Сервіс на базі штучного інтелекту для допомоги у форматуванні академічних робіт

https://doi.org/10.31713/MCIT.2025.054

Вячеслав Шендеровський
Кафедра інженерії програмного забезпечення
Національний університет «Одеська політехніка»
Одеса, Україна
slavikshend@gmail.com

Вікторія Рувінська
Кафедра інженерії програмного забезпечення
Національний університет «Одеська політехніка»
Одеса, Україна
victoriya.ruvinskaya@gmail.com

Анотація— **У статті запропоновано сервіс на базі штучного інтелекту для автоматизованого форматування академічних робіт відповідно до тексту вимог конкретної конференції, журналу чи університету. Сервіс аналізує норму стилю оформлення, виявляє відхилення (відступи, шрифти, посилання, формули) і автоматично приводить текст до потрібного формату. Розроблене рішення покликане значно скоротити час на ручне форматування та знизити ймовірність помилок.**

Ключові слова—сервіс, штучний інтелект, форматування, академічні роботи.

I. ВСТУП

Форматування – одна з найбільш трудомістких і рутинних частин підготовки академічного документу. Часто автор мусить вручну приводити текст, таблиці, рисунки, списки літератури до специфікацій конференції, університету чи журналу – і ці вимоги можуть кардинально відрізнятися. При цьому стандарти форматування постійно змінюються – оновлюються шаблони, вводяться нові правила, іноді змінюються стиль цитування або вимоги до оформлення рисунків та таблиць.

Наприклад, дослідження LeBlanc et al. показало, що середній час форматування однієї статті – 14 годин, а в середньому це приносить втрати приблизно 52 години на рік для одного дослідника [1]. При цьому опитування показують, що авторів постійні зміни вимог до форматування відволікають від наукового змісту і знижують продуктивність. Також часто викладачі-керівники кваліфікаційних робіт, члени університетських комісій малого захисту витрачають багато особистого часу на ітеративну перевірку академічних робіт [2], часто власноруч, що може призводити до помилок через людський фактор.

Крім того, в Україні вже існують спроби автоматизувати перевірку академічних робіт: наприклад, створено систему перевірки кваліфікаційних робіт, яка дозволяє керівнику витрачати лише кілька хвилин на аналіз оформлення й змісту роботи замість десятків годин ручної перевірки [2]. Це демонструє як потребу, так і можливість розробки подібних інструментів для форматування в ширшому масштабі.

Метою цієї роботи є створення сервісу на базі штучного інтелекту, який допомагатиме автоматично форматувати академічні роботи під стандарти конкретної конференції або видання.

Об’єкт роботи – процес форматування текстових документів академічного характеру.

Предмет роботи – програмна система (сервіс), що аналізує документ і автоматично застосовує потрібні стилі, правила й шаблони форматування.

II. АНАЛІЗ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

A. Опис предметної області

Сервіс на базі ш тучного інтелекту для допомоги у форматуванні академічних робіт є прикладом сучасного вебзастосунку освітньо-наукового спрямування. Основна його мета – автоматизація процесів підготовки та оформлення текстових документів (курсових, дипломних робіт, наукових статей, тез конференцій тощо) відповідно до стандартів та вимог різних організацій.

У системі передбачені такі основні сутності: документ, користувач, підписка.

Для кожного документа зберігається його назва, вибрані вимоги для оформлення (наприклад, ДСТУ або власний документ з вимогами), статус форматування та фінальний відформатований результат;

Користувач має ім’я, прізвище, адресу електронної пошти, пароль, а також роль, що визначає його можливості у системі. Для клієнта доступні функції реєстрації, авторизації, завантаження документів, вибору стандарту форматування, оплати підписки та зверення до технічної підтримки. Адміністратор системи, окрім цього, має можливість переглядати звітність, керувати акаунтами клієнтів, а також відповідати на запити до технічної підтримки.

Для доступу до повного функціоналу користувач може оформити підписку. Вона містить інформацію про тип підписки (базова, розширена,

професіональна), дату початку та завершення, статус (активна, завершена, призупинена), а також кількість доступних для форматування сторінок документів на місяць.

B. Аналіз аналогів

Порівняльна характеристика наявних рішень представлена на таблиці 1.

ТАБЛИЦЯ І. Порівняльна характеристика наявних рішень

	Chat GPT	slite	textformatter .ai	Сист ема
Власні вимоги до форматування	+	-	-	+
Перевірка граматики, орфографії, пунктуації	+	-	-	+
Форматування по ДСТУ	+	-	-	+
Захищеність робіт	-	-	-	+

Отже, аналіз аналогів показав, що жодне з існуючих рішень не забезпечує повного набору функцій, необхідних для зручного та якісного форматування академічних робіт. Зокрема, сервіси ChatGPT, Slite та textformatter.ai частково виконують завдання перевірки граматики чи базового форматування, однак вони не підтримують форматування за стандартами ДСТУ та не гарантують захищеності робіт.

C. Визначення функціональних вимог до системи

Цей підрозділ визначає ключові вимоги до функціонала системи автоматизованого форматування академічних робіт із використанням штучного інтелекту. Функціональні вимоги можна представити у вигляді наборів прецедентів:

– для неавторизованих користувачів: «Авторизація», «Реєстрація».

– для авторизованих користувачів: окрім можливостей, що передбачені для неавторизованих користувачів, доступні такі прецеденти: «Оплата підписки», «Завантаження документа», «Форматування документа за заданими вимогами», «Форматування документа за стандартом ДСТУ»,

– для адміністраторів системи: окрім функцій клієнтів, також доступні: «Керування підписками», «Адміністрування шаблонів форматування» (додавання/оновлення стандартів), «Перегляд та аналіз звітності».

Діаграма прецедентів представлена на рис. 1.

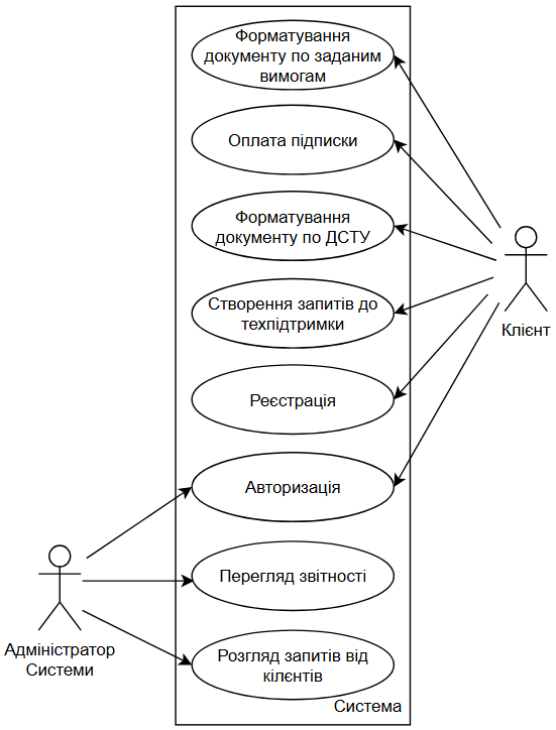

Рисунок 1. Діаграма варіантів використання

D. Визначення нефункціональних вимог до системи

Окрім функціональних вимог, мають бути визначені й нефункціональні вимоги, що описують загальну якість системи для роботи з документами та задають критерії для оцінки якості її роботи. Нижче наведені основні атрибути якості застосунку відповідно до структури стандартів ISO/IEC 25000 [4].

1. Сценарії надійності:

– система повинна забезпечувати коректність збереження документа з імовірністю не менше 99,9%;

– у випадку збою робота з документом повинна відновлюватися протягом не більше 2 хвилин без втрати даних;

– автоматичне резервне копіювання версій документа повинно виконуватись щонайменше раз на добу.

2. Сценарії ефективності:

– час відкриття документа має становити не більше 2 секунд для 90% користувачів;

– час перевірки документа на помилки (граматика, орфографія, стиль) не повинен перевищувати 5 секунд для тексту обсягом до 10 сторінок;

– максимальна кількість одночасних користувачів, що редагують документи, повинна бути не менше 100.

3. Сценарії зручності використання:

– середня тривалість внесення та збереження виправлень у документі має бути не більше 5 секунд;

– система повинна автоматично виправляти до 90% граматичних і стилістичних помилок;

– інтерфейс перевірки та виправлення повинен бути зрозумілим для користувачів без попереднього навчання.

4. Сценарії супроводжуваності:

– виправлення виявленої помилки у модулі перевірки документів має здійснюватися протягом 24 годин;

– додавання нового модуля перевірки (наприклад, перевірка пунктуації, стилістики) має здійснюватися протягом 2 тижнів;

– система повинна підтримувати швидку інтеграцію з іншими сервісами (наприклад, хмарними сховищами) без зміни базового коду.

5. Сценарії безпеки:

– доступ до документів має здійснюватися виключно авторизованими користувачами;

– система повинна забезпечувати захищене зберігання документів за допомогою шифрування;

– система має бути захищена від спроб несанкціонованого редагування або видалення документа з ефективністю не менше 99,9%.

III. Проєктування системи

A. Опис архітектури системи

У цьому розділі розглядається розроблювана система з погляду архітектурного проєктування.

Система належить до класу вебзастосунків для роботи з документами, тому у такому випадку найбільш оптимальним рішенням є реалізація проєкту на базі клієнт-серверної архітектури з мікросервісним підходом.

Щодо альтернативних архітектурних рішень, може скластися враження, що використання шаблону Модель-Вид-Контролер (MVC) могло б забезпечити необхідний функціонал. Проте типовий проєкт із використанням MVC передбачає монолітний характер системи, що ускладнює масштабування, тестування та інтеграцію з зовнішніми інструментами перевірки тексту. Крім того, при MVC сервер часто відповідає за рендеринг сторінок, що збільшує навантаження та знижує продуктивність.

На відміну від цього, мікросервісна архітектура з Web API дозволяє виділити окремі компоненти системи (перевірка тексту, зберігання даних, облік логів, робота з клієнтами) у незалежні сервіси, що взаємодіють між собою за допомогою REST та протоколу HTTP. Такий підхід підвищує гнучкість, забезпечує простоту оновлення та полегшує масштабування окремих модулів.

До складу системи входять наступні ключові компоненти:

1. Клієнтська частина (Client Side) – реалізована у вигляді односторінкового застосунку (SPA), що динамічно відображає дані та забезпечує зручний інтерфейс користувача. Рендеринг виконується на боці клієнта (CSR), що зменшує навантаження на сервер.

2. Web API – серверна частина, яка виконує роль посередника між клієнтом, базою даних та іншими мікросервісами. Саме API обробляє запити, реалізує бізнес-логіку та надає уніфікований доступ до даних і функціонала.

3. LanguageTool Service – окремий мікросервіс, відповідальний за автоматичний аналіз та виправлення орфографії, пунктуації. Його інтеграція з Web API дозволяє забезпечити перевірку граматики, орфографії та стилістики.

4. Database – основна реляційна база даних, що зберігає документи, користувацькі дані та результати перевірки.

5. Logs Database – додаткова спеціалізована база для зберігання логів взаємодії користувачів із системою, що дозволяє відстежувати помилки, вести аудит та моніторинг ефективності роботи сервісів.

На рис. 2 зображена архітектура системи.

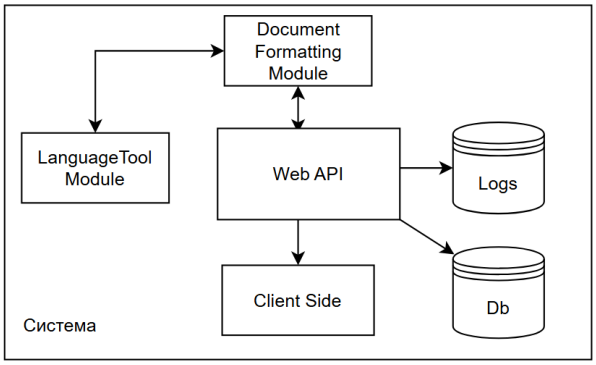

Рисунок 2. Архітектура системи

B. Визначення стеку технологій

Для реалізації системи було обрано сучасний стек технологій, який забезпечує гнучкість, масштабованість та простоту інтеграції з зовнішніми сервісами.

Серверна частина:

– Python FastAPI – фреймворк для створення високопродуктивних Web API. Забезпечує асинхронність, простоту опису REST-інтерфейсів та інтеграцію з іншими сервісами. Використовується для реалізації бізнес-логіки та координації взаємодії клієнта з базами даних і мікросервісами.

– Mammoth – бібліотека для перетворення документів у форматі DOCX у HTML. Використовується для збереження структури тексту без надлишкового форматування.

– Python-docx – бібліотека для створення та модифікації документів DOCX. Дає можливість працювати з документами у програмному режимі, наприклад, для генерації відформатованих файлів після перевірки.

Бази даних:

– PostgreSQL – реляційна база даних, що використовується для зберігання основних даних системи: документів, користувачів та результатів перевірки. Підтримує складні запити, транзакції та забезпечує цілісність даних.

– Apache Cassandra – розподілена база даних, яка застосовується для зберігання логів та великого обсягу неструктурованих даних. Завдяки горизонтальному масштабуванню Cassandra забезпечує високу доступність і стійкість до відмов.

Клієнтська частина: Angular – фреймворк для створення односторінкових застосунків (SPA). Використовується для реалізації клієнтського інтерфейсу системи, що забезпечує інтерактивну взаємодію з користувачем, динамічне відображення даних та інтеграцію з Web API.

Список літератури

[1] LeBlanc AG, Barnes JD, Saunders TJ, Tremblay MS, Chaput J-P (2019) Scientific sinkhole: The pernicious price of formatting. PLoS ONE 14(9): e0223116. https://doi.org/10.1371/journal.pone.0223116

[2] Кучерук, О. В., Желізко, В. В., Савіцький Р. С., & Фант, М. О. (2024). Система перевірки дипломних робіт як інструмент автоматизації роботи студента та викладача. Технічна інженерія, (1(93), с. 176–184.

[3] W. Perdomo, C. M. Zapata, “Software quality measures and their relationship with the states of the software system alpha,” Ingeniare. Revista chilena de ingeniería, vol. 29, no. 2, pp. 346–363, 2021.