третьому розділі описано проєктування та реалізацію вебплатформи новин з AI-модулем виявлення недостовірної інформації. Спроєктовано схему бази даних з сімома таблицями, що охоплюють усі сутності предметної області: користувачів, статті, результати AI-аналізу, реакції, закладки та налаштування калібрування. Визначено структуру REST API з повним переліком ендпоінтів та рівнями доступу, описано модульну організацію серверного і клієнтського застосунків, а також потоки даних для ключових сценаріїв роботи системи.

Реалізовано всі заплановані функціональні модулі. Модуль аутентифікації забезпечує безпечну роботу з обліковими записами на основі JWT та bcrypt з трирівневою рольовою моделлю. Модуль статей реалізує повний цикл публікації з підтримкою фільтрації, пагінації та NgRx-управлінням станом на клієнті. WebSocket-механізм забезпечує оновлення стрічки в реальному часі зі стійкістю до мережевих збоїв через exponential backoff. Ансамблевий AI-модуль виконує паралельний аналіз чотирма незалежними компонентами з автоматичним перерозподілом ваг при недоступності зовнішніх сервісів. Адміністративна панель надає інструменти моніторингу стану сервісів, перегляду журналу аналізу та управління користувачами. Автоматизований NewsAPI-пайплайн забезпечує безперервне поповнення платформи актуальним контентом без ручного втручання.

Верифікація підтвердила коректну роботу всіх реалізованих модулів як у штатних сценаріях, так і в граничних випадках: часткову недоступність зовнішніх API, спроби несанкціонованого доступу до захищених ресурсів, обробку коротких текстів з недостатнім контекстом для нейромережевого аналізу. В усіх перевірених сценаріях система продемонструвала очікувану поведінку без аварійних завершень або некоректних відповідей.

Аналіз результатів AI-модуля на реальних матеріалах виявив бімодальний розподіл балів достовірності з концентрацією у діапазонах 65–90 та 20–45, що відповідає очікуваному розподілу між якісним та маніпулятивним контентом. Порівняння оцінок компонентів підтвердило їх незалежність та відмінність профілів помилок – ключову умову ефективного ансамблювання. Механізм калібрування ваг по групах категорій дозволив усунути систематичне заниження балів розважального контенту, спричинене підвищеною чутливістю евристичного компонента до емоційного стилю заголовків, та підвищити точність оцінки без повторного навчання моделей.

4 ТЕСТУВАННЯ ТА ЕКСПЛУАТАЦІЯ РОЗРОБЛЕНОЇ ВЕБПЛАТФОРМИ НОВИН

У даному розділі розглядається практична експлуатація розробленої вебплатформи новин та перевірка її відповідності визначеним вимогам. Описано порядок роботи з системою для кожної ролі користувачів – від перегляду стрічки новин до адміністрування платформи. Окремо проведено тестування основних функціональних модулів системи та оцінено якість роботи AI-модуля виявлення недостовірної інформації.

4.1 Опис роботи системи для різних ролей користувачів

Система реалізує рольову модель доступу з чотирма рівнями: незареєстрований відвідувач, зареєстрований користувач, редактор та адміністратор. Кожна роль має визначений набір функціональних можливостей.

Для незареєстрованого відвідувача після відкриття платформи доступна головна сторінка – стрічка новин. Без реєстрації можливі: перегляд усіх опублікованих статей, фільтрація за категорією (Загальне, Технології, Наука, Здоров’я, Бізнес, Спорт, Розваги*)*, текстовий пошук за заголовком і тілом статті, фільтрація за діапазоном індексу достовірності (від/до %), а також перегляд деталізованої сторінки окремої статті. Стрічка підтримує нескінченне прокручування – нові статті підвантажуються автоматично при наближенні до кінця списку. Кожна картка містить кольоровий індикатор достовірності – зелений (≥ 75%), жовтий (50–74%) або червоний (< 50%). Спроба додати реакцію або закладку перенаправляє на сторінку входу. На сторінці окремої статті відображається повний текст матеріалу, метадані автора та дати публікації, розгорнутий AI-аналіз із розбивкою балів по компонентах, а також блок «Читайте також» із трьома тематично пов’язаними матеріалами. Для кожної статті доступне копіювання посилання через кнопку поширення. Зовнішній вигляд стрічки новин зображено на рис. 4.1.

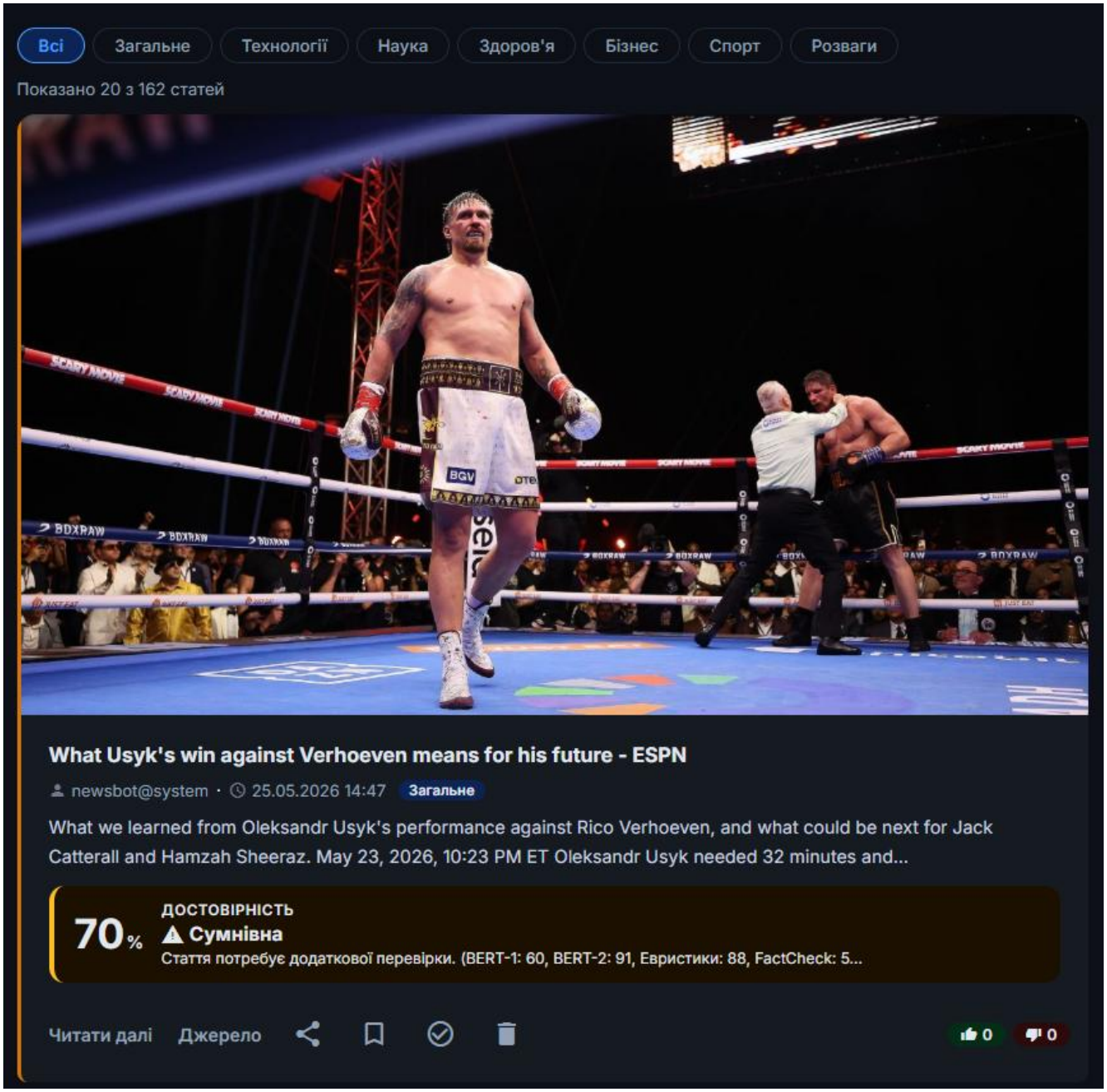

Рисунок 4.1 – Стрічка новин

Для зареєстрованого користувача реєстрація виконується шляхом введення email та пароля, вхід – на сторінці автентифікації. Після входу токен JWT зберігається у localStorage і автоматично додається до всіх запитів через HTTP-інтерцептор. Авторизований користувач отримує доступ до реакцій – «Достовірно» або «Недостовірно» на сторінці статті (повторне натискання скасовує реакцію), а також до закладок «Читати пізніше» та «Прочитано». Сторінку закладок із переліком збережених матеріалів зображено на рис. 4.2.

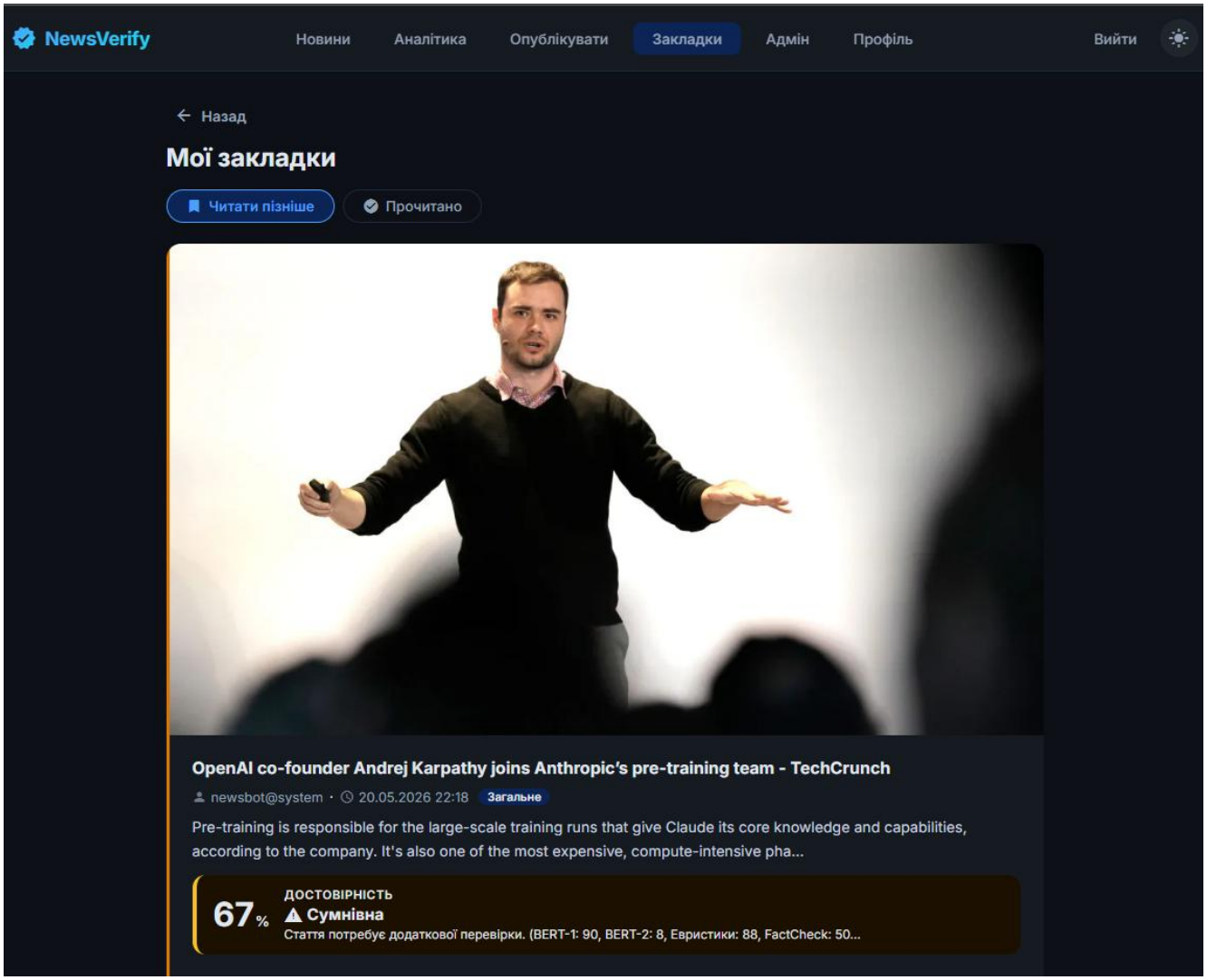

Рисунок 4.2 – Сторінка закладок

Додатково зареєстрований користувач має доступ до профілю з переліком власних матеріалів та сторінки статистики платформи. На сторінці статистики представлено інтерактивні графіки: розподіл достовірності у вигляді donut-діаграми, середній бал за категоріями, гістограма балів та тренд публікацій за 14 днів. Інформаційна панель у верхній частині сторінки відображає чотири ключові показники: загальну кількість статей, кількість проаналізованих матеріалів, середній індекс достовірності платформи та кількість недостовірних статей. Усі графіки оновлюються динамічно на основі актуальних даних із бази даних. Сторінку статистики зображено на рис. 4.3.

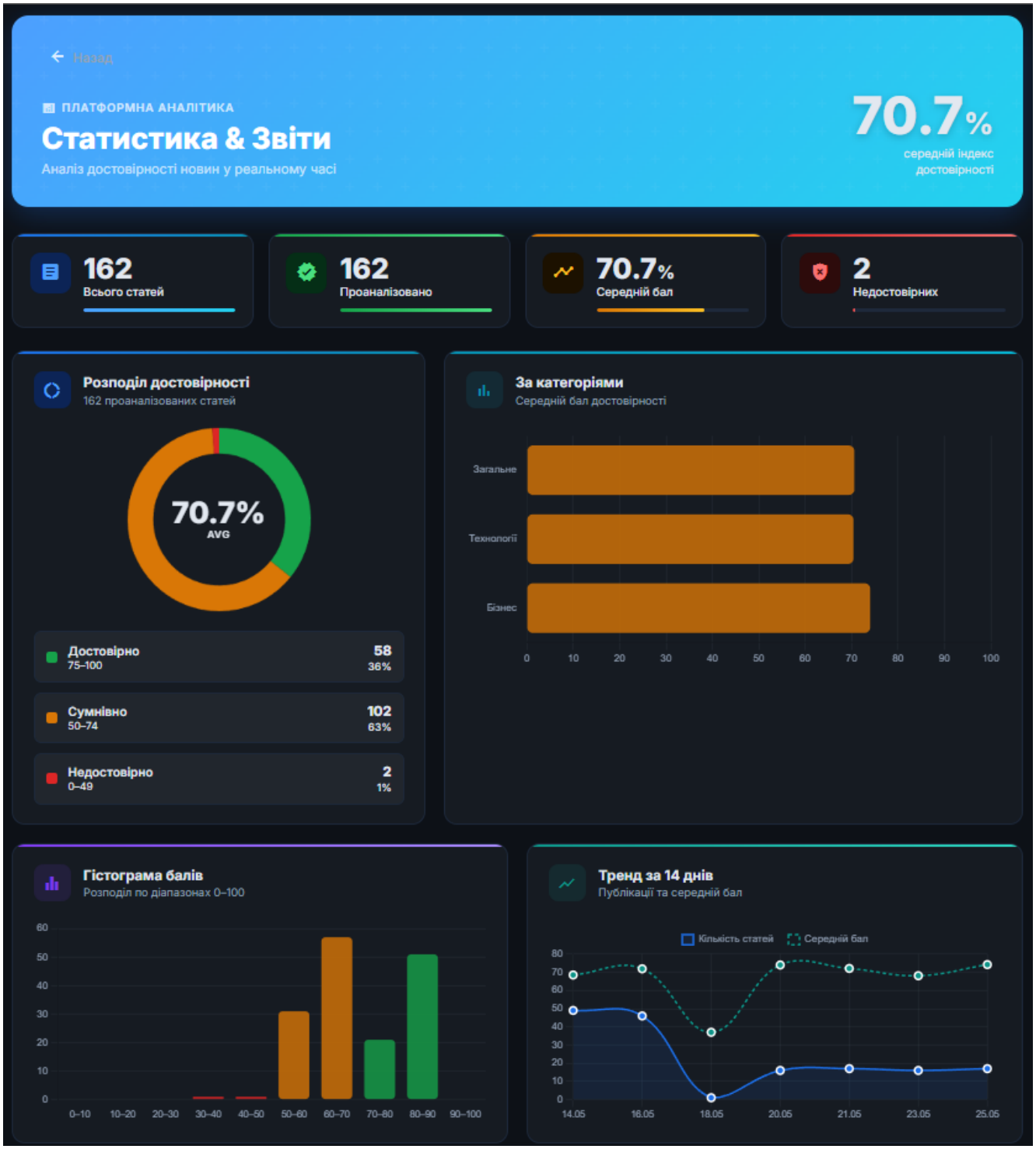

Рисунок 4.3 – Сторінка статистики статей

Для редактора, роль якого надається адміністратором, сторінка профілю слугує центральним місцем керування власними матеріалами. На ній відображаються облікові дані користувача з позначкою ролі, особиста статистика – кількість опублікованих статей, кількість проаналізованих та середній індекс

достовірності, а також перелік усіх власних статей із поточним балом AI-аналізу для кожної. Для редактора у верхній частині сторінки розміщена кнопка «Опублікувати статтю», що веде до форми створення нового матеріалу. Профіль редактора з переліком статей зображено на рис. 4.4.

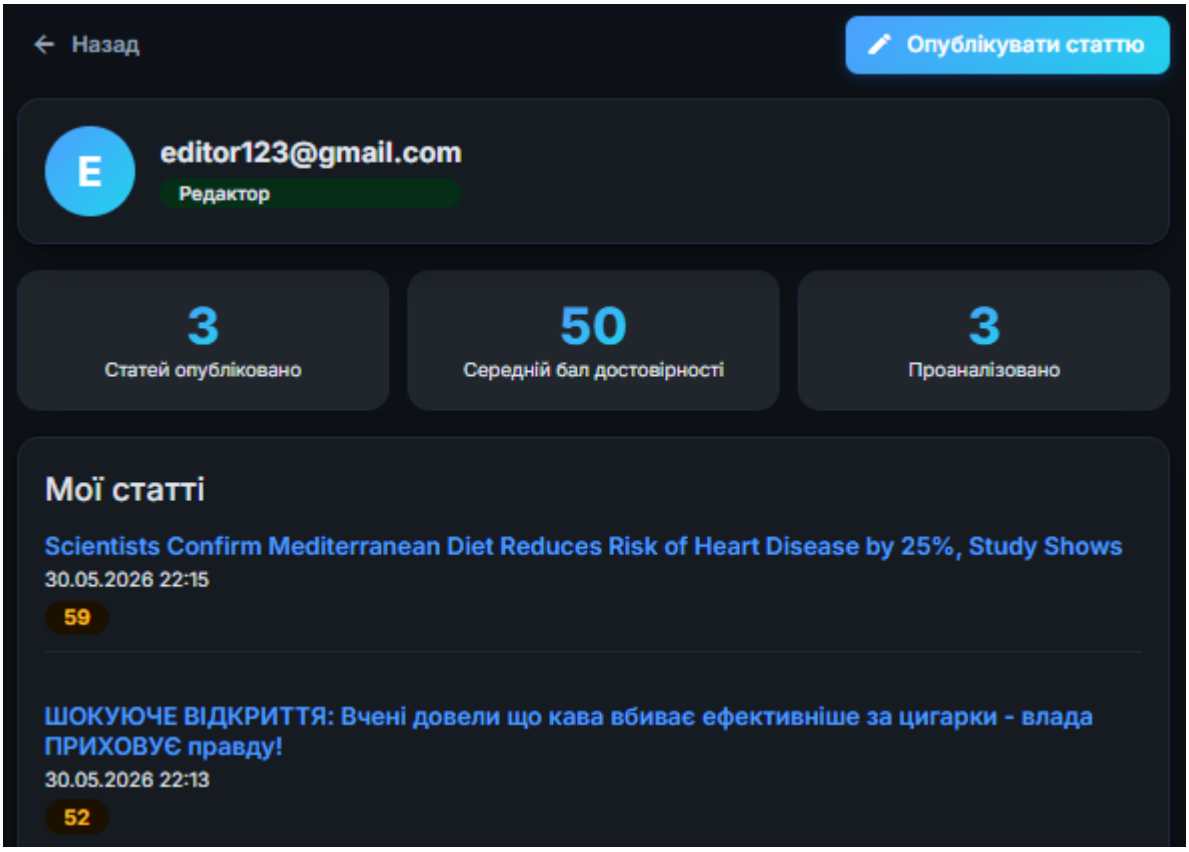

Рисунок 4.4 – Профіль редактора з переліком власних статей

Для адміністратора доступний повний набір функцій редактора, а також розширена панель адміністратора з чотирма секціями. Перша секція відповідає за управління користувачами. Адміністратор бачить таблицю всіх акаунтів і може змінювати роль кожного користувача через випадаючий список із варіантами user, editor та admin. Власний акаунт при цьому заблокований від редагування. Друга секція призначена для калібрування балів AI-ансамблю – тут налаштовуються ваги компонентів BERT-1, BERT-2, Евристики та FactCheck окремо для категорій новин і розважального контенту. Зміни відображаються у вигляді нормалізованої кольорової смуги, а натискання кнопки «Застосувати до всіх» перераховує бали наявних статей без повторного запуску моделей. Третя секція показує статус

сервісів: для кожного компонента системи відображається індикатор доступності та кількість статей, що очікують на аналіз. Четверта секція містить журнал аналізів із таблицею останніх 50 оброблених статей і розбивкою балів по кожному компоненту окремо. Крім панелі, зі стрічки новин адміністратор може видаляти будь-які статті, імпортувати нові матеріали з NewsAPI за обраною категорією та запускати повторний аналіз усіх наявних публікацій.

4.2 Тестування та оцінка роботи системи

Тестування системи проводилося за двома напрямами: перевірка коректності роботи основних функціональних модулів та оцінка якості AI-модуля виявлення недостовірної інформації.

Функціональне тестування виконувалося методом ручного тестування шляхом послідовного відтворення типових сценаріїв використання системи. Результати наведено у табл. 4.1.

Таблиця 4.1 – Результати функціонального тестування

№	Сценарій	Очікуваний результат	Результат
1	Реєстрація нового користувача	Акаунт створено, виконано автоматичний вхід	Пройдено
2	Вхід із некоректним паролем	Повідомлення про помилку автентифікації	Пройдено
3	Публікація статті редактором	Стаття збережена, AI-аналіз запущено у фоні	Пройдено
4	Відображення результату аналізу через WebSocket	Бал з’являється без перезавантаження сторінки	Пройдено
5	Фільтрація стрічки за категорією	Відображаються лише статті обраної категорії	Пройдено
6	Текстовий пошук за заголовком	Повертаються статті з відповідним текстом	Пройдено
7	Додавання статті до закладок	Стаття відображається на сторінці закладок	Пройдено
8	Реакція «Достовірно» / «Недостовірно»	Лічильник оновлюється, повторне натискання скасовує	Пройдено
9	Зміна ролі користувача адміністратором	Роль змінено, нові права застосовано	Пройдено
10	Видалення статті адміністратором	Стаття видалена зі стрічки та бази даних	Пройдено

Для оцінки роботи AI-модуля було проведено тестування на трьох статтях із різними характеристиками тексту. Перший тест – стаття з явними ознаками недостовірності: сенсаційний заголовок із великими літерами, анонімні «вчені», відсутність реальних джерел, заклики до термінового поширення та теорії змови. Результат аналізу першої тестової статті зображено на рис. 4.5.

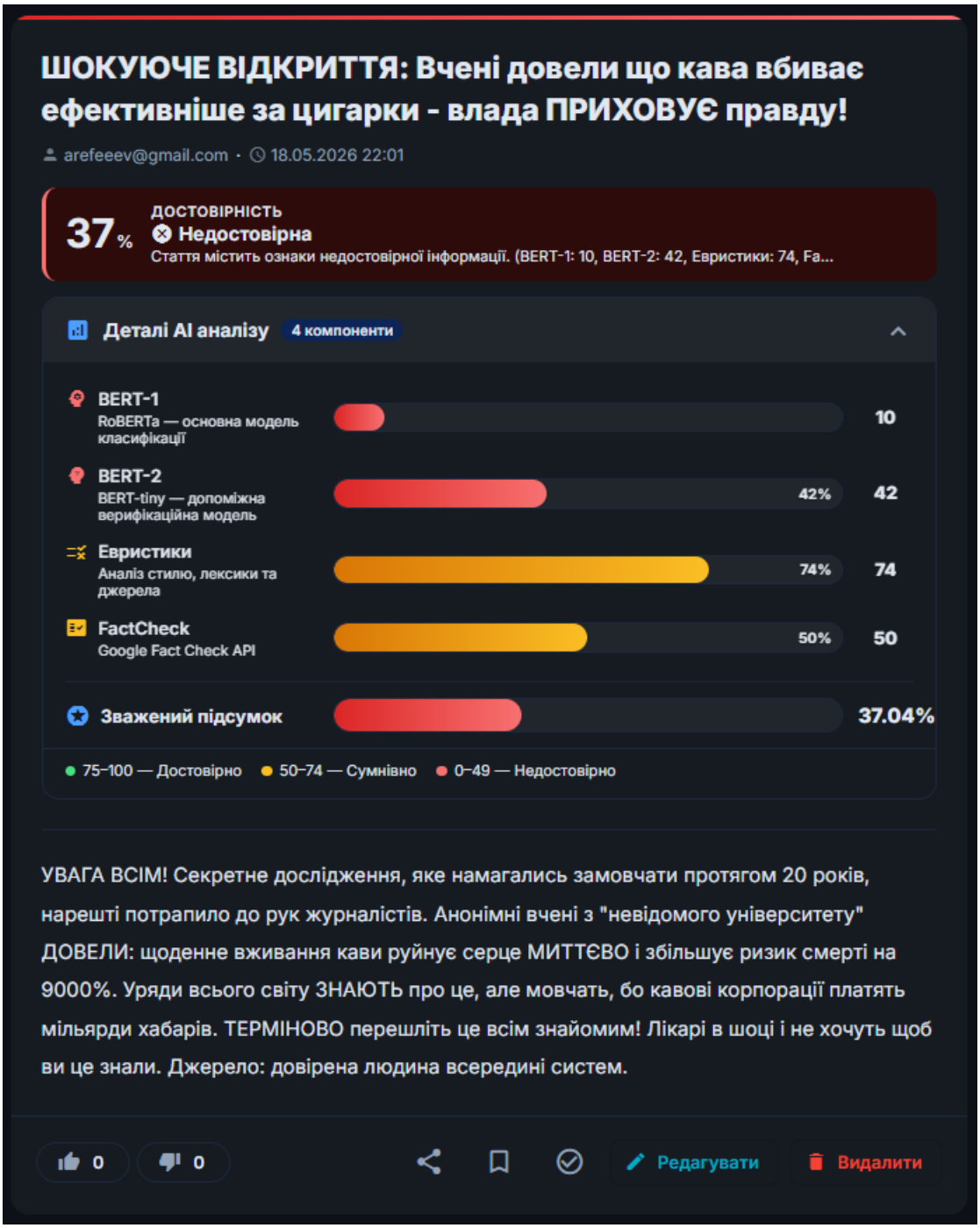

Рисунок 4.5 – Результат аналізу фейкової статті

Система присвоїла статті бал 37% з вердиктом «Недостовірна». Евристичний компонент визначив сенсаційний стиль і відсутність джерел (74 бали), BERT-1 та BERT-2 також зафіксували ознаки маніпулятивного тексту (10 та 42 бали відповідно). Результат свідчить про коректне спрацювання системи для очевидного фейкового контенту.

Другий тест – це якісна фактологічна стаття про регуляторні рішення Євросоюзу в сфері кібербезпеки, написана нейтральним стилем із посиланням на офіційні джерела. Система присвоїла їй 38% з вердиктом «Недостовірна», що є хибно негативним результатом. Аналіз компонентів показує: евристики коректно оцінили якість тексту (88 балів), проте BERT-1 та BERT-2 видали низькі значення (14 та 28 балів відповідно). Це пояснюється тим, що обидві моделі навчені переважно на англомовних датасетах політичних фейків і не адаптовані до україномовного фактологічного контенту регуляторної тематики – фраза з офіційним документом ЄС не потрапляє до патернів, які моделі асоціюють із достовірністю. Результат аналізу другої тестової статті зображено на рис. 4.6.

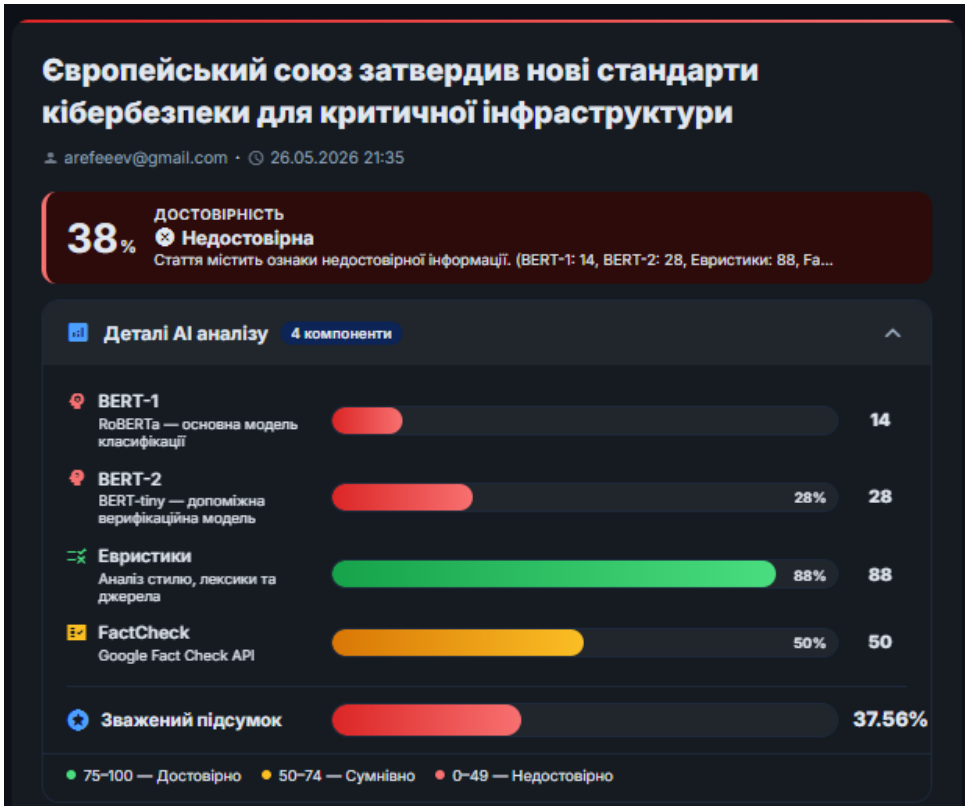

Рисунок 4.6 – Результат аналізу достовірної статті про кібербезпеку

І останній третій тест про англомовний огляд відеогри категорії «Технології», імпортований через NewsAPI. Система присвоїла статті 68% з вердиктом «Сумнівна». Показово, що BERT-1 та BERT-2 суттєво розійшлись в оцінці (6 та 92 бали), що активувало логіку корекції розбіжності: при різниці понад 45 балів ансамбль застосовує зважене усереднення з пріоритетом вищого значення. Евристики підтвердили якість тексту (88 балів). Підсумковий бал 68% є прийнятним результатом для розважального контенту, достовірність якого складніше верифікувати зовнішніми джерелами. Результат аналізу третьої тестової статті зображено на рис. 4.7.

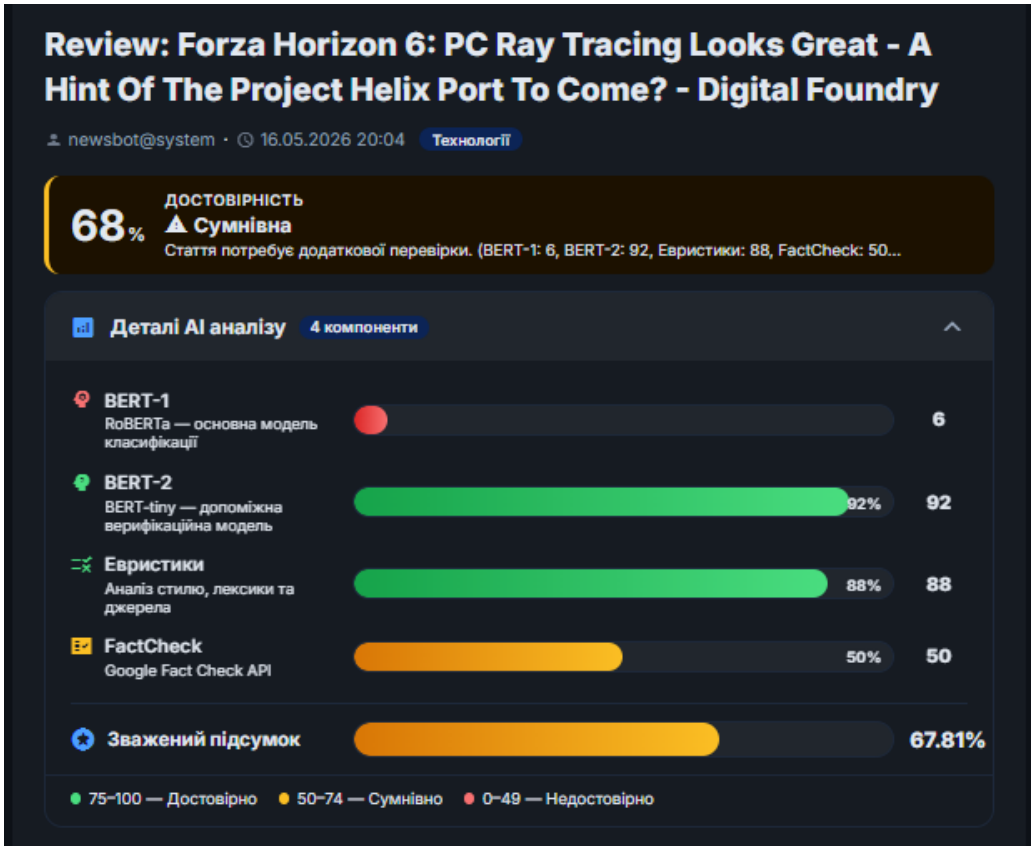

Рисунок 4.7 – Результат аналізу англомовного огляду відеогри

Проведене тестування виявило як сильні сторони системи, так і наявні обмеження. AI-модуль впевнено ідентифікує явний маніпулятивний контент із характерними ознаками фейку та коректно обробляє ситуації розбіжності між компонентами ансамблю. Водночас виявлено обмеження BERT-моделей щодо україномовних текстів нейтральної тематики – через специфіку тренувальних

даних моделі не завжди здатні коректно класифікувати якісний україномовний контент, що не відноситься до політичної сфери. Усунення цього обмеження потребує донавчання моделей на україномовних корпусах.

Висновки до розділу 4

У четвертому розділі розглянуто практичну експлуатацію та тестування розробленої вебплатформи новин. Описано роботу системи для чотирьох рівнів доступу. Незареєстрований відвідувач може переглядати стрічку новин, здійснювати пошук і фільтрацію матеріалів. Зареєстрований користувач отримує доступ до реакцій, закладок і персонального профілю. Редактор має можливість публікувати та редагувати власні статті з автоматичним запуском AI-аналізу. Адміністратор здійснює повне керування системою через спеціалізовану панель із функціями управління користувачами, калібрування ваг AI-ансамблю, моніторингу сервісів та перегляду журналу аналізів.

За результатами функціонального тестування всі десять перевірених сценаріїв виконані достовірно, що підтверджує відповідність реалізованих модулів визначеним вимогам. Тестування AI-модуля на трьох статтях різного характеру показало, що система впевнено виявляє явний маніпулятивний контент та правильно обробляє розбіжності між компонентами ансамблю. Разом із тим виявлено обмеження BERT-моделей щодо україномовних текстів нейтральної тематики, що зумовлено специфікою їх тренувальних даних і потребує донавчання на україномовних корпусах у подальшій роботі.

Загалом, проведене тестування підтверджує працездатність розробленої системи та достатній рівень точності AI-модуля для виявлення недостовірного контенту з вираженими ознаками маніпуляції. Отримані результати демонструють перспективність ансамблевого підходу до оцінки достовірності новин, а виявлені обмеження окреслюють напрями подальшого вдосконалення платформи.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено вебплатформу новин з інтегрованим AI-модулем для автоматичного виявлення недостовірної інформації. Вебплатформа надає наступний функціонал: публікацію, редагування, перегляд, оцінку та статистику аналізу новин. Здійснено верифікацію новинних матеріалів у реальному часі з автоматичним формуванням рейтингу достовірності.

Перший розділ роботи містить комплексний аналіз предметної сфери. Досліджено феномен недостовірної інформації, в результаті чого було встановлено, що в науковій літературі розрізняють три основні категорії: дезінформацію (свідоме введення в оману), міс-інформацію (ненавмисне поширення неточних даних) і мал-інформацію (маніпулятивне використання реальних фактів). Було встановлено, що неправдиві матеріали поширюються у соціальних мережах до шести разів швидше за правдиві. Згідно з даними Reuters Institute Digital News Report 2025, основними каналами дезінформації є Facebook (49% респондентів) та TikTok (48%). Для України проблема набуває особливої гостроти в умовах інформаційного протистояння. Кількість виявлених фейків зросла щонайменше у три рази порівняно з довоєнним періодом.

Проведено огляд та порівняльний аналіз п’яти провідних фактекінгових платформ (Snopes, FactCheck.org, Full Fact, PolitiFact, Stop Fake) та декількох академічних систем (ClaimBuster, FEVER) за тринадцятьма критеріями. Аналіз показав, що жодна розглянута платформа не поєднує в собі повноцінний новинний портал із вбудованим AI-модулем автоматичної верифікації, системою реакцій, автоматичним імпортом новин і прозорим поясненням результатів оцінювання. Спільним недоліком усіх аналогів є реактивний принцип роботи. Спростування публікується значно пізніше за фейк і охоплює набагато меншу аудиторію.

У другому розділі визначено і обґрунтовано методи, технології та архітектурні рішення. Вирішено реалізувати AI-модуль з ансамблевим підходом, який поєднує чотири компоненти. Перший та основний класифікатор RoBERTa

(навчений на датасеті FEVER), другий допоміжний класифікатор BERT-tiny з відмінним профілем помилок, евристичний аналізатор маркерів маніпуляції та Google Fact Check Tools API. Результати усіх компонентів агрегуються зваженим ансамблем у єдиний рейтинг достовірності від 0 до 100. Присутня підтримка калібрування ваг по тематичних категоріях. Технологічний стек системи побудований на Angular 17 та FastAPI з використанням PosgreSQL (Neon) як хмарної бази даних.

Третій розділ описує проєктування та реалізацію системи. Спроєктовано схему бази даних з сімома таблицями. На рівні серверної частини реалізовано JWT-аутентифікацію з трирівневою рольовою моделлю та адміністративну панель з журналом аналізу і управлінням ваговими коефіцієнтами. Клієнтська частина реалізована з використанням NgRx-управління станом та WebSocket-механізму для оновлення стрічки у реальному часі з автоматичним відновленням з’єднання. Ансамблевий AI-модуль виконує паралельний аналіз через asyncio.gather із середнім часом обробки 3–6 секунд. Автоматизований NewsAPI-пайплайн забезпечує безперервне поповнення платформи актуальним контентом.

Тестування та порядок експлуатації системи для кожної ролі користувачів проведено у четвертому розділі. Функціональне тестування охопило шість ключових сценаріїв – від реєстрації та автентифікації до публікації статей і адміністрування. Усі сценарії виконано успішно. Окремо проведено оцінку роботи AI-модуля. Було взято три типи контенту: маніпулятивний, нейтральний і розважальний. Результати підтвердили здатність системи розрізняти контент із характерними ознаками недостовірності. Водночас виявлено обмеження BERT-моделей на текстах нейтральної україномовної тематики.

Практичне значення роботи полягає у реалізації автоматичної верифікації контенту в момент публікації. Ансамблевий підхід підвищує точність оцінки за рахунок взаємної компенсації помилок компонентів. Калібрування вагових коефіцієнтів забезпечує гнучкість системи без необхідності повторного навчання моделей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Wardle C., Derakhshan H. Information disorder: Toward an interdisciplinary framework for research and policymaking. Council of Europe report. 2017. 107 p. URL: https://rm.coe.int/information-disorder-report/168076277c (дата звернення: 31.01.2026).

2. Vosoughi S., Roy D., Aral S. The spread of true and false news online. Science. 2018. Vol. 359, № 6380. P. 1146–1151. DOI: 10.1126/science.aap9559.

3. Wardle C. Fake news. It’s complicated. First Draft News. 2017. URL: https://firstdraftnews.org/articles/fake-news-complicated (дата звернення: 03.02.2026).

4. Tandoc E. C., Lim Z. W., Ling R. Defining "fake news": A typology of scholarly definitions. Digital Journalism. 2018. Vol. 6, № 2. P. 137–153. DOI: 10.1080/21670811.2017.1360143.

5. Детектор медіа. Моніторинг дезінформації в українському медіапросторі : звіт за 2023 рік. Київ, 2024. URL: https://detector.media (дата звернення: 03.02.2026).

6. EUvsDisinfo. Disinformation in the context of war: Annual report 2023. European External Action Service. 2024. URL: https://euvsdisinfo.eu/reports (дата звернення: 04.02.2026).

7. Zhou X., Zafarani R. A survey of fake news: Fundamental theories, detection methods, and opportunities. ACM Computing Surveys. 2020. Vol. 53, № 5. Article 109. DOI: 10.1145/3395046.

8. Vaswani A. et al. Attention is all you need. Advances in Neural Information Processing Systems. 2017. Vol. 30. P. 5998–6008. URL: https://arxiv.org/abs/1706.03762 (дата звернення: 12.02.2026).

9. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of NAACL-HLT 2019. Minneapolis, 2019. P. 4171–4186. URL: https://arxiv.org/abs/1810.04805 (дата звернення: 12.02.2026).

10. Silverman C., Alexander L. How teens in the Balkans are duping Trump supporters with fake news. BuzzFeed News. 2016. URL: https://www.buzzfeednews.com/article/craigsilverman/how-macedonia-became-a-global-hub-for-pro-trump-misinfo (дата звернення: 17.02.2026).

11. Newman N. et al. Reuters Institute Digital News Report 2025. Reuters Institute for the Study of Journalism. Oxford, 2025. URL: https://reutersinstitute.politics.ox.ac.uk/digital-news-report/2025 (дата звернення: 19.02.2026).

12. Full Fact. The State of Automated Fact Checking: report. London, 2016. URL: https://fullfact.org/automated (дата звернення: 23.02.2026).

13. Benkler Y., Faris R., Roberts H. Network propaganda: Manipulation, disinformation, and radicalization in American politics. Oxford University Press, 2018. 464 p. DOI: 10.1093/oso/9780190923624.001.0001.

14. Hassan N. et al. ClaimBuster: The first-ever end-to-end fact-checking system. Proceedings of the VLDB Endowment. 2017. Vol. 10, № 12. P. 1945–1948. DOI: 10.14778/3137765.3137815.

15. Arslan F. et al. A benchmark dataset of check-worthy factual claims. Proceedings of ICWSM 2020. 2020. Vol. 14. P. 821–829. DOI: 10.1609/icwsm.v14i1.7346.

16. Thorne J. et al. FEVER: A large-scale dataset for fact extraction and verification. Proceedings of NAACL-HLT 2018. New Orleans, 2018. P. 809–819. URL: https://arxiv.org/abs/1803.05355 (дата звернення: 23.02.2026).

17. Popat K. et al. DeClarE: Debunking fake news and false claims using evidence-aware deep learning. Proceedings of EMNLP 2018. Brussels, 2018. P. 22–32. DOI: 10.18653/v1/D18-1003.

18. Jones M. et al. JSON Web Token (JWT). RFC 7519. Internet Engineering Task Force (IETF). 2015. URL: https://www.rfc-editor.org/rfc/rfc7519 (дата звернення: 28.02.2026).

19. SQLAlchemy: документація: вебсайт. URL: https://docs.sqlalchemy.org (дата звернення: 02.03.2026).

20. Liu Y. et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint. 2019. URL: https://arxiv.org/abs/1907.11692 (дата звернення: 02.03.2026).

21. Sagi O., Rokach L. Ensemble learning: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2018. Vol. 8, № 4. Article e1249. DOI: 10.1002/widm.1249.

22. Kaliyar R. K. et al. FakeBERT: Fake news detection in social media with a BERT-based deep learning approach. Multimedia Tools and Applications. 2021. Vol. 80. P. 11765–11788. DOI: 10.1007/s11042-020-10183-2.

23. Thorne J. et al. FEVER: A large-scale dataset for fact extraction and verification. Proceedings of NAACL-HLT 2018. New Orleans, 2018. P. 809–819. URL: https://arxiv.org/abs/1803.05355 (дата звернення: 05.03.2026).

24. Python asyncio: документація: вебсайт. URL: https://docs.python.org/3/library/asyncio.html (дата звернення: 05.03.2026).

25. Hassan N. et al. ClaimBuster: The first-ever end-to-end fact-checking system. Proceedings of the VLDB Endowment. 2017. Vol. 10, № 12. P. 1945–1948. DOI: 10.14778/3137765.3137815.

26. Provos N., Mazieres D. A future-adaptable password scheme. Proceedings of the USENIX Annual Technical Conference. 1999. URL: https://www.usenix.org/legacy/events/usenix99/provos/provos.pdf (дата звернення: 10.03.2026).

27. PostgreSQL: документація: вебсайт. URL: https://www.postgresql.org/docs (дата звернення: 10.03.2026).

28. Neon: serverless PostgreSQL: вебсайт. URL: https://neon.tech/docs (дата звернення: 10.03.2026).

29. Angular: документація: вебсайт. URL: https://angular.dev/overview (дата звернення: 11.03.2026).

30. NgRx: документація: вебсайт. URL: https://ngrx.io/docs (дата звернення: 11.03.2026).

31. NewsAPI: документація: вебсайт. URL: https://newsapi.org/docs (дата звернення: 18.03.2026).

32. Google Fact Check Tools API: документація. URL: https://developers.google.com/fact-check/tools/api/reference/rest (дата звернення: 23.03.2026).

33. Fette I., Melnikov A. The WebSocket Protocol. RFC 6455. Internet Engineering Task Force (IETF). 2011. URL: https://www.rfc-editor.org/rfc/rfc6455 (дата звернення: 24.03.2026).

34. Wolf T. et al. HuggingFace’s Transformers: State-of-the-art natural language processing. Proceedings of EMNLP 2020: System Demonstrations. 2020. P. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6.

ДОДАТОК А

Код AI-ансамблю

```
    import asyncio
    from app.ai import bert, bert2, factcheck, heuristics
    from app.config import settings

    _ENTERTAINMENT_CATS = {"sports", "entertainment", "sport"}
    _NEWS_CATS = {"general", "technology", "science", "health", "business"}

    def _get_weights(category: str | None, has_fc: bool, bert2_ok: bool) ->
dict:
        is_entertainment = (category or "").lower() in _ENTERTAINMENT_CATS

        if is_entertainment:
            w = {"bert": 0.25, "bert2": 0.20, "heuristics": 0.35, "fc": 0.20}
        else:
            w = {"bert": 0.38, "bert2": 0.24, "heuristics": 0.18, "fc": 0.20}

        if not has_fc:
            extra = w["fc"]
            w["bert"] += extra * 0.55
            w["heuristics"] += extra * 0.45
            w["fc"] = 0.0

        if not bert2_ok:
            extra = w["bert2"]
            w["bert"] += extra * 0.60
            w["heuristics"] += extra * 0.40
            w["bert2"] = 0.0

        return w

    def _adjust_bert_scores(b1: float, b2: float, bert2_ok: bool) ->
tuple[float, float]:
        if not bert2_ok:
```

```
        return b1, b1

    divergence = abs(b1 - b2)
    avg = (b1 + b2) / 2

    if avg < 20 and divergence < 30:
        return 50.0, 50.0

    if divergence > 45:
        higher = max(b1, b2)
        lower = min(b1, b2)
        adjusted = round(higher * 0.72 + lower * 0.28, 2)
        return adjusted, adjusted

    return b1, b2

async def analyze_article(
    title: str,
    body: str,
    source_url: str | None = None,
    category: str | None = None,
) -> dict:
    text = f"{title}\n\n{body}"

    bert_score, bert2_score, fc_score = await asyncio.gather(
        bert.classify(text),
        bert2.classify(text),
        factcheck.analyze(text),
    )
    heuristic_score = heuristics.analyze(text, source_url)

    has_fc = bool(settings.google_factcheck_api_key)
    bert2_ok = bert2.is_available()

    bert_raw, bert2_raw = bert_score, bert2_score
```

```
        bert_adj, bert2_adj = _adjust_bert_scores(bert_score, bert2_score,
bert2_ok)

        w = _get_weights(category, has_fc, bert2_ok)
        score = round(
            bert_adj        * w["bert"] +
            bert2_adj       * w["bert2"] +
            heuristic_score * w["heuristics"] +
            fc_score        * w["fc"],
            2,
        )

        is_entertainment = (category or "").lower() in _ENTERTAINMENT_CATS
        if score >= 75:
            verdict = "Стаття виглядає достовірною"
        elif score >= 50:
            verdict = "Стаття потребує додаткової перевірки"
        else:
            verdict = "Стаття містить ознаки недостовірної інформації"
        if is_entertainment:
            verdict += ". Для розважального контенту аналіз оцінює стиль та
якість тексту"
        parts = [f"BERT-1: {bert_raw:.0f}"]
        if bert2_ok:
            parts.append(f"BERT-2: {bert2_raw:.0f}")
        parts.append(f"Евристики: {heuristic_score:.0f}")
        if has_fc:
            parts.append(f"FactCheck: {fc_score:.0f}")
        return {
            "score": score,
            "bert_score": bert_score,
            "bert2_score": bert2_score if bert2_ok else None,
            "claimbuster_score": round(heuristic_score, 2),
            "factcheck_score": fc_score if has_fc else None,
            "explanation": f"{verdict}. ({', '.join(parts)})",
        }
```

ДОДАТОК Б

Маршрути API статей

```
import uuid
import os
from fastapi import APIRouter, Depends, HTTPException, BackgroundTasks, UploadFile, File
from sqlalchemy.ext.asyncio import AsyncSession
from app.database import get_db
from app.auth.dependencies import get_current_user, require_editor, require_admin, get_optional_user
from app.models import User
from app.articles.schemas import ArticleCreate, ArticleUpdate, ArticleOut, ArticlesPage, AiScoreOut, ReactionsOut
from app.articles import service

router = APIRouter(prefix="/articles", tags=["articles"])

_UPLOAD_DIR = os.path.normpath(os.path.join(os.path.dirname(__file__), "..", "..", "static", "uploads"))
_ALLOWED_TYPES = {"image/jpeg", "image/png", "image/webp", "image/gif"}
_MAX_SIZE = 5 * 1024 * 1024

def _serialize(article, likes: int = 0, dislikes: int = 0) -> ArticleOut:
    ai = AiScoreOut.from_orm_obj(article.ai_score) if article.ai_score else None
    return ArticleOut(
        id=article.id,
        title=article.title,
        body=article.body,
        imageUrl=article.image_url,
        sourceUrl=article.source_url,
        category=article.category,
        status=article.status.value,
        createdAt=article.created_at,
        author=article.author,
```

```
            aiScore=ai,
            likes=likes,
            dislikes=dislikes,
        )


    @router.get("", response_model=ArticlesPage)
    async def list_articles(
        page: int = 1,
        limit: int = 20,
        search: str | None = None,
        category: str | None = None,
        min_score: float | None = None,
        max_score: float | None = None,
        db: AsyncSession = Depends(get_db),
    ):
        articles, total = await service.get_articles(
            page, limit, db, search=search, category=category,
            min_score=min_score, max_score=max_score,
        )
        reactions = await service.get_batch_reactions([a.id for a in articles],
db)
        return ArticlesPage(
            data=[_serialize(a,
                             likes=reactions.get(a.id, {}).get("like", 0),
                             dislikes=reactions.get(a.id, {}).get("dislike", 0))
                  for a in articles],
            total=total,
        )


    @router.post("/upload-image")
    async def upload_image(
        file: UploadFile = File(...),
        _: User = Depends(require_editor),
    ):
        if file.content_type not in _ALLOWED_TYPES:
```

```
            raise HTTPException(status_code=400, detail="Дозволені формати:
JPEG, PNG, WebP, GIF")
        data = await file.read()
        if len(data) > _MAX_SIZE:
            raise HTTPException(status_code=400, detail="Файл занадто великий
(макс. 5 МБ)")
        ext = file.filename.rsplit(".", 1)[-1].lower() if file.filename and "."
in file.filename else "jpg"
        filename = f"{uuid.uuid4().hex}.{ext}"
        os.makedirs(_UPLOAD_DIR, exist_ok=True)
        with open(os.path.join(_UPLOAD_DIR, filename), "wb") as f:
            f.write(data)
        return {"url": f"/static/uploads/{filename}"}

    @router.post("", response_model=ArticleOut, status_code=201)
    async def create_article(
        body: ArticleCreate,
        background_tasks: BackgroundTasks,
        db: AsyncSession = Depends(get_db),
        current_user: User = Depends(require_editor),
    ):
        article = await service.create_article(body.title, body.body,
current_user, db, image_url=body.image_url)
        background_tasks.add_task(
            service.run_ai_analysis,
            article.id, article.title, article.body,
            article.source_url, article.category,
        )
        return _serialize(article)

    @router.get("/my", response_model=list[ArticleOut])
    async def my_articles(
        db: AsyncSession = Depends(get_db),
        current_user: User = Depends(get_current_user),
    ):
        articles = await service.get_my_articles(current_user.id, db)
```

```
        return [_serialize(a) for a in articles]


    @router.get("/stats")
    async def stats(db: AsyncSession = Depends(get_db)):
        return await service.get_stats(db)


    @router.get("/{article_id}/related", response_model=list[ArticleOut])
    async def get_related(article_id: int, db: AsyncSession = Depends(get_db)):
        articles = await service.get_related_articles(article_id, db)
        reactions = await service.get_batch_reactions([a.id for a in articles],
db)
        return [
            _serialize(a,
                       likes=reactions.get(a.id, {}).get("like", 0),
                       dislikes=reactions.get(a.id, {}).get("dislike", 0))
            for a in articles
        ]


    @router.get("/{article_id}", response_model=ArticleOut)
    async def get_article(article_id: int, db: AsyncSession = Depends(get_db)):
        article = await service.get_article(article_id, db)
        if not article:
            raise HTTPException(status_code=404, detail="Article not found")
        return _serialize(article)


    @router.delete("/{article_id}", status_code=204)
    async def delete_article(
        article_id: int,
        db: AsyncSession = Depends(get_db),
        _: User = Depends(require_admin),
    ):
        deleted = await service.delete_article(article_id, db)
        if not deleted:
            raise HTTPException(status_code=404, detail="Article not found")


    @router.get("/{article_id}/reactions", response_model=ReactionsOut)
```

```
    async def get_reactions(
        article_id: int,
        db: AsyncSession = Depends(get_db),
        current_user: User | None = Depends(get_optional_user),
    ):
        return await service.get_reactions(article_id, db,
user_id=current_user.id if current_user else None)

    @router.post("/{article_id}/react", response_model=ReactionsOut)
    async def react(
        article_id: int,
        body: dict,
        db: AsyncSession = Depends(get_db),
        current_user: User = Depends(get_current_user),
    ):
        reaction = body.get("reaction")
        if reaction not in ("like", "dislike"):
            raise HTTPException(status_code=400, detail="reaction must be 'like'
or 'dislike'")
        article = await service.get_article(article_id, db)
        if not article:
            raise HTTPException(status_code=404, detail="Article not found")
        return await service.toggle_reaction(article_id, current_user.id,
reaction, db)

    @router.patch("/{article_id}", response_model=ArticleOut)
    async def update_article(
        article_id: int,
        body: ArticleUpdate,
        background_tasks: BackgroundTasks,
        db: AsyncSession = Depends(get_db),
        current_user: User = Depends(require_editor),
    ):
        article = await service.update_article(
            article_id, body.title, body.body, body.image_url,
            current_user.id, db, background_tasks,
```

```
        )
        if not article:
            raise HTTPException(status_code=404, detail="Article not found or
access denied")
        return _serialize(article)


    @router.post("/reanalyze-all")
    async def reanalyze_all(
        background_tasks: BackgroundTasks,
        db: AsyncSession = Depends(get_db),
        _: User = Depends(require_admin),
    ):
        count = await service.reset_and_reanalyze_all(db, background_tasks)
        return {"queued": count, "message": f"Повторний аналіз запущено для
{count} статей"}
```

ДОДАТОК В

NgRx ефекти стрічки новин

```
import { Injectable } from '@angular/core';
import { Actions, createEffect, ofType } from '@ngrx/effects';
import { Router } from '@angular/router';
import { catchError, exhaustMap, map, mergeMap, switchMap, tap } from
'rxjs/operators';
import { of } from 'rxjs';
import { ArticlesActions } from './articles.actions';
import { ArticlesService } from '../../core/services/articles.service';
import { WebsocketService } from '../../core/services/websocket.service';

@Injectable()
export class ArticlesEffects {
  loadFeed$ = createEffect(() =>
    this.actions$.pipe(
      ofType(ArticlesActions.loadFeed),
      switchMap(({ page, search, category, minScore, maxScore }) =>
        this.articlesService.getAll(page, 20, search, category, minScore,
maxScore).pipe(
          map((res) => ArticlesActions.loadFeedSuccess({ articles: res.data,
total: res.total })),
          catchError((err) => of(ArticlesActions.loadFeedFailure({ error:
err.message }))),
        ),
      ),
    ),
  );

  loadArticle$ = createEffect(() =>
    this.actions$.pipe(
      ofType(ArticlesActions.loadArticle),
      switchMap(({ id }) =>
        this.articlesService.getOne(id).pipe(
          map((article) => ArticlesActions.loadArticleSuccess({ article })),
```

```
          catchError((err) => of(ArticlesActions.loadArticleFailure({ error:
err.message }))),
        ),
      ),
    ),
  );

  publish$ = createEffect(() =>
    this.actions$.pipe(
      ofType(ArticlesActions.publishArticle),
      switchMap(({ title, body, imageUrl }) =>
        this.articlesService.create(title, body, imageUrl).pipe(
          map((article) => ArticlesActions.publishArticleSuccess({ article })),
          catchError((err) => of(ArticlesActions.publishArticleFailure({ error:
err.error?.message ?? 'Publish failed' }))),
        ),
      ),
    ),
  );
  redirectAfterPublish$ = createEffect(
    () =>
      this.actions$.pipe(
        ofType(ArticlesActions.publishArticleSuccess),
        tap(({ article }) => this.router.navigate(['/articles', article.id])),
      ),
    { dispatch: false },
  );
  loadFeedMore$ = createEffect(() =>
    this.actions$.pipe(
      ofType(ArticlesActions.loadFeedMore),
      exhaustMap(({ page, search, category, minScore, maxScore }) =>
        this.articlesService.getAll(page, 20, search, category, minScore,
maxScore).pipe(
          map((res) => ArticlesActions.loadFeedMoreSuccess({ articles: res.data,
total: res.total })),
```

```
          catchError((err) => of(ArticlesActions.loadFeedMoreFailure({ error:
err.message }))),
        ),
      ),
    ),
  );
  deleteArticle$ = createEffect(() =>
    this.actions$.pipe(
      ofType(ArticlesActions.deleteArticle),
      switchMap(({ id }) =>
        this.articlesService.delete(id).pipe(
          map(() => ArticlesActions.deleteArticleSuccess({ id })),
          catchError((err) => of(ArticlesActions.deleteArticleFailure({ error:
err.message }))),
        ),
      ),
    ),
  );
  feedScores$ = createEffect(() =>
    this.wsService.onFeedUpdates().pipe(
      mergeMap(({ articleId, score, explanation }) =>
        of(
          ArticlesActions.scoreUpdated({ articleId, score, explanation }),
          ArticlesActions.showScoreToast({ articleId, score }),
        ),
      ),
      catchError(() => of({ type: '[Articles] WS Feed Error' })),
    ),
  );
  constructor(
    private actions$: Actions,
    private articlesService: ArticlesService,
    private wsService: WebsocketService,
    private router: Router,
  ) {}
}
```