Пояснювальна записка
до дипломного проєкту

на тему: Вебзастосунок для агрегації та інтелектуального аналізу новинного контенту

КПІ.ІП-1130.045440.02.91

Київ – 2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application programming interface, прикладний програмний Інтерфейс.
SDK	–	Software development kit.
IT	–	Інформаційні технології.
ER	–	Entity-Relation diagram.
ОС	–	Операційна система.
БД	–	База даних.
LLM	–	Велика мовна модель
RSS	–	Вебстрічка новин
NLP	–	Обробка природньої мови
GPT	–	Генеративний натренований трансформер
ORM	–	Об’єктно-реляційна модель

ВСТУП

У сучасному суспільстві 21 століття питання надійності новинного контенту набуло особливої актуальності. Збільшення кількості фейків у щоденному інформаційному просторі становить серйозну загрозу для країн та її громадян, особливо в умовах збройного конфлікту[1]. В Україні, де інфопростір зазнає значного тиску внаслідок війни, потреба в засобах для перевірки фактів та аналізу контенту сильно зросла [2].

Можливі розв’язки таких задач включають використання:

- алгоритми глибоких нейронних мереж, зокрема LLM для аналізу оцінки статей за метриками якості а також автоматизованої категоризації текстів для пошуку закономірностей;
- семантичний пошук з використанням засобів векторизації тексту для пошуку схожих за значенням текстів [3];
- використання розслідувань і статей від відомих і визнаних організацій факт-чекерів, наприклад VoxUkraine[4] & StopFake[5].

Задача цього проекту - створення вебзастосунку для агрегації та інтелектуального аналізу новинного контенту, який інтегруватиме сучасні технології штучного інтелекту для автоматизованої оцінки якості новин за критеріями достовірності , фактичності, емоційності та сенсаційності.

Метою проекту є покращення доступу користувачів до достовірної інформації шляхом створення системи автоматизованого контролю якості новинного контенту з використанням великих мовних моделей і аналізом природної мови.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання дипломного проєктування

Даний проєкт націлений на використання сучасних засобів штучного інтелекту та алгоритмів обробки природної мови для аналізу новинних статей від декількох джерел новинних статей, зокрема від таких новинних видань: espreso.tv[6], hromadske[7], ukrinform[8], Радіо «Свобода»[9]. Збір контенту можна виконувати за допомогою багаточисельних технологій вебскрейпінгу або API новин.

Використання вище зазначених технологій відкриває наступні можливості у сфері аналізу новинного контенту:

– оцінка якості новинних статей за деякими метриками;
– фільтрування статей за вищевказаними метриками або за видавниками статей;
– категоризація розслідувань від факт-чекінгових організацій для визначення трендів дезінформації;
– пошук схожих статей за тематикою і сенсом;
– створення звітів про дезінформацію за деякий проміжок часу.

Реалізація даних завдань надасть можливість створити зрозумілий і простий інтерфейс для користувачів для доступу до цього функціоналу.

1.2 Аналіз предметної області

За даними міжнародної PR-кампанії Edelman у її розслідуванні “Edelman Trust Barometer” 2023 року, в Об’єднаному Королівстві лише 37% громадян довіряють традиційним медіа [10], що свідчить про кризу довіри до журналістів.

Також варто зазначити, що український медіа-простір все більше заповнюється фейками. З найвидатніших – це підробка фактів про «біо-лабораторії США» [11], дезінформація про події в Бучі в 2022 році [12], і також

не можна не згадати багаточисельні фейки про пошкодження інфраструктури через масовані російські обстріли [13].

Через такі кампанії зростає попит не тільки на якісну журналістику, а й на незалежних факт-чекерів, та можливість розрізняти якісну журналістику від неякісної.

Тим не менш, в інтернеті існує велика кількість онлайн-сервісів індивідуальних видань, де користувачі можуть подивитися їхні публікації, проте їхня проблема полягає у відстутності інтеграції з сучасними технологіями для аналізу тексту. Також велика кількість подібних сервісів мають суб'єктивні погляди і намагаються проштовхнути власні корисні для них наративи.

За останні роки також почали з'являтися агрегатори новин з метою подолання проблем, які виникають при обмеженні сприйняття новинного контенту від лише індивідуальних організацій.

1.3 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації моєї власної розробки вебзастосунку «NewsCheck». Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

1.3.1 Аналіз відомих програмних продуктів

Проведемо аналіз існуючих агрегаторів новин:

– Google News [14]: агрегатор новин від корпорації Google. Спеціалізується на агрегації новинного контенту від великої кількості організацій, та використовує кластеризацію для пошуку схожих статей;
– Ground News [15]: агрегатор новин, який використовує інноватівний підхід для показу різних поглядів на одну подію від декількох видань;

– NewsGuard [16]: агрегатор новин, який використовує експертний підхід до оцінки достовірності сайтів медіа.

Для порівняння проєкту з аналогом можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогами

Функціонал	News Check	Ground News	Google News	News Guard	Пояснення
Українські джерела новин	+	+-	+	+-	Google News підтримує українські джерела, Ground News лише висвітлює події України від міжнародних медіа, а NewsGuard має рейтинг для малої кількості українських медіа
Використання LLM для аналізу новин	+	-	-	-	Жодний з сервісів не використовує LLM для аналізу новин
Пошук схожих статей за семантичним пошуком	+	+-	+-	-	Google News і Ground News мають базовий пошук по ключовим словам і тематиці.
Інтеграція факт-чекерів	+	-	-	-	Жодний з сервісів не мають інтеграції з факт-чекерами
Опис дезінормаційних кампаній	+	-	-	-	Жодний з сервісів не має опису дезінформаційних кампаній

1.3.2 Аналіз відомих алгоритмічних та технічних рішень

Для виконання функціональних вимог проєкту, варто проаналізувати технічні і алгоритмічні рішення, які є доступними.

а) збір новинного контенту

Для збору новинного контенту, є наступні опції:

1) статичний скрапінг: використання бібліотек для отримання вихідного коду вебсторінки для подальшого збору;
2) динамічний скрапінг: використання вебкролерів, тобто емуляції браузера, для можливості взаємодії з JavaScript для кращої взаємодії зі сторінкою;
3) гібридний підхід: використання обох попередніх підходів в залежності від сайту.

Було прийнято рішення обрати підхід с.

б) алгоритми векторизації тексту:

1) використання власної невеликої моделі, наприклад TF-IDF. Є швидкодійним варіантом, проте потрібно проводити тренування;
2) використання більшої моделі, розгорнутої локально, наприклад BERT. Є кращим варіантом, але потребує великих обчислювальних ресурсів;
3) використання моделей для ембедингу на основі GPT. Має найкращі семантичні здібності, але потребує колосальних обчислювальних ресурсів, або використання API компаній VoyageAI або OpenAI, що збільшує витрати.

Було прийнято рішення обрати варіант с. з використанням API VoyageAI;

в) алгоритми категоризації статей факт-чекерів:

1) локальна модель машинного навчання або невелика нейронна мережа. Ефективний варіант, але потребує

тренування, і також більше вразливий до зроблення помилок;

2) надсилання запитів для категоризації до LLM, наприклад OpenAI, що збільшує витрати.

Було прийнято рішення використати варіант b;

г) оцінка статті за метриками якості:

було прийнято рішення використати запити до LLM від OpenAI;

д) зберігання векторів:

1) налаштування бази даних векторів, наприклад Milvus, з локальним розгортанням;

2) використання хмарного серверного векторного сховища, наприклад Pinecone.

Враховуючи простоту локального розгортання та зручність використання, було обрано Milvus.

1.4 Аналіз та моделювання бізнес-процесів

Для моделювання бізнес-процесу використовується BPMN модель (рисунок 1.1, рисунок 1.2, рисунок 1.3):

Опис послідовності аналізу статті новин:

а) відбувається виклик завдання на аналіз статей;

б) дані збираються методом гібридного вебскрейпінгу;

в) дані зберігаються у базі даних;

г) надсилається запит до LLM для оцінки статті;

д) метрики оцінки зберігаються у базі даних;

е) текст статті відправляється на векторизацію;

ж) вектор статті зберігається у векторній базі даних;

з) виконується пошук схожих статей за допомогою семантичного пошуку;

і) результати пошуку зберігаються у базі даних.

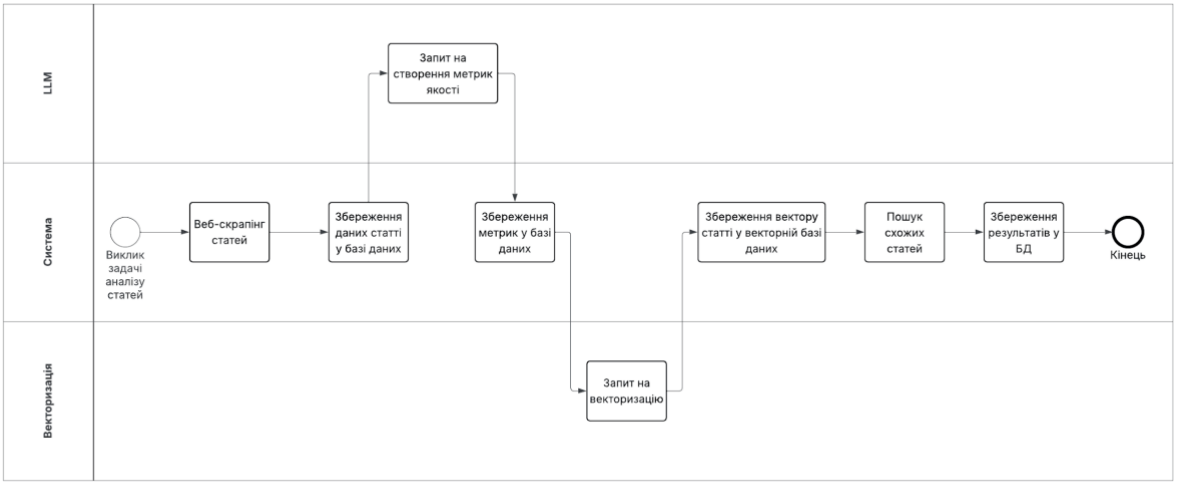


Рисунок 1.1 – Модель бізнес-процесу для збору і аналізу новинних статей

Опис послідовності аналізу розслідувань:

а) викликається задача аналізу розслідувань;

б) проводиться збір методом вебскрапінгу;

в) дані розслідування зберігаються у базі даних;

г) відправляється запит до LLM для категоризації статті;

д) результати аналізу зберігаються у базі даних.

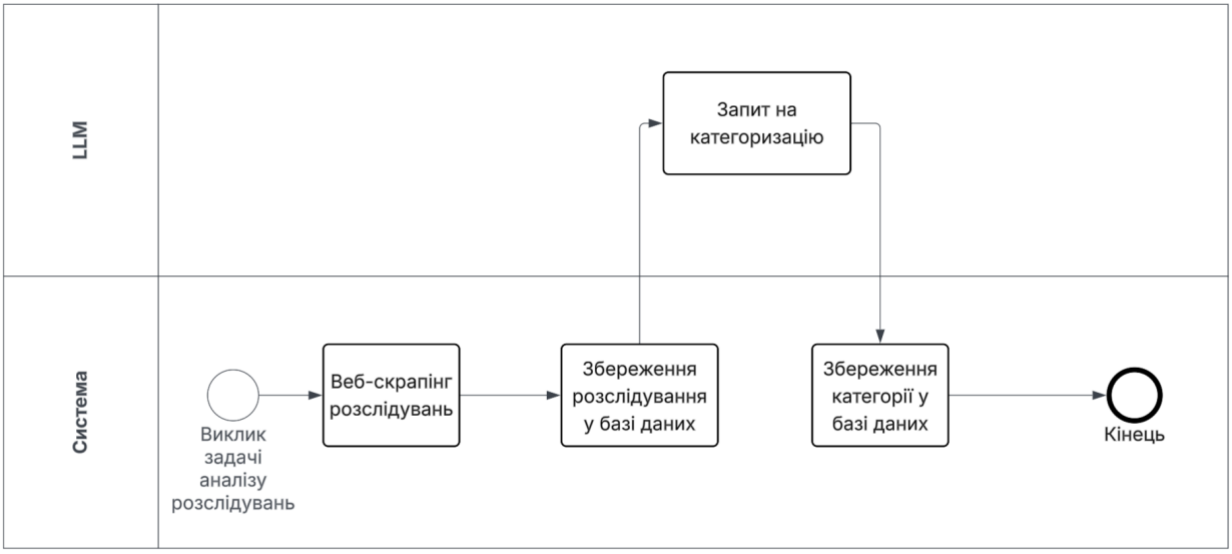


Рисунок 1.2 – Модель бізнес процесу для збору і аналізу розслідувань

Опис послідовності створення щотижневої статистики і дайджесту про дезінформацію:

а) викликається задача для щотижневого аналізу;

б) з бази даних агрегуються усі розслідування за минулий тиждень;

в) підраховуються усі кількість розслідувань для кожної категорії для статистики;

г) до LLM надсилається запит для створення дайджесту;

д) створений дайджест і статистика зберігаються у базі даних.

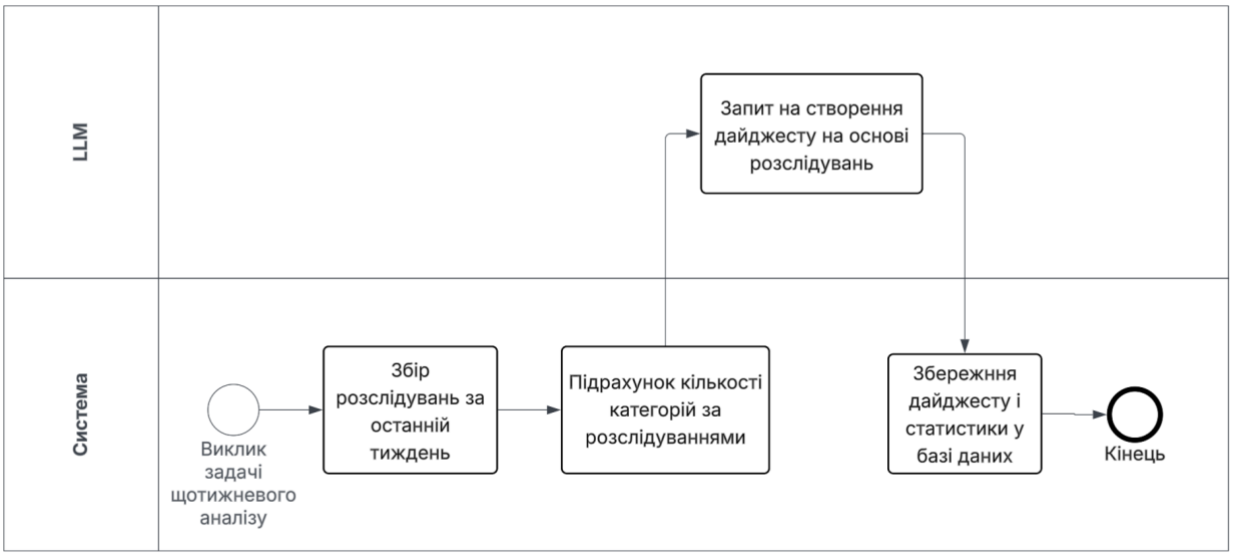


Рисунок 1.3 – Модель бізнес-процесу для щотижнеового аналізу дезінформації

Висновки до розділу

У цьому розділі було виконане детальне обстеження предметної області.

У першому підрозділі було наведено аргументацію щодо актуальності розробки цього проєкту, наведено технології, якими можна втілити функціонал проєкту.

У другому підрозділі було проведено ретельний аналіз предметної області, було наведено загальну теорію і контекст, що оточує дану предметну область. Було також представлено проблеми, з якими стикаються сучасні сервіси для представлення новинного контенту.

На початку третього підрозділу, було проведено аналіз існуючих рішень та існуючих програмних продуктів, складено таблицю для порівняння функціоналу власного проєкту і вище вказаних програмних продуктів. По результатам таблиці можна стверджувати, що функціонал і технології, які використовуються у проєкті є дійсно інноваційними. Далі було проведено аналіз існуючих технологій та алгоритмів, які можна було б використовувати для втілення програмного забезпечення проєкту.

У четвертому підрозділі було наведено опис бізнес-процесів, а також було створено нотації моделей бізнес-процесів за допомогою діаграм BPMN. Було узагальнено мету і цілі програмного забезпечення.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Головним функціоналом застосунку є здатність користувача переглядати новини, розслідування, і щотижневі дайджести і статистику.

Діаграма варіантів використання з ключовими акторами та основними функціями показана на рисунку 2.1:

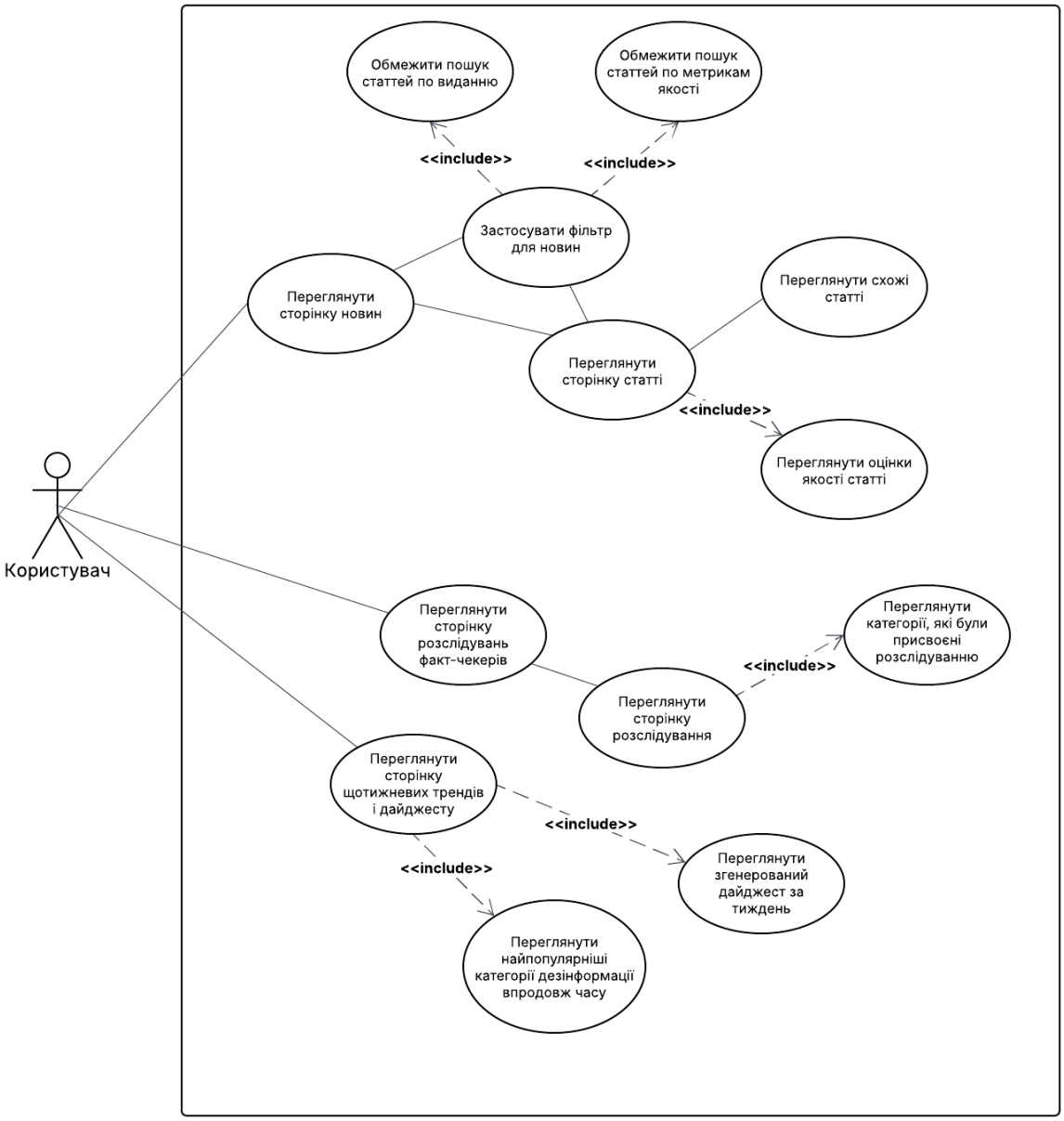


Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.1-2.13 наведено опис варіантів використання програмного забезпечення.

Таблиця 2.1 – Варіант використання UC-01

Use case name	Перехід на сторінку новин
Use case ID	UC-01
Goals	Користувач бачить список статей новин
Actors	Користувач
Trigger	Користувач натискає на кнопку «Новини»
Pre-conditions	-
Flow of Events	Користувач заходить на сторінку зі списком новин. Система відображає список статей, які позначені заголовком, виданням, та часом публікації. Статті відсортовані у порядку спадання за часом публікації. Система також надає можливість фільтрувати статті.
Extension	-
Post-Condition	Користувач знаходиться на сторінці новин

Таблиця 2.2 – Варіант використання UC-02

Use case name	Застосування фільтру
Use case ID	UC-02
Goals	Список статей обмежується за параметрами, виставленими користувачем
Actors	Користувач
Trigger	Користувач натискає на позначку фільтра на сторінці новин
Pre-conditions	Користувач знаходиться на сторінці новин
Flow of Events	Користувач натискає на позначку фільтра на сторінці новин. Відкривається вікно фільтра. Користувач може закрити вікно фільтра, виставити параметри пошуку за замовченням або зберегти налаштування.
Extension	-
Post-Condition	Користувач, після налаштування фільтра, повертається на сторінку новин

Таблиця 2.3 – Варіант використання UC-03

Use case name	Обмежити пошук статей по виданню
Use case ID	UC-03
Goals	Користувач може бачити на сторінці новин статті тільки від тих видань, від яких користувач бажає
Actors	Користувач
Trigger	Користувач натискає на позначку фільтра на сторінці новин
Pre-conditions	Користувач знаходиться на сторінці фільтра
Flow of Events	Знаходячись на сторінці фільтра, користувач може обирати за допомогою check-box видання, від яких він бажає бачити статті.
Extension	-
Post-Condition	Користувач залишається на сторінці фільтра.

Таблиця 2.4 – Варіант використання UC-04

Use case name	Обмежити пошук статей по оцінкам якості
Use case ID	UC-04
Goals	Користувач бачить на сторінці новин тільки ті новини, які підпадають по параметрам якості виставленим користувачем у фільтрі
Actors	Користувач
Trigger	Користувач натискає на позначку фільтра на сторінці новин
Pre-conditions	Користувач знаходиться на сторінці фільтра
Flow of Events	Користувач, у сторінці фільтра може використати такі елементи як check-box та range slider для обмеження статей по метрикам емоційності, сенсаційності, фактичності та достовірності.
Extension	-
Post-Condition	Користувач залишається на сторінці фільтра.

Таблиця 2.5 – Варіант використання UC-05

Use case name	Переглянути сторінку статті
Use case ID	UC-05
Goals	Користувач знаходиться на сторінці статті
Actors	Користувач
Trigger	Користувач натискає на заголовок статті на сторінці списку статей
Pre-conditions	Користувач знаходиться на сторінці списку статей
Flow of Events	Користувач, по натисканню заголовку статті, переходить на сторінку статті. На цій сторінці користувач може переглянути заголовок статті, текст статті, джерела, та посилання на джерело від видання, з якого воно взяте.
Extension	-
Post-Condition	Користувач залишається на сторінці статті.

Таблиця 2.6 – Варіант використання UC-06

Use case name	Переглянути метрики якості
Use case ID	UC-06
Goals	Користувач може побачити оцінки статті
Actors	Користувач
Trigger	Користувач натискає на заголовок статті на сторінці списку статей
Pre-conditions	Користувач знаходиться на сторінці списку статей
Flow of Events	Нижче заголовку і тексту статті, користувач може побачити метрики якості, які були присвоєні статті. На сторінці це відображається у форматі метрика – оцінка. При наведені курсору на метрику, користувач може побачити пояснення від ШІ, чому саме така оцінка була поставлена.
Extension	-
Post-Condition	Користувач залишається на сторінці статті.

Таблиця 2.7 – Варіант використання UC-07

Use case name	Переглянути схожі новинні статті
Use case ID	UC-07
Goals	Користувач бачить схожі до статті, на сторінці якої користувач зараз перебуває, статті
Actors	Користувач
Trigger	-
Pre-conditions	Користувач знаходиться на сторінці статті
Flow of Events	Під метриками якості на сторінці статті, користувач може побачити схожі статті. Ці статті подаються у форматі заголовку, видання, що опублікувало статтю і часу публікації. По натисканню на схожу статтю, користувач може моментально потрапити на сторінку статті.
Extension	-
Post-Condition	Користувач знаходиться на сторінці схожої статті.

Таблиця 2.8 – Варіант використання UC-08

Use case name	Переглянути сторінку розслідувань факт-чекерів
Use case ID	UC-08
Goals	Користувач знаходиться на сторінці зі списком розслідувань від факт-чекерів
Actors	Користувач
Trigger	Користувач натискає на кнопку «Викривання фейків»
Pre-conditions	-
Flow of Events	Користувач потрапляє до списку розслідувань. Розслідування, знаходяться в такому самому форматі як і статті новин.
Extension	
Post-Condition	Користувач знаходиться на сторінці з розслідуваннями.

Таблиця 2.9 – Варіант використання UC-09

Use case name	Переглянути сторінку розслідування
Use case ID	UC-09
Goals	Користувач бачить дані про розслідування
Actors	Користувач
Trigger	Користувач натискає на заголовок розслідування
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку розслідування, де він може побачити заголовок, посилання на оригінальну публікацію, текст, джерела.
Extension	
Post-Condition	Користувач перебуває на сторінці розслідування.

Таблиця 2.10 – Варіант використання UC-10

Use case name	Переглянути категорії, які були присвоєні розслідуванню
Use case ID	UC-10
Goals	Користувач бачить категорії, які були присвоєні розслідуванню
Actors	Гість (незареєстрований користувач)
Trigger	-
Pre-conditions	Користувач знаходиться на сторінці розслідування
Flow of Events	Під заголовком розслідування, користувач може побачити категорії, які були присвоєні розслідуванню під час аналізу LLM.
Extension	-
Post-Condition	Користувач перебуває на сторінці розслідування

Таблиця 2.11 – Варіант використання UC-11

Use case name	Переглянути сторінку щотижневих трендів і дайджесту
Use case ID	UC-11
Goals	Користувач знаходиться на сторінці щотижневих трендів і дайджесту
Actors	Користувач
Trigger	Користувач натискає на кнопку «Тренди дезінформації»
Pre-conditions	-
Flow of Events	Користувач переходить до сторінки з щотижневими трендами і дайджестом.
Extension	-
Post-Condition	Користувач перебуває на сторінці «Тренди дезінформації»

Таблиця 2.12 – Варіант використання UC-12

Use case name	Переглянути згенерований дайджест за тиждень
Use case ID	UC-12
Goals	Користувач бачить дайджест за останній тиждень у
Actors	Користувач
Trigger	-
Pre-conditions	Користувач перебуває на сторінці «Тренди дезінформації»
Flow of Events	Користувач бачить дайджест за останній тиждень. Цей дайджест складається з заголовку «Дайджест за {Дата}», та тексту, який коротко описує об'єкти розслідувань за останній тиждень.
Extension	
Post-Condition	Користувач перебуває на сторінці «Тренди дезінформації»

Таблиця 2.13 – Варіант використання UC-13

Use case name	Переглянути найпопулярніші категорії дезінформації впродовж часу
Use case ID	UC-13
Goals	Користувач
Actors	Користувач бачить на графіку
Trigger	-
Pre-conditions	Користувач перебуває на сторінці «Тренди дезінформації»
Flow of Events	Користувач бачить статистику найчастіше з категоріями, що найчастіше зустрічаються у розслідуваннях факт-чекерів за останні 3-5 тижнів. Кожна категорія вираховується як сума усіх розслідувань з відповідною категорією.
Extension	-
Post-Condition	Користувач перебуває на сторінці «Тренди дезінформації»

2.2 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблиці 2.14 наведено загальну модель вимог, а в таблиці 2.15 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.2.

Таблиця 2.14 – Загальна модель вимог

№	Назва	ID Вимоги	Пріоритети	Ризики
1	Перехід на сторінку новин	FR-1	Високий	Низький
2	Застосування фільтру для новин	FR-2	Середній	Низький
2.1	Обмежити пошук статей по виданню	FR-3	Середній	Низький
2.2	Обмежити пошук статей по метрикам якості	FR-4	Середній	Середній

Продовження таблиці 2.14

№	Назва	ID Вимоги	Пріоритети	Ризики
3	Переглянути сторінку статті	FR-5	Високий	Низький
3.1	Переглянути схожі статті	FR-6	Середній	Низький
3.2	Переглянути оцінки якості статті	FR-7	Низький	Низький
4	Переглянути сторінку розслідувань факт-чекерів	FR-8	Високий	Низький
4.1	Переглянути сторінку розслідування	FR-9	Високий	Низький
4.2	Переглянути категорії, які були присвоєні розслідуванню	FR-10	Середній	Низький
5	Переглянути сторінку щотижневих трендів і дайджесту	FR-11	Середній	Низький
5.1	Переглянути згенерований дайджест за тиждень	FR-12	Середній	Низький
5.2	Переглянути найпопулярніші категорії дезінформації впродовж часу	FR-13	Низький	Низький

Таблиця 2.15 – Перелік функціональних вимог

Назва	Опис
FR-1	Перехід на сторінку новин Функція дозволяє користувачу переходити на головну сторінку зі списком усіх останніх новинних статей від підтримуваних видань. Система відображає статті у хронологічному порядку дати публікації статей з можливістю навігації.

Продовження таблиці 2.15

Назва	Опис
FR-2	Застосування фільтру для новин Система надає функціонал фільтрації новинного контенту за різними критеріями. Користувач може застосувати комбінацію фільтрів для знаходження статей що відповідають його інтересам.
FR-3	Обмежити пошук статей по виданню Користувач має можливість фільтрувати статті за конкретними новинними видавниками (Espreso, Hromadske, Ukrinform, Радіо Свобода). Система дозволяє обирати один або кілька видавників одночасно.
FR-4	Обмежити пошук статей по метрикам якості Функція дозволяє користувачу фільтрувати статті за показниками якості: емоційність, сенсаційність, фактичність, достовірність джерел. Кожна метрика має свою шкалу оцінювання.
FR-5	Переглянути сторінку статті При натисканні на заголовок статті користувач переходить на детальну сторінку з повним текстом статті, видавником, часом публікації та посиланням на оригінальну публікацію.
FR-6	Переглянути схожі статті На сторінці статті система відображає список семантично схожих статей, знайдених за допомогою семантичного аналізу векторів статей. Користувач може перейти до будь-якої зі схожих статей.
FR-7	Переглянути оцінки якості статті Система відображає автоматично згенеровані оцінки статті за метриками: емоційність, фактичність, достовірність джерел, сенсаційність. При наведенні на оцінку відображається пояснення від AI.

Продовження таблиці 2.15:

Назва	Опис
FR-8	Переглянути сторінку розслідувань факт-чекерів Користувач має доступ до окремої сторінки із списком розслідувань дезінформації від організацій VoxUkraine та StopFake. Статті відображаються в хронологічному порядку.
FR-9	Переглянути сторінку розслідування Детальний перегляд факт-чекінгового розслідування з повним текстом, джерелами, та посиланням на оригінальну публікацію. Відображається контекст викритого фейку.
FR-10	Переглянути категорії, які були присвоєні розслідуванню Система автоматично категоризує розслідування за типами дезінформації (виправдання агресії, маніпуляції з військовими діями, політичні процеси тощо). Категорії відображаються як теги.
FR-11	Переглянути сторінку щотижневих трендів і дайджесту Користувач має доступ до аналітичної сторінки з оглядом тенденцій дезінформації. Сторінка містить графіки трендів та резюме подій за тиждень.
FR-12	Переглянути згенерований дайджест за тиждень Система автоматично генерує короткий текстовий огляд основних дезінформаційних кампаній за попередній тиждень, створений за допомогою LLM на основі зібраних розслідувань.
FR-13	Переглянути найпопулярніші категорії дезінформації впродовж часу Відображення графіків та статистики, які показують динаміку поширеності різних типів дезінформації протягом останніх тижнів. Дані представлені у вигляді інтерактивних діаграм.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13
UC-1	+												
UC-2		+											
UC-3			+										
UC-4				+									
UC-5					+								
UC-6						+							
UC-7							+						
UC-8								+					
UC-9									+				
UC-10										+			
UC-11											+		
UC-12												+	
UC-13													+

Рисунок 2.2 – Матриця трасування вимог

2.3 Розроблення нефункціональних вимог

а) система має бути швидкодійною:

1) завантаження сторінок не має займати більше трьох секунд;

2) час на аналіз і вебскрейпінг статей має займати стільки часу, щоб не перевантажувати систему.

б) система має бути зручною у використанні;

в) система має бути захищена від:

1) CSRF-атак;

2) XSS-атак;

3) SQL-ін’єкцій.

г) система має бути сумісною з різними браузерами.

2.4 Аналіз системних вимог

Системні вимоги не висовуються.

2.5 Аналіз економічних показників програмного забезпечення

Для аналізу економічних показників ПЗ я прийняв рішення обрати базову модель COCOMO.

Припустимо, що проект має 3,000 вихідних рядків коду, отже проект належить до типу «Розповсюджений» (<50K SLOC)

$$KSLOC = 3$$

Відповідно до рисунку 2.4, приймаємо параметри $a_i = 2.4$,

$$b_i = 1.05,\ c_i = 2.5, d_i = 0.38$$

Тип проєкту	a_i	b_i	c_i	d_i
Розповсюджений	2,4	1,05	2,5	0,38
Напівнезалежний	3,0	1,12	2,5	0,35
Вбудований	3,6	1,20	2,5	0,32

Рисунок 2.3 – Таблиця коефіцієнтів моделі COCOMO

Отже, можемо визначити економічні показники:

$$PM = a_i * KSLOC^{b_i} = 2.4 * 3^{1.05} = 7.45 \text{ люд.* міс.}$$

$$TM = c_i * (PM)^{d_i} = 2.5 * 7.45^{0.38} = 5.36 \text{ міс.}$$

Показник PM – означає трудомісткість, а показник TM – означає орієнтований час на розробку у календарних місяцях.

Отже, при TM = 5.36, та середній заробітній платні у 1800 $ розробника відповідно до DOU.ua [17], витрати на створення дане ПЗ становлять:

EXP = 3300 * 5.36 = 17,688 $

2.6 Постановка завдання на розробку програмного забезпечення

Отже, розробка програмного забезпечення полягатиме у наступних ключових етапах:

а) побудова програмного забезпечення для автоматичного вебскрейпінгу статей від різних видань та розслідувань від факт-чекерів;

б) створення алгоритму для аналізу новинних статей та розслідувань;

в) інтеграція сервісів LLM та алгоритмів для векторизації тексту;

г) налаштування традиційної бази даних і векторної бази даних;

д) створення вебсторінок для забезпечення взаємодії користувача і сервісу;

е) написання запитів до баз даних для показу актуальної інформації.

Висновки до розділу

Отже, у цьому розділі було детально проаналізовано вимоги до програмного забезпечення.

У першому підрозділі було створено діаграму UML Use-Case для візуалізації варіантів використання застосунку. Також було описано кожен варіант використання. Всього було наведено 13 варіантів використання.

У другому підрозділі було створено список функціональних вимог, яких теж виявилось 13, було детально описано кожну вимогу. Також у цьому підрозділі було створено таблицю трасування вимог і варіантів використання.

У третьому підрозділі було зазначено перелік нефункціональних вимог, де було зазначено вимоги до швидкодії, безпеки, зручності та сумісності.

У підрозділі про аналіз економічних показників було взято метод базової моделі COCOMO та підраховано показники трудомісткості, часових вимог і потенційних витрат.

В кінці було виконано узагальнену постановку задачі, де було наведено ключові етапи в розробці програмного забезпчення.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Як було зазначено у вимогах до програмного забезпечення, застосунок повинен виконувати велику кількість задач, деякі з яких взаємодіють одна з іншим. У проектуванні і конструюванні програмного забезпечення, найголовнішими підходами до архітектури є монолітна і мікросервісна архітектури.

Монолітна архітектура передбачає створення програмного забезпечення всередині одного компонента, де всі модулі тісно переплетені і розгортаються разом. Перевагами монолітної архітектури програмного забезпечення є простота розробки, зручність тестування, і легке розгортання. Проте дуже важливими недоліками є складність масштабування, та високий ризик загального збою усього компонента при збої навіть в одному з модулів[18].

Тим часом, мікросервісна архітектура розбиває застосунок на набір невеликих автономних компонентів, які відповідають за відповідну функцію. Дані сервіси взаємодіють через свій власний API[19]. Перевагами даного підходу до розробки є висока гнучкість і незалежне масштабування, та ізоляція відмов, які компенсуються недоліками у виді складністю управління, вищої затримки та складності тестування.

Для розробки свого програмного забезпечення, було обрано гібридний підхід, який дозволяє об’єднати переваги мікросервісної архітектури у плані стійкості до відмов та гнучкості, та переваги монолітної архітектури, у виді простоти розгортання і розробки.

Для візуалізації архітектури програмного забезпечення, було використано діаграми моделі C4[20]. Діаграма найвищого рівня моделі – це контекстна діаграма, яка дає змогу продемонструвати загальну картину, представляє взаємодію акторів і систем. Діаграму для ПЗ наведено на рисунку 3.1:

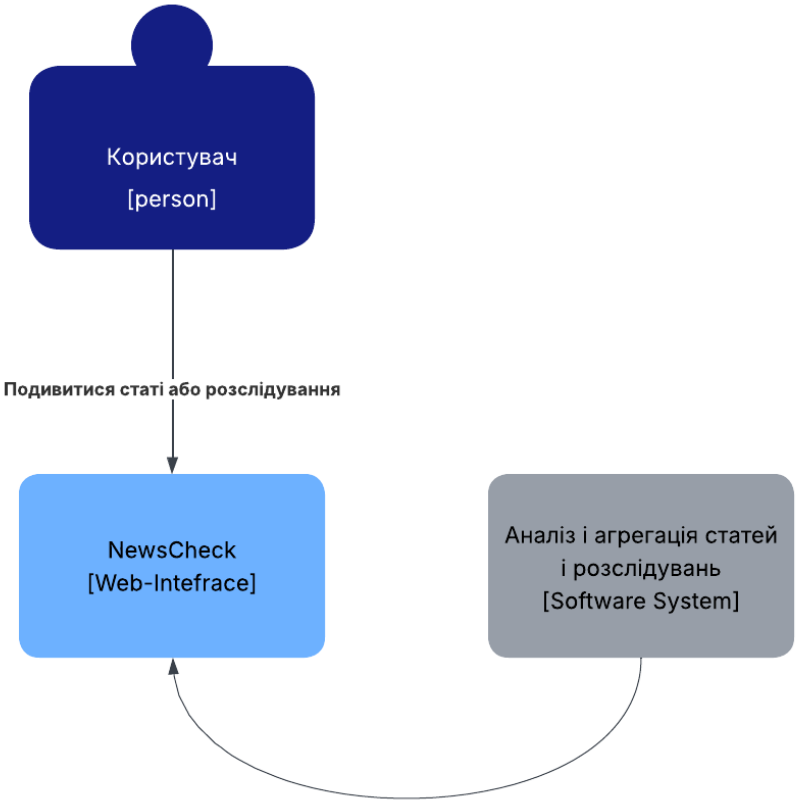


Рисунок 3.1 – Контекстна діаграма

Наступний рівень моделі C4 – це діаграма контейнерів, яка більш детально описує компоненти системи, продемонстрована на рисунку 3.2.

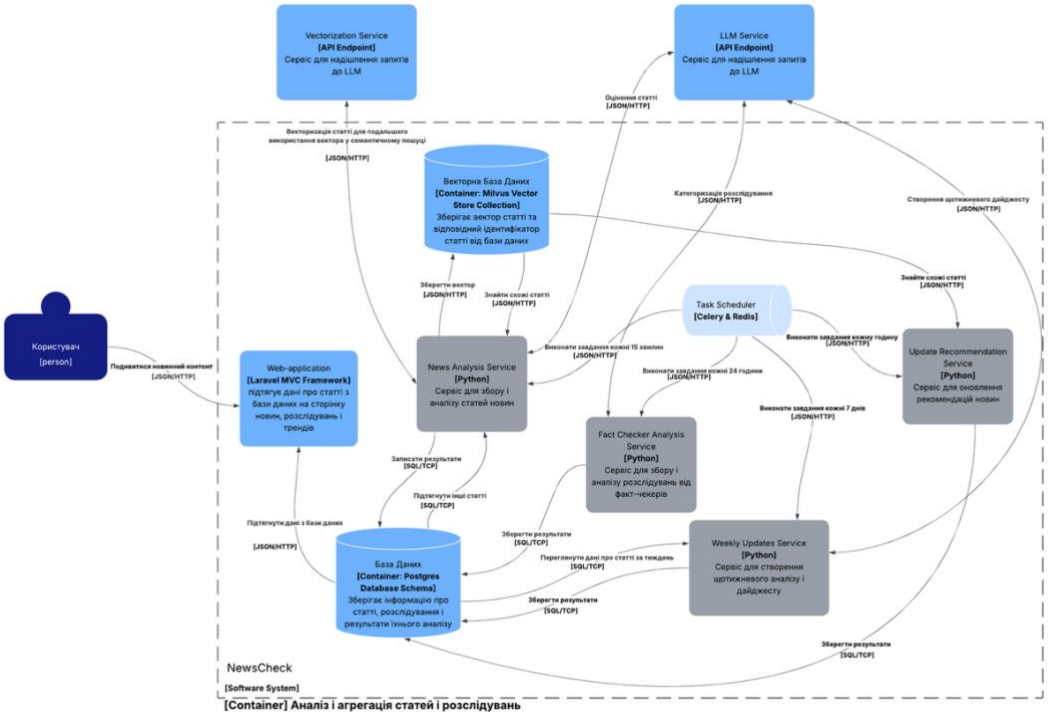


Рисунок 3.2 – Діаграма контейнерів

Діаграма зображує зв’язки наступних елементів:

- користувач (Актор);
- web-Application (Laravel) – вебзастосунок який надає інтерфейс для доступу до проаналізованих статей і розслідувань;
- база даних (PostgreSQL) – База даних яка використовується для збереження усіх результатів аналізу;
- векторна база даних (Milvus) – Колекція для зберігання векторів статей;
- сервіс для аналізу новин (Python) – Функція для отримання статей, виконання аналізу за допомогою LLM, та потім виконання семантичного аналізу за допомогою векторної бази даних Milvus. Результати аналізу зберігаються у базі даних PostgreSQL;
- сервіс для аналізу розслідувань (Python) – Функція для отримання розслідувань від факт-чекерів, та надання певних категорій методом аналізу за допомогою LLM;
- сервіс для оновлення рекомендацій (Python) – Функція для оновлення схожих статей для усіх новинних статей, час публікації яких менше або дорівнює годині. Використовує векторну базу даних для пошуку найближчого вектора та базу даних для збереження результатів;
- сервіс для щотижневих дайджестів (Python) – Функція для створення статистики розслідувань факт-чекерів за останній тиждень, та написання дайджесту на основі даних розслідувань;
- планувальник завдань (Celery) – Система для планування завдань, що використовує чергу завдань Redis для виконання сервісу для аналізу новин кожні 15 хвилин; сервісу для оновлення рекомендацій кожну годину; сервісу для збору і аналізу розслідувань кожні 24 години; та виконання сервісу для щотижневого аналізу розслідувань виконуються кожні 7 днів.

Наступний етап – це створення діаграм третього рівня моделі C4, тобто діаграм компонентів. Дані діаграми ще глибше занурюються у архітектуру системи, тому кожен компонент, який на попередній діаграмі не розписувався, матиме свою діаграму.

На рисунках 3.3 – 3.6 зображено діаграми компонентів для вище описаних сервісів 5) - 8).

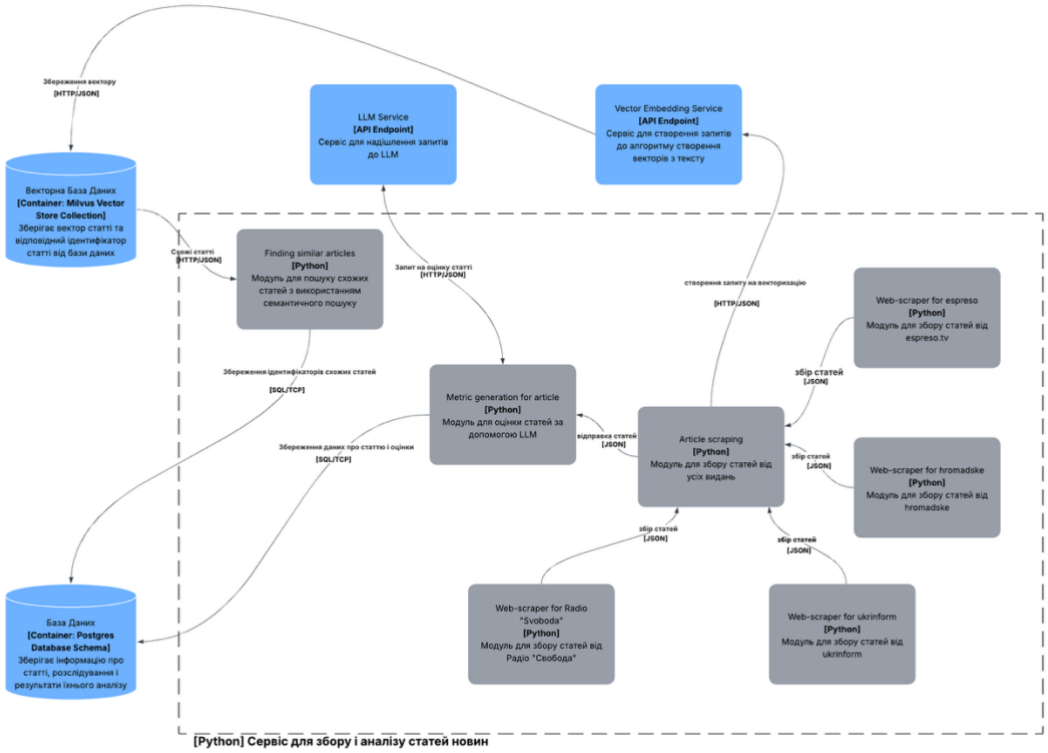


Рисунок 3.3 – Діаграма компонента «Сервіс для збору і аналізу новин»

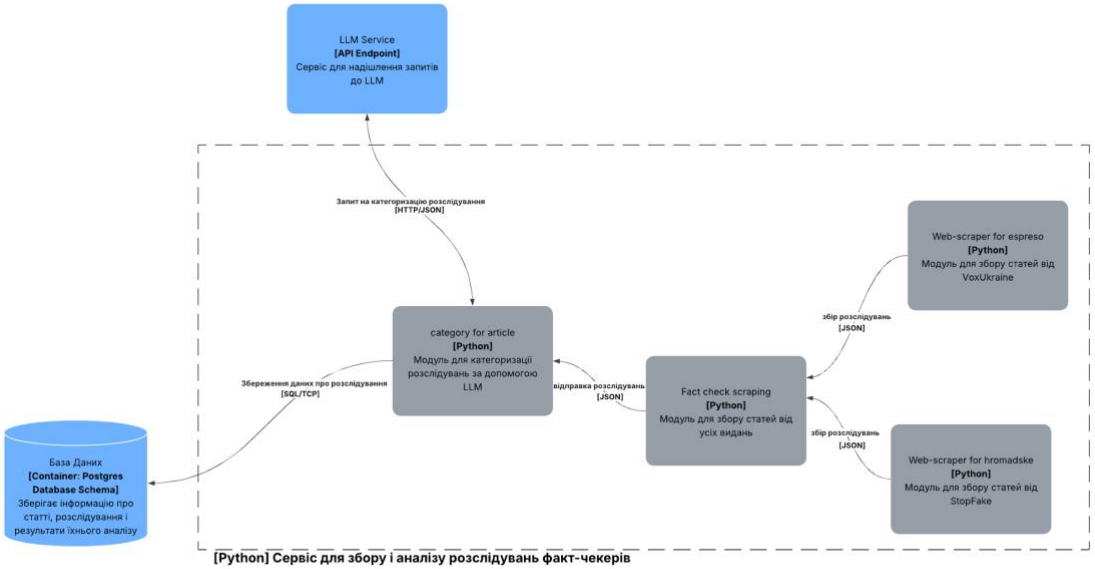


Рисунок 3.4 – Діаграма компонента «Сервіс для збору і аналізу розслідувань»

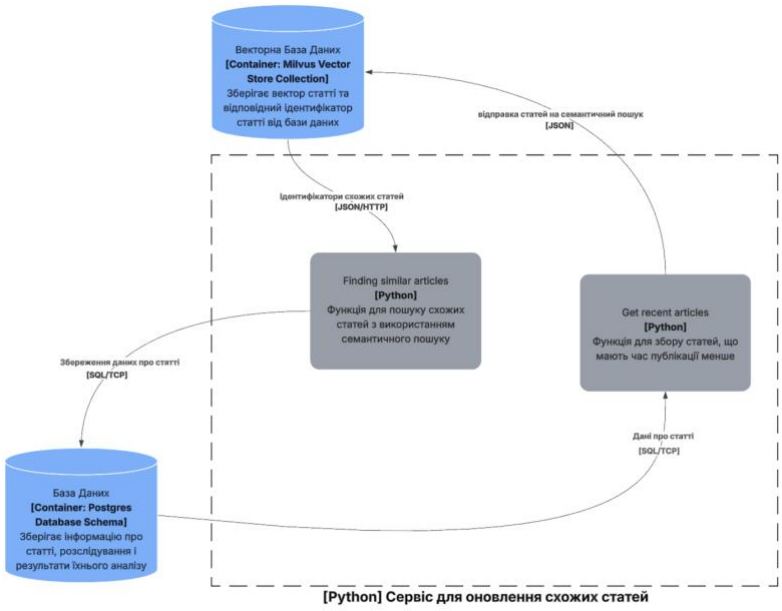


Рисунок 3.5 – Діаграма компонента «Сервіс для оновлення схожих статей»

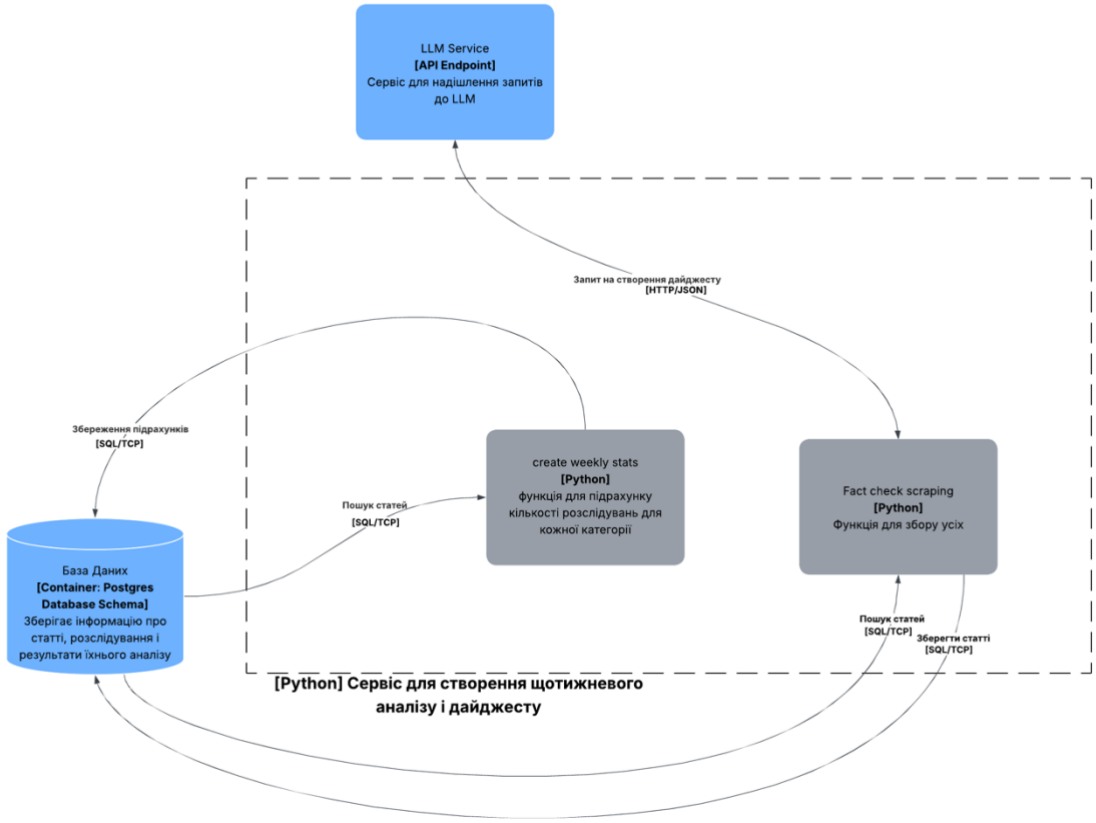


Рисунок 3.6 – Діаграма компонента «Сервіс для створення щотижневого »

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

Було прийнято рішення використати наступні мови програмування:

Для серверної частини програмного забезпечення було прийнято рішення використати мову програмування Python, через її нескладний синтаксис, широкий перелік бібліотек для аналізу та вебскрейпінгу, і також через широкий вибір взаємодій з базами даних.

Django надає потужні засоби для швидкої розробки з автоматичною адмін-панеллю та відмінним ORM. Основною перевагою Django є інтеграція з екосистемою Python для машинного навчання та аналізу даних. Однак для даного проєкту Laravel[21] обраний через кращу інтеграцію з фронтенд-рішеннями, простішу конфігурацію хостингу та менші вимоги до серверних ресурсів. Laravel також забезпечує більш зручну роботу через вбудовані засоби.

Laravel надає оптимальний баланс між швидкістю розробки та функціональністю. Eloquent ORM дозволяє ефективно працювати зі складними відношеннями між статтями, метриками та категоріями. Вбудована система міграцій забезпечує версійний контроль структури бази даних, що критично важливо для проєкту з еволюційною схемою даних.

Blade шаблонізатор Laravel спрощує створення динамічних інтерфейсів для відображення новин та їх аналізу. Artisan CLI автоматизує рутинні завдання розробки та дозволяє створювати кастомні команди для специфічних потреб проєкту, таких як запуск скрапінгу або генерація звітів.

Для баз даних, було обрано СУБД PostgreSQL[25] через її швидкодію та сумісність з різними інструментами, та Milvus[24] як векторну базу даних через її сумісність з Python у вигляді бібліотеки pymilvus, яка дозволяє легко і зручно взаємодіяти з колекціями, при локальному розгортанні.

Система не передбачає складної системи аутентифікації користувачів. Всі функції доступні без реєстрації, що спрощує доступ до інформації в умовах швидко змінюваної новинної обстановки. Це рішення обґрунтовується публічним характером новинного контенту, необхідністю швидкого доступу до інформації та спрощенням архітектури системи.

Система також інтегрується з наступними зовнішніми сервісами:

а) OpenAI API[22]: використовується для аналізу якісних характеристик статей (емоційність, фактичність, достовірність джерел, сенсаційність). Нещодавня модель GPT-4.1-Mini обрана завдяки оптимальному співвідношенню якості аналізу, вартості використання, та швидкодії;

б) VoyageAI API[23]: створює семантичні вектори статей для пошуку схожого контенту. Модель voyage-3 забезпечує високу якість векторизації українського тексту;

в) Вебскрапінг українських медіа: система автоматично збирає новини з провідних українських видань (ukrinform, hromadske, espreso.tv,

Радіо Свобода, StopFake, VoxUkraine) за допомогою бібліотек, доступних у Python Selenium[26] та requests[27].

Система відповідає нефункціональним вимогам:

а) Використання СУБД PostgreSQL для відображення матеріалу через об’єктно-реляційну модель Eloquent для Laravel дає змогу задовольнити вимогу у швидкодії;

б) Також використання Eloquent дозволяє запобігти SQL-ін’єкціям;

в) Фреймворк Laravel також є сумісним для браузерів з версіями JavaScript ES6+;

г) Зручність забезпечена інтуїтивним інтерфейсом.

Для системи черг було обрано Celery через свою здатність ефективно асинхронно виконувати задачі, які було описано у діаграмах компонентів (рис. 3.3 – 3.6).

Як БД для повідомлень було обрано Redis, що використовує БД в пам’яті для брокерування повідомлень для Celery та кешування часто запитуваних даних.

Для середовищ розробки було обрано PyCharm & PhpStorm від JetBrains, через їх сумісність з версіонуванням, а також великої кількості плагінів, автодоповнення за допомогою JetBrains-AI, та інших невеликих, але зручних переваг.

3.3 Конструювання програмного забезпечення

3.3.1 Алгоритм пошуку схожих статей

Як було зазначено на діаграмі компонентів “Сервіс для збору та аналізу новин” та “Сервіс для оновлення схожих статей”, для пошуку схожих статей використовується векторна база даних, яка спеціально оптимізована для математичних операцій з векторами.

Перед операцією пошуку, сервіс VoyageAI дає змогу отримати вектор ебедингів на основі архітектури трансформера. Дані вектори отримуються з

розмірністю [1024; 1], що дає відкриває змогу порівняти напрямки даних векторів у багатовимірному просторі.

Чим більш співнапрямлений вектор до іншого вектора, тим більше вони подібні. З цієї причини було прийнято рішення використати алгоритм косинусу подібності, що вираховується за наступною формулою:

$$cos(\theta) = \frac{A \cdot B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2}\sqrt{\sum_{i=1}^{n} {B_i}^2}},$$

де θ - це кут відхилення між векторами, A, B - вектори однакової розмірності.

Дана операція дозволяє отримати легко порівняльну метрику. Чим більш подібні вектори, тим їх косинус подібності буде ближчий до одного, та чим ближче косинус до нуля, тим більша семантична різниця між текстами цих векторів. Це дає змогу відрізнити подібні тексти від неподібних, тому у алгоритмі було обрано поріг для схожих статей у 0.5. Тобто Стаття вважається схожою тоді і тільки тоді, коли косинус подібності більше або дорівнює 0.5.

3.3.2 Опис структури бази даних

В якості системи управління базами даних використовується PostgreSQL. База даних серверу призначена для зберігання статей та їхнього аналізу. Опис таблиць бази даних наведено у таблицях 3.1-3.14. Модель бази даних наведена на рисунку 3.2.

Таблиця 3.1 – Опис таблиці article

Назва поля	Тип даних	Опис
id	serial	Ідентифікаційний номер статті
title	varchar	Оригінальний заголовок статті
href	int	Посилання на оригінальну статтю
outlet	varchar	Назва видання

Продовження таблиці 3.1

Назва поля	Тип даних	Опис
published_at	datetime	Час публікації оригінальної статті
created_at	datetime	Час занесення статті у базу даних
status	varchar	Статус обробки статті

Таблиця 3.2 – Опис таблиці paragraph

Назва поля	Тип даних	Опис
id	serial	Ідентифікаційний номер параграфу статті
paragraph_text	varchar	Текст параграфу
paragraph_num	int	Номер параграфу як це було у статті
article_id	int	Ключ з посиланням на articles

Таблиця 3.3 – Опис таблиці source

Назва поля	Тип даних	Опис
id	serial	Ідентифікаційний номер посилання на джерело в статті
source_url	varchar	Посилання на джерело взяте зі статті
source_num	int	Номер джерела в порядку зростання зі статті
article_id	int	Ключ з посиланням на articles

Таблиця 3.4 – Опис таблиці recommended_article

Назва поля	Тип даних	Опис
id	serial	Ідентифікаційний номер схожої статті
article_id	int	Ключ з посиланням на article, який вказує на статтю для якої підбираються схожі статті
recommended_article_id	int	Ключ з посиланням на article, який вказує на схожу статтю
similarity_score	float	Ключ з посиланням на articles
last_updated	datetime	Час, коли була зроблена рекомендація

Таблиця 3.5 – Опис таблиці metric

Назва поля	Тип даних	Опис
Id	serial	Ідентифікаційний номер метрики оцінювань від LLM
article_id	int	Ключ з посиланням на article, який вказує на статтю для якої зроблені оцінки
emotionality_score	varchar	Оцінка емоційності статті (нейтральна, дещо емоційна, дуже емоційна)
emotionality_reason	varchar	Обґрунтування для оцінки емоційності
factuality_score	int	Оцінка фактуальності статті (чи посилається стаття на реальні події)

Продовження таблиці 3.5

Назва поля	Тип даних	Опис
factuality_reason	varchar	Обґрунтування для оцінки фактуальності
credibility_score	int	Оцінка достовірності статті (наскільки надійні джерела)
credibility_reason	varchar	Обґрунтування для оцінки достовірності
clickbaitness_score	int	Оцінка сенсаційності статті (наскільки гучний заголовок статті)
clickbaitness_reason	varchar	Обґрунтування для оцінки сенсаційності

Таблиця 3.6 – Опис таблиці factcheck_category

Назва поля	Тип даних	Опис
id	serial	Ідентифікаційний номер присвоєної до статті категорії
article_id	int	Ключ з посиланням на article, який вказує на статтю для якої підбираються схожі статті
category_id	int	Ключ з посиланням на category, який вказує на ідентифікатор категорії

Таблиця 3.7 – Опис таблиці category

Назва поля	Тип даних	Опис
id	serial	Ідентифікаційний номер категорії
name	int	Назва категорії
description	int	Опис категорії

Таблиця 3.8 – Опис таблиці weekly_stats

Назва поля	Тип даних	Опис
id	serial	Ідентифікаційний номер статистики
category_id	int	Ключ з посиланням на ідентифікатор категорії
num_articles	int	Кількість статей, які були знайдені за відповідною категорією за останній тиждень
date	date	Останній день тижня, на якому збиралися статті

Таблиця 3.9 – Опис таблиці weekly_digest

Назва поля	Тип даних	Опис
id	serial	Ідентифікаційний номер дайджесту
digest_date	date	Дата останнього дня тижня, коли був створений дайджест
report_text	varchar	Текст дайджесту

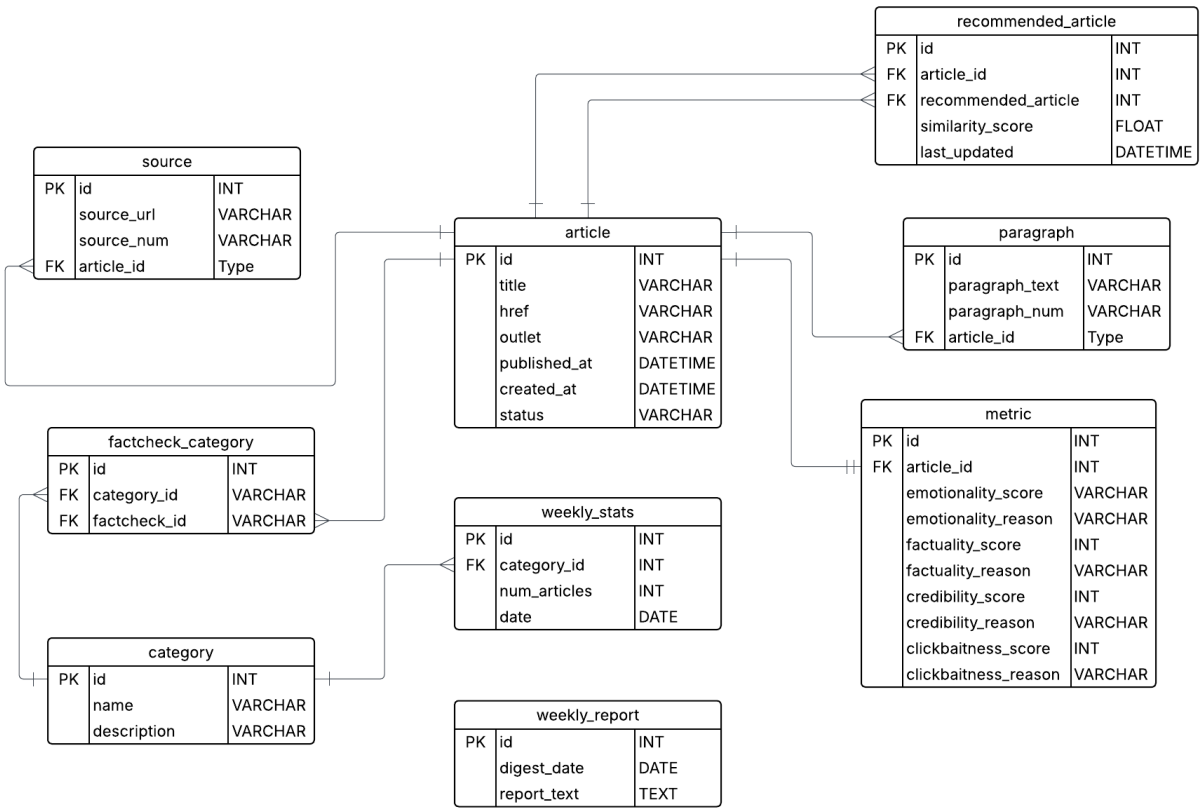


Рисунок 3.7 – ER діаграма усіх сутностей бази даних

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.10.

Таблиця 3.10 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	PyCharm	Головне середовище розробки програмного забезпечення серверної частини дипломного проєкту.
2	PhpStorm	Головне середовище розробки програмного забезпечення клієнтної частини курсової роботи.
3	Composer	Менеджер залежностей для PHP
4	Git	Система контролю версій, використовувалася для відстежування змін у коді
5	pip	Менеджер залежностей для Python

Продовження таблиці 3.10

№ п/п	Назва утиліти	Опис застосування
6	Docker	Застосунок для контейнеризації, який використовувався для розгортання векторної бази данх Milvus
7	SQLalchemy	Бібліотека для Python, яка надає можливість використовувати ORM.
8	pymilvus	Бібліотека для взаємодії з колекціями векторної бази даних milvus
9	Requests	Бібліотека для надсилання запитів у Python. Використовувалася для отримання HTML-коду сторінок новин і розслідувань
8	pymilvus	Бібліотека для взаємодії з колекціями векторної бази даних milvus
9	Requests	Бібліотека для надсилання запитів у Python. Використовувалася для отримання HTML-коду сторінок новин і розслідувань
10	Selenium	Бібліотека для автоматизації роботи з браузером у Python. Використовувалася для вебскрейпінгу для сторінок які використовували JavaScript, що унеможливлювало вебскрейпінг з requests.
11	openai	Бібліотека для Python від OpenAI, розроблена для полегшення роботи з сервісами від OpenAI
12	pydantic	Бібліотека для створення спеціальних структур даних. Використовується для генерації стабільних типізованих відповідей від LLM

3.4 Аналіз безпеки даних

Розроблена система має кілька рівнів захисту від різних типів атак та загроз безпеки. В контексті управління вразливостями система використовує Laravel фреймворк який має вбудовані засоби захисту від SQL ін'єкцій завдяки використанню Eloquent ORM та підготовлених запитів. Всі користувацькі вводи проходять валідацію через контролери Laravel та Pydantic схеми в Python частині що запобігає некоректному введенню даних та потенційним атакам через маніпуляцію параметрів. CSRF токени автоматично генеруються Laravel для всіх форм що захищає від Cross-Site Request Forgery атак. Система використовує HTTPS протокол для шифрування трафіку між клієнтом та сервером що запобігає перехопленню даних під час передачі.

Безпека даних забезпечується кількома способами. API ключі для зовнішніх сервісів OpenAI та VoyageAI зберігаються у змінних середовища через dotenv файли та не потрапляють до репозиторію коду. База даних PostgreSQL має обмежені права доступу та використовує з'єднання через захищені паролі. Персональні дані користувачів не збираються оскільки система не передбачає реєстрації або аутентифікації що знижує ризики пов'язані з витоком особистої інформації. Всі зібрані новини містять лише публічно доступну інформацію з офіційних медіа ресурсів.

В сфері тестування безпеки система проходить перевірку через статичний аналіз коду в IDE PyCharm та PhpStorm які виявляють потенційні вразливості на етапі розробки. Laravel має вбудовані засоби захисту від XSS атак через автоматичне екранування змінних у Blade шаблонах. Python код використовує типізацію через Pydantic що попереджає помилки типів даних та можливі експлойти. Веб скрапінг модулі мають обмеження за часом виконання та обсягом завантажуваних даних що запобігає DoS атакам на цільові ресурси.

Архітектура системи має кілька рівнів захисту. Веб додаток Laravel працює як фронтенд інтерфейс, та не має прямого доступу до логіки збору новин. Python сервіси працюють ізольовано та взаємодіють з базою даних

через встановлені права доступу. Redis брокер для Celery має обмежений доступ лише до необхідних черг завдань. Milvus векторна база даних працює в локальному режимі що знижує ризики несанкціонованого доступу через мережу.

Система не використовує контейнери Docker у базовій конфігурації але може бути розширена для додаткового рівня ізоляції. Всі зовнішні API викликаються через HTTPS що забезпечує шифрування трафіку. Логування системи ведеться на рівні Laravel та Python що дозволяє відстежувати підозрілу активність та виявляти потенційні атаки. Бекап стратегія включає регулярне копіювання бази даних PostgreSQL для відновлення у випадку втрати даних або компрометації системи.

Висновки до розділу

Отже, у цьому розділі було детально виконано конструювання і архітектура програмного забезпечення.

У першому підрозділі було детально описано архітектуру програмного забезпечення. Було сконструйовано три рівні моделі C4 та відповідні діаграми контексту, контейнерів і компонентів. Було також обрано загальний патерн до розробки програмного забезпечення. Було сформовано компоненти програмного забезпечення.

У другому підрозділі було детально проаналізовано засоби розробки програмного забезпечення, СУБД. Було описано сторонні сервіси та API, які використовуються створеним програмним забезпеченням. Також було названо методи, як програмне забезпечення виконує поставлені нефункціональні вимоги.

У третьому підрозділі було детально описано базу даних застосунку та відповідні сутності. По результатам було створено діаграму сутностей, яка наочно відображає сутності (таблиці бази даних) та їх взаємозв’язки.

В останньому підрозділі було проаналізовано безпеку даних застосунку. Було продумано декілька можливих типів атак, до яких варто адаптувати

застосунок. По результатам аналізу ми переконалися, що застосунок є захищеним від таких атак як SQL-ін’єкції, CSRF та XSS.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Метриками для оцінки якості ПЗ обрано наступні:

– Часова затримка відповіді на запит з клієнтської сторони;

– Навантаження сервера;

– Часова затримка обробки запитів аналізу новин.

Для аналізу часової затримки з клієнтської сторони, було використано функціонал платформи render.com[28]. Після розгортання вебзастосунку можна подивитися на Dashboard у вкладці Metrics. Графік часової затримки наведено на рисунку 4.1.

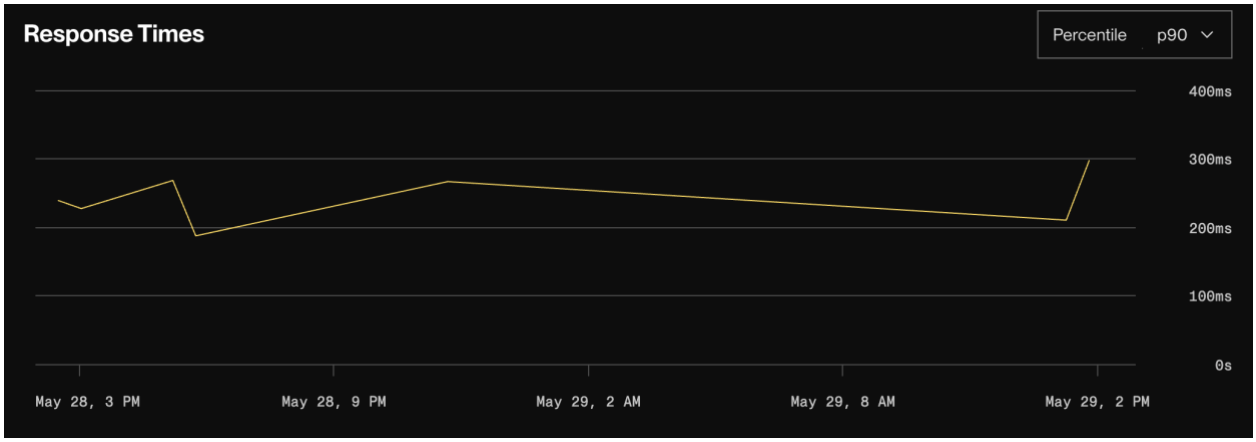


Рисунок 4.1 – Часова затримка відповіді з клієнтської сторони

Як можна спостерігати на графіку, часова затримка не перевищує 400 мс. Це повністю задовольняє вимогу про швидкодію системи.

Для демонстрації навантаження сервера можна також застосувати функціонал платформи render.com подивитися на Dashboard у вкладці Metrics. Можна ознайомитись з показниками навантаження пам'яті і процесора відповідно на рисунках 4.2 і 4.3

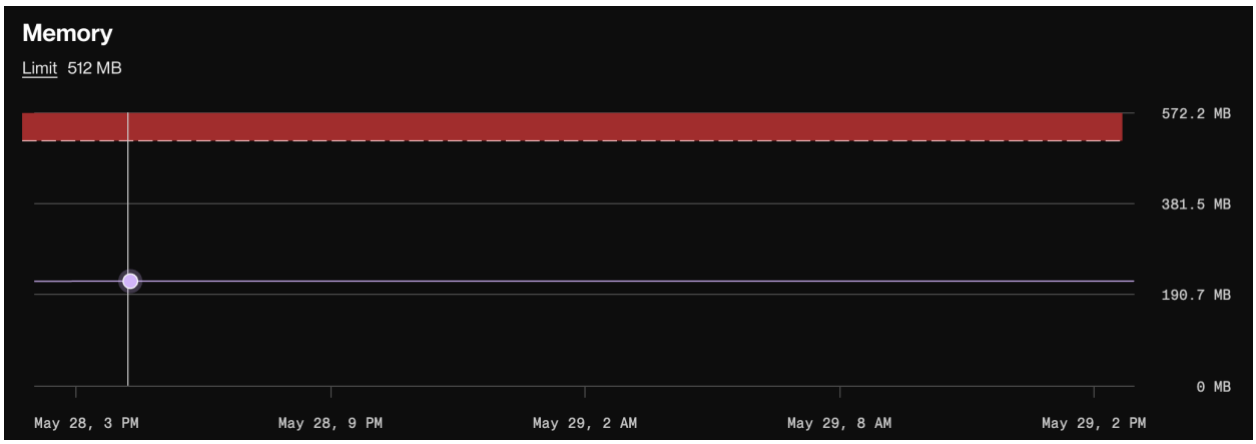


Рисунок 4.2 – Навантаження пам’яті відповіді з клієнтської сторони

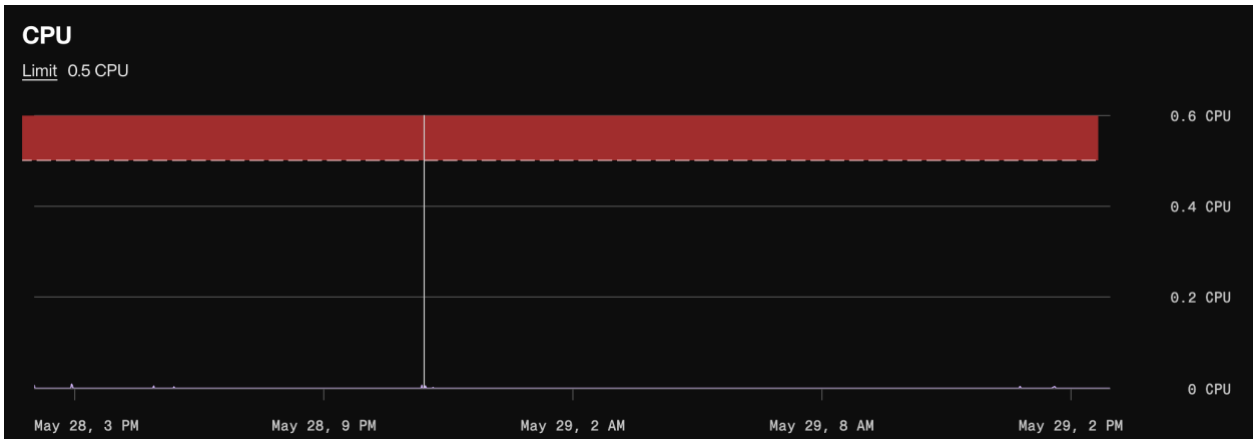


Рисунок 4.3 – Навантаження процесора відповіді з клієнтської сторони

Знову можемо переконатися, що система не споживає значних ресурсів і функціонує стабільно, оскільки система використовує дуже малу частку виділених ресурсів процесора, і використання пам’яті також залишається дуже стабільним

Оскільки для виконання цих задач використовується Celery та база даних Redis, то можна відслідковувати часову затримку за допомогою скрипту, який надасть можливість аналізувати час виконання кожної задачі. Результат середньої часової затримки аналізу новин наведено на рисунку 4.4.

Можна переконатися, що аналіз новин, який відбувається кожні 15 хвилин, не перевищує по тривалості більше 15 хвилин навіть у найгіршому випадку, за 100 замірів виконання. Також можна побачити, що виконання інших задач займає адекватну кількість часу.

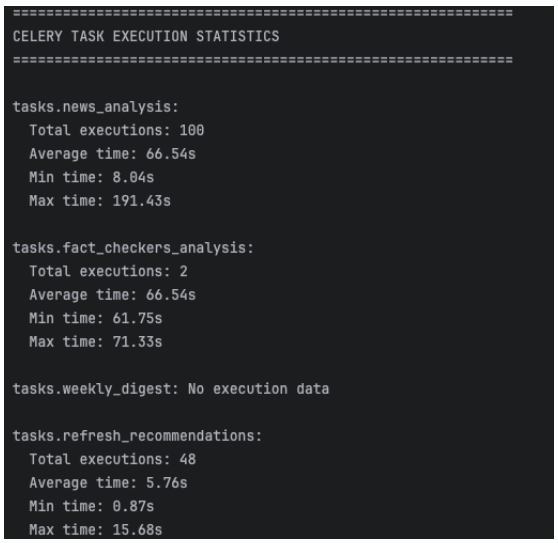


Рисунок 4.4 – Середня часова затримка процесів аналізу

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.1 – 4.15.

Таблиця 4.1 – Тест 1.1 Перехід на головну сторінку новин

Початковий стан системи	Користувач відкриває вебзастосунок
Вхідні дані	URL вебзастосунку
Опис проведення тесту	Користувач вводить URL застосунку в адресний рядок браузера. Система повинна завантажити першу сторінку з новинами.
Очікуваний результат	Завантажується сторінка зі списком новин, які упорядковані у хронологічному порядку спадання. Користувачу доступні також опції фільтру та перейти на сторінки «Викривання фейків» та «Тренди дезінформації»
Фактичний результат	Збігається з очікуваним.

Таблиця 4.2 – Тест 1.2 Відображення списку новин

Початковий стан системи	Користувач знаходиться на сторін
Вхідні дані	-
Опис проведення тесту	Користувач бачить список статей на першій сторінці новин. Перевіряється правильне відображення заголовків, часу публікації та видання.
Очікуваний результат	Кожна стаття має заголовок, який відповідає заголовку оригінальної статті. Час публікації відповідає часу публікації оригінальної статті, а також часовому поясу регіону. Список статей розбитий на сторінки, по 10 статей на кожну сторінку.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.3 – Тест 2.1 Відкриття фільтра новин

Початковий стан системи	Користувач знаходиться на сторін
Вхідні дані	Натискання іконки фільтра
Опис проведення тесту	Користувач натискає на іконку фільтра, після чого бачить перед собою вікно фільтра.
Очікуваний результат	Відкривається вікно фільтра, де користувач може обрати чекбоксами видання, які він бажає бачити, також чекбоксами користувач може обрати бажані рівні емоційності новин. За допомогою dual-range slider користувач може обирати прийнятні рівні фактичності, сенсаційності та достовірності джерел.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.4 – Тест 2.2 Фільтрація за виданням

Початковий стан системи	Користувач знаходиться на сторінці фільтра
Вхідні дані	Вибір видань Укрінформ і Радіо «Свобода»
Опис проведення тесту	Користувач натискає на чекбокси «Укрінформ» і «Радіо «Свобода»», після чого натискає на «Зберегти налаштування». Після цього на сторінці новин з’являються лише статті від відповідних видань.
Очікуваний результат	Після цього на сторінці новин з’являються лише статті від відповідних видань.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.5 – Тест 2.3 Фільтрація за метриками якості

Початковий стан системи	Користувач знаходиться на сторінці фільтра
Вхідні дані	Налаштування слайдера фактичності від 3 до 5 та емоційності на «нейтральні»
Опис проведення тесту	Користувач натискає на чекбокс «Нейтральні» для показника емоційності та встановлює максимальне і мінімальне значення показника фактичності відповідно 5 і 3. Після цього користувач натискає на кнопку «Зберегти налаштування»
Очікуваний результат	Кожна стаття, яка відображається після застосування фільтра відповідає налаштуванням фільтра.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.6 – Тест 2.4 Скасування налаштувань фільтра

Початковий стан системи	Користувач знаходиться на сторінці фільтра
Вхідні дані	Користувач натискає на кнопку «Скасувати налаштування»
Опис проведення тесту	Користувач, маючи деякі налаштування не за замовчуванням у фільтрі, натискає на кнопку «Скасувати налаштування»
Очікуваний результат	Користувач залишається у вікні фільтра, та всі налаштування повертаються до початкової конфігурації.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.7 – Тест 3.1 Перегляд статті новин

Початковий стан системи	Користувач знаходиться на сторінці новин
Вхідні дані	Натискання заголовку статті
Опис проведення тесту	Користувач натискає на заголовок статті.
Очікуваний результат	Користувач переходить на сторінку статті. На сторінці користувач може побачити заголовок, текст, час публікації, посилання на оригінальну статтю оцінки від LLM, джерела, та схожі статті.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.8 – Тест 3.2 Перегляд метрик якості статті

Початковий стан системи	Користувач знаходиться на сторінці статті
Вхідні дані	Наведення курсору на оцінки від LLM
Опис проведення тесту	Користувач наводить курсор на усі метрики якості, які були згенеровані LLM.
Очікуваний результат	При наведені курсору з’являється текст-пояснення конкретної оцінки від LLM.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.9 – Тест 3.3 Перегляд схожих статей

Початковий стан системи	Користувач знаходиться на сторінці статті
Вхідні дані	-
Опис проведення тесту	Користувач прокручує сторінку статті вниз.
Очікуваний результат	Під оцінками статті, користувач має бачити список схожих статей, якщо такі є.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.10 – Тест 4.1 Перехід на сторінку «Викривання фейків»

Початковий стан системи	Користувач перебуває на сторінці новин
Вхідні дані	Натискання кнопки «Викривання фейків»

Продовження таблиці 4.10

Опис проведення тесту	Користувач, перебуваючи на будь-якій сторінці вебзастосунку, зверху натискає на кнопку «Викривання фейків».
Очікуваний результат	Завантажується сторінка «Викривання фейків». Розслідування повинні бути упорядковані за датою публікації у порядку спадання. Розслідування, які статті новин, розбиті з пагінацією по 10 розслідувань на сторінку.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.11 – Тест 5.1 Перегляд розслідування

Початковий стан системи	Користувач перебуває на сторінці «Викривання фейків»
Вхідні дані	Натискання на заголовок розслідування
Опис проведення тесту	Користувач натискає на заголовок розслідування, після чого перенапрямлений до сторінки розслідування.
Очікуваний результат	Завантажується сторінка розслідування. На сторінці користувач бачить заголовок розслідування, текст, посилання на джерело розслідування, використані джерела та категорії дезінформації, до яких належить розслідування. Категорії розслідування призначаються LLM до розслідування на основі всієї інформації розслідування і одне розслідування може належати декільком категоріям.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.12 – Тест 6.1 Перехід на сторінку «Тренди дезінформації»

Початковий стан системи	Користувач знаходиться на сторінці «Викривання фейків»
Вхідні дані	Натискання кнопки «Тренди дезінформації»
Опис проведення тесту	Користувач, перебуваючи на будь-якій сторінці у застосунку, натискає на кнопку «Тренди дезінформації».
Очікуваний результат	Завантажується сторінка «Тренди дезінформації» з графіком найпопулярніших категорій дезінформації та дайджестом фейків за останній тиждень.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.13 – Тест 6.2 Перегляд графіку дезінформації

Початковий стан системи	Користувач перебуває на сторінці «Тренди дезінформації»
Вхідні дані	-
Опис проведення тесту	Користувач переглядає графік дезінформації, проводить курсором по результатам стовпчастої діаграми.
Очікуваний результат	Графік описує найпопулярніші категорії дезінформації. Стовпці на графіку представляють категорії, для кожного тижня відбирається три найпопулярніші категорії. При наведені курсору на кожний стовпець, користувач може побачити кількість статей за останній тиждень з відповідною категорією.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.14 – Тест 6.3 Перегляд щотижневого дайджесту

Початковий стан системи	Користувач перебуває на сторінці «Тренди дезінформації»
Вхідні дані	-
Опис проведення тесту	Користувач бачить щотижневий дайджест Перевіряється правильне тексту.
Очікуваний результат	Текст дайджесту відображає поверхневий погляд на деякі розслідування, таким чином створюючи короткий погляд на деякі розслідування та їхній контекст.
Фактичний результат	Збігається з очікуваним.

4.3 Опис контрольного прикладу

При завантаженні головної сторінки, гість бачить перед собою головну сторінку новин, де користувач може вільно дивитися перелік останніх новин. Гість не зобов’язаний мати власний аккаунт для перегляду новин. Головну сторінку новин можна побачити на рисунку 4.5:

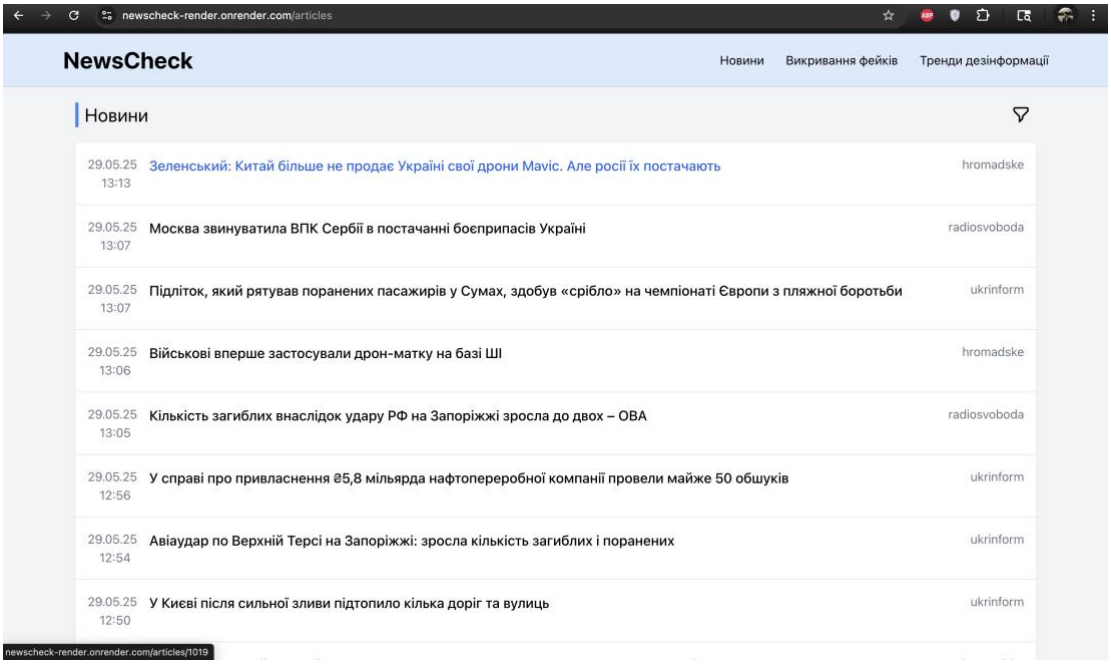


Рисунок 4.5 – Головна сторінка новин

Повний список новин розділений на сторінки, де на кожній сторінці знаходиться по 10 статей, розташовані у хронологічному порядку по спаданню:

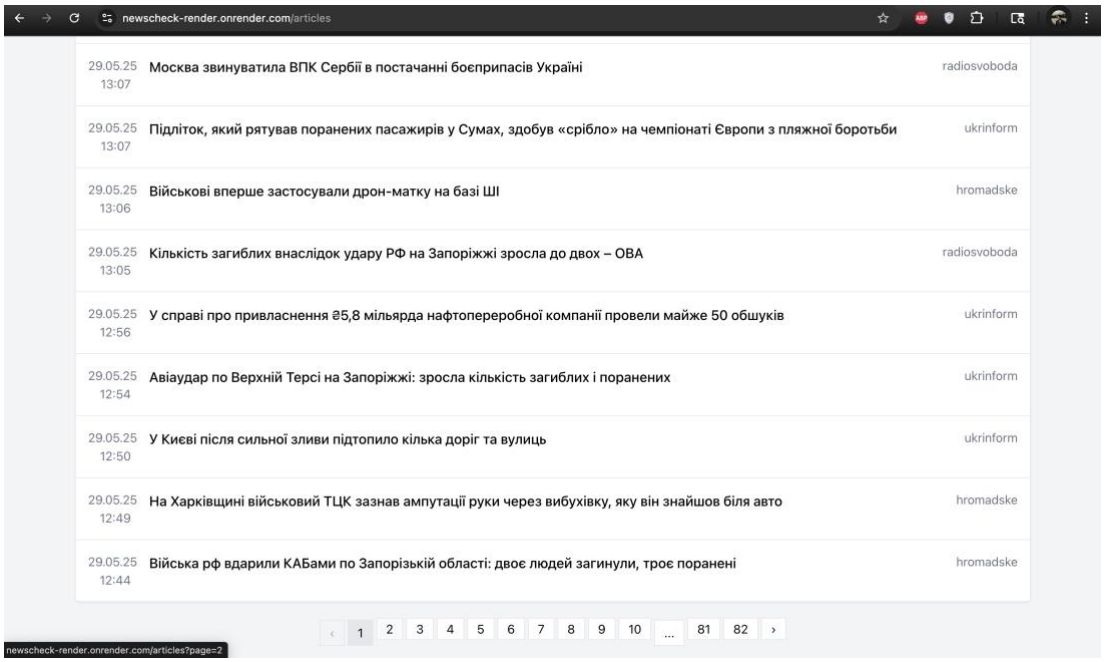


Рисунок 4.6 – Сторінка новин нижче, з пагінацією статей

Якщо користувач бажає подивитися зміст статті, то користувач робить це за допомогою натискання на заголовок бажаної статті. Після цього користувач зможе подивитися текст статті, заголовок, джерела, які були наведені у статті, а також оцінки від LLM щодо сенсаційності, фактичності, достовірності джерел і емоційності. При наведені на кожну з оцінок, користувач може подивитися обґрунтування кожної з оцінок.

Також користувач зможе подивитися статтю з оригінального джерела, де стаття була опублікована та звідки вона була взята за допомогою вебскрапінгу.

На рисунках 4.7 і 4.8 відповідно наведені скріншоти з однією зі статей, яка демонструє її функіонал:

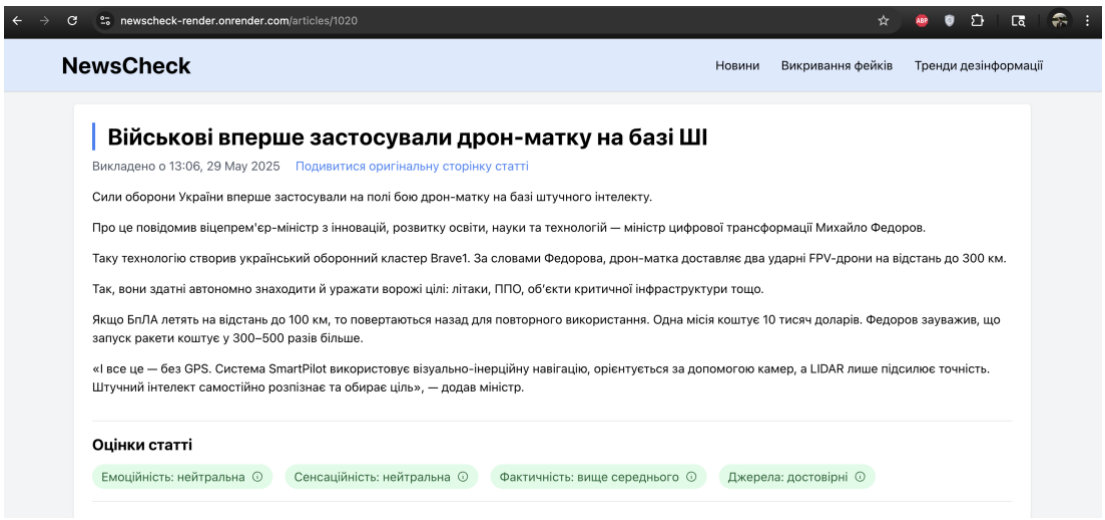


Рисунок 4.7 – Текст статті і її оцінки

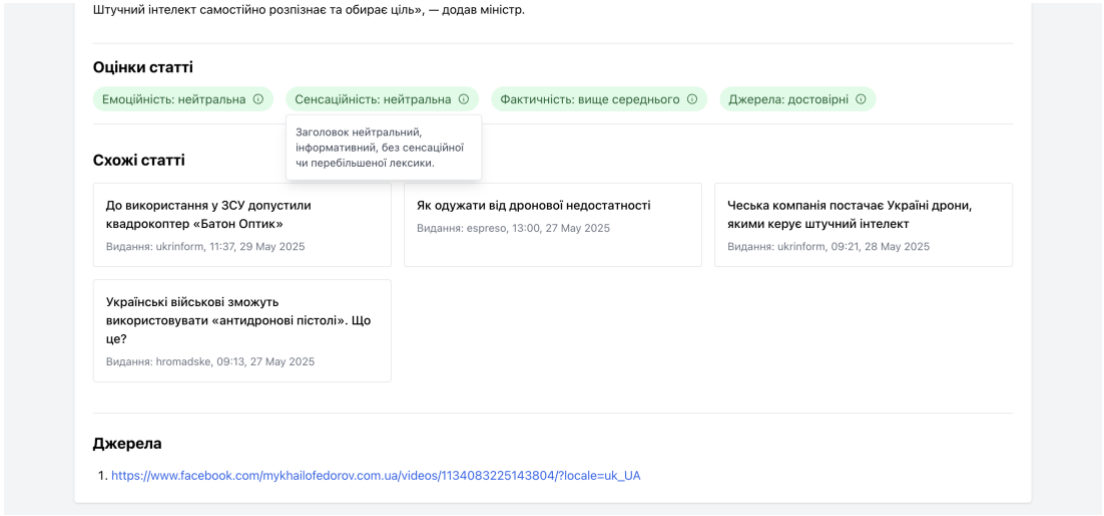


Рисунок 4.8 – Знайдені схожі статті, обґрунтування однієї з оцінок, знайдені джерела

Користувач також може застосувати фільтр новин для відбору новин за показниками якості, а також за виданнями. На рисунку 4.9 наведено приклад використання фільтру новин:

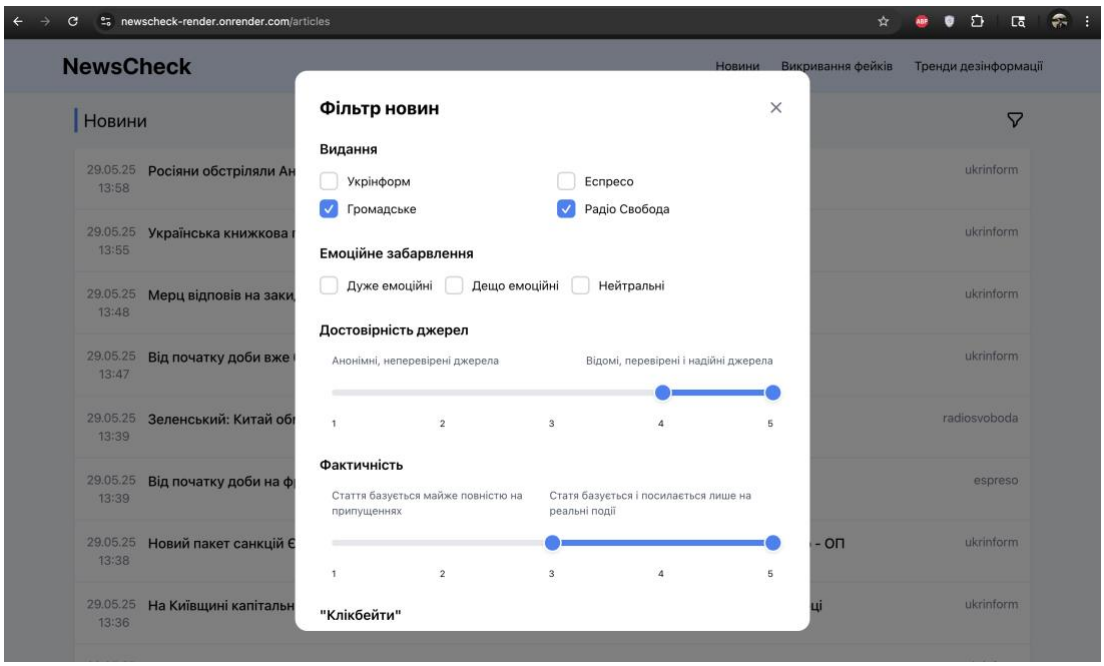


Рисунок 4.9 – Фільтр новин

Користувач може скасувати налаштування фільтру, при натисканні на кнопку «Скасувати налаштування», застосувати фільтр може користувач при натисканні на кнопку «Зберегти налаштування».

Список статей подається у тому самому форматі що і список новин, з пагінацією на 10 розслідувань на кожну сторінку. Кожна стаття має прев'ю з датою публікації, заголовком, та організацією, яке виклало це розслідування.

Користувач також може подивитися розслідування від факт-чекерів при натисканні на кнопку «Викривання фейків» зверху зліва. На рисунку 4.10 наведено екран зі списком розслідувань від факт-чекерів:

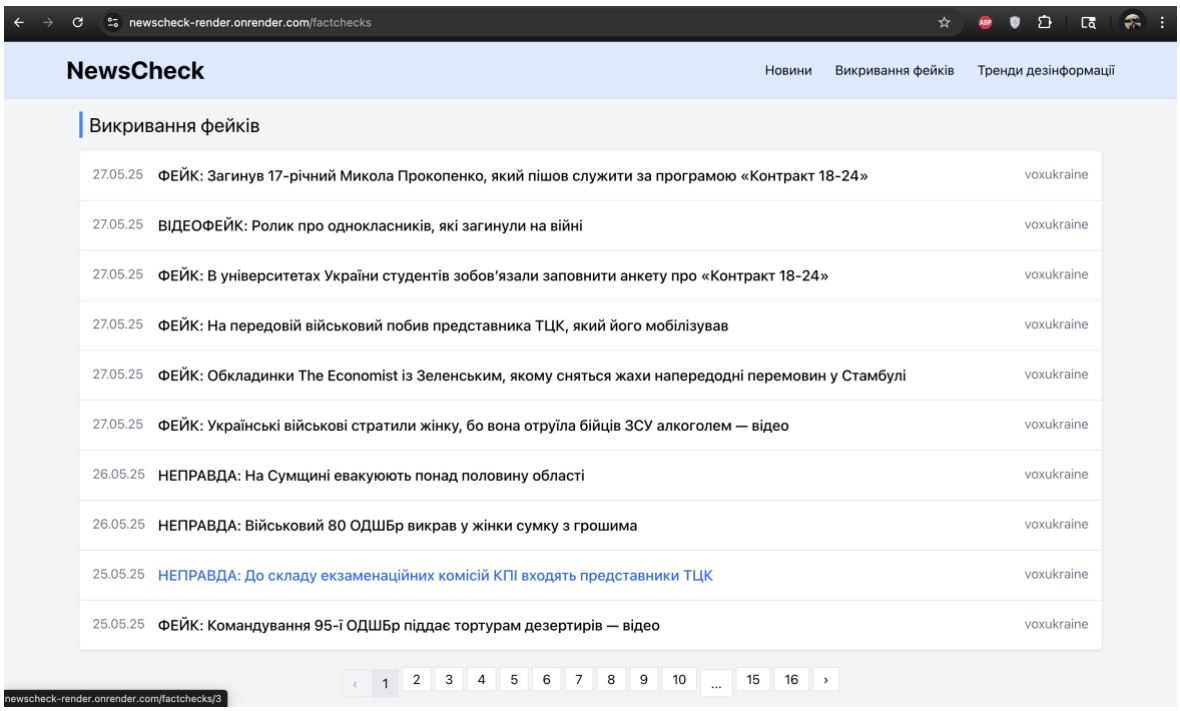


Рисунок 4.11 – Список розслідувань від факт-чекерів.

Користувач може переглянути розслідування аналогічно статті новин. Кожне розслідування має ті самі характеристики що й статті новин, окрім того, що розслідування мають категорії дезінформації, але не мають списку схожих розслідувань або оцінок від LLM. На рисунку 4.12 наведено приклад розслідування від факт-чекерів:

Рисунок 4.12 – Приклад розслідування від факт-чекерів

Остання сторінка – це сторінка з щотижневими статистиками дезінформації, а також їх щотижневим дайджестом. Зліва можна побачити графік дезінформації, який містить щотижневі кумулятивні заміри категорій дезінформації, які можна побачити у кожного розслідування під заголовком. Справа можна побачити дайджест дезінформації, який представляє собою текстовий огляд деяких розслідувань за останній тиждень. На рисунку 4.13 можна побачити сторінку «Тренди дезінформації»

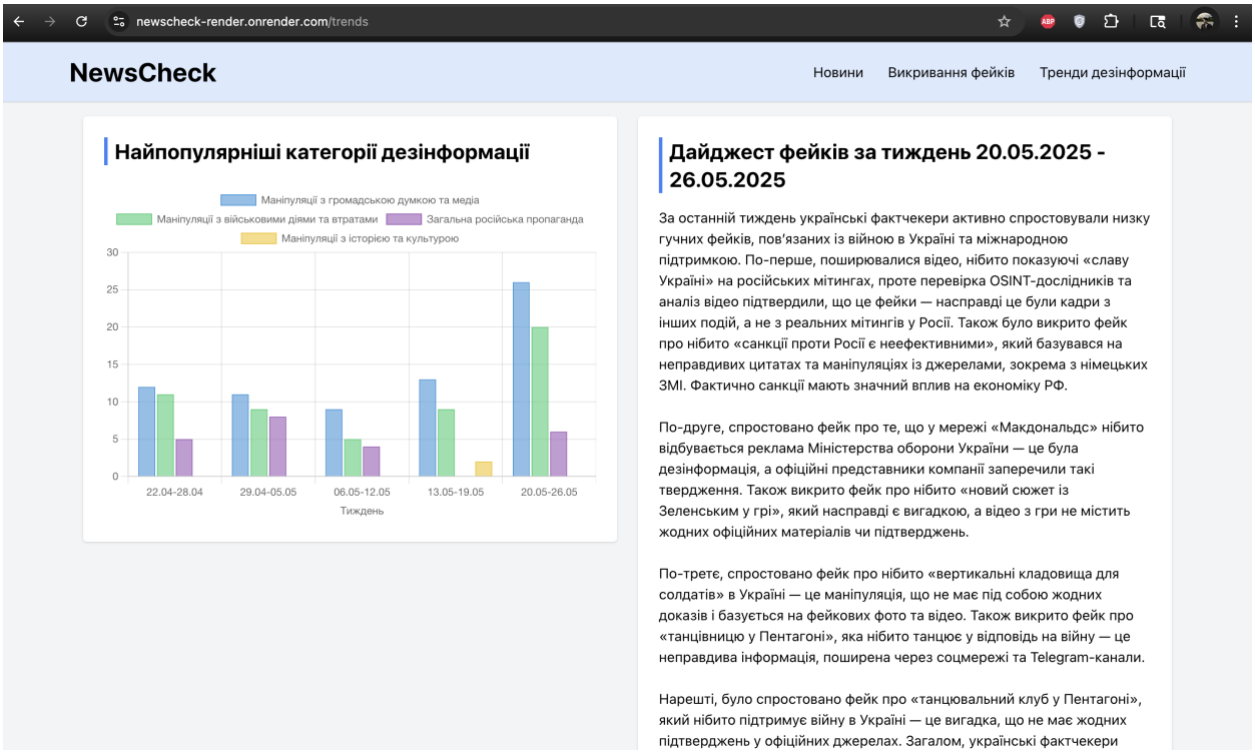


Рисунок 4.13 – Сторінка «Тренди дезінформації»

Висновки до розділу

У цьому розділі було проведено ретельний аналіз якості і тестування програмного забезпечення.

У першому підрозділі було здійснено аналіз якості програмного забезпечення за трьома ключовими метриками: швидкодія вебзастосунку з клієнтської сторони, навантаження пам’яті та процесора сервера з вебзастосунку, та швидкодія аналізу новин. Усі метрики повністю втілюють нефункціональні вимоги висунуті до програмного забезпечення.

У другому підрозділі було виконане мануальне тестування програмного забезпечення з 14 тест-кейсами, які повністю покривають ключовий функціонал та демонструють відповідність програмного забезпечення до функціональних вимог. Тестування включало перевірку переходу між сторінками, відображення новин і розслідувань, а також перегляд щотижневого дайджесту і графіку дезінформації.

В останньому підрозділі було детально описано контрольний приклад використання вебзастосунку, який демонструє повний цикл взаємодії користувача з системою. Контрольний приклад повністю охоплює три найголовніші сторінки застосунку «Новини», «Викривання фейків», та «Тренди дезінформації».

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Для розгортання програмного забезпечення було прийнято рішення використати платформу render.com, яка дозволяє швидко і надійно розгорнути різні види програмного забезпечення.

Програмне забезпечення застосунку складається з трьох ключових частин: вебзастосунок, база даних PostgreSQL та аналізатор новин і розслідувань. Оскільки останній компонент потребує дуже велику кількість ресурсів для вебскрапінгу, та аналізу новин в тому числі за допомогою векторної бази даних, яку також треба розгорнути, було прийнято рішення розгорнути лише вебзастусонок і базу даних PostgreSQL. Аналізатор новин буде взаємодіяти з розгорнутою бази даних локально. Тим часом вебзастосунок також зможе взаємодіяти з розгорнутою базою даних, таким чином ніяк не впливаючи на функціонал.

База даних PostgreSQL розгортається найлегше, тому її було розгорнуто з самого початку. Для цього, переходимо на головну сторінку та обираємо опцію “New”, та обираємо PostgreSQL. Розгортання бази даних не потребує конфігураційних файлів. Обираємо ім’я бази даних та додаємо користувачів. Оскільки застосунок не передбачає взаємодії користувачів з застосунком з операціями створення або написання, то створюємо лише користувача “admin”. У регіоні вказуємо Frankfurt (EU Central) щоб запобігти високим затримкам через великі відстані до серверів. Також вказуємо налаштування процесорів і пам’яті. Оскільки бізнес-процеси не потребують багато ресурсів, то було прийнято рішення взяти безкоштовний варіант бази даних з 256 МБ оперативної пам’яті, 0.1 ЦП, та 1 ГБ постійного сховища. Більше не потрібно вказувати налаштувань для бази даних, тому натискаємо “Create Database”.

На рисунку 5.1 показано успішно розгорнуту базу даних newscheck-PostgreSQL:

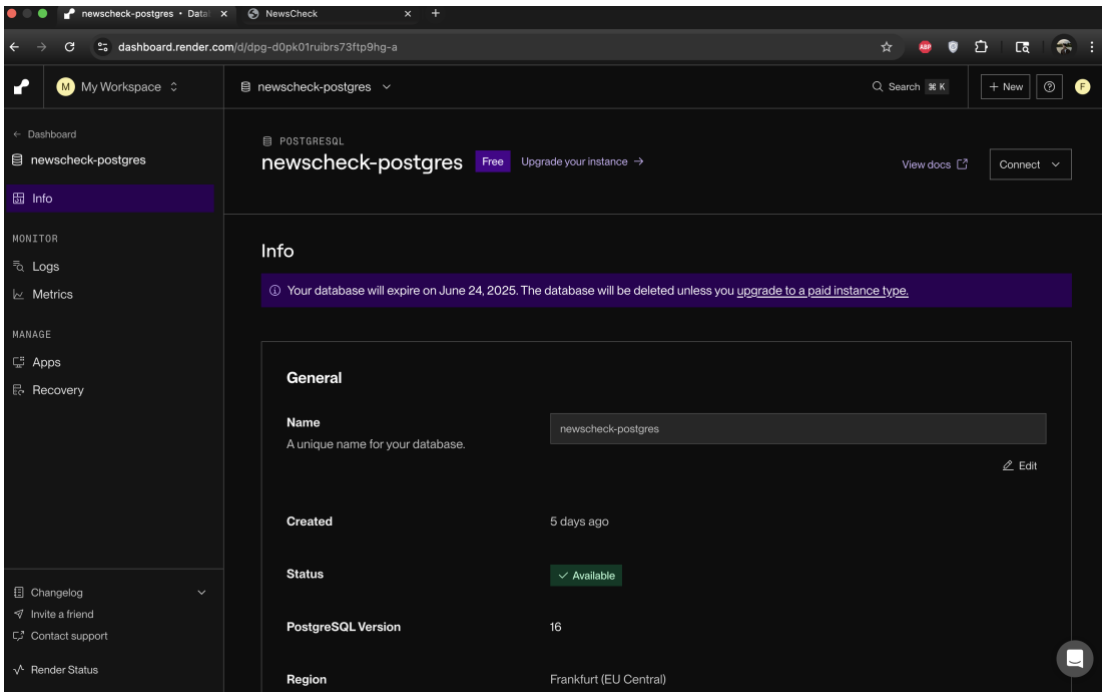


Рисунок 5.1 – Успішно розгорнута база даних newscheck-postgres

Після успішного розгортання, було виконано підключення до бази даних за допомогою інструментів Render CLI через локальний термінал та створено усі необхідні таблиці для бази даних.

Наступний крок – це розгортання застосунку фреймворку Laravel на render.com. Для цього, обираємо “New” на головній сторінці, та обираємо “Web Service”. При натисканні цієї опції, render автоматично запросить доступ до репозиторію Git з застосунком та конфігураційними файлами. Тому даємо доступ до репозиторію.

При отриманні доступу до репозиторію, маємо вказати назву застосунку. Після цього, обираємо у Language опцію «docker», оскільки розгортання виконується за допомогою Dockerfile. Також вказуємо гілку репозиторію, з якої варто виконати розгортання, вказуємо master. Аналогічно базі даних, вказуємо регіон Frankfurt (EU Central).

Для потреб пам'яті і процесора, було обрано опцію на 512 МБ оперативної пам’яті та 0.5 ЦП.

Також є опції для більш складних налаштувань, такі як власний шлях до конфігураційних файлів, файлів зі змінними середовища, і так далі. Дані опції

не було використано, оскільки багато з них можна вказати у конфігураційних файлах. Тому обираємо Deploy Web Service.

Після цього, render автоматично почне розгортання відповідно до останніх змін у репозиторії. Після запуску усіх команд з конфігураційних файлів сайт буде успішно розгорнутий та доступний для всіх за посиланням: https://newscheck-render.onrender.com

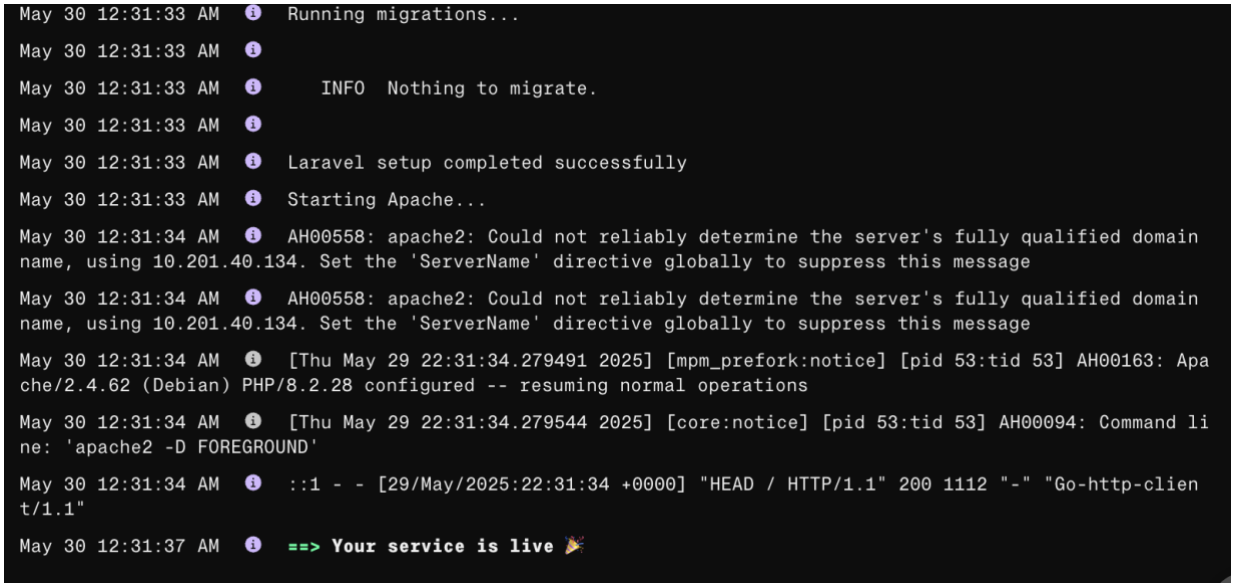


Рисунок 5.2 – Повідомлення з логів про успішне розгортання застосунку

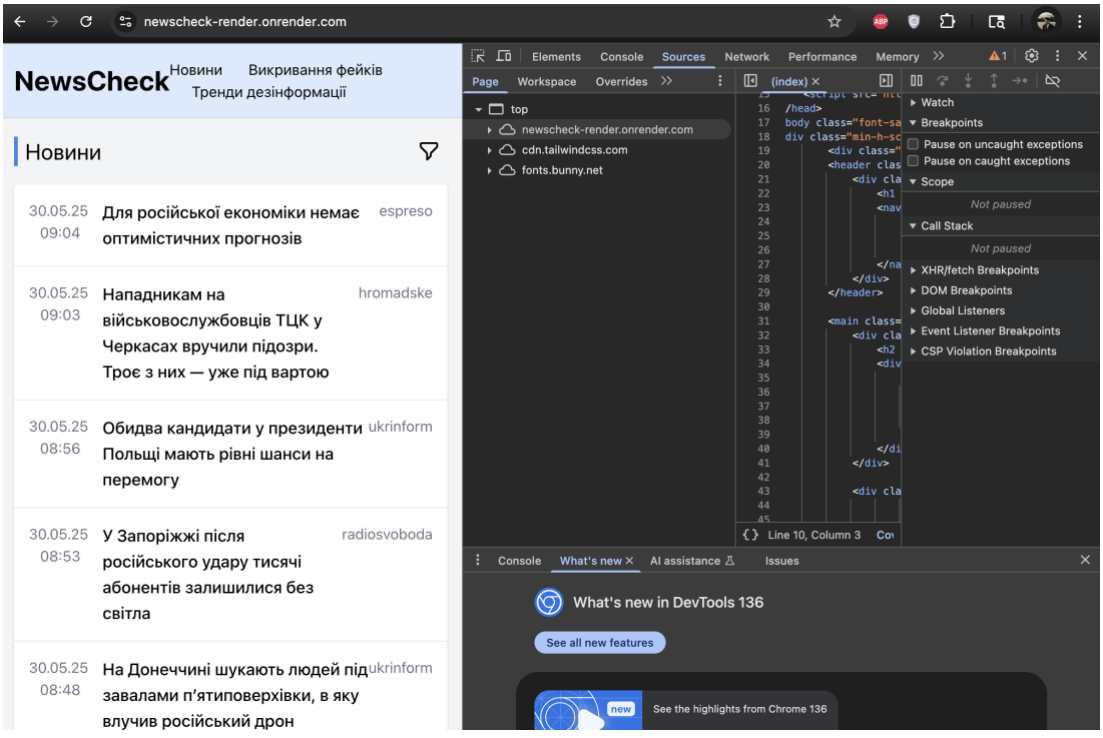


Рисунок 5.3 – Завантаження сторінки розгорнутого вебзастосунку

Для розгортання було використано декілька конфігураційних файлів.

По-перше, було використано файл Dockerfile, як було вказано раніше. У цьому файлі вказано основні команди, які були використані для розгортання.

```
FROM php:8.2-apache

WORKDIR /var/www/html

RUN apt-get update && apt-get install -y \
    git \
    curl \
    libpng-dev \
    libonig-dev \
    libxml2-dev \
    libpq-dev \
    zip \
    unzip \
    && docker-php-ext-install pdo pdo_pgsql mbstring exif pcntl bcmath gd \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

COPY --from=composer:latest /usr/bin/composer /usr/bin/composer

COPY . /var/www/html

RUN composer install --no-dev --optimize-autoloader
```

Рисунок 5.4 – Частковий скріншот Dockerfile

Було прийнято рішення використати Apache Server для хостингу сайту. Також було налаштовано директорію для збереження файлів проєкту, завантажено composer, тобто менеджер пакетів для PHP, який широко використовується у Laravel. Завантажуються інші інструменти, наприклад git, curl, та інші.

Також копіюються конфігураційні файли apache.conf, який містить налаштування сервера, і також файл start.sh, який запускає деякі конфігураційні команди для PHP для створення налаштувань для роботи з базою даних, копіювання файлу з секретними змінними, наприклад даними для підключення до бази даних, очищення кешів, та інші. Останній також виводить у консоль допоміжні повідомлення про результати операцій, що спрощує розгортання у майбутньому.

На рисунках 5.5 і 5.6 наведені скріншоти з конфігураційними файлами apache.conf та start.sh.

```
<VirtualHost *:80>
    ServerName localhost
    DocumentRoot /var/www/html/public

    <Directory /var/www/html/public>
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Рисунок 5.5 – Скріншот файлу apache.conf

```
#!/bin/bash

if [ ! -f /var/www/html/.env ]; then
    echo "Creating .env file from .env.example..."
    cp /var/www/html/.env.example /var/www/html/.env
    echo ".env file created successfully"
else
    echo ".env file already exists"
fi

echo "=== Environment Variables ==="
echo "DB_CONNECTION: $DB_CONNECTION"
echo "DB_HOST: $DB_HOST"
echo "DB_PORT: $DB_PORT"
echo "DB_DATABASE: $DB_DATABASE"
echo "DB_USERNAME: $DB_USERNAME"
echo "DB_PASSWORD: [HIDDEN]"
echo "APP_KEY: $APP_KEY"
echo "============================"
```

Рисунок 5.6 – Частковий скріншот файлу start.sh

Для локального розгортання аналізатора новин, були використані Redis та Celery, які надають зручний інтерфейс у Python. Було створено файл celery_app.py, який дозволяє з певним проміжком часу виконувати завдання.

Відповідно, наступні задачі виконуються з наступним інтервалом:

– пошук і аналіз новин: кожні 15 хвилин;
– виконання семантичного аналізу для пошуку схожих новин: кожна година;
– пошук і аналіз розслідувань від факт-чекерів: кожні 24 години;
– створення щотижнвої статистики і дайджесту: кожні 7 днів.

Налаштування відповідних задач наведено на рисунку 5.7:

```python
app = Celery( main: 'tasks',
            broker='redis://localhost:6379/0',
            include=['tasks'])

app.conf.beat_schedule = {
    'task-news-analysis': {
        'task': 'tasks.news_analysis',
        'schedule': 15 * 60,
    },
    'task-fact-recommendations_update': {
        'task': 'tasks.refresh_recommendations',
        'schedule': 60 * 60,
    },
    'task-fact-checkers-analysis': {
        'task': 'tasks.fact_checkers_analysis',
        'schedule': 24 * 60 * 60,
    },
    'task-weekly-digest': {
        'task': 'tasks.weekly_digest',
        'schedule': 7 * 24 * 60 * 60,
    },
}
```

Рисунок 5.7 – Налаштування інтервалів виконання задач у Celery

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі.

Для отримання нової версії програмного забезпечення для користувачів, потрібно створити зміни у репозиторії та зберегти їх на головній гілці master. Render автоматично помічає будь-які зміни у репозиторії і виконує розгортання автоматично. Також з Render можна виконати розгортання конкретної версії програмного забезпечення, якщо така потреба з’явиться.

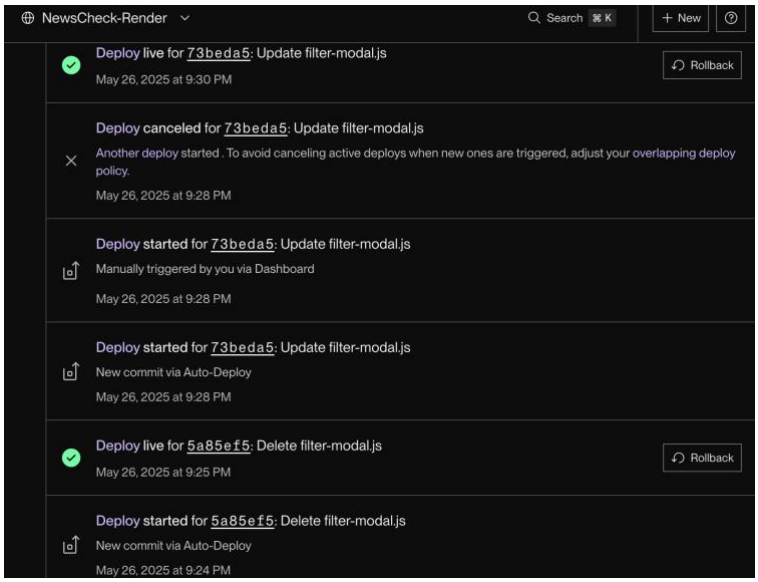


Рисунок 5.8 – Приклад розгортання кількох версій вебзастосунку

Висновки до розділу

У цьому розділі було описано процес супроводу та розгортання програмного забезпечення.

У першому підрозділі було виконано розгортання програмного забезпечення на платформі render.com. Було виконано розгортання бази даних PostgreSQL та вебзастосунок з використанням контейнеризації за допомогою Docker. Аналізатор новин було розгорнуто локально за допомогою Celery та Redis з автоматичного виконання завдань за вказаним розкладом.

У другому підрозділі було описано процес супроводу програмного забезпечення. Render надає автоматичне розгортання при внесені змін до репозиторію GitHub, що забезпечує швидке оновлення системи.

ВИСНОВКИ

У ході виконання дипломного проєкту було розроблено повністю функціональний вебзастосунок NewsCheck для агрегації та інтелектуального аналізу новинного контенту. Система включає автоматизований збір новин з деяких видань, та використовує засоби LLM для аналізу якості за метриками емоційності, сенсаційності, фактичності та достовірності джерел, а також векторні бази даних для пошуку схожих новин за допомогою семантичного аналізу тексту.

Була втілена мета дипломного проєктування – покращення доступу користувачів до достовірної інформації шляхом системи автоматизованого контролю якості новинного контенту.

Система надає користувачам дивитися оцінки новинних статей, переглядати розслідування факт-чекерів щодо викривання фейків та спостерігати тренди дезінформації. Використання сучасних великих мовних моделей дозволяє проводити оцінювання статей та наводити обґрунтування оцінки, даючи користувачам можливість робити власні оцінки статті у випадку якщо стаття оцінена некоректно та користувачі з цим не згодні.

Програмне забезпечення проєкту було виконане за допомогою мови програмування Python і фреймворку Laravel, для реалізації аналізатора новин і розслідувань, і реалізації вебзастосунку відповідно. Також було використано модель GPT-4.1-mini від OpenAI, а для векторизації тексту для отримання векторів ембедингів було використано модель Voyage-3 від VoyageAI, обидві з яких визнані як дуже якісні моделі з низькими ймовірностями відмови, а також прийнятними цінами за свої послуги.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. Science, 359(6380), 1146-1151. https://doi.org/10.1126/science.aap9559

2) Starbird, K., Spiro, E., Edwards, I., et al. (2023). Information warfare during armed conflict: Lessons from Ukraine. Proceedings of the CHI Conference on Human Factors in Computing Systems. https://doi.org/10.1145/3544548.3581518

3) Wang, W. Y. (2017). "Liar, Liar Pants on Fire": A new benchmark dataset for fake news detection. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-2067

4) VoxUkraine: https://voxukraine.org/category/voks-informue

5) StopFake: https://www.stopfake.org/uk/category/context-ua/

6) Espreso.tv - https://espreso.tv/

7) Hromadske - https://hromadske.ua/

8) Ukrinform - https://www.ukrinform.ua/

9) Радіо «Свобода» - https://www.radiosvoboda.org/

10) Edelman Trust Barometer : https://www.edelman.com/trust/2023/trust-barometer

11) Фейк про «біолабораторії» в Україні: https://voxukraine.org/fejk-biolaboratoriyi-ssha-v-ukrayini-tse-voyenni-ob-yekty-nato

12) Фейки про події у Бучі: https://voxukraine.org/fejk-u-buchi-ukrayinski-vijskovi-obstrilyuvaly-budynky-tyh-hto-brav-dopomogu-vid-rosiyan

13) Фейки про пошкодження інфраструктури: https://voxukraine.org/fejk-cherez-poshkodzhennya-energetychnoyi-infrastruktury-v-ukrayini-pochnutsya-spalahy-tyfu-ta-chumy

14) Google News: https://news.google.com/home

15) Ground News: https://ground.news/

16) NewsGuard: https://www.newsguardtech.com/

17) DOU.ua: https://jobs.dou.ua/salaries/?period=2024-12&position=Software%20Engineer

18) What is monolithic architecture? https://www.ibm.com/think/topics/monolithic-architecture

19) Microservice Architecture Pattern: https://microservices.io/patterns/microservices.html

20) The C4 model for visualizing software architecture: https://c4model.com/

21) Laravel: https://laravel.com/

22) OpenAI API: https://platform.openai.com/docs/overview

23) VoyageAI API: https://www.voyageai.com/

24) PostgreSQL: https://www.postgresql.org/

25) Milvus: https://milvus.io/

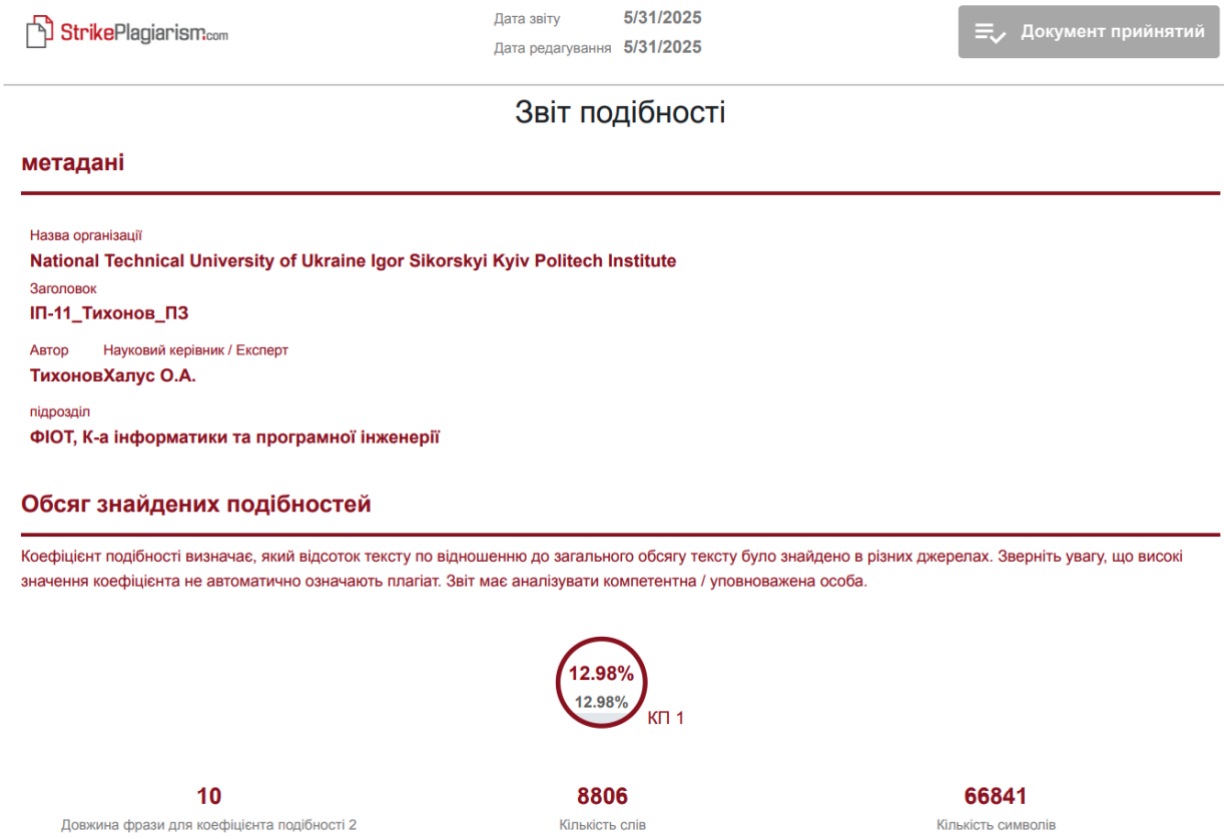
26) Requests: https://pypi.org/project/requests/

27) Selenium: https://www.selenium.dev/

28) Render.com: https://render.com/

ДОДАТКИ

Додаток А Звіт подібності



StrikePlagiarism.com

Дата звіту 5/31/2025
Дата редагування 5/31/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute
Заголовок
ІП-11_Тихонов_ПЗ
Автор Науковий керівник / Експерт
ТихоновХалус О.А.
підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

12.98%
12.98%
КП 1

10	8806	66841
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів