НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

ДО ЗАХИСТУ ДОПУЩЕНО
В.о. завідувача кафедри
_____Олександр КОВАЛЬ
«_____»_____2025р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці»
спеціальності 121 Інженерія програмного забезпечення
на тему: «Вебплатформа для створення навчальних матеріалів на основі LLM»

Виконав (-ла):
студент (-ка) IV курсу, групи _ТВ-12__
_Гончаренко Микита Едуардович_____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник:
___Доцент, к. т. н. Кузьміних В.О___ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент:
_____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент (ка)_____
(підпис)

Київ – 2025

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

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

ЗАТВЕРДЖУЮ В.о. завідувача кафедри
_____ Олександр КОВАЛЬ (підпис)
«____»_____2025р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Гончаренко Микиті Едуардовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи
_Вебплатформа для створення навчальних матеріалів на основі LLM_____
_____
керівник роботи_____Доцент, к. т. н. Кузьміних В.О_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від "__"_____202_ року №_____
2. Строк подання студентом роботи :_____
3. Вихідні дані до роботи:__Мова програмування Python, MongoDB ChromaDB, Мова програмування JavaScript_____
_____
_____
_____
4. Зміст (дипломної роботи) пояснювальної записки (перелік завдань, які потрібно розробити):__ Проведення аналізу існуючих рішень; проведення аналізу поставленої задачі; проектування системи; розробка адаптивної стохастичної системи; розробка користувацького інтерфейсу. _____
_____
5. Перелік ілюстративного матеріалу:_ Опис використаних інструментів; структура програмного забезпечення; блок-схема алгоритму; взаємодія користувачів з системою._____
_____
6. Дата видачі завдання «29» _вересня_ 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Строки виконання етапів роботи	Примітка
1	Отримання завдання	29.09.2024	
2	Дослідження предметної області	14.01.2025 - 27.01.2025	
3	Постановка вимог до проєктування системи	28.01.2025 - 01.02.2025	
4	Дослідження існуючих рішень	02.02.2025 - 10.02.2025	
5	Розробка програмного продукту	11.02.2025 – 24.04.2025	
6	Тестування	25.04.2025 – 10.05.2025	
7	Захист програмного продукту	12.05.2025	
8	Оформлення дипломної роботи	18.05.2025 – 28. 05.2025	
9	Передзахист	02.06.2025	
10	Захист	16.06.2025	

Студент _____ _Гончаренко Микита_
(підпис) (ім'я, прізвище)

Керівник роботи _____ _ Валерій Кузьміних_
(підпис) (ім'я, прізвище)

РЕФЕРАТ

Структура та обсяг роботи.

Дипломна робота викладена на 59 сторінках, містить 39 рисунків, 2 таблиці та 9 джерел.

Мета роботи — створення інтелектуальної навчальної платформи з автоматичною генерацією навчальних матеріалів на основі користувацьких запитів і завантажених файлів за допомогою мовних моделей.

Для досягнення мети виконано:

- проаналізовано сучасні підходи до генерації контенту за допомогою LLM;
- реалізовано семантичний пошук через LangChain і векторне сховище;
- створено адаптивний інтерфейс для різних пристроїв;
- реалізовано інтерактивні смарт-картки й тести;
- додано чат-бота «База знань» з можливістю відповідати на запити на основі дисципліни «Управління ІТ-проєктами»;
- створено систему обліку прогресу користувача.

Практичне значення роботи полягає у створенні повноцінного прототипу платформи, придатного для подальшого вдосконалення в магістратурі або впровадження у навчальний процес.

Апробація результатів.

Застосунок протестовано на реальних даних, порівняно з аналогами за зручністю інтерфейсу, генерації та інтерактивності.

Ключові слова:

навчальна платформа, генерація знань, LLM, LangChain, векторне сховище, смарт-картки, тести, чат-бот, FastAPI, React, MongoDB.

ABSTRACT

Structure and scope of the work.

The thesis consists of 59 pages, includes 39 figures, 2 tables and 9 references.

The aim of the work is to develop an intelligent learning platform with automatic generation of educational materials based on user queries and uploaded files using language models.

To achieve this goal, the following tasks were completed:

- analyzed modern approaches to content generation using LLMs;
- implemented semantic search using LangChain and a vector store;
- developed an adaptive user interface for different screen sizes;
- implemented interactive smart flashcards and tests;
- added a knowledge base chatbot capable of answering questions based on the discipline "IT Project Management";
- created a user progress tracking system.

Practical significance lies in the creation of a fully functional prototype of the platform, ready for further development during master's studies or for integration into the educational process.

Research validation.

The application was tested on real data and compared with analogs in terms of interface usability, content generation, and interactivity.

Keywords:

learning platform, knowledge generation, LLM, LangChain, vector store, smart flashcards, tests, chatbot, FastAPI, React, MongoDB.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IDE (Integrated Development Environment) – інтегроване середовище розробки, яке об'єднує редактор коду, компілятор, налагоджувач і інші інструменти для зручної розробки програмного забезпечення.

Frontend – клієнтська частина сайту або застосунку. Це те, що бачить користувач у браузері або на екрані пристрою, включаючи інтерфейс, кнопки, меню тощо.

Backend – серверна частина застосунку. Відповідає за обробку логіки, взаємодію з базою даних, обробку API-запитів та безпеку. Користувач безпосередньо її не бачить.

API (Application Programming Interface) – інтерфейс прикладного програмування. Набір правил, що дозволяють різним програмам або сервісам обмінюватися даними та функціональністю.

AI (Artificial Intelligence) – штучний інтелект. Сфера комп’ютерних наук, що займається створенням систем, здатних виконувати завдання, які зазвичай потребують людського інтелекту.

СУБД (Система управління базою даних) – програма або набір програм, що забезпечують створення, обслуговування та використання бази даних (наприклад, MongoDB, PostgreSQL, MySQL).

ПЗ (Програмне забезпечення) – сукупність програм і пов’язаних з ними даних, що використовуються для керування комп’ютером або виконання конкретних завдань.

БД (База даних) – організована структура для зберігання, управління та пошуку даних.

LLM (Large Language Model) – велика мовна модель, навчена на великих обсягах тексту для генерації, перекладу, відповіді на запити та інших задач обробки природної мови (наприклад, GPT-4).

RAG (Retrieval-Augmented Generation) – метод генерації тексту з попереднім пошуком у базі знань або документах. LLM отримує не лише запит, а й релевантну інформацію, знайдену в сховищі (векторна база), для точнішої відповіді.

ВСТУП

Наш світ стрімко змінюється та розвивається під впливом цифрових технологій, що охоплюють усі сфери життя, зокрема й освіту. Одним із ключових трендів останніх років стало використання штучного інтелекту для автоматизації та персоналізації навчального процесу. Особливої популярності набули великі мовні моделі (LLM), які дозволяють швидко створювати навчальні матеріали на основі вхідних даних користувача. При пошуку аналогів, на жаль, не було знайдено підходящої платформи, яка б задовольняла всі потреби: інтерактивний підхід та автоматичну генерацію матеріалів на основі моїх даних, тому підчас практики було розпочато розробку вебплатформи Dumka, яка має на меті поєднати вже існуючі рішення під дах однієї зручної платформи. Рішення базується на технологіях FastAPI, React, MongoDB та інтегрує мовну модель через LangChain і підхід RAG (Retrieval-Augmented Generation) для генерації на основі наших даних. Актуальність теми зумовлена необхідністю підвищення ефективності навчання за рахунок автоматизації рутинних процесів. Існуючі освітні сервіси часто не надають можливості швидко адаптувати матеріали під конкретні потреби студентів чи викладачів, а впровадження LLM дозволяє створювати персоналізований контент в короткий час. Метою проходження практики було поглиблення професійних навичок у сфері веброзробки, проєктування архітектури вебзастосунків та інтеграції сучасних інструментів штучного інтелекту. У процесі реалізації було розроблено повнофункціональний застосунок, впроваджено механізми авторизації, генерації матеріалів, збереження прогресу користувача та статистики.

1 МЕТА, ПОСТАНОВКА ЗАДАЧІ ТА ОСНОВНІ ЗАВДАННЯ ПРАКТИКИ

На сьогоднішній день існує багато навчальних платформ та цифрових інструментів для генерації інформації. Завдяки ним ми можемо спрощувати освітній процес, особливо за умов великих обсягів навчального матеріалу. Одним із ключових факторів популярності таких платформах це інтерактивний підхід, який надає не лише пасивне сприйняття інформації у вигляді текстів, а також активну взаємодію користувача з навчальним матеріалом. Це у значній мірі покращує засвоєння знань і робить процес навчання більш ефективним та цікавим.

Нажаль, більшість сучасних платформ, що пропонують інтерактивний формат навчання мають певні обмеження. Зокрема, вони не завжди дозволяють користувачам створювати власні інтерактивні компоненти на основі авторських або навчальних матеріалів. Це може бути критичним фактором, оскільки навчальні заклади зазвичай володіють більш точними, детальними та адаптованими до навчальних програм ресурсами. Таким чином можливість інтеграції власних навчальних матеріалів у такі системи значно підвищила б їхню ефективність та практичну цінність.

Окрім того, розвиток мовних моделей штучного інтелекту відкрив нові можливості у сфері освіти. Такі моделі здатні генерувати навчальний контент на основі заданих матеріалів, що може суттєво полегшити створення навчальних ресурсів. Проте, вони, як правило, не підтримують інтерактивний підхід, що обмежує їхню практичну застосовність у навчальному процесі.

У зв’язку з цим недоліком буде актуальним створення освітньої платформи, яка поєднувала б переваги обох підходів: здатність мовних моделей до генерації якісного навчального контенту та можливість інтерактивної взаємодії з ним. Така система сприяла б більш глибокому засвоєнню знань, забезпечувала б гнучкість у

використанні навчальних матеріалів та підвищувала ефективність освітнього процесу в цілому.

Тому метою проєкту стало створення вебплатформи «Dumka», яка б стала цінним інструментом для навчання студентів. На етапі планування було обрано дисципліну «Управління ІТ-проєктами» як ключову для формування інтерактивних навчальних матеріалів, спрямованих на підвищення кваліфікації керівників проєктів і членів команд. Основними завданнями платформи визначено створення зручного та інтуїтивно зрозумілого інтерфейсу, реалізацію можливості генерації інтерактивних матеріалів на основі методичних посібників та підручників, впровадження функціоналу для відстеження статистики проходження матеріалів, а також реалізацію пошуку матеріалів, створених іншими користувачами.

1.1 Ключові етапи розробки системи

Першочерговим завданням у процесі розробки стало визначення основних компонентів застосунку та послідовності їх реалізації. Оскільки створювана система має освітнє призначення, особливу увагу було приділено як функціональним можливостям, так і зручності користувацького інтерфейсу. Для подібного типу застосунків ці два аспекти є критично важливими. Було поставлено наступні задачі:

1) Аналіз аналогічних рішень та формування основних вимог до системи. Було складено список функцій, які повинна підтримувати платформа, а також визначено пріоритетність завдань.

2) Проєктування архітектури системи, включаючи розподіл компонентів між клієнтською та серверною частинами.

3) Вибір інструментів розробки, зокрема фреймворків, мов програмування, бази даних та технологій для інтеграції з LLM.

4) Безпосередня реалізація застосунку, що охоплює створення фронтенду, бекенду, підключення бази даних та генерації навчальних матеріалів.

5) Тестування платформи з метою перевірки відповідності функціоналу заявленим вимогам, стабільності роботи та зручності користування.

Ці загальні кроки були розбиті на менші під задачі для більш детальної розробки платформи і досягнення загального успіху в роботі системи.

Загальний функціонал системи:

- Генерація навчальних матеріалів – на основі введених користувачем даних система формує набір навчальних карток або тестових завдань з урахуванням складності та цільового формату;

- Застосування мовної моделі (LLM) – використовується генеративна модель для формування змістовних, адаптивних запитань і відповідей на основі вхідних матеріалів, включаючи техніку RAG (Retrieval-Augmented Generation);

- Створення запитів до генератора – формування внутрішніх запитів на генерацію залежно від потреб користувача (тип матеріалу, бажаний стиль тощо);

- Збереження результатів та статистики – збереження даних щодо проходження тестів, перегляду карток, позначення карток як "знані/незнані", а також відображення прогресу користувача у зручному вигляді;

- Інтерфейс користувача – зручний веб-інтерфейс із розділенням функціональності на окремі сторінки (панель керування, навчальні матеріали, профіль, статистика);

- Система авторизації – захист доступу до функцій системи, включаючи JWT-авторизацію та хешування паролів за допомогою bcrypt;

- Обмеження повторного проходження – контроль за унікальністю проходження тестів: користувач не може пройти тест повторно, поки не буде очищено результат;

- Адаптивність платформи – система дозволяє масштабуватись та розширювати функціонал у разі зміни форматів вхідних даних або навчальних сценаріїв.

1.2 Вимоги до системи

Функціональні вимоги:

- Система повинна надавати можливість користувачу створювати освітні матеріали (навчальні-картки, тести, конспекти) на основі введених даних та на основі завантажених матеріалів.

- Користувач повинен мати можливість зберігати створені матеріали у власному обліковому записі.

- Реалізація функціоналу авторизації/реєстрації з використанням JWT та безпечного зберігання паролів.

- Користувач повинен мати змогу позначати картки як «знані» або «незнані» та переглядати персональну статистику.

- Система повинна не дозволяти повторне проходження тесту до очищення попереднього результату.

- Адміністративна панель (опціонально) має забезпечувати модерацію публічних колекцій або керування користувачами.

Нефункціональні вимоги:

- Надійність — система має стабільно функціонувати при навантаженні від одночасного використання кількома користувачами.

- Безпека — особисті дані користувачів (електронна пошта, пароль, навчальні матеріали) мають бути захищені.

- Швидкодія — генерація навчальних матеріалів має займати мінімальний час.

- Адаптивність інтерфейсу — платформа повинна коректно відображатися на різних пристроях (ПК, планшет, смартфон).

- Масштабованість — система має підтримувати можливість розширення функціоналу (додавання типів матеріалів, локалізація, розширення аналітики).

- Юзабіліті — інтерфейс має бути інтуїтивно зрозумілим, без необхідності сторонніх інструкцій.

1.3 Основні завдання

Система реалізує інтерактивну веб-платформу для створення, збереження та проходження навчальних матеріалів, таких як флеш-картки та тести, з можливістю їх генерації за допомогою великих мовних моделей (LLM). Функціонал платформи охоплює як адміністративні можливості, так і зручність користування для звичайних користувачів.

Основні можливості системи:

- Аутентифікація та авторизація користувачів:
    - Реєстрація та вхід через email та пароль.
    - Захист сесії за допомогою JWT.
    - Хешування паролів (bcrypt) для безпечного зберігання.
- Генерація навчальних матеріалів:
    - Можливість завантажити документи (тексти, лекції).
    - Генерація навчальних-карток або тестів на основі LLM (через LangChain та RAG-підхід).

    - Налаштування рівня складності, стилю-мовлення та типу запитань.
- Робота з навчальними-картками:
    - Створення колекцій карток.
    - Позначення карток як “знаю / не знаю”.
    - Прогрес навчання.
- Робота з тестами:
    - Можливість проходження згенерованих тестів один раз.
    - Збереження результатів проходження.
    - Блокування повторного проходження до очищення результатів.
- Користувацький інтерфейс:
    - Сучасний дизайн з розділенням на блоки.
    - Навігація між сторінками через меню.
- Система статистики:
    - Відображення прогресу користувача по кожному типу матеріалів.
    - Збір інформації про результати тестів та рівень засвоєння матеріалу.
- Адміністрування:
    - Можливість керувати матеріалами.
    - Підтримка лише авторизованого використання платформи.

Висновки до розділу 1

У першому розділі було визначено мету та поставлено основні задачі для дипломного проекту, що полягають у створенні сучасної навчальної платформи з

інтеграцією можливостей великих мовних моделей для автоматизованого генерування контенту. Розглянуто ключові етапи розробки системи, серед яких: проєктування архітектури, формування вимог, вибір технологій, створення інтерфейсу та реалізація логіки генерації і взаємодії з користувачем.

Сформульовано функціональні та нефункціональні вимоги до системи, що включають адаптивність, масштабованість, інтерактивність та безпеку збереження даних. Також було деталізовано основні завдання, зокрема створення інтерфейсу, реалізація генерації матеріалів, впровадження семантичного пошуку та системи відстеження прогресу.

2 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ, МЕТОДІВ І МОДЕЛЕЙ

Коли вирішуєш розробляти будь який продукт слід завжди спочатку проаналізувати ринок, які аналогічні програми вже існують і чим відрізняються один від одної. Важливо не копіювати систему а придумати чим моя реалізація буде кращою за інші і чому саме моєю платформою будуть користуватись.

2.1 Аналіз подібних рішень та сервісів

Для аналізу існуючих рішень у сфері створення та використання освітніх матеріалів були розглянуті такі популярні платформи, як Quizlet, Kahoot!, Anki та ChatGPT. Кожна з них має унікальний набір функцій, які можуть бути корисними при реалізації власної платформи. Метою аналізу було виявлення сильних сторін кожного сервісу та подальше поєднання їх у єдину, гармонійно реалізовану систему.

Quizlet спеціалізується на створенні та використанні флешкарт, що активно застосовується в навчальному процесі. Важливою перевагою є можливість обміну матеріалами між користувачами, що робить платформу зручною для освітніх цілей. Однак платформа має низку обмежень: відсутня автоматизація створення контенту, не підтримується інтерактивне тестування, а також бракує інструментів для комплексного супроводу навчального процесу.

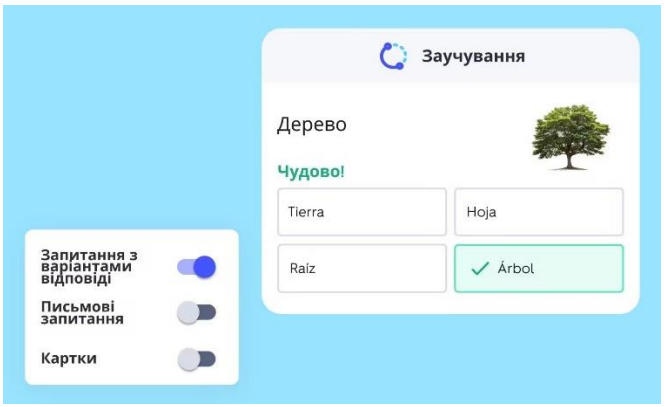


Рисунок 2.1 – Застосунок Quizlet

Kahoot! орієнтований на створення інтерактивних вікторин і використання гейміфікованого підходу до навчання. Такий підхід особливо ефективний у групових заняттях і дозволяє зробити навчання динамічним та залученим. Водночас платформа не передбачає можливості роботи з спеціальними навчальними матеріалами і є малопридатною для індивідуального навчання.

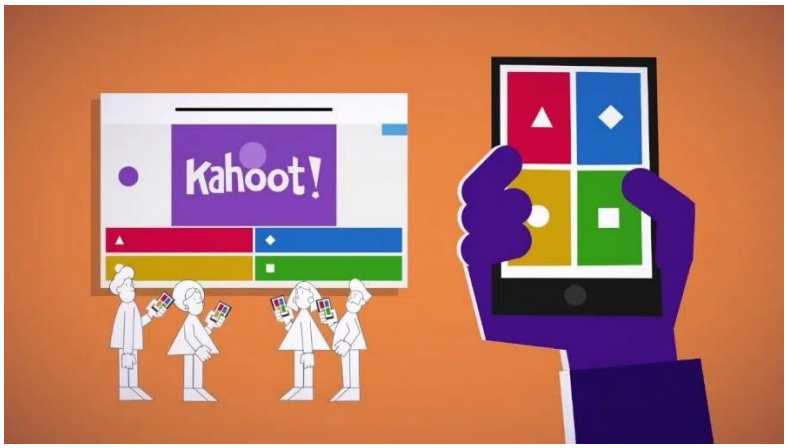


Рисунок 2.2 – Застосунок Kahoot!

Anki вирізняється використанням алгоритмів інтервального повторення, що позитивно впливає на запам’ятовування інформації. Система дозволяє створювати гнучко налаштовані картки для навчання. Проте інтерфейс програми є складним для нових користувачів, а відсутність інтерактивних функцій та тестування обмежує її освітній потенціал.

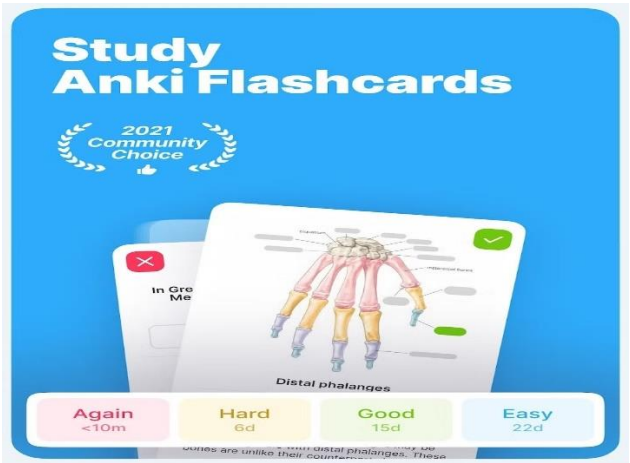


Рисунок 2.3 – Застосунок Anki

Duolingo це платформа що спеціалізується на вивченні іноземних мов і здобула широку популярність завдяки поєднанню освітньої ефективності та інтерактивного підходу. Програма використовує елементи гейміфікації, що підтримують зацікавленість користувача, а також механізми повторення для закріплення матеріалу. Такий формат навчання демонструє потенціал інтерактивних технологій у підвищенні мотивації та залучення аудиторії.

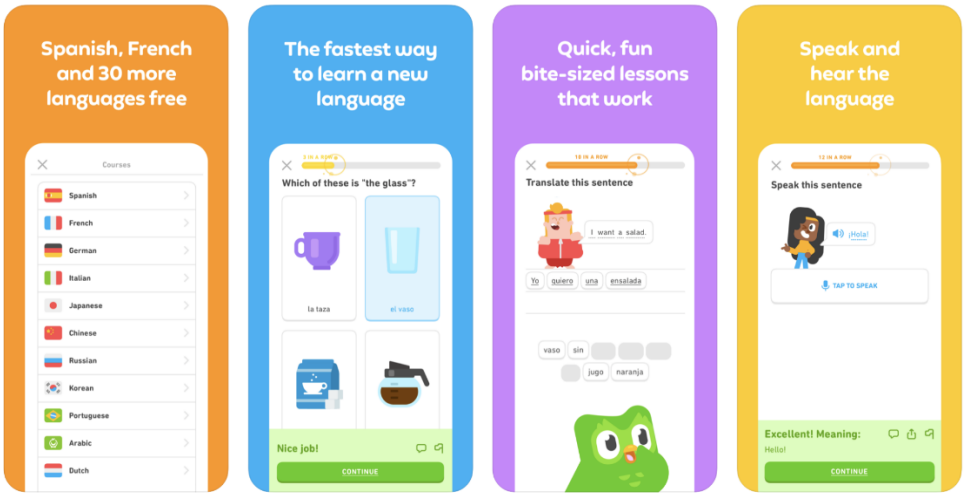


Рисунок 2.4 – Застосунок Duolingo

ChatGPT API дозволяє генерувати різноманітні тексти та навчальні матеріали за індивідуальними запитами. Водночас платформа не надає готових рішень для збереження, структуризації чи тестування матеріалів, а також не інтегрується з системами для організації навчального процесу.

На основі проведеного аналізу було визначено ключовий функціонал, який доцільно реалізувати у власному проєкті. Він охоплює генерацію навчального контенту за допомогою LLM, збереження матеріалів у вигляді карток, тестів та колекцій, інтерактивну взаємодію з користувачем, а також зручний інтерфейс для перегляду та аналізу прогресу. Ключові моменти порівняння аналогів були занесені у таблицю 2.1 для наглядного і зручного порівняння та аналізу сильних та слабких сторін, щоб поєднати найкращий функціонал та створити приємний для використання інтерфейс додатка.

Таблиця 2.1 - Порівняння аналогів

Платформа	Навчальні картки	Тести	Автоматизація	Інтерактивність	Власні дані для матеріалів
Quizlet	Так	Так	Ні	Середня	Так, але не автоматично
Kahoot!	Ні	Так	Ні	Висока	Частково так
Anki	Так	Ні	Ні	Низька	Не автоматично
Duolingo	Так	Частково	Ні	Висока	Ні
Дипломний проєкт «Dumka»	Так	Так	Так	Висока	Так

2.2 Проблеми, які вирішує розроблена система

Як уже було розглянуто у попередніх розділах, основною метою створеної системи є розробка навчальної платформи, яка буде ефективною для самопідготовки та засвоєння знань на основі індивідуальних матеріалів користувача або академічної літератури. У процесі навчання студенти часто стикаються з проблемами, пов'язаними з пошуком та структуризацією якісних навчальних матеріалів, нестачею часу на створення флешкарт або тестів, а також з труднощами в оцінці власного прогресу.

Запропонована система вирішує ці проблеми шляхом автоматизації процесу створення навчального контенту. Користувачу достатньо лише надати джерело (наприклад, конспект, підручник чи інший документ) або вказати тему, а система на основі LLM згенерує структуровані флешкарти, тести або інші інтерактивні елементи. Це значно зменшує навантаження на користувача та підвищує ефективність підготовки.

Крім того, система усуває проблему фрагментованості: замість використання кількох окремих сервісів користувач отримує єдине інтегроване середовище. Також платформа вирішує проблему адаптації до змін у навчальному контенті: при оновленні джерел система дозволяє повторно генерувати актуальні матеріали без

ручного втручання. На проблематику та її вирішення можна подивитись у таблиці 2.2.

Таблиця 2.2 - Проблематику та її вирішення

Проблема	Як вирішується в системі
Необхідність витрачати багато часу на створення навчальних матеріалів	Автоматична генерація карток і тестів на основі введеного тексту або документів
Відсутність інтегрованої системи для карток і тестування	Об’єднання обох форматів навчання в одному застосунку
Складність у відстеженні особистого прогресу	Наявність модуля статистики, який зберігає результати і прогрес навчання
Обмеження платформ типу Quizlet, Anki, Kahoot (одностороння функціональність)	Інтеграція найсильніших сторін кожної з платформ: інтервальне повторення, тестування, генерація контенту
Відсутність автоматизації навчання на основі університетських матеріалів	Підтримка PDF та інших форматів для генерації карток без потреби в ручному створенні
Складність повторення пройденого матеріалу	Інтервальні нагадування, статус “вивчено/не вивчено” у картках, збереження історії

Таким чином, розроблена система забезпечує гнучкий, інтуїтивний та сучасний інструмент для самостійного навчання, усуваючи ключові бар'єри, з якими стикаються користувачі під час підготовки.

2.3 Опис предметної області

Для забезпечення ефективності та практичної цінності розроблюваного застосунку як основну предметну область було обрано дисципліну «Управління ІТ-проєктами». Такий вибір дозволяє наочно продемонструвати функціональні можливості платформи та перевірити її працездатність у реальному навчальному середовищі.

Управління ІТ-проєктами є складним багаторівневим процесом, що охоплює планування, координацію, реалізацію та контроль над розробкою ІТ-продуктів і сервісів. Основною метою такого управління є досягнення результату в межах визначених термінів, бюджету та якості. Системний підхід до управління

проєктами сприяє оптимізації робочих процесів, підвищенню продуктивності команд та ефективному використанню ресурсів.

У межах дипломного проєкту дисципліна «Управління ІТ-проєктами» виступає основною темою для генерації навчальних матеріалів. Це дає змогу реалізувати ключові функції платформи: створення карток, тестів, формування структурованих знань для підвищення кваліфікації як керівників проєктів, так і членів ІТ-команд. Таким чином, платформа демонструє свою актуальність, практичну цінність і здатність підтримувати професійне навчання в конкретній предметній галузі.

Висновки до розділу 2

У другому розділі було проведено порівняльний аналіз схожих освітніх рішень, зокрема платформ, які надають функціонал створення тестів і карток. Виявлено, що більшість із них або не використовують генерацію за допомогою LLM, або не підтримують інтерактивну взаємодію з контентом. Розроблена система вирішує ці обмеження, поєднуючи генерацію на основі матеріалів користувача з можливістю самоперевірки через картки й тести. У підрозділі про предметну область було охарактеризовано контекст дисципліни «Управління ІТ-проєктами», на основі якої побудоване векторне сховище.

3 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У цьому розділі наведено обґрунтування вибору технологій, інструментів і середовища, що були використані під час розробки програмного забезпечення. Правильний вибір стеку напряму впливає на зручність реалізації, масштабованість, безпеку та продуктивність системи.

3.1 Вибір стеку технологій

Під час вибору технологій для розробки системи було проаналізовано кілька альтернатив з урахуванням таких критеріїв: зручність розробки, продуктивність, можливість масштабування, підтримка спільноти та інтеграція з сучасними інструментами.

Першим завданням було визначитись на чому буде написана серверна частина застосунку, адже вся основна логіка та взаємодія з LLM буде відбуватись саме там. Я обрав мову програмування Python бо вже маю багатий досвід роботи з нею і не одноразово створював застосунки з використанням мовних моделей. Python виділяється на фоні інших мов для серверної частини великою кількістю фреймворків та бібліотек що дають змогу будувати різносторонні високонавантажені застосунки з використанням технологій штучного інтелекту.

Після вибору мови програмування треба обрати спосіб написання бекенду, на Python є три основні фреймворки для цього:

- Django — потужний фреймворк з великою кількістю вбудованих можливостей.

- Flask — мінімалістичний, гнучкий, але потребує ручної реалізації багатьох функцій.

- FastAPI — сучасний, асинхронний фреймворк з автоматичною генерацією документації.

Вибір зупинився на FastAPI, оскільки він дозволяє легко створювати високопродуктивний асинхронний API, забезпечує валідацію даних, і має хорошу документацію та підтримку інтеграцій.

Не менш важливою складовою будь-якого сучасного застосунку є його клієнтська частина, адже саме інтерфейс створює перше враження користувача. Навіть високофункціональні системи часто втрачають потенційну аудиторію через недосконалий або застарілий фронтенд. Для більшості користувачів саме зовнішній вигляд та зручність взаємодії є визначальними факторами у виборі продукту — а не складність чи якість серверної логіки, прихованої «під капотом».

З огляду на це, у межах мого проєкту постала задача створити сучасний, інтуїтивно зрозумілий та гнучкий інтерфейс, який водночас добре поєднувався б із потужними серверними можливостями. Для реалізації клієнтської частини було розглянуто кілька популярних JavaScript-фреймворків:

- Vue.js — легкий та простий у використанні фреймворк, який ідеально підходить для швидкого створення інтерфейсів;
- Angular — повноцінне фронтенд-рішення для великих корпоративних застосунків із вбудованими інструментами для масштабування;
- React — компонентно-орієнтована бібліотека з величезною екосистемою, активною спільнотою та широким набором готових рішень.

Зважаючи на власний досвід та потреби проєкту, я обрав React. Цей підхід забезпечив високу модульність, гнучкість структури, а також легкість у підтримці й масштабуванні застосунку. Крім того, React має велику кількість сторонніх бібліотек для інтерфейсних компонентів, що значно прискорює розробку.

З-поміж багатьох доступних UI-бібліотек мій вибір пав на DaisyUI — компонентну бібліотеку, створену як надбудову над Tailwind CSS. Основними перевагами DaisyUI є:

- Проста інтеграція з Tailwind CSS, який вже широко використовується у сучасних фронтенд-проєктах;

- Готові, стилізовані компоненти, які дозволяють швидко створювати інтерфейс без необхідності писати вручну великий обсяг CSS;
- Можливість гнучкої кастомізації тем оформлення (light/dark та інші);
- Легка вага — бібліотека не перевантажує проєкт зайвими залежностями;
- Активна спільнота та регулярні оновлення, що забезпечує надійність і актуальність у довготривалій перспективі.

На відміну від більш складних рішень, таких як Material UI або Ant Design, DaisyUI дає більше свободи в стилізації без нав'язування строгої візуальної парадигми. Саме ця гнучкість і простота стали визначальними факторами при виборі, адже для освітнього застосунку важливо мати зрозумілий, легкий і водночас естетичний інтерфейс, який не відволікає користувача від основного функціоналу.

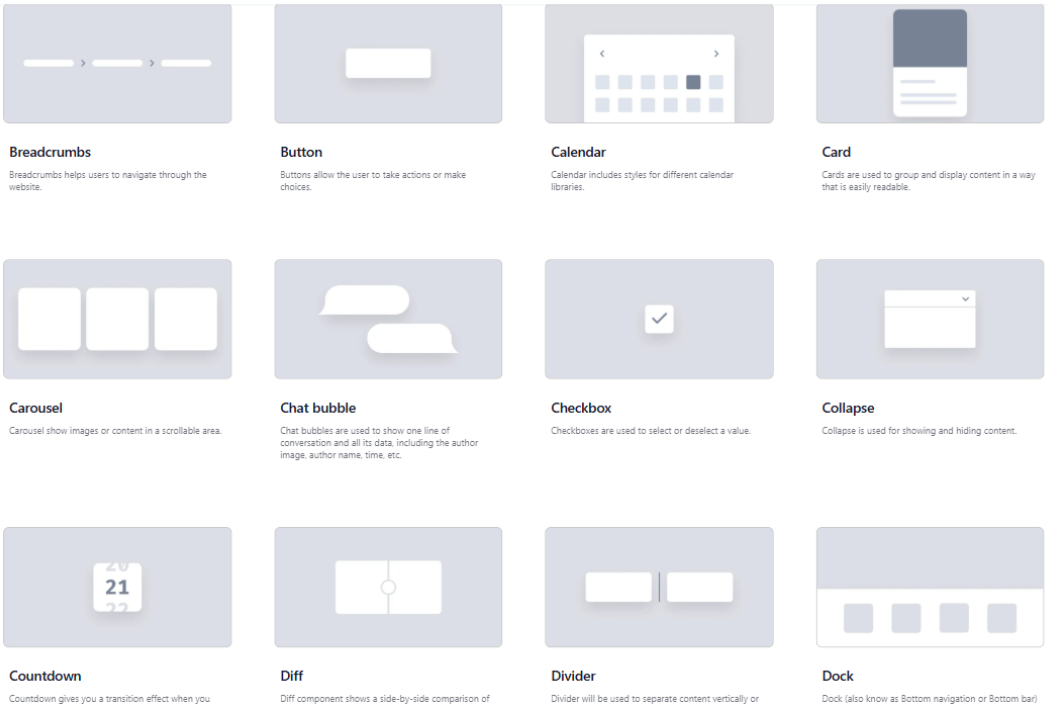


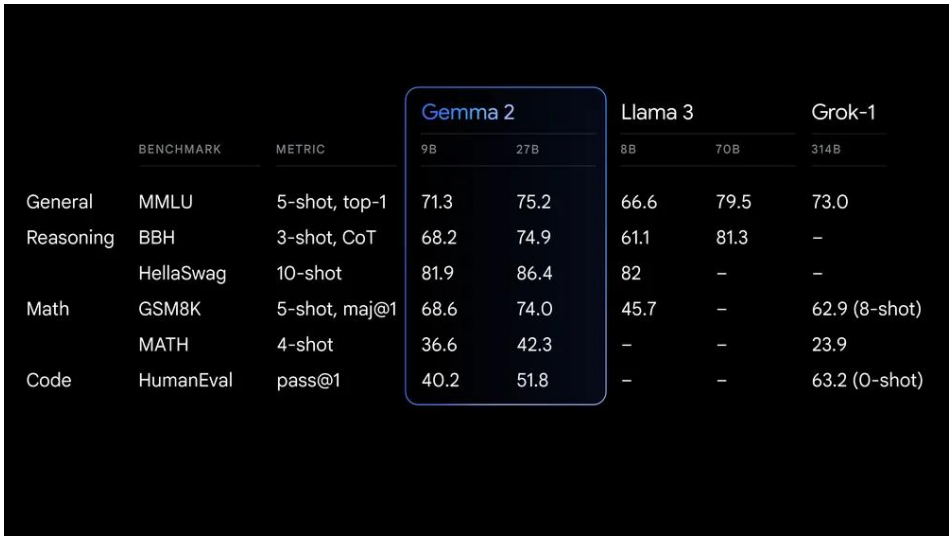
Рисунок 3.1 – Приклад компонентів DaisyUI

Для збирання клієнтської частини використовується Vite — сучасний інструмент, який забезпечує блискавичну швидкість старту та гаряче перезавантаження під час розробки. Така комбінація дозволяє зберігати високу продуктивність розробницького процесу без шкоди для зручності користувача.

Після визначення технологічногій для серверної та клієнтської частини системи настав час обрати ядро – мовну модель, яка відповідає за генерацію навчального контенту. Оскільки на сьогоднішній день існує велика кількість різних моделей, важливо було провести попереднє дослідження з метою підбору оптимального рішення для поставлених задач.

Першим етапом цього дослідження стало тестування локальних моделей за допомогою інструменту Ollama, який дозволяє запускати велику кількість мовних моделей безпосередньо на локальному пристрої. Я протестував модель Gemma 2 у двох варіантах з 2 та 9 мільярдами параметрів. Модель з 2 мільярдами параметрів не забезпечувала достатньої якості відповідей і не відповідала вимогам проєкту. Натомість версія з 9 мільярдами параметрів показала значно кращі результати, успішно справляючись із більшістю завдань.

Проте використання локальної мовної моделі має значні обмеження зокрема, високі вимоги до апаратного забезпечення. У більшості випадків її доцільно запускати лише на власних серверах або високопродуктивних робочих станціях. У зв’язку з цим було прийнято рішення шукати інші, більш доступні та масштабовані альтернативи.



			Gemma 2		Llama 3		Grok-1
	BENCHMARK	METRIC	9B	27B	8B	70B	314B
General	MMLU	5-shot, top-1	71.3	75.2	66.6	79.5	73.0
Reasoning	BBH	3-shot, CoT	68.2	74.9	61.1	81.3	–
	HellaSwag	10-shot	81.9	86.4	82	–	–
Math	GSM8K	5-shot, maj@1	68.6	74.0	45.7	–	62.9 (8-shot)
	MATH	4-shot	36.6	42.3	–	–	23.9
Code	HumanEval	pass@1	40.2	51.8	–	–	63.2 (0-shot)

Рисунок 3.2 – Порівняння моделі gemma2 з іншими аналогами

Таким чином, я розпочав пошук сервісів, які надають доступ до мовних моделей через API. Такий підхід дозволяє забезпечити високу швидкість обробки

запитів, стабільність роботи системи та належну якість генерації контенту без необхідності розгортання власної інфраструктури.

Я звернув увагу на найпопулярніше рішення ChatGPT від компанії OpenAI. Він є лідером у сфері генеративного ШІ та пропонує широкий вибір моделей із високою точністю та стабільністю роботи. Проте в процесі аналізу я зіштовхнувся з серйозним обмеженням - вартість використання моделей виявилась надто високою для використання в межах дипломного проєкту.

У зв'язку з цим виникла потреба пошуку альтернативних рішень. Одним з останніх гучних анонсів стала поява китайської мовної моделі DeepSeek, яку в мережі активно обговорювали як потенційного "вбивцю ChatGPT". Я вирішив протестувати цю модель і в процесі ознайомлення виявилось, що вартість її використання є в декілька разів нижчою за аналоги, при цьому якість генерації залишалася на високому рівні. Це робить DeepSeek перспективним варіантом для інтеграції в систему.

Модель / Провайдер	Вхідні токени (звичайні)	Вхідні токени (кешовані)	Вихідні токени	Контекст (макс.)
DeepSeek Chat (V3)	$0.27	$0.07	$1.10	64K
DeepSeek Reasoner (R1)	$0.55	$0.14	$2.19	64K
OpenAI GPT-4o	$2.50	$1.25	$10.00	128K
OpenAI GPT-4.5	$75.00	$37.50	$150.00	128K
OpenAI o3-mini	$1.10	$0.55	$4.40	200K

Рисунок 3.3 – Порівняння цін OpenAI та DeepSeek

За ціну, що є в кілька разів нижчою порівняно з лідерами ринку, ми отримуємо доступ до однієї з найпотужніших сучасних мовних моделей — DeepSeek V3, яка має 671 мільярд параметрів. Це надзвичайно вигідне рішення з огляду на співвідношення ціни та якості, що робить його особливо привабливим для проєктів з обмеженим бюджетом.

Зважаючи на це, було прийнято остаточне рішення використовувати саме DeepSeek V3 як основну мовну модель у системі генерації навчальних матеріалів.

Такий вибір дозволяє забезпечити високу якість відповіді, швидкість обробки запитів та економічну доцільність у довгостроковій перспективі.

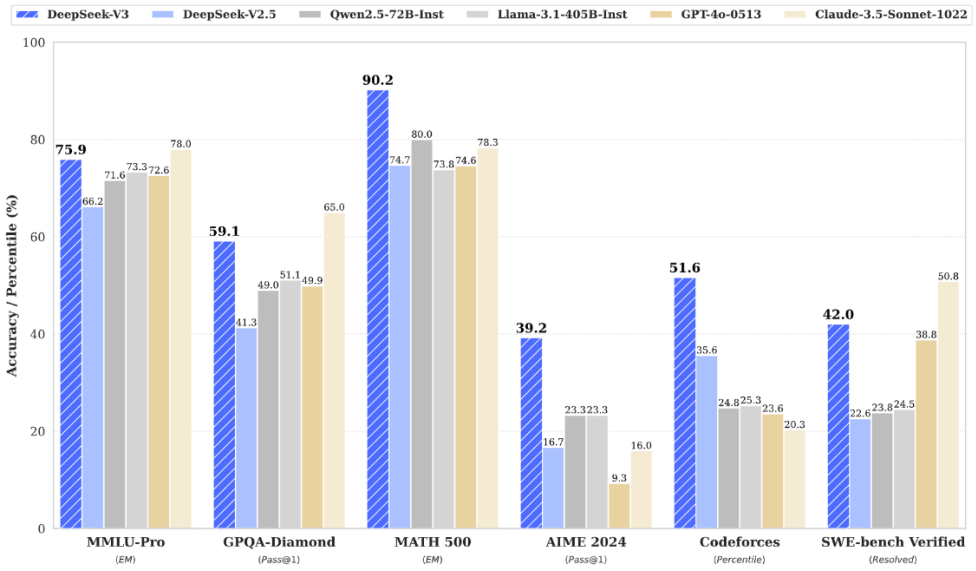


Рисунок 3.4 – Порівняння DeepSeek V3 з аналогами

Крім того, DeepSeek має чудову інтеграцію з фреймворком LangChain, що суттєво спрощує процес підключення моделі до власного проєкту. Завдяки підтримці відповідного інтерфейсу, інтеграція відбувається швидко та без додаткових ускладнень, що дозволяє зосередитись безпосередньо на реалізації бізнес-логіки замість налаштування низькорівневих API-запитів. Така сумісність робить DeepSeek не лише економічно вигідним, а й технічно зручним вибором для впровадження у систему генерації навчальних матеріалів.

```python
import os
from dotenv import load_dotenv
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain.chat_models import ChatOpenAI

load_dotenv()

class Model:  4 usages
    def __init__(self):
        # Підключення embeddings через LangChain адаптер
        self.embeddings_model = HuggingFaceEmbeddings(
            model_name='sentence-transformers/all-MiniLM-L6-v2'
        )

        # Використання DeepSeek LLM через OpenAI-сумісний API
        self.model_deep_seek = ChatOpenAI(
            model="deepseek-chat",
            temperature=1.5,
            openai_api_base="https://api.deepseek.com/v1",  # Цей URL незмінний
            openai_api_key=os.getenv("DEEPSEEK_API_KEY")  # API-ключ беремо з .env
        )
```

Рисунок 3.5 – Підключення моделі DeepSeek у коді

Наступним важливим етапом у розробці повноцінної багатофункціональної системи стало визначення підходящої бази даних. Оскільки мій застосунок є навчальною платформою, логічно передбачити збереження великого обсягу інформації: даних про зареєстрованих користувачів, навчальних матеріалів, результатів тестування, прогресу проходження курсів тощо.

На сучасному етапі розвитку технологій найчастіше використовуються три основні типи рішень:

- Реляційні бази даних — класичний варіант для структурованих систем (наприклад, PostgreSQL);
- Хмарні бази даних, орієнтовані на швидкий запуск і масштабування (Firebase Realtime DB / Firestore);
- Документо-орієнтовані бази даних, зокрема MongoDB, які належать до типу NoSQL.

Попри популярність реляційних СУБД, які зручні у розробці та забезпечують чітку схему зберігання, я вирішив відмовитися від них на користь більш гнучкого рішення. Зокрема, через те, що структура даних у моєму застосунку є динамічною, неоднорідною та схильною до змін у майбутньому (наприклад, навчальні картки, адаптивні тести, індивідуальні відповіді, різні типи статистики).

Варіант з Firebase також було відхилено, оскільки це комерційне хмарне рішення, яке передбачає регулярні витрати, а також обмежує гнучкість у розгортанні локального або контейнеризованого середовища.

Зрештою, після аналізу вимог та оцінки доступних варіантів, було прийнято рішення використати MongoDB як основну базу даних. Це рішення продиктоване насамперед гнучкістю структури зберігання, яка ідеально відповідає характеру даних у застосунку — картки, відповіді, навчальні модулі, прогрес користувача тощо мають різну структуру та можуть змінюватися в процесі розвитку системи. MongoDB дозволяє зберігати такі об’єкти у вигляді документів без необхідності жорстко визначати схему, що значно спрощує як початкову розробку, так і подальше масштабування.

Крім того, MongoDB добре інтегрується з середовищем розробки на Python, а також дозволяє працювати як у локальному, так і у хмарному середовищі без критичних обмежень. Враховуючи, що на початковому етапі важливо було мати контрольоване середовище з можливістю швидкого розгортання, MongoDB стала оптимальним вибором для реалізації першої стабільної версії застосунку.

У межах проєкту, окрім основної бази даних, виникла необхідність у використанні векторної бази даних для зберігання embedding-представлень даних, що застосовуються в механізмах пошуку та генерації контенту. На сьогоднішній день існує низка популярних рішень у цій сфері, зокрема:

- ChromaDB
- Pinecone
- Qdrant

Кожна з цих систем має широкий функціонал і здатна забезпечити високопродуктивний пошук по векторному простору. Втім, з огляду на вимоги до гнучкості архітектури, швидкості інтеграції та можливості подальшого масштабування, було прийнято рішення на початковому етапі використати ChromaDB. Це локальна векторна база, яка відзначається легкою інтеграцією з Python та повною відповідністю поточним технічним потребам проєкту.

У перспективі, з розвитком системи, планується перехід на потужніші векторні сховища — такі як Pinecone чи Qdrant. Їх впровадження дозволить суттєво підвищити ефективність пошуку релевантних embedding-відповідників, оптимізувати час відгуку застосунку та забезпечити масштабовану обробку великих обсягів векторних даних.

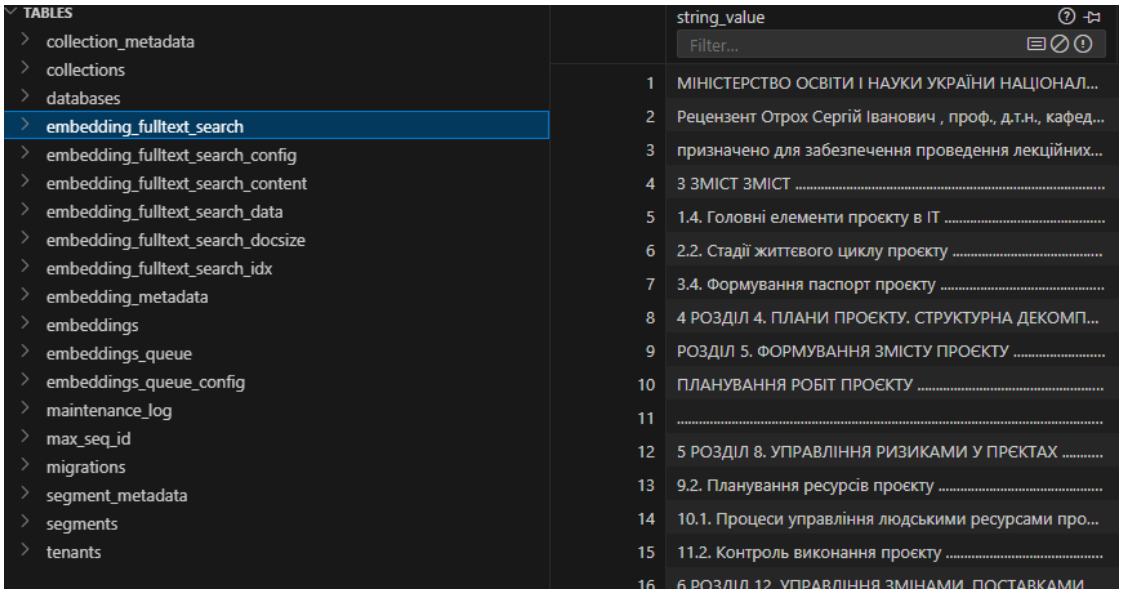


Рисунок 3.6 – Приклад зберігання даних у ChromaDB

Сучасна навчальна система передбачає обов’язкову наявність аутентифікації та облікових записів користувачів. Враховуючи специфіку застосунку, було прийнято рішення реалізувати власну систему аутентифікації. На початковому етапі було проаналізовано найбільш поширені сучасні підходи:

- Серверні сесії (збереження стану на сервері)
- OAuth-провайдери (Google, GitHub тощо)
- JWT (JSON Web Token) — безстатевий токенний підхід

У підсумку я обрав саме JWT, оскільки він найкраще підходить для SPA-архітектури (Single Page Application) — з ним немає потреби у серверних сесіях, що полегшує масштабування та забезпечує повну автономність клієнтської частини.

JWT-токен після успішної автентифікації зберігається у cookies, доступних із JavaScript. Це дозволяє гнучко керувати авторизаційним станом прямо з клієнтської частини (наприклад, для автоматичного оновлення інтерфейсу або логіки перенаправлення). Такий підхід обрано тому, що він спрощує реалізацію SPA-логіки та не потребує складної конфігурації запитів із додатковими заголовками, як у випадку з токенами в Authorization header.

Щодо зберігання паролів, то система жодним чином не зберігає їх у відкритому вигляді. При реєстрації паролі шифруються за допомогою бібліотеки

bcrypt (Python), яка забезпечує сучасний рівень безпеки за рахунок соління та багатократного хешування. Під час логіну відбувається перевірка введених даних з використанням bcrypt-методів верифікації.

Рисунок 3.7 – Приклад зберігання паролів у БД

Для генерації контенту на основі користувацьких запитів потрібно було реалізувати зв’язок із мовною моделлю. Вибір зроблено на користь LangChain з використанням підходу RAG (Retrieval-Augmented Generation).

Під час вибору мовної моделі основними критеріями були швидкодія, зручність використання та вартість. Розглядалися варіанти локального розгортання моделей, однак такий підхід вимагає високопродуктивного обладнання, що не відповідало доступним технічним можливостям.

У результаті було прийнято рішення зосередитися на використанні хмарних сервісів, які надають доступ до моделей через API. Серед кількох доступних рішень було обрано DeepSeek, оскільки цей сервіс пропонує високу якість генерації тексту, конкурентну швидкість відповіді та доступну вартість використання API. Завдяки цьому вдалося інтегрувати LLM у проєкт ефективно й без надмірних витрат на інфраструктуру.

Це дозволяє поєднати генеративні можливості LLM із контекстом з наданих файлів (наприклад, навчальних документів), що значно підвищує точність і користь згенерованого матеріалу.

Таким чином, обраний стек технологій виглядає так:

- Python FastAPI для бекенду
- React + Vite та DaisyUI для фронтенду
- MongoDB для бази даних
- ChromaDB для зберігання векторних ембедингів
- JWT + bcrypt для авторизації
- LangChain + RAG для генерації навчальних матеріалів

- DeepSeek китайсяка LLM для генерації контенту

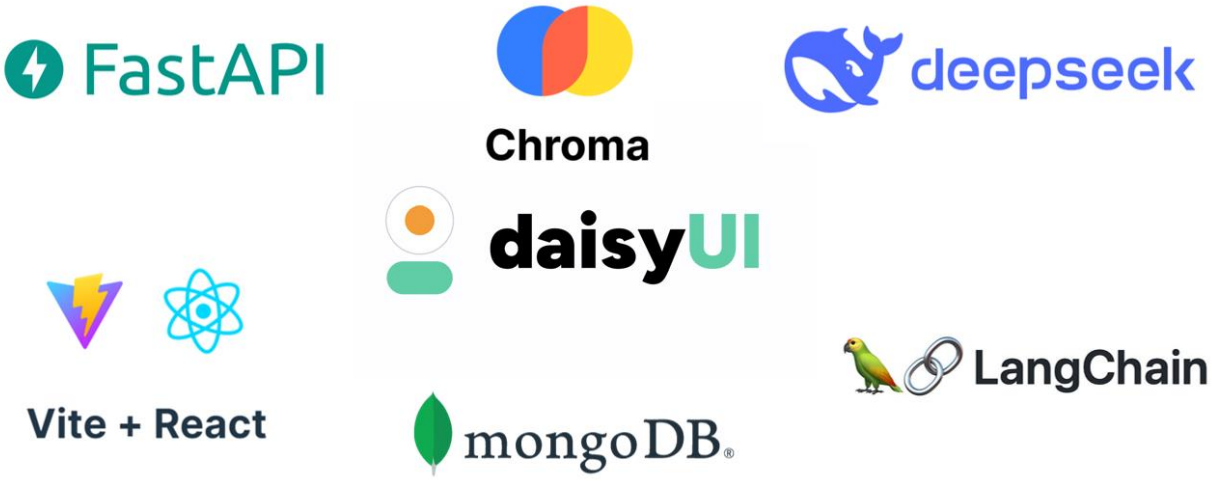


Рисунок 3.8 – Основний стек технологій

3.2 Використані інструменти та бібліотеки

Для забезпечення функціональності системи було використано низку бібліотек і фреймворків, які доповнюють основний стек і спрощують реалізацію окремих підсистем.

На бекенді (FastAPI):

- pydantic — для валідації та типізації вхідних даних.
- fastapi — головний фреймворк для створення REST API.
- motor — асинхронний драйвер для MongoDB, що дозволяє ефективно працювати з базою даних у рамках FastAPI.
- bcrypt — для безпечного хешування паролів користувачів.
- python-jose — для створення та перевірки JWT-токенів.
- langchain — бібліотека для роботи з LLM та побудови генеративних ланцюжків з підтримкою RAG.
- unstructured, chromadb — для попередньої обробки документів та побудови векторних індексів, що використовуються при генерації контенту.

На фронтенді (React):

- react-router-dom — для організації маршрутизації між сторінками.

- axios — для виконання HTTP-запитів до бекенду.
- jwt-decode — для зчитування інформації з токена доступу.
- tailwindcss — для швидкої розробки інтерфейсу у заданому стилі.
- daisyUI — набір готових React-компонентів із підтримкою Tailwind.
- framer-motion — для додавання плавних анімацій, що покращують візуальне сприйняття.

3.3 Середовище розробки та інфраструктура

Для організації ефективного робочого процесу було обрано сучасні інструменти, які спрощують розробку, налагодження та розгортання системи.

Основна розробка бекенду здійснювалась у середовищі PyCharm, яке забезпечує зручну роботу з Python-проєктами, має вбудовану підтримку для Docker, інтеграцію з системами контролю версій та широкі можливості для налагодження коду. Фронтенд розроблявся у Visual Studio Code, який підтримує численні розширення для JavaScript, React і TailwindCSS.

Для контейнеризації проєкту застосовано Docker: бекенд, фронтенд та база даних описані як окремі сервіси у файлі docker-compose.yml, що дозволяє запускати всю інфраструктуру за допомогою однієї команди.

Локально бекенд запускався у віртуальному середовищі Python, а фронтенд через локальний сервер Vite. У продакшн-середовищі вся система працює в контейнерах Docker.

Висновки до розділу 3

У третьому розділі обґрунтовано вибір сучасного технологічного стеку: FastAPI для бекенду, React для фронтенду та MongoDB як документоорієнтованої бази даних. Вказано інструменти, що полегшують розробку, зокрема LangChain для взаємодії з LLM, bcrypt та JWT для безпеки. Розгортання відбувається в Docker-

інфраструктурі. Створене середовище є гнучким, масштабованим і готовим до подальшого розвитку, наприклад у межах магістерської роботи.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Архітектура системи

Під час вибору відповідної архітектури системи було проведено аналіз функціональних і нефункціональних вимог, які необхідно було виконати. Однією з ключових умов була обмеженість у часі, тож архітектурне рішення мало бути достатньо простим для швидкої реалізації, але водночас — достатньо гнучким і масштабованим.

На початковому етапі було відхилено варіант монолітної архітектури, оскільки він не відповідав специфіці проєкту. Зокрема, передбачувана кількість функціональних модулів і перспектива подальшого розширення зробили монолітний підхід малопридатним.

Альтернативним варіантом розглядалася мікросервісна архітектура, яка сьогодні є однією з найпопулярніших у галузі. Вона дозволяє створювати незалежні модулі, які легко масштабуються та можуть використовуватись повторно в інших проєктах. Проте реалізація мікросервісного підходу потребує значних витрат часу, високої технічної дисципліни та досвіду з налагодження взаємодії між сервісами. З огляду на обмежені часові ресурси та відносно невеликий масштаб поточного проєкту, було вирішено відкласти використання мікросервісів до майбутніх етапів розвитку системи, з можливістю їх поступового додавання як окремих функціональних блоків.

У результаті остаточним вибором стала клієнт-серверна архітектура з чітким поділом на фронтенд, бекенд і базу даних. Такий підхід забезпечив баланс між простотою реалізації та можливістю масштабування. Архітектура виявилася оптимальною для створення повноцінного прототипу з перспективою подальшого розвитку.

У реалізованій архітектурі фронтенд відповідає за взаємодію з користувачем, бекенд за обробку запитів, авторизацію, а також реалізацію логіки генерації навчальних матеріалів. База даних забезпечує збереження облікових записів

користувачів, колекцій навчальних карток, результатів тестувань та іншої необхідної інформації.

На концептуальній діаграмі представлено загальну схему роботи застосунку, де чітко розмежовані компоненти клієнта, сервера та зовнішніх сервісів. Зокрема, застосунок використовує стороннє API (наприклад, DeepSeek) для генерації навчального контенту, що дозволяє розширити функціональність системи без ускладнення внутрішньої структури.

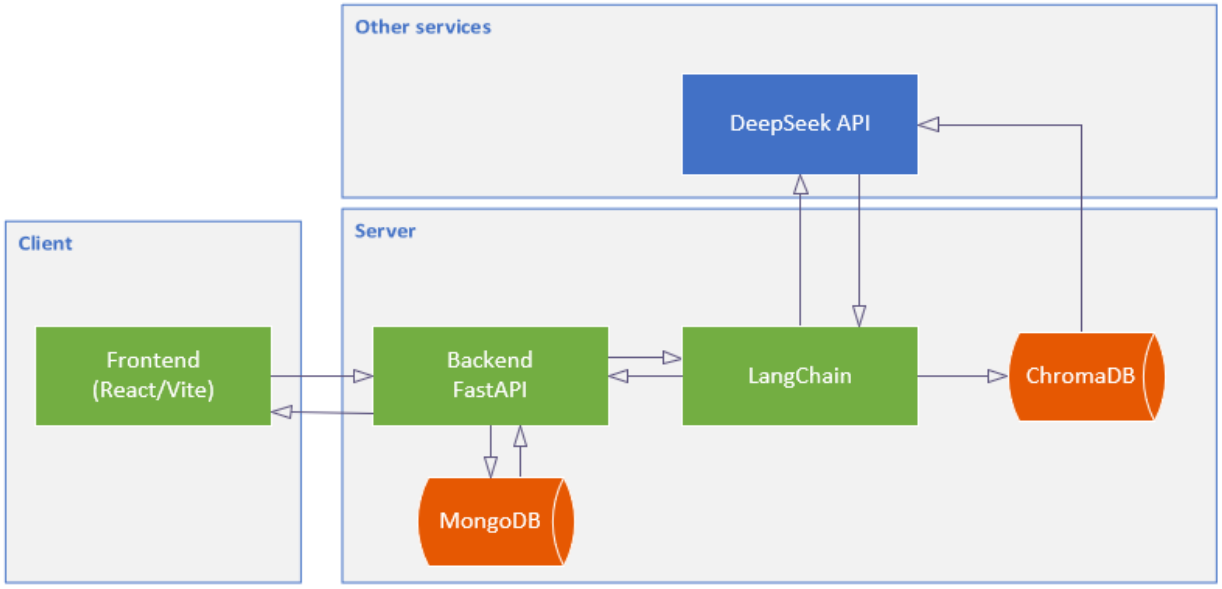


Рисунок 4.1- Архітектура системи

4.2 Реалізація бекенду (FastAPI)

Бекенд реалізовано з використанням фреймворку FastAPI, який забезпечує швидке створення продуктивних REST API з підтримкою OpenAPI та автоматичною генерацією документації. Усі маршрути згруповано за логічними модулями відповідно до функціонального призначення.

Основні групи та приклади маршрутів:

- Auth — забезпечує базову автентифікацію користувача:
    - POST /auth/signup — реєстрація користувача;
    - POST /auth/login — вхід у систему;
    - POST /auth/logout — вихід користувача.

- User — запити, пов’язані з даними облікового запису:
    - GET /user/me — отримання поточного користувача;
    - GET /user/user — інформація про користувача;
    - GET /user/is_author — перевірка ролі користувача.
- LLM — робота з генерацією матеріалів на основі LLM:
    - POST /generate_cards — генерація карток;
    - POST /generate_test — генерація тесту;
    - POST /generate_summary — генерація конспекту;
    - POST /save_smart_cards, /save_smart_test, /save_smart_summary — збереження відповідних результатів.
- RAG_LLM — генерація з використанням механізму RAG:
    - POST /knowledge_base — відповіді на основі бази знань;
    - POST /rag_flashcard, /rag_test, /rag_summary — генерація карток, тестів та конспектів з використанням Retrieval-Augmented Generation.
- Material — управління навчальними матеріалами:
    - GET /material/get_material — отримати всі матеріали;
    - GET /material/get_material/{material_id} — отримати матеріал за ID;
    - POST /material/add_material_to_favorites/{item_id} — додати до обраного;
    - GET /material/get_favourite_materials — отримати список улюблених матеріалів;
    - DELETE /material/delete/{item_id} — видалити матеріал.
- Statistic — збереження та аналіз статистики користувача:
    - /statistic_cards/* — збереження, оновлення, отримання та видалення статистики по картках;
    - /statistic_test/* — збереження, оновлення, отримання та видалення статистики по тестах.

Рисунок 4.2 – Автоматично згенерована документація

У проєкті використано асинхронну обробку запитів за допомогою конструкції async def, що забезпечує масштабованість та дозволяє ефективно обробляти велику кількість одночасних звернень.

Для валідації даних застосовується бібліотека Pydantic, яка гарантує строгий контроль типів і структури вхідної інформації. Авторизація користувачів реалізована через JWT (JSON Web Token), що дозволяє захищати маршрути та зберігати інформацію про сесії без використання сторонніх сервісів.

Робота з MongoDB здійснюється за допомогою Motor — асинхронного драйвера, який забезпечує швидку та ефективну взаємодію з базою даних у межах асинхронного середовища FastAPI.

4.3 Реалізація фронтенду (React)

Фронтенд реалізовано за допомогою React із використанням TailwindCSS для стилізації та бібліотеки компонентів DaisyUI, що забезпечує швидку побудову

інтерфейсу в єдиному стилі. Для маршрутизації між сторінками застосовано React Router.

Інтерфейс включає основні сторінки: головну з авторизацією, список колекцій, перегляд карток, інтерфейс генерації навчальних матеріалів та перегляд статистики користувача.

Застосовано компонентний підхід, де кожна частина інтерфейсу винесена в окремий React-компонент з власною логікою. Для обміну даними з API використовується axios. Управління станами реалізовано за допомогою useState та useEffect. Токени авторизації зберігаються у cookie, що забезпечує зручну та відносно безпечну передачу даних між клієнтом і сервером.

4.4 Робота з базою даних (MongoDB)

Для зберігання даних у проєкті було обрано MongoDB, оскільки вона є документоорієнтованою базою даних, що дозволяє зберігати дані у форматі JSON-подібних документів. Така структура дає високу гнучкість, що важливо для проєкту, де структура даних може змінюватися та доповнюватися з часом. MongoDB також забезпечує простоту масштабування і дозволяє зберігати великі обсяги даних без значних проблем з продуктивністю.

```
MONGO_URI = os.getenv("MONGO_URI")
DB_NAME = os.getenv("DB_NAME")

client = AsyncIOMotorClient(MONGO_URI)
db = client[DB_NAME]

users_collection = db["users"]
materials_collection = db["materials"]
favourite_materials_collection = db["favourite"]
statistic_collection = db["statistic"]
```

Рисунок 4.3 – Код підключення БД та створення колекцій

У результаті роботи застосунку формуються відповідні колекції в базі даних, які зберігають. За допомогою програми MongoDB Atlas ми можемо здійснювати взаємодію з базою даних: переглядати створені колекції, аналізувати їх вміст та виконувати необхідні операції з даними.

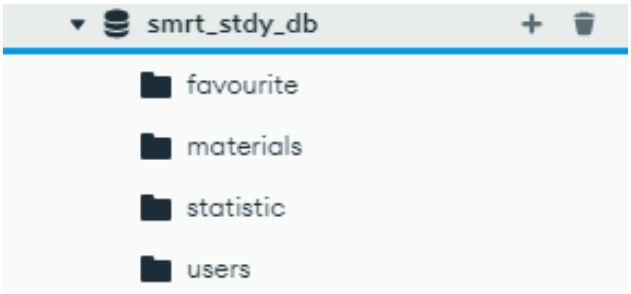


Рисунок 4.4 - Структура БД у MongoDB Atlas

Вибір колекцій базується на необхідності зберігання та швидкої обробки різних типів даних:

users - колекція з інформацією про користувачів, включаючи хешовані паролі. Кожен документ може містити індивідуальні дані про користувача та його уподобання, що дозволяє швидко отримувати потрібну інформацію без необхідності створювати складні зв’язки між таблицями.

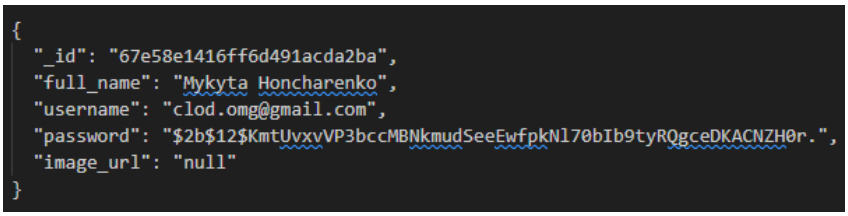


Рисунок 4.5 - Приклад даних users

favorites - колекція для зберігання улюблених матеріалів користувачів. У ній зберігаються посилання на документи в колекції materials, що дає можливість зручно отримувати матеріали, обрані користувачем, без необхідності дублювати інформацію.

```
{
  "_id": "680fd680e08150f35ce3248d",
  "user_id": "67e58e1416ff6d491acda2ba",
  "item_id": "680e6441b1259d1a3b5c17cd",
  "author_id": "680e1deddd93cf68e144599d"
}
```


Рисунок 4.6 - Приклад даних favorites

materials - колекція для зберігання навчальних матеріалів, таких як картки, тестові питання та статті. Завдяки гнучкості MongoDB, кожен документ може мати різну структуру, що дозволяє легко адаптувати колекцію до нових вимог.

Як приклад, розглянемо структуру класів, що формують модель даних для смарт-карток. Основою є головний клас Material, який успадковується від BaseModel бібліотеки Pydantic.

Pydantic це бібліотека для валідації даних та автоматичного створення моделей на основі Python типів. Клас BaseModel використовується як базовий клас, що надає можливість описувати структуру даних, здійснювати перевірку правильності типів полів даних.

Клас Material слугує узагальненою моделлю, що визначає базові поля, спільні для всіх типів навчальних матеріалів. Усі спеціалізовані класи, наприклад для карток, тестів чи текстових пояснень, наслідуються від Material і розширюють його додатковими специфічними полями. Це дозволяє дотримуватись принципів об’єктно-орієнтованого програмування, зокрема повторного використання коду та підтримки масштабованості.

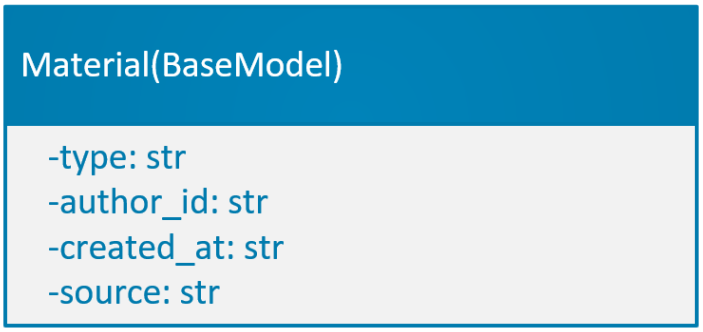


Рисунок 4.7 – Клас Material

На основі базового класу Material створюється основний клас навчального матеріалу зі специфічними полями та деталізованою інформацією. Цей клас уточнює загальну структуру, додаючи атрибути, характерні саме для смарт-карток, наприклад: назву теми, опис, автора, мітки та інші метадані.

Особливу увагу варто приділити останньому атрибуту цього класу — масиву, що містить вміст смарт карток. Цей масив є ключовим компонентом бо він формує навчальне наповнення картки. Кожен елемент у цьому масиві представлений як окремий об’єкт і може доповнюватися додатковою інформацією, наприклад, поясненнями чи тегами для подальшої класифікації.

Такий підхід дозволяє ефективно зберігати та обробляти навчальні дані, забезпечуючи зручну структуру для подальшого використання як на стороні сервера, так і у фронтенд частині застосунку.

Рисунок 4.8 – Клас Flashcard

І ось ми підходимо безпосередньо до класу, який відповідає за представлення смарт-карток. Ці картки також реалізуються у вигляді окремого класу, що наслідується від BaseModel бібліотеки Pydantic.

Як уже зазначалося, BaseModel це базовий клас, що використовується для створення строго типізованих моделей даних. Клас смарт-картки містить основні поля, зокрема запитання, відповідь. У базі даних ці об’єкти зберігаються у вигляді масиву всередині основного документа матеріалу. Таким чином, кожен навчальний матеріал містить набір смарт карток у структурованому форматі, що дозволяє легко працювати з ними як під час генерації, так і при подальшому використанні в інтерфейсі користувача.

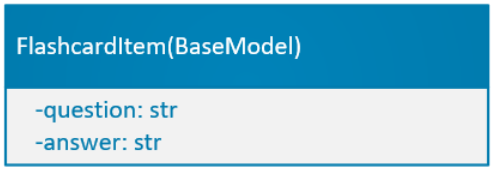


Рисунок 4.9 – Клас FlashcardItem

У результаті, під час збереження навчального матеріалу, всі відповідні класи об’єднуються в єдину структуровану модель даних. Ця модель формується відповідно до визначених схем і серіалізується у формат, придатний для зберігання в базі даних MongoDB.

Таким чином, у сховищі зберігається повний опис матеріалу разом із усіма вкладеними елементами, зокрема масивом смарт-карток. Під час перегляду в інтерфейсі MongoDB Atlas можна побачити цю структуру у вигляді вкладених об’єктів, що наочно відображає ієрархію та взаємозв’язок між складовими частинами навчального матеріалу. Це спрощує процес аналізу збережених даних, їх налагодження та підтримки.


```
{
  "_id": "680e62f6b1259d1a3b5c17cb",
  "type": "smart_cards",
  "author_id": "680e1deddd93cf68e144599d",
  "source": "general",
  "created_at": "2025-04-27 17:01:42.682878",
  "topic": "Управління ризиками в IT-проєктах",
  "num_cards": 5,
  "text_style": "formal",
  "detail_level": "medium",
  "answer_format": "description",
  "flashcards": [
    {
      "question": "Що таке управління ризиками в IT-проєктах?",
      "answer": "Управління ризиками в IT-проєктах – це систематичний …"
    }…
  ]
}
```


Рисунок 4.10 - Приклад даних materials

statistics - колекція для зберігання статистики тестувань, результатів і прогресу користувачів. Завдяки підтримці вкладених структур, MongoDB дозволяє зберігати інформацію про спроби тестувань та їхні оцінки в одному документі.

```
{
  "_id": {
    "$oid": "680fd6a8e08150f35ce3248e"
  },
  "user_id": "67e58e1416ff6d491acda2ba",
  "item_id": "680e6441b1259d1a3b5c17cd",
  "author_id": "680e1deddd93cf68e144599d",
  "known_cards": ["0", "1", "2", "3", "4"
  ],
  "need_to_remind": [],
  "unknown_cards": [],
  "completed": true
}
```


Рисунок 4.11 - Приклад даних statistics

Колекції були обрані відповідно до основних потреб проєкту. users зберігає інформацію про користувачів і їхні уподобання, що дозволяє легко взаємодіяти з даними користувача. favorites дає змогу зберігати персоналізовані вибори користувачів без дублювання інформації. materials є основною колекцією для зберігання навчальних матеріалів, а statistics дозволяє зберігати історію тестувань та прогрес користувачів у зручному вигляді.

4.5 Авторизація, безпека та обробка даних

Як я вже вище описував вибір принципу для аутентифікації користувачів, то хочу більш детально про це описати, що для аутентифікації реалізована система з використанням JWT (JSON Web Token). Паролі зберігаються в базі у хешованому вигляді завдяки bcrypt. При кожному запиті користувача перевіряється токен доступу, що гарантує захист персональних даних.
Також реалізована валідація вхідних даних через pydantic, що дозволяє уникнути ін'єкцій і забезпечити цілісність системи. Доступ до окремих ендпоїнтів обмежено на основі ролі користувача.

```python
@router.post('/login')
async def login(user_data: LoginUser, response: Response):
    try:
        # Пошук користувача в базі даних
        existing_user = await get_user(user_data.username)

        if existing_user and verify_password(user_data.password, existing_user.password):
            access_token = auth.create_access_token(uid=user_data.username)
            response.set_cookie(auth_config.JWT_ACCESS_COOKIE_NAME, access_token)
            return {"access_token": access_token}

        # Якщо користувач знайдений, але пароль неправильний
        raise HTTPException( status_code: 401, detail="Невірний логін або пароль")
```

Рисунок 4.12 – Приклад ендпоїнту входу у систему

4.6 Інтеграція з LLM через LangChain та RAG

Однією з ключових особливостей системи є інтеграція з LLM (мовною моделлю), що реалізована за допомогою бібліотеки LangChain. Механізм побудований на основі RAG (Retrieval-Augmented Generation) який допомагає розробити системи з гнучкою адаптацією до контексту матеріалу. Саме тому був обраний такий підхід а не донавчення моделі, бо таким чином можна постійно коригувати інформацію на основі якої йде генерація та додавати нову, що в контексті навчання грає дуже велику та важливу роль.

Давайте розглянемо принцип як в нас відбувається генерація на основі наданого контексту:

- Спочатку виконується індексація матеріалу Індексація знань (виконується заздалегідь). Навчальні матеріали дисципліни («Управління IT-проєктами») розбиваються на логічні фрагменти зазвичай по кілька речень. Кожен фрагмент перетворюється у вектор (числове представлення його змісту). Це робиться за допомогою спеціальної ембедінгової моделі(написати що це таке). Усі вектори зберігаються у векторному сховищі, яке дозволяє швидко здійснювати семантичний пошук. У моєму випадку це Chroma DB.

Після завершення етапу підготовки БД, система переходить до режиму взаємодії з користувачем. Коли користувач вводить запит, відбувається попередня обробка, яка включає семантичний пошук відповідного контенту в базі знань.

Семантичний пошук це метод пошуку, що враховує не лише ключові слова, а й сенс та змістовне значення введеного запиту. На відміну від класичного повнотекстового пошуку, семантичний аналіз використовує векторне представлення тексту та обчислює схожість між запитом і наявними текстовими фрагментами за допомогою спеціальних моделей. Це дозволяє знаходити найбільш релевантні та змістовно близькі частини контенту, навіть якщо вони сформульовані іншими словами.

Найбільш відповідні фрагменти, знайдені в результаті семантичного пошуку, передаються мовній моделі у вигляді контексту (додаткової інформації,) на основі якої модель виконує генерацію навчального матеріалу. Завдяки цьому модель працює з урахуванням конкретної тематики та надає точні, адаптовані до запиту результати.

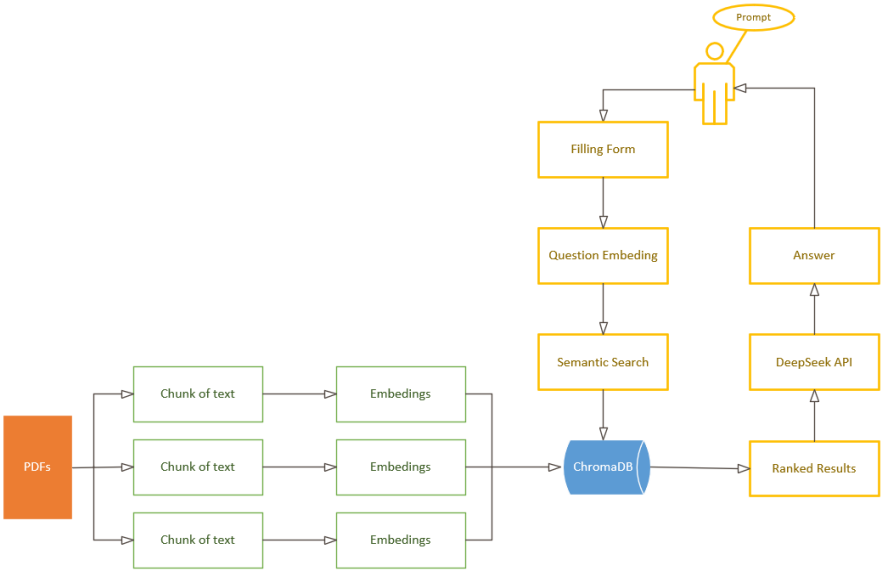


Рисунок 4.13 - Базовий приклад RAG алгоритму

Висновок до розділу 4

У четвертому розділі детально описано реалізацію основних компонентів системи. Розроблена розподілена архітектура, що забезпечує розширюваність і підтримуваність коду. Бекенд-логіка реалізує генерацію матеріалів, обробку запитів, збереження результатів, роботу з LangChain. Фронтенд надає адаптивний, інтуїтивний інтерфейс із вкладками: Головна, Пошук, Обране, База знань, Профіль. Забезпечено захищений обмін даними між клієнтом і сервером. Інтеграція RAG-механізму дозволяє давати точні відповіді на запити з урахуванням контексту матеріалів.

5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

5.1 Інтерфейс користувача

Однією з ключових задач проєкту було розробити інтерфейс користувача, оскільки саме через нього відбувається основна взаємодія користувача із застосунком, а також формується перше враження про платформу. Відповідно, інтерфейс повинен бути сучасним, унікальним та максимально зручним у використанні.

Головна сторінка платформи мала виконувати роль інформативного вітального екрану, який одразу ж дає зрозуміти користувачу, що це за платформа і які функції вона надає. Початковий дизайн був функціональним, але занадто мінімалістичним і простим, що створювало відчуття дешевизни і недостатньої професійності застосунку.

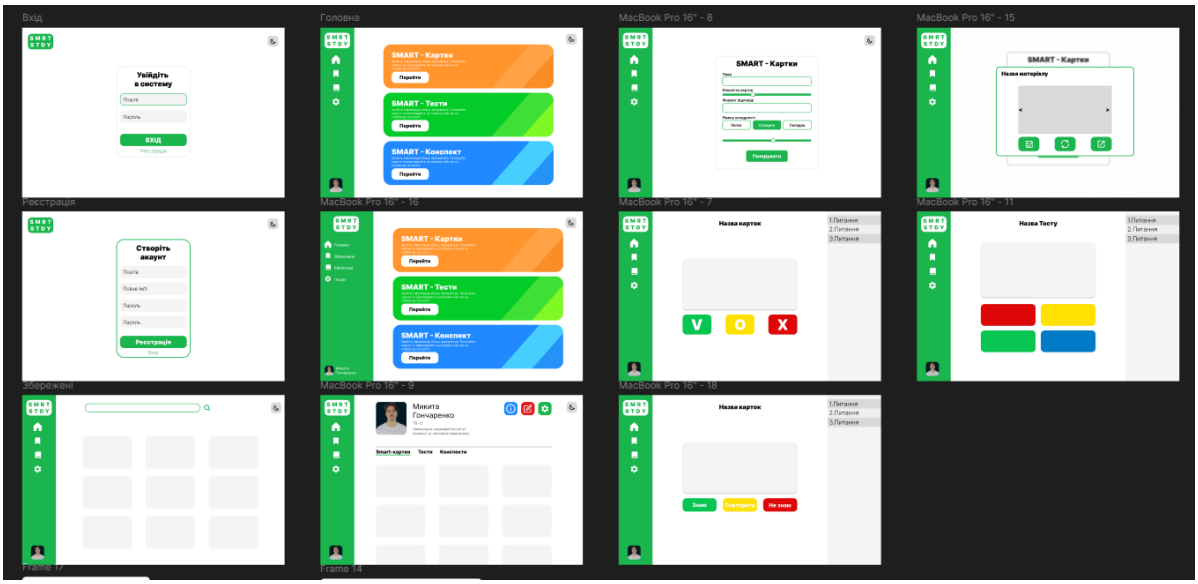


Рисунок 5.1 – Шаблони початкового дизайну застосунка, створено у Figma

З огляду на це я вирішив пошукати альтернативні дизайнерські рішеня, які б поєднували сучасність з оригінальністю, відходили від шаблонних і часто використовуваних у багатьох сайтах стилістичних підходів, і водночас забезпечували комфортну навігацію та інтуїтивно зрозумілий інтерфейс для користувачів.

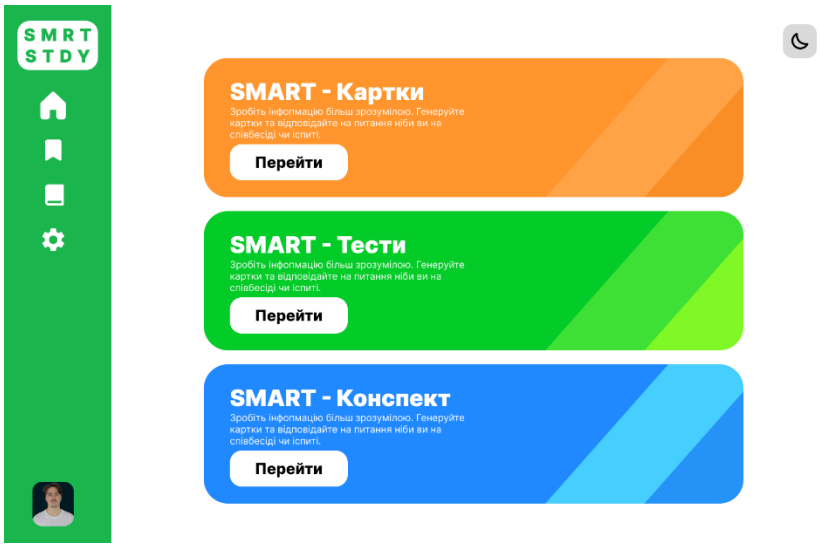


Рисунок 5.2 – Початковий дизайн головної сторінки

Таким чином, дизайн застосунку було практично повністю перероблено, оскільки навіть початкове меню з боковою навігацією не забезпечувало достатньої зручності при користуванні на мобільних пристроях.

Для нового оформлення я обрав стриману кольорову палітру з ніжними відтінками та градієнтами, які асоціюються з навчанням і не відволікають користувача від основного процесу опанування матеріалу.

Головна сторінка стала значно інформативнішою, містить різноманітні корисні елементи, що допомагають орієнтуватися у функціоналі платформи та максимально спрощують доступ до потрібної інформації.

Інтерфейс застосунку реалізовано у сучасному стилі з акцентом на зручність, доступність і зрозумілість. Головне меню містить такі навігаційні вкладки: «Головна», «Пошук», «Обране», «База знань», «Профіль користувача». Усі сторінки адаптовані під різні розміри екранів, що забезпечує комфортне використання як на мобільних пристроях, так і на персональних комп’ютерах.

На головній сторінці користувач бачить кнопку «Створити новий матеріал», рекомендовані картки/тести для проходження, а також статистику нещодавньої активності.

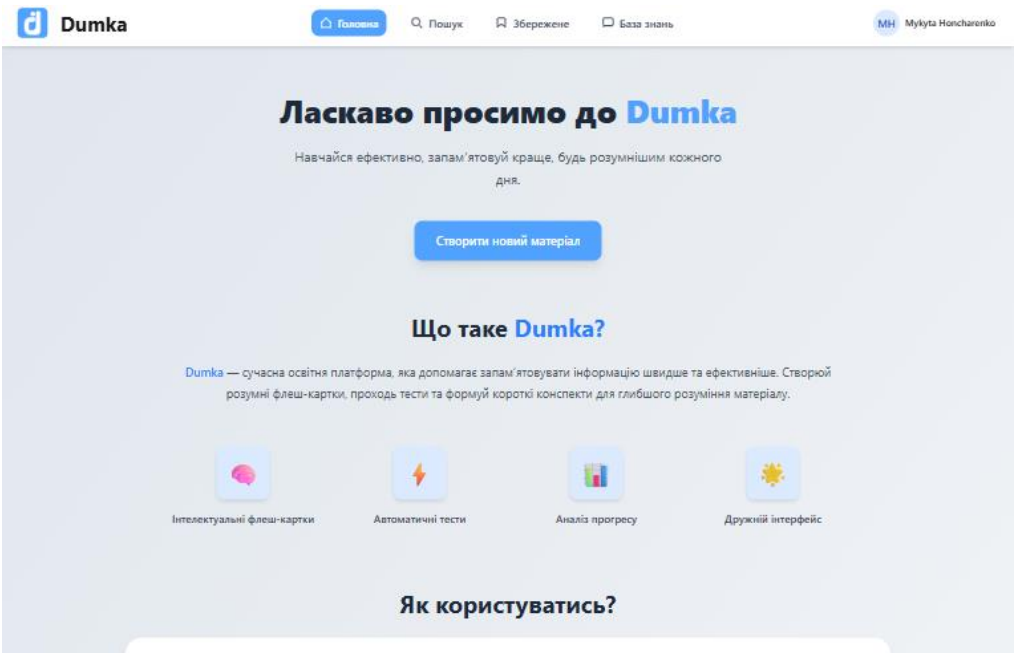


Рисунок 5.3 - Головна сторінка застосунку із кнопкою створення нового матеріалу та панеллю навігації

Сторінка пошуку реалізована спеціально для того щоб модна було не самотужки генерувати матеріал, а знайти матеріал який вже створив інший користувач. Переглянути зміст сторінки можна на рис 5.2. Крім того пошук містить в собі декілька фільтрацій для більш точного знаходження потрібного матеріалу.

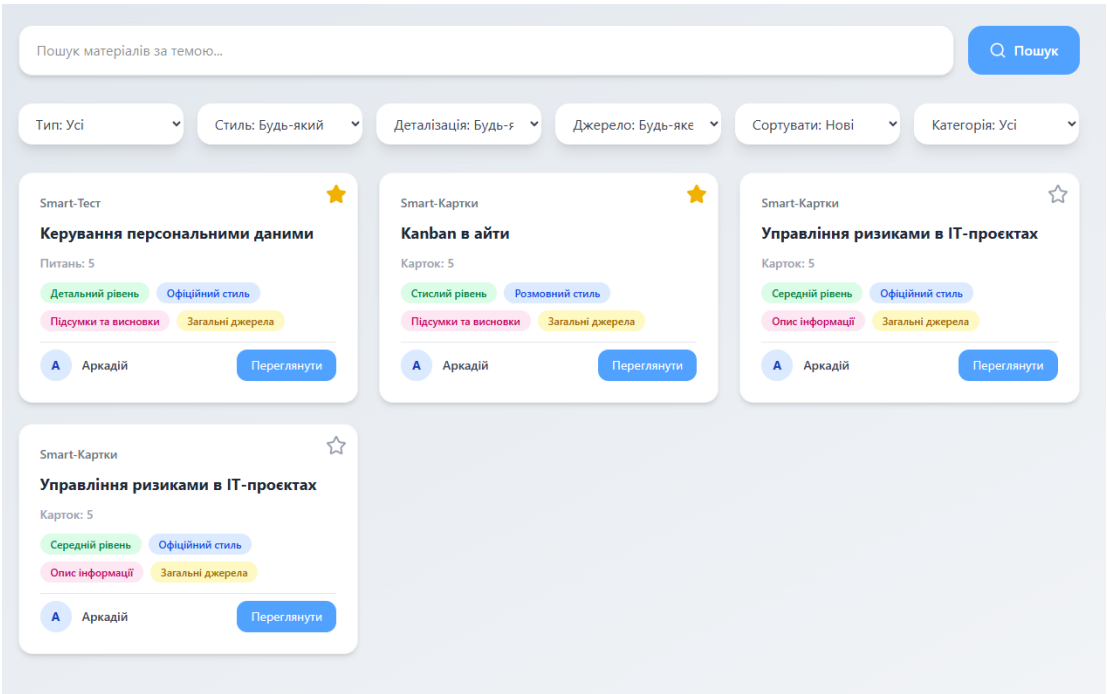


Рисунок 5.4 - Сторінка пошуку матеріалів з фільтрами

5.2 Функціонал застосунку

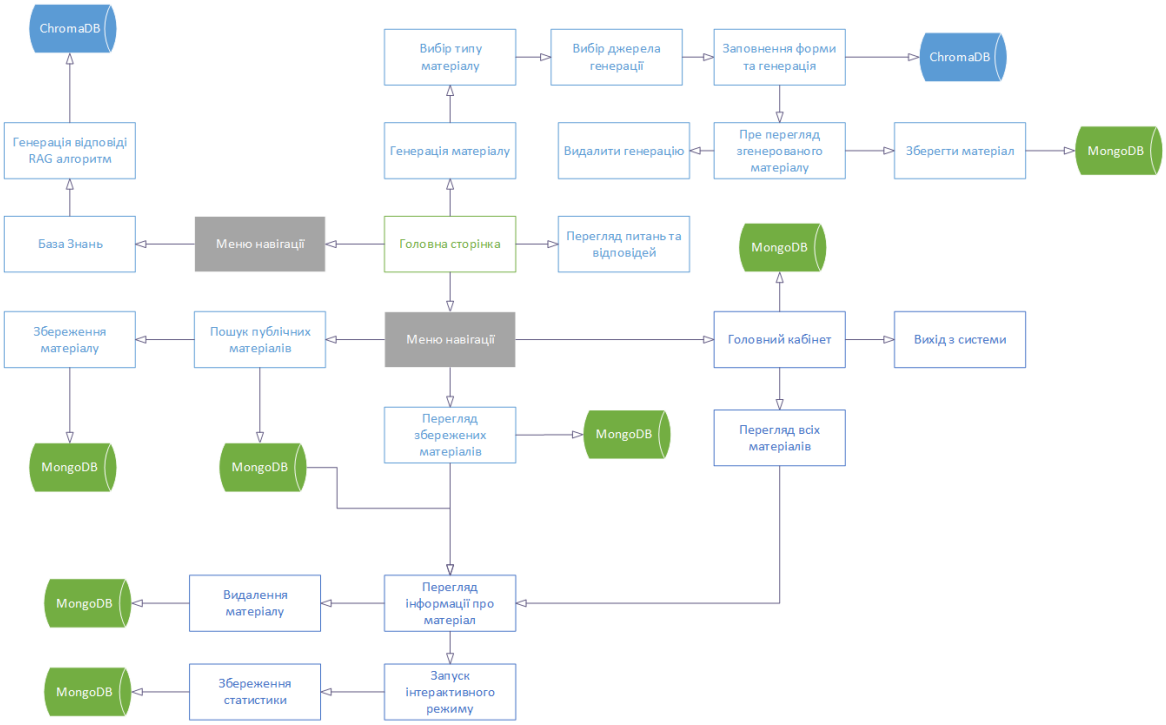


Рисунок 5.5 - Структура застосунку

У задачі стояло на меті створити максимально багатофункціональний навчальний застосунок, який одразу міг би бути готовим прототипом для подальшої доробки у рамках магістерської роботи.

Процес створення нового навчального матеріалу починається на головній сторінці шляхом натискання кнопки «Створити новий матеріал». Після цього користувач обирає тип матеріалу, наприклад, смарт-картки або тест, що дозволяє адаптувати подальші дії під конкретні вимоги й формат навчального контенту.

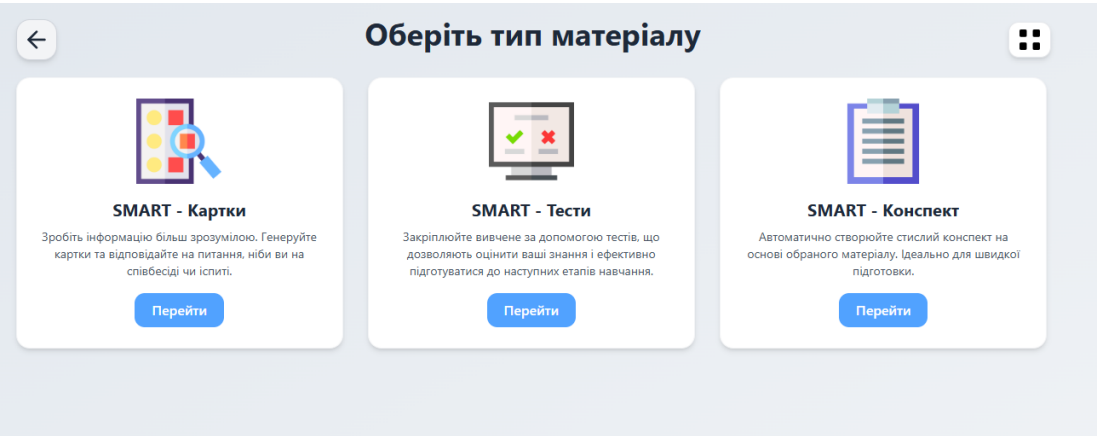
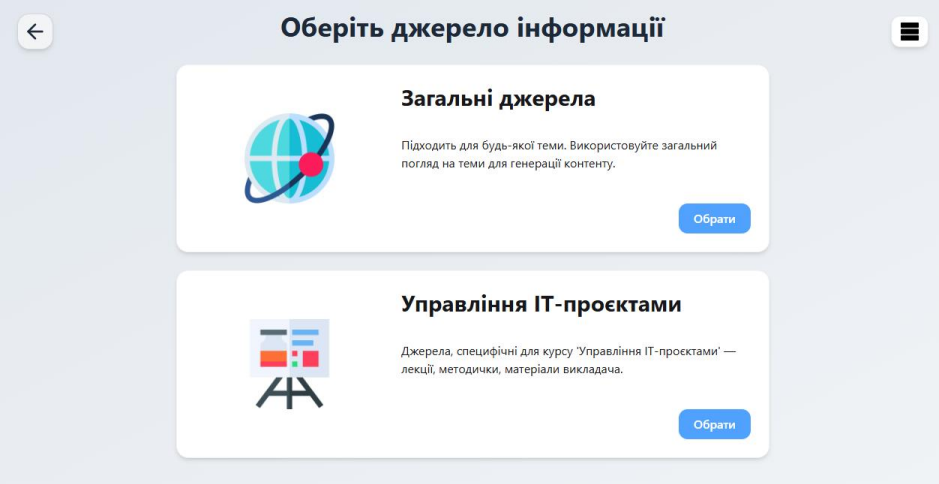


Рисунок 5.6 - Сторінка вибору типу інтерактивного матеріалу

На наступному кроці користувач має обрати джерело інформації для генерації:

- Використати загальні дані мовної моделі (в моєму випадку DeepSeek)
- Скористатися матеріалами з векторного сховища з дисципліни «Управління IT-проєктами»



Зображення 5.7 - Сторінка вибору джерела генерації

Після цього відкривається форма з полями, де вводяться параметри генерації (наприклад, кількість питань, мова тощо). Після заповнення форми запускається генерація, яка займає декілька секунд або хвилин, залежно від складності запиту.

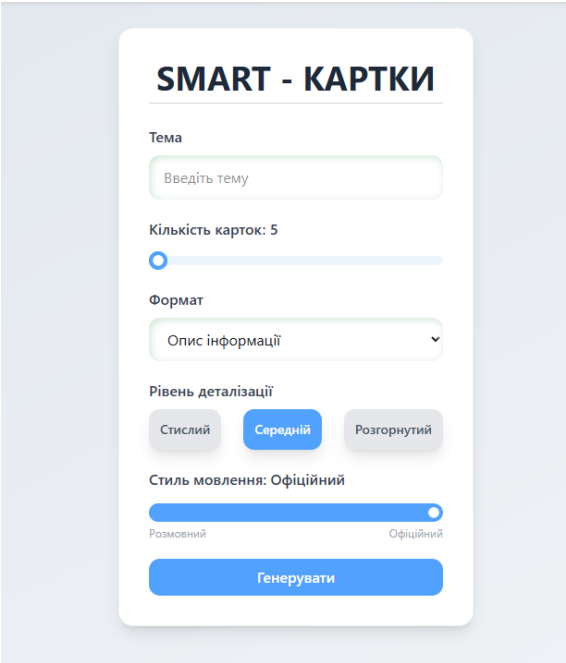


Рисунок 5.8 - Типова форма для створення інтерактивного матеріалу

Після завершення генерації матеріал можна переглянути та зберегти до свого профілю чи видалити.

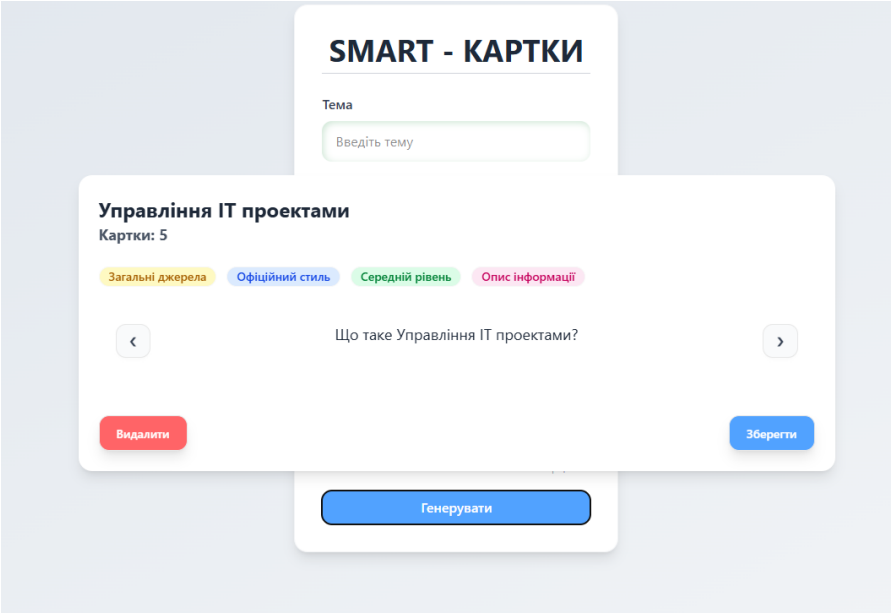


Рисунок 5.9 – Пре перегляд згенерованого матеріалу

У профілі користувача матеріал буде доступний для перегляду, редагування або проходження. У режимі проходження (тест або картки) користувач має змогу переглядати свій прогрес, а також оновлювати статистику засвоєння матеріалу.

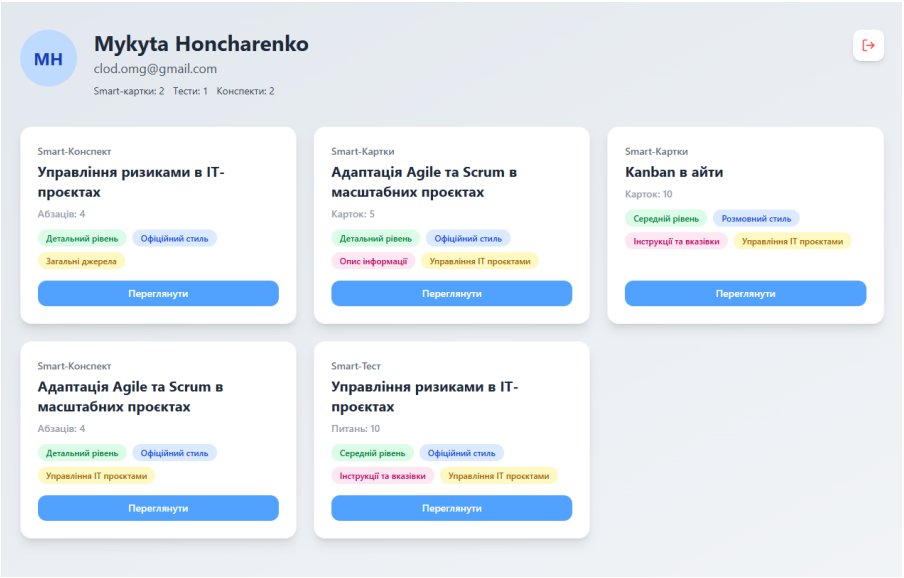


Рисунок 5.10 - Особистий кабінет із відображенням збережених матеріалів та графіком прогресу користувача

Кожен створений або пройдений матеріал фіксується у системі. Користувач може переглядати, наскільки він засвоїв матеріал, рівень правильних відповідей, динаміку засвоєння тощо. Ця статистика доступна на сторінці перегляду матеріалу.

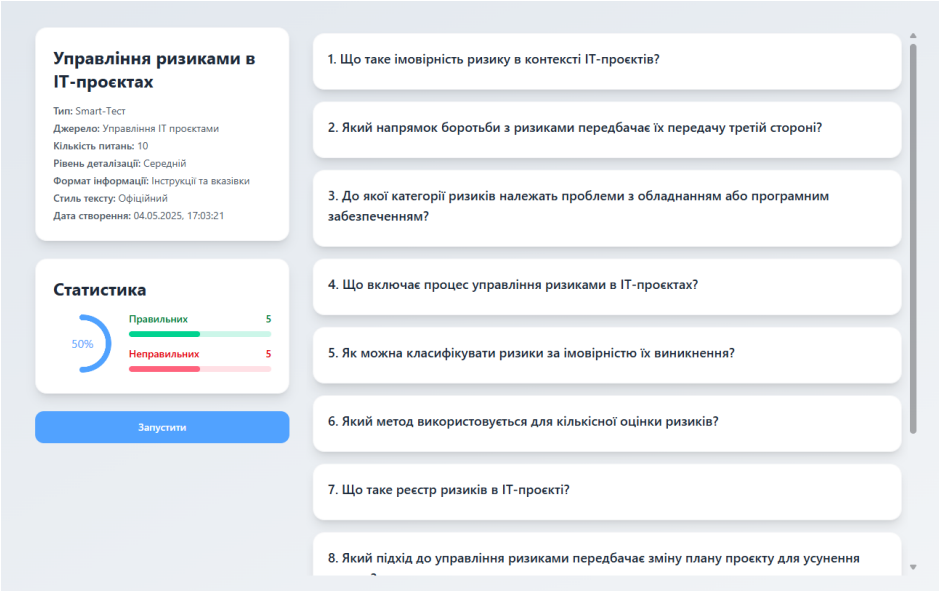


Рисунок 5.11 - Сторінка матеріалу із відображенням збережених матеріалів та графіком прогресу користувача

Також певні навчальні матеріали мають інтерактивний режим, що є ключовою ідеєю створення платформи це поєднати автоматичну генерацію контенту на основі власних даних користувача з інтерактивністю через зручний інтерфейс.

Для смарт карток передбачено три категорії стану: «Знаю», «Не знаю» та «Повторити», які допомагають користувачу організувати процес самоперевірки та закріплення вивченого матеріалу.

Інтерфейс містить бокову панель із переліком карток та статистикою, що відображає прогрес користувача. При перегляді картка має дві сторони: питання та відповідь. Такий формат дозволяє студенту ефективно та швидко запам’ятовувати матеріал за допомогою активного повторення.

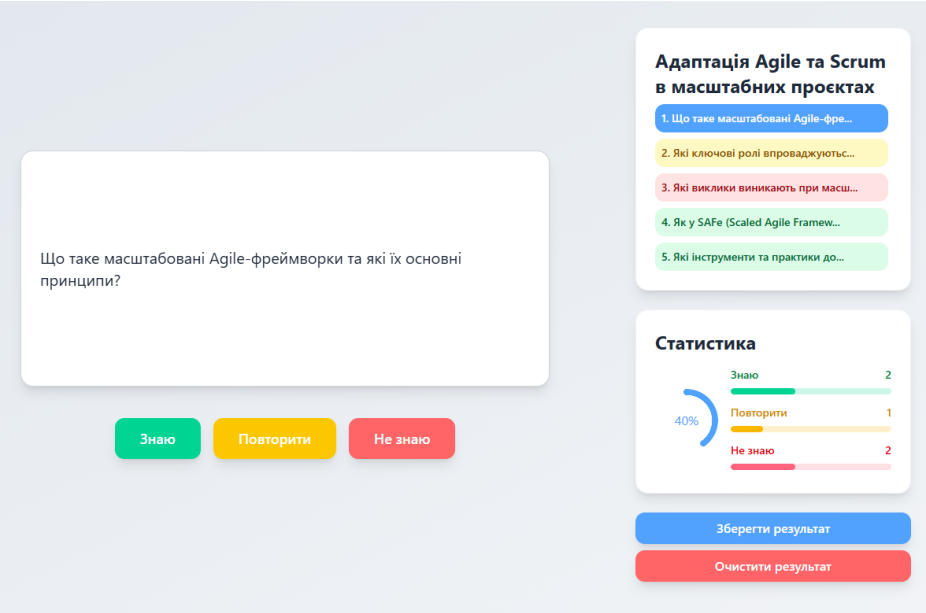


Рисунок 5.12 - Сторінка інтерактивного режиму для смарт карток

Інтерактивний режим також реалізовано для смарт-тестів. Інтерфейс у цьому випадку подібний до режиму смарт-карток, однак адаптований під особливості тестування.

Зліва розташоване активне питання з чотирма варіантами відповіді та кнопкою «Відповісти» для підтвердження вибору. Праворуч знаходиться список усіх питань, статистика проходження тесту, а також кнопки для збереження або видалення накопиченої статистики.

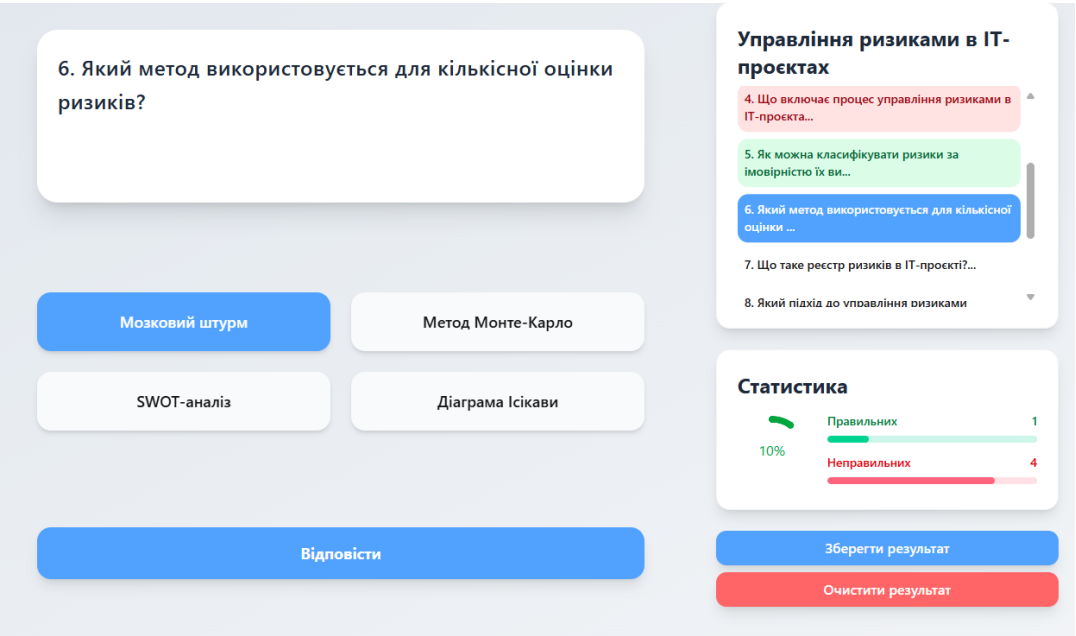


Рисунок 5.13 - Сторінка інтерактивного режиму для смарт тестів

Ще однією цікавою особливістю застосунку є чат-бот під назвою «База знань», який виконує роль віртуального викладача, до якого користувач може звертатися з питаннями.

На сторінці «База знань» користувач має змогу ввести текстове запитання, яке обробляється великою мовною моделлю через LangChain із доступом до векторного сховища, побудованого на основі навчальних матеріалів з дисципліни «Управління IT-проєктами».

Модель генерує релевантну відповідь у вигляді текстового пояснення, доповненого за потреби посиланнями на джерела або рекомендаціями для поглибленого вивчення теми.

Рисунок 5.14 - Інтерфейс сторінки «База знань» з полем для запиту та згенерованою відповіддю

Висновки до розділу 5

Розділ описує взаємодію користувача з інтерфейсом системи. Проведено редизайн із фокусом на зручність і мінімалізм. Обрані м’які кольори, які не відволікають від навчання. Застосунок адаптований до мобільних пристроїв і ПК. Основний функціонал реалізовано через розділи: створення матеріалів, перегляд і проходження карток та тестів, перегляд статистики, використання чат-бота. Особливою функцією є інтерактивність — можливість самоперевірки через режим

«Знаю / Не знаю / Повторити» та інтерактивні тести з можливістю збереження прогресу. Також реалізовано чат-бот як інтелектуального помічника на основі векторного сховища знань, що робить платформу потужним засобом навчання.

ВИСНОВКИ

Під час розробки дипломного проекту була реалізована повнофункціональна веб-платформа для генерації та проходження навчальних матеріалів із використанням технологій штучного інтелекту.

Архітектура системи була розроблена з акцентом на масштабованість і модульність, що дозволяє ефективно працювати з великими обсягами даних. Бекенд був створений на базі FastAPI, що забезпечує швидку обробку запитів і інтеграцію з базою даних та мовною моделлю. Для клієнтської частини був обраний React, що дозволяє створити зручний, адаптивний інтерфейс.

Інтеграція LangChain і підхід Retrieval-Augmented Generation (RAG) дали змогу створювати персоналізовані навчальні матеріали на основі файлів користувача або предметної бази знань. Дані зберігаються в MongoDB, з можливістю авторизації, трекінгу прогресу користувача та управління навчальними матеріалами. Платформа підтримує генерацію карток і тестів, перегляд статистики, а також можливість пошуку відповідей у базі знань.

У майбутньому розвиток платформи буде спрямований на розширення її функціональності та покращення користувацького досвіду. Планується додавання нових видів інтерактивних навчальних матеріалів. Це дозволить зробити процес навчання більш цікавим і захоплюючим для користувачів.

Також буде розширено кількість доступних дисциплін, що дасть можливість користувачам готуватися до більш широкого спектру предметів.

Для покращення аналізу прогресу буде розроблена вдосконалена статистична система, яка дозволить точніше відстежувати динаміку навчання, визначати слабкі місця та ефективність проходження матеріалів.

Ці покращення зроблять платформу ще більш гнучкою, масштабованою та адаптованою до різних категорій користувачів, включаючи як студентів, так і викладачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tiangolo S. FastAPI: The modern, fast (high-performance) web framework for building APIs with Python 3.7+. FastAPI. URL: https://fastapi.tiangolo.com/ (дата звернення: 15.04.2025).

2. React – A JavaScript library for building user interfaces. React Documentation. URL: https://reactjs.org/docs/getting-started.html (дата звернення: 25.04.2025).

3. MongoDB – The Developer Data Platform. MongoDB Documentation. URL: https://www.mongodb.com/docs/ (дата звернення: 16.05.2025).

4. LangChain – Building language models with LLMs. LangChain Documentation. URL: https://langchain.com/docs/ (дата звернення: 15.04.2025).

5. Motor Documentation. URL: https://motor.readthedocs.io/en/latest/ (дата звернення: 16.04.2025).

6. DaisyUI Documentation. URL: https://daisyui.com/ (дата звернення: 01.05.2025).

7. Axios Documentation. URL: https://axios-http.com/docs/intro (дата звернення: 01.05.2025).

8. bcrypt Documentation. URL: https://pypi.org/project/bcrypt/ (дата звернення: 20.04.2025).

9. DeepSeek Documentation. URL: https://www.deepseek.com/en (дата звернення: 15.04.2025).

ДОДАТОК А Лістинг розробленої системи

```
from authx.exceptions import MissingTokenError
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from starlette.responses import JSONResponse
from src.routers.statistic_routers.statistics_cards import router as statistic_cards_router
from src.routers.statistic_routers.statistics_test import router as statistic_test_router
from src.routers.model_router import model_router
from src.routers.rag_model_router import router as rag_model_router
from src.routers.auth_router import router as auth_router
from src.routers.user_router import router as user_router
from src.routers.material_router import router as material_router
from src.routers.data_upload_router import router as data_upload_router

import httpx
app = FastAPI()

origins = [
    "http://localhost",
    "http://localhost:5173",
    "http://127.0.0.1:5173",
"http://192.168.0.51:5173",
"http://localhost:3000",
    "http://frontend:80"

]

app.add_middleware(
    CORSMiddleware,
```

```
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

@app.exception_handler(MissingTokenError)
async def missing_token_exception_handler(request, exc):
    return JSONResponse(
        status_code=401,
        content={"detail": "Missing authentication token. Please log in."},
    )

app.include_router(auth_router)
app.include_router(user_router)
app.include_router(model_router)
app.include_router(rag_model_router)
app.include_router(material_router)
app.include_router(statistic_cards_router)
app.include_router(statistic_test_router)
app.include_router(data_upload_router)



from langchain_core.output_parsers import JsonOutputParser
from src.llm_pipeline.llm_model import Model
from langchain_community.vectorstores import Chroma
```

```
# Ініціалізація моделі
model = Model()
llm = model.model_deep_seek

CHROMA_PATH = "src/llm_pipeline/chroma_db"
COLLECTION_NAME = "it_collection"

retriever = Chroma(
    collection_name=COLLECTION_NAME,
    embedding_function=model.embeddings_model,
    persist_directory=CHROMA_PATH
).as_retriever(search_kwargs={"k": 10})



async def rag_generate(query, pars_model, prompt_template):
    parser = JsonOutputParser(pydantic_object=pars_model)

    try:
        # 1. Витягуємо релевантні документи через новий API
        docs = await retriever.ainvoke(query)
        context = "\n\n".join([doc.page_content for doc in docs])

        # 2. Створюємо фінальний вхід
        input_vars = {
            "input": f"Контекст:\n{context}\n\nЗапитання:\n{query}"
        }

        # 3. Створюємо ланцюжок з prompt + LLM + parser
        chain = prompt_template | llm | parser
```

```
        # 4. Викликаємо chain
        result = await chain.ainvoke(input_vars)
        return result

    except Exception as e:
        print("Помилка:", e)
        return None
```

```
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import axios from "axios";
import { motion } from "framer-motion";
import { ToastContainer, toast } from "react-toastify";

import "react-toastify/dist/ReactToastify.css"; // Імпортуємо стилі для Toast

function LoginPage() {
  const navigate = useNavigate();

  const handleClick = () => {
    navigate('/signup'); // Змініть шлях на потрібний
  };

  const [users, setUsers] = useState([]);
  const [user, setUser] = useState({
    username: "",
    password: "",
```

```
  });

  const handleUserChange = (e) => {
    const { name, value } = e.target;
    setUser((prevUser) => ({
      ...prevUser,
      [name]: value,
    }));
  };

  const sendUserData = () => {
    if (!user.username.trim()) {
      console.error("Ім'я користувача не може бути порожнім");
      toast.error("Ім'я користувача не може бути порожнім"); // Тост для порожнього
імені
      return;
    }

    axios
      .post("http://localhost:8000/auth/login", user, {
        withCredentials: true,
      })
      .then((res) => {
        console.log("Відповідь сервера:", res.data);
        setUsers(res.data.Users || []); // Безопасная проверка
        setUser({ username: "", password: "" }); // Очищаем поля
        navigate("/", { replace: true });
        window.history.replaceState(null, "", window.location.href);
        toast.success("Успішний вхід!"); // Успішний вхід
```

```
    })
    .catch((err) => {
      console.error("Помилка:", err.response?.data || err.message);
      // Тости для різних помилок
      if (err.response) {
        const errorMessage = err.response.data.detail || "Щось пішло не так.";
        if (err.response.status === 401) {
          toast.error("Невірний логін або пароль.");
        } else if (err.response.status === 404) {
          toast.error("Користувач не знайдений.");
        } else if (err.response.status === 503) {
          toast.error("Проблеми з підключенням до бази даних. Спробуйте пізніше.");
        } else {
          toast.error(errorMessage);
        }
      } else {
        toast.error("Невідома помилка. Перевірте підключення до Інтернету.");
      }
    });
};

return (
  <motion.div
    initial={{ opacity: 0, y: 20 }}
    animate={{ opacity: 1, y: 0 }}
    exit={{ opacity: 0, y: -20 }}
    transition={{ duration: 0.5 }}
    className="min-h-screen w-full flex flex-col text-gray-800 font-sans relative"
  >
```

```
      {/* Логотип зверху зліва */}
<div className="flex items-center absolute top-4 left-4 z-10">
  <button
    onClick={() => navigate('/')}
    className="text-3xl font-bold flex items-center sm:inline text-gray-800 hover:text-blue-500 transition-colors cursor-pointer"
  >
    <img
      src="/dumka_logo_new.svg"
      alt="Description"
      className="w-20 h-20 rounded-full mr-2"  // Відступ праворуч між зображенням і текстом
    />
  </button>
</div>

      {/* Центрована форма */}
      <div className="flex flex-1 items-center justify-center px-4 py-8">
        <div className="w-full max-w-md rounded-3xl p-10 bg-base-100 shadow-lg">
          <h1 className="text-3xl font-bold mb-8 text-center text-blue-700">Увійдіть в систему</h1>

          <form className="space-y-6">
            <div>
              <label className="block mb-2 font-medium text-blue-700">Email</label>
              <input
                type="text"
                name="username"
                value={user.username}
```

```
            onChange={handleUserChange}
            required
            placeholder="Email"
            className="w-full px-4 py-3 rounded-xl bg-base-100 shadow-
[inset_2px_2px_8px_#cfe7d6,inset_-2px_-2px_8px_#ffffff] focus:outline-none"
          />
        </div>

        <div>
          <label className="block mb-2 font-medium text-blue-700">Пароль</label>
          <input
            type="password"
            name="password"
            value={user.password}
            onChange={handleUserChange}
            required
            placeholder="Пароль"
            className="w-full px-4 py-3 rounded-xl bg-base-100 shadow-
[inset_2px_2px_8px_#cfe7d6,inset_-2px_-2px_8px_#ffffff] focus:outline-none"
          />
        </div>

        <div className="space-y-4 pt-4">
          <button
            type="button"
            onClick={sendUserData}
            className="w-full py-3 rounded-xl font-semibold text-white bg-blue-400
shadow-lg hover:brightness-105 transition-all duration-200 cursor-pointer"
          >
```

```
            Увійти
          </button>

          <button
            type="button"
            onClick={handleClick}
            className="w-full py-3 rounded-xl font-semibold text-blue-700 bg-base-100 shadow-[inset_2px_2px_8px_#cfe7d6,inset_-2px_-2px_8px_#ffffff] hover:shadow-[2px_2px_12px_#b0d4bb,-2px_-2px_12px_#ffffff] hover:bg-[#f0fef4] transition-all duration-100 cursor-pointer"
          >
            Реєстрація
          </button>
        </div>
      </form>
    </div>
  </div>

  {/* Додаємо контейнер для тостів */}
  <ToastContainer />
 </motion.div>
 );
}

export default LoginPage;
```

ДОДАТОК Б Презентація

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

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

Вебплатформа для створення навчальних матеріалів на основі LLM

Гончаренко Микита, ТВ-12
Кузьміних Валерій Олександрович, Доцент, Кандидат наук

2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Актуальність теми

Сучасні освітні платформи та цифрові інструменти значно спрощують навчальний процес завдяки інтерактивному підходу, що активізує взаємодію з матеріалом і покращує засвоєння знань. Однак, більшості платформ бракує можливості створювати власні інтерактивні компоненти, що знижує їхню адаптованість до потреб навчальних закладів. Мовні моделі ШІ здатні генерувати навчальний контент, але рідко підтримують інтерактивність. Поєднання цих переваг у єдиній платформі сприяло б глибшому засвоєнню знань та підвищенню ефективності освіти.

Постановка задачі

Розробка веб-платформи, яка реалізує підхід до генерації та оцінки навчальних матеріалів (флеш карток, тестів та ін.) на основі Large Language Models (LLM), із подальшим тестуванням її ефективності у реальному середовищі.

Етапи реалізації:

1. Аналіз існуючих LLM та вибір оптимальної моделі для генерації навчального контенту.
2. Розробка веб-системи для створення, збереження та керування навчальними матеріалами.
3. Підготовка тестового середовища для перевірки функціональності платформи.
4. Проведення функціонального та користувацького тестування системи.
5. Аналіз якості згенерованих матеріалів та ефективності використаної моделі.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Аналіз предметної області

- Обрано курс “Управління IT-проєктами”
- Вивчається на більшості IT-спеціальностей
- Містить теоретичні та практичні матеріали
- Дає змогу протестувати генерацію термінів, понять, кейсів

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Стек технологій

FastAPI

Chroma

deepseek

daisyUI

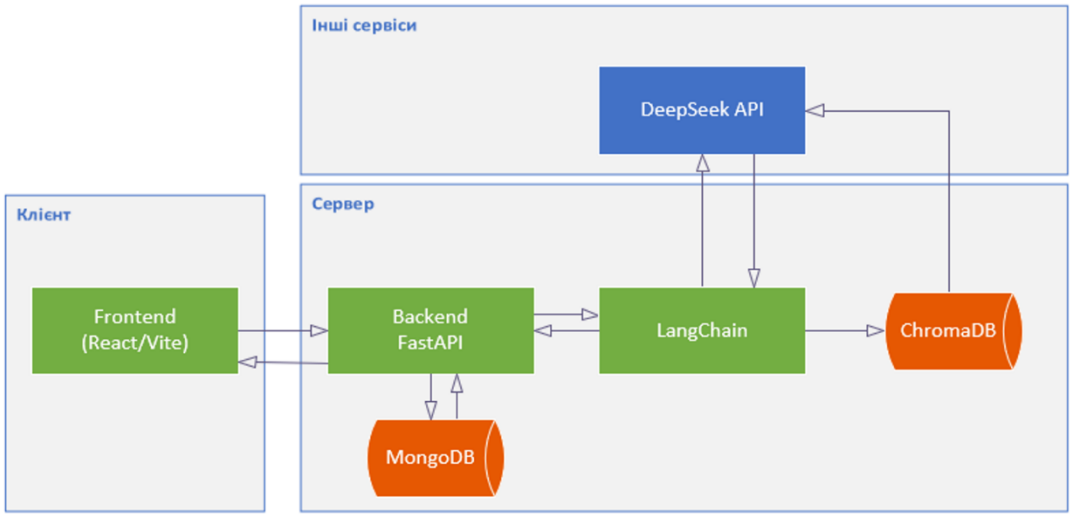
Vite + React

LangChain

mongoDB.

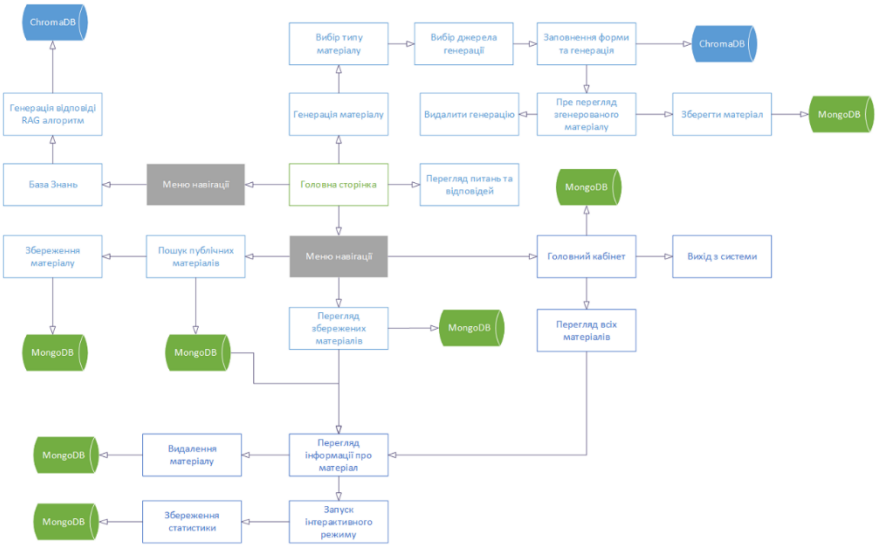
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Архітектура застосунку



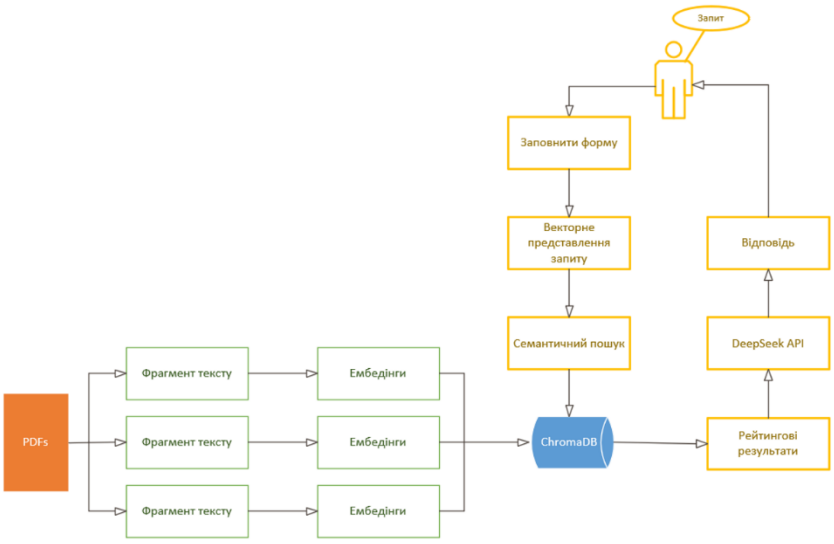
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Структура застосунку



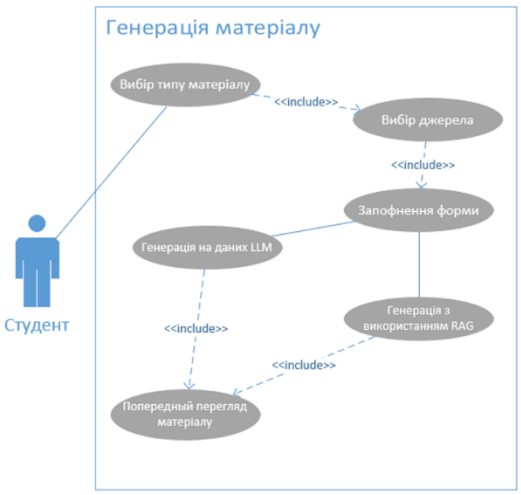
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

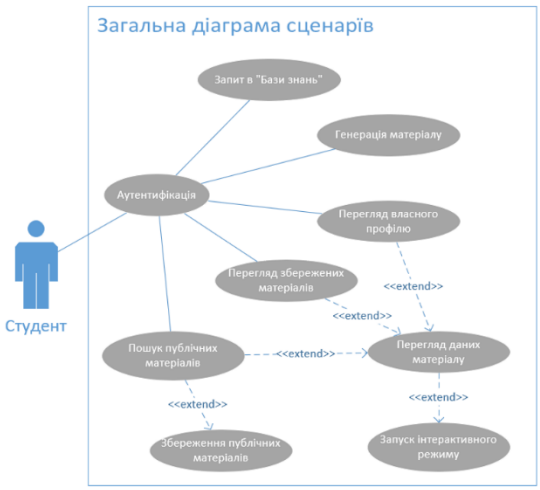
Алгоритм генерації



НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

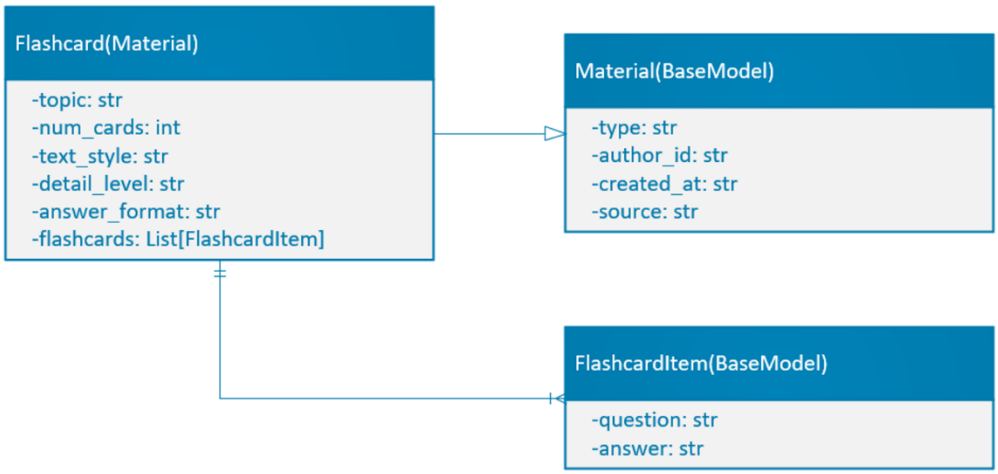
USE-Case діаграми

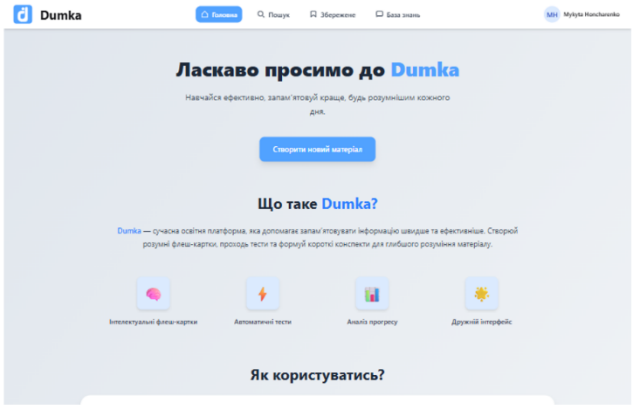




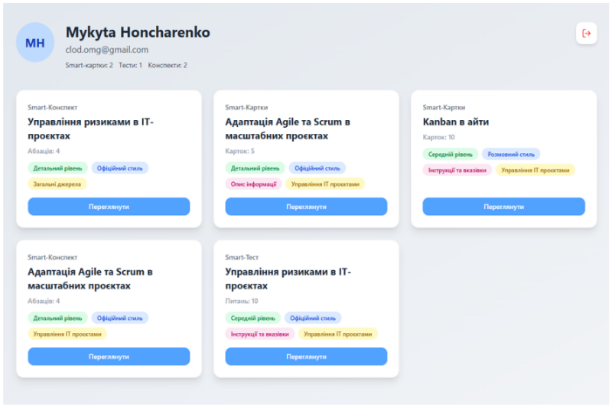
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Діаграми класів

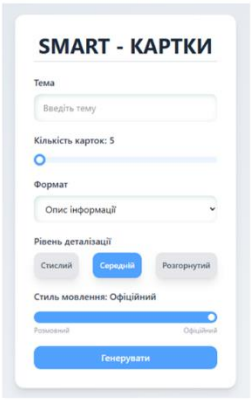




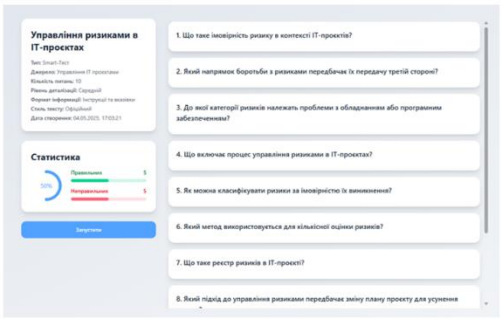
Головна сторінка застосунку



Особистий кабінет



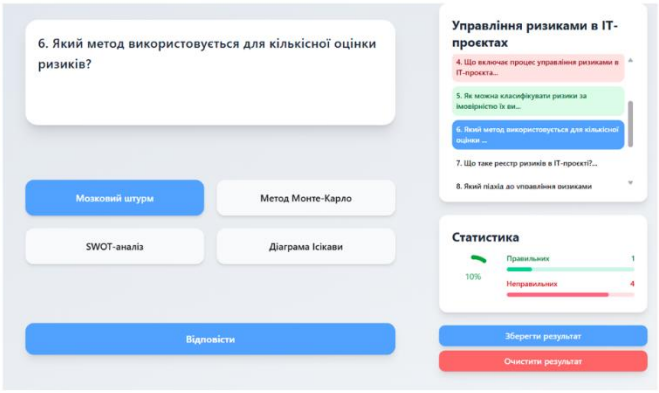
Форма для генерації Smart-карток



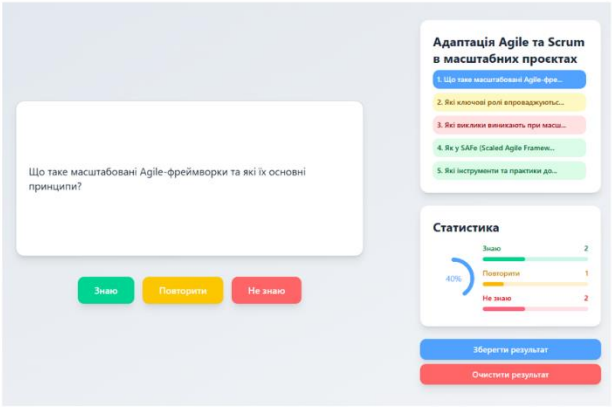
Сторінка перегляду матеріалів

Сторінка «База знань»

Інтерфейс



Інтерактивний режим для тестів



Інтерактивний режим для карток

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Висновок

- Розроблено веб-платформу для генерації навчальних матеріалів
- Підтверджено ефективність генерації навчального контенту з LLM
- Проєкт має потенціал для розвитку та масштабування в освітньому середовищі