ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ І ТЕХНОЛОГІЙ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

бакалавра
(освітньо-кваліфікаційний рівень)

на тему *Розробка інтелектуального модуля для автоматизованого редагування текстів для веб-системи публікації творів*

Виконав: здобувач *4* року навчання
групи *4ПР1*
напряму підготовки (спеціальності)
121-Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)

Ринденко Є.І.
(підпис, прізвище та ініціали)

Керівник *Огнєва О.Є.*
(підпис, прізвище та ініціали)

Рецензент *к.т.н., доцент Корніловська Н.В.*
(прізвище та ініціали)

Нормоконтролер *Комісаров О. С.*
(підпис, прізвище та ініціали)

Хмельницький – 2026

ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Інститут, факультет, відділення *Інформаційних технологій та дизайну*
Кафедра, циклова комісія *Програмних засобів і технологій*
Освітньо-кваліфікаційний рівень *бакалавр*
Освітньо-професійна підготовка *Програмна інженерія*
(шифр і назва)
Спеціальність *121-Інженерія програмного забезпечення*
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
програмних засобів і технологій

_____ О.Є. Огнєва
«___» _____
2026 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Ринденко Єлизаветі Ігорівні
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) *Розробка інтелектуального модуля для автоматизованого редагування текстів для веб-системи публікації творів*

керівник проєкту (роботи) *к.т.н. доцент Огнєва О.Є.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «16» січня 2026 року №82-с

2. Строк подання здобувачем проєкту(роботи) 10.06.2026

3. Вихідні дані до проєкту (роботи) *літературні та періодичні джерела, матеріали переддипломної практики, результати проведеного дослідження, офіційна технологічна документація.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) Аналіз предметної області та постановка задачі
2) Проєктування архітектури та логіки роботи модуля
3) Розробка модулю автоматизованого редагування
4) Тестування, впровадження та оцінка результатів

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *14.01.2026*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів проєкту (роботи)	Примітка
1	Отримання завдання	14.01.2026	Виконано
2	Пошук та аналіз літературних джерел	16.01.2026-26.01.2026	Виконано
3	Організація та проведення дослідження щодо сприйняття ШІ-тексту.	27.01.2026-16.02.2026	Виконано
4	Обробка та аналіз результатів дослідження	17.02.2026-20.02.2026	Виконано
5	Формалізація вимог до модуля	21.02.2026-07.03.2026	Виконано
6	Проєктування архітектури та логіки інтелектуального модуля, алгоритму визначення ШІ-редагування	08.03.2026-20.03.2026	Виконано
7	Програмна реалізація	20.03.2026-15.04.2026	Виконано
8	Проведення тестування та інтеграція до платформи Dragon Tale	16.04.2026-29.04.2026	Виконано
9	Написання пояснювальної записки	30.04.2026-27.05.2026	Виконано
10	Захист кваліфікаційної роботи	24.06.2026	Виконано

Здобувач _____ *Ринденко Є.І.*
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ *Огнєва О.Є.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 104 сторінки, 27 рисунков, 9 таблиць, 48 джерел, 1 додаток.

Мета роботи – дослідження ефективності контрольованого використання великих мовних моделей у процесі редагування художнього тексту та на основі отриманих результатів проєктування, розробка й інтеграція інтелектуального модуля автоматизованого редагування до літературної веб-платформи.

Об'єкт дослідження – процес редагування художнього тексту в межах літературних веб-платформ.

Предмет дослідження – методи та способи контрольованого автоматизованого редагування художнього тексту на базі ШІ з урахуванням механізмів забезпечення прозорості його застосування.

Новизна роботи:

- інтелектуальний модуль автоматизованого редагування на базі ШІ реалізовано для контрольованої обробки художнього тексту та інтегровано безпосередньо у вбудований редактор платформи;
- використаний перелік архітектурних рішень (контроль обсягу і перевірка фрагменту, human-in-the-loop підхід і бібліотека структурованих шаблонів запитів до LLM) дозволяє мінімізувати як неконтрольоване використання ШІ, так і можливості втрати під час обробки авторського стилю;
- впроваджений алгоритм автоматичного визначення факту використання ШІ-редагування на основі снапшотів і методу шинглів, а також реалізовано механізм незворотної фіксації відповідної мітки прозорості на рівні бази даних;
- практично продемонстрована можливість етичного застосування ШІ в межах літературних веб-платформ завдяки контрольованості та прозорості його використання.

Інструменти та засоби розробки: для практичної реалізації модуля використовувався технологічний стек, що притаманний цільовій платформі – full- stack Next.js з мовою TypeScript. Завдяки платформі Supabase відбувається робота з даними та автентифікацією. У якості мовної моделі обрано Llama-3.1-8B-Instant, взаємодія з якою відбувається через інференс-платформою Groq Cloud за використання відповідного API-інтерфейса.

Практична значущість: розроблений інтелектуальний модуль автоматизованого редагування інтегровано у наявний редактор літературної веб-платформу Dragon Tale. Загальні архітектурні рішення та головні принципи, що лягли в його основу (human-in-the-loop, попередній контроль обмеження обсягу фрагменту, бібліотека готових шаблонів із структурованими запитами до LLM та механізм мітки прозорості), можуть слугувати готовим прикладом етичного застосування сучасних мовних моделей в межах літературних веб-платформ для спрощення процесу ручного редагування і підвищення загального рівня якості наявного на них контенту.

КЛЮЧОВІ СЛОВА: *ЛІТЕРАТУРНА ВЕБ-ПЛАТФОРМА, ХУДОЖНІЙ ТЕКСТ, АВТОМАТИЗОВАНЕ РЕДАГУВАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ, ВЕЛИКІ МОВНІ МОДЕЛІ, ІНТЕЛЕКТУАЛЬНИЙ МОДУЛЬ, ПРОЗОРІСТЬ ВИКОРИСТАННЯ ШІ, HUMAN-IN-THE-LOOP*.

АНОТАЦІЯ

Кваліфікаційна робота складається зі вступу, 4 головних розділів, висновку, списку використаних джерел та додатку.

Розділ 1 присвячений огляду особливостей, проблем сучасних літературних веб-платформ й існуючих інструментів автоматизованого редагуванням. Частину уваги приділено аналізу етичних аспектів застосування ШІ в літературній сфері, що підводить до потреби дослідження сприйняття контрольованого ШІ-редагування тексту аудиторією. Отримані результати проведеного опитування дозволили впевнитися у доцільності реалізації інтелектуального інструмента автоматизованого редагування та сформувати завдання на його розробку для платформи Dragon Tale.

Розділ 2 орієнтований на проєктування модулю з аналізом існуючої організації платформи, формалізацією функціональних і нефункціональних вимог, розглядом ключових архітектурних рішень і логіки роботи інструмента. У ньому описується формування структури бібліотеки варіантів редагування з її складовими одиницями – шаблонами, що містять системні запити до ШІ й інформативні складові для відображення, а також логіка виявлення наявності ШІ-відредагованого тексту, яка визначає встановлення мітки прозорості на рівні БД.

Розділ 3 містить перелік обраних технологій та описує 4 етапи практичної розробки модуля: клієнтської частини – попереднього контролю текстового фрагмента й складових багатокрокового модального вікна; серверної – бібліотеки шаблонів, формування запиту до ШІ та взаємодію з Groq API; тригерного захисту незворотності мітки прозорості на рівні БД; реалізації алгоритму визначення ШІ-відредагованого тексту при збереженні внесених змін.

Розділ 4 першочергово демонструє результати проведеного тестування: ручного функціонального – роботи модуля та алгоритму визначення наявності ШІ-відредагованого тексту в фінальному варіанті глави, а також модульного –

ключових технічних механізмів. Після тестів він містить пояснення фінального впровадження розробленого модуля до загальнодоступної версії веб-платформи Dragon Tale та аналіз результатів ефективності такої інтеграції. Додаткового у цьому розділі розглядаються напрямки подальшого розвитку реалізованого рішення.

ABSTRACT

This bachelor's thesis consists of an introduction, 4 main chapters, a conclusion, a list of sources, and an appendix.

Chapter 1 provides an overview of the features and challenges of modern literary web-platforms and existing automated editing tools. Particular attention is paid to analyzing the ethical aspects of the use of AI in the literary sphere, which leads to the need to study the audience's perception of controlled AI-editing text. The results of the survey confirmed the feasibility of implementing an intelligent automated editing tool and defined the requirements for its development for the Dragon Tale platform.

Chapter 2 focuses on the design of the module, including an analysis of the platform's existing architecture, the formalization of functional and non-functional requirements, and an examination of key architectural decisions and the tool's operational logic. It describes the formation of the structure of the editing options library with its constituent units – templates containing system requests to the AI and informative components for display, as well as the logic for detecting the presence of AI-edited text, which determines the setting of a transparency tag at the database level.

Chapter 3 contains a list of selected technologies and describes the 4 stages of the module's practical development: the client-side component – pre-checking of the text fragment and the components of the multi-step modal window; the server-side – the template library, forming a request to the AI, and interaction with the Groq API; trigger-based protection of the irreversibility of the transparency tag at the database level; implementation of the algorithm for detecting AI-edited text while saving the changes made.

Chapter 4 primarily presents the results of the testing conducted: manual functional testing – of the module's operation and the algorithm for detecting the presence of AI-edited text in the final version of the chapter, as well as modular testing – of key technical mechanisms. Following the tests, it provides an explanation of the final implementation of the developed module into the publicly available version of the

Dragon Tale web-platform and an analysis of the effectiveness of such integration. Additionally, this chapter discusses directions for the further development of the implemented solution.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API (Application Programming Interface) – прикладний програмний інтерфейс (інтерфейс прикладного програмування).

БД – база даних.

ШІ – штучний інтелект.

LLM (Large Language Model) – велика мовна модель.

LPU (Language Processing Unit) – спеціалізований мовний процесор (архітектура обробки мови з високою швидкістю).

JSON (JavaScript Object Notation) – текстовий формат обміну даними, заснований на JavaScript.

CSV (Comma-Separated Values) – текстовий формат, призначений для представлення табличних даних.

XML (Extensible Markup Language) – розширювана мова розмітки для зберігання та передачі даних.

HTML (HyperText Markup Language) – стандартизована мова розмітки документів для перегляду вебсторінок у браузері.

UI (User Interface) – користувацький інтерфейс.

MVP (Minimum Viable Product) – мінімально життєздатний продукт.

ВСТУП

Вже багато років сучасні літературні веб-платформи виступають основним осередком публікації художнього текстового контенту в мережі поза межами традиційних видавництв. Вони надають зручне місце для публікації, читання та пошуку різноманітних творів, а також, найголовніше, активну спільноту письменників і читачів, готову до контакту й обговорень у будь-який момент. Проте, попри всі зручності, однією з найвиразніших проблем таких платформ, навіть після років розвитку, все ще залишається процес редагування, який повністю покладається на авторів. Для більшості він є довгим, виснажливим і неприємний етапом роботи над твором, що часто супроводжується явищем «авторської сліпоти», значними витратами часу та обмеженістю вбудованих редакторів. Через відсутність будь-якої інтелектуальної допомоги безпосередньо в межах систем багато авторів або публікують тексти «як є» і більше не повертаються до них, або витрачають надмірну кількість сил на самостійне редагування. У результаті загальна якість контенту на платформах є дуже нерівномірною – від мінімально опрацьованих до професійних. Саме через це створення інтелектуального допоміжного інструменту для редагування, який працюватиме всередині редактора платформи, є дуже актуальним. З урахуванням поточного рівня розвитку великих мовних моделей саме підхід з їх використанням є одним з найбільш доцільних рішень, здатних підтримати процес авторського редагування без прямого втручання в його індивідуальний стиль.

Метою роботи є дослідити ефективність контрольованого використання великих мовних моделей у процесі редагування художнього тексту та на основі отриманих результатів спроєктувати, розробити й інтегрувати інтелектуальний модуль автоматизованого редагування до літературної веб-платформи. Для досягнення цього необхідно виконати такі завдання:

- провести аналіз сучасних літературних платформ та процесу редагування на них із урахуванням проблематики застосування доступних інструментів автоматизованого редагування;
- дослідити етичні аспекти застосування штучного інтелекту в сфері художнього письма;
- створити контрольований спосіб використання ШІ для редагування тексту без втрати авторського стилю;
- спроєктувати інтелектуальний інструмент ШІ-редагування з урахуванням необхідності підлаштування архітектурних рішень під існуючу організацію системи, до якої він інтегрується;
- розробити інструмент відповідно до визначених вимог і спроєктованого функціоналу;
- провести тестування створеного рішення та інтегрувати його до загальнодоступної версії Dragon Tale.

Об’єктом дослідження даної роботи є процес редагування художнього тексту в межах літературних веб-платформ. У той же час предметом дослідження виступають методи та способи контрольованого автоматизованого редагування художнього тексту на базі ШІ з урахуванням механізмів забезпечення прозорості його застосування.

Методами дослідження, які використовувалися в процесі роботи, стали:

- для аналізу сучасних літературних платформ, проблем редагування на них та обмежень існуючих інструментів автоматизованого редагування в контексті художнього тексту – методи теоретичного аналізу (аналіз, порівняння, узагальнення);
- для дослідження сприйняття якості контрольованого редагування тексту з використанням ШІ – метод онлайн-анкетування;
- для формалізації функціональних і нефункціональних вимог – метод системного аналізу;
- для побудови архітектури модуля, бібліотеки шаблонів, алгоритму встановлення мітки прозорості та загального макету

користувацького інтерфейсу – методи структурного та об’єктно-орієнтованого проєктування;

- для розробки та впровадження інструмента – методи програмної реалізації;
- для перевірки коректності роботи створеного рішення – методи тестування (функціонального та модульного).

Результатом виконаної кваліфікаційної роботи бакалавра виступає створений інтелектуальний інструмент автоматизованого редагування художнього тексту, інтегрований до вбудованого редактора платформи Dragon Tale, з механізмами попереднього контролю, уніфікованою взаємодією з Groq API та прозорістю застосування ШІ-відредагованого тексту. Створене рішення полегшує роботу з текстом, забезпечує контрольоване використання штучного інтелекту, не порушує авторського стилю та непомітно вписується в звичний процес редагування.

Наукова новизна одержаних результатів:

- інтеграція створеного контрольованого рішення автоматизованого редагування художнього тексту безпосередньо у вбудований редактор платформи.;
- використаний перелік архітектурних рішень дозволяє мінімізувати як неконтрольоване використання ШІ, так і можливості втрати авторської індивідуальності під час обробки.
- завдяки реалізованому алгоритму автоматичного визначення факту використання ШІ-редагування та механізму незворотної фіксації відповідної мітки прозорості на рівні бази даних забезпечено достатній рівень прозорості застосування LLM в межах системи;
- практично продемонстрована можливість етичного застосування ШІ в межах літературних веб-платформ завдяки контрольованому та прозорому характеру його використання.

Практичне значення одержаних результатів: розроблений модуль автоматизованого редагування інтегровано у наявний редактор літературної веб-

платформу Dragon Tale. Загальні розглянуті архітектурні рішення та головні принципи, що лягли в його основу (human-in-the-loop, попередній контроль обмеження обсягу фрагменту, бібліотека готових шаблонів із структурованими запитами до LLM та механізм мітки прозорості), можуть виступати готовим прикладом того, як етичне застосування сучасних мовних моделей в межах літературних веб-платформ може одночасно значно спростити процес редагування для авторів і підвищити загальну якість контенту.

Структура роботи містить 104 сторінки, 4 розділи, 27 рисунків, 9 таблиць, 1 додаток та 48 джерел.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Огляд сучасних веб-платформ для публікації художнього текстового контенту

Популярність сучасних платформ для публікації художнього текстового контенту стабільно зростає з року в рік. Вони об’єднують мільйони людей різних національностей, професій, інтересів та вподобань спільною любов’ю до літератури. У таких системах авторам надається повна свобода – вони самі вирішують, як, коли і в якому вигляді публікувати власні рукописи, без будь-яких обмежень з боку традиційного видавничого процесу. Одними із головних переваг цих платформ також можна назвати відкритий доступ до контенту на будь-який смак, зручна форма подачі та широкі можливості міжособової взаємодії.

Здебільшого саме згадані платформи у наш час стають першим місцем публікації для новачків, які тільки починають свій шлях художнього письма. Літературні веб-платформи надають стійкий фундамент для практики та набуття впевненості у власній творчості, пошуку першої аудиторії та однодумців.

Для багатьох читачів такі веб-системи багато років виступають затишним місцем комфорту, де занурившись у світ історій вони можуть відпочити. Платформи надають доступ до неймовірної кількості безкоштовного літературного контенту різних жанрів, обсягів і напрямів. Користувачі можуть не просто обирати твори, які відповідають конкретним запитам, а й взаємодіяти з ними через коментарі та оцінки. Така взаємодія робить процес читання приємнішим, а літературу – «живішою».

Навіть попри все різноманіття подібних веб-систем, їхня базова структура доволі одноманітна:

- користувач створює профіль, в якому публікує власні рукописи;
- твори характеризуються метаінформацією (тегами, жанрами, анотацією, коментарями автора та іншим) й складаються з глав;

- читачі класично взаємодіють з контентом: коментують, оцінюють та додають роботи до власних колекцій.

Але з таким класичним фундаментом, деякі платформи мають свої унікальні акценти:

- Wattpad – одне з найпопулярніших місць для публікації веб-новел і фанатських творів з найбільшою користувацькою базою. Ця платформа орієнтована здебільшого на серійну публікацію оригінальної прози. Її особливостями є зручний мобільний додаток й активна спільнота, яка активно коментує глави в реальному часі [1].
- AO3 (Archive of Our Own) – некомерційне онлайн-сховище насамперед фанатських робіт з відкритим ПЗ. Основною особливістю платформи є одна з найпотужніших систем тегування, що дозволяє здійснювати дуже параметризований пошук [2]. Також її вбудований редактор підтримує HTML-розмітку, яка дозволяє авторам у текстах навіть використовувати зображення.
- Webnovel – глобальна веб-платформа, зосереджена на публікації веб-новел у жанрах фентезі, романтики та сянься [3]. Також вона підтримує комікси, мангу, манхву та інших мальований сюжетний контент. Важливою її складовою є можливість монетизації творчості.
- BookNet – платформа самвидаву, що дозволяє письменникам безкоштовно розміщувати художні твори. Вона нерідко співпрацює з різними видавництвами для проведення літературних конкурсів. Також окрім сучасних робіт у відкритому доступі на ній знаходиться української класика [4].

З редагуванням на них все доволі обмежено – воно базується лише на візуальному форматуванні (виділенні тексту курсивом чи жирним, підкресленні та зміні вирівнювання). Навіть HTML-розмітка на AO3 не робить нічого окрім візуалу.

Платформи не надають авторам ніякої інтелектуальної допомоги, залишаючи їх самостійно «боротися» з текстами. Така самостійність неодмінно підкреслює унікальність і самобутність, але часто призводить невиправлених граматичних і стилістичних помилок. Це особливо помітно у роботах початківців і може дуже відлякувати читачів.

1.2. Особливості та проблеми процесу редагування художнього тексту авторами у межах літературних платформ

Для того, щоб розглядати специфіку редагування на літературних платформах, необхідно початково визначити, що саме мається на увазі під поняттям «редагування художнього тексту».

Редагування художнього тексту – це процес, який спрямований на контроль за дотриманням естетичних норм оформлення, виправлення помилок і покращення загальної якості тексту без суттєвого втручання в особливості авторського стилю [5]. Його умовно можна розділити на такі рівні:

- Лінгвістичний – передбачає вдосконалення з точки зору граматики, орфографії, пунктуації, лексики та екстралінгвістики задля дотримання базових правил мови [6].
- Стилістичний – спрямований на покращення загальної читабельності, усунення тавтологій та надмірного ускладнення речень, щоб зробити тон більш плавним й узгодженим [7].
- Композиційний – передбачає вдосконалення структури тексту, побудови сцен і забезпечення цілісності та плавності викладу [8].
- Семантичний – пов’язаний із забезпеченням правильності використання мовних засобів, усуненням логічних невідповідностей та посиленням емоційного впливу на читача [9].

У професійному книговидавництві редагування ніколи не є одноосібною роботою автора. Воно базується на командній взаємодії різних фахівців, до яких можуть входити: редактори, коректори, бета- та гамма-рідери. Така специфіка роботи забезпечує багаторівневу перевірку тексту, що майже у всіх випадках

мінімізує як технічні, так і логічні помилки, гарантуючи високий рівень фінального твору. Але сучасні літературні платформи такого комплексу фахівців (й інструментів) не мають, тому усі обов’язки щодо забезпечення якості тексту лягають виключно на автора, що накладає на нього значний тягар виконання кількох ролей одночасно.

Як уже було зазначено, більшість літературних платформ надають своїм користувачам лише базові засоби візуального редагування, тому воно зводиться переважно до зовнішньої обробки, а не фактичного вдосконалення. Платформи буквально залишають автора один на один зі своїм текстом, без будь-якої системної інтелектуальної підтримки. Саме це, поряд зі страхом технічних збоїв та втрати прогресу, є ключовою причиною того, чому багато користувачів нечасто користуються вбудованими системними редакторами, надаючи перевагу роботі в окремих текстових процесорах із подальшим копіюванням готового тексту на сайт.

У такій самостійній «боротьбі» з власним текстом з’являється ряд специфічних проблем та особливостей. Однією з найпоширеніших є так звана «авторська сліпота» – нездатність об’єктивно помічати стилістичні та граматичні проблеми у власному рукописі через тривалу роботу над ним [10]. Це призводить до того, що письменник зосереджується лише на тих аспектах, які вважає найважливішими (сюжет, персонажі, власний стиль), а інші проблеми, які очевидні для стороннього читача, ним ігноруються. Нерідко такі недоліки аргументуються унікальністю стилістичної самобутності: технічні та логічні недоліки тексту починають вважати частиною авторського стилю, що ще більше ускладнює об’єктивну оцінку якості твору. Проте зловживання такою аргументацією фактично призводить до зниження реальної якості.

Окремо варто виділити популярність своєрідної течію виправдання «без бета-рідера» – культуру публікації без попередньої вичитки та редагування. Автори на сучасних літературних платформах у примітках нерідко зазначають фразу на кшталт: «Без бета-рідера. Якщо помітите помилки – дайте знати!». Таким чином вони ніби заздалегідь намагаються зняти з себе відповідальність за

якість тексту, керуючись логікою «я ж попереджав». Така практика вже стала природною частиною платформ та елементом стилю спілкування авторів із читачами. Вона фактично нормалізує публікацію творів без попередньої якісної вичитки та редагування.

Якщо все ж таки письменник знаходить і залучає бета-рідера до свого рукопису, якість такого редагування все одно не гарантовано буде високою. Це зумовлюється тим, що на відміну від професійного середовища, такі бета-рідери зазвичай не мають спеціалізованої підготовки та достатнього досвіду. Вони можуть думати інакше, ніж автор оригіналу, редагувати «під себе», пропускати або навмисно ігнорувати окремі помилки, надмірно прикрашати або шаблонізувати текст, приводячи його до стандартних літературних патернів. Парадоксально, що таку людську шаблонність у редагуванні прийнято вважати нормою та ставитися до неї цілком адекватно, а в той же час аналогічне приведення тексту до шаблонів штучним інтелектом активно критикується спільнотою.

За своєю суттю літературні веб-платформи об’єднують дуже різних людей – із різним рівнем підготовки, сферою діяльності та досвідом. Вони містять творчі напрацювання як філологів і досвідчених письменників, так і технічних спеціалістів, школярів та людей без профільної гуманітарної освіти. Це формує середовище, яке поєднує тексти різної якості та стилю оформлення, де відчутно проявляється людська суб’єктивність – те, що один автор вважає «унікальним стилем», для іншого читача може виявитися просто абсурдом.

Також важливо пам’ятати, що редагування будь-якого тексту є дуже енергозатратним процесом як з точки зору часових рамок, так і з точки зору психологічних ресурсів людини. Якісне опрацювання навіть власного, невеликого за обсягом фрагмента (наприклад, 500-700 слів) може займати від 40 до 60 хвилин. Якщо ж ідеться про редагування тексту іншої людини, час значно зростає, адже необхідно спочатку зрозуміти, де авторський стиль, а де помилка. А якщо враховувати те, що середній розмір однієї глави на таких платформах становить приблизно 5000 слів, то час, необхідний на редагування всього твору,

досягає лякаючих значень. Саме тому нерідко письменники опиняються перед вибором: публікувати «як є» тут і зараз, щоб не втрачати власну мотивацію до письма та увагу читачів, які чекають на вихід продовження, або ж ідеалізувати якість тексту, витративши на це купу часу та нервів, і ризикувати втратити мотивацію, щоб опублікувавши матеріал, про який усі вже забули. І для багатьох цей вибір є дійсно очевидним – зберегти мотивацію та аудиторію набагато важливіше, а низьку якість завжди можна виправдати простою приміткою «без бети».

Не потрібно забувати й про те, що значна частина авторів таких платформ становлять початківці, для яких публікація своїх перших напрацювань є насамперед методом набуття досвіду та пошуку власного стилю, а не спробою створити ідеальний літературний продукт. Новачки можуть добре відчувати сюжет і персонажів, але при цьому недостатньо володіти технікою редагування. І це є цілком нормальним, проте суттєво впливає на рівень уваги до якості тексту.

Використання сторонніх інструментів для перевірки тексту ніким ніколи не заборонялося. Але часто для повноцінної роботи вони вимагають виконання певних умов або платної підписки, що для багатьох користувачів літературних платформ здається абсолютно зайвим. Також більшість із інструментів не враховує специфіку художнього стилю, працюючи за чітко визначеними патернами поведінки, й не інтегровані безпосередньо у процес роботи на платформах, через що часто ігноруються авторами.

Через усе це можна виділити наступний основний перелік особливостей та проблем, що стосуються процесу редагування художнього тексту в контексті сучасних літературних веб-платформ:

1. обмеженість вбудованого інструментарію виключно візуальним форматуванням;
2. нездатність автора об'єктивно помічати власні помилки («авторська сліпота»);
3. нормалізація культури «без бета-рідера»;

4. суб’єктивність та людська шаблонізації при залученні сторонніх помічників;
5. різний рівень освіти, знань та навичок користувачів платформ;
6. значні витрати часу та енергії на редагування;
7. постійний конфлікт між швидкістю публікації та якістю тексту;
8. переважання мотивації до самого процесу написання над виправленням помилок, особливо серед початківців;
9. обмеженість і неінтегрованість сторонніх допоміжних інструментів.

Саме через це виникає потреба у пошуку та впровадженні інструментів редагування, які будуть не тільки ефективно допомагати у роботі з художнім текстом, а й зніматимуть частину технічного навантаження з автора, не позбавляючи його повного контролю над власним рукописом.

1.3. Аналіз існуючих інструментів автоматизованого редагування тексту

Сучасна вибірка інструментів редагування тексту пропонує рішення на будь-які потреби – від класичних вбудованих коректорів найпоширеніших текстових процесорів до потужних спеціалізованих сервісів із використанням штучного інтелекту. Усі ці рішення за ступенем їхнього втручання у текст можна умовно поділити на 4 основні групи, кожна з яких має свої переваги й обмеження, особливо якщо розглядати їх у контексті літературних веб-платформ.

Представниками першої групи є класичні інструменти популярних текстових процесорів Microsoft Word та Google Docs. До них належать вбудовані засоби перевірки орфографії, граматики та пунктуації, що працюють на основі статичних словників і строго визначених мовних правил. Такі інструменти добре та ненав’язливо допомагають під час базовій роботі з написання будь-якого тексту у знайомому інтерфейсі, акцентуючи увагу на виявлених помилках і не вимагаючи обов’язкового їх виправлення. Загальний принцип взаємодії з ними є

дуже простим і зрозумілим майже кожному, що робить їх достатньо комфортними у повсякденному використанні.

Проте основною їх проблемою можна назвати те, вони не враховують контекст стилю, жанру чи авторської індивідуальності. Такі інструменти не розрізняють, чи є текст діловим листом, науковою статтею, заявою на розлучення або ж художнім фентезійним твором. Для них будь-який авторський елемент, нетипове словосполучення чи знак, що не відповідає стандартній «методичці», буде вважатися помилкою. Це робить такі інструменти не надто оптимальними у випадках творчого письма. Також очевидним є те, що їх пряма інтеграція в інтерфейс літературних веб-платформ є неможливою, тому вони залишаються лише допоміжним засобом на етапі підготовки тексту до публікації, а не її складовою частиною.

Інструменти другої групи – це універсальні мовні асистенти, такі як популярні нині Grammarly та LanguageTool, які використовують значно складніші алгоритми порівняно з класичними коректорами. Вони аналізують текст у контексті цілого речення чи абзацу та на основі цього не лише знаходять та пояснюють помилки, а й надають рекомендації щодо їх виправлення (наприклад, правильного вживання словосполучення, уникнення тавтологій чи перевантаження тексту займенниками). Для багатьох використання таких інструментів уже давно є звичною процедурою, особливо враховуючи можливість інтеграції їх у браузери та текстові процесори у ролі доповнення або повної заміни класичних коректорів.

Однак, попри всі зручності, вони все ще залишаються обмеженими алгоритмічними патернами та стандартними літературними конструкціями. Навіть з урахуванням того, що деякі з них використовують ШІ, нав’язування ними «правильних» формулювань без врахування авторської індивідуальності є суперечливим, особливо для художніх текстів. Такі інструменти більш підходять для написання грамотних електронних листів і курсових робіт, ніж для роботи з творчим письмом [11]. Також, варто також зазначити, що значна частина функціоналу доступна лише у платних версіях, що знижує їх доступність для

більшості авторів літературних веб-платформ. До того ж, ці інструменти безпосередньо не можуть бути інтегровані в інтерфейси платформ і не забезпечують фіксацію факту їх використання.

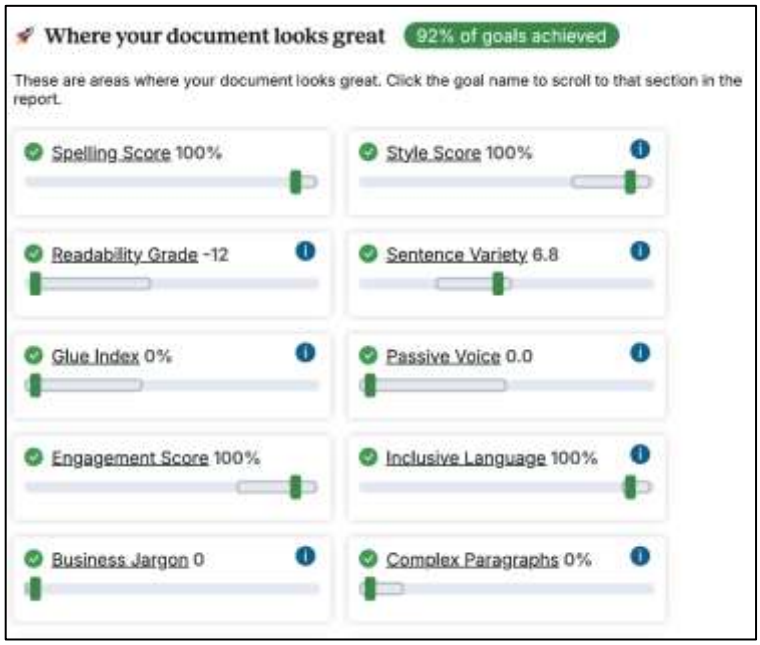


Рисунок 1.1 – Приклад частини оцінки художнього тексту за допомогою функціоналу ProWritingAid

До третьої групи належать спеціалізовані інструменти стилістичного аналізу, наприклад ProWritingAid. Вони направлені на комплексний аналіз (рис. 1.1) та оцінку якості тексту за безліччю різних параметрів (наприклад: рівень читабельності, довжина речень, рівень бізнес-жаргону тощо) задля проведення глибокого лексичного розбору. На основі отриманих даних такі інструменти формулюють персоналізовані рекомендації щодо покращення та редагування.

Проте саме така надмірна кількість оцінок і метрик зазвичай стає відлякуючим фактором для пересічних авторів, для яких письмо є передусім хобі. Різноманітні оцінки, графіки та відсотки викликають збентеження та нерозуміння, чого текст отримує такий низький бал за тим чи іншим параметром [12]. Загалом освоєння таких інструментів потребує чимало часу, що прямо суперечить бажанню швидкого редагування за принципом «раз і готово». До того ж, хоч би якими корисними та багатошаровими були такі інструменти, вони все

ще не повністю враховують унікальність авторського стилю. Наприклад, навмисне стилістичне рішення автора починати певну кількість речень підряд з однієї фрази задля емоційного нагнітання системою автоматично маркується як недолік і пропонується до виправлення.

Такі інструменти можуть бути корисними, але є досить профільними рішеннями, орієнтованими більше для тих, хто професійно займається чи планує займатися письмом. Вони потребують глибокого вивчення, тривалого звикання та готовності платити за розблокування повного потенціалу. Крім того, стандартно такі інструменти існують як окремі платформи та не передбачають будь-які інтеграції в інші системи.

Четвертою і останньою виділеною групою є інструменти генеративного співавторства та великі мовні моделі. Цю групу можна вважати найбільш технологічно просунутою, адже такі інструменти повністю базуються на використанні штучного інтелекту для допомоги. На відміну від усіх попередніх, ці системи не лише аналізують текст і пропонують виправлення, але й здатні генерувати новий контент, фактично виступаючи як співавтори.

До прикладу, сервіс Sudowrite позиціонується як інструмент для творчості, що може допомогти значно покращити та пришвидшити письмовий процес, а також подолати творчу кризу [13]. Проте на практиці більшість його функціоналу (переписування, мозковий штурм, покращення описів та інше) фактично зводиться до звичайної генерації нового контенту замість автора. Це відбувається попри заяви розробників: «Is this cheating? Sudowrite is a tool for creativity. It's not intended to write for you, but can help you…». Так остаточно стирається і без того тонка межа між редагуванням і генерацією нового контенту. Це несе за собою знецінення авторської праці, створює ризики видачі ШІ-тексту за людський, що неминуче веде до втрати авторського стилю через уніфікацію та шаблонізацію.

Майже все перелічене стосується і використання великих мовних моделей (ChatGPT, Claude Sonnet, Gemini, Grok тощо) для редагування, яке останнім часом набуло більшого поширення. Їх очевидною перевагою є максимальна

гнучкість: автор може самостійно обирати, що саме редагувати, за якими параметрами оцінювати текст та які аспекти покращувати.

Проте їх уявна простота може здаватися дуже оманливою. Ефективність використання LLM значною мірою залежить від здатності користувача правильно формулювати запити для отримання оптимального результату, якою володіє далеко не кожен. А за відсутності таких навичок робота з цими високоякісними інструментами зводиться до примітивних запитів по типу «відредагуй цей текс» у комплекті з великими обсягами даних. Це неминуче призводить до «з'їдання» авторського стилю, порушень сюжету та структури, шаблонізації та інших небажаних змін.

Звинувачувати самі платформи в цьому неможливо – вони лише надають потужний та добре спроєктований інструментарій, а відповідальність за його використання вже лежить на людині. Але етичні ризики такого підходу залишаються досить серйозними.

Також однією з ключових проблем залишається відсутність обмежень і неможливість ефективного виявлення та фіксації фактичного використання ШІ для генерації навіть частини тексту. Оскільки зазвичай процес редагування відбувається поза межами літературних платформ, контролювати його неможливо. Так звані «детектори штучного інтелекту» демонструють повну неефективність: вони або завищують відсоток «штучності» добре написаного людського тексту, або, навпаки, пропускають фрагменти, які дійсно були штучно створені [14].

Загалом усі інструменти цієї групи, попри свій значний потенціал, мають спільну суттєву проблему – відсутність дійсно працюючих механізмів прозорості та контролю. Це робить їх безконтрольне використання потенційно неприйнятним як для читачів, так і репутації платформ загалом.

Проведений аналіз існуючих інструментів автоматизованого редагування дозволяє чітко виділити основний перелік проблем, пов'язаних з їх використанням у контексті літературних веб-платформ:

- майже повна неможливість інтеграції безпосередньо до вбудованих редакторів веб-платформ;
- орієнтація переважно на встановлені правила, словники та алгоритмічні патерни;
- неврахування стилістичних і структурних особливостей художнього тексту та авторського стилю;
- розмита межа в ШІ-інструментах між редагуванням і генерацією нового контенту;
- відсутність ефективних механізмів контролю та прозорості використання ШІ;
- перевантаженість функціоналу та обмеженість безкоштовного доступу до нього.

Саме через ці фактори повноцінного й ефективного вирішення проблеми редагування художнього тексту в межах літературних веб-платформ, згаданої раніше, все ще не існує.

1.4. Етичні аспекти використання штучного інтелекту у редагуванні художніх текстів

Навіть зважаючи на весь розглянутий технічний потенціал інструментів автоматизованого редагування на базі ШІ, їх застосування у літературній сфері породжує значну кількість етичних питань. Вони стосуються не просто якості кінцевого результату, а і його сприйняття читачами й авторами. Також актуальним є страх шаблонізації через втрату творчої індивідуальності. Немаловажливим виступає й загальне негативне ставлення спільноти до штучного інтелекту навіть в якості допоміжного інструменту.

Значна частина критики щодо ШІ в творчому письмі небезпідставна. Найбільш обговорювані проблеми сьогодні зводяться до таких напрямків:

- Навчання моделей на неліцензованих даних – стосується великої кількості скандалів, коли під час навчання великих мовних моделей були використані творчі напрацювання без жодної згоди зі сторони

авторів. Особливо неприйнятним це є ще й через їх отримання здебільшого незаконним шляхом з піратських ресурсів. Така ситуація очевидно викликає обурення у літературній спільноти та формує базову недовіру до будь-яких інструментів на базі генеративного ШІ [15].

- Страх знецінення людської праці та втрати професії – базується на побоюванні письменників, що створення творчого контенту за допомогою алгоритмів буде більш економічно-вигідним. Це може відчутно знецінити людську працю. До прикладу, можна розглянути дослідження «The Impact of Generative AI on the Novel», проведене докторкою Клементиною Коллетт. У ньому 51% опитаних романістів вважали, що ШІ, ймовірно, зможе повністю витіснити їх зі сфери. Вони аргументували це неконтрольованим копіюванням їхніх творчих напрацювань для навчання моделей, які у подальшому здатні легко імітувати людське мислення та генерувати подібні тексти автоматично [16].
- Страх втрати індивідуальності – виникає через те, що велика кількість авторів боїться застосовувати ШІ навіть для простого редагування через шанси втрати стилю, шаблонізації тексту та загального зниження оригінальності. У випадку читачів це часто стосується об’єктивного небажання споживати «бездушний» машинний контент [17].

Якщо ж розглядати ставлення до ШІ безпосередньо у межах літературних веб-платформ, то воно вбирає в себе все найкритичніше. У них текст – це передусім спосіб унікального самовираження та міжособової комунікації. Через це навіть легке застосування ШІ – наприклад, для генерації ідей чи покращення формулювань – часто розцінюється як порушення неписаних правил спільноти або обманом читача, який прийшов за щирими людськими емоціями.

Більша подібних платформ пропонують власні політики та рекомендації щодо використання ШІ. Вони, як правило, забороняють повне використання

згенерованого контенту та видачу його за власний, але не перешкоджають помірному застосуванню допомоги моделей для мозкового штурму, генерації ідей або редагування [18].

Але, розглядаючи ситуацію ШІ в літературі загалом, можна побачити, що все негативне ставлення насамперед базується на узагальненому і спрощеному його сприйнятті виключно як інструменту створення контенту з нуля. Його застосування як допоміжного асистента майже не згадується. Саме це підводить до ключової проблеми – відсутності чіткого розуміння різниці між генерацією нового тексту та редагуванням уже існуючого.

Генерація тексту передбачає створення нового контенту замість автора: сюжету, персонажів та стилістичних особливостей. Під час цього процесу існує високий ризик відтворення навчальних даних, що може нести за собою несвідоме копіювання стилю інших авторів, прямого плагіату, втрати «людяності», породження галюцинацій та навіть використання дискримінаційних конструкцій [15]. Генерація повністю замінює людський творчий процес, тому видавання такого тексту за свій не просто порушує довіру, а й суперечить базовим принципам творчого письма.

На відміну від генерації, редагування передбачає взаємодію з уже існуючим авторським матеріалом. Воно стосується виправлення граматичних або пунктуаційних помилок, покращення читабельності або уточнення формулювань без прямого втручання до основного змісту та стилю. За умов коректного використання ШІ такий підхід не замінює людину і не стає співавтором, а виступає тільки професійним асистентом [19].

Розуміння різниці між цими двома підходами є ключовим для етичного використання ШІ у сфері творчого письма. Воно можливо дозволить хоча б трохи зменшити необґрунтовані упередження спільноти та розглядати LLM в якості ефективного інструменту, який може допомагати вдосконалювати текст без втрати його ідентичності.

Окремо варто згадати ще й про прозорість застосування ШІ. Дослідження демонструють, що як приховування факту використання моделей, так і його

пряме розкриття можуть негативно впливати на рівень довіри аудиторії [18]. При цьому ситуація стає суттєво гіршою, коли розкривається приховане застосування, адже рівень недовіри зростає у рази через відчуття прямого обману.

На професійному рівні (наприклад, у видавництвах та на платформах по типу Kindle Direct Publishing) станом на 2026 рік вже існує чітка вимога розкриття інформації щодо використання ШІ. У той же час любительські веб-платформи досі не мають сформованих єдиних правил, які б регулювали характер і ступінь залучення мовних моделей. Навіть за наявності загальних рекомендацій та всього негативного ставлення до повної генерації контенту, дієвих технічних механізмів визначення обсягу та характеру використання ШІ все ще не існує.

З відсутністю цих необхідних чітких правил деяка частина авторів продовжує приховувати застосування великих мовних моделей, видаючи згенерований контент за власний, а інша – уникає навіть просто згадувати їх допомогу при редагування через ризик негативної реакції спільноти. Така ситуація сильно підриває довіру між авторами та читачами, що складає фундамент подібних платформ. Варто ще й взяти до уваги, що якісне ШІ-редагування стає все важче відрізнити від повністю людського

Саме через усе згадане свідома демонстрація використання мовних моделей має стати нормою прозорості. Таке рішення дозволить відокремити чесних письменників, які бажають просто покращити свій рукопис, від тих, хто зловживає генеративним створенням творчого контенту.

Важливою частино етичного використання ШІ є збереження повного контролю зі сторони людини. Незалежно від ступеня його залучення, остаточна відповідальність за зміст, стиль і етичну коректність опублікованого твору завжди лежить виключно на людині. Моделі можуть галюцинувати, генерувати помилки, підтримувати упередження та дискримінацію, тому автор повинен виступати фінальним редактором і здійснювати критичну оцінку результатів.

Абсолютно неприйнятним є намагання виправдати наявність проблемного контенту в тексті звичайною «машинною» помилкою. Якщо автор не перевірив зміни, додав їх до свого твору й опублікував, відповідальність все одно несе саме людина, а не інструмент. З урахуванням цього використання ШІ як самостійного редактора без повторної перевірки людино є некоректним.

На основі всього аналізу можна впевнено сказати, що етичне використання штучного інтелекту в процесі редагування художнього тексту можливе лише за певних умов: чіткого розуміння різниці між генерацією та редагуванням, обов'язкового контролю з боку автора, якісної побудови запитів та забезпечення певного рівня прозорості для читачів. Тільки при виконанні усього цього ШІ може виступати у ролі етичного допоміжного інструменту вдосконалення тексту, а не ставати джерелом ще більших проблем у літературній сфері.

1.5. Дослідження сприйняття якості редагування тексту з використанням ШІ

1.5.1. Методологія проведеного дослідження

Для практичного підтвердження можливості використання штучного інтелекту в якості допоміжного інструменту в процесі редагування художнього тексту було проведено дослідження.

Його метою стало не тільки визначення здатності респондентів відрізняти текст, відредагований людиною, від тексту, відредагованого ШІ, а й – ефективність контрольованого використання великих мовних моделей у контексті редагування художньої прози.

За об'єкт дослідження було прийнято сприйняття респондентами відредагованих текстових фрагментів, а за предмет – різницю у сприйнятті варіантів редагування людиною та штучним інтелектом.

Методом дослідження обрано анонімне сліпе онлайн-тестування [20] через Google-форми. Такий підхід дозволив усунути вплив упереджень опитаних щодо

конкретних LLM та забезпечити об’єктивність оцінювання представлених текстових фрагментів.

Для проведення тестування було обрано оригінальний текстовий фрагмент художнього тексту обсягом 2 абзаци (138 слів). Його було відредаговано 5 способами:

- 1 спосіб – ручне редагування людиною з певним досвідом у цьому процесі;
- 4 способи – автоматичне редагування за використання популярних на момент проведення дослідження LLM: GPT-5.2, Claude Sonnet 4.6, Gemini 3.1 Pro та Grok 4.

Для забезпечення рівних умов експерименту до всіх моделей ШІ було поставлено один і той самий запит [21] з базовими інструкціями щодо редагування тексту (рис. 1.2).

Відредагуй поданий фрагмент художнього тексту. Виправ можливі граматичні, пунктуаційні та стилістичні помилки. Покращ читабельність і логічність викладу, зберігаючи авторський стиль і зміст. Не додавай нову інформацію та не змінюй основну ідею. Поверни лише відредагований текст без пояснень.

Рисунок 1.2 – Використаний запит для редагування

Загалом було створено 3 варіанти Google-форми, кожна з яких містила набір 5 відредагованих фрагментів у різній послідовності (табл. 1.1). Це було зроблено для уникнення ефекту порядку [22] та підвищення достовірності результатів.

Таблиця 1.1

Розподіл варіантів редагування у Google-формах

Порядок джерела редагування	Форми		
	1	2	3
1	Людське	GPT	Claude

2	GPT	Gemini	Grok
3	Claude	Людське	GPT
4	Gemini	Grok	Людське
5	Grok	Claude	Gemini

Опитування складалося з таких логічних блоків:

1. Визначення демографії – вік респондентів, гендерна самоідентифікація, сфера навчання чи професійної діяльності та частота читання художньої літератури.
2. Оцінка якості редагування – оцінювання респондентами кожного з фрагментів за 5-бальною шкалою Лайкерта [23] за параметрами: наскільки текст легко і приємно читається; чи відчувається авторський стиль як цілісний; чи помітні граматичні, стилістичні або пунктуаційні помилки. Також перед переходом до іншого фрагменту надавалася можливість залишити невеликий коментар («що сподобалося або не сподобалося в цьому тексті»).
3. Ключові питання – фінальні питання: «Який з фрагментів сподобався вам найбільше?» та «Який фрагмент, на вашу думку, відредагований людиною?». Також було поставлено питання щодо того, чи може ШІ замінити людину у редагуванні художнього тексту.

Такий перелік питань дозволив комплексно проаналізувати сприйняття якості редагування тексту різними групами респондентів з урахуванням їхнього віку, професійної діяльності та досвіду взаємодії з художньою літературою.

Участь в опитуванні була анонімною та на добровільній основі. Загалом у дослідженні взяло участь 36 респондентів, відповіді яких відносно рівномірно збиралися з 3 форм, що дозволило отримати достатню вибірку результатів для подальшого аналізу.

1.5.2. Аналіз отриманих результатів

Отримана вибірка, що складається з 36 відповідей, охопила віковий діапазон від 18 до 55+ років, основну частину якого складали люди 18-24 та 25-

34 років. Опитані представляли різні сфери діяльності, найпопулярнішою з яких була технічна – IT, інженерія, математика. За отриманими відповідями щодо рівня взаємодії з художньою літературою можна побачити, що лише 42% читають її на регулярній основі (щотижня або частіше), тоді як 30% майже не читає або взагалі не робить цього. Такий демографічний склад вибірки доволі сильно міг вплинути на оцінку якості тексту та авторського стилю.

Таблиця 1. 2

Середні бали оцінки відредагованих фрагментів за шкалою Лайкерта

Редагування	Легкість читання (сер. бал)	Авторський стиль (сер. бал)	Відсутність помилок (сер. бал)
Людське	3,86	3,64	4,25
GPT-5.2	4,03	3,61	4,17
Claude Sonnet 4.6	4,06	4,11	4,42
Gemini 3.1 Pro	4,22	3,69	4,33
Grok 4	4,14	3,92	4,17

На основі вирахованих середніх оцінок за шкалою Лайкерта (табл. 1.5.2.1) найкраще за параметром «легкість читання» респонденти оцінили редагування моделі Gemini, що може свідчити про її намагання зробити текст більш доступним для швидкого сприйняття. Також можна побачити, що фрагмент, відредагований Claude, отримав найвищі оцінки за критеріями «авторський стиль» та «відсутність помилок». Проте такий результат став доволі несподіваним, адже, на думку автора оригінального фрагменту, саме Claude найбільше порушив авторську стилістику, додаючи надмірно літературні вирази та конструкції, яких в оригінальному творі не було. Це дозволяє підкреслити, що недостатньо контрольоване редагування за допомогою ШІ може легко порушувати стиль автора, «навішуючи зайві прикраси». Для читачів це може сприйматися тільки збагаченням тексту, а для автора бути втратою своєї індивідуальності.

Найкраще відредагованим текстом, попри високі результати інших моделей, був обраний фрагмент, над яким працювала модель GPT, хоча й відрив від Claude становив лише 1 голос (рис. 1.3). Але загалом можна простежити, що текст, над яким працював ШІ, частіше подобався респондентами ніж той, над яким працювала людина. Це свідчить про високу якість редагування, виконаного сучасними великими мовними моделями.

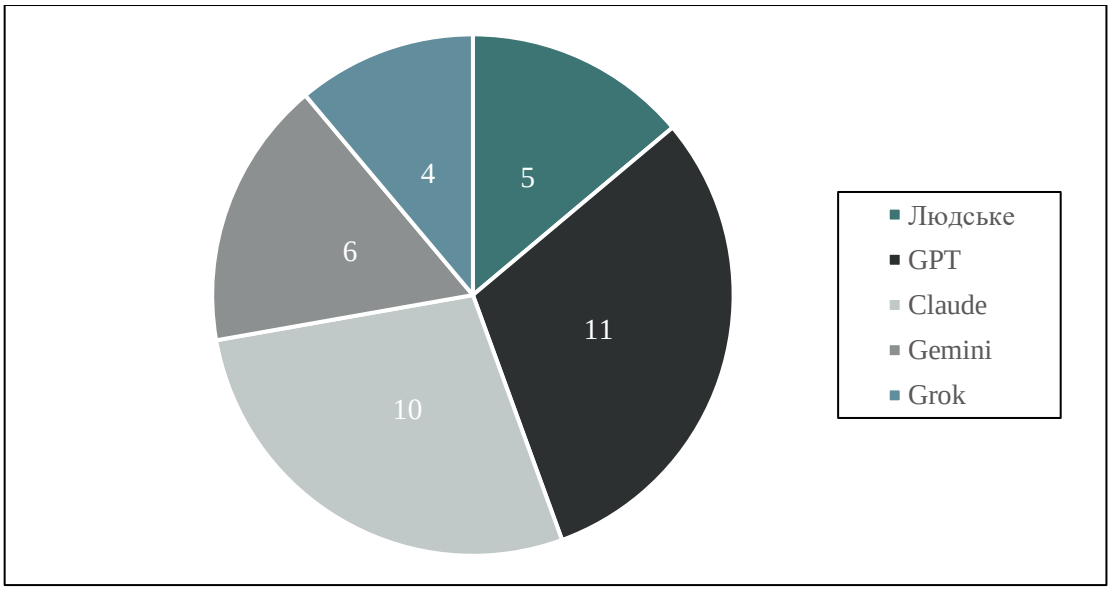


Рисунок 1.3 – Розподіл вибору за критерієм «найкраще відредагований фрагмент»

За результатами вибору опитаних, найбільшу «людяність» має фрагмент відредагований моделлю GPT – він був обраний 12 разів, значно переважаючи усі інші варіанти (рис. 1.4). А ось, справжнє людське редагування було ідентифіковано відчутно менше – лише 7 разів. Це дозволяє стверджувати, що люди все частіше сприймають справді якісне ШІ-редагування як більш природнє та близьке до ручного.

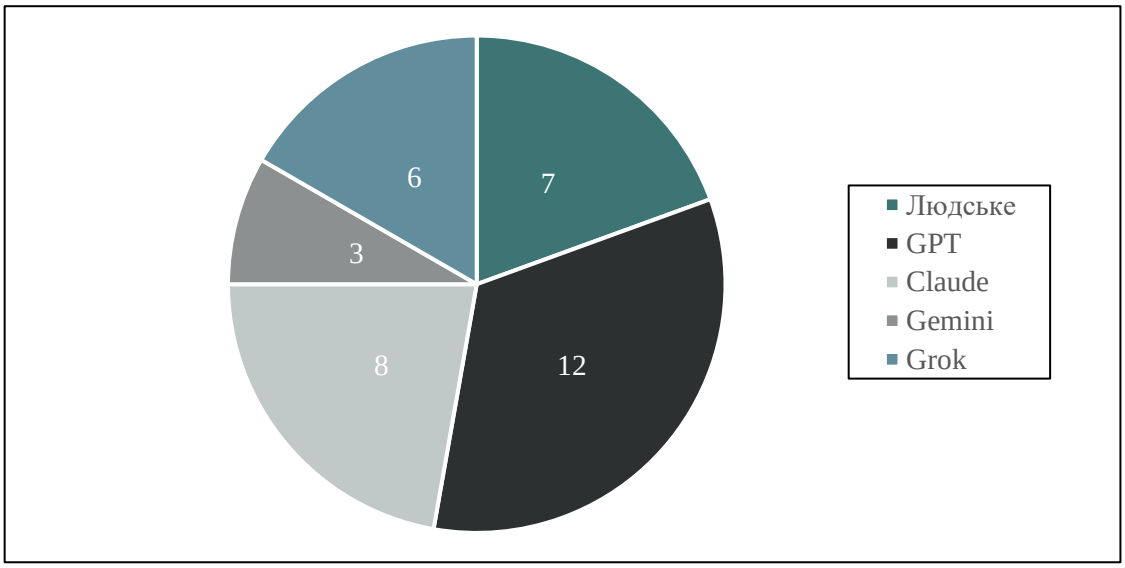


Рисунок 1.4 – Розподіл результатів вибору за критерієм «на мою думку, відредаговано людиною»

Під час аналізу, наданих деякими респондентами, коментарів до обґрунтування їх вибору «людського» фрагменту, можна побачити декілька очевидних шаблонів. Нерідко опитані обирали той чи інший варіант саме через наявність у ньому «людських» недоліків або, навпаки, через відчуття літературної «природності» та «рівності» тексту (табл. 1.3).

Такі коментарі чітко демонструють, що сприйняття «людяності» тексту є суб’єктивним і часто базується не на об’єктивних характеристиках, а на очікуваннях кожної людини. Тому наявність помилок буде вважатися ознакою редагування людиною, а «чистий» та «структурований» текст – сприйматися результатом машинної роботи.

Таблиця 1.3

Аргументація деяких респондентів щодо вибору фрагменту як «відредагованого людиною»

Джерело редагування	Аргументація респондента («Чому це людина?»)

GPT-5.2	«Наявність помилки, що часто допускається людиною».
Claude Sonnet 4.6	«Текст найбільш наближений до художньої стилістики... хоч і є слова русифіковані».
	«Спрощено».
Gemini 3.1 Pro	«Враження, що я читаю книгу, було більш вираженим».
Grok 4	«Текст виглядає більш стилістично вирівняним і природним, менше відчувається шаблонність».

За відповідями, отриманими на питання щодо перспектив використання ШІ в редагуванні художнього тексту, респондентів можна поділити на 3 групи:

- Прагматики (50%) – вважають, що ШІ зможе ефективно працювати з більш технічними аспектами редагування (граматикою, пунктуацією та структурою), але не з індивідуальним стилем та «душею» творів.
- Оптимісти (30%) – вірять, що LLM зможуть повністю витіснити людське редагування у майбутньому.
- Скептики (20%) – наголошують на тому, що через обмеження шаблонами ШІ не зможе ефективно працювати з унікальністю художнього тексту.

Отримані результати підтверджують доцільність використання штучного інтелекту в ролі допоміжного інструменту під час редагування художнього твору. Він здатен легко полегшити та прискорити роботу автора, але тільки в асистуючій ролі, з обов’язковим контролем зі сторони людини.

1.6. Постановка задачі на розробку інтелектуального модулю автоматизованого редагування

На основі результатів проведеного раніше аналізу предметної області чітко видно, що процес редагування у межах сучасних літературних веб-платформ супроводжується рядом суттєвих обмежень та труднощів. Існуючі інструменти автоматизованого редагування не мають можливості безпосередньої інтеграції в такі системи та не дозволяють ефективно вирішувати виявлені проблеми, а іноді

й взагалі їх використання супроводжується виникненням етичних ризиків. Водночас результати проведеного дослідження довели, шо застосування сучасних мовних моделей для редагування художнього тексту є ефективним і за якістю дуже наближено до людського, що підтверджує доцільність їх застосування. Саме тому виникає потреба у проєктуванні та розробці нових рішень, орієнтованих на допомогу авторам у межах подібних систем.

Прикладом такого рішенням може стати інтелектуальний модуль автоматизованого редагування, спроєктований, розроблений і впроваджений відповідно до потреб та особливостей нової літературної платформи Dragon Tale. Вона орієнтована здебільшого на авторів-початківців, підтримує багатомовну публікацію контенту та відкрита до використання сучасних технологій, зокрема штучного інтелекту, але лише за чітких обмежень. При цьому платформа виступає категорично проти публікації повністю згенерованих текстів, але передбачає надання авторам ефективного й етичного допоміжного інструменту, який стимулюватиме їх активніше працювати над якістю своїх рукописів безпосередньо у вбудованому редакторі.

Метою розробки такого інтелектуального модуля є надання письменникам зручного, швидкого, ефективного й етичного інструменту граматичного, стилістичного та лінгвістичного вдосконалення власного тексту безпосередньо у межах платформи. Таке рішення дозволить частково усунути ряд, виявлених під час аналізу, проблем.

Ключовою особливістю розроблюваного модуля має стати контрольований характер взаємодії з ШІ. Частиною цієї взаємодії є обмеження обсягу тексту, який надається на редагування, що дозволить не тільки забезпечити повний контроль, ітеративний підхід до роботи автора над текстом і уникнути надмірної автоматизацій, а й часткового запобігти проявам когнітивної інерції. Також, зважаючи на те, що ефективність використання великих мовних моделей здебільшого залежить від якості побудованого запиту, чого не вміє робити більшість аудиторії, взаємодія користувача з автоматизованим модулем має здійснюватися за чітко визначеними сценаріями.

Цими сценаріями можуть виступати готові шаблони редагування, що дозволять забезпечити різноманітні варіанти обробки та уникнути безпосереднього формування запитів користувачами та прямого контакту з ШІ.

Важливими аспектами роботи над модулем є забезпечення усвідомленої згоди користувача на обробку тексту із залученням ШІ та відкритості факту такої взаємодії. Використання інтелектуального інструменту не повинно приховуватися, що потребує реалізації механізмів прозорості. Вони дозволять ідентифікувати факт редагування тексту за допомогою ШІ саме як засобу підвищення його якості, а не повного створення. Це сприятиме запобіганню формуванню негативного сприйняття з боку користувачів та нормалізації розуміння різниці між редагуванням і генерацією контенту.

Так у межах даної роботи передбачається вирішення таких задач:

1. Проєктування, розробка, тестування та інтеграція інтелектуального модуля автоматизованого редагування на базі штучного інтелекту для платформи Dragon Tale.
2. Впровадження механізмів отримання чіткої згоди користувача на взаємодію з ШІ.
3. Реалізація обмеження обсягу тексту, що надається на редагування.
4. Розробка набору шаблонів редагування, орієнтованих на різні потреби авторів.
5. Забезпечення повного контролю зі сторони людини під час взаємодії з модулем.
6. Реалізація механізмів прозорості: автоматична фіксація факту використання інструменту редагування та ненав’язлива демонстрація цього факту.

Висновки до розділу 1

У цьому розділі було проведено комплексний аналіз сучасних літературних веб-платформ (Wattpad, Archive of Our Own, Webnovel і BookNet) та особливостей взаємодії авторів з ними у контексті редагування художнього

тексту. У його результаті було виявлено, що, попри зручність публікації та взаємодії з аудиторією, редагування тексту у межах цих систем все ще не дуже ефективне. Основними проблемами залишаються обмеження вбудованого інструментарію, «авторська сліпота», культура «без бета-рідера» та суб'єктивність сторонніх помічників. Також доволі сильно на процес редагування впливають великі витрати часу й енергії та постійний пошук компромісу між швидкістю публікації і якістю тексту.

Розгляд існуючих інструментів автоматизованого редагування (класичних текстових процесорів, мовних асистентів, спеціалізованих систем аналізу тексту та інструментів на базі штучного інтелекту) показав, що жоден із них не забезпечує ефективного вирішення проблем редагування художнього тексту у межах літературних веб-платформ. Більшість із них не інтегрується безпосередньо у такі системи, що ускладнює їх практичне застосування. Також використання частини інструментів, особливо на базі ШІ, несе за собою низку етичних питань, підняття яких розбурхує особливо чутливі теми у літературній спільноті.

Особливу увагу було приділено глибокому аналізу існуючих етичних аспектів використання штучного інтелекту та причин їх виникнення. Визначено різницю між генерацією нового контенту та редагуванням уже існуючого, що є основоположним принципом застосування ШІ як ефективного асистенту. Доведено необхідність впровадження механізмів прозорості використання великих мовних моделей, а також збереження повного контролю зі сторони автора та тільки його відповідальності за фінальний результат.

Додатково було проведено дослідження для порівняння сприйняття якості тексту, відредагованого великими мовними моделями та людиною. Отримані результати продемонстрували, що велика частина аудиторії не бачить різниці між текстом, над яким працювала людина, від тексту, обробленого ШІ. Це підтверджує, що правильне та контрольоване використання штучного інтелекту в процесі редагування художнього тексту є доцільним та ефективним.

На основі всіх отриманих результатів поставлено задачі на розробку інтелектуального інструменту редагування художнього тексту на базі ШІ, який повинен надавати допомогу для покращення якості тексту без втрати авторського стилю, бути інтегрованим безпосередньо до платформи Dragon Tale, реалізовувати механізми контролю та прозорості його використання. Правильна та етична реалізація такого інструменту дозволить хоча б частково вирішити ряд, виявлених раніше, проблем та підвищити загальний рівень якості контенту на літературних веб-платформах.

РОЗДІЛ 2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА ЛОГІКИ РОБОТИ МОДУЛЯ

2.1. Аналіз структури даних та існуючої організації платформи Dragon Tale

Перед початком роботи над проєктуванням інтелектуального модуля автоматизованого редагування необхідним є проведення аналізу існуючої організації літературної веб-платформи Dragon Tale. Через те, що розроблюваний інструмент стане лише функціональним доповненням уже існуючої системи, доцільним є дослідження її архітектури та структури даних. Це дозволить глибше зрозуміти внутрішню структуру роботи системи та визначити місце інтеграції модуля та механізмів фіксації використання штучного інтелекту.

Dragon Tale представляє собою відносно нову веб-систему, орієнтовану на публікацію художніх текстових творів і взаємодію з ними. Платформа надає весь необхідний базовий функціонал: публікацію, редагування, видалення, пошук, читання та групування творів. Важливою складовою системи є багатомовність – вона не тільки підтримує публікацію контенту різними мовами, а й має інтерфейс доступний українсько та англійською мовами.

Загальна архітектура системи (рис. 2.1) побудована за класичною клієнт-серверною моделлю. Клієнтська та серверна логіка реалізовані з використанням full-stack фреймворку Next.js та мови TypeScript, що забезпечує узгодженість і типізацію даних на всіх рівнях системи [24]. Зберігання даних та автентифікація користувачів відбувається на основі Supabase, яка поєднує реляційну базу даних PostgreSQL та ряд додаткових серверних можливостей [25].

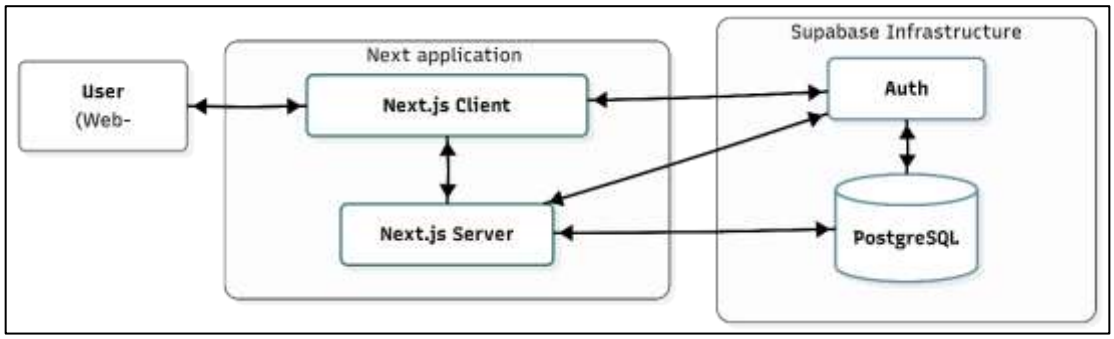


Рисунок 2.1 – Загальна архітектура системи Dragon Tale

На архітектурному рівні системи реалізується принцип багаторівневої валідації [26]. На боці клієнта інтеграція з сервісом автентифікації використовується для оперативного керування станом інтерфейсу, а на рівні сервера – для забезпечення строгого контролю доступу з перевіркою прав користувача на взаємодію з відповідними об’єктами бази даних (наприклад, при редагуванні характеристик або тексту твору). Це дозволяє поєднати зручність використання single page application із високим рівнем безпеки.

Існуюча організація даних платформи (рис. 2.2) містить набір сутностей, що забезпечують збереження інформації про користувачів, літературні твори, їхній вміст, систему тегів та заявок.

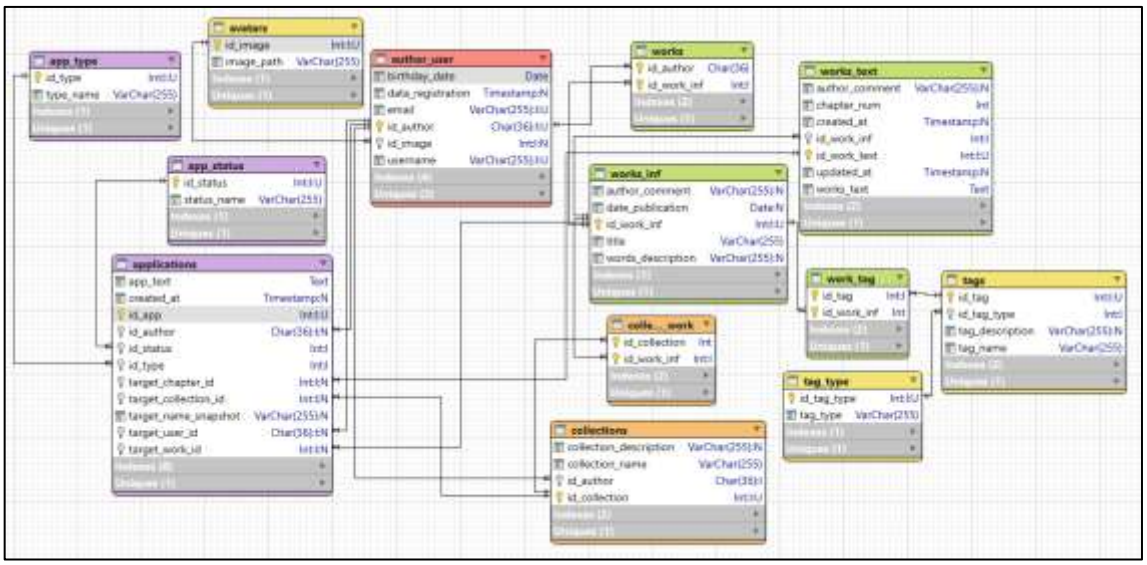


Рисунок 2.2 – Схема даних платформи Dragon Tale

Основною одиницею системи виступає сутність «твір», яка у БД реалізована через таблицю «works_info» та містить базову інформацію про нього (назву, опис та інше). Ця інформація (умовна «шапка» твору) виступає центральним елементом структури та пов’язана з автором («author_user»), характеристиками («tags»), главами («works_text») і колекціями («collections»). Безпосередній текстовий контент знаходиться на рівні глав і це визначає їх сутність ключовою для подальшої роботи з текстом, особливо у контексті впровадження механізмів маркування використання ШІ.

Внутрішня структура даних відповідно відображається у логіці інтерфейсу платформи:

- Створення «шапки» твору та його глав відбувається окремо, на різних сторінках, що забезпечує розмежування логіки, дозволяючи існування твору без жодної глави.
- Редагування інформації твору та текстового наповнення глав також реалізовано на окремих сторінках.
- Сторінка твору містить його загальну інформацію, характеристики, список глав і колекції, до яких він входить.
- Кожна глава відкривається окремо у, зручному для читання, режимі з можливістю налаштування візуального оформлення.

Внесення змін до текстового контенту глав відбувається у вбудованому редакторі платформи, доступ до якого обмежений відповідно до прав користувача. Кожен автор має можливість редагувати тільки власні творчі напрацювання, що забезпечує базовий рівень контролю доступу та захисту даних.

З урахуванням проаналізованої організації даних та реалізації інтерфейсу найбільш доцільним місцем для інтеграції інтелектуального інструменту є саме сторінка редагування глави, адже на ній відбувається безпосередня зміна текстового контенту, який виступає основним об’єктом обробки. Такий підхід робить можливим взаємодію з уже існуючим матеріалом, що дозволить забезпечити більш чіткий контроль змін і точніше відслідковування

використання інструменту. Його застосування прямо на сторінці створення глави є також можливим, але тоді відокремлення авторського внеску від результатів автоматизованого редагування буде доволі складним, що понизить забезпечення прозорості використання. Саме через це інтеграцію інструменту раціонально обмежити тільки сторінкою редагування.

Прозорість використання інструменту доцільно зробити у вигляді невеликої мітки з коротким поясненням про застосування ШІ саме для редагування, а не генерації. Застосування таких міток варто обмежити лише двома зонами інтерфейсу: на сторінці твору – у списку глав (лише для тих, що були відредаговані із застосуванням інструменту), а також безпосередньо на сторінках відповідних глав.

Відмова від розміщення загальної мітки на рівні всього твору (у «шапці») зумовлена прагненням уникнути перевантаження інтерфейсу та можливого негативного сприйняття твору як повністю згенерованого. Через це точкове та контекстне відображення має створити баланс між прозорістю, інформативністю та зручністю.

2.2. Формалізація функціональних та нефункціональних вимог до модуля

На основі визначених архітектурних особливостей платформи Dragon Tale та обраних точок інтеграції інтелектуального модуля автоматизованого редагування необхідною є формалізація вимог до його реалізації.

За допомогою функціональних вимог описуються конкретні можливості, що мають бути присутніми у системі для забезпечення роботи модуля [27]. Їх формалізація дозволяє чітко спроєктувати поведінку системи та її реакцію на дії користувача. Функціональні вимоги до інструменту:

1. Вибір текстового фрагмента – користувачу повинна надаватися можливість на вибір довільного текстового фрагмента для редагування.

2. Перевірка коректності тексту – повинна здійснюватися перевірка вибраного текстового фрагмента на відповідність встановленим вимогам (наявність дійсного тексту, а не випадкового набору символів).
3. Обмеження обсягу тексту – повинен обмежуватися дозволений обсяг, виділяємого на редагування, фрагменту тексту (1 абзац та обмежена кількість символів).
4. Отримання згоди користувача – повинно реалізовуватися явне отримання згоди користувача на надання тексту для редагування штучному інтелекту перед початком взаємодії з модулем.
5. Отримання списку шаблонів – повинно забезпечувати отримання та коректне відображення списку доступних шаблонів редагування.
6. Вибір шаблону редагування – користувачеві повинна надаватися можливість вибору шаблону редагування.
7. Автоматизоване редагування – повинно виконуватися автоматизоване редагування вибраного текстового фрагменту.
8. Порівняння результатів редагування – відредагований фрагмент повинен відображатися разом із оригінальною версією у режимі порівняння, що дозволить користувачу обирати той варіант тексту, який він хоче залишити.
9. Повторний вибір шаблону – повинна надаватися можливість повторного вибору шаблону редагування без зміни початкового тексту.
10. Відмова від редагування – повинна забезпечуватися можливість повної відмови від використання інтелектуального інструменту та повернення до звичайного редагування.
11. Підтвердження внесення змін – внесення змін до загального тексту глави змін повинно відбуватися тільки після підтвердження користувача.

12. Перевірка фінального тексту – повинна реалізовуватися перевірка фінального тексту глави на наявність змін, внесених під час використання автоматизованого редагування.
13. Встановлення мітки використання ШІ – мітка «Відредаговано з використанням ШІ» повинна автоматично встановлюватися у разі виявлення відповідних змін.
14. Відображення мітки використання ШІ – відображення мітки повинно забезпечуватися на рівні сторінки глави та «шапки» твору в списку глав.
15. Незалежність редагування від мови – редагування текстових фрагментів повинно відбуватися незалежно від мови їх написання.
16. Можливість звичайного редагування – повинна забезпечуватися можливість редагування без використання інтелектуального модуля.
17. Обробка помилок – повинно забезпечуватися інформування користувача у разі виникнення помилок під час роботи інтелектуального модуля.
18. Відображення стану виконання – стан виконання редагування повинен відображатися у вигляді індикатора.
19. Довідкова інформація – у веб-платформі повинна надаватися довідкова інформація щодо використання автоматизованого редагування.

На відміну від функціональних, нефункціональні вимоги описують властивості та умови роботи системи [27]. Їх формалізація дозволяє чітко визначити очікування щодо ефективності, стабільності та зручності використання системи в умовах експлуатації. Нефункціональні вимоги до інструменту:

1. Швидкодія – загальний час обробки запиту на редагування текстового фрагмента не повинен перевищувати 10 секунд у стандартних умовах.

2. Зручність використання – інтерфейс інтелектуального модуля повинен бути інтуїтивно зрозумілим для всіх користувачів.
3. Контроль користувача (human-in-the-loop) – повинен забезпечуватися повний контроль користувача над власним текстом та результатами його редагування.
4. Багатомовність інтерфейсу – інтелектуальний модуль повинен підтримувати багатомовний (англійський та український) інтерфейс із можливістю простого подальшого розширення.
5. Захист мітки використання ШІ – повинна забезпечуватися неможливість несанкціонованої зміни мітки «Відредаговано з використанням ШІ» після її встановлення.
6. Цілісність даних – збереження внесених змін повинно бути коректним, без втрат даних або їх спотворення.
7. Взаємодія з сервісами – повинна забезпечуватися стабільна робота системи під час взаємодії зі сторонніми сервісами.
8. Обмеження копіювання – повинна обмежуватися можливість копіювання результатів редагування.
9. Масштабованість – повинна забезпечуватися можливість розширення функціоналу без внесення суттєвих змін.

Весь цей перелік вимог чітко визначає основні аспекти й особливості роботи інтелектуального модуля та стає базою для подальшого його проєктування.

2.3. Загальна архітектура системи з урахуванням інтеграції модуля

Створення інтелектуального інструменту та його інтеграція до сторінки редагування глави не потребує повної перебудови або внесення значних змін до існуючої організації платформи Dragon Tale, проте передбачає її логічне розширення під потреби модуля. Це зумовлено на необхідності реалізації нового рівня взаємодії системи зі сторонніми сервісами обробки природньої мови.

Для повноцінної реалізації функціоналу автоматизованогоредагування необхідною є інтеграція з зовнішніми API великих мовних моделей. У якості такого сервісу може використовуватися, зокрема, Groq Cloud – спеціалізована інференс-платформа, яка побудована на архітектурі LPU (Language Processing Unit), що дозволяє запускати LLM зі значно вищою швидкістю [28]. Застосування подібного рішення забезпечить високий рівень продуктивності та низьку затримку, що є однією з головних виділених нефункціональних вимог.

Будь-яка взаємодія з зовнішніми ШІ-сервісами повинна відбуватися виключно на серверній стороні. У такому випадку сервер виступає центральним елементом – оркестратором між клієнтом і зовнішнім API. Це дозволяє надійно інкапсулювати ключі та системні інструкції (шаблони запитів), унеможливлюючи несанкціонований до них на клієнтському рівні. Також централізована обробка запитів дозволяє значно полегшити загальний контроль їх структури та логіки виконання.

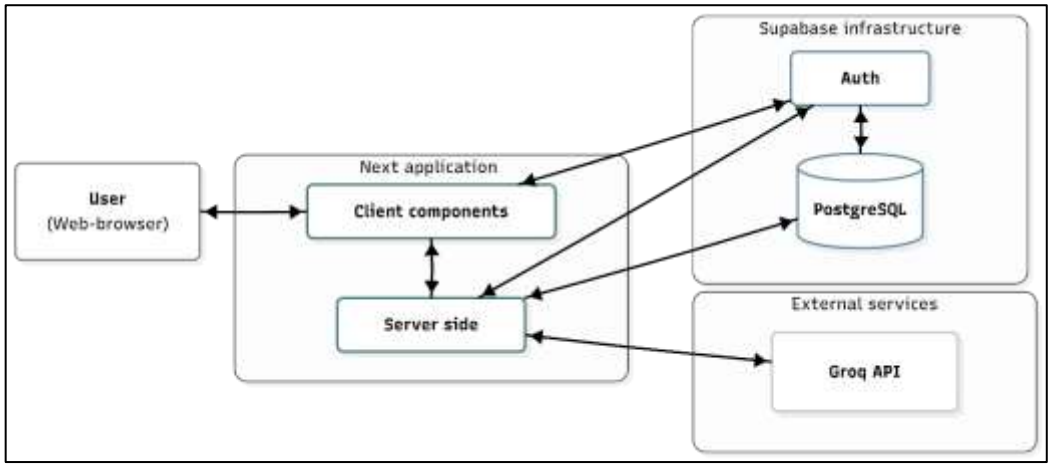


Рисунок 2.3 – Загальна архітектура системи Dragon Tale з урахуванням взаємодії з Groq API

Загальна архітектура, що поєднує існуючу структуру платформи із зовнішнім сервісом Groq API, представлена на рис. 2.3.

При такій схемі взаємодії клієнтська частина передає на сервер необхідний набір даних. На серверному рівні відбувається їх поєднання з системними

інструкціями для того, щоб передати вже повний запит до API. Результат роботи ШІ повертається до серверу в структурованому вигляді, після чого очищується та передається клієнту для демонстрації користувачеві.

2.4. Архітектура та внутрішня логіка модуля інтелектуального редагування

Для ефективної інтеграції інтелектуального інструменту необхідно поєднати певну кількість архітектурних рішень [29]. Вони повинні забезпечувати стабільність роботи та передбачуваність результатів редагування.

Побудова внутрішньої логіки модуля починається зі створення багаторівневого попереднього контролю вхідних даних – текстового фрагмента, що обирає користувач. Найпростішим варіантом реалізації такого механізму є жорстке обмеження обсягу виділеного тексту (наприклад, у діапазоні від 200 до 2000 символів). Таке рішення дозволить уникнути надмірного втручання моделі у великий об’єм авторського контенту.

Окрім обмеження обсягу, надаваємого на редагування, фрагменту, важливим механізмом попереднього контролю є перевірка його «чистоти». Для цього передбачається валідація тексту на відсутність випадкових символів, одиноких літер або надмірної кількості пробілів. Така перевірка на стороні клієнта дозволить уникнути некоректних запитів ще до їх відправки на сервер.

Активація елементу виклику інструменту (доступ до його використання) повинна відбуватися тільки у разі виконання всіх умов попереднього контролю. Завдяки цьому реалізований модуль вдасться зробити ненав’язливим, а також він не буде втручатися в звичний сценарій редагування без використання ШІ.

Наступним етапом побудови логіки є забезпечення усвідомленої взаємодії зі штучним інтелектом через механізм обов’язкової згоди на обробку даних. Впровадження цього моменту дозволить не тільки створити етичні умови застосування ШІ, а й запобігти неусвідомленому використанню відповідного функціоналу.

Підтвердження автором розуміння, що його текст буде наданий для опрацювання LLM, потребує отримання явної згоди. Без неї доступ до модулю надаватися не буде, що дозволить виключити можливість прихованого його залучення до процесу редагування. Таке рішення доволі важливе, адже воно створить чітку межу авторської відповідальності – використання інструменту на базі ШІ є свідомим вибором людини, а не прихованою функцією системи.

Отримана згода буде ініціювати перехід до вибору конкретного варіанту редагування з бібліотеки шаблонів, де автор обиратиме необхідний сценарій обробки тексту. Завдяки такому рішенню взаємодія з ШІ не вимагає від користувача самостійного написання промпту, що загалом уніфікує та значно полегшує цей процес.

Кожен шаблон у межах бібліотеки являє собою структурований набір даних, які використовуються як на клієнті, так і на сервері. Для коректного відображення та вибору в інтерфейсі він буде складатися з назви, короткого опису й унікальний ідентифікатора, що лаконічно пояснять автору суть та особливості конкретного типу редагування. Для логіки ж серверного рівня кожному ідентифікатору відповідає відповідний системний запит, який містить формалізовані вимоги до моделі щодо умов обробки тексту. Такий поділ дозволяє ефективно розмежувати доступність даних у системі, ізолюючи внутрішню логіку формування запитів.

Критичним моментом взаємодії зі стороннім сервісом є отримання результату в зручному для подальшої обробки вигляді (наприклад, JSON, CSV або XML). Таким підхід дозволить системі без помилок відокремити відредагований фрагмент тексту серед інших елементів відповіді та швидко інтегрувати його в користувацький інтерфейс. Відповідні вимоги до очікуваної структури відповіді будуть знаходитися у шаблоні та включати такі складові частини: відредагований текст, визначену мову та додаткові зауваження (за потреби). Завдяки цьому зменшиться вірогідність виникнення «дружніх» коментарів і пояснень моделі, які нерідко супроводжують базові текстові відповіді.

Після того, як автор визначився з варіантом редагування, клієнтська частина ініціює запит, передаючи на сервер мінімально-необхідний набір даних: ідентифікатор обраного шаблону та виділений фрагмент тексту. На серверному рівні реалізується механізм оркестрації запиту, після якої вже йде взаємодія з ШІ.

Під процесом оркестрації мається на увазі витяг системного запиту з бібліотеки за отриманим ідентифікатором шаблону, додавання до нього обраного фрагменту тексту та формування повного запиту до моделі. Після нього здійснюється передача сформованого промпту через API (також на рівні серверу). Таке рішення доволі важливе для безпеки, адже API-ключі та складні системні інструкції залишаються недоступними для клієнтської сторони, що унеможливлює їх перехоплення або модифікацію.

Користувацький інтерфейс відображає індикатор виконання, поки триває обробка та очікування відповіді. Це забезпечить безперервність користувацького досвіду та втримає увагу людини, сигналізуючи про стан обробки тексту.

Як тільки сервер отримує відповідь, то здійснює базову перевірку її цілісності. Якщо все нормально, він обробляє дані та передає їх на клієнт. Переданий структурований контент відразу готовий до моментального відображення.

Дуже важливим також є реалізація механізму human-in-the-loop [30]. Він передбачає використання штучного інтелекту виключно у ролі допоміжного інструменту із збереженням повного контролю та відповідальності за текст виключно за його автором. У межах модулю це забезпечується через можливість порівняння оригінального текстового фрагменту з ШІ-відредагованим, що дозволяє користувачеві самостійно робити вибір щодо подальшого використання отриманого результату.

Для цього необхідна реалізація середовища порівняння, де жодна зміна не вноситься в основний текст без прямого підтвердження. В інтерфейсі автору мають надаватися можливість прийняти запропонований варіант, відхилити його, змінити тип редагування або повністю відмовитися від використання інструменту на будь-якому етапі роботи. Завдяки цьому повний контроль

зберігається за письменником і він сам вирішує що і як робити з власним текстом, аналізуючи кожне запропоноване рішення.

Додатково, для підвищення прозорості використання ШІ-редагування та ускладнення приховування факту залучення інструмента, на рівні UI передбачається блокування прямого копіювання відредагованого тексту з етапу порівняння.

Останнім архітектурний рішенням стала перевірка фінального тексту в межах однієї сесії редагування з метою встановленням мітки прозорості. Вона спрацьовуватиме безпосередньо перед ініціалізацією запиту на збереження змін, що дозволить автоматизувати контроль за використанням інтелектуального інструменту та результатів його роботи.

Перевірка повинна відбуватися на клієнтській стороні задля високої продуктивності та мінімізації засмічення серверу та БД тимчасовими даними. Вона ґрунтується на порівнянні фінального варіанту всього тексту зі снапшотами фрагментів, отриманих раніше від моделі. Це потенційно дозволить ідентифікувати використання ШІ-відредагованого тексту навіть за умови його часткової ручної корекції.

У разі виявлення хоча б одного збігу до запиту на збереження буде додаватися спеціальна мітка «AI Edited». За такого підходу на сервер йде тільки фінальний текст глави, коментар автора та логічний прапорець, який означає застосування ШІ-редагування. Це значно спрощує подальшу логіку взаємодії з БД і не потребує додаткових перевірок.

Саме завдяки мітці «Відредаговано з використанням ШІ» на платформі досягається прийнятний рівень прозорості. Вона повинна чітко підкреслювати посил, який вкладений до неї: «редагування – не генерація контенту». Відкрита демонстрація повністю відповідатиме етичним принципам використання ШІ та водночас захищатиме репутацію авторів.

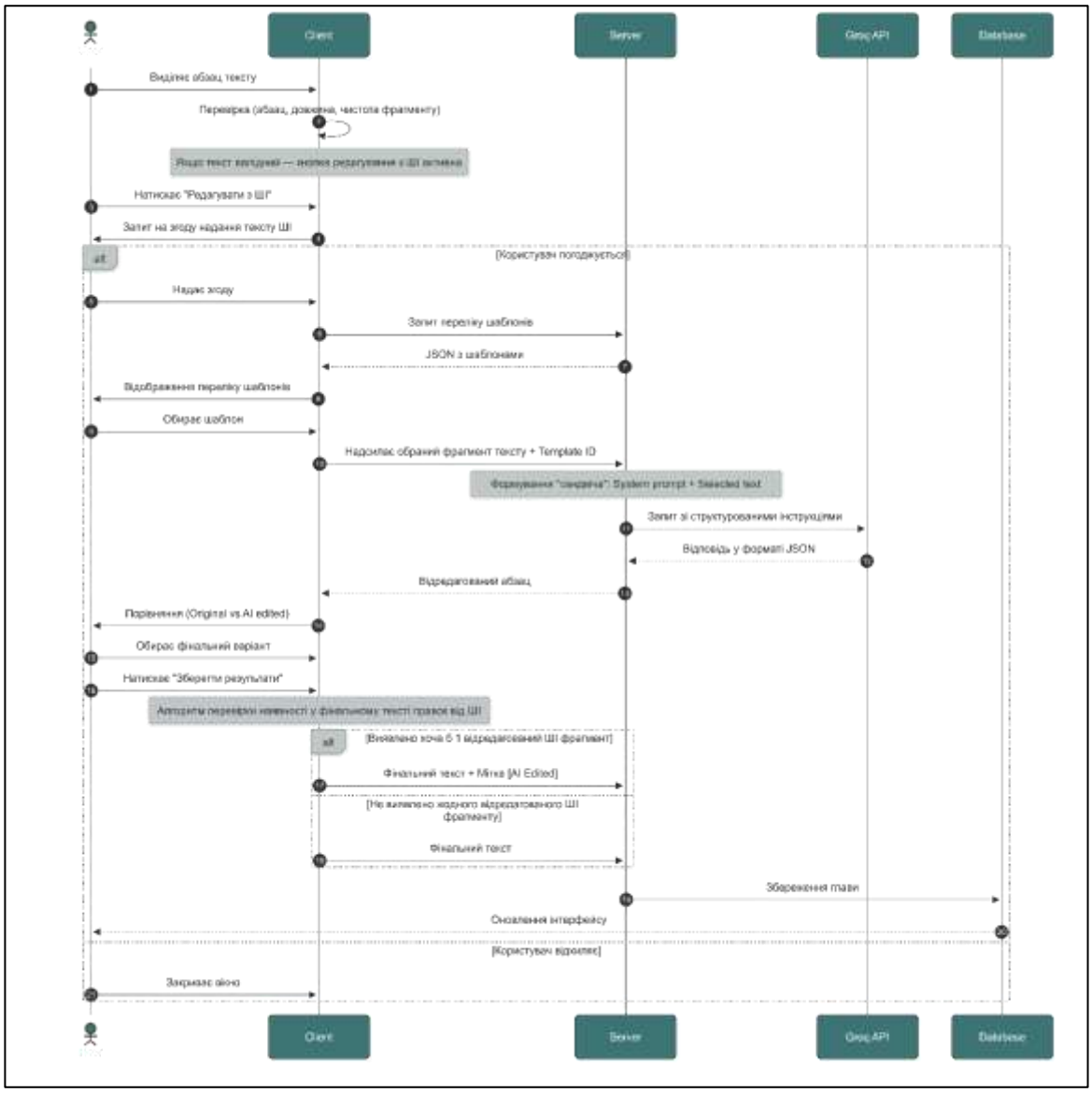


Рисунок 2.4 – Схема взаємодії компонентів модуля інтелектуального редагування

Гармонійне поєднання всіх цих архітектурних рішень формує цілісний та контрольований процес ШІ-обробки тексту в рамках внутрішнього редактора платформи. Схема взаємодії складових інструменту представлена на рис. 2.4.

2.5. Механізм фіксації використання ШІ на рівні бази даних

У архітектурних рішеннях було визначено необхідність фіксації факту застосування інструменту редагування для забезпечення прозорості, що реалізується за допомогою певної мітки. Вона формується відповідно до алгоритму визначення використання ШІ та виступає механізмом відображення

цього факту для користувачів. Через свою значущу роль, мітка потребує правильного та надійного зберігання у системі.

Враховуючи те, що основним об’єктом редагування на платформі та напрямом застосування інструменту є глава твору, найбільш доцільним місцем для зберігання мітки прозорості виступає таблиця бази даних «works_text». Таке рішення є раціональним ще й з точки зору того, що передбачене відображення мітки на сторінках платформи також прив’язане саме до рівня глави – на сторінці перегляду глави та в списку глав на сторінці твору.

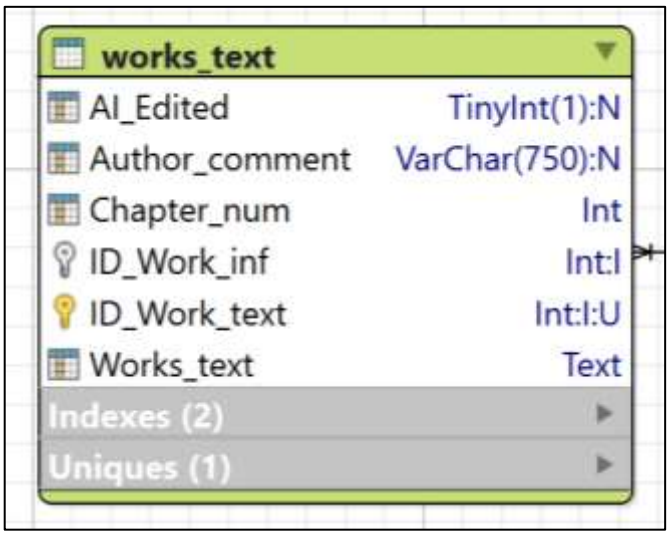


Рисунок 2.5 – Таблция «works_text» з доданим полем «ai_edited»

Для реалізації механізму фіксації мітки необхідним є розширення таблиці «works_text» новим булевим полем «ai_edited» (рис. 2.5). Логічний тип даних передбачає лише 2 стани:

- «FALSE» – дефолтне значення, що текст глави є суто авторським і до нього не було застосовано редагування на базі ШІ;
- «TRUE» – статус, що встановлюється системою у разі підтвердження факту використання інтелектуального модуля для обробки фрагментів тексту в межах поточної глави.

Однією з фундаментальних особливостей літературної платформи Dragon Tale є її позиція щодо перманентності маркування – факт використання

інтелектуального інструменту редагування тексту є незворотнім і не підлягає скасуванню.

Незважаючи на те, що архітектурні рішення модуля забезпечують те, що мітка не може з’явитися випадково або без свідомих дій зі сторони автора, існує теоретична можливість спроб обходу механізму прозорості. Користувач може зробити спробу вручну модифікувати дані, що йдуть на сервер (наприклад, через інструменти розробника у браузері), намагаючись приховати факт використання інструменту. З урахуванням цього постає нагальна потреба у реалізації механізмів захисту на рівні БД.

Ним може виступати механізм тригерного захисту, який буде спрацьовувати перед кожним оновленням запису в таблиці «works_text». Його логіка полягає у перевірці поля «ai_edited» перед виконанням оновлення та забороні зміни його значення зі стану «TRUE» на «FALSE». У разі такої спроби БД примусово зберігатиме значення у стані «TRUE», що забезпечує незворотність фіксації факту використання інтелектуального редагування.

2.6. Побудова логіки взаємодії користувача з модулем редагування

Ефективність використання інтелектуального інструменту залежить не тільки від коректності технічних архітектурних рішень, що лежать у його основі, а й від практичної організації взаємодії з автором, який ним користується. Особливу роль у побудові такої взаємодії, як і всієї архітектури загалом, відіграє створення послідовного, безперервного та контрольованого сценарію роботи над текстом з прозорим застосуванням штучного інтелекту, де людина зберігає керівну роль над внесенням змін до власного тексту.

Логіка такої взаємодії повинна природньо інтегруватися у звичний процес ручного редагування глави та водночас чітко відокремлювати його від використання допоміжного інструменту на базі ШІ. Через це логічно винести відповідний функціонал в окремий рівень інтерфейсу, наприклад у формат модального вікна. Його використання дозволить чітко відділити стандартну

роботу з текстом й автоматизовану обробку, а також сконцентрувати увагу користувача саме на конкретному етапі взаємодії.

Розглядаєма логіка реалізується у рамках основного сценарію, коли автор, перебуваючи в режимі редагування глави, ініціює застосування інтелектуального модуля для вдосконалення виділеного фрагмента тексту.

Все починається з виділення користувачем необхідного для редагування текстового фрагменту. На інтерфейсному рівні одразу здійснюється автоматична перевірка його відповідності до визначених вимог обсягу, кількості абзаців та чистоти тексту. При будь-якому відхиленні елемент активації інструменту залишається неактивним, а для розуміння відображається пояснення причини.

Після активації інструменту відображається модальне вікно, яке інформує користувача про те, що виділений текст буде передано для обробки зовнішньому ШІ-сервісу. У разі незгоди доступ до інструменту надаватися не буде.

При підтвердженні користувачу демонструється список доступних шаблонів редагування (рис. 2.6). Назви шаблонів у поєднанні з їхніми описами дозволять визначити оптимальний режим, що відповідає потребам виділеного текстового фрагменту. Обраний шаблон супроводжується відображенням його характеристик безпосередньо при перегляді.

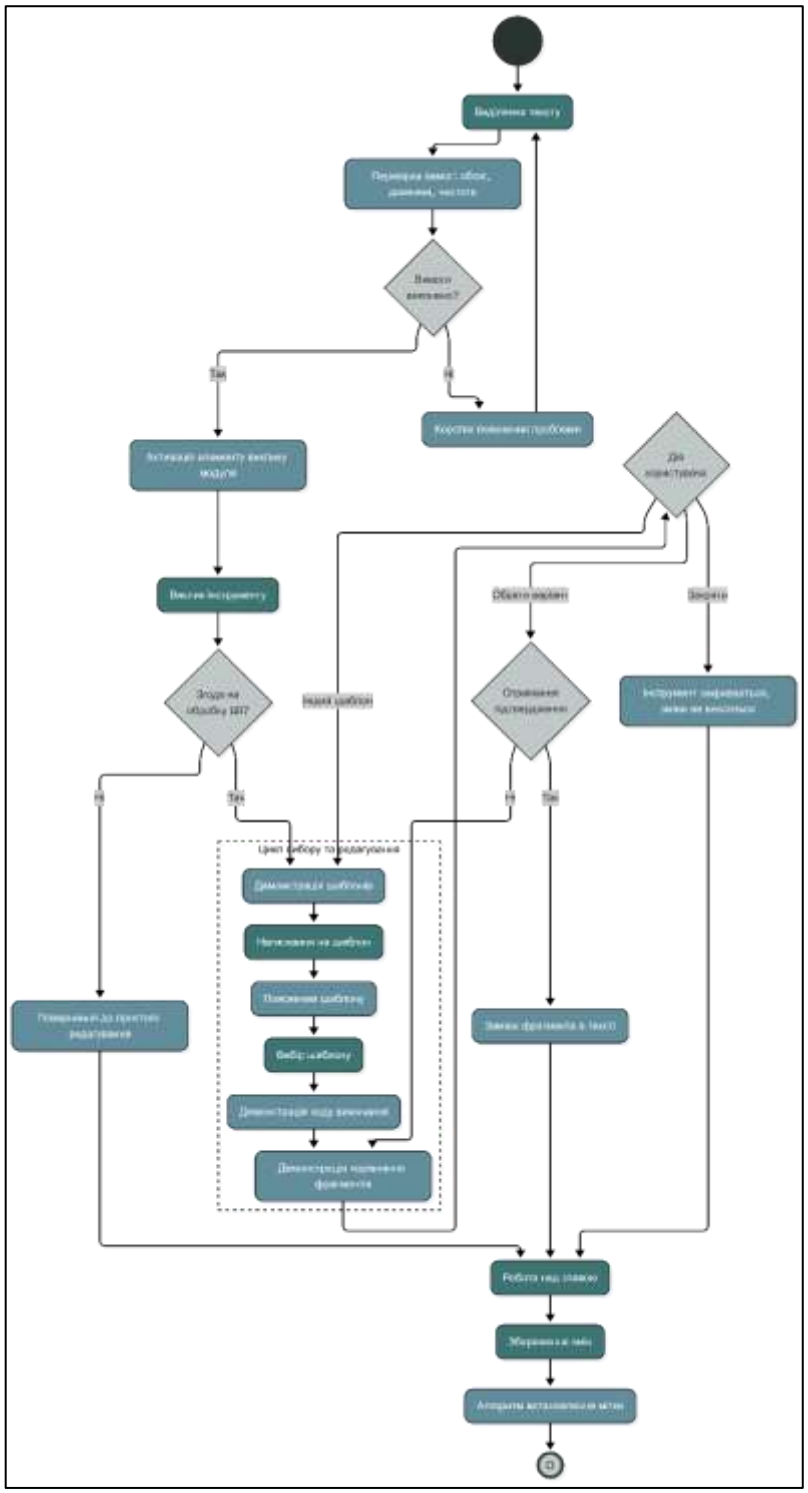


Рисунок 2.6 – Послідовність взаємодії користувача з модулем редагування

Вибір конкретного шаблону запускає процес редагування тексту, який на інтерфейсі супроводжується відображенням стану виконання операції, що втримає увагу користувача під час очікування результату.

Отриманий результат редагування буде відображатися поряд з оригінальним варіантом тексту для наглядного порівняння. Це дозволить користувачу оцінити внесені зміни, не замінюючи основний текст одразу. На цьому етапі також передбачається можливість повернення до вибору іншого шаблону, якщо отриманий результат не задовольняє автора.

Важливою особливістю взаємодії з інструментом є можливість виходу з нього на будь-якому етапі без внесення змін до тексту.

Для запобігання випадкової втрати варіантів тексту внаслідок помилкового натискання реалізується механізм підтвердження заміни фрагмента. Лише після отримання явної згоди автора обраний варіант буде інтегровано на місце оригінального фрагмента в загальному обсязі глави.

Після завершення автоматизованого редагування одного фрагменту користувач може продовжувати далі працювати над текстом або застосовувати його повторно до інших частин. Завершення роботи над главою відповідно переходить у збереження внесених змін, що за потреби буде супроводжуватися фоновим процесом аналізу для встановлення мітки використання ШІ-редагування.

2.7. Проєктування бібліотеки шаблонів редагування

2.7.1. Загальна структура шаблону

Побудова та використання бібліотеки шаблонів дозволяє стандартизувати процес роботи модулю редагування та усуває необхідність надання користувачу можливості самостійно писати запити до великої мовної моделі. Застосування вже готових шаблонів з професійно побудованими промптами забезпечить передбачуваність результатів обробки тексту та підвищить загальний рівень контролю над процесом.

Шаблони всередині бібліотеки є структурованими об’єктами єдиної структури. Вони інкапсулюють логіку конкретного сценарію обробки тексту, гарантуючи багаторазовість його застосування з передбаченим результатом. У

такому випадку забезпечується неймовірна зручність як підтримки та зміни наявних шаблонів, так і подальше додавання нових. Також передбачена структура об’єктів дозволяє поєднати дані різних частин веб-системи: для клієнта ними виступає інформація, що необхідна для інтерфейсного відображення, а для сервера – критичні системні інструкції, які використовуються під час формування запиту до моделі. Цю структуру складають такі поля:

Структура шаблону включає такий перелік обов’язкових полів:

- «id» – унікальний числовий ідентифікатор;
- «key» – унікальний строковий ідентифікатор;
- «label» – назва шаблону;
- «description» – опис функціонального призначення шаблону;
- «system_prompt» – найважливіша частина шаблону, яка містить структуровані інструкції для великої мовної моделі.

Беручи до уваги двомовний (український та англійський) інтерфейс веб-платформи Dragon Tale з подальшими можливостями розширення кількості підтримуваних мов, текстові поля «label» і «description» підлягають локалізації. Базово вони зберігаються у шаблоні англійською мовою, що реалізовано як механізм перестраховки у разі збою локалізації або відсутності перекладу. Це дозволить інструменту редагування зберігати працездатність та зрозумілість незалежно від стану локалізаційних ресурсів. Динамічна зміна назви й опису реалізовуватиметься на рівні компонентів інтерфейсу, де за значенням «key» буде здійснюватися підстановка відповідного перекладу зі словників.

2.7.2. Методологія формування структурованих запитів

Якість будь-якої отриманої відповіді від великої мовної моделі напряму залежить від якості сформованого до неї запиту. Особливо це помітно під час використання ШІ у рамках редагування художнього тексту. У такому випадку модель повинна не створювати щось нове, а коректно модифікувати вже

існуючий контент, зберігаючи при цьому авторський стиль, зміст та унікальне поєднання художніх засобів.

Художній текст, на відміну від формалізованого, технічного чи наукового, характеризується високим рівнем індивідуальності, що потребує специфічних підходів до роботи з ним, особливо у процесі редагування із застосуванням ШІ [31]. Таке редагування передбачає не повну перегенерацію, а максимально точне та обережне втручання.

Існуючі методи prompt engineering (англомовні інструкції, role prompting, negative instructions, structured output, hard/soft control) глибоко досліджені та технічно обґрунтовані, проте майже всі вони орієнтовані на генерацію тексту з нуля (creative writing). Через це у межах інтелектуального інструменту редагування веб-платформи Dragon Tale ці підходи не будуть використовуватися у чистому вигляді, а будуть адаптовані та поєднані з урахуванням потреб контрольованого редагування.

У процесі проєктування бібліотеки шаблонів основна увага приділяється виключно роботі з художньою прозою, яка є найбільш поширеним типом контенту на літературних платформах. На відміну від поезії, де форма тексту з його ритмом і римою є жорсткою основою твору, вона дозволяє легко та гнучко застосовувати методи автоматизованого редагування без ризику руйнування художньої структури за умови коректного використання підходів редагування.

У розглядуваній системі запити до LLM будуються за такими ключовими принципами:

- Мова побудови запитів – інструкції для моделі формулюються виключно англійською мовою, адже сучасні великі мовні моделі демонструють вищу точність виконання складних запитів саме в цьому мовному середовищі через переважання англомовних даних у процесі попереднього навчання [32]. Сам же текст для редагування подається мовою оригіналу, а у самому запиті додатково фіксується вимога щодо її збереження.

- Поділ запиту – кожна інструкція складається з фіксованої частини, що містить загальні правила роботи з художнім текстом, та змінної частини, яка визначає конкретний тип редагування.
- Структурований вивід – у запиті задається вимога щодо повернення відповіді виключно у форматі валідного JSON-об’єкта [33]. Це дозволить забезпечити надійний парсинг на стороні серверу та чітко відокремити відредагований текст від додаткових «дружніх» коментарів моделі.

Поле шаблону «system_prompt» має єдину логічну структуру, яка складається з:

1. Role-prompting– моделі призначається роль «досвідченого літературного редактора», що дозволяє активувати лінгвістичні патерни, орієнтовані на роботу з художнім текстом (наприклад, «you are a professional literary editor with 20 years of experience in editing prose») [34].
2. Negative instructions – перелік обмежень, що задає рамки небажаної поведінки моделі та мінімізує шанси неконтрольованої генерацією контенту (наприклад, «don’t change the plot» або «don’t add your own descriptions»).
3. Soft control – інструкції, які регулюють збереження стилістики: авторського голосу, настрою, емоційного забарвлення та іншого (наприклад, «preserve the original author’s voice, rhythm, unique linguistic features and emotional tone as much as possible»).
4. Hard control – точний опис конкретного типу редагування (наприклад, «fix grammar, punctuation and spelling errors») [35].
5. Language instruction – принцип визначення мови вхідного тексту та використання її для відповіді (наприклад, «the output must be in the same language»)

6. Output format – чітка вимога до формату відповіді, що визначає її структуру заданій структурі (наприклад, «response must be a valid JSON object with the following structure…»).
7. Output rules – обмеження щодо вмісту відповіді, які зменшують вірогідність проявлення «дружнього» супроводу (наприклад, «return only the JSON object).

Усі складові цієї структуру у поєднанні дозволяють досягти необхідного рівня фінальної якості результату та контрольованості ШІ-редагування.

2.8. Проєктування алгоритму визначення використання ШІ

Прозорість використання інструменту виступає важливою складовою політики платформи Dragon Tale. Через це для її забезпечення необхідне створення ефективного механізму для визначення самого факту наявності ШІ-відредагованого тексту у фінальній версії глави. Він повинен працювати на етапі збереження та приймати рішення про встановлення мітки на основі порівняння результатів роботи модуля, отриманих протягом поточної сесії, з остаточним варіантом глави.

Фундаментом запропонованого механізму виступають так звані снапшоти – ШІ-відредаговані фрагменти тексту, отримані від моделі протягом однієї сесії роботи над главою. Снапшот створюється тільки у разі успішного використання інструменту (тобто отриманий ШІ-фрагмент стає доступним для подальшого порівняння). Це відбувається незалежно від того, чи інтегується він у подальшому до основного тексту. Таке рішення дозволяє хоча б трохи мінімізувати ризики обходу системи, коли автор може використовувати результати ШІ-редагування поза межами інтерфейсу порівняння для уникнення встановлення мітки (наприклад, через скріншот або ручне введення).

Спроєктований механізм визначення наявності ШІ-відредагованого фрагменту в фінальному варіанті загального тексту базується на:

- запуску перевірки тільки тоді, коли глава ще не має мітки «ai_edited» у значенні «TRUE»;
- здійсненні обробки тільки в тому випадку, коли є снапшоти, що підтверджують сам факт успішного застосування модуля;
- встановленні мітки за першого ж виявлення позитивного збігу та відповідному перериванні роботи алгоритму;
- двоетапній нормалізації та декількох перевірках для точного визначення наявності снапшоту або підправленого його варіанту у фінальній версії тексту;
- визначенні часткової схожості за можливих змін, які може внести автор.

Завдяки нормалізації фінальний текст глави та снапшотів підготовлюється до подальшого порівняння. Передбачено застосування 2 її рівнів:

- м’якої нормалізації, коли текст зводиться до нижнього регістру, очищується від зайвих пробілів та зальотних HTML-тегів для проходження першої фази перевірки;
- жорсткої нормалізації, за якої прибираються розділові знаки і символи, що не є літерами та числами у форматі Unicode, для більш комплексної другої фази.

Першою фазою є базова перевірка – пошук точного входження снапшоту до загального обсягу тексту. Завдяки ній відбувається проста фіксація прямого використання ШІ-редагування без внесення будь-яких змін (наприклад, у разі цільової зміни лише пунктуації). Якщо перевірка цього етапу точного збігу не виявила, то відбувається перехід до другої фази, яка базується на оцінці ступеня схожості.

Для розрахунку такої оцінки було обрано метод шинглів, який традиційно застосовується в процесі пошуку плагіату та копій у нормалізованих текстах [36]. Для алгоритму, що реалізується, цей підхід адаптовано для визначення факту використання ШІ-результатів шляхом аналізу подібності текстів.

Такий метод передбачає розбиття як загального тексту глави, так і снапшотів на шингли – послідовні фрагменти заданої довжини (обрано n = 3). Для оцінки присутності ШІ-фрагментів використовується коефіцієнт перекриття (2.1), що визначає частку шинглів снапшота, які збереглися у фінальному тексті.

$$O(A,B) = \frac{|\ A \cap B\ |}{|A|} \tag{2.1}$$

A – це множина шинглів снапшоту, а B – множина шинглів тексту глави [37]. Пороговим значенням схожості обрано 45%. Таке рішення є компромісом між чутливістю до внесених змін та стійкістю до часткового редагування.

У разі виконання всіх попередніх умов (відсутність встановленої мітки та наявність переліку снапшотів) запускається алгоритм для визначення необхідності встановлення мітки. Все починається м’якої нормалізації та пошуку прямого входження. При негативному результаті базової перевірки відбувається перехід до більш глибокого аналізу на основі жорсткої нормалізації та розрахунку коефіцієнту перекриття. Таким чином здійснюється обробка кожного зафіксованого снапшоту і тільки тоді, коли жоден з етапів алгоритму нічого не виявив, мітка не встановлюється. В інших випадках – мітка «Відредаговано з використанням ШІ» незворотно додається до поточної глави.

Загальна логіка роботи алгоритму на рис.2.7.

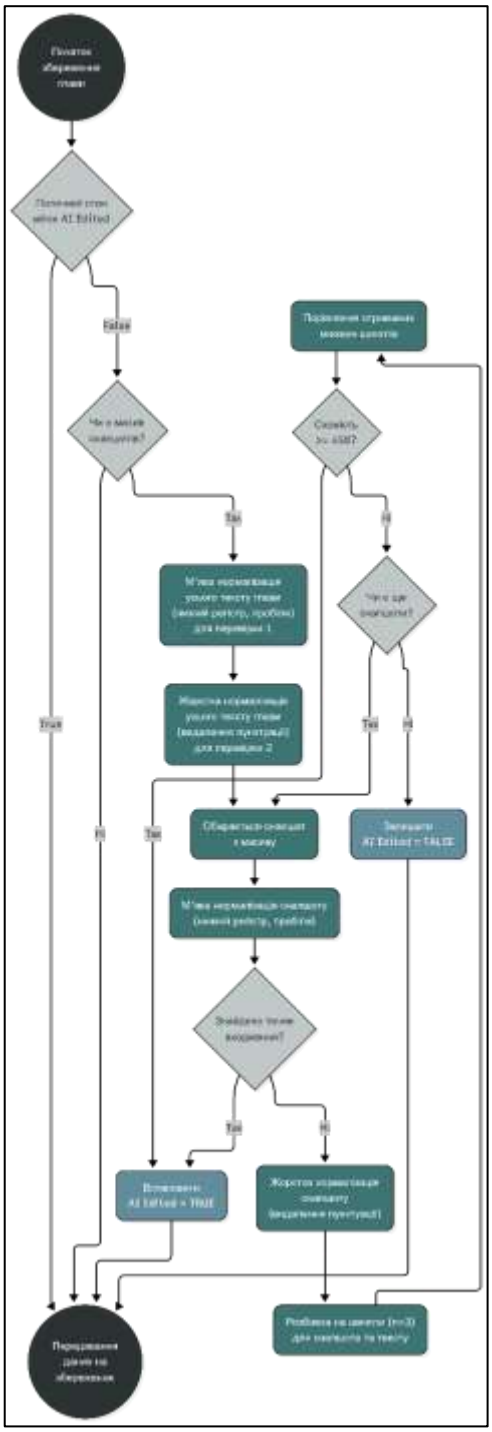


Рисунок 2.7 – Алгоритм визначення стану мітки

2.9. Проєктування користувацького інтерфейсу модуля

Проєктування користувацького інтерфейсу модуля автоматизованого редагування є завершальним етапом роботи над плануванням архітектурних і логічних рішень, розглянутих до цього.

Точкою входу до інструменту виступає інтерактивна кнопка, що знаходиться у межах сторінки редагування глави (див. додаток А рис. А.1). Вона стає активною лише у разі виділення користувачем коректного текстового фрагменту. В іншому випадку (виділення більше одного абзацу, перевищення ліміту символів або порожній текст) кнопка залишається неактивною, а при наведенні на неї з’являються контекстні підказки з причиною неможливості використання інструменту.

Перед доступом до самого функціоналу інструменту користувач обов’язково надає згоду у відповідному модальному вікні (див. додаток А рис. А.2), підтверджуючи розуміння того, що виділений текст буде передаватися на обробку великою мовною моделлю.

Основна частина взаємодії з модулем відбувається через багатокрокове модальне вікно, яке можна закрити на будь-якому етапі з поверненням до попереднього кроку або повною відмовою від використання інструменту.

На першому кроці користувачеві пропонується обрати шаблон редагування із доступної бібліотеки, який відповідає конкретним потребам виділеного текстового фрагмента (рис. 2.8). Кожен шаблон супроводжується назвою та коротким описом функціонального призначення.

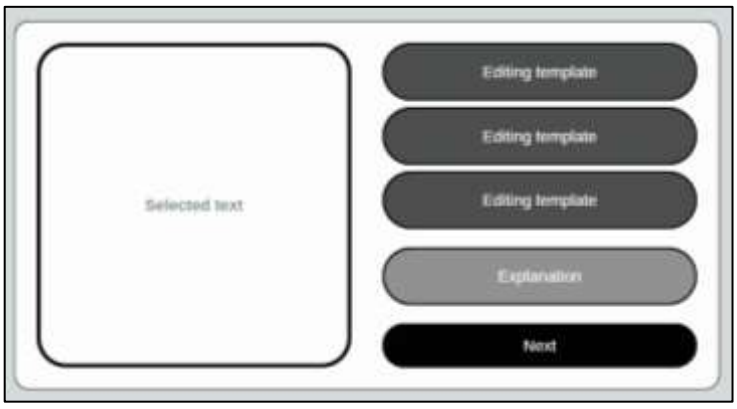


Рисунок 2.8 – Макет модального вікна вибору шаблону редагування

Другий крок передбачає відображення індикатору виконання запиту з текстом інформування про тривалість обробки.

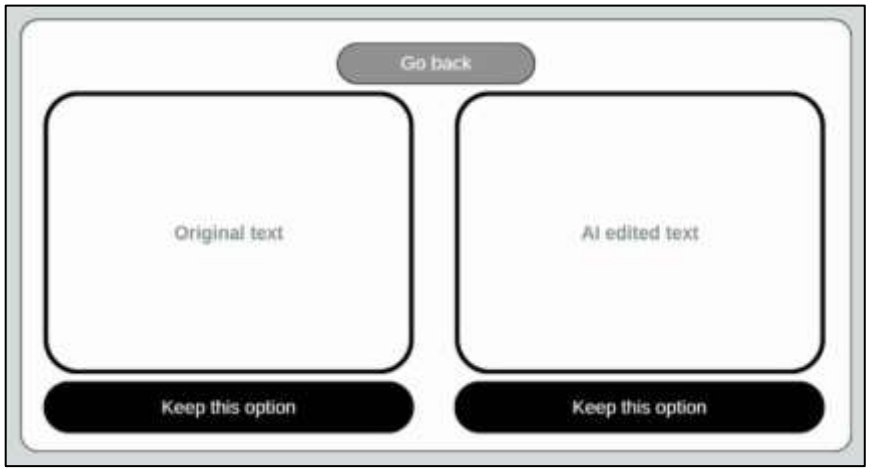


Рисунок 2.9 – Макет режиму порівняння оригінального та відредагованого тексту

Після отримання результату від сервера реалізується третій, завершальний крок, що представлений режимом порівняння (рис. 2.9). У його межах поряд один з одним відображаються 2 варіанти обраного текстового фрагменту: зліва – оригінальний, справа – відредагований штучним інтелектом. Важливим аспектом цього етапу є технічна реалізація блокування копіювання відредагованого тексту, що запобігатиме неконтрольованому використанню результату обробки поза межами модуля.

Після вибору прийнятного варіанту з’являється фінальне модальне вікно підтвердження заміни фрагмента в основному тексті глави (див. додаток А рис. А.3).

У разі успішного збереження глави, яка містить фрагменти тексту, що були відредаговані за допомогою інструменту, на неї встановлюється мітка «Відредаговано з використанням ШІ», що відображається у вигляді невеликого фірмового значка на сторінці твору та безпосередньо на сторінці перегляду цієї глави (рис. 2.10).

Рисунок 2.10 – Відображення мітки «Відредаговано з використанням ШІ» на сторінці твору та сторінці глави

Дизайн інтерфейсу модуля повністю узгоджуватиметься з загальною стилістикою Dragon Tale через використання тих самих шрифтів, кольорової схеми та базових UI-компонентів (зокрема модальних вікон підтвердження). Робоча назва інструменту «Подих дракона (ШІ)» природньо впишеться до загальної концепції брендингу платформи, акцентуючи при цьому, що інструмент виконує редагування за допомогою великої мовної моделі.

Інтерфейс модуля буде виконано ефективно-мінімалістичним без зайвих компонентів з адаптивністю для різних типів пристроїв.

Висновки до розділу 2

Під час роботи над цим розділом було здійснено комплексне проєктування інтелектуального інструменту автоматизованого редагування, для якого було спочатку проаналізовано існуючу архітектуру веб-платформи Dragon Tale, та, у результаті, визначено зону й об’єкт його інтеграції. Проведений аналіз дозволив чітко зрозуміти внутрішню організацію системи, структуру даних, взаємозв’язки між наявними сутностями та особливості роботи користувача з вбудованим редактором. У результаті обґрунтовано доцільність вибору глави як основної точки інтеграції інструменту, а також визначено підхід до реалізації мітки прозорості його використання.

Було детально визначено 19 функціональних та 9 нефункціональних вимог до модуля, що описують як безпосередній функціонал інструменту, так і умови його стабільної, безпечної та зручної роботи. На їх основі сформовано загальну архітектуру системи з урахуванням інтеграції інструменту редагування, у межах якої сервер виступає центральним елементом взаємодії із зовнішнім сервісом обробки природної мови. Також сформовані вимоги стали фундаментом проєктування внутрішньої логіки модулю, що базується на багаторівневому контролі вхідних даних, отриманні обов'язкової згоди користувача, застосуваннф бібліотеці шаблонів, формуванні структурованих запитів, реалізації механізму human-in-the-loop та прозорій обробці результатів.

Окремо було розглянуто механізм фіксації факту використання ШІ на рівні бази даних шляхом розширення існуючої таблиці глав «works_text» булевим полем «ai_edited». Також обґрунтовано необхідність впровадження тригерного захисту для унеможливлення зміни цієї мітки після встановлення.

Детально описано логіку взаємодії користувача з інструментом, яка природньо інтегрується в процес редагування глави через систему модальних вікон. Її головною особливістю є надання автору повного контролю на кожному етапі роботи з чітким розмежуванням звичайного редагування та використання ШІ-інструменту, що спрямовано на формування культури свідомого та прозорого застосування великих мовних моделей у процесі редагування.

Велику увагу приділено проєктуванню бібліотеки шаблонів редагування. Визначено єдину структуру шаблону, яка включає поля «id», «key», «label», «description» та «system_prompt». Також детально опрацьовано методологію формування структурованих запитів до LLM для редагування художньої прози, зокрема надання ролі, negative instructions, soft та hard control, інструкції щодо збереження мови оригіналу та вимогу відповіді в форматі JSON.

Не меншу увагу було приділено розробці алгоритму визначення факту використання відредагованого ШІ варіанту тексту в загальному обсязі глави під час її збереження, що дозволяє надійно фіксувати використання інструменту навіть у разі часткової ручної корекції отриманих результатів автором. Алгоритм

базується на збереженні снапшотів відредагованих фрагментів, двофазній нормалізації тексту та поєднанні перевірки точного входження з методом шинглів і адаптованою мірою Жаккара.

Фінальним етапом проєктування стала робота над плануванням користувацького інтерфейсу модуля. Було визначено ключові інтерфейсні компоненти: елементи візуалізації початкового вибору тексту, контекстну кнопку активації, трьохкрокове модальне вікно основного функціоналу інструменту та мітки прозорості.

РОЗДІЛ 3. РОЗРОБКА МОДУЛЮ АВТОМАТИЗОВАНОГО РЕДАГУВАННЯ

3.1. Обґрунтування вибору інструментів та технологій

Технологічний стек для розробки інтелектуального інструменту обирався на основі потреби його інтеграції вже до працюючої веб-платформи Dragon Tale. Через те, що модуль виступає лише функціональним розширенням існуючої системи, оптимальним рішенням є використання тих самих інструментів і технологій. Вони дозволять провести його безшовне впровадження у наявну клієнт-серверну архітектуру без виникнення технічних конфліктів.

Клієнт і сервер реалізуються на основі фреймворку Next.js у поєднанні з мовою TypeScript. Full-stack характер Next.js дозволяє поєднувати логіку обох архітектурних частин в межах одного проєкту, що значно спрощує процес розробки. Використання API Routes дає можливість створювати серверні ендпоінти безпосередньо в межах додатку, не потребуючи розгортання окремого бекенд-сервера [38].

Розгортання здійснюється на платформі Vercel – нативному середовищі для Next-проєктів. Вона забезпечує високу швидкість роботи serverless-функцій, дозволяє автоматизувати процес деплою та гарантує безпечне зберігання змінних оточення (API-ключів і параметрів підключення до БД) [39].

Платформа Supabase слугує для роботи з даними та автентифікацією. В її основі лежить PostgreSQL, що дозволяє легко розширити наявну структуру БД під мітку прозорості та її тригерний захист.

Через сервер модуль взаємодіятиме з інференс-платформою Groq Cloud за використання відповідного API-інтерфейса [40]. Вибір саме Groq ґрунтується на високій швидкості обробки запитів, що досягається завдяки спеціалізованій архітектурі LPU. Для некомерційної платформи Dragon Tale така швидкість і відносна безкоштовна доступність використання є оптимальним рішенням для першого впровадження модуля автоматизованого редагування. Цільовою

моделлю виступає Llama-3.1-8B-Instant, що продемонструвала достатню якість розуміння контексту та стилістичних нюансів художньої прози [41].

Завдяки спроєктованій архітектурі існує можливість легкої заміни Groq на будь-якого іншого провайдера (OpenAI, Anthropic тощо) без суттєвих змін у загальній логіці роботи інструмента. Це забезпечує масштабованість рішення, адже при переході до комерційного використання, можливе застосування більш потужною мовної моделі без потреб у внесені значних змін.

3.2. Реалізація клієнтської частини

Для реалізації інтелектуального інструмента автоматизованого редагування на клієнтській стороні веб-додатку використано поєднання вже існуючих компонентів платформи Dragon Tale з новими. Основними складовими клієнтської частини модуля є:

- «Page» – виступає серверною точкою входу до сторінки редагування глави. Цей компонент виконує початкову перевірку користувача, валідацію його прав щодо редагування конкретної глави та здійснює перенаправлення у випадку їх відсутності.
- «EditChapterPage» – реалізує основну клієнтську логіку сторінки редагування. Він відповідає за отримання даних глави та шаблонів ШІ-редагування, виклик функцій збереження та видалення глави, а також передачу обраних даних до подальшого компонування запиту до великої мовної моделі.
- «ChapterForm» – містить основний об’єм розмітки сторінки та реалізує попередній контроль над фрагментом тексту, який обирається для ШІ-редагування. У його межах відбувається підготовка даних для подальшої взаємодії з модулем.
- «EditWithAIModal» – виступає інтерфейсною складовою інтелектуального інструменту. З ним взаємодіє користувач для вибору шаблону редагування, візуального перегляду інформації про

процес роботи над текстом, а також для порівняння початкового та відредагованого ШІ варіантів фрагмента.

- «ConfirmModal» – використовується як універсальне модальне вікно підтвердження дій у різних частинах веб-додатку. У межах взаємодії з модулем автоматизованого редагування воно застосовується для отримання згоди користувача на передачу тексту до ШІ та для остаточного підтвердження вибору фінального варіанта тексту.

Усі зазначені компоненти, окрім «Page», використовують директиву «use client», що дозволяє їм виконуватися безпосередньо на клієнті [42]. Це є необхідним для використання React hooks, обробки подій, керування станами компонентів та динамічного оновлення інтерфейсу під час взаємодії з користувачем.

Попередня перевірка виділеного текстового фрагмента, що реалізується у компоненті «ChapterForm», базується на обробці події «onSelect» у текстовому полі редагування (ліст. 3.1). Завдяки ній отримується обраний користувачем текстовий фрагмент, який у подальшому буде перевірятися, та межі його початку та завершення у загальному обсязі глави. Отримані координати у подальшому використовуються для коректної заміни оригінального фрагмента на фінальний обраний варіант.

Лістинг 3.1 – Обробка події «onSelect»

```
onSelect={(e) => { const target = e.target as HTMLTextAreaElement;
                  setSelectedText(
                    target.value.substring(
                      target.selectionStart,
                      target.selectionEnd, ),);
                  setSelectionRange({
                    start: target.selectionStart,
                    end: target.selectionEnd,});}}
```

Основна логіка перевірки знаходиться у методі «isAISelectionValid», що використовує React-хук «useMemo» для оптимізації продуктивності шляхом повторного виконання обчислень лише у разі зміни виділеного тексту [43]. У межах цього методу здійснюється перевірка відповідності обраного тексту

встановленим вимогам: контроль кількості абзаців фрагмента (AI_MAX_PARAGRAPHS = 1), валідація відповідності мінімальній (AI_MIN_CHARS = 200) та максимальній (AI_MAX_CHARS = 2000) кількості символів, а також оцінка «чистоти» тексту для уникнення передачі випадкових символів або некоректних їх наборів до мовної моделі.

Для визначення «чистоти» фрагмента початково виконується його базова нормалізація шляхом видалення надлишкових пробілів. Потім відбувається підрахунок кількості символів, які є літерами у стандарті Unicode. На основі отриманих значень обчислюється щільність тексту як відношення кількості літер до загальної кількості символів нормалізованого фрагмента (ліст. 3.2).

Лістинг 3.2 – Визначення «чистоти» фрагменту

```
const lettersCount = (normalizedSelection.match(/\p{L}/gu) || []).length;
    const letterDensity =
      contentLength === 0 ? 0 : lettersCount / contentLength;
```

Такий підхід дозволяє уникнути обробки невалідних фрагментів, які складаються переважно з пробілів, спеціальних знаків або випадковому набору символів. Якщо щільність становить менше 40% (0.4), виділений текстовий фрагмент вважається непридатним для застосування інтелектуального редагування.

На основі здійснених перевірок формується об’єкт стану типу «isValid: boolean, reason: "string"». Значення «isValid» керує можливістю використання інструменту автоматизованого редагування, а «reason» містить ключ причини недоступності, який використовується для рендерингу невеликого текстового повідомлення для користувача. Для локалізації текстового супроводу використовується бібліотека «next-intl» (як і по всій платформі загалом), яка дозволяє динамічно підтягувати потрібні повідомлення зі словників залежно від ситуації [44].

Відображення підказок реалізується за допомогою компонента «OverlayTrigger» бібліотеки React-Bootstrap, який відслідковує наведення

курсора на кнопку та забезпечує появу спливаючого вікна з поясненням причин недоступності інструмента (рис. 3.1).

Рисунок 3.1 – Відображення підказки при виборі фрагмента, який перевищує задану кількість абзаців

Додатково, для покращення користувацького досвіду реалізовано візуальне розділення активного та неактивного станів кнопки шляхом зміни її кольору, що дозволяє користувачу інтуїтивно визначати поточний стан доступності інструменту (рис. 3.2).

Рисунок 3.2 – Вигляд кнопки активації інструмента в активному та неактивному станах

Для отримання підтвердження користувача щодо передачі його тексту на обробку штучним інтелектом використовується модальне вікно універсального компоненту «ConfirmModal» (рис. 3.3).

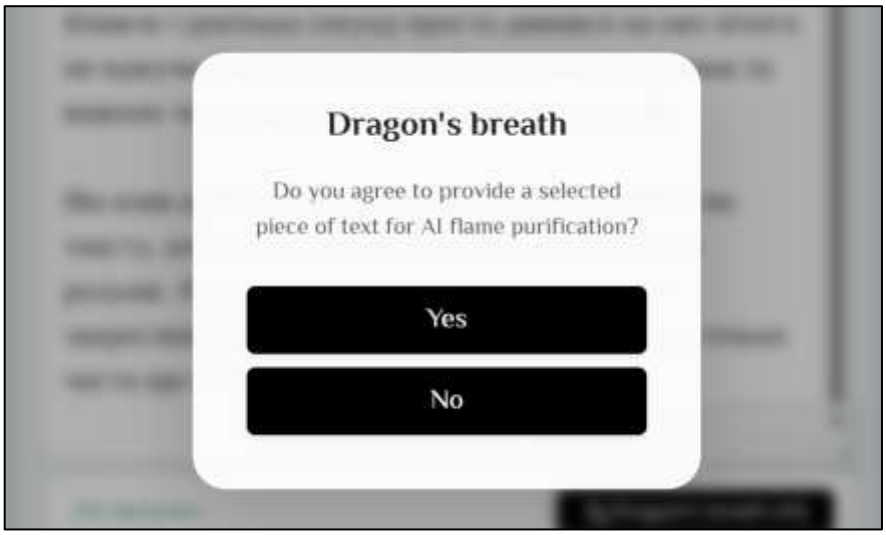


Рисунок 3. 3 – Вигляд модального вікна отримання згоди на ШІ-обробку тексту

Його текстове наповнення параметризується відповідно до сценарію отримання згоди. Завдяки реалізованій логіці гарантується доступ до інтелектуального інструмента лише у разі свідомого погодження користувача, що запобігає випадкового запуску ШІ-редагування внаслідок помилкових кліків (ліст. 3.2).

Лістинг 3.3 – Використання компонента «ConfirmModal» для отримання згоди на ШІ-обробку тексту

```
<ConfirmModal
        isOpen={showAIConsentModal}
        title={t("modals.aiConsent.title")}
        message={t("modals.aiConsent.message")}
        confirmText={t("modals.aiConsent.confirm")}
        cancelText={t("modals.aiConsent.cancel")}
        onConfirm={() => {
          setShowAIConsentModal(false);
          setShowAIEditModal(true);}}
        onCancel={() => setShowAIConsentModal(false)}/>
```

Основна частина модуля реалізується за допомогою компонента «EditWithAIModal», який побудований на основі переходу між трьома функціональними станами (рис. 3.4): «template» – вибір шаблону редагування, «processing» – візуалізація процесу обробки тексту; «compare» – порівняння

оригінального та відредагованого варіантів фрагменту з подальшим підтвердженням застосування обраних змін.

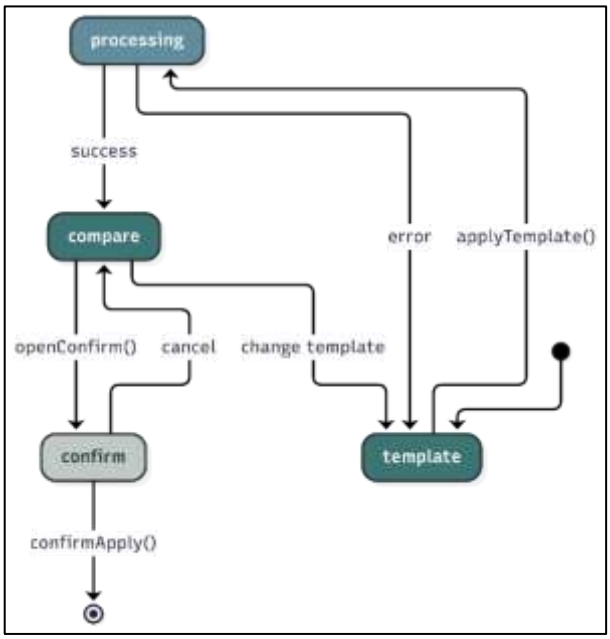


Рисунок 3.4 – Перехід між станами модального вікна інтелектуального інструмента

Все починається на етапі «template», для якого у компоненті «EditChapterPage» під час ініціалізації сторінки здійснюється асинхронне отримання переліку доступних шаблонів редагування (ліст. 3.4). Кожен шаблон містить набір параметрів («id», «key», «label», «description»), що необхідні для його відображення та подальшого використання.

Лістинг 3.4 – Асинхронний запит на отримання шаблонів редагування

```
const initPage = async () => {
      setIsLoading(true);
      try {
        const [chapterResult, templates] = await Promise.all([
          getChapterForEdit(workId, chapterId),
          getAITemplatesAction(), ]);
        …
        setAITemplates(templates);
```

Отримані шаблони передаються до компонента модального вікна та відображаються поряд з обраним текстовим фрагментом у вигляді списку радіо-

кнопок. На кожному елементі вибору реалізовано обробник відслідковування змін, який не лише фіксує обраний об’єкт у стані «selectedTemplate», а й динамічно відображає коротко пояснення очікуваного результату редагування (рис. 3.5).

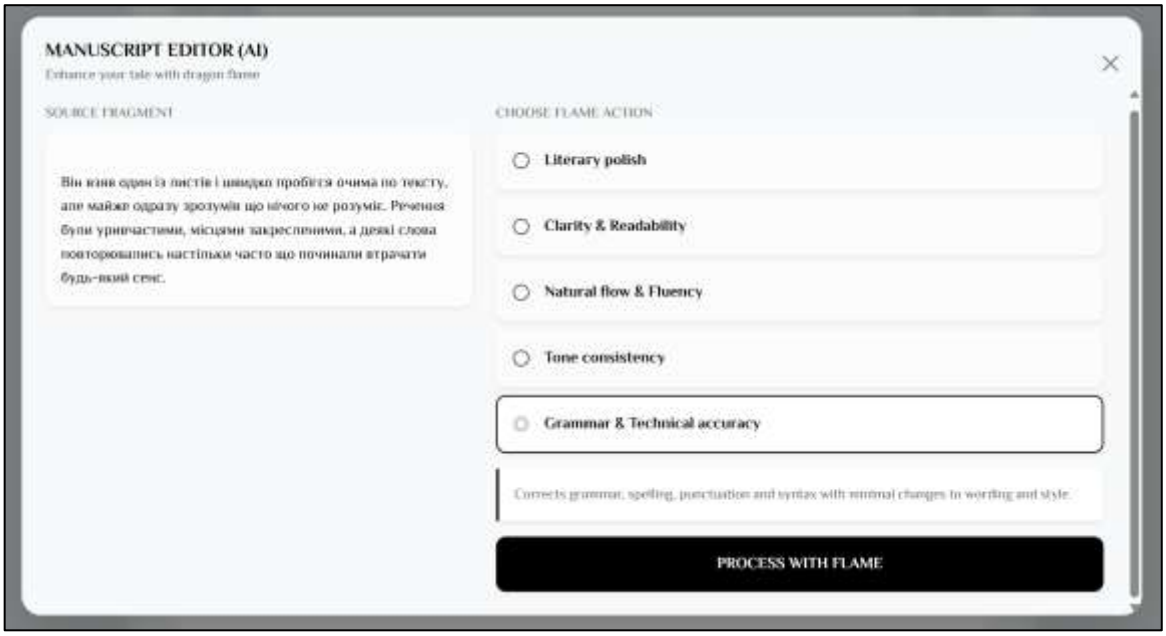


Рисунок 3.5 – Етап вибору шаблону редагування модального вікна

Перехід до наступного етапу відбувається лише після вибору одного з шаблонів – в іншому випадку кнопка запуску залишається неактивною. При натисканні на неї викликається метод «applyTemplate», який виконує 2 дії:

1. встановлює стан модального вікна на «processing»;
2. ініціює асинхронний запит на редагування тексту через передану callback-функцію «onRequestEdit».

Метод «applyTemplate» не виконує запит безпосередньо, а використовує пропс-інтерфейс «onRequestEdit», який поєднує інструмент із основною логікою сторінки, що міститься у компоненті «EditChapterPage» (ліст. 3.5).

Лістинг 3.5 – Методу «applyTemplate»

```
const applyTemplate = async () => {
    if (!selectedTemplate) return;
    setStep("processing");
    setError(null);
```

```
        try {
                const editedText = await onRequestEdit(selectedTemplate,
selectedText);
```

Наявний у компоненті «EditChapterPage», метод «handleRequestAIEdit» викликає серверну дію «editWithAIAction», передаючи ключ обраного шаблону та текстовий фрагмент (ліст. 3.6). У відповідь повертається оброблений текст у вигляді рядка.

Лістинг 3.6 – Частина методу «handleRequestAIEdit»

```
const handleRequestAIEdit = useCallback(
    async (template: aiTemplate, text: string): Promise<string> => {
      try {
        const result = await editWithAIAction(template.key, text);
        return result.edited_text;}
```

Поки йде очікування відредагованого тексту користувачеві демонструється індикатор виконання, що дозволяє утримати його увагу та сигналізує про процес обробки (рис. 3.6).

Рисунок 3.6 – Етап процесу обробки тексту

Як тільки асинхронний запит у методі «applyTemplate» успішно завершується, відбувається перехід до завершальної частини модального вікна – «compare». У випадку виникнення помилки здійснюється повернення до етапу «template» з демонстрацією повідомлення про помилку.

Під час порівняння модальне вікно візуально ділиться на 2 частини: зліва знаходиться оригінальний варіант текстового фрагмента, а справа – відредагований ШІ (рис. 3.7). Під кожним із них розміщені кнопки вибору відповідного результату для подальшого використання у загальному обсязі глави. При натисканні на кнопки відбувається виклик універсального компоненту підтвердження «ConfirmModal», що відкривається поверх основного модального вікна інструмента. Його використання забезпечує додатковий рівень підтвердження остаточного вибору перед внесенням змін. У разі відмови модальне вікно підтвердження просто закривається, повертаючи користувача до порівняння.

Рисунок 3.7 – Етап порівняльного вибору між оригінальним варіантом фрагменту та відредагованим ШІ

Для запобігання прямого копіювання відредагованого ШІ варіанту використовується CSS-властивість «userSelect» у значенні «none». Вона забороняє користувачеві виділяти та копіювати текст за допомогою миші або клавіатури [45].

Якщо користувача не влаштовує отриманий результат, він має можливість повернення до етапу вибору шаблону. Це реалізовано за допомогою обробника події «onClick» на кнопці, який викликає метод «setStep("template")» та переводить модальне вікно до початкового стану.

Після підтвердження вибору викликається метод «handleApplyAIText», до якого передається фінальний варіант текстового фрагмента – або оригінальний, або відредагований ШІ (ліст. 3.7).

Лістинг 3.7 – Метод «handleApplyAIText»

```
const handleApplyAIText = (finalText: string) => {
    if (!selectionRange) return;
    setChapterText(
      (prev) =>
        prev.slice(0, selectionRange.start) +
        finalText +
        prev.slice(selectionRange.end),);
    handleRegisterAIText(finalText);
    setSelectedText("");
    setSelectionRange(null);};
```

Цей метод використовує раніше збережені межі виділення («selectionRange.start» та «selectionRange.end») і виконує вставку фрагмента шляхом конкатенації 3 частин: тексту від початку до стартового індексу виділення, фінального обраного варіанта фрагмента та частини рядка від кінцевого індексу виділення до завершення глави.

Після першого використання інструменту користувач може повторно застосовувати його необмежену кількість разів без необхідності перезавантаження сторінки або зміни логіки роботи клієнтської частини.

3.3. Реалізація серверної частини

Реалізація серверної частини інтелектуального модуля починається з формування бібліотеки шаблонів редагування. Вона реалізована у вигляді статичного масива об’єктів «aiTemplates». Завдяки ньому забезпечується централізоване зберігання та швидкий доступ до даних під час виконання запитів. Кожен елемент масиву відповідає структурі, визначеній інтерфейсом «AITemplate» (ліст. 3.8).

Лістинг 3.8 – Інтерфейс «AITemplate»

```
export interface AITemplate {
  id: number;
  key: string;
```

```
label: string;
description: string;
system_prompt: string;}
```

Поле «system_prompt» кожного шаблону, як і передбачалося, складається з двох частин:

- змінної, яка визначає конкретний тип редагування;
- фіксованої, що є однаковою для всіх шаблонів.

У фіксованій частині знаходиться набір базових інструкцій для мовної моделі: визначення ролі, негативні обмеження, правила роботи з авторським стилем, вказівки щодо мови, формат вихідних даних та правила формування JSON-відповіді (рис. 3.7).

ROLE: you are a professional literary editor with 20 years of experience in editing prose. Your only goal is to assist the author without imposing your own vision or style.

RULES: Do NOT change the plot, character behavior, or dialogue meaning. Do NOT add new descriptions, metaphors, or ideas. Do NOT remove specific meaningful details. Do NOT "modernize" or "simplify" the vocabulary if it serves the author's style. Only perform edits that align with professional literary standards. Do NOT insert any introductory or concluding phrases (e.g., "Here is the edited text").

LANGUAGE: Detect the language of the input text. The output MUST be in the same language.

OUTPUT FORMAT: Your response MUST be a valid JSON object with the following structure:

{"edited_text": "the final edited version of the provided text",

"notes": " short notes only if there are critical issues that require author's attention (otherwise null)}

OUTPUT RULES: Return ONLY the JSON object. Do NOT include markdown blocks. Do NOT include extra text. The JSON must be syntactically valid.

Рисунок 3. 8 – Фіксована частина системного запиту

Було створено 5 базових варіантів редагування, кожен із яких містить певну специфіку обробки тексту. Детальні характеристики реалізованих шаблонів знаходяться у табл.3.1.

Таблиця 3.1

Наявні у бібліотеці шаблони редагування

Назва шаблону (key)	Короткий опис	Скорочений варіант змінної частини запиту
Літературне шліфування (literary_polish)	Комплексне літературне редагування, що покращує стилістичну якість тексту.	Improve clarity, flow, rhythm and stylistic consistency. Refine sentence structure and transitions. Eliminate awkward phrasing.
Чіткість і читабельність (clarity_readability)	Вдосконалення тексту через спрощення складних конструкцій для кращого розуміння.	Enhance clarity and readability. Simplify overly complex sentences where necessary. Break long sentences if needed.
Природний потік і плавність (natural_flow)	Внесення змін для більшої природності та легкості читання.	Improve natural flow and fluency. Remove rigid and awkward constructions. Ensure seamless sentence connections.
Граматика та технічна правильність (grammar_technical)	Виправлення граматичних, орфографічних та пунктуаційних помилок.	Correct grammar, spelling, punctuation and syntax. Make minimal changes to wording.
Узгодженість тону (tone_consistency)	Забезпечення більш стабільного емоційного та стилістичного тону.	Ensure consistent tone and narrative voice. Eliminate tonal inconsistencies.

Асинхронна функція «getAITemplatesAction» відповідає за передачу даних шаблонів на клієнт, де завдяки ним формується список доступних варіантів редагування на початковому етапі модального вікна. З бібліотеки передається обмежений набір даних, який необхідний виключно для побудови UI (ліст. 3.9).

Лістинг 3.9 – Функція «getAITemplatesAction»

```
export async function getAiTemplatesAction() {
 return aiTemplates.map(({ id, key, label, description }) => ({
   id, key, label, description, }));}
```

Поле «system_prompt» навмисно не передається, адже містить внутрішню логіку взаємодії з мовною моделлю, що не має бути доступною на клієнті.

Для спрощення підключення та подальшої роботи з LLM використовується окремий конфігураційний файл «aiConfig» (ліст. 3.10). Він відповідає за централізоване керування параметрами взаємодії з зовнішнім сервісом.

Лістинг 3.10 – Вміст конфігураційного файлу «aiConfig»

```
export const aiConfig = {
  apiKey: process.env.GROQ_API_KEY,
  baseUri: "https://api.groq.com/openai/v1",
  model: "llama-3.1-8b-instant",};
```

Конфігураційний об’єкт «aiConfig» включає:

- «apiKey» – ключ доступу до API зовнішнього сервісу Groq, який зберігається у змінній середовища для уникнення його розкриття;
- «baseUri» – базову URL-адресу API Groq, через яку здійснюється взаємодія з ШІ;
- «model» – ідентифікатор обраної моделі, яка використовується для редагування тексту.

Асинхронна функція «editWithAIAction» відповідає за основну взаємодію з Groq API. Її вхідними параметрами є: ключ шаблону «templateKey» та обраний користувачем текстовий фрагмент «text».

Спочатку здійснюється перевірка прав доступу. Для цього створюється серверне з’єднання з Supabase та отримуються дані поточної сесії користувача за допомогою методу «getUser» (ліст. 11).

Лістинг 3.11 – Початкова перевірка прав доступу

```
const {
    data: { user },
  } = await supabase.auth.getUser();
  if (!user) throw new Error("Unauthorized");
```

Результат виконання запиту – об’єкт авторизованого користувача. Якщо такого знайдено не було, то подальша взаємодія з Groq API не відбувається, а робота функції завершується помилкою авторизації.

Наступним етапом є пошук потрібного шаблону редагування за ключем шаблону «templateKey», що було передано з клієнта (ліст. 3.12).

Лістинг 3.12 – Пошук необхідного шаблона

```
const template = aiTemplates.find((t) => t.key === templateKey);
 if (!template) throw new Error("Template not found");
```

У випадку, коли необхідний шаблон не знайдено, виникає помилка і виконання функції припиняється. Якщо ж все відбувається нормально, то отримується об’єкт шаблону, який містить поле «system_prompt». Воно необхідне для подальшого формування запиту до мовної моделі.

Перед взаємодією з Groq API необхідною є ініціалізація клієнта бібліотеки «openai», яка відбувається на основі даних конфігураційного файлу «aiConfig» (ліст. 3.13).

Лістинг 3.13 – Ініціалізація OpenAI-клієнта

```
const client = new OpenAI({
  apiKey: aiConfig.apiKey,
  baseURL: aiConfig.baseUri,});
```

Використання офіційного OpenAI-клієнта зумовлено тим, що Groq повністю підтримує OpenAI-сумісний формат взаємодії. Це дозволяє застосовувати стандартний інструментарій для керування потоками даних, обробки помилок та форматування відповідей [46]. Також завдяки такому підходу подальша потенційна заміна AI-провайдера буде проходити без внесення суттєвих змін до коду.

Основна взаємодія з Groq API відбувається через асинхронний метод «client.chat.completions.create» (ліст. 3.14), який відповідає за формування та відправки запиту. Його складовими є:

1. «model» – ідентифікатор конкретної моделі, яка буде обробляти запит.
2. «messages» – масив об’єктів, для формування контексту взаємодії з ШІ:
    - частина з роллю «system» містить системні інструкції шаблону редагування «system_prompt», що визначають правила поведінки моделі;
    - частина з роллю «user» передає текстовий фрагмент, який буде оброблятися.
3. «response_format» – параметр, що задає жорстку вимогу повернення відповіді у форматі JSON-об’єкта.

Лістинг 3.14 – Формування та відправка запиту

```
const response = await client.chat.completions.create({
  model: aiConfig.model,
  messages: [
    { role: "system", content: template.system_prompt },
    { role: "user", content: text }, ],
  response_format: { type: "json_object" }, });
```

Хоча й вимога щодо формату відповіді прописується у самому системному запиті, використання «response_format» забезпечує додатковий контроль на рівні API. Завдяки цьому ризики отримання некоректного текстового результату замість валідного JSON-об’єкта мінімізуються ще більше.

Отримана відповідь проходить двоетапну обробку (ліст. 3.15):

- Спочатку система витягує текстовий контент зі структури «response.choices[0].message.content» та перевіряє його на наявність даних. Якщо відповідь порожня, виконання переривається з помилкою.
- Потім отримані дані перетворюються зі звичайного string-рядка на дійсний JavaScript-об’єкт за допомогою методу «JSON.parse».

Лістинг 3.15 – Обробка отриманої відповіді

```
const content = response.choices[0].message.content;
if (!content) throw new Error("Empty AI response");
return JSON.parse(content);
```

Якщо все проходить успішно, то на клієнт повертається розпарсений JSON-об’єкт, який містить «edited_text» із результатом редагування. Цей текст відразу ж демонструється на порівняльному етапі модального вікна інструмента. У випадку ж помилки або під час виконання запиту, або під час обробки відповіді на клієнт повертається повідомлення «AI editing failed».

3.4. Реалізація механізму захисту даних на рівні бази даних

Захист мітки «ai_edited» забезпечено на рівні БД з використанням синтаксису, притаманному PostgreSQL. Таке рішення обрано для створення умов коректного збереження значення навіть при спробі прямої зміни запису.

Механізм захисту реалізовано завдяки функції «ai_flag_protection» (ліст. 3.16, яка базується на:

1. Перехопленні спроби зміни – явній перевірці, коли є спроба зміни значення «ai_edited» з «TRUE» на «FALSE».
2. Примусовій фіксації значення – забезпеченні незворотності шляхом автоматичного збереження мітки у значенні «TRUE», якщо вона вже була раніше активована.

Лістинг 3.16 – Функція «ai_flag_protection»

```
IF OLD.ai_edited = TRUE AND NEW.ai_edited = FALSE THEN
        NEW.ai_edited := TRUE;
    END IF;
    IF OLD.ai_edited = TRUE THEN
        NEW.ai_edited := TRUE;
    END IF;
    RETURN NEW;
```

Тригером «ai_flag_protection_trigger» забезпечено автоматичне виконання створеної функції завдяки конструкції «BEFORE UPDATE» (ліст. 3.17). Завдяки ній відбувається перехоплення будь-яких спроб оновлення записів в таблиці

«works_text» ще до фактичного внесення змін, що дозволяє примусово відновлювати коректне значення мітки [47].

Лістинг 3.17 – Створення тригеру «ai_flag_protection_trigger»

```
CREATE TRIGGER ai_flag_protection_trigger
BEFORE UPDATE ON works_text
FOR EACH ROW
EXECUTE FUNCTION ai_flag_protection();
```

На рис. 3.9 продемонстровано загальний принцип роботи «ai_flag_protection_trigger».

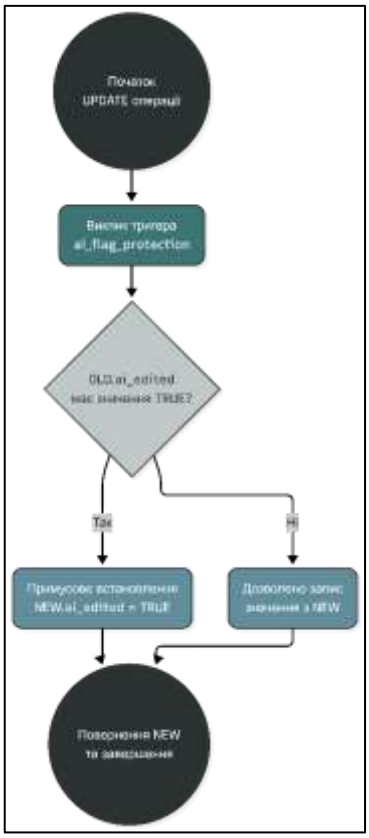


Рисунок 3.9 – Алгоритм роботи тригера «ai_flag_protection_trigger»

Спрацювання тригеру відбувається автоматично перед кожним виконанням оператора «UPDATE» для таблиці «works_text». Порівнюються значення поля «ai_edited» у старій та новій версіях. У разі спроби змінити мітку з «TRUE» на «FALSE» система примусово повертає значення «TRUE», не допускаючи зняття мітки.

3.5. Реалізація алгоритму визначення використання ШІ при редагуванні тексту

Механізм визначення наявності ШІ-відредагованого тексту в фінальному обсязі глави реалізовано через клас «AITracking», який містить усю спроєктовану логіку перевірки та викликається після натискання кнопки збереження у компоненті «ChapterForm».

Перевірка виконується тільки тоді, коли глава ще не має встановленої мітки використання ШІ, а в межах поточної сесії редагування було зафіксовано застосування інтелектуального інструменту через наявність снапшотів відредагованих текстових фрагментів (ліст 3.18).

Лістинг 3.18 – Початкові умови для виконання перевірки

```
const handleSubmit = (e: React.FormEvent) => {
   …
    let aiDetected = aiAlreadyMarked;
    if (!aiDetected && aiSnapshots.length > 0) {
      aiDetected = AITracking.checkIfAIPresent(chapterText, aiSnapshots);}
    onSubmit({ chapterText, authorComment, aiEdited: aiDetected, });};
```

При кожному успішному отриманні відповіді від ШІ відредагований текст передається через callback-функцію модального вікна «onGenerated» до компонента «ChapterForm», де фіксується у вигляді снапшота. Це відбувається через метод «handleRegisterAIText», що викликає «prepareSnapshot» класу «AITracking» для підготовки тексту, а також додає отриманий результат до локального масиву «aiSnapshots» (ліст. 3.19).

Лістинг 3.19 – Метод «handleRegisterAIText»

```
const handleRegisterAIText = (text: string) => {
    const snapshot = AITracking.prepareSnapshot(text);
    setAISnapshots((prev) => {
      if (prev.includes(snapshot)) return prev;
      return [...prev, snapshot];});};
```

Снапшоти зберігаються виключно у стані компонента «ChapterForm» незалежно від того, чи застосував користувач відредагований фрагмент до фінального тексту глави. Вони існують лише протягом поточної сесії

редагування, що дозволяє не перевантажувати сервер та БД зайвими тимчасовими даними.

Логіка визначення наявності ШІ-відредагованого тексту реалізується через інкапсульований функціонал класу «AITracking». Основні методи, їх вхідні параметри, вихідні дані та короткий опис призначення наведено у табл. 3.2.

Таблиця 3.2

Методи класу «AITracking»

Назва методу	Вхідні параметри	Вихідні дані	Короткий опис логіки
checkIfAIPresent	фінальний текст глави, масив снапшотів	булеве значення	Основний метод визначення наявності ШІ-відредагованих фрагментів у загальному обсязі глави.
normalize	текст, прапорець видалення пунктуації	нормалізований рядок	Приведення до нижнього регістру, видалення HTML-тегів і зайвих пробілів; у жорсткому режимі – додаткове видалення розділових знаків.
getShingles	нормалізований текст	колекція шинглів	Розбиття тексту на n-грами (шингли) розміром по 3 слова.
calculateOverlap	2 колекції шинглів	числовий коефіцієнт схожості	Розрахунок частки шинглів снапшота, які присутні у фінальному тексті глави.

Продовження таблиці 3.2

prepareSnapshot	відредагований ШІ текстовий фрагмент	підготовлений снапшот	Попередня нормалізація та підготовка тексту перед збереженням у масив снапшотів.

Основний метод «checkIfAIPresent» використовує поєднання функціоналу всіх допоміжних та реалізує багаторівневий пошук відредагованих ШІ текстових фрагментів тексту у фінальному обсязі глави.

На початку виконується базова перевірка вхідних даних – чи переданий текст глави та масив снапшотів не є порожніми (ліст. 3.20). Далі проходить м'яка нормалізація як фінального тексту, так і кожного снапшота: приведення до нижнього регістру, очищення від випадкових HTML-тегів, якщо такі присутні, та видалення надмірних пробілів.

Лістинг 3.20 – Швидка перевірка точного входження

```
const softNormalizedFullText = this.normalize(fullText, false);
    for (const rawSnapshot of snapshots) {
      const softSnapshot = this.normalize(rawSnapshot, false);
      if (softSnapshot.length < this.MIN_SNAPSHOT_LENGTH) continue;
      if (softNormalizedFullText.includes(softSnapshot)) {return true;}}
```

Після нормалізації здійснюється швидка перевірка точного входження снапшота у фінальний текст глави через «includes». Занадто короткі снапшоти після нормалізації пропускаються, адже можуть спричиняти хибні збіги.

Якщо вже на цьому етапібуло виявлено входження відредагованого ШІ фрагмента до загального обсягу тексту, то метод одразу повертає позитивний результат без виконання подальших перевірок. В іншому випадку здійснюється перехід до більш глибокого аналізу на основі шинглів.

На цьому етапі і фінальний текст, і снапшоти проходять вже жорстку нормалізацію, під час якої додатково видаляються розділові знаки. Після цього вони розбиваються на множини шинглів – n-грами розміром по 3 слова (ліст 3.21).

Лістинг 3.21 – Нормалізація та отримання множин шинглів для другого рівня перевірки

```
const hardNormalizedFullText = this.normalize(fullText, true);
    const fullTextShingles = this.getShingles(hardNormalizedFullText);
    if (fullTextShingles.size === 0) return false;
    for (const rawSnapshot of snapshots) {
      const hardSnapshot = this.normalize(rawSnapshot, true);
      if (hardSnapshot.length < this.MIN_SNAPSHOT_LENGTH) continue;
      const snapshotShingles = this.getShingles(hardSnapshot);
```

Додатково виконується перевірка на наявність шинглів у фінальному тексті, бо за їх повної відсутності суті у подальшому аналізі немає.

На основі отриманих колекцій шинглів розраховується коефіцієнт перекриття між шинглами снапшота та фінального тексту глави (ліст. 3.22).

Лістинг 3.22 – Розрахунок коефіцієнту перекриття

```
const similarity = this.calculateOverlap( snapshotShingles, fullTextShingles,);
          if (similarity >= this.SIMILARITY_THRESHOLD) {return true;}
```

Коефіцієнт перекриття визначає, яка частка шинглів снапшота присутня у фінальному тексті глави. Якщо отримане значення дорівнює або становить більше заданої межі в 45%, то системою встановлюється позитивна мітка використання ШІ під час редагування. Якщо ж навіть після проходження всіх етапів перевірки заданої межі не досягнуто, мітка залишається негативною.

Визначене значення мітки прозорості передається разом з іншими даними форми у метод «handleSave» компонента «EditChapterPage», який використовує серверну дію «updateChapterAction» для оновлення даних глави (ліст. 3.23).

Лістинг 3.23 – Оновлення даних глави

```
const result = await updateChapterAction(workId, chapterId, {
  chapterText: data.chapterText,
  authorComment: data.authorComment,
  aiEdited: data.aiEdited,});
```

Так результат роботи алгоритму інтегрується у вже існуючу логіку збереження глави без потреби у реалізації додаткової серверної обробки.

Значення мітки передається як звичайний параметр оновлення даних та надалі зберігається у відповідному полі таблиці «works_text» (ліст. 3.24).

Лістинг 3.24 – Оновлення даних таблиці «works_text»

```
const { error: updateError } = await supabase
    .from("works_text")
    .update({
      works_text: data.chapterText,
      author_comment: data.authorComment,
      ai_edited: data.aiEdited,})
    .eq("id_work_text", chapterId)
    .eq("id_work_inf", workId);
```

Отримана мітка «ai_edited» використовується не тільки для фіксації факту застосування ШІ-відредагованого тексту, а й для його подальшого візуального відображення в інтерфейсі. При завантаженні даних глави та списку глав твору на клієнті здійснюється перевірка значення цього поля. Якщо воно позитивне, то на сторінці читання глави та в списку глав «шапки» твору відображається відповідна візуальна мітка прозорості (рис. 3.10).

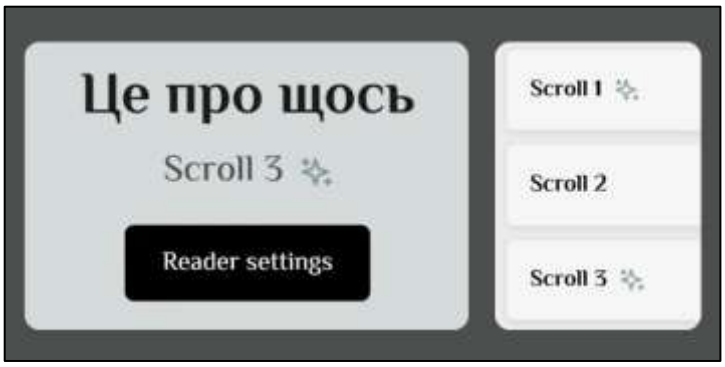


Рисунок 3.10 – Відображення мітки прозорості на сторінках сайту

Висновки до розділу 3

Під час роботи над цим розділом було реалізовано весь спроєктований функціонал інтелектуального інструмента автоматизованого редагування. Для цього спочатку було обґрунтовано вибір інструментарію та технологічного стеку (Next.js з TypeScript, Supabase, Vercel, Groq Cloud з моделлю Llama-3.1-8B-

Instant), який забезпечив простоту процесу розробки та високу продуктивність створеного рішення.

На клієнтській частині було реалізовано повний цикл взаємодії користувача з інструментом: виділення текстового фрагмента з попередньою валідацією (перевірка обмеження в 1 абзац, меж кількості символів та «чистоти» тексту), отримання свідомої згоди автора, трьохетапне модальне вікно («template» – вибір шаблону, «processing» – демонстрація процесу обробки, «compare» – порівняння оригінального та ШІ-відредагованого варіантів), а також механізм впровадження змін до основного обсягу глави з використанням збереженого діапазону виділення.

На серверній стороні було створено централізовану бібліотеку шаблонів редагування (5 різних шаблонів, які складаються зі спільної для всіх частини та унікальної змінної). Серверна логіка забезпечує формування структурованого запиту на основі системного шаблону та користувацького тексту до Groq API з подальшим отриманням відповіді у жорстко визначеному JSON-форматі. На рівні БД для поля «ai_edited» впроваджено тригерний захист «ai_flag_protection_trigger», який гарантує незворотність мітки прозорості та унеможливлює її зняття навіть при прямому оновленні.

Особливу увагу було приділено реалізації алгоритму визначення використання ШІ-редагування на основі класу «AITracking». Він працює на основі порівняння отриманих снапшотів відредагованих фрагментів під час поточної сесії редагування з фінальним текстом глави за допомогою двоетапної перевірки (точне входження після м’якої нормалізації та розрахунок перекриття шинглів після жорсткої). Отримане значення мітки прозорості просто інтегрується до стандартного процесу збереження глави без порушення існуючої логіки. Крім того, воно відображається у вигляді візуальної мітки прозорості на сторінці глави та в списку глав «шапки» твору.

РОЗДІЛ 4. ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ

4.1. Організація та проведення тестування модуля

4.1.1. Функціональне тестування модуля

Після закінчення розробки інтелектуального модуля автоматизованого редагування та перед його прямим диплоєм до основного проєкту було проведено комплексне тестування функціоналу. Особлива увага приділялася перевірці якості взаємодії користувача з інструментом, обробці можливих помилок, стабільності інтеграції з Groq API та надійності механізмів забезпечення прозорості використання штучного інтелекту.

Загалом було перевірено 14 сценаріїв, які стосуються основних етапів роботи модуля: попередню валідацію виділеного текстового фрагменту, повний цикл взаємодії користувача з самим інструментом (модальним вікном), обробку відповідей від ШІ, збереження змін, а також роботу механізмів фіксації, відображення та захисту мітки прозорості. Тестування проводилося вручну шляхом відтворення сценаріїв взаємодії у локальному середовищі розробки з реальним підключенням до Groq API [48]. У табл. 4.1 наведено перелік протестованих сценаріїв та результати їх виконання.

Отримані результати тестування демонструють повну відповідність роботи модуля редагування поставленим вимогам та запланованій функціональній логіці. Інтелектуальний інструмент коректно обробляє всі основні сценарії, забезпечуючи інтуїтиву та передбачувану взаємодію з користувачем, надійну обробку помилок і жорстке дотримання механізмів прозорості. Під час перевірок не було виявлено жодних критичних помилок або проблем у роботі модулю та системи загалом.

Таблиця 4.1

Результати функціонального тестування

Сценарій	Очікуваний результат
Виділення валідного фрагменту	Кнопка активації інструменту стає активною.
Виділення невалідного фрагменту (усі можливі ситуації)	Кнопка неактивна, а при наведенні на неї відображається відповідна контекстна підказка.
Спроба запустити інструмент без виділеного тексту	Кнопка неактивна та запуск інструменту неможливий.
Відмова надання згоди на ШІ-обробку	Модальне вікно згоди закривається, а доступ до інструменту не надається.
Відмова від застосування змін на етапі порівняння (вибір оригіналу)	Основний текст глави залишається без змін.
Підтвердження застосування ШІ-відредагованого варіанту	Відредагований фрагмент вставляється у потрібне місце загального тексту глави.
Спроба копіювання ШІ-відредагованого варіанту	Копіювання тексту неможливе.
Успішне отримання відповіді від Groq API	Відображення відредагованого ШІ тексту в порівнянні.
Некоректна відповідь від моделі	Відображення повідомлення помилки і прохання спробувати ще.
Повторне використання інструменту в межах однієї сесії редагування	Інструмент працює без змін при повторному виклику.
Збереження глави без використання інструменту	Міткапрозорості не встановлюється.
Збереження глави з підтвердженим використанням ШІ-редагування	Мітка прозорості встановлюється автоматично.
Відображення мітки у списку глав та на сторінці конкретної глави	Мітка правильно відображається в інтерфейсі.
Спроба зняти мітку вручну (через БД або API)	Тригерний захист не дозволяє змінити значення з TRUE на FALSE.

4.1.2. Функціональне тестування алгоритму виявлення використання ШІ-редагування

Алгоритм визначення факту використання інтелектуального модуля редагування є однією з ключових складових механізму забезпечення прозорості платформи Dragon Tale, тому потребує окремої уваги в контексті тестування.

Важливим моментом є те, що реалізований алгоритм не позиціонується як детектор штучного інтелекту. Він не класифікує текст на «людський» чи «згенерований», а виконує цілеспрямований пошук наявності ШІ-відредагованих фрагментів у фінальному обсязі глави шляхом порівняння снапшотів (зафіксованих нормалізованих версій тексту від мовної моделі) із загальним фінальним вмістом.

Тестування проводилося з метою підтвердження працездатності алгоритму в реальних сценаріях взаємодії автора з модулем, оцінки його чутливості до різного рівня внесення подальших змін у отримані текстові фрагменти, а також виявлення потенційних обмежень реалізації.

Через специфіку художнього тексту та необмежену можливість подальшого редагування отриманого результату формалізовані перевірки алгоритму є обмеженими. Результат його роботи залежить як від самого тексту, так і від рівня подальших змін, загального обсягу глави та співвідношення конкретного фрагмента до нього. Тому тестування проводилося вручну в локальному середовищі розробки на тестових главах обсягом від 1500 до 4000 символів. Для кожного з 7 виділених сценаріїв було проведено повний цикл взаємодії: виділення фрагменту, запуск інструменту, вибір шаблону, отримання результату, вибір необхідного варіанту тексту, внесення змін різного ступеня та збереження глави з подальшою перевіркою алгоритмом. Результати тестування наведено в табл. 4.2.

Таблиця 4.2

Результати тестування алгоритму визначення використання ШІ-редагування

Сценарій змін після отримання результату від ШІ	Очікуваний результат	Фактичний результат
Повне застосування ШІ-варіанту без внесення змін	Мітка встановлюється	Мітка встановлюється
Незначне редагування (1-5 слів, виправлення пунктуації, синоніми)	Мітка встановлюється	Мітка встановлюється
Часткове редагування (20-40% фрагменту змінено)	Мітка встановлюється	Часткове спрацювання
Значне редагування (понад 60% фрагменту змінено автором)	Мітка може не встановлюватися	Часткове спрацювання
Повна відмова від отриманого ШІ-варіант з подальшим ручним редагуванням	Мітка не повинна встановлюватися	Можливе хибне спрацювання
Невеликий обсяг ШІ-фрагменту до відносно великого обсягу глави	Спрацювання залежить від порогового значення	Часткове спрацювання
Багаторазове використання ШІ-редагування до одного фрагмента	Мітка встановлюється	Мітка встановлюється

Отримані результати тестування не є еталонними або універсальними, адже сама структура художнього тексту дуже різна від твору до твору, а фінальний результат сильно залежить від подальших дій автора. Проте проведене тестування дозволило оцінити алгоритм в типових сценаріях використання й чітко визначити його основні обмеження.

Показовим є випадок, коли автор використовує інструмент редагування, переглядає отриманий результат, але надалі відмовляється від його прямого застосування та продовжує самостійне редагування тексту. Алгоритм у такому

сценарії може ініціювати встановлення мітки, якщо вона ще не була активована раніше. Це пов’язано насамперед з особливостями реалізованої логіки збереження снапшотів та механізму перевірки текстових збігів. Такий підхід є свідомим архітектурним компромісом між точністю визначення та забезпеченням прозорості використання інструменту, адже пріоритетом для платформи загалом є мінімізація прихованого використання ШІ-редагування та формування культури чесності.

Також обмеження були виявлені у випадках, коли автор значно переписує отриманий від ШІ варіант фрагмента, що призводить до втрати характерних шинглів. У такій ситуації відсоток текстового перекриття може не досягати встановленого порогового значення, що, відповідно, не дозволяє алгоритму встановити мітку правильно.

Навіть з урахуванням виявлених обмежень, запропонованого механізму достатньо для поточної MVP-версії платформи.

4.1.3. Модульне тестування ключових механізмів

Додатково до ручного функціонального тестування інтелектуального модуля було проведено автоматизоване модульне. Під час нього основна увага була зосереджена виключно на 2 аспектах реалізованої логіки: механізмі допуску тексту до редагування з використанням інструменту (попередньому контролі) та алгоритмі виявлення наявності ШІ-відредагованого тексту.

Тестування проводилося за допомогою фреймворку Vitest, що дозволило провести швидку перевірку ізольованого чистого функціоналу. Всього було створено 27 тестових кейсів, які охопили як основні сценарії роботи, так і різні граничні випадки.

Через специфіку реалізації редактора та значну залежність попереднього контролю від UI-логіки його основний функціонал було винесено в окремий незалежний модуль. Це дозволило провести об’єктивне автоматизоване тестування без прямої прив’язки до React-компонентів та клієнтської взаємодії. Для цього було проведено комплексну перевірку функції

«validateAISelection(text)», яка відповідає за надання допуску до інструменту редагування (табл. 4.3).

Таблиця 4.3

Результати тестування функції «validateAISelection»

Сценарій	Очікуваний результат	Фактичний результат
Порожній рядок або тільки пробіли	isValid: false, reason: "noSelection"	Успішно
Текст < 200 символів	isValid: false, reason: "tooShort"	Успішно
Текст > 2000 символів	isValid: false, reason: "tooLong"	Успішно
Більше одного абзацу	isValid: false, reason: "tooManyParagraphs"	Успішно
Низька щільність літер	isValid: false, reason: "lowQuality"	Успішно
Валідний текст (200-2000 символів, 1 абзац, достатня щільність літер)	isValid: true, reason: "ready"	Успішно
Нормалізація множинних пробілів	isValid: true, reason: "ready"	Успішно

Під час тестування класу «AITracking» було проведено комплексну перевірку кожного з наявних методів: нормалізації тексту, розбиття на шингли, розрахунку коефіцієнту перекриття, пошуку наявності ШІ-відредагованого тексту та підготовки снапшотів. Отримані результати наведено у таблиці 4.4.

Таблиця 4.4

Результати тестування методів класу «AITracking»

Метод	Перевірені сценарії	Результат
normalize	Приведення тексту до lowercase, видалення HTML, видалення пунктуації, збереження Unicode-символів, нормалізація множинних пробілів.	Успішно
getShingles	Створення шинглів по 3 слова, робота з коротким та порожнім текстом.	Успішно
calculateOverlap	Повне співпадіння, часткове співпадіння, відсутність перетину, пустий снапшот.	Успішно
checkIfAIPresent	Точне співпадіння в загальному тексті, реакція на пустий снапшот, ігнорування коротких снапшотів, часткова схожість через використання шинглів, низька схожість.	Успішно
prepareSnapshot	Обрізання пробілів, обробка снапшоту тільки з пробілів.	Успішно

Усі 27 реалізованих unit-тестів виконано виявлення жодних помилок (рис. 4.1).

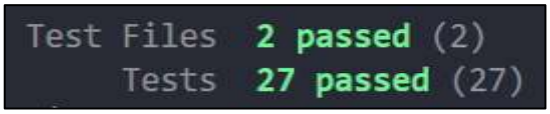


Рисунок 4.1 – Успішне виконання модульних тестів у Vitest

Додатково було проведено аналіз покриття протестованих модулів, результати якого продемонстрували високий рівень покриття основної реалізованої бізнес-логіки (рис. 4.2).

Рисунок 4.2 – Результати покриття модульних тестів (Vitest coverage)

Отримані результати додатково підтвердили коректність роботи основних функціональних механізмів розробки та стабільність реалізованої логіки як у типових, так і у граничних сценаріях використання.

4.2. Розгортання системи

Після проведення локального тестування, що довело працездатність розробленого інтелектуального інструмента автоматизованого редагування, здійснюється його інтеграція до загальнодоступної версії веб-платформи Dragon Tale. Через те, що система вже працює на хмарній платформі Vercel, внесення нового функціоналу не потребує ручного налаштування, адже воан забезпечує автоматизований деплой і спрощує оновлення веб-системи.

Загальний проєкт зберігається у репозиторії GitHub. Платформа Vercel під’єднана безпосередньо до гілки «main», що забезпечує автоматичний запуск процесу розгортання при кожному новому коміті. Після push-оновлень автоматично виконується збірка Next.js-додатку, перевірка залежностей, компіляція TypeScript та, якщо все пройшло без помилок, здійснюється миттєве оновлення загальнодоступної версії веб-сайту. У середньому весь процес внесення змін не перевищує 1-2 хвилин.

Під час інтеграції функціоналу важливим було безпечне налаштування збереження чутливих даних – ключа доступу для взаємодії з Groq Cloud. Для

цього було створено змінну середовища «GROQ_API_KEY», значення якої зберігається у зашифрованому вигляді в налаштуваннях Vercel і доступно виключно на серверній стороні (рис. 4.3).

Рисунок 4. 3 – Змінна середовища «GROQ_API_KEY» у Vercel

4.3. Аналіз ефективності розробленого модуля

Під час роботи над реалізацією інструмента було повністю пройдено шлях від проєктування до розробки, тестування і фінального впровадження в загальнодоступну версію Dragon Tale.

Відповідно до 6, поставлених на розробку, задач було виконано:

1. Модуль редагування повністю спроєктований, розроблений, протестований та інтегрований у вбудований редактор платформи. Взаємодія з обраною моделлю (Llama-3.1-8B-Instant) відбувається через Groq API, що забезпечує високу швидкість обробки.
2. Отримання згоди користувача на ШІ-обробку його тексту реалізується через обов’язкове модальне вікно підтвердження. Без явної згоди доступ до інструменти не надається, що унеможливлює випадкове його використання та сприяє більш свідомому застосуванню допомоги LLM.
3. Обмеження обсягу фрагменту для редагування відбувається через попередню валідацію, основою якої є перевірка виділеного тексту на дотримання встановлених умов (1 абзац від 200 до 2000 символів з достатньою «чистотою»). Завдяки цьому забезпечується ітеративний підхід до редагування та краща робота моделі зі збереженням авторського стилю.
4. Створена бібліотека шаблонів містить 5 варіантів редагування, що орієнтовані на роботу з художньою прозою: «літературне

шліфування», «чіткість і читабельність», «природний потік і плавність», «узгодженість тону», «граматика та технічна правильність». Їх наявність дозволяє уніфікувати взаємодію користувача з модулем, прибираючи потребу в самостійному формуванні промптів.

5. Повний контроль зі сторони користувача присутній на всіх етапах: від виділення фрагмента та надання згоди до вибору шаблона, порівняння й прийняття остаточного рішення щодо застосування або відхилення запропонованих змін. Таким чином окреслюється повна відповідальність автора за якість та зміст внесених змін до фінального тексту.
6. Для забезпечення прозорості застосування ШІ-редагування створений алгоритм на основі методу шинглів визначає відсоток входження снапшоту відредагованого фрагменту в загальний обсяг глави при збереженні. Отримане значення порівнюється з встановленою межею і на основі цього вже відбувається вирішення потреби встановлення мітки «ai_edited». Вона є перманентною та ненав’язливо відображається тільки у визначених місцях інтерфейсу (на сторінці глави та у переліку глав «шапки» твору).

Розроблений інтелектуальний модуль дозволяє зменшити вплив основних проблем, притаманних процесу редагуванню на сучасних літературних веб-платформ:

- Найголовніше – він інтегрований прямо до вбудованого редактора платформи, що усуває потребу в зверненні до сторонніх сервісів для отримання допомоги в обробці тексту.
- Концепція ітеративного підходу до редагування окремих фрагментів допомагає частково позбутися проявів «авторської сліпоти». Також робота з невеликими частинами тексту мотивує письменника аналізувати власний рукопис та контролювати внесені зміни.

- Висока швидкість обробки з доволі гарною якістю результату ШІ-редагування суттєво зменшує часові та енергетичні витрати. Це підтримує автора щодо одночасної швидкої публікації та хорошої якості тексту, частково зменшуючи постійний конфлікт вибору та балансування між ними.
- Отримання обов’язкової згоди користувача, обмеження обсягу обробки та принцип human-in-the-loop гарантують етичний характер використання штучного інтелекту у сфері художньої літератури.
- Мітка «Відредаговано з використанням ШІ» дозволяє формувати правильну культуру ставлення до подібних інструментів. Вона допомагає як авторам, так і читачам краще розуміти різницю між повною генерацією тексту та його допоміжним редагуванням, нормалізуючи відкрите й чесне використання ШІ.

Під час тестування модуль продемонстрував високу швидкість та стабільність роботи завдяки використанню Groq Cloud з моделлю Llama-3.1-8B-Instant. Користувацький інтерфейс реалізовано інтуїтивно зрозумілим для авторів з непомітним виділенням у загальній концепції платформи Dragon Tale.

Найслабшим місцем реалізації наразі виступає алгоритм визначення факту використання ШІ-редагування. Його обмеження та можливі хибні спрацювання є прямими наслідками свідомо прийнятих архітектурнихїрішень, адже пріоритет був зміщений у сторону забезпечення прозорості використання інструменту, а не у ідеальну точність визначення. Для поточної MVP-версії платформи алгоритм надає достатній рівень аналізу для встановлення мітки прозорості.

У результаті розроблений модуль автоматизованого редагування повністю відповідає всім, поставленим на розробку, задачам. Він виступає ефективним допоміжним інструментом, який зменшує технічне навантаження на автора, при цьому зберігаючи за ним повний контроль над текстом і забезпечуючи прозорість використання штучного інтелекту.

4.4. Напрями подальшого розвитку модуля

Реалізований модуль автоматизованого редагування художнього тексту, як і будь-який інший програмний продукт не має повністю ідеальної фінальної форми. Це дозволяє йому постійно покращуватись або під нові потреби користувачів, або під розвиток технологічних рішень.

На даний момент можна виділити 3 основні напрями, що зможуть вивести створений інструмент на відчутно вищий рівень:

1. Використання більш потужного ШІ – поточна модель Llama-3.1-8B-Instant надає прийнятний рівень редагування, а її застосування через платформу Groq Cloud забезпечує високу швидкість обробки. Але для об’єктивно кращої якості роботи необхідним є перехід на модель вищого рівня (GPT, Claude або іншої сучасної LLM), яка краще розумітиме тонкощі авторського стилю й структури художнього твору, а також точніше слідуватиме інструкціям.
2. Розширення та покращення бібліотеки шаблонів – очевидним покращенням є просте розширення наявного переліку варіантів редагування більш точковими та вузькоспеціалізованими шаблонами. Також важливим є вдосконалення системних інструкцій під подальший розвиток LLM, для більшого контролю поведінки ШІ та ще точнішого збереження авторського стилю.
3. Покращення алгоритму визначення використання ШІ-редагування – тестування поточної версії алгоритму продемонструвало його основні слабкі сторони, що повинні стати пріоритетом для оптимізації. У подальшому алгоритм можна розвивати методом поєднання ще більш комплексних варіантів перевірок, або ж впровадженням альтернативних підходів до аналізу змін тексту. У довгостроковій перспективі можлива навіть інтеграція детекторів ШІ, якщо їх точність і надійність все ж досягне прийнятного рівня для практичного застосування.

Висновки до розділу 4

У межах цього розділу було проведено успішне тестування інтегрованого у процес редагування інтелектуального модуля. Перевірка 14 ключових сценаріїв взаємодії з інструментом дозволила підтвердити працездатність реалізованого функціоналу, а тестування 7 окремих сценаріїв роботи алгоритму визначення використання ШІ-редагування – оцінити його поведінку в реальних умовах і виявити основні обмеження реалізації, що стали наслідками свідомо прийнятих архітектурних рішень у сторону забезпечення прозорості використання ШІ. Додатково було проведено модульне тестування ключових механізмів системи за допомогою фреймворку Vitest, що дозволило перевірити стабільність роботи попереднього контролю тексту та всіх методів алгоритму виявлення ШІ-відредагованих фрагментів у різних сценаріях використання.

Після завершення всіх тестувань було виконано інтеграцію розробленого функціоналу до загальнодоступної версії веб-платформи Dragon Tale через оновлення GitHub-репозиторію проєкта, під'єднаного до платформи Vercel.

Також було проведено аналіз ефективності розробленого інтелектуального модуля автоматизованого редагування, що продемонстрував відповідність реалізованого рішення поставленим на початку роботи задачам. У результаті інструмент став органічним доповненням вбудованого редактора платформи, надаючи користувачам зручний, контрольований та етичний спосіб покращення власного художнього тексту без втрати авторського стилю. Завдяки інтеграції безпосередньо у робочий процес, обов'язковому отриманні згоди, обмеженню обсягу фрагмента та механізму прозорості модуль дозволяє значно спростити процес редагування, знизити технічне навантаження на автора та забезпечити базовий рівень етичного використання штучного інтелекту в межах платформи.

Окремо було окреслено головні напрямки подальшого розвитку інструменту (використання потужнішої LLM, розширення бібліотеки шаблонів та покрашення алгоритму визначення використання ШІ), які у перспективі допоможуть перетворити поточне MVP-рішення на повноцінного інтелектуального асистента редагування художніх текстів.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було спроєктовано та практично реалізовано інтелектуальний модуль автоматизованого редагування художнього тексту для вбудованого редактора системи Dragon Tale. Він став результатом успішного виконання всіх поставлених завдань і собою підтверджує повне досягнення поставленої мети – дослідження ефективності контрольованого використання великих мовних моделей у процесі редагування художнього тексту та створення на основі цих результатів відповідного рішення з інтеграцією до існуючої літературної веб-платформи.

Для початку був проведений комплексний аналіз сучасних літературних платформ та процесу редагування на них. Виділено основні проблеми редагування, а також розглянуто обмеження доступних інструментів автоматизованої обробки тексту.

На основі розглянутих етичних аспектів застосування штучного інтелекту в сфері художнього письма було виділено головні причини, що викликають занепокоєння спільноти щодо LLM. Для практичної перевірки контрольованого використання ШІ при редагуванні, яке не повинно призводити до втрати авторського стилю, та для оцінки сприйняття такої обробки аудиторією проведено дослідження. Воно продемонструвало, що результати такого редагування дуже важко відрізнити від людського, та підтвердило доцільність такого контрольованого способу застосування ШІ, яке лягло в основу подальшої розробки.

Під час етапу проєктування після аналізу існуючої організації цільової системи Dragon Tale було сформовано основні вимоги до розробки та визначено низку архітектурних рішень. Ці рішення виступають основою етичного та контрольованого використання ШІ для редагування художнього тексту в межах літературних веб-платформах і включають: механізми попереднього контролю фрагмента, human-in-the-loop, обов’язкову чітку згоду користувача, бібліотеку готових шаблонів зі спеціальними структурованими системними запитами до

моделі, що мінімізують ризики втрати авторського стилю, а також алгоритм автоматичного встановлення мітки прозорості на основі снапшотів і методу шинглів.

Модуль було розроблено відповідно до визначених вимог і спроєктованого функціоналу з використанням технологічного стеку, який притаманний Dragon Tale (full-stack Next.js з TypeScript, Supabase),, що забезпечило подальшу безшовну інтеграцію. Взаємодія з обраною моделлю Llama-3.1-8B-Instant здійснюється через API платформи Groq Cloud, що гарантує високу швидкість виконання запитів. Створені 5 шаблонів редагування надають варіативність обробки тексту під різні потреби користувачів. Механізм обов'язкової початкової згоди дозволяє уникнути несвідомого використання інструменту. Загальна реалізована логіка надає модулю лише роль інтелектуального асистента, не порушуючи загальний процес редагування й авторський контроль за текстом. Мітка «Відредаговано з використанням ШІ» ненав'язливо демонструє сам факт застосування інструмента саме в допоміжній ролі, забезпечуючи бажану прозорість у межах платформи з метою формування розуміння різниці між генерацією нового контенту та контрольованим редагуванням.

Функціональне та модульне тестування підтвердили працездатність створеного рішення, що дозволило провести успішну інтеграцію модуля до загальнодоступної версії системи. Виявлені обмеження алгоритму визначення ШІ-редагування стали наслідками усвідомленого компромісу на користь надійності фіксації використання інструменту. Загалом інструмент у межах вбудованого редактора працює стабільно, надаючи необхідну допомогу авторам та забезпечуючи достатній рівень роботи механізмів прозорості.

Ця робота містить у собі певну наукову та практичну новизну. На відміну від існуючих інструментів, створений модуль працює безпосередньо у вбудованому редакторі платформи та забезпечує контрольоване й уніфіковане використання ШІ завдяки шаблонам, обмеженню обсягу фрагмента для ітеративного характеру роботи і механізму human-in-the-loop. Також він гарантує

прозорість застосування штучного інтелекту завдяки незворотній мітці. Усі ці аспекти дозволяють поєднати зручність сучасних мовних моделей з етичними способами їх використання в контексті літературних платформ.

Загалом, розроблений інтелектуальний інструмент автоматизованого редагування виступає ефективним сучасним рішенням частини проблем редагування на веб-платформах публікації творів. Він значно полегшує процес роботи авторів над текстом, підвищує загальну якість контенту та сприяє формуванню культури свідомого використання й адекватного сприйняття допомоги ШІ в сфері художнього письма. Подальший розвиток модуля може включати розширення бібліотеки шаблонів, використання потужнішої моделі та вдосконалення алгоритму визначення ШІ-редагування.

Код розробленого модуля доступний у відкритому репозиторії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Білоножно І. Онлайн-платформа Wattpad: місце для творчого самовираження чи кузня бестселерів? URL: https://www.methodwriting.com.ua/post/онлайн-платформа-wattpad-місце-для-творчого-самовираження-чи-кузня-бестселерів (дата звернення: 12.02.2026).

2. Archive of Our Own (Наш Власний Архів). URL: https://surl.lu/uniklw (дата звернення: 12.02.2026).

3. Webnovel. URL: https://webcatalog.io/uk/apps/webnovel (дата звернення: 13.02.2026).

4. Букнет // Вікіпедія. URL: https://uk.wikipedia.org/wiki/Букнет (дата звернення: 13.02.2026).

5. Що таке редагування тексту? URL: https://text.com.ua/info/shcho-take-redahuvannia-tekstu (дата звернення: 17.02.2026).

6. Сологуб Д. Редагування та коректура – в чому різниця? URL: https://lnk.ua/34yq8RZEg (дата звернення: 17.02.2026).

7. Дарада А. Що таке редагування тексту: види, завдання і значення. URL: https://redactor.com.ua/shho-take-redaguvannya-tekstu-vidi-zavdannya-i-znachennya/ (дата звернення: 18.02.2026).

8. Композиція // Вікіпедія. URL: https://lnk.ua/nzBtJIncD (дата звернення: 19.02.2026).

9. Levels of Stylistic Analysis. URL: https://www.scribd.com/document/997885429/Levels-of-Stylistic-Analysis#google_vignette (дата звернення: 19.02.2026).

10. Бриж Є. Як редагувати власний текст: 10 порад. URL: https://detector.media/infospace/article/245830/2025-11-24-yak-redaguvaty-vlasnyy-tekst-10-porad/ (дата звернення: 20.02.2026).

11. Peveto J. Grammarly Review: Is It Good for Novelists? (2025). URL: https://www.groundcreweditorial.com/blog/grammarly-review-2025 (дата звернення: 24.02.2026).

12. Peveto A. ProWritingAid Review: Is It Good for Novelists? (2024). URL: https://www.groundcreweditorial.com/blog/prowritingaid-review-2026 (дата звернення: 24.02.2026).

13. Frequently Asked Questions: What does Sudowrite do, really? URL: https://sudowrite.com/faq 2026 (дата звернення: 24.02.2026).

14. Myers A. AI-Detectors Biased Against Non-Native English Writers. URL: https://hai.stanford.edu/news/ai-detectors-biased-against-non-native-english-writers (дата звернення: 26.02.2026).

15. AI Best Practices for Authors. URL: https://authorsguild.org/resource/ai-best-practices-for-authors/ (дата звернення: 26.02.2026).

16. Collett, C. The Impact of Generative AI on the Novel. URL: https://www.mctd.ac.uk/wp-content/uploads/2025/11/MCTD-AIAndTheNovel-PolicyBrief-Accessible.html (дата звернення: 27.02.2026).

17. Milliot J. New Report Examines Writers' Attitudes toward AI. URL: https://www.publishersweekly.com/pw/by-topic/digital/copyright/article/99019-new-report-examines-writers-attitudes-toward-ai.html (дата звернення: 27.02.2026).

18. Wiberg K. Navigating AI Ethics in the Writing Process. URL: https://clearsightbooks.com/navigating-ai-ethics-in-the-writing-process/ (дата звернення: 28.02.2026).

19. AI for Authors: Ethical & Practical Guidelines. URL: https://selfpublishingadvice.org/ai-for-authors-guidelines/ (дата звернення: 28.02.2026).

20. Сліпий експеримент // Вікіпедія. URL: https://uk.wikipedia.org/wiki/Сліпий_експеримент (дата звернення: 09.03.2026).

21. Sotala K. Creative writing with LLMs, part 1: Prompting for fiction. URL: https://www.lesswrong.com/posts/D9MHrR8GrgSbXMqtB/creative-writing-with-llms-part-1-prompting-for-fiction (дата звернення: 25.03.2026).

22. How does question order impact survey responses? URL: https://yougov.com/en-us/articles/50446-how-does-question-order-impact-survey-responses (дата звернення: 09.03.2026).

23. Шкала Лайкерта // Вікіпедія. URL: https://uk.wikipedia.org/wiki/Шкала_Лайкерта (дата звернення: 11.03.2026).

24. Next.js Documentation. URL: https://nextjs.org/docs (дата звернення: 26.03.2026).

25. Supabase. URL: https://supabase.com/ (дата звернення: 28.03.2026).

26. Defence in depth and how it applies to web applications. URL: https://www.acunetix.com/websitesecurity/defence-in-depth-and-how-it-applies-to-web-applications/ (дата звернення: 28.03.2026).

27. Functional and Non-Functional Requirements. URL: https://www.geeksforgeeks.org/software-engineering/functional-vs-non-functional-requirements/ (дата звернення: 30.03.2026).

28. What is a Language Processing Unit? URL: https://groq.com/blog/the-groq-lpu-explained (дата звернення: 01.04.2026).

29. Ринденко Є.І. Архітектурні рішення інтеграції LLM для контрольованого редагування текстів у веб-платформу для публікації художніх творів. Матеріали конф. «Наукові відкриття та фундаментальні наукові дослідження: світовий досвід». Житомир, 2026. С. 168-171 (дата звернення: 18.04.2026).

30. Stryker C. What is human-in-the-loop? URL: https://www.ibm.com/think/topics/human-in-the-loop (дата звернення: 02.04.2026).

31. Ullah F. Literary vs Non-Literary Texts Explained. URL: https://www.scribd.com/document/621017041/Literary-and-Non-Literary-Stylistics (дата звернення: 02.04.2026).

32. Schulhoff S. et al. The Prompt Report: A Systematic Survey of Prompt Engineering Techniques. URL: https://arxiv.org/pdf/2406.06608 (дата звернення: 05.04.2026).

33. Ye J. et al. Structured Outputs in Prompt Engineering: Enhancing LLM Adaptability on Counterintuitive Instructions. URL: https://aclanthology.org/2025.wasp-main.13.pdf (дата звернення: 05.04.2026).

34. Chen B. et al. Unleashing the potential of prompt engineering for large language models. URL: https://arxiv.org/pdf/2310.14735 (дата звернення: 05.04.2026).

35. Liang X. et al. (2024). Controllable Text Generation for Large Language Models: A Survey. URL: https://arxiv.org/pdf/2408.12599 (дата звернення: 06.04.2026).

36. Алгоритм шинглів // Вікіпедія. URL: https://surl.li/brobgy (дата звернення 10.04.2026).

37. Overlap coefficient // Вікіпедія. URL: https://en.wikipedia.org/wiki/Overlap_coefficient (дата звернення: 10.04.2026).

38. API Routes. URL: https://nextjs.org/docs/pages/building-your-application/routing/api-routes (дата звернення: 12.04.2026).

39. Deploying to Vercel. URL: https://vercel.com/docs/deployments (дата звернення: 14.04.2026).

40. GroqCloud. URL: https://groq.com/groqcloud (дата звернення: 14.04.2026).

41. Documentation. Llama 3.1 8B. URL: https://console.groq.com/docs/model/llama-3.1-8b-instant (дата звернення: 15.04.2026).

42. react@18.2.0. 'use client'. URL: https://react.org.ua/reference/react/use-client (дата звернення: 16.04.2026).

43. react@18.2.0. useMemo. URL: https://react.org.ua/reference/react/useMemo (дата звернення: 16.04.2026).

44. Internationalization for Next.js. URL: https://next-intl.dev/ (дата звернення: 17.04.2026).

45. css властивість user-select. URL: https://css.in.ua/css/property/user-select (дата звернення: 17.04.2026).

46. OpenAI Compatibility. URL: https://console.groq.com/docs/openai (дата звернення: 30.04.2026).

47. PostgreSQL CREATE TRIGGER. URL: https://www.postgresql.org/docs/current/sql-createtrigger.html (дата звернення: 02.05.2026).

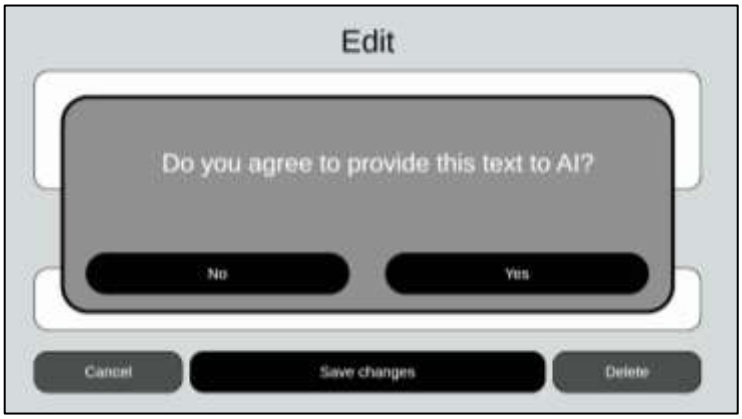
48. Nagori N. 10 Manual Testing Best Practices. URL: https://www.geeksforgeeks.org/software-testing/manual-testing-best-practices/ (дата звернення: 03.05.2026).

ДОДАТОК А

Макети інтерфейсу

А.1 Макет розташування кнопки активації інструменту та лічильника символів

А.2 Макет модального вікна згоди на обробку тексту ШІ



А.3 Макет модального вікна підтвердження заміни тексту