Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка програмного забезпечення для ефективного вивчення матеріалів за допомогою карток методом інтервального повторення»

Виконав: студент IV курсу
групи 4ПІ-21б спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Франчук Б. В.
(прізвище та ініціали)

Керівник: д.т.н., проф. каф. ПЗ Ліщинська Л. Б.
(прізвище та ініціали)

Рецензент: к.т.н., доцент каф. ОТ Черняк О. І.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ Романюк О. Н.

«09» червня 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
«21» березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Франчуку Богдану Вікторовичу

1. Тема роботи – розробка програмного забезпечення для ефективного вивчення матеріалів за допомогою карток методом інтервального повторення.

Керівник роботи: Ліщинська Людмила Броніславівна, д.т.н., професор кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97.

2. Строк подання студентом роботи: 2 червня 2025.

3. Вихідні дані до роботи: середовище розробки – Visual Studio Code; мови програмування – Javascript, Typescript; фреймворки – Express, React; операційна система – Windows 11.

4. Зміст текстової частини: вступ, аналіз розвитку систем для вивчення матеріалу та формулювання завдань дослідження, розробка моделі та алгоритму системи, розробка програмних модулів системи для вивчення матеріалу за допомогою карток методом інтервального повторення, тестування програми, висновки, список використаних джерел, додатки, ілюстративна частина.

5. Перелік ілюстративного матеріалу: титульний слайд – автор, науковий керівник бакалаврської кваліфікаційної роботи, тема; актуальність розробки; мета, об'єкт та предмет дослідження; постановка задачі; наукова новизна; практична цінність; огляд та аналіз аналогів; модель архітектури системи; алгоритм пошуку зворотних посилань; діаграма послідовності; діаграми класів; блок-схема алгоритму «Half-time Regression»; функціонал розробленої системи; висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Ліщинська Л. Б., д.т.н., професор кафедри ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз розвитку систем для вивчення матеріалу та формулювання завдань дослідження	25.03.25 – 31.03.25	вик
2	Розробка методів та алгоритмів системи	01.04.25 – 18.04.25	вик
3	Розробка програмного забезпечення	18.04.25 – 07.05.25	вик
4	Тестування програми	07.05.25 – 20.05.25	вик
5	Оформлення матеріалів до захисту БКР	20.05.25 – 30.05.25	вик

Студент _____ **Франчук Б. В.**
(підпис) (ініціали та прізвище)

Керівник бакалаврської кваліфікаційної роботи _____ **Ліщинська Л. Б.**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

УДК 004.4

Франчук Б.В. Розробка програмного забезпечення для ефективного вивчення матеріалів за допомогою карток методом інтервального повторення : бакалаврська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 64с.

На укр. мові. Бібліогр. : 26 назв ; рис. : 30; табл. 5.

У бакалаврській кваліфікаційній роботі розроблено вебзастосунок для вивчення навчального матеріалу методом інтервального повторення. Проведено порівняльний аналіз аналогів, сформовано вимоги до системи та реалізовано алгоритм автоматизованого оцінювання знань. Запропоновано модуль створення нотаток зі зворотними посиланнями для збереження контексту. Вперше реалізовано режим опитування у форматі інтерв'ю зі штучним інтелектом, здатним оцінювати повноту, логіку та глибину відповідей користувача. Система зберігає статистику навчання та підтримує візуалізацію знань через зворотні посилання між картками й нотатками. Інтерфейс створено з урахуванням інтуїтивної зручності та можливості масштабування. У розробці використано сучасні вебтехнології: React, Node.js, TypeScript, Prisma. Отримані результати можуть бути використані в освітніх закладах, для самоосвіти та в корпоративному навчанні, а також як основа для розвитку адаптивних навчальних систем.

Ключові слова: інтервальне повторення, картки для навчання, освітнє програмне забезпечення, веб-застосунок.

ABSTRACT

Franchuk B.V. Development of software for effective study of materials using cards by the method of interval repetition: bachelor's qualification work in specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2025. 64pp.

In Ukrainian. Bibliogr.: 26 titles ; fig. 30; tab. 5.

In the bachelor's thesis, a web application for studying educational material using the interval repetition method was developed. A comparative analysis of analogues was conducted, requirements for the system were formed, and an algorithm for automated knowledge assessment was implemented. A module for creating notes with backlinks to preserve context is proposed. For the first time, a survey mode in the interview format with artificial intelligence capable of assessing the completeness, logic, and depth of user responses was implemented. The system stores learning statistics and supports knowledge visualization through backlinks between cards and notes. The interface is designed to be intuitive and scalable. Modern web technologies were used in the development: React, Node.js, TypeScript, Prisma. The results obtained can be used in educational institutions, for self-education and corporate training, as well as as a basis for the development of adaptive learning systems.

Keywords: interval repetition, learning cards, educational software, web application

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасних умовах інтенсивного розвитку інформаційних технологій особливої актуальності набувають системи, що сприяють продуктивному управлінню знаннями та організації самонавчання. Обсяги інформації, з якими стикаються користувачі, постійно зростають, а вимоги до швидкого засвоєння матеріалу та гнучкої адаптації навчального процесу зумовлюють потребу у створенні інноваційних програмних засобів. Сучасні підходи до навчання акцентують увагу на персоналізації, візуалізації знань, підтримці когнітивних процесів і застосуванні методів інтервального повторення [1]. Отже, тема розробки програмної системи для підвищення ефективності самонавчання є актуальною і відповідає потребам сучасної освіти.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення рівня засвоєння навчальних матеріалів та досягнення кращого рівня структурування великих об'ємів інформації.

Відповідно до поставленої мети потрібно вирішити такі завдання:

– провести порівняльний аналіз аналогів та сформувати функціональні та нефункціональні вимоги;

– розробити алгоритм інтервального повторення з автоматизованим оцінюванням;

– розробити модуль створення нотаток із підтримкою зворотних посилань;

– забезпечити можливість оцінки рівня знань користувача за допомогою ШІ у форматі інтерв'ю;

– інтегрувати базу даних для збереження навчальних матеріалів користувача та розробити контролер для керування;

– створити зручний і інтуїтивно зрозумілий графічний інтерфейс користувача;

– провести тестування системи та розробити інструкцію користувача.

Об’єктом дослідження є процес структурування та вивчення навчальних матеріалів з використанням методу інтервального повторення.

Предметом дослідження є методи та засоби організації вивчення навчальних матеріалів.

Методи дослідження. У процесі дослідження використовувалися наступні методи:

– методи аналізу існуючих програмних систем для виявлення їх сильних і слабких сторін, а також для обґрунтування актуальності розробки;

– принципи математичних моделей забування та методи інтервального повторення для розробки дієвого алгоритму запам’ятовування навчального матеріалу;

– методи проєктування інтерфейсів для створення зручного та інтуїтивно зрозумілого інтерфейсу системи;

– способи тестування інтегрованих компонентів системи для перевірки її працездатності і відповідності поставленим вимогам.

Новизна отриманих результатів.

1. Отримала подальшого розвитку система зворотних посилань, яка, на відміну від існуючих, забезпечує тісний зв’язок між картками та розділами конспектів, що дозволяє зберігати контекст засвоюваного матеріалу та покращити розуміння матеріалу за рахунок структурованої організації інформації в межах єдиного навчального середовища.

2. Вперше запропоновано метод аналізу відповідей користувача, особливість якого полягає у здатності штучного інтелекту об’єктивно оцінювати рівень знань та

глибину розуміння матеріалу під час вивчення карток, що дозволяє системі адаптувати навчальний процес та надавати пояснення для досягнення більш глибокого засвоєння навчальних матеріалів.

3. Вперше запропоновано режим опитування у форматі інтерв'ю з використанням штучного інтелекту для роботи з навчальними матеріалами, особливість якого полягає в здатності аналізувати відповіді користувача на предмет повноти, логічності та глибини розуміння, що дозволяє виявляти пробіли в пам'яті та неправильне трактування навчального матеріалу.

Практичне значення полягає у можливості використання розробленого програмного забезпечення для вивчення матеріалу за допомогою карток з використанням методу інтервального повторення, що у взаємодії з удосконаленою системою зворотних посилань дозволяє зберігати контекст засвоюваної інформації. Завдяки інноваційному методу аналізу відповідей користувача та режиму опитування у форматі інтерв'ю зі штучним інтелектом забезпечується об'єктивне оцінювання рівня знань, виявлення прогалин у розумінні та адаптація навчального процесу для досягнення кращого засвоєння матеріалу.

Особистий внесок здобувача. Усі результати, викладені в бакалаврській кваліфікаційній роботі, здобуті автором самостійно.

Апробація матеріалів бакалаврської кваліфікаційної роботи. Основні положення бакалаврської кваліфікаційної роботи доповідалися та обговорювалися на всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету «LIV Всеукраїнська науково-технічна конференція підрозділів ВНТУ» (Вінниця, 2025).

Публікації. Результати досліджень опубліковано в матеріалах конференції [2].

1 АНАЛІЗ РОЗВИТКУ СИСТЕМ ДЛЯ ВИВЧЕННЯ МАТЕРІАЛУ ТА ФОРМУЛЮВАННЯ ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд стану систем для вивчення матеріалу

На сьогоднішній день цифрові технології стали важливою складовою освітнього процесу. Вони відкривають нові можливості для навчання і дозволяють засвоювати знання швидше. Одним з відомих способів запам’ятовування є метод інтервального повторення. Його суть полягає у повторенні матеріалу через певний час, що допомагає краще його закріпити і зменшити ймовірність забування. Найбільш популярним додатком, який реалізує цей метод, вважається Anki. Він використовує алгоритм SuperNemo 2, що автоматично налаштовує інтервали повторень для кожного користувача залежно від того, як швидко він запам’ятовує інформацію. Як результат, його добре організований підхід до запам’ятовування є настільки простим і результативним, що його активно використовують як студенти, які готуються до екзаменів з пріоритетом на довготривале запам’ятовування, так і викладачі і ті, хто просто займається саморозвитком.

Хоча такі додатки як Anki і дають можливість швидко вчити теорію, але не можуть забезпечити цілісного сприйняття, особливо, якщо матеріалу занадто багато. Причиною цьому являється те, що карти вміщають в собі не весь матеріал, а лише окремі вирвані з контексту фрагменти інформації. У результаті користувач засвоює факти ізольовано, без розуміння загальної картини. Це обмежує глибину навчання і створює ризик поверхневого засвоєння знань, що особливо шкідливо при вивченні складних або взаємопов’язаних тем. У таких випадках краще підходять інструменти, які дозволяють зручно зберігати і пов’язувати велику кількість записів. Серед найпопулярніших Notion та Obsidian, вони базуються на ідеї пов’язування документів між собою через зворотні посилання, подібно до того, як це реалізовано на сторінках Вікіпедії, де кожен розділ веде на інший, пов’язаний з

ним. Завдяки цьому користувачі можуть створювати зручні бази знань, у яких легко орієнтуватися і які відповідають власним потребам. Проте, незважаючи на їхню гнучкість, ці застосунки не мають вбудованої підтримки інтервального повторення, що обмежує можливості ефективного запам’ятовування.

Хоча обидві групи інструментів довели свою користь, кожна з них розв’язує лише частину проблеми. Одні дозволяють ефективно запам’ятовувати, інші організовувати знання, але жоден з них не поєднує обидві функції. Через це виникає потреба у створенні нової системи, яка дозволить не тільки структурувати великі об’єми важливої інформації, а й вивчати її простіше та швидше, що забезпечить максимальну продуктивність навчання.

1.2 Порівняльний аналіз аналогів

Серед систем, що використовують для вивчення матеріалів метод інтервального повторення найпопулярнішими є ті, що подають процес вивчення у форматі флеш-карток. Ідея полягає в тому, що користувач розділяє необхідну для вивчення інформацію на питання та відповіді та оформлює у вигляді картки, на одній стороні якої знаходиться питання, а на іншій стороні відповідь. Це створює умови для більш стійкого закріплення знань у довготривалій пам’яті. Коли користувач витягає картку з колоди, він спершу читає питання і намагається спам’ятати відповідь і якщо не це вдається, то користувач переглядає відповідь. Дієвість такого підходу прихований в тому як людина запам’ятовує інформацію. Щоб інформація запам’ятовувалась легко, мозок повинен позначити її як важливу, але не завжди мозок самостійно розпізнає важливу для нас інформацію як дійсно значущу. Він орієнтується на сигнали, які вказують на актуальність або цінність даних, наприклад, емоційне забарвлення, повторюваність або дефіцит [3]. Саме тому в навчальному процесі важливо створити для мозку ілюзію важливості інформації. Одним із дієвих способів є обмеження доступу до правильної відповіді,

користувач не може просто одразу її переглянути, замість цього він має спробувати самостійно її пригадати. Це активізує зусилля пам’яті, створює стан когнітивного напруження і підвищує ймовірність того, що мозок позначить цю інформацію як цінну. Такий підхід імітує дефіцит інформації, в якому можна надіятись тільки на свою пам’ять і саме це стимулює формування довготривалого сліду в пам’яті.

Одним із найвідоміших представників даного підходу є Anki, програма з відкритим кодом, яка забезпечує реалізацію інтервального повторення на основі алгоритму SM-2. Anki дозволяє користувачам створювати власні набори флеш-карток або використовувати загальнодоступні готові колоди. Вона підтримує складне форматування вмісту карток за допомогою HTML та LaTeX, що робить її зручною для відображення як текстової, так і ілюстративної інформації. Важливою перевагою Anki є її кросплатформеність та наявність синхронізації даних через AnkiWeb. Водночас, програму часто критикують за застарілий інтерфейс і відсутність інтеграції з різними методами організації знань, що обмежує її дієвість при засвоєнні великого об’єму інформації [4] (див. рисунок 1.1).

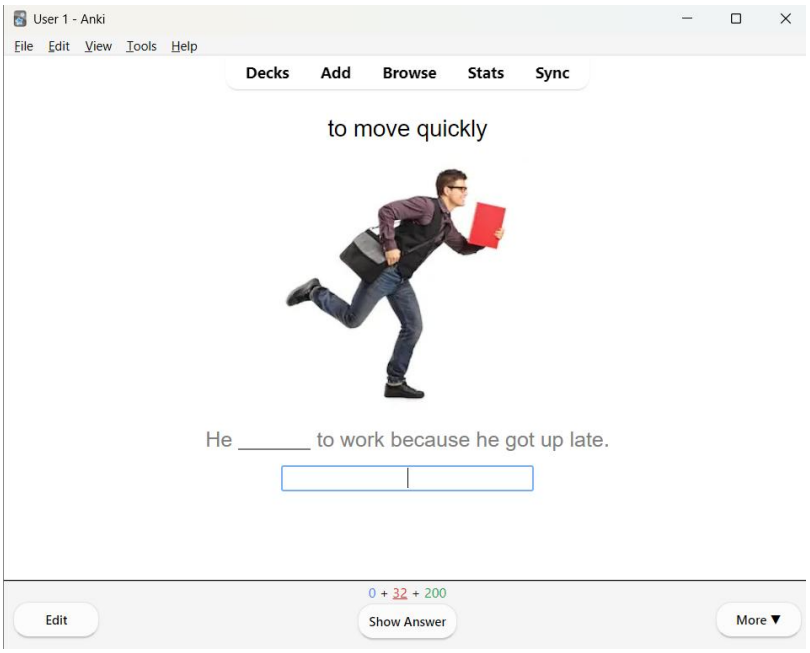

Рисунок 1.1 – Інтерфейс «Anki»

На відміну від Anki, Traverse орієнтується на візуалізацію логічних зв’язків між даними шляхом інтеграції мапи думок, зворотних посилань та адаптивного повторення. Користувачі можуть працювати з інформацією у вигляді документів, які потім можуть бути розташовані на віртуальній дошці, що надає візуальне представлення про структуру навчального матеріалу. Документи в Traverse також дозволяють вставляти відео, зображення, списки, математичні формули, таблиці та блоки коду. Також доступні функції виділення тексту кольором і створення скетчів, що теж доповнює візуальний підхід до навчання. Але Traverse не дозволяє вільно малювати на мапі думок або створювати складні графічні елементи, що частково обмежує можливості візуального подання [5] (див. рисунок 1.2).

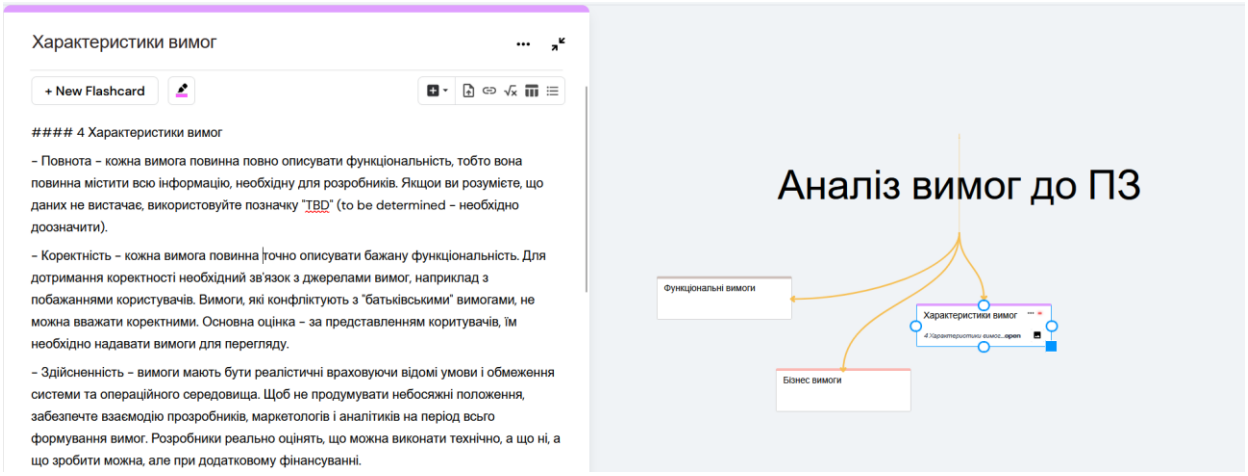

Рисунок 1.2 – Інтерфейс «Traverse»

Сервіс Recall відрізняється фокусом на автоматизації процесу створення навчального контенту. Він дозволяє створювати нотатки й флеш-картки безпосередньо з веб-сторінок, відео та статей. За допомогою вбудованого штучного інтелекту, програма генерує картки на основі резюме доданого контенту, що значно пришвидшує підготовку матеріалів до вивчення. Використовується розмітка Markdown, яка підтримує форматування, додавання таблиць, коду, відео та списків.

Проте Recall не підтримує мапу думок, що являється недоліком для користувачів, які надають перевагу візуальній організації матеріалу [6] (див. рисунок 1.3).

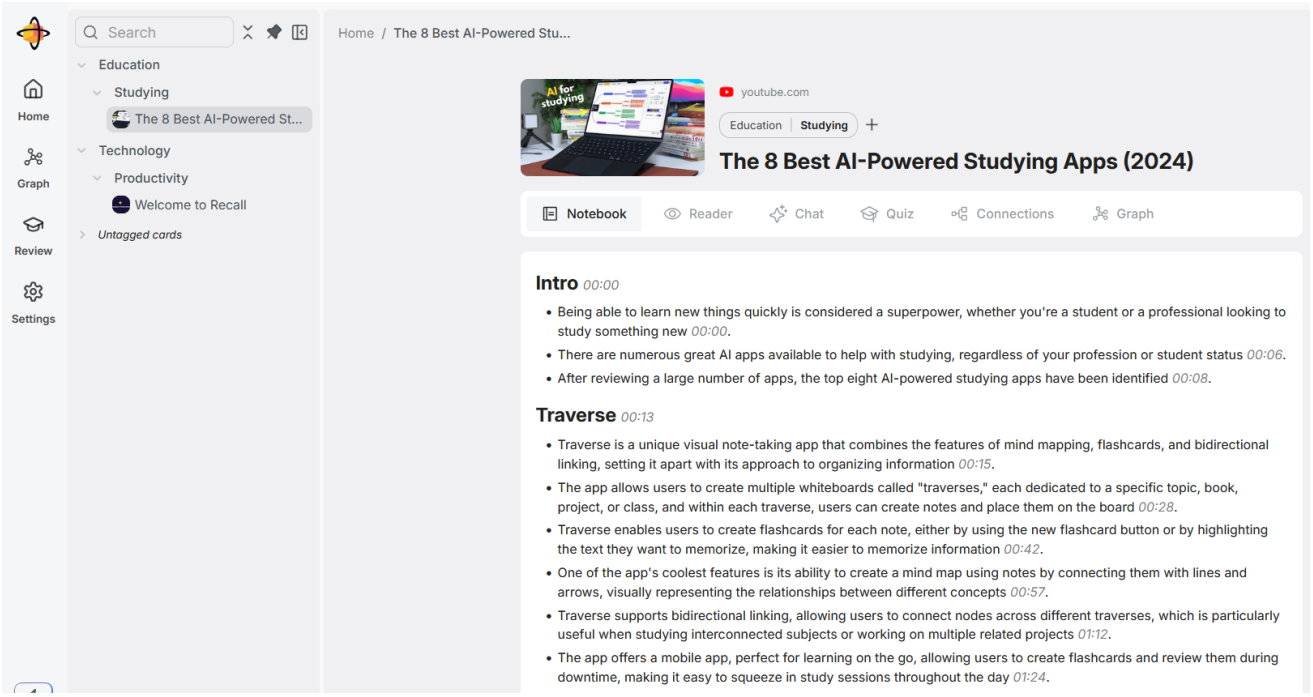

Рисунок 1.3 – Автоматичне формування конспектів в «Recall»

RemNote це ще один приклад інструмента, який поєднує можливості створення карток для вивчення в поєднані з матеріалами, оформленими у виді нотаток. Він дозволяє організовувати інформацію у вигляді тез, які можуть бути автоматично перетворені у флеш-картки. Унікальною особливістю RemNote є можливість перегляду контексту для кожного питання під час повторення, що покращує глибину розуміння. Додатково надаються інструменти для встановлення пріоритету вивчення, додавання тегів, керування дедлайнами, а також функціонал ШІ, що генерує картки на основі введеного тексту, але не підтримує мапу думок або візуальні інструменти для представлення інформації [7] (див. рисунок 1.4).

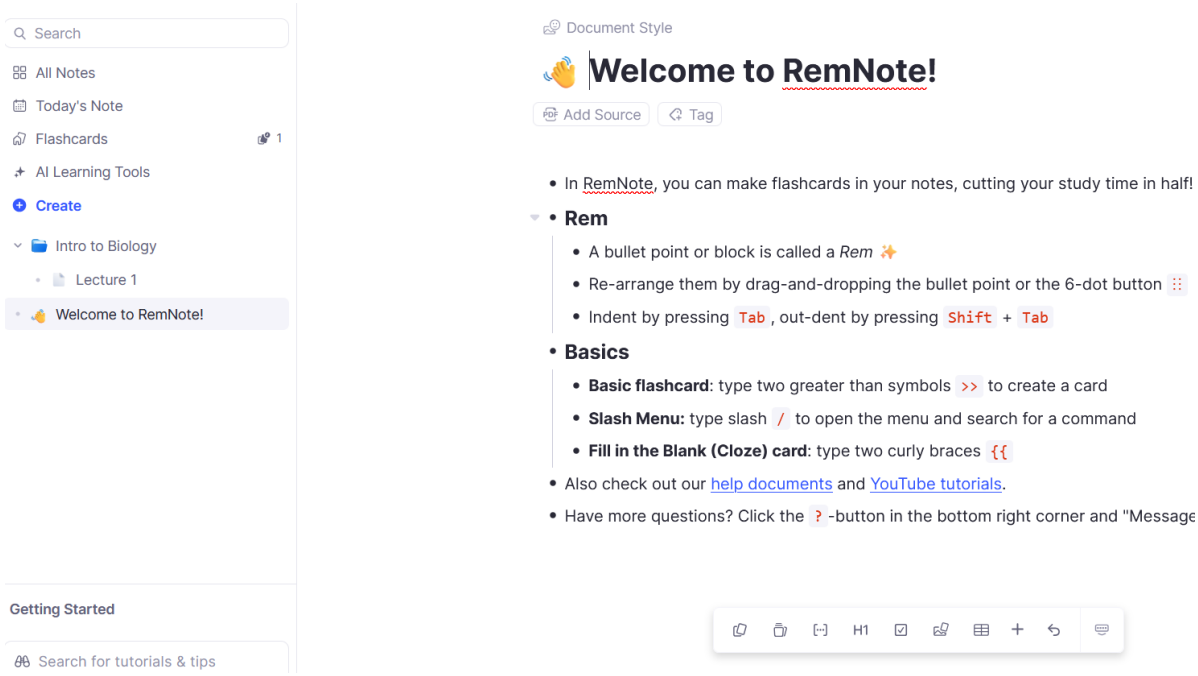

Рисунок 1.4 – Створення нотаток в «RemNote»

Аналогічні системи, хоча й підтримують навчання за допомогою карток, зосереджені переважно на запам’ятовуванні окремих термінів чи визначень, вирваних із загального контексту. У них відсутній режим, який би сприяв формуванню цілісного розуміння теми в цілому. Цю проблему можна вирішити інтегрувавши систему опитування користувача по загальній темі у форматі інтерв’ю із штучним інтелектом, задачі якого знайти слабкі місця в розумінні теми. Через серію відкритих запитань та автоматичний аналіз відповідей, користувач отримує повноцінний зворотний зв’язок щодо своїх знань, що значно покращить процес засвоєння матеріалу.

На основі проведеного аналізу була побудована порівняльна таблиця (див. таблиця 1.1), яка узагальнює функціональні можливості згаданих систем за основними критеріями, а саме наявність алгоритму інтервального повторення, створення конспектів, мапи думок, зворотних посилань, графа, та використання ШІ для автоматизації створення конспектів, карток та оцінки знань користувача.

Таблиця 1.1 – Порівняльна характеристика систем

Критерії	Anki	Traverse	Recall	RemNote	Власна розробка
Алгоритм інтервального повторення	1	1	1	1	1
Документи для створення конспектів	0	1	1	1	1
Мапа думок	0	1	1	0	1
Асистент зі штучним інтелектом	0	1	1	1	1
Зворотні посилання	0	1	1	1	1
Граф	0	1	1	0	1
Режим опитування у форматі інтерв’ю	0	0	0	0	1
Сумарний коефіцієнт	1	6	6	4	7

Провівши аналіз аналогів було визначено, що хоча окремі існуючі програмні засоби успішно реалізують певні процеси продуктивного навчального процесу, жоден із них не охоплює повного спектра необхідних функцій у межах єдиної системи. Це створює необхідність для розробки нового інструмента, здатного

об’єднати ключові переваги вивчених рішень у цілісну та логічно узгоджену систему. Запропонована система передбачає інтеграцію методу інтервального повторення, мапи думок, створення нотаток, зворотні посилання, граф, інтеграцію штучного інтелекту для оцінки знань, що надає повноцінний інструмент, який покриває всі необхідні потреби в організації навчального процесу.

1.3 Аналіз методів розв’язання задачі

Метод інтервального повторення базується на кривій забування Еббінгауза, яка показує, що без регулярного повторення засвоєна інформація забувається експоненційно швидко з плином часу. Основна ідея інтервального повторення полягає в тому, щоб повторювати навчальний матеріал з оптимальними інтервалами, які збільшуються в залежності від того, наскільки добре інформація була засвоєна. Це дозволяє зменшити кількість повторень без втрати продуктивності навчання [8].

Вибір найбільш продуктивного алгоритму інтервального повторення є ключовим чинником для створення успішної системи навчання, яка дозволяє максимально покращити засвоєння інформації та знизити ймовірність її забування. Оскільки на сьогоднішній день існує кілька підходів із різним ступенем складності та адаптивності, важливо провести детальний аналіз існуючих алгоритмів.

Серед алгоритмів, які реалізують принцип інтервального повторення, найвідомішим є алгоритм SM-2, розроблений для системи SuperMemo і пізніше адаптований у популярній платформі Anki [9]. Такий підхід потребує користувача оцінювати наскільки легко йому було згадати відповідь після кожного повторення. Виходячи з цієї оцінки, система коригує наступний інтервал повторення. SM-2 є достатньо простим у реалізації та дає хороші результати, однак має певні обмеження, наприклад, він не враховує реальну складність інформації та суб’єктивність оцінки користувача, що може впливати на точність інтервалів.

Ще один часто використовуваний метод система Лейтнера, суть якого полягає на розподілі карток за віртуальними «коробками». Кожна коробка має свій інтервал повторення. Якщо користувач правильно відповідає на картку, вона переміщується в коробку з довшим інтервалом, а у разі помилки повертається на початок. Цей метод є надзвичайно простим і досить дієвим для базового навчання, але не підлаштовується під складність навчального матеріалу, що обмежує його адаптивність [10].

Більш адаптивний підхід представляє алгоритм Half-Life Regression, який використовує математичну модель для оцінки «періоду напіврозпаду» знань. В основі лежить статистичний аналіз історії взаємодій користувача з карткою. Це дозволяє моделювати й передбачати оптимальні інтервали повторення для кожної картки, що дозволяє алгоритму максимально підлаштовуватись під темп користувача [11]. Реалізація такого методу складніша, але він забезпечує високу точність розрахунку інтервалів повторення.

Ще одним прикладом є математична модель Forgeting Curve Model, яка базується на експоненційному забуванні. Даний підхід часто використовується як основа для розробки алгоритмів інтервального повторення, включаючи Half-Life Regression. Програма прогнозує момент, коли знання буде втрачено, і встановлює повторення саме в цей момент, але її головним недоліком є те, що на відміну від Half-Life Regression, вона не враховує індивідуальні темп запам’ятовування користувача та оцінку попередніх повторень. Вона базується на узагальненій експоненціальній моделі забування, яка однакова для всіх, тоді як Half-Life Regression аналізує відповіді користувача і підлаштовує інтервали під конкретного користувача[12].

Провівши аналіз переваг і недоліків цих алгоритмів інтервального повторення було побудовано порівняльну таблицю (див. таблиця 1.2).

Таблиця 1.2 – Порівняльна характеристика алгоритмів інтервального повторення

Критерії	SM-2	Leitner System	Forgetting Curve Model	Half-Life Regression
Адаптація до користувача	1	0	1	1
Врахування складності матеріалу	0	0	0	1
Динамічне формування інтервалів	1	0	1	1
Проста реалізація	1	1	0	0
Урахування історії повторень	1	0	0	1
Сумарний коефіцієнт	4	1	2	4

В результаті проведення аналізу було визначено, що для впровадження у власній розробці найбільш прийнятним є використання алгоритму Half-Life Regression, оскільки він добре підлаштовується під користувача, формує інтервали в залежності від кількості повторень без помилок, враховує результат попередніх повторень та, на відміну від алгоритму SM-2, враховує складність матеріалу, що являється більш важливим критерієм, аніж простота реалізації.

1.4 Постановка задач дослідження

Проаналізувавши переваги та недоліки існуючих програмних засобів для вивчення матеріалу за допомогою карток методом інтервального повторення, було визначено функціональні та нефункціональні вимоги до програмного забезпечення.

Функціональні вимоги:

- програмне забезпечення має підтримувати адаптивне навчання шляхом впровадження алгоритму «Half-Life Regression», який забезпечує індивідуальне налаштування інтервалів повторення залежно від рівня засвоєння користувачем;
- повинна бути реалізована можливість створення, редагування та збереження текстових конспектів;
- програма має містити інструменти створення мапи думок, що дає змогу відображати інформацію з використанням графічних зображень;
- необхідно реалізувати зворотні посилання, що дозволять додавати логічні зв’язки між катками та нотатками, а також граф, який візуально відображає ці зв’язки;
- повинен бути впроваджений режим вивчення карток зі штучним інтелектом, який буде об’єктивно оцінювати відповідь користувача, що дозволить точніше налаштовувати інтервал повторення;
- в програмі має бути впроваджений режим опитування у форматі інтерв’ю, що використовує штучний інтелект в якості інтерв’юера, який опитує користувача по вибраній нотатці для визначення рівня розуміння теми.

Нефункціональні вимоги:

- програма має забезпечувати швидкий відгук інтерфейсу, мінімальні затримки при перемиканні між модулями, обробці введення та збереженні даних;
- система повинна працювати без збоїв під час тривалої сесії а також підтримувати збереження даних у реальному часі;

- архітектура програмного застосунку має підтримувати можливість розширення функціоналу без кардинальних змін основного коду;

- необхідно забезпечити захист персональних даних користувачів, включаючи авторизацію, шифрування інформації та захист від несанкціонованого доступу.

- програмний засіб має функціонувати у сучасних браузерах на різних операційних системах та мобільних пристроях.

1.5 Висновки

Проведено порівняльний аналіз сучасних програмних засобів для навчання за допомогою інтервального повторення, зокрема Anki, Traverse, Recall та RemNote. Аналіз було проведено шляхом вивчення функціональних можливостей зазначених систем з особливою увагою на реалізацію алгоритмів повторення, підтримку створення конспектів, візуалізацію графа, можливість створення мапи думок та інтеграцію з штучним інтелектом для оцінки знань користувача в форматі інтерв'ю та при вивченні інформації з використанням карток.

Також було проведено аналіз різних алгоритмів інтервального повторення, в результаті чого вирішено, що вимогам роботи найкраще підходить алгоритм «Half-Life Regression». В результаті аналізу було визначено актуальність і необхідність розробки власної системи, також було визначено її функціональні та нефункціональні вимоги.

2 РОЗРОБКА МЕТОДІВ ТА АЛГОРИТМІВ СИСТЕМИ

2.1 Розробка алгоритму інтервального повторення з автоматизованим оцінюванням

Одним із найважливіших елементів адаптивної логіки навчання, реалізованих у програмному забезпеченні, є алгоритм інтервального повторення, що базується на моделі Half-Life Regression (див. рисунок 2.1).

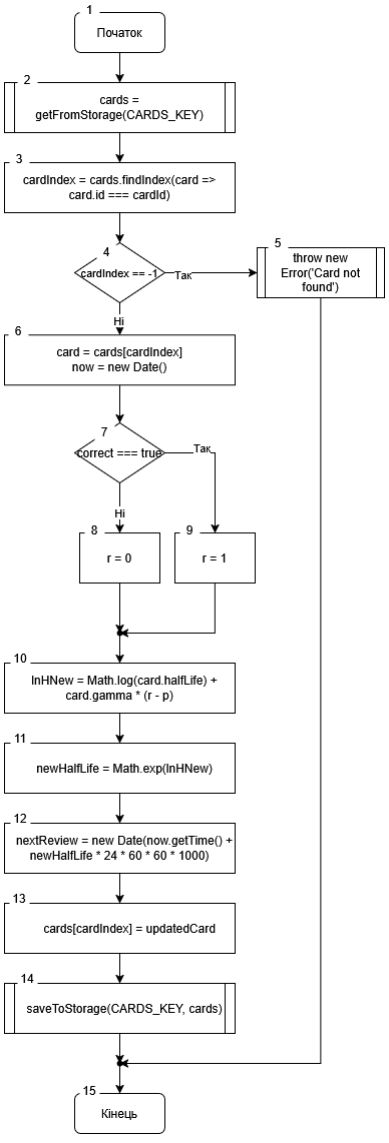

Рисунок 2.1 – Блок-схема алгоритму «Half-time Regression»

Цей алгоритм застосовується для динамічного оновлення інтервалів між повтореннями навчальних карток залежно від коректності відповідей користувача.

Його головна мета полягає в тому, щоб максимально наблизити повторення інформації до моменту її можливого забування, тим самим забезпечуючи оптимальне використання часу і покращити процес запам’ятовування знань [13].

Першим етапом є завантаження списку навчальних карток з бази даним за допомогою унікального ключа, наприклад, CARDS_KEY. Це забезпечує збереження персоналізованих даних навіть при відсутності з’єднання з мережею.

Далі система здійснює пошук картки за її унікальним ідентифікатором cardId. У разі, якщо картка з відповідним ID не знайдена, генерується виключення throw new Error('Card not found'), що запобігає подальшому виконанню з некоректними даними. Якщо ж картка виявлена, алгоритм переходить до етапу обробки інформації про неї.

У першу чергу зберігається поточна дата та обчислюється кількість днів, що минула від останнього перегляду цієї картки, інтервал dt. Цей часовий проміжок є основою для обчислення ймовірності правильного згадування картки за допомогою експоненційної функції (2.1).

$$p = e^{\left\{\frac{-dt}{h}\right\}} \tag{2.1}$$

Змінна e це попередній період часткового забування для цієї картки. Вона враховує, що з часом ймовірність правильного згадування експоненційно зменшується, що підтверджено численними емпіричними дослідженнями у когнітивній психології [14].

Наступним кроком є інтерпретація відповіді користувача у вигляді бінарної змінної r значення якої дорівнює одиниці, якщо відповідає правильній відповіді, і нулю, коли відповідає неправильній. Використовуючи логарифмічну регресію,

система оновлює значення натурального логарифму періоду забування за формулою (2.2).

$$ln\, h_{\{new\}} = ln\, h + \alpha \cdot (r - p) \quad (2.2)$$

де α коефіцієнт навчання, який визначає, наскільки сильно нова відповідь вплине на зміну інтервалу повторення. Після цього обчислюється період часткового забування (2.3).

$$h_{new} = e^{\ln h_{new}} \quad (2.3)$$

Визначається нова дата наступного повторення як сума поточного моменту часу та оновленого періоду h_new у днях.

Далі створюється новий об’єкт картки, у якому оновлюються параметри періоду часткового забування, дата останнього перегляду, дата наступного перегляду, кількість переглядів, середній рівень успішності користувача, а також дата останньої зміни. Оновлена картка замінює попередню версію у загальному списку, після чого цей список перезаписується в базу даних.

В кінці виконання алгоритму повертається оновлений об’єкт картки з актуальними значеннями, що дозволяє системі підлаштовувати навчальний процес до швидкості запам’ятовування кожного користувача. Такий підхід забезпечує дієве використання принципу інтервального повторення, що довів свою продуктивність у багатьох навчальних системах, таких як SuperMemo, Anki та Duolingo [15].

Важливою складовою розробленого застосунку є модуль взаємодії з штучним інтелектом, який значно підвищує дієвість навчання за допомогою карток. Зазвичай у системах інтервального повторення користувач самостійно оцінює правильність

своєї відповіді після перегляду правильної відповіді, що може призводити до суб'єктивності оцінки, що в свою чергу погано вплине на дієвість методу. Запропонований модуль штучного інтелекту усуває цей недолік, надаючи об'єктивну оцінку введеної відповіді.

Основне призначення модуля полягає в автоматичній перевірці текстових відповідей користувача за допомогою великої мовної моделі. Після введення відповіді користувачем, вона надсилається до сервера, де передається в API ChatGPT, або аналогічної нейронної мережі. На основі контексту картки та очікуваної правильної відповіді, модель аналізує відповідь користувача та формує наступні оцінку правильності, а також, при необхідності, надає додаткові пояснення.

Таким чином, користувач отримує розгорнутий зворотний зв'язок, що дозволяє не лише фіксувати помилки, а й краще зрозуміти вивчений матеріал. Це створює більш адаптивне і глибоке навчальне середовище порівняно з традиційним підходом.

2.2 Розробка алгоритму пошуку зворотних посилань

У рамках створення системи навчання на основі нотаток і карток з інтервальним повторенням важливо забезпечити можливість створення логічних зв’язків між різними навчальними матеріалами. Для цього необхідно розробити механізм зворотних посилань, який дозволяє встановлювати зв’язки між картками та нотатками або їх розділами. Такий підхід сприяє формуванню змістовної структури знань, подібної до графа, та полегшує навігацію між нотатками та картками.

Зворотні посилання в даній реалізації це спеціальний текстовий маркер у форматі [[НазваНотатки#Розділ]], який зберігається в картці або нотатці й дозволяє за потреби швидко перейти до відповідного розділу іншої нотатки [16]. Такий

формат запозичено з популярних систем організації баз знань, в тому числі Obsidian.

Щоб реалізувати роботу цієї функції, розроблено алгоритм, який виконує пошук і обробку зворотних посилань у кілька етапів. На рисунку 2.2 зображено логіку роботи алгоритму пошуку зворотних посилань.

Рисунок 2.2 - Алгоритм пошуку зворотних посилань

Алгоритм починає свою роботу з отримання зворотного посилання, після чого виконується його розбір на назву нотатки та назву розділу. За назвою

виконується пошук відповідної нотатки в базі даних за допомогою ORM-бібліотеки Prisma. Якщо нотатку знайдено, здійснюється пошук вмісту розділу, який виконується через парсинг Markdown-формату. У разі успіху повертається відповідний фрагмент тексту для подальшого відображення або обробки. Якщо ж нотатка або розділ не знайдені, користувач отримує відповідне повідомлення про помилку.

Таким чином, алгоритм забезпечує зручний і масштабований спосіб реалізації зворотних посилань у системі, підтримуючи зручну та інтуїтивно зрозумілу навігацію по навчальних матеріалах.

Для візуалізації таких зв’язків реалізовано граф знань, який візуалізує кожну нотатку або картку як вузол, а кожне зворотне посилання як ребро між цими вузлами. Це дозволяє користувачу бачити повну картину зв’язків між картками та нотатками і їх розділами.

2.3 Розробка методу оцінювання знань у форматі інтерв’ю

Модуль «Інтерв’ю» є являється найбільш оригінальним компонентом розробленого застосунку, адже він не використовується в аналогічних системах. Його наявність дозволяє поєднати методи інтервального повторення та аналіз загального розуміння теми. Головна мета полягає в наданні користувачу можливість пройти згенероване штучним інтелектом опитування за повним змістом конспекту, що дозволяє комплексно оцінити рівень розуміння теми, а не лише окремих фактів чи термінів.

Користувач обирає одну з існуючих нотаток, збережених у базі даних, після чого інтерфейс перемикається у режим діалогу з штучним інтелектом, який в свою чергу формує серію відкритих запитань на основі вмісту обраної нотатки, аналізує відповіді користувача та виявляє не тільки прості помилки, але й певні неточності або неповне розуміння матеріалу. Результати аналізу зберігаються у вигляді сесії

інтерв’ю, що може бути використана надалі для формування персоналізованих карток або рекомендацій.

З технічної точки зору, модуль «Інтерв’ю» є розширенням модуля оцінки відповідей користувача при вивчення карток з участю штучного інтелекту Але цей модуль має власну логіку обробки матеріалу, формування запитань і оцінювання результатів. На рисунку 2.3 зображена діаграма послідовності, яка відображає послідовність взаємодії користувача з модулями системи.

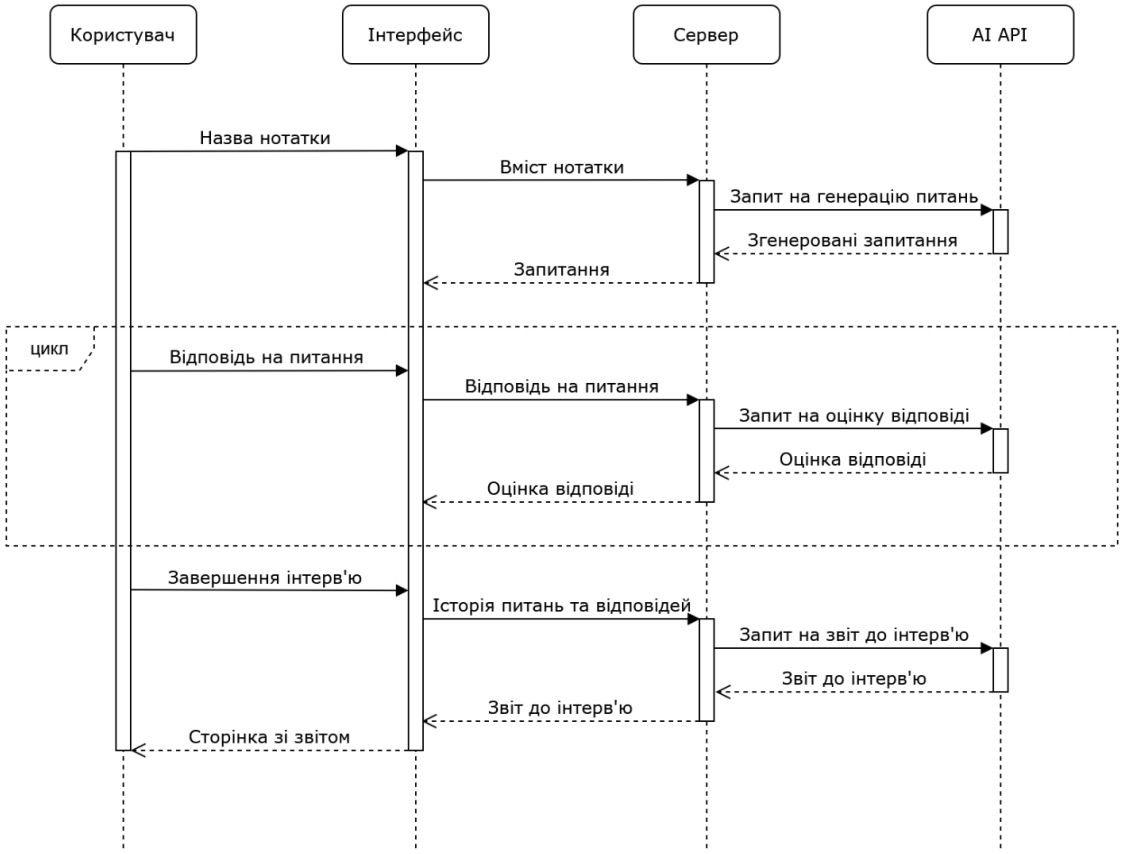

Рисунок 2.3 - Діаграма послідовності

Інтерфейс реалізовано у вигляді класичного чату з можливістю переглядати історію діалогу, попереднього перегляду нотатки, а також завершувати сесію

інтерв’ю в любий момент, без необхідності спершу давати відповіді на фіксовану кількість запитань. Після завершення інтерв’ю штучний інтелект надсилає повідомлення, яке містить підсумковий аналіз, у якому перелічуються теми, які засвоєні добре та вказуються фрагменти, які потребують повторення.

2.4 Висновки

Розроблено методи та алгоритми, які забезпечують новизну і функціональну повноту програмного забезпечення для навчання. Реалізовано алгоритм інтервального повторення, заснований на моделі Half-Life Regression, що дозволяє адаптувати навчальний процес до індивідуального рівня запам’ятовування користувача. Впроваджено автоматизований процес оцінки відповідей за допомогою штучного інтелекту, що усуває суб’єктивність самооцінювання і покращує точність роботи алгоритму інтервального повторення.

Крім того розроблено алгоритм зворотних посилань, який забезпечує формування логічних зв’язків між навчальними матеріалами у форматі графа знань, сприяючи цілісному сприйняттю інформації та зручній навігації. Також проведено розробку модуля «Інтерв’ю», що дозволяє оцінювати глибину розуміння матеріалу через опитування у форматі діалогу з штучним інтелектом, на основі змісту обраної нотатки.

3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИВЧЕННЯ МАТЕРІАЛУ ЗА ДОПОМОГОЮ КАРТОК МЕТОДОМ ІНТЕРВАЛЬНОГО ПОВТОРЕННЯ

3.1 Варіантний аналіз та обґрунтування вибору засобів для реалізації програмного забезпечення

У межах розробки програмного забезпечення було вирішено використовувати JavaScript як основну мову розробки. Такий вибір зумовлений її універсальністю та широкою екосистемою як для клієнтської, так і для серверної частини застосунку. Завдяки підтримці мови TypeScript, яка розширює можливості JavaScript за рахунок статичної типізації, досягається підвищена надійність роботи і полегшується процес налагоджування коду .

Оскільки проєкт включає в себе реалізацію як фронтенду, так і бекенду з використанням сучасного стеку технологій, було проведено порівняльний аналіз популярних фреймворків для обох рівнів архітектури [17].

У процесі проведення аналізу були використані чітко визначені технічні критерії, які безпосередньо пов’язані з функціональністю та вимогами роботи. Один із важливих аспектів полягає у підтримці мови TypeScript [18], яка забезпечує розробку з чітким контролем типізації змінних на стороні клієнта і на сервері. Для фронтенду також корисною є сумісність з Vite для компіляції та розгортання застосунків, з автоматичним оновленням модулів без необхідності перезапускати компіляцію проєкту вручну. Оскільки проєкт інтегрує можливість створення конспектів, до критеріїв було включено підтримку роботи з бібліотекою TipTap, яка відповідає за редагування та форматування текстових документів. Також необхідна реалізація Canvas для побудови схем і малюнків, з цією задачею добре вправиться бібліотека Excalidraw. Для побудови інтерфейсів, які можна легко обновлювати навіть при великому розмірі елементів необхідно, щоб фреймворк підтримував

поділ на компоненти, адже такий підхід дозволяє легко адаптувати, комбінувати та масштабувати інтерфейс з появою нового функціоналу. Також важливо враховувати, наскільки велика спільнота підтримує фреймворк, адже це напряму впливає на швидкість розробки, бо чим активніша спільнота, тим легше знайти відповіді на поширені питання та готові рішення типових проблем.

Ще одним критерієм була підтримка бібліотек для візуалізації графів, що необхідно для візуального відображення зв’язків між картками та нотатками.

Для розробки клієнтської частини розглядалися такі фреймворки як React, Vue.js та Angular.

React це фреймворк для створення інтерфейсів, що дозволяє будувати складні користувацькі інтерфейси, спираючись на компонентну архітектуру, де кожна частина користувацького інтерфейсу може бути подана у вигляді незалежного елементу. Розробка на React зазвичай вимагає додаткових інструментів для управління станом або маршрутизацією, що з одного боку забезпечує гнучкість, але з іншого потребує більше часу для освоєння [19].

Vue це фреймворк, орієнтований на створення інтерфейсів користувача. Його основна ідея полягає в тому, щоб зробити початок розробки максимально простим, дозволяючи водночас масштабуватись до повноцінного проєкту. Він використовує декларативний підхід до побудови UI, має зрозумілий синтаксис і добре документований API. Часто його обирають за простоту вивчення і логічну структуру, хоча при розробці складніших застосунків може бути важко орієнтуватись в проєкті [20].

Angular це фреймворк, розроблений Google для розробки масштабованих вебдодатків. Він пропонує цілий набір інструментів без необхідності початкового налаштування, включно з маршрутизацією, формами, перевіркою прав доступу, що спрощує створення повноцінного додатку без зовнішніх бібліотек. Його строга структура дозволяє підтримувати великі проєкти у чистому вигляді, але через

складність концепцій і велику кількість шаблонного коду новачкам може бути складно швидко почати [21].

Для порівняння характеристик аналогів була вибрана така шкала оцінки: 2 бали, якщо є повна відповідність критерію, 1 бал, якщо часткова відповідність; і 0 балів, якщо не відповідає критерію. В результаті аналізу фреймворків для фронтенду була побудована порівняльна таблиця характеристик (див. таблиця 3.1).

Таблиця 3.1 – Порівняльна характеристика фронтенд фреймворків

Характеристика	React	Vue	Angular
Підтримка TypeScript	2	2	2
Сумісність із Vite	2	2	1
Робота з TipTap	2	2	1
Підтримка Excalidraw	2	1	0
Гнучкість компонентної структури	2	2	1
Поширеність у спільноті	2	1	1
Продуктивність	2	2	1
Підтримка графів	2	1	1
Загальний коефіцієнт	16	13	8

В результаті порівняння фреймворків було визначено, що React краще відповідає вимогам проекту в порівнянні з Vue та Angular, адже відповідає найбільшій кількості критеріїв.

Що стосується бекенду, важливою передумовою була підтримка TypeScript, що дозволяє забезпечити чітку типізацію між фронтендом і серверною частиною, що сприяє зниженню кількості помилок і полегшує рефакторинг. Простота налаштування фреймворку оцінювалася з точки зору швидкого старту та зручного конфігурування базових функцій маршрутизації та обробки запитів. Окрему увагу

приділено гнучкості та контролю, можливості розробника повністю керувати логікою обробки запитів, не обмежуючись жорсткою структурою або надлишковими абстракціями. Оскільки у проєкті використовується ORM Prisma, фреймворк мав бути сумісним із цією бібліотекою для швидкої взаємодії з базою даних.

Серед критеріїв також розглядалася безпека, наскільки легко реалізуються механізми аутентифікації, хешування паролів, контроль CORS-запитів тощо. Поширеність фреймворку та наявність якісної документації відіграють роль у зниженні ризиків, пов’язаних з впровадженням менш відомих технологій. Продуктивність аналізувалась на основі швидкості обробки запитів та масштабованості. Завершальним критерієм була сумісність із фронтендом, тобто зручність побудови API для клієнтської частини, створеної на React.

В результаті аналізу фреймворків для бекенду була побудована порівняльна таблиця характеристик (див. таблиця 3.2).

Таблиця 3.2 – Таблиця порівняння функціональних характеристик backend фреймворків

Характеристика	Express	NestJS	Fastify
Підтримка TypeScript	2	2	2
Простота налаштування	2	1	2
Гнучкість та контроль	2	1	2
Сумісність із Prisma	2	2	2
Безпека	2	2	2
Поширеність, документація	2	2	1
Продуктивність	1	1	2
Сумісність із frontend	2	2	2
Загальний коефіцієнт	15	13	15

В результаті порівняння фреймворків було вирішено використовувати для розробки Express [22], адже він краще відповідає вимогам проекту аніж NestJS[] та Fastify [23].

В результаті вибору засобів для реалізації програмного забезпечення було розроблено багаторівневу архітектурну модель інтерактивної системи самонавчання, що поєднує сучасні вебтехнології, адаптивні алгоритми інтервального повторення та візуалізацію графу (див. рисунок 3.1). Обрана архітектура ґрунтується на принципах розділення відповідальностей і дозволяє забезпечити модульність, масштабованість та легкість у подальшій підтримці й розвитку програмного засобу.

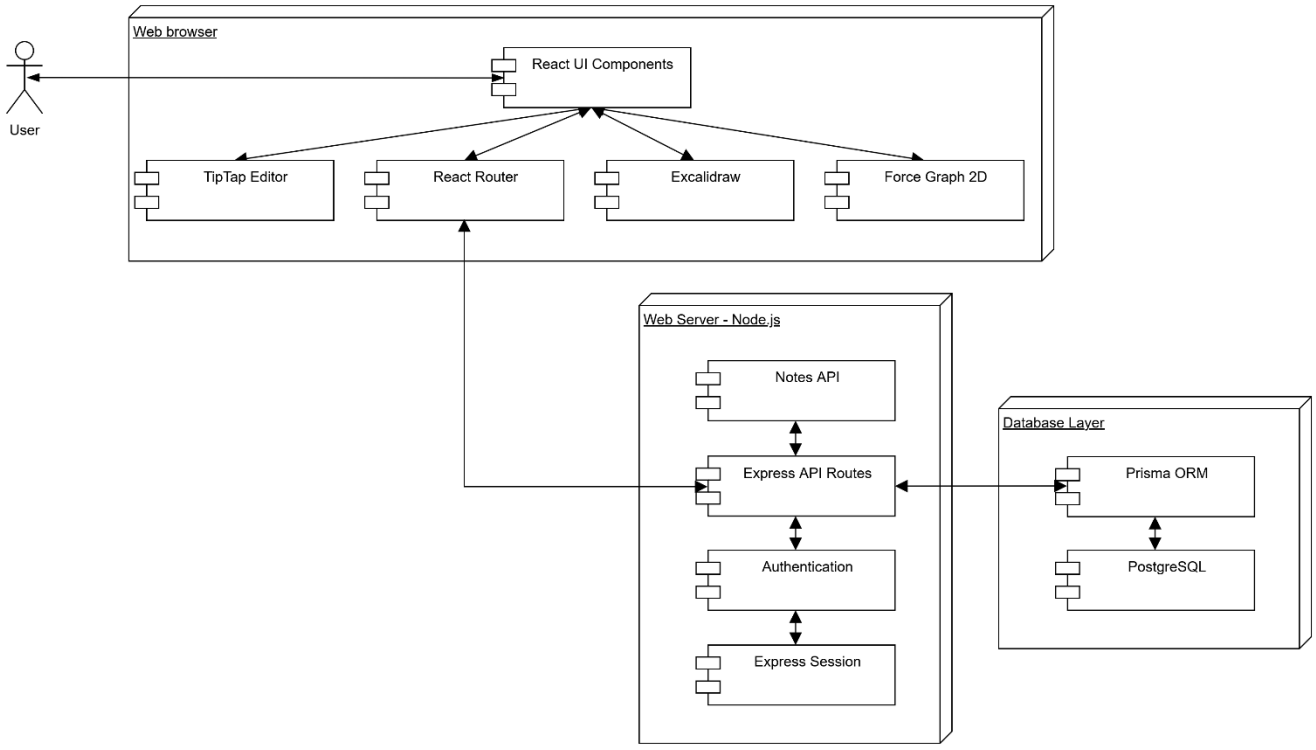

Рисунок 3.1 – Модель архітектури системи

Система реалізована на основі клієнт-серверної архітектури, що включає інтерфейс користувача, серверну логіку та рівень бази даних. Компоненти системи

були описані у вигляді графічної діаграми, яка демонструє модулі, їхні взаємозв’язки та функціональні зв’язки між підсистемами.

На рівні інтерфейсу користувача використовуються бібліотека React та компілятор Vite, що забезпечує високу продуктивність та швидку збірку проєкту. Головним завданням цього рівня є забезпечення взаємодії з користувачем через інтуїтивно зрозумілий графічний інтерфейс.

Frontend взаємодіє з сервером за допомогою HTTP-запитів через RESTful API, реалізоване у рамках Backend-рівня на основі Node.js та Express.

На рівні бази даних застосовується реляційна СУБД PostgreSQL [24]. Для взаємодії з нею використовується ORM-бібліотека Prisma, яка забезпечує типізований доступ до даних і спрощує розробку, зменшуючи кількість помилок, пов’язаних з роботою з SQL-запитами. У структурі бази даних зберігаються користувацькі дані, картки для інтервального повторення, графові зв’язки між нотатками та інші структуровані дані.

3.2 Розробка модуля створення та вивчення матеріалу за допомогою карток

Модуль cardService є центральним сервісом логіки роботи з навчальними картками у системі інтервального повторення (див. рисунок 3.2). Він реалізований у вигляді об'єкта з асинхронними методами, які забезпечують повний цикл CRUD-операцій над колодами та картками, а також реалізують механізм повторення відповідно до алгоритму Half-Life Regression.

Метод getDecks відповідає за отримання всіх колод, що зберігаються в базі даних. Він використовує допоміжну функцію getFromStorage, яка зчитує серіалізовані об'єкти за визначеним ключем DECKS_KEY.

Метод createDeck дозволяє створити нову колоду, приймаючи на вхід її назву та опис. Після генерації унікального ідентифікатора за допомогою generateId()

створюється новий об’єкт типу Deck, який ініціалізується порожнім списком карток, поточною міткою часу створення та оновлення, а також умовним userId, що вказує на користувача. Нова колода додається до загального списку і зберігається у локальному сховищі.

Для видалення колоди передбачено метод deleteDeck, який видаляє як саму колоду, так і пов’язані з нею картки. Це забезпечує консистентність даних, запобігаючи появі карток, що залишились без колоди і по суті стали сміттям, яке з часом буде накопичуватись і займати лишнє місце.

Метод getCards дозволяє отримати список карток, що належать до конкретної колоди. Він виконує фільтрацію карток за їхнім полем deckId, яке вказує на належність до певної колоди.

Метод createCard створює нову картку в межах заданої колоди, приймаючи на вхід питання, відповідь та, при необхідності, період напіврозпаду знання і коефіцієнт чутливості. Картка ініціалізується з початковими значеннями reviewCount = 0, successRate = 0, а також встановлюється початковий час перегляду і час наступного перегляду. Після цього нова картка додається до бази даних.

Оновлення картки виконується за допомогою методу updateCard, який приймає ідентифікатор картки та часткові зміни. Якщо картку знайдено, її поля оновлюються, включно з міткою часу updatedAt.

Метод deleteCard дозволяє видалити картку за її унікальним ідентифікатором, виключаючи її зі списку та оновлюючи базу даних.

Найважливішим методом модуля є reviewCard, який реалізує логіку інтервального повторення за алгоритмом Half-Life Regression. Цей метод оновлює параметри картки залежно від того, чи правильно відповів користувач.

Метод getDueCards дозволяє визначити картки, що потребують повторення на поточний момент. Він порівнює дату наступного перегляду кожної картки з

поточною датою і повертає лише ті, які вже «прострочені» або ніколи не переглядалися.

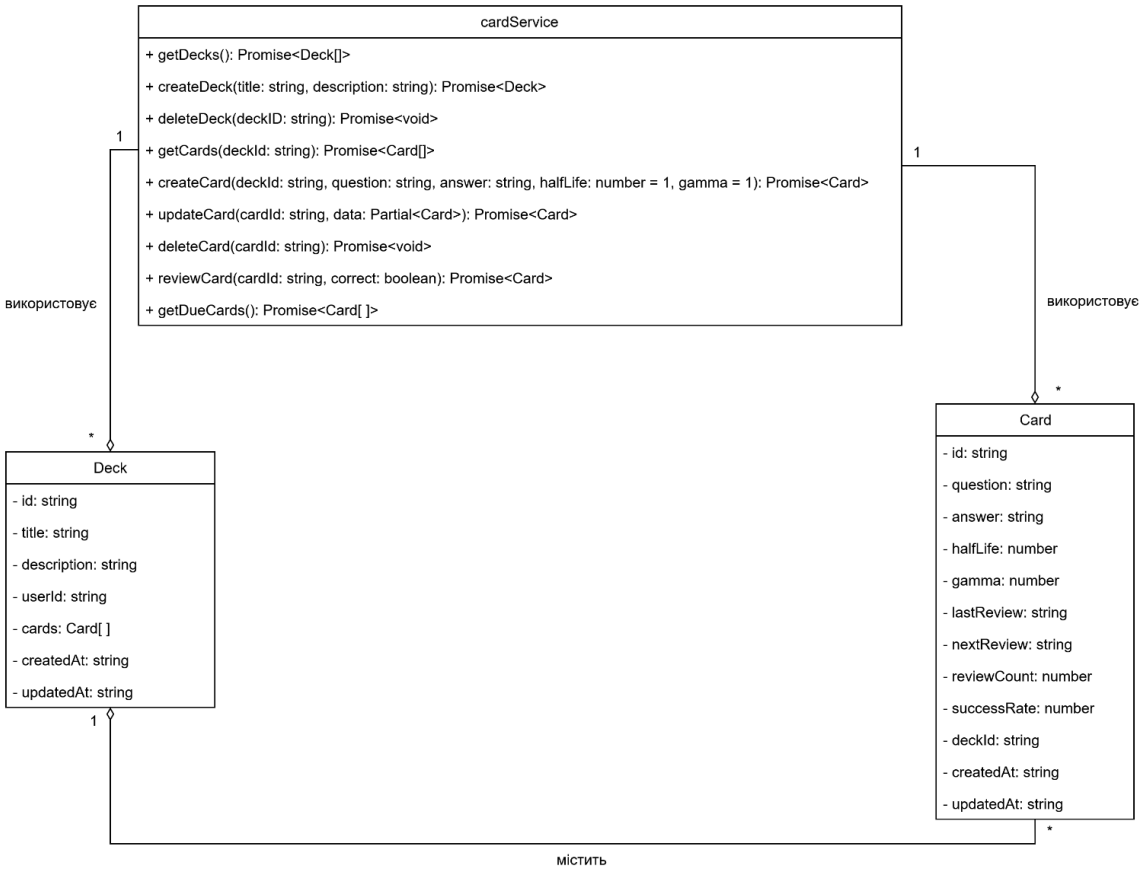

Рисунок 3.2 – Діаграма класів модуля «cardService»

Завдяки модулю cardService реалізується повноцінна логіка керування навчальним контентом від створення і видалення колод до адаптивного оновлення параметрів інтервального повторення карток. Він забезпечує зручний інтерфейс доступу до бази даних. Інтеграція методу reviewCard робить систему адаптивною до потреб конкретного користувача, підвищуючи швидкість засвоєння матеріалу відповідно до його особистого темпу навчання.

Механізм зворотних посилань реалізовано як функціональність розпізнавання та обробки спеціальних текстових маркерів у форматі

[[НазваНотатки#Розділ]], що використовуються у вмісті нотаток і карток. На етапі клієнтської обробки виконується парсинг тексту за допомогою регулярних виразів, які дозволяють виявити всі входження шаблонів зворотних посилань. Кожен знайдений маркер розбивається на назву цільової нотатки та заголовок її секції. Після цього фронтенд надсилає запит на серверний API, в якому передається інформація про назву нотатки та назву секції.

На сервері, побудованому за допомогою Node.js та Express, реалізовано логіку обробки таких запитів. Спочатку виконується пошук нотатки в базі даних за допомогою ORM-бібліотеки Prisma. У випадку успішного знаходження нотатки її вміст розбирається на структурні блоки Markdown, після чого здійснюється пошук відповідної секції за заголовком. Якщо знайдено збіг, сервер повертає фрагмент тексту або HTML-вміст секції для подальшого відображення на клієнті. Якщо ж нотатку або секцію знайти не вдається, користувачу повертається повідомлення про помилку.

Зворотні посилання виконують не лише функцію гіперпосилань для навігації, але й слугують основою для формування графу, який полегшує розуміння взаємозв’язків між матеріалами. Реалізація такої функції передбачає інтеграцію клієнтського парсингу, серверної обробки запитів, збереження структури зв’язків у базі даних і інтерактивної візуалізації.

3.3 Розробка модуля оцінювання знань у форматі інтерв’ю

Модуль «Інтерв’ю» реалізовано як клієнт-серверну систему, у якій фронтенд взаємодіє з API для ініціалізації та підтримки діалогу між користувачем і нейронною мережею на основі вмісту обраної нотатки. При відкритті компонента Interview.tsx ініціалізується стан, у якому isModalOpen встановлено в true, що активує рендеринг компонента NoteSelectionModal. Цей компонент завантажує нотатки користувача через виклик API, реалізованого у src/api/notes.ts, застосовує

фільтрацію на стороні клієнта, а після вибору конкретної нотатки викликає зворотний виклик handleNoteSelect, який передає об'єкт Note до батьківського компонента та закриває вікно. Після цього активується ChatInterface.tsx, який ініціює перше повідомлення шляхом виклику функції sendInterviewMessage, передаючи noteId, текст першого повідомлення і порожній масив історії.

На сервері маршрут POST /api/interview, оголошений у server/routes/interview.ts, виконує перевірку токена автентифікації, дістає текст нотатки з бази даних за допомогою Prisma ORM і передає всі вхідні параметри до сервісної функції generateInterviewResponse, яка міститься у server/services/LMService.ts. У цій функції формується промпт, що складається з вмісту нотатки, попередньої історії діалогу та поточного повідомлення користувача, після чого виконується запит до нейромережі, яка повертає відповідь у вигляді тексту.

Усі повідомлення відкритої сесії інтерв’ю зберігаються у локальному стані компонента ChatInterface у масиві messages. Кожне нове повідомлення користувача додається до масиву, після чого надсилається на сервер. У відповідь приходить згенероване повідомлення, яке також додається до цього масиву. Під час кожного запиту встановлюється isLoading = true, що активує візуальну індикацію генерації відповіді.

Користувач має можливість у будь-який момент переключитися на перегляд вмісту нотатки, що забезпечується умовним рендерингом на основі isViewingNote. Перехід назад до діалогу можливий без перевантаження сторінки чи втрати історії.

Для правильної роботи режиму опитування необхідно реалізувати також роботу з нотатками. Модуль «Notes» відповідає за взаємодію з API-інтерфейсом для управління нотатками користувача в системі. Він реалізує набір асинхронних функцій, які забезпечують базові CRUD-операції, а саме створення, отримання,

оновлення та видалення нотаток. Комунікація з сервером здійснюється за допомогою HTTP-запитів до ресурсу, розміщеного за базовою адресою /api/notes.

Перша функція getNotesByUser(userId: string) надсилає GET-запит на сервер з параметром userId, що дозволяє отримати всі нотатки, пов’язані з конкретним користувачем. У разі успішної відповіді функція повертає отримані дані у вигляді масиву об’єктів, нотатки. Якщо запит не було виконано, у консоль виводиться повідомлення про помилку, і повертається об’єкт із властивістю success: false та відповідним описом помилки.

Функція createNote(note) використовується для створення нової нотатки. Вона перевіряє, чи вказано заголовок нотатки, і лише після цього надсилає POST-запит до API . Якщо запит проходить успішно, повертається результат у вигляді створеної нотатки. У разі помилки повертається об’єкт із полем error, що вказує причини помилки.

Оновлення нотаток реалізовано через функцію updateNote(id, updates), яка надсилає PUT-запит. Вона передає ідентифікатор нотатки та об’єкт з оновленими значеннями . Результатом є оновлена версія нотатки або повідомлення про помилку.

Остання функція deleteNote(id) реалізує видалення нотатки за її унікальним ідентифікатором. Запит надсилається методом DELETE, і, як і в інших випадках, результатом є об’єкт із відповіддю API, або підтвердження про видалення, або повідомлення про помилку. Таким чином, модуль забезпечує централізовану логіку управління нотатками в клієнтській частині застосунку.

3.4 Розробка контролера бази даних

У даному підрозділі представлено модуль db, який реалізує взаємодію з базою даних за допомогою ORM-бібліотеки Prisma. Цей контролер охоплює операції з користувачами, нотатками та картками для навчання, забезпечуючи централізований доступ до збережених у базі даних записів (див. рисунок 3.3).

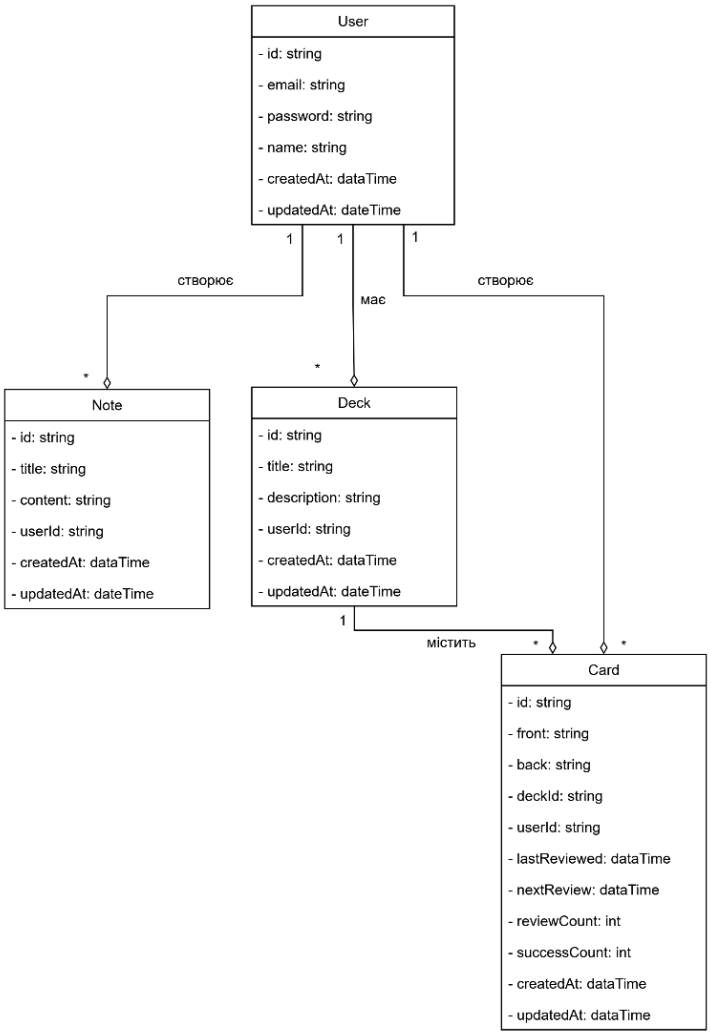

Рисунок 3.3 – Діаграма класів контролера баз даних

Модуль дозволяє створювати нового користувача за допомогою функції createUser. Перед збереженням пароль користувача хешується за допомогою бібліотеки bcryptjs, що забезпечує базовий рівень безпеки. Функція findUserByEmail дозволяє отримати користувача за його email, це корисно під час авторизації. Для перевірки правильності введеного пароля використовується метод verifyPassword, який порівнює хешований пароль з введеним.

Функція createNote дозволяє створити нову нотатку, вказавши заголовок, вміст та userId власника. Метод getNotesByUser повертає список усіх нотаток, створених конкретним користувачем, відсортованих за останнім оновленням.

Функція updateNote дозволяє оновити заголовок чи вміст нотатки за її ідентифікатором, а deleteNote видаляє нотатку з бази.

Модуль підтримує створення навчальних карток через функцію createCard, де вказуються обидві сторони картки, ID колоди та власник. Метод getCardsByUser повертає список карток, створених конкретним користувачем. Оновлення вмісту картки відбувається через updateCard, а видалення через deleteCard.

Функція updateCardReview відповідає за оновлення статистики перегляду картки. Вона визначає, чи відповів користувач правильно, і на основі цього оновлює дату наступного перегляду, кількість переглядів та рівень успішності.

Таким чином модуль db ізольовано керує всіма основними CRUD-операціями з базою даних і забезпечує логіку перегляду та навчання за допомогою карток. Він служить проміжним шаром між клієнтською логікою та низькорівневою взаємодією з базою даних.

3.5 Розробка модуля інтерфейсу програмного застосунку

Проєктування інтерфейсу програмного застосунку виконувалося з урахуванням принципів зручності користування, інтуїтивної навігації та швидкого доступу до основних функціональних можливостей. Особлива увага приділялася забезпеченню логічної структури розташування елементів керування, узгодженості стилю та підтримці сучасних вимог до адаптивності інтерфейсу [26].

Інтерфейс програмного додатку розроблено таким чином, щоб забезпечити швидкий доступ до основних функцій системи та зручність користування. Основна навігація здійснюється через бічну панель «Sidebar», яка містить кнопки для переходу між основними модулями (див. рисунок 3.4). Передбачено можливість згортання або розгортання панелі за допомогою спеціальної кнопки, що дозволяє користувачу виділити більше простору в інтерфейсі для основного компоненту.

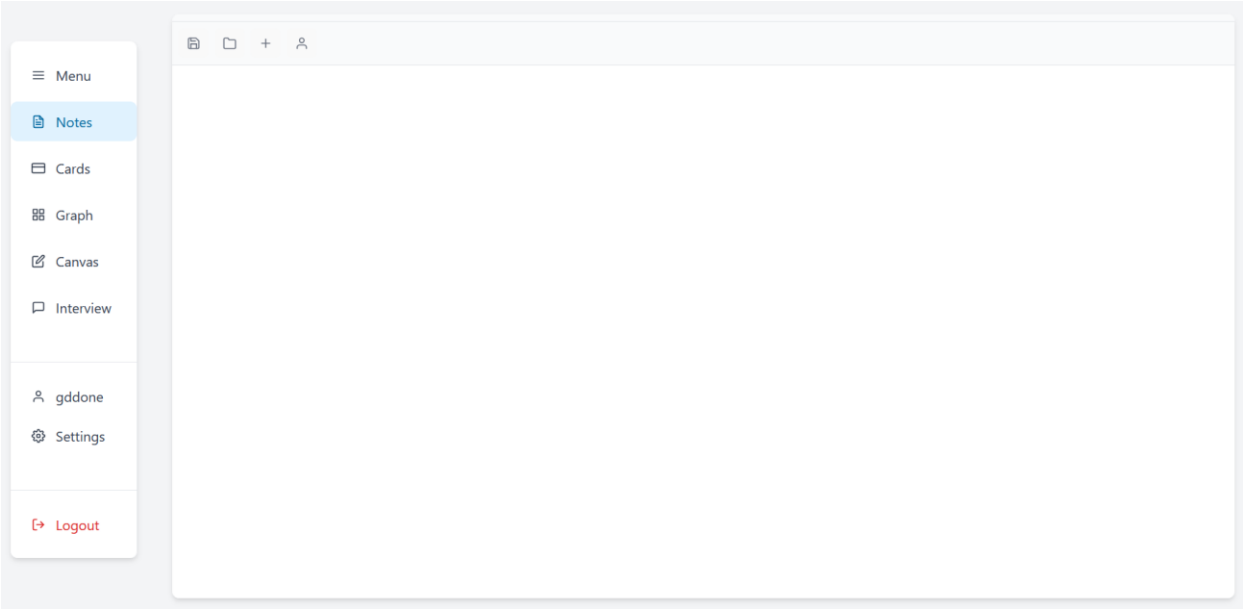

Рисунок 3.4 – Інтерфейс головної сторінки

Основні вкладки навігації надають доступ до модулів роботи з нотатками, картками, графом знань та Canvas. Кожна вкладка позначена відповідною піктограмою для швидкої ідентифікації, «Notes» для доступу до функціоналу створення та редагування нотаток, «Cards» для роботи з навчальними картками, «Graph» для візуалізації зв'язків між документами та «Canvas» для створення діаграм і малюнків.

Інтерфейс модуля «Notes» реалізує можливість відкривати кілька нотаток одночасно у вигляді вкладок (див. рисунок 3.5). Вбудований редактор тексту підтримує базові функції форматування, включно з виділенням жирним шрифтом, курсивом, створенням списків, вставкою коду та вирівнюванням тексту. Для збереження змін у нотатках використовується кнопка Save, а також передбачено можливості відкриття вже існуючих нотаток, створення нових документів та закриття активної вкладки.

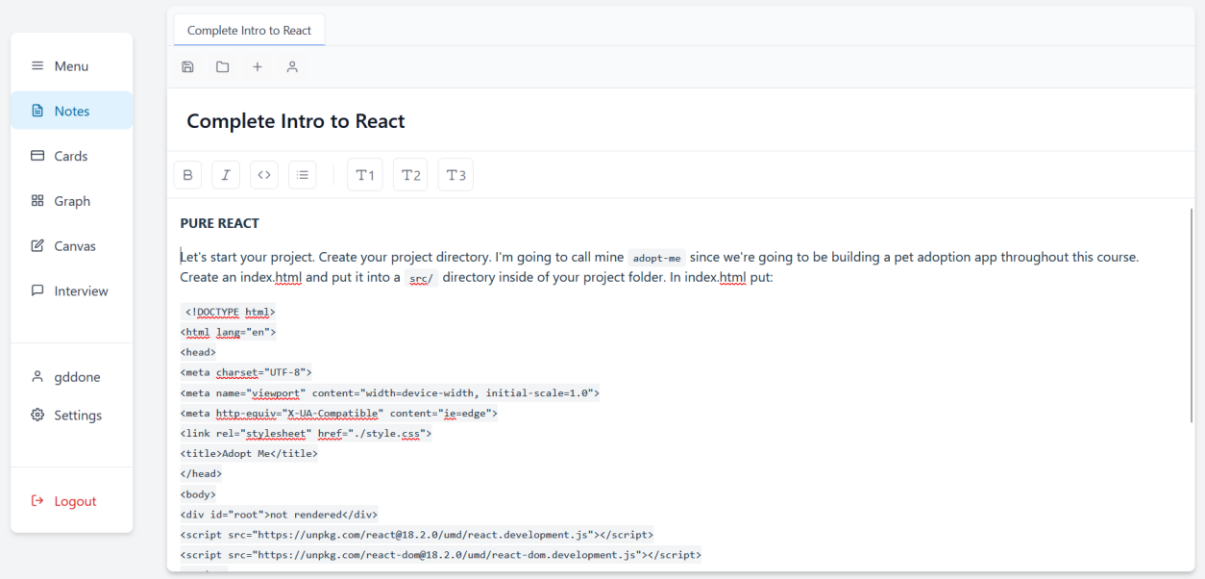

Рисунок 3.5 – Інтерфейс модуля «Notes»

Інтерфейс модуля «Cards» надає можливість переглядати всі існуючі колоди карток, створювати нові колоди, додавати нові картки до колод, редагувати та видаляти їх (див. рисунок 3.6). Для початку навчальної сесії реалізовано режим «Study Mode». Кожна колода має власну панель управління для перегляду вмісту, редагування властивостей або видалення.

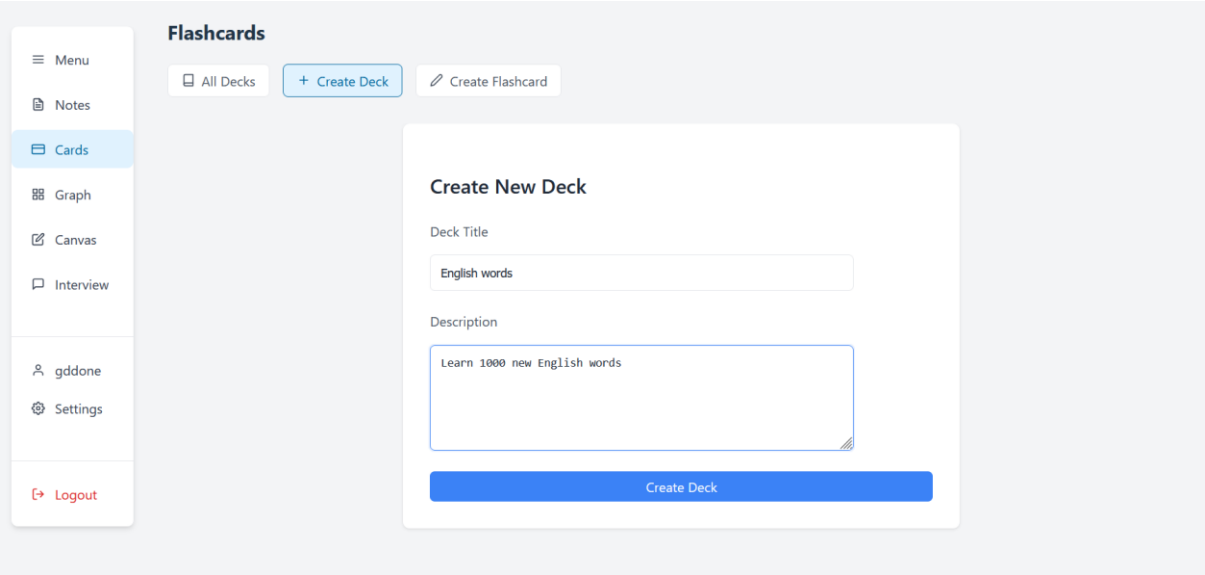

Рисунок 3.6 – Інтерфейс модуля «Cards»

Модуль «Graph» дозволяє шукати документи за допомогою рядка пошуку та взаємодіяти з візуалізованими вузлами (див. рисунок 3.7). При наведенні на вузол відображається коротка інформація, а при натисканні відкривається детальна інформація про документ і перелік пов'язаних об'єктів. Вузли графа можна перетягувати для кращого візуального упорядкування.

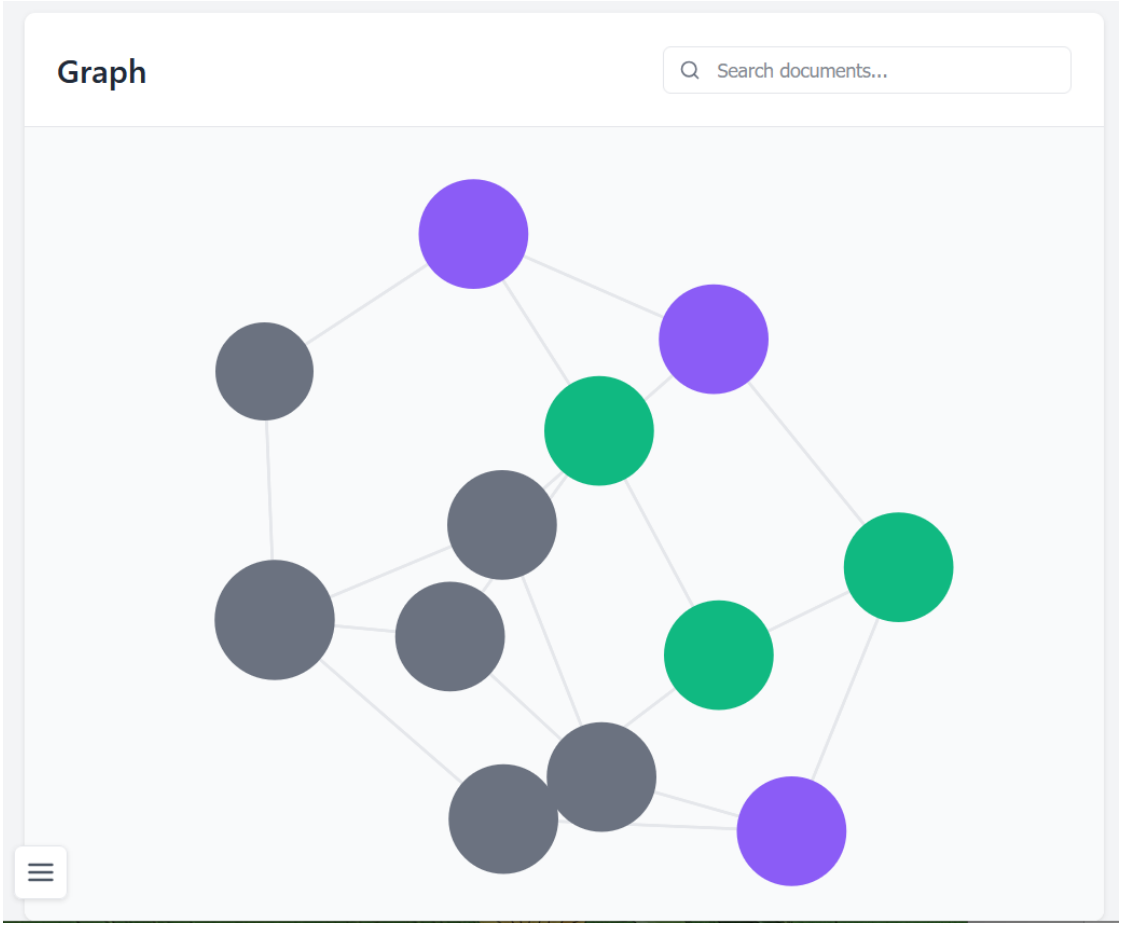

Рисунок 3.7 – Модуль «Graph»

Модуль «Canvas» забезпечує інструменти для малювання, створення фігур, вставки тексту та вибору кольорів (див. рисунок 3.8). Передбачено можливість масштабування полотна, переміщення області перегляду, скасування та повторення останніх дій, а також збереження або завантаження створених діаграм.

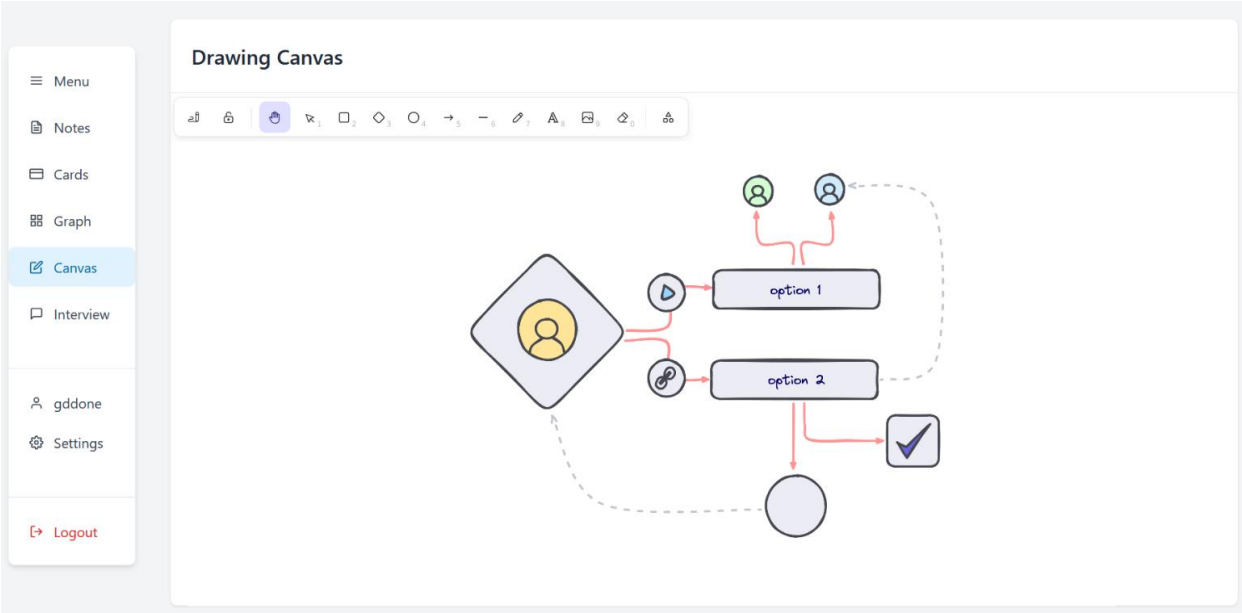

Рисунок 3.8 – Модуль «Canvas»

Модуль «Interview» являється альтернативним режимом вивчення матеріалу. В ньому користувач відповідає на запитання по конкретному конспекту у вигляді діалогу з штучним інтелектом, що дозволяє оцінити знання і вказати на помилки не по одному конкретному запитанню, а цілому конспекту без втрати контексту (див. рисунок 3.9).

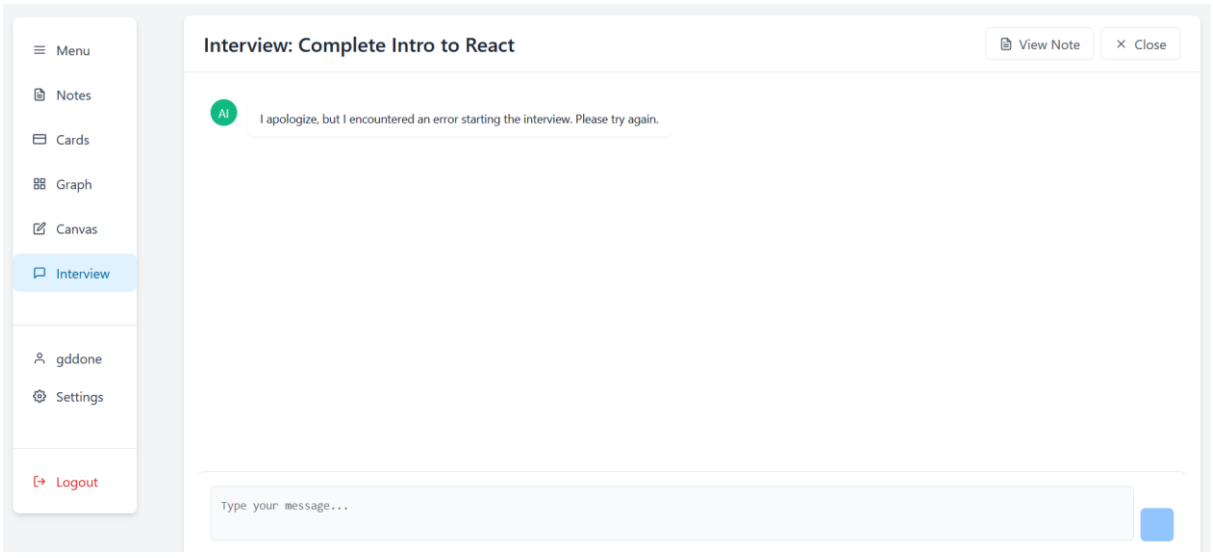

Рисунок 3.9 – Модуль «Interview»

Після запуску модуля користувач спершу бачить вікно вибору конспекту, де можна знайти потрібну нотатку через пошук по списку. Після вибору нотатки відкривається головний чат-інтерфейс, в якому користувача зустрічає штучний інтелект, ініціюючи перше питання за вмістом обраного конспекту. Якщо користувач натискає «Переглянути нотатку», інтерфейс перемикається у режим перегляду повного тексту.

У нижній частині бічної панелі розміщено елементи управління профілем користувача, налаштуваннями додатку та виходом із системи. Перехід до профілю відкриває інформацію про користувача та статистику навчання, налаштування дозволяють змінити особисті дані та параметри навчання, а кнопка виходу дозволяє завершити сесію.

Панель профілю користувача містить базову інформацію про ім'я та електронну пошту, а також статистику навчальної активності, що відображає щоденні, тижневі та місячні результати (див. рисунок 3.10).

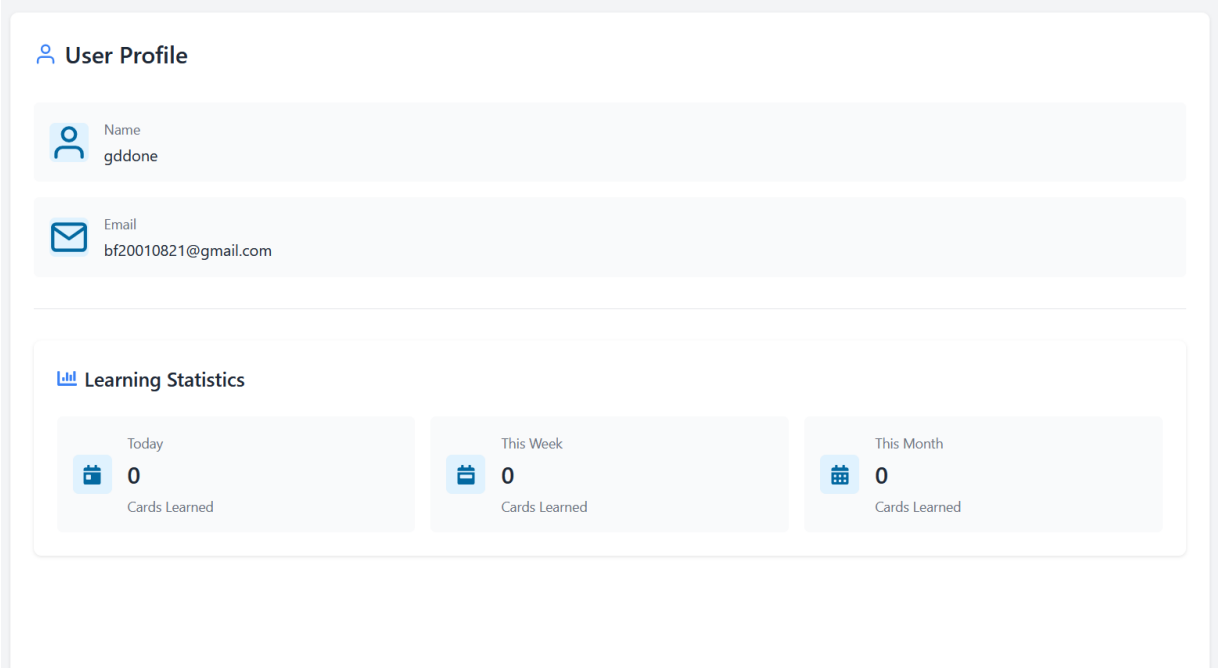

Рисунок 3.10 – Профіль і статистика користувача

У панелі налаштувань реалізовано зміну пароля через введення поточного і нового паролів, а також конфігурацію параметрів навчання, таких як встановлення щоденної мети, налаштування інтервалів повторення, активація підказок та випадкове перемішування карток під час вивчення.

Інтерфейс аутентифікації включає окремі форми для входу та реєстрації користувачів (див. рисунок 3.11). Форма входу очікує введення електронної пошти та пароля, а форма реєстрації додатково вимагає введення імені та підтвердження пароля. Для зручності передбачено швидкі посилання для переходу між формами входу та реєстрації.

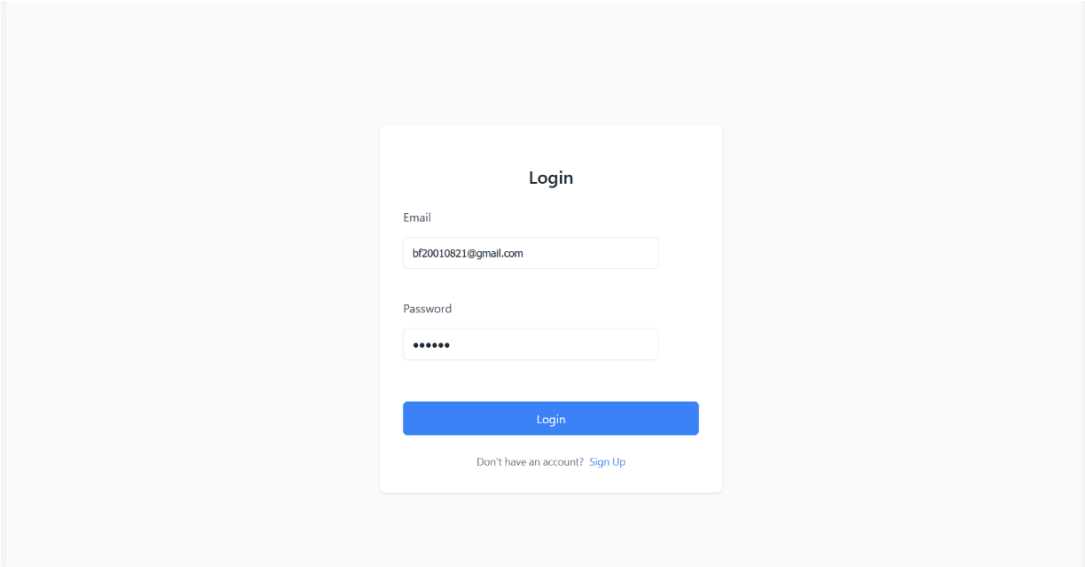

Рисунок 3.11 – Інтерфейс авторизації користувача

Дана організація елементів інтерфейсу дозволила створити інтуїтивно зрозуміле середовище, яке забезпечує просту та інтуїтивну взаємодію користувача з усіма модулями системи та сприяє досягненню навчальних цілей з мінімальними зусиллями.

3.6 Висновки

Розроблено модуль створення нотаток із підтримкою зворотних посилань і побудовою графа взаємозв’язків на базі React Force Graph 2D. Для цього створено RESTful API з асинхронними CRUD-функціями. Реалізовано інтеграцію з штучним інтелектом для автоматичної перевірки текстових відповідей, що забезпечує об’єктивну оцінку знань і зворотний зв’язок.

Інтегровано базу даних через Prisma з підтримкою PostgreSQL. Контролер охоплює операції з користувачами, нотатками та навчальними картками. Створено модуль cardService, що реалізує алгоритм Half-Life Regression для персоналізованого інтервального повторення, з повною підтримкою CRUD-функціоналу.

Розроблено модуль інтерв’ю, який генерує питання на основі обраної нотатки та дозволяє тренувати знання у діалоговому форматі. Графічний інтерфейс реалізовано на React із використанням TypeScript, Vite, TipTap і Excalidraw, з модульною структурою та підтримкою аутентифікації, що забезпечує зручну навігацію між основними розділами системи.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Тестування програмного забезпечення проводилося з метою всебічної перевірки правильності функціонування всіх компонентів системи, коректністю обробки виняткових ситуацій, відповідності очікуваним сценаріям користувацької поведінки та забезпечення конфіденційності і цілісності даних користувача. Особливу увагу було зосереджено на перевірці взаємодії користувача з інтерфейсом системи у типових ситуаціях, а також у випадках, що виходять за межі штатної експлуатації [27].

У рамках перевірки модуля автентифікації тестувалися обидва сценарії успішної та неуспішної авторизації. Перевірка включала введення коректних облікових даних, а також моделювання поширених помилок, таких як введення неправильного пароля або електронної адреси, відсутність введених даних у полях форми, використання надто короткого пароля, а також невідповідність пароля при його повторному введенні під час реєстрації (див. рисунок 4.1).

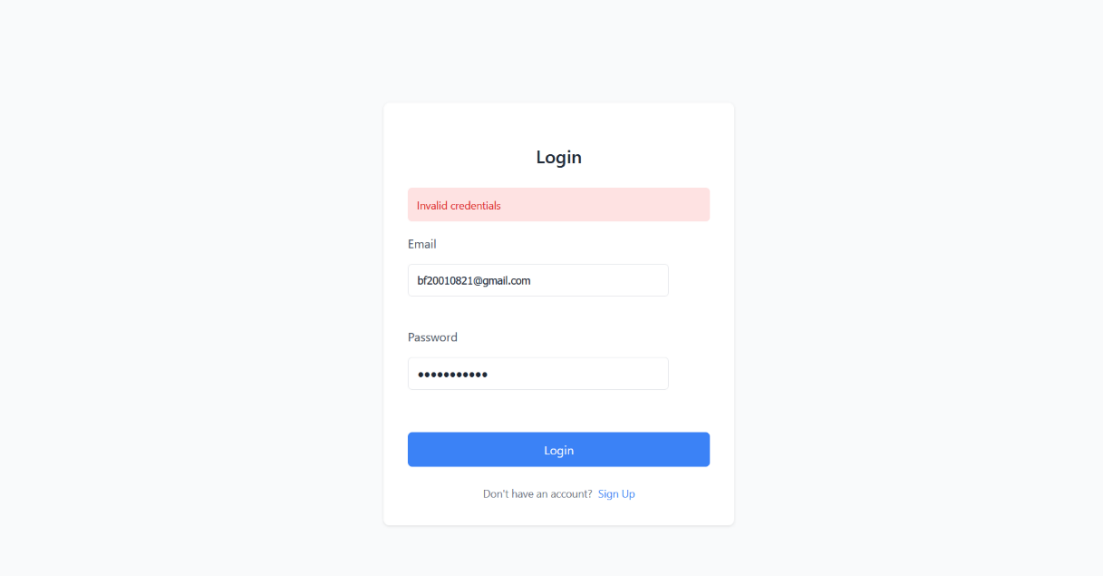

Рисунок 4.1 – Попередження про помилку

Для перевірки функціональності компонента Cards було проведено поетапне ручне тестування його основних вкладок, функції перегляду колод, створення нової колоди, додавання карток та запуск режиму вивчення. Під час тестування основна увага зверталась на коректність відображення інтерфейсу, роботу з базою даних через cardService, валідацію форм, обробку помилок і відображення повідомлень для користувача (див. рисунок 4.2).

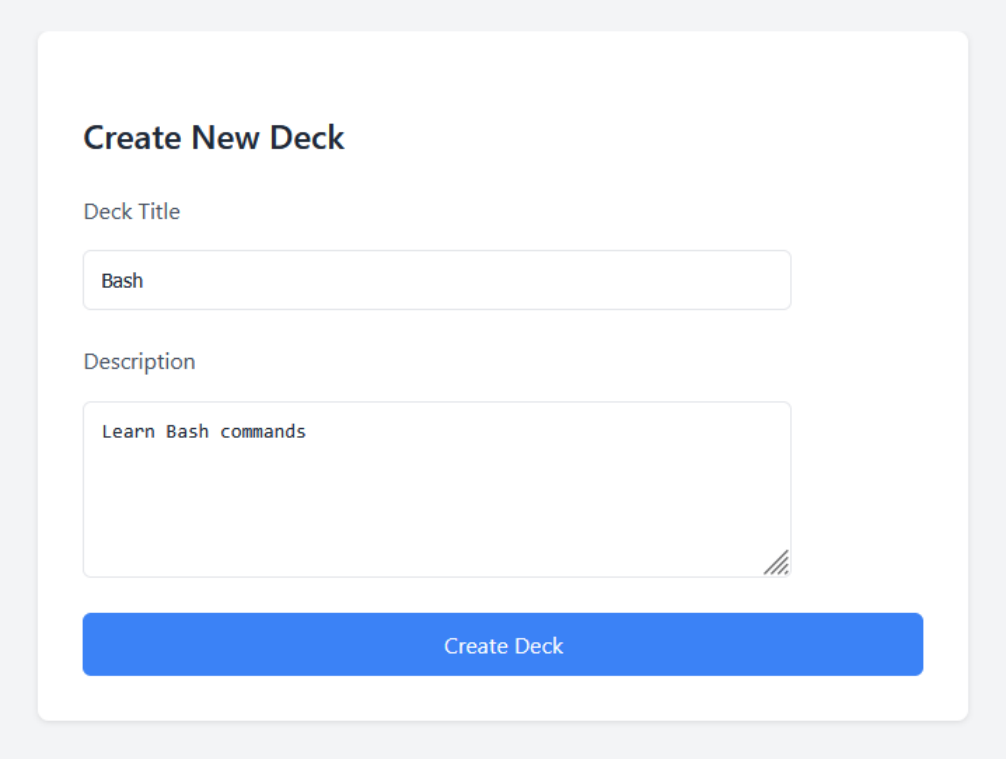

Рисунок 4.2 – Створення колоди

На першому етапі тестування було виконано перехід до вкладки Create Deck, після чого введено назву нової колоди «Bash» і короткий опис «Learn Bash commands». Після натискання кнопки «Create Deck» колода успішно збереглась, і користувач автоматично було перенаправлено на вкладку перегляду всіх колод.

На випадок некоректного заповнення, наприклад, порожнє поле назви або опису, форма блокувала відправку і підсвічувала відповідні поля (див. рисунок 4.3).

У разі симуляції помилки серверу, на екрані з’являлось повідомлення «Failed to create deck».

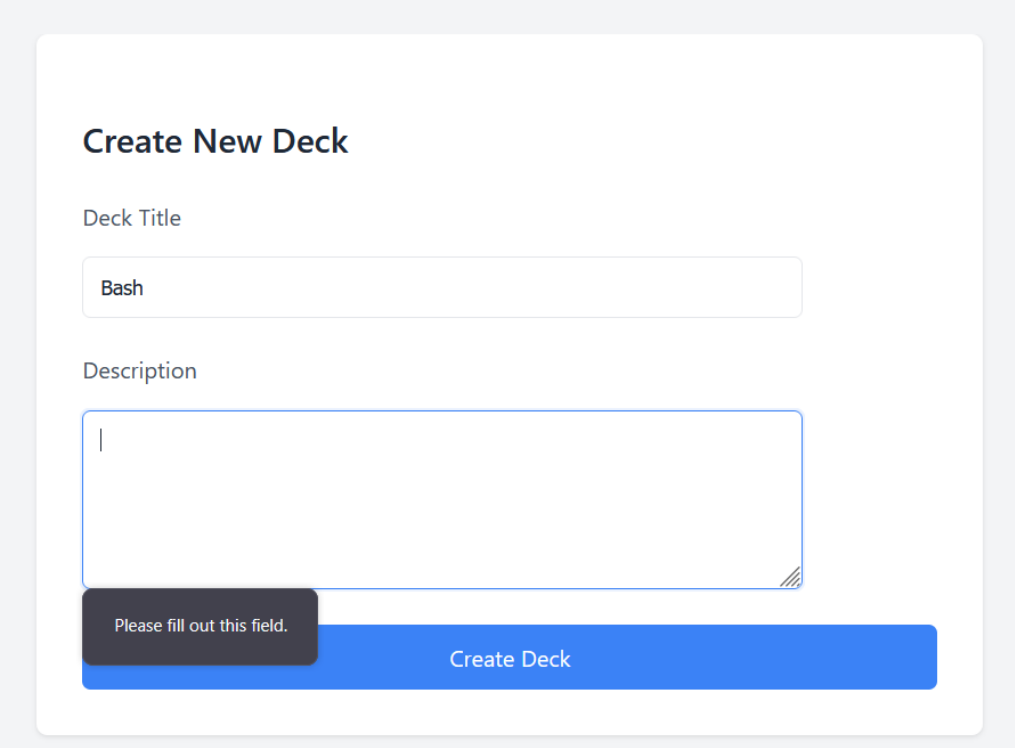

Рисунок 4.3 – Повідомлення про необхідність ввести дані в поле

Після створення колоди було протестовано вкладку Create Flashcard. У формі створення картки було вибрано новостворену колоду, введено запитання і відповідь. Параметри halfLife і gamma залишалися за замовчуванням. Також було вручну перевірено поведінку форми при незаповнених обов’язкових полях. Наприклад, якщо користувач не обирає колоду або залишає порожнім питання, кнопка створення стає неактивною, і валідація запобігає відправці форми. При штучному введенні нечислових значень у поля halfLife або gamma форма відхиляє введення, що підтверджує наявність валідації типів (див. рисунок 4.4). Після успішного створення картки вона одразу з’явилась у списку карток відповідної колоди, що свідчить про оновлення стану інтерфейсу в реальному часі.

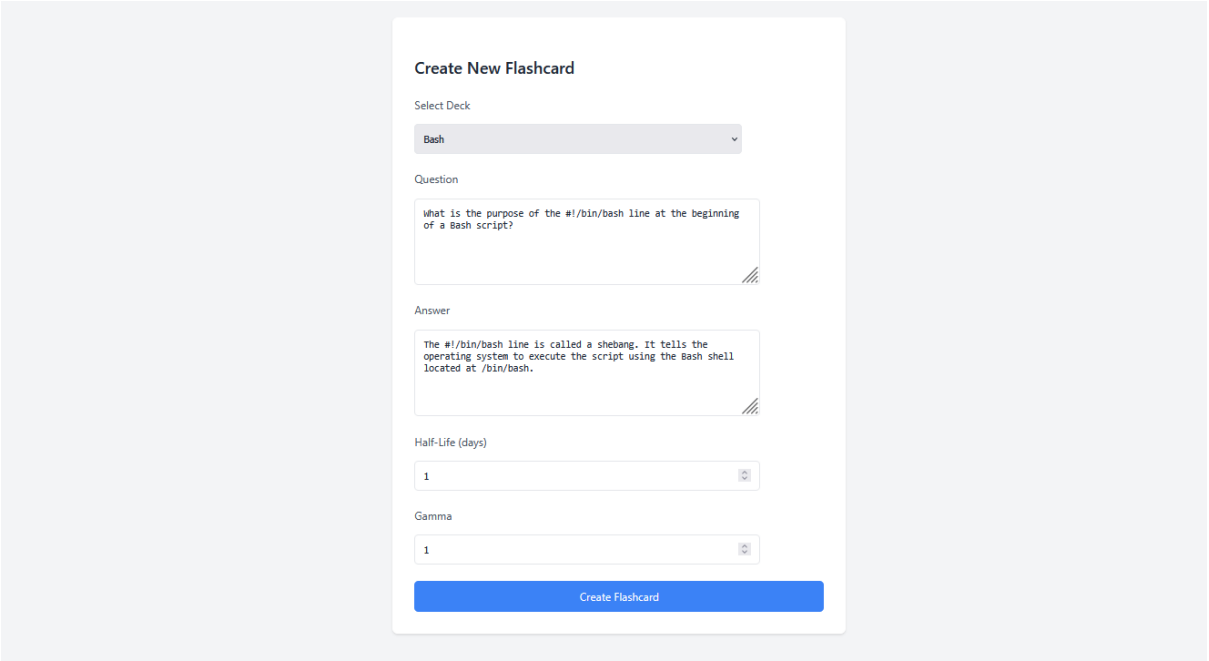

Рисунок 4.4 – Заповнена форма створення нової картки

На рисунку 4.5 за допомогою pgAdmin відображено доданий запис, створений для збереження нової картки в базі даних PostgreSQL.

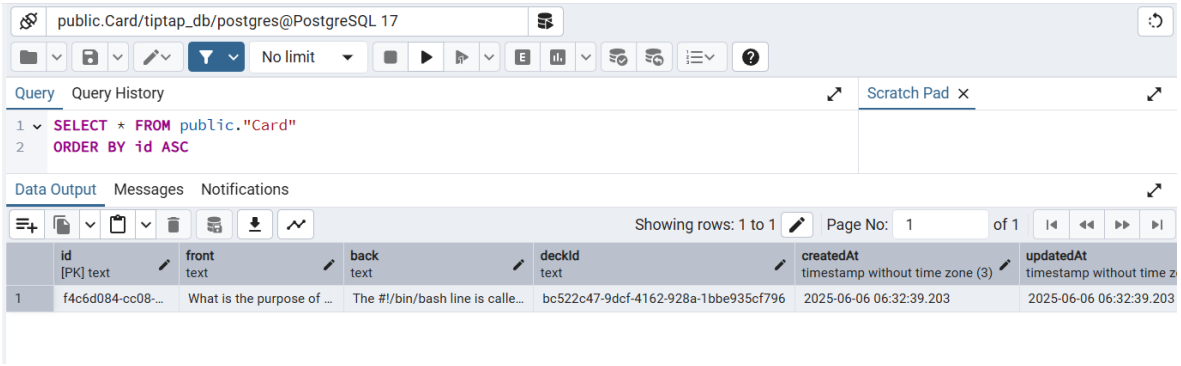

Рисунок 4.5 – Новий запис для збереження картки

Далі, при переході до вкладки All Decks, у таблиці колод відображалась створена колода з відповідною статистикою, загальною кількістю карток, кількістю нових та тих, що підлягають повторенню (див. рисунок 4.6). Натискання на кнопку «Start Studying» запускало режим вивчення. У випадку, коли в колоді ще не було

жодної картки з терміном повторення або нової картки, кнопка повторення ставала неактивною.

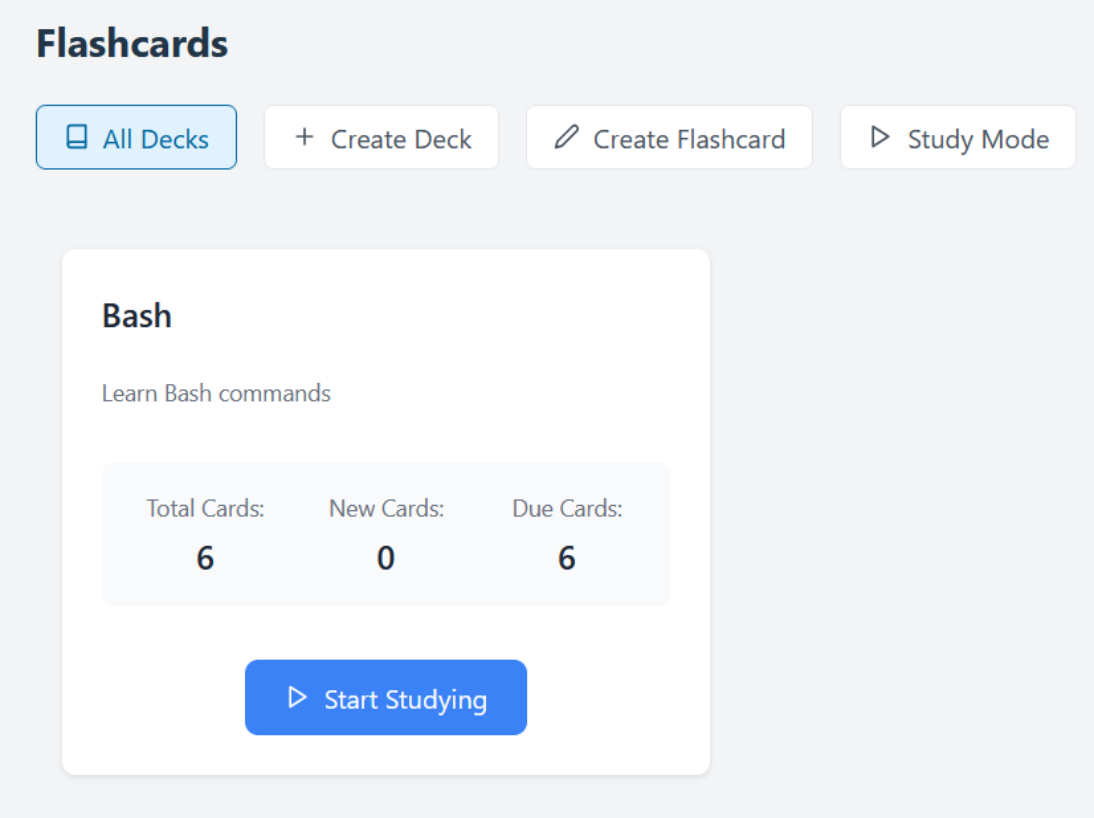

Рисунок 4.6 – Відображення створеної колоди з картками

В режимі вивчення тестувалась логіка переходу між питанням і відповіддю, а також реакція на натискання кнопок «Correct» та «Incorrect». При кожній відповіді викликався асинхронний запит до бекенду для оновлення статистики картки. У разі втрати з’єднання під час відповіді з’являлось повідомлення про помилку з сервера, що підтверджувало обробку помилок і на цьому етапі.

Після завершення усіх дій було виконано перевірку бази даних вручну через Prisma Studio. На рисунку 4.7 зображено дані про створену колоду, картку і її параметри збереження, які правильно збереглись та відповідають тим, що вводились під час тестування.

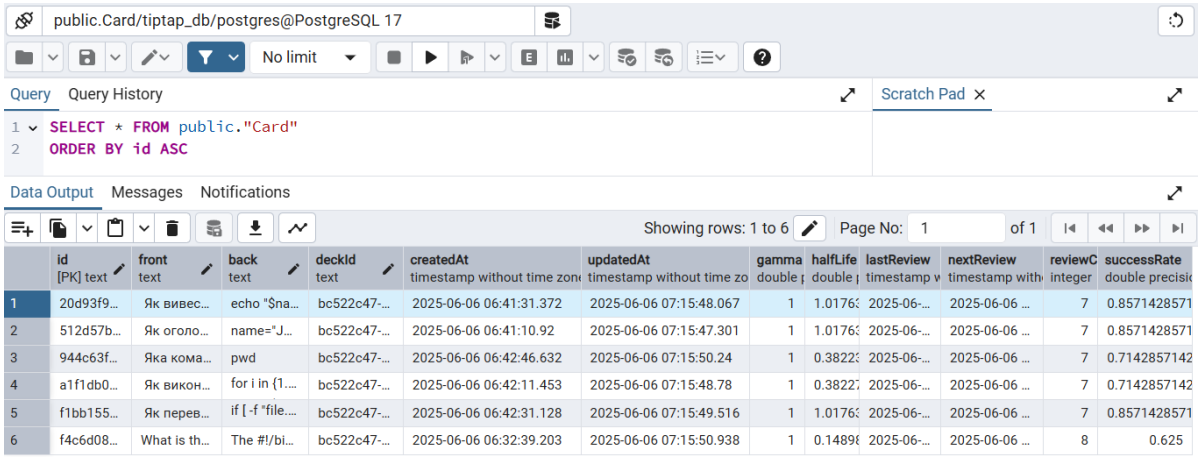

Рисунок 4.7 – Записи створених карток в базі даних

Окремо тестувалась функція видалення колоди. Після натискання кнопки «Delete», колода зникала зі списку, а її запис з бази даних також видалявся. У разі, якщо запит видалення не проходив, користувач отримував повідомлення «Failed to delete deck».

Таким чином, компонент «Cards» продемонстрував стабільну роботу з даними, коректне управління станами завантаження, правильну валідацію форм і інформативну систему обробки помилок.

Під час тестування модуля «Notes» було перевірено основні функції створення, редагування, збереження, вибору та завантаження нотаток. Тестування проводилось у звичайному користувацькому режимі через інтерфейс застосунку.

Спочатку було створено нову нотатку, натиснувши на кнопку додавання, було введено назву, у текстове поле нотатки було додано і відформатовано конспект Після цього було натиснуто кнопку збереження. Після успішного створення, на екрані з’явилось повідомлення «Note saved successfully!».

Під час тестування було також імітовано помилку з’єднання з мережею. Для цього при збереженні нотатки з’єднання з сервером було вимкнено. Система зловила помилку та показала повідомлення про помилку «Network error while saving...» (див. рисунок 4.8).

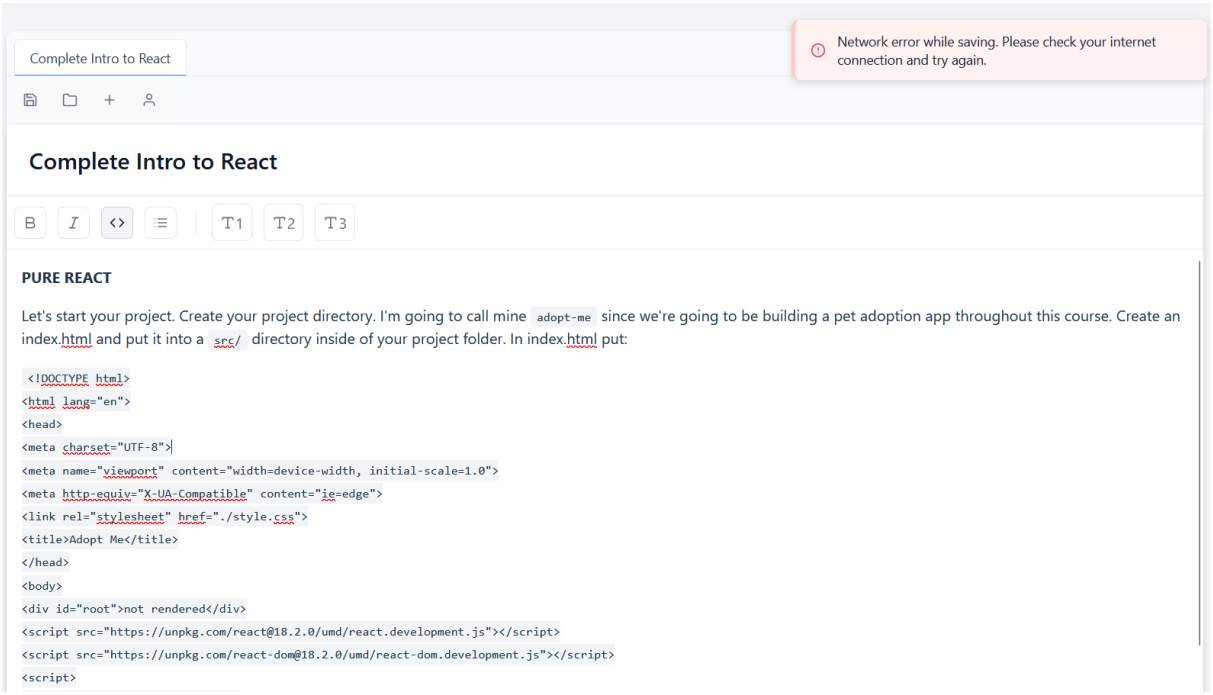

Рисунок 4.8 – Відображення помилки про втрату з’єднання

Зрештою, після завершення всіх дій було відкрито базу даних, де підтверджено, що заголовки, контент, ID користувача, дати створення та оновлення нотаток збереглись правильно та були актуальними. Для збережених нотаток збігались як вміст, так і HTML-форматування (див. рисунок 4.9).

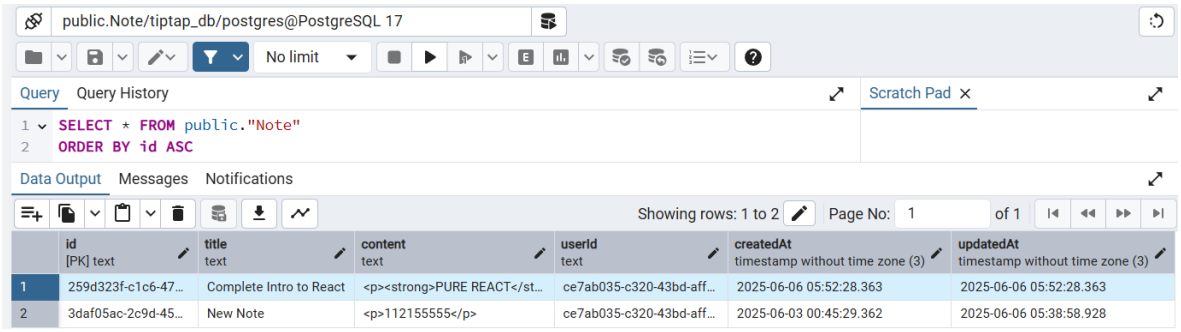

Рисунок 4.9 – Запис створеної нотатки в базі даних

Далі було протестовано відкриття раніше збережених нотаток. Успішне завантаження відкрило вікно вибору нотаток із їхніми заголовками та датами

останнього оновлення. Один з елементів списку було обрано, після чого відповідна нотатка відкрилась, а її вміст завантажився в редактор. У разі виникнення помилки API, система показала повідомлення «Failed to load notes» та не відкривала модальне вікно (див. рисунок 4.10).

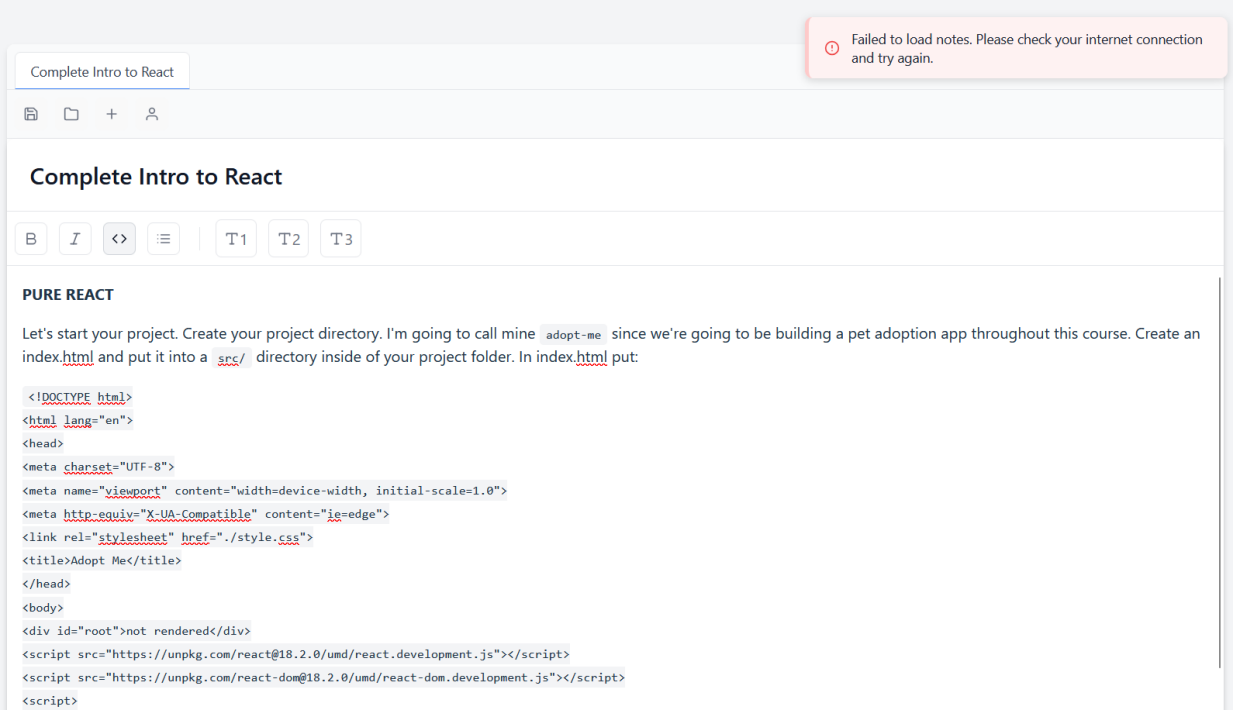

Рисунок 4.10 – Відображення помилки відкриття нотатки

Далі було протестовано функціональність керування нотатками, що включає створення, редагування, збереження змін і видалення об’єктів. Система коректно реагувала на введення некоректних або неповних даних, забезпечуючи своєчасне попередження користувача про помилку. Перевірялися також ситуації втрати з’єднання з сервером, що моделювалися вручну, аби оцінити стійкість модуля до зовнішніх факторів. У разі виявлення неавторизованого доступу система коректно перенаправляла користувача на сторінку входу для повторної автентифікації, попередньо відобразивши помилку.

Модуль візуалізації графа знань також пройшов тестування на надійність у типових сценаріях, наприклад, вибір вузлів, виконання пошуку, завантаження пов’язаних даних. Система успішно справлялася із запитами навіть у разі часткових збоїв, забезпечуючи плавний користувацький досвід та уникнення втрати даних. Модуль малювання на інтерактивному полотні дозволяв створювати, редагувати та зберігати графічні об’єкти. Під час тестування виявлено, що система здатна вчасно реагувати на спроби некоректних дій з боку користувача або перевантаження клієнтської частини пам’яті, повідомляючи про ці помилки відповідними повідомленнями.

Ще одним важливим аспектом стала перевірка модуля керування налаштуваннями. Система демонструвала стабільність при оновленні параметрів навчання, зміні пароля або інших персоналізованих параметрів. У разі виявлення некоректних даних користувач отримував повідомлення з підказкою щодо правильного заповнення (рисунок 4.11).

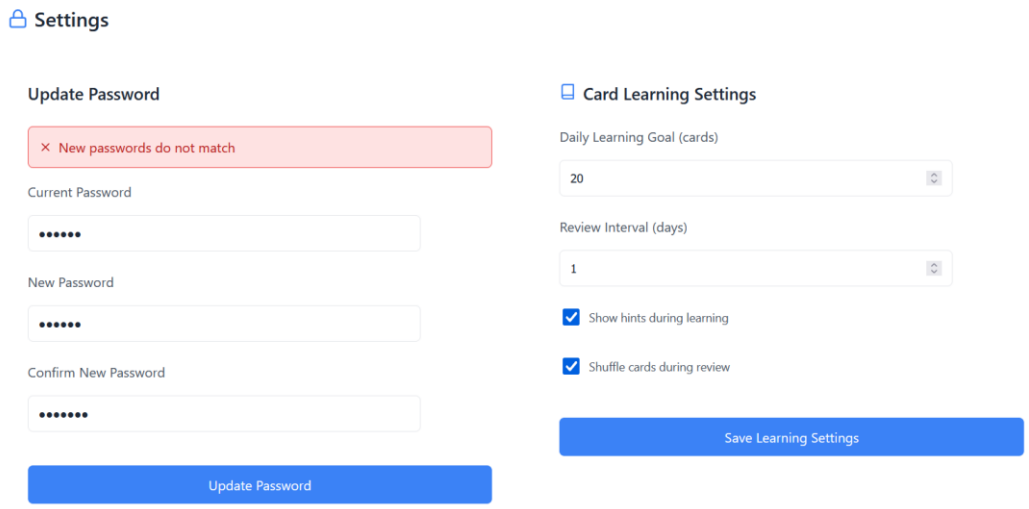

Рисунок 4.11 – Попередження про несумісність даних

Крім того, було протестовано модуль управління профілем користувача. Система належно обробляла як успішні оновлення, так і спроби внесення недійсних змін або неавторизованого редагування. Аналіз модуля збору статистики підтвердив, що система здатна відображати навчальний прогрес користувача в узагальненому вигляді, враховуючи вибраний часовий діапазон. У разі введення некоректного діапазону або відсутності статистичних даних система не зупиняла роботу через помилки, а інформувала користувача відповідним повідомленням.

Під час тестування модуля «Interview» було перевірено основний сценарій взаємодії користувача з інтерфейсом штучного інтелекту, який проводить опитування за обраним конспектом. На початку відкривається модальне вікно з переліком доступних конспектів. Користувач обирає один з них, у тестовому випадку використовувався конспект із теми «Основи React».

Після вибору конспекту, з’являється інтерфейс чату, стилізований аналогічно до класичного месенджера. Штучний інтелект одразу ініціює діалог, ставлячи перше питання по темі. Користувач надає відповідь у текстовому полі, і ШІ або ставить наступне питання, або дає уточнення чи відгук (див. рисунок 4.12). Весь діалог проходить у реальному часі, без оновлення сторінки. В разі втрати мережі, ШІ надсилає відповідне повідомлення користувачу.

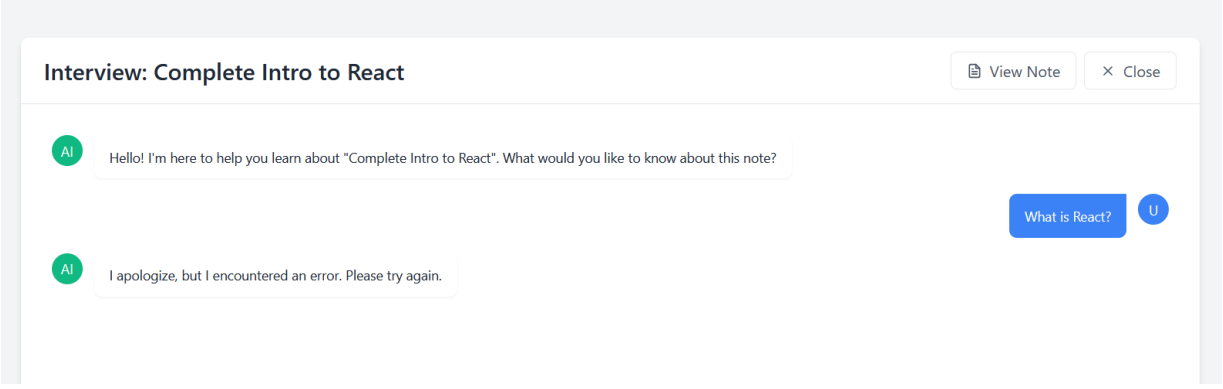

Рисунок 4.12 – Взаємодія з штучним інтелектом в режимі «Інтерв’ю»

Загалом проведене тестування підтвердило, що система відповідає базовим вимогам до сучасних вебзастосунків, включаючи стабільність, безпечність, обробку винятків, збереження цілісності даних та інформативність взаємодії з користувачем.

4.2 Розробка інструкції користувача

На таблиці 4.1 зображено визначені мінімальні системні вимоги для роботи системи.

Таблиця 4.1 – Системні вимоги

Процесор	Будь-який сучасний процесор (1,6 ГГц або вище)
RAM	Мінімум 4 ГБ (рекомендовано 8 ГБ)
Сховище	Щонайменше 500 МБ вільного місця
Веб-браузер	- Chrome 80+ - Firefox 75+ - Safari 13+ - Edge 80+

Інструкція користувача призначена для забезпечення правильної взаємодії із системою управління навчанням. Вона містить опис основних етапів взаємодії із системою, починаючи від реєстрації та навігації, і завершуючи рекомендаціями для підвищення продуктивності навчального процесу.

Щоб розпочати роботу із системою, користувач повинен створити обліковий запис, натиснувши кнопку «Register» на головній сторінці. При реєстрації необхідно заповнити поля електронної пошти, імені та пароля, при цьому пароль має містити щонайменше шість символів. Для входу в систему потрібно ввести

адресу електронної пошти та пароль, які були вказані під час реєстрації. Навігація системою здійснюється через бокове меню, де основні розділи «Notes», «Cards», «Graph», «Canvas», «Profile» та «Settings» доступні для переходу. Меню можна розгортати або згортати за допомогою відповідної іконки.

Для створення нотаток користувач переходить до розділу «Notes», де необхідно натиснути кнопку «New Note». У відповідних полях вводяться заголовок і вміст нотатки, а для форматування тексту можна скористатися панеллю стилізації. Збереження здійснюється натисканням кнопки «Save». Усі створені нотатки відображаються у списку нотаток, де користувач може вибрати будь-яку нотатку для редагування. Передбачена можливість одночасної роботи з декількома нотатками через систему вкладок. Для видалення нотатки слід натиснути на іконку кошика поруч із відповідним записом.

Процес створення карток розпочинається у розділі «Cards», де користувач натискає кнопку «Create Deck» для створення нової колоди. До колоди додаються картки із запитаннями та відповідями. За бажанням можна встановити додаткові параметри навчання. Для вивчення карток потрібно обрати відповідну колоду і перейти у режим «Study Mode». У процесі навчання спочатку відображається запитання, після чого за допомогою кнопки «Show Answer» відкривається правильна відповідь. Налаштування параметрів навчання, таких як встановлення щоденних цілей, коригування інтервалів повторення, активація підказок і зміна порядку карток, доступні через розділ налаштувань.

У розділі «Graph» користувач може переглядати взаємозв'язки між нотатками та картками у вигляді інтерактивного графа. Натиснувши на вузол можна отримати додаткову інформацію про пов'язаний об'єкт. Для пошуку певного елемента передбачено зручний пошук за ключовими словами. Користувач може змінювати розташування вузлів шляхом їх перетягування, застосовувати фільтри для

відображення певних типів об'єктів та використовувати масштабування для кращого огляду графа.

Для створення діаграм та малюнків користувач переходить до розділу «Canvas». У цьому розділі доступні інструменти малювання, створення фігур, додавання тексту, вибору кольорів та можливості скасування і повторення останніх дій. Усі зміни автоматично зберігаються системою без потреби додаткового збереження вручну.

Перегляд особистого профілю відбувається через натискання на ім'я користувача у боковій панелі. У профілі відображається особиста інформація користувача та основні показники навчальної активності. Статистика включає щоденний прогрес, щотижневі досягнення, місячний огляд результатів та рівень успішності відповідей.

У розділі налаштувань користувач має можливість оновити пароль, змінити електронну пошту або редагувати інші дані профілю. Крім того, можна налаштувати навчальні параметри, такі як встановлення щоденних цілей, коригування інтервалів повторення карток, конфігурація преференцій у процесі навчання, а також активація або деактивація додаткових функцій системи.

Для результативного використання карток рекомендується регулярно їх переглядати, чесно оцінювати свої відповіді, використовувати систему інтервального повторення та створювати чіткі й зрозумілі картки. Ведення нотаток варто здійснювати із використанням форматування для покращення структури, створенням посилань між пов’язаними записами та регулярним оновленням матеріалів. Граф знань допомагає краще візуалізувати взаємозв'язки між темами. Плануючи навчання, слід встановлювати реалістичні щоденні цілі, орієнтуватися на показники статистики для коригування навчального процесу та підтримувати стабільний графік занять.

4.3 Висновки

Проведено комплексне тестування системи для перевірки функціональності та надійності всіх компонентів. Задача була вирішена шляхом всебічної перевірки модулів системи з використанням методів функціонального та нефункціонального тестування. Тестування охопило модуль автентифікації, функціональність керування нотатками, модуль вивчення карток, граф, «Canvas», модуль опитування та модуль управління налаштуваннями. Також проведено тестування стійкості до технічних збоїв, включаючи втрату мережевого з'єднання. Підтверджено відповідність системи вимогам до сучасних вебзастосунків, включаючи стабільність, безпечність, правильну обробку винятків та збереження цілісності даних користувача.

Розроблено детальну інструкцію користувача для роз’яснення того як потрібно працювати з системою. Задача була вирішена за рахунок створення користувацького керівництва, яке охоплює всі аспекти роботи з системою від початкової реєстрації до використання просунутих функцій. Інструкція включає визначення мінімальних системних вимог, покрокові алгоритми реєстрації та авторизації, детальний опис роботи з кожним модулем системи, керування профілем та налаштуваннями, а також практичні рекомендації для підвищення продуктивності навчального процесу.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи було виконано комплекс завдань, спрямованих на створення інноваційного інструменту для навчання з інтервальним повторенням, з використанням сучасних вебтехнологій та штучного інтелекту. Проведено порівняльний аналіз існуючих рішень, на основі якого сформульовано функціональні та нефункціональні вимоги до системи. Це дозволило обґрунтовано визначити архітектурні підходи, обрати відповідні технології та забезпечити узгодженість між усіма компонентами.

Розроблено алгоритм інтервального повторення на основі Half-Life Regression, який дозволяє адаптувати навчальний процес до індивідуальних потреб користувача, автоматизовано оцінювання за допомогою штучного інтелекту та впроваджено механізм об’єктивного зворотного зв’язку. Модуль створення нотаток реалізовано з підтримкою зворотних посилань і візуалізацією зв’язків через граф, що сприяє глибшому розумінню навчального матеріалу.

Реалізовано функціонал інтерв’ю, який дає змогу оцінювати знання користувача в діалоговому форматі за допомогою штучного інтелекта, що розширює можливості інтерактивного навчання. Інтегровано базу даних із використанням ORM-бібліотеки Prisma, створено контролер для керування користувачами, нотатками та картками, що забезпечує безпечне та ефективне збереження навчальних даних.

Розроблено зручний графічний інтерфейс користувача. Проведено тестування основних функціональних блоків системи та створено інструкцію користувача.

Таким чином, усі поставлені завдання було виконано в повному обсязі, що дозволило досягти мети роботи та створити функціональний програмний засіб для навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Quantifying cognitive and affective impacts of Quizlet on learning. Frontiers in Psychology. URL: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2024.1349835/full (дата звернення: 22.03.2025)

2. Франчук Б.В, Ліщинська Л.Б. «Розробка програмного забезпечення для ефективного вивчення матеріалів за допомогою карток методом інтервального повторення» / Матеріали LIV Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії Вінницького національного технічного університету: збірник доповідей [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025.

3. Notion for Academic Research & Note-Taking. Reddit. URL: https://www.reddit.com/r/Notion/comments/ki8ju7/notion_for_academic_research_notetaking/ (дата звернення: 22.03.2025)

4. A Cohort Study Assessing the Impact of Anki as a Spaced Repetition Tool. URL: https://pubmed.ncbi.nlm.nih.gov/37546209/ (дата зверення: 23.03.2025)

5. Traverse Research. URL: https://traverse.link/research (дата звернення: 23.03.2025)

6. Active Recall and Spaced Repetition with Recall. Recall Review. URL: https://www.getrecall.ai/post/supercharge-your-memory-using-spaced-repetition-2023 (дата звернення: 20.03.2025)

7. RemNote. The All-in-One Tool for Thinking and Learning. URL: https://www.remnote.com/ (дата звернення: 23.03.2025)

8. Spacing Effect - The Decision Lab. URL: https://thedecisionlab.com/biases/spacing-effect (дата звернення: 24.03.2025)

9. Comparing spaced repetition algorithms for legal digital flashcards. ResearchGate. URL: https://www.researchgate.net/publication/387331417_Comparing_spaced_repetition_algorithms_for_legal_digital_flashcards (дата звернення: 24.03.2025)

10. The Leitner System – Study & revision: a Practical Guide. University of York. URL: https://subjectguides.york.ac.uk/study-revision/leitner-system (дата звернення: 24.03.2025)

11. Settles B., Meeder B. A Trainable Spaced Repetition Model for Language Learning. Proceedings of ACL, 2016. URL: https://www.semanticscholar.org/paper/A-Trainable-Spaced-Repetition-Model-for-Language-Settles-Meeder/cb836d2b8e126dc31ded5e674d73021604dcc6e0 (дата звернення: 25.03.2025)

12. Zaidi A. et al. Adaptive Forgetting Curves for Spaced Repetition Language Learning. Cambridge Assessment, 2019. URL: https://www.researchgate.net/publication/342677560_Adaptive_Forgetting_Curves_for_Spaced_Repetition_Language_Learning (дата звернення: 25.03.2025)

13. Spacing Effect - Wikipedia. URL: https://en.wikipedia.org/wiki/Spacing_effect (дата звернення: 26.03.2025)

14. Optimization of SM2. URL: https://www.researchgate.net/publication/386162306_Optimization_of_SM2_Algorithm_Based_on_Polynomial_Segmentation_and_Parallel_Computing (дата звернення: 25.03.2025)

15. Settles B., Meeder B. A Trainable Spaced Repetition Model. URL: https://www.researchgate.net/publication/306093511_A_Trainable_Spaced_Repetition_Model_for_Language_Learning (дата звернення: 25.03.2025)

16. Obsidian. URL: https://obsidian.md/ (дата зверення: 25.03.2025)

17. Best Practices for RESTful Web API Design. Microsoft Azure. URL: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design (дата звернення: 26.03.2025)

18. TypeScript Documentation. URL: typescriptlang.org (дата звернення: 26.03.2025)

19. React: A JavaScript library for building user interfaces. URL: react.dev (дата звернення: 26.03.2025)

20. Vue.js documentation. URL: vuejs.org. (дата звернення: 26.03.2025)

21. Angular documentaion. URL: angular.io. (дата звернення: 26.03.2025)

22. Express documentation. URL: expressjs.com. (дата звернення: 26.03.2025)

23. Fastify documentation. URL: fastify.io. (дата звернення: 26.03.2025)

24. PostgreSQL. URL: postgresql.org. (дата звернення: 26.03.2025)

25. Banks A., Porcello E. Learning React: Modern Patterns… Sebastopol: O'Reilly, 2020. 392 p.

26. REST API Best Practices: Principles & methods for testing web services. restfulapi.net. URL: https://restfulapi.net/ (дата звернення: 26.03.2025)

Додатки

66

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
« 25 » березня 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу
«Розробка програмного забезпечення для ефективного вивчення матеріалів за допомогою карток методом інтервального повторення»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
д.т.н., проф. Ліщинська Л. Б.
« 24 » березня 2025 р.

Виконав:
студент гр. 4ПІ-21б Франчук Б. В.
« 24 » березня 2025 р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка програмного забезпечення для ефективного вивчення матеріалів за допомогою карток методом інтервального повторення».

Галузь застосування – сфера навчання.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від 20 березня 2025 року ректора по ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки.

Метою дослідження є підвищення рівня засвоєння навчальних матеріалів та досягнення кращого рівня структурування великих об’ємів інформації.

Призначення роботи – розробка програмного забезпечення для ефективного вивчення матеріалів за допомогою карток методом інтервального повторення.

4. Вихідні дані для проведення НДР

1. Огляд сучасних систем інтервального повторення та платформ для навчання.

2. Публікації з когнітивної психології щодо ефективності інтервального повторення в засвоєнні знань.

3. Алгоритми планування повторень на основі кривої забування.

4. Документація до бібліотек React, TypeScript, Vite, Prisma, Node.js, Express, Tiptap, Excalidraw.

5. Опис API та можливостей GPT-моделей для реалізації інтелектуальної взаємодії.

5. Технічні вимоги

Комп’ютер з процесором та тактовою частотою 1,6 ГГц або вище.

Об’єм оперативної пам’яті 4 ГБ, сховище повинне мати щонайменше 500 МБ вільного місця.

Один з перелічених браузерів: Chrome (з версією 80 і вище), Firefox (з версією 75 і вище), Safari (з версією 13 і вище), Edge (з версією 80 і вище).

6. Конструктивні вимоги.

Вебсайт має бути зручним для користувачів, з інтуїтивно зрозумілим інтерфейсом для створення та повторення навчальних карток, роботи з нотатками та візуалізації зв’язків між ними. Система повинна бути доступною для користувачів з різним рівнем технічної підготовки та забезпечувати просту взаємодію з інструментами ШІ. Застосунок має стабільно працювати у сучасних браузерах на різних пристроях, включно з комп’ютерами, планшетами та мобільними телефонами, адаптуючи інтерфейс до розміру екрана.

7. Перелік технічної документації, що пред’являється по закінченню робіт:

- пояснювальна записка до бакалаврської кваліфікаційної роботи;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

Розробка програмного забезпечення має відповідати сучасним міжнародним стандартам веброзробки, включаючи вимоги до доступності (WCAG), безпеки персональних даних користувачів (GDPR), а також принципам модульності, масштабованості та повторного використання коду.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз розвитку систем для вивчення матеріалу та формулювання завдань дослідження	25.03.25 – 31.03.25
2	Розробка моделі та алгоритму системи	01.04.25 – 18.04.25
3	Розробка програмних модулів системи для вивчення матеріалу за допомогою карток методом інтервального повторення	19.04.25 – 07.05.25
4	Тестування програми	08.05.25 – 20.05.25
5	Оформлення матеріалів до захисту БКР	21.05.25 – 30.05.25

10. Порядок контролю та прийняття.

Всі етапи бакалаврської кваліфікаційної роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

70

Додаток Б.

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Розробка програмного забезпечення для ефективного вивчення матеріалів за допомогою карток методом інтервального повторення»

Тип роботи: Бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра програмного забезпечення, ФІТКІ
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 5,2 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Ліщинська Л. Б.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Франчук Б. В.
(підпис) (прізвище, ініціали)

Додаток В. Лістинг програми

```
import express, { Request, Response, Router, RequestHandler } from 'express';
import { PrismaClient } from '@prisma/client';
import { Session } from 'express-session';

declare module 'express-session' {
  interface Session {
    userId: string;
  }
}

const router: Router = express.Router();
const prisma = new PrismaClient();

const getNotes: RequestHandler = async (req, res): Promise<void> => {
  try {
    const notes = await prisma.note.findMany({
      where: { userId: req.session.userId },
      orderBy: { updatedAt: 'desc' },
    });
    res.json({ success: true, data: notes });
  } catch (error) {
    console.error('Error fetching notes:', error);
    res.status(500).json({ success: false, error: 'Failed to fetch notes' });
  }
};

const createNote: RequestHandler = async (req, res): Promise<void> => {
  try {
    const { title, content } = req.body;
    if (!title || !content) {
```

```
      res.status(400).json({ success: false, error: 'Missing required fields' });
      return;
    }

    if (!req.session.userId) {
      res.status(401).json({ success: false, error: 'Not authenticated' });
      return;
    }

    const newNote = await prisma.note.create({
      data: {
        title,
        content,
        userId: req.session.userId,
      },
    });
    res.json({ success: true, data: newNote });
  } catch (error) {
    console.error('Error creating note:', error);
    res.status(500).json({ success: false, error: 'Failed to create note' });
  }
};

const updateNote: RequestHandler = async (req, res): Promise<void> => {
  try {
    const { id, title, content } = req.body;
    if (!id) {
      res.status(400).json({ success: false, error: 'Note ID is required' });
      return;
    }

    if (!req.session.userId) {
```

```
      res.status(401).json({ success: false, error: 'Not authenticated' });
      return;
    }

    const note = await prisma.note.findUnique({
      where: { id },
    });

    if (!note || note.userId !== req.session.userId) {
      res.status(403).json({ success: false, error: 'Not authorized' });
      return;
    }

    const updatedNote = await prisma.note.update({
      where: { id },
      data: { title, content },
    });
    res.json({ success: true, data: updatedNote });
  } catch (error) {
    console.error('Error updating note:', error);
    res.status(500).json({ success: false, error: 'Failed to update note' });
  }
};

const deleteNote: RequestHandler = async (req, res): Promise<void> => {
  try {
    const id = req.query.id as string;
    if (!id) {
      res.status(400).json({ success: false, error: 'Note ID is required' });
      return;
    }
```

```
    if (!req.session.userId) {
      res.status(401).json({ success: false, error: 'Not authenticated' });
      return;
    }

    const note = await prisma.note.findUnique({
      where: { id },
    });

    if (!note || note.userId !== req.session.userId) {
      res.status(403).json({ success: false, error: 'Not authorized' });
      return;
    }

    await prisma.note.delete({
      where: { id },
    });
    res.json({ success: true });
  } catch (error) {
    console.error('Error deleting note:', error);
    res.status(500).json({ success: false, error: 'Failed to delete note' });
  }
};

router.get('/notes', getNotes);
router.post('/notes', createNote);
router.put('/notes', updateNote);
router.delete('/notes', deleteNote);

export default router;
import { Card, Deck } from '../types';
```

```
const DECKS_KEY = 'flashcards_decks';
const CARDS_KEY = 'flashcards_cards';

const getFromStorage = <T>(key: string): T[] => {
  const data = localStorage.getItem(key);
  return data ? JSON.parse(data) : [];
};

const saveToStorage = <T>(key: string, data: T[]): void => {
  localStorage.setItem(key, JSON.stringify(data));
};

const generateId = (): string => {
  return Math.random().toString(36).substring(2) + Date.now().toString(36);
};

export const cardService = {
  async getDecks(): Promise<Deck[]> {
    return getFromStorage<Deck>(DECKS_KEY);
  },

  async createDeck(title: string, description: string): Promise<Deck> {
    const decks = getFromStorage<Deck>(DECKS_KEY);
    const newDeck: Deck = {
      id: generateId(),
      title,
      description,
      userId: 'local-user',
      cards: [],
      createdAt: new Date().toISOString(),
      updatedAt: new Date().toISOString(),
    };
```

```
    decks.push(newDeck);
    saveToStorage(DECKS_KEY, decks);
    return newDeck;
  },

  async deleteDeck(deckId: string): Promise<void> {
    const decks = getFromStorage<Deck>(DECKS_KEY);
    const cards = getFromStorage<Card>(CARDS_KEY);

    const updatedDecks = decks.filter(deck => deck.id !== deckId);
    saveToStorage(DECKS_KEY, updatedDecks);

    const updatedCards = cards.filter(card => card.deckId !== deckId);
    saveToStorage(CARDS_KEY, updatedCards);
  },

  async getCards(deckId: string): Promise<Card[]> {
    const cards = getFromStorage<Card>(CARDS_KEY);
    return cards.filter(card => card.deckId === deckId);
  },
  async createCard(deckId: string, question: string, answer: string, halfLife: number = 1, gamma:
number = 1): Promise<Card> {
    const cards = getFromStorage<Card>(CARDS_KEY);
    const now = new Date().toISOString();
    const newCard: Card = {
      id: generateId(),
      question,
      answer,
      halfLife,
      gamma,
      lastReview: now,
      nextReview: now,
```

```
    reviewCount: 0,
    successRate: 0,
    deckId,
    createdAt: now,
    updatedAt: now,
  };
  cards.push(newCard);
  saveToStorage(CARDS_KEY, cards);
  return newCard;
},
async updateCard(cardId: string, data: Partial<Card>): Promise<Card> {
  const cards = getFromStorage<Card>(CARDS_KEY);
  const cardIndex = cards.findIndex(card => card.id === cardId);
  if (cardIndex === -1) throw new Error('Card not found');
  const updatedCard = {
    ...cards[cardIndex],
    ...data,
    updatedAt: new Date().toISOString(),
  };
  cards[cardIndex] = updatedCard;
  saveToStorage(CARDS_KEY, cards);
  return updatedCard;
},
async deleteCard(cardId: string): Promise<void> {
  const cards = getFromStorage<Card>(CARDS_KEY);
  const updatedCards = cards.filter(card => card.id !== cardId);
  saveToStorage(CARDS_KEY, updatedCards);
},

async reviewCard(cardId: string, correct: boolean): Promise<Card> {
  const cards = getFromStorage<Card>(CARDS_KEY);
  const cardIndex = cards.findIndex(card => card.id === cardId);
```

```
  if (cardIndex === -1) throw new Error('Card not found');

  const card = cards[cardIndex];
  const now = new Date();
  const dt = (now.getTime() - new Date(card.lastReview).getTime()) / (1000 * 60 * 60 * 24);
  const p = Math.exp(-dt / card.halfLife);
  const r = correct ? 1 : 0;

  const lnHNew = Math.log(card.halfLife) + card.gamma * (r - p);
  const newHalfLife = Math.exp(lnHNew);
  const nextReview = new Date(now.getTime() + newHalfLife * 24 * 60 * 60 * 1000);

  const updatedCard: Card = {
    ...card,
    halfLife: newHalfLife,
    lastReview: now.toISOString(),
    nextReview: nextReview.toISOString(),
    reviewCount: card.reviewCount + 1,
    successRate: ((card.successRate * card.reviewCount) + r) / (card.reviewCount + 1),
    updatedAt: now.toISOString(),
  };

  cards[cardIndex] = updatedCard;
  saveToStorage(CARDS_KEY, cards);
  return updatedCard;
},

async getDueCards(): Promise<Card[]> {
  const cards = getFromStorage<Card>(CARDS_KEY);
  const now = new Date();
  return cards.filter(card => {
    if (!card.nextReview) return true;
```

```
      return new Date(card.nextReview) <= now;
    });
  },
};
// src/Tiptap.tsx
import { EditorProvider, useCurrentEditor, EditorContent } from "@tiptap/react";
import StarterKit from "@tiptap/starter-kit";
import { useCallback } from "react";

const MenuBar = () => {
  const { editor } = useCurrentEditor();

  if (!editor) {
    return null;
  }

  return (
    <div className="flex gap-2 p-2 border-b bg-gray-100">
                    <button onClick={() => editor.chain().focus().toggleBold().run()}
className={editor.isActive("bold") ? "bg-gray-300 p-1 rounded" : "p-1"}>
      <b>B</b>
    </button>
                    <button onClick={() => editor.chain().focus().toggleItalic().run()}
className={editor.isActive("italic") ? "bg-gray-300 p-1 rounded" : "p-1"}>
      <i>I</i>
    </button>
                    <button onClick={() => editor.chain().focus().toggleStrike().run()}
className={editor.isActive("strike") ? "bg-gray-300 p-1 rounded" : "p-1"}>
      <s>S</s>
    </button>
        <button onClick={() => editor.chain().focus().toggleHeading({ level: 1 }).run()}
className={editor.isActive("heading", { level: 1 }) ? "bg-gray-300 p-1 rounded" : "p-1"}>
```

```
            H1
          </button>
              <button onClick={() => editor.chain().focus().toggleHeading({ level: 2 }).run()}
className={editor.isActive("heading", { level: 2 }) ? "bg-gray-300 p-1 rounded" : "p-1"}>
            H2
          </button>
                  <button onClick={() => editor.chain().focus().toggleBulletList().run()}
className={editor.isActive("bulletList") ? "bg-gray-300 p-1 rounded" : "p-1"}>
            • List
          </button>
                <button onClick={() => editor.chain().focus().toggleOrderedList().run()}
className={editor.isActive("orderedList") ? "bg-gray-300 p-1 rounded" : "p-1"}>
            1. List
          </button>
                    <button onClick={() => editor.chain().focus().setParagraph().run()}
className={editor.isActive("paragraph") ? "bg-gray-300 p-1 rounded" : "p-1"}>
            P
          </button>
          <button onClick={() => editor.chain().focus().undo().run()} className="p-1">
            Undo
          </button>
          <button onClick={() => editor.chain().focus().redo().run()} className="p-1">
            Redo
          </button>
        </div>
      );
    };

    const extensions = [StarterKit];

    const Tiptap = () => {
```

```
  return (
    <EditorProvider extensions={extensions} content="<p>Hello World!</p>">
      <div className="max-w-2xl mx-auto mt-5 border rounded-lg shadow-md">
        <MenuBar />
        <EditorContent className="p-4 min-h-[200px] bg-white border" />
      </div>
    </EditorProvider>
  );
};

export default Tiptap;

import React, { useState, useEffect, useRef } from 'react';
import { EditorContent, useEditor } from '@tiptap/react';
import StarterKit from '@tiptap/starter-kit';
import { FiBold, FiItalic, FiCode, FiList, FiEdit, FiType, FiSave, FiFolder, FiPlus, FiX, FiUser }
from 'react-icons/fi';
import { getNotesByUser, createNote, updateNote, deleteNote } from '../api/notes';
import { useAuth } from '../contexts/AuthContext';

interface Note {
  id: string;
  title: string;
  content: string;
  userId: string;
  createdAt: Date;
  updatedAt: Date;
  isLocal?: boolean;
}

const Notes: React.FC = () => {
  const { user } = useAuth();
```

```
const [notes, setNotes] = useState<Note[]>([]);
const [activeNoteId, setActiveNoteId] = useState<string | null>(null);
const [isEditingTitle, setIsEditingTitle] = useState(false);
const [newTitle, setNewTitle] = useState('');
const [isLoading, setIsLoading] = useState(false);
const [showNoteSelector, setShowNoteSelector] = useState(false);
const [availableNotes, setAvailableNotes] = useState<Note[]>([]);

const editor = useEditor({
  extensions: [StarterKit],
  content: '',
  onUpdate: ({ editor }) => {
    if (activeNoteId) {
      const content = editor.getHTML();
      updateLocalNote(activeNoteId, content);
    }
  },
});

useEffect(() => {
  if (activeNoteId && editor) {
    const note = notes.find(n => n.id === activeNoteId);
    if (note) {
      editor.commands.setContent(note.content);
      setNewTitle(note.title);
    }
  }
}, [activeNoteId, editor]);

const updateLocalNote = (id: string, content: string) => {
  setNotes(notes.map(note =>
    note.id === id ? { ...note, content, updatedAt: new Date() } : note
```

```
  ));
};

const loadAvailableNotes = async () => {
  if (!user) return;

  try {
    setIsLoading(true);
    const response = await getNotesByUser(user.id);
    if (response.success) {
      setAvailableNotes(response.data);
      setShowNoteSelector(true);
    } else {
      console.error('Failed to load available notes:', response.error);
    }
  } catch (error) {
    console.error('Error loading available notes:', error);
  } finally {
    setIsLoading(false);
  }
};

const addNewNote = () => {
  if (!user) return;

  const newNote: Note = {
    id: `local-${Date.now()}`,
    title: 'New Note',
    content: '',
    userId: user.id,
    createdAt: new Date(),
    updatedAt: new Date(),
```

```
    isLocal: true
  };
  setNotes([newNote, ...notes]);
  setActiveNoteId(newNote.id);
  setNewTitle('New Note');
  setIsEditingTitle(true);
  if (editor) {
    editor.commands.setContent('');
  }
};

const saveNote = async () => {
  if (!activeNoteId || !user) return;

  const noteToSave = notes.find(note => note.id === activeNoteId);
  if (!noteToSave) return;

  try {
    if (noteToSave.isLocal) {
      const response = await createNote({
        title: noteToSave.title,
        content: noteToSave.content,
        userId: user.id,
      });
      if (response.success) {
        setNotes(notes.map(note =>
          note.id === activeNoteId ? { ...response.data, isLocal: false } : note
        ));
        setActiveNoteId(response.data.id);
      }
    } else {
      const response = await updateNote(activeNoteId, {
```

```
        title: noteToSave.title,
        content: noteToSave.content,
      });
      if (response.success) {
        setNotes(notes.map(note =>
          note.id === activeNoteId ? { ...response.data, isLocal: false } : note
        ));
      }
    }
  } catch (error) {
    console.error('Error saving note:', error);
  }
};

const handleNoteSelect = (note: Note) => {
  const currentNote = notes.find(n => n.id === activeNoteId);
  if (currentNote?.isLocal) {
    if (!notes.some(n => n.id === currentNote.id)) {
      setNotes([currentNote, ...notes]);
    }
  }

  if (!notes.some(n => n.id === note.id)) {
    setNotes([note, ...notes]);
  }

  setActiveNoteId(note.id);
  setShowNoteSelector(false);
  if (editor) {
    editor.commands.setContent(note.content);
  }
  setNewTitle(note.title);
```

```
};

const handleTabClick = (id: string) => {
  const currentNote = notes.find(n => n.id === activeNoteId);
  if (currentNote?.isLocal) {
    if (!notes.some(n => n.id === currentNote.id)) {
      setNotes([currentNote, ...notes]);
    }
  }

  setActiveNoteId(id);
  setIsEditingTitle(false);
};

const handleTitleClick = () => {
  setIsEditingTitle(true);
};

const handleTitleChange = (e: React.ChangeEvent<HTMLInputElement>) => {
  setNewTitle(e.target.value);
  if (activeNoteId) {
    setNotes(notes.map(note =>
      note.id === activeNoteId ? { ...note, title: e.target.value } : note
    ));
  }
};

const handleTitleBlur = () => {
  setIsEditingTitle(false);
};

const handleTitleKeyDown = (e: React.KeyboardEvent<HTMLInputElement>) => {
```

```
  if (e.key === 'Enter') {
    e.currentTarget.blur();
  }
};

const createTestUser = async () => {
  try {
    const response = await fetch('http://localhost:3000/users/test', {
      method: 'POST',
    });
    const data = await response.json();
    if (data.success) {
      console.log('Test user created:', data.data);
    } else {
      console.error('Failed to create test user:', data.error);
    }
  } catch (error) {
    console.error('Error creating test user:', error);
  }
};

return (
  <div className="notes-container">
    <div className="tabs-container">
      <div className="tabs-list">
        {notes.map(note => (
          <div
            key={note.id}
            className={`tab ${activeNoteId === note.id ? 'active' : ''}`}
            onClick={() => handleTabClick(note.id)}
          >
            <span>{note.title || 'Untitled'}</span>
```

```
      {notes.length > 1 && (
       <button
        onClick={(e) => {
         e.stopPropagation();
         setNotes(notes.filter(n => n.id !== note.id));
         if (activeNoteId === note.id) {
          setActiveNoteId(notes[0]?.id || null);
         }
        }}
        className="tab-close"
       >
        <FiX className="w-4 h-4" />
       </button>
      )}
    </div>
  ))}
</div>
<div className="tab-actions">
  <button
    onClick={saveNote}
    className="tab-action-button"
    title="Save Note (Ctrl+S)"
  >
    <FiSave className="w-5 h-5" />
  </button>
  <button
    onClick={loadAvailableNotes}
    className="tab-action-button"
    title="Open Saved Notes"
  >
    <FiFolder className="w-5 h-5" />
  </button>
```

```
    <button
      onClick={addNewNote}
      className="tab-action-button"
      title="New Note"
    >
      <FiPlus className="w-5 h-5" />
    </button>
    <button
      onClick={createTestUser}
      className="tab-action-button"
      title="Create Test User"
    >
      <FiUser className="w-5 h-5" />
    </button>
  </div>
</div>

{showNoteSelector && (
  <div className="note-selector-overlay">
    <div className="note-selector">
      <div className="note-selector-header">
        <h2>Select a Note</h2>
        <button
          onClick={() => setShowNoteSelector(false)}
          className="note-selector-close"
        >
          <FiX className="w-5 h-5" />
        </button>
      </div>
      <div className="note-selector-list">
        {availableNotes.map(note => (
          <div
```

```
          key={note.id}
          className="note-selector-item"
          onClick={() => handleNoteSelect(note)}
        >
          <h3>{note.title || 'Untitled'}</h3>
          <p className="text-sm text-gray-500">
            Last updated: {new Date(note.updatedAt).toLocaleString()}
          </p>
        </div>
      ))}
    </div>
  </div>
 </div>
)}

{activeNoteId && (
 <div className="notes-editor">
  <div className="note-title-container">
    {isEditingTitle ? (
     <input
      type="text"
      value={newTitle}
      onChange={handleTitleChange}
      onBlur={handleTitleBlur}
      onKeyDown={handleTitleKeyDown}
      className="note-title-input"
      autoFocus
     />
    ) : (
     <h1 className="note-title" onClick={handleTitleClick}>
      {notes.find(note => note.id === activeNoteId)?.title || 'Untitled'}
     </h1>
```

```
  )}
</div>
<div className="formatting-toolbar">
  <button
    onClick={() => editor?.chain().focus().toggleBold().run()}
    className={`formatting-button ${editor?.isActive('bold') ? 'active' : ''}`}
    title="Bold"
  >
    <FiBold />
  </button>
  <button
    onClick={() => editor?.chain().focus().toggleItalic().run()}
    className={`formatting-button ${editor?.isActive('italic') ? 'active' : ''}`}
    title="Italic"
  >
    <FiItalic />
  </button>
  <button
    onClick={() => editor?.chain().focus().toggleCode().run()}
    className={`formatting-button ${editor?.isActive('code') ? 'active' : ''}`}
    title="Code"
  >
    <FiCode />
  </button>
  <button
    onClick={() => editor?.chain().focus().toggleBulletList().run()}
    className={`formatting-button ${editor?.isActive('bulletList') ? 'active' : ''}`}
    title="List"
  >
    <FiList />
  </button>
  <div className="formatting-divider" />
```

```
          <button
            onClick={() => editor?.chain().focus().toggleHeading({ level: 1 }).run()}
           className={`formatting-button ${editor?.isActive('heading', { level: 1 }) ? 'active' : ''}`}
            title="Heading 1"
          >
            <FiType className="w-4 h-4" />
            <span className="ml-1 text-xs">1</span>
          </button>
          <button
            onClick={() => editor?.chain().focus().toggleHeading({ level: 2 }).run()}
           className={`formatting-button ${editor?.isActive('heading', { level: 2 }) ? 'active' : ''}`}
            title="Heading 2"
          >
            <FiType className="w-4 h-4" />
            <span className="ml-1 text-xs">2</span>
          </button>
          <button
            onClick={() => editor?.chain().focus().toggleHeading({ level: 3 }).run()}
           className={`formatting-button ${editor?.isActive('heading', { level: 3 }) ? 'active' : ''}`}
            title="Heading 3"
          >
            <FiType className="w-4 h-4" />
            <span className="ml-1 text-xs">3</span>
          </button>
        </div>
        <div className="notes-editor-content">
          <EditorContent editor={editor} />
        </div>
      </div>
    )}
  </div>
);
```

```
};

export default Notes;
const API_BASE_URL = '/api/notes';

export async function getNotesByUser(userId: string) {
  try {
    const response = await fetch(`${API_BASE_URL}?userId=${userId}`);
    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Error fetching notes:', error);
    return { success: false, error: 'Failed to fetch notes' };
  }
}

export async function createNote(note: { title: string; content: string; userId: string }) {
  try {
    if (!note.title.trim()) {
      return {
        success: false,
        error: 'Note title is required. Please add a title before saving.'
      };
    }

    console.log('Sending note data:', note);
    const response = await fetch(API_BASE_URL, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(note),
```

```
    });
    const data = await response.json();
    if (!response.ok) {
      throw new Error(data.error || 'Failed to create note');
    }
    return data;
  } catch (error) {
    console.error('Error creating note:', error);
    return { success: false, error: 'Failed to create note' };
  }
}

export async function updateNote(id: string, updates: { title?: string; content?: string }) {
  try {
    const response = await fetch(API_BASE_URL, {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ id, ...updates }),
    });
    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Error updating note:', error);
    return { success: false, error: 'Failed to update note' };
  }
}

export async function deleteNote(id: string) {
  try {
    const response = await fetch(`${API_BASE_URL}?id=${id}`, {
```

```
      method: 'DELETE',
    });
    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Error deleting note:', error);
    return { success: false, error: 'Failed to delete note' };
  }
}
```

Додаток Г. Ілюстративна частина

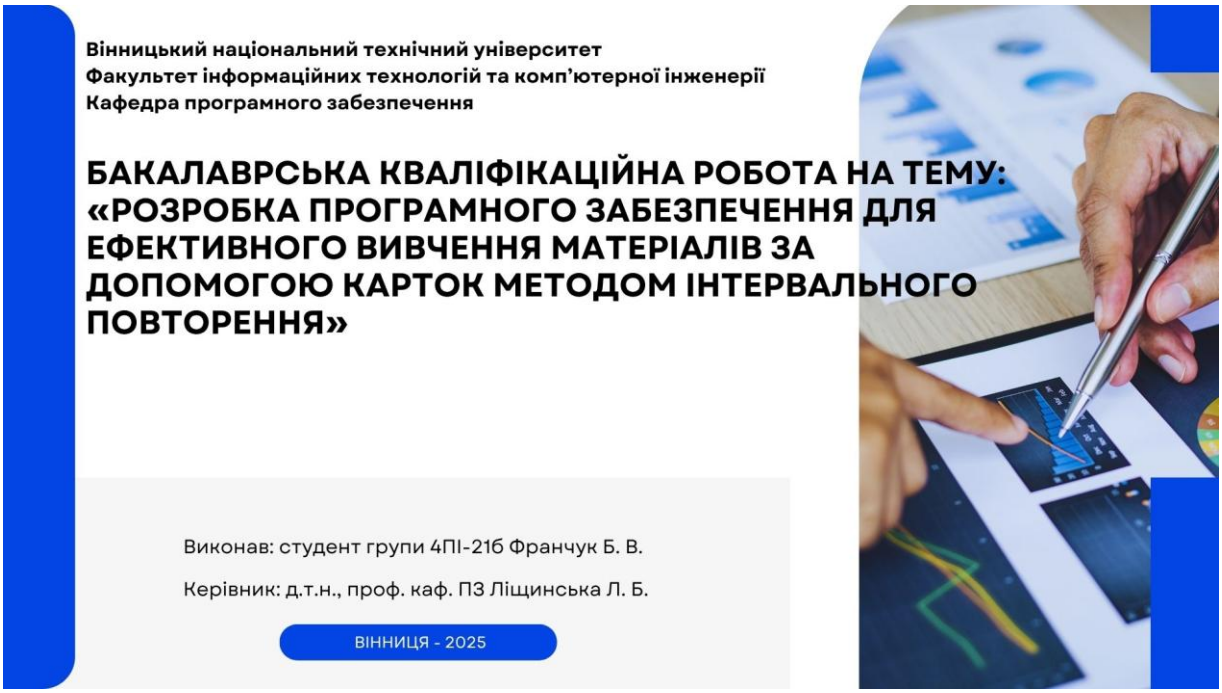

Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Мета та задачі дослідження

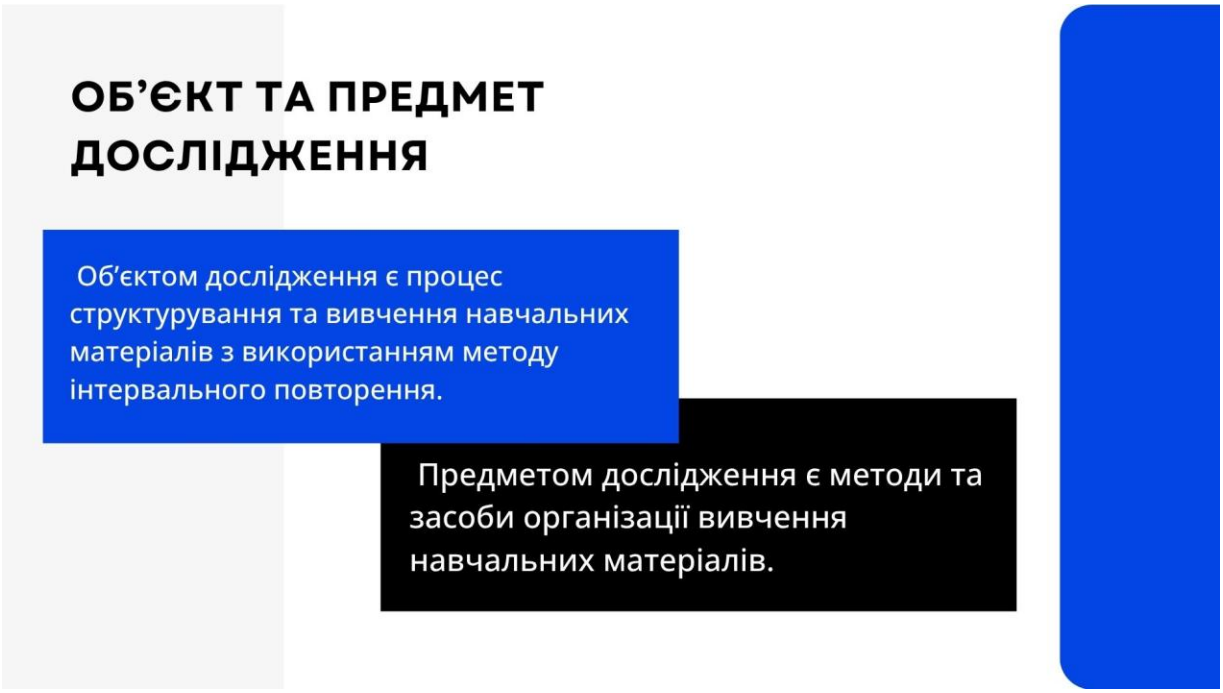

Рисунок Г.3 – Об’єкт та предмет дослідження

Рисунок Г.4 – Новизна отриманих результатів

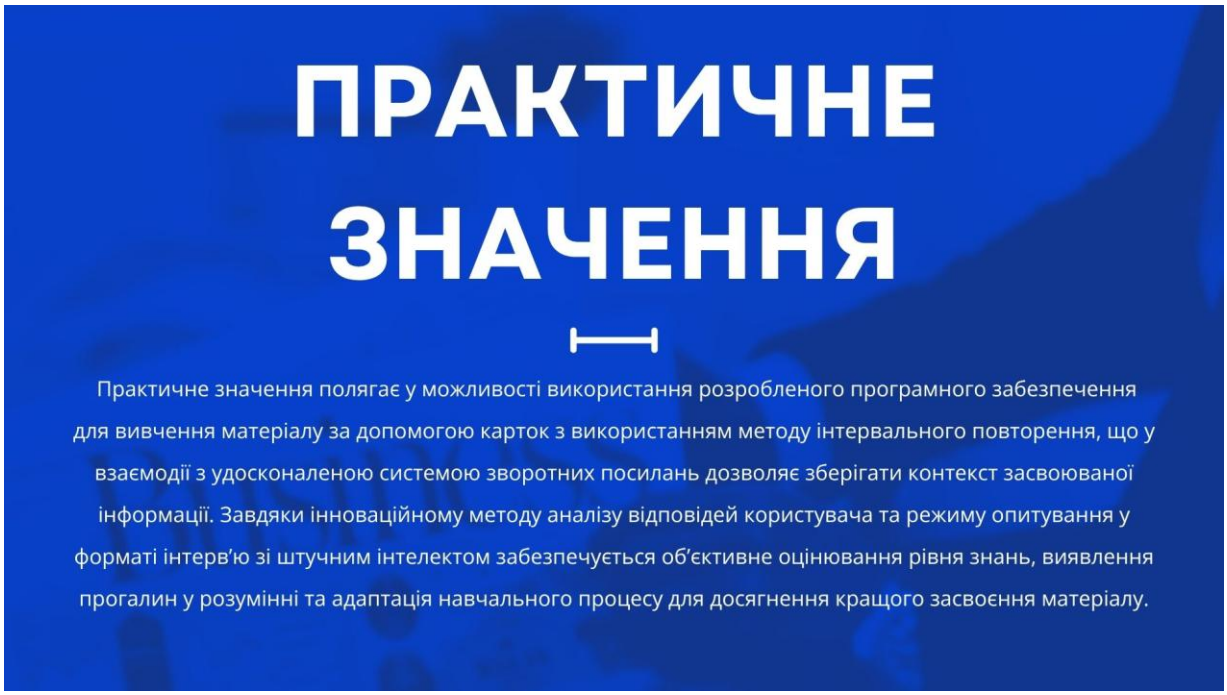

Рисунок Г.5 – Практичне значення

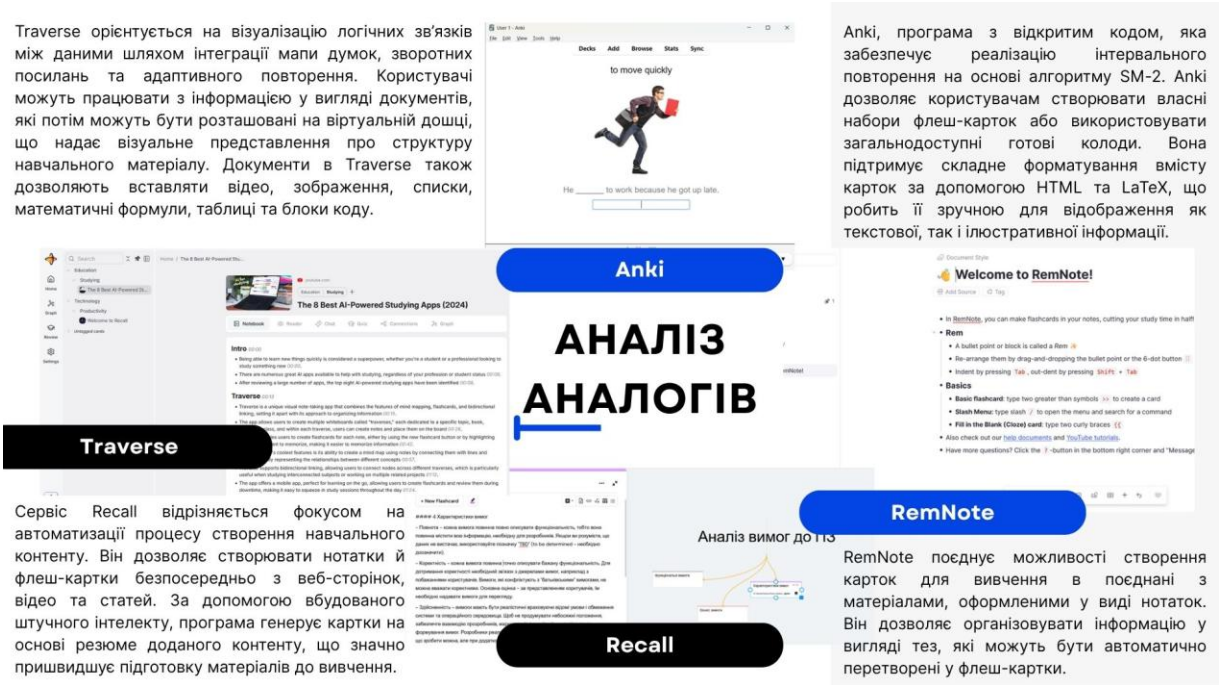

Рисунок Г.6 – Аналіз аналогів

ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА СИСТЕМ

Аналогічні системи зосереджені переважно на запам'ятовуванні окремих термінів чи визначень, вирваних із загального контексту. У них відсутній режим, який би сприяв формуванню цілісного розуміння теми в цілому. Цю проблему можна вирішити інтегрувавши систему опитування користувача по загальній темі у форматі інтерв'ю із штучним інтелектом, задачі якого знайти слабкі місця в розумінні теми.

Провівши аналіз аналогів було визначено, що хоча окремі існуючі програмні засоби успішно реалізують певні процеси продуктивного навчального процесу, жоден із них не охоплює повного спектра необхідних функцій у межах єдиної системи. Це створює необхідність для розробки нового інструмента, здатного об'єднати ключові переваги вивчених рішень у цілісну та логічно узгоджену систему.

Критерії	Anki	Traverse	Recall	RemNote	Власна розробка
Алгоритм інтервального повторення	✓	✓	✓	✓	✓
Документи для створення конспектів	✗	✓	✓	✓	✓
Мапа думок	✗	✓	✓	✗	✓
Асистент зі штучним інтелектом	✗	✓	✓	✓	✓
Зворотні посилання	✗	✓	✓	✓	✓
Граф	✗	✓	✓	✗	✓
Режим опитування у форматі інтерв'ю	✗	✗	✗	✗	✓
Сумарний коефіцієнт	1	6	6	4	7

Рисунок Г.7 – Порівняльна характеристика систем

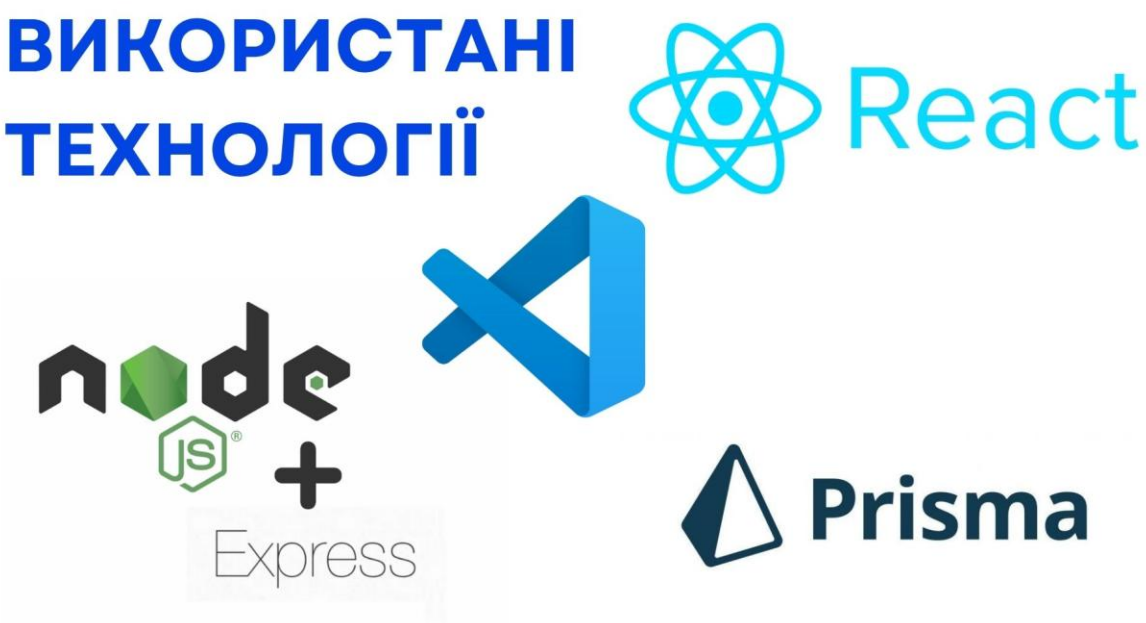

Рисунок Г.8 – Використані технології

Рисунок Г.9 – Алгоритм «Half-time regression»

Рисунок Г.10 – Алгоритм пошуку зворотних посилань

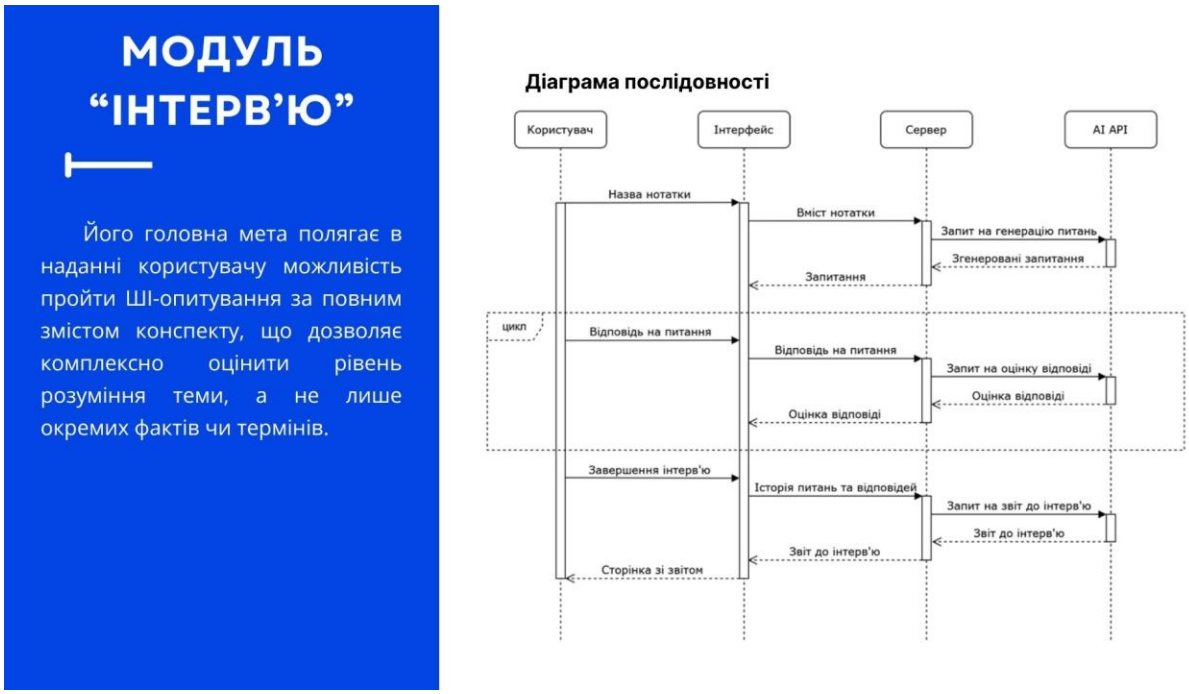

Рисунок Г.11 – Діаграма послідовності модуля «Інтерв’ю»

Рисунок Г.12 – Модель архітектури системи

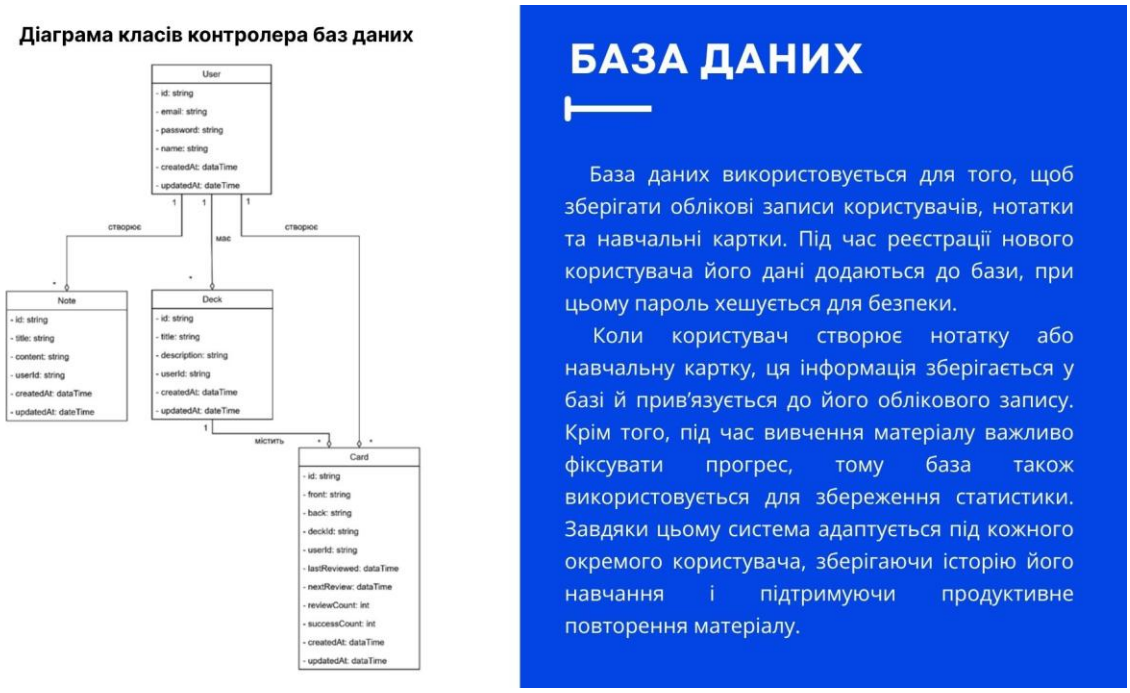

Рисунок Г.13 – Діаграма класів контролера баз даних

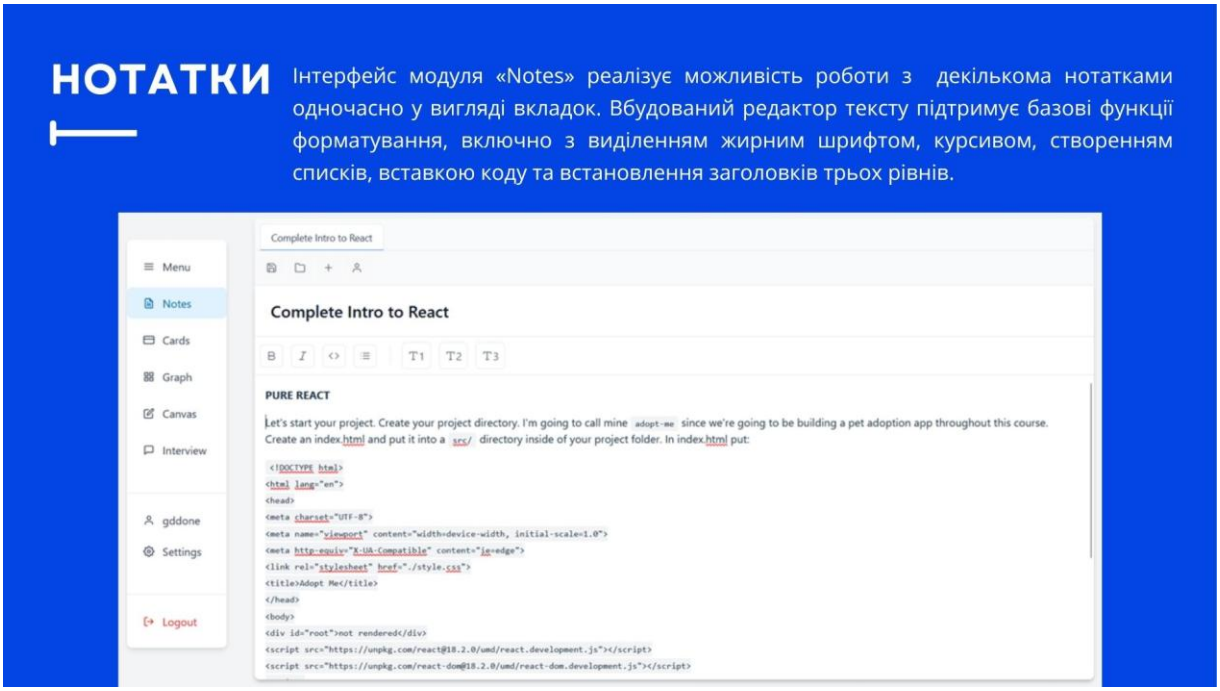

Рисунок Г.14 – Функціонал нотаток

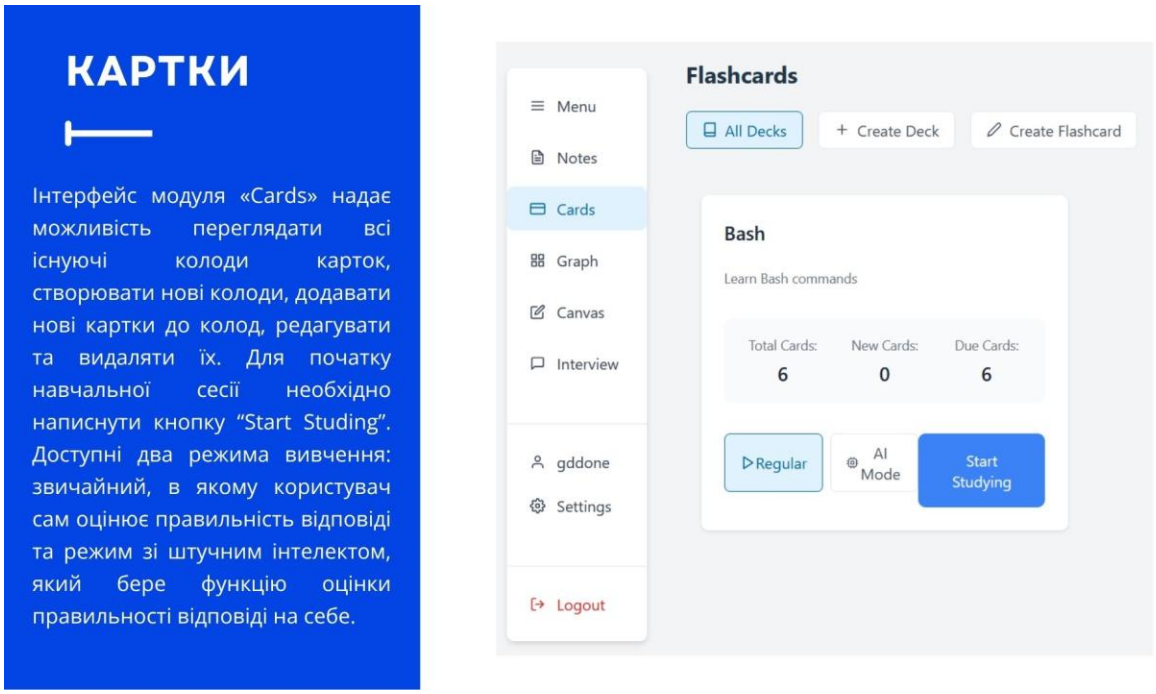

Рисунок Г.15 – Функціонал карток

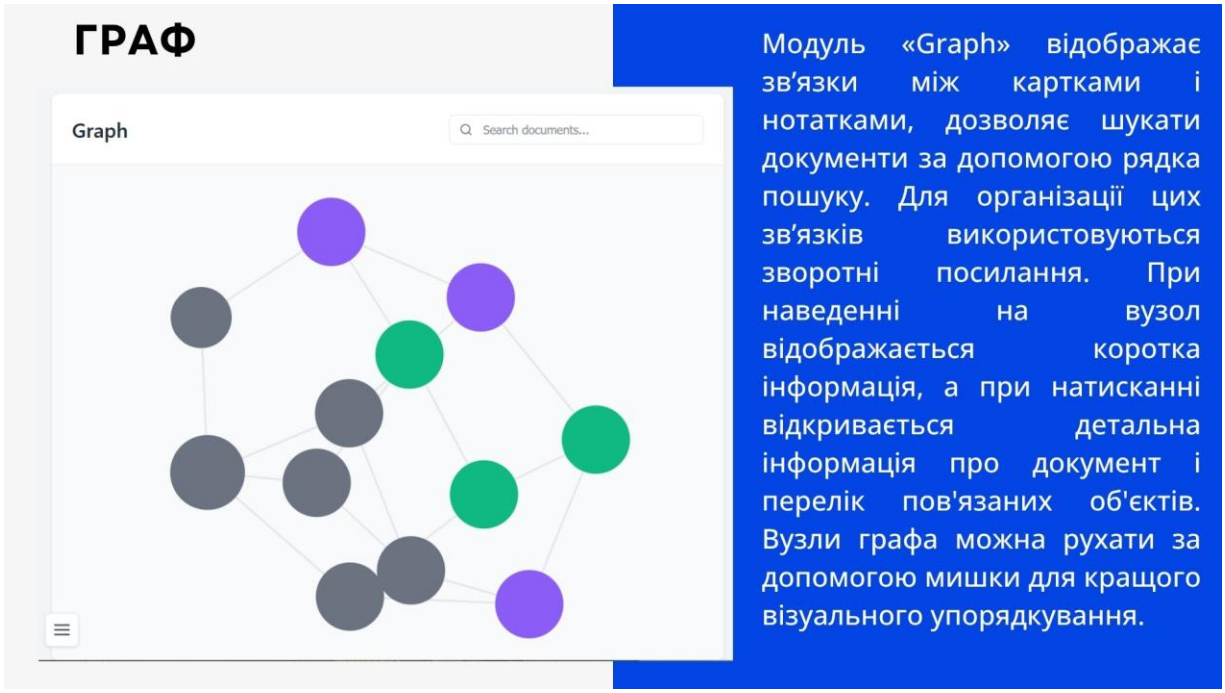

Рисунок Г.16 – Функціонал графу

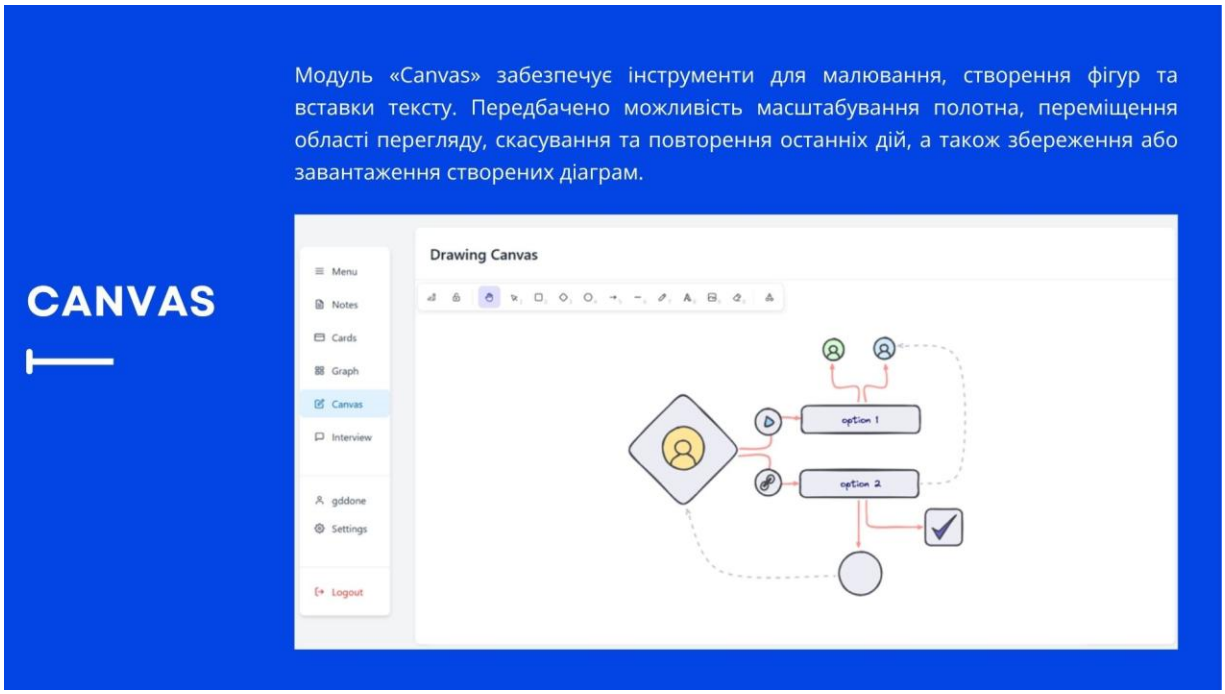

Рисунок Г.17 – Функціонал «Canvas»

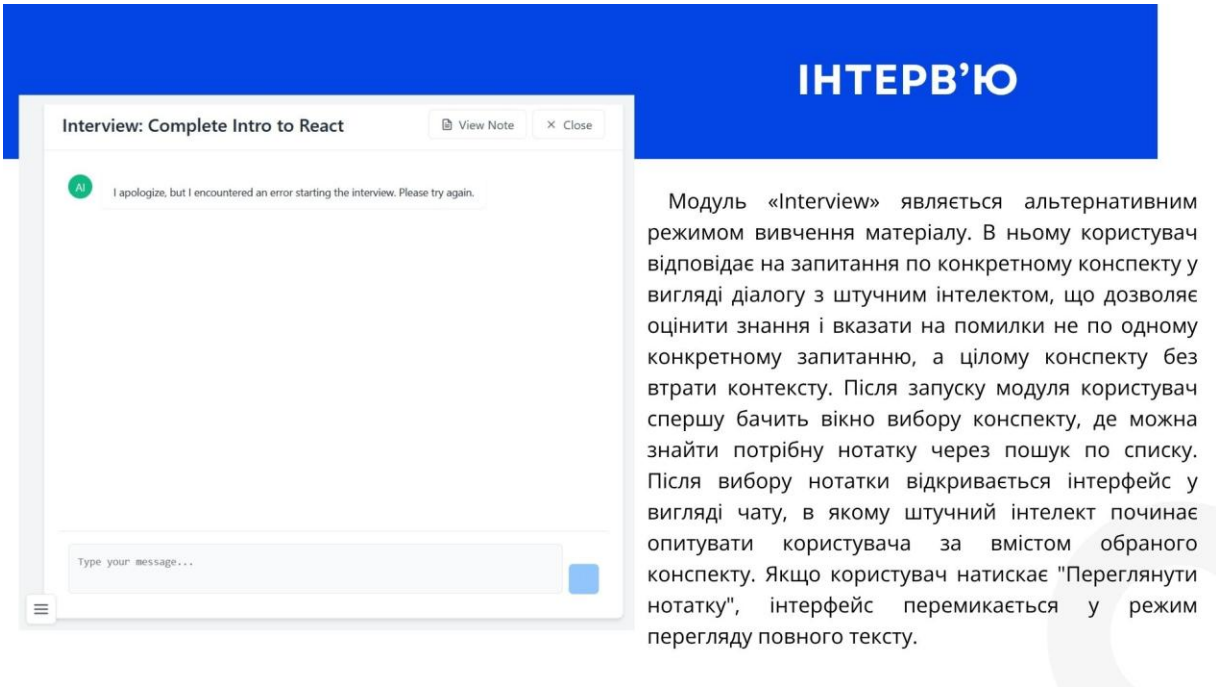

Рисунок Г.18 – Функціонал «Інтерв'ю»

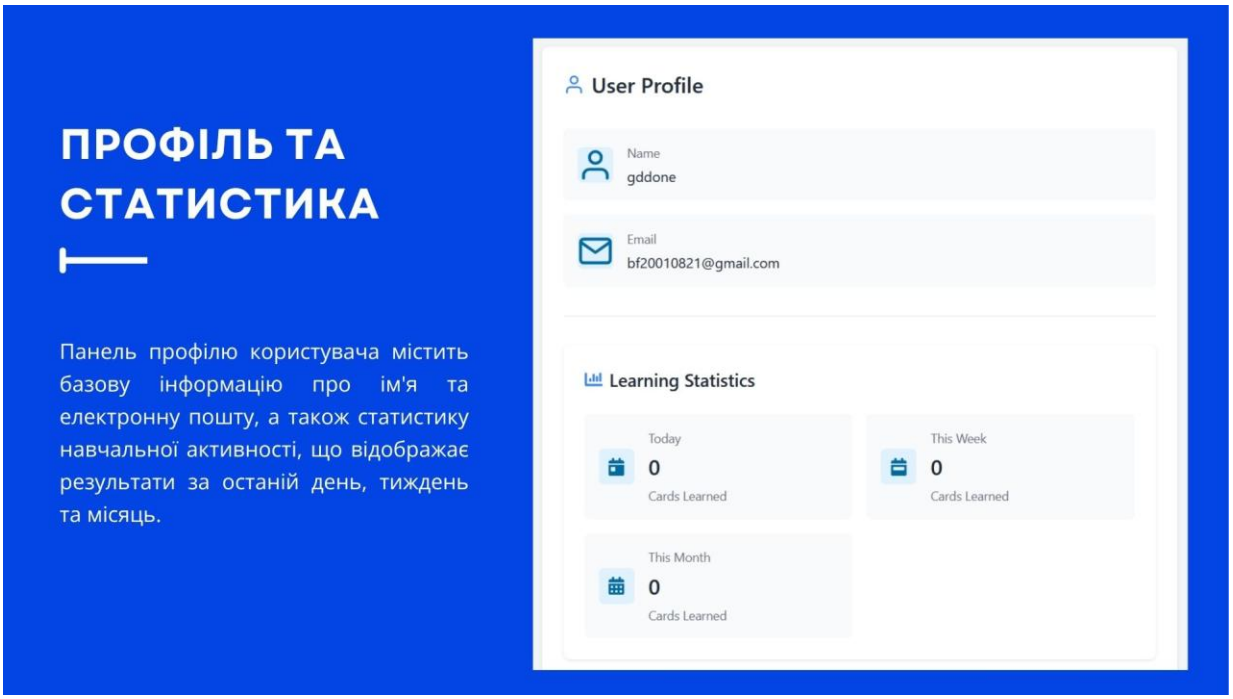

Рисунок Г.19 – Функціонал статистики

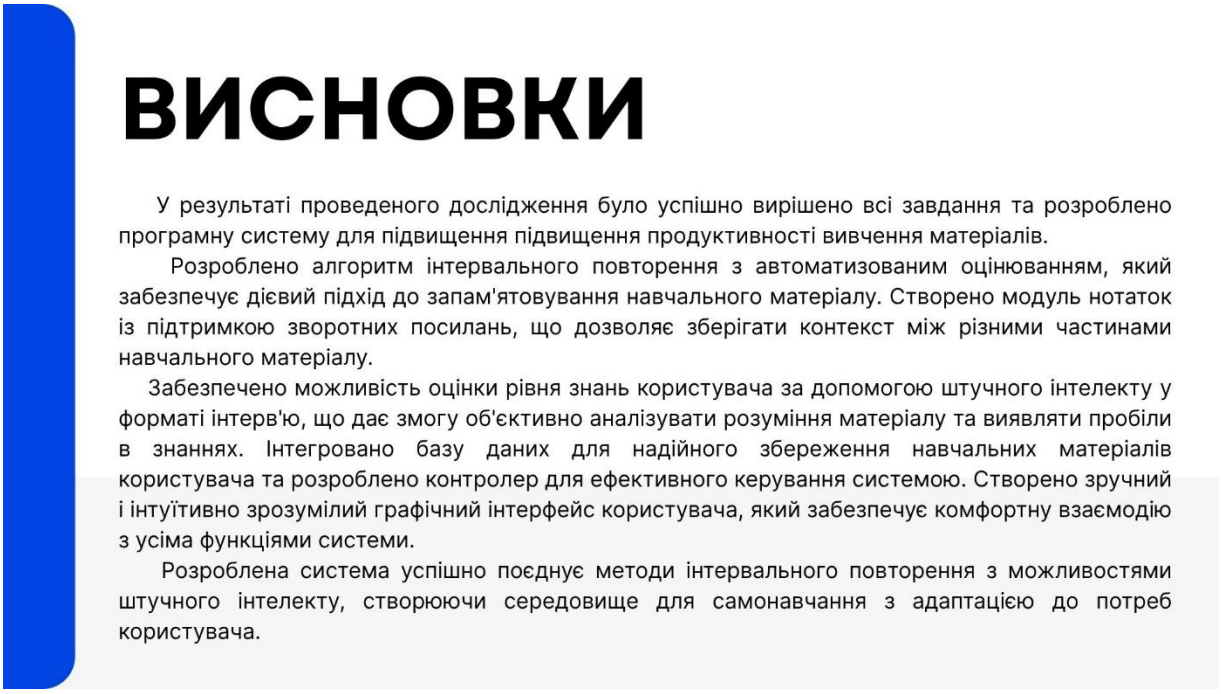

Рисунок Г.20 – Висновки