ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Веб додаток для генерації текстових завдань на основі навчальних матеріалів»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
освітньо-професійної програми Штучний інтелект

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Андрій СИНГАЄВСЬКИЙ
(підпис)

Виконав: здобувач вищої освіти гр.ШІД-42
Андрій СИНГАЄВСЬКИЙ

Керівник: Євгеній ЧИЧКАРЬОВ
д.т.н, професор

Рецензент:
_____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ

Завідувач кафедрою Штучного інтелекту
_____ Ольга ЗІНЧЕНКО
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Сингаєвський Андрій Олегович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Веб додаток для генерації текстових завдань на основі навчальних матеріалів

керівник кваліфікаційної роботи д.т.н., професор, Євгеній Чичкарьов
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» 02.2025р. № 56

2. Строк подання кваліфікаційної роботи «23» травня 2025р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд сучасних технологій для розробки

5. Перелік графічного матеріалу: *презентація*

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	25.02-20.03.2025	Виконано
2	Визначення вимог до вебсистеми: функціонал, структура, вибір технологій (Flask, API, SQLite)	21.03-29.03.2025	Виконано
3	Розробка архітектури застосунку: модулі генерації, інтерфейс користувача, база даних	30.03-10.04.2025	Виконано
4	Інтеграція мовної моделі (OpenAI API / локальна GPT) для генерації текстових завдань	10.04-21.04.2025	Виконано
5	Створення інтерфейсу: форма завантаження матеріалів, перегляд згенерованих завдань, PDF-експорт	21.04.-29.04.2025	Виконано
6	Тестування функціоналу: генерація різних типів питань, стабільність роботи, швидкодія	29.04-02.05.2025	Виконано
7	Оформлення роботи: вступ, висновки, реферат	12.05-19.05.2025	Виконано
8	Розробка демонстраційних матеріалів	20.05-23.05.2025	Виконано

Здобувач вищої освіти _____ Андрій СИНГАЄВСЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Євгеній ЧИЧКАРЬОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 66 стор., 14 рис., 20 джерел.

Мета роботи: автоматизація процесу створення текстових завдань на основі навчальних матеріалів шляхом розробки веб-додатку з використанням технологій обробки природної мови.

Об'єкт дослідження: процес формування контрольних і тестових завдань у навчальному процесі.

Предмет дослідження: веб-додаток, який реалізує генерацію текстових завдань із використанням NLP та штучного інтелекту.

Короткий зміст роботи: У кваліфікаційній роботі проаналізовано сучасні підходи до автоматичної генерації текстових завдань у навчальному процесі з використанням технологій обробки природної мови (NLP) та штучного інтелекту. Розглянуто принципи побудови запитань різних типів (множинний вибір, пропуски, так/ні) на основі навчальних текстів, а також досліджено можливості використання мовних моделей (GPT, LLaMA) для цього завдання. Обґрунтовано доцільність застосування AI для автоматизації процесу створення контрольних матеріалів у закладах освіти. Реалізовано вебдодаток із підтримкою авторизації, завантаження навчальних матеріалів у форматі .txt/.docx/.pdf, потокової генерації тестів та експорту у формат PDF. Серверна частина реалізована з використанням Flask, клієнтська – HTML/CSS/JavaScript. Інтегровано NLP-модуль для аналізу ключових понять і генерації завдань у режимі реального часу. Проведено тестування роботи системи на реальних освітніх матеріалах та оцінено якість отриманих результатів, що підтвердило ефективність запропонованого рішення в навчальному середовищі.

КЛЮЧОВІ СЛОВА: ГЕНЕРАЦІЯ ЗАВДАНЬ, НАВЧАЛЬНИЙ ПРОЦЕС, NLP, ШТУЧНИЙ ІНТЕЛЕКТ, LLAMA, FLASK, PDF, ВЕБЗАСТОСУНОК, ТЕСТИ, АВТОМАТИЗАЦІЯ.

ЗМІСТ

ВСТУП

У сучасному освітньому процесі значна увага приділяється персоналізації навчання, оцінюванню знань та гнучкій адаптації навчальних матеріалів під індивідуальні потреби учнів. Одним із важливих інструментів цього процесу є тестові завдання, які дозволяють об’єктивно оцінити рівень засвоєння матеріалу. Проте створення якісних завдань потребує значних часових та інтелектуальних ресурсів з боку викладача. Особливо це стосується великих обсягів текстів або частих оновлень навчального контенту. Використання технологій обробки природної мови (NLP) та штучного інтелекту відкриває нові можливості для автоматизації процесу формування тестів. Завдяки NLP можливо здійснювати розпізнавання тематичних одиниць, класифікацію речень, аналіз семантики та синтаксису, що суттєво покращує якість автоматично згенерованих запитань. Сучасні мовні моделі, такі як GPT, LLaMA, Claude, PaLM, здатні не лише інтерпретувати зміст тексту, а й створювати різні типи завдань: з множинним вибором, заповненням пропусків, так/ні, а також формулювати відкриті питання. Інтеграція таких технологій у веборієнтовані додатки надає можливість створювати інтуїтивні та ефективні інструменти для викладачів, які дозволяють скоротити час на підготовку тестів, підвищити їхню об’єктивність і варіативність, а також адаптувати завдання під рівень знань учнів. Це особливо важливо в умовах дистанційного навчання та цифрової трансформації освіти.

Актуальність теми зумовлена потребою у створенні інноваційної системи, яка б дозволяла швидко та ефективно генерувати текстові завдання на основі навчальних матеріалів різного формату. Така система здатна зменшити навантаження на педагогів, підвищити об’єктивність оцінювання та адаптувати завдання до рівня складності навчального тексту.

Мета дослідження – розробити вебдодаток, що реалізує функціонал автоматичної генерації текстових завдань з використанням технологій обробки природної мови та мовних моделей штучного інтелекту.

Завдання дослідження:

– Провести аналіз сучасних підходів до генерації тестових завдань;

– Дослідити можливості NLP і мовних моделей у створенні запитань;

– Визначити вимоги до архітектури вебдодатку;

– Розробити модулі аналізу тексту, генерації питань та експорту результатів;

– Забезпечити потокове виведення результатів генерації в режимі реального часу;

– Провести тестування якості роботи системи на навчальних текстах.

Об’єкт дослідження – процес автоматизованого створення текстових завдань у навчальному середовищі.

Предмет дослідження – вебдодаток, що реалізує генерацію тестових завдань на основі навчального контенту з використанням NLP і мовних моделей.

Методи дослідження – аналіз літературних джерел, методи NLP (токенізація, витяг ключових слів, класифікація), розробка архітектури вебдодатку, Python/Flask-програмування, тестування на навчальних прикладах.

Практична значущість дослідження полягає в можливості використання розробленого вебдодатку в освітніх закладах різних рівнів — від загальноосвітніх шкіл до закладів вищої освіти — для автоматичного формування контрольних завдань. Це дозволяє викладачам економити час при підготовці матеріалів, уникати суб’єктивності при оцінюванні, створювати завдання різного типу і складності в залежності від потреб учнів або студентів. Крім того, система може бути використана в онлайн-курсах, корпоративному навчанні, при формуванні самостійних робіт або дистанційного тестування. Завдяки можливості збереження історії генерацій, система сприяє формуванню аналітики успішності та побудові адаптивних освітніх траєкторій, що є актуальним в умовах цифрової трансформації освіти.

1 ТЕОРЕТИЧНІ ОСНОВИ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ТЕКСТОВИХ ЗАВДАНЬ

1.1 Поняття та види тестових завдань у навчальному процесі

Тестові завдання є одним із найважливіших і найпоширеніших інструментів контролю знань у сучасній системі освіти. Вони забезпечують стандартизований підхід до оцінювання навчальних досягнень учнів і студентів, дозволяючи точно, об’єктивно та швидко визначити рівень засвоєння знань. Завдяки своїй структурованості та чітким критеріям перевірки тестові завдання значно підвищують ефективність контролю навчального процесу. Вони також зменшують суб’єктивність при оцінюванні, що особливо важливо в умовах масового навчання або дистанційної освіти.

Крім основної функції — перевірки знань — тестові завдання сприяють формуванню навчальної мотивації, розвитку аналітичного та логічного мислення, закріпленню знань через повторення. Вони можуть бути використані не лише для контролю, але й для самонавчання, адаптивного тестування, тренування пам’яті, формування навичок критичного осмислення матеріалу. Саме тому дедалі більшого поширення набувають електронні тестові системи, які дозволяють гнучко варіювати складність завдань, адаптувати їх до навчального контексту та відслідковувати прогрес учня. Тестові завдання можуть бути застосовані на різних етапах навчального процесу: на початку теми (вхідний контроль), у процесі навчання (поточний контроль), після завершення теми або розділу (підсумковий контроль), а також у формі самостійної перевірки знань, тренувальних завдань або фінального тестування. Вони також широко використовуються при вступі до освітніх закладів, при сертифікації та ліцензуванні, у корпоративному навчанні, що свідчить про універсальність цього інструменту.

У науковій та педагогічній літературі виділяють кілька основних типів тестових завдань, які відрізняються за структурою, метою та рівнем когнітивної

складності. Їх правильне поєднання дозволяє досягти найкращих результатів у навчальному процесі.

– Завдання з вибором однієї правильної відповіді (multiple choice) – один з найпоширеніших форматів, що передбачає вибір правильної відповіді з кількох запропонованих варіантів (зазвичай чотири або п’ять). Такі завдання мають високу надійність, легко піддаються автоматичній перевірці та дозволяють оцінити не лише фактологічні знання, а й навички аналізу та узагальнення. При правильному формулюванні дистракторів (неправильних варіантів) такі завдання є надзвичайно ефективними для перевірки розуміння теми.

– Завдання на встановлення відповідностей (matching) – полягають у необхідності зіставити два списки елементів, наприклад терміни та їх визначення, події та дати тощо. Вони активно використовуються для перевірки асоціативного мислення, здатності систематизувати інформацію та швидко знаходити логічні зв’язки.

– Завдання з відкритою відповіддю (open-ended) – це запитання, на які учень повинен самостійно сформулювати відповідь у вигляді слова, фрази або повноцінного тексту. Такі завдання дають змогу глибоко оцінити рівень осмислення матеріалу, критичне мислення, вміння аргументувати. Вони складніші в реалізації та автоматичному оцінюванні, але надзвичайно корисні для формувального оцінювання.

– Завдання на заповнення пропусків у тексті (fill-in-the-blanks) – передбачають вписування відсутніх слів або виразів у заданий контекст. Ці завдання зручні для перевірки лексичного запасу, граматичних навичок або логіки викладення матеріалу. Їх можна адаптувати як для початкового рівня, так і для поглибленого аналізу, додаючи підказки або обираючи складні формулювання.

– Правда/неправда (true/false) – передбачають оцінку конкретного твердження на предмет його істинності. Цей формат є найбільш легким для сприйняття, проте має обмежену діагностичну цінність через велику

ймовірність вгадування. Для підвищення ефективності часто використовують супутні завдання на виправлення помилкових тверджень.

– Послідовність (ordering/sequencing) – завдання, що потребують впорядкувати елементи у правильному хронологічному або логічному порядку. Це дозволяє перевірити глибоке розуміння процесів, історичних подій, етапів явищ або алгоритмів дій.

– Завдання на класифікацію (categorization) – вимагають розподілу понять або об’єктів за групами відповідно до певної ознаки. Ці завдання розвивають здатність систематизувати знання та бачити структуру в інформації.

Кожен тип завдань має свої переваги й обмеження, тому ефективна тестова система передбачає їх комбінацію залежно від цілей навчання, складності матеріалу, рівня підготовки аудиторії та технічних можливостей платформи. У рамках автоматизованих систем, особливо з підтримкою NLP, особливої популярності набувають multiple choice та fill-in-the-blanks через легкість їх реалізації, проте поєднання з відкритими запитаннями забезпечує глибше оцінювання.

Кожен тип завдань має свої переваги й обмеження, і їх вибір повинен бути ретельно обґрунтований з урахуванням низки педагогічних, методичних та технічних чинників.

По-перше, цілі тестування відіграють ключову роль: якщо метою є перевірка знань фактів, доцільно використовувати завдання типу «вибір правильної відповіді» або «правда/неправда», тоді як для оцінки аналітичного мислення доцільнішими є відкриті запитання або класифікаційні завдання.

По-друге, вікові особливості аудиторії впливають на складність формулювання запитань, обсяг варіантів, тип мови тощо. Молодшим учням слід пропонувати завдання з короткими формулюваннями та візуальною підтримкою, тоді як для студентів вищих навчальних закладів — більш складні завдання з логічними зв’язками між частинами.

По-третє, складність навчального матеріалу визначає тип когнітивної активності, яку необхідно задіяти: для поверхневого знання достатньо простих

тестів, а для глибокого розуміння важливо перевірити здатність до порівняння, аналізу, синтезу інформації.

По-четверте, технічні можливості програмного забезпечення чи вебдодатку є вирішальними у питанні реалізації того чи іншого формату. Наприклад, генерація та автоматична перевірка відкритих запитань потребує NLP-аналізу, тоді як multiple choice можна перевірити простим порівнянням з шаблоном.

Таким чином, ефективна система генерації тестових завдань має враховувати всі ці параметри одночасно, адаптуючи типи запитань до конкретного навчального контексту, забезпечуючи гнучкість, точність та педагогічну доцільність автоматизованого оцінювання знань..

1.2 Методи обробки природної мови (NLP) у сфері освіти

Обробка природної мови (NLP — Natural Language Processing) є міждисциплінарною галуззю, що об'єднує знання з лінгвістики, інформатики, штучного інтелекту та когнітивних наук з метою розуміння й обробки природної людської мови за допомогою комп’ютерних алгоритмів. Основне завдання NLP полягає у перетворенні неструктурованого тексту в структуровану форму, придатну для автоматичного аналізу, прийняття рішень, навчання моделей або генерування нових текстів. Ці можливості мають широке застосування у сфері освіти, де ефективна взаємодія з текстовою інформацією є критично важливою для успішного засвоєння знань. У навчальному процесі NLP забезпечує автоматизацію цілої низки рутинних і ресурсозатратних завдань. Це охоплює перевірку граматики та орфографії, автоматичне оцінювання відкритих відповідей, створення текстів для вправ, резюмування навчальних матеріалів, автоматичний переклад, виявлення плагіату, формування індивідуальних підсумкових звітів та зворотного зв’язку для учня. Одним з найперспективніших напрямів є автоматична генерація тестових завдань на основі текстів навчальних дисциплін. За допомогою NLP можна не лише обробляти синтаксичні конструкції, а й формувати завдання, що ґрунтуються на семантиці, контексті, структурі тексту та ключових іменованих

сутностях. Наприклад, алгоритми можуть виявляти найважливіші елементи тексту, що мають дидактичне значення: дати, терміни, події, причиново-наслідкові зв’язки, класифікації. Далі з цих даних формуються запитання різного типу: на встановлення фактів, заповнення пропусків, виявлення логічних зв’язків, перевірку розуміння абзацу або тексту в цілому. Такий підхід дозволяє не лише автоматизувати процес створення матеріалів, а й підвищити їхню якість через глибший аналіз змісту та структури тексту.

Системи з NLP здатні також адаптувати сформовані тести під рівень складності: наприклад, формулювати ті самі запитання різними словами або перетворювати складні запитання у спрощені варіанти. Крім того, NLP дає можливість багатомовного навчання, дозволяючи автоматично перекладати або формулювати запитання іншими мовами. Завдяки поєднанню NLP з інтерфейсами штучного інтелекту, такими як великі мовні моделі (LLMs), можна досягти рівня генерації, максимально наближеного до людського. Це відкриває перспективи для створення інтелектуальних освітніх систем нового покоління, в яких взаємодія з текстом буде не лише інтерактивною, а й змістовною, гнучкою та навчально-орієнтованою.

Серед основних методів NLP, що застосовуються в освіті, можна виділити:

– Токенізація — розбиття тексту на менші одиниці: речення, слова, символи або навіть морфеми. Це один з найперший і базових етапів обробки тексту, який дозволяє перевести текст у формат, придатний для подальшого аналізу. Залежно від поставленого завдання, токенізація може бути застосована до речень або слів, що особливо важливо для правильного визначення меж синтаксичних конструкцій. У сучасних системах токенізація може бути контекстно-залежною, з урахуванням пунктуації, скорочень, лексичних злиттів, що актуально для обробки неформальних або навчальних текстів.

– Стемінг і лематизація — процеси приведення слів до їх базової форми. Стемінг — це усічення слова до основи без врахування граматичних правил (наприклад, "навчання" → "навч"), тоді як лематизація враховує контекст і

повертає коректну словникову форму ("навчання" → "навчати"). Ці методи відіграють важливу роль у нормалізації лексичних форм, що дозволяє ефективніше зіставляти синонімічні або граматично змінені слова в навчальному тексті. Для генерації завдань це критично важливо, особливо при формуванні варіантів відповіді, пошуку синонімів або створенні запитань на узагальнення.

– Part-of-Speech Tagging (POS) — визначення частин мови для кожного слова в реченні, що дозволяє встановити граматичну структуру тексту. Цей етап важливий для того, щоб розуміти, яке слово є іменником, дієсловом, прикметником тощо. У контексті освіти POS-тегінг дозволяє створювати граматично правильні запитання, наприклад, на виявлення означень, присудків або заповнення пропусків певними частинами мови. Крім того, він є основою для синтаксичного аналізу та створення завдань із граматичного контролю.

– Named Entity Recognition (NER) — метод визначення та класифікації іменованих сутностей у тексті, таких як імена людей, географічні об’єкти, дати, назви організацій, історичні події тощо. Для генерації запитань у навчальному процесі це відкриває широкі можливості: формулювання запитань типу «Хто відкрив…?», «Коли відбулася подія…?», «У якому місті…?» тощо. Використання NER дозволяє автоматизувати створення завдань з історії, географії, літератури, де ключові факти пов’язані з конкретними об’єктами. У сучасних реалізаціях NER застосовує глибоке навчання та контекстуальне розпізнавання, що підвищує точність навіть у складних, двозначних фразах.

– Dependency Parsing — побудова дерев синтаксичних залежностей, яка дозволяє аналізувати логіку побудови речень, визначати головні і залежні члени речення, виявляти підмети, присудки, додатки, обставини та інші елементи. Цей метод є надзвичайно важливим при генерації граматично правильних і логічно узгоджених запитань, а також при побудові складнопідрядних конструкцій. У навчальних задачах dependency parsing дозволяє автоматично визначити, до якого слова поставити запитання, яке

слово приховати для створення завдань із пропусками або які зв'язки є ключовими для формування питального речення.

– TF-IDF і word embeddings (наприклад, Word2Vec, GloVe) — статистичні та векторні методи представлення слів у вигляді чисел для подальшого машинного аналізу. TF-IDF (term frequency-inverse document frequency) дозволяє визначати важливість слова в межах документа або корпусу, що допомагає виявляти ключові терміни, на основі яких можуть бути згенеровані запитання. Word embeddings перетворюють слова у багатовимірні вектори з урахуванням їх контекстуального значення. Це дає змогу системі «розуміти» семантику тексту: знаходити синоніми, класифікувати терміни, уникати повторень, а також формулювати запитання, що перевіряють не лише знання, але й розуміння суті.

– Класифікація тексту — метод автоматичного розподілу текстового контенту за тематиками, категоріями або рівнем складності. У сфері освіти текстова класифікація може застосовуватись для попередньої обробки навчальних матеріалів, зокрема для визначення тематики параграфа або типу запитання, що має бути згенероване. Наприклад, тексти з домінуванням фактологічної інформації — для створення запитань типу «хто/що/коли», а тексти з аналітичними міркуваннями — для відкритих запитань. Класифікація також дозволяє сегментувати тексти на логічні блоки для покращення якості побудови тестових сесій.

– Автоматичне перефразування — процес зміни структури речення без зміни його змісту. У генерації тестових завдань цей підхід застосовується для створення кількох варіантів запитання з однаковим смисловим навантаженням, що дозволяє варіювати тести та уникати шаблонного повторення. Наприклад, можна перетворити запитання «Який рік вважається початком події?» на «У якому році відбулася подія?». Такий підхід не лише покращує якість оцінювання, а й стимулює учнів до більш глибокого осмислення навчального матеріалу.

– Генерація тексту — створення нових фраз, речень або запитань на основі заданого тексту. Це один із найскладніших, але й найперспективніших методів NLP, особливо в поєднанні з великими мовними моделями (LLMs), такими як GPT-4, LLaMA, Claude, Gemini. Такі моделі навчені на мільярдах слів і здатні продукувати людськоподібні тексти, розуміти контекст, логіку викладу, стилістику. У сфері освіти генерація тексту дає змогу створювати адаптивні, варіативні тести, пояснення до завдань, зворотній зв’язок. Генеративні можливості LLM використовуються також для побудови відкритих запитань, варіантів відповідей, пояснень до них і навіть оцінювання розгорнутих відповідей учнів.

Застосування NLP у сфері освіти має цілу низку глибоких переваг, які змінюють підхід до створення, подання й оцінювання навчального контенту. Автоматизовані системи на основі NLP значно пришвидшують процес підготовки навчальних матеріалів, дозволяючи миттєво створювати варіативні завдання, генерувати запитання на основі будь-якого тексту, адаптувати складність до потреб конкретної аудиторії. Вони сприяють індивідуалізації навчання: замість одноманітних контрольних робіт учні отримують персоналізовані завдання, що відповідають їхньому рівню знань, прогресу й інтересам. Зменшується навантаження на викладача — найбільш рутинні й повторювані операції (створення тестів, перевірка відкритих відповідей, оцінювання грамотності) виконуються автоматично, що дозволяє зосередитись на методичній та комунікативній складовій навчання. NLP забезпечує гнучкість у форматах навчання: система може працювати як у традиційних аудиторних умовах, так і в умовах дистанційного чи змішаного навчання.

Особливо важливо, що NLP забезпечує масштабованість — можливість одночасної роботи з великою кількістю учнів, збирання статистики відповідей, аналізу помилок і створення рекомендацій для вдосконалення освітнього процесу. Крім того, системи на основі NLP сприяють забезпеченню доступності освіти для різних категорій користувачів, зокрема осіб з порушеннями мовлення або письма, завдяки інтеграції з мовними інтерфейсами,

синтезаторами мовлення та інструментами спрощеного викладу змісту. Отже, NLP не лише оптимізує процес навчання, а й трансформує його у більш гнучку, інклюзивну, ефективну та технологічно орієнтовану систему. NLP відкриває можливість до створення адаптивних тестів, які динамічно змінюють рівень складності запитань у реальному часі залежно від рівня підготовки та попередніх відповідей учня. Це дозволяє створювати персоналізовані траєкторії оцінювання, що краще відображають індивідуальний прогрес кожного здобувача освіти. Такі тести можуть знижувати або підвищувати складність наступного питання, в залежності від успішності попередніх відповідей, що формує ефект «інтелектуального супроводу». NLP підтримує багатоетапну генерацію завдань, яка охоплює кілька ключових фаз: вибір тематики, автоматичний аналіз навчального тексту, визначення ключових понять та іменованих сутностей, класифікація інформації за типами запитань, формування кількох варіантів завдань (із закритою або відкритою формою), перевірка результатів та їх інтерпретація. Це дозволяє формувати тести, які враховують не лише фактологічну, а й семантичну, логічну та критичну складову засвоєння матеріалу. NLP-технології дають змогу відстежувати час відповіді, розуміння контексту, навіть граматичні помилки, що розширює спектр аналізу результатів. Із впровадженням багаторівневого логічного оцінювання та когнітивної аналітики тестування стає не просто перевіркою знань, а повноцінною інтегрованою частиною освітнього процесу, яка розвиває вміння, рефлексію та здатність до самонавчання.

Завдяки сучасним NLP-алгоритмам, які можуть працювати як автономно, так і в зв’язці з мовними моделями великого масштабу, освітній процес отримує нову якість. Він стає більш гнучким, доступним, ефективним, адаптивним і здатним до масштабування, що є критично важливим в умовах цифрової трансформації освіти у XXI столітті.

1.3 Використання мовних моделей (GPT, LLaMA) для генерації запитань

Сучасні мовні моделі, зокрема GPT (Generative Pre-trained Transformer) від OpenAI та LLaMA (Large Language Model Meta AI) від Meta, здійснили справжню революцію в обробці природної мови та автоматизованій генерації текстового контенту. Вони відкрили нові перспективи в реалізації інтелектуальних освітніх технологій, зокрема в галузі автоматичного створення навчальних завдань, тестів і підсумкових перевірок знань. Ці моделі належать до класу великих мовних моделей (LLM — Large Language Models), які натреновані на масивних корпусах тексту — від енциклопедичних статей і наукових праць до форумів, художньої літератури та роз transkribovanoї усної мови. Завдяки такому різноманітному навчальному матеріалу, моделі демонструють здатність не тільки до граматично правильного формування речень, але й до логічної побудови змісту, розпізнавання контексту та розуміння стилістики запитань. Моделі GPT і LLaMA побудовані на основі трансформерної архітектури — інноваційної структури, яка дозволяє ефективно обробляти контексти довжиною до тисяч слів, виявляючи неочевидні залежності між віддаленими фрагментами тексту. Це критично важливо при генерації змістовно обґрунтованих тестових завдань, де важливо врахувати не лише окремі факти, але й загальний зміст, логіку викладу та стилістичні особливості матеріалу. На практиці це означає, що модель може зчитати весь навчальний текст, виокремити з нього найважливіше, згенерувати до нього запитання відповідного типу, а також запропонувати правдоподібні варіанти відповідей — і все це без втручання людини. Крім суто технічної здатності формувати запитання, ці моделі вирізняються високим ступенем адаптивності. Вони можуть бути навченою на конкретних корпусах текстів у межах однієї дисципліни, що забезпечує відповідність термінології, стилю й очікуваної складності запитань. GPT та LLaMA здатні генерувати завдання як для початкової школи, так і для магістерських програм або професійних сертифікацій. Завдяки застосуванню техніки zero-shot або few-shot learning,

моделі можуть генерувати коректні завдання навіть без спеціального донавчання — достатньо надати приклад або опис бажаного результату. У поєднанні з можливістю використання інструкцій (prompt engineering), це дозволяє викладачеві формувати чітке технічне завдання до моделі: стиль запитань, формат відповіді, рівень складності, мова тощо.

GPT і LLaMA — це не просто інструменти текстогенерації, а повноцінні освітні партнери, які здатні адаптуватись під конкретні освітні завдання, знижувати навантаження на викладача, забезпечувати індивідуальний підхід до кожного учня та значно підвищувати ефективність навчального процесу.

Процес генерації запитань за допомогою мовних моделей включає кілька етапів, кожен із яких базується на застосуванні передових методів обробки природної мови. Спочатку система приймає на вхід навчальний текст або ключову тему, пов'язану з певною навчальною дисципліною. Модель виконує його попередній аналіз — здійснює токенізацію, лематизацію, синтаксичний та семантичний аналіз, а також визначає ключові сутності: іменовані об’єкти (історичні особи, географічні назви), поняття, дати, терміни, зв’язки між подіями тощо. Це дозволяє формувати змістовну карту тексту, на основі якої відбувається побудова запитань. Далі модель, керуючись мовними шаблонами, навчальним контекстом і запитаною складністю, формує відповідні запитання. При цьому використовуються різні типи завдань: фактологічні ("Хто...?", "Коли...?"), пояснювальні ("Чому...?", "Яка мета...?"), аналітичні ("Які наслідки...?", "Порівняйте..."), завдання з вибором відповіді, на встановлення відповідностей, з відкритими відповідями тощо. У більш просунутих реалізаціях системи можуть генерувати завдання з елементами критичного мислення або багаторівневої логіки. Однією з ключових переваг використання GPT або LLaMA є здатність до генерації варіативних формулювань, що дозволяє уникати повторень і шаблонності. Наприклад, на основі одного й того ж фрагмента тексту модель може створити кілька запитань з різним кутом аналізу: одне — на факт, друге — на причину, третє — на інтерпретацію. Це формує багатогранне оцінювання знань і дозволяє краще відобразити рівень

розуміння учнем змісту матеріалу. Завдяки великій кількості прикладів, на яких були натреновані ці моделі, вони мають здатність до генерації контекстно точних і лінгвістично природних запитань, які виглядають так, ніби їх створила людина. Також моделі демонструють здатність до логічного міркування, розпізнавання глибших семантичних зв’язків, виявлення імпліцитної інформації та навіть часткове розуміння наміру користувача (user intent), що робить можливим інтерактивну й адаптивну генерацію завдань в освітніх застосунках нового покоління.

Іншим важливим аспектом є підтримка декількох мов, що дозволяє створювати запитання різними мовами або здійснювати переклад вже згенерованих тестів для багатомовної аудиторії. Це особливо актуально у міжнародних програмах або онлайн-курсах, де присутні користувачі з різними мовними профілями, а також у мультикультурному навчальному середовищі, яке потребує інклюзивного підходу до подачі навчального контенту. Мовні моделі з можливістю багатомовного аналізу можуть самостійно адаптувати стиль мовлення, враховуючи лексичні, граматичні та культурні особливості різних мов, що сприяє підвищенню якості й точності формулювання запитань для іноземних студентів. Крім багатомовності, мовні моделі також можуть бути налаштовані під конкретні потреби навчального закладу, дисципліни або методичних рекомендацій викладача. За допомогою інструкцій (prompt engineering) можна точно вказати стиль, структуру, бажану складність, тип запитань (відкриті, закриті, комбіновані), а також формат відповіді (текстовий, числовий, варіантний). Наприклад, можна сформулювати запит до моделі такого типу: "Створи п’ять запитань з варіантами відповідей для 8 класу з історії України, складність — базовий рівень, тема — козацька доба". У відповідь модель згенерує релевантні запитання з урахуванням стилістики, вікового рівня, тематики та типу оцінювання. Мовні моделі перетворюються на потужний дидактичний інструмент, який дозволяє реалізовувати персоналізоване навчання, дотримуватись освітніх стандартів, забезпечувати академічну доброчесність і гнучко реагувати на потреби конкретної аудиторії.

Високий рівень керованості процесом генерації дає змогу викладачу повноцінно використовувати модель як асистента, а не як непрозору «чорну скриньку». Це відкриває нові горизонти в розробці інтелектуальних освітніх систем та автоматизованих платформ оцінювання знань.

У рамках вебдодатків, таких як розроблений у цій роботі, інтеграція мовних моделей GPT або LLaMA дозволяє реалізувати високоефективну потокову генерацію запитань у режимі реального часу. Це означає, що користувач (викладач або студент) не лише завантажує навчальний текст у систему, а й одразу отримує інтерактивну відповідь — у вигляді набору автоматично сформованих завдань. Система виконує попередню обробку: розпізнає ключові терміни, виявляє структуру тексту, будує семантичні залежності, і на основі цього формує запитання, що відповідають визначеним критеріям. Кожне сформоване запитання може супроводжуватись варіантами відповідей, поясненням вибору правильної відповіді, контекстним посиланням на фрагмент тексту та навіть інтерактивною підказкою. Це дозволяє не тільки оцінити знання, а й сформувати зворотній зв’язок, пояснити помилку, закріпити матеріал. Такий підхід дозволяє створювати повноцінні адаптивні навчальні платформи нового покоління, які поєднують автоматизацію перевірки знань із можливістю навчання в процесі тестування.

Використання GPT або LLaMA у вебсередовищі надає додаткові переваги: можливість масштабування для великої кількості користувачів, безперервного навчання моделі на нових даних, інтеграції з іншими освітніми системами (електронні щоденники, LMS, CRM), ведення статистики та формування аналітики результатів. Розроблений вебдодаток може виступати не лише інструментом тестування, а й середовищем для аналізу прогресу учнів, планування подальших етапів навчання та автоматичного формування підсумкових оцінок.

Застосування мовних моделей у сфері генерації тестових завдань є не лише технічним удосконаленням, а й педагогічною трансформацією, що впливає на всі рівні освітнього процесу. Воно змінює філософію взаємодії між

викладачем і учнем: роль викладача поступово переходить від носія знань до фасилітатора та куратора навчального середовища, тоді як учень перетворюється з пасивного реципієнта на активного учасника навчального процесу. Мовні моделі дають змогу формувати не лише завдання, а й контекст навколо них: пояснення, приклади, рекомендації до подальшого вивчення. Це створює основу для розвитку м'яких навичок, таких як критичне мислення, аналітична обробка інформації, здатність до обґрунтованих суджень. Такі системи сприяють формуванню навичок самонавчання, адже учень має змогу взаємодіяти із завданням динамічно: не лише відповідати, а й уточнювати, переглядати підказки, бачити логіку формування запитання. Таким чином, знання формуються не лише як результат відтворення інформації, а як наслідок активної розумової діяльності. У результаті формується більш глибоке розуміння матеріалу, яке не обмежується поверхневим відтворенням фактів.

Варто відзначити, що такі системи впливають на методику оцінювання: автоматизація дозволяє перейти від одноразових контрольних заходів до системного моніторингу прогресу, формувального оцінювання, виявлення прогалин у знаннях. Аналітичні можливості моделей забезпечують побудову індивідуальних освітніх траєкторій, адаптивну складність завдань, персоналізовані звіти для учня та викладача. Такі рішення відкривають новий вимір освіти — коли навчання стає безперервним, інтерактивним, орієнтованим на результат і заснованим на даних. Мовні моделі, впроваджені в освітнє середовище, дозволяють переосмислити саму суть навчального процесу — зробити його не лише ефективнішим, а й більш захопливим, індивідуалізованим та глибоко інтегрованим у цифрову культуру сучасного покоління учнів і студентів.

Висновок до розділу 1

У першому розділі було розглянуто теоретичні основи автоматичної генерації тестових завдань, що базуються на сучасних технологіях обробки природної мови (NLP) та використанні великих мовних моделей (LLM), таких як GPT та LLaMA. Детально проаналізовано місце й значення тестових завдань

у навчальному процесі, класифікацію їх видів, педагогічні цілі, які вони покликані реалізувати, а також їх роль у забезпеченні об’єктивного оцінювання знань. Значна увага приділена сучасним NLP-методам: токенізації, лематизації, синтаксичному аналізу, розпізнаванню сутностей, класифікації та генерації тексту. Було встановлено, що ці інструменти забезпечують ефективну трансформацію тексту в завдання різного типу, з урахуванням навчального контексту та освітніх цілей. Особливо важливим є застосування мовних моделей, здатних не лише розуміти зміст, а й продукувати логічно та лінгвістично коректні запитання, що дозволяє автоматизувати створення дидактичних матеріалів із високим ступенем адаптації та персоналізації.

У результаті аналізу зроблено висновок, що використання NLP і мовних моделей у сфері освіти відкриває новий рівень якості в оцінюванні знань, підвищує ефективність навчального процесу, дозволяє зменшити навантаження на викладача, а також забезпечує можливість створення адаптивних, індивідуалізованих тестових систем. Ці технології створюють умови для трансформації освітньої моделі: від статичного відтворення знань — до гнучкого, інтерактивного й інтелектуального навчання з постійним зворотним зв’язком..

2 АНАЛІЗ, ПРОЄКТУВАННЯ ТА АРХІТЕКТУРА ВЕБДОДАТКУ

2.1 Аналіз існуючих сервісів автоматичної генерації тестів

У сучасному освітньому середовищі, що стрімко трансформується під впливом цифрових технологій, все більшої популярності набувають онлайн-інструменти, які дозволяють автоматизувати створення тестів, контрольних завдань і навчальних вправ на основі текстових матеріалів. Ці сервіси стали необхідною складовою сучасної педагогіки, орієнтованої на гнучкість, адаптивність, масштабованість та оперативний зворотний зв’язок. Їхнє використання сприяє зменшенню навантаження на викладачів, підвищенню об'єктивності оцінювання та забезпеченню індивідуального підходу до навчання. Онлайн-сервіси автоматичної генерації тестів використовуються не лише в традиційних освітніх установах, таких як заклади загальної середньої чи вищої освіти, а й у корпоративному навчанні для підготовки персоналу, сертифікаційних програмах, системах дистанційного навчання (LMS), підготовці до зовнішніх незалежних іспитів (ЗНО, IELTS, TOEFL), та навіть у мобільних застосунках для самонавчання. Більше того, такі інструменти дозволяють інтегрувати штучний інтелект у навчальний процес, трансформуючи традиційне оцінювання у більш динамічну та персоналізовану форму, яка відповідає потребам цифрової епохи. Оскільки попит на подібні рішення постійно зростає, з’являються нові платформи, що пропонують часткову або повну автоматизацію генерації запитань. Проте рівень їх функціональності, гнучкості, якості та адаптації до освітніх цілей істотно різниться, що зумовлює актуальність критичного аналізу наявних рішень і потребу у створенні ефективнішого, універсального вебзастосунку.

Серед найвідоміших сервісів можна виділити:

Google Forms з доповненням Quizizz (рис. 2.1) або Certify'em. Цей інструмент широко використовується в освітньому середовищі завдяки простоті створення форм і можливості додавання тестових запитань. Quizizz і Certify'em доповнюють функціонал Google Forms інтерактивними іграми,

автоматичним оцінюванням та сертифікацією. Проте ці інструменти не підтримують автоматичну генерацію тестів на основі тексту, не мають NLP-аналізу та потребують повністю ручного введення запитань.

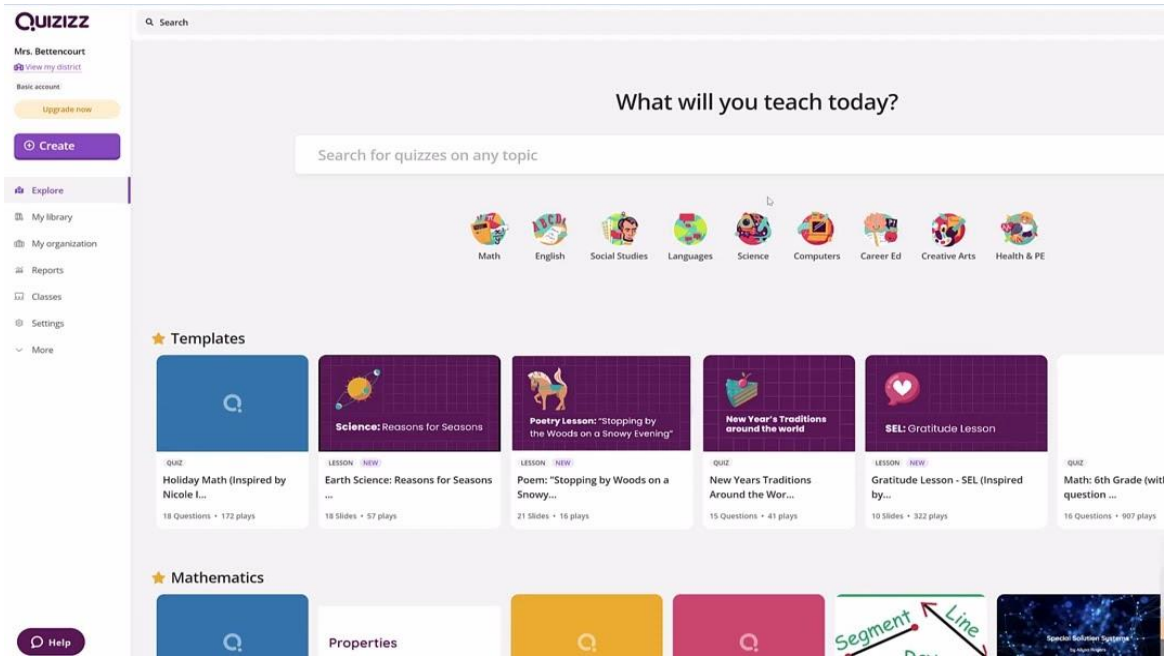


Рисунок 2.1 - Google Forms з доповненням Quizizz

Quizlet.Це популярний сервіс створення флеш-карток, тестів і тренувальних вправ. Quizlet дозволяє створювати навчальні модулі, які включають терміни, визначення, тести, ігрові завдання. Проте система обмежена шаблонною генерацією і не підтримує повноцінний аналіз тексту. Всі запитання вводяться вручну або формуються за примітивними правилами.

Testportal.net, Easy LMS, ClassMarker (рис. 2.2). Ці платформи орієнтовані на створення професійних онлайн-тестів для навчання, сертифікації, HR. Вони підтримують гнучке налаштування структури тестів, таймерів, оцінювання та звітів. Проте в усіх випадках тестові завдання створюються вручну або імпортуються з шаблонів. Відсутній аналіз тексту та автоматична генерація запитань.

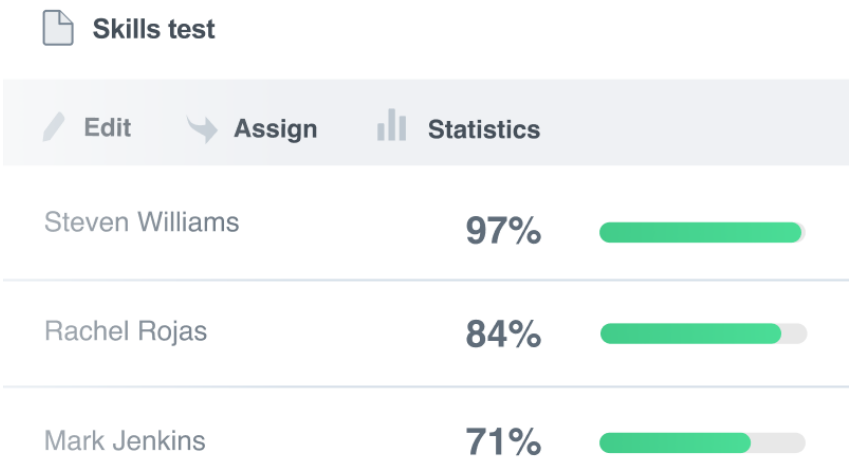


Рисунок 2.2 - Testportal.net

ChatGPT + Prompt Generator (рис. 2.3).У цьому підході користувач використовує генеративну модель ChatGPT у поєднанні з вручну підготовленими інструкціями (prompt'ами). Це дозволяє частково автоматизувати генерацію завдань із довільного тексту. Проте якість залежить від навичок формулювання prompt, відсутній інтерфейс перевірки відповідей, історія сесій або підтримка формального тестування. Це рішення радше експериментальне, ніж готове до масового використання в освіті.

Рисунок 2.3 - ChatGPT + Prompt Generator

AIQuizMaker, Quizgecko (рис. 2.4). Це сучасні генератори тестів, що використовують GPT- або аналогічні мовні моделі. Вони аналізують введений текст і автоматично генерують запитання. Сервіси працюють швидко, мають простий інтерфейс. Але у безкоштовній версії кількість запитань обмежена, не

підтримуються всі типи запитань, часто бракує контролю над логікою генерації. Більшість таких платформ орієнтовані на англомовний контент, що ускладнює застосування в україномовному навчальному середовищі. Мають обмежений безкоштовний функціонал, часто прив’язані до англомовного контенту та не мають повного контролю над логікою генерації.

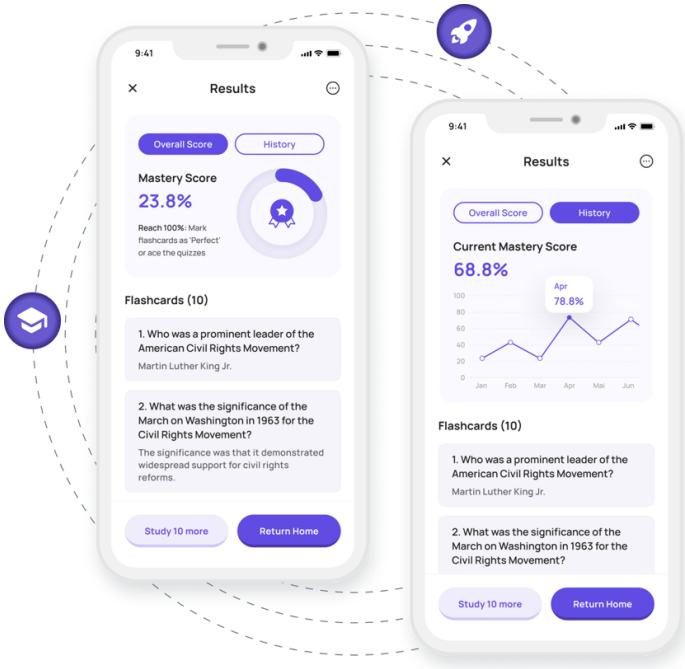


Рисунок 2.4 - AIQuizMaker, Quizgecko

Більшість доступних інструментів мають такі недоліки:

– Відсутність глибокої інтеграції з навчальним контекстом — більшість систем працює на основі шаблонного підходу, не аналізуючи семантику та логіку вихідного навчального матеріалу. В результаті завдання часто втрачають смислову зв’язність із джерельним текстом, що знижує їхню ефективність у навчальному процесі.

– Шаблонність запитань — значна частина сервісів генерує запитання за заздалегідь визначеними структурами, без варіативності формулювань або адаптації до стилістики предмету, що створює ефект механічного тестування та знижує рівень зацікавленості учня.

– Обмежений контроль над типами завдань — користувачі не завжди мають можливість самостійно визначити, який саме тип запитання (multiple

choice, так/ні, відкрите, пропуски тощо) повинен бути згенерований. Це обмежує гнучкість системи в умовах диференційованого навчання.

– Відсутність потокового режиму виведення — більшість рішень працюють у форматі «запит — відповідь», що створює затримки, особливо при обробці великих текстів. Відсутність потокового (streaming) режиму унеможливлює поступове відображення результатів та робить взаємодію менш інтерактивною.

– Не зберігають історію генерацій для подальшого аналізу — системи не надають викладачеві можливість переглядати, повторно використовувати чи модифікувати попередні генерації. Відсутність журналу активності або архіву завдань ускладнює контроль і моніторинг успішності.

– Слабка підтримка PDF-експорту або персоналізації — не всі сервіси дозволяють користувачам експортувати створені тести у формат PDF із брендуванням, стилізацією, підписами чи поділом на рівні складності. Це обмежує можливість подальшого розповсюдження, друку або архівації матеріалів.

На цьому фоні розробка власного вебдодатку, представленого в цій дипломній роботі, вирішує низку критичних обмежень і пропонує новий підхід до формування цифрового освітнього інструментарію. На відміну від більшості існуючих рішень, система реалізована з урахуванням гнучкої архітектури, масштабованої логіки обробки даних та інтерактивної взаємодії з користувачем. У презентації та технічній документації було сформульовано низку ключових вимог, які визначили функціональну модель вебдодатку:

– автоматична генерація завдань на основі завантажених або вставлених вручну навчальних текстів;

– глибока інтеграція NLP-моделей для аналізу структури, семантики та ключових понять у текстах;

– підтримка кількох типів запитань (multiple choice, на заповнення пропусків, так/ні, відкриті);

– реалізація потокового (streaming) виводу генерації з поступовим оновленням результату;

– збереження історії генерацій з можливістю перегляду, редагування, дублювання або експорту;

– підтримка роботи з файлами різних форматів (TXT, DOCX, PDF) та текстовими блоками;

– можливість авторизації користувачів, ведення сесій, збереження персональних результатів;

– підтримка PDF-експорту сформованого тесту із брендуванням, стилізацією та підписами;

– забезпечення безпеки взаємодії через HTTPS, CSRF-захист, cookies, ізоляцію сесій;

– висока продуктивність і стабільна обробка великих обсягів текстових даних.

Уся система проєктувалась із фокусом на доступність, універсальність, масштабованість та адаптивність до різних категорій користувачів. Ключова увага приділялася тому, щоб зробити застосунок інтуїтивно зрозумілим для викладачів, зручним для студентів і гнучким для подальшої інтеграції у вже наявні освітні екосистеми. Розробка фронтенду орієнтувалася на створення чистого, мінімалістичного й адаптивного інтерфейсу, що підтримує як роботу з невеликими текстовими фрагментами (наприклад, для генерації одного-двох завдань), так і з великими за обсягом текстами (розділи підручника, лекції, навчальні посібники). Вебінтерфейс реалізовано з урахуванням вимог до швидкості завантаження, інтерактивності та реактивного зворотного зв’язку. Для викладача — це інструмент швидкого формування перевірочних матеріалів, а також збереження створених сесій для подальшого повторного використання чи редагування. Передбачено підтримку фільтрації завдань за типом, складністю, структурною одиницею тексту (абзац, розділ тощо). Для студентів — це платформа самоперевірки, де завдання можуть виводитись поступово (streaming), з підказками, варіантами відповідей та поясненнями.

Окремо реалізовано механізм багатокористувацької взаємодії: особистий кабінет, авторизація, історія генерацій, можливість завантаження результатів у PDF, перегляд статистики, збереження обраних завдань. Передбачено режим "тільки читання" для учнів і розширений режим редагування для викладачів.

Запропонований вебдодаток виступає не просто як альтернатива існуючим системам, а як повноцінна адаптивна освітня платформа нового покоління. Вона поєднує найкращі практики сучасних AI-інструментів (модельні ядра GPT/LLaMA), потужну серверну логіку, гнучкий інтерфейс і педагогічну ефективність. Його впровадження дозволяє переосмислити класичну систему тестування, трансформувавши її у відкритий, керований, прозорий і результатоорієнтований процес оцінювання знань і формування компетентностей XXI століття.

2.2 Формування вимог та архітектури застосунку

Вимоги до системи були деталізовані на основі глибокого аналізу освітніх потреб та функціональних обмежень існуючих платформ, а також на базі результатів попередньої оцінки сучасних AI-сервісів. Допомогли чітко структурувати запити до системи за категоріями функціональних і нефункціональних характеристик. Вони стали основою для архітектурного проєктування та реалізації системи. На основі вивчених практик автоматизованих сервісів тестування, таких як Quizlet, Testportal, ChatGPT-подібні генератори, було виявлено низку недоліків — шаблонність завдань, відсутність адаптації до навчального контексту, нестача потокового виведення результатів, обмежений функціонал експорту. Усі ці обмеження стали драйверами для розробки власного інноваційного рішення.

З урахуванням сучасних можливостей великих мовних моделей, таких як GPT і LLaMA, та потреб українського освітнього середовища, були сформульовані такі принципи до побудови вимог:

– Максимальна адаптація до освітніх сценаріїв (школа, університет, онлайн-курси).

- Підтримка української мови, локалізація й розширена мовна підтримка.
- Відкритість до модифікацій та налаштувань: як для викладача, так і адміністратора.
- Сумісність із мобільними пристроями, стабільність при роботі з великими обсягами інформації.
- Безпечність, захищеність персональних даних та підтримка сесійної взаємодії.

Сформульовані вимоги стали не лише технічним описом функцій, а й концептуальним баченням нового покоління інструменту автоматизованої генерації тестових завдань. Це рішення має відповідати сучасним педагогічним викликам, бути інтегрованим у цифрове середовище та надавати викладачам і студентам ефективний, зручний і безпечний інструмент для оцінювання знань.

Функціональні вимоги:

Реєстрація та авторизація користувачів.

Здійснюється через форму зберігання облікових даних у захищеній базі. Забезпечує ідентифікацію, персоніфікацію результатів та безпеку доступу.

Завантаження навчальних матеріалів у форматах різних.

Підтримуються формати TXT, DOCX, PDF. Це дозволяє викладачам використовувати вже наявні матеріали без попередньої обробки.

Можливість введення тексту вручну для генерації завдань.

Інтерфейс передбачає окреме текстове поле для ручного введення матеріалу - зручно для коротких сесій або редагування фрагментів.

Аналіз тексту за допомогою NLP для витягнення ключових понять.

Застосовується розпізнавання сутностей, частин мови, семантичного контексту (на базі spaCy або NLTK).

Генерація запитань різних типів.

- Підтримуються завдання з вибором відповіді, відкриті, true/false, на заповнення пропусків. Це забезпечує дидактичну різноманітність.
- Потокове (streaming) виведення згенерованих завдань у режимі реального часу.

- Використання SSE або WebSocket для поступового оновлення інтерфейсу в ході генерації — для кращого UX та сприйняття.
- Експорт тестів у PDF.
- Реалізовано за допомогою ReportLab/pdfkit з адаптацією до дизайну та включенням відповідей/пояснень.

Панель адміністратора для керування генерацією.

Передбачено адміністрування генерацій, пріоритетів, обмежень, а також керування сесіями користувачів.

Нефункціональні вимоги:

Підтримка захищених сесій (https, cookies, CSRF-захист).

Важливо для запобігання атак типу man-in-the-middle та забезпечення цілісності взаємодії.

Надійне зберігання паролів у базі даних (bcrypt або аналог).

Реалізовано з хешуванням для уникнення компрометації облікових записів.

Висока швидкодія при обробці великих обсягів тексту.

- Оптимізація обробки довгих текстів (до 10 000 слів) забезпечується через кешування, потокову генерацію та ефективну роботу NLP.
- Відповідність принципам зручності інтерфейсу (UX) для використання у стресових умовах.
- Мінімалістичний, адаптивний інтерфейс для швидкої генерації, перегляду й експорту — навіть при навантаженні або обмеженому часі.

Архітектура застосунку (рис. 2.5) реалізована за принципом клієнт-серверної моделі та подана у вигляді взаємодії чотирьох основних компонентів:

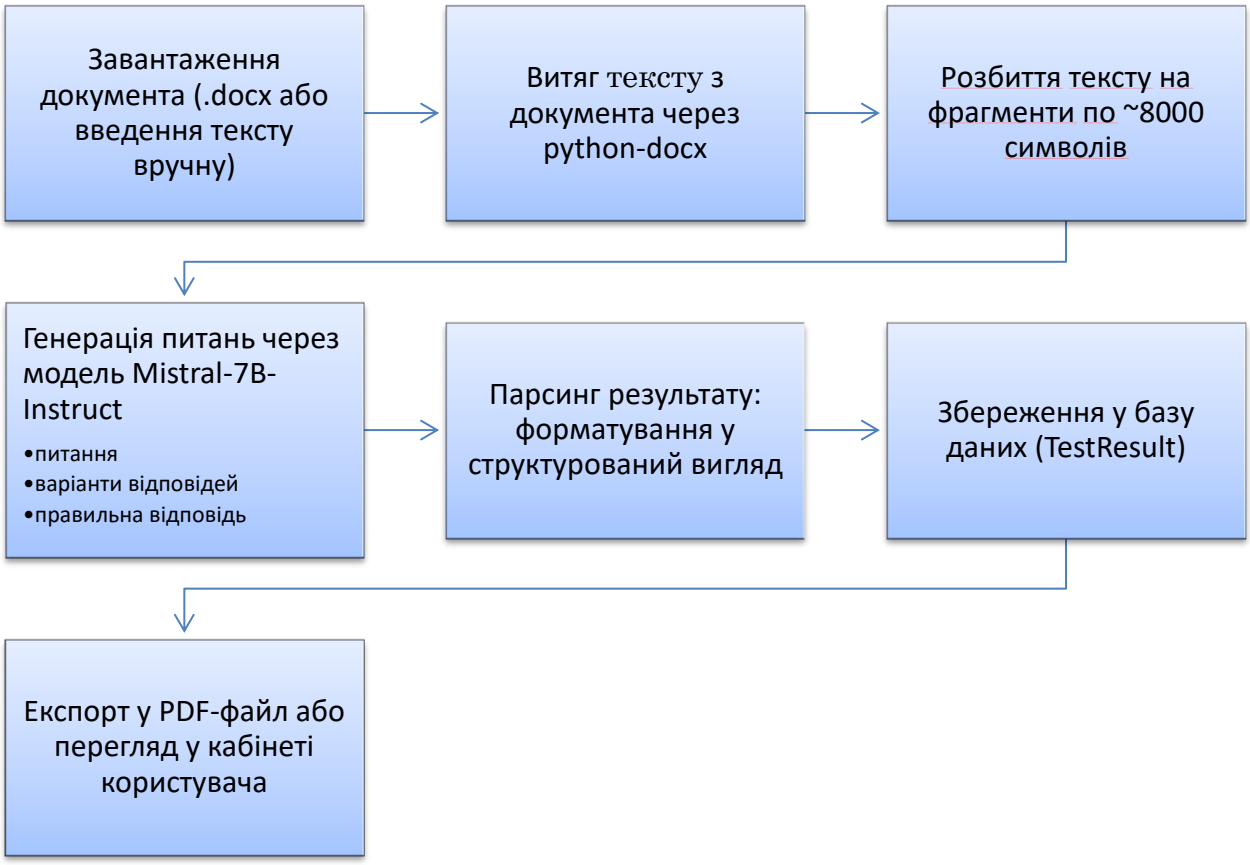


Рисунок 2.5 - Архітектура застосунку

1. Client (Клієнт):

– Побудований на основі HTML, CSS, JavaScript.

– Містить форму для завантаження текстових матеріалів (txt, docx, pdf) або ручного введення.

– Підтримує HTTP-запити до сервера через Ajax та приймає відповідь у вигляді JSON або PDF.

– Працює в потоковому режимі через технологію SSE (Server-Sent Events), яка забезпечує реальну взаємодію з генерацією запитань у режимі реального часу.

2. Server (Flask):

Складається з кількох логічних рівнів:

– API Layer — обробляє запити користувача, автентифікацію, завантаження матеріалів, старт генерації.

- NLP Layer — аналізує текст за допомогою NLP-інструментів (spaCy/NLTK або LLaMA) для виділення ключових понять.
- Генератор — формує запитання за заданими сценаріями: multiple choice, заповнення пропусків, відкриті.
- PDF Layer — генерує фінальний документ для збереження та експорту (з використанням ReportLab).

3. Database Layer:

- Використовуються СУБД SQLite або PostgreSQL.
- База даних зберігає облікові записи користувачів, історію генерацій, результати запитань і метадані по кожній сесії.

4. File Storage Layer:

Зберігання PDF-файлів реалізовано або локально, або на Amazon S3, що забезпечує масштабованість та безпечний доступ до згенерованих тестів.

Основна логіка взаємодії:

- Користувач завантажує або вводить навчальний текст.
- Сервер обробляє його за допомогою NLP-модуля.
- Результати поступово виводяться у клієнтському інтерфейсі через потік SSE.
- Користувач за потреби експортує результати у PDF.
- Вся інформація автоматично зберігається в базі даних для повторного використання.

Така архітектура дозволяє досягти високої продуктивності, адаптивності та гнучкості — як для масштабованого розгортання на сервері, так і для зручного використання у стресових умовах навчального процесу.

Для взаємодії з користувачем вебзастосунок використовує механізм потокового оновлення інтерфейсу, який реалізований через комбінацію технологій JavaScript, Ajax та SSE (Server-Sent Events). Це забезпечує відображення процесу генерації тестових завдань у режимі реального часу: користувач одразу бачить появу сформованих запитань на сторінці, без

необхідності очікувати завершення повної генерації. Такий підхід значно покращує користувацький досвід, підвищує динамічність взаємодії та дозволяє оцінити зміст кожного завдання ще до завершення сесії. Згенеровані сесії не просто відображаються в інтерфейсі, а й автоматично зберігаються у базі даних. Це дає змогу користувачеві повертатися до попередньо створених тестів, переглядати їх, редагувати, доповнювати або повторно експортувати у форматі PDF. Завдяки такій логіці кожна сесія генерації виступає як повноцінна одиниця збереженого навчального контенту, який можна керовано використовувати в різних освітніх контекстах. У межах інтерфейсу реалізовано можливість персоналізації генерації. Користувач самостійно обирає тип тестових завдань (вибір правильної відповіді, пропуски, відкриті запитання), рівень складності, бажану кількість запитань. Також можна здійснювати фільтрацію сформованих блоків, редагувати окремі запитання вручну, додавати авторські варіанти відповідей або адаптувати формулювання відповідно до змісту заняття.

Взаємодія між користувачем і застосунком не обмежується лише завантаженням тексту й отриманням результату — вона є гнучким, інтерактивним процесом із високим рівнем кастомізації. У поєднанні з модульною архітектурою, що легко адаптується до змін, це робить запропонований вебдодаток технологічно сучасним і методично ефективним рішенням для автоматизованого створення навчальних тестів.

2.3 Моделювання бази даних та логіки генерації запитань

Ефективна робота вебзастосунку неможлива без чітко спроєктованої та оптимізованої структури (рис. 2.6) збереження даних, яка здатна підтримувати взаємодію численних користувачів, забезпечувати надійне зберігання результатів генерації та дозволяти швидке отримання інформації для подальшого використання. Проєктування бази даних є одним із ключових етапів розробки системи, адже саме база забезпечує основу для стабільної роботи всіх основних модулів: авторизації, збереження навчальних текстів, журналювання подій, обробки запитань та відповідей. Моделювання структури

БД здійснювалося з урахуванням низки принципів: нормалізації, уникаючи дублювання даних; гнучкості, що дозволяє легко розширювати модель новими типами запитань чи атрибутів; цілісності, шляхом реалізації зовнішніх ключів між таблицями; а також ефективності — шляхом створення індексів для швидкого пошуку. Особливу увагу було приділено масштабованості — система повинна залишатися стабільною при зростанні кількості користувачів і сесій генерації. Враховувалася можливість інтеграції з іншими модулями: збереження PDF-файлів, статистики успішності, відгуків тощо.

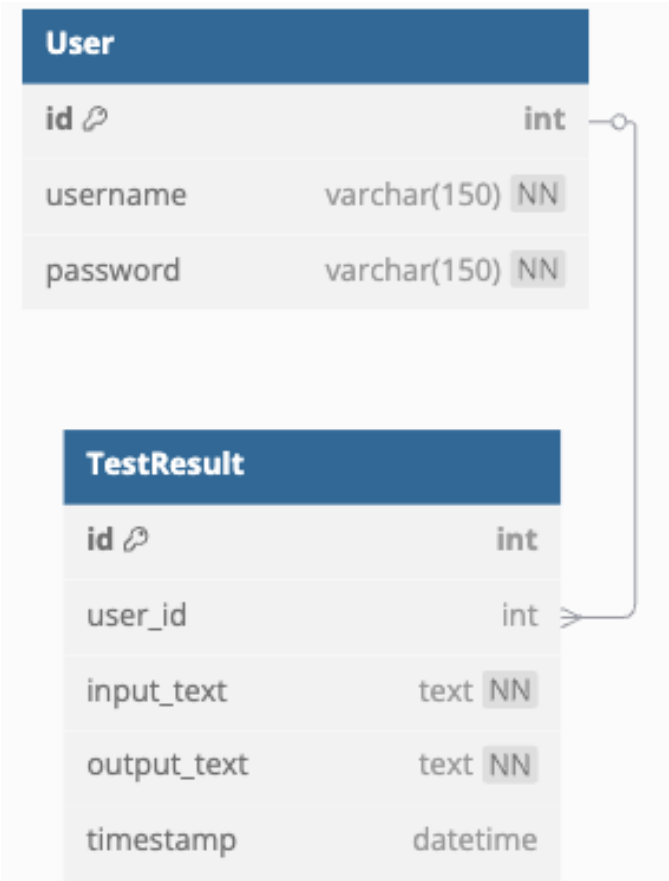


Рисунок 2.6 – Структура бази даних

База даних не лише виконує роль сховища, а й є засобом управління контентом, на основі якого працює логіка генерації: від вхідного тексту до остаточного варіанту тесту з усіма запитаннями, варіантами відповідей, поясненнями та метаданими. Структура таблиць має бути достатньо детальною, щоб зберігати кожен запит, контекст генерації, тип завдань, час створення, а також пов’язати все це з конкретним користувачем. Такий підхід дозволяє не

лише вести аналітику та історію використання, а й реалізувати функції адаптивного навчання, коли система пропонує запитання з урахуванням попередніх результатів студента або складності пройдених тем.

На етапі реалізації було використано ORM-бібліотеку SQLAlchemy для формування структури бази даних у вигляді Python-класів, що суттєво полегшує взаємодію з даними та пришвидшує розробку. Цей підхід дозволяє не лише створювати таблиці безпосередньо з коду, але й ефективно реалізовувати зв'язки між ними, забезпечуючи логічну цілісність і підтримку транзакцій. Класи User та TestResult є базовими складовими цієї моделі, які відображають найважливіші аспекти системи — збереження облікових записів користувачів і результатів генерацій. Такий підхід забезпечує гнучкість при оновленні схеми, дозволяє легко масштабувати систему при розширенні функціональності (наприклад, додавання нових типів користувачів, ролей, варіантів оцінювання тощо), і створює основу для аналітики, безпеки та персоналізованої роботи кожного користувача. Модель повністю відповідає структурі, представленій у презентації (слайд «Структура бази даних»), і дозволяє легко інтегрувати нові логічні модулі — такі як розділ статистики, банк тестових запитань, журнал активності, або систему рекомендацій для покращення якості тестування.

Фрагмент коду моделі SQLAlchemy:

```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), unique=True, nullable=False)
    password = db.Column(db.String(150), nullable=False)
class TestResult(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    input_text = db.Column(db.Text, nullable=False)
    output_text = db.Column(db.Text, nullable=False)
    timestamp = db.Column(db.DateTime, server_default=db.func.now())
```

Опис структури бази даних:

User (Користувачі) - зберігає інформацію про зареєстрованих користувачів:

- id — унікальний ідентифікатор користувача
- username — логін (унікальний, не null)
- password — зашифрований пароль користувача (зберігається у хешованому вигляді)
- TestResult (Результати генерації): зберігає результати згенерованих тестів:
- id — унікальний ідентифікатор результату
- user_id — зовнішній ключ, який вказує на автора генерації
- input_text — текст, який завантажив або ввів користувач
- output_text — згенеровані запитання (у форматі JSON або plain text)
- timestamp — дата та час створення (встановлюється автоматично)

Ці дві таблиці забезпечують критичну основу для функціонування всієї системи, виступаючи центральним елементом зберігання й організації даних у процесі роботи вебзастосунку. Вони забезпечують персоналізоване збереження історії генерацій, що дозволяє кожному користувачеві мати доступ до власних результатів у будь-який момент. Це важливо як з позиції гнучкої роботи в реальному часі, так і для відкладеного використання — наприклад, підготовки підсумкових звітів або повторного експорту тестів у PDF. Реалізоване збереження даних створює можливість для ведення аналітики — система може надавати статистику про кількість створених тестів, типи запитань, середній рівень складності, частоту використання тих чи інших конструкцій. Це дозволяє як адміністраторам, так і викладачам отримувати об’єктивні дані про освітній процес, коригувати підходи до навчання та виявляти прогалини в знаннях учнів. Наявність чітко спроєктованих зв’язків між таблицями (user_id як зовнішній ключ) дозволяє реалізувати фільтрацію та сегментацію:

наприклад, можна отримати всі генерації конкретного користувача за певний період часу, вивести тільки завдання певного типу або обробити лише ті, що були експортовані. Окрему цінність становить можливість повторного використання вже згенерованих матеріалів — викладач може створити банк запитань, формувати тести з наявних блоків, адаптувати їх для різних груп студентів, а також зберігати версії для контролю змін. Усі ці можливості роблять реалізовану модель бази даних не просто технічним інструментом, а ключовим механізмом підтримки інтелектуальної логіки системи автоматичного формування навчального контенту.

Логіка генерації запитань:

Попередній NLP-аналіз:

Першим етапом логіки генерації запитань є глибинний лінгвістичний аналіз вхідного тексту. Текст розбивається на логічні сегменти — абзаци та речення — із застосуванням токенізації. Далі кожен фрагмент аналізується для виявлення ключових слів, граматичних категорій, частин мови (іменники, дієслова, прикметники тощо), синтаксичних залежностей та іменованих сутностей (NER — наприклад, дати, прізвища, географічні об’єкти). Такий аналіз дозволяє побудувати внутрішню модель тексту, що відповідає його семантичному навантаженню та контексту. Застосовуються бібліотеки spaCy або nltk, що дозволяють виконати морфологічний розбір, виявити структуру речення, а також тематичні зв’язки між словами. Особливу увагу приділяється частотному аналізу лексики, визначенню тематичних ядер, повторюваності термінів, структурі питальних/ствердних речень. Це дає змогу адаптувати логіку генерації до особливостей предмету (наприклад, гуманітарний чи технічний контент), а також фільтрувати інформаційно незначущі фрагменти.

Визначення типів завдань:

Після завершення первинного аналізу система переходить до етапу категоризації запитань. На основі розміченого синтаксичного дерева та семантичних зв’язків модель обирає найбільш відповідні типи завдань. Наприклад, якщо в тексті домінують визначення або пояснення термінів —

пріоритет отримують запитання з вибором правильної відповіді (multiple choice); при наявності причинно-наслідкових зв’язків — true/false або відкриті запитання. У випадку структурованого тексту із визначеннями, списками або хронологічними подіями — доречними є завдання на заповнення пропусків або встановлення відповідності. Алгоритм також враховує довжину речення, граматичну побудову, рівень вкладеності підрядних конструкцій, що впливає на складність запитання. Визначення типу завдань — це не статичне правило, а адаптивна процедура, яка підлаштовується під вхідні дані.

Формування шаблонів запитань:

Наступним кроком є формування тексту запитання на основі адаптованих шаблонів. Модель використовує набір генеративних патернів, що відповідають типам завдань: наприклад, «Що сталося в [дата]?»; «Яке з наведених тверджень є правильним щодо...?»; «Обери правильний варіант, щоб завершити речення...». У випадку дат, місць, імен — формується хронологія або географічна прив’язка. Для дефініцій — генерація за логікою «Що таке...?». Шаблони постійно уточнюються під конкретний навчальний контекст. У процесі беруть участь як машинні правила, так і логічні трансформації речень — заміна активних конструкцій на пасивні, формулювання питального формату, інверсія речень, спрощення граматичних структур для молодшого віку. Це дозволяє отримати природні та зрозумілі запитання, які не виглядають штучно або випадково сформованими.

Генерація варіантів відповідей:

Етап формування варіантів відповідей є одним із ключових у забезпеченні педагогічної ефективності тестових завдань. Від якості дистракторів (неправильних варіантів) значною мірою залежить складність, коректність і здатність завдання дійсно оцінювати знання, а не випадкове вгадування.

У випадку завдань типу multiple choice модель спочатку фіксує правильну відповідь, отриману з тексту або бази знань, після чого за допомогою генеративних трансформацій формує 2–4 дистрактори. Ці дистрактори можуть

створюватися кількома способами: перефразуванням (LLM-моделлю), логічним узагальненням, семантично близькими поняттями (за допомогою word embeddings), або використанням помилок, які є типовими для студентів. Наприклад, у запитанні «Коли відбулась битва при Грюнвальді?» — правильна відповідь: «1410 рік», а дистрактори можуть бути: «1380 рік», «1430 рік», «1415 рік». У завданнях на заповнення пропусків (fill-in-the-blank) система аналізує граматичну структуру речення, і виділяє ключове слово (іменник, дієслово, числівник або термін), яке вилучається. При цьому речення залишається синтаксично цілісним, а варіанти відповідей формуються з синонімів, антонімів, або фрагментів того ж тексту з подібним контекстом. У відкритих запитаннях (open-ended) система формує лише саме запитання, залишаючи простір для власної відповіді користувача. У цьому випадку особливу увагу приділено тому, щоб запитання було чітким, недвозначним, орієнтованим на одну правильну думку. В майбутньому такі відповіді можуть автоматично оцінюватися за допомогою LLM-класифікатора. Для запитань типу true/false система бере фактичне твердження з тексту й генерує його інверсію або змінює ключовий факт (дату, суб'єкта, числову величину), щоб сформувати хибний варіант. Наприклад: «Франція приєдналася до ЄС у 1995 році» — інверсія факту, яка буде хибною. Такі запитання генеруються з урахуванням того, щоб хибність не була очевидною, а вимагала від учня аналізу.

Фінальна оцінка та перевірка:

На завершальному етапі перед виведенням запитання модель виконує багаторівневу перевірку, яка охоплює як мовну, так і логічну цілісність завдань. Насамперед система аналізує сформовані запитання на наявність дублікатів — як повних, так і часткових, що можуть виникати внаслідок обробки схожих фрагментів тексту. Виявлення дублювань відбувається шляхом порівняння лексичних конструкцій, змістового ядра та семантичної близькості. Якщо виявляються подібні запитання, вони видаляються або замінюються альтернативними формулюваннями. Далі перевіряється стилістична єдність — наскільки запитання відповідають заданому тону й граматичному стилю тексту.

Це включає виявлення різнорівневих формулювань (наприклад, коли одне запитання звучить як академічне, а інше — як розмовне), перевірку узгодженості часу, відсутність орфографічних помилок і логічних зсувів. Використовуються вбудовані інструменти стилістичної нормалізації або мовні моделі з навчанням на корпусі навчальних матеріалів. Особливу увагу приділяється перевірці логічної цілісності: чи дійсно запитання має сенс у контексті поданого тексту, чи немає двозначності у формулюванні, чи відповідає правильна відповідь структурі завдання. Наприклад, у випадку multiple choice — перевіряється, що лише одна відповідь є правильною, а distractors не дублюють її частково або повністю. У true/false — факт повинен мати чітку ознаку істинності або хибності, без контроверсійних трактувань. Якщо за результатами автоматичної перевірки система фіксує потенційні порушення, такі запитання позначаються як «неперевірені» або пропонуються до перегляду вручну. Таким чином, користувач отримує змогу самостійно редагувати запитання або змінювати їх тип, покращуючи загальну якість підсумкового тесту. Це також дозволяє адаптувати завдання під конкретні навчальні цілі, рівень учнів або структуру заняття.

У майбутньому планується інтеграція методу автоматичної перевірки якості за допомогою LLM або моделей ранжування, які на основі великих корпусів навчальних матеріалів оцінюватимуть, наскільки запитання є педагогічно релевантними та пізнавально значущими. Таким чином, фінальна оцінка стає не лише технічним кроком, а повноцінним етапом підвищення якості навчального контенту. Завдяки такому підходу забезпечується багаторівнева, структурована, гнучка та масштабована логіка роботи, яка формує технічний фундамент і педагогічну цінність системи. Вона дозволяє не лише базову автоматизовану генерацію тестових завдань, а й формує простір для повноцінного керування навчальним контентом. Кожна дія — від обробки тексту до виведення завдання — проходить через низку аналітичних та трансформаційних стадій, що підвищує якість, логічну послідовність та адаптованість матеріалу до потреб конкретного викладача або студента.

Структурованість реалізується через чітке розбиття етапів аналізу: спочатку відбувається попередня лінгвістична обробка (токенізація, лематизація, розпізнавання іменованих сутностей), потім — визначення релевантних фрагментів тексту, на основі яких формуються питання. Це дозволяє уникати випадкового або контекстно незначущого формування завдань. Усі параметри фіксуються у базі даних для забезпечення прозорості, повторного доступу та накопичення навчального корпусу. Гнучкість системи полягає не лише у виборі типу запитання (multiple choice, open-ended, true/false, fill in the blanks), а й у можливості змінювати рівень складності, стиль формулювань, кількість варіантів, навіть на льоту. Користувач може налаштовувати генерацію відповідно до вікових особливостей, цілей оцінювання, конкретних методичних рекомендацій. Додатково передбачено ручне редагування, що дозволяє перетворити автоматично згенероване завдання в індивідуальний елемент освітньої програми. Масштабованість проявляється як у технічному, так і в методологічному аспектах. Система здатна обслуговувати велику кількість паралельних користувачів, підтримувати історію генерацій, формувати статистику за дисциплінами, класами, рівнями навчання. У перспективі можлива побудова аналітичного модуля, що рекомендує викладачеві ті типи завдань або рівні складності, які найкраще підходять конкретному учневі або групі на основі попередньої взаємодії.

Сформована логіка не обмежується однокроковою генерацією, а перетворює систему на інтелектуальну освітню платформу, яка підтримує гнучкий цикл взаємодії: від введення знань — до перевірки, оцінювання, повторного навчання та вдосконалення педагогічної стратегії. Це відповідає сучасним вимогам цифрової трансформації освіти й відкриває нові горизонти автоматизованого навчання.

Висновок до розділу 2

У другому розділі було здійснено комплексний аналіз архітектури, логіки функціонування та реалізації програмного забезпечення для автоматичної генерації тестових завдань на основі навчальних матеріалів. Розгляд почався з

аналізу сучасних сервісів генерації запитань, таких як Quizlet, Testportal, AIQuizMaker, ChatGPT та ін., що дало змогу виявити їх обмеження, зокрема відсутність потокової генерації, слабку персоналізацію, обмеження за мовою, типами завдань та збереженням історії. На цьому тлі була сформульована необхідність створення власного вебдодатку. Було детально сформульовано функціональні та нефункціональні вимоги до системи: підтримка різних форматів вхідного тексту, багатьох типів завдань, генерація в режимі реального часу, PDF-експорт, авторизація користувачів, інтеграція з мовними моделями та забезпечення безпеки даних. Архітектура системи описана як багаторівнева клієнт-серверна модель з чіткою побудовою: фронтенд, бекенд на Flask, AI-модуль, база даних та файлове сховище. Особливу увагу приділено моделюванню бази даних, яка забезпечує ефективне збереження користувацьких сесій, результатів генерації, варіантів відповідей та подій. Також було докладно розглянуто алгоритм логіки генерації: від попереднього NLP-аналізу до фінального редагування запитань. Було продемонстровано, як система враховує синтаксичну структуру тексту, визначає тип запитання, формує шаблон, генерує distractors, виконує стилістичну і логічну перевірку.

Другий розділ заклав технічну та алгоритмічну основу для реалізації повноцінного інтелектуального вебдодатку нового покоління, здатного не лише автоматизувати створення завдань, а й адаптувати їх до педагогічного контексту, стилю навчання та рівня підготовки користувача.

3 РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА ОЦІНКА ЯКОСТІ

3.1 Розробка клієнтської та серверної частини застосунку

Розробка застосунку відбувалась за класичною клієнт-серверною архітектурою, яка дозволяє ефективно розділити логіку взаємодії користувача та обробки даних на дві окремі частини: клієнтську (frontend) та серверну (backend). Такий підхід забезпечує масштабованість, легкість у супроводі та розвиток системи. Для розробки інтерфейсу користувача було використано сучасні вебтехнології — HTML5 для розмітки, Tailwind CSS для стилізації та JavaScript (разом з бібліотекою jQuery) для реалізації логіки на боці клієнта. Це дозволило створити інтуїтивно зрозумілий, адаптивний і швидкий інтерфейс, який працює однаково добре на десктопах та мобільних пристроях. Інтерфейс включає форму для введення тексту, область виведення згенерованих результатів, елементи управління процесом генерації та збереженням. Серверна частина реалізована з використанням мікрофреймворку Flask на мові програмування Python. Flask дозволяє будувати REST API, яке обробляє запити від клієнта, запускає логіку генерації запитань, взаємодіє з базою даних і формує відповіді. Завдяки модульному підходу реалізовано окремі компоненти для аутентифікації, логіки генерації, експорту в PDF, потокової передачі даних (SSE), обробки помилок та логування. Це дозволяє легко підтримувати, тестувати та оновлювати кожен компонент без шкоди для цілісності всієї системи.

Окрему увагу було приділено підтримці потокової генерації запитань (streaming), що значно покращує взаємодію з користувачем. Завдяки застосуванню SSE (Server-Sent Events) результати передаються з сервера до браузера поступово, у міру їх генерації. Це дозволяє користувачу бачити процес у реальному часі та взаємодіяти з результатами негайно, не чекаючи повної обробки великого тексту. Вибрана архітектура та технологічний стек забезпечують баланс між швидкістю розробки, зручністю використання,

можливістю масштабування і готовністю до подальшої інтеграції з мовними моделями, такими як GPT або LLaMA.

Клієнтська частина (HTML + JS)

Форма для введення тексту користувачем реалізована у вигляді HTML-компонента з використанням стилів Tailwind. Вона включає текстову область для введення, кнопку для запуску генерації та блок для динамічного виводу результатів. Ця форма є центральним елементом клієнтського інтерфейсу й ініціює основну взаємодію із сервером. Її структура дозволяє не тільки зручно ввести великий обсяг навчального матеріалу, але й оперативно отримати згенеровані запитання в режимі реального часу через механізм потокової передачі даних (SSE).

Нижче подано фрагмент HTML-коду, який використовується в реалізації форми:

```
<form id="generate-form">
  <textarea id="input-text" class="w-full p-2 border" rows="10"></textarea>
  <button type="submit" class="mt-2 px-4 py-2 bg-blue-600 text-white">Генерувати</button>
</form>
<div id="output" class="mt-4"></div>
```

JavaScript для обробки події й потокового SSE-з'єднання реалізує логіку зв’язку клієнтської частини з сервером у режимі реального часу. Цей фрагмент коду дозволяє при натисканні кнопки надсилати введений текст на сервер і підписуватись на потік подій, які надсилаються через Server-Sent Events (SSE). Кожен фрагмент тексту, який генерується на сервері, одразу відображається у блоці результатів на сторінці. Користувач не чекає завершення повної обробки всього тексту — перші запитання починають з’являтись миттєво. Це особливо зручно при роботі з великими обсягами тексту й дозволяє підвищити інтерактивність інтерфейсу.

Повний фрагмент JavaScript-коду:

```
$("#generate-form").on("submit", function(e) {
  e.preventDefault(); // блокує перезавантаження сторінки
  const input = $("#input-text").val();
  const source = new EventSource(`/stream?text=${encodeURIComponent(input)}`);
  // очищення попереднього виводу
  $("#output").empty();
  // обробка кожного повідомлення з потоку
  source.onmessage = function(event) {
    $("#output").append(`<p>${event.data}</p>`);
  };
  // обробка завершення з'єднання (необов'язково)
  source.onerror = function() {
    source.close();
  };
});
```

Цей код забезпечує асинхронну взаємодію між інтерфейсом та генератором, дозволяючи уникнути затримок і покращити сприйняття системи як «живої». Він може бути адаптований для покрокового відображення прогресу, вставки завантажувача, додавання звуків або підсвітки нових запитань.

```
$("#generate-form").on("submit", function(e) {
  e.preventDefault();
  const input = $("#input-text").val();
  const source = new EventSource(`/stream?text=${encodeURIComponent(input)}`);
  $("#output").empty();
  source.onmessage = function(event) {
    $("#output").append(`<p>${event.data}</p>`);
  };
});
```

Серверна частина (Flask, Python)

stream.py – потокова відповідь:

```
from flask import Flask, Response, request
import time
from generator import generate_questions
app = Flask(__name__)
@app.route('/stream')
def stream():
    text = request.args.get("text", "")
    def generate():
        if not text.strip():
            yield "data: [ПОМИЛКА] Текст не може бути порожнім"
            return
        questions = generate_questions(text)
        for idx, q in enumerate(questions, 1):
            yield f"data: {idx}. {q}"
            time.sleep(0.4)    # штучна затримка для ефекту потоковості
        yield "data: [ГЕНЕРАЦІЮ ЗАВЕРШЕНО]"
    return Response(generate(), mimetype='text/event-stream')
```

У цьому прикладі серверна функція stream() ініціює генерацію питань із тексту, отриманого від клієнта. Після кожного згенерованого питання воно відправляється в браузер як SSE-подія. Якщо введений текст порожній, користувачу одразу надсилається повідомлення про помилку. Наприкінці потоку надсилається спеціальний маркер про завершення генерації. Завдяки time.sleep() модель імітує поступову генерацію, що візуально підсилює ефект живого потоку для користувача.

generator.py – основна логіка генерації:

```
def generate_questions(text):
    # Проста реалізація для прикладу
    sentences = text.split('.')
    for s in sentences:
        if s.strip():
            yield f"Що означає: {s.strip()}?"
```

Авторизація (auth.py)

```
@app.route("/login", methods=["POST"])
def login():
    data = request.json
    user = User.query.filter_by(username=data['username']).first()
    if user and check_password_hash(user.password, data['password']):
        session['user_id'] = user.id
        return jsonify({"status": "success"})
    return jsonify({"status": "error"}), 401
```

Збереження в базу даних (models.py)

```
class TestResult(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    input_text = db.Column(db.Text)
    output_text = db.Column(db.Text)
    timestamp = db.Column(db.DateTime, default=datetime.utcnow)
```

Експорт у PDF (export.py)

Модуль експорту в PDF є важливою частиною вебзастосунку, оскільки він забезпечує можливість збереження результатів генерації в зручному форматі для подальшого друку, розповсюдження або архівування. Генерація PDF-файлів реалізована за допомогою бібліотеки ReportLab, яка дозволяє створювати документи з кастомізованим виглядом, шрифтами та форматуванням.

PDF-файл містить основну інформацію — заголовок, фрагмент навчального тексту, список згенерованих запитань та дату створення. У майбутньому можна розширити функціонал додаванням логотипу закладу, номерованих сторінок, динамічного підпису викладача, QR-коду, графіків, зображень або інтерактивних форм для зворотного зв’язку.

Нижче наведено приклад реалізації модуля експорту:

```
from flask import send_file
from io import BytesIO
from reportlab.pdfgen import canvas
@app.route("/export/<int:result_id>")
def export_pdf(result_id):
    result = TestResult.query.get_or_404(result_id)
    buffer = BytesIO()
    pdf = canvas.Canvas(buffer)
    pdf.setFont("Helvetica-Bold", 14)
    pdf.drawString(50, 800, "Результати генерації тестових завдань")
    pdf.setFont("Helvetica", 10)
    y = 760
    pdf.drawString(50, y, "Дата: " + result.timestamp.strftime("%d.%m.%Y %H:%M"))
    y -= 20
    pdf.drawString(50, y, "Текст для генерації (скорочено):")
    y -= 20
    pdf.setFont("Helvetica-Oblique", 9)
    for line in result.input_text[:300].split(". "):
```

```
                pdf.drawString(60, y, line.strip())
                y -= 15
                if y < 100:
                    pdf.showPage()
                    y = 800
            y -= 30
            pdf.setFont("Helvetica-Bold", 11)
            pdf.drawString(50, y, "Згенеровані запитання:")
            y -= 20
            pdf.setFont("Helvetica", 10)
            for i, line in enumerate(result.output_text.split("
        "), 1):
                pdf.drawString(60, y, f"{i}. {line.strip()}")
                y -= 15
                if y < 100:
                    pdf.showPage()
                    y = 800
            pdf.save()
            buffer.seek(0)
            return        send_file(buffer,        as_attachment=True,
download_name=f'result_{result.id}.pdf')
```

Код дозволяє експортувати результат генерації у вигляді структурованого PDF-документа з налаштовуваним виглядом. За потреби його легко адаптувати до корпоративного стилю або нормативних вимог документації.

3.2 Інтеграція NLP-моделі та генерація тестів у режимі реального часу

У цьому підрозділі розглядається технічна інтеграція модуля обробки природної мови (NLP) із системою генерації завдань, а також реалізація механізму передачі результатів у реальному часі з використанням потокових технологій (Streaming). Основна мета — забезпечити не лише автоматизовану, а й адаптивну, логічно структуровану і педагогічно обґрунтовану генерацію тестових запитань. Система повинна мати здатність адаптуватися до змісту,

складності та структури будь-якого навчального тексту, незалежно від предметної області чи стилістики викладу. Це особливо важливо в умовах сучасної цифрової освіти, де навчальні матеріали можуть бути представлені у різних форматах, містити терміни, дати, імена або наукові поняття. Інтеграція NLP-модуля дозволяє проводити комплексний аналіз вхідного тексту: від базової токенізації до глибинного семантичного розбору. Таким чином, система здатна визначати, які частини тексту є ключовими, яка роль окремих слів у реченнях, які факти можна трансформувати у запитання. Це дає змогу створювати завдання не випадковим чином, а на основі змістовної релевантності. Так, речення, що містить події, імена, терміни чи причинно-наслідкові зв’язки, буде використано як основа для формування питань із вибором відповіді, відкритих завдань або завдань на встановлення відповідності. Завдяки потоковій технології SSE (Server-Sent Events) забезпечується миттєва передача згенерованих завдань до клієнтського інтерфейсу. Це означає, що користувач бачить нові запитання у режимі реального часу, як тільки вони формуються на сервері. Такий підхід значно покращує UX — користувач не очікує завершення повного аналізу тексту, а може одразу бачити результат, взаємодіяти з ним, переглядати, коригувати або зберігати.

Інтеграція NLP із потоковою генерацією також створює фундамент для майбутнього розвитку системи — зокрема, для побудови адаптивного навчання. На базі аналізу складності речень, частоти понять, стилістичних характеристик або відповідей студентів можна налаштувати рівень генерації: простіші запитання — для молодших класів; складніші, багаторівневі — для вищої школи. Крім того, така структура відкриває можливості для аналітики: з яких типів текстів найчастіше генеруються корисні запитання, які структури дають найкращі відповіді, які частини тексту не використовуються і потребують додаткової уваги.

Такий підхід поєднує в собі точність машинного аналізу та гнучкість педагогічних моделей, що робить систему не лише інструментом для генерації

тестів, а й повноцінною платформою для супроводу освітнього процесу, орієнтованого на персоналізацію, швидкий зворотний зв'язок і змістову якість тестових матеріалів.

Використання NLP для аналізу навчального тексту

Для попередньої обробки тексту в системі застосовується бібліотека spaCy, яка дозволяє:

- розбивати текст на речення (sentence segmentation);
- виконувати токенізацію та лематизацію слів;
- визначати частини мови (POS-tagging);
- виявляти іменовані сутності (NER);
- аналізувати залежності між словами (dependency parsing).

```
from llama_cpp import Llama
llm = Llama(
    model_path="models/mistral-7b-instruct-v0.2-q4_k_m.gguf",
    n_ctx=4096,
    n_threads=4,
    verbose=False
)
def analyze_text(text):
    prompt = f"Проаналізуй текст та сформуй 5 коротких запитань на основі його змісту. Текст: {text}"
    output = llm(prompt, max_tokens=512)
    return output["choices"][0]["text"].strip().split('
')
```

Аналізовані речення та сутності використовуються як основа для генерації запитань. Наприклад, при виявленні особових імен або дат, формуються запитання типу "Хто...?", "Коли...?".

Генерація запитань на основі шаблонів

Кожне речення, відібране для генерації, піддається шаблонізації. Залежно від частини мови та структури, застосовуються такі типові шаблони:

– "Що означає термін: ...?"
– "Яка подія відбулась у ...?"
– "Хто виконав дію ...?"

```
def generate_question(sentence, entity):
    if entity[1] == "PERSON":
        return f"Хто згадується у фразі: '{sentence}'?"
    elif entity[1] == "DATE":
        return f"Коли відбулась подія у фразі: '{sentence}'?"
    else:
        return f"Що означає: '{sentence}'?"
```

Потокова передача результатів користувачу (SSE)

Після генерації кожного окремого запитання, система негайно передає його у браузер користувача за допомогою технології Server-Sent Events (SSE). Це дозволяє сформувати відчуття безперервного, динамічного процесу генерації, де кожен фрагмент результату надсилається окремим повідомленням — без потреби оновлення сторінки або очікування повної генерації всього тесту. Завдяки SSE користувач бачить нові запитання практично миттєво після їх формування на сервері. Такий підхід знижує психологічне навантаження, дозволяє краще сприймати матеріал частинами та створює інтерактивне освітнє середовище. Крім того, ця технологія дозволяє реалізувати додаткові механізми: показ прогресу, повідомлення про помилки, індикатори завершення, а також налаштування темпу виводу, що робить процес генерації не лише швидким, а й педагогічно ефективним.

```
@app.route('/stream')
def stream():
    text = request.args.get("text", "")
    sentences, entities = analyze_text(text)
    def generate():
        for sentence in sentences:
            for ent in entities:
```

```
                if ent[0] in sentence:
                    q = generate_question(sentence, ent)
                    yield f"data: {q}\n\n"
                    time.sleep(0.3)
        yield "data: [ГЕНЕРАЦІЮ ЗАВЕРШЕНО]\n\n"
    return Response(generate(), mimetype='text/event-stream')
```

Переваги такого підходу

- Реальний час: користувач не чекає завершення всього процесу.
- Якість: запитання генеруються з урахуванням частин мови, сутностей та контексту.
- Масштабованість: підхід працює з будь-яким текстом обсягом до 10 тис. слів.
- Модульність: можна замінити spaCy на іншу NLP-бібліотеку або LLM.

Інтеграція NLP-аналізу з потоковою генерацією забезпечує глибоку адаптацію системи до семантики, структури та освітнього змісту навчального тексту. Завдяки цьому підходу система не просто механічно формує запитання, а фактично аналізує контекст, виявляє ключові сутності, логічні зв’язки та тематичні вузли, що дозволяє створювати завдання з високим рівнем змістової релевантності. Крім того, потокова генерація забезпечує негайний зворотний зв’язок — кожне згенероване запитання з’являється миттєво, формуючи ефект «живої» взаємодії з системою. Це значно підвищує залученість користувача, покращує UX та дозволяє оперативно коригувати результат, що особливо важливо в динамічному навчальному процесі або під час пілотного тестування генератора..

3.3 Тестування роботи застосунку та оцінка якості згенерованих завдань

У цьому розділі здійснюється всебічна перевірка працездатності всіх ключових компонентів реалізованого вебзастосунку, включаючи як клієнтську, так і серверну частину. Основна увага приділяється аналізу функціональності інтерфейсу користувача, взаємодії з моделлю штучного інтелекту, обробки

текстів, збереження результатів і якості сформованих тестових завдань. Особливу увагу також приділено зручності користування системою, адаптивності під різні пристрої та сценарії використання. Тестування проводиться в умовах, наближених до реального використання: із входом користувача, створенням нового тесту, генерацією в режимі реального часу, отриманням результату та його перевіркою в PDF-форматі. Усі етапи перевіряються на стабільність, відповідність функціональним вимогам та якість генерації контенту. Таким чином, аналіз дозволяє не лише виявити можливі недоліки, але й оцінити загальну ефективність і надійність розробленого застосунку у практичному середовищі.

Сторінка входу до системи (рис. 3.1). На етапі авторизації перевіряється коректність введених логіна та пароля. У разі успішного входу користувач автоматично переходить до особистого кабінету. Реалізовано перевірку валідності даних і простий інтерфейс взаємодії.

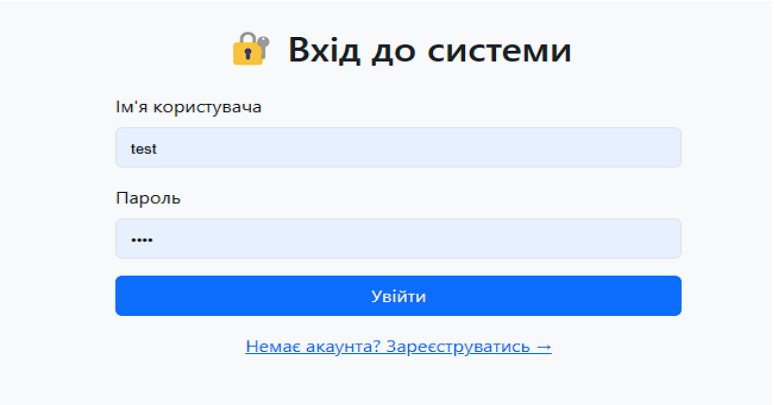


Рисунок 3.1 - Сторінка входу до системи

Сторінка реєстрації нового користувача (рис. 3.2).Забезпечує створення облікового запису з перевіркою унікальності і збереженням пароля в хешованому вигляді. Реєстрація завершується автоматичним переходом до сторінки входу.

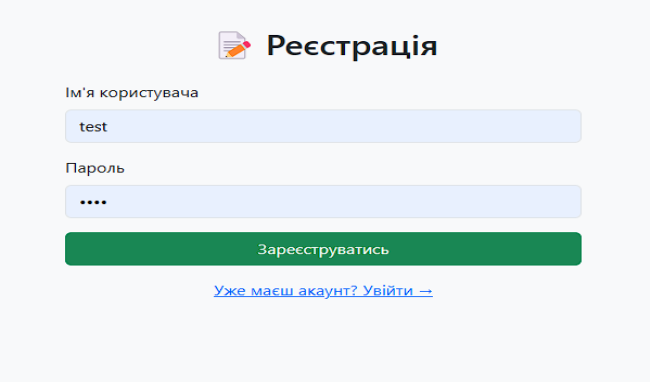


Рисунок 3.2 - Сторінка реєстрації нового користувача

Особистий кабінет користувача (рис. 3.3).Після входу користувач бачить список збережених результатів генерації з можливістю завантаження у PDF-форматі або видалення відповідного тесту. Є кнопка створення нового тесту.

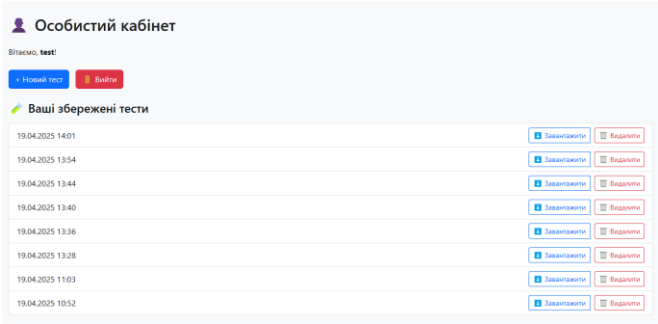


Рисунок 3.3 - Особистий кабінет користувача

Сторінка генерації тесту(рис.3.4).Забезпечує введення навчального тексту вручну або через завантаження файлу. Користувач може обрати рівень складності (легкий, середній, складний) і кількість запитань. Після натискання кнопки запускається процес генерації із використанням LLM (наприклад, Mistral 7B).

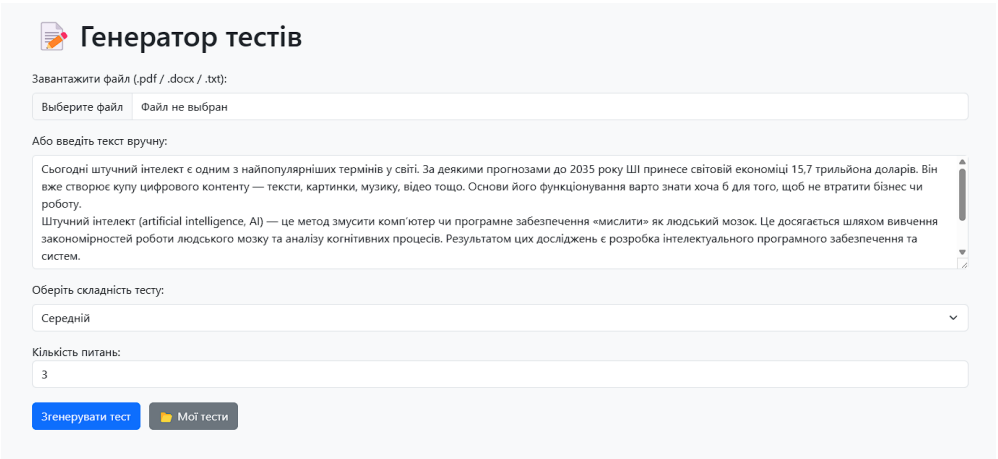


Рисунок 3.4 - Сторінка генерації тесту

Процес генерації (інтерфейс очікування) (рис. 3.5). Користувач у реальному часі бачить повідомлення про хід обробки тексту: від старту генерації до обробки кожної частини (chunk). Потокове виведення реалізовано через технологію SSE.

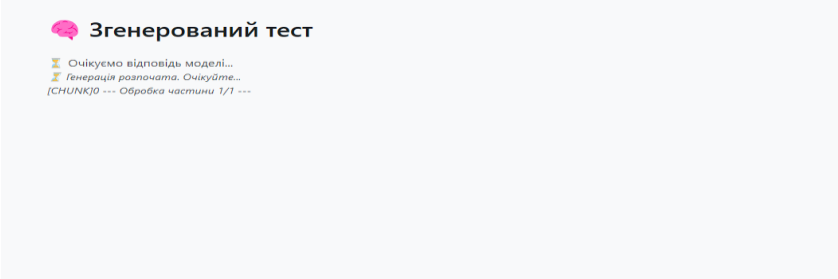


Рисунок 3.5 - Процес генерації (інтерфейс очікування)

Відображення згенерованих запитань (рис. 3.6). У процесі виводу запитання поступово з’являються на екрані у структурованому вигляді: [QUESTION], [A], [B], [C], [D], [ANSWER].

Рисунок 3.6 - Відображення згенерованих запитань

Це дозволяє перевірити форматування, якість формулювань і відповідність заданим параметрам.

Завершення генерації та збереження PDF (рис. 3.7). Після успішної генерації користувач бачить повідомлення про завершення роботи та можливість завантажити створений PDF-документ. Результат автоматично зберігається до бази даних користувача.

Рисунок 3.7 - Завершення генерації та збереження PDF(рис.3.7).

Згенерований PDF-документ (рис. 3.8). Файл містить згенеровані тестові запитання в упорядкованому вигляді, готовому для друку або подальшого використання. Це дозволяє швидко перевірити відповідність формату та якість побудованих запитань.

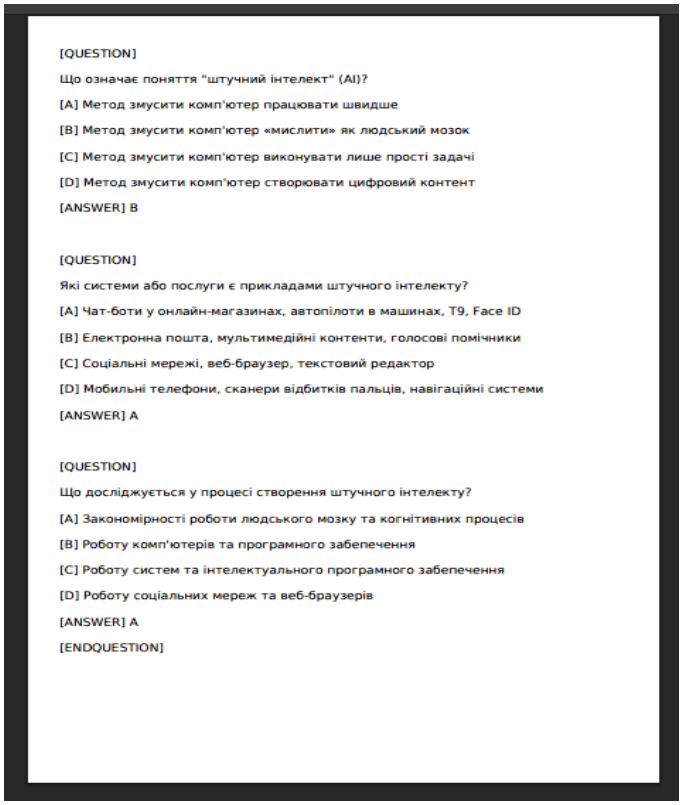


Рисунок 3.8 - Згенерований PDF-документ(рис.3.8).

Під час тестування підтверджено стабільну роботу та коректне функціонування всіх основних модулів системи: реєстрація нового користувача

із перевіркою даних, авторизація з обробкою введених облікових даних, генерація тестів з використанням мовної моделі Mistral, обробка вхідного тексту в режимі реального часу (через механізм потокового SSE), а також автоматичне збереження згенерованих результатів у базі даних користувача та їх експорт у форматі PDF. Генератор успішно обробляв як короткі абзаци, так і великі обсяги навчального матеріалу (до кількох тисяч слів), без помилок або збоїв.

Усі згенеровані запитання були логічно зв’язані зі змістом введеного тексту, відповідали вибраній складності, мали чітку структуру та дотримувалися встановленого формату ([QUESTION], [A], [B], [C], [D], [ANSWER]). Це забезпечує можливість безпосереднього використання сформованого контенту в навчальному процесі, включаючи роздрук, використання в LMS або перевірку знань у тестовій формі.

Додатково було перевірено відповідність системи не функціональним вимогам: високий рівень швидкодії, безпечне збереження даних (використання хешування паролів), підтримка стабільного сеансу користувача, інтуїтивна адаптивна верстка та коректне відображення результатів навіть при перезавантаженні сторінки. Це дозволяє зробити висновок, що система є надійним інструментом автоматизованої генерації навчального контенту і відповідає сучасним вимогам до веборієнтованих освітніх сервісів.

Висновок до розділу 3

У третьому розділі було здійснено практичну реалізацію, інтеграцію та перевірку роботи вебзастосунку для генерації тестових завдань на основі навчального тексту із використанням сучасної мовної моделі Mistral. Було створено повноцінний клієнт-серверний застосунок із потоковим інтерфейсом, підтримкою SSE, реєстрацією користувачів, генерацією запитань, збереженням історії, а також експортом результатів у PDF. Окрему увагу приділено інтеграції NLP-моделі, яка дозволяє адаптувати формування запитань до контексту тексту, що значно підвищує їх якість, змістовність та логічну послідовність. Запитання формуються в реальному часі, поступово

надсилаються користувачу, що створює ефект «живої взаємодії» і забезпечує високий рівень зручності.

У ході тестування підтверджено коректну роботу основних функціональних модулів: реєстрація, логін, генерація, вивід, збереження, експорт. Система стабільно працює при великому обсязі тексту, демонструє високий рівень точності парсингу та формування структури питань. Згенеровані PDF-документи відповідають вимогам до оформлення та зручності використання в навчальному середовищі.

Реалізований функціонал підтверджує практичну доцільність запропонованого підходу та дозволяє розглядати систему як повноцінне інструментальне рішення для автоматизованої підтримки навчального процесу. У наступному розділі буде зроблено загальний підсумок та визначено напрямки подальшого вдосконалення системи.

ВИСНОВКИ

У межах дипломної роботи було реалізовано вебзастосунок для генерації тестових завдань на основі навчального тексту з використанням сучасних мовних моделей штучного інтелекту. Система створена з урахуванням актуальних вимог до цифрових освітніх інструментів: швидкість генерації, якість формулювання, інтерактивність, персоналізація та підтримка багаторівневої структури.

У ході дослідження досягнуто наступних результатів:

Проведено детальний аналіз існуючих сервісів автоматизованої генерації тестів, виявлено їх обмеження, що обґрунтувало потребу в розробці унікального, адаптивного рішення з підтримкою української мови. Визначено ключові вимоги до системи: гнучкість формулювання, підтримка різних типів запитань, можливість обробки великих обсягів тексту, інтеграція мовної моделі, а також стабільна серверна архітектура з підтримкою потокової взаємодії (SSE).

Було розроблено повноцінну архітектуру вебзастосунку, що включає:

- клієнтську частину (HTML, Tailwind CSS, JavaScript) для зручної взаємодії з користувачем;
- серверну частину (Flask, Python) з логікою авторизації, генерації, збереження, експорту;
- інтеграцію з локальною LLM-моделлю (Mistral) через llama-cpp для обробки тексту й формування запитань;
- базу даних (SQLite) для збереження історії користувачів, згенерованих результатів і експорту у PDF через ReportLab.

Система підтримує генерацію запитань у режимі реального часу, структурує результати за шаблоном, зберігає їх до БД та дозволяє одразу експортувати у форматі PDF. Користувач має змогу обирати рівень складності, кількість питань, типи тексту. Потоковий інтерфейс забезпечує зручну інтерактивну взаємодію та миттєвий зворотний зв’язок. Результати тестування підтвердили стабільну роботу системи, високу якість генерації, адаптивність до різного обсягу текстів, а також ефективність парсингу та подальшого збереження результатів. Інтерфейс зрозумілий і доступний для непідготовлених користувачів, система легко масштабована та готова до впровадження в освітнє середовище (школи, університети, онлайн-курси).

Розроблений застосунок має значний практичний потенціал і може бути розширений у наступних напрямах: підтримка мобільної версії, мультимовність, інтеграція з LMS (Moodle, Google Classroom), адаптивна

перевірка відповідей, рейтинги складності, автоматичне оцінювання та використання в освітніх тренажерах або системах дистанційного навчання.

ПЕРЕЛІК ПОСИЛАНЬ

1. Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press, 2016.

2. Vaswani A. et al. Attention Is All You Need. – NIPS, 2017.

3. Chollet F. Deep Learning with Python. – Manning, 2017.

4. Jurafsky D., Martin J. Speech and Language Processing. – Pearson, 2021.

5. Goldberg Y. Neural Network Methods for Natural Language Processing. – Morgan & Claypool Publishers, 2017.

6. Копитин О.Є. Штучний інтелект і нейронні мережі: навч. посібник – Київ: КНУ, 2020.

7. Назаренко І.М. Інформаційні системи в освіті: методи і засоби – К., 2019.

8. Овчарук О. Цифрова трансформація освіти в Україні. – Освіта України, №4, 2020.

9. Глушков В.М. Основи штучного інтелекту. – К.: Наукова думка, 2018.

10. Шевченко А. Інформаційні технології в навчанні. – Львів: ЛНУ, 2021.

11. OpenAI. Introducing ChatGPT [Електронний ресурс]. – https://openai.com/blog/chatgpt

12. Mistral AI. Mistral 7B Model Card [Електронний ресурс]. – https://mistral.ai/news/announcing-mistral-7b/

13. Hugging Face. Transformers Documentation [Електронний ресурс]. – https://huggingface.co/docs/transformers

14. llama.cpp – Lightweight LLM inference [Електронний ресурс]. – https://github.com/ggerganov/llama.cpp

15. Python Software Foundation. Python 3 Documentation [Електронний ресурс]. – https://docs.python.org/3/

16. Flask. Flask Documentation [Електронний ресурс]. – https://flask.palletsprojects.com/

17. SQLite. SQLite Documentation [Електронний ресурс]. – https://www.sqlite.org/docs.html

18. ISO/IEC 2382:2015. Information technology – Vocabulary. – International Organization for Standardization.

19. Міністерство освіти і науки України. Концепція розвитку цифрової освіти – 2020.

20. Mairal J., Bach F., Ponce J., Sapiro G. Online Learning for Matrix Factorization and Sparse Coding. – JMLR, 2010.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Веб додаток для генерації текстових завдань на основі навчальних матеріалів

Виконала студентка 4 курсу
групи ШІ-41
Сингаєвський Андрій
Керівник роботи
К.п.н., доц., доцент кафедри ШІ ПІБ

Київ – 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – автоматизація процесу створення текстових завдань на основі навчальних матеріалів шляхом розробки веб-додатку з використанням технологій обробки природної мови.

Об'єкт дослідження – процес формування контрольних і тестових завдань у навчальному процесі.

Предмет дослідження – веб-додаток, який реалізує генерацію текстових завдань із використанням NLP та штучного інтелекту.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

Дослідити існуючі сервіси та інструменти для створення тестів на основі навчального контенту, визначити їхні переваги та недоліки.

Оцінити можливості сучасних мовних моделей (GPT, LLaMA тощо) у задачах генерації запитань.

Сформулювати вимоги до веб-додатку для генерації текстових завдань.

Спроєктувати архітектуру веб-системи з урахуванням потокової генерації, обробки PDF і роботи з навчальними матеріалами.

- Реалізувати веб-додаток з підтримкою:
- Завантаження навчальних матеріалів;
- Обробки тексту (NLP);
- Генерації запитань різних типів;
- Збереження історії та експорту у PDF.

ВИМОГИ ДО ДОДАТКУ

Функціональні вимоги:

- Реєстрація та авторизація користувачів.
- Завантаження навчальних матеріалів у форматах різних.
- Можливість введення тексту вручну для генерації завдань.
- Аналіз тексту за допомогою NLP для витягнення ключових понять.
- Генерація запитань різних типів.
- Потокове (streaming) виведення згенерованих завдань у режимі реального часу
- Експорт тестів у PDF.
- Панель адміністратора для керування генерацією.

Нефункціональні вимоги:

- Підтримка захищених сесій (https, cookies, CSRF-захист).
- Надійне зберігання паролів у базі даних (bcrypt або аналог).
- Висока швидкодія при обробці великих обсягів тексту.
- Відповідність принципам зручності інтерфейсу (UX) для використання у стресових умовах.

Flask	серверна частина вебзастосунку.
SQLAlchemy	ORM для роботи з базою даних (SQLite).
Flask-Login	система аутентифікації користувачів.
llama-cpp-python	запуск LLM моделі Mistral-7B локально
python-docx	витяг тексту з .docx файлів
ReportLab	генерація PDF-файлів
HTML, CSS, JS	інтерфейс користувача

Використані бібліотеки та їх призначення

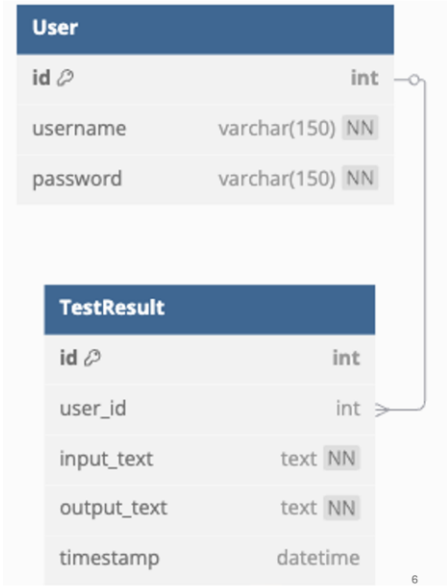
СТРУКТУРА БАЗИ ДАННИХ

Таблиця User

- id – унікальний ідентифікатор користувача
- username – логін, має бути унікальним
- password – зашифрований пароль користувача

Таблиця TestResult

- id – унікальний ідентифікатор запису
- user_id – зовнішній ключ, що посилається на User(id)
- input_text – початковий навчальний текст, введений або завантажений користувачем
- output_text – текст згенерованих запитань (тест)
- timestamp – дата та час створення генерації (встановлюється автоматично



Загальна схема роботи системи

Користувач завантажує навчальний матеріал (.docx або вручну).

Система витягує текст і розбиває його на частини

Кожен фрагмент передається до моделі Mistral-7B для генерації

Модель повертає тестові питання у форматі

- Питання
- Варіанти відповідей
- Правильна відповідь
- Результат зберігається у базі даних

Користувач переглядає, редагує або експортує тести в PDF

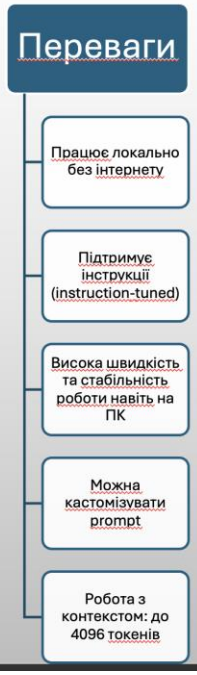
Модель штучного інтелекту Mistral-7B

Mistral-7B-Instruct – основа генерації тестів у системі

Велика мовна модель (LLM) на 7 мільярдів параметрів

gguf, сумісний із llama-cpp-python

Генерує питання з тексту, варіанти відповідей, вказує правильну відповідь



Алгоритм генерації тестових завдань

Prompt-запит до моделі Mistral-7B

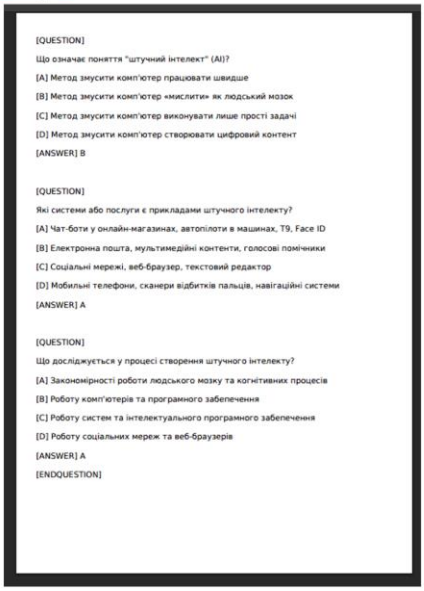
Результати генерації

Структурований список питань:

- Текст питання
- 3–4 варіанти відповідей

Позначення правильної відповіді

- Дані автоматично зберігаються в базі даних
- Доступна функція експорту в PDF.
- Виведення в особистому кабінеті

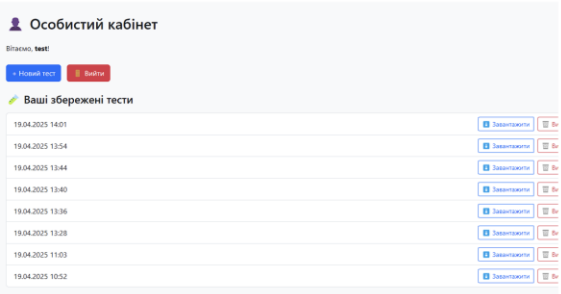


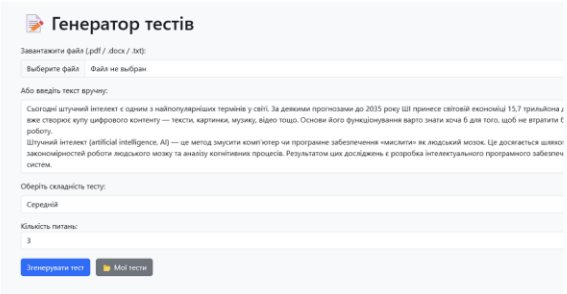
[QUESTION]
Що означає поняття "штучний інтелект" (AI)?
[A] Метод змусити комп'ютер працювати швидше
[B] Метод змусити комп'ютер «мислити» як людський мозок
[C] Метод змусити комп'ютер виконувати лише прості задачі
[D] Метод змусити комп'ютер створювати цифровий контент
[ANSWER] B

[QUESTION]
Які системи або послуги є прикладами штучного інтелекту?
[A] Чат-боти у онлайн-магазинах, автопілоти в машинах, T9, Face ID
[B] Електронна пошта, мультимедійні контенти, голосові помічники
[C] Соціальні мережі, веб-браузер, текстовий редактор
[D] Мобільні телефони, сканери відбитків пальців, навігаційні системи
[ANSWER] A

[QUESTION]
Що досліджується у процесі створення штучного інтелекту?
[A] Закономірності роботи людського мозку та когнітивних процесів
[B] Роботу комп'ютерів та програмного забезпечення
[C] Роботу систем та інтелектуального програмного забезпечення
[D] Роботу соціальних мереж та веб-браузерів
[ANSWER] A
[ENDQUESTION]

ОСОБИСТИЙ КАБІНЕТ ТА ФОРМА ГЕНЕРАЦІЇ ТЕСТІВ





Особистий кабінет користувача

- переглядати збережені тести (із зазначенням дати/часу)
- натискати для перегляду PDF або повторного експорту
- створювати нові тести.

Форма генерації тестів

- завантажити файл у форматі .docx, .pdf, .txt або ввести текст вручну
- вибрати складність (початковий, середній, складний)
- вказати кількість запитань
- згенерувати тест за допомогою ШІ

ВИСНОВКИ

Проведено аналіз літературних джерел щодо сучасних підходів до автоматичної генерації навчальних завдань та технологій обробки природної мови (NLP).

Досліджено існуючі рішення для формування тестів, визначено їхні обмеження та актуальність впровадження AI-моделі для генерації змістовних та адаптивних запитань.

Сформовано вимоги до веб-додатку, що реалізує функціонал генерації тестових завдань на основі завантажених або введених текстів.

Спроєктовано архітектуру веб-застосунку з поділом на клієнтську, серверну та модельну частини.

Реалізовано веб-додаток на основі Flask з інтеграцією NLP-моделі (LLaMA) та функцією потокового виведення запитань у реальному часі.

Додано можливість експорту тестів у форматі PDF та ведення історії генерацій, що підвищує зручність і функціональність системи.

Проведено тестування працездатності та якості згенерованих завдань, підтверджено коректність логіки та відповідність поставленим вимогам.