

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

ІНТЕЛЕКТУАЛЬНОГО ЧАТУ ДЛЯ ОСВІТНІХ

ЦІЛЕЙ

SOFTWARE IMPLEMENTATION OF AN

INTELLIGENT EDUCATIONAL CHATBOT SYSTEM

Виконав(ла): студент(ка) 4 курсу, групи КНТ-112

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

СТЕПАНОВ І.А.

(ПРИЗВИЩЕ та ініціали)

Керівник ГОФМАН Є.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ДЬЯЧУК Т.С.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н., проф.

Сергій

СУББОТІН

“ ” 2026 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

СТЕПАНОВА Ігоря Андрійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація застосунку інтелектуального чату для освітніх цілей.

Software Implementation of an Intelligent Educational Chatbot System

керівник проєкту (роботи) к.т.н., доцент, ГОФМАН Євгеній Олександрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 07 ” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 10 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Розробка програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГОФМАН Є. О., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 20 ” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка архітектури програми.	4 тиждень	Розділ 2
4	Розробка програми.	5 тиждень	Розділ 3
5	Тестування та експериментальне дослідження програми.	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування.	7 тиждень	Додатки
7	Захист роботи.	8 тиждень	

Студент(ка)

(підпис)

Ігор СТЕПАНОВ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Євгеній ГОФМАН
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 98 с., 10 табл., 42 рис., 3 дод., 21 джерело.

ІНТЕЛЕКТУАЛЬНИЙ ЧАТ-ЗАСТОСУНОК, ПЕРСОНАЛІЗОВАНЕ НАВЧАННЯ, ОСВІТНЯ ПЛАТФОРМА, ШТУЧНИЙ ІНТЕЛЕКТ, ГЕНЕРАЦІЯ ТЕСТІВ, ФЛЕШ-КАРТКИ, АДАПТИВНЕ НАВЧАННЯ.

Об'єкт дослідження – процес побудови інтелектуальних клієнт-серверних програмних систем для обробки та аналізу навчального контенту.

Предмет дослідження – методи та програмні засоби генерації навчального контенту (із використанням моделей штучного інтелекту) та адаптації матеріалу до рівня знань користувача.

Мета роботи – підвищення ефективності та зменшення часу засвоєння навчального матеріалу користувачем за рахунок автоматичної генерації структурованого контенту з урахуванням рівня підготовки користувача.

Матеріали, методи та технічні засоби: Використано методи об'єктно-орієнтованого програмування та обробки природної мови. Серверну частину реалізовано на фреймворку NestJS із використанням TypeScript, Prisma ORM та бази даних PostgreSQL. Для інтеграції з моделями штучного інтелекту використано API OpenAI. Клієнтську частину побудовано на фреймворку Next.js з використанням TypeScript, Tailwind CSS та користувацьких компонентів бібліотеки Shadcn UI. Для управління станом запитів використано React Query. Реалізовано обробку PDF-документів із використанням бібліотеки pdf-parse. Розроблення проводилося в середовищі Visual Studio Code на персональному комп'ютері з операційною системою Windows 11.

Результати. Розроблено вебзастосунок, що забезпечує інтерактивну взаємодію з користувачем у форматі чату та підтримує генерацію

структурованих пояснень навчального матеріалу. У системі реалізовано функціональність створення тестових сесій із автоматичним оцінюванням результатів і наданням детальних пояснень до відповідей. Також передбачено формування флеш-карток із можливістю їх перегортання та зручної навігації. Застосунок підтримує аналіз PDF-документів із подальшою генерацією на їх основі навчального контенту. Додатково реалізовано адаптацію складності матеріалу відповідно до рівня підготовки користувача (trainee, junior, middle, senior), а також збереження історії навчання та взаємодії з системою.

Висновки. Досягнуто мети роботи: реалізовано систему, що дозволяє скоротити час опрацювання навчального матеріалу за рахунок автоматизованої генерації конспектів, тестів та флеш-карток, а також їх адаптації до рівня підготовки користувача. Застосунок забезпечує інтерактивну взаємодію, персоналізацію навчання та ефективне використання навчальних матеріалів, зокрема у форматі PDF.

Галузь використання – освітні платформи, самостійне навчання, підготовка студентів та розробників, а також системи підвищення кваліфікації в IT-сфері.

Економічна ефективність. Використання застосунку дозволяє скоротити час на підготовку навчальних матеріалів, зменшити витрати на створення тестів і навчального контенту, а також прискорити процес засвоєння інформації користувачем за рахунок персоналізації та автоматизації навчання.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 98 pages, 10 tables, 42 figures, 3 appendixes, 21 sources.

INTELLIGENT CHAT APPLICATION, PERSONALIZED LEARNING, EDUCATIONAL PLATFORM, ARTIFICIAL INTELLIGENCE, TEST GENERATION, FLASHCARDS, ADAPTIVE LEARNING.

Research object – the process of building intelligent client–server software systems for processing and analyzing educational content.

Research subject – methods and software tools for generating educational content (using artificial intelligence models) and adapting materials to the user's level of knowledge.

The purpose of the work – to increase the efficiency and reduce the time required for learning educational material through the automatic generation of structured content tailored to the user's level of preparation.

Materials, methods, and technical means: object-oriented programming methods and natural language processing techniques were used. The server side (backend) was implemented using the NestJS framework with TypeScript, Prisma ORM, and a PostgreSQL database. The OpenAI API was used for integration with artificial intelligence models. The client side (frontend) was built using the Next.js framework with TypeScript, Tailwind CSS, and Shadcn UI components. React Query was used for managing request state. PDF document processing was implemented using the pdf-parse library. Development was carried out in the Visual Studio Code environment on a personal computer running Windows 11.

Results. A web application was developed that provides interactive user interaction in a chat format and supports the generation of structured explanations of educational material. The system implements functionality for creating test sessions with automatic evaluation of results and detailed explanations of answers.

It also provides the creation of flashcards with flipping functionality and convenient navigation. The application supports PDF document analysis with subsequent generation of educational content based on them. Additionally, adaptation of material complexity according to the user's level (trainee, junior, middle, senior) and storage of learning history and user interactions were implemented.

Conclusions. The objective of the work was achieved: a system was developed that reduces the time required to process educational material through automated generation of summaries, tests, and flashcards, as well as their adaptation to the user's level of knowledge. The application provides interactive interaction, personalized learning, and efficient use of educational materials, including PDF-based content.

Field of application – educational platforms, self-learning, training of students and developers, as well as professional development systems in the IT field.

Economic efficiency. The application reduces the time required for preparing educational materials, decreases the cost of creating tests and learning content, and accelerates the learning process through personalization and automation.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	11
Вступ	12
1 Аналіз предметної області	15
1.1 Короткий опис досліджуваної предметної області.....	15
1.2 Огляд існуючих рішень проблеми.....	18
1.2.1 Застосунок ChatGPT	18
1.2.2 Застосунок Quizlet	19
1.2.3 Застосунок Notion AI.....	20
1.2.4 Порівняльна таблиця аналогів та розроблюваної системи	21
1.3 Опис основних вимог до програми	21
1.3.1 Визначення функціональних вимог	21
1.3.2 Нефункціональні вимоги.....	23
1.4 Постановка завдання роботи.....	23
1.5 Висновки за розділом 1	24
2 Розробка архітектури програми	25
2.1 Структура системи	25
2.1.1 Основні компоненти системи	25
2.1.2 Взаємодія компонентів системи	26
2.2 Функціонування системи	27
2.2.1 Основні процеси системи.....	27
2.2.2 Логіка роботи системи	28
2.3 Вибір мови програмування та інструментарію для розробки програми.....	29
2.3.1 Мова програмування	29
2.3.2 Середовище розробки.....	30
2.3.3 База даних	31
2.4 Вимоги до технічних і програмних засобів	33
2.4.1 Необхідні технічні засобів для розробки та експлуатації	33

2.4.2 Вимоги до апаратного та програмного забезпечення.....	34
2.5 Висновки за розділом 2	34
3 Розробка програми	35
3.1 Архітектура системи	35
3.1.1 Структурна схема системи.....	37
3.2 Функціонування системи	38
3.2.1 Функціональна схема	38
3.2.2 Опис процесу обміну даними у системі	39
3.3 Проєктування бази даних.....	40
3.3.1 Структура бази даних.....	40
3.3.2 Взаємозв'язки між таблицями	40
3.4 Висновки за розділом 3	41
4 Експлуатація, тестування та експериментальне дослідження програми	43
4.1 Опис застосування програми	43
4.1.1 Огляд планової цільової аудиторії.....	43
4.1.2 Візуалізація планованої цільової аудиторії.....	45
4.2 Інструкція по роботі з програмою	46
4.2.1 Встановлення програми.....	46
4.2.2 Запуск та початкове налаштування програми	47
4.2.3 Методика роботи з програмою	49
4.3 Приклад роботи користувача з програмою	51
4.4 Тестування роботи програми	54
4.4.1 Функціональне тестування.....	54
4.4.2 Аналіз функціонального тестування	55
4.4.3 Тестування продуктивності	57
4.4.4 Аналіз продуктивності	58
4.4.5 Тестування безпеки	60
4.4.6 Аналіз безпеки	60
4.4.7 Аналіз користувацького досвіду	62
4.5 Рекомендації щодо використання.....	63

	10
4.6 Плани з розвитку	64
Висновки.....	66
Перелік джерел посилення	68
Додаток А Технічне завдання.....	70
Додаток Б Текст програми.....	77
Додаток В Слайди презентації	93

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AI	– Artificial Intelligence;
API	– Application Programming Interface;
CSS	– Cascading Style Sheets;
DOM	– Document Object Model;
HTML	– HyperText Markup Language;
HTTP	– HyperText Transfer Protocol;
JSON	– JavaScript Object Notation;
JWT	– JSON Web Token;
ORM	– Object-Relational Mapping;
PDF	– Portable Document Format;
REST	– Representational State Transfer;
SPA	– Single Page Application;
SQL	– Structured Query Language;
UI	– User Interface;
UX	– User Experience.

ВСТУП

Стрімкий розвиток технологій штучного інтелекту та цифрових освітніх платформ суттєво змінює підходи до отримання й опрацювання навчальної інформації [1]. У сучасному середовищі особливої актуальності набувають системи, здатні не лише надавати інформацію, а й адаптувати її до рівня підготовки користувача, автоматизувати процес створення навчальних матеріалів та забезпечувати інтерактивну взаємодію. Значні обсяги інформації, висока складність технічних тем і потреба у швидкому засвоєнні знань створюють необхідність у використанні інтелектуальних інструментів підтримки навчання. Незважаючи на наявність великої кількості освітніх платформ та чат-застосунків, більшість із них або не підтримують персоналізовану адаптацію контенту, або мають обмежені можливості генерації навчальних матеріалів на основі користувацьких джерел даних.

Актуальність теми підтверджується активним впровадженням технологій штучного інтелекту в освітню сферу [2]. Сучасні великі мовні моделі дозволяють автоматизувати створення пояснень, тестів, конспектів і навчальних карток, що значно скорочує час підготовки матеріалів та підвищує зручність самостійного навчання. Водночас існуючі рішення часто не забезпечують цілісного підходу, який би поєднував інтерактивний чат, аналіз навчальних документів, генерацію тестових сесій і персоналізацію матеріалу відповідно до рівня знань користувача. Особливо актуальною є проблема ефективного опрацювання PDF-документів та технічної літератури, яка зазвичай потребує значних витрат часу на структурузацію та аналіз.

Проблема дослідження полягає в необхідності створення інтелектуальної системи, здатної автоматизувати процес формування навчального контенту, забезпечувати адаптацію матеріалу до рівня підготовки користувача та зменшувати час засвоєння інформації. Аналіз сучасних платформ, таких як ChatGPT, Quizlet чи Coursera, показує, що вони переважно орієнтовані на окремі аспекти навчального процесу та не забезпечують

комплексного поєднання інтерактивної взаємодії, генерації тестів, флеш-карток і аналізу користувацьких документів у межах єдиної системи. Провідні дослідження та сучасні тенденції веброзробки вказують на перспективність використання клієнт-серверної архітектури, фреймворків Next.js та NestJS, а також моделей штучного інтелекту для побудови адаптивних освітніх платформ.

Об'єктом дослідження є процес побудови інтелектуальних клієнт-серверних програмних систем для обробки та аналізу навчального контенту.

Предметом дослідження є методи та програмні засоби генерації навчального контенту із використанням моделей штучного інтелекту та адаптації результатів до рівня знань користувача.

Мета роботи полягає у розробці вебсистеми, яка зменшує час засвоєння навчального матеріалу користувачем за рахунок автоматичної генерації структурованих конспектів, тестових завдань та флеш-карток із текстових і PDF-джерел із урахуванням рівня підготовки користувача.

Для досягнення цієї мети було окреслено та виконано такі етапи дослідження:

- проведення аналізу сучасних технологій та платформ у сфері інтелектуального навчання і генерації освітнього контенту;
- проєктування архітектури клієнт-серверного вебзастосунку з підтримкою інтеграції моделей штучного інтелекту;
- розробка функціоналу інтерактивного чату, генерації пояснень, тестових сесій та флеш-карток, а також механізму аналізу PDF-документів а також адаптації складності матеріалу;
- проведення тестування програмної системи та перевірка стабільності, коректності і якості роботи реалізованого функціоналу.

Методологічну основу дослідження складають принципи об'єктно-орієнтованого програмування, клієнт-серверної взаємодії та методи обробки природної мови. Для реалізації серверної частини використано фреймворк NestJS, Prisma ORM та систему керування базами даних PostgreSQL.

Клієнтську частину побудовано на основі Next.js із використанням TypeScript, Tailwind CSS та бібліотеки компонентів Shadcn UI. Для інтеграції моделей штучного інтелекту використано API OpenAI, а для обробки PDF-документів – бібліотеку pdf-parse.

Практична цінність роботи полягає у створенні інструменту, який дозволяє автоматизувати підготовку навчальних матеріалів, скоротити час на опрацювання складної інформації та забезпечити персоналізований підхід до навчання. Розроблений застосунок може використовуватися студентами, розробниками-початківцями та користувачами, які займаються самостійним опануванням технічних дисциплін.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Короткий опис досліджуваної предметної області

Сучасний розвиток інформаційних технологій та стрімке поширення систем штучного інтелекту суттєво трансформують підходи до навчання, самостійного опрацювання інформації та підготовки фахівців у сфері ІТ. Велика кількість навчальних матеріалів, технічної документації та освітніх ресурсів створює проблему швидкого й ефективного засвоєння інформації. Особливо актуальною ця проблема є для студентів, початківців-розробників та спеціалістів, які постійно працюють із великими обсягами складного текстового контенту.

Традиційні підходи до навчання, що базуються на ручному створенні конспектів, тестів або навчальних карток, потребують значних витрат часу та не забезпечують достатнього рівня персоналізації. Крім того, більшість освітніх платформ орієнтовані лише на зберігання або відображення контенту, не надаючи користувачеві інструментів для автоматичного аналізу матеріалів та адаптації їх до індивідуального рівня знань.

Ключові поняття, що формують основу сучасних інтелектуальних освітніх систем:

- інтелектуальний чат-застосунок – програмна система, що забезпечує взаємодію користувача із моделями штучного інтелекту у форматі діалогу;
- тестова сесія (test session) – інтерактивний модуль перевірки знань із автоматичним оцінюванням та поясненням результатів;
- флеш-картки (flashcards) – навчальний інструмент для закріплення знань шляхом повторення інформації у форматі “запитання – відповідь”;
- персоналізоване навчання – підхід до подання матеріалу з урахуванням рівня підготовки, темпу навчання та потреб користувача;

– генерація навчального контенту – автоматичне створення конспектів, пояснень, тестів і флеш-карток із використанням алгоритмів штучного інтелекту.

На сьогодні освітня галузь стикається з низкою важливих викликів. По-перше, обсяги інформації постійно зростають, через що користувачам складно швидко структурувати та засвоювати нові знання. По-друге, існує проблема адаптації навчального матеріалу до різного рівня підготовки користувачів. По-третє, значна частина навчальних сервісів не підтримує інтегровану роботу з документами користувача, зокрема PDF-файлами, що ускладнює автоматизацію процесу навчання. Крім того, більшість платформ не поєднують в одному середовищі чатову взаємодію, генерацію тестів, флеш-карток та аналіз навчальних матеріалів.

Серед основних тенденцій розвитку сучасних інтелектуальних освітніх систем варто виділити активне впровадження великих мовних моделей (LLM), персоналізацію навчального контенту, автоматичну генерацію тестових завдань, а також інтеграцію мультимодальної обробки даних. Особливої популярності набувають системи, здатні аналізувати текстові документи, формувати структуровані конспекти, генерувати інтерактивні флеш-картки та адаптувати складність матеріалу відповідно до рівня знань користувача. Використання технологій штучного інтелекту дозволяє значно скоротити час на підготовку навчальних матеріалів і підвищити доступність освітніх ресурсів для широкого кола користувачів.

Окрему увагу сучасні дослідження приділяють розвитку адаптивного навчання, у межах якого система автоматично підлаштовує формат і складність контенту під індивідуальні особливості користувача. Такий підхід сприяє підвищенню швидкості засвоєння інформації, покращенню залученості до навчального процесу та зменшенню навантаження на викладачів і авторів навчальних матеріалів. Крім того, поширення хмарних технологій та API сервісів штучного інтелекту значно спростило інтеграцію AI-функціоналу у вебзастосунки та освітні платформи.

Значного поширення набувають платформи, що використовують штучний інтелект для автоматизації створення освітнього контенту, генерації пояснень і тестування знань користувачів. Такі системи дозволяють значно спростити процес підготовки до навчання, підвищити доступність навчальних матеріалів та забезпечити більш персоналізований підхід до засвоєння інформації. Динаміку користувацького інтересу платформ на базі штучного інтелекту, таких як OpenAI та Quizlet, проаналізовано за допомогою даних Google Trends, отриманих через бібліотеку pytrends [3], та відображено на рисунках 1.1 та 1.2.

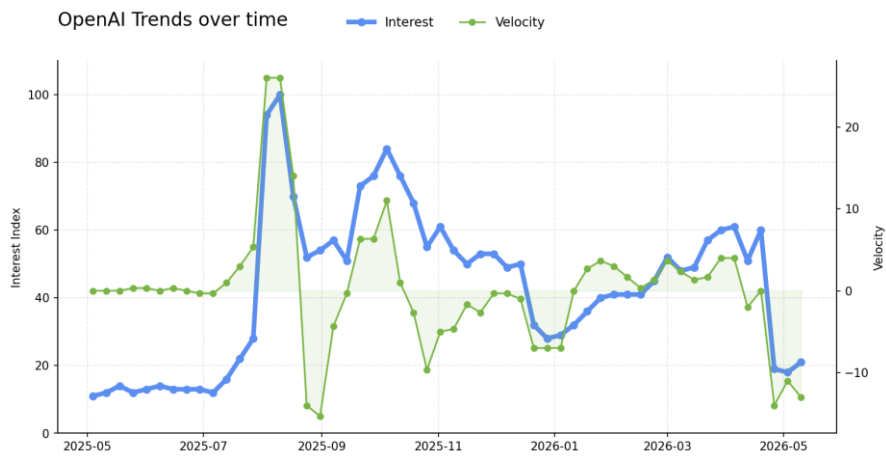


Рисунок 1.1 – Динаміка користувацького інтересу платформи OpenAI за останній рік [3]

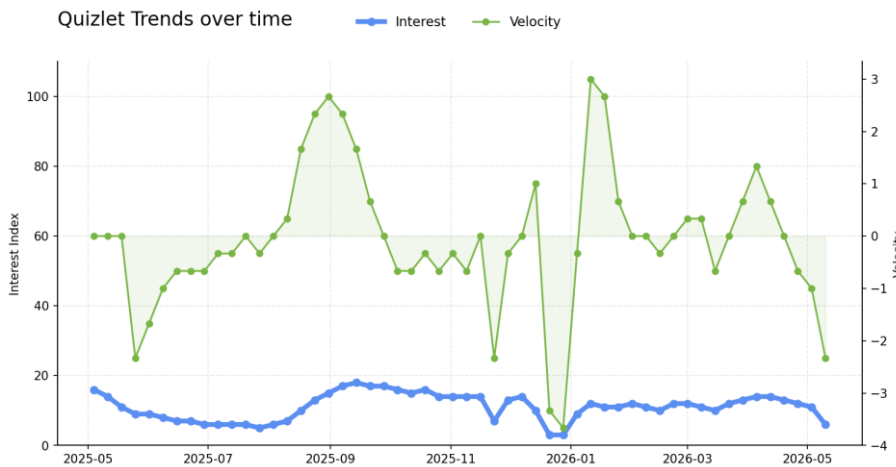


Рисунок 1.2 – Динаміка користувацького інтересу платформи Quizlet за останній рік [3]

1.2 Огляд існуючих рішень проблеми

Для аналізу сучасних підходів до автоматизації навчального процесу розглянемо функціональні можливості платформ ChatGPT, Quizlet та Notion AI. Кожне з цих рішень надає окремі інструменти для роботи з навчальним контентом, проте жодне не поєднує інтерактивний навчальний чат, генерацію тестових сесій, флеш-карток і аналіз PDF-документів у межах єдиної системи.

1.2.1 Застосунок ChatGPT

OpenAI ChatGPT – інтелектуальна чат-платформа, що використовує великі мовні моделі для генерації текстових відповідей, пояснень і навчального контенту [4].

Переваги:

- високоякісна генерація тексту та пояснень;
- підтримка інтерактивного діалогу;
- підтримка завантаження документів.

Недоліки:

- відсутність повноцінної системи тестових сесій;
- обмежені можливості роботи з флеш-картками;
- відсутність структурованого збереження навчального прогресу;

Зовнішній вигляд сторінки чату сервісу ChatGPT зображено на рисунку 1.3.

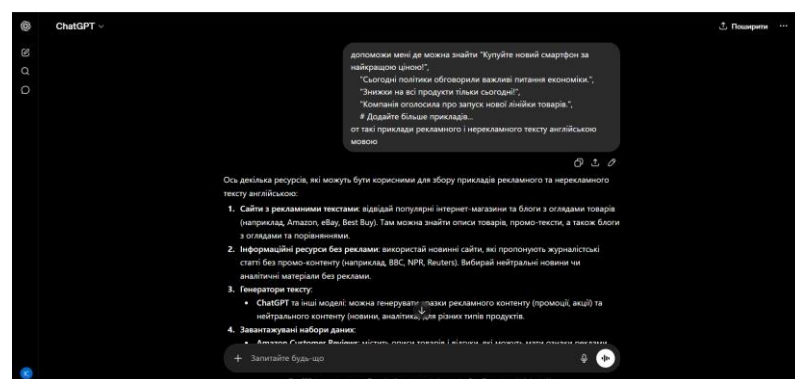


Рисунок 1.3 – Сторінка чату сервісу ChatGPT [4]

1.2.2 Застосунок Quizlet

Quizlet – популярна освітня платформа для створення флеш-карток та тестів[5].

Переваги:

- зручний механізм роботи з флеш-картками;
- інтерактивні режими перевірки знань;
- простий та зрозумілий інтерфейс;
- підтримка мобільних платформ.

Недоліки:

- відсутність повноцінного AI-чату;
- слабка персоналізація пояснень;
- обмежений аналіз PDF-документів;
- недостатня адаптація матеріалу до рівня користувача.

Зовнішній вигляд сторінки навчального посібника сервісу Quizlet зображено на рисунку 1.4.

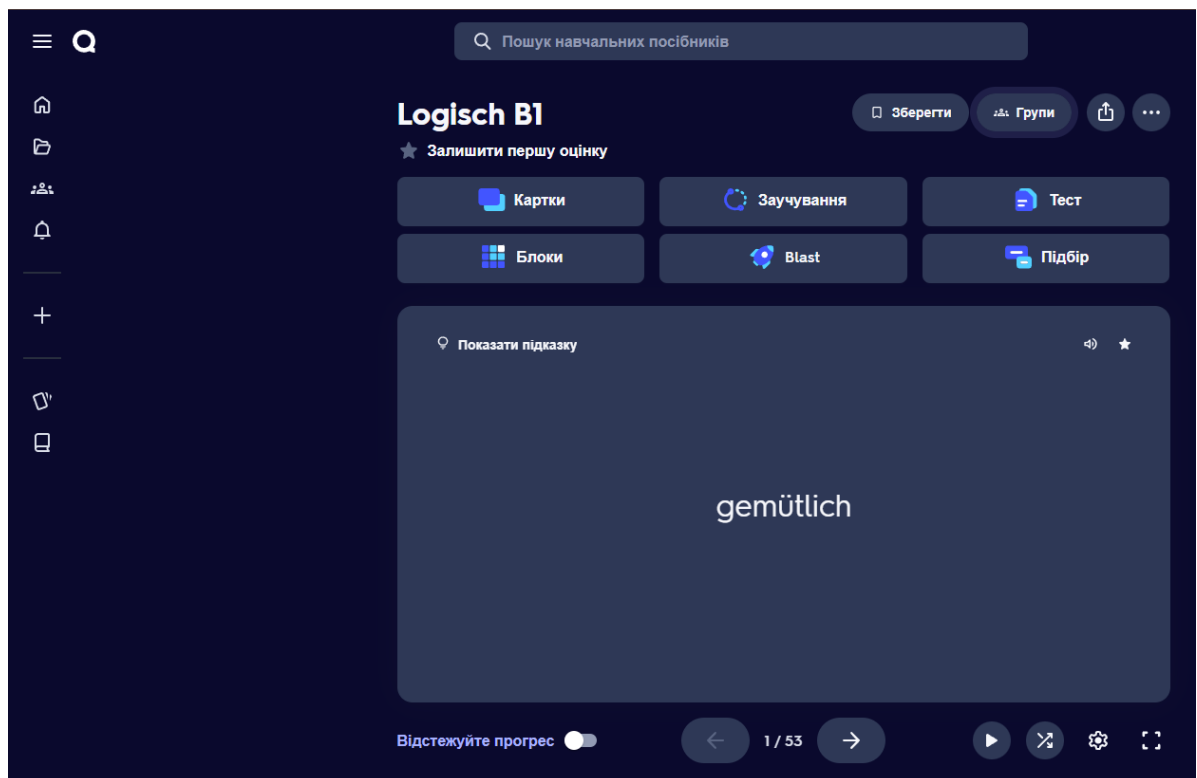


Рисунок 1.4 – Сторінка навчального посібника сервісу Quizlet [5]

1.2.3 Застосунок Notion AI

Notion AI – інструмент для роботи з текстом і автоматизації створення нотаток на базі штучного інтелекту[6].

Переваги:

- зручне структурування інформації;
- генерація конспектів та коротких пояснень;
- інтеграція з нотатками та документами;
- сучасний інтерфейс.

Недоліки:

- відсутність повноцінних тестових сесій;
- немає інтерактивного режиму навчання;
- відсутній механізм флеш-карток;
- обмежені можливості адаптації складності матеріалу.

Зовнішній вигляд початкової сторінки чату сервісу Notion AI зображено на рисунку 1.5.

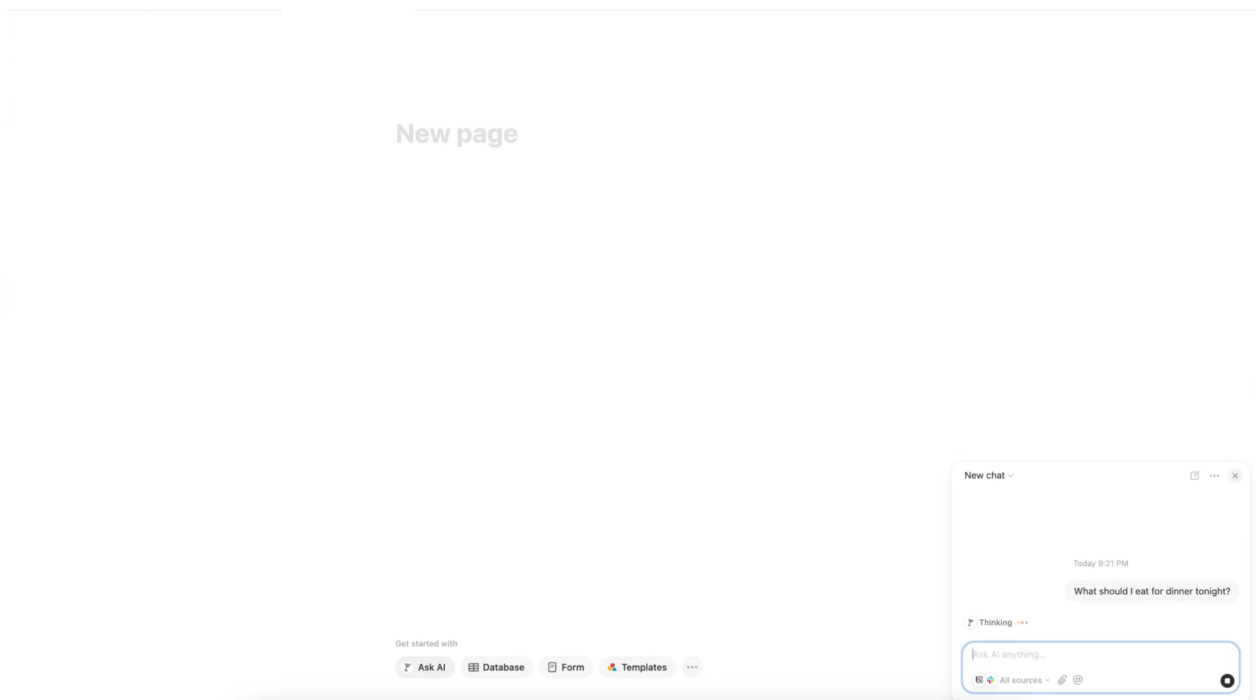


Рисунок 1.5 – Початкова сторінка чату сервісу Notion AI [6]

1.2.4 Порівняльна таблиця аналогів та розроблюваної системи

Для визначення функціональних переваг розроблюваної системи проведемо порівняння із платформами ChatGPT, Quizlet та Notion AI за ключовими критеріями. Результати аналізу наведено в таблиці 1.1.

Таблиця 1.1 – Функціональні характеристики систем освітніх платформ

Критерій	ChatGPT	Quizlet	Notion AI	Edu AI Platform
AI-чат	+	-	Частково	+
Генерація тестів	Частково	+	-	+
Флеш-картки	-	+	-	+
Аналіз PDF	+	-	Частково	+
Персоналізація навчання	Частково	Частково	-	+
Адаптація рівня складності	-	-	-	+
Збереження історії навчання	Частково	+	+	+

1.3 Опис основних вимог до програми

1.3.1 Визначення функціональних вимог

Функціональні вимоги визначають основні можливості системи та сценарії взаємодії користувача із застосунком. Для вебзастосунку «Edu AI Platform» встановлено такі вимоги:

а) авторизація та керування користувачами:

- 1) реєстрація та автентифікація користувачів за email і паролем;
- 2) підтримка JWT-авторизації;

- 3) збереження історії навчання та взаємодії (фіксація дій користувача для подальшого аналізу прогресу);
- 4) керування профілем користувача;
- б) робота з навчальним контентом:
 - 1) генерація пояснень і конспектів;
 - 2) адаптація контенту відповідно до рівня користувача (підлаштування складності матеріалу під знання користувача);
 - 3) підтримка режимів trainee, junior, middle та senior;
 - 4) інтерактивна чат-взаємодія;
- в) тестування та перевірка знань:
 - 1) генерація тестових сесій;
 - 2) автоматичне оцінювання відповідей (перевірка правильності відповідей без участі викладача);
 - 3) пояснення правильних відповідей(надання обґрунтування правильних варіантів для навчання);
 - 4) збереження результатів тестування;
- г) робота з флеш-картками:
 - 1) автоматичне створення флеш-карток;
 - 2) перегортання карток;
 - 3) навігація між картками;
 - 4) групування карток за секціями;
- д) робота з PDF-документами:
 - 1) завантаження PDF-файлів (додавання документів у систему для аналізу);
 - 2) автоматичне зчитування тексту;
 - 3) генерація навчального контенту на основі PDF;
 - 4) структуризація отриманої інформації.

1.3.2 Нефункціональні вимоги

Нефункціональні вимоги визначають основні характеристики якості програмної системи, такі як продуктивність, безпека, зручність використання та сумісність:

а) безпека:

- 1) шифрування паролів за допомогою bcrypt;
- 2) захист API-ендпоінтів через JWT;
- 3) серверна валідація даних (перевірка коректності введених даних на стороні сервера для запобігання помилкам і атакам);

б) продуктивність:

- 1) швидка генерація відповідей;
- 2) оптимізація запитів до бази даних;
- 3) кешування клієнтських запитів через React Query;

в) користувацький інтерфейс та зручність використання:

- 1) адаптивний інтерфейс;
- 2) сучасний інтерфейс із використанням Tailwind CSS та Shadcn UI;
- 3) естетичний та простий у використанні інтерфейс (інтуїтивно зрозуміле та візуально привабливе оформлення системи);

г) сумісність:

- 1) підтримка сучасних браузерів;
- 2) стабільна робота на мобільних і десктопних пристроях;
- 3) підтримка Windows, macOS та Linux;

1.4 Постановка завдання роботи

Метою роботи є розроблення інтелектуального вебзастосунку для автоматизації процесу навчання та генерації освітнього контенту з використанням технологій штучного інтелекту.

Основні завдання розробки:

- проєктування архітектури клієнт-серверної системи;
- реалізація AI-чату для взаємодії користувача із системою;
- розробка модулів генерації тестів, конспектів і флеш-карток, аналізу PDF-документів;

- реалізація персоналізації навчального контенту;
- забезпечення безпечної авторизації та збереження даних;
- тестування стабільності та коректності роботи системи.

Цілі розробки:

- скоротити час опрацювання навчальних матеріалів;
- скоротити час опрацювання навчальних матеріалів;
- автоматизувати створення освітнього контенту;
- надати користувачеві єдине середовище для навчання та перевірки знань.

1.5 Висновки за розділом 1

Проведений аналіз сучасних освітніх платформ показав, що існуючі рішення здебільшого орієнтовані на окремі аспекти навчального процесу: чатову взаємодію, створення тестів або роботу з флеш-картками. Водночас більшість із них не забезпечує комплексного поєднання генерації навчального контенту, аналізу PDF-документів, адаптації складності матеріалу та інтерактивної перевірки знань у межах єдиної системи. Розроблюваний вебзастосунок спрямований на усунення зазначених недоліків шляхом інтеграції технологій штучного інтелекту, персоналізації навчального процесу та автоматизації створення освітніх матеріалів.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Структура системи

2.1.1 Основні компоненти системи

Архітектура вебзастосунку базується на клієнт-серверному підході з розділенням відповідальності між frontend та backend частинами системи. Для забезпечення масштабованості, підтримуваності та гнучкості застосовано модульну архітектуру з використанням Next.js для клієнтської частини та NestJS для серверної логіки. Система складається з таких основних компонентів.

Шар представлення (Frontend / View):

- сторінки застосунку (Pages) – реалізовані за допомогою App Router у Next.js;
- клієнтські компоненти (Client Components) – відповідають за інтерактивність інтерфейсу;
- UI-компоненти – реалізовані на базі бібліотеки Shadcn UI та Tailwind CSS [7];
- модулі взаємодії з API – забезпечують обмін даними між клієнтом та сервером через HTTP-запити.

Шар управління станом та логікою клієнта:

- React Query – використовується для кешування та синхронізації серверного стану;
- Context API та custom hooks – забезпечують управління локальним станом застосунку;
- middleware та guards – відповідають за перевірку авторизації користувача та доступу до приватних сторінок.

Серверний шар (Backend):

- модулі NestJS – реалізують бізнес-логіку системи;
- REST API – забезпечує взаємодію між frontend та backend;

- JWT Authentication Module – реалізує систему автентифікації та авторизації;
- AI Module – забезпечує взаємодію з OpenAI API;
- PDF Processing Module – відповідає за обробку PDF-документів;
- Test Session Module – реалізує логіку тестових сесій та оцінювання відповідей;
- Flashcards Module – забезпечує створення та збереження флеш-карток.

Шар роботи з даними (Data Layer):

- Prisma ORM – забезпечує взаємодію із базою даних;
- PostgreSQL – основне сховище користувацьких даних;
- моделі даних (Entities) – описують структуру таблиць та взаємозв'язків між ними.

Шар зовнішніх інтеграцій:

- OpenAI API – генерація пояснень, тестів та флеш-карток;
- pdf-parse – аналіз та витяг тексту з PDF-документів;
- bcrypt – хешування паролів;
- JWT – підтримка захищених сесій користувачів.

2.1.2 Взаємодія компонентів системи

Взаємодія між компонентами системи «Edu AI Platform» реалізована за принципом клієнт-серверної архітектури.

Користувацький інтерфейс – API:

- frontend надсилає HTTP-запити до backend-сервісів;
- сервер повертає структуровані JSON-відповіді;
- React Query автоматично синхронізує стан даних у застосунку.

Frontend – Backend:

- клієнт передає JWT-токен через HTTP-only cookies;
- сервер виконує автентифікацію та перевірку доступу;

- дані трансформуються сервісами у формат, придатний для відправлення клієнту;
- результати обробки повертаються користувачу без перезавантаження сторінки.

Backend – AI-сервіси:

- сервер формує запити до OpenAI API;
- модель генерує пояснення, тести або флеш-картки;
- результати обробляються та зберігаються в базі даних.

Backend – База даних:

- Prisma ORM виконує CRUD-операції;
- PostgreSQL забезпечує централізоване зберігання даних;
- реалізовано зв'язки між користувачами, чатами, тестами та флеш-картками.

2.2 Функціонування системи

2.2.1 Основні процеси системи

Функціонування системи можна описати через основні сценарії використання.

Процес реєстрації та входу:

- користувач проходить реєстрацію за email та паролем;
- пароль хешується за допомогою bcrypt;
- після авторизації система генерує JWT-токен;
- токен зберігається в HTTP-only cookies.

Процес створення навчального контенту:

- користувач вводить тему або надсилає PDF-файл;
- сервер виконує аналіз тексту;
- OpenAI API генерує структуровані пояснення;
- результати відображаються у вигляді конспекту або чату.

Процес генерації тестів:

- користувач обирає режим Test Room;

- система генерує тестові питання;
- користувач проходить тестову сесію;
- сервер автоматично оцінює результати та формує пояснення до відповідей.

Процес роботи з флеш-картками:

- система генерує флеш-картки за темою;
- картки групуються за секціями;
- користувач може перегортати картки та зберігати їх у власній бібліотеці.

Процес адаптації складності:

- користувач обирає рівень знань (trainee, junior, middle, senior);
- система адаптує складність пояснень та тестів;
- генерація контенту виконується відповідно до рівня підготовки.

2.2.2 Логіка роботи системи

Логіка роботи системи «Edu AI Platform» базується на наступних принципах, що забезпечать зручність користування та стабільність:

Принцип персоналізації навчання:

- контент адаптується до рівня знань користувача;
- система враховує тип навчального матеріалу;
- забезпечується індивідуальний підхід до навчання;
- система автоматизовує подачу контенту за допомогою флеш-карток та тестів.

Принцип централізованого зберігання:

- усі дані користувача зберігаються у PostgreSQL;
- історія навчання доступна після повторного входу;
- забезпечується цілісність даних.

Принцип безпечної автентифікації:

- паролі не зберігаються у відкритому вигляді;
- JWT-токени використовуються для підтримки сесій;

- HTTP-only cookies захищають токени від XSS-атак.

Принцип реактивного оновлення інтерфейсу:

- React Query автоматично оновлює дані;
- користувач отримує миттєвий відгук інтерфейсу;
- зменшується кількість повторних запитів до сервера.

2.3 Вибір мови програмування та інструментарію для розробки програми

2.3.1 Мова програмування

Для реалізації вебзастосунку обрано мову програмування TypeScript, яка використовується як у фронтенд-, так і бекенд-частині системи[8]-[10].

Вибір TypeScript обумовлений такими перевагами:

- типізація: TypeScript забезпечує статичну перевірку типів, що дозволяє зменшити кількість помилок під час розробки та підвищити стабільність системи;
- єдина мова для frontend та backend: використання TypeScript у Next.js та NestJS спрощує підтримку проєкту та забезпечує єдиний стиль розробки;
- масштабованість: TypeScript добре підходить для створення великих модульних систем із великою кількістю взаємопов'язаних компонентів;
- підтримка сучасних технологій: мова має активну екосистему та підтримується більшістю сучасних фреймворків і бібліотек.

2.3.1.1 Опис переваг та обмежень вибраної мови

Переваги мови TypeScript у системі «Edu AI Platform»:

- статична типізація та зменшення кількості помилок;
- висока швидкість розробки;
- зручна інтеграція з Next.js та NestJS[11]-[12];

- велика кількість бібліотек та документації.

Обмеження:

- необхідність додаткової компіляції;
- складніша конфігурація порівняно зі звичайним JavaScript;
- більший поріг входження для початківців.

Порівняльна характеристика мов програмування для вебплатформ наведена в таблиці 2.1.

Таблиця 2.1 – Порівняльний аналіз мов програмування

Критерій	TypeScript	JavaScript	Python
Типізація	Статична	Динамічна	Динамічна
Продуктивність	Висока	Висока	Середня
Масштабованість	Висока	Середня	Середня
Підтримка frontend	Так	Так	Обмежена
Підтримка backend	Так	Так	Так
Зручність підтримки	Висока	Середня	Висока

TypeScript обрано завдяки високій масштабованості, безпечній типізації та повній інтеграції із сучасним стеком веброзробки.

2.3.2 Середовище розробки

Для розробки застосунку обрано середовище Visual Studio Code [13].

Основні причини вибору:

- підтримка TypeScript, Next.js та NestJS;
- інтеграція з Git;
- велика кількість розширень;
- вбудовані інструменти налагодження;
- підтримка Docker та PostgreSQL.

2.3.2.1 Обґрунтування вибору та опис його можливостей

VS Code – це універсальне середовище розробки, яке оптимально поєднує низьке споживання системних ресурсів із розширеним функціоналом для Full-stack програмування..

VS Code забезпечує:

- автодоповнення коду;
- рефакторинг;
- підтримку ESLint та Prettier;
- тестування та налагодження застосунку.

Порівняння популярних середовищ для веброзробки представлено в таблиці 2.2.

Таблиця 2.2 – Порівняльна характеристика інструментів розробки

Критерій	VS Code	WebStorm	Sublime Text
Підтримка TypeScript	Повна	Повна	Часткова
Інтеграція з Git	Вбудована	Вбудована	Обмежена
Швидкодія	Висока	Середня	Висока
Ресурсомісткість	Низька	Висока	Мінімальна
Вартість	Безкоштовно	Платно	Платно

Оптимальність VS Code для проєкту «Edu AI Platform» зумовлена балансом між функціональною потужністю та легкістю архітектури, що гарантує стабільну роботу інструментів налагодження та типізації для всього технологічного стеку.

2.3.3 База даних

Для стабільного функціонування платформи «Edu AI Platform», що передбачає складні ієрархічні зв'язки між користувачами, історією чатів з AI, навчальними тестами та флеш-картками, було обрано реляційну модель

збереження даних. Основним технологічним стеком для роботи з інформацією стали PostgreSQL як серверна база даних та Prisma як інструмент об'єктно-реляційного відображення (ORM).

2.3.3.1 Реляційна база даних PostgreSQL

PostgreSQL є потужною системою керування базами даних з відкритим кодом, яка забезпечує високу надійність та цілісність інформації. У проєкті «Edu AI Platform» вона виступає централізованим сховищем усіх даних.

Ключові переваги:

- цілісність та транзакційність: база гарантує, що жоден крок генерації тесту чи флеш-картки не буде втрачено через технічний збій – дані або зберігаються повністю, або не зберігаються зовсім;
- робота з лінкованими структурами: PostgreSQL блискуче справляється із запитам, де потрібно витягнути історію діалогів конкретного користувача, прив'язану до певних контекстів навчання;
- масштаб на перспективу: система дозволяє легко розширювати обсяги даних, що критично при накопиченні великої кількості згенерованого контенту від OpenAI.

Обмеження: система працює як віддалений сервер, що потребує стабільного з'єднання для синхронізації між backend-частиною та сховищем.

2.3.3.2 Об'єктно-реляційне відображення (Prisma ORM)

Замість прямого написання громіздких SQL-запитів, у проєкті використовується Prisma – сучасний прошарок, що «перекладає» мову бази даних на мову програмування TypeScript.

Переваги використання Prisma:

- захист від помилок: завдяки автоматичній генерації типів, розробник бачить помилку в структурі даних ще на етапі написання коду;

- еволюція схеми: будь-які зміни в структурі системи відбуваються через декларативні міграції, які легко відстежувати в Git;
- швидкість розробки: високорівневий API дозволяє маніпулювати об'єктами бази даних так само просто, як і звичайними масивами в коді.

Під час аналізу альтернатив розглядалася MongoDB, проте її нереляційна природа ускладнює підтримку чітких зв'язків між навчальними курсами та прогресом юзерів. Також було відхилено SQLite, оскільки вона не розрахована на паралельну роботу багатьох користувачів у вебсередовищі.

Детальне порівняння обраного рішення з іншими системами наведено в таблиці 2.3.

Таблиця 2.3 – Порівняльний аналіз систем керування базами даних

Критерій	PostgreSQL + Prisma	MongoDB	SQLite
1	2	3	4
Тип моделі	Реляційна(таблиці)	Документна	Реляційна (файл)
Цілісність даних	Повна (ACID)	Часткова	Висока
Складність зв'язків	Висока	Обмежено	Базова
Гнучкість системи	Сувора	Висока	Сувора
Типізація	Автоматична	Відсутня	Ручна
Масштабування	Вертикальне/Горизонтальне	Горизонтальне	Обмежене

Таким чином, використання PostgreSQL у зв'язці з Prisma ORM забезпечує «Edu AI Platform» архітектурну стійкість та дозволяє зосередитися на бізнес-логіці модулів, не відволікаючись на рутинне керування базою даних [14]-[16].

2.4 Вимоги до технічних і програмних засобів

2.4.1 Необхідні технічні засоби для розробки та експлуатації

Для розробки та подальшої експлуатації платформи «Edu AI Platform» необхідні:

- персональний комп'ютер або ноутбук;

- стабільне інтернет-з'єднання;
- сучасний веббраузер;
- сервер для розгортання backend-частини.

2.4.2 Вимоги до апаратного та програмного забезпечення

Апаратне забезпечення:

- процесор Intel Core i5 або аналогічний;
- оперативна пам'ять від 8 Гб;
- SSD-диск із вільним простором від 10 Гб.

Програмне забезпечення:

- Windows 10/11, Linux або macOS;
- Node.js версії 20+;
- PostgreSQL;
- Visual Studio Code;

2.5 Висновки за розділом 2

У другому розділі було розглянуто архітектуру вебзастосунку для генерації навчального контенту з використанням технологій штучного інтелекту. Описано структуру системи, взаємодію її компонентів та логіку функціонування основних модулів. Обґрунтовано вибір TypeScript як основної мови програмування, а також використання Next.js, NestJS, PostgreSQL та Prisma ORM для реалізації клієнт-серверної архітектури. Використання сучасних вебтехнологій дозволило забезпечити масштабованість, безпечну автентифікацію користувачів, зручний інтерфейс та стабільну роботу системи.

3 РОЗРОБКА ПРОГРАМИ

3.1 Архітектура системи

Архітектура навчальної AI-платформи «Edu AI Platform» побудована на базі клієнт-серверної моделі з модульною структурою на бекенді та компонентним підходом на фронтенді. Такий підхід забезпечує чітке розділення бізнес-логіки, інтеграцію з моделями штучного інтелекту та ефективну обробку великих масивів навчальних даних, що сприяє високій продуктивності та зручності підтримки системи.

Ключові складові архітектури.

Рівень даних – відповідає за структуру навчальних даних та взаємодію з базою даних (PostgreSQL):

- сутності, спроектовані за допомогою Prisma (User, ChatSession, Message, TestSession, FlashcardDeck, PdfDocument);
- сервіс доступу до даних (PrismaService), що контролює цілісність транзакцій та збереження контексту навчального процесу.

Рівень представлення – фокусується на візуальній частині та клієнтському досвіді (Next.js):

- функціональні блоки сторінок (ChatRoom, TestCenter, FlashcardsGallery тощо);
- набір інтерактивних елементів (інтерфейс чату, завантажувач файлів, квіз-картки);
- сучасний дизайн у темних тонах, реалізований на Tailwind CSS, що сприяє комфортному тривалому навчанню.

Рівень логіки – виступає посередником між інтерфейсом та даними:

- серверні контролери для обробки вхідних запитів (Auth, Chat, Education, Pdf);
- система клієнтських провайдерів та хуків, що синхронізують стан навчальних сесій у браузері користувача.

Функціональні сервіси – забезпечують роботу спеціалізованих модулів:

- AuthService – відповідає за безпеку, JWT-авторизацію та ідентифікацію користувачів;
- AIService – забезпечує комунікацію з OpenAI API для формування експертних відповідей та навчальних матеріалів;
- PdfService – реалізує алгоритми обробки та індексації документів для їх подальшого використання ШІ;
- ChatService – керує логами діалогів та забезпечує спадкоємність контексту в бесідах;
- TestService – автоматизує процес створення перевірочних тестів на основі вивчених тем;
- FlashcardService – допомагає структурувати та зберігати картки для закріплення знань;

Взаємодія між усіма компонентами системи деталізована на рисунку 3.1.

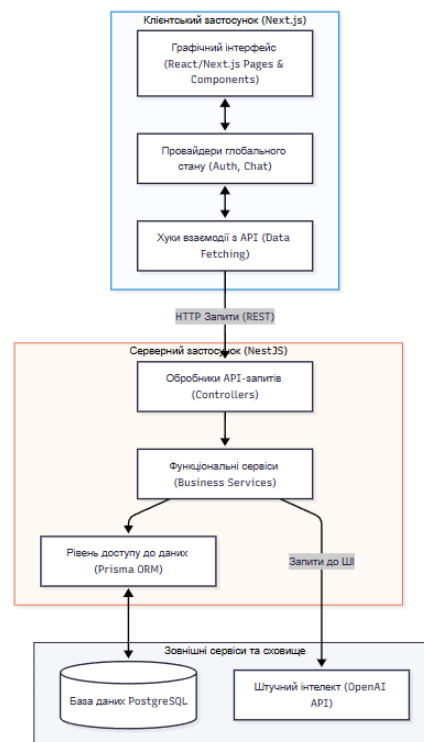


Рисунок 3.1 – Схема взаємодії компонентів системи

3.1.1 Структурна схема системи

Проект «Edu AI Platform» спроектовано за принципом модульної декомпозиції, де програмний код сегментовано на незалежні логічні блоки: auth, ai, chat, pdf, education. Це дозволило мінімізувати взаємозалежність компонентів та забезпечити високу спеціалізацію кожного модуля, що значно полегшує тестування окремих функцій та впровадження нових освітніх інструментів.

Структурна ієрархія та взаємозв'язок основних модулів системи представлені на рисунку 3.2.

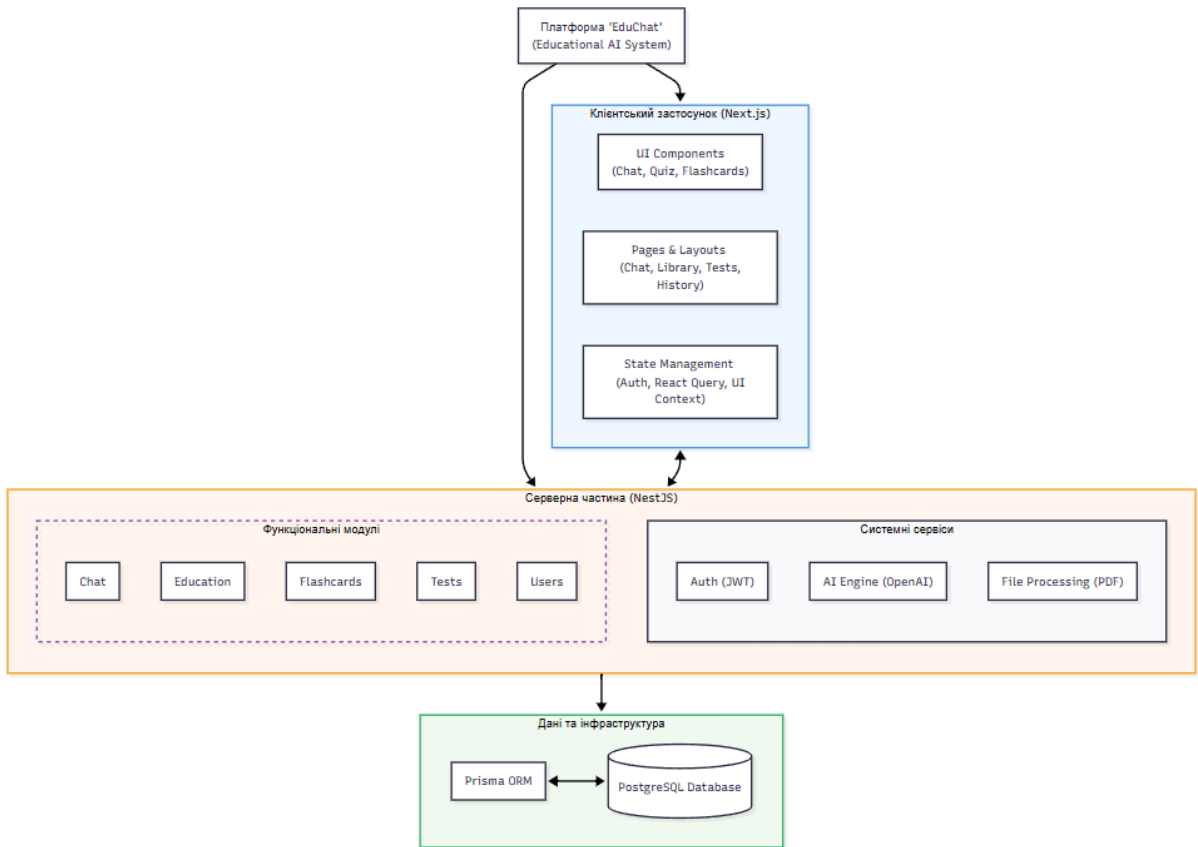


Рисунок 3.2 – Структурна схема системи

3.2 Функціонування системи

3.2.1 Функціональна схема

Робота платформи базується на синергії алгоритмів обробки природної мови та інструментів управління навчанням. Система забезпечує безперервний цикл генерації знань: від аналізу завантажених PDF-джерел до створення персоналізованих квізів з адаптивним рівнем складності. Це дозволяє користувачу плавно переходити від базового рівня (Trainee) до експертного (Senior). Загальна функціональна схема платформи наведена на рисунку 3.3.

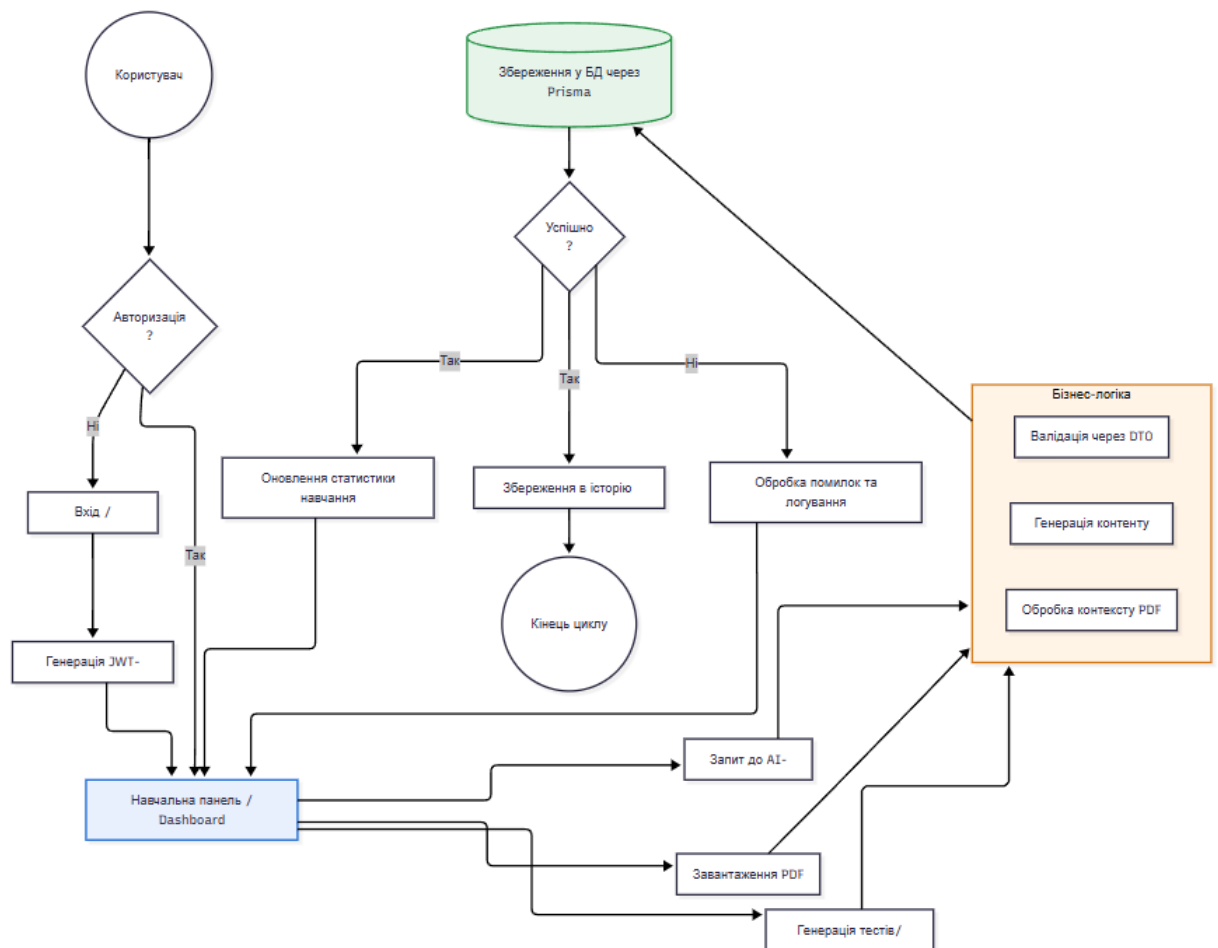


Рисунок 3.3 – Функціональна схема системи

3.2.2 Опис процесу обміну даними у системі

Потоки інформації в «Edu AI Platform» організовані за принципом контекстного збагачення знань користувача. Процес обміну даними складається з наступних етапів (рис. 3.4):

- ініціація запиту: користувач взаємодіє з інтерфейсом (завантаження файлу або запитання в чат);
- валідація та формування запиту: NestJS-сервер перевіряє права доступу та вилучає необхідну інформацію з документів для створення контекстного запиту до ШІ;
- генерація інтелектуальної відповіді: AIService передає запит мовній моделі, отримуючи структуровану відповідь або навчальний контент;
- фіксація результатів: історія діалогів, успішність тестів та нові картки зберігаються в PostgreSQL через Prisma ORM для подальшого аналізу;
- миттєве відображення: завдяки React Query користувач отримує результат без затримок та перезавантажень сторінки.

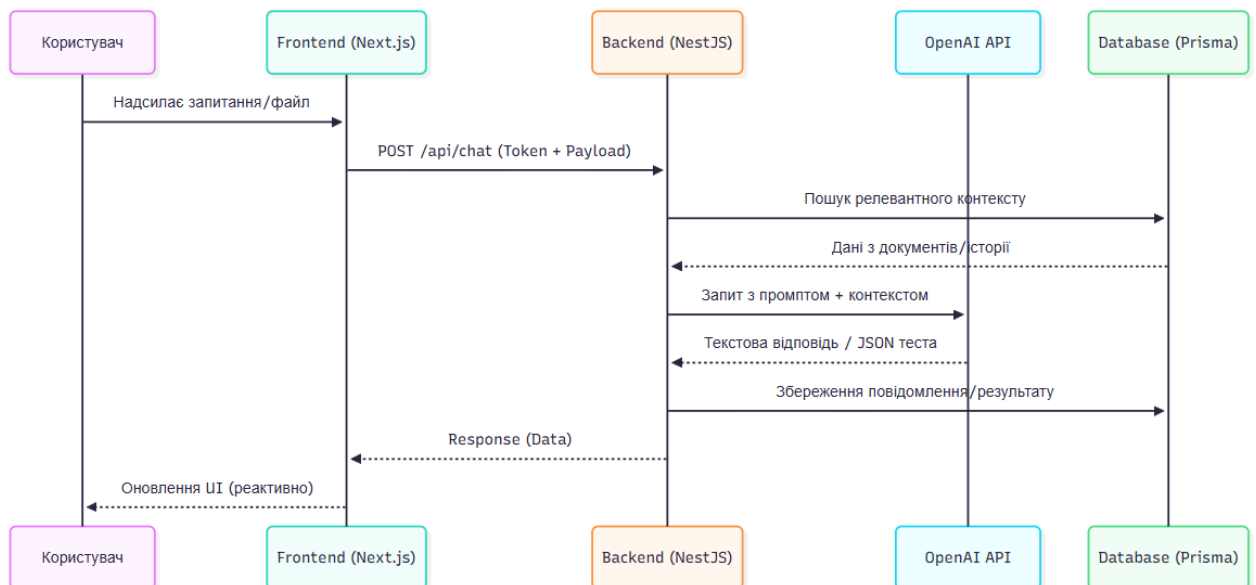


Рисунок 3.4 – Схема процесу обміну даними у системі

3.3 Проєктування бази даних

3.3.1 Структура бази даних

База даних системи розроблена на базі PostgreSQL з використанням Prisma ORM для забезпечення типізації та зручного управління міграціями. Архітектурне рішення орієнтоване на швидкий пошук релевантного контексту серед завантажених матеріалів та збереження детальної історії навчального прогресу. Основні сутності охоплюють профілі користувачів, сесії чатів, фрагменти документів та результати тестування.

3.3.2 Взаємозв'язки між таблицями

Система зв'язків підтримує цілісність освітньої траєкторії:

- Users – ChatSessions: ієрархія один-до-багатьох для ведення історії діалогів;
- ChatSessions – Messages: лінійний зв'язок повідомлень у межах однієї сесії;
- Users – PdfDocuments: персональна бібліотека знань кожного учня;
- PdfDocuments – PdfChunks: декомпозиція тексту для точного пошуку інформації III;
- Users – TestSessions: облік спроб проходження тестів та їх результатів;
- Users – FlashcardDecks: організація матеріалів для повторення.

Візуалізація взаємозв'язків між таблицями бази даних представлена на рисунку 3.5.

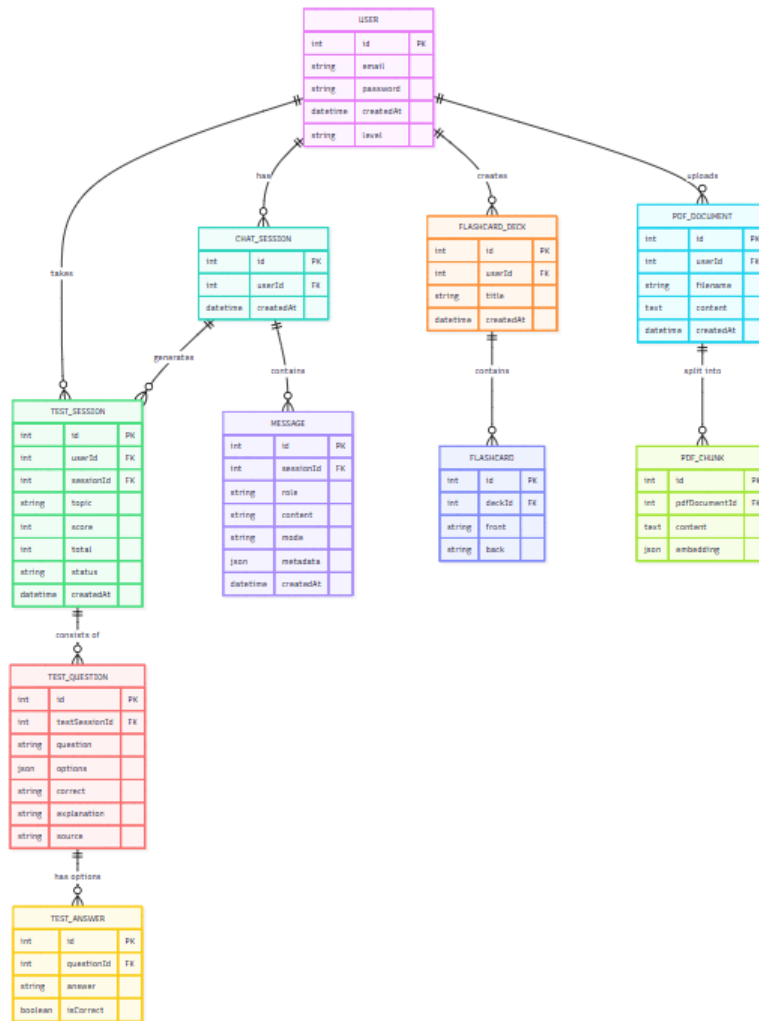


Рисунок 3.5 – Схема взаємозв'язків між таблицями бази даних «Edu AI Platform»

3.4 Висновки за розділом 3

У межах третього розділу було проведено детальне проєктування та технічну реалізацію платформи «Edu AI Platform». Було розроблено архітектурні рішення та алгоритми, що забезпечують ефективну взаємодію користувача зі штучним інтелектом на основі власних навчальних матеріалів.

Завдяки використанню сучасного стеку (NestJS, Next.js, Prisma) вдалося створити масштабовану систему, що складається з таких модулів:

- модуль авторизації та безпеки: безпечний доступ через JWT та збереження профілю користувача з його рівнем компетенцій. Ключові фрагменти реалізації наведено у Додатках Б.1–Б.2;

- модуль інтелектуального чату: ядро системи, що дозволяє вести діалог з ШІ, отримувати пояснення та зберігати контекст навчання. Реалізацію обробки запитів наведено в Додатку Б.3;
- модуль обробки документів: забезпечує завантаження, парсинг та використання зовнішніх матеріалів для навчання. Код обробки PDF представлено в Додатку Б.4;
- модуль генерації тестів: автоматичне створення квізів на основі діалогів або документів для перевірки знань. Ключові фрагменти коду наведено в Додатках Б.5–Б.6;
- модуль інтерактивних карток: інструментарій для інтервальних повторень та візуального вивчення термінів. Реалізацію логіки карток наведено в Додатку Б.7;

Результатом роботи стала цілісна програмна екосистема, готова до впровадження в освітній процес, що поєднує високу швидкість роботи та інтуїтивно зрозумілий інтерфейс.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис застосування програми

Навчальна AI-платформа «Edu AI Platform» — це інтелектуальне середовище для підтримки навчання та підготовки до іспитів, розроблене для оптимізації освітнього процесу за допомогою технологій обробки природної мови. Програма дозволяє користувачам вести контекстні діалоги з персоналізованим AI-тьютором, автоматично генерувати інтерактивні тести та картки (flashcards) на основі власних матеріалів, а також аналізувати складні навчальні PDF-документи для швидкого засвоєння інформації.

Застосунок охоплює ключові етапи сучасного навчального процесу: від первинного аналізу джерел знань до закріплення матеріалу через самоперевірку. Функціонал включає інтелектуальну підтримку діалогу, автоматичне структурування навчального контенту, адаптивне налаштування рівня складності (від початківця до експерта) та візуалізацію прогресу засвоєння тем через історію успішності.

Цільова аудиторія охоплює студентів вищих навчальних закладів, фахівців, що займаються самоосвітою (upskilling), старшокласників та викладачів, які прагнуть автоматизувати створення навчальних матеріалів. Завдяки інтуїтивно зрозумілому інтерфейсу та темній колірній гамі, адаптованій для тривалої роботи з текстом, система є доступною як для активних дослідників, так і для користувачів, які шукають швидкі відповіді на технічні чи академічні питання.

4.1.1 Огляд планової цільової аудиторії

Для оцінки потенційного ринку та бази користувачів «Edu AI Platform» було проведено аналіз трендів у сфері EdTech (Educational Technology) та ринку інструментів на базі штучного інтелекту для навчання. Нижче наведено дві прогнозовані структури аудиторії за віковими категоріями та форматом

використання платформи. Ці дані базуються на глобальних звітах про зростання популярності AI-асистентів у навчанні та специфіці українського ринку цифрової освіти..

4.1.1.1 Розподіл за віком

Прогнозований розподіл базується на даних про активність груп, залучених до активного навчання та професійного розвитку. Основне ядро користувачів зосереджено в сегменті молоді та фахівців, що постійно навчаються.

Група 18–24 роки (50%): найбільш чисельний сегмент, що включає студентів університетів та коледжів. Вони використовують «EduChat» для підготовки до сесій, аналізу наукових публікацій та швидкого освоєння складних тем за допомогою AI-пояснень.

Група 25–35 років (30%): спеціалісти (свічери, IT-фахівці, менеджери), які займаються самоосвітою або підвищенням кваліфікації. Їхній інтерес зумовлений потребою в інтенсивному навчанні без відриву від роботи та можливістю автоматичного створення карток для запам'ятовування термінів.

Група 14–17 років (15%): Старшокласники, які готуються до вступних іспитів (НМТ/ЗНО). Платформа допомагає їм структурувати великі обсяги шкільного матеріалу та тренуватися через генерацію квізів.

Група 36+ років (5%): Викладачі, науковці та люди, що практикують lifelong learning. Вони цінують платформу як інструмент для швидкого аналізу нових методичних матеріалів та створення тестів для студентів.

4.1.1.2 Розподіл за організаційною структурою

Прогнозована структура базується на типах освітньої діяльності та організаційному підході до навчання.

Індивідуальна самоосвіта (45%): користувачі, які використовують платформу як персонального тьютора для вивчення мов, програмування або інших професійних дисциплін у власному темпі.

Академічне навчання (35%): студенти, які застосовують «EduChat» для роботи з університетськими курсами, підготовки рефератів та курсових робіт на основі завантаженої літератури.

Підготовка до сертифікації та іспитів (10%): спеціалізоване використання для інтенсивного тренування через модулі тестів та флеш-картки перед важливими перевітками знань.

Освітні мікро-групи та викладачі (10%): використання платформи для підготовки контенту, яким діляться в межах невеликих навчальних груп або курсів.

4.1.2 Візуалізація планованої цільової аудиторії

На основі проведеного аналізу нижче представлено графічне відображення структури потенційних користувачів системи «Edu AI Platform» за віковими показниками та форматом освітньої діяльності (рис. 4.1 та 4.2).

Розподіл аудиторії за віком (у %)

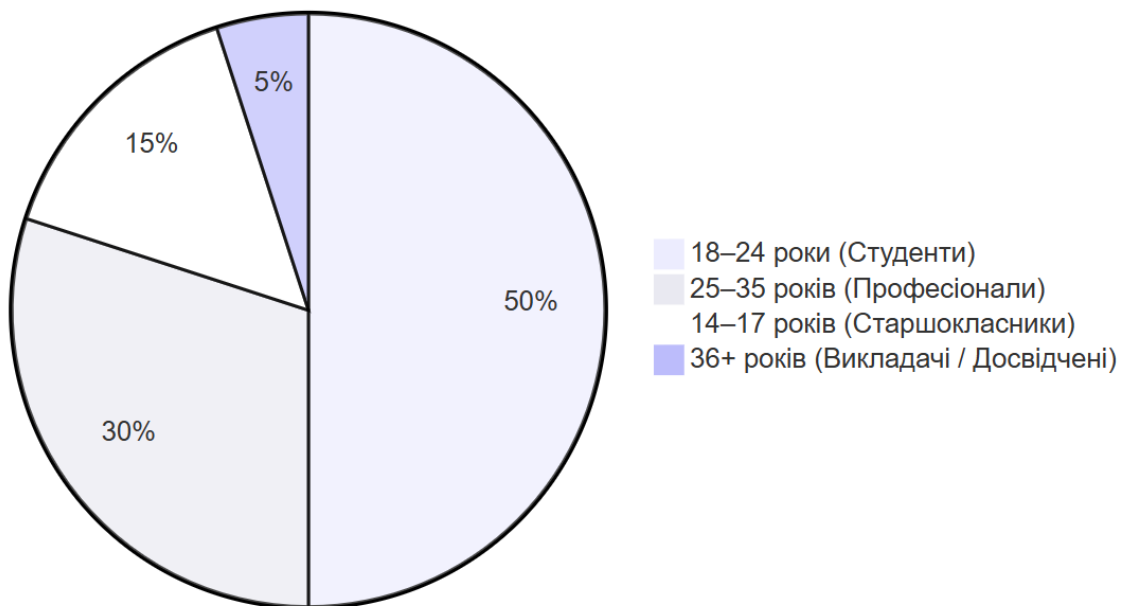


Рисунок 4.1 – Розподіл аудиторії Edu AI Platform за віковими категоріями

Розподіл за типом використання (у %)

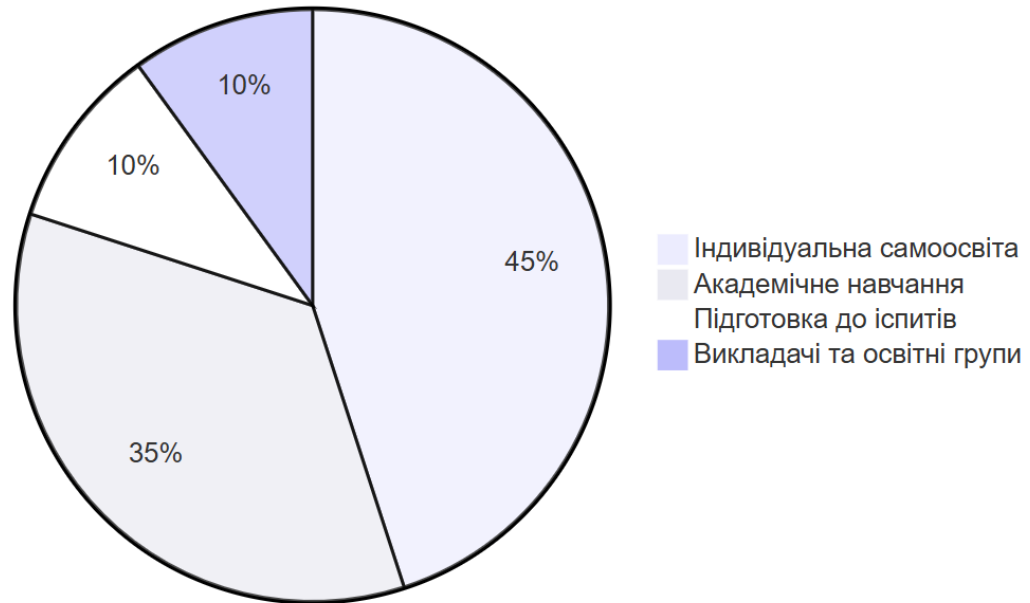


Рисунок 4.2 – Розподіл за типом організаційної структури

4.2 Інструкція по роботі з програмою

4.2.1 Встановлення програми

Оскільки «Edu AI Platform» є кросплатформним вебзастосунком, він працює за моделлю SaaS (Software as a Service) і не потребує класичного встановлення на локальний пристрій. Для стабільної роботи системи необхідно дотримуватися наступних технічних вимог:

а) технічні вимоги до середовища:

- 1) актуальна версія браузера (Google Chrome 100+, Firefox 95+, Safari 15+);
- 2) мінімальна роздільна здатність екрана 1280x800 пікселів;
- 3) стабільне підключення до мережі Інтернет (для комунікації з сервером та OpenAI API);
- 4) увімкнена підтримка JavaScript та Web APIs у налаштуваннях браузера;

б) доступ через вебінтерфейс:

- 1) перейдіть за посиланням на хостинг, де розгорнуто фронтенд застосунку;

2) додайте сторінку в закладки для швидкого доступу до навчальних сесій;

в) необхідні дозволи браузера:

1) дозвіл на завантаження та читання файлів (для роботи з PDF-документами);

2) дозвіл на використання локального сховища;

3) доступ до Cookies для забезпечення безпеки сесії.

Схему процесу авторизації та автентифікації зображено на рисунку 4.3.

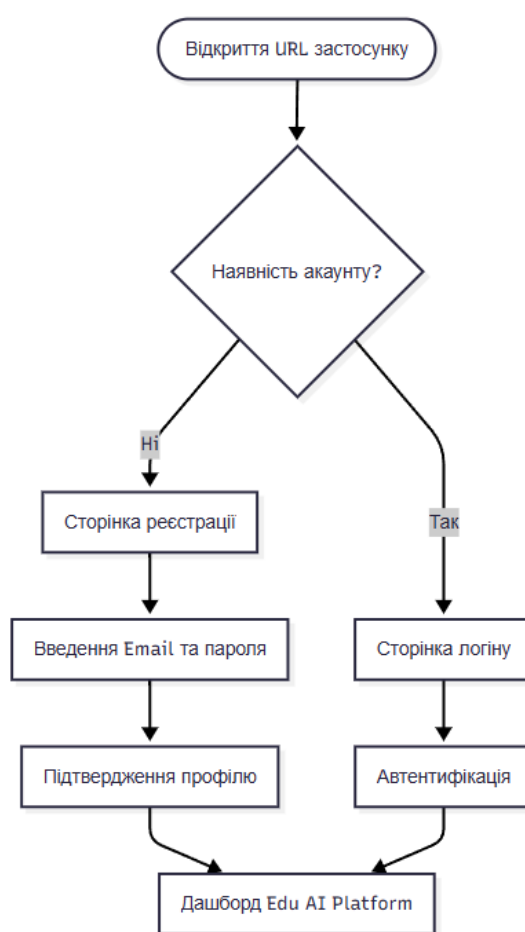


Рисунок 4.3 – Схема процесу авторизації в системі

4.2.2 Запуск та початкове налаштування програми

Після відкриття вебзастосунку користувачу необхідно пройти етап ідентифікації та налаштувати свій навчальний профіль:

а) перший запуск та реєстрація:

1) відкрийте головну сторінку. Якщо ви новий користувач, система запропонує створити акаунт;

2) у розділі реєстрації введіть електронну пошту та надійний пароль. Дані валідуються на стороні сервера через DTO;

3) після успішної реєстрації система автоматично авторизує вас;

б) налаштування профілю та рівня знань:

1) у розділі налаштувань профілю користувач може обрати свій поточний рівень компетенцій (Trainee, Junior, Middle, Senior);

2) обраний рівень впливає на те, як ШІ буде формулювати відповіді: від простих пояснень для початківців до глибоких технічних розборів для досвідчених фахівців;

в) створення першої навчальної сесії:

1) для початку роботи необхідно перейти в розділ «Chat»;

2) користувач може одразу почати діалог або завантажити навчальний PDF-файл у розділі «Library», який стане базою знань для ШІ;

Сторінки реєстрації, авторизації та налаштування профілю наведено на рисунках 4.4, 4.5 та 4.6 відповідно.

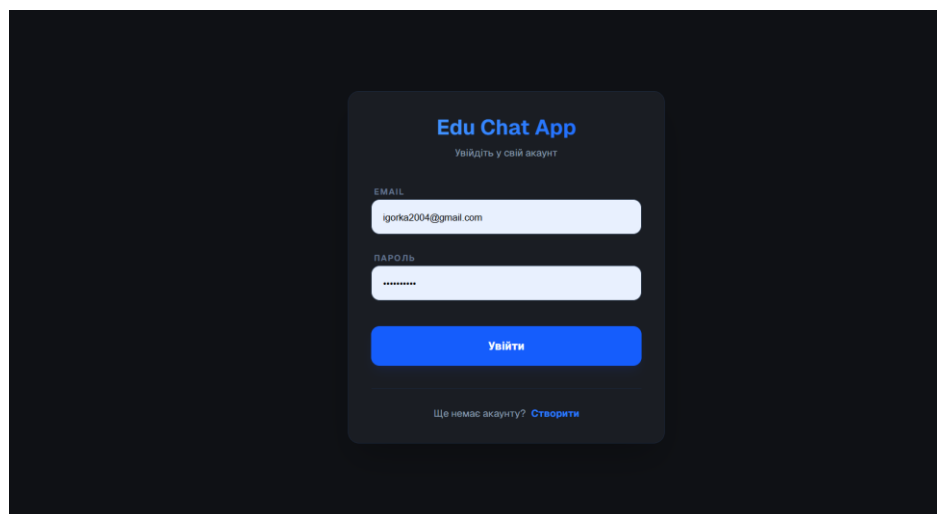


Рисунок 4.4 – Сторінка авторизації

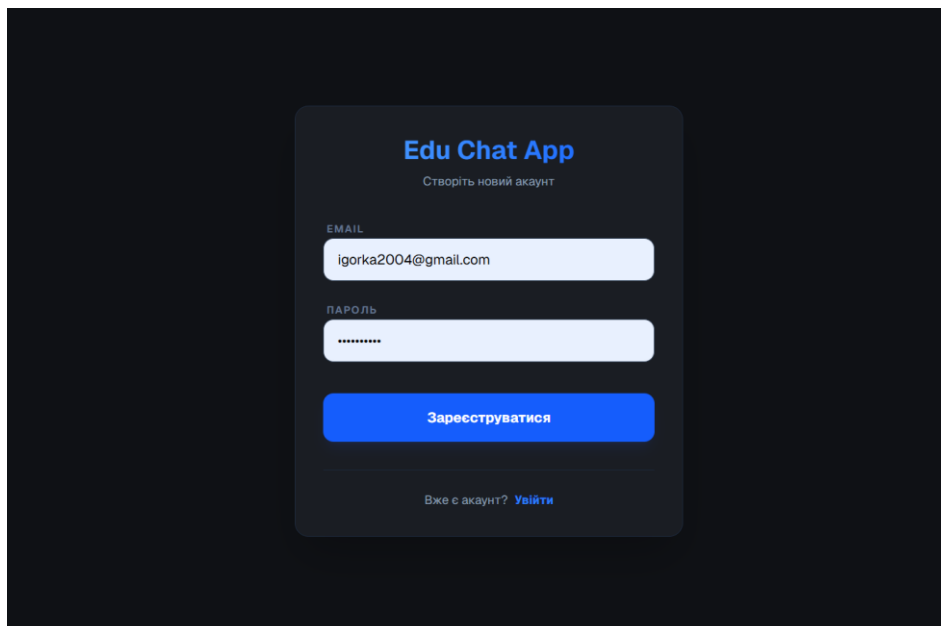


Рисунок 4.5 – Сторінка реєстрації

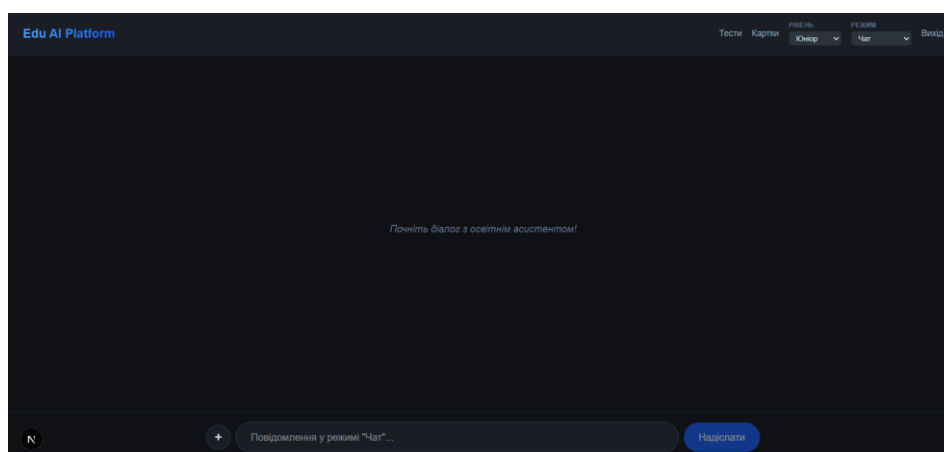


Рисунок 4.6 – Сторінка налаштувань рівня навчання та профілю

4.2.3 Методика роботи з програмою

Процес взаємодії з системою «Edu AI Platform» базується на циклічному підході до освоєння знань (рис. 4.7):

а) робота з джерелами знань:

- 1) завантаження навчальних матеріалів, конспектів або підручників;
- 2) автоматична індексація тексту для подальшого контекстного пошуку;

б) інтелектуальна взаємодія (Issues):

- 1) постановка запитань до ШІ щодо завантажених матеріалів або загальних тем;
- 2) отримання розгорнутих пояснень, прикладів коду або тез;
- 3) можливість уточнення деталей у режимі діалогу;

в) верифікація та закріплення знань:

- 1) генерація інтерактивних тестів на основі поточної бесіди або документа;
- 2) проходження квізів з миттєвою перевіркою відповідей та розбором помилок;
- 3) робота з картками для інтервального повторення ключових термінів.

Для досягнення найкращого результату рекомендується завантажувати структуровані PDF-документи, що дозволяє ШІ надавати максимально точні відповіді з посиланням на джерело.



Рисунок 4.7 – Схема життєвого циклу проєкту в системі

4.3 Приклад роботи користувача з програмою

Розглянемо типовий сценарій використання «EduChat» на прикладі студента, який вивчає тему «Основи баз даних».

Користувач реєструється в системі та в налаштуваннях встановлює рівень «Junior». Далі він переходить до розділу бібліотеки та завантажує PDF-файл з методичними вказівками щодо SQL. Система обробляє документ, фрагментує його та зберігає в базі даних.

Студент відкриває чат і ставить запитання: "Поясни різницю між JOIN та UNION на основі завантаженого файлу". Система за допомогою AIService знаходить релевантні частини в документі та формує відповідь, адаптовану під рівень Junior (зрозумілою мовою з прикладами).

Після обговорення теми студент натискає кнопку «Generate Test». Система створює 10 запитань на основі щойно обговореного матеріалу. Користувач проходить тест, бачить свій результат (наприклад, 8/10) та отримує пояснення щодо неправильних відповідей. Для закріплення термінів студент переходить у розділ «Flashcards», де система автоматично створила набір карток з ключовими SQL-командами.

Вебінтерфейс інтелектуального чату та вікно завантаження документів зображені на рисунках 4.8–4.9.

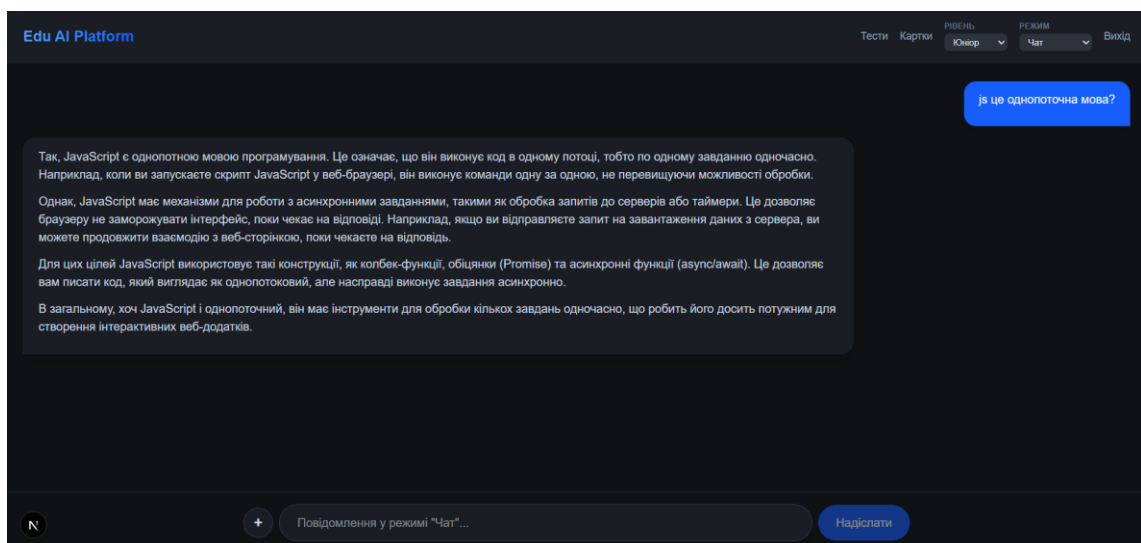


Рисунок 4.8 – Інтерфейс інтелектуального чату з AI-тьютором

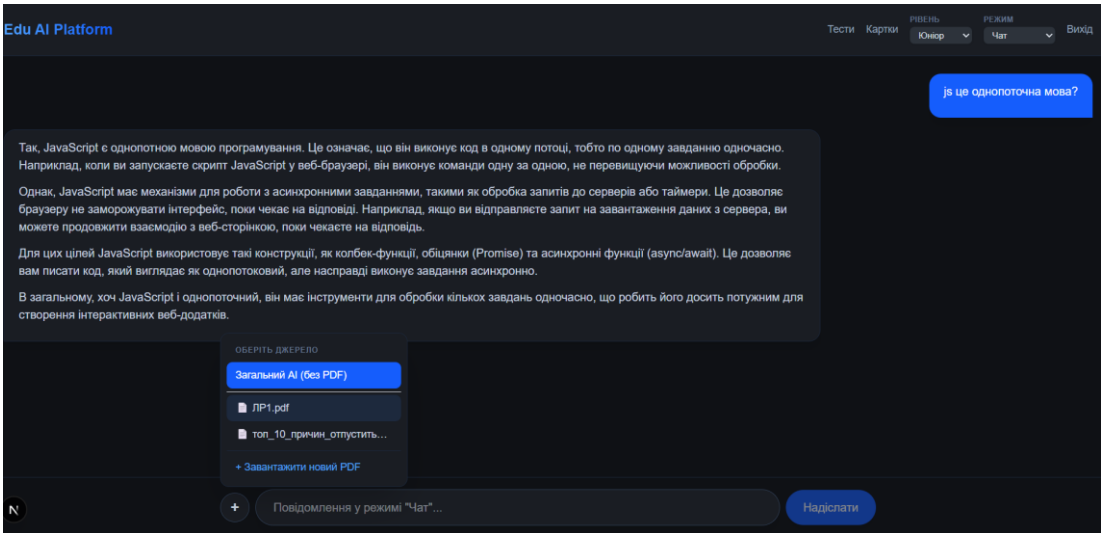


Рисунок 4.9 – Модуль завантаження та управління навчальними документами

Процес проходження генеративного тесту та інтерфейс перевірки знань зображені на рисунках 4.10–4.11.

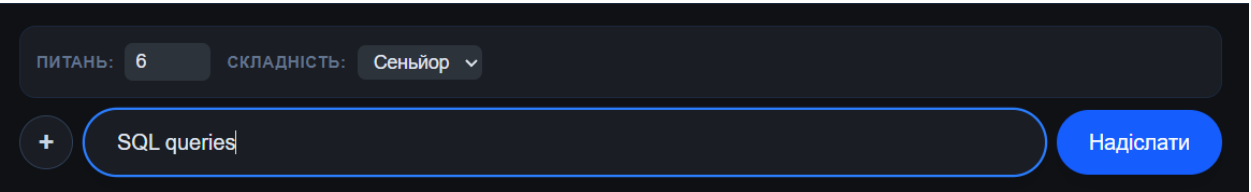


Рисунок 4.10 – Генерація та налаштування параметрів тесту

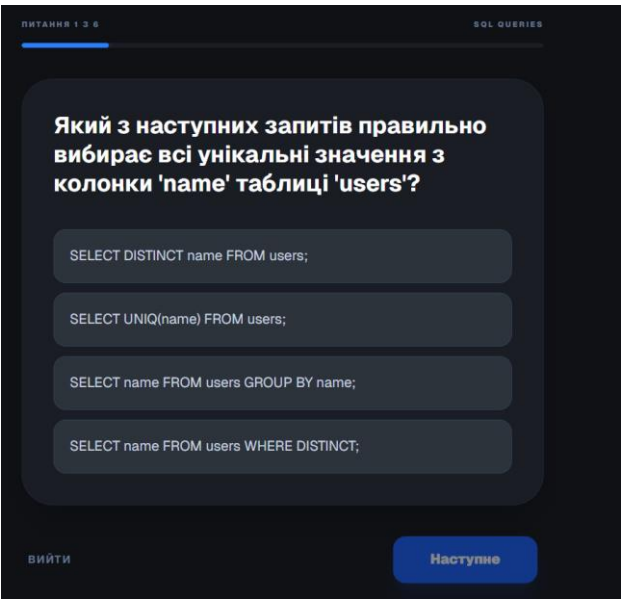


Рисунок 4.11 – Інтерфейс проходження інтерактивного тесту

Вікна роботи з навчальними картками та візуалізація прогресу навчання зображені на рисунках 4.12–4.14.

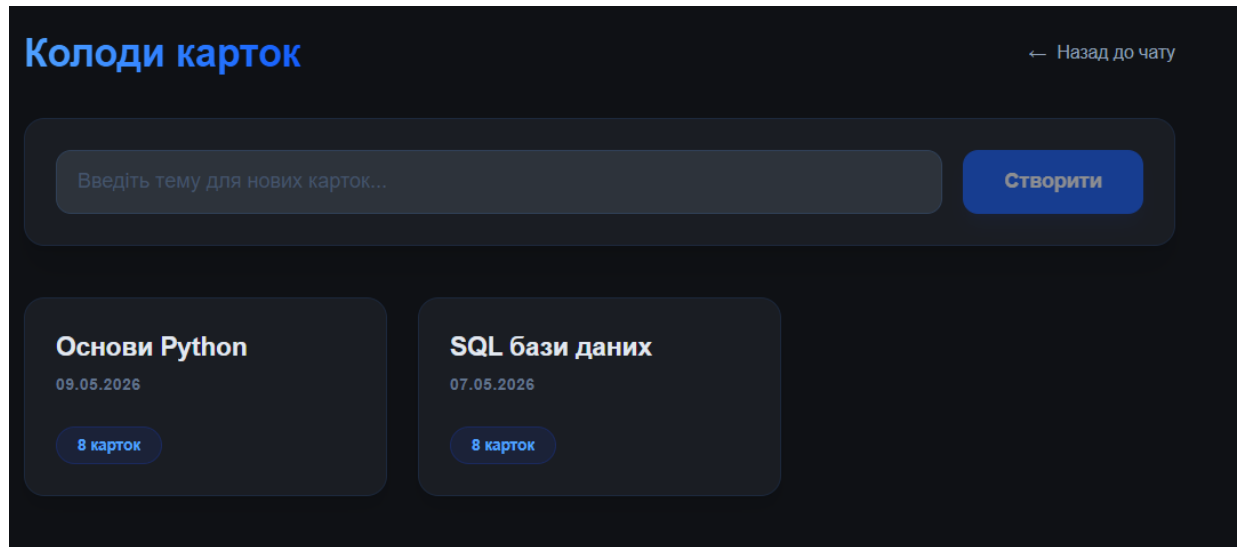


Рисунок 4.12 – Галерея навчальних карток (Flashcards)

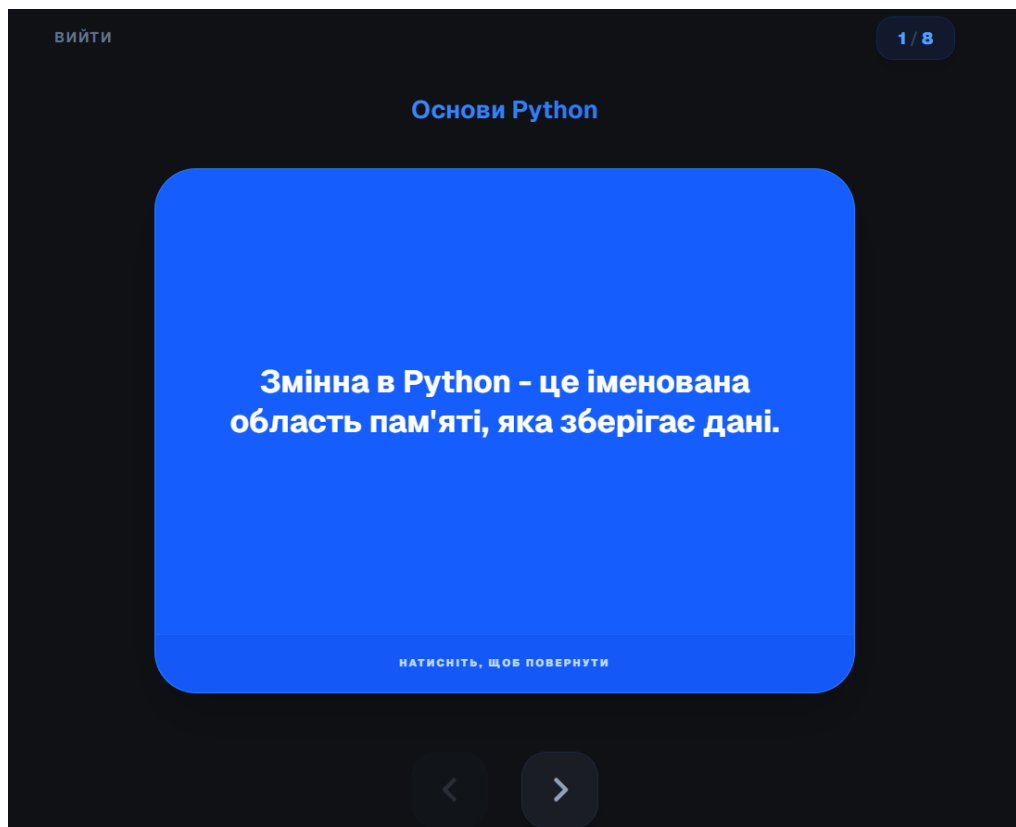


Рисунок 4.13 – Перегляд зворотного боку картки з поясненням

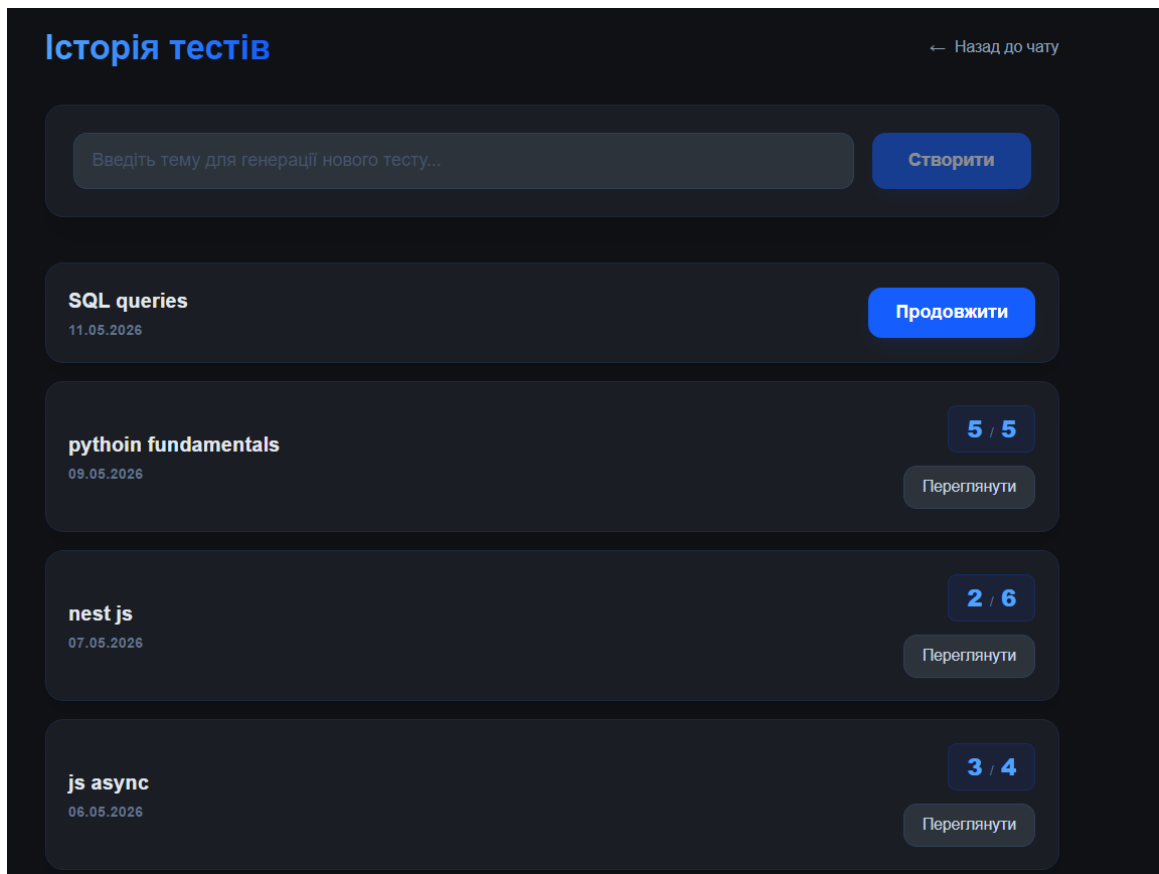


Рисунок 4.14 – Історія успішності та динаміка прогресу користувача

4.4 Тестування роботи програми

4.4.1 Функціональне тестування

Для забезпечення надійності та коректності роботи всіх модулів системи «Edu AI Platform» було проведено комплексне функціональне тестування, яке включало такі методи:

а) модульне тестування:

- 1) перевірка окремих сервісів бекенду (AIService, PdfService, TestService);
- 2) тестування логіки розрахунку балів у тестах та визначення навчального рівня (Junior, Senior);
- 3) валідація DTO-моделей для запитів до ШІ та завантаження документів;

б) інтеграційне тестування:

1) перевірка взаємодії між API та базою даних PostgreSQL через Prisma ORM;

2) тестування механізмів автентифікації та збереження сесій на основі JWT [17];

3) валідація обміну даними між бекендом та зовнішнім API OpenAI (обробка затримок та форматів відповідей) [18];

в) тестування користувацького інтерфейсу:

1) перевірка адаптивності темного інтерфейсу на різних роздільних здатностях;

2) тестування навігації між чатом, бібліотекою PDF та модулем тестів;

3) перевірка анімацій та коректності рендерингу Markdown-відповідей від ІІІ.

Результати функціонального тестування основних модулів наведено в таблиці 4.1.

Таблиця 4.1 – Результати функціонального тестування Edu AI Platform

Функціональний модуль	Тест-кейси	Результат	Виправлені помилки
Реєстрація та профілі	8	Успішно	1
Авторизація	6	Успішно	1
Інтелектуальний чат	15	Успішно	3
Тести та флеш-картки	12	Успішно	2
Обробка документів	10	Успішно	2

4.4.2 Аналіз функціонального тестування

Детальна оцінка результатів підтвердила стабільність основних компонентів «Edu AI Platform». Показники надійності системи відповідають встановленим вимогам для освітніх SaaS-платформ.

Результати тестування (рис. 4.15):

- загальна кількість виконаних тестів: 51;
- успішно пройдено: 47 (~92%);
- виявлено технічних невідповідностей: 11;
- успішно виправлено дефектів: 9.

Результативність тестування (у %)



Рисунок 4.15 – Статистика успішності тестування модулів Edu AI Platform

Основні виправлені дефекти:

- дублювання повідомлень: усунуто помилку повторного надсилання запиту до ШІ при швидкому подвійному натисканні клавіші Enter;
- парсинг PDF: виправлено логіку обробки документів із нестандартним кодуванням тексту, що призводило до пустих відповідей ШІ;
- генерація тестів: виправлено помилку, при якій у квізах іноді генерувалося два однакових варіанти відповіді;
- збереження прогресу: ліквідовано збій у підрахунку загального балу за тест при розриві HTTP-з'єднання під час відправки результатів.

Деталізація критичних помилок наведена в таблиці 4.2.

Таблиця 4.2 – Аналіз виявлених та усунених помилок у Edu AI Platform

Тип помилки	Пріоритет	Час виявлення	Час виправлення	Статус
Помилка обробки контексту PDF	Високий	07.04.2026	09.04.2026	Виправлено
Валідація токенів OpenAI	Критичний	11.04.2026	11.04.2026	Виправлено
Логіка підрахунку результатів	Середній	16.04.2026	18.04.2026	Виправлено
Анімація карток	Низький	22.04.2026	22.04.2026	Виправлено

4.4.3 Тестування продуктивності

Метою було визначення стабільності «Edu AI Platform» під навантаженням та оцінка часу відгуку при взаємодії з ШІ.

Методи оцінки продуктивності:

- а) навантажувальне тестування:
 - 1) моделювання паралельної роботи: перевірка обробки запитів від 30+ користувачів одночасно;
 - 2) швидкість AI-генерації: оцінка затримки відповіді від OpenAI при великих обсягах вхідного контексту (PDF);
 - 3) продуктивність БД: аналіз швидкості вибірки історії повідомлень через Prisma при великому об'ємі даних;
- б) аналіз споживання ресурсів:
 - 1) використання пам'яті: моніторинг RAM при рендерингу довгих діалогів у чаті;

2) навантаження на процесор: оцінка інтенсивності обчислень під час парсингу PDF на стороні сервера;

Графік використання пам'яті клієнтом під час активної сесії навчання (від завантаження PDF до проходження тесту) зображено на рисунку 4.16.

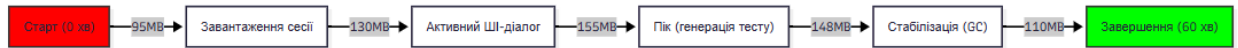


Рисунок 4.16 – Графік використання пам'яті під час роботи застосунку

4.4.4 Аналіз продуктивності

Комплексне оцінювання проводилося на Desktop та Mobile пристроях. Результати демонструють високу швидкість інтерфейсу (рис. 4.17).

Час завантаження системи (LCP):

- робочі станції: 1.0–1.5 секунди;
- мобільні пристрої: 1.4–2.2 секунди.

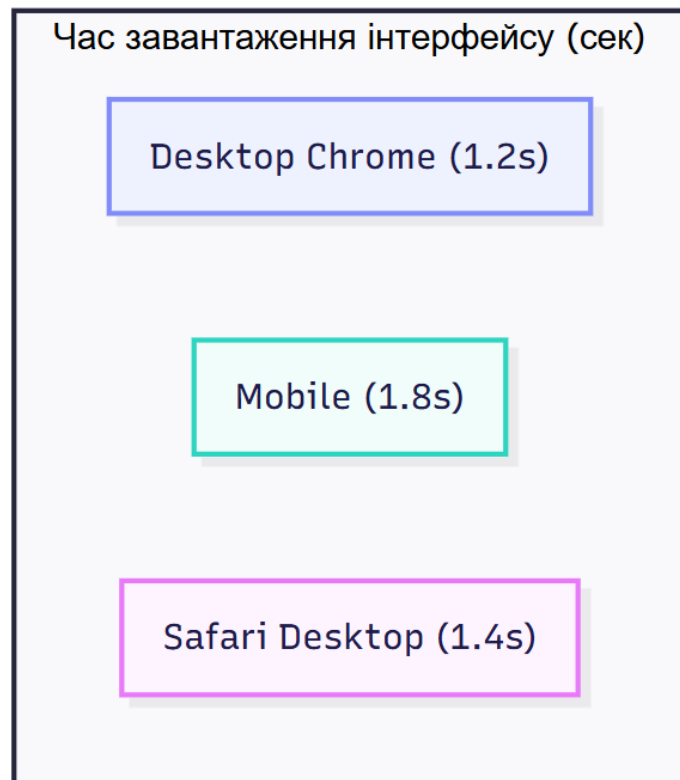


Рисунок 4.17 – Графік часу завантаження інтерфейсу на різних платформах

Результати показників споживання системних ресурсів при різних станах використання системи наведено в таблиці 4.3.

Таблиця 4.3 – Споживання ресурсів клієнтською частиною Edu AI Platform

Режим роботи	Використання пам'яті	CPU	Вплив на батарею
Очікування (вкладка неактивна)	35 MB	0.1%	<0.1%/год
Активний ІІІ-діалог	95 MB	3–6%	~2.0%/год
Завантаження та парсинг PDF	140 MB	10–12%	~3.5%/год

Швидкість виконання ключових операцій:

- надсилання запиту та ініціалізація відповіді ІІІ: <0.6 секунди;
- парсинг та індексація PDF-документа (до 20 сторінок): <1.5 секунди;
- перевертання навчальної картки (Flashcard): <0.1 секунди;
- автоматична генерація тесту на 10 запитань: <2.0 секунди.

Аналіз цих показників свідчить про те, що архітектура Single Page Application (SPA) у поєднанні з асинхронною обробкою запитів та інтеграцією з потужними API забезпечує високу динамічність системи. Це є критично важливим для створення безперервного та комфортного навчального процесу, де користувач отримує інтелектуальну підтримку без відчутних затримок.

4.4.5 Тестування безпеки

Для гарантування конфіденційності навчальних даних було проведено багаторівневе тестування безпеки.

Основні напрямки:

а) автентифікація та сесій:

- 1) перевірка JWT-токенів (термін дії, підпис);
- 2) безпечне збереження паролів (Bcrypt);

б) захист даних:

- 1) тестування DTO-моделей на стійкість до запитів з некоректними даними;
- 2) тестування ізоляції PDF-документів (користувач не може бачити файли інших користувачів через прямі посилання);
- 3) перевірка SQL-безпеки запитів Prisma та валідація Markdown-контенту для запобігання XSS.

Результати тестування безпеки наведено в таблиці 4.4.

Таблиця 4.4 – Результати тестування безпеки

Категорія тестування	Тестів	Вразливості	Усунені вразливості
Автентифікація(JWT, bcrypt)	8	0	0
Валідація та ізоляція PDF	12	1	1
Захист від XSS та SQL-ін'єкцій	10	0	0

4.4.6 Аналіз безпеки

За результатами проведеного тестування було виконано комплексний аналіз безпеки застосунку «Edu AI Platform», який підтвердив високу ефективність впроваджених механізмів захисту конфіденційних даних користувачів.

Особлива увага приділялася надійності алгоритму Bcrypt, що використовується для одностороннього хешування паролів, та безпеці передачі даних через шифрований протокол HTTPS [19]. Аналіз автентифікації на основі JWT (JSON Web Tokens) показав, що токени мають обмежений термін дії та захищені криптографічним підписом сервера, що практично виключає можливість їх підробки або викрадення сесії. Перевірка механізмів ізоляції даних підтвердила, що користувачі мають доступ виключно до власних навчальних матеріалів, історії діалогів та результатів тестування.

Результати автоматизованого сканування та тестування на проникнення наведено в таблиці 4.5.

Таблиця 4.5 – Результати сканування безпеки Planner

Метод перевірки	Кількість вразливостей	Критичні дефекти	Статус усунення
Статичний аналіз коду	3	0	Виправлено
Динамічний аналіз	2	0	Виправлено
Ручний аналіз	4	1	Виправлено

Основні результати аналізу та виправлення:

- критична вразливість: було виявлено можливість отримання доступу до приватного PDF-документа або історії чату іншого користувача через пряму зміну ID в URL-запиті до API. Проблему усунено шляхом впровадження обов'язкової перевірки власності ресурсу у сервісах ChatService та PdfService для кожного запиту;

- статичний аналіз: виявлено використання застарілих версій бібліотек для обробки PDF у package.json, які мали потенційні вразливості. Залежності було оновлено до актуальних стабільних версій;

– динамічний аналіз: виправлено відсутність заголовків безпеки CSP, що значно посилило захист від XSS-атак при рендерингу динамічного контенту, який генерується штучним інтелектом [20].

4.4.7 Аналіз користувацького досвіду

Для оцінки ергономічності «Edu AI Platform» проведено апробацію групою з 5 студентів. Результати наведені на рисунку 4.18.

Показники задоволеності користувацьким інтерфейсом:

- високий рівень задоволеності: 60% (відзначили зручність генерації тестів);
- задовільний рівень: 40% (позитивно оцінили швидкість відповідей ШІ);

Оцінка задоволеності інтерфейсом Edu AI Platform

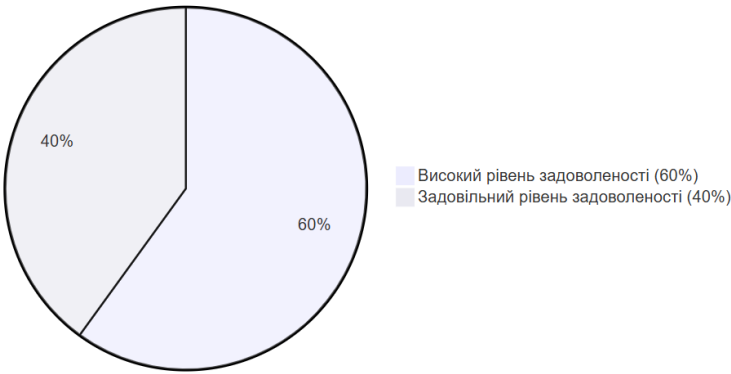


Рисунок 4.18 – Оцінка задоволеності інтерфейсом системи Edu AI Platform

Результати аналізу частоти використання ключових модулів та їхньої суб’єктивної оцінки користувачами наведено в таблиці 4.6.

Таблиця 4.6 – Популярність модулів Edu AI Platform

Функціональний модуль	Частота звернень	Середня оцінка
1	2	3
Навчальний чат з ШІ	98%	4.9/5
Функціональний модуль	Частота звернень	Середня оцінка

Кінець таблиці 4.6

1	2	3
Генератор тестів	88%	4.8/5
Обробка PDF-документів	75%	4.6/5
Флеш-картки	40%	4.5/5

Тестувальники відзначили високу ефективність модуля генерації тестів, що дозволяє значно економити час на самоперевірку знань. Найвищий рейтинг отримав чат за його здатність пояснювати складні концепції зрозумілою мовою.

4.5 Рекомендації щодо використання

На основі результатів апробації та аналізу відгуків користувачів було сформовано перелік рекомендацій для максимально ефективного впровадження системи «Edu AI Platform» в індивідуальний навчальний процес.

Для оптимального налаштування навчального простору рекомендується:

- чітко визначати свій поточний рівень компетенцій (Trainee, Junior, Middle, Senior) у профілі для отримання відповідей III, адаптованих під ваші знання;
- завантажувати структуровані PDF-документи (підручники, методичні вказівки), що дозволяє системі надавати більш точні відповіді з конкретними посиланнями на джерело;

Для ефективного використання функціоналу платформи варто:

- формулювати запити до III максимально конкретно, вказуючи контекст (наприклад: «Поясни це визначення на прикладах з програмування на Java»);
- генерувати інтерактивні тести одразу після опрацювання нової теми або документа для миттєвої перевірки засвоєного матеріалу;

- регулярно використовувати модуль флеш-карток для повторення ключових термінів, що сприяє переходу інформації з короткострокової в довгострокову пам'ять.

Для забезпечення безпеки та цілісності даних слід:

- уникати завантаження PDF-файлів, що містять конфіденційну або персональну інформацію, яка не стосується навчального процесу;
- періодично очищувати історію навчальних сесій, які втратили актуальність, для підтримки лаконічності робочого простору;
- завершувати сеанс роботи через функцію Logout при використанні платформи на спільних або публічних пристроях.

4.6 Плани з розвитку

На основі аналізу отриманих відгуків та дослідження ринку EdTech-рішень було визначено стратегію подальшого розвитку платформи «Edu AI Platform» у кількох ключових напрямках.

У короткостроковій перспективі передбачається:

- розширення підтримки форматів файлів для завантаження (DOCX, PPTX, а також зображення з функцією OCR для розпізнавання рукописних конспектів);
- впровадження системи голосового введення запитів для зручної взаємодії з ІІІ-тьютором у мобільному режимі;
- реалізація PWA-версії (Progressive Web App) для забезпечення стабільного доступу до навчальних матеріалів безпосередньо з робочого столу смартфона [21].

У довгострокових планах основна увага зосереджена на наступних аспектах:

- колективному навчанні: впровадження спільних навчальних кімнат, де декілька користувачів можуть одночасно працювати з одним документом та обговорювати його з ІІІ;

- глибокій аналітиці: розробка системи AI-прогнозування успішності, яка на основі результатів тестів буде пропонувати індивідуальну дорожню карту навчання (Learning Path);

- інтеграції з LMS: реалізація можливості синхронізації з популярними системами управління навчанням (наприклад, Moodle або Google Classroom) для автоматичного імпорту навчальних планів.

Крім того, розробляються рішення для інтеграції з відеоплатформами, що дозволить ШІ аналізувати транскрипти навчальних відео на YouTube та автоматично створювати на їхній основі конспекти та тестові завдання, максимально автоматизуючи процес підготовки студентів.

ВИСНОВКИ

У рамках дипломної кваліфікаційної роботи було розроблено та успішно впроваджено інтелектуальну освітню систему «Edu AI Platform», покликану автоматизувати процес засвоєння знань та підвищити ефективність навчання за допомогою сучасних AI-технологій. Отримані результати підтвердили, що створений програмний продукт є надійним, швидким та інтуїтивно зрозумілим інструментом, здатним адаптуватися під індивідуальні потреби користувача.

Під час виконання роботи було досліджено сучасний стан галузі EdTech та проаналізовано існуючі рішення на базі штучного інтелекту. Виявлено, що більшість доступних сервісів мають обмежені можливості у контексті глибокої роботи з локальними навчальними матеріалами. Це стало ключовим обґрунтуванням для розробки власної платформи, яка поєднує можливості великих мовних моделей з персоналізованим контекстом користувача та інструментами для активного закріплення матеріалу.

Технічна архітектура системи реалізована на базі сучасного Full-stack стеку: Next.js для фронтенду та NestJS для бекенду. Використання Prisma ORM та СУБД PostgreSQL дозволило побудувати гнучку структуру бази знань, а інтеграція з OpenAI API забезпечила високий рівень "інтелектуальності" системи при генерації відповідей та навчального контенту. Обрані технологічні рішення гарантують високу продуктивність застосунку та його стійкість до зростаючих навантажень.

Програмна реалізація охоплює модулі інтерактивного чату, систему аналізу PDF-документів, а також генератори тестів та навчальних карток. Особливістю «Edu AI Platform» є впровадження рівневої системи адаптації контенту (Trainee, Junior, Middle, Senior), що дозволяє подавати інформацію в найбільш прийнятній для користувача формі, забезпечуючи високу якість освітнього процесу.

Етап тестування підтвердив технічну зрілість розробленого продукту. Показник успішного виконання модульних тестів на рівні 92% та середній час відгуку інтерфейсу в межах 1.0–2.2 секунди свідчать про стабільність та високу швидкість роботи платформи. Усунення виявлених дефектів у логіці обробки документів на етапі розробки дозволило досягти високої надійності системи в експлуатаційному режимі.

Захист інформації забезпечено шляхом впровадження багаторівневої системи безпеки, що включає автентифікацію через JWT, криптографічне хешування паролів та суворе розмежування прав доступу до ресурсів на рівні серверної логіки. Тестування на стійкість до вразливостей підтвердило повну конфіденційність навчальних матеріалів та захищеність персональних профілів користувачів.

Аналіз цільової аудиторії показав, що платформа має найбільший потенціал серед студентів (50%) та спеціалістів, які займаються самоосвітою (30%). Це дає можливість чітко фокусувати маркетингову стратегію та подальше розширення функціонала на найбільш активних сегментах ринку.

Визначена стратегія розвитку проекту передбачає впровадження багатомовної підтримки, інтеграцію з відеохостингами для аналізу лекцій та створення мобільного застосунку. Розроблена платформа «Edu AI Platform» не лише автоматизує рутинні аспекти навчання, а й закладає міцний фундамент для створення повноцінної екосистеми адаптивної цифрової освіти.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Artificial Intelligence in Education. UNESCO. URL: <https://www.unesco.org/en/digital-education/artificial-intelligence> (date of access: 02.05.2026).
2. Artificial Intelligence. U.S. Department of Education. URL: <https://www.ed.gov/about/ed-overview/artificial-intelligence> (date of access: 02.05.2026).
3. pytrends. General Mills. URL: <https://pypi.org/project/pytrends> (date of access: 02.05.2026).
4. ChatGPT. OpenAI. URL: <https://chatgpt.com> (date of access: 02.05.2026).
5. Quizlet. Quizlet Inc. URL: <https://quizlet.com> (date of access: 02.05.2026).
6. Notion AI. Notion Labs, Inc. URL: <https://www.notion.so> (date of access: 02.05.2026).
7. Tailwind CSS Documentation. Tailwind Labs, Inc. URL: <https://tailwindcss.com/docs> (date of access: 02.05.2026).
8. TypeScript. Microsoft Corporation. URL: <https://www.typescriptlang.org> (date of access: 02.05.2026).
9. TypeScript Handbook. Microsoft Corporation. URL: <https://www.typescriptlang.org/docs/handbook> (date of access: 02.05.2026).
10. Cherny B. Programming TypeScript. Sebastopol : O'Reilly Media, 2019. 352 p.
11. NestJS Documentation. Trilon. URL: <https://docs.nestjs.com> (date of access: 02.05.2026).
12. Next.js Documentation. Vercel. URL: <https://nextjs.org/docs> (date of access: 02.05.2026).
13. Visual Studio Code. Microsoft Corporation. URL: <https://code.visualstudio.com> (date of access: 02.05.2026).

14. PostgreSQL Documentation. The PostgreSQL Global Development Group. URL: <https://www.postgresql.org/docs> (date of access: 02.05.2026).
15. PostgreSQL database connector. Prisma Data, Inc. URL: <https://www.prisma.io/docs/orm/core-concepts/supported-databases/postgresql> (date of access: 02.05.2026).
16. Fontaine D. The Art of PostgreSQL. Lyon : Lulu Press, 2019. 430 p.
17. JSON Web Tokens. Auth0 Inc. URL: <https://jwt.io/introduction> (date of access: 02.05.2026).
18. OpenAI API Documentation. OpenAI. URL: <https://platform.openai.com/docs> (date of access: 02.05.2026).
19. bcrypt. npm, Inc. URL: <https://www.npmjs.com/package/bcrypt> (date of access: 02.05.2026).
20. Content Security Policy (CSP). Mozilla Developer Network. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP> (date of access: 02.05.2026).
21. Progressive Web Apps. Google Developers. URL: <https://web.dev/progressive-web-apps> (date of access: 02.05.2026).

ДОДАТОК А
Технічне завдання

Вступ

Вебплатформа «Edu AI Platform» призначена для автоматизації освітніх процесів та персоналізації навчання за допомогою технологій штучного інтелекту. Вона розроблена для використання студентами, викладачами та особами, що займаються самоосвітою, які потребують інтелектуального інструменту для аналізу навчальних матеріалів, генерації перевірочних тестів та отримання миттєвих роз'яснень складних тем. Область застосування охоплює сферу цифрової освіти (EdTech), а об'єктом використання є кросплатформний вебінтерфейс, адаптований для десктопних та мобільних пристроїв.

A.1 Підстави до розробки

Підставою для розробки є завдання на дипломну роботу, видане 20 квітня 2026 року. Найменування теми розробки – «Програмна реалізація застосунку інтелектуального чату для освітніх цілей».

A.2 Призначення розробки

Застосунок розроблений для автоматизації циклу засвоєння знань, включаючи інтелектуальну підтримку в режимі діалогу та автоматичне створення навчального контенту (тестів, карток). Система забезпечує глибокий аналіз PDF-документів користувача, адаптує складність відповідей під рівень знань учня та надає інструменти для ефективного закріплення матеріалу. Експлуатаційне призначення продукту полягає у спрощенні самоосвіти та підвищенні якості підготовки до іспитів через персоналізований AI-супровід.

А.3 Вимоги до програми чи програмного виробу

А.3.1 Вимоги до функціональних характеристик

Вебплатформа «Edu AI Platform» розроблена для вирішення широкого спектра завдань, пов'язаних із цифровим навчанням. Система забезпечує:

- безпечну автентифікацію через JWT-токени для доступу до персональної історії навчання;
- ведення діалогу з AI-тьютором, який здатний пояснювати концепції, розв'язувати задачі та структурувати інформацію;
- завантаження та індексацію PDF-документів для використання їх як контексту для відповідей ШІ (RAG-підхід);
- автоматичну генерацію інтерактивних тестів на основі опрацьованого матеріалу з миттєвою перевіркою відповідей;
- створення наборів навчальних карток (Flashcards) для інтервального повторення ключових термінів;
- адаптивне налаштування рівня складності (Trainee, Junior, Middle, Senior) для персоналізації відповідей ШІ.

Вхідними даними є текстові запити користувача та навчальні документи у форматі PDF. Вихідними результатами є згенеровані відповіді ШІ, структуровані тести, навчальні картки та звіти про прогрес навчання.

А.3.2 Вимоги до надійності

Оскільки «Edu AI Platform» є клієнт-серверним застосунком, необхідно забезпечити стабільне підключення до Інтернету для комунікації з OpenAI API та сервером. Використання NestJS та Next.js гарантує швидку обробку запитів та сумісність із сучасними браузерами. Безпека даних підтримується завдяки впровадженню JWT-авторизації та шифруванню паролів (Bcrypt). У разі виникнення помилок при взаємодії з ШІ, система має передбачати механізми обробки виключень та інформування користувача про необхідність

повторного запиту. Валідація вхідних даних на рівні DTO запобігає передачі некоректної інформації до бази даних або API.

A.3.3 Умови експлуатації

Застосунок є веборієнтованим, тому ключовою вимогою є наявність сучасного браузера (Chrome, Firefox, Safari) та інтернет-з'єднання. Рівень підготовки користувача передбачає базові навички роботи з вебінтерфейсами. Система розроблена за принципами інтуїтивно зрозумілого дизайну (UI/UX) у темній колірній гамі, що мінімізує втому очей під час тривалого навчання.

A.3.4 Вимоги до складу параметрів технічних засобів

Для забезпечення стабільної роботи вебплатформи «Edu AI Platform», пристрій користувача повинен задовільняти наступні мінімальні технічні вимоги:

- операційна система: Windows 10/11, macOS, Linux або актуальні мобільні ОС;
- ПЗ: сучасний веббраузер із підтримкою стандарту ES6+;
- оперативна пам'ять: мінімум 2 ГБ;
- мережеве з'єднання: обов'язкове підключення до інтернету (мінімум 1 Мбіт/с) для стабільної роботи з ШІ-моделями.

A.3.5 Вимоги до інформаційної й програмної сумісності

Обмін даними здійснюється у форматі JSON. Програмна реалізація базується на TypeScript. Система має наступний стек технологій:

- Frontend: Next.js (React);
- Backend: NestJS;
- СУБД: PostgreSQL (через Prisma ORM);
- AI Engine: OpenAI API (GPT-моделі).

Для миттєвого відгуку інтерфейсу використовуються інструменти React Query. Доступ до даних ізольований на рівні Row-Level логіки в Prisma, що гарантує конфіденційність документів кожного користувача.

А.3.6 Вимоги до маркування й упакування

Дистрибуція «Edu AI Platform» здійснюється через глобальну мережу за допомогою унікальної веб адреси, що забезпечує миттєвий доступ до освітнього інструментарію без необхідності локального встановлення. Програмний продукт розповсюджується за моделлю SaaS, що базується на хмарній інфраструктурі, проте архітектура також передбачає можливість автономного розгортання на власних серверах користувача (Self-hosting) із застосуванням Docker-контейнеризації для максимального контролю над приватними даними.

Оскільки платформа є інтелектуальним програмним активом, будь-які вимоги до фізичної упаковки, матеріальних етикеток або логістичного супроводу не висувуються. Всі процеси передачі прав, конфігурації профілів та регулярного оновлення алгоритмів штучного інтелекту відбуваються у цифровому форматі. Це дозволяє оперативно впроваджувати нові методи аналізу знань та забезпечувати користувачів актуальними інструментами без переривання їхньої освітньої траєкторії.

А.3.7 Вимоги до транспортування й збереження

Враховуючи цифрову природу системи та хмарну стратегію її розгортання, будь-які вимоги щодо фізичного переміщення, логістичних операцій чи спеціальних умов зберігання матеріальних компонентів є неактуальними. Отримання доступу до системи, а також інсталяція патчів безпеки та розширення AI-модулів виконуються миттєво через мережеву інфраструктуру. Такий підхід повністю нівелює потребу у фізичному

пакуванні чи механічному транспортуванні, гарантуючи екологічність рішення та безперебійний доступ до освітніх ресурсів із будь-якої точки світу.

A.4 Вимоги до програмної документації

Комплект документації до «Edu AI Platform» включає це технічне завдання та розгорнуту пояснювальну записку. Ці матеріали детально описують архітектурний стек (Next.js/NestJS), функціональні компоненти (інтелектуальний помічник, модуль аналізу PDF, генератор квізів та карток), а також встановлюють чіткі параметри для середовища розробки та експлуатації. Окремий акцент зроблено на регламентах налаштування AI-промптів та протоколах взаємодії з OpenAI API.

A.5 Техніко–економічні показники

Розроблена система має значний потенціал для трансформації підходів до самоосвіти та академічної підготовки. Прогнозується, що впровадження платформи дозволить оптимізувати час на опрацювання наукової літератури, створення конспектів та самоконтроль на 40-60%.

Доступність базових інтелектуальних функцій робить «Edu AI Platform» привабливим рішенням для студентів та незалежних дослідників. Економічна ефективність полягає у наданні персоналізованого AI-тьюторингу без необхідності фінансових витрат на репетиторів чи дорогі курси. Це забезпечує суттєву економію інтелектуальних зусиль та часу, дозволяючи зосередитися на розвитку професійних компетенцій та аналітичних навичок.

A.6 Стадії й етапи розробки

Створення програмного продукту розподілено на наступні стадії:

- формулювання концепції та вимог до інтелектуальних функцій (1 тиждень);

- дослідження предметної області та проєктування RAG-алгоритмів (2 тижні);
- проєктування системної архітектури та схеми даних у PostgreSQL (1 тиждень);
- безпосередня розробка клієнтського та серверного програмного коду (2 тижні);
- верифікація відповідей ШІ та тестування швидкості обробки документів (1 тиждень);
- фіналізація документації та підготовка посібників для користувачів (1 тиждень).

A.7 Порядок контролю та приймання

Підтвердження функціональності «Edu AI Platform» передбачає багаторівневий цикл випробувань. Модульне тестування (Unit testing) гарантує стабільність ізольованих сервісів, зокрема логіки авторизації та генерації тестів. Інтеграційні випробування спрямовані на перевірку зв'язку між фронтендом та бекендом, з особливим акцентом на коректності запитів до хмарних AI-моделей та точності вилучення даних із PDF. Окремо аналізується інтерфейс користувача (UI/UX) на предмет зручності взаємодії та коректності роботи компонентів на базі shadcn/ui.

Працездатність системи перевіряється в сучасних браузерях як на десктопних, так і на мобільних пристроях. Особлива увага приділяється валідації генерованого навчального контенту та точності автоматичного підрахунку балів. Після завершення всіх етапів перевірки та підтвердження повної відповідності вимогам безпеки та цілісності даних, система вважається готовою до введення в експлуатацію.

ДОДАТОК Б
Текст програми

Б.1 Файл auth.dto.ts

У цьому файлі знаходиться валідація вхідних даних користувача (email та пароль) за допомогою декораторів class-validator для забезпечення безпеки та коректності формату даних при реєстрації:

```
import { IsEmail, IsString, MinLength } from 'class-validator';

export class AuthDto {
  @IsEmail({}, { message: 'Некоректний формат Email' })
  email: string;

  @IsString()
  @MinLength(6, { message: 'Пароль має бути не менше 6 символів' })
  password: string;
}
```

Б.2 Файл auth.service.ts

У цьому файлі знаходиться реалізація логіки автентифікації, хешування паролів за допомогою bcrypt та генерації JWT-токенів для створення захищених сесій користувачів:

```
import { Injectable, UnauthorizedException, BadRequestException } from '@nestjs/common';
import { JwtService } from '@nestjs/jwt';
import { PrismaService } from '../prisma/prisma.service';
import * as bcrypt from 'bcrypt';

@Injectable()
export class AuthService {
  constructor(
    private prisma: PrismaService,
    private jwtService: JwtService
  ) {}

  async register(email: string, pass: string) {
    const existing = await this.prisma.user.findUnique({ where: { email } });
    if (existing) {
      throw new BadRequestException('User already exists');
    }

    const hashedPassword = await bcrypt.hash(pass, 10);
    const user = await this.prisma.user.create({
      data: {
        email,
        password: hashedPassword,
      },
    });
  }
}
```

```

    });

    return { id: user.id, email: user.email };
  }

  async login(email: string, pass: string) {
    const user = await this.prisma.user.findUnique({ where: { email } });
    if (!user) {
      throw new UnauthorizedException('Invalid credentials');
    }

    const isMatch = await bcrypt.compare(pass, user.password);
    if (!isMatch) {
      throw new UnauthorizedException('Invalid credentials');
    }

    const payload = { email: user.email, userId: user.id };
    return {
      access_token: this.jwtService.sign(payload),
    };
  }
}

```

Б.3 Файл ai.service.ts

Ядро системи, що відповідає за взаємодію з OpenAI API. Тут реалізовано логіку формування промптів, обробку контексту та отримання інтелектуальних відповідей у режимі реального часу:

```

import { Injectable } from '@nestjs/common';
import { ConfigService } from '@nestjs/config';
import OpenAI from 'openai';

@Injectable()
export class AiService {
  private openai: OpenAI;

  constructor(private readonly configService: ConfigService) {
    this.openai = new OpenAI({
      apiKey: this.configService.get<string>('OPENAI_API_KEY'),
    });
  }

  async generateResponse(prompt: string): Promise<string> {
    try {
      const response = await this.openai.chat.completions.create({
        model: 'gpt-4o-mini',
        messages: [{ role: 'user', content: prompt }],
      });
    }
  }
}

```

```

    return response.choices[0]?.message?.content || 'AI service temporarily unavailable';
  } catch (error) {
    console.error('OpenAI Error:', error);
    return 'AI service temporarily unavailable';
  }
}

async generateEmbedding(text: string): Promise<number[]> {
  try {
    const response = await this.openai.embeddings.create({
      model: 'text-embedding-3-small',
      input: text,
    });

    return response.data[0].embedding;
  } catch (error) {
    console.error('OpenAI Embedding Error:', error);
    throw error;
  }
}
}

```

Б.4 Файл pdf.service.ts

У цьому файлі знаходиться модуль для роботи з документами, що забезпечує зчитування тексту з PDF-файлів, його фрагментацію (chunking) та підготовку даних для використання мовною моделлю:

```

import { Injectable } from '@nestjs/common';
import { PrismaService } from '../prisma/prisma.service';
import { AiService } from '../ai/ai.service';
// @ts-ignore
import * as pdf from 'pdf-parse';

@Injectable()
export class PdfService {
  constructor(
    private readonly prisma: PrismaService,
    private readonly aiService: AiService
  ) {}

  async uploadPdf(userId: number, filename: string, buffer: Buffer) {
    // @ts-ignore
    const parser = new pdf.PDFParse({ data: buffer });
    const result = await parser.getText();
    const fullContent = result.text;

    const doc = await this.prisma.pdfDocument.create({
      data: {
        userId,

```

```

    filename,
    content: fullContent.substring(0, 5000), // Keep preview
  },
});

// Chunking logic
const chunks = this.chunkText(fullContent, 1000, 200);

for (const chunkContent of chunks) {
  const embedding = await this.aiService.generateEmbedding(chunkContent);
  await this.prisma.pdfChunk.create({
    data: {
      pdfDocumentId: doc.id,
      content: chunkContent,
      embedding: embedding as any,
    },
  });
}

return doc;
}

private chunkText(text: string, size: number, overlap: number): string[] {
  const chunks: string[] = [];
  let start = 0;
  while (start < text.length) {
    const end = Math.min(start + size, text.length);
    chunks.push(text.substring(start, end));
    start += size - overlap;
    if (start >= text.length - overlap && text.length > size) break;
  }
  return chunks;
}

async searchChunks(pdfId: number, queryEmbedding: number[], topK = 5): Promise<string[]>
{
  const chunks = await this.prisma.pdfChunk.findMany({
    where: { pdfDocumentId: pdfId },
  });

  const scores = chunks.map((chunk: any) => {
    const embedding = chunk.embedding as number[];
    const similarity = this.cosineSimilarity(queryEmbedding, embedding);
    return { content: chunk.content, similarity };
  });

  return scores
    .sort((a, b) => b.similarity - a.similarity)
    .slice(0, topK)
    .map(s => s.content);
}

```

```

private cosineSimilarity(vecA: number[], vecB: number[]): number {
  if (!vecA || !vecB) return 0;
  let dotProduct = 0;
  let normA = 0;
  let normB = 0;
  for (let i = 0; i < vecA.length; i++) {
    dotProduct += vecA[i] * vecB[i];
    normA += vecA[i] * vecA[i];
    normB += vecB[i] * vecB[i];
  }
  return dotProduct / (Math.sqrt(normA) * Math.sqrt(normB));
}

```

```

async getPdfs(userId: number) {
  return this.prisma.pdfDocument.findMany({
    where: { userId },
    select: { id: true, filename: true, createdAt: true },
  });
}

```

```

async getPdfContent(userId: number, pdfId: number) {
  const doc = await this.prisma.pdfDocument.findUnique({
    where: { id: pdfId },
  });
  if (!doc || doc.userId !== userId) return null;
  return doc.content;
}
}

```

Б.5 Файл test.service.ts

У цьому файлі знаходиться сервіс автоматичної генерації тестових запитань. Містить алгоритми перетворення навчального матеріалу у структурований JSON-формат із варіантами відповідей та поясненнями:

```

import { Injectable, ForbiddenException, NotFoundException } from '@nestjs/common';
import { AiService } from '../ai/ai.service';
import { PrismaService } from '../prisma/prisma.service';

```

```

@Injectable()
export class TestService {
  constructor(
    private readonly aiService: AiService,
    private readonly prisma: PrismaService
  ) {}

```

```

  async startTest(userId: number, topic: string, sessionId?: number, context?: string, level?:
string, questionCount: number = 5) {
    const baseContext = context ? `Використовуючи наданий матеріал:\n${context}\n\n` : "";
    const levelInstr = level ? `Рівень складності: ${level}.` : "";

```

```
const prompt = `You are a professional test generator.
${baseContext}
Difficulty level: ${level}.
```

```
Generate exactly ${questionCount} multiple-choice questions on the topic: "${topic}".
```

IMPORTANT: The questions, options, and explanations MUST BE in the same language as the topic or context provided (likely Ukrainian or Russian).

```
If context material is provided, create questions STRICTLY based on it.
```

```
Return ONLY valid JSON:
```

```
{
  "questions": [
    {
      "question": "...",
      "options": ["...", "...", "...", "..."],
      "correct": "exact text of the correct option",
      "explanation": "..."
    }
  ]
};
```

```
let aiResponse = "";
let parsedData: any = null;
let attempts = 0;
```

```
while (attempts < 2) {
  aiResponse = await this.aiService.generateResponse(prompt);
  try {
    const cleanJson = aiResponse.replace(/```json/g, "").replace(/```/g, "").trim();
    parsedData = JSON.parse(cleanJson);
    if (parsedData && parsedData.questions && Array.isArray(parsedData.questions)) {
      break;
    }
  } catch (e) {
    attempts++;
  }
}
```

```
if (!parsedData || !parsedData.questions) {
  throw new Error('Failed to generate test questions. AI returned malformed data.');
```

```
}

const testSession = await this.prisma.testSession.create({
  data: {
    userId,
    sessionId,
    topic,
    total: parsedData.questions.length,
    status: 'active',
```

```

    }
  });

  for (const q of parsedData.questions) {
    await this.prisma.testQuestion.create({
      data: {
        testSessionId: testSession.id,
        question: q.question,
        options: q.options,
        correct: q.correct,
        explanation: q.explanation,
      }
    });
  }
}

const questions = await this.prisma.testQuestion.findMany({
  where: { testSessionId: testSession.id },
  select: {
    id: true,
    question: true,
    options: true,
  }
});

return { testSessionId: testSession.id, questions };
}

async getTest(userId: number, testSessionId: number) {
  const session = await this.prisma.testSession.findUnique({
    where: { id: testSessionId },
    include: {
      questions: {
        select: {
          id: true,
          question: true,
          options: true,
          correct: true,
          explanation: true
        }
      }
    },
  });

  if (!session || session.userId !== userId) {
    throw new ForbiddenException('Access denied');
  }

  if (session.status !== 'finished') {
    // Strip correct answers for active tests
    session.questions = session.questions.map(q => {
      const { correct, explanation, ...rest } = q as any;
      return rest;
    });
  }
}

```

```

    }) as any;
    return session;
  }

  const answers = await this.prisma.testAnswer.findMany({
    where: { question: { testSessionId: session.id } }
  });

  const results = session.questions.map(q => {
    const ans = answers.find(a => a.questionId === q.id);
    return {
      questionId: q.id,
      userAnswer: ans ? ans.answer : null,
      correctAnswer: q.correct,
      isCorrect: ans ? ans.isCorrect : false,
      explanation: q.explanation
    };
  });

  return { ...session, results };
}

async submitTest(userId: number, testSessionId: number, userAnswers: { questionId: number,
answer: string }[]) {
  const session = await this.prisma.testSession.findUnique({
    where: { id: testSessionId },
    include: { questions: true },
  });

  if (!session || session.userId !== userId) {
    throw new ForbiddenException('Access denied');
  }

  if (session.status === 'finished') {
    throw new ForbiddenException('Test already submitted');
  }

  let score = 0;
  const results = [];

  for (const ans of userAnswers) {
    const question = session.questions.find(q => q.id === ans.questionId);
    if (!question) continue;

    const isCorrect = question.correct === ans.answer;
    if (isCorrect) score++;

    await this.prisma.testAnswer.create({
      data: {
        questionId: question.id,
        answer: ans.answer,
        isCorrect,

```

```

    }
  });

  results.push({
    questionId: question.id,
    userAnswer: ans.answer,
    correctAnswer: question.correct,
    isCorrect,
    explanation: question.explanation,
  });
}

await this.prisma.testSession.update({
  where: { id: testSessionId },
  data: {
    score,
    status: 'finished',
  }
});

return { score, total: session.total, results };
}

async getHistory(userId: number) {
  return this.prisma.testSession.findMany({
    where: { userId },
    orderBy: { createdAt: 'desc' },
  });
}
}

```

Б.6 Файл test/[testsessionId]/page.tsx

У цьому файлі знаходиться логіку динамічного відображення запитань, управління станом відповідей користувача, візуалізацію прогресу та інтерактивний вивід результатів тестування з використанням Tailwind CSS:

```

"use client";
import { useEffect, useState, use } from 'react';
import { useRouter } from 'next/navigation';
import { testApi } from '../api/test';
import Link from 'next/link';

export default function TestRoomClient({ params }: { params: Promise<{ testSessionId: string }> }) {
  const unwrappedParams = use(params);
  const testSessionId = unwrappedParams.testSessionId;
  const router = useRouter();
  const [session, setSession] = useState<any>(null);
  const [loading, setLoading] = useState(true);

```

```

const [currentIdx, setCurrentIdx] = useState(0);
const [answers, setAnswers] = useState<{questionId: number, answer: string}[]>([]);
const [results, setResults] = useState<any>(null);
const [submitting, setSubmitting] = useState(false);

useEffect(() => {
  testApi.getTest(Number(testSessionId))
    .then(res => {
      setSession(res.data);
      if (res.data.status === 'finished') {
        setResults({ score: res.data.score, total: res.data.total, results: res.data.results });
      }
    })
    .catch(console.error)
    .finally(() => setLoading(false));
}, [testSessionId]);

if (loading) return <div className="p-8 text-center text-slate-500 bg-[#0f1115] flex h-screen items-center justify-center">Завантаження тесту...</div>;
if (!session) return <div className="p-8 text-center text-red-500 bg-[#0f1115] flex h-screen items-center justify-center">Тест не знайдено.</div>;
if (session.status === 'finished' && !results) {
  return <div className="p-8 text-center bg-[#0f1115] flex h-screen items-center justify-center text-slate-400">Завантаження результатів...</div>;
}

if (results) {
  return (
    <div className="min-h-screen bg-[#0f1115] text-slate-200 p-6 py-12">
      <div className="max-w-3xl mx-auto">
        <h1 className="text-4xl font-black mb-10 text-center text-blue-500 animate-score-reveal">
          Результат: {results.score} <span className="text-slate-600"></span> {results.total}
        </h1>
        <div className="space-y-6">
          {results.results.map((r: any, i: number) => (
            <div key={r.questionId} className={`p-6 rounded-3xl border shadow-xl ${r.isCorrect ? 'bg-green-900/10 border-green-800/30' : 'bg-red-900/10 border-red-800/30'}`}>
              <div className="flex items-start gap-4 mb-4">
                <span className={`w-8 h-8 rounded-full flex items-center justify-center font-bold shrink-0 ${r.isCorrect ? 'bg-green-600 text-white' : 'bg-red-600 text-white'}`}>
                  {i + 1}
                </span>
                <p className="font-bold text-lg leading-tight text-slate-100">{session.questions.find((q: any) => q.id === r.questionId)?.question}</p>
              </div>
              <div className="ml-12 space-y-2">
                <p className="text-sm">Ваша відповідь: <span className={`font-bold ${r.isCorrect ? 'text-green-400' : 'text-red-400'}`}>{r.userAnswer}</span></p>
                <{!r.isCorrect && <p className="text-sm text-green-400 font-bold">Правильно: {r.correctAnswer}</p>}

```

```

<div className="mt-6 text-sm text-slate-400 bg-slate-900/50 p-4 rounded-2xl border
border-slate-800 leading-relaxed italic">
  <span className="font-bold text-slate-500 block mb-1 not-italic uppercase
tracking-widest text-[10px]">Пояснення:</span>
    {r.explanation}
  </div>
</div>
</div>
))}
</div>
<div className="mt-12 text-center flex flex-col sm:flex-row justify-center gap-4">
  <Link href="/test/history" className="px-8 py-4 bg-[#1a1d23] text-slate-300 border
border-slate-800 rounded-2xl hover:bg-slate-800 transition-all font-bold shadow-
lg">Історія</Link>
  <Link href="/chat" className="px-8 py-4 bg-blue-600 text-white rounded-2xl hover:bg-
blue-700 transition-all font-bold shadow-lg shadow-blue-900/20">До чату</Link>
</div>
</div>
</div>
);
}

```

```
const question = session.questions[currentIdx];
const currentAnswer = answers.find(a => a.questionId === question.id)?.answer;
```

```
const handleSelect = (option: string) => {
  const newAnswers = answers.filter(a => a.questionId !== question.id);
  newAnswers.push({ questionId: question.id, answer: option });
  setAnswers(newAnswers);
};
```

```
const handleNext = () => {
  if (currentIdx < session.questions.length - 1) {
    setCurrentIdx(currentIdx + 1);
  } else {
    setSubmitting(true);
    testApi.submitTest(Number(testSessionId), answers)
      .then(res => setResults(res.data))
      .catch(console.error)
      .finally(() => setSubmitting(false));
  }
};
```

```

return (
  <div className="min-h-screen bg-[#0f1115] text-slate-200 p-6 flex flex-col items-center justify-center font-sans">
    <div className="w-full max-w-2xl">
      <div className="mb-10">
        <div className="flex justify-between text-[10px] text-slate-500 mb-4 font-black uppercase tracking-[0.2em]">
          <span>Питання {currentIdx + 1} з {session.questions.length}</span>
          <span>{session.topic}</span>

```

```

    </div>
    <div className="w-full bg-[#1a1d23] rounded-full h-1.5 overflow-hidden">
      <div className="bg-blue-500 h-full rounded-full transition-all duration-700 ease-out
shadow-[0_0_8px_rgba(59,130,246,0.5)]" style={{ width: `${((currentIdx + 1) /
session.questions.length) * 100}%` }}></div>
    </div>
  </div>

  <div className="bg-[#1a1d23] p-8 sm:p-10 rounded-[2.5rem] shadow-2xl border border-
slate-800 mb-10 transition-all hover:border-slate-700">
    <h2 className="text-2xl sm:text-3xl font-bold mb-10 leading-tight text-
white">{question.question}</h2>
    <div className="space-y-3">
      {question.options.map((opt: string, i: number) => (
        <button
          key={i}
          onClick={() => handleSelect(opt)}
          className={`w-full text-left p-5 rounded-2xl border transition-all duration-300 text-lg
group relative overflow-hidden ${currentAnswer === opt ? 'bg-blue-600 border-blue-500 text-
white shadow-lg shadow-blue-900/30' : 'bg-[#2d333b] border-slate-700 text-slate-300
hover:border-slate-500 hover:bg-slate-800`} `}
        >
          <span className="relative z-10">{opt}</span>
        </button>
      ))}
    </div>
  </div>

  <div className="flex justify-between items-center w-full px-2">
    <Link href="/chat" className="text-sm font-bold text-slate-500 hover:text-slate-300
transition-colors uppercase tracking-widest">Вийти</Link>
    <button
      onClick={handleNext}
      disabled={!currentAnswer || submitting}
      className="px-12 py-4 bg-blue-600 text-white rounded-2xl font-black text-lg hover:bg-
blue-700 disabled:opacity-50 transition-all active:scale-95 shadow-xl shadow-blue-900/30"
    >
      {submitting ? 'Надсилення...' : (currentIdx === session.questions.length - 1 ?
'Завершити' : 'Наступне')}
    </button>
  </div>
</div>
</div>
);
}

```

Б.7 Файл `flashcard.service.ts`

У цьому файлі знаходиться логіка створення та управління навчальними картками. Файл містить методи для генерації фронтальної та зворотної сторін карток для ефективного вивчення термінів:

```
'use client';
import { Injectable, ForbiddenException } from '@nestjs/common';
import { AiService } from '../ai/ai.service';
import { PrismaService } from '../prisma/prisma.service';

@Injectable()
export class FlashcardService {
  constructor(
    private readonly aiService: AiService,
    private readonly prisma: PrismaService
  ) {}

  async generateDeck(userId: number, topic: string, context?: string, level?: string) {
    const baseContext = context ? `Використовуючи наданий матеріал:\n${context}\n\n` : "";
    const levelInstr = level ? `Рівень складності: ${level}.` : "";

    const prompt = `You are an educational assistant.
    ${baseContext}
    Difficulty level: ${level}.

    Generate exactly 8 memory flashcards on the topic: "${topic}"`.

    IMPORTANT: The deck title, and all content on the front and back of the cards MUST BE in
    the same language as the topic or context provided (likely Ukrainian or Russian).

    If context material is provided, create flashcards STRICTLY based on it.

    Return ONLY valid JSON:
    {
      "deckTitle": "...",
      "cards": [
        { "front": "...", "back": "..." }
      ]
    }`;

    let aiResponse = "";
    let parsedData: any = null;
    let attempts = 0;

    while (attempts < 2) {
      aiResponse = await this.aiService.generateResponse(prompt);
      try {
        const cleanJson = aiResponse.replace(/```json/g, "").replace(/```/g, "").trim();
        parsedData = JSON.parse(cleanJson);
      }
    }
  }
}
```

```

    if (parsedData && parsedData.cards && Array.isArray(parsedData.cards)) {
      break;
    }
  } catch (e) {
    attempts++;
  }
}

if (!parsedData || !parsedData.cards) {
  throw new Error('Failed to generate flashcards. AI returned malformed data.');
```

```

}

const deck = await this.prisma.flashcardDeck.create({
  data: {
    userId,
    title: parsedData.deckTitle || topic,
  }
});

for (const card of parsedData.cards) {
  await this.prisma.flashcard.create({
    data: {
      deckId: deck.id,
      front: card.front,
      back: card.back,
    }
  });
}

const cards = await this.prisma.flashcard.findMany({
  where: { deckId: deck.id }
});

return { deckId: deck.id, title: deck.title, cards };
}

async getDecks(userId: number) {
  return this.prisma.flashcardDeck.findMany({
    where: { userId },
    orderBy: { createdAt: 'desc' },
    include: {
      _count: {
        select: { cards: true }
      }
    }
  });
}

async getDeck(userId: number, deckId: number) {
  const deck = await this.prisma.flashcardDeck.findUnique({
    where: { id: deckId },
    include: { cards: true },
  });
}

```

```
});  
  
if (!deck || deck.userId !== userId) {  
    throw new ForbiddenException('Access denied');  
}  
  
return deck;  
}  
}
```

ДОДАТОК В
Слайди презентації

ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ІНТЕЛЕКТУАЛЬНОГО ЧАТУ ДЛЯ ОСВІТНІХ ЦІЛЕЙ

Виконав
студент групи КНТ-112

Igor СТЕПАНОВ

Керівник
к.т.н. доцент

Євгеній ГОФМАН

Рисунок В.1 – Слайд №1

Мета та задачі дипломної роботи

Мета роботи – розробка вебсистеми, яка зменшує час засвоєння навчального матеріалу користувачем за рахунок автоматичної генерації структурованих конспектів, тестових завдань та флеш-карток із текстових і PDF-джерел із урахуванням рівня підготовки користувача.

Об'єкт дослідження – процес побудови інтелектуальних клієнт-серверних програмних систем для обробки та аналізу навчального контенту.

Предмет дослідження – методи та програмні засоби генерації навчального контенту із використанням моделей штучного інтелекту та адаптації результатів до рівня знань користувача.

Задачі роботи:

- аналіз сучасних технологій та платформ у сфері інтелектуального навчання і генерації освітнього контенту;
- розробка архітектури програмного забезпечення;
- розробка функціоналу програмного забезпечення;
- проведення тестування програмного забезпечення та перевірка якості роботи реалізованого функціоналу;

2

Рисунок В.2 – Слайд №2

Аналіз вимог до роботи

З метою досягнення поставлених у дипломному проєкті цілей було визначено та систематизовано ключові функціональні можливості та технічні вимоги до розроблюваної системи

Функціональні вимоги:

- авторизація та керування користувачами(реєстрація, автентифікація, збереження сесій навчання).
- робота з навчальним контентом(генерація пояснень і конспектів, підлаштування складності матеріалу під знання користувача).
- генерація тестових сесій, автоматичне оцінювання відповідей, пояснення правильних відповідей, збереження результатів тестування.
- робота з флеш-картками(автоматичне створення флеш-карток, перегортання карток, навігація між картками, групування карток за секціями).
- робота з PDF-документами(завантаження PDF-файлів, генерація навчального контенту на основі PDF).

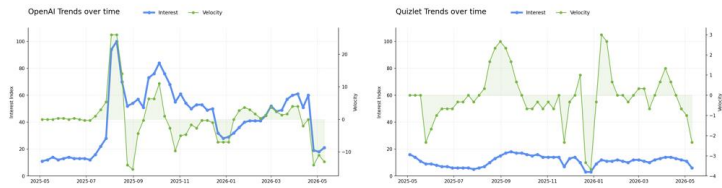
Нефункціональні вимоги:

- безпека(шифрування паролів, захист API-ендпоінтів, серверна валідація даних).
- продуктивність(швидка генерація відповідей, оптимізація запитів до бази даних, кешування клієнтських запитів).
- зручність використання(адаптивний, сучасний, естетичний та простий у використанні інтерфейс).
- сумісність(підтримка сучасних браузерів, стабільна робота на мобільних і десктопних пристроях).

3

Рисунок В.3 – Слайд №3

Актуальність роботи

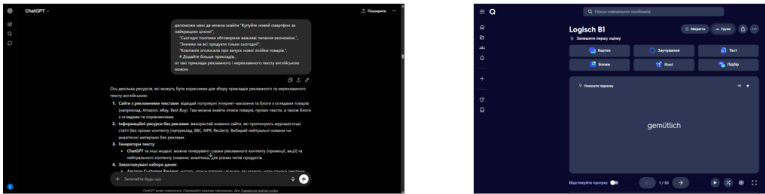


Користувачський інтерес існуючих систем(ChatGPT, Quizlet)

4

Рисунок В.4 – Слайд №4

Актуальність роботи



Приклади існуючих систем(ChatGPT, Quizlet)

5

Рисунок В.5 – Слайд №5

Порівняльна характеристика освітніх платформ

Критерій	ChatGPT	Quizlet	Notion AI	Edu AI Platform
AI-чат	+	-	Частково	+
Генерація тестів	Частково	+	-	+
Флеш-картки	-	+	-	+
Аналіз PDF	+	-	Частково	+
Персоналізація навчання	Частково	Частково	-	+
Адаптація рівня складності	-	-	-	+
Збереження історії навчання	Частково	+	+	+

6

Рисунок В.6 – Слайд №6

Порівняльна характеристика мов програмування

Критерій	TypeScript	JavaScript	Python
Типізація	Статична	Динамічна	Динамічна
Продуктивність	Висока	Висока	Середня
Масштабованість	Висока	Середня	Середня
Підтримка frontend	Так	Так	Обмежена
Підтримка backend	Так	Так	Так
Зручність підтримки	Висока	Середня	Висока

7

Рисунок В.7 – Слайд №7

Порівняльна характеристика середовища розробки

Критерій	TypeScript	JavaScript	Python
Підтримка TypeScript	Повна	Повна	Часткова
Інтеграція з Git	Вбудована	Вбудована	Обмежена
Швидкодія	Висока	Середня	Висока
Ресурсомісткість	Низька	Висока	Мінімальна
Вартість	Безкоштовно	Платно	Платно

8

Рисунок В.8 – Слайд №8

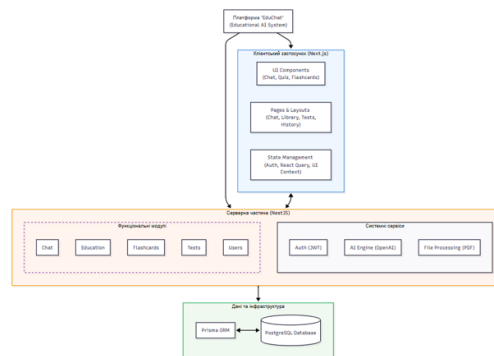
Порівняльна характеристика СУБД

Критерій	PostgreSQL + Prisma	MongoDB	SQLite
Тип моделі	Реляційна	Динамічна	Реляційна
Цілісність даних	Повна	Часткова	Повна
Складність зв'язків	Висока	Обмежено	Базова
Гнучкість системи	Суворі	Висока	Суворі
Типізація	Автоматична	Відсутня	Ручна
Масштабування	Вертикальне/горизонтальне	Горизонтальне	Обмежене

9

Рисунок В.9 – Слайд №9

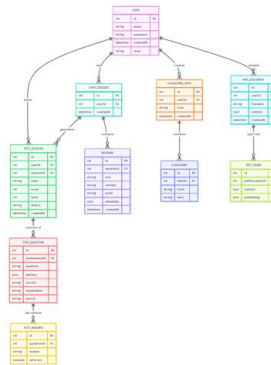
Структурна схема системи



10

Рисунок В.10 – Слайд №10

Схема взаємозв'язків між таблицями бази даних



11

Рисунок В.11 – Слайд №11

Алгоритм роботи системи освітньої платформи



12

Рисунок В.12 – Слайд №12

Приклади роботи системи

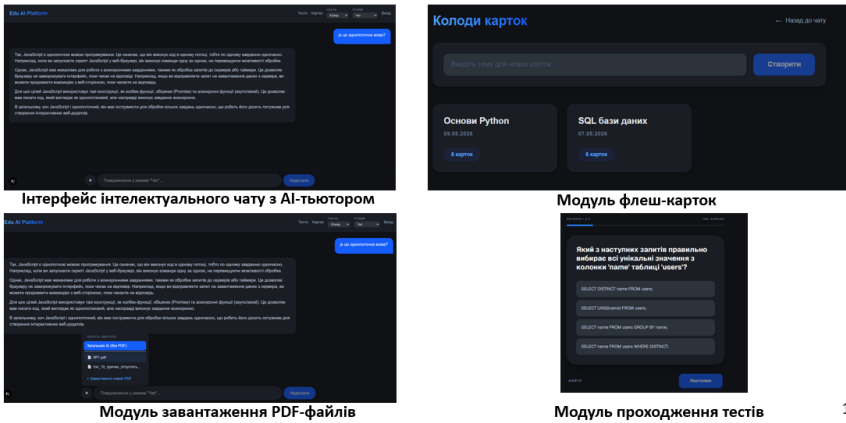


Рисунок В.13 – Слайд №13

Висновки

- **Наукова новизна** полягає в удосконаленні методів персоналізованого навчання шляхом інтеграції моделей штучного інтелекту з технологією аналізу користувацьких PDF-джерел. Зокрема запропоновано комплексний підхід до генерації навчального контенту, який поєднує інтерактивну взаємодію в чаті, семантичний аналіз документів та автоматичне створення різноманітних перевіірочних матеріалів у межах єдиної системи. Також розроблено алгоритм адаптації складності матеріалу, що генерується штучним інтелектом, відповідно до рівня підготовки користувача, що дозволяє динамічно змінювати глибину пояснень та складність перевіірочних завдань.
- **Практична цінність** полягає у створенні вебплатформи «Edu AI Platform», яка автоматизує формування конспектів та перевіірочних завдань на основі технічної літератури. Застосування системи мінімізує час на опрацювання великих обсягів документації та забезпечує інструментарій для ефективної самоосвіти студентів і спеціалістів без допомоги викладачів.
- **Перспективи подальшого розвитку** передбачає інтеграцію аналізу відеолекцій, розробку мобільного застосунку та впровадження модулів колаборативного навчання. Планується додавання предиктивної аналітики для автоматичного коригування навчальної траєкторії на основі прогресу користувача.

Рисунок В.14 – Слайд №14