Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Інтеграція інтелектуального асистента та технологій ШІ для організації наукових конференцій на основі LLM та PFE
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-3

Максим Теплюк
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки

(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект

(повна назва освітньої програми)
Керівник ст. викл. Олена Гриньова
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Теплюку Максиму Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інтеграція інтелектуального асистента та технологій ШІ для організації наукових конференцій на основі LLM та PFE

затверджена наказом університету від 29 травня 20 25 р. № 378Ст
2. Термін подання студентом роботи до екзаменаційної комісії 19 червня 20 25 р.
3. Вихідні дані до роботи Python, FastAPI, Huggingface Transformers, PyMuPDF, Jinja2, HTML, CSS, JavaScript, OpenAI API, Copyleaks API, матеріали конференцій, RoBERTa, SBERT, Visual Studio Code.

4. Перелік питань, що потрібно опрацювати в роботі
1) Теоретичні основи інтеграції ШІ в організацію наукових конференцій
2) Постановка задачі і загальна концепція реалізації асистента
3) Проєктування та розробка інтелектуального помічника для наукових конференцій
4) Візуальне представлення та елементи інтерфейсу користувача

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	20.05 – 23.05	виконано
3	Постановка задачі	24.05 – 26.05	виконано
4	Обґрунтування вибору використаних технологій	27.05 – 30.05	виконано
5	Програмна реалізація	31.05 – 05.06	виконано
6	Оформлення пояснювальної записки	06.06 – 14.06	виконано
7	Підготовка до захисту	15.06 – 18.06	виконано
8	Захист перед ЕК	19.06.2025	виконано

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Олена Гриньова
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 71 с., 5 рис., 1 дод., 26 джерел.

ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ, КОНФЕРЕНЦІЙНИЙ МЕНЕДЖМЕНТ, AI-ПОМІЧНИК, GENERATIVE PRE-TRAINED TRANSFORMER, LLM, NLP, NLU, PFE, REINFORCEMENT LEARNING WITH HUMAN FEEDBACK, SBERT.

Об’єкт дослідження – процеси автоматизації та цифровізації організації наукових конференцій з використанням сучасних моделей штучного інтелекту.

Предмет дослідження – застосування інтелектуального асистента, побудованого на основі LLM і PFE, для аналізу, валідації та супроводу подання наукових матеріалів у середовищі конференції.

Мета роботи – проєктування, формалізація та апробація інтелектуального асистента для підтримки учасників, організаторів та рецензентів наукової конференції.

Методи дослідження – архітектурне проєктування багатокомпонентних цифрових систем, методи обробки природної мови (NLP/NLU), технології Prompt Flow Engineering.

У роботі обґрунтовано необхідність впровадження інтелектуальних інструментів у цифрову інфраструктуру наукових заходів. Розроблений концепт асистента орієнтовано на три основні типи користувачів – учасників, організаторів і рецензентів – та забезпечено відповідні сценарії взаємодії. Показано, що застосування LLM та PFE дозволяє реалізувати гнучкі сценарії комунікації, прискорити процес оцінювання, зменшити дублювання тематики та покращити досвід користувача.

ABSTRACT

Bachelor’s thesis contains: 71 pp., 5 fig., 1 ann., 26 references.

AI-ASSISTANT, CONFERENCE MANAGEMENT, GENERATIVE PRE-TRAINED TRANSFORMER, INTELLIGENT SYSTEMS, LLM, NLP, NLU, PFE, REINFORCEMENT LEARNING WITH HUMAN FEEDBACK, SBERT.

The object of the study is the processes of automation and digitalization of the organization of scientific conferences using modern artificial intelligence models.

The subject of the study is the use of an intelligent assistant built on the basis of LLM and PFE for analysis, validation and support of the presentation of scientific materials in the conference environment.

The purpose of the work is the design, formalization and testing of an intelligent assistant to support participants, organizers and reviewers of a scientific conference.

Research methods are architectural design of multi-component digital systems, natural language processing methods (NLP/NLU), Prompt Flow Engineering technologies.

The work substantiates the need to implement intelligent tools in the digital infrastructure of scientific events. The developed assistant concept is focused on three main types of users - participants, organizers and reviewers - and provides appropriate interaction scenarios. It is shown that the use of LLM and PFE allows for the implementation of flexible communication scenarios, accelerate the evaluation process, reduce duplication of topics and improve the user experience.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

GPT – Generative Pre-trained Transformer – генеративний попередньо навчений трансформер;

LLM – Large Language Model – велика мовна модель;

NLP – Natural Language Processing – обробка природної мови;

NLU – Natural Language Understanding – розуміння природної мови;

PFE – Prompt Flow Engineering – інженерія потоків запитів;

RLHF – Reinforcement Learning with Human Feedback – навчання з підкріпленням на основі зворотного зв’язку людини;

SBERT – Sentence-BERT – модель для семантичного порівняння речень.

ВСТУП

У добу цифрової трансформації та стрімкого розвитку інтелектуальних технологій відбуваються глибокі зміни у всіх сферах людської діяльності, зокрема й у науці, освіті та академічному адмініструванні. Традиційні форми організації наукових заходів, які ще донедавна базувалися переважно на ручному модераторстві, статичних інструкціях і людській комунікації, дедалі більше потребують адаптації до нових викликів. Масове зростання кількості подань, підвищення вимог до академічної доброчесності, очікування учасників щодо інтерактивності та швидкості отримання зворотного зв'язку зумовлюють необхідність переосмислення функціонування наукових платформ. У цьому контексті штучний інтелект (ШІ) постає не як абстрактне технологічне нововведення, а як реальний інструмент вирішення практичних задач конференційного менеджменту.

Однією з найперспективніших технологій у цій сфері є великі мовні моделі, здатні до контекстного аналізу, генерації тексту, семантичного зіставлення та діалогової взаємодії з користувачем. Зокрема, інтеграція таких моделей у платформу наукової конференції дозволяє реалізувати інтелектуального асистента − цифрового агента, що може не лише відповідати на питання учасників, а й аналізувати структуру поданих робіт, здійснювати пошук схожих матеріалів, перевіряти академічну якість та формувати звіти для організаторів. У поєднанні з методологією Prompt Flow Engineering (PFE), що забезпечує гнучке проєктування сценаріїв взаємодії з ШІ, такі асистенти можуть радикально змінити підхід до адміністрування форумів, симпозіумів і конференцій.

У контексті україномовного академічного простору, інтеграція подібних рішень вимагає врахування низки специфічних викликів. По- перше, не всі LLM мають достатню лінгвістичну адаптацію до української мови, що ускладнює реалізацію повноцінної підтримки

користувачів. По-друге, більшість платформ і систем організації заходів орієнтовані на англомовну аудиторію, а отже, потребують лінгвістичного та концептуального доопрацювання. По-третє, культурно-комунікативний стиль взаємодії у вітчизняному академічному середовищі має власні особливості, які слід ураховувати при побудові логіки інтерфейсу та діалогу з користувачем. Саме тому створення україномовного, контекстно орієнтованого інтелектуального асистента, адаптованого до завдань реального наукового форуму, є як науково важливим, так і практично цінним напрямом дослідження.

1 ТЕОРЕТИЧНІ ОСНОВИ ІНТЕГРАЦІЇ ШІ В ОРГАНІЗАЦІЮ НАУКОВИХ КОНФЕРЕНЦІЙ

1.1 Сучасні виклики в організації наукових заходів

Упродовж останніх десятиліть наукові конференції зазнали значної трансформації, викликаної як зростанням обсягів поданих матеріалів, так і розвитком цифрових технологій. Науково-дослідницьке середовище дедалі активніше залучає до участі молодих учених та дослідників з усього світу, що, з одного боку, сприяє формуванню глобального інтелектуального простору, а з іншого − створює суттєві виклики в організації самих заходів. Усе це актуалізує необхідність перегляду традиційних методів організації конференцій, зокрема в аспекті автоматизації, цифровізації та забезпечення об’єктивності оцінювання наукових матеріалів.

Одним з головних викликів, із яким стикаються організатори наукових форумів, є перевантаження рецензентського корпусу. Зі зростанням кількості поданих матеріалів відбувається збільшення обсягу роботи, яку мають виконати експерти. Рецензенти змушені витрачати значні ресурси часу та уваги на аналіз кожного подання, перевірку його новизни, структури, наукової цінності та відповідності вимогам конференції. В умовах обмежених людських і часових ресурсів така практика призводить до затримок, суб’єктивних оцінок, а іноді й до пропусків важливих аспектів поданих досліджень. Як наслідок, знижується якість експертного оцінювання, що безпосередньо впливає на загальний науковий рівень заходу [1].

Другим аспектом є потреба в автоматизації рутинних процесів, що супроводжують організацію конференцій. Йдеться, зокрема, про попередню перевірку структури робіт, відповідність шаблону оформлення, тематичну класифікацію, виявлення дублікатів або схожих досліджень, а також надання відповідей на типові питання учасників. Усе це традиційно

здійснюється вручну організаторами або технічними секретарями, що у великих конференціях може вимагати залучення окремих робочих груп. Автоматизація таких задач за допомогою технологій штучного інтелекту дозволила б суттєво знизити навантаження на персонал, прискорити обробку матеріалів і забезпечити більш точне дотримання регламентів [2].

Третім ключовим викликом є проблема масштабування. У глобалізованому світі наукові заходи набувають міжнародного статусу, що вимагає забезпечення багатомовності, підтримки учасників з різних часових поясів, різних рівнів цифрової грамотності, а також гнучкої технічної інфраструктури для інтеграції з базами даних, системами рецензування, засобами комунікації. Масштабування традиційних моделей конференцій без відповідної цифрової трансформації стає неможливим або вкрай ресурсозатратним. Інтеграція інтелектуальних систем, здатних працювати з великими мовними моделями (LLM), відкриває нові можливості для ефективної комунікації, перекладу, виявлення дублювання, а також побудови рекомендаційних систем для тематичної класифікації [3].

Особливу увагу в цьому контексті заслуговує питання якості наукового контенту. Унаслідок збільшення кількості заявок та зниження бар’єру входу до участі, виникає проблема наукової репутації заходів: частина матеріалів може бути поверхневою, компілятивною або навіть з ознаками плагіату. Для підтримання рівня конференції організатори мають вдаватися до інструментів попереднього семантичного аналізу текстів, виявлення схожості, перевірки новизни тощо. Використання спеціалізованих систем на основі глибинного навчання дозволяє не лише здійснювати поверхневу перевірку, але й аналізувати глибинні контекстуальні відповідності між роботами, що особливо актуально для міждисциплінарних заходів [4].

На сучасному етапі організація наукових конференцій є не лише логістичним викликом, а й складною системною задачею, яка потребує впровадження цифрових інструментів, зокрема засобів штучного інтелекту.

1.2 Поняття та роль інтелектуального асистента в цифровому середовищі конференцій

1.2.1 Визначення та функціональні можливості інтелектуального асистента

На відміну від традиційних чат-ботів, інтелектуальні асистенти мають глибше розуміння контексту завдяки використанню великих мовних моделей. Такі моделі, зокрема GPT, RoBERTa, BERT та їхні похідні, забезпечують семантичну обробку тексту, підтримку багатомовності, здатність до генерації, узагальнення та класифікації інформації. Водночас, зростання складності моделей призвело до виникнення потреби в більш гнучкому управлінні їх поведінкою. У цьому контексті сформувався підхід Prompt Flow Engineering, система побудови ланцюгів промптів, яка дозволяє організувати багатокроковий діалог з LLM і динамічно змінювати логіку взаємодії залежно від типу запиту [5].

З архітектурної точки зору, інтелектуальний асистент складається з таких базових компонентів: модуль обробки природної мови (NLU/NLP), блок генерації або вибору відповіді, промпт-менеджер (якщо використовується PFE), база знань або підключення до зовнішніх джерел даних, а також інтерфейс користувача. У разі інтеграції в середовище наукової конференції інтелектуальний асистент може бути реалізований як вебкомпонент, підключений до бази заявок, шаблонів документів, FAQ та системи рецензування. Основною перевагою LLM у цьому контексті є здатність не лише аналізувати текст, але й адаптувати відповіді відповідно до наукового стилю, семантичного контексту й навіть виявляти порушення формату [6].

Серед ключових функціональних можливостей інтелектуального асистента виокремлюються:

− семантичний пошук. На відміну від ключового пошуку, LLM дозволяють аналізувати зміст запиту і порівнювати його з текстами інших документів, визначаючи не лише поверхневу схожість, а й глибинну тематичну відповідність. Це критично важливо у випадку пошуку аналогічних робіт, тем чи концептів у межах конференції;

− класифікація та категоризація. Інтелектуальні асистенти можуть автоматично розподіляти подані роботи за тематичними напрямами, визначати їхню належність до певного наукового поля або навіть фільтрувати за ступенем відповідності вимогам конференції. Це реалізується через попереднє навчання моделей на прикладах раніше класифікованих матеріалів;

− контекстно-залежна генерація відповідей. Система здатна генерувати інформативні, граматично та стилістично коректні відповіді на запити користувачів (учасників або організаторів), у тому числі в межах FAQ. На відміну від класичних шаблонних рішень, LLM дозволяє формувати гнучкі відповіді, які враховують і попередні запити, і поточну ситуацію (наприклад, статус заявки, дедлайн тощо);

− валідація структури робіт. Завдяки використанню моделей типу ChatGPT або інструментів, налаштованих на форматні перевірки, інтелектуальний асистент може перевіряти, чи містить наукова робота обов’язкові елементи: анотацію, ключові слова, розділи «Вступ», «Методологія», «Результати», «Висновки» тощо. Це дозволяє виявити роботи, які не відповідають вимогам, ще до етапу рецензування;

− розпізнавання мовних і стилістичних аномалій. Моделі, на кшталт RoBERTa або Ukr-RoBERTa-base, можуть бути налаштовані для виявлення надмірного використання штампів, повторів, мовних помилок, що має значення для підвищення якості представлених матеріалів.

Інтелектуальні асистенти можуть реалізовувати інтеграцію з перевіркою на плагіат, рекомендаційними системами, системами голосового введення та навіть автоматичною оцінкою наукового потенціалу

роботи на основі її структури та контенту. Це відкриває перспективи їх застосування не лише в наукових конференціях, а й у ширших академічних і дослідницьких процесах.

1.2.2 Роль інтелектуального асистента в організації наукових заходів

У контексті сучасної цифрової трансформації наукової спільноти, інтелектуальні асистенти набувають особливої ваги як елемент підтримки організаційної, комунікаційної та аналітичної інфраструктури академічних заходів. Зокрема, у середовищі конференцій, симпозіумів та форумів такі інструменти виступають не лише технічним посередником між користувачем і системою, але й дієвим агентом цифрового управління процесами, що мають бути точними, масштабованими та водночас гнучкими до людської динаміки наукового спілкування.

Перш за все, інтелектуальний асистент відіграє роль персонального інтерфейсу між учасником і організаційною платформою заходу. У середовищі наукової конференції, де подається велика кількість робіт, виникає потреба в системному доступі до інформації про вимоги до подання, терміни, форматування, умови участі, процедуру рецензування тощо. Традиційне забезпечення цього інформаційного поля, як правило, реалізується у вигляді статичних розділів на вебсайті або через електронну пошту, однак такий підхід не враховує індивідуальних запитів користувача, не дає змоги працювати в режимі реального часу та вимагає залучення додаткового персоналу для комунікаційної підтримки. Інтелектуальний асистент, навпаки, здатен автоматично відповідати на запитання, виявляти наміри користувача, уточнювати інформацію, перенаправляти до релевантних розділів ресурсу або навіть генерувати необхідні інструкції. Така комунікаційна функція дозволяє не лише зменшити навантаження на організаторів, але й формує якісно новий досвід участі для доповідачів.

Важливою функцією, яку виконує інтелектуальний асистент, є попередній аналіз і валідація поданих матеріалів. Подання наукових робіт, які не відповідають базовим вимогам до структури, форматування або стилю, призводить до затримок у розгляді, додаткових витрат часу з боку рецензентів, а іноді й до несправедливого відхилення через формальні помилки. У цьому сенсі інтеграція інтелектуального асистента, що працює на основі LLM і здатен аналізувати науковий текст, відкриває можливість автоматичного визначення наявності ключових структурних компонентів (вступу, методології, результатів, висновків), перевірки наявності анотації та ключових слів, а також оцінки загального рівня академічної якості тексту. Така автоматизована валідація не лише оптимізує процес попереднього відбору, але й підвищує об’єктивність, зменшуючи ймовірність упередженого рецензування [7].

Інтелектуальні асистенти можуть бути залучені до семантичного аналізу змісту поданих досліджень, що має критичне значення в умовах необхідності виявлення схожих або дублікатних матеріалів. У межах однієї конференції можуть подаватися дослідження, що концептуально або тематично перетинаються між собою, особливо якщо тематика є вузькоспеціалізованою або актуальною для широкого кола науковців. Використання системи, яка здатна не лише виконувати ключовий пошук, але й аналізувати глибинну семантичну подібність текстів на основі ембеддінгів та контекстуального порівняння, дозволяє ефективно ідентифікувати випадки дублювання, потенційного плагіату або невиправданої реплікації ідей. Таким чином, інтелектуальний асистент виступає запобіжником зниження наукової цінності заходу та водночас гарантом дотримання академічної доброчесності.

Роль інтелектуального асистента є також важливою у підтримці рецензентського корпусу. З огляду на обсяг подань, рецензентам необхідно швидко ознайомлюватися з численними роботами, визначати релевантність і якість кожної. Наявність попередньо згенерованих аналітичних висновків,

що містять короткий зміст, оцінку відповідності структури та потенційні схожості з іншими поданнями, суттєво полегшує цей процес. Це дозволяє зосередити увагу експертів не на формальних аспектах, а на справжньому науковому змісті, що, зрештою, підвищує якість наукової оцінки.

У перспективі, роль інтелектуального асистента може виходити за межі статичних відповідей і шаблонних перевірок. Завдяки технологіям адаптивного навчання та контекстної оптимізації, такі системи здатні змінювати свою поведінку залежно від користувацького профілю, минулих дій, тематики дослідження чи навіть рівня досвідченості учасника. Це відкриває новий вимір персоналізованої підтримки, де кожен учасник отримує релевантну допомогу відповідно до своєї ролі, будь то молодий дослідник, керівник секції чи адміністратор конференції [8].

Інтеграція інтелектуального асистента у процеси організації наукових заходів створює умови для підвищення ефективності, прозорості та наукової якості подій. Його функціональність охоплює як адміністративні, так і змістовні аспекти організації, сприяючи зниженню людського навантаження, уникненню помилок та забезпеченню динамічного й інтуїтивного середовища для всіх учасників.

1.3 Архітектура інтелектуального асистента: використання LLM та Prompt Flow Engineering

Інтелектуальні асистенти нового покоління формуються на перетині досягнень у сфері обробки природної мови, нейромережевих архітектур і сучасного програмного дизайну. Ключовими технологічними компонентами цих систем виступають великі мовні моделі та методи керованої генерації відповідей на основі підходу Prompt Flow Engineering (PFE). Саме поєднання цих технологій дозволяє реалізовувати асистентів, які не тільки генерують граматично правильні фрази, а й розуміють семантичний контекст, структуру діалогу та мету користувача.

В основі архітектури сучасного інтелектуального асистента лежить багаторівнева система, що включає декілька логічно взаємопов’язаних модулів. На рівні взаємодії з користувачем функціонує інтерфейсна оболонка − графічна або голосова, яка приймає запит і формалізує його у вигляді структурованих параметрів. Далі задіюється блок попередньої обробки, де відбувається токенізація, визначення наміру користувача, виявлення ключових слів і контекстних ознак запиту. На наступному етапі вступає в дію механізм PFE, який визначає послідовність взаємодії з LLM. У цій взаємодії формуються запити-модулі, які йдуть до LLM у вигляді промптів, а відповіді аналізуються, перетворюються та у разі потреби передаються далі по ланцюгу логіки діалогу [9].

Великі мовні моделі, такі як GPT-4, PaLM чи LLaMA, функціонують на основі трансформерної архітектури, де кожне наступне слово чи токен прогнозується з урахуванням усього попереднього контексту. Ця здатність до контекстного кодування інформації дає змогу моделям не просто завершувати речення, а й опрацьовувати запити зі складною логікою, узагальнювати інформацію, проводити аргументоване міркування або витягувати релевантні дані з великих масивів тексту. Проте потенціал LLM вимагає правильної постановки задачі, саме в цьому полягає роль PFE.

Prompt Flow Engineering − це методологія проєктування багатокрокових логічних сценаріїв взаємодії з великою мовною моделлю. Вона виникла як відповідь на обмеження традиційного «zero-shot» і «few-shot» підходів, коли одному великому запиту надсилається весь контекст, приклади, інструкції, очікування. У таких випадках моделі часто генерують узагальнені або неконкретні відповіді.

Натомість PFE пропонує розбивати задачу на менші логічні кроки та реалізовувати так звані «ланцюги промптів», структуровані послідовності звернень до LLM із уточненнями, перевірками, умовними гілками. Такий підхід дозволяє досягати більш точних результатів, керованої генерації тексту й адаптації до конкретного типу запиту [10].

Промпт-ланцюг − це серія взаємопов’язаних кроків, кожен з яких формує окремий запит до мовної моделі, отримує відповідь, інтерпретує її і залежно від результату або завершує сценарій, або ініціює новий запит. Наприклад, у випадку перевірки наукової роботи на відповідність структурним вимогам конференції, перший промпт може сформулювати запит на виявлення наявності вступу та анотації.

Якщо відповідь моделі свідчить про відсутність хоча б одного елементу, активується другий промпт, який уточнює, чи може користувач надати доопрацьований варіант. Далі запускається третій етап, а саме перевірка стилістики, четвертий − визначення відповідності тематиці тощо. Завдяки цьому ланцюгу можливе не тільки поетапне опрацювання складної задачі, а й гнучке керування логікою системи в залежності від конкретного користувача чи формату поданої інформації [11].

Ключовою перевагою PFE є можливість модульного проєктування логіки асистента. Кожен сценарій − FAQ, валідація роботи, пошук схожих матеріалів, реєстрація користувача, формується як окремий блок сценарного дизайну, що дозволяє легко масштабувати систему, змінювати окремі частини без перебудови всієї моделі та швидко оновлювати логіку відповідно до нових вимог або політик конференції. Це особливо важливо в умовах обмежених технічних ресурсів, коли повна перебудова системи є надто затратною [12], [13].

Ще одним важливим аспектом архітектури асистента є можливість зберігати проміжні відповіді, формувати звіти на основі ланцюгів промптів, а також виводити результати у зручному для користувача вигляді, у вигляді табличних звітів, списків перевірених пунктів або навіть PDF-файлів для організаторів. Це забезпечує прозорість системи та її адаптивність до адміністративної роботи.

Працюючи з LLM, архітектура асистента має зменшити обсяг переданої інформації без втрати контексту. Промпт-ланцюги в цьому випадку допомагають локалізувати запит: замість передавання всієї роботи

як єдиного тексту, система витягує окремі фрагменти (наприклад, заголовок, методологію, висновки) й обробляє їх окремо. Це не лише пришвидшує обробку, але й зменшує вартість використання моделі у хмарному середовищі [14].

Архітектура інтелектуального асистента на основі LLM та PFE поєднує в собі потужність глибокого контекстного аналізу з точністю модульного сценарного управління. Це дозволяє реалізовувати інтелектуальні системи, що працюють не лише як інструменти генерації, але й як самостійні агенти прийняття рішень у складному середовищі наукової конференції.

1.4 Моделі ШІ, що використовуються в системі

Ефективність інтелектуального асистента напряму залежить від якості обраних моделей штучного інтелекту, які реалізують ключові функціональні блоки системи. Кожна з моделей має власну архітектурну специфіку, навчальну базу, мовну орієнтацію та спектр завдань, до яких вона застосовується. У рамках розробки цифрового помічника для підтримки наукової конференції було обрано три моделі, що виконують різні ролі у загальній архітектурі: Sentence-BERT (SBERT), ChatGPT та Ukr-RoBERTa-base.

1.4.1 Sentence-BERT як інструмент семантичного порівняння текстів

SBERT є вдосконаленою версією моделі BERT, спеціально адаптованою для виконання задач, пов’язаних із порівнянням речень, визначенням семантичної подібності та кластеризацією текстових даних. На відміну від класичного BERT, який генерує контекстуальні представлення окремих слів, SBERT формує стабільні векторні ембеддінги на рівні речень або текстових блоків, що дає змогу безпосередньо вимірювати відстань між

цими представленнями у векторному просторі. Саме ця властивість робить SBERT надзвичайно ефективним для реалізації функції пошуку схожих наукових робіт на вебплатформі конференції.

Архітектурно SBERT побудовано як сіамську нейронну мережу на основі BERT або RoBERTa, що дозволяє паралельно обробляти два або більше текстів і порівнювати їх ембеддінги за допомогою косинусної подібності або евклідової відстані. Це радикально змінює логіку порівняння текстів у порівнянні з класичними підходами на основі tf-idf або простих словникових моделей. Ключова відмінність полягає у здатності SBERT враховувати глибокі контекстуальні залежності між словами, а також у створенні репрезентацій, які зберігають значущу семантичну інформацію навіть у скороченій формі [15].

У контексті організації наукової конференції SBERT використовується як модель M1, що виконує аналіз схожості між поданими роботами. Коли учасник завантажує свій текст, система SBERT створює ембеддінг для цього документа, після чого здійснюється пошук у базі вже поданих робіт за критерієм семантичної подібності. Це дозволяє автоматично виявляти не лише очевидні випадки дублювання тем, але й дослідження, які можуть бути потенційно пов’язані, хоч і подані в різній термінологічній формі. Такий підхід суттєво покращує якість первинного аналізу та допомагає рецензентам сконцентрувати увагу на унікальних дослідженнях.

SBERT демонструє оптимальний баланс між точністю та продуктивністю. У той час як класичний BERT потребує повторного запуску моделі для кожного нового порівняння, що є ресурсозатратним, SBERT дозволяє попередньо згенерувати ембеддінги для всієї бази робіт і зберігати їх у вигляді індексу, що забезпечує миттєвий пошук у режимі реального часу [16]. Це надзвичайно важливо в умовах масштабної конференції, де одночасно подається кілька сотень або тисяч документів. До того ж, SBERT добре масштабується і не потребує доступу до надпотужного

обчислювального середовища, що робить її придатною для вебінтерфейсів, розгорнутих навіть на середніх серверних потужностях.

Для українськомовного корпусу існують адаптовані версії, зокрема мультимовна версія SBERT (distiluse-base-multilingual-cased), яка підтримує українську, англійську та понад 50 інших мов. Це дає змогу використовувати SBERT не лише для порівняння англомовних матеріалів, але й у контексті україномовних подань, що критично важливо для конференцій в українських ЗВО.

Інтеграція SBERT у структуру інтелектуального асистента створює можливість реалізації високоякісного, масштабованого та об’єктивного інструменту семантичного аналізу, що підвищує прозорість і академічну доброчесність конференційного процесу. У поєднанні з іншими моделями, SBERT виконує роль «семантичного фільтра», крізь який проходять усі нові подання, що дозволяє автоматизувати виявлення дублікатів, знизити навантаження на рецензентів та забезпечити більш ефективне управління науковим контентом.

1.4.2 ChatGPT як модель для структурної валідації та генерації відповідей

У контексті створення інтелектуального асистента для підтримки наукових конференцій модель ChatGPT відіграє надзвичайно важливу роль, оскільки забезпечує не лише генерацію відповідей у діалоговому форматі, а й виконує складні завдання з аналізу структури текстів, перевірки формату та адаптації інформації відповідно до специфіки користувацьких запитів. Її ключова перевага, це здатність працювати з великими фрагментами природної мови, зберігаючи контекст, логіку і стиль, притаманний науковому середовищу.

ChatGPT є реалізацією архітектури GPT (Generative Pre-trained Transformer), яка навчається у два етапи: попереднє навчання на великих

корпусах тексту, а далі − тонке налаштування через Reinforcement Learning with Human Feedback (RLHF). Завдяки цьому модель може не лише завершувати речення або відповідати на запити, а й з високим ступенем точності визначати логічні помилки, структурні порушення або стилістичні невідповідності у складних текстах [17]. У випадку конференційного середовища така функціональність дозволяє автоматизувати перевірку наукових робіт до моменту рецензування, виключаючи недопустимі варіанти ще на етапі подання.

У межах проєкту цифрового асистента ChatGPT реалізує функцію моделі M4 − системи, яка виконує структурну валідацію тексту. При поданні наукової роботи користувачем, модель аналізує її логічну побудову: наявність вступу, чітке формулювання мети дослідження, обґрунтування актуальності, виклад методології, представлення результатів, а також висновків. У разі відсутності хоча б одного з критичних компонентів, асистент не просто повідомляє про помилку, а надає рекомендацію щодо її виправлення, дотримуючись формальних і стилістичних вимог конференції. Це суттєво підвищує якість поданих матеріалів і водночас економить час рецензентів.

ChatGPT активно використовується в режимі діалогового інтерфейсу для генерації відповідей на індивідуальні запити учасників конференції. Якщо користувач цікавиться вимогами до оформлення, дедлайнами, структурою рецензування або тематичними секціями, асистент на базі ChatGPT здатен сформулювати точну й релевантну відповідь у режимі реального часу. Завдяки глибокому контекстуальному розумінню мови та можливості підтримувати тривалий діалог, система реагує не як шаблонний бот, а як цифровий консультант із гнучким стилем ведення бесіди. Це суттєво покращує користувацький досвід та знижує навантаження на оргкомітет [18].

Окрему увагу слід приділити здатності ChatGPT адаптувати свій стиль до заданого шаблону. У науковому середовищі це означає можливість

підтримувати академічну лексику, уникати публіцистичних чи емоційних виразів, дотримуватись логічної послідовності викладу, а також уникати тавтології. Цей рівень гнучкості забезпечується завдяки глибокій попередній підготовці моделі на академічних корпусах тексту, наукових статтях, технічній документації та міждисциплінарних публікаціях.

На практиці, використання ChatGPT як ядра діалогової системи дозволяє поєднати автоматизовану перевірку, генерацію тексту, динамічну адаптацію до типу користувача та навіть прості інструменти редагування. У порівнянні з іншими LLM, ця модель демонструє високу стабільність, передбачуваність у генерації та зручну інтеграцію через API, що робить її ідеальним вибором для побудови багатофункціонального асистента в сфері науки.

1.4.3 Ukr-RoBERTa-base для обробки українськомовних запитів і FAQ

Інтеграція систем штучного інтелекту в україномовне інформаційне середовище викликає необхідність використання спеціалізованих мовних моделей, здатних точно інтерпретувати природну мову в її граматичних, семантичних і лексичних особливостях. Однією з найефективніших моделей для таких задач є Ukr-RoBERTa-base, попередньо натренована трансформерна модель, адаптована під обробку української мови. У межах проєкту інтелектуального асистента для наукової конференції ця модель використовується для розуміння запитів користувачів, зокрема в режимі частих запитань (FAQ), а також для побудови адаптивної системи допомоги на основі природної мови.

Ukr-RoBERTa-base базується на архітектурі RoBERTa (Robustly Optimized BERT Pretraining Approach), яка була вдосконалена для кращого збереження контексту в межах довших послідовностей тексту, використання більшої кількості тренувальних даних і модифікованих гіперпараметрів. Стандартна RoBERTa була навчена на корпусах

англійської мови, проте завдяки зусиллям українських дослідників і спільноти lang-uk, створено повноцінну україномовну версію, що зберігає всі технічні переваги оригінальної моделі, адаптуючи їх під морфологічну та синтаксичну структуру української мови [19].

Ukr-RoBERTa-base має 12 шарів трансформера, 12 головок уваги й 125 мільйонів параметрів. Вона була навчена на великому корпусі українських текстів, що включає Вікіпедію, новини, літературні та академічні джерела. Це забезпечує високу якість обробки навіть у складних запитах, які включають терміни, багатозначні слова або синтаксичні звороти, характерні для науково-освітнього дискурсу. У порівнянні з універсальними багатомовними моделями (такими як mBERT або XLM-RoBERTa), Ukr- RoBERTa демонструє більш точне семантичне зіставлення, кращу інтерпретацію скорочень, стилістичних маркерів та складних граматичних конструкцій [20].

У рамках реалізації інтелектуального асистента Ukr-RoBERTa використовується для розпізнавання намірів користувача у межах запитів: «Як подати роботу?», «Чи потрібна анотація?», «Де завантажити шаблон?». Модель класифікує запити за заздалегідь визначеними категоріями (оформлення, дедлайни, реєстрація, тематика тощо) і передає їх у відповідний блок сценарію (Prompt Flow). Завдяки цьому забезпечується миттєва і точна реакція системи на запити, незалежно від формулювання. Навіть запити типу «я не можу знайти дедлайн» або «до якого числа ще можна подати» будуть інтерпретовані як звернення до блоку «терміни подання».

Модель може використовуватись як інструмент семантичної фільтрації у базі знань конференції: на основі україномовного запиту система може витягати найбільш релевантні фрагменти інструкцій, регламентів, технічних вимог, навіть якщо у запиті вжито неточні або неповні формулювання. Це суттєво підвищує зручність користування та

зменшує когнітивне навантаження на учасників, особливо новачків, які лише ознайомлюються з процедурою.

Завдяки відкритому коду та підтримці з боку української дослідницької спільноти, Ukr-RoBERTa-base може бути донавчена або адаптована під потреби конкретної конференції, наприклад, шляхом додавання специфічної термінології, абревіатур або лексичних форм, які часто зустрічаються в поданих матеріалах. Це дозволяє отримати гібридну модель, оптимізовану не лише під українську мову, а й під конкретну предметну галузь.

1.5 Переваги впровадження асистента у платформу Міжнародного молодіжного форуму

Упровадження інтелектуального асистента на базі великих мовних моделей та методології Prompt Flow Engineering в організаційну платформу Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» є не просто модернізацією цифрової інфраструктури, це стратегічний крок до побудови інноваційного, ефективного й людиноцентричного середовища підтримки наукового потенціалу. Такий асистент виступає не як допоміжний інструмент, а як інтелектуальний партнер як для організаторів, так і для учасників форуму.

Однією з найочевидніших переваг є значне зростання швидкості обробки інформації. У звичайному сценарії подання та попереднього перегляду наукових матеріалів вимагає ручної перевірки десятків критеріїв: відповідність структурі, стиль оформлення, дотримання регламентів, відповідність тематиці, наявність анотацій, ключових слів тощо. Автоматизований асистент здатен здійснювати ці перевірки за лічені секунди, надаючи зворотний зв’язок у режимі реального часу. Це зменшує кількість помилкових або неповних подань, а також скорочує навантаження на технічний персонал і рецензентів. При цьому швидкість не погіршує

якість, завдяки великим мовним моделям оцінка відбувається не формально, а з урахуванням семантичного й контекстуального аналізу.

Іншою важливою перевагою є об’єктивність і послідовність у перевірці матеріалів. Людський фактор, що неминуче присутній у процесі оцінювання, часто призводить до суб’єктивного підходу: один і той самий недолік у роботі може бути проігнорований одним рецензентом і жорстко оцінений іншим. Автоматизована перевірка на основі єдиної логіки усуває цей ризик, забезпечуючи стандартизовану оцінку всіх подань. Крім того, такий підхід дозволяє зафіксувати критерії, за якими здійснюється аналіз, що є основою для прозорості процесу рецензування. Асистент може не лише відхилити роботу, а й обґрунтувати причину на рівні логіки структури, відсутності обов’язкових розділів або тематичної невідповідності.

Підвищення унікальності та релевантності представлених матеріалів. Завдяки інтеграції моделі SBERT, система здійснює попередню перевірку схожості поданої роботи з уже наявними в базі, що дає змогу уникати дублікатів, компіляцій та тематичних повторів. Це не лише зменшує кількість однотипних виступів, а й стимулює учасників готувати справді нові дослідження, фокусуючись на власному науковому внеску. Виявлення плагіату або паралелізму змісту на основі семантичної подібності є важливим кроком до підтримання академічної доброчесності.

У традиційній моделі зв’язку користувачі змушені шукати інформацію самостійно, перечитуючи розділи сайту або звертаючись до організаторів електронною поштою. В умовах великого обсягу комунікацій це призводить до втрати часу, непорозумінь і невдоволення. Натомість інтелектуальний асистент на базі ChatGPT та Ukr-RoBERTa забезпечує природну, інтуїтивну взаємодію: учасник може задати запитання у вільній формі українською мовою, і миттєво отримати точну, граматично правильну і стилістично адаптовану відповідь. Це створює ефект «живого» консультанта, але доступного 24/7, без очікування черги або перенавантаження.

Крім функції FAQ, асистент може виконувати роль гіда по конференції, допомагаючи учаснику зорієнтуватися в секціях, знайти шаблон оформлення, згенерувати резюме роботи, сформулювати тему чи навіть перевірити текст на наявність помилок або стилістичних недоліків. В умовах, коли значна частина учасників є студентами або молодими дослідниками без великого досвіду, така підтримка критично важлива. Вона не лише підвищує якість поданих матеріалів, а й формує навчальний компонент: користувач отримує поради, які може використати для покращення власних навичок наукового письма.

З погляду організаційної ефективності, асистент також є інструментом глибокої аналітики. Усі звернення, запити, типові помилки, рівень відповідності шаблонам, усе це може бути зафіксовано й узагальнено для подальшого аналізу. Це дає змогу оргкомітету коригувати інформаційну політику, оновлювати інструкції, покращувати структуру сайту або секцій, виходячи з реальних потреб користувачів. У довготривалій перспективі це формує адаптивну, самонавчальну систему управління форумом, де рішення базуються не на припущеннях, а на даних.

Упровадження інтелектуального асистента в платформу Міжнародного молодіжного форуму відкриває нову парадигму організації академічних заходів. Вона поєднує швидкість обробки, об’єктивність перевірки, унікальність наукового контенту та персоналізовану підтримку кожного учасника. У світі, де час і точність мають вирішальне значення, така система формує конкурентну перевагу не лише окремого заходу, а й національного академічного середовища загалом.

2 ПОСТАНОВКА ЗАДАЧІ І ЗАГАЛЬНА КОНЦЕПЦІЯ РЕАЛІЗАЦІЇ АСИСТЕНТА

2.1 Мета та постановка задачі

Активне впровадження інтелектуальних технологій у сферу академічної комунікації, зокрема в процеси організації та підтримки наукових заходів, пов’язано з постійним зростанням обсягів поданих матеріалів, розширенням кількості учасників, зростанням вимог до академічної доброчесності та ефективності внутрішніх процесів. У цьому контексті особливої актуальності набувають цифрові рішення, здатні не лише обробляти інформацію, а й виконувати інтелектуальні функції з аналізу, порівняння, структуризації та персоналізованої взаємодії. Саме тому розробка й апробація інтелектуального асистента для автоматизованої підтримки учасників та організаторів наукового форуму становить основну дослідницьку мету даної практичної роботи.

Метою дослідження є створення ефективного та функціонального асистента, заснованого на технологіях великих мовних моделей та методології Prompt Flow Engineering, який дозволяє реалізувати гнучкі багатокрокові сценарії взаємодії з користувачем у форматі природної мови. Такий асистент повинен забезпечити оперативну обробку наукових матеріалів, семантичний аналіз тексту, перевірку відповідності вимогам конференції, виявлення схожих робіт, а також надання відповідей на типові запити учасників форуму в режимі реального часу. Йдеться не лише про технічну перевірку поданих документів, а про повноцінну інтелектуальну підтримку академічного процесу, що охоплює комунікацію, оцінювання, навігацію та зворотний зв’язок.

Одним із ключових акцентів дослідження є адаптація системи до україномовного середовища. В умовах, коли більшість LLM орієнтовані на англомовні корпуси, забезпечення високоякісної обробки української мови

є особливо актуальним для вітчизняних наукових заходів. Використання моделі Ukr-RoBERTa-base у поєднанні з іншими мовними моделями дозволяє досягти точного розуміння запитів користувачів, обробки текстів заявок та надання релевантних відповідей, зберігаючи при цьому академічний стиль та правильну граматику. Мета дослідження охоплює не лише технічну реалізацію, а й лінгвістичну локалізацію інтелектуального рішення, що має забезпечити зручність, доступність та коректність використання асистента в українському академічному просторі.

Дослідження також орієнтується на забезпечення масштабованості та адаптивності створеної системи. Передбачається, що запропонована архітектура інтелектуального асистента буде гнучкою до змін: оновлення регламентів, зміна типових запитів, зростання кількості користувачів або розширення функціоналу не вимагатимуть повної перебудови системи, а можуть бути реалізовані через оновлення логіки промптів та додавання нових модулів. Це дозволить у майбутньому застосовувати рішення не лише в межах одного форуму, а й адаптувати його до інших наукових заходів, як регіонального, так і міжнародного масштабу.

Побудована система має також інтеграційний потенціал, можливість з’єднання з базами даних поданих робіт, інтерфейсами адміністрування, системами рецензування, а також зовнішніми сервісами (перевірки плагіату або автоматичної оцінки стилістики текстів). Це відкриває перспективу створення єдиного цифрового середовища управління конференцією, у межах якого інтелектуальний асистент стане центральною ланкою між користувачами та даними.

Мета передбачає як програмну реалізацію системи, так і її концептуальне осмислення в контексті цифровізації науки, інтерфейсного дизайну та лінгвістичної доступності. Реалізація поставленої мети дозволить отримати ефективний, масштабований, адаптивний інструмент академічної підтримки, що відповідає актуальним викликам цифрового наукового простору.

Для досягнення окресленої мети передбачається реалізувати низку взаємопов’язаних завдань, що охоплюють аналітичну, проєктну та експериментальну частини дослідження. Насамперед необхідно було здійснити теоретичний аналіз сучасних підходів до побудови інтелектуальних асистентів у сфері наукових конференцій. Це включало вивчення архітектурних рішень, методів застосування великих мовних моделей (LLM), механізмів генерації відповідей, а також алгоритмів обробки природної мови у багатомовному середовищі. Особливу увагу приділити підходам, що використовують Prompt Flow Engineering як засіб побудови керованих сценаріїв взаємодії з мовними моделями.

Наступний етап − проєктування архітектурної моделі інтелектуального асистента, яка б відповідала вимогам модульності, масштабованості та інтеграційної відкритості. У межах цієї моделі виокремити функціональні блоки для семантичного аналізу, генерації відповідей, перевірки структури текстів і підтримки україномовного інтерфейсу. На основі цього архітектурного бачення розробити конкретні технічні реалізації кожного з модулів.

Одне із ключових завдань − розробка модуля для семантичного порівняння наукових робіт із метою виявлення схожих подань та запобігання дублюванню. Цей модуль необхідно реалізувати з використанням моделі Sentence-BERT, яка дає змогу формувати векторні репрезентації текстів і здійснювати їх порівняння на основі косинусної подібності. Паралельно впровадження механізму для перевірки структурної відповідності поданих робіт вимогам конференції. Такий механізм мав функціонувати на основі можливостей ChatGPT як інструмента для обробки складних текстових структур і генерації діалогових відповідей.

Створення україномовного модуля підтримки користувачів, що функціонує на базі моделі Ukr-RoBERTa-base. Він повинен забезпечити інтерпретацію запитів природною українською мовою, а також відповідати за точність класифікації питань у межах системи частих запитань (FAQ). На

цьому етапі також передбачається налаштування логіки багатокрокової взаємодії користувача з асистентом шляхом формування промпт-ланцюгів і організації сценаріїв за допомогою технології Prompt Flow.

Окреме завдання − формування тестового середовища, у межах якого планувалося перевірити функціонування усіх модулів асистента на прикладах реальних або змодельованих подань до конференції. У процесі тестування − моделювання користувацьких сценаріїв, включно із запитами, поданням робіт, виявленням типових помилок і оцінкою релевантності відповідей. Після завершення випробувань необхідно провести аналітичне узагальнення результатів функціонування системи за ключовими критеріями ефективності: швидкістю обробки запитів, точністю виявлення структурних і семантичних відхилень, рівнем унікальності поданих матеріалів, а також загальною зручністю взаємодії з інтерфейсом асистента з позиції користувача.

Поставлені завдання охоплюють повний цикл дослідницької діяльності: від аналізу предметної галузі до практичної реалізації й тестування, що дозволяє розглядати інтелектуального асистента не як ізольований програмний модуль, а як елемент цілісної цифрової системи академічного середовища.

2.2 Формалізація функціональних компонентів асистента

Проєктування інтелектуального асистента для підтримки процесів наукової конференції передбачає чітке розмежування його функціональних компонентів, кожен з яких реалізує специфічну підзадачу в межах загальної архітектури. Така формалізація дозволяє забезпечити як внутрішню узгодженість системи, так і гнучкість її налаштування, масштабування та подальшої адаптації. У межах запропонованого підходу асистент умовно поділено на чотири основні функціональні модулі, що відповідають за ключові напрями взаємодії з користувачем: семантичний пошук тез (M1),

аналіз українськомовних запитів (М2), оцінка рівня унікальності текстів (М3) та генерація відповідей, включно з перевіркою структури поданих матеріалів (М4).

Першим функціональним компонентом, що розглядається як модель М1, є семантичний пошук – аналіз поданих робіт з метою виявлення їх схожості, який виконується за допомогою Sentence-BERT. Ця модель дозволяє переводити текстові блоки в компактні векторні представлення, які зберігають семантичну інформацію, і які можна порівнювати між собою за допомогою косинусної подібності. Формалізовано, кожна робота, подана користувачем, повинна бути сегментована на ключові елементи (назва, анотація, вступ, висновки), які окремо подаються на обробку SBERT. У процесі генерації ембеддінгів здійснюється їх зіставлення з уже наявними в базі робіт форуму. При виявленні високого ступеня подібності система повертає попередження для адміністратора або рецензента, дозволяючи на ранньому етапі виявити можливі дублікати або компіляції. Такий підхід дозволяє автоматизувати первинне фільтрування подань та уникнути формального перегляду великої кількості повторюваних матеріалів. При цьому SBERT не просто порівнює слова, вона працює на рівні змісту, що є ключовим у міждисциплінарному середовищі.

Наступним модулем, М2, є аналіз україномовних запитів у форматі FAQ і підтримки, реалізований на основі моделі Ukr-RoBERTa-base. Цей компонент відповідає за обробку природної мови користувачів, що звертаються до асистента з інформаційними, процедурними або технічними питаннями. Модель навчена на корпусі україномовних текстів, що дозволяє їй точно інтерпретувати запити навіть у випадках неформальних, фрагментарних або стилістично складних формулювань. На рівні формалізації, цей модуль виконує семантичне порівняння запиту користувача з базою типових питань, класифікує його за категоріями (реєстрація, оформлення, дедлайни, техпідтримка) і передає до відповідного блоку сценарію. Ukr-RoBERTa-base забезпечує мовну

інтерфейсну гнучкість асистента, його адаптивність до українськомовного контексту й можливість масштабного розширення бази знань без необхідності модифікації коду.

Третій функціональний блок, М3, пов’язаний із перевіркою рівня унікальності та доброчесності поданих текстів. Тут можливі два сценарії реалізації: або через інтеграцію з API зовнішньої антиплагіатної системи типу Turnitin, або через застосування локального класифікатора або детектора AI-контенту (OpenAI AI Detector, GPTZero або інші моделі перевірки на ймовірність машинного генерування). У контексті формалізації проєкту цей модуль відповідає за попереднє оцінювання ступеня схожості роботи з загальнодоступними джерелами, а також за ймовірність того, що поданий текст було згенеровано штучним інтелектом без суттєвого редагування. Результати перевірки передаються в модуль звітування або рецензування як попередній сигнал для модерації. Незалежно від обраної реалізації, цей компонент формує основу для підтримки академічної доброчесності на ранньому етапі, забезпечуючи конфіденційність і захист від недоброчесного копіювання.

Четвертий модуль, М4, виконує генерацію відповідей, структурну перевірку та діалогову взаємодію, і реалізується на основі моделі ChatGPT. Він є центральним елементом комунікаційного інтерфейсу асистента, що відповідає за логіку реакції системи на складні або відкриті запити. Формально, модель приймає на вхід як окремі фрази, так і фрагменти наукових робіт, аналізує їхню структуру (наявність вступу, цілей, методології, висновків тощо) і повертає або автоматичну оцінку, або рекомендацію щодо доопрацювання. ChatGPT здатен проводити діалог на декілька кроків, підтримуючи контекст і логіку розмови, а також варіативно реагуючи на уточнення з боку користувача. Це дозволяє реалізувати не просто довідкову систему, а повноцінний консультаційний модуль з високим рівнем адаптивності.

Сукупність зазначених модулів забезпечує високий ступінь функціональної завершеності асистента. Кожна модель виконує свою чітко визначену роль, при цьому їх взаємодія дозволяє формувати єдину логіку користувацького досвіду: від моменту подання матеріалу, до отримання зворотного зв’язку або інтеграції в рецензійний процес. Подібна багатокомпонентна структура дозволяє не лише масштабувати систему під інші формати заходів, а й адаптувати її під зміну політик, розширення бази знань або оновлення вимог до академічного оцінювання.

2.3 Типи користувачів та сценарії використання асистента на сайті конференції

Функціональність інтелектуального асистента має бути спроєктована не лише з урахуванням технічних характеристик модулів, а й відповідно до реальних ролей та потреб користувачів, які взаємодіють із платформою наукової конференції. Передбачається, що система підтримуватиме роботу щонайменше трьох основних типів користувачів: учасника (автора наукової роботи), організатора (члена технічного або програмного комітету) та рецензента (експерта, що оцінює подані матеріали). Для кожної з цих категорій передбачається специфічний набір функцій, що активується в межах відповідного сценарію взаємодії.

З точки зору учасника конференції, інтелектуальний асистент виконує роль цифрового помічника, який супроводжує процес подання роботи від моменту ознайомлення з умовами участі до фінального збереження заявки в базі. Одразу після авторизації на сайті користувач має змогу звертатися до асистента з інформаційними питаннями. У цьому контексті активується модуль M2 (Ukr-RoBERTa), який відповідає за розпізнавання запиту, визначення його категорії та передавання в логіку Prompt Flow. Асистент повертає релевантну відповідь, з урахуванням стилістичних та мовних норм.

У процесі завантаження наукової роботи система автоматично запускає ланцюг перевірок, які активують модулі М1, М3 та М4. Передусім, модуль М4 (ChatGPT) аналізує структуру документа: визначає наявність обов'язкових компонентів (вступ, мета, методологія, результати, висновки), надає поради щодо покращення формулювань або виявляє недоліки в логіці викладу. У разі серйозних порушень структури асистент пропонує учаснику виправити роботу до остаточного подання. Після цього активується М1 − модуль SBERT, який проводить семантичне порівняння із наявними роботами в базі й повідомляє користувача про наявність подібних тем. Якщо збіг високий, система рекомендує уточнити формулювання теми або переформатувати підхід до дослідження. Такий сценарій забезпечує активну участь асистента у формуванні якісного, унікального академічного контенту ще до того, як робота буде розглянута рецензентом.

З боку організатора конференції (секретаря, координатора секції або технічного адміністратора), асистент відіграє переважно функцію автоматизованого фільтра та попереднього аудитора. Після подання великої кількості заявок організатор отримує доступ до модуля перевірки структури, в якому зібрані зведені дані, згенеровані ChatGPT: наявність структурних елементів, відповідність шаблону, загальна оцінка якості тексту. Кожен аналіз супроводжується зведенням, яке дозволяє адміністратору оперативно переглядати проблемні місця, не читаючи повністю кожен документ. Також організатор має змогу формувати вибірки робіт за рівнем унікальності, відповідно до показників, наданих модулем М3. Завдяки цьому можлива швидка модерація подань, виявлення потенційно проблемних текстів та розподіл їх між рецензентами або повернення авторам на доопрацювання.

Усі дії, які здійснює асистент у взаємодії з роботою, фіксуються у внутрішньому журналі − лог-файлі, доступному організатору. Це дозволяє контролювати всі процеси, зберігати прозорість та відтворюваність попередньої оцінки, а також адаптувати логіку промптів у разі потреби.

Наприклад, у разі надмірної кількості запитів на одну й ту саму тему, адміністратор може оперативно додати нове FAQ-запитання до сценарію.

З погляду рецензента, інтелектуальний асистент виступає не як замінник аналітичної роботи експерта, а як попередній фільтр і джерело зведеної інформації про кожну подану роботу. Передусім, рецензент має доступ до результатів аналізу схожості, які надає SBERT. Це дозволяє одразу побачити, чи є схожі роботи, які вже були подані, і порівняти їх між собою. Додатково рецензент отримує коротке автоматично згенероване резюме, що формується ChatGPT за логікою Prompt Flow: визначаються основні тези, наукова новизна, застосована методологія та узагальнення висновків.

Такий формат дозволяє рецензенту швидко зорієнтуватися в змісті роботи, не втрачаючи час на технічні чи стилістичні недоліки. За потреби рецензент може скористатися асистентом у режимі діалогу: поставити уточнююче питання щодо структури, попросити згенерувати перелік можливих питань до автора або вказати на суперечності у формулюваннях. У перспективі така співпраця створює гібридну систему оцінювання, де автоматизований аналіз доповнює професійне експертне судження.

Розробка сценаріїв використання асистента враховує не лише функціональну диференціацію користувачів, а й контекст їхньої взаємодії з платформою. Учасники отримують динамічну підтримку на етапі підготовки та подання роботи. Організатори − інструменти для швидкої модерації, зведеної перевірки й логістики подань. Рецензенти отримують інтелектуальний фільтр та аналітичну підтримку в оцінюванні.

2.4 Очікувані результати та вигоди впровадження

Інтеграція інтелектуального асистента у платформу наукового форуму, як очікується, матиме значний позитивний вплив як на внутрішні процеси організації заходу, так і на досвід взаємодії з платформою для всіх

категорій користувачів. Запропонована система, що поєднує можливості великих мовних моделей, модульної логіки та сценаріїв Prompt Flow Engineering, має потенціал не лише підвищити технічну ефективність, а й сприяти формуванню нового рівня академічної якості, прозорості та зручності участі у науковому заході.

Одним із ключових очікуваних результатів є суттєве скорочення часу, необхідного для перевірки поданих наукових робіт. У традиційному форматі ця процедура вимагає ручної перевірки кожного подання на відповідність формальним вимогам: наявність структури, відповідність обсягу, правильне оформлення анотації та ключових слів, логіка викладу тощо. Застосування інтелектуального асистента дозволяє автоматизувати цей процес, оскільки модуль ChatGPT здатен миттєво аналізувати структуру поданої роботи, виявляти відсутні компоненти або порушення формату. Завдяки цьому роботи, які не відповідають базовим критеріям, можуть бути автоматично повернуті авторам на доопрацювання ще до моменту рецензування. Така перевірка триває не більше кількох секунд, що дозволяє оптимізувати навантаження на програмний комітет та рецензентів, особливо при великій кількості подань.

Другою важливою перевагою є підвищення якості оцінювання поданих матеріалів. Рецензент, отримуючи на руки десятки робіт, часто змушений фокусуватися на формальних критеріях та не має змоги глибоко аналізувати кожен текст. Система інтелектуального асистента забезпечує попередню обробку матеріалів: здійснює стислий контекстуальний огляд, витягує основні тези, проводить семантичну класифікацію теми, а також оцінює ступінь структурованості документа. Такий підхід дозволяє рецензенту зосередитися на суті наукового внеску, а не витрачати час на пошук базових елементів тексту або виправлення форматних помилок. Оцінювання набуває більш фахового й об’єктивного характеру, а організатори можуть бути впевнені, що усі подані роботи проходять початковий рівень автоматичної експертизи.

Одним із пріоритетних завдань форуму є забезпечення унікальності представленого наукового контенту. З розвитком відкритих джерел, публікацій у вільному доступі та широким використанням штучного інтелекту для генерації текстів, виникає реальна загроза дублювання тематики, повторення висновків або навмисного копіювання чужих матеріалів. Упровадження інтелектуального асистента дозволяє запобігати таким ситуаціям на етапі подання. Модуль SBERT виконує порівняння кожної нової роботи з уже наявними в базі форуму, виявляючи високий ступінь семантичної подібності навіть у випадках переформульованих або змінених фрагментів.

Окремо варто наголосити на вигодах для самих учасників, зокрема для студентів, аспірантів та молодих науковців, які лише починають свій шлях у дослідницькій діяльності. Завдяки функціоналу асистента учасник отримує постійну підтримку, у вигляді швидких відповідей на запитання щодо процедури подання, пояснень до правил оформлення, порад щодо структури роботи тощо. Взаємодія з асистентом відбувається в зручному діалоговому форматі, що зменшує стрес і невизначеність, особливо в умовах першого досвіду участі у форумі. Крім того, асистент може супроводжувати процес редагування: підказувати формулювання, виявляти стилістичні помилки, надавати шаблонні конструкції для академічного тексту. Це створює середовище м’якого навчання, де користувач покращує власну наукову грамотність у процесі взаємодії з системою.

Із боку організаторів та адміністраторів очікується значне спрощення логістики обробки заявок. Завдяки наявності логів взаємодії, звітів перевірки структури, оцінок унікальності та журналів запитів, організатори зможуть проводити аналітику, планувати робоче навантаження, адаптувати регламент та актуалізовувати контент платформи. Система надає гнучкий інструментарій для масштабування: у разі збільшення кількості учасників або зміни у політиках конференції достатньо змінити логіку промптів або додати нові модулі, не змінюючи архітектуру цілком. Це дає змогу

перетворити форум із класичного заходу на адаптивну, цифрово-керовану подію, що відповідає вимогам XXI століття.

Таким чином, очікувані результати впровадження інтелектуального асистента можна умовно поділити на чотири рівні: технічний (швидкість і стабільність перевірок), академічний (підвищення якості матеріалів), організаційний (оптимізація рецензійних та адміністративних процесів) і людський (підтримка й навчання учасників). Усі ці рівні об’єднані спільною логікою: створити ефективне, надійне, зручне й доброчесне цифрове середовище для наукової комунікації, в якому штучний інтелект виступає помічником, а не замінником людини. У перспективі така система здатна не лише трансформувати локальний форум, а й стати універсальним рішенням для академічних подій нового покоління.

3 ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНТЕЛЕКТУАЛЬНОГО ПОМІЧНИКА ДЛЯ НАУКОВИХ КОНФЕРЕНЦІЙ

3.1 Технічне завдання на розробку

Завданням проєкту є створення інтелектуального вебсервісу, призначеного для автоматизованої роботи з науковими статтями, що подаються авторами на конференцію. Система покликана забезпечити ряд функцій, що мають на меті оптимізувати процес рецензування, покращити якість контенту та розвантажити організаторів. Основними функціями системи є пошук аналогічних робіт, відповіді на поширені питання, перевірка структури згідно зі стандартами, виявлення плагіату та оцінка ймовірності використання ШІ для написання тексту.

У системі планується використовувати поєднання сучасних технологій обробки природної мови, зокрема, великих мовних моделей, попередньо навчених трансформерів для української мови, а також методів семантичного порівняння текстів. Програмне забезпечення реалізується як вебдодаток із REST API для обробки запитів та зрозумілим інтерфейсом користувача, який дає змогу працювати з сервісом без необхідності спеціальної технічної підготовки.

До функціональних вимог належать:

– виконання семантичного пошуку для знаходження схожих робіт з використанням моделі Sentence-BERT;

– визначення ступеня схожості між новою статтею та базою попередніх матеріалів конференцій;

– обробка найчастіших запитань учасників (FAQ) з допомогою моделі Ukr-RoBERTa;

– перевірка наявності обов’язкових структурних елементів (назва, автори, мета, висновки тощо) відповідно до вимог ДСТУ 8302:2015, з використанням GPT API;

– аналіз коректності оформлення бібліографічних посилань на основі регулярних виразів та мовної моделі;

– виявлення рівня ймовірності створення тексту за допомогою штучного інтелекту;

– вилучення та зчитування тексту з PDF-файлів, завантажених користувачами;

– підтримка роботи з інтерфейсом на базі HTML і JavaScript для завантаження файлів та надсилання текстових запитів.

Нефункціональні вимоги в свою чергу включають наступні характеристики:

– підтримка обробки PDF-документів розміром до 5 мегабайт;

– час відповіді сервера на один запит не повинен перевищувати 5 секунд при типовому навантаженні;

– повна підтримка української мови в усіх модулях системи;

– сумісність клієнтського інтерфейсу із зальновживаними браузерами Google Chrome, Mozilla Firefox і Microsoft Edge;

– можливість локального розгортання системи в межах інфраструктури організаторів конференції.

Для розробки проєкту вибрано мову програмування Python. Основою серверної частини буде фреймворк FastAPI, що дозволяє реалізувати асинхронну обробку HTTP-запитів. Для мовних моделей буде використано бібліотеки Huggingface Transformers та SentenceTransformers. Обробка PDF-файлів здійснюватиметься з допомогою PyMuPDF. Інтерфейс користувача побудований на HTML, CSS та JavaScript без сторонніх фреймворків.

В результаті розробки передбачається створення стабільної системи, здатної витримувати навантаження та забезпечувати перевірку наукових текстів за вищенаведеними критеріями та виконувати поставлені функції. Сервіс допоможе скоротити витрати часу на ручне рецензування, виявити дублювання тематики, а також поліпшити відповідність вимогам до структури та оформлення.

3.2 Архітектура та структура проєкту

Архітектура інтелектуального помічника для конференцій базується на модульному розподілі функцій, що сприяє створенню стійкого, масштабованого і зрозумілого рішення. Кожен компонент системи відповідає за певну функціональність, проте вони взаємодіють між собою у межах єдиного логічного середовища, що полегшує супровід коду, спрощує майбутнє розширення функціоналу та дозволяє ізолювати можливі помилки.

Загалом, система складається з трьох основних шарів – шар взаємодії з користувачем, шар обробки даних та аналітики, а також шар управління ресурсами і допоміжної логіки. На першому шарі реалізовано механізм отримання запитів, як у форматі файлів, так і текстових повідомлень. Головним чином, це дозволяє охопити різноманітні сценарії взаємодії від перевірки структури наукових документів до надання консультативної підтримки.

Середній шар відіграє найважливішу роль з погляду функціональності. В ньому відбувається попередня обробка інформації, розподіл запитів за типами, запуск відповідних алгоритмів і формування відповідей. Також шар відповідає за перевірку структурних вимог, виконання семантичного пошуку, генерацію відповідей на часті запитання, а також аналіз схожості текстів і виявлення ознак використання генеративних мовних моделей. Кожна з цих задач виконується окремим підкомпонентом, який працює ізольовано, але взаємодіє з іншими частинами через чітко визначені інтерфейси.

Допоміжний шар забезпечує технічне функціонування сервісу. До нього входять інструменти для обробки PDF-документів, ведення журналів подій, підготовки запитів до зовнішніх сервісів, роботи з форматами даних, збереження результатів. Елементи не впливають безпосередньо на

прийняття рішень, але є особливо важливими для забезпечення цілісності процесу.

Узагальнену архітектурну модель системи показано на рисунку 3.1. У ній зображено основні функціональні блоки та напрямки обміну даними між ними. Проєкт розроблено з врахуванням можливості подальшого розвитку. Використання модульного підходу дає змогу без проблем інтегрувати нові сервіси, змінювати моделі або розширювати джерела вхідної інформації без необхідності змін у загальній архітектурі. Кожен компонент проєкту легко ізолюється, тестується та оновлюється окремо, що є надзвичайно важливим у випадку, коли система має адаптуватися до нових вимог конференцій, змін правил оформлення або появи нових джерел плагіату.

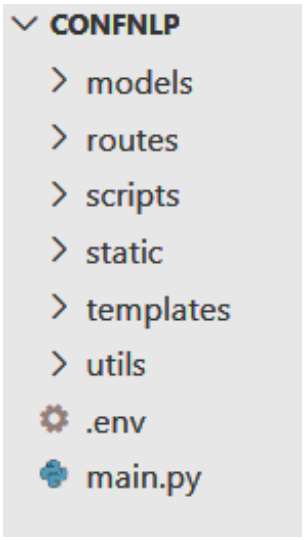

Рисунок 3.1 – Структура модулів проєкту

Інтерфейс містить в собі мінімум елементів, які розділені на логічні зони. Користувач безпосередньо не взаємодіє з технічною частиною, а лише надсилає файл або ставить питання. Подальша обробка відбувається на сервері, який забезпечує повний цикл аналізу без втручання людини.

Очевидно, що в структурі також передбачено підтримку української мови на всіх рівнях. Мовні моделі, які використовуються, попередньо

навчаються на україномовних моделях, що забезпечує високу точність при роботі з текстами, характерними саме для наукових конференцій та дозволяє обробляти специфічні формулювання, скорочення та терміни.

3.3 Опис базових сервісів системи

3.3.1 Семантичний пошук

Семантичний пошук – це ключовий функціонал системи, без якого неможливо здійснити інтелектуальний аналіз подібності наукових текстів. Його використовують не лише для виявлення плагіату чи дублювання робіт, а й для групування доповідей за темами, оцінки релевантності матеріалу для конкретної секції, а також створення внутрішньої контент-аналітики конференції. Головна мета цього механізму – не знаходження формального збігу слів, а глибоке розпізнавання смислової відповідності між текстовими фрагментами.

Для реалізації пошуку застосовується модель Sentence-BERT, що дає змогу кодувати речення у багатовимірні вектори з урахуванням їх семантики, забезпечуючи визначення подібності між реченнями, навіть якщо їх лексична форма значно відрізняється. Замість порівняння текстів як послідовностей символів, система працює з числовими представленнями, котрі формують геометричну модель змісту. Відтак, два тексти, що різняться за формулюванням, можуть бути розпізнані як схожі, якщо їхня тематика чи основна ідея співпадають.

Архітектурно реалізація поділена на два основні етапи: індексація корпусу та обробка запиту. На першому етапі система читає всі наявні наукові документи з бази даних, попередньо підготовленої у форматі JSON. Кожен документ містить текст та супровідні метадані. Після зчитування всі тексти кодуються у векторні представлення один раз і зберігаються в оперативній пам’яті (лістинг 3.1).

Лістинг 3.1 – Завантаження корпусу наукових робіт

```
def _load_papers(self) -> List[Dict]:
    if not os.path.exists(self.data_path):
        raise FileNotFoundError("Файл бази не знайдено.")
    with open(self.data_path, "r", encoding="utf-8") as f:
        try:
            return json.load(f)
        except json.JSONDecodeError:
            raise ValueError("Формат JSON некоректний.")
```

Основна перевага методології полягає в уникненні повторних обчислень ембедингів для корпусу при кожному запиті. Генерація векторів – процес, що вимагає значних ресурсів, тому оптимізація шляхом кешування та попереднього кодування значно підвищує продуктивність і здатність системи до масштабування (лістинг 3.2).

Лістинг 3.2 – Побудова векторних представлень для корпусу

```
    def _encode_all_papers(self) -> np.ndarray:
        texts = [paper.get("text", "") for paper in
self.papers]
        embeddings = self.model.encode(
            texts,
            convert_to_numpy=True,
            normalize_embeddings=True
        )
        return embeddings
```

Наступний крок – це обробка запиту, який виконується в реальному часі. Користувач надсилає текст, який кодується тією ж моделлю, а вже далі здійснюється порівняння цього вектора з усіма векторами корпусу за допомогою косинусної метрики. Головний критерій: чим менший кут між векторами, тим вищий рівень подібності (лістинг 3.3).

Лістинг 3.3 – Пошук за семантичним подібністю

```
    def search_semantic(self, query_text: str, top_k: int = 5):
        query_vec = self.model.encode([query_text],
convert_to_numpy=True)
        scores = cosine_similarity(query_vec,
self.paper_embeddings)[0]
        sorted_idx = scores.argsort()[::-1][:top_k]
        return [{"title": self.papers[i].get("title"),
                "author": self.papers[i].get("author"),
                "similarity": round(float(scores[i]), 4)
            }for i in sorted_idx]
```

Семантичне порівняння не обмежується лише загальною подібністю. У системі передбачено додаткові фільтри для звуження області пошуку: за секцією, кодом УДК або конкретною конференцією, що дозволяє, наприклад, шукати лише серед подань певного року або тільки серед робіт з близькою тематикою.

Пошукова система в цій розробці – це значно більше, ніж просто інструмент для знаходження схожих документів. Вона є основою для подальшої роботи: створення звітів, організації матеріалів за тематикою, визначення тематичних повторів. У майбутньому планується розширення функціоналу, наприклад, додавання карт кластеризації документів, автоматичне присвоєння секцій та генерування статистики з популярних напрямів наукових досліджень.

Окремо слід відзначити, що ця система працює автономно, без потреби у зовнішніх сервісах, що є важливим для забезпечення безпеки даних користувачів. Ефективність роботи досягається завдяки використанню векторної індексації та оптимізованого кодування.

Підсумовуючи, модуль семантичного пошуку, реалізований на базі SBERT у даній системі, відіграє не тільки прикладну, але й аналітичну роль. Він дає можливість створити екосистему сервісів, що в цілому покращують якість проведення конференцій – як у технічному, так і у змістовному плані.

3.3.2 Блок FAQ

Компонент для пошуку відповідей на поширені запитання (FAQ) у системі втілений як індексатор документів. Його функція полягає у створенні бази даних потенційних відповідей на основі сирих текстових даних (методичних рекомендацій, інструкцій, інформаційних листів у форматі PDF). Користувач формулює запит у вільній формі, а система намагається відшукати найбільш відповідне речення серед інформації, що міститься у наданих файлах.

На відміну від звичного FAQ, де питання та відповіді створюються вручну, ця реалізація використовує обробку повнотекстових документів. Кожен файл, що знаходиться у власній директорії, відкривається, далі з нього витягується текст, після чого здійснюється його сегментація на речення (лістинг 3.4). Кожне речення сприймається як окремий кандидат на потенційну відповідь.

Лістинг 3.4 – Розбиття тексту організаційного характеру на речення та ембеддінги

```
    def _prepare_sentences(self):
        for filename in os.listdir(self.data_dir):
            if not filename.endswith(".pdf"): continue
            path = os.path.join(self.data_dir, filename)
            text = self._extract_text_from_pdf(path)
            sentences = self._split_to_sentences(text)
            for sent in sentences:
                emb = self._encode(sent)
                self.sentences.append({"text": sent, "source":
filename, "embedding": emb})
```

Отже, база відповідей – це збірка окремих речень з документів, кожне з яких має власне векторне відображення. Ці вектори генеруються з використанням моделі Ukr-RoBERTa, що демонструє позитивні показники

на мовному масиві. Коли користувач задає питання, система перетворює його у вектори аналогічно до індексованих речень. Після цього здійснюється порівняння запиту з усіма реченнями корпусу за допомогою косинусної подібності. Додатково, для поліпшення точності, перед порівнянням, відбувається класифікація запиту за тематикою: терміни, реквізити, форматування, обсяг. Категорія визначається на основі ключових слів у тексті питання. Вона використовується для посилення ваги тих речень, які ймовірно належать до цієї теми, що дозволяє компенсувати випадки, коли схожість формально нижча, але контекст – релевантний.

Лістинг 3.5 – Вибір найрелевантнішої відповіді серед усіх речень

```
def find_best_answer(self, question: str) -> Dict:
    query_emb = self._encode(question)
    category = self._detect_category(question)
    scored = []
    for sent in self.sentences:
        sim = F.cosine_similarity(query_emb, sent["embedding"], dim=0).item()
        boost = 0.4 if any(kw in sent["text"].lower() for kw in self.keywords[category]) else 0
        scored.append((sim + boost, sent))
    best = max(scored, key=lambda x: x[0])
    return {
        "matched_text": best[1]["text"],
        "source_file": best[1]["source"],
        "score": round(best[0], 4)
    }
```

Відповідь формується як речення, взяте з документа, разом з назвою джерела. На відміну від генеративного методу, де GPT самостійно формує відповідь, у даному випадку модель нічого нового не вигадує. Вона просто обирає найдоречніше речення з наявних в документах, що забезпечує

достовірність відповідей офіційній інформації та усуває ймовірність хибних тверджень. Реалізація в такій формі тим самим уникає генерації тексту, який міг би суперечити офіційним формулюванням. Важливо враховувати те, що помилка у формулюванні може призвести до неправильного тлумачення вимог конференції.

Блок дозволяє працювати з неструктурованими текстами, не вимагаючи ручного редагування або формалізації документації. Система індексує все виявлене, гарантуючи максимальне пряме відтворення змісту у відповіді на питання користувача, полегшуючи роботу координаторів конференцій та дозволяє надавати оперативну й узгоджену інформаційну підтримку.

3.3.3 Перевірка структури

Одним із основних завдань системи є автоматизована верифікація структури наукових праць, щоб переконатися у відповідності формальним стандартам. У цьому блоці використовується модель великої мови (LLM), яка, базуючись на підготовленому промпті, здійснює аналіз тексту та встановлює наявність важливих складових, керуючись конкретними вимогами (наприклад, ДСТУ 8302:2015), що дозволяє оптимізувати процес перевірки поданих робіт, а також покращити об’єктивність оцінювання наявності необхідних розділів.

Перш ніж відправити текст до LLM, система виокремлює основний контент із PDF-файлу. З метою зменшення навантаження на модель застосовується обрізання зайвої інформації, що сприяє охопленню ключових розділів статті: назви, вступу, мету, методи тощо. Головна логіка перевірки реалізована через промпт, що включає деталізований перелік вимог. Модель зобов’язана надати відповідь у форматі JSON, де для кожного структурного компоненту вказується його наявність або супровідний коментар. Запит скерується до моделі GPT (через API),

супроводжуваний текстом. Від відповіді передбачається структурований формат, придатний для програмної обробки. У разі неможливості розпізнати відповідь як коректний JSON, система повідомить про помилку.

Після отримання відповіді, вона розбирається та результати виводяться. Для кожного компонента буде відображена оцінка наявності, а у випадку неоднозначності – короткий коментар, що дає змогу редакторам та рецензентам оперативно визначити повноту тексту, не вдаючись до його детального аналізу (лістинг 3.6).

Лістинг 3.6 – Обробка відповіді моделі та перетворення у Python-структуру

```
def _try_parse_json(response: str) -> dict:
    start = response.find("{")
    end = response.rfind("}") + 1
    try:
        return json.loads(response[start:end])
    except Exception:
        return {"error": "Не вдалося розпізнати JSON-відповідь GPT."}
```

Перевірка структури втілена як напівформалізована операція: модель не вдається до глибокого лінгвістичного розбору, але дає можливість виявити очевидну відсутність секцій або вади оформлення. Рішення доречне для початкового відбору матеріалів перед рецензуванням, а його результати можуть бути застосовані як додатковий інструмент у загальному процесі технічного контролю.

3.3.4 Перевірка на плагіат

Перевірка на плагіат – це компонент системи, що має на меті виміряти унікальність представленого тексту, яке здійснюється шляхом зіставлення

тексту з різноманітними ресурсами: відкритими джерелами інформації, науковими базами даних, а також з внутрішнім сховищем попередньо збережених робіт. У втіленому рішенні використовується зовнішній API сервісу Copyleaks, який пропонує багатомовну перевірку, працює з PDF-файлами та здійснює аналіз на рівні сутності змісту.

Інтеграція з Copyleaks базується на REST API з використанням спеціальних токенів для автентифікації. Щоб здійснити перевірку тексту, потрібно виконати два ключові етапи: спочатку згенерувати маркер авторизації (access token), а потім передати дані для аналізу. Перед відправкою текст видобувається з PDF-файлу, його очищують від непотрібних деталей та форматування, після чого надсилають у формі простого тексту (plain-text) або рядка, закодованого у форматі base64, залежно від формату запиту.

Для отримання маркера доступу потрібно надіслати запит до OAuth- ендпоінту Copyleaks. API повертає текстовий токен у відповідь, який має обмежений термін дії, який в свою чергу зберігається в оперативній пам’яті з метою повторного застосування, якщо час його чинності дозволяє (лістинг 3.7).

Лістинг 3.7 – Генерація токену для Copyleaks API

```
def get_access_token(user_id, api_key):
    url = "https://id.copyleaks.com/v3/account/login/api"
    payload = {"email": user_id, "key": api_key}
    headers = {"Content-Type": "application/json"}
    response = requests.post(url, json=payload, 
headers=headers)
    return response.json().get("access_token")
```

Отримавши токен, система ініціює сесію сканування. Для цього текст або файл відправляється POST-запитом на сервер Copyleaks, де вказуються параметри. До них належать мова, бажана глибина сканування, включення

звітів тощо. API реалізує асинхронне опрацювання, тобто перевірка не відбувається миттєво. Система мусить чекати на сповіщення про завершення (callback) або регулярно запитувати статус вручну.

Інтеграція з Copyleaks реалізована у вигляді окремого модуля з підтримкою аутентифікації, передачі текстів та обробки результатів. Перевірка здійснюється через зовнішнє API з можливістю отримання повного звіту про збіги. У разі потреби система також підтримує локальне порівняння з внутрішньою базою даних за допомогою SBERT.

Результати сканування від Copyleaks надходять як структура, що містить відсоток подібності, перелік збігів та уривки схожих частин тексту. Дані подаються користувачу у спрощеному форматі: загальний рівень плагіату, назви найсхожіших документів, а також текстовий коментар (його формування наведено у лістингу 3.8).

Лістинг 3.8 – Обробка результатів перевірки Copyleaks

```
def handle_plagiarism_response(data: dict) -> dict:
    results = data.get("results", [])
    top_matches = [
        {"title":        r["title"],        "similarity":
round(r["score"], 2)}
        for r in results[:5]
    ]
    return {
        "plagiarism_percent":
round(data.get("overall_score", 0), 2),
        "top_matches": top_matches,
        "comment": data.get("commentary", "")
    }
```

Варто врахувати факт, що відсоткові показники, надані Copyleaks, є орієнтовними. Великий відсоток збігів не обов’язково свідченням про плагіат. Цитування, використання тих самих методик або цитати з

нормативних документів також можуть бути причинами. Саме тому в інтерфейсі передбачено можливість перегляду всіх збігів, щоб можна було провести візуальний аналіз.

На додаток до використання зовнішнього API, система проводить перевірку шляхом локального семантичного порівняння, яке базується на Sentence-BERT, що дозволяє виявляти схожість із роботами, які не розміщені у відкритому доступі, наприклад, попередніми роботами, поданими на конференції, внутрішніми тезами та іншими матеріалами. Комбінований підхід підвищує здатність використання системи як у внутрішньому контексті, так і для виявлення порушень в рамках зовнішніх джерел.

3.3.5 Виявлення згенерованого тексту

Модуль виявлення AI-тексту – функціональний елемент системи, призначений автоматично визначати ймовірність того, що наданий матеріал був створений за допомогою великої мовної моделі (LLM), зокрема GPT. Завдання полягає у виявленні випадків автоматичного генерування текстів, які намагаються виглядати авторськими, але мають ознаки шаблонності, поверховості або плагіату. Перевірка є надзвичайно важливою з огляду на поширення моделей текстової генерації та загрозу заміни реального наукового внеску автоматизованим контентом.

Інтеграція з GPT-4 реалізована через офіційне API OpenAI. Для кожного тексту формується окремий запит, який містить інструкції для моделі: оцінити ймовірність того, що представлений фрагмент був створений ШІ, надати числову оцінку (у відсотках) та короткий коментар. Система самостійно формує цей промпт на основі поданого PDF, вилучаючи з нього основний зміст. Запит відбувається шляхом прямого звернення до ChatCompletion.create, а обробка результату відбувається системою у фоновому режимі. Відповідь GPT передбачається у вигляді текстового

блоку, що включає два елементи: числове значення оцінки та письмовий висновок. Інформація автоматично опрацьовується та інтегрується в кінцевий звіт про роботу (лістинг 3.9).

Лістинг 3.9 – Обробка відповіді

```
def detect_ai_generated(text: str) -> dict:
    prompt = build_ai_prompt(text)
    response = openai.ChatCompletion.create(
        model="gpt-4",
        messages=[
            {"role": "system", "content": "Ти аналізуєш тексти і визначаєш, чи вони створені GPT."},
            {"role": "user", "content": prompt}
        ],
        temperature=0.2
    )
    return {"ai_analysis": response["choices"][0]["message"]["content"]}
```

Модуль функціонує автономно, не потребуючи втручання рецензента чи адміністратора. Він не впливає на кінцеве рішення напряму, але виступає додатковим критерієм оцінювання. Скажімо, якщо AI-аналіз виявляє потенційне генерування тексту, рецензент може звернути увагу на структурні особливості або зіставити їх з результатами перевірки на плагіат.

На відміну від ручного аналізу, цей механізм гарантує стандартизований підхід до оцінювання текстів, зменшуючи суб’єктивний вплив. Більше того, застосування промпт-інженерії дозволяє отримати від моделі лаконічні та однозначні результати, які легко інтегруються в загальну систему технічної перевірки. Модуль детекції AI-тексту є ключовим елементом архітектури, що забезпечує додатковий рівень контролю якості поданих матеріалів та сприяє дотриманню академічної доброчесності.

3.4 Розробка інтерфейсу користувача

Користувацький інтерфейс системи спроєктовано як односторінковий вебдодаток, застосовуючи HTML, CSS та JavaScript, не залучаючи сторонніх фреймворків. Таке рішення прийнято з огляду на простоту, можливість повного контролю та гарантовану взаємодію з FastAPI- сервером, який обслуговує обидві складові системи: API та шаблони.

Інтерфейс працює на основі HTML-шаблону index.html, який обробляє шаблонізатор Jinja2. У шаблоні передбачено два головні функціональні блоки: для введення користувачем текстових запитів та для аналізу PDF-документів. Блоки організовано як вкладки, між якими можна перемикатися без перезавантаження сторінки. JavaScript відповідає за логіку відтворення вмісту, відправку запитів на сервер та обробку отриманих відповідей.

Будь-яка взаємодія користувача (введення тексту чи завантаження файлу) ініціює надсилання відповідного запиту до бекенду через API-ендпоінт. Для цього формується об’єкт FormData, що передається методом POST через fetch. Відповідь від сервера може включати структуровані результати з декількох модулів: аналізу структури, перевірки на плагіат, оцінки AI-генерованого контенту, пошуку подібних робіт або вилучення даних з організаційних документів.

На рівні JavaScript обробка запитів чітко диференційована залежно від типу вхідних даних. Якщо подано лише текст, він спрямовується на аналіз у режимі org, або, якщо це науковий запит – на семантичний пошук (лістинг 3.10). Якщо ж надано файл, запускається перевірка повного PDF-документу, результатом якої стає сукупність секцій: структура, джерела, плагіат, AI-аналіз. Для кожного виду відповіді передбачено відповідну функцію відображення. Результати показуються у заздалегідь визначених блоках сторінки, які динамічно оновлюються. Користувач

отримує результати без перезавантаження сторінки, що забезпечує інтерактивність та зручність використання.

Лістинг 3.10 – Відправка PDF-файлу для перевірки

```
pdfForm.addEventListener("submit", async (e) => {
    e.preventDefault();
    const file = document.getElementById("file-pdf").files[0];
    if (!file) return;

    const formData = new FormData();
    formData.append("file", file);

    const res = await fetch("/process", {
        method: "POST",
        body: formData
    });

    const result = await res.json();
});
```

Особливу увагу зосереджено на створенні результатів у зручному для сприйняття вигляді. В розділі перевірки структури кожен компонент (заголовок, вступ, перелік джерел тощо) представлений як позиція списку, позначена знаком відповідності, або коротким зауваженням, якщо модель вказує на двозначність, що забезпечує швидку оцінку якості структури документа. Під час демонстрації результатів перевірки плагіату, система показує відсоткову оцінку схожості, а також перелік найближчих збігів. За наявності даних з внутрішньої бази, вказуються автор, назва та ступінь подібності. Крім того, якщо в модулі активна детекція AI-тексту, додатково відображаються оцінка вірогідності генерації та короткий коментар від GPT.

Для стилізації інтерфейсу застосовано окремий CSS-файл, який визначає адаптивні розміри контейнерів, колірні схеми для різних блоків, а також стилі для компонентів управління, таких як кнопки, вкладки та списки. Головний акцент зроблено на зручність сприйняття інформації та логічну організацію виводу. Кожен блок результатів чітко розмежовано, має назву, заголовок та внутрішню структуру.

З технічної точки зору, інтерфейс не виконує жодної логічної перевірки. Його функція обмежується формуванням запиту та відображенням відповіді. Обчислення, обробка тексту та порівняння здійснюються на серверній частині, що дозволяє легко оновлювати логіку на бекенді без необхідності змінювати фронтенд, а також сприяє підвищенню безпеки та стабільності системи. Реалізація ж JavaScript коду є компактною, структурованою та не містить сторонніх залежностей, забезпечуючи легкість інтеграції в будь-яке інше середовище або розширення новими функціями, не ризикуючи порушити поточну логіку.

3.5 Допоміжні модулі

Окрім основних складових, проєкт реалізує низку допоміжних модулів, що забезпечують функціональність, придатну для повторного використання, та сприяють структурованості й стабільності системи. Їхня мета полягає в оптимізації рутинних операцій, інкапсуляції логіки обробки файлів, логуванні процесів, підготовці шаблонів запитів до мовних моделей, а також формалізованій перевірці форматів бібліографічних даних. Модулі мають чітко окреслені функції, розташовані в директорії utils/ та підключаються до основного коду за потреби.

Одним із ключових елементів є модуль file_utils.py. Його функціонал охоплює створення унікальних імен файлів за допомогою UUID, формування повних шляхів до тимчасових або кінцевих директорій, а також створення структури папок у випадку їхньої відсутності. Це дає змогу

мінімізувати ймовірність конфліктів імен, дублювання даних чи несанкціонованого перезапису. Використання цих функцій у всій системі дозволяє централізовано керувати розміщенням файлів та забезпечує їхню ізольовану обробку. До того ж, це спрощує тестування та налагодження: кожен файл має унікальне ім’я, а тимчасові директорії можна очищати окремо.

Для відстеження внутрішніх процесів, виявлення помилок, моніторингу запитів до API та інших критичних подій, використовується модуль logger.py. Він базується на стандартному інструментарії Python-модуля logging, але налаштований для одночасного виведення логів як у файл (logs/system.log), так і в консоль. Кожен запис містить часову мітку, рівень логування (INFO, ERROR тощо) та саме повідомлення, що дозволяє ефективно виконувати дебагінг системи, виявляти проблеми під час роботи з API, а також вести хроніку звернень до моделей або обробки PDF-файлів.

Окремо реалізовано модуль pdf_parser.py, що призначений для вилучення тексту з PDF-документів. Він побудований на бібліотеці PyMuPDF (fitz) та орієнтований на максимально повне видобування тексту зі сторінок без втрати структури. У межах функціоналу відкривається документ, послідовно обробляються сторінки, а отриманий текст об’єднується в один рядок, що надалі використовується у таких модулях, як структурна перевірка, аналіз AI-генерації, плагіат-детектор тощо. Реалізація враховує потенційні помилки відкриття чи некоректні файли: у таких випадках повертається порожній рядок, що дозволяє обробляти винятки без зупинки програми.

Ще одним важливим компонентом є regex_checkers.py. Цей модуль містить регулярні вирази, розроблені для перевірки бібліографічних посилань згідно з базовими вимогами ДСТУ 8302:2015. Основна перевірка здійснюється на рівні шаблону: визначення порядку елементів (автор, назва, місто, рік), форматування і наявності необхідних розділових знаків. Результатом роботи модуля є перелік об’єктів, у яких зазначено рядок,

валідність та причину невідповідності у випадку помилки. Така реалізація дозволяє уніфіковано перевіряти перелік джерел без залучення GPT-моделі та отримувати швидкий зворотний зв’язок щодо коректності оформлення. Крім того, модуль легко доповнюється новими шаблонами – для статей з DOI, онлайн-ресурсів, нормативних документів тощо.

З метою підтримки цілісності запитів до великих мовних моделей та уникнення дублювання коду застосовується модуль prompt_templates.py. У ньому зберігаються функції, які повертають готові шаблони запитів для різних задач: перевірки структури за ДСТУ, виявлення AI-генерованого тексту та інших. Кожна функція приймає текст як аргумент, вставляє його у шаблон з попередньо прописаними інструкціями для LLM, а також уточненнями щодо формату очікуваної відповіді (наприклад, JSON або список з об’єктами). Централізоване зберігання промптів дозволяє підтримувати єдиний стиль взаємодії з моделлю, а також уніфікувати обробку її відповіді на бекенді.

Модулі у сукупності формують базову інфраструктуру, без якої робота основної системи була б громіздкою, нестабільною та складною у підтримці. Їхнє ізольоване застосування дозволяє спростити розгортання, масштабування та налагодження: кожен модуль має чітку відповідальність і взаємодіє лише через визначені інтерфейси. Архітектура відповідає принципам модульності, повторного використання та супроводу, що критично важливо для систем, орієнтованих на аналіз та обробку великої кількості вхідних документів.

4 ВІЗУАЛЬНЕ ПРЕДСТАВЛЕННЯ ТА ЕЛЕМЕНТИ ІНТЕРФЕЙСУ КОРИСТУВАЧА

4.1 Головна сторінка

Головна сторінка вебдодатку – це перше, з чим зустрічається користувач, взаємодіючи з системою. Дизайн, який реалізований, гранично стислий і зосереджений на простоті, зручній навігації та чіткому розмежуванню функціоналу. У верхній частині сторінки розташована назва системи: «ConfNLP – Асистент для перевірки PDF та допомоги співробітникам». Під заголовком знаходиться панель навігації, представлена двома кнопками, які дозволяють перемикатися між основними режимами роботи додатку. Кожна кнопка відповідає за окремий функціональний блок:

– допомога працівнику, режим для формування текстових запитів до бази організаційних документів (рисунок 4.1);

– перевірка PDF, режим для завантаження документів та всебічної перевірки їх на відповідність вимогам і плагіат (рисунок 4.2).

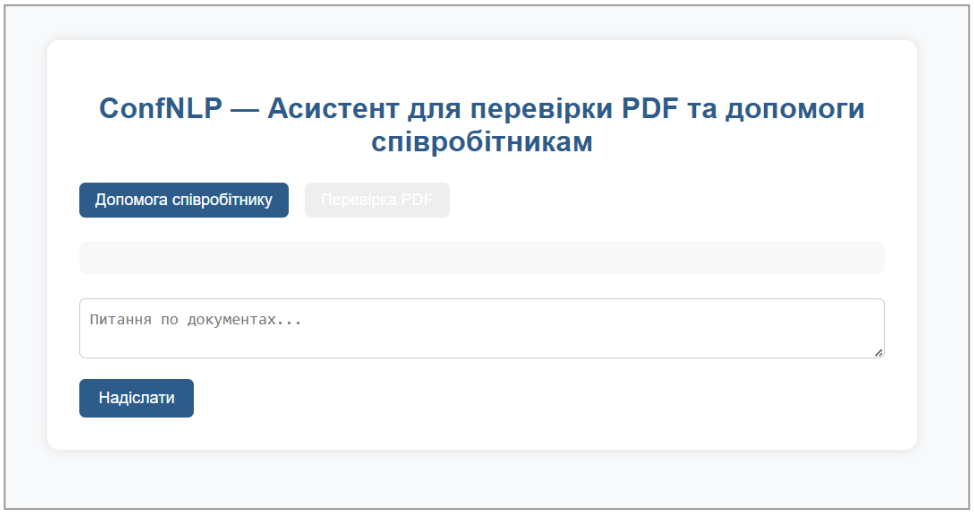

Рисунок 4.1 – Головна сторінка в режимі «Допомога співробітнику»

Навігаційні кнопки мають просту прямокутну форму, чітке шрифтове оформлення та змінюють колір при активації, що полегшує орієнтування в тому, яка вкладка активна на даний момент. Кнопки розміщені горизонтально та візуально відокремлені, що робить перехід між режимами легким. Перемикання між вкладками відбувається без перезавантаження сторінки – активується відповідний блок введення або завантаження даних, а інший приховується. Завдяки цьому взаємодія відбувається плавно, а час відгуку інтерфейсу – мінімальний.

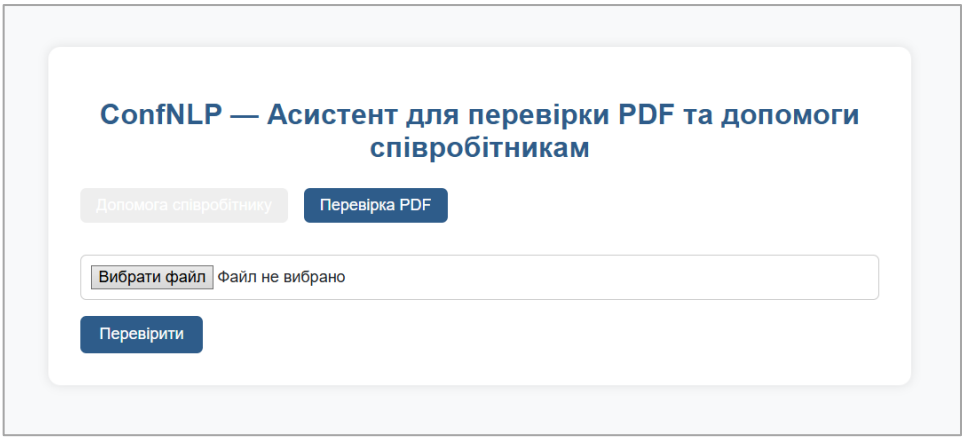

Рисунок 4.2 – Головна сторінка в режимі «Перевірка PDF»

Під панеллю навігації розміщено головні елементи для введення інформації. У активній вкладці відображається або форма для введення тексту запиту, або форма для завантаження PDF-документа, залежно від обраного режиму. Головна сторінка має центровану компоновку, тобто всі основні елементи розміщено по центру екрану, що полегшує взаємодію незалежно від розміру пристрою. Реалізація забезпечує адаптивність інтерфейсу: він коректно виглядає як на великих екранах настільних ПК, так і на мобільних пристроях з меншими дисплеями.

Кнопки навігації мають темно-синій фон і білий текст, що забезпечує достатній контраст і зручність для користувача. Вся логіка навігації спрямована на мінімізацію кількості кліків і переходів. Користувачу

достатньо вибрати потрібний режим роботи за допомогою навігаційних кнопок і перейти до введення тексту або завантаження файлу. Додаткові елементи, як-от поля для введення чи вибору файлів, з’являються динамічно лише в межах обраної вкладки, що забезпечує чистоту і простоту головної сторінки.

4.2 Опис основних функціональних блоків

Інтерфейс вебдодатку містить ключові функціональні блоки, що реалізують повний цикл взаємодії користувача з системою: введення запиту, завантаження файлів, ініціювання перевірки та перегляд результатів. Блок для текстового запиту доступний у вкладці «Допомога співробітнику». Він являє собою однорядкове текстове поле для введення питань щодо організаційних аспектів конференції. Поле має підказку у вигляді тексту, що пропонує ввести запитання. Під полем розміщена кнопка «Надіслати», яка запускає процес обробки запиту. Натиснення кнопки ініціює асинхронне надсилання даних на серверну частину системи. Результатом є вивід відповіді, знайденої серед організаційних документів за допомогою векторного пошуку. Вивід реалізовано у вигляді чат-логу, де відображається як запит користувача, так і відповідь системи, створюючи зрозумілу візуальну модель діалогу, полегшуючи сприйняття інформації (рисунок 4.3).

Другий блок розміщений у вкладці «Перевірка PDF». Його функціональне призначення – завантаження файлів для комплексного аналізу тексту. Форма містить кнопку «Вибрати файл», що відкриває стандартне вікно вибору файлів на пристрої користувача. Після вибору назва обраного файлу відображається поруч із кнопкою, що забезпечує зворотний зв’язок та запобігає помилкам при виборі.

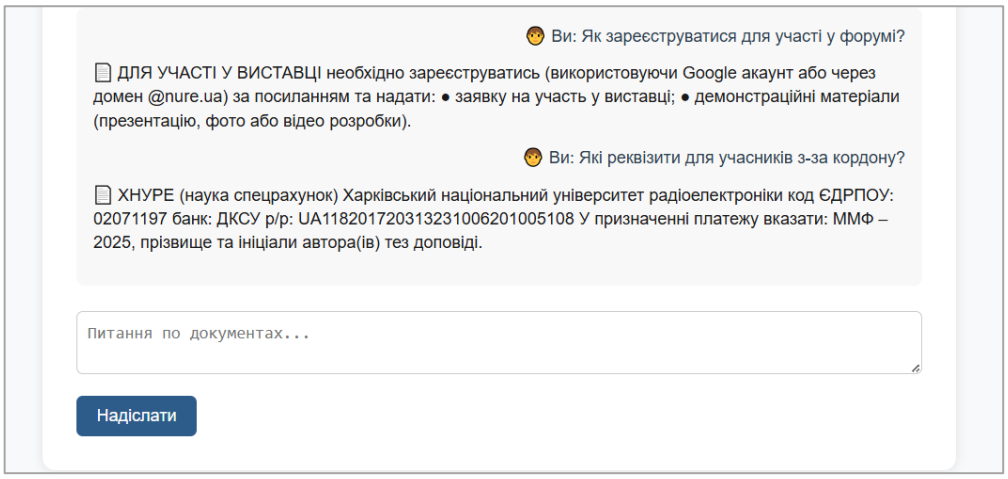

Рисунок 4.3 – Діалогове вікно відповідей на запитання

Кнопка «Перевірити», яка активує процес відправки PDF-файлу на сервер для подальшого аналізу. Відправка реалізується як асинхронний запит без перезавантаження сторінки. У разі помилок завантаження або некоректного формату файлу система виводить відповідне повідомлення для користувача (рисунок 4.4).

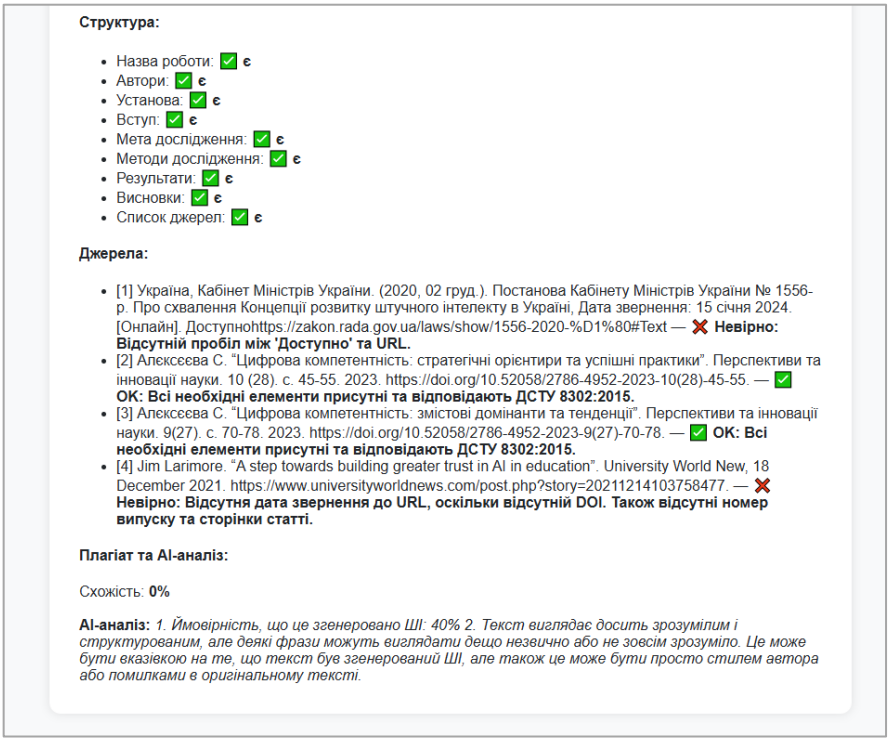

Рисунок 4.4 – Діалогове вікно аналізу наданого тексту

Після обробки запиту або перевірки документа, результати відображаються в окремих секціях, кожна з яких відповідає певному модулю системи. Секції з’являються динамічно, після завершення обробки, що дозволяє уникнути перевантаження інтерфейсу.

Секція перевірки структури документа містить перелік обов’язкових елементів, які повинні бути в науковій роботі згідно з вимогами ДСТУ. Кожен пункт структури має окремий рядок з відповідним маркуванням: зеленою галочкою, якщо елемент знайдено, або заборонним знаком у разі його відсутності.

Секція перевірки на плагіат містить кілька блоків. Перший – загальний відсоток подібності тексту з іншими роботами. Далі виводиться список найбільш схожих документів із зазначенням назв, авторів та рівня збігу у відсотках. Список оформлений як маркований перелік, де кожен рядок містить стислу інформацію про знайдену відповідність.

Секція аналізу згенерованості тексту містить інформацію про ймовірність того, що текст створено мовною моделлю, та короткий аналітичний коментар. Ймовірність виводиться як числове значення у відсотках, а коментар надає текстову оцінку характеру тексту з точки зору його структури, стилю та мовних особливостей.

4.3 Аналіз роботи інтерфейсу

Інтерфейс системи сконструйовано так, аби мінімізувати когнітивне навантаження на користувача. Кожен елемент має чітку функцію, відсутнє перенасичення екрану зайвими деталями чи непотрібною інформацією. Чітке розмежування на вкладки забезпечує логічну сегментацію функцій. Користувач одразу бачить доступні опції взаємодії та може вибрати потрібний режим без зайвих дій. Всі операції зведено до одного або двох натискань: введення запиту, завантаження файлу, отримання результатів.

Окрема увага приділена відображенню результатів. Всі результати структуровано за типами перевірок, що усуває плутанину та дозволяє зосередитись на ключових аспектах: структурі документа, рівні плагіату, аналізі тексту на предмет AI-генерації. Подання результатів у вигляді окремих блоків з чіткими заголовками і внутрішньою структурою дає змогу не тільки переглядати інформацію, але й швидко оцінювати її за допомогою індикаторів (позначок відповідності, відсотків збігу тощо).

Інтерфейс демонструє стабільну роботу як на стаціонарних пристроях, так і на мобільних телефонах. Завдяки адаптивній верстці всі елементи коректно масштабуються під ширину екрану, забезпечуючи збереження читабельності й функціональності. Особливо важливо, що навіть при зменшенні розміру екрану зберігається повноцінна доступність всіх функцій, без необхідності горизонтального скролу або приховування інформації.

Швидкість роботи інтерфейсу є ще однією позитивною рисою. Використання асинхронних запитів дозволяє миттєво отримувати відповіді без перезавантаження сторінки, що істотно зменшує час очікування, що позитивно впливає на користувацький досвід і створює відчуття безперервної взаємодії із системою.

Інтерфейс поєднує функціональність та простоту. Він забезпечує швидкий доступ до всіх можливостей системи, дозволяючи користувачу без зайвих труднощів виконувати перевірку документів або отримувати необхідні відповіді на запити. Всі функціональні блоки логічно пов’язані між собою і працюють як єдина система, що підвищує ефективність роботи та мінімізує ймовірність помилок з боку користувача.

ВИСНОВКИ

У процесі роботи над кваліфікаційною роботою було розроблено та досліджено інтелектуального помічника для підтримки організації наукових форумів, використовуючи технології штучного інтелекту [26]. Головний акцент було зроблено на інтеграції великих мовних моделей (LLM) та методології Prompt Flow Engineering (PFE), що дозволило створити систему, яка ефективно обробляє запити користувачів, аналізує надані матеріали та надає автоматизовану підтримку в процесах подання та перевірки наукових праць.

На основі аналізу предметної області та існуючих рішень була спроєктована архітектура асистента, яка складається з чотирьох основних модулів: SBERT для семантичного пошуку, Ukr-RoBERTa-base для обробки україномовних запитів, інструменти перевірки текстів на академічну доброчесність, а також модуль ChatGPT для діалогової взаємодії та валідації текстів на відповідність структурним вимогам.

В рамках дослідження було розроблено типові сценарії використання системи для різних категорій користувачів – учасників конференцій, рецензентів та організаторів. Описано логіку обробки запитів та перевірки документів відповідно до визначених критеріїв якості.

Результати дослідження підтверджують доцільність і перспективність впровадження інтелектуальних асистентів на базі наведених моделей. Запропонована модель відзначається модульністю, масштабованістю та лінгвістичною адаптивністю, що забезпечує її відповідність потребам української академічної спільноти. Створена структура є основою для подальшої реалізації асистента, його впровадження та тестування в умовах реального функціонування наукового форуму.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Șerbu R. S., Mârza B. Ș. From crisis to opportunity: embracing sustainable development goals and artificial intelligence in the transformative innovations world. *SpringerLink*. URL: https://link.springer.com/chapter/10.1007/978-3-031-66434-2_4 (date of access: 26.04.2025).

2. Murdan A. P., Halkhoree R. Integration of Artificial Intelligence for educational excellence and innovation in higher education institutions. URL: https://doi.org/10.1109/sesai61023.2024.10599402 (date of access: 26.04.2025).

3. Digital transformations: artificial intelligence in higher education / A. de Bem Machado et al. *EAI/Springer innovations in communication and computing*. Cham, 2024. P. 1–23. URL: https://doi.org/10.1007/978-3-031-52296-3_1 (date of access: 26.04.2025).

4. Complexity analysis and construction of conceptual graphs of mass open online courses / V. I. Zherebkin et al. *Bionics of Intelligence*. 2023. Vol. 1, no. 99. P. 26–37. URL: https://doi.org/10.30837/bi.2023.1(99).04 (date of access: 14.06.2025).

5. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Stroudsburg, PA, USA, 2019. URL: https://doi.org/10.18653/v1/d19-1410 (date of access: 13.06.2025).

6. The role of artificial intelligence in achieving the Sustainable Development Goals / R. Vinuesa et al. *Nature communications*. 2020. Vol. 11, no. 1. URL: https://doi.org/10.1038/s41467-019-14108-y (date of access: 26.04.2025).

7. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv.org*. URL: https://arxiv.org/abs/1907.11692 (date of access: 27.04.2025).

8. Reimers N., Gurevych I. Sentence-BERT: sentence embeddings using siamese bert-networks. *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, Hong Kong, China. Stroudsburg, PA, USA, 2019. URL: https://doi.org/10.18653/v1/d19-1410 (date of access: 27.04.2025).

9. Gao A. Prompt engineering for large language models. *SSRN electronic journal*. 2023. URL: https://doi.org/10.2139/ssrn.4504303 (date of access: 27.04.2025).

10. Event technology and trends archives. *Congrex Switzerland*. URL: https://congrex.com/topics-tags/event-technology-trends/ (date of access: 28.04.2025).

11. Artificial intelligence in congress and conference organization '2024 – CME. *CME – Congress Management & Events*. URL: https://cmebg.com/en/2024/01/18/artificial-intelligence-in-congress-and-conference-organization-2024 (date of access: 28.04.2025).

12. Prompt flow in azure AI foundry portal - azure AI foundry. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/azure/ai-foundry/concepts/prompt-flow (date of access: 28.04.2025).

13. IBM. What is prompt chaining? | IBM. *IBM − United States*. URL: https://www.ibm.com/think/topics/prompt-chaining (date of access: 28.04.2025).

14. Prompt design and engineering: introduction and advanced methods. *arXiv.org e-Print archive*. URL: https://arxiv.org/html/2401.14423v4 (date of access: 28.04.2025).

15. SentenceTransformers documentation − sentence transformers documentation. URL: https://sbert.net/ (date of access: 29.04.2025).

16. Sentence-transformers. Hugging Face – *The AI community building the future*. URL: https://huggingface.co/sentence-transformers (date of access: 29.04.2025).

17. Language models are few-shot learners. *List of Proceedings*. URL: https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (date of access: 29.04.2025).

18. Youscan/ukr-roberta-base hugging face. *Hugging Face – The AI community building the future*. URL: https://huggingface.co/youscan/ukr-roberta-base (date of access: 30.04.2025).

19. Predicting informativeness of semantic triples. *ACL Anthology*. URL: https://aclanthology.org/2021.ranlp-1.126/ (date of access: 30.04.2025).

20. Using llms as peer reviewers for revising essays − the WAC clearinghouse. *The WAC Clearinghouse*. URL: https://wac.colostate.edu/repository/collections/textgened/rhetorical-engagements/using-llms-as-peer-reviewers-for-revising-essays/ (date of access: 02.05.2025).

21. Andreychikov A., Andreychikova O. Intelligent information systems and artificial intelligence methods. ru : INFRA-M Academic Publishing LLC., 2021. URL: https://doi.org/10.12737/1009595 (date of access: 04.05.2025).

22. Session AI: artificial intelligence. *2020 15th international conference on computer engineering and systems (ICCES)*, Cairo, Egypt, 15–16 December 2020. 2020. URL: https://doi.org/10.1109/icces51560.2020.9334674 (date of access: 04.05.2025).

23. Lemos R., Grzes M. Self-Adaptive artificial intelligence. *2019 IEEE/ACM 14th international symposium on software engineering for adaptive and self-managing systems (SEAMS)*, Montreal, QC, Canada, 25 May 2019. 2019. URL: https://doi.org/10.1109/seams.2019.00028 (date of access: 04.05.2025).

24. Теплюк М. В., Гриньова О. Є. Підхід до створення інтелектуального помічника для наукових конференцій на основі LLM та PFE. *Радіоелектроніка та молодь у XXI столітті*, м. Харків. 2025. С. 63−65. URL: https://openarchive.nure.ua/server/api/core/bitstreams/fb910422-c426-4940-ba77-a209eb9a99ee/content (date of access: 05.06.2025).