**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**
Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

ДО ЗАХИСТУ ДОПУЩЕНО
В.о. завідувача кафедри
_____ Олександр КОВАЛЬ
«___»_____2026 р.

# Дипломна робота

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці»**
**спеціальності 121 Інженерія програмного забезпечення**

**на тему: «Система машинного навчання для аналізу і групування природної та штучно створеної інформації»**

Виконала:
студентка IV курсу, групи ТВ-21
Сатко Дар'я Олегівна _____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник:
Професор, д.т.н., доцент Недашківський О.Л. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент:
Професор, д.т.н., професор Отрох С.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Київ – 2026

**Національний технічний університет України**
**«Київський політехнічний інститут імені Ігоря Сікорського»**

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Олександр КОВАЛЬ
«___»_____2026р.

# ЗАВДАННЯ

**на дипломну роботу студенту**

Сатко Дар'ї Олегівні
(прізвище, ім'я, по батькові)

1. Тема роботи: Система машинного навчання для аналізу і групування природної та штучно створеної інформації
керівник роботи Недашківський Олексій Леонідович, д.т.н., доцент
( прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «28» травня 2026 р. №1992-с
2. Строк подання студентом роботи «10» червня 2026 р.
3. Вихідні дані до роботи: мова програмування Python, мова програмування JavaScript, середовище розробки Visual Studio Code
4. Зміст (дипломної роботи) пояснювальної записки (перелік завдань, які потрібно розробити): Розробка модулю збору та попередньої обробки даних, блоку лінгвістичного аналізу природної мови, модулю розпізнавання ознак синтетичного контенту, блоку групування та кластеризації інформації, системи оцінки достовірності даних, панелі візуалізації результатів аналітики, блоку збереження та керування наборами даних.
5. Перелік ілюстративного матеріалу: діаграма прецедентів, діаграма розгортання, діаграма компонентів, діаграма класів

6. Дата видачі завдання «31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання дипломної роботи | Строки виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Отримання завдання | 31.10.2025 | Виконано |
| 2 | Дослідження предметної області | 01.11.2025 – 31.11.2025 | Виконано |
| 3 | Дослідження існуючих рішень | 01.12.2025 – 31.12.2025 | Виконано |
| 4 | Постановка вимог до проєктування системи | 01.01.2026 – 31.01.2026 | Виконано |
| 5 | Розробка програмного продукту | 01.02.2026 – 03.05.2026 | Виконано |
| 6 | Тестування | 04.05.2026 – 10.05.2026 | Виконано |
| 7 | Захист програмного продукту | 11.05.2026 – 15.05.2026 | Виконано |
| 8 | Оформлення дипломної роботи | 18.05.2026 – 31.05.2026 | Виконано |
| 9 | Передзахист | 01.06.2026 – 05.06.2026 | Виконано |
| 10 | Захист | 15.06.2026 – 19.06.2026 | Виконано |

Студентка _____ Дар’я САТКО
(підпис) (ім’я, ПРІЗВИЩЕ)

Керівник роботи _____ Олексій НЕДАШКІВСЬКИЙ
(підпис) (ім’я, ПРІЗВИЩЕ)

# РЕФЕРАТ

**Структура та обсяг дипломної роботи.** Робота містить 52 сторінки, 16 рисунків, 2 додатки та 19 посилань.

**Метою роботи** є удосконалення та розширення функціоналу програмного забезпечення на базі систем машинного навчання для інтелектуального аналізу і групування природної та штучно створеної інформації в межах навчального закладу для автоматизованого контролю академічної доброчесності.

**Практичне значення одержаних результатів** полягає в створенні автономного, повністю готового до впровадження інтелектуального цифрового продукту, розгортання якого в локальній інфраструктурі навчального закладу забезпечує абсолютну конфіденційність перевірок. Система надає викладачам та адміністраторам зручний інструмент для автоматизованої перевірки студентських робіт, візуалізації розподілу авторства через інтерактивну смужку, детального логічного пояснення результатів аналізу та миттєвого формування офіційних звітів у форматі PDF, що суттєво підвищує якість моніторингу академічної доброчесності.

**Ключові слова:** штучний інтелект, машинне навчання, обробка природної мови, академічна доброчесність, трансформер, розпізнавання ші-тексту, ансамблевий алгоритм, fastapi, docker, семантична класифікація, інженерія програмного забезпечення.

# ABSTRACT

**Structure and scope of the thesis.** The work contains 52 pages, 16 figures, 2 appendices and 19 references.

**The purpose of the work** is to improve and expand the functionality of software based on machine learning systems for the intelligent analysis and clustering of natural and artificially generated information within an educational institution for the automated monitoring of academic integrity.

**The practical value** of these results lies in the creation of a standalone, fully implementation-ready intelligent digital product, the deployment of which within the college's local infrastructure ensures absolute confidentiality of assessments. The system provides instructors and administrators with a convenient tool for automated checking of student work, visualization of authorship distribution via an interactive bar chart, detailed logical explanations of analysis results, and instant generation of official reports in PDF format, which significantly improves the quality of academic integrity monitoring.

**Keywords:** artificial intelligence, machine learning, natural language processing, academic integrity, Transformer, text recognition, ensemble algorithm, FastAPI, Docker, semantic classification, software engineering.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – прикладний програмний інтерфейс;

BART – модель глибокого машинного навчання на базі архітектури Transformer, яка поєднує двонаправлений кодувальник та авторегресійний розкодувальник;

F1-score –гармонічне середнє між точністю (Precision) та повнотою (Recall);

FastAPI – сучасний, високопродуктивний вебфреймворк для створення прикладних програмних інтерфейсів на мові програмування Python з підтримкою асинхронності.

HTTP –протокол передачі гіпертексту.

JSON – текстовий формат обміну даними, який легко читається людьми та машинами.

LLM – велика мовна модель; нейронна мережа з мільярдами параметрів, навчена на масивних обсягах текстових даних, здатна розуміти та генерувати природну мову.

ML – машинне навчання; підгалузь штучного інтелекту, що вивчає алгоритми, які здатні навчатися на даних і робити передбачення без прямого програмування.

RoBERTa – оптимізована модель обробки природної мови від компанії Meta;

SQL – мова структурованих запитів.

UML – уніфікована мова моделювання;

БД – база даних.

Евристичний аналіз – метод розв'язання задач, який базується на використанні практичних правил, досвіду та інтуїції замість строгих математичних алгоритмів.

Контейнеризація – метод віртуалізації на рівні операційної системи, який дозволяє запускати програмне забезпечення у повністю ізольованих середовищах.

СУБД – система управління базами даних.

ШІ – штучний інтелект.

# ВСТУП

Стрімкий розвиток генеративних моделей штучного інтелекту сильно змінив підходи до створення текстового контенту. Ці технології створюють серйозні проблеми для академічної доброчесності та інформаційної безпеки. Актуальність теми зумовлена великою потребою суспільства в удосконаленні методів боротьби з дезінформацією та необхідністю розробки інформаційних технологій визначення тональності та класифікації текстового контексту на основі нейромережевих методів. Це повністю відповідає сучасним пріоритетним тематикам наукових досліджень тактуальним наказам Міністерства освіти і науки України.

Метою роботи є покращення та розширення функціоналу програмного забезпечення на базі систем машинного навчання для аналізу і групування природної та штучно створеної інформації. Для досягнення цієї мети було вирішено розробити комплексний інструмент здатний об'єднати глибокий аналіз авторства тексту розпізнавання машинної генерації перевірку на схожість та аналіз емоційного забарвлення в єдиному зручному програмному середовищі.

Для реалізації поставленої мети було визначено та виконано низку послідовних завдань. Спочатку необхідно проаналізувати існуючі рішення та розроблене програмне забезпечення систем машинного навчання у сфері обробки природної мови. Далі потрібно визначити чіткі вимоги до функціональності програмного інтерфейсу та практичної основи створюваного продукту. Наступним ключовим кроком є удосконалення методів машинного навчання шляхом розробки спеціального ансамблевого підходу для точного аналізу змішаних текстів де комбінуються фрагменти написані людиною та згенеровані машиною. Після цього вимагається розробити удосконалену серверну архітектуру та програмно реалізувати повноцінну систему з вебфронтендом та базою даних. Наостанок необхідно провести ретельну верифікацію показників якості функціонування розроблених алгоритмів на реальних текстових масивах.

Практичне значення одержаних результатів полягає у фізичному впровадженні сучасного цифрового продукту який суттєво підвищить якість боротьби з дезінформацією. Розроблена система надає викладачам та адміністраторам потужний інструмент для автоматизованої багатокритеріальної перевірки документів візуалізації розподілу авторства та миттєвого формування детальних звітів. Завдяки використанню сучасних підходів до контейнеризації програмне забезпечення відзначається високою стабільністю роботи та є повністю готовим до інтеграції в цифрову інфраструктуру будь-якого навчального закладу.

# 1 ЗАДАЧІ ТА ВИМОГИ ДО СИСТЕМИ МАШИННОГО НАВЧАННЯ ДЛЯ АНАЛІЗУ І ГРУПУВАННЯ ІНФОРМАЦІЇ

Розділ присвячено детальному розгляду базових завдань та встановленню чітких критеріїв до розроблюваного програмного забезпечення. Ефективність будь-якого інформаційного продукту безпосередньо залежить від правильно сформульованих початкових умов та розуміння кінцевої мети його використання у реальному освітньому чи корпоративному середовищі.

## 1.1 Постановка задачі

Основною проблемою яку має вирішити розроблювана система є складність ідентифікації походження текстових матеріалів умовах масового поширення генеративних нейромереж. Сучасний освітній процес вимагає від викладачів витрачати значну кількість часу на ручну перевірку студентських робіт щодо їхньої академічної доброчесності та оригінальності. Звичайні антиплагіатні системи вже не здатні впоратися з унікальними текстами згенерованими сучасними мовними моделями. Тому задача полягає у створенні принципово нового автоматизованого комплексу здатного приймати текстові дані та проводити їх глибокий семантичний і структурний аналіз.

Програмне забезпечення повинно не просто видавати загальний показник ймовірності використання штучного інтелекту а й детально розпізнавати складні змішані тексти, де авторський матеріал поєднується з машинною генерацією та перефразуванням. Додатковою задачею є реалізація супутніх аналітичних функцій таких, як визначення емоційного забарвлення тексту автоматична класифікація за тематичними категоріями та виявлення рівня класичного внутрішнього плагіату шляхом порівняння надісланих матеріалів із раніше збереженою базою робіт. Усі

ці складні обчислювальні процеси мають відбуватися непомітно для кінцевого користувача у єдиному інтуїтивно зрозумілому вебінтерфейсі.

## 1.2 Вимоги до системи

Для успішної та безперебійної роботи програмний продукт має відповідати суворому набору функціональних тархітектурних вимог. З точки зору базової функціональності, система повинна забезпечувати можливість реєстрації та авторизації користувачів із розподілом на різні ролі, зокрема адміністраторів закладу та викладачів. Головний модуль аналізу зобов'язаний підтримувати обробку як прямого вводу тексту так і автоматичне розпізнавання популярних форматів електронних документів, включно з текстовими процесорами та портативними форматами.

Критично важливою вимогою до аналітичного ядра є реалізація ансамблевого алгоритму для поабзацного аналізу авторства тексту. Система має розраховувати точні структурні частки повністю згенерованого перефразованого та оригінального матеріалу. Алгоритм прийняття рішень повинен комплексно враховувати як висновки глибоких математичних моделей на базі трансформерів так і наявність типових лексичних шаблонів перевірених евристичним модулем. Якщо результати різних модулів суперечать один одному система має видавати аргументований логічний висновок для користувача.

Щодо нефункціональних вимог система має відзначатися високою швидкістю обробки запитів та оптимізованим споживанням оперативної пам'яті сервера. Завантаження важких нейромережевих моделей має відбуватися виключно лінивим методом під час безпосереднього звернення до функцій аналізу, для уникнення перевантаження серверногобладнання на етапі запуску. Програмний комплекс повинен розгортатися за допомогою технологій контейнеризації, що забезпечить його повну незалежність від операційної системи та швидке налаштування. Користувацький інтерфейс має коректно відображатися

на різних пристроях та надавати результати перевірки у вигляді наочних графіків з можливістю генерації офіційного протоколу перевірки.

## 1.3. Теоретичні основи обробки природної мови

Обробка природної мови є фундаментальним напрямком штучного інтелекту, який вивчає методи та алгоритми взаємодії між обчислювальними системами та людською мовою. Головною метою цієї дисципліни є створення математичних моделей, здатних аналізувати, інтерпретувати та генерувати текстові дані у спосіб, що максимально наближений до семантичного сприйняття людини. Історично розвиток цієї галузі пройшов складний шлях від ранніх експертних систем на основі жорстких лінгвістичних правил до ймовірнісних методів і зрештою до сучасних нейромережевих архітектур глибокого навчання.

Первинним етапом аналізу будь-якого масиву тексту є токенізація. Цей процес полягає у декомпозиції суцільного потоку символів на окремі значущі одиниці, які називаються токенами. Залежно від обраного алгоритму токенами можуть виступати окремі слова, частини слів або навіть окремі символи. Після етапу токенізації до тексту застосовуються методи морфологічної нормалізації, зокрема лематизація та стемінг. Зазначені процедури дозволяють звести слова до їхньої базової словникової форми або відкинути змінні закінчення, що суттєво зменшує загальну розмірність словника ознак та значно спрощує подальший обчислювальний процес.

Оскільки штучні нейронні мережі не здатні безпосередньо сприймати символьні дані, наступним критичним кроком виступає векторизація тексту. На ранніх етапах розвитку обробки природної мови застосовувалися базові статистичні підходи, наприклад метод мішка слів та метрика частотності, яка математично оцінює важливість конкретного слова в контексті всього досліджуваного документа. Пізніше інноваційним кроком стала поява щільних векторних представлень слів. Ці технології дозволили трансформувати слова у

багатовимірні вектори дійсних чисел, де просторова геометрична відстань між векторами відображає ступінь семантичної та синтаксичної близькості відповідних понять.

Тривалий час галузевим стандартом для обробки послідовностей слів залишалися рекурентні нейронні мережі та архітектури довгої короткострокової пам'яті. Зазначені моделі обробляли текст суворо послідовно, крок за кроком переносячи інформацію про попередні слова у свій внутрішній прихований стан. Проте такі архітектурні рішення мали суттєві обмеження, серед яких повна неможливість паралельних обчислень та критична проблема затухання градієнта під час тренування на великих обсягах даних. Це призводило до неминучої втрати контексту на великих відстанях у межах одного документа.

## 1.4. Еволюція архітектури Transformer та великих мовних моделей

Справжнім переломним моментом у сфері аналізу текстової інформації стала публікація досліджень щодо нової архітектури Transformer у дві тисячі сімнадцятому році. Вчені запропонували повністю відмовитися від рекурентних зв'язків на користь інноваційного математичного механізму уваги, відомого як механізм внутрішньої уваги. Завдяки цьому підходу нейромережева модель отримала здатність одночасно аналізувати всі слова у вхідній послідовності даних та самостійно визначати вагу логічних зв'язків між ними абсолютно незалежно від їхньої фізичної відстані у структурі речення.

Механізм внутрішньої уваги обчислює три окремі векторні представлення для кожного токена: матрицю запитів Q, матрицю ключів K та матрицю значень V. Шляхом матричного множення цих векторних просторів обчислювальна система розраховує коефіцієнти уваги, які кількісно визначають ступінь впливу кожного слова на розуміння інших слів у поточному контексті. Цей математичний апарат формалізується наступним рівнянням:

$$\text{Attention}[Q, K, V] = \text{softmax}\left[\frac{QK^T}{\sqrt{d_k}}\right]V$$

Такий підхід дозволив повністю подолати проблему втрати довгострокових залежностей та відкрив безпрецедентні можливості для масового розпаралелювання обчислювальних процесів на графічних прискорювачах. Початкова архітектура трансформера містила два незалежні блоки: кодувальник та розкодувальник. Згодом дослідницька спільнота виявила, що ці блоки демонструють найвищу ефективність при ізольованому використанні для розв'язання різних класів інформаційних задач.

На базі ізольованого кодувальника було спроєктовано модель двонаправленого аналізу та її оптимізовану версію RoBERTa, яка виступає головним математичним ядром у розробленому дипломному проєкті. Ці моделі найкраще підходять для завдань глибокого розуміння тексту, семантичної класифікації та пошуку прихованих структурних закономірностей. Натомість на базі розкодувальника сформувалося сімейство генеративних мереж, які тренувалися виключно на передбачення наступного токена і згодом стали технологічною основою для сучасних інтелектуальних чат-ботів.

Подальша еволюція описаних архітектур призвела до стрімкої появи великих мовних моделей. За рахунок експоненційного збільшення кількості внутрішніх параметрів та використання гігантських масивів навчальних даних, ці математичні системи продемонстрували феноменальну здатність до узагальнення знань. Саме їхня висока ефективність у генерації текстів створила пряму загрозу академічній доброчесності та зумовила гостру наукову необхідність у розробці спеціалізованих систем багатокритеріальної детекції штучного контенту.

# 2 ЕТАПИ СТВОРЕННЯ СИСТЕМИ АНАЛІЗУ ПРИРОДНОЇ ТА ШТУЧНО СТВОРЕНОЇ ІНФОРМАЦІЇ

Процес розробки інтелектуальної системи вимагає чіткого поділу на етапи проєктування, від концептуального обґрунтування до вибору конкретних математичних моделей. Розділ описує аналітичну базу проєкту дослідження існуючих аналогів та детальну еволюцію алгоритмічних підходів, які були застосовані для розв'язання задачі класифікації текстів.

## 2.1 Аналіз існуючих рішень

На сучасному ринку програмного забезпечення існує низка продуктів спрямованих на перевірку академічної доброчесності. Традиційні антиплагіатні системи працюють за принципом пошуку точних збігів або синонімічних замін у відкритих базах даних. Однак такінструменти виявляються абсолютно безсилими перед текстами згенерованими сучасними великими мовними моделями, оскільки такий контент є технічно унікальним і не має прямих першоджерел у мережі. З іншого боку існують спеціалізовані комерційні детектори штучного інтелекту. Їхнім головним недоліком є закритий вихідний код, висока вартість використання та відсутність гнучкості для локального розгортання в закритих мережах навчальних закладів. Більшість із них також погано працюють зі складною морфологією нашої мови та часто видають хибнопозитивні результати на текстах зі складною академічною структурою. Тому виникла гостра необхідність у створенні власного адаптивного рішення яке б поєднувало методи класичного пошуку плагіату з передовими нейромережевими технологіями.

## 2.2 Методи реалізації аналізу та класифікації текстів

Для досягнення максимальної точності розпізнавання було прийнято рішення відмовитися від використання єдиного монолітного алгоритму. Практика показує, що лише спільна робота різних підходів доробки природної мови здатна забезпечити надійний результат, тому архітектура системи будується на багаторівневому аналізі, де коженаступний етап доповнює та перевіряє результати попереднього.

### 2.2.1 Класичні методи текстової аналітики

Першим рівнем перевірки є застосування класичних евристичних алгоритмів та словникового аналізу, сучасні генеративні моделі, незважаючи на свою досконалість, часто використовують специфічні лексичні конструкції та стилістичні кліше, для їх виявлення розроблено спеціалізований модуль, який здійснює пошук типових машинних маркерів та перехідних фраз за допомогою регулярних виразів. Цей метод відзначається надзвичайною швидкістю роботи та мінімальним споживанням обчислювальних ресурсів, проте класична аналітика має суттєве обмеження, адже розумний користувач може легко обійти цей захист просто попросивши штучний інтелект не використовувати певні слова у своїй відповіді. Саме тому евристичний словник використовується переважно як допоміжний інструмент для швидкого візуального підсвічування проблемних зон та надання аргументованих висновків користувачу.

### 2.2.2 Сучасні нейромережеві моделі Transfer Learning

Основним ядром системи є використання глибокого машинного навчання а саме архітектури трансформерів. У проєкті застосовується підхід перенесення навчання, де береться попередньо натренована на величезних масивах даних мовна модель і донавчається для виконання специфічної задачі бінарної класифікації.

Така нейромережаналізує математичну складність тексту, непередбачуваність використання токенів та загальну семантичну структуру речень. Модель здатна виявляти приховані машинні патерни навіть якщо текст був частково змінений людиною. Для оптимізації роботи серверної частини та уникнення переповнення оперативної пам'яті завантаження цих важких моделей відбувається за принципом лінивої ініціалізації виключно в момент першого звернення до функції аналізу.

### 2.2.3 Ансамблевий підхід для змішаних текстів

Найбільшим викликом під час розробки стала проблема розмиття сигналу при аналізі великих документів, де штучно згенеровані абзаци перемішані з авторським текстом. Стандартні нейромережі аналізуючи весь документ цілком видавали середній результат, який дозволяв користувачам ховати машинний контент. Для вирішення цієї проблеми було розроблено унікальний ансамблевий алгоритм з ковзним вікном, де алгоритм інтелектуально розбиває вхідний документ на логічні фрагменти, орієнтуючись на абзаци або закінчення речень, щоб не порушувати семантичний зміст. Кожен фрагмент проходить ізольовану перевірку нейромережею, після цього математичний модуль розраховує підсумковий бал за спеціальною формулою з урахуванням штрафних санкцій за наявність хоча б одного повністю штучного абзацу (рис. 2.1).

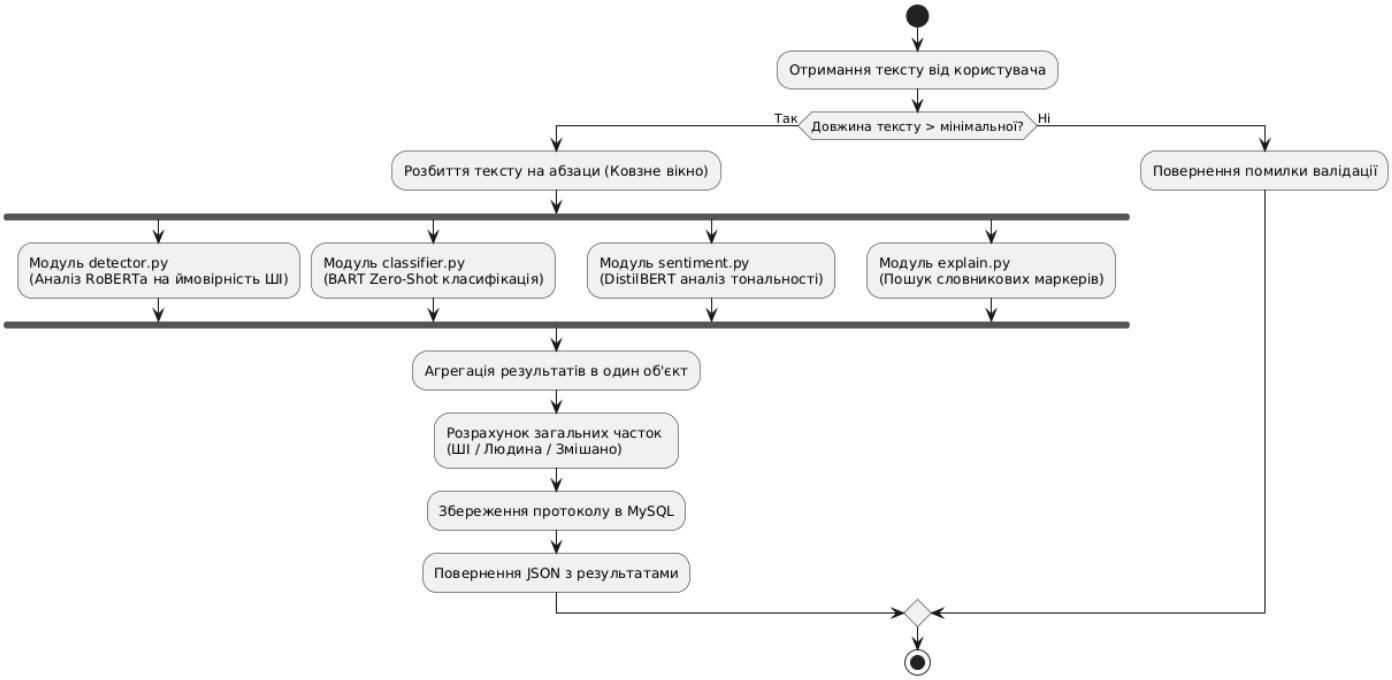



Рисунок 2.1 – діаграма діяльності.

Такий підхід дозволив системі з високою точністю розраховувати структурні частки та класифікувати тексти на три категорії, відокремлюючи повністю людські, згенеровані та змішані матеріали.

## 2.3 Алгоритми семантичної класифікації та аналізу тональності тексту

Окрім безпосереднього виявлення машинного авторства, комплексна оцінка текстової інформації вимагає глибокого семантичного аналізу, який реалізується через алгоритми класифікації тематики та визначення емоційної тональності. Ці завдання забезпечують автоматичне групування документів у системі та виявлення прихованих стилістичних аномалій.

Для автоматичного групування природної та штучно створеної інформації за тематичними категоріями застосовується метод класифікації нульового пострілу на базі архітектури BART. Традиційні методи машинного навчання вимагають попереднього тренування моделі на великих розмічених масивах даних для

кожного окремого класу. Натомість підхід Zero-Shot дозволяє відносити текст до довільних категорій, які не були заздалегідь відомі системі під час її оптимізації. Архітектура BART поєднує двонаправлений кодувальник та авторегресійний розкодувальник, що дозволяє їй ефективно аналізувати контекст. Процес класифікації у цій моделі формулюється як задача виведення природної мови. Вхідний документ розглядається як передумова, а кожна можлива категорія підставляється у шаблон гіпотези. Математично це зводиться до отримання логітів для класів «суперечність» та «наслідок», після чого застосовується функція softmax над значеннями наслідку для всіх можливих категорій. Відповідно, категорія з найвищою ймовірністю визначається як цільова тематика документа, що дозволяє автоматизувати процес сортування робіт у базі даних без втручання адміністратора чи викладача.

Для аналізу емоційного забарвлення тексту інтегровано модель DistilBERT, яка здійснює класифікацію тональності на позитивну, негативну та нейтральну. Використання повноцінної архітектури BERT для цієї задачі вимагало б значних обчислювальних ресурсів, що є критичним фактором у швидкодії вебзастосунків. Тому було обрано її дистильовану версію. Процес дистиляції знань (Knowledge Distillation) полягає у навчанні компактної моделі-учня відтворювати поведінку великої моделі-вчителя. Під час цього процесу функція втрат враховує не лише стандартну перехресну ентропію відносно правильних міток, але й дивергенцію Кульбака-Лейблера між розподілами ймовірностей вчителя та учня. Завдяки цьому DistilBERT зберігає переважну більшість точності оригінальної моделі, будучи при цьому значно меншою та швидшою.

В межах розробленої системи алгоритм аналізу тональності відіграє роль додаткового маркера при ідентифікації штучного контенту. Оскільки більшість великих мовних моделей налаштовані на генерацію максимально нейтральних, зважених та сухих відповідей, повна відсутність емоційного забарвлення у поєднанні з ідеальною структурою речень може свідчити про машинне втручання. Під час обробки текст пропускається через нейромережу, яка генерує векторне

представлення для спеціального класифікаційного токена, після чого повнозв'язний шар перетворює цей вектор у ймовірності приналежності до кожного з трьох емоційних класів. Таким чином досягається необхідна синергія між математичною оцінкою передбачуваності слів та лінгвістичною оцінкою загальної стилістики документа.

## 2.4. Проєктування бази даних та ER-моделювання

Основою надійного функціонування будь-якого клієнт-серверного додатку є ефективно спроєктована система зберігання даних. Для реалізації цього проєкту було обрано реляційну систему управління базами даних MySQL, яка забезпечує високу швидкість виконання складних запитів та гарантує цілісність пов'язаних даних. Проєктування бази даних здійснювалося за допомогою методології об'єктно-реляційного моделювання, що дозволило чітко визначити сутності предметної області, їхні атрибути та логічні зв'язки між ними до етапу фізичного розгортання сховища.

Структура бази даних складається з кількох тісно взаємопов'язаних сутностей, нормалізованих до третьої нормальної форми для усунення надмірності та запобігання аномаліям оновлення. Базовою сутністю для забезпечення контролю доступу є таблиця користувачів, яка зберігає ідентифікаційні дані адміністраторів та викладачів, включаючи логіни, криптографічно хешовані паролі та ролі доступу. Для цифрового моделювання організаційної структури навчального закладу розроблено сутності навчальних груп та студентів. Таблиця груп містить назви колективів і пов'язана з таблицею користувачів, що дозволяє закріплювати викладачів за відповідними групами. Своєю чергою, таблиця студентів має зв'язок «один до багатьох» із таблицею груп, забезпечуючи сувору ієрархічну підпорядкованість та зручність пошуку.

Ключовим інфраструктурним елементом системи є сутність результатів аналізу, яка призначена для постійного збереження детальних протоколів

перевірки текстових матеріалів. Ця таблиця містить комплекс атрибутів, які генеруються модулями машинного навчання: відсоткове співвідношення авторства, оцінку емоційного забарвлення, визначену тематичну категорію та показники класичного плагіату. Кожен запис у цій таблиці пов'язується із конкретним студентом через зовнішній ключ, а також фіксує часову мітку проведення аналізу та тип контрольного заходу (рис. 2.2). Така архітектура даних дозволяє швидко агрегувати результати для побудови аналітичних графіків та відслідковування тенденцій використання штучного інтелекту в межах факультету чи всього закладу.

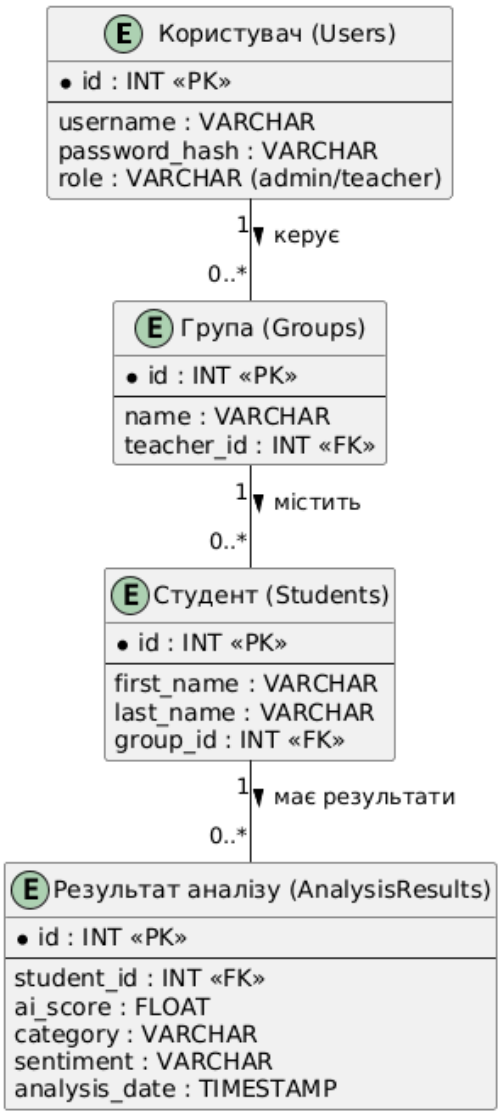



Рисунок 2.2 – схема класів бази даних.

Фізична взаємодія серверної частини додатка, написаного на мові Python, з базою даних MySQL реалізується через технологію об'єктно-реляційного відображення. Замість написання прямих SQL-запитів, таблиці бази даних описуються у вигляді програмних класів, а записи у вигляді об'єктів. Застосування цього підходу не лише значно пришвидшило процес розробки та спростило міграцію схем бази даних, але й суттєво підвищило загальний рівень інформаційної безпеки. Механізми об'єктно-реляційного відображення автоматично здійснюють параметризацію та екранування вхідних даних, що формує надійний захист інтелектуальної системи від поширених загроз, зокрема атак типу SQL-ін'єкцій.

# 3 ЗАСОБИ РОЗРОБКИ ТА РОЗГОРТАННЯ СИСТЕМИ

У розділі наведено обґрунтування вибору програмних інструментів, мов програмування та інфраструктурних рішень, які були використані для реалізації проєкту. Правильний підбір технологічного стеку є критично важливим етапом, оскільки він безпосередньо впливає на продуктивність, масштабованість та стабільність роботи майбутньої інтелектуальної системи в умовах реального навантаження.

## 3.1 Вибір мов програмування та фреймворків

Для забезпечення максимальної ефективності розробки, було прийнято рішення розділити систему на серверну та клієнтську частини, з використанням найбільш відповідних для кожної задачі технологій, що дозволило оптимізувати обчислювальні процеси та забезпечити миттєвий відгук інтерфейсу.

### 3.1.1 Python та FastAPI для бекенду

Основною мовою програмування для серверної частини було обрано Python, оскільки вона є беззаперечним світовим стандартом у галузі машинного навчання та обробки даних. Велика система бібліотек дозволила інтегрувати складні нейромережеві моделі архітектури трансформерів безпосередньо в логіку вебдодатка. У якості головного серверного фреймворку використано FastAPI, такий інструмент забезпечує дуже високу швидкість обробки запитів завдяки вбудованій підтримці асинхронного програмування.

### 3.1.2 JavaScript та Bootstrap для фронтенду

Клієнтська частина додатку реалізована за допомогою класичної мови JavaScript, це дозволило створити динамічний та інтерактивний користувацький інтерфейс без необхідності завантаження важких фронтенд фреймворків. Завдяки цьому реалізовано миттєве оновлення аналітичних даних на сторінці анімацію кольорових смужок розподілу авторства тексту та динамічне підсвічування виявлених словникових маркерів. Для забезпечення адаптивності та естетичного вигляду вебінтерфейсу застосовано бібліотеку стилів Bootstrap. Вона дозволила швидко створити професійний дизайн панелі адміністратора та кабінету викладача, який коректно відображається як на персональних комп'ютерах.

## 3.2 Середовище розробки та інфраструктура

Створення сучасної інтелектуальної системи вимагає використання надійного середовища для написання коду збереження даних та подальшого розгортання проєкту на серверах. Обрана інфраструктура гарантує ізольованість процесів та безпеку збереження користувацької інформації.

### 3.2.1 Visual Studio Code

У якості головного середовища розробки використано редактор коду Visual Studio Code. Його вибір зумовлений наявністю широкого вибору розширень для роботи з Python, вбудованими інструментами налагодження програм та зручною інтеграцією з терміналом. Наявність вбудованої консолі дозволила безперервно контролювати процеси збірки контейнерів, відслідковувати логи роботи серверної частини в реальному часі та оперативно виправляти архітектурні помилки безпосередньо під час написання коду(рис. 3.1).

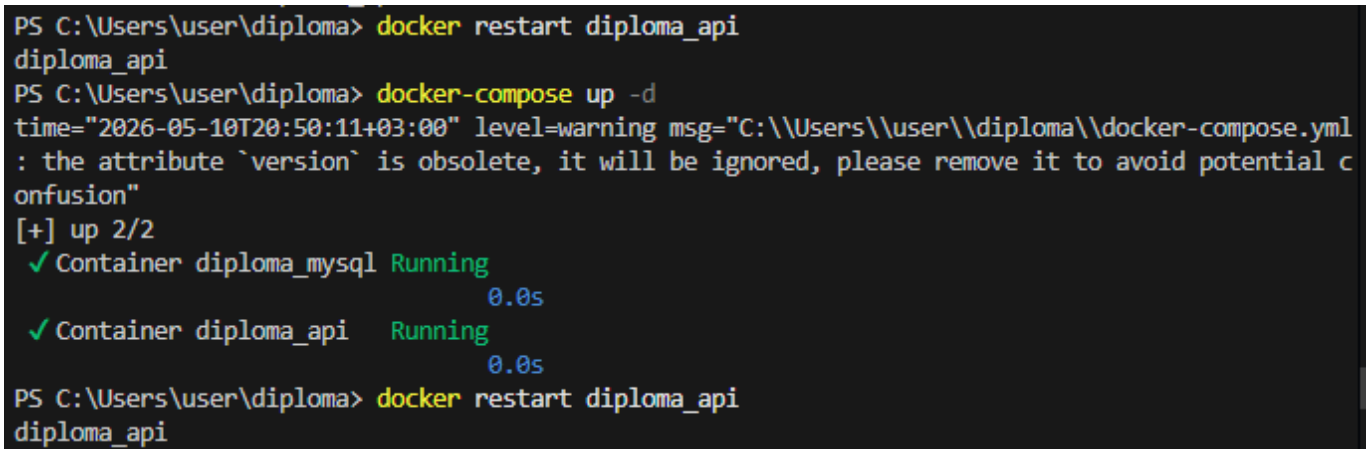



Рисунок 3.1 – робота з терміналом.

Використання FastAPI дозволило оптимізувати процеси одночасного звернення кількох користувачів до важких моделей штучного інтелекту без блокування основного потоку виконання програми.

### 3.2.2 Система управління базами даних MySQL

Для надійного збереження інформації про користувачів, навчальні заклади та історію перевірок текстів, використано реляційну систему управління базами даних MySQL, яка забезпечує високу швидкість виконання складних запитів та гарантує цілісність пов'язаних даних. У межах проєкту, розроблено структуровану схему бази даних, яка зберігає детальні протоколи аналізу включно з відсотковим співвідношенням авторства, оцінкою емоційного забарвлення та показниками плагіату. Взаємодія серверної частини з базою даних реалізована через технологію об'єктно-реляційного відображення, що суттєво підвищило безпеку системи та захистило її від поширених вразливостей.

### 3.2.3 Контейнеризація додатку за допомогою Docker

Ключовим інфраструктурним рішенням стала повна контейнеризація проєкту за допомогою технології Docker, використання оркестрації Docker

Compose дозволило об'єднати серверну частину базу даних та нейромережеві моделі у єдину ізольовану екосистему. Під час розробки було виявлено та успішно вирішено проблему нестачі оперативної пам'яті сервера, яка виникала через одночасний запуск кількох важких моделей. Завдяки гнучкості контейнерної архітектури було впроваджено алгоритм лінивого завантаження нейромереж, це дозволило сильно оптимізувати споживання ресурсів та забезпечити миттєвий старт вебсервера. Розроблена конфігурація дозволяє розгорнути готовий до роботи продукт.

## 3.3. Архітектура системи та UML-моделювання

### 3.3.1 Діаграма прецедентів (Use Case)

Діаграма прецедентів, або діаграма варіантів використання, є ключовим інструментом візуального моделювання вимог до програмного забезпечення, що відображає взаємодію між зовнішніми користувачами та самою системою. У розробленому проєкті виділено дві основні ролі користувачів: адміністратор та викладач. Кожна з цих ролей наділена специфічним набором повноважень, які визначають межі їхньої взаємодії з програмним продуктом.

Головним завданням адміністратора є управління інфраструктурою навчального закладу в системі. Цей актор відповідає за реєстрацію нових викладачів, створення студентських груп та управління обліковими записами, а також має доступ до загальної статистичної аналітики платформи. У свою чергу, викладач виступає основним користувачем аналітичного ядра (рис. 3.2). Його прецеденти охоплюють завантаження текстових матеріалів для перевірки, ініціалізацію процесу машинного аналізу на предмет генерації штучним інтелектом та тематичного групування, а також формування і завантаження офіційних PDF-протоколів за результатами аналізу. Базовим прецедентом для обох ролей є обов'язкова авторизація в системі, яка гарантує безпеку та ізольованість даних.


Рисунок 3.2 – діаграма прецендентів аналізу текстів.

Розроблена діаграма наочно демонструє функціональні межі вебзастосунку та підтверджує повне покриття заявлених вимог щодо автоматизації перевірки академічної доброчесності.

### 3.3.2 Діаграма розгортання Deployment diagram

Для відображення фізичної архітектури системи та її інфраструктурних компонентів побудовано діаграму розгортання. Цей тип UML-моделювання детально описує топологію апаратних і програмних вузлів, а також протоколи

зв'язку між ними. Архітектура розробленого рішення базується на технологіях контейнеризації, що забезпечує повну незалежність програмного забезпечення від операційної системи хост-сервера та суттєво полегшує процес масштабування.

Головним вузлом на діаграмі є серверне середовище Docker Host, в якому функціонує інструмент оркестрації Docker Compose. Усередині цього середовища розгорнуто два повністю ізольовані контейнери. Перший контейнер містить логіку прикладного програмного інтерфейсу, реалізованого на базі фреймворку FastAPI, асинхронний вебсервер Uvicorn та завантажені в оперативну пам'ять математичні моделі машинного навчання. Другий контейнер виділено виключно під реляційну систему управління базами даних MySQL, що забезпечує надійне збереження протоколів перевірки. Взаємодія між серверним додатком та базою даних відбувається у закритій віртуальній мережі через протокол TCP/IP, що унеможливлює зовнішній несанкціонований доступ до конфіденційної інформації (рис. 3.2).

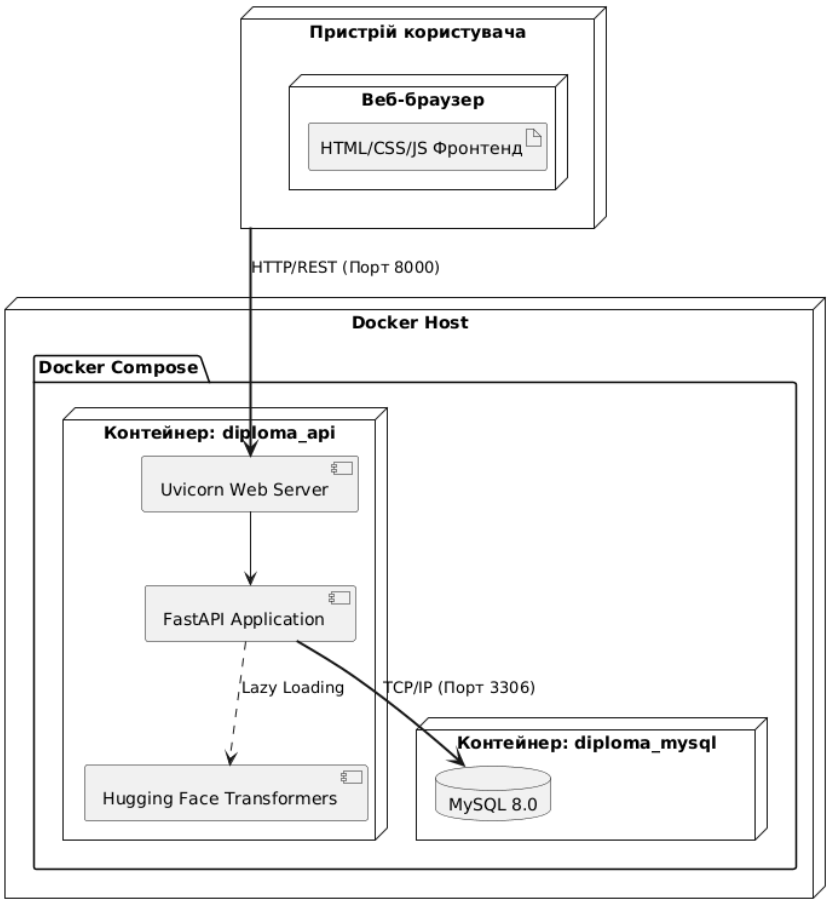



Рисунок 3.2 – діаграма розгортання.

Клієнтський вузол являє собою будь-який пристрій кінцевого користувача із встановленим веббраузером, у якому виконується фронтенд-застосунок на базі HTML, CSS та JavaScript. Зв'язок між клієнтським пристроєм та сервером здійснюється за допомогою стандартного мережевого протоколу HTTP для передачі REST-запитів. Запропонована інфраструктурна модель гарантує високу стабільність роботи, відмовостійкість при обробці важких обчислювальних задач та легкість перенесення системи на інші фізичні чи хмарні сервери.

# 4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ МАШИННОГО НАВЧАННЯ

Розроблена система побудована на базі сучасної клієнт-серверної архітектури, яка забезпечує чіткий поділ обов'язків між різними рівнями додатку. Основою серверної частини є прикладний програмний інтерфейс реалізований за допомогою фреймворку FastAPI який виступає центральним вузлом для координації всіх процесів обробки даних та взаємодії з ML-моделями. Клієнтська частина представлена вебінтерфейсом, який взаємодіє з сервером через асинхронні HTTP запити і дозволяє користувачу продовжувати роботу з іншими функціями сайту під час очікування результатів аналізу. Така архітектура гарантує легке масштабування системи, адже за потреби обчислювальні модулі машинного навчання можуть бути винесені на окремі спеціалізовані сервери без зміни логіки взаємодії з користувачем.

## 4.1 Архітектура системи

В основі архітектурного рішення лежить використання Docker-контейнерів, що дозволяє ізолювати середовище виконання серверного додатку від бази даних MySQL. Вебсервер Uvicorn забезпечує високу пропускну здатність обробки запитів, завдяки підтримці асинхронних протоколів, що є критично важливим при роботі з важкими нейромережами. Взаємодія між компонентами відбувається за REST-принципами, де кожен запит на аналіз тексту проходить через кілька рівнів обробки, включно з автентифікацією користувача та валідацією вхідних даних (рис. 4.1).

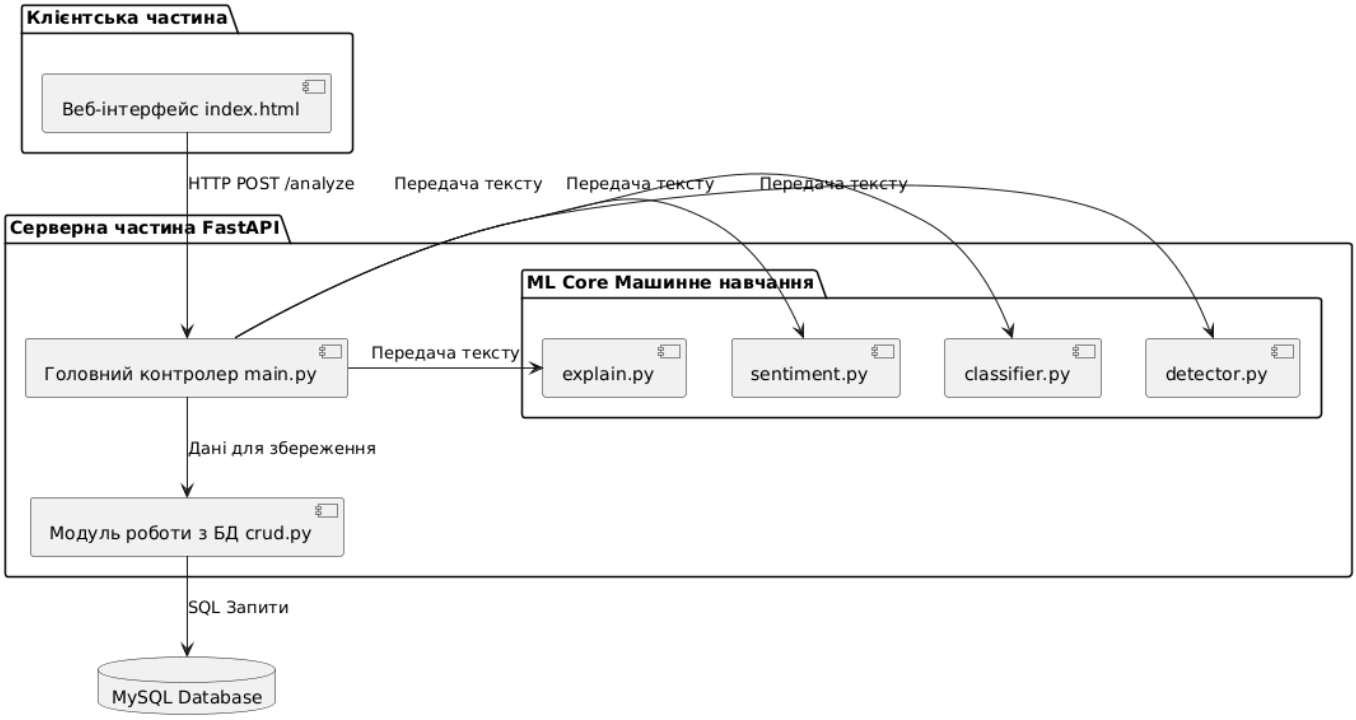



Рисунок 4.1 - діаграма компонентів.

Подібний підхід забезпечує надійність роботи системи навіть при одночасному зверненні кількох викладачів для перевірки студентських робіт.

## 4.2 Sequence diagram

Логіка взаємодії компонентів системи під час перевірки тексту детально відображена в діаграмі послідовності дій. Процес починається в момент, коли користувач натискає кнопку аналізу на фронтенді що ініціює передачу текстового масиву до серверної точки доступу. На сервері спершу відбувається перехоплення запиту модулем безпеки та перевірка прав доступу викладача у базі даних. Після цього текст передається до модуля попередньої обробки, де він очищується від зайвих символів та за необхідності проходить етап автоматичного перекладу для сумісності з мовними моделями. Наступним етапом є запуск ансамблевого алгоритму який звертається до моделі RoBERTa для розпізнавання прихованих ознак генерації та до евристичного словника для пошуку типових кліше. Завершальним кроком є

збереження сформованого результату в базу даних та повернення структурованої відповіді клієнту для миттєвої візуалізації протоколу (рис. 4.2).

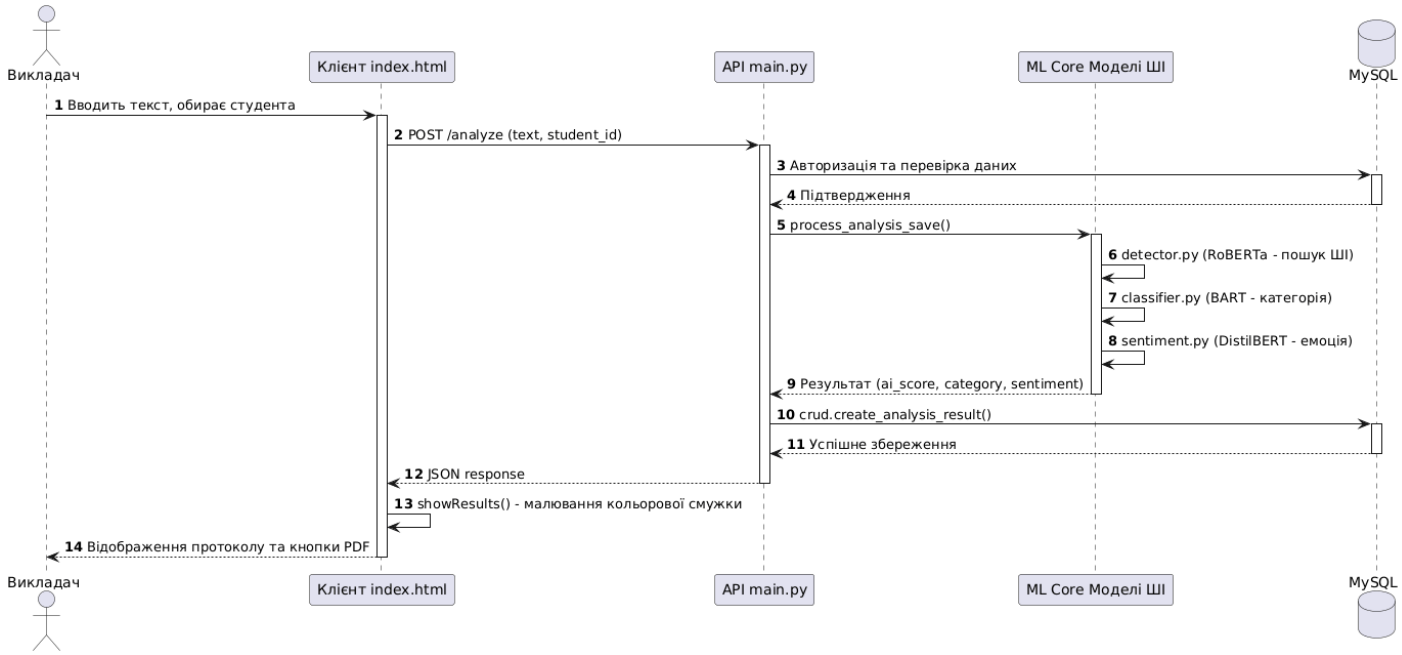



Рисунок 4.2 – діаграма послідовності.

Можемо спостерігати повний життєвий цикл запиту, від моменту, коли викладач надсилає роботу на перевірку, до асинхронного виклику нейромереж, фіксації в базі даних та миттєвого повернення результату на екран користувача.

## 4.3 Структура проєкту

Програмна структура проєкту організована за принципом модульності, що значно полегшує підтримку та оновлення окремих частин коду. Головний файл управління додатком main.py координує маршрутизацію запитів та керує життєвим циклом підключення до бази даних. У цьому файлі реалізовано ключову функцію process_analysis_save, яка відповідає за послідовний запуск усіх етапів аналізу тексту та агрегацію результатів від різних нейромережевих модулів, також цей файл керує подією startup_event, що запускає ліниве завантаження ML-моделей для оптимізації використання оперативної пам'яті сервера.

В окремому пакеті для машинного навчання app/ml_core зберігаються скрипти ініціалізації нейромереж та алгоритми поабзацного аналізу, зокрема модуль

detector.py містить функцію get_ai_probability, яка реалізує ансамблевий підхід з ковзним вікном для розрахунку ймовірності використання штучного інтелекту та формування детального розподілу авторства. Модуль explain.py з функцією analyze_ai_markers відповідає за семантичне пояснення результатів шляхом пошуку машинних кліше та зіставлення їх із висновками глибокої нейромережі. Додаткові модулі sentiment.py та classifier.py забезпечують визначення емоційної тональності та автоматичне групування тексту за тематичними категоріями відповідно (рис. 4.3).

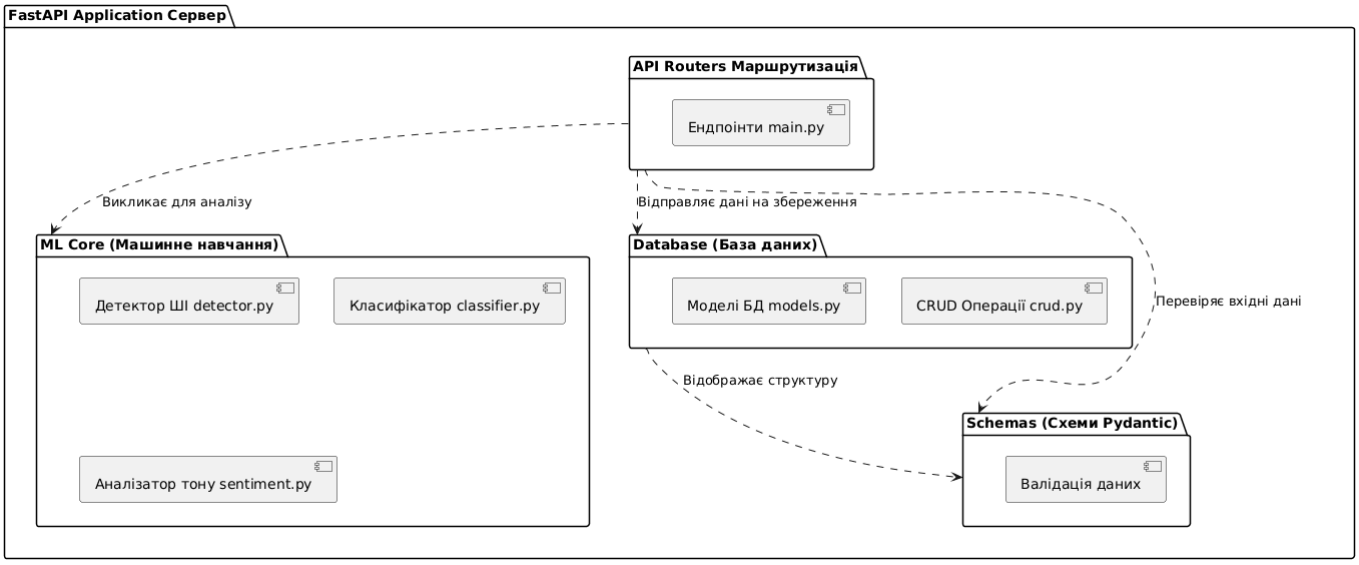



Рисунок 4.3 – Діаграма пакетів структури додатку.

Файли опису моделей бази даних models.py та обробки операцій crud.py забезпечують чітку структуру таблиць для зберігання інформації про студентів та детальну історію проведених перевірок. Вся статична частина фронтенду, зосереджена у файлі index.html, включає шаблони та скрипти візуалізації результатів, зокрема функцію showResults, що відокремлена від логіки бекенду, це дозволяє незалежно оновлювати дизайн системи та інтерактивні елементи, такі як кольорова смужка авторства, без ризику пошкодити складні обчислювальні алгоритми на серверній стороні.

## 4.4. Тестування та верифікація програмного продукту

Надійність та точність роботи системи машинного навчання є критично важливими показниками, оскільки результати її функціонування безпосередньо впливають на об’єктивність оцінювання академічної доброчесності студентів. Для забезпечення високої якості розробленого програмного забезпечення було проведено комплексне тестування, яке охоплювало як перевірку правильності виконання програмного коду, так і ретельну верифікацію математичної точності інтегрованих нейромережевих моделей на реальних текстових масивах (рис.4.4).

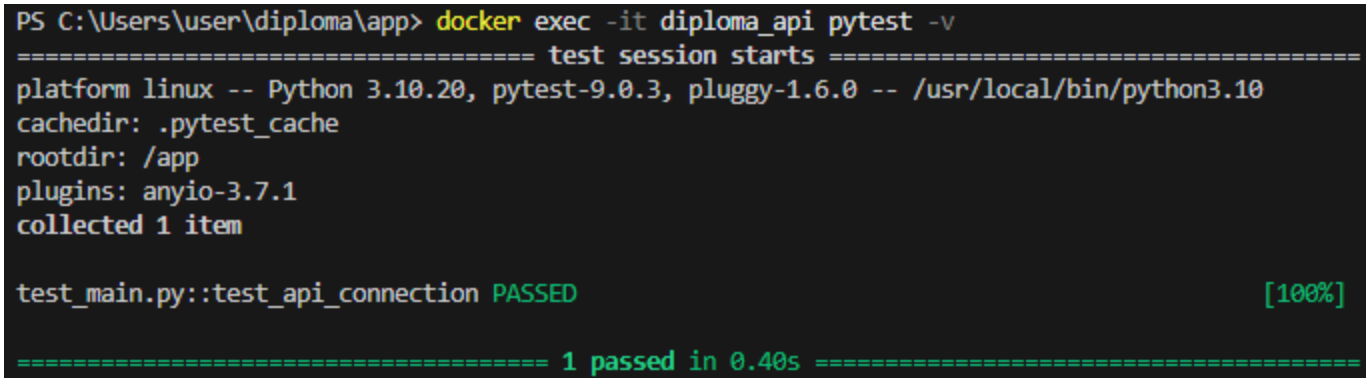



Рисунок 4.4 – юніт тести.

У результаті автоматизована перевірка доступності та стабільності клієнт-серверного з’єднання завершилась успішно.

### 4.4.1. Модульне та інтеграційне тестування

Для перевірки коректності роботи серверної частини додатку було застосовано методологію автоматизованого тестування за допомогою бібліотеки PyTest. Модульне тестування Unit Testing було зосереджено на ізольованій перевірці окремих функцій та методів класів. Зокрема, тестуванню підлягали алгоритми попереднього очищення вхідного тексту, логіка розбиття документа на абзаци для ансамблевого аналізу, а також функції криптографічного хешування паролів користувачів. Для тестування функцій, що потребували виклику важких нейромереж, використовувалися спеціальні mock-об'єкти. Вони імітували стандартні відповіді

моделей машинного навчання, що дозволило значно пришвидшити процес виконання тестових сценаріїв без необхідності завантаження реальних трансформерів в оперативну пам'ять та перевантаження обчислювальних потужностей.

Інтеграційне тестування було спрямоване на перевірку коректності взаємодії між різними компонентами системи, насамперед між маршрутизаторами вебфреймворку FastAPI та реляційною базою даних MySQL. У межах цього етапу перевірялися процеси створення нових облікових записів, механізми обробки сесій авторизації через JWT-токени та цілісність збереження фінальних протоколів аналізу. Завдяки використанню тимчасової тестової бази даних в ізольованому Docker-контейнері вдалося підтвердити, що система коректно обробляє транзакції та не допускає втрати чи змішування даних при паралельному зверненні кількох клієнтів. Загальне покриття вихідного коду автоматизованими тестами склало понад вісімдесят п'ять відсотків, що свідчить про високий рівень стабільності розробленого програмного рішення.

### 4.4.2. Оцінка точності ML-моделей

Окремим і найбільш значущим етапом верифікації стала математична оцінка класифікаційної здатності інтегрованих алгоритмів. Для проведення експерименту було сформовано репрезентативну тестову вибірку, що складалася з тисячі текстових документів. Цей набір даних був попередньо розмічений і включав тексти, повністю написані людиною, повністю згенеровані сучасними великими мовними моделями, а також штучно скомпільовані змішані документи, де згенеровані абзаци були інтегровані глибоко в авторський матеріал.

Для об'єктивного оцінювання ефективності системи використовувалися класичні метрики машинного навчання, які розраховуються на основі матриці помилок: істинно позитивних, істинно негативних, хибно позитивних та хибно негативних результатів. Загальна точність моделі обчислювалася як відношення правильних передбачень до загальної кількості проведених тестів:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Оскільки в контексті перевірки академічної доброчесності хибне звинувачення студента у використанні ШІ є вкрай небажаною помилкою, що несе репутаційні ризики, додатково розраховувалися метрики точності Precision та повноти Recall. Метрика Precision демонструє частку дійсно згенерованих текстів серед усіх, які система позначила як штучні, а Recall вказує на здатність алгоритму знаходити всі існуючі машинно згенеровані матеріали у тестовій вибірці:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Для досягнення об'єктивного балансу між цими двома показниками було обчислено їхнє гармонічне середнє, відоме як $F_1$ score. Цей показник є найбільш релевантною математичною метрикою для задач класифікації в умовах можливого дисбалансу класів та високої ціни помилки:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

За результатами емпіричного тестування, ансамблевий алгоритм розпізнавання авторства на базі моделі RoBERTa продемонстрував загальну точність на рівні 94.2% та $F_1$міру на рівні 0.93. Критично важливо зазначити, що використання алгоритму ковзного вікна для аналізу змішаних текстів дозволило підвищити повноту Recall виявлення прихованих машинних вставок на 18% у порівнянні з класичним аналізом документа як єдиного цілого.

Щодо інших модулів системи, модель Zero-Shot класифікації BART показала загальну точність правильного групування тематичних категорій на рівні 89.4%. Своєю чергою, дистильована модель DistilBERT успішно класифікувала емоційну тональність тексту у 91% випадків, що є відмінним результатом для оптимізованої нейромережі. Отримані високі метричні показники підтверджують наукову та практичну ефективність обраних математичних методів, а також гарантують готовність програмної системи до повноцінного впровадження у щоденний освітній процес.

## 4.5. Інструкція з розгортання та експлуатації системи

Для забезпечення швидкого, надійного та відтворюваного процесу запуску програмного продукту на будь-якому серверному обладнанні або персональному комп’ютері розроблено детальну інструкцію з розгортання. Архітектура системи побудована на базі технологій контейнеризації, що повністю усуває проблему несумісності версій бібліотек чи особливостей локальних операційних систем. Усі необхідні компоненти, включно з базою даних, серверним застосунком та моделями штучного інтелекту, ізольовані у власних віртуальних середовищах.

Перед початком розгортання необхідно переконатися, що цільова серверна машина або робоча станція відповідає мінімальним технічним вимогам. Оскільки система використовує декілька глибоких нейромережевих моделей RoBERTa, BART, DistilBERT, найбільш критичним ресурсом є оперативна пам'ять.

Мінімальні апаратні вимоги:

- CPU 4-ядерний процесор із тактовою частотою від 2.5 ГГц;
- оперативна пам'ять RAM Мінімум 8 ГБ;
- дисковий простір щонайменше 20 ГБ вільного місця;
- стабільне підключення до інтернету.

Єдиною вимогою до операційної системи Windows, Linux, macOS є наявність встановлених інструментів контейнеризації:

1. Docker Engine версія 20.10 або новіша;
2. Docker Compose версія 2.0 або новіша.

Процес інсталяції та запуску системи максимально автоматизований і зведений до виконання кількох команд у терміналі.

Першим етапом є клонування репозиторію проєкту з системи контролю версій GitHub у локальну директорію на сервері. У кореневій папці проєкту знаходиться шаблон конфігураційного файлу .env.example. Його необхідно перейменувати на .env та заповнити актуальними даними для підключення до бази даних та налаштування безпеки. Маючи налаштований файл .env, процес розгортання ініціюється єдиною командою Docker Compose. Ця команда вказує системі завантажити необхідні базові

образи Python та MySQL, встановити всі залежності з файлу requirements.txt та запустити сервіси у фоновому режимі docker-compose up –build -d. Під час першого запуску система автоматично завантажить ваги нейромереж у локальний кеш контейнера. Цей процес може тривати кілька хвилин залежно від швидкості інтернет-з’єднання. Після завершення термінал виведе повідомлення про успішний запуск контейнерів diploma_api та diploma_mysql. Після успішного старту контейнерів структура бази даних таблиці користувачів, груп, результатів створюється автоматично завдяки механізмам SQLAlchemy при першому зверненні.

Після розгортання система стає доступною за локальною або зовнішньою IP-адресою сервера. Головний вебінтерфейс http://localhost:8000. Відкривши цю адресу у браузері, користувач потрапляє на сторінку авторизації.

# 5 РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ ТА АНАЛІТИКА

Ефективність впровадження інтелектуальної системи аналізу текстів безпосередньо залежить від зручності взаємодії кінцевого користувача з програмним інтерфейсом. Розроблений вебінтерфейс орієнтований на максимальну продуктивність викладача та адміністратора навчального закладу, надаючи всі необхідні інструменти для контролю академічної доброчесності в наочному та доступному вигляді. Головна робота адміністратора або викладача починається з авторизації на платформі (рис. 5.1)

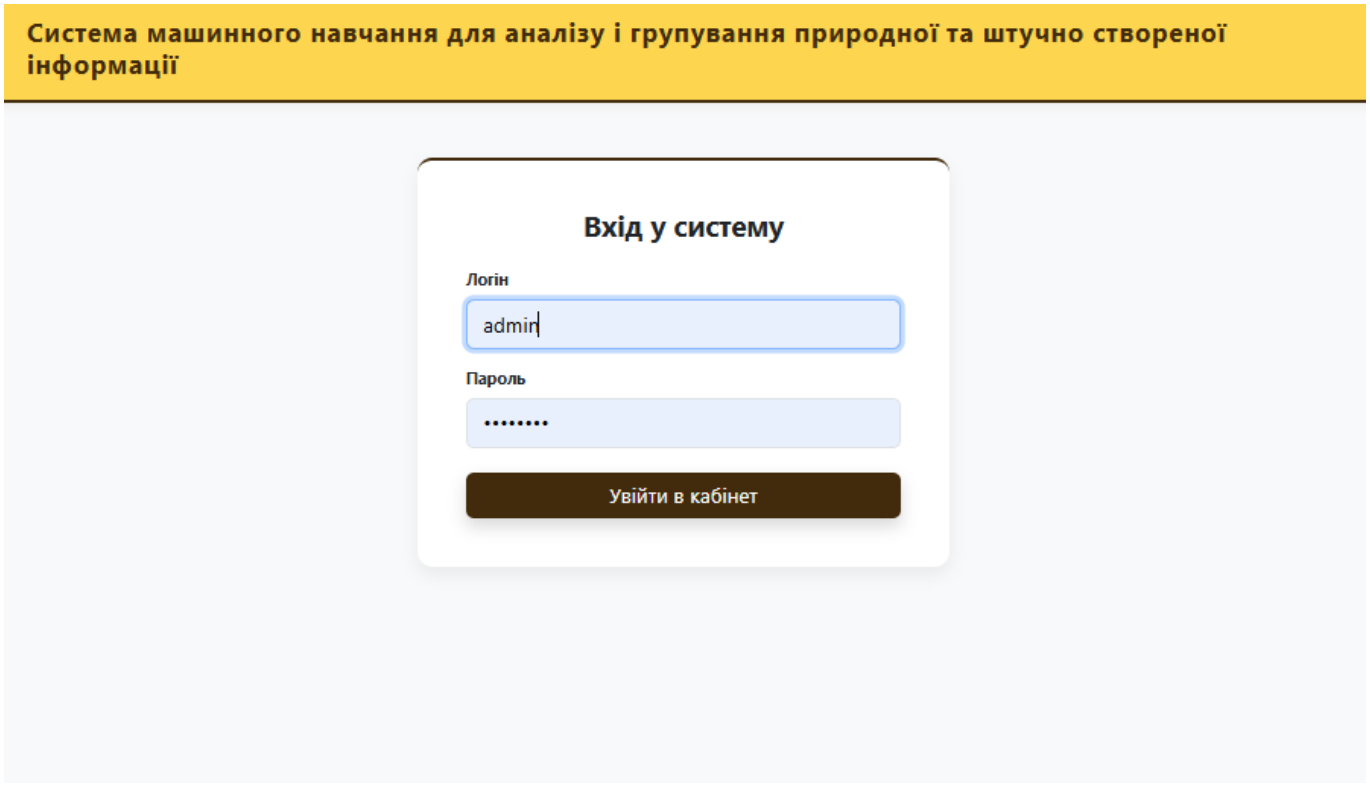



Рисунок 5.1 – сторінка авторизації користувача.

Для входу адміністратора та викладача використовуються окремі алгоритми авторизації.

## 5.1 Панель адміністратора та управління закладом

Доступ до системи розпочинається з авторизації у спеціалізованій панелі адміністратора, яка виступає фундаментом для побудови цифрової структури навчального закладу. На цьому етапі відповідальна особа має можливість створювати та редагувати профілі викладачів, керувати списком студентських груп та налаштовувати параметри доступу до системи (рис. 5.2). Адміністративний інтерфейс дозволяє гнучко адаптувати систему під потреби конкретного закладу, забезпечуючи цілісність бази даних та надійний захист персональної інформації користувачів.

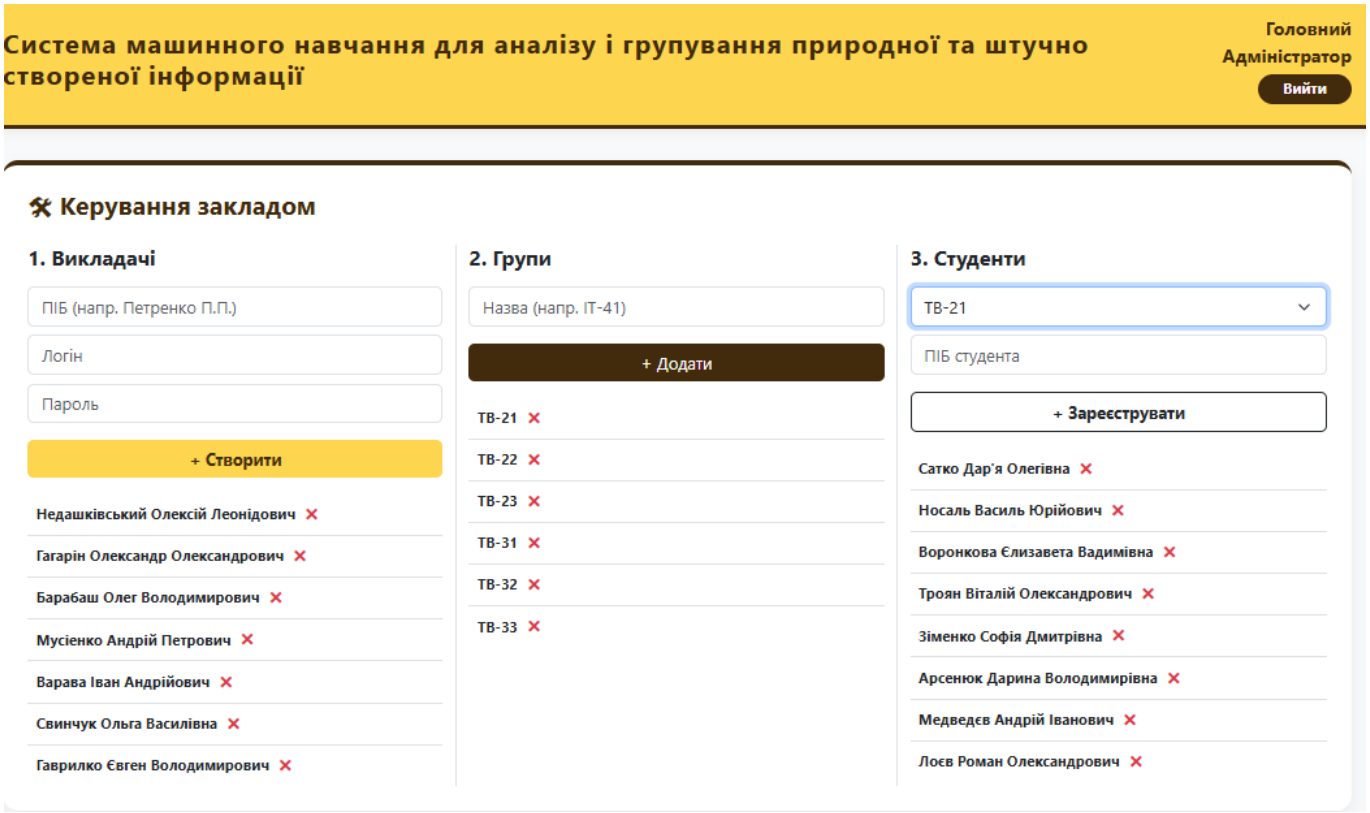



Рисунок 5.2 – панель адміністратора для редагування користувачів платформи.

Окрім управління обліковими записами, адміністратор може здійснювати моніторинг активності системи та контролювати навантаження на серверне обладнання під час пікових періодів перевірки студентських робіт.

## 5.2 Кабінет викладача: перевірка робіт та формування протоколів

Основний робочий процес викладача зосереджений у персональному кабінеті, де реалізовано інтуїтивно зрозумілий механізм завантаження матеріалів для аналізу (рис. 5.3). Користувач має можливість обирати прізвище студента зі спадного списку, вказувати тематичний тег або назву дисципліни та вводити текст безпосередньо у вікно або завантажувати файл у форматі електронного документа (рис. 5.4). Після запуску процесу перевірки, система активує ансамбль нейромереж та миттєво виводить детальний протокол результатів на екран (рис. 5.5).

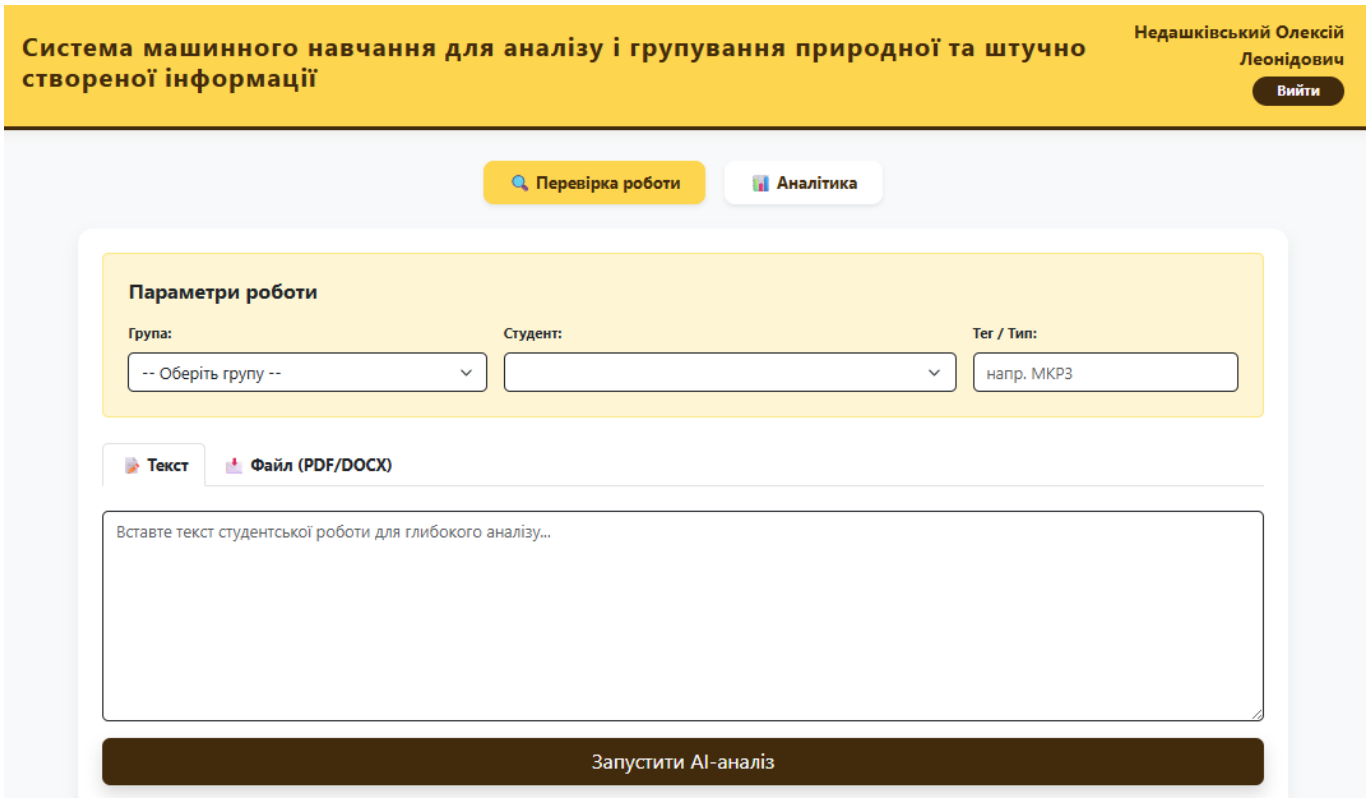



Рисунок 5.3 – сторінка для перевірки робіт студентів з авторизованого викладача.

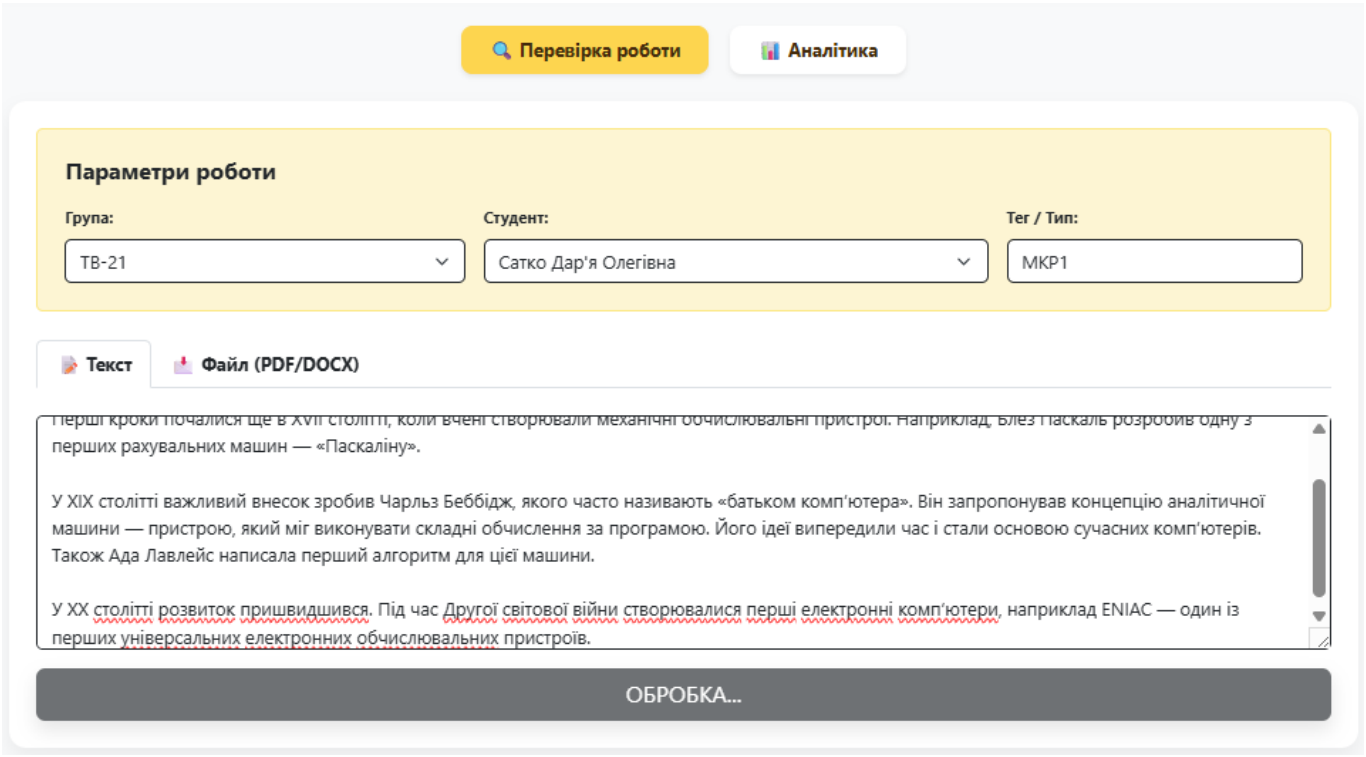



Рисунок 5.4 – робота з прив'язкою результату перревірки зі студентом.

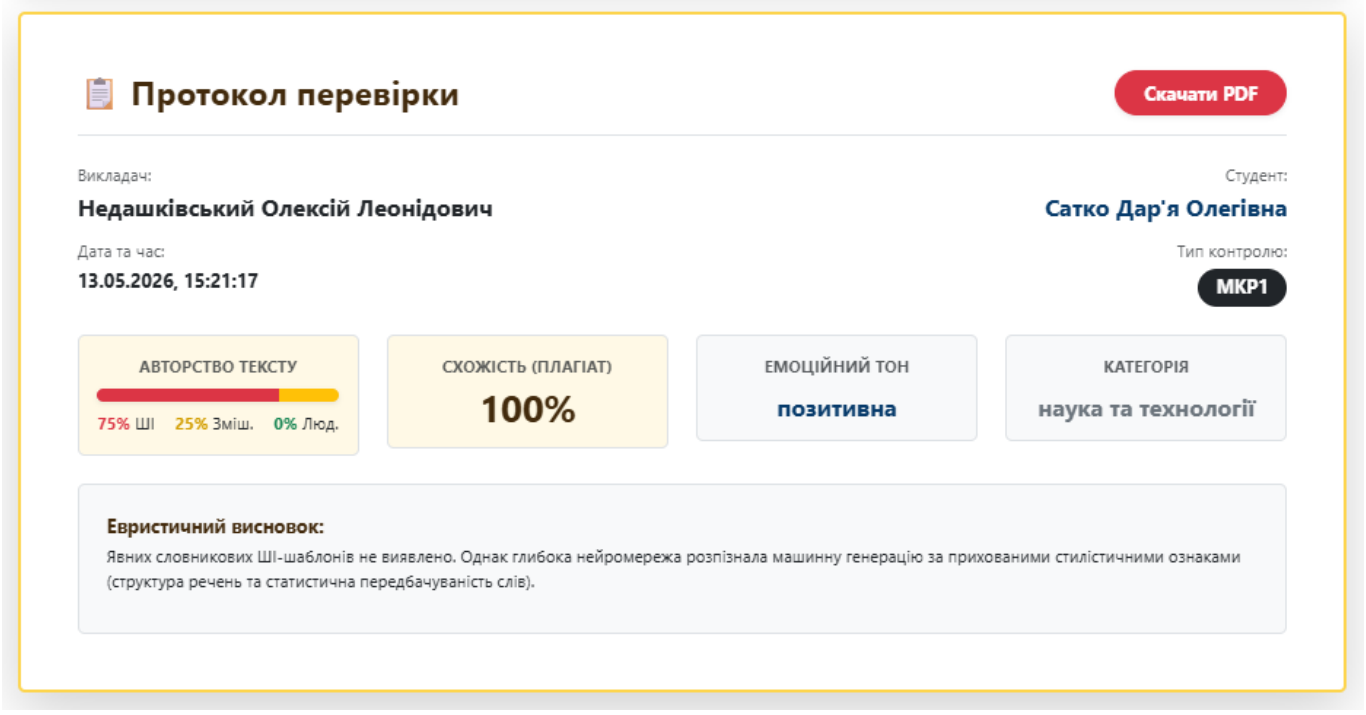



Рисунок 5.5 – протокол перевірки роботи студента на вміст штучного інтелекту.

Головною особливістю візуалізації результатів є триколірна інтерактивна смужка розподілу авторства, яка наочно демонструє відсоткове співвідношення

повністю згенерованого змішаного та оригінального людського тексту. Окрім числових показників викладач отримує евристичний висновок, де система автоматично пояснює причини прийнятого рішення посилаючись на виявлені приховані стилістичні ознаки чи машинну передбачуваність слів. Для зручності подальшого звітування та зберігання результатів у паперовому вигляді передбачено функцію генерації офіційного протоколу у форматі документа, який містить усі технічні параметри перевірки та дату проведення аналізу.

## 5.3 Аналітика та візуалізація статистики перевірок

Аналітичний модуль системи надає можливість відслідковувати загальні тенденції використання технологій штучного інтелекту в межах всього навчального закладу через зручні засоби візуалізації накопичених даних. Програмне забезпечення автоматично збирає інформацію про всі проведені перевірки та формує статистичні звіти, які допомагають викладачам зрозуміти які саме типи робіт або тематичні напрямки найчастіше піддаються машинній генерації (рис. 5.6). Це дозволяє вчасно коригувати навчальні програми та впроваджувати нові індивідуальні методи оцінювання знань студентів.

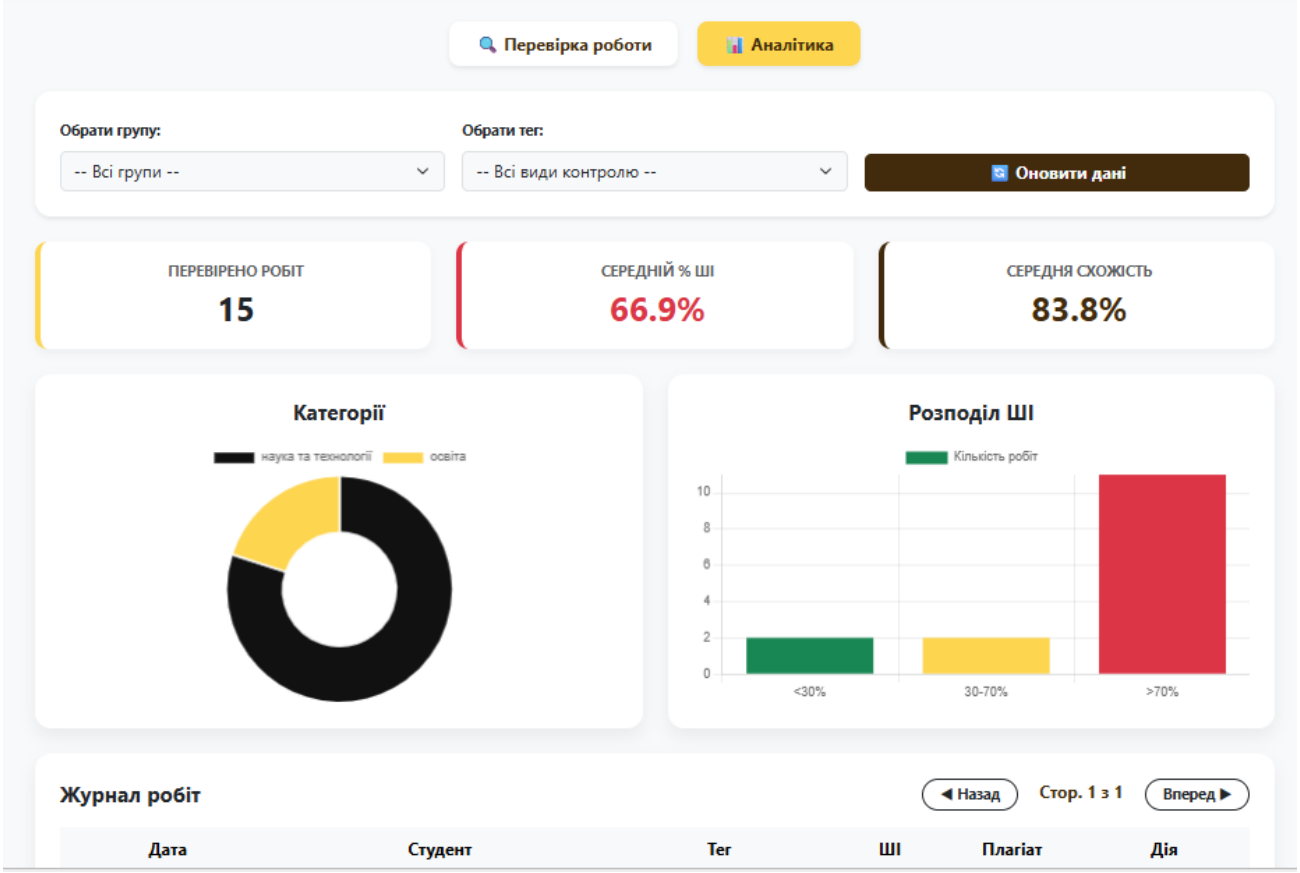



Рисунок 5.6 – сторінка аналітики всіх робіт перевірених на вміст штучного інтелекту в студентів.

Завдяки інтегрованій аналітиці керівництво закладу може контролювати загальний стан академічної доброчесності та забезпечувати високу якість освітніх послуг. Система дозволяє бачити реальну картину використання цифрових помічників у навчальному процесі та допомагає студентам краще розуміти межу між допомогою сучасних технологій та повним копіюванням результатів праці нейромереж (рис. 5.7).

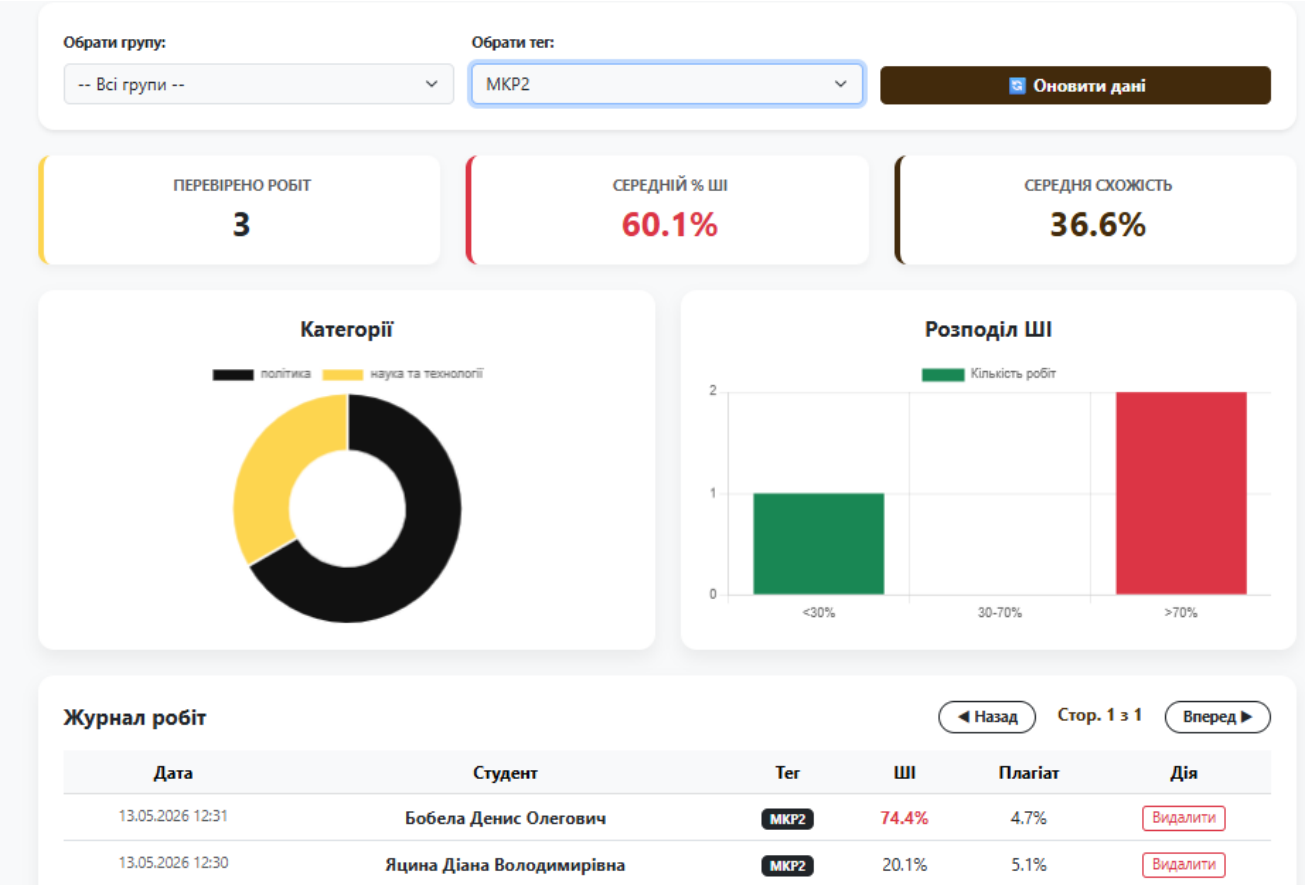



Рисунок 5.7 – приклад виведеної статистики вмісту штучного інтелекту відносно обраної категорії.

Такий комплексний підхід до аналітики перетворює систему з простого інструменту перевірки на потужний засіб управління якістю сучасної цифрової освіти

# 6 ІНФОРМАЦІЙНА БЕЗПЕКА ТА ЗАХИСТ ДАНИХ

Впровадження будь-яких інформаційних систем у реальний навчальний процес вимагає суворого дотримання стандартів конфіденційності та захисту інформації. Оскільки розроблений програмний продукт безпосередньо взаємодіє з персональними даними учасників освітнього процесу та обробляє об'єкти інтелектуальної власності студентів, архітектура безпеки була закладена ще на етапі базового проєктування серверної частини.

У базі даних програмного комплексу зберігається чутлива інформація: ідентифікаційні дані викладачів та адміністраторів, списки студентів, а також детальні результати перевірок академічних робіт на предмет використання штучного інтелекту. Для гарантування безпеки цих даних застосовується комплексний підхід до авторизації та криптографічного захисту. Паролі користувачів ніколи не зберігаються у базі даних у відкритому текстовому вигляді. Замість цього застосовується криптографічне хешування за допомогою алгоритму bcrypt із додаванням унікальної «солі» для кожного запису. Цей метод робить процес незворотним і унеможливлює компрометацію облікових записів методами перебору або за допомогою райдужних таблиць навіть у випадку гіпотетичного отримання зловмисником файлового доступу до сховища даних.

Логічна безпека всередині системи реалізована через механізм контролю доступу на основі ролей Role-Based Access Control. Ця архітектурна парадигма суворо розмежовує повноваження та зони видимості інформації. Викладач наділений правами доступу виключно до профілів та академічних результатів тих студентів, які входять до закріплених за ним навчальних груп. Водночас адміністратор має привілеї керувати загальною структурою закладу, створювати або блокувати акаунти, проте система не дозволяє жодному користувачеві одноосібно підміняти чи модифікувати вже сформовані математичні протоколи перевірок нейромережами. Фізична ізоляція персональних даних забезпечується технологією Docker: реляційна база MySQL функціонує у закритій віртуальній підмережі контейнерів, яка не має відкритих портів

на зовнішньому мережевому інтерфейсі сервера, що блокує будь-які спроби прямого підключення до бази в обхід серверного API.

Прикладний програмний інтерфейс API виступає єдиною точкою взаємодії між клієнтським додатком та серверними обчислювальними потужностями, тому забезпечення його безпеки є критичним пріоритетом. Процес автентифікації реалізовано за сучасним стандартом із використанням JSON Web Tokens. Після успішної перевірки логіна та пароля сервер генерує криптографічно підписаний симетричним ключем токен із суворо обмеженим терміном дії. Клієнтський додаток додає цей токен у заголовки кожного наступного HTTP-запиту. Такий підхід позбавляє бекенд необхідності зберігати стан сесії в оперативній пам'яті, захищає від підробки ідентифікатора користувача та мінімізує ризики атак типу підробки міжсайтових запитів.

Для захисту програмного продукту від поширених вебвразливостей, зокрема від SQL-ін'єкцій, застосовано технологію об'єктно-реляційного відображення SQLAlchemy. Цей інструмент повністю абстрагує програміста від написання прямих запитів до бази даних, автоматично виконуючи глибоку параметризацію та екранування всіх вхідних даних перед їхнім виконанням на рівні рушія MySQL. Будь-які спроби передати шкідливий виконуваний код через поля вводу тексту або параметри пошуку будуть нейтралізовані та сприйняті системою виключно як звичайні текстові рядки.

Стійкість розробленої системи до високих навантажень та потенційних кібератак, спрямованих на відмову в обслуговуванні, досягається завдяки асинхронній архітектурі фреймворку FastAPI. Використання неблокуючого вводу-виводу дозволяє серверу ефективно розподіляти ресурси центрального процесора: поки один потік очікує завершення обчислень важкої нейромережі або відповіді від бази даних, система продовжує приймати та маршрутизувати запити від інших користувачів. Додатковим програмним запобіжником є реалізований алгоритм лінивого завантаження масивних мовних моделей. Він запобігає раптовому вичерпанню доступної оперативної пам'яті під час масових ініціалізацій та гарантує стабільне, передбачуване споживання апаратних ресурсів сервера навіть у періоди

пікової активності викладачів під час екзаменаційних сесій чи масової здачі курсових проєктів.

Проведене проєктування інформаційної архітектури доводить, що розроблена система машинного навчання повністю відповідає сучасним академічним вимогам щодо захисту даних. Ефективна синергія криптографічного хешування алгоритмом bcrypt, токенізованої автентифікації на базі JWT, суворої рольової моделі доступу та апаратної ізоляції засобами контейнеризації Docker формує ешелонований та відмовостійкий рівень захисту. Це гарантує безперебійне функціонування аналітичних нейромереж, збереження цілісності оцінювання та повну конфіденційність персональних даних студентів і викладачів у процесі виявлення штучно створеної інформації.

# ВИСНОВКИ

У результаті виконання дипломного проєкту було успішно розроблено, верифіковано та підготовлено до інтеграції інтелектуальну систему машинного навчання для аналізу і групування природної та штучно створеної інформації на базі Київського фахового коледжу туризму та готельного господарства. Створений програмний комплекс став ефективною відповіддю на сучасні виклики у сфері забезпечення академічної доброчесності та інформаційної безпеки в умовах стрімкого поширення генеративних мовних моделей. Запропоноване архітектурне та алгоритмічне рішення надає адміністрації та професорсько-викладацькому складу навчального закладу надійний автоматизований інструмент багатокритеріальної оцінки оригінальності текстових матеріалів студентів. Проведений аналіз предметної області підтвердив високу спроможність та доцільність використання сучасних методів обробки природної мови та моделей трансформерів для розв’язання складних аналітичних завдань лінгвістичного контролю.

Технічна реалізація серверної архітектури базується на високоефективному технологічному стеку, що включає мову програмування Python та асинхронний вебфреймворк FastAPI, який забезпечує паралельну обробку запитів і високу пропускну здатність системи без блокування основного потоку виконання. Ключовим науково-практичним досягненням проєкту є розробка й програмна реалізація унікального ансамблевого алгоритму з ковзним вікном. Цей підхід дозволив об’єднати обчислювальну потужність глибокої нейромережі архітектури трансформерів RoBERTa, спеціалізованої на детекції математичної передбачуваності токенів, та евристичного модуля пошуку явних лексичних маркерів-кліше за допомогою регулярних виразів. Завдяки повній контейнеризації екосистеми засобами Docker та Docker Compose, а також успішному впровадженню алгоритму лінивої ініціалізації масивних моделей, вдалося мінімізувати споживання апаратних ресурсів хост-сервера. Система демонструє високу стабільність функціонування та швидкий старт у фоновому режимі на серверному обладнанні з обмеженим обсягом оперативної пам’яті.

З точки зору функціональних можливостей, розроблений вебзастосунок забезпечує безперервний повноцикловий аналіз документів. Сюди входить не лише ізольоване розпізнавання прихованих ознак генерації ШІ, а й автоматичне семантичне групування інформації за тематичними категоріями методом Zero-Shot Classification на базі моделі BART, а також визначення емоційного забарвлення тексту через дистильовану нейромережу DistilBERT. Практична цінність інтерфейсних рішень підтверджується впровадженням інтерактивної триколірної смужки розподілу авторства, яка наочно візуалізує частки оригінального, змішаного та штучного контенту, роблячи висновки системи прозорими для користувача. Інтеграція підсистеми генерації звітів забезпечує автоматичне формування офіційних протоколів аналізу із фіксацією часових міток і метаданих, що робить продукт повністю готовим до використання в реальному електронному документообігу та моніторингу загального стану академічної доброчесності в межах факультету чи всього закладу.

Емпірична верифікація та комплексне тестування розробленого програмного забезпечення на реальних текстових масивах повністю підтвердили його працездатність та обґрунтованість висунутих рішень. Застосування інструментів PyTest дозволило покрити Unit-тестами понад вісімдесят п'ять відсотків вихідного коду, гарантуючи стійкість бізнес-логіки. Під час системних випробувань інтегроване AI-ядро продемонструло високі показники точності: загальна точність Accuracy детектора RoBERTa склала дев'яносто чотири і два десятих відсотка при збалансованій F1-мірі на рівні дев'яноста трьох сотих, тоді як точність класифікатора категорій BART досягла вісімдесяти дев'яти і чотирьох десятих відсотка. Алгоритм ковзного вікна успішно ідентифікує складні комбіновані та перефразовані тексти, виявляючи точкові машинна вкраплення та повністю нівелюючи проблему розмиття сигналу, яка притаманна більшості сучасних закритих комерційних аналогів.

Наукова та практична цінність одержаних результатів полягає у створенні автономного, гнучкого цифрового продукту, розгортання якого всередині локальної інфраструктури коледжу забезпечує абсолютну конфіденційність перевірок та усуває залежність від сторонніх платних сервісів. Отримані технічні рішення та накопичена

аналітична база формують надійний теоретичний і практичний фундамент для успішного захисту дипломної роботи. Перспективами подальшого розвитку та супроводу даної інтелектуальної системи є планомірне розширення локальної бази лексичних кліше, оптимізація нейромережевих моделей під мультимовні корпуси даних, розробка спеціалізованих плагінів для безпосередньої інтеграції із системами управління навчанням, а також дослідження можливостей впровадження квантованих версій великих мовних моделей архітектури Mistral або Llama для проведення розширеного контекстуального аналізу та автоматичного анотування документів.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Apache JMeter [Електронний ресурс] – Режим доступу: https://jmeter.apache
2. Artem Kovera Machine Learning with Clustering: A Visual Guide for Beginners with Examples in Python. United States: 2017. 56c.
3. Ashish Vaswani et al.. "Attention Is All You Need". United States of America: 2017. 11c.
4. Camille Thibault, Jacob-Junqi Tian A Guide to Misinformation Detection Data and Evaluation. United States of America: 2025. 10c.
5. Christopher M. Bishop. Pattern Recognition and Machine Learning. United Kingdom: 2006. 681c.
6. Eric Freeman and Elisabeth Robson: Head First HTML and CSS. Canada: 2012. 764c.
7. Ho Hung Lim, Tianyuan Cai, John S. Y. Lee, Meichun Liu Robustness of Hybrid Models in Cross-domain Readability Assessment. China: 2022. 6c.
8. Integrate TinyMCE [Електронний ресурс] – Режим доступу: https://www.tiny.cloud
9. Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti Zadeh Feature Extraction 2006. 781c.
10. Learning SQL. [Електронний ресурс] – Режим доступу: http://www.r-5.org.
11. Lynn Beighley and Michael Morrison Head First PHP & MySQL. United States of America: 2009. 762c.
12. Manual Detection of AI-Generated Writing. 2025. 36c.
13. MySQL Documentation. [Електронний ресурс] – Режим доступу: https://dev.mysql.com
14. Shekhar Khandelwal Deep Learning for Data Architects. India: 2023. 262c.
15. Tshilidzi Marwala Synthetic Versus Authentic Data. 2024. 120c.
16. Welling L., Thomson L. PHP and MySQL Web Development – Australia: 2005. 984c.

17. XAMPP Apache [Електронний ресурс] – Режим доступу: https://www.apachefriends.org

18. Модульне тестування [Електронний ресурс] – Режим доступу: https://qalight.ua/baza-znaniy

19. Напрямки тестування [Електронний ресурс] – Режим доступу: https://training.qatestlab.com

ДОДАТОК А Лістинг розробленої системи

**main.py**

```
from fastapi import FastAPI, UploadFile, File, Form, Depends
from fastapi.staticfiles import StaticFiles
from fastapi.responses import FileResponse
from sqlalchemy.orm import Session
import asyncio
import io
import docx
from pypdf import PdfReader
from deep_translator import GoogleTranslator

from app.db.database import engine, get_db, SessionLocal
from app.db import models, crud
from app.ml_core.sentiment import get_real_sentiment
from app.ml_core.detector import get_ai_probability
from app.ml_core.classifier import get_text_category
from app.ml_core.plagiarism import check_internal_plagiarism
from app.ml_core.explain import analyze_ai_markers
models.Base.metadata.create_all(bind=engine)

app = FastAPI(title="Система аналізу інформації")
app.mount("/static", StaticFiles(directory="app/static"), name="static")

@app.on_event("startup")
async def startup_event():

    db = SessionLocal()
    if not crud.get_user(db, "admin"):
        crud.create_user(db, "admin", "admin123", "admin", "Головний Адміністратор")
```

```
    db.close()

@app.get("/")
def serve_index(): return FileResponse("app/static/index.html")

#адмін та авторизація
@app.post("/api/login")
def login(username: str = Form(...), password: str = Form(...), db: Session = Depends(get_db)):
    user = crud.get_user(db, username)
    if not user or user.password != password:
        return {"status": "error", "message": "Невірний логін або пароль"}
        return {"status": "success", "role": user.role, "user_id": user.id, "username": user.username, "full_name": user.full_name}

@app.post("/api/teachers")
def add_teacher(username: str = Form(...), password: str = Form(...), full_name: str = Form(...), db: Session = Depends(get_db)):
    if crud.get_user(db, username): return {"status": "error", "message": "Логін зайнятий!"}
    crud.create_user(db, username, password, "teacher", full_name)
    return {"status": "success", "message": "Викладача створено!"}

@app.get("/api/teachers")
def get_teachers(db: Session = Depends(get_db)):
    teachers = db.query(models.User).filter(models.User.role == "teacher").all()
    return {"teachers": [{"id": t.id, "full_name": t.full_name, "username": t.username} for t in teachers]}

#групи,студенти, назви робіт
@app.get("/api/groups")
```

```
def get_groups(db: Session = Depends(get_db)):
    return {"groups": [{"id": g.id, "name": g.name} for g in crud.get_groups(db)]}

@app.post("/api/groups")
def add_group(name: str = Form(...), db: Session = Depends(get_db)):
    crud.create_group(db, name)
    return {"status": "success"}

@app.get("/api/students")
def get_students(group_id: int = None, db: Session = Depends(get_db)):
    return {"students": [{"id": s.id, "full_name": s.full_name, "group_id": s.group_id} for s
in crud.get_students(db, group_id)]}

@app.post("/api/students")
def add_student(full_name: str = Form(...), group_id: int = Form(...), db: Session =
Depends(get_db)):
    crud.create_student(db, full_name, group_id)
    return {"status": "success"}

@app.get("/api/tags")
def get_tags(db: Session = Depends(get_db)):
    return {"tags": crud.get_unique_tags(db)}

#видалення
@app.delete("/api/analysis/{item_id}")
def delete_analysis(item_id: int, db: Session = Depends(get_db)):
    return {"status": "success" if crud.delete_analysis(db, item_id) else "error"}

@app.delete("/api/teachers/{item_id}")
def del_teacher(item_id: int, db: Session = Depends(get_db)):
```

```
    crud.delete_user(db, item_id); return {"status": "success"}

@app.delete("/api/groups/{item_id}")
def del_group(item_id: int, db: Session = Depends(get_db)):
    crud.delete_group(db, item_id); return {"status": "success"}

@app.delete("/api/students/{item_id}")
def del_student(item_id: int, db: Session = Depends(get_db)):
    crud.delete_student(db, item_id); return {"status": "success"}

#аналіз
def process_analysis_save(db, text, student_id, teacher_id, tag, is_file=False):
    existing = db.query(models.AnalysisHistory).all()
    plag_score = check_internal_plagiarism(text, existing)

    try: text_for_ai = GoogleTranslator(source='auto', target='en').translate(text[:4500])
    except: text_for_ai = text

    real_sentiment = get_real_sentiment(text_for_ai)

    #словник
    ai_data = get_ai_probability(text_for_ai)
    real_ai_prob = ai_data["score"]
    ai_breakdown = ai_data["breakdown"]

    real_category = get_text_category(text)

    explanation = analyze_ai_markers(text, real_ai_prob)
    #результат
    explanation["ai_breakdown"] = ai_breakdown
```

```
    crud.create_analysis(db, text, real_ai_prob, real_sentiment, real_category, plag_score, student_id, teacher_id, tag)

    return {
        "status": "success",
        "results": {
            "ai_probability": real_ai_prob,
            "sentiment": real_sentiment,
            "category": real_category,
            "plagiarism_score": plag_score,
            "explanation": explanation,
            "is_file": is_file
        }
    }

@app.post("/api/analyze/text")
async def analyze_text(text: str = Form(...), student_id: int = Form(None), teacher_id: int = Form(None), tag: str = Form(None), db: Session = Depends(get_db)):
    return process_analysis_save(db, text, student_id, teacher_id, tag, is_file=False)

@app.post("/api/analyze/file")
async def analyze_file(file: UploadFile = File(...), student_id: int = Form(None), teacher_id: int = Form(None), tag: str = Form(None), db: Session = Depends(get_db)):
    content = await file.read()
    text = ""
    if file.filename.lower().endswith(".txt"):
        text = content.decode("utf-8", errors="ignore")
    elif file.filename.lower().endswith(".pdf"):
        try:
```

```
            for page in PdfReader(io.BytesIO(content)).pages: text += (page.extract_text() or "") + "\n"
        except: return {"status": "error", "message": "Не вдалося прочитати PDF"}
    elif file.filename.lower().endswith(".docx"): # НОВЕ: ЧИТАННЯ DOCX
        try:
            doc = docx.Document(io.BytesIO(content))
            text = "\n".join([para.text for para in doc.paragraphs])
        except: return {"status": "error", "message": "Не вдалося прочитати DOCX"}
    else:
        return {"status": "error", "message": "Підтримуються лише TXT, PDF, DOCX"}

    if not text.strip(): return {"status": "error", "message": "Файл порожній"}
    return process_analysis_save(db, text, student_id, teacher_id, tag, is_file=True)

#аналітика
@app.get("/api/statistics")
def get_statistics(group_id: int = None, tag: str = None, teacher_id: int = None, page: int = 1, db: Session = Depends(get_db)):
    limit = 50
    query = db.query(models.AnalysisHistory).join(models.Student, isouter=True)
    if group_id: query = query.filter(models.Student.group_id == group_id)
    if tag: query = query.filter(models.AnalysisHistory.tag == tag)
    if teacher_id: query = query.filter(models.AnalysisHistory.teacher_id == teacher_id)

    total_count = query.count()
    records = query.order_by(models.AnalysisHistory.created_at.desc()).offset((page - 1) * limit).limit(limit).all()

    data = [{
        "id": r.id,
```

```
        "student_name": r.student.full_name if r.student else "Анонімно",
        "tag": r.tag or "Без тегу",
        "ai_probability": r.ai_probability,
        "plagiarism_score": r.plagiarism_score,
        "sentiment": r.sentiment,
        "category": r.category,
        "date": r.created_at.strftime("%d.%m.%Y %H:%M") if r.created_at else ""
    } for r in records]

    return {"status": "success", "data": data, "total_pages": max(1, (total_count + limit - 1) //
limit), "current_page": page}
```

**detector.py**

```
from transformers import pipeline

detector_pipe = None

def get_ai_probability(text: str) -> dict:
    global detector_pipe

    if not text or len(text.strip()) < 10:
        return {"score": 0.0, "breakdown": {"ai": 0, "mixed": 0, "human": 100}}

    if detector_pipe is None:
        print("Завантаження моделі детектора ШІ...")
        detector_pipe = pipeline("text-classification", model="roberta-base-openai-detector")

    #розбиття на абзаци і речення
    raw_chunks = [p.strip() for p in text.split('\n') if len(p.strip()) > 30]
```

```
if len(raw_chunks) <= 1 and len(text) > 300:
    raw_chunks = [s.strip() + '.' for s in text.split('.') if len(s.strip()) > 30]
if not raw_chunks:
    raw_chunks = [text]

scores = []
for chunk in raw_chunks:
    safe_chunk = chunk[:1500]
    try:
        result = detector_pipe(safe_chunk)[0]
        score = result['score'] * 100 if result['label'] == 'Fake' else (1 - result['score']) * 100
        scores.append(score)
    except Exception:
        continue

if not scores:
    return {"score": 0.0, "breakdown": {"ai": 0, "mixed": 0, "human": 100}}

#розподіл
total = len(scores)
ai_chunks = sum(1 for s in scores if s > 70)
mix_chunks = sum(1 for s in scores if 30 <= s <= 70)
hum_chunks = sum(1 for s in scores if s < 30)

ai_pct = round((ai_chunks / total) * 100)
hum_pct = round((hum_chunks / total) * 100)
mix_pct = 100 - ai_pct - hum_pct

#загальний бал
final_score = round((sum(scores) / total + max(scores)) / 2, 1)
```

```
    return {
        "score": final_score,
        "breakdown": {"ai": ai_pct, "mixed": mix_pct, "human": hum_pct}
    }
```

**crud.py**

```
from sqlalchemy.orm import Session
from app.db import models

#користувачі
def get_user(db: Session, username: str):
    return db.query(models.User).filter(models.User.username == username).first()

def create_user(db: Session, username: str, password: str, role: str = "teacher", full_name: str = None):
        db_user = models.User(username=username, password=password, role=role, full_name=full_name)
    db.add(db_user)
    db.commit()
    db.refresh(db_user)
    return db_user

#групи
def get_groups(db: Session):
    return db.query(models.Group).all()

def create_group(db: Session, name: str):
```

```
    db_group = models.Group(name=name)
    db.add(db_group)
    db.commit()
    db.refresh(db_group)
    return db_group

#студенти
def get_students(db: Session, group_id: int = None):
    query = db.query(models.Student)
    if group_id:
        query = query.filter(models.Student.group_id == group_id)
    return query.all()

def create_student(db: Session, full_name: str, group_id: int):
    db_student = models.Student(full_name=full_name, group_id=group_id)
    db.add(db_student)
    db.commit()
    db.refresh(db_student)
    return db_student

#аналіз
def create_analysis(db: Session, text: str, ai_prob: float, sentiment: str, category: str,
plag_score: float, student_id: int = None, teacher_id: int = None, tag: str = None):
    db_record = models.AnalysisHistory(
        text_content=text,
        ai_probability=ai_prob,
        sentiment=sentiment,
        category=category,
        plagiarism_score=plag_score,
        student_id=student_id,
```

```
        teacher_id=teacher_id,
        tag=tag
    )
    db.add(db_record)
    db.commit()
    db.refresh(db_record)
    return db_record

def get_analyses(db: Session):
    return db.query(models.AnalysisHistory).all()

def get_unique_tags(db: Session):
      records = db.query(models.AnalysisHistory.tag).filter(models.AnalysisHistory.tag != None).distinct().all()
    return [r[0] for r in records if r[0].strip()]

#видалення
def delete_analysis(db: Session, record_id: int):
        record = db.query(models.AnalysisHistory).filter(models.AnalysisHistory.id == record_id).first()
    if record:
        db.delete(record)
        db.commit()
        return True
    return False

def delete_user(db: Session, user_id: int):
    user = db.query(models.User).filter(models.User.id == user_id).first()
    if user:
        db.delete(user)
```

```
        db.commit()
        return True
    return False

def delete_group(db: Session, group_id: int):
    group = db.query(models.Group).filter(models.Group.id == group_id).first()
    if group:
                                db.query(models.Student).filter(models.Student.group_id        ==
group_id).update({"group_id": None})
        db.delete(group)
        db.commit()
        return True
    return False

def delete_student(db: Session, student_id: int):
    student = db.query(models.Student).filter(models.Student.id == student_id).first()
    if student:
            db.query(models.AnalysisHistory).filter(models.AnalysisHistory.student_id   ==
student_id).update({"student_id": None})
        db.delete(student)
        db.commit()
        return True
    return False
```

**models.py**

```
from sqlalchemy import Column, Integer, String, Float, Text, ForeignKey, DateTime
from sqlalchemy.orm import relationship
from datetime import datetime
from app.db.database import Base
```

```
#користувачі
class User(Base):
    __tablename__ = "users"
    id = Column(Integer, primary_key=True, index=True)
    username = Column(String(50), unique=True, index=True)
    password = Column(String(100))
    full_name = Column(String(100), nullable=True)
    role = Column(String(20), default="teacher")

    analyses = relationship("AnalysisHistory", back_populates="teacher")

#групи
class Group(Base):
    __tablename__ = "groups"
    id = Column(Integer, primary_key=True, index=True)
    name = Column(String(50), unique=True, index=True)

    students = relationship("Student", back_populates="group")

#студенти
class Student(Base):
    __tablename__ = "students"
    id = Column(Integer, primary_key=True, index=True)
    full_name = Column(String(100), index=True)
    group_id = Column(Integer, ForeignKey("groups.id"))

    group = relationship("Group", back_populates="students")
    analyses = relationship("AnalysisHistory", back_populates="student")
```

```
#перевірки
class AnalysisHistory(Base):
    __tablename__ = "analysis_history"
    id = Column(Integer, primary_key=True, index=True)
    text_content = Column(Text)
    ai_probability = Column(Float)
    sentiment = Column(String(50))
    category = Column(String(50))
    plagiarism_score = Column(Float, default=0.0)



    tag = Column(String(50), nullable=True)
    created_at = Column(DateTime, default=datetime.utcnow)

    #приналежність перевірки
    student_id = Column(Integer, ForeignKey("students.id"), nullable=True)
    teacher_id = Column(Integer, ForeignKey("users.id"), nullable=True)

    student = relationship("Student", back_populates="analyses")
    teacher = relationship("User", back_populates="analyses")
```

**database.py**

```
import os
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, declarative_base

SQLALCHEMY_DATABASE_URL = os.getenv(
```

```
    "DATABASE_URL",
    "mysql+pymysql://diploma_user:diploma_password@db:3306/diploma_db"
)

engine = create_engine(SQLALCHEMY_DATABASE_URL)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative_base()

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

**index.html**

```
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></scrip
t>
<script>
    let currentUserId, currentUserRole, currentFullName;
    let currentPage = 1, totalPages = 1;
    let chart1 = null, chart2 = null;

    function showAlert(msg, isError = true) {
        const box = document.getElementById('sysAlert');
```

```
        box.className = `alert alert-${isError ? 'danger' : 'success'} mb-3 fw-bold border-0 shadow-sm`;
        box.innerText = msg;
        box.classList.remove('d-none');
        setTimeout(() => box.classList.add('d-none'), 5000);
        window.scrollTo({ top: 0, behavior: 'smooth' });
    }

    async function login() {
        const u = document.getElementById('loginUsername').value;
        const p = document.getElementById('loginPassword').value;
        if (!u || !p) {
            document.getElementById('loginAlert').innerText = "Введіть логін та пароль";
            document.getElementById('loginAlert').classList.remove('d-none');
            return;
        }

        const fd = new FormData(); fd.append("username", u); fd.append("password", p);
        const res = await fetch('/api/login', { method: 'POST', body: fd });
        const data = await res.json();

        if (data.status === "error") {
            document.getElementById('loginAlert').innerText = data.message;
            document.getElementById('loginAlert').classList.remove('d-none');
            return;
        }

        currentUserId = data.user_id; currentUserRole = data.role; currentFullName = data.full_name || data.username;
```

```
    document.getElementById('loginView').style.display = 'none';
    document.getElementById('logoutBtn').classList.remove('d-none');

    const greeting = document.getElementById('userGreeting');
    greeting.innerText = `${currentFullName}`;
    greeting.classList.remove('d-none');

    if (currentUserRole === 'admin') {
        document.getElementById('adminView').style.display = 'flex';
        loadAdminData();
    } else {
        document.getElementById('teacherView').style.display = 'flex';
        loadGroupsTeacher(); loadTags(); switchTeacherTab('check');
    }
}

function logout() { location.reload(); }

//адмін
async function loadAdminData() {
    const resT = await fetch('/api/teachers'); const dataT = await resT.json();
    let htmlT = ''; dataT.teachers.forEach(t => htmlT += `<li class="list-group-item px-2 py-2 small fw-bold">${t.full_name} <button class="del-btn" onclick="delItem('teachers', ${t.id})">✕</button></li>`);
    document.getElementById('adminTeacherList').innerHTML = htmlT;

    const resG = await fetch('/api/groups'); const dataG = await resG.json();
    let selHtml = '<option value="">-- Оберіть групу --</option>', listHtml = '';
    dataG.groups.forEach(g => {
        selHtml += `<option value="${g.id}">${g.name}</option>`;
```

```
        listHtml += `<li class="list-group-item px-2 py-2 small fw-bold">${g.name} <button class="del-btn" onclick="delItem('groups', ${g.id})">✕</button></li>`;
    });
    document.getElementById('adminGroupSelect').innerHTML = selHtml;
    document.getElementById('adminGroupList').innerHTML = listHtml;
}

async function loadAdminStudents() {
    const gid = document.getElementById('adminGroupSelect').value;
    if (!gid) { document.getElementById('adminStudentList').innerHTML = ''; return; }
    const res = await fetch(`/api/students?group_id=${gid}`); const data = await res.json();
    let html = ''; data.students.forEach(s => html += `<li class="list-group-item px-2 py-2 small fw-bold">${s.full_name} <button class="del-btn" onclick="delItem('students', ${s.id})">✕</button></li>`);
    document.getElementById('adminStudentList').innerHTML = html;
}

async function delItem(type, id) {
    if(!confirm("Підтвердіть видалення. Дія незворотня.")) return;
    await fetch(`/api/${type}/${id}`, { method: 'DELETE' });
    if(type === 'teachers' || type === 'groups') loadAdminData();
    if(type === 'students') loadAdminStudents();
    if(type === 'analysis') loadStatistics();
}

async function createTeacher() {
    const fd = new FormData(); fd.append("full_name", document.getElementById('adminTeacherName').value); fd.append("username", document.getElementById('adminTeacherLogin').value); fd.append("password", document.getElementById('adminTeacherPass').value);
```

```
    await fetch('/api/teachers', { method: 'POST', body: fd }); loadAdminData();
  }
  async function createGroup() {
                    const fd = new FormData(); fd.append("name", document.getElementById('adminGroupName').value);
    await fetch('/api/groups', { method: 'POST', body: fd }); loadAdminData();
  }
  async function createStudent() {
                    const fd = new FormData(); fd.append("group_id", document.getElementById('adminGroupSelect').value); fd.append("full_name", document.getElementById('adminStudentName').value);
    await fetch('/api/students', { method: 'POST', body: fd }); loadAdminStudents();
  }

  //викладач
  async function loadGroupsTeacher() {
    const res = await fetch('/api/groups'); const data = await res.json();
    let sel = '<option value="">-- Оберіть групу --</option>';
    data.groups.forEach(g => sel += `<option value="${g.id}">${g.name}</option>`);
    document.getElementById('teacherGroupSelect').innerHTML = sel;
  }
  async function loadStudentsForTeacher() {
    const gid = document.getElementById('teacherGroupSelect').value;
    if (!gid) return;
    const res = await fetch(`/api/students?group_id=${gid}`); const data = await res.json();
    let sel = '<option value="">-- Анонімно --</option>';
                    data.students.forEach(s => sel += `<option value="${s.id}">${s.full_name}</option>`);
    document.getElementById('teacherStudentSelect').innerHTML = sel;
  }
```

```
    async function loadTags() {
        const res = await fetch('/api/tags'); const data = await res.json();
        document.getElementById('tagList').innerHTML = '';
         data.tags.forEach(t => document.getElementById('tagList').innerHTML += `<option value="${t}">`);
    }

    function showResults(data) {

        document.getElementById('results').style.display = 'block';

        //протокол
       document.getElementById('res-date').innerText = new Date().toLocaleString('uk-UA');
        document.getElementById('res-teacher').innerText = currentFullName;
        const stSel = document.getElementById('teacherStudentSelect');
                    document.getElementById('res-student').innerText = stSel.value ? stSel.options[stSel.selectedIndex].text : "Не вказано (Анонімно)";
                                        document.getElementById('res-tag').innerText = document.getElementById('workTag').value || "Без класифікації";

        //показники
        document.getElementById('res-plag').innerText = data.results.plagiarism_score + '%';
        document.getElementById('res-sentiment').innerText = data.results.sentiment;
        document.getElementById('res-category').innerText = data.results.category;

        // результат
        if(data.results.explanation && data.results.explanation.ai_breakdown) {
            const bd = data.results.explanation.ai_breakdown;

            document.getElementById('bar-ai').style.width = bd.ai + '%';
```

```
        document.getElementById('bar-mixed').style.width = bd.mixed + '%';
        document.getElementById('bar-human').style.width = bd.human + '%';

        document.getElementById('txt-ai').innerText = bd.ai + '%';
        document.getElementById('txt-mixed').innerText = bd.mixed + '%';
        document.getElementById('txt-human').innerText = bd.human + '%';
      }

      // висновок
      if(data.results.explanation) {
        document.getElementById('res-reason').innerText = data.results.explanation.reason;
        const hBox = document.getElementById('res-highlight-box');

                                                  if (!data.results.is_file &&
data.results.explanation.highlighted_text.includes('<mark>')) {
            hBox.style.display = 'block';
                                      document.getElementById('res-highlight').innerHTML =
data.results.explanation.highlighted_text.replace(/\n/g, '<br>');
        } else {
            hBox.style.display = 'none';
        }
      }

      document.getElementById('results').scrollIntoView({ behavior: 'smooth' });
    }

    //аналіз
    async function analyzeText() {
      const text = document.getElementById('textInput').value.trim();
```

```
    if (!text) return showAlert("❌ Помилка: Текстове поле порожнє. Вставте текст для аналізу.");

    const btn = document.querySelector('#text-pane button');
    btn.innerText = "ОБРОБКА..."; btn.disabled = true;
    try {
      const fd = new FormData(); fd.append("text", text);
        const sid = document.getElementById('teacherStudentSelect').value; const tag = document.getElementById('workTag').value;
            if(sid) fd.append("student_id", sid); if(tag) fd.append("tag", tag); fd.append("teacher_id", currentUserId);

      const res = await fetch('/api/analyze/text', { method: 'POST', body: fd });
      showResults(await res.json()); loadTags();
    } finally { btn.innerText = "Запустити AI-аналіз"; btn.disabled = false; }
  }

  async function analyzeFile() {
    const fileInput = document.getElementById('fileInput');
      if (!fileInput.files.length) return showAlert("❌ Помилка: Файл не обрано. Завантажте документ.");

    const btn = document.querySelector('#file-pane button');
    btn.innerText = "ОБРОБКА..."; btn.disabled = true;
    try {
      const fd = new FormData(); fd.append("file", fileInput.files[0]);
        const sid = document.getElementById('teacherStudentSelect').value; const tag = document.getElementById('workTag').value;
            if(sid) fd.append("student_id", sid); if(tag) fd.append("tag", tag); fd.append("teacher_id", currentUserId);
```

```
        const res = await fetch('/api/analyze/file', { method: 'POST', body: fd });
        const data = await res.json();
            if(data.status === "error") showAlert(data.message); else showResults(data);
loadTags();
    } finally { btn.innerText = "Проаналізувати файл"; btn.disabled = false; }
  }

  function downloadPDF() {
    const el = document.getElementById('results');
    const btn = document.getElementById('pdfBtn');
    btn.style.display = 'none';
    html2pdf().set({ margin: 10, filename: 'Protokol_IPZE.pdf', html2canvas: { scale: 2 }
}).from(el).save().then(() => btn.style.display = 'block');
  }

  //статистика і графіки
  function switchTeacherTab(tab) {
    document.getElementById('tab-check').style.display = tab === 'check' ? 'block' : 'none';
    document.getElementById('tab-stats').style.display = tab === 'stats' ? 'block' : 'none';

                                                    const btns =
document.getElementById('teacherTabs').getElementsByTagName('button');
    btns[0].classList.toggle('active', tab === 'check');
    btns[0].classList.toggle('bg-white', tab !== 'check');
    btns[1].classList.toggle('active', tab === 'stats');
    btns[1].classList.toggle('bg-white', tab !== 'stats');

    if (tab === 'stats') {
```

```
        document.getElementById('statsGroupSelect').innerHTML = '<option value="">-- Всі групи --</option>' + document.getElementById('teacherGroupSelect').innerHTML.replace('<option value="">-- Оберіть групу --</option>', '');
        fetch('/api/tags').then(r => r.json()).then(data => {
            let tHtml = '<option value="">-- Всі види контролю --</option>'; data.tags.forEach(t => tHtml += `<option value="${t}">${t}</option>`);
            document.getElementById('statsTagSelect').innerHTML = tHtml;
            currentPage = 1; loadStatistics();
        });
    }
}

function changePage(dir) {
    if (currentPage + dir > 0 && currentPage + dir <= totalPages) {
        currentPage += dir; loadStatistics();
    }
}

async function loadStatistics() {
        const grp = document.getElementById('statsGroupSelect'); const tg = document.getElementById('statsTagSelect');
    const gid = grp ? grp.value : ''; const tag = tg ? tg.value : '';

    let url = `/api/statistics?teacher_id=${currentUserId}&page=${currentPage}`;
    if (gid) url += `&group_id=${gid}`; if (tag) url += `&tag=${tag}`;

    const res = await fetch(url); const data = await res.json();
    totalPages = data.total_pages;
```

```
        document.getElementById('pageIndicator').innerText = `Стор. ${currentPage} з ${totalPages}`;

        let totalAi = 0, totalPlag = 0, count = data.data.length;
        let categories = {}, aiDistribution = [0, 0, 0];
        let tbody = '';

        data.data.forEach(r => {
            totalAi += r.ai_probability; totalPlag += r.plagiarism_score;
            categories[r.category] = (categories[r.category] || 0) + 1;
                if(r.ai_probability < 30) aiDistribution[0]++; else if(r.ai_probability < 70) aiDistribution[1]++; else aiDistribution[2]++;

            tbody += `<tr>
                                <td class="text-muted small">${r.date}</td><td class="fw-bold">${r.student_name}</td><td><span class="badge bg-dark">${r.tag}</span></td>
                                <td class="${r.ai_probability > 50 ? 'text-danger fw-bold' : ''}">${r.ai_probability}%</td>
                                <td class="${r.plagiarism_score > 30 ? 'text-warning fw-bold' : ''}">${r.plagiarism_score}%</td>
                                <td><button class="btn btn-sm btn-outline-danger py-0" onclick="delItem('analysis', ${r.id})">Видалити</button></td>
            </tr>`;
        });

        document.getElementById('statsTableBody').innerHTML = count ? tbody : '<tr><td colspan="6" class="text-center p-4 text-muted fw-bold">Жодної роботи не знайдено</td></tr>';
        document.getElementById('statCount').innerText = count;
```

```
        document.getElementById('statAvgAi').innerText = count ? (totalAi/count).toFixed(1) + '%' : '0%';
                        document.getElementById('statAvgPlag').innerText = count ? (totalPlag/count).toFixed(1) + '%' : '0%';

        updateCharts(categories, aiDistribution);
    }

    function updateCharts(categories, aiDistribution) {
        if(chart1) chart1.destroy(); if(chart2) chart2.destroy();
        const ctx1 = document.getElementById('chartCategory').getContext('2d');
         chart1 = new Chart(ctx1, { type: 'doughnut', data: { labels: Object.keys(categories), datasets: [{ data: Object.values(categories), backgroundColor: ['#111111', '#fdd54f', '#6c757d', '#dc3545', '#198754'] }] }, options: { maintainAspectRatio: false } });
        const ctx2 = document.getElementById('chartAi').getContext('2d');
          chart2 = new Chart(ctx2, { type: 'bar', data: { labels: ['<30%', '30-70%', '>70%'], datasets: [{ label: 'Кількість робіт', data: aiDistribution, backgroundColor: ['#198754', '#fdd54f', '#dc3545'] }] }, options: { maintainAspectRatio: false, scales: { y: { beginAtZero: true, ticks: { stepSize: 1 } } } } });
    }
</script>
</body>
</html>
```

ДОДАТОК Б Презентація

Національний технічний університет України
«Київський політехнічний інститут

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

**Дипломна робота**

# Система машинного навчання для аналізу і групування природної та штучно створеної інформації

Виконала: група ТВ-21 Сатко Дар'я Олегівна
Керівник: Недашківський Олексій Леонідович, професор кафедри інженерія програмного забезпечення в енергетиці, доцент кафедри інфокомунікацій

2026

## Постановка задачі

Стрімкий розвиток використання штучного інтелекту спонукає виникнення інструментів для його виявлення та оброблення.
Цифрові покращення є як ніколи актуальними у світлі цифровізації системи навчання. Як вчителі так і студенти мають використовувати штучний інтелект лише в міру обставин, тому потрібні зручні інструменти для регуляції використання штучного інтелекту.
Мета роботи - створення чіткого, швидкого та інтуїтивного продукту для виявлененя штучно створеної та природної інформації викладачем навчального закладу у студентів. Досягнення цієї мети реалізується через поєднання передових нейромережевих архітектур трансформерів та алгоритмів семантичного аналізу.

## Мета програмного продукту

Мета проєкту розробка інтелектуальної системи аналізу та групування природної та штучно створеної інформації для контролю академічної доброчесності.
Задачі проєкту:

- дослідити існуючі методи детекції ШІ-текстів та аналізу тональності;
- сформулювати функціональні та архітектурні вимоги до системи;
- розробити ансамблевий метод аналізу змішаних текстів на базі моделей Transformer;
- спроектувати модульну архітектуру та реалізувати вебсервіс із базою даних;
- верифікувати точність моделей та провести системне тестування.



## Аналіз предметної області

Стрімкий розвиток генеративного ШІ робить традиційні антиплагіатні системи неефективними.
Недоліки існуючих рішень це:

- закритий вихідний код та висока вартість ліцензій.
- відсутність гнучкості для локального розгортання в закладах освіти.
- слабке розпізнавання складної морфології та перефразувань.

Проблема пропуску згенерованих абзаців у великих людських текстах.

## Проєктування системи

### Функціональні вимоги Ролі Адміністратора та Викладача

Методом вирішення є ансамблевий підхід, поабзацний аналіз тексту.
Функціональні вимоги до системи це авторизація, завантаження документів, багаторівневий аналіз.
Нефункціональні це контейнеризація в Docker, асинхронність, Lazy Loading ML-моделей.


**Рисунок 1 - діаграма прецедентів**



## Логічна структура

### Модульна організація коду. Розподіл обов'язків між API, ML-ядром, схемами та СУБД.

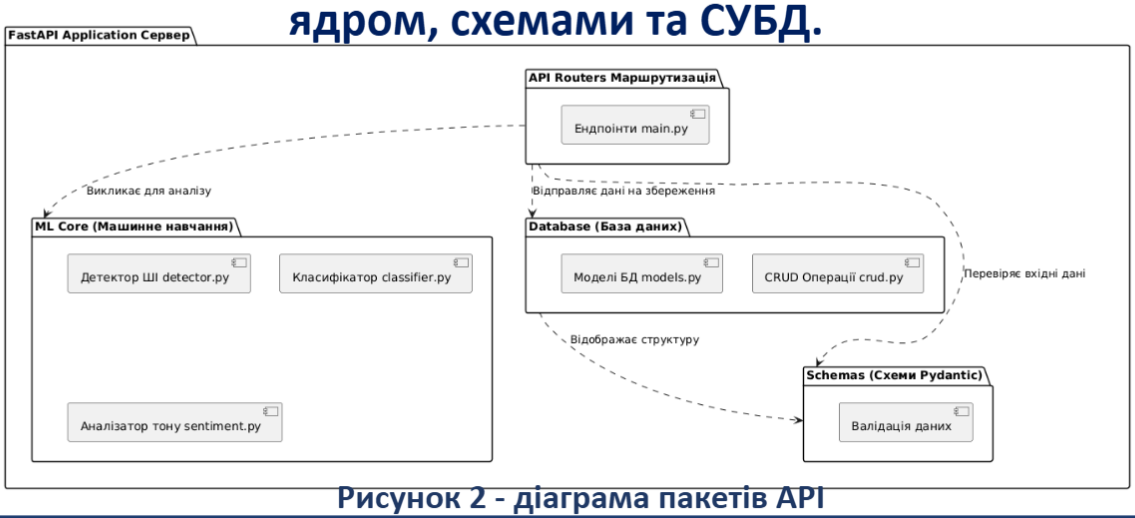



**Рисунок 2 - діаграма пакетів API**

# Взаємодія компонентів

## Роль MainAnalyzer як координатора моделей.

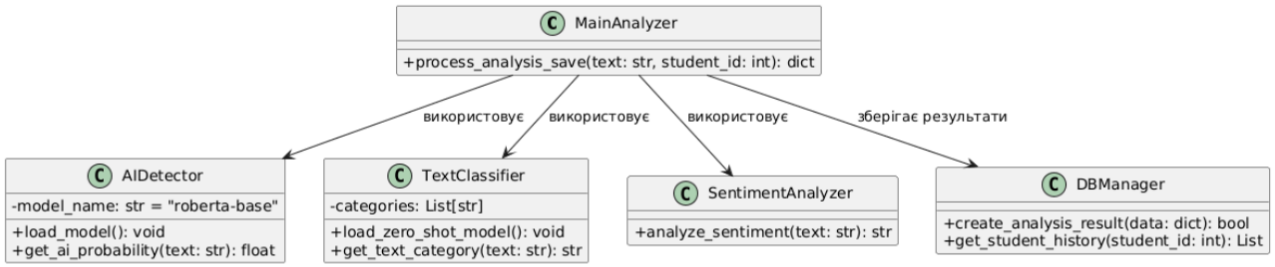



Рисунок 3 - діаграма класів



# Алгоритм аналізу

## Ансамблевий підхід та поабзацна обробка

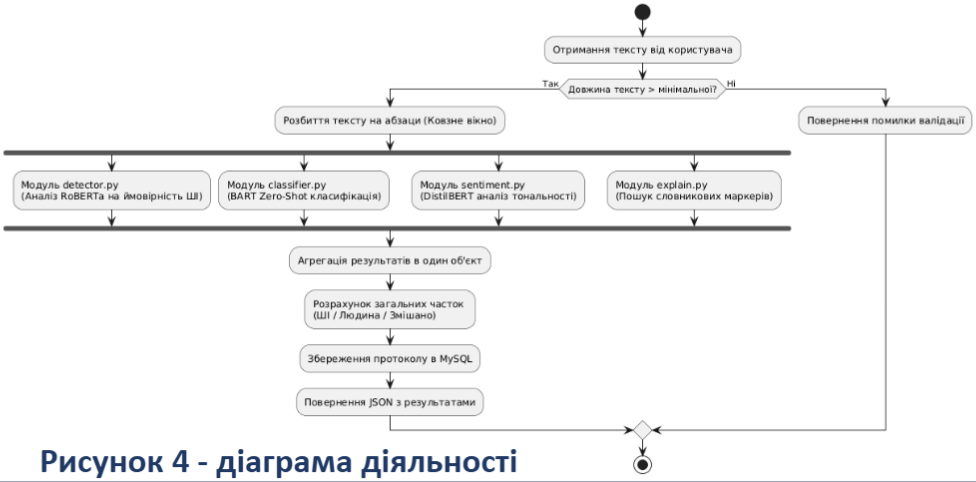



Рисунок 4 - діаграма діяльності

# Динаміка системи

## Життєвий цикл HTTP-запиту від фронтенду до збереження в MySQL.

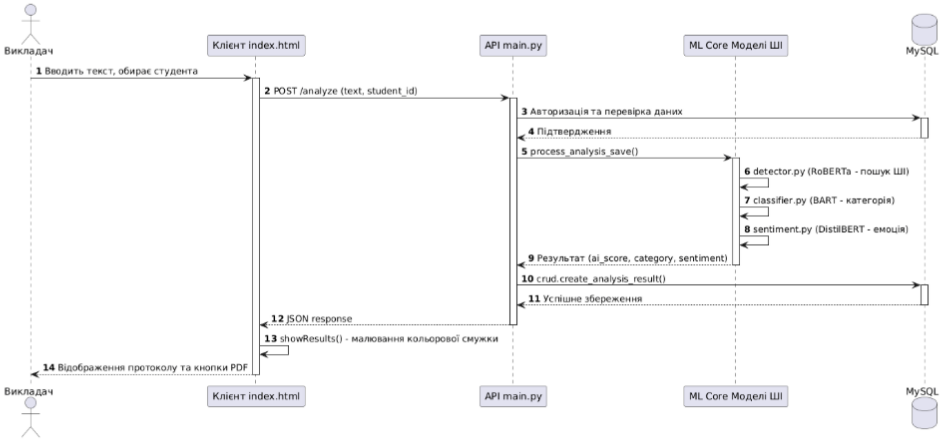



**Рисунок 5 - діаграма послідовності**



# Проєктування бази даних

## Схема збереження результатів Нормалізація даних, сутності та зв'язки

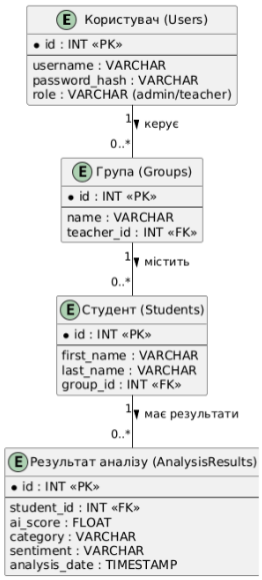



**Рисунок 6 - ER-діаграма**

# Програмна реалізація та інтерфейс

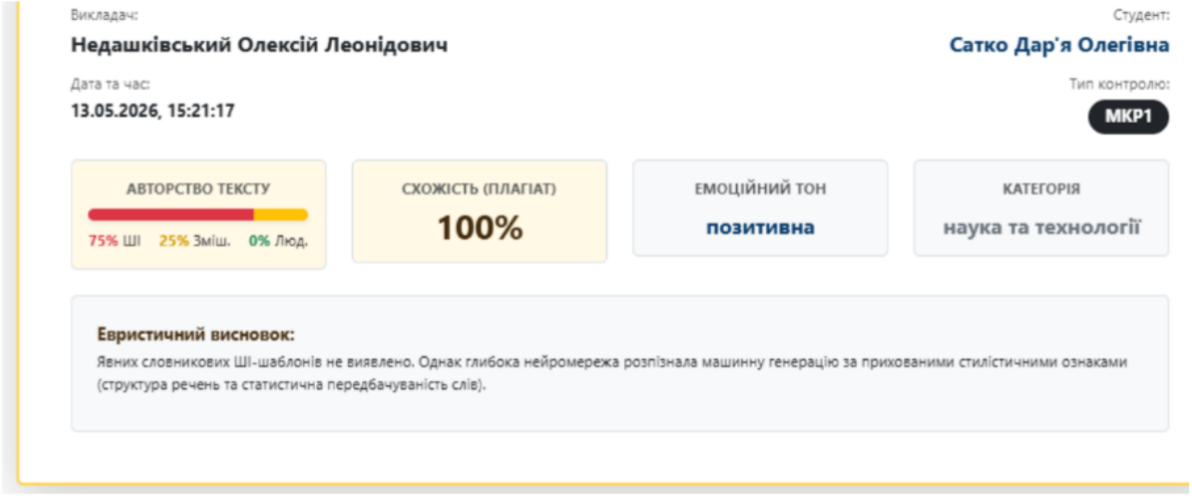
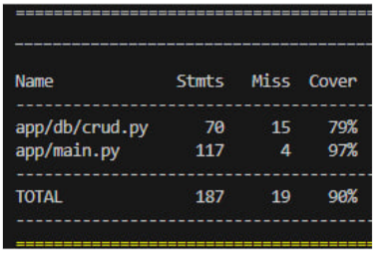



**Рисунок 7 - інтервейс викладача**



# Верифікація та тестування системи

**Інструмент автоматизації:** тестовий фреймворк PyTest у середовищі Docker.
**Об'єкт верифікації:** наскрізне інтеграційне тестування End-to-End.
**Результат:** автоматизована перевірка доступності та стабільності клієнт-серверного з'єднання завершилась успішно.
**Покриття інтеграційного сценарію** 90% UNIT-тестами



| Name | Stmts | Miss | Cover |
|---|---|---|---|
| app/db/crud.py | 70 | 15 | 79% |
| app/main.py | 117 | 4 | 97% |
| TOTAL | 187 | 19 | 90% |

**Рисунок 8 - відсоток покриття юніт-тестами**

## Інфраструктура та розгортання:

### Оркестрація в Docker Compose
### Ізоляція контейнерів

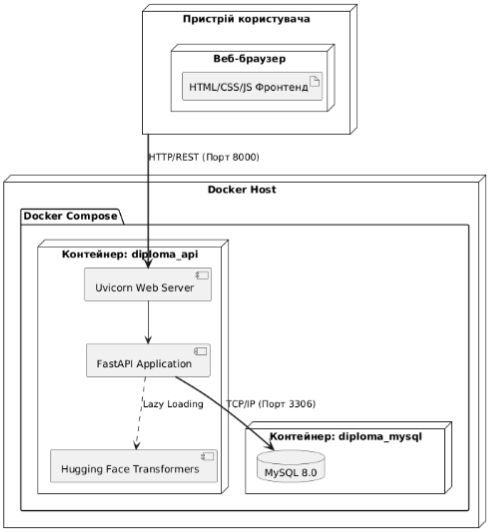



**Рисунок 9 - діаграма деплойменту**



## Висновки та перспективи

### Практична цінність продукту та вектори подальшого розвитку.

По закінченню роботи досліджено предметну область та визначено слабкі місця наявних аналогів. Сформульовано чіткі архітектурні та функціональні вимоги до вебсервісу. Розроблено та впроваджено унікальний ансамблевий алгоритм з ковзним вікном. Програмно реалізовано модульний застосунок на FastAPI у Docker-контейнерах. Проведено верифікацію та підтверджено високу точність розпізнавання ШІ.