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

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
“___”_____2026 р.

Дипломна робота
на здобуття ступеня бакалавр

За освітньою програмою «Цифрові технології в енергетиці»

Спеціальності 122 «Комп’ютерні науки»

на тему: «Вебзастосунок для вивчення іноземних мов»

Виконала: студентка 4 курсу, групи ТР-21
Діброва Євгенія Василівна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н, доцент Лариса КУБЛІЙ _____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Рецензент доцент каф. ТАЕ, к.т.н., Олександр СІРИЙ _____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Н.контроль ст.викладач Ольга БЕСПАЛА _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ — 2026

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)
спеціальність 122 «Комп’ютерні науки»
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА
(підпис)
“___” _____2026 р.

ЗАВДАННЯ

на дипломну роботу студЕнту

Діброва Євгенія Василівна
(прізвище, ім’я, по батькові)

1. Тема роботи «Вебзастосунок для вивчення іноземних мов»
Науковий керівник Кублій Лариса Іванівна, к.т.н, доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “ 28 ” травня 2026 року № 1992-с
2. Термін подання студентом роботи 04.06.2026
3. Вихідні дані до роботи мова програмування PHP, фреймворк Laravel, СУБД MySQL
4. Перелік питань, які потрібно розробити _____
1) провести аналіз серед наявних аналогів
2) визначити основні функції вебзастосунку, який керує навчальним матеріалом і процесом його повторення
3) аргументувати вибрані інструменти, алгоритми та технології для реалізації вебсистеми
4) побудувати архітектуру програмного забезпечення і бази даних
5) створити прикладний програмний вебінтерфейс
6) подати процес застосування вебінтерфейсу

5. Орієнтований перелік ілюстративного матеріалу схема алгоритму опрацювання відповіді користувача, схема архітектури проєкту, ER-діаграма сутностей бази даних, діаграма прецедентів, скриншоти роботи користувачів з вебзастосунком.

6. Дата видачі завдання 15.09.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вибір теми роботи	01.06.2025-21.06.2025	виконано
2.	Аналіз методів та засобів розв’язання задачі	23.03.2026-05.04.2026	виконано
3.	Розробка архітектури та загальної структури системи	20-23.04.26	виконано
4.	Розробка окремих підсистем	24.04.-01.05.26	виконано
5.	Програмна реалізація системи	02-06.05.26	виконано
6.	Оформлення пояснювальної записки	07-11.05.26	виконано
7.	Захист програмного забезпечення	12.05.2026	виконано
8.	Передзахист	02.06.2026	виконано
9.	Захист	17.06.2026	виконано

Студент _____ Євгенія ДІБРОВА
(підпис) (прізвище та ініціали)

Керівник _____ Лариса КУБЛІЙ
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота виконана на 66 сторінках, містить 52 сторінок основного тексту, 25 ілюстрацій, 3 таблиці, 21 джерело у переліку посилань, 2 додатки.

Мета розробки — розробка вебзастосунку для систематичного розширення словникового запасу з можливістю додавання користувачем власного матеріалу.

Методи і засоби: мова програмування PHP, фреймворк Laravel, архітектурний шаблон MVC, мова програмування JavaScript, фреймворк Vue, система керування базами даних MySQL.

Результат — розроблено вебзастосунок для вивчення іноземних мов за допомогою алгоритму інтервального повторення, з персональними категоріями слів.

Ключові слова: інтервальні повторення, автоматизація навчання, PHP, Laravel, Vue.

ANNOTATION

The thesis is 66 pages long, with 52 pages of main text, contains 25 illustrations, 3 tables, 2 appendices, 21 sources in the list of references.

The purpose of this work is to develop a web application for systematically expanding vocabulary, with the ability to add user-generated content.

Methods and tools: PHP programming language, Laravel framework, MVC architectural pattern, JavaScript programming language, Vue framework, MySQL database management system.

Result: a web application developed for learning foreign languages using the spaced repetition algorithm, with personalized word categories.

Keywords: spaced repetition, learning automation, PHP, Laravel, Vue.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) — набір стандартизованих правил, за допомогою яких програмні компоненти обмінюються між собою даними.

REST (Representational State Transfer) — архітектурний стиль побудови вебслужб, що базується на використанні стандартних методів протоколу HTTP.

HTTP (HyperText Transfer Protocol) — протокол передачі даних між веббраузером і сервером.

JSON (JavaScript Object Notation) — текстовий формат для обміну структурованими даними.

URL (Uniform Resource Locator) — унікальна текстова адреса ресурсу в мережі Інтернет.

ORM (Object-Relational Mapping) — технологія роботи з таблицями бази даних як з об’єктами мови програмування.

СКБД — система керування базами даних.

SQL (Structured Query Language) — мова структурованих запитів до баз даних.

PHP (Hypertext Preprocessor) — скриптова мова програмування загального призначення, яка найчастіше застосовується для розробки вебзастосунків.

Laravel — відкритий PHP-фреймворк для розробки вебзастосунків, побудований на основі архітектурного шаблону MVC.

MVC (Model-View-Controller) — архітектурний шаблон, який розділяє логіку застосунку на три взаємопов’язані компоненти: дані (модель), інтерфейс (представлення) і керування (контролер).

Laravel Sanctum — офіційний пакет фреймворку Laravel для безпечної автентифікації користувачів через токени.

Токен доступу — цифровий ключ користувача у вигляді рядка символів, який підтверджує права користувача під час HTTP-запитів до захищеного API.

Eloquent — вбудована в Laravel ORM-система для простої роботи з базою даних засобами мови PHP.

MySQL — реляційна система керування базами даних з відкритим вихідним кодом.

Vue — сучасний фреймворк JavaScript для побудови інтерактивних користувацьких інтерфейсів.

Composition API — підхід до організації логіки компонентів у фреймворку Vue 3 на основі функцій.

Pinia — офіційна бібліотека керування глобальним станом для фреймворку Vue 3.

Vue Router — офіційна бібліотека маршрутизації для фреймворку Vue.

Axios — бібліотека JavaScript для виконання HTTP-запитів із клієнта на сервер.

SM-2 (SuperMemo 2) — математичний алгоритм для розрахунку оптимального часу повторення інформації.

Інтервальне повторення — метод навчання, який передбачає повторення матеріалу через часові проміжки, що постійно збільшуються.

Крива забування — графік, який відображає закономірність того, як людська пам’ять з часом забуває інформацію, якщо її не повторювати.

ВСТУП

У сучасному міжнародному ринку праці володіння іноземними мовами є основною вимогою до кваліфікованого працівника. Зокрема в індустрії інформаційних технологій знання англійської мови критично необхідне для використання технічної документації, комунікації всередині команд і взаємодії з замовниками з-за кордону.

Попри велику кількість навчальних ресурсів, головною перешкодою для багатьох фахівців залишається брак словникового запасу. Більшість сучасних платформ, спрямованих на розв’язання цієї проблеми, орієнтовані на пасивне споживання матеріалу без активної взаємодії з ним. Це призводить до зниження ефективності засвоєння й ускладнює подальше практичне застосування нової лексики.

Згідно з дослідженнями в галузі когнітивної психології, найкращим способом перевести інформацію з короткотривалої пам’яті в довготривалу вважається метод інтервальних повторень. Цей підхід базується на циклічному повторенні матеріалу через щоразу більші інтервали часу, оптимальний розмір яких розраховується залежно від відповіді користувача.

Зі збільшенням обсягів інформації, використання для цього методу паперових карток стає неефективним, оскільки складність керування матеріалами теж збільшується.

Метою роботи є розробка вебзастосунку, який автоматизує рутинні обчислення і надасть користувачеві можливість самостійно формувати навчальний матеріал.

Завданнями роботи для досягнення мети є:

— аналіз наявних програмних засобів для вивчення іншомовної лексики;

— дослідження особливостей методу інтервальних повторень;

— реалізація системи вивчення іноземних мов з використанням інтервальних повторень;

— тестування створеного програмного забезпечення і виправлення помилок.

Практичне значення роботи полягає в тому, що створений вебзастосунок може бути використаний як повноцінний інструмент для вивчення іноземної лексики користувачами різного рівня підготовки. Система надасть користувачам можливість автоматизувати процес і спростить організацію навчального матеріалу. Крім цього, застосунок також може бути розвинутий у більш масштабну навчальну платформу.

Результати роботи апробовано на XXIII Міжнародній науково-практичній конференції молодих вчених та студентів “Сучасні проблеми наукового забезпечення енергетики: безпека, стійкість, ІТ та екологічний моніторинг” (до 40-річчя Чорнобильської катастрофи), м. Київ, 17-22 квітня 2026 року [1] (додаток А).

Дипломна записка складається з чотирьох розділів. У першому розділі описано задачу розробки вебсистеми для вивчення іншомовної лексики, проаналізовано потенційних користувачів, визначено функціональні вимоги, розглянуто метод інтервальних повторень і проведено аналіз наявних програмних аналогів. У другому розділі обґрунтовано вибір технологій і засобів розробки серверної і клієнтської частин застосунку. Третій розділ присвячений програмній реалізації вебзастосунку, описові архітектури системи і структури бази даних. У четвертому розділі наведено опис взаємодії користувача із системою.

Повний обсяг дипломної записки — 66 сторінок. Основна частина складається із 52 сторінок, містить 25 рисунків, 3 таблиці, 2 додатки і 21 джерело в переліку посилань.

1 РОЗРОБКА ВЕБСИСТЕМИ ДЛЯ ВИВЧЕННЯ ІНШОМОВНОЇ ЛЕКСИКИ

Для реалізації гнучкого масштабованого рішення необхідно детально дослідити потенційних користувачів і сформувати чіткий перелік вимог до системи. Ще одним важливим етапом є аналіз сучасних підходів до цифрової освіти, оскільки все більше платформ у цій ніші фокусуються не просто на реалізації функціональних вимог, вони створюють зручний інструмент для повсякденного використання і підтримки мотивації до навчання. Тому під час розробки системи важливо приділяти увагу не лише технічним аспектам. Такий підхід закладе надійний фундамент для проєкту, узгодить очікування користувачів із технічними можливостями і допоможе уникнути критичних помилок на стадіях проєктування й написання коду.

1.1 Аналіз потенційних користувачів системи

За останні роки у зв’язку з багатьма факторами популярність електронного навчання стрімко зросла і цей формат став одним із основних способів здобуття нових знань [2]. Мільйони людей комбінують старі методи з новими, проте чимало з них вирішує повністю перейти на навчання за допомогою цифрових платформ, відмовляючись від використання паперових підручників.

Головна причина таких змін полягає у високому рівні адаптивності електронної освіти до потреб користувача, оскільки це процес, який повністю базується на взаємодії з навчальним контентом через спеціальні мобільні застосунки, вебсайти або комплексні онлайн-платформи за допомогою смартфонів, планшетів чи комп’ютерів [3]. Ще однією перевагою є заощадження часу, оскільки на відміну від класичної системи, за якої навчання має відбуватися за єдиним графіком, цифровий формат надає можливість самостійно керувати власним

розкладом і темпом. Таким чином, лекції, які потрібно відвідати, більше не є фіксованими в часі і здобувачеві знань не потрібно підлаштовуватися під інших студентів чи витрачати час, щоб потрапити на заняття.

Такий формат використовується для розв'язання широкого спектру проблем: від вивчення іноземних мов, програмування і до підвищення кваліфікації на роботі. З кожним роком кількість цифрових навчальних програм стрімко зростає, завдяки чому кожен користувач, незалежно від рівня його знань, здатен підібрати матеріал, розрахований саме під його індивідуальні потреби.

Розроблюваний вебзастосунок призначений для максимально широкого кола осіб, які вивчають іноземні мови і прагнуть ефективно поповнювати свій лексичний запас.

Потенційними користувачами можуть бути не тільки студенти лінгвістичних напрямків. Крім них, можна виділити студентів технічних і гуманітарних спеціальностей, яким потрібно засвоювати велику кількість академічних чи технічних термінів іноземною мовою. Іншу важливу групу становлять IT-спеціалісти, які часто вивчають вузькоспеціалізовану лексику для співбесід і комунікацій з клієнтами. Застосунок також може стати корисний особам, які готуються до переїзду або до подорожі в іншу країну, оскільки їм потрібен зручний інструмент для швидкого збереження побутових і ситуативних виразів для подальшого вивчення. Завдяки своїй універсальності вебзастосунок здатен зацікавити людей будь-якого віку, які шукають безкоштовний інструмент без нав'язливої реклами.

Окрему увагу варто приділити користувачам, які вивчають іноземні мови самостійно без участі викладачів. Оскільки велика кількість людей припиняє цей процес через втрату мотивації або надмірну складність навчальних платформ, під час проєктування системи вкрай важливо враховувати їхні психологічні особливості. Інтерфейс застосунку має бути лаконічним, а сам процес навчання побудовано таким чином, щоб користувач міг чітко бачити власний прогрес. Наявність гейміфікації у вигляді статистики і системи відстеження результатів

позитивно впливає на мотивацію і сприяє регулярному поверненню до навчання [4].

Разом із цим, аналіз потенційної аудиторії демонструє необхідність підтримки різних форматів і сценаріїв використання системи. Зокрема, одним із ключових факторів є швидкість доступу до навчального матеріалу, що зумовлене тим, що багато користувачів повторюють слова в короткі проміжки часу, наприклад, під час поїздок або перерв. Саме тому система повинна забезпечувати швидке завантаження сторінок і мінімальну кількість дій для запуску тренувань. Такий підхід надасть користувачеві можливість обирати, як і коли навчатися упродовж дня, не прив’язуючи цей процес виключно до персонального комп’ютера чи мобільного телефону.

1.2 Розробка функціональних вимог

Створюваний застосунок не має прив’язки до конкретної мови, завдяки чому користувач отримує максимальну свободу дій і змогу персоналізувати навчальний матеріал під власні потреби й поточний рівень знань. Крім того, для розв’язання проблеми неефективного використання часу, система має відійти від хаотичного перегляду матеріалу. Натомість рішення орієнтоване на механізм адаптивного підбору слів, який базується на рівні їхнього засвоєння користувачем.

Для забезпечення життєздатності проєкту система включає обов’язкові модулі автентифікації й авторизації. Вони відповідають за реєстрацію нових користувачів і вхід із обов’язковою валідацією введених даних. Зокрема, під час реєстрації перевіряється унікальність електронної адреси, а сам процес авторизації супроводжується виданням користувачеві токена доступу, який автоматично додається до всіх його наступних HTTP-запитів. Для захисту паролів користувачів вони зберігаються тільки в захешованому вигляді. Також передбачено можливість повного видалення профілю користувача разом з усіма його персональними даними.

Система також забезпечує розмежування прав доступу. Користувач має доступ лише до власних категорій, словникових записів і статистики. Будь-які спроби отримання або модифікації чужих даних блокуються серверною частиною застосунку.

Робота з навчальним контентом базується на чіткій організації слів. Окремі картки слів не можуть існувати самостійно і створюються виключно в межах визначених категорій. У результаті користувач має швидку навігацію по матеріалу і динамічний підрахунок кількості слів всередині кожної категорії та за статусом.

Система надає користувачеві повний набір інструментів для створення, модифікації та видалення персональних категорій, а також окремих слів. Видалення категорії є каскадним, тобто автоматично вилучає всі слова всередині неї. Картка слова, крім обов’язкового значення самого слова, перекладу і категорії, може містити додаткову необов’язкову інформацію, наприклад, транскрипцію чи контекстуальні приклади. Для навігації по словах реалізовано фільтрацію за статусом, динамічний пошук слів і їхнє сортування за будь-якою колонкою таблиці.

Найважливішим компонентом архітектури є модуль повторень, який реалізує алгоритм інтервального навчання. Система автоматично формує вибірку слів на кожну навчальну сесію, враховуючи час наступного показу карток слів. Процес тренування організовано таким чином, що після введення користувачем відповіді вона одразу порівнюється з правильною і відображається результат. Далі, залежно від результату, на серверному боці застосунку автоматично перераховується наступна дата появи слова і змінюється рівень його засвоєння. Складні слова з’являються частіше, а добре засвоєні навпаки відкладаються на довші терміни.

Для наочного відстеження прогресу навчання система містить модуль статистики, що забезпечує візуалізацію прогресу вивчення у вигляді графіка і значення кількості слів за їхніми поточними статусами.

1.3 Розробка вимог до інтерфейсу користувача

Інтерфейс відіграє одну з найважливіших ролей у вебзастосунку, оскільки саме за допомогою нього відбувається взаємодія користувача із системою. Головною вимогою до інтерфейсу є інтуїтивність. Користувач повинен легко і швидко розуміти основний функціонал системи [5] без необхідності шукати додаткову інформацію на сторонніх ресурсах і мати можливість за мінімальну кількість дій перейти до створення категорії, додавання слова або запуску навчальної сесії.

Дизайн застосунку потрібно виконувати в мінімалістичному стилі. Інтерфейс не має бути перевантаженим зайвими декоративними елементами, щоб увага користувача концентрувалася виключно на навчальному матеріалі. За допомогою такого підходу зменшується когнітивне навантаження на користувача і взаємодія із системою стає комфортнішою [6].

Для забезпечення цілісності дизайну використовується єдина кольорова палітра, однакові стилі кнопок, таблиць, форм введення і модальних вікон. Повторюваність елементів інтерфейсу формує передбачувану модель взаємодії із системою і допомагає користувачеві швидко адаптуватися до роботи із застосунком.

Навігація між сторінками повинна бути реалізована так, щоб користувач мав постійний доступ до основних розділів вебзастосунку. Для цього використовується фіксована бокова панель. Активний пункт меню візуально виділяється, що допомагає користувачеві легко орієнтуватися в поточному розділі застосунку.

Особливу увагу потрібно приділити формам введення даних. Усі поля повинні мати зрозумілі назви, а інтерфейс має забезпечувати швидкий зворотний зв’язок під час взаємодії із системою. У випадку введення некоректних даних повинно виводитися відповідне повідомлення про помилку з поясненням причини. Також, після додавання, редагування або видалення якогось із об’єктів зміни

повинні миттєво відображатися на сторінці без необхідності її повторного завантаження.

Окремо варто враховувати питання доступності інтерфейсу. Використання контрастних кольорів, зрозумілих елементів навігації і читабельних шрифтів зробить застосунок більш зручним [7].

1.4 Аналіз методу інтервальних повторень

У процесі засвоєння іноземної мови ключовою проблемою є швидке забування лексичного матеріалу через відсутність постійної практики. Традиційні методи вивчення інформації демонструють низьку ефективність у довгостроковій перспективі. Це зумовлено механізмами мозку, що відповідають за очищення з пам’яті інформації, яка не використовується регулярно або не вважається пріоритетною.

Однією з основних проблем традиційних підходів вивчення слів є нерівномірний розподіл повторень. Користувачі часто намагаються вивчити велику кількість матеріалу за один раз, проте без систематичного закріплення ця інформація дуже швидко забувається. З цієї причини значна частина часу, витраченого на навчання, використовується неефективно. Розв’язання цієї проблеми полягає не в збільшенні інтенсивності навантаження, а в оптимізації навчання, використовуючи перевірені когнітивні методики і автоматизуючи їх за допомогою програмного забезпечення. Саме тому сучасні цифрові освітні системи дедалі більше використовують адаптивні механізми навчання, здатні автоматично підлаштовуватися під рівень знань конкретного користувача.

В основі розроблюваного вебзастосунку лежить метод інтервальних повторень. Він ґрунтується на двох психологічних феноменах: кривій забування Еббінгауза та ефекті інтервального навчання. Герман Еббінгауз довів, що значна частина нової інформації зникає з пам’яті вже впродовж перших годин після

засвоєння. Проте, якщо повторити матеріал перш ніж це станеться, швидкість забування сповільниться, а сама інформація перейде в довготривалу пам’ять [8].

Завдяки повторенню інформації через певні часові проміжки мозок отримує сигнал про те, що вона важлива і краще закріплює її. Такий підхід активно використовується в сучасних освітніх платформах, мобільних застосунках і професійних системах підготовки, де необхідно працювати з великими обсягами інформації.

Практична реалізація методу передбачає систему рівнів. Щойно додане слово перебуває на першому рівні з мінімальним інтервалом повторення. Після кожної правильної відповіді під час тренування воно переходить на наступний рівень з більшим інтервалом. При неправильній відповіді слово втрачає свій прогрес і автоматично повертається на перший рівень. У результаті користувач витрачає більше часу саме на проблемний матеріал, а не на вже добре засвоєні слова.

Крім підвищення ефективності запам’ятовування, використання інтервальних повторень позитивно впливає і на мотивацію користувача. Завдяки поступовому переходу слів між рівнями користувач може бачити власний прогрес, що стимулює продовження навчання.

Метод інтервальних повторень ідеально поєднується з концепцією мікронавчання, яка передбачає засвоєння матеріалу невеликими частинами в короткі проміжки часу і теж демонструє покращення рівня засвоєння інформації [9]. Такий підхід є особливо актуальним для сучасного ритму життя, оскільки надає можливість ефективно навчатися в перервах між роботою або іншими повсякденними справами. Поєднання цих двох методів допомагає створити гнучкий і комфортний процес засвоєння матеріалу: користувач проходить короткі тренувальні сесії у вільний для нього час, повторюючи саме ті слова, які система визначила як пріоритетні на основі алгоритму.

Таким чином, у результаті вдається отримати ефективний механізм вивчення іншомовної лексики, який враховує особливості людської пам’яті, а також індивідуальний темп навчання користувача.

1.5 Аналіз програмних аналогів

На сьогодні існує велика кількість цифрових платформ і застосунків для вивчення іноземних мов. Вони відрізняються за функціональними можливостями, підходами до навчання і складністю інтерфейсу. Аналіз наявних рішень допомагає визначити їхні переваги, недоліки і виявити особливості, які доцільно врахувати під час розробки власної системи.

Одним із найвідоміших інструментів для вивчення лексики є програма Anki [10]. Цей застосунок базується на алгоритмі інтервальних повторень SM-2 і надає користувачам можливість створювати власні набори карток. До основних переваг застосунку належать висока гнучкість налаштувань, відкритий програмний код і підтримка різних типів контенту. Користувачі можуть додавати текст, зображення і навіть аудіо. Проте значна кількість налаштувань, а також складний застарілий інтерфейс, можуть створювати труднощі у використанні, особливо на початковому етапі роботи із системою.

Іншим популярним рішенням є платформа Quizlet [11], яка орієнтується на використання флеш-карток. Система має сучасний зручний інтерфейс, підтримує спільні набори слів і містить інтерактивні режими навчання. Водночас значна частина функціоналу, зокрема повноцінне інтервальне повторення, доступна тільки в платній версії застосунку, що створює труднощі у використанні цієї платформи для широкого кола користувачів.

Застосунок Memrise [12] також належить до популярних платформ для вивчення мов. Його особливістю є орієнтація на використання готових курсів і навчання популярних фраз у контексті реальних ситуацій. Платформа активно використовує елементи гейміфікації і мультимедійний контент. Проте можливості роботи з власними матеріалами є обмеженою, а частина функцій доступна тільки за підпискою.

Також варто вказати сервіс Duolingo [13], який є одним із найпоширеніших у світі застосунків для вивчення мов. Основний акцент у системі зроблено на

гейміфікації навчального процесу. Користувач отримує бали і досягнення за регулярне проходження вправ. Такий підхід позитивно впливає на мотивацію, проте система не орієнтована на роботу з власним матеріалом користувача, а персоналізація контенту відсутня.

Проведений аналіз показує, що більшість сучасних платформ або мають складний інтерфейс і високий поріг входження, або обмежують користувача у роботі з власним навчальним матеріалом. Крім того, частина функціоналу часто потребує оплати. Таким чином, існує актуальна потреба у створенні простого, безкоштовного і зручного вебзастосунку, який поєднуватиме можливість роботи з довільним контентом, підтримку інтервальних повторень і сучасний мінімалістичний інтерфейс без перевантаження зайвими функціями.

2 ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ І ЗАСОБІВ РОЗРОБКИ

Для реалізації запланованого функціоналу і створення клієнт-серверної архітектури був обраний сучасний набір технологій. Таким чином, проєкт поділяється на дві незалежні частини: серверне API, яке виконує всі складні обчислення і роботу з базою даних, а також клієнтський інтерфейс.

Використання популярних технологій полегшує подальшу модернізацію системи, а також інтеграцію нових модулів. Окремою перевагою є наявність великої кількості документації і активних спільнот розробників, що значно полегшує процес розробки і підтримки проєкту в майбутньому.

2.1 Мова програмування PHP

Для розробки серверної частини вебзастосунку було вирішено застосувати мову програмування PHP [14]. Основним фактором вибору цієї мови стала її популярність і висока швидкість розробки.

Крім цього, однією з важливих переваг мови PHP є можливість легкої реалізації REST API завдяки наявності великої кількості готових бібліотек і фреймворків. Це суттєво спрощує створення серверної логіки та інтеграцію з базою даних.

Ця мова також є оптимальним вибором для створення вебзастосунків із великою кількістю HTTP-запитів і взаємодією з реляційними базами даних. Для системи вивчення іноземної лексики це є особливо важливим фактором, оскільки застосунок постійно працює із запитами на створення, редагування чи отримання слів, категорій і статистики навчання. Ще однією важливою перевагою є підтримка сучасних механізмів типізації та оптимізації продуктивності в нових версіях мови програмування PHP.

2.2 Фреймворк Laravel

Фреймворк Laravel [15] було обрано завдяки його чіткій структурі, а також великій кількості готових рішень. У проєкті серверна логіка побудована так, що кожен елемент виконує тільки відведені йому задачі. Маршрути відповідають за обробку URL і HTTP-запитів, контролери координують роботу системи, сервіси містять бізнес-логіку, а моделі забезпечують взаємодію з базою даних.

Фреймворк також спрощує роботу щодо авторизації, валідації даних і формування JSON-відповідей, завдяки чому значно скорочується час розробки. Крім цього, вбудована система міграцій виконує роль контролю версій для бази даних, що надає можливість швидко розгортати структуру таблиць на будь-якому комп’ютері.

Фреймворк Laravel також надає потужну ORM Eloquent для зручної роботи з базою даних за допомогою мови PHP без написання складних SQL-запитів вручну. Це робить код читабельним і полегшує його підтримку в майбутньому.

2.3 Пакет Laravel Sanctum

Для організації безпечної автентифікації було використано Laravel Sanctum [16]. Використання цього пакету зумовлене тим, що клієнтська і серверна частини працюють окремо і взаємодіють виключно через API, який забезпечує гнучку архітектуру і спрощує можливість підключення мобільного застосунку до вже створеного серверного API в майбутньому.

Цей пакет забезпечує механізм авторизації на основі токенів. Після успішного входу система створює користувачеві токен доступу, який надалі використовується для підтвердження його особи під час кожного запиту до API.

Такий підхід допомагає уникнути повторної авторизації при кожному оновленні сторінки і забезпечує безпечну роботу із захищеними ресурсами системи.

2.4 Система керування базами даних MySQL

Для збереження інформації про користувачів, колекції, слова та історію повторень була обрана реляційна система керування базами даних MySQL [17]. За її допомогою у вебзастосунку забезпечується швидка обробка запитів, підтримка зовнішніх ключів і можливість коректно встановлювати зв’язки між таблицями. Це особливо важливо для роботи з категоріями слів та історією повторень, де між сутностями існують складні взаємозв’язки.

Зокрема, зв’язок «один-до-багатьох» між категоріями і словами реалізовано з використанням каскадного видалення на рівні системи керування базами даних. Це гарантує цілісність даних і автоматичне очищення бази від застарілих або видалених користувачем матеріалів. Крім того, індексація полів із ідентифікаторами користувачів і статусами слів дає можливість виконувати миттєву вибірку набору слів для тренувальних сесій, що важливо для високої продуктивності вебзастосунку.

2.5 Фреймворк Vue

Візуальна частина застосунку побудована за допомогою фреймворку Vue [18] з використанням Composition API. Завдяки цьому з’являється можливість створювати незалежні елементи інтерфейсу, наприклад, модальні вікна, картки слів чи таблиці, і повторно використовувати їх у різних частинах застосунку, що значно пришвидшує процес розробки і зменшує дублювання коду. Логіка всередині них організована у вигляді окремих функцій, що робить код чистим і легким для подальшого масштабування або тестування.

Фреймворк Vue також забезпечує швидку реакцію інтерфейсу на дії користувача, завдяки чому всі зміни в даних миттєво відображаються на екрані без необхідності постійного оновлення сторінки. Це особливо важливо для застосунку, де користувач часто взаємодіє з таблицями слів і статистикою.

2.6 Бібліотека Pinia

Для керування глобальним станом клієнтської частини застосунку використовується бібліотека Pinia [19]. Вона дає можливість зберігати дані й логіку спільно між різними компонентами застосунку, грає роль глобального сховища даних, що значно спрощує взаємодію між сторінками і компонентами. Таким чином, компоненти миттєво дізнаються про зміни в даних через глобальні сховища поточного користувача, список слів чи категорій, забезпечуючи оновлення інтерфейсу в реальному часі.

2.7 Бібліотека Vue Router

Навігацію сторінок у вебзастосунку забезпечує бібліотека Vue Router [20]. Вона організована так, щоб користувач мав можливість переходити між основними сторінками системи: загальним списком слів, персональними категоріями, особистим профілем.

За допомогою цієї бібліотеки реалізується така структура навігації, що переходи між сторінками виконуються миттєво і без повного оновлення браузера. Це забезпечує для користувача плавну і комфортну роботу із системою. Окремо варто зазначити, що використання маршрутизації спрощує подальше масштабування системи, оскільки нові сторінки можна інтегрувати в уже наявну структуру без суттєвих змін архітектури.

Додатково на рівні маршрутизатора реалізовано механізм перевірки прав доступу й авторизації безпосередньо під час переходу між сторінками. Якщо неавторизований користувач намагається отримати доступ до захищеного компонента або якщо термін дії його поточної сесії завершився, система блокує цей запит і автоматично перенаправляє його на сторінку входу для проходження автентифікації.

2.8 Бібліотека Axios

Усі HTTP-запити до серверного API здійснюються за допомогою бібліотеки Axios [21]. Бібліотека Axios автоматично додає токен авторизації до кожного запиту користувача, спрямованого на сервер. Таке рішення повністю автоматизує процес верифікації сесії на стороні клієнта, усуває потребу вручну прописувати заголовки в кожному окремому методі, гарантує надійний захист під час обміну даними.

Крім цього, бібліотека Axios забезпечує централізовану обробку помилок сервера і мережевих збоїв. Наприклад, якщо сервер повертає помилку авторизації або тимчасово недоступний, застосунок може автоматично повідомити користувача про проблему або виконати перенаправлення на сторінку входу. Такий підхід підвищує стабільність роботи клієнтської частини і спрощує підтримку мережевої логіки у великих проєктах.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

Застосунок побудований з використанням сучасного підходу, що включає розділення системи на клієнтську і серверну частини. Така архітектура допомагає забезпечити гнучкість системи й спрощує її подальше масштабування. Обмін даними між частинами застосунку здійснюється за допомогою HTTP-запитів до REST API, що реалізовано з чітким розподілом на шари.

3.1 Архітектура системи

Обробка кожного клієнтського запиту на сервері відбувається через чітку послідовність взаємопов'язаних шарів, де кожен рівень виконує визначену для нього функцію (рисунок 3.1).

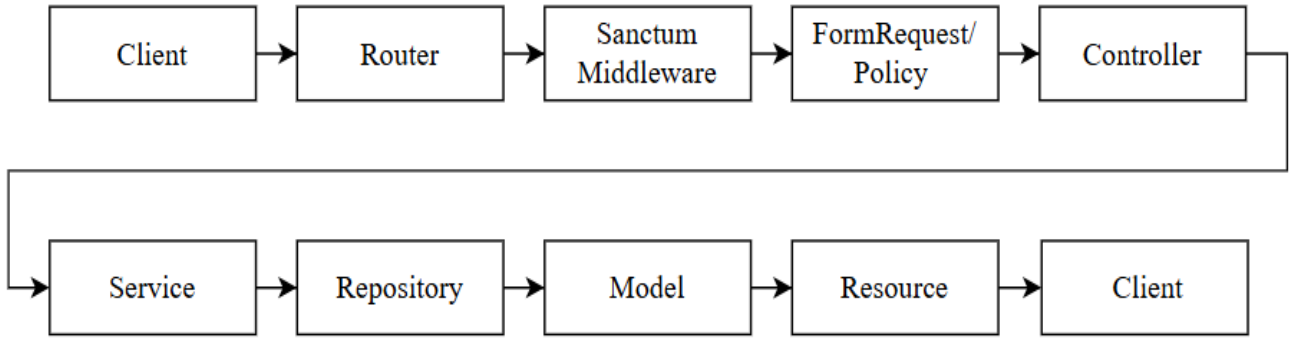

Рисунок 3.1 — Діаграма послідовності обробки HTTP-запиту

Першою точкою входу є маршрутизатор, який визначає цільову URL-адресу і відповідний HTTP-метод. Одразу після цього проміжне програмне забезпечення здійснює первинну перевірку запиту, зокрема автентифікацію користувача за допомогою інструментів пакету Sanctum. Далі до роботи долучається клас валідації, який перевіряє вхідні дані на відповідність встановленим правилам і делегує права доступу відповідним політикам безпеки.

Після цього запит перехоплює контролер, який займає роль координатора між шарами, викликаючи необхідний сервіс бізнес-логіки і повертаючи кінцевий API-ресурс. Сама ж бізнес-логіка застосунку розроблена в шарі сервісу. Наступний шар — репозиторій. Він відповідає за безпосередню взаємодію з базою даних і повне приховання логіки побудови запитів ORM Eloquent. Безпосередня структура таблиць і зв’язки між ними описуються в Eloquent-моделі, яка відображає сутності бази даних. На завершальному етапі обробки дані проходять через API-ресурс, який трансформує модель у стандартизовану JSON-відповідь для передачі її назад клієнтові.

Взаємодія на боці клієнта побудована за принципом передачі даних від інтерфейсу користувача до мережевого рівня через послідовні програмні рівні. Весь процес починається зі сторінок, які відповідають за загальне компонування екрана. Керування даними і синхронізація станів між цими компонентами покладена на сховища, які керують глобальним станом усього застосунку. Для взаємодії із сервером сховища звертаються до виділених API-сервісів, які містять у собі функції для виконання конкретних запитів. Наостанок HTTP-клієнт здійснює мережеве з’єднання з API-сервером і передає сформовані запити.

3.2 Структура бази даних

Проєктування бази даних виконувалося з урахуванням принципів нормалізації.

Кожна таблиця відповідає за збереження окремого типу даних, а зв’язки між ними реалізовано за допомогою зовнішніх ключів. Для забезпечення цілісності даних у системі використовуються каскадні операції видалення, механізм автоматичного знищення всіх залежних записів у пов’язаних таблицях у разі видалення основного об’єкта.

Структура створеної бази даних (рисунок 3.2) складається з п’яти взаємопов’язаних таблиць.

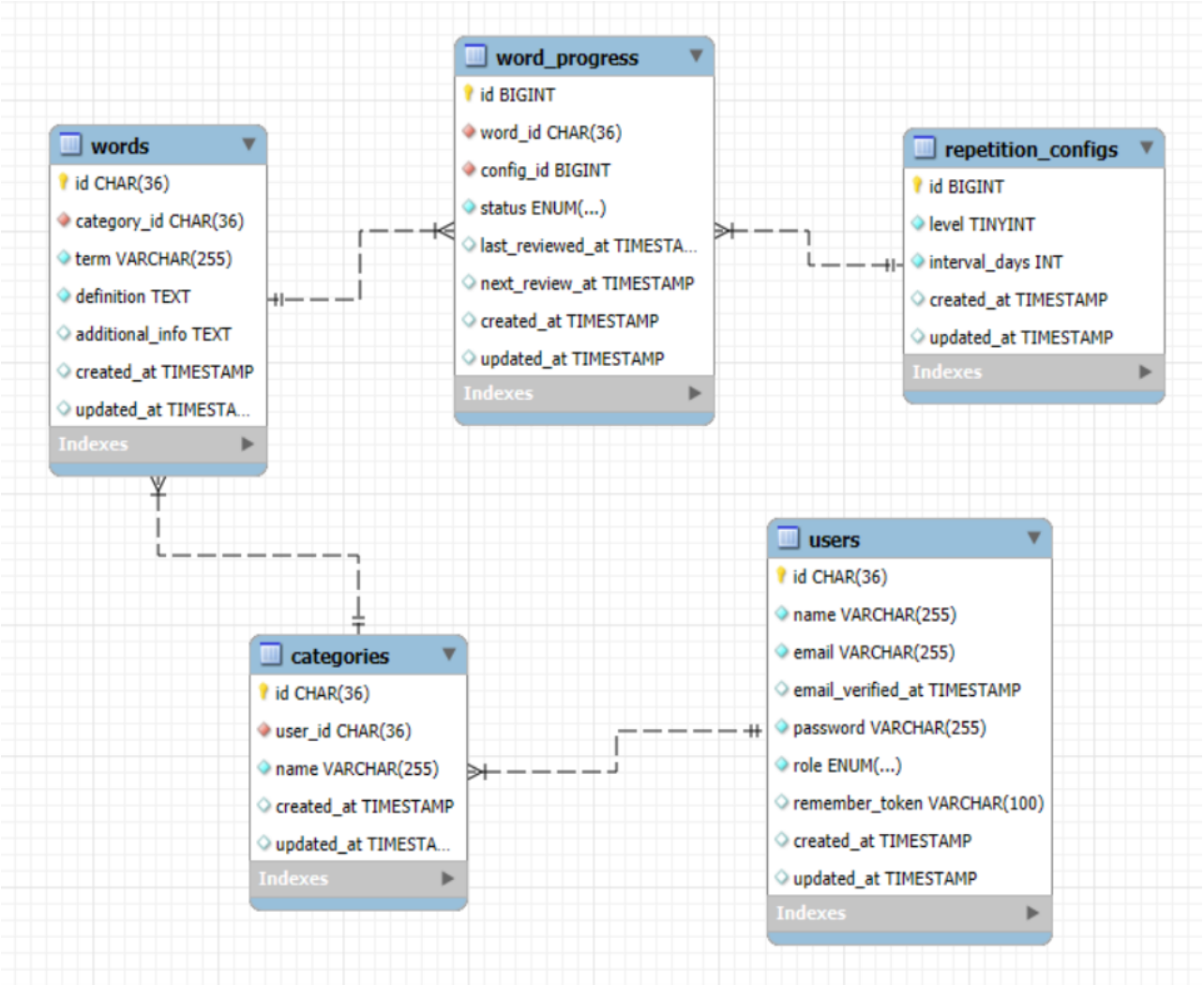

Рисунок 3.2 — Структура створеної бази даних

Першою є таблиця users, призначена для збереження облікових даних користувачів, забезпечення їхньої автентифікації, розмежування ролей у системі. У цій таблиці зберігаються унікальні ідентифікатори користувачів, їхні імена, електронні адреси, а також хешовані паролі. Із таблицею users безпосередньо пов’язана таблиця categories. Вона використовується для створення і збереження персональних категорій слів, які допомагають користувачеві структурувати власний словниковий запас. Один користувач може створювати необмежену кількість категорій, що реалізує зв’язок типу «один до багатьох» через зовнішній ключ user_id.

Наступною важливою сутністю є таблиця words, яка містить записи слів користувача. Кожне слово прив’язується до конкретної категорії і може містити сам іншомовний термін, переклад або додаткову інформацію про слово, таку як транскрипція, примітки або приклади використання цього слова. Між таблицями categories і words також реалізовано зв’язок «один до багатьох», оскільки одна категорія може містити значну кількість слів.

Логіка вивчення та алгоритм повторення реалізовано за допомогою таблиці word_progress. Таблиця відповідає за збереження інформації про прогрес вивчення кожного окремого слова, його поточний статус, часову мітку останнього перегляду і дату його наступного показу. Саме ця таблиця забезпечує автоматичне формування списку слів для повторення на серверному боці застосунку шляхом фільтрації записів. Між таблицями words і word_progress реалізовано зв’язок «один до одного», оскільки кожне слово має тільки один актуальний запис прогресу його вивчення.

Останньою таблицею є repetition_configs, яка використовується для збереження глобальних конфігурацій рівнів для інтервального повторення. У ній визначаються часові проміжки між повтореннями для різних етапів засвоєння матеріалу.

Кожен запис із таблиці word_progress містить посилання на відповідний рівень через зовнішній ключ config_id.

Завдяки такому архітектурному рішенню, система динамічно перераховує майбутній час показу картки, просто зчитуючи кількість днів із конфігураційної таблиці відповідно до поточного рівня вивчення цього слова в користувача, що робить алгоритм гнучким для потенційного масштабування або для зміни інтервалів у майбутньому.

Усі таблиці містять поля для автоматичної фіксації часу створення й оновлення записів. Це може спростити контроль змін у таблицях і подальший аналіз даних.

Конкретні часові інтервали для реалізації алгоритму наведено у таблиці 3.1.

Таблиця 3.1 — Конфігурація рівнів інтервального повторення

Рівень	Часовий інтервал	Пояснення дати повторення
1	1	Через день
2	1	Через день
3	3	Через три дні
4	7	Через тиждень
5	14	Через два тижні
6	30	Через місяць
7	30	Через місяць
8	30	Через місяць

На початкових етапах система вимагає щоденного повернення до матеріалу. Надалі, за умови успішних відповідей, часовий проміжок збільшується: спочатку до трьох днів, потім до тижня і до двох тижнів. Останні рівні передбачають фіксовані інтервали у тридцять днів. Такий підхід гарантує, що слово продовжуватиме перевірятися через місяць кілька разів поспіль, перш ніж система змінить його статус і визнає слово засвоєним.

У слова може бути три різні статуси (таблиця 3.2).

Таблиця 3.2 — Конфігурація статусів слів

Статус	Опис
added	Початковий статус слова після додавання
learning	Статус слова, яке знаходиться у процесі вивчення
mastered	Статус слова, вивчення якого було завершено

Впровадження цієї системи статусів допомагає керувати вибіркою даних під час генерації тренувальних сесій. Нові слова не з’являються під час тренувань, доки

користувач на вирішить почати їхнє вивчення, а повністю засвоєна лексика прибирається зі щоденних перевірок.

Крім цього, статистика користувача обраховується і відображається за допомогою цих статусів.

3.3 Реалізація алгоритму інтервальних повторень

Основою навчального процесу в вебзастосунку є алгоритм інтервальних повторень, призначений для ефективного запам’ятовування іншомовної лексики. Його головна задача полягає в формуванні тренувальних сесій для кожного користувача залежно від рівня засвоєння конкретних слів. Такий підхід надає можливість оптимізувати процес навчання і зосередити увагу користувача саме на тому матеріалі, який потребує повторення.

Під час створення тренувальної сесії система виконує автоматичний відбір слів із бази даних. До вибірки потрапляють тільки ті слова, які мають статус learning і відповідають умові next_review_at ≤ now(). Це означає, що система враховує не тільки поточний день, а й усі пропущені дати повторень. Завдяки цьому навіть у випадку тривалої перерви в навчанні користувач не втратить слова, які мали бути повторені раніше. Вони залишаться в черзі і повернуться в наступній навчальній сесії, а не втратяться назавжди. Таке рішення дає можливість підтримувати безперервність процесу навчання й уникати втрати прогресу через нерегулярне використання застосунку.

Логіка оновлення прогресу слова реалізована на серверному боці в методі applyAnswer класу WordProgressRepository. Саме цей метод відповідає за аналіз відповіді користувача під час тренування і подальшу зміну стану слова в системі. Після завершення кожної перевірки клієнтська частина надсилає результат на сервер, де автоматично виконується оновлення відповідних параметрів у базі даних.

Метод applyAnswer приймає два основні параметри: об’єкт поточного прогресу слова WordProgress і логічне значення isCorrect, яке визначає правильність відповіді користувача. На основі цього значення система обирає один із двох сценаріїв подальшої обробки.

Графічне подання послідовності роботи алгоритму наведено на рисунку 3.3. На схемі відображено етап перевірки відповіді, оновлення рівня слова і розрахунку дати останнього і наступного повторення.

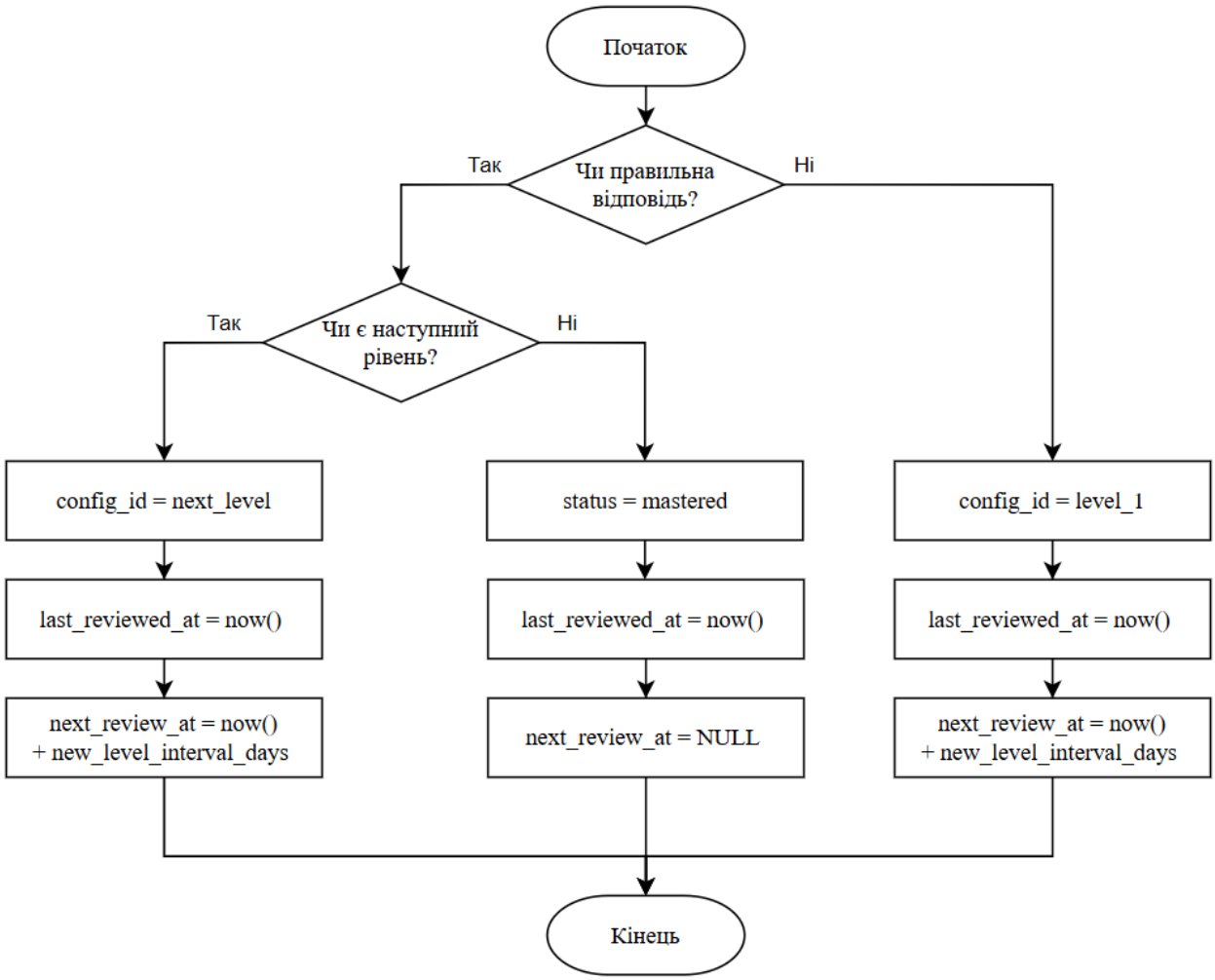

Рисунок 3.3 — Схема алгоритму опрацювання відповіді користувача

У випадку правильної відповіді користувача система перевіряє, чи існує для поточного слова наступний рівень. Якщо такий рівень доступний, слово переводиться на новий етап засвоєння, а дата його наступного повторення

автоматично розраховується відповідно до конфігурації інтервалів наступного рівня. Одночасно в базі даних оновлюється час останньої успішної перевірки. Завдяки цьому добре засвоєні слова демонструються користувачеві значно рідше.

Якщо ж слово вже перебуває на останньому рівні, система автоматично присвоює йому статус mastered, а наступна дата повторення не розраховується. У такому випадку слово вважається повністю вивченим і більше не потраплятиме до наступних тренувальних сесій, оскільки запит на вибірку слів ігноруватиме об’єкти з цим статусом. Таким чином відбувається поступове зменшення кількості матеріалу і користувач не відволікається на вже добре засвоєні слова.

У разі неправильної відповіді алгоритм працює за іншим принципом. Система скидає поточний прогрес слова і повертає його до початкового рівня вивчення. Після цього для слова встановлюється мінімальний інтервал повторення, що забезпечує його повторну появу вже найближчим часом. Такий механізм допомагає швидко виявляти проблемні слова, відразу ж збільшуючи частоту їхнього повторення.

Подібна організація алгоритму дає можливість динамічно адаптувати навчальний процес під індивідуальні особливості користувача. Слова, які запам’ятовуються легко, поступово переходять до довших інтервалів повторення, тоді як складні повторюються більш інтенсивно. Це також мінімізує втрату відчуття прогресу через монотонне вивчення того самого матеріалу.

Використання такого підходу позитивно впливає не тільки на ефективність запам’ятовування, але й на загальний рівень мотивації користувача. Завдяки автоматизованій вибірці слів для навчання на основі часових міток система уникає ситуацій, коли користувачеві доводиться занадто часто повторювати вже добре засвоєний матеріал або, навпаки, надовго втрачати з уваги складні слова.

Окремою перевагою реалізованого алгоритму є можливість його подальшого розширення. У майбутньому система може враховувати додаткові фактори, зокрема швидкість відповіді користувача, кількість попередніх помилок або середній рівень успішності під час тренувань.

3.4 Маршрути вебзастосунку

У таблиці 3.3 наведено структуру всіх маршрутів вебзастосунку, через які клієнтська частина взаємодіє із серверним API. Кожен маршрут відповідає за виконання конкретної операції, використовує відповідний HTTP-метод залежно від типу дії над даними. Завдяки побудові API за принципами REST структура запитів стає зрозумілою і зручною для подальшого розширення функціоналу системи.

Таблиця 3.3 — Маршрути вебзастосунку

Метод	URL	Опис маршруту
POST	/api/auth/register	Реєстрація
POST	/api/auth/login	Вхід у систему
POST	/api/auth/logout	Вихід із системи
GET	/api/auth/me	Отримання профілю поточного користувача
DELETE	/api/auth/account	Видалення облікового запису користувача
GET	/api/categories	Отримання списку всіх категорій
POST	/api/categories	Створення нової категорії
GET	/api/categories/{category}	Отримання інформації про конкретну категорію
PUT/PATCH	/api/categories/{category}	Оновлення даних категорії
DELETE	/api/categories/{category}	Видалення категорії
GET	/api/categories/{category}/words	Отримання слів конкретної категорії з фільтрацією
GET	/api/words/stats	Загальна статистика слів
POST	/api/words/batch-start-learning	Масове переведення слів у статус активного вивчення

Кінець таблиці 3.3

GET	/api/words	Отримання загального списку всіх слів
POST	/api/words	Додавання слова
GET	/api/words/{word}	Отримання інформації про конкретне слово
PUT/PATCH	/api/words/{word}	Редагування слова
DELETE	/api/words/{word}	Видалення слова
POST	/api/words/{word}/start-learning	Переведення слова у статус активного вивчення
POST	/api/words/{word}/answer	Фіксація відповіді і перерахунок інтервалу
GET	/api/review/today	Слова для повторення сьогодні

Маршрути вебзастосунку поділяються на публічні й захищені. Публічні маршрути доступні без попередньої автентифікації. Вони використовуються для створення нового облікового запису або входу користувача в систему. Захищені ж маршрути, навпаки, потребують наявності токена доступу і забезпечують доступ до функцій керування категоріями, словами, статистикою і безпосередньо модулем інтервальних повторень. Завдяки такому підходу обмежується доступ до персональних даних і запобігається виконання неавторизованих дій.

Взаємодія між частинами вебзастосунку відбувається у форматі JSON. Кожен API-запит повертає відповідь, яка містить необхідні дані або інформацію про помилку. Формат відповідей є уніфікованим, що забезпечує просту інтеграцію між компонентами системи, а також допомагає легко обробляти результати різних запитів на боці клієнта.

Важливу роль відіграє система обробки помилок. Якщо користувач надсилає некоректні або неповні дані, сервер повертає статус 422 і перелік помилок валідації (рисунок 3.4). Таким чином на клієнтській частині можуть відображатися повідомлення, завдяки яким користувач зрозуміє, які дії йому потрібно виконати, щоб отримати бажаний результат.

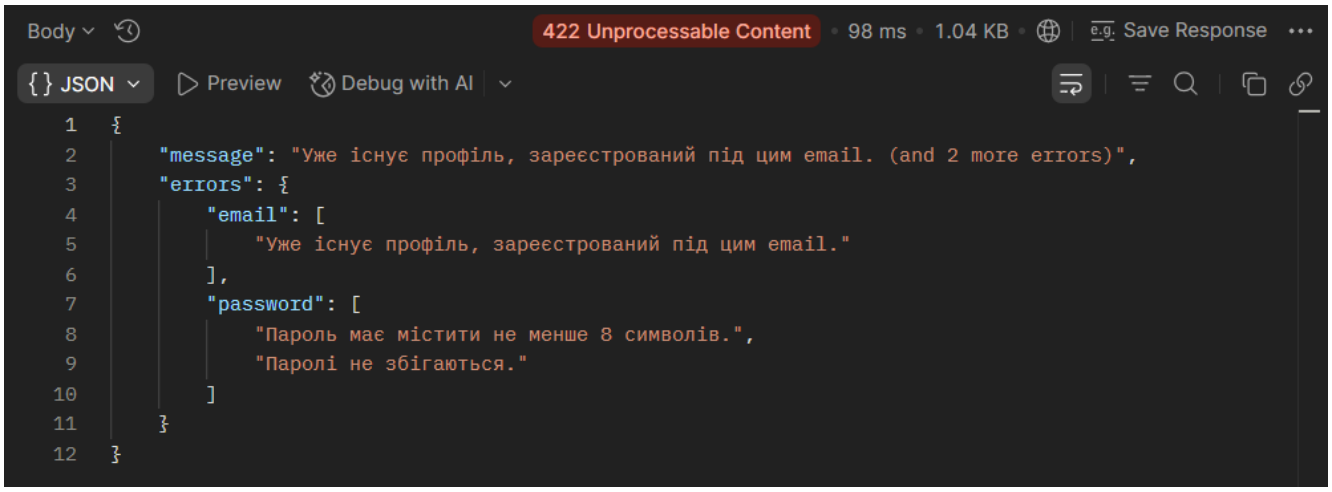

Рисунок 3.4 — Відповідь сервера під час помилки валідації

У випадку успішного виконання запиту сервер повертає статус 200 або 201 і об’єкт із необхідними даними. Наприклад, після авторизації користувача система поверне токен доступу разом із інформацією про профіль користувача (рисунок 3.5).

Рисунок 3.5 — Відповідь сервера на запит авторизації користувача

Під час створення нових записів сервер повертає створений об’єкт і статус 201, що підтверджує успішне додавання інформації до бази даних. На рисунку 3.6 демонструється відповідь сервера на запит створення нової категорії.

Рисунок 3.6 — Відповідь сервера на запит створення нової категорії

Для отримання інформації про об’єкт використовуються GET-запити, які повертають структуровані масиви даних. На рисунку 3.7 демонструється відповідь сервера на запит отримання списку всіх слів користувача.

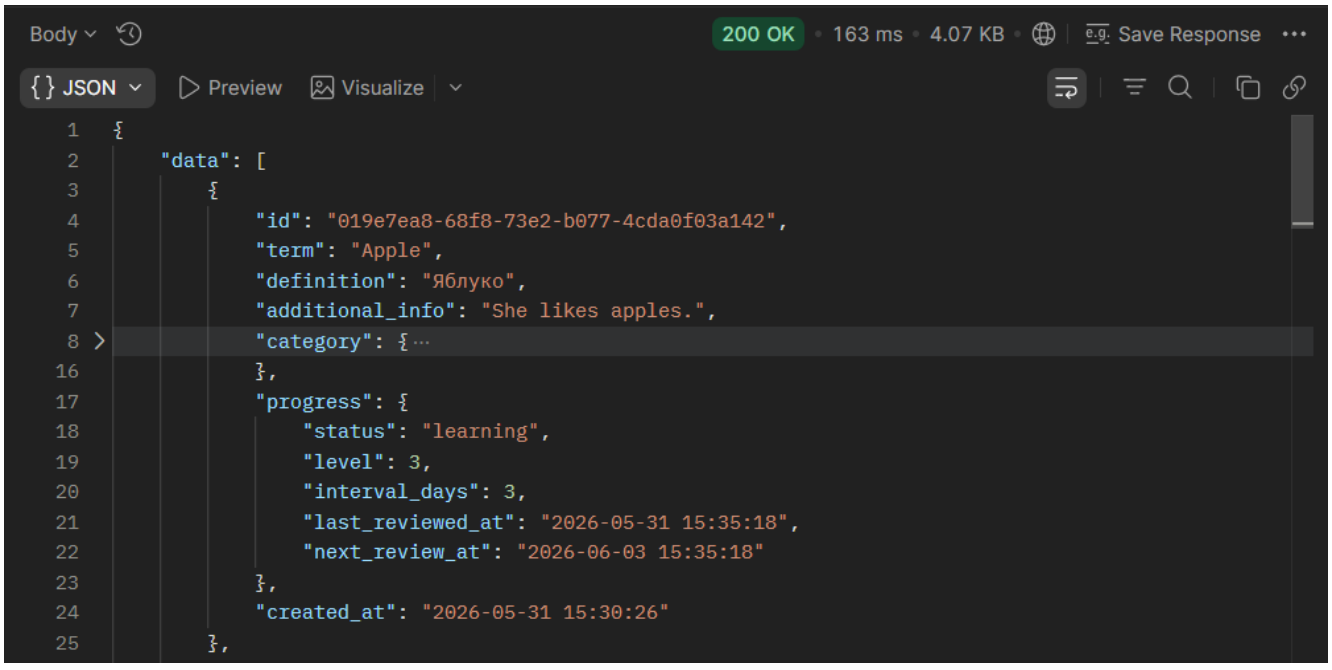

Рисунок 3.7 — Приклад отримання списку слів користувача

У випадку спроби доступу до захищених маршрутів без авторизації сервер повертає статус 401 (рисунок 3.8).

Такий механізм забезпечує захист персональних даних користувачів і запобігає виконанню несанкціонованих дій.

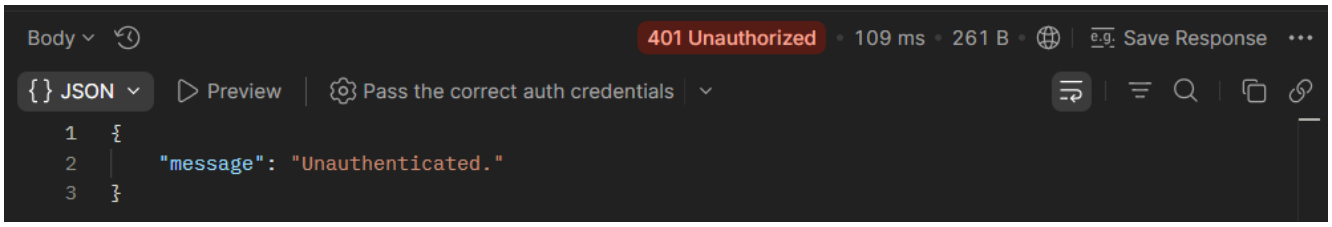

Рисунок 3.8 — Відповідь сервера при відсутності авторизації

Отже, використання JSON як основного формату передачі даних забезпечує просту взаємодію між клієнтом і сервером, уніфікує структуру API-відповідей і спрощує подальше масштабування вебзастосунку.

3.5 Діаграма прецедентів

Використання діаграми прецедентів спрощує аналіз вимог до застосунку, допомагає визначити межі відповідальності окремих модулів і забезпечує більш зрозуміле подання логіки роботи системи на етапі проєктування. Представлена на рисунку 3.9 діаграма прецедентів наочно відображає всі функціональні можливості системи. Єдиним актором у цій моделі виступає користувач, який є головним ініціатором усіх процесів у вебзастосунку. Діаграма охоплює функції, пов’язані з керуванням обліковим записом, організацією навчального матеріалу, взаємодією зі словником і проходженням тренувальних сесій.

Процес взаємодії з платформою починається з прецеденту реєстрації. На цьому етапі користувач створює персональний профіль, який надалі використовується для збереження записів слів, категорій і статистики навчання. Після успішної реєстрації користувач отримує можливість увійти до профілю через прецедент авторизації. Цей процес забезпечує безпечний доступ до персональних даних і функціоналу вебзастосунку.

Після входу до системи користувач отримує доступ до функцій керування профілем. До них належать перегляд статистики навчання, вихід із профілю і повне видалення облікового запису.

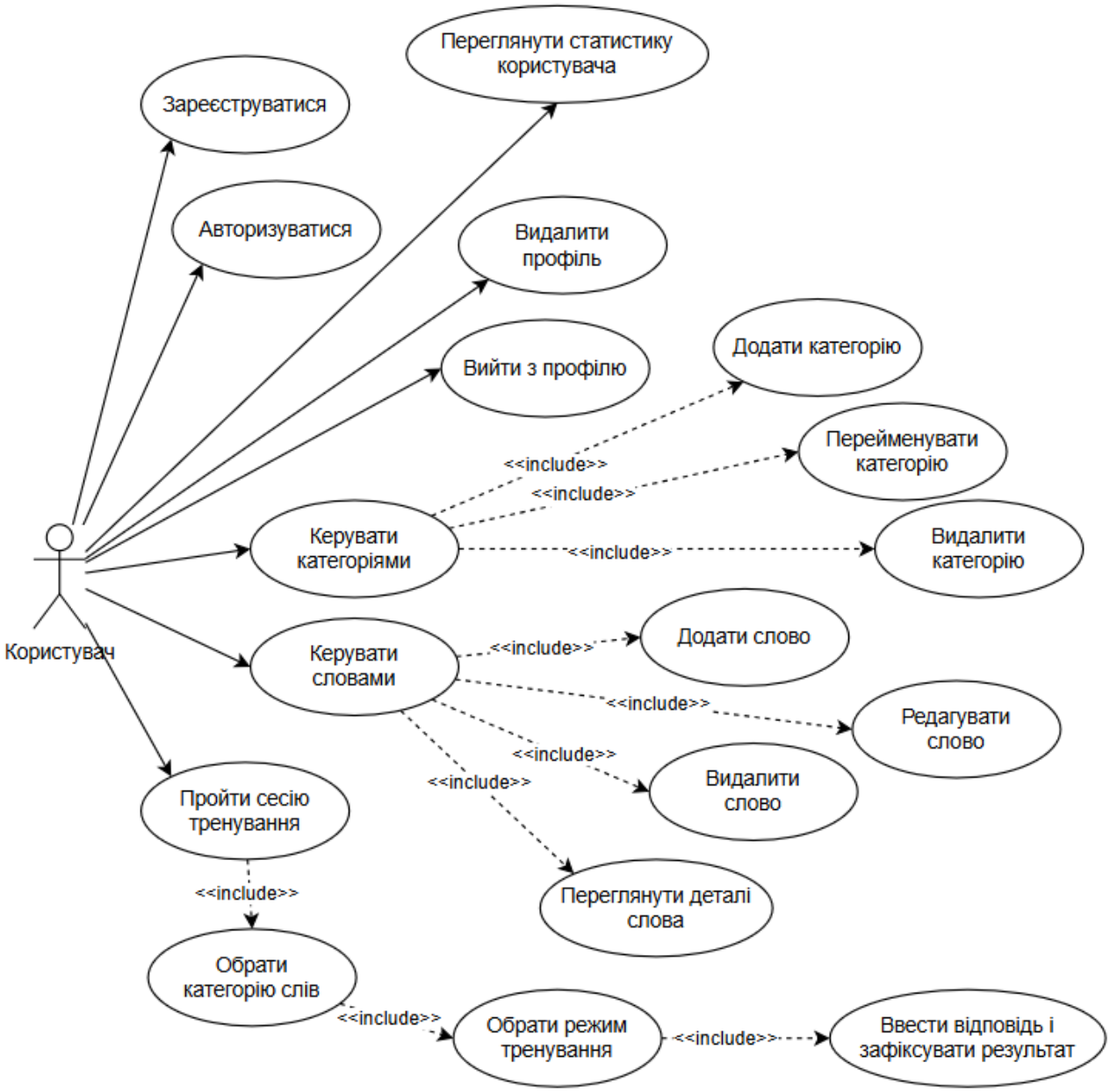

Рисунок 3.9 — Діаграма прецедентів

Прецедент перегляду статистики дає можливість відстежувати динаміку вивчення лексики, аналізувати кількість засвоєних слів та оцінювати ефективність навчання. Видалення профілю забезпечує повне очищення персональних даних користувача і пов’язаного навчального контенту.

Окремим функціональним блоком є керування категоріями слів. Цей прецедент об’єднує процеси створення, перейменування й видалення категорій. Категорії використовуються для систематизації навчального матеріалу і поділу

словникового запасу за темами, мовами або рівнями складності. Завдяки такому підходу користувач може зручно організовувати власний словник і швидко знаходити потрібний матеріал.

Наступним важливим елементом діаграми є керування словами. Цей прецедент включає додавання нових слів, редагування вже створених записів, перегляд деталей слова і його видалення. Можливість редагування забезпечує швидке оновлення або виправлення інформації, а функція перегляду деталей дає можливість отримати повну інформацію про конкретне слово.

Центральним елементом системи є прецедент проходження тренувальної сесії, який реалізує основну навчальну функцію вебзастосунку. Перед початком тренування користувач обирає категорію слів, з якою бажає працювати. Після цього система пропонує вибір режиму навчання. Під час проходження тренування користувач вводить власну відповідь, а система автоматично перевіряє її правильність і фіксує результат у базі даних. На основі отриманої відповіді виконується оновлення прогресу слова і розрахунок дати наступного повторення.

4 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Після першого відкриття вебзастосунку користувач потрапляє на стартову сторінку системи, яка виконує роль початкового екрана (рисунок 4.1). На цій сторінці відображається назва застосунку, тематична ілюстрація і основні кнопки навігації для переходу до авторизації або створення нового облікового запису. Інтерфейс стартової сторінки розроблено в мінімалістичному стилі, щоб не перевантажувати користувача зайвою інформацією. Його основною метою є швидке ознайомлення користувача з призначенням системи і надання доступу до початкових функцій.

Рисунок 4.1 — Стартова сторінка

Для створення нового облікового запису користувач переходить на сторінку реєстрації, де розміщена форма введення персональних даних (рисунок 4.2). Вона містить поля для введення імені користувача, електронної пошти, пароля і підтвердження пароля. Усі поля забезпечені текстовими підказками для полегшення процесу заповнення форми.

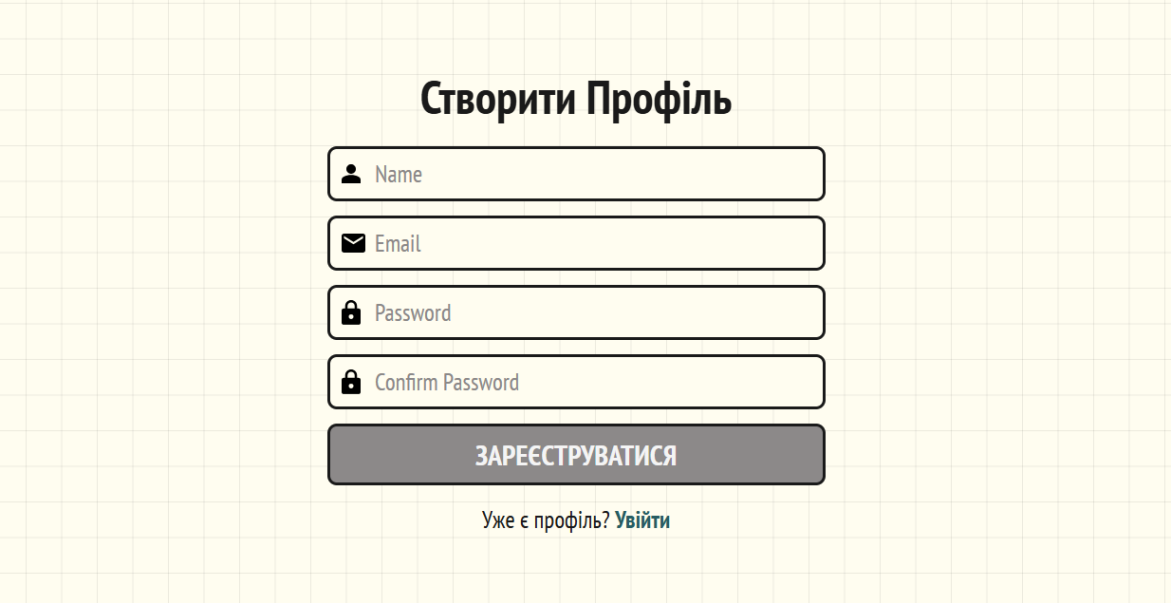

Рисунок 4.2 — Сторінка реєстрації

Кнопка підтвердження реєстрації залишається неактивною доти, доки користувач не заповнить усі обов’язкові поля. Створення нового профіля також не відбудеться доти, доки користувач не виконає встановлені умови для значень у полях. Пароль повинен містити щонайменше вісім символів, а значення в полях введення пароля і його підтвердження мають повністю збігатися.

У випадку виникнення помилки система відобразить відповідне повідомлення над формою у червоній рамці (рисунок 4.3). Завдяки цьому користувач може швидко зрозуміти, які саме дані були введені некоректно.

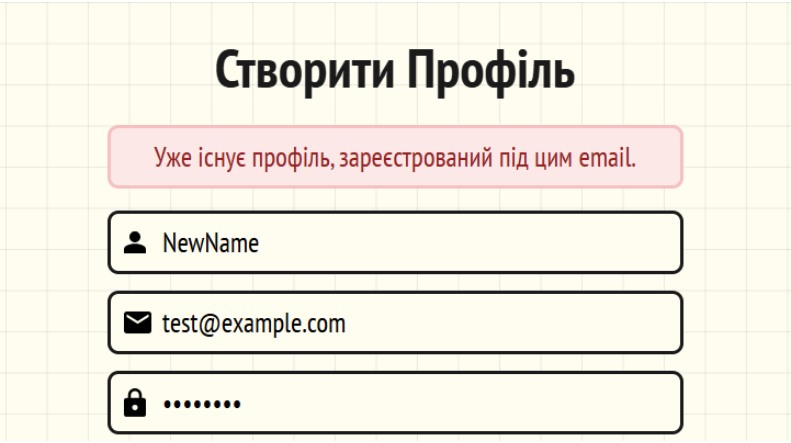

Рисунок 4.3 — Повідомлення про помилку на сторінці реєстрації

Після успішної авторизації користувач автоматично перенаправляється на головну сторінку застосунку. На всіх основних сторінках вебзастосунку зліва присутня фіксована бокова панель навігації. Вона містить основні розділи системи і забезпечує швидкий перехід між ними. Активний пункт меню виділяється темнішим кольором, що допомагає користувачеві легко визначити поточний розділ застосунку. Фіксоване положення панелі забезпечує швидкий перехід між розділами без необхідності повертатися на головну сторінку.

Головна сторінка використовується для роботи зі словами користувача (рисунок 4.4). На ній відображається загальна кількість слів, а також статистика за окремими статусами: нові слова, слова в процесі вивчення і вже засвоєна лексика. Такий підхід допомагає швидко оцінити поточний стан навчального процесу.

Рисунок 4.4 — Головна сторінка

Нижче від статистики розташовані вкладки для перемикання між різними категоріями слів залежно від їхнього статусу. Активна вкладка підсвічується зеленим кольором, що покращує орієнтацію користувача під час роботи із системою. Для швидкого пошуку необхідних записів реалізовано текстове поле пошуку, яке виконує фільтрацію слів за самим терміном, його перекладом або назвою категорії.

Відображення слів реалізовано у вигляді таблиці. Кожен рядок таблиці містить основну інформацію про слово і доступні з ним дії: перегляд детальної інформації, його видалення. Завдяки такому підходу таблиця має лаконічний вигляд, а інтерфейс не перевантажується великою кількістю інформації.

Для зручності реалізовано можливість сортування таблиці за будь-якою колонкою. Після натискання на заголовок виконується сортування записів, а повторне натискання змінює напрямок сортування. Поточний напрямок додатково позначається спеціальною стрілкою біля назви колонки.

Для перегляду повної інформації про слово використовується окреме модальне вікно, яке відкривається після натискання піктограми перегляду (рисунок 4.5).

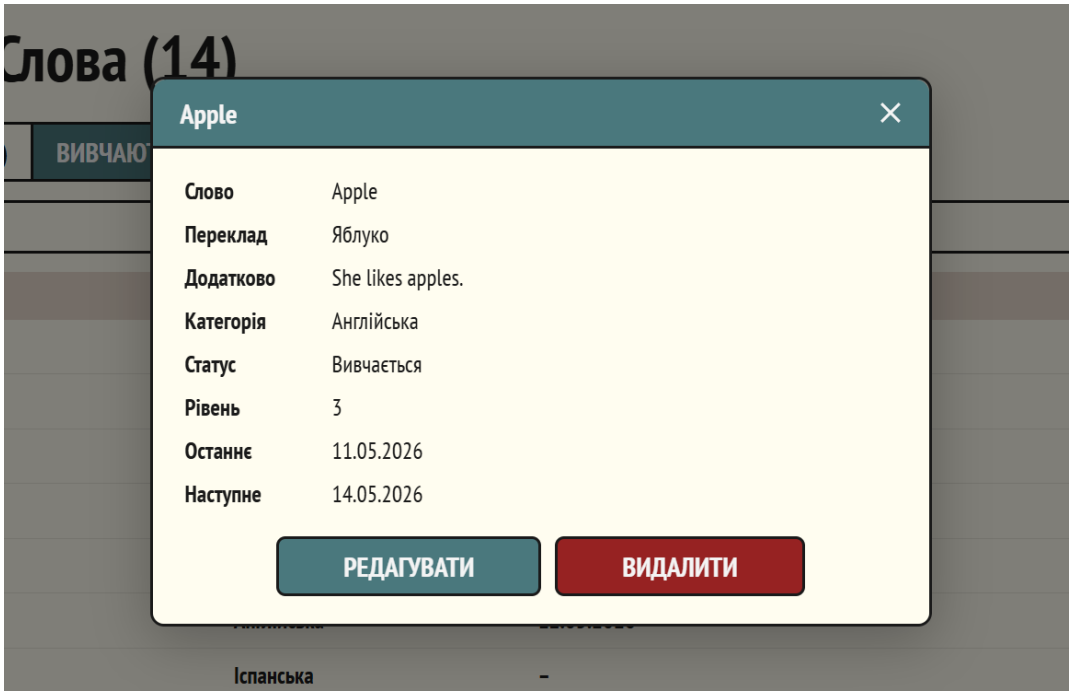

Рисунок 4.5 — Перегляд деталей слова

У цьому вікні відображається термін, переклад, додаткова інформація, категорія слова, поточний статус навчання, рівень слова, дата останнього перегляду і наступного повторення. У нижній частині вікна розташовані кнопки для редагування або видалення слова. Це дає користувачеві можливість швидко виконати необхідні дії без повернення до таблиці.

На головній сторінці також реалізовано функцію додавання нових слів. Для цього використовується окреме модальне вікно, яке відкривається після натискання кнопки «Додати слово».

Модальне вікно містить форму для введення інформації про нове слово, його переклад, категорію і додаткову інформацію (рисунок 4.6). Для запобігання створенню неповних записів кнопка додавання залишається недоступною доти, доки користувач не заповнить усі обов’язкові поля.

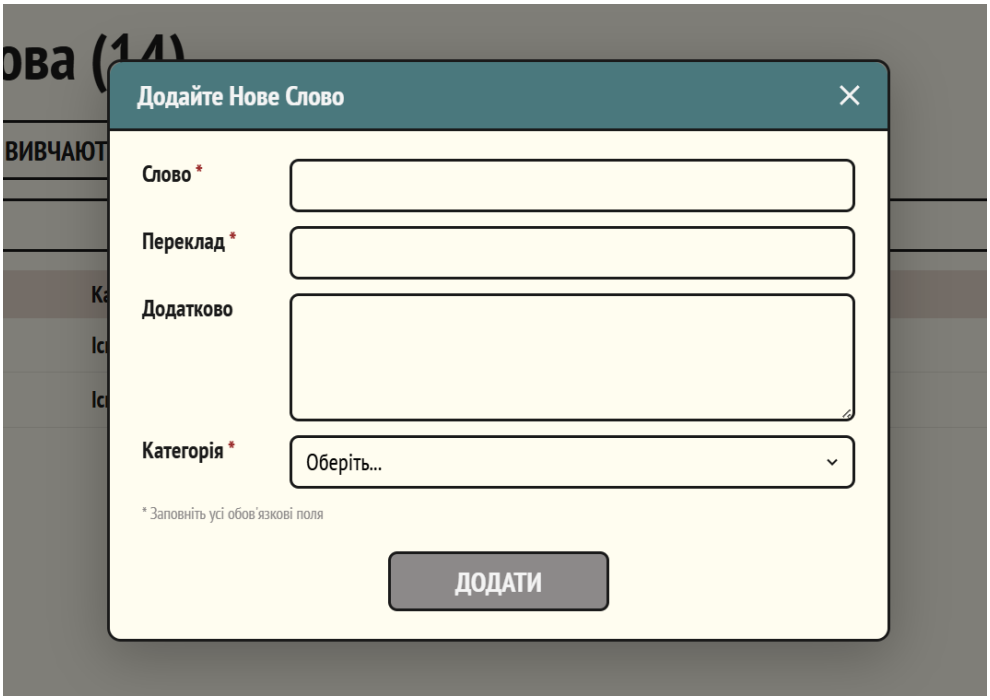

Рисунок 4.6 — Додавання слова

Функцію редагування слова реалізовано через модальне вікно, структура якого майже повністю повторює форму створення нового словникового запису. Основна відмінність полягає в тому, що всі поля автоматично заповнюються поточними даними слова, завдяки чому користувач може швидко внести необхідні зміни без повторного введення інформації. У формі доступне редагування самого слова, перекладу, додаткового контексту і категорії, до якої належить запис.

Користувач може змінювати категорію слова навіть у випадку, якщо редагування виконується безпосередньо зі сторінки конкретної категорії. Після

успішного збереження оновлені дані миттєво відображаються в таблиці без необхідності перезавантаження сторінки.

Сторінка категорій призначена для організації слів користувача за окремими тематичними групами. Вона відображає всі створені категорії у вигляді окремих елементів, що дає можливість швидко переходити між різними наборами слів та ефективно структурувати навчальний матеріал (рисунок 4.7).

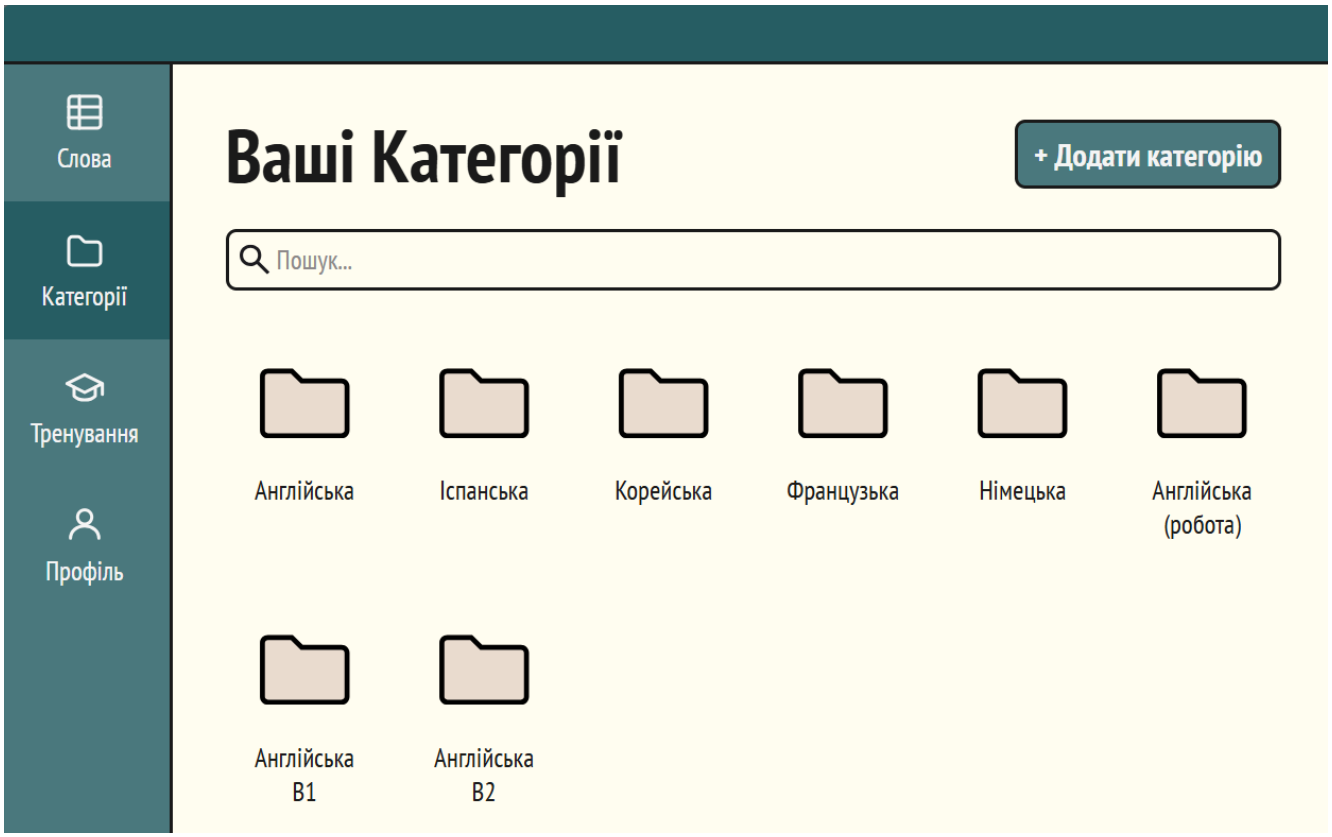

Рисунок 4.7 — Список категорій користувача

Для спрощення навігації на сторінці реалізовано поле пошуку, за допомогою якого користувач може швидко знайти потрібну категорію за її назвою. Це особливо важливо у випадку великої кількості створених категорій, коли ручний перегляд усього списку стає незручним.

Створення нової категорії здійснюється через окреме модальне вікно, яке відкривається після натискання кнопки додавання категорії. Вікно містить поле для введення назви і кнопку підтвердження створення (рисунок 4.8).

Після успішного створення нова категорія автоматично додається до загального списку без оновлення сторінки.

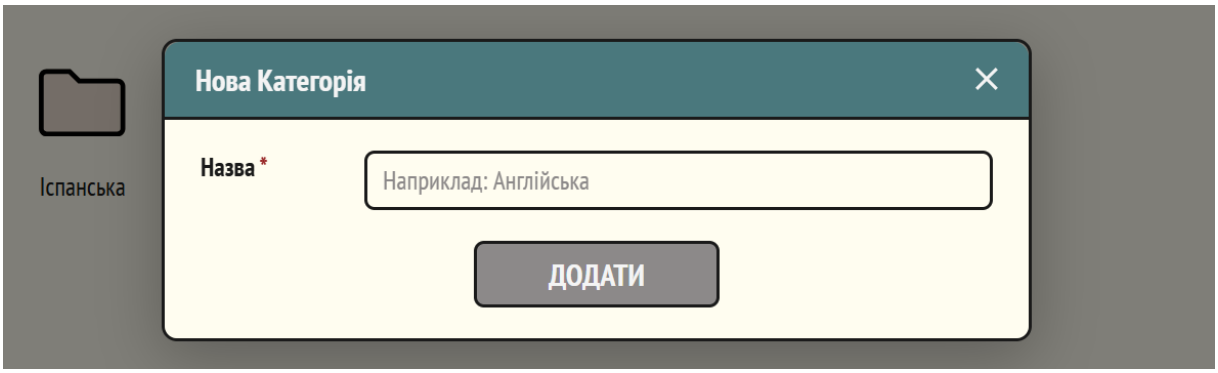

Рисунок 4.8 — Створення категорії

Сторінка конкретної категорії за своєю структурою подібна до головної сторінки слів, однак містить кілька важливих відмінностей, пов’язаних із роботою всередині окремої тематичної групи (рисунок 4.9).

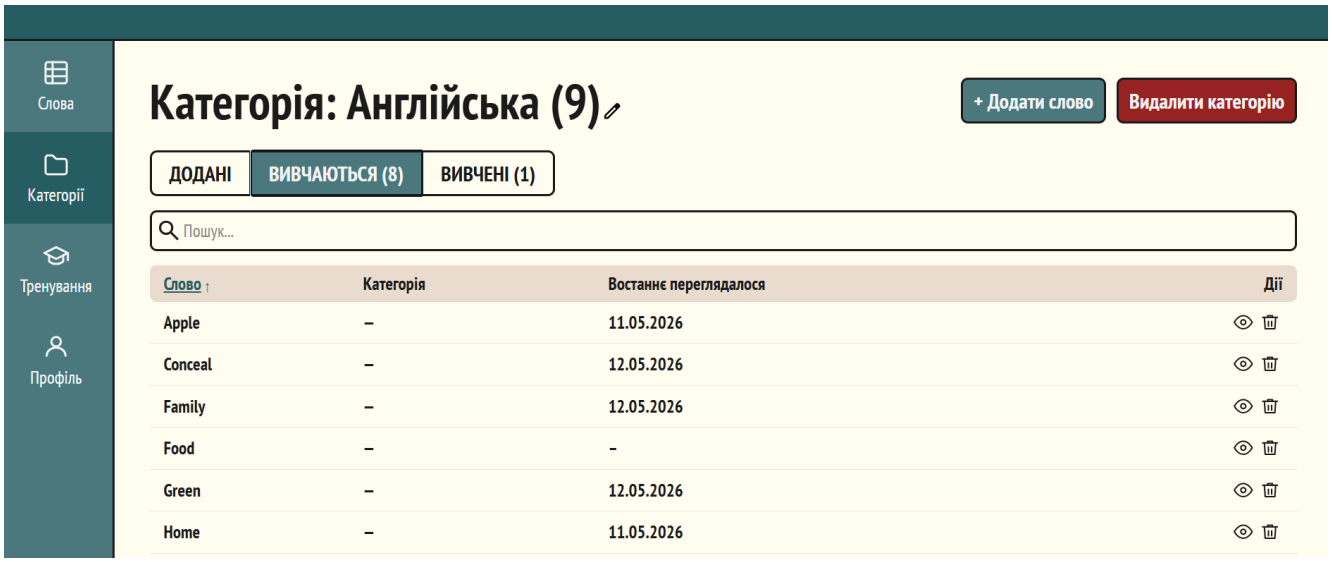

Рисунок 4.9 — Сторінка категорії

У верхній частині сторінки відображається назва поточної категорії і кнопка її редагування. Після натискання відкривається модальне вікно, яке за структурою повністю відповідає формі створення нової категорії. На відміну від порожньої

форми створення, під час редагування поле назви вже містить поточне значення, що дає можливість користувачеві швидко змінити назву категорії.

Таблиця слів на сторінці категорії також має дещо змінений вигляд. Оскільки всі слова вже належать до однієї конкретної категорії, колонка з назвою категорії не відображається. Таким чином інформація не дублюється, а інтерфейс стає більш компактним.

На сторінці також розміщена кнопка видалення категорії. Оскільки видалення категорії автоматично призводить до видалення всіх слів, що належать до неї, для підтвердження цієї дії використовується окреме модальне вікно (рисунок 4.10).

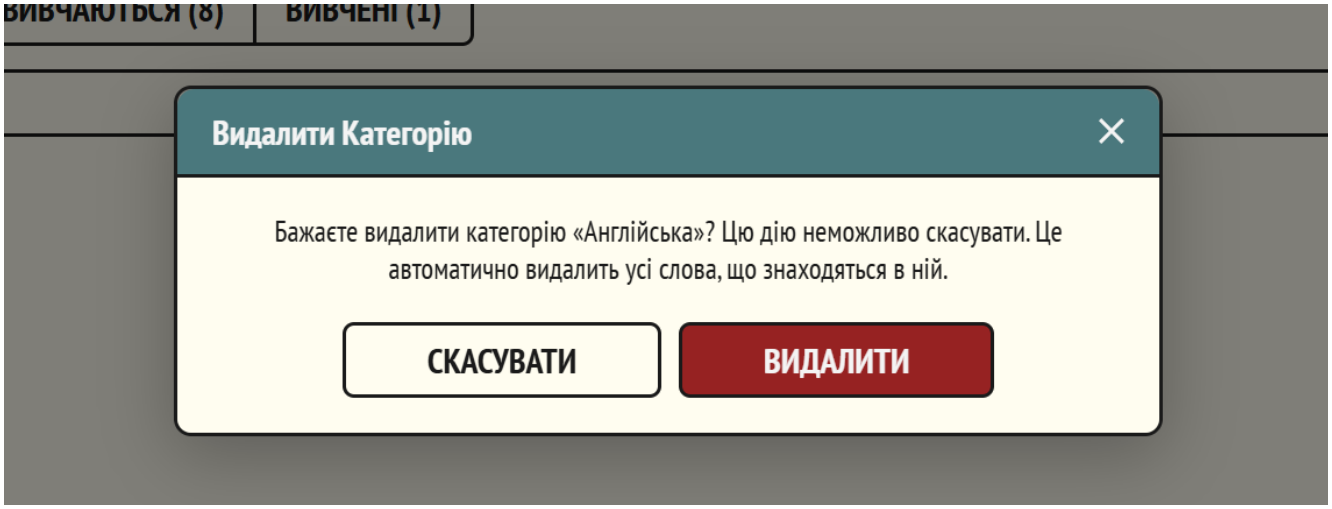

Рисунок 4.10 — Підтвердження видалення

У вікні підтвердження користувачеві відображається коротке пояснення наслідків видалення і доступні дії: скасування операції або її підтвердження. Такий підхід допомагає уникнути випадкової втрати важливих даних і підвищує безпеку взаємодії із системою.

Сторінка тренування використовується для запуску навчальних сесій. Перед початком тренування користувач має змогу обрати категорію слів, з якою бажає працювати. За замовчуванням система автоматично обирає першу доступну категорію (рисунок 4.11).

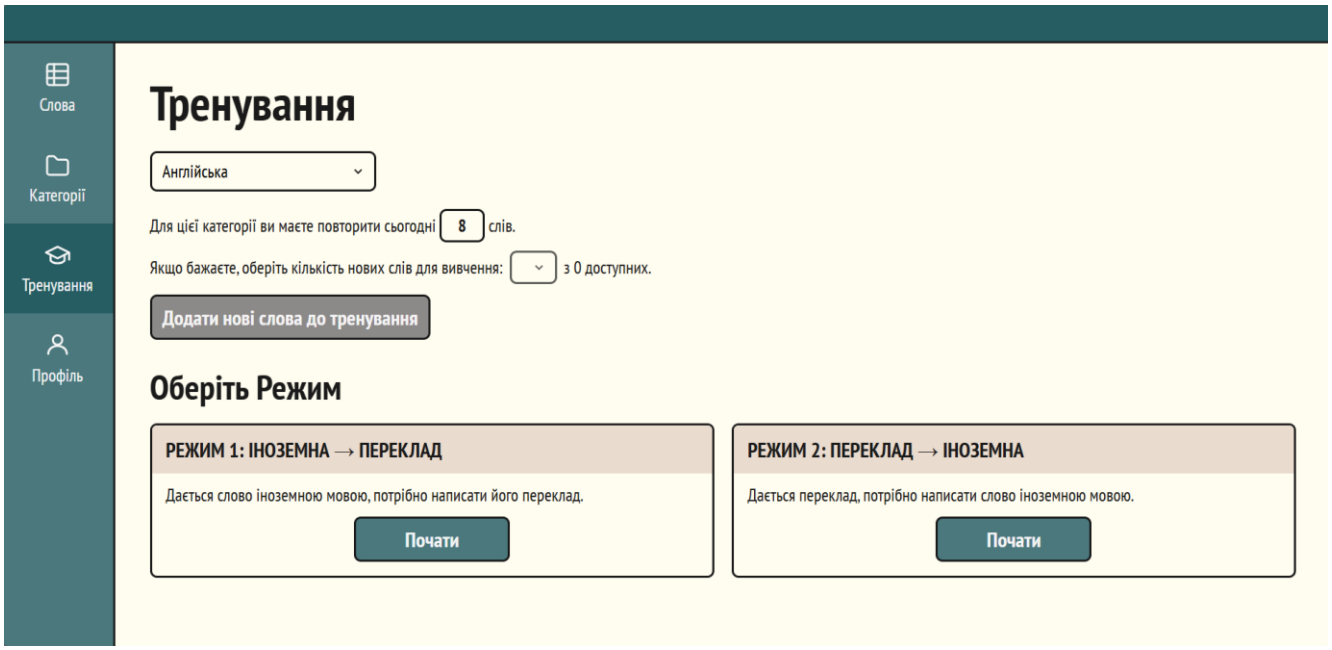

Рисунок 4.11 — Сторінка тренування

Після вибору категорії система відображає кількість слів, які необхідно повторити сьогодні, а також кількість нових слів, доступних для додавання до процесу навчання. Користувач може самостійно визначити, скільки нових слів буде включено до поточної сесії. Після цього необхідно обрати режим тренування і запустити навчальний процес.

Після початку тренування відкривається новий екран (рисунок 4.12). У верхній частині екрана розташована смужка прогресу, що показує кількість уже пройдених слів і загальний обсяг слів у поточній сесії. Поруч розміщується кнопка завершення тренування, яка дає користувачеві можливість у будь-який момент повернутися на попередню сторінку.

У центральній частині екрана залежно від обраного режиму відображається слово або його переклад, а також поле для введення відповіді. Після введення варіанта відповіді користувач натискає кнопку підтвердження, після чого система автоматично виконує перевірку результату.

У разі правильної відповіді відображається повідомлення про успішне проходження перевірки, правильний переклад і додаткова інформація, якщо вона була вказана для слова (рисунок 4.13).

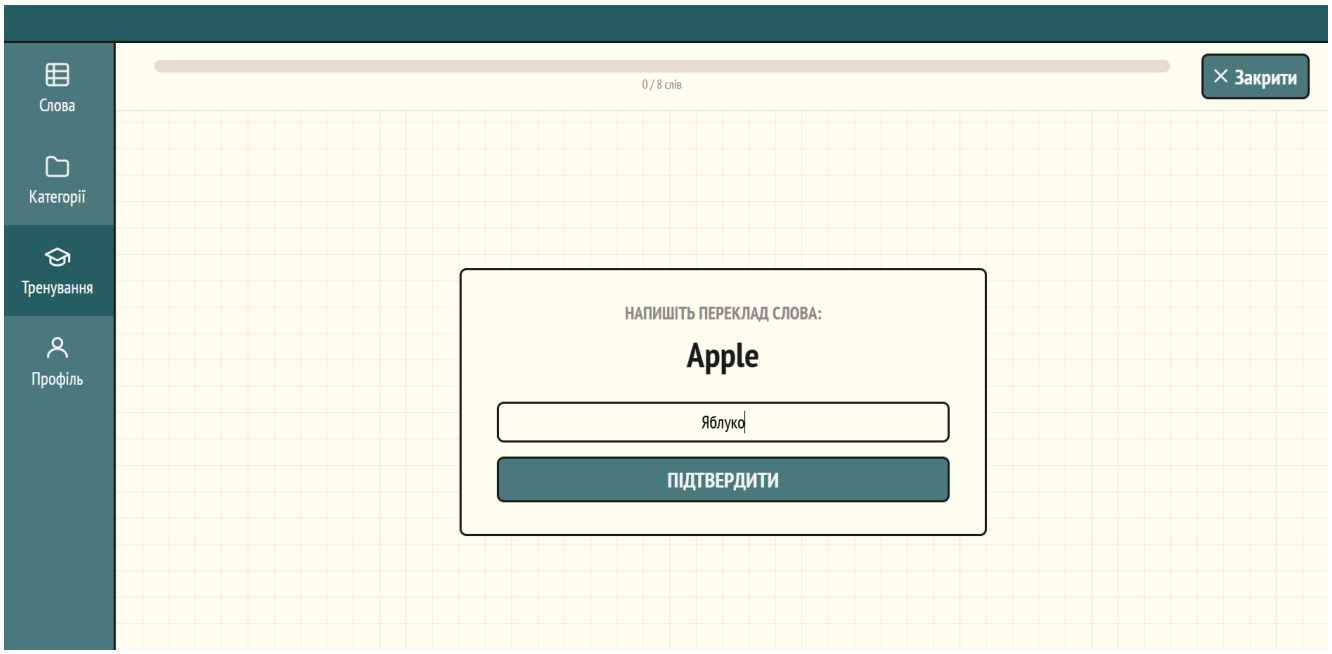

Рисунок 4.12 — Сесія тренування

Рисунок 4.13 — Відображення правильної відповіді

Якщо відповідь користувача є неправильною, система повідомляє про це і показує правильний варіант (рисунок 4.14). Такий механізм допомагає користувачеві одразу бачити помилку і краще закріпити матеріал у пам’яті.

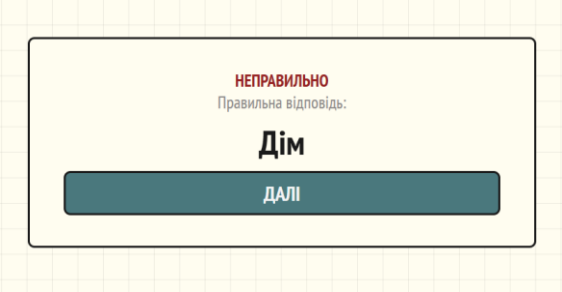

Рисунок 4.14 — Відображення неправильної відповіді

Після завершення перевірки користувач може перейти до наступного слова за допомогою кнопки «Далі». Навчальна сесія триває доти, доки всі слова з поточного списку не будуть опрацьовані або користувач не завершить її, натиснувши на кнопку (рисунок 4.15).

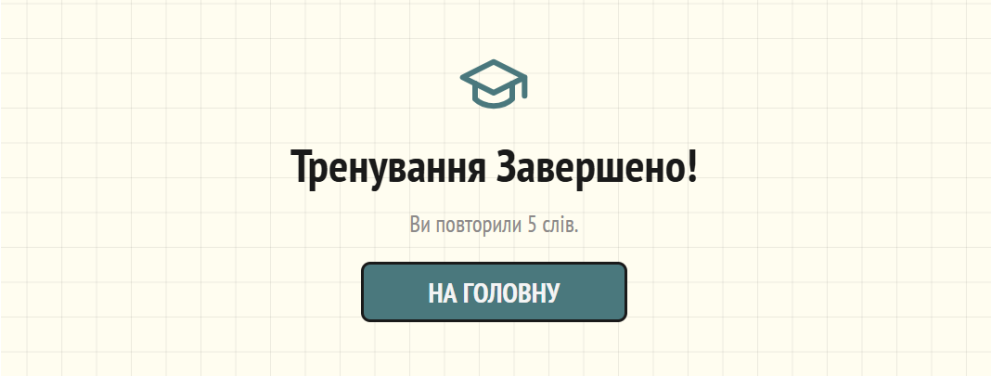

Рисунок 4.15 — Завершення тренувальної сесії

На сторінці профілю відображаються основні дані, зокрема ім’я й електронна пошта користувача (рисунок 4.16).

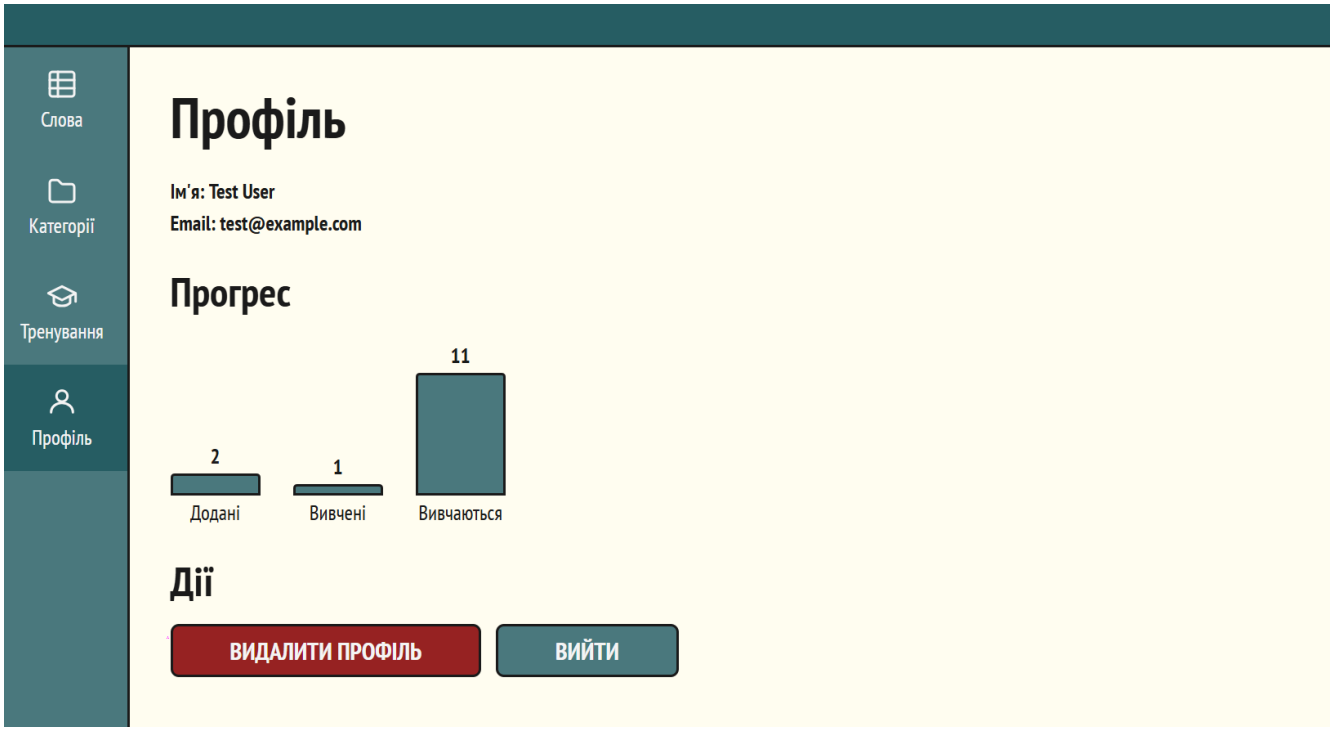

Рисунок 4.16 — Профіль користувача

Окремий блок містить статистику навчання у вигляді стовпчиків діаграми, які наочно демонструють співвідношення між доданими, вивченими і словами, які вивчаються.

Такий спосіб візуалізації дає користувачеві можливість швидко оцінити власний прогрес не тільки в текстовому вигляді, а ще й графічно.

У нижній частині сторінки розташовані кнопки «Видалити профіль» і «Вийти». Перша забезпечує повне видалення облікового запису разом із пов’язаними даними, тоді як друга завершує поточну сесію користувача й повертає його на стартову сторінку застосунку.

ВИСНОВКИ

У ході виконання дипломної роботи відповідно до визначеної мети і поставлених завдань було проведено комплексний аналіз наявних програмних засобів для засвоєння іншомовної лексики, що допомогло виявити їхні ключові недоліки, зокрема обмежені можливості користувачів щодо персоналізації навчального матеріалу. На основі аналізу когнітивних методик було досліджено метод інтервальних повторень для покращення засвоєння нової інформації.

У результаті розроблено повноцінний вебзастосунок для вивчення іноземних слів з використанням методу інтервального повторення, який складається з REST API на фреймворці Laravel і клієнтської частини на фреймворці Vue. Серверна частина побудована за принципами чистої архітектури з розподілом на шари: репозиторій, сервіс, контролер. У межах завдання реалізовано такі основні функціональні можливості як реєстрація і автентифікація користувачів через токени; керування словниковим запасом з організацією за довільними категоріями; система інтервального повторення з рівневим алгоритмом і автоматичним розрахунком дати наступного повторення; два режими тренування; відображення статистики прогресу.

На завершальному етапі проведено тестування створеного програмного забезпечення і виправлення виявлених помилок.

Порівняно з наявними рішеннями розроблений застосунок надає можливість роботи з довільним контентом без прив’язки до мови. До перспектив розвитку можна віднести додавання до словникових записів підтримки зображень та аудіо; удосконалення алгоритму повторення слів; мобільний застосунок на основі того самого API.

Результати роботи було апробовано на XXIII Міжнародній науково-практичній конференції молодих вчених та студентів “Сучасні проблеми наукового забезпечення енергетики: безпека, стійкість, ІТ та екологічний моніторинг” (до 40-річчя Чорнобильської катастрофи), м. Київ, 17-22 квітня 2026 року (додаток А).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Діброва Є.В., Кублій Л.І. Вебзастосунок для вивчення іноземних мов. *Сучасні проблеми наукового забезпечення енергетики.* Матеріали XXIII Міжнародної науково-практичної конференції молодих вчених та студентів “Сучасні проблеми наукового забезпечення енергетики: безпека, стійкість, ІТ та екологічний моніторинг” (до 40-річчя Чорнобильської катастрофи), м. Київ, 17-22 квітня 2026 року. Київ: КПІ ім. Ігоря Сікорського, Вид-во “Політехніка”, 2026. С. 432-433. URL: https://iate.kpi.ua/uploads/p_182_80374158.pdf (дата звернення: 30.05.2026).

2. Maatuk A. M. et al. The COVID-19 pandemic and E-learning: challenges and opportunities from the perspective of students and instructors. *Journal of Computing in Higher Education*. 2022. Vol. 34. P. 21–38. URL: https://doi.org/10.1007/s12528-021-09274-2 (дата звернення: 30.05.2026).

3. Digital Education Action Plan: policy background. European Education Area. URL: https://education.ec.europa.eu/focus-topics/digital-education/actions/plan (дата звернення: 30.05.2026).

4. Fitria T. N. The impact of gamification on students’ motivation: *A Systematic Literature Review. LingTera.* 2023. Vol. 9, no. 2. P. 47–61. URL: https://doi.org/10.21831/lt.v9i2.56616 (дата звернення: 30.05.2026).

5. Nielsen J. 10 Usability Heuristics for User Interface Design. *Nielsen Norman Group.* URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (дата звернення: 30.05.2026).

6. Mayer R. E., Moreno R. Nine Ways to Reduce Cognitive Load in Multimedia Learning. *Educational Psychologist*. 2003. Vol. 38, no. 1. P. 43–52. URL: https://doi.org/10.1207/s15326985ep3801_6 (дата звернення: 30.05.2026).

7. Web Content Accessibility Guidelines (WCAG) 2.1. W3C. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 30.05.2026).

8. The right time to learn: mechanisms and optimization of spaced learning — PMC. PMC Home. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC5126970/ (дата звернення: 24.05.2026).

9. Sirwan Mohammed G., Wakil K., Sirwan Nawroly S. The Effectiveness of Microlearning to Improve Students' Learning Ability. *International Journal of Educational Research Review*. 2018. Vol. 3, no. 3. P. 32–38. URL: https://doi.org/10.24331/ijere.415824 (дата звернення: 30.05.2026).

10. Anki — powerful, intelligent flashcards. Anki. URL: https://apps.ankiweb.net/ (дата звернення: 24.05.2026).

11. Quizlet. URL: https://quizlet.com/ (дата звернення: 24.05.2026).

12. Learn a language. Memrise is authentic, useful & personalised. URL: https://www.memrise.com/ (дата звернення: 24.05.2026).

13. Duolingo. URL: https://www.duolingo.com/ (дата звернення: 30.05.2026).

14. PHP: Посібник з PHP — Manual. PHP. URL: https://www.php.net/manual/uk/ (дата звернення: 24.05.2026).

15. Laravel 13.x — The clean stack for Artisans and agents. Laravel. URL: https://laravel.com/docs/13.x (дата звернення: 24.05.2026).

16. Laravel Sanctum | Laravel 13.x — The clean stack for Artisans and agents. Laravel. URL: https://laravel.com/docs/13.x/sanctum (дата звернення: 30.05.2026).

17. What is MySQL. MySQL. URL: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html (дата звернення: 24.05.2026).

18. Vue.js — The Progressive JavaScript Framework: веб-сайт. URL: https://vuejs.org/.

19. Defining a Store | Pinia. Pinia | The intuitive store for Vue.js. URL: https://pinia.vuejs.org/core-concepts/ (дата звернення: 30.05.2026).

20. Getting Started | Vue Router. Vue Router | The official Router for Vue.js. URL: https://router.vuejs.org/guide/ (дата звернення: 30.05.2026).

21. First steps | axios | Promise based HTTP client. axios | Promise based HTTP client. URL: https://axios.rest/pages/getting-started/first-steps (дата звернення: 30.05.2026).

ДОДАТОК А

Апробація

Матеріали XXIII Міжнародної науково-практичної конференції молодих вчених і студентів «Сучасні проблеми наукового забезпечення енергетики: безпека, стійкість, ІТ та екологічний моніторинг»

«Вебзастосунок для вивчення іноземних мов»

УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ІАТЕ_ЦТЕ_ТР_21

Аркушів 6

Київ — 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

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

СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ:

БЕЗПЕКА, СТІЙКІСТЬ, ІТ ТА ЕКОЛОГІЧНИЙ МОНІТОРИНГ

XXIII Міжнародна науково-практична конференція
молодих вчених та студентів

До 40-річчя Чорнобильської катастрофи

17–22 квітня 2026 року, м. Київ

Київ-2026

УДК 004.42:378.1

[1] Бакалаврант 4 курсу Діброва Є. В.
[1] Доц., к.т.н. Кублій Л. І.
https://scholar.google.com.ua/citations?hl=en&user=rXA1eMIAAAAJ

[1] КПІ ім. Ігоря Сікорського

ВЕБЗАСТОСУНОК ДЛЯ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ

Постановка проблеми та її актуальність. На сучасному ринку праці знання іноземної мови є важливою і часто обов'язковою вимогою до кваліфікованого працівника в багатьох галузях, зокрема й у сфері інформаційних технологій. Насамперед ключовим компонентом мовної компетентності виступає великий і добре структурований словниковий запас, оскільки саме він визначає здатність фахівця працювати з технічною документацією, брати участь у міжнародних проєктах, впевнено опановувати нові технології та інформацію.

Наявні застосунки для вивчення мов не завжди враховують професійну спрямованість лексики або не пропонують достатньо гнучких механізмів повторення слів, які адаптувалися б до реального прогресу користувача. Часто такі ресурси орієнтовані на пасивне споживання матеріалу без активної взаємодії, що знижує ефективність засвоєння нової лексики та ускладнює її подальше практичне застосування.

Аналіз останніх досліджень. Питання ефективного вивчення іноземної лексики активно досліджується. Серед відомих способів особливо виділяють метод інтервального повторення. Дослідження підтверджують, що інтервальне навчання допомагає формувати довготривалу пам'ять [1] і зменшує забування вивченого матеріалу завдяки повторному перегляду інформації через щоразу довші проміжки часу.

Розповсюдженим способом виконання цього методу є розподілення карток з інформацією на секції і їхнє подальше переміщення між секціями залежно від того, чи було дано правильну відповідь під час повторення. Недоліком такого методу є поступове накопичення великої кількості матеріалу, через що підтримувати чітку організацію секцій самостійно стає дедалі складніше. Тому для довготривалого систематичного навчання зручно використовувати саме цифрові застосунки.

До поширених інструментів, які реалізують принцип інтервального повторення, належать застосунок Anki [2] і навчальний сервіс Quizlet [3]. Вебзастосунок Anki базується на алгоритмі SM-2, розробленому спеціально для оптимізації інтервалів повторення. Користувач самостійно оцінює, наскільки добре запам'ятав слово чи поняття, після чого система автоматично визначає наступний інтервал появи картки. Завдяки цьому, а також можливості створювати власні набори карток і додавати до них звуковий супровід, Anki забезпечує високу адаптивність і гнучкість. На відміну від нього Quizlet використовує інтервальне повторення більше як допоміжний інструмент і акцентує увагу радше на тестах, тренуваннях і різноманітних режимах перевірки знань. Цей сервіс є зручним і все ще залишається розповсюдженим серед тих, хто надає перевагу більш ігровому формату навчання.

Формулювання мети. Метою дослідження є розробка вебзастосунку для вивчення іноземної мови шляхом ефективного розширення словникового запасу. Вебзастосунок повинен надавати користувачам можливість додавати власний матеріал, мати зручний інтерфейс для доступу до його функціональних можливостей.

Основна частина. Створений вебзастосунок орієнтований на вивчення іноземної лексики шляхом систематичного повторення доданих користувачем слів. Застосунок побудований за клієнт-серверною архітектурою і складається з клієнтської частини, серверної частини та бази даних. Клієнтська частина відповідає за відображення інтерфейсу

користувача, інструменти додавання нових слів, їхнє повторення і взаємодію з сервером. Серверна частина реалізує бізнес-логіку застосунку, обробляє запити користувачів, виконує обчислення інтервалів повторення, забезпечує доступ до бази даних. У базі даних зберігається інформація про користувачів, результати повторень, а також кожне додане слово разом із його визначенням, додатковою інформацією, набором спеціальних міток. Ці мітки відображають рівень засвоєння конкретного слова і визначають оптимальний момент для його повторення відповідно до інтервального принципу навчання.

У системі також передбачається роль адміністратора, який відповідає за керування користувачами, здійснює контроль структури бази даних і моніторинг стабільності роботи системи.

Користувач має можливість створювати власні колекції слів, додавати нові записи, редагувати наявні. Під час сеансів повторення система автоматично формує добірку слів відповідно до розрахованого інтервалу, фіксує результати відповідей. На основі отриманих даних застосунок оновлює рівень засвоєння слів та обчислює дату їхньої наступної появи. Механізм повторення реалізовано таким чином, що слова з нижчим рівнем засвоєння з'являються частіше, а добре засвоєні – рідше. Це дає можливість зосередити увагу на проблемній лексиці і зменшити появу вже вивченого матеріалу. Такий підхід реалізовує повністю адаптивне навчальне середовище, де кожне слово повторюється з тією частотою, яка гарантує його запам'ятовування, а поява нових слів залежить саме від рішень користувача.

Клієнтська частина вебзастосунку створена з використанням мови розмітки гіпертексту HTML, мови стилів CSS і мови для розробки інтерактивних динамічних вебсайтів JavaScript, що забезпечує інтуїтивний і сучасний інтерфейс користувача, швидку взаємодію з системою і динамічне оновлення даних без повного перезавантаження сторінки. Серверна частина створена з використанням мови програмування PHP, яка добре підходить для розробки веб-застосунків, оскільки забезпечує гнучкість реалізації бізнес-логіки та ефективну обробку HTTP-запитів.

Для зберігання даних застосовано систему керування базами даних MySQL, яка забезпечує підтримку складних запитів, високу швидкість роботи, цілісність даних. Використання бази даних дає можливість логічно структурувати доданий користувачами навчальний матеріал та організовувати набори слів. Це значно оптимізує витрати часу на роботу із системою, роблячи навчальний процес не тільки якісним, а й набагато ефективнішим.

Сукупність використаних технологій забезпечила можливість створити стабільний, продуктивний і доступний вебзастосунок. Оскільки доступ до системи здійснюється через веббраузер, то головною перевагою є відсутність необхідності встановлення спеціального програмного забезпечення – вебзастосунок можна безперешкодно використовувати з різних пристроїв та операційних систем.

Висновки. Розроблений вебзастосунок забезпечує структуроване зберігання лексичного матеріалу, адаптивне планування повторень, враховує індивідуальний прогрес користувача. Він створює умови для підвищення ефективності самостійного вивчення іноземної мови. Цей вебзастосунок також може бути використаний як основа для подальшого вдосконалення цифрових інструментів вивчення іноземних мов.

Перелік посилань:

1. Smolen P., Zhang Yi., Byrne J. H. The right time to learn: mechanisms and optimization of spaced learning. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC5126970/ (дата звернення: 09.02.2026).

2. Anki : веб-сайт. URL: https://apps.ankiweb.net/ (дата звернення: 09.02.2026).

3. Quizlet : веб-сайт. URL: https://quizlet.com/ (дата звернення: 09.02.2026).

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

ДИПЛОМ

ІІІ с т у п е н я

ЗА АКТИВНУ УЧАСТЬ У

XXIII МІЖНАРОДНІЙ НАУКОВО-ПРАКТИЧНІЙ КОНФЕРЕНЦІЇ МОЛОДИХ ВЧЕНИХ ТА СТУДЕНТІВ «СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ: БЕЗПЕКА, СТІЙКІСТЬ, ІТ ТА ЕКОЛОГІЧНИЙ МОНІТОРИНГ (ДО 40-РІЧЧЯ ЧОРНОБИЛЬСЬКОЇ КАТАСТРОФИ)»

нагороджується студентка гр. ТР – 21

кафедри ЦТЕ

Діброва Євгенія Василівна

В.о. директора НН ІАТЕ Ольга ЧЕРНОУСЕНКО

2026

ДОДАТОК Б

Фрагменти основних складових застосунку

«Вебзастосунок для вивчення іноземних мов»

УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ІАТЕ_ЦТЕ_ТР_21

Мова програмування — PHP

Аркушів 6

Київ — 2026

```
// Файл WordProgressRepository.php
<?php

namespace App\Repositories;

use App\Models\RepetitionConfig;
use App\Models\Word;
use App\Models\WordProgress;
use App\Repositories\Contracts\WordProgressRepositoryInterface;
use Illuminate\Support\Carbon;

class WordProgressRepository implements WordProgressRepositoryInterface
{

    /**
     * Create an initial progress record for the given word.
     *
     * Assigns the word to the first repetition configuration and
     * sets its status to "added".
     *
     * @return \App\Models\WordProgress
     */
    public function createForWord(Word $word): WordProgress
    {
        $firstConfig = RepetitionConfig::orderBy('level', 'asc')->firstOrFail();

        return WordProgress::create([
            'word_id'   => $word->id,
            'config_id' => $firstConfig->id,
            'status'    => 'added',
```

```
        ]);
    }

    /**
     * Move the word progress to the learning state.
     *
     * Updates the status to "learning" and schedules the next review
     * for the current time.
     *
     * @return \App\Models\WordProgress
     */
    public function moveToLearning(WordProgress $progress): WordProgress
    {
        $progress->update([
            'status'         => 'learning',
            'next_review_at' => now(),
        ]);

        return $progress->fresh('config');
    }

    /**
     * Reset the word progress back to the initial added state.
     *
     * Assigns the first repetition configuration and clears
     * review-related timestamps.
     *
     * @return \App\Models\WordProgress
     */
    public function resetToAdded(WordProgress $progress): WordProgress
```

```
{
    $firstConfig = RepetitionConfig::orderBy('level', 'asc')->firstOrFail();

    $progress->update([
        'status'    => 'added',
        'config_id' => $firstConfig->id,
        'next_review_at' => null,
        'last_reviewed_at' => null,
    ]);

    return $progress->fresh('config');
}

/**
 * Apply the result of a user's answer.
 *
 * Moves the progress forward or resets it depending on whether
 * the answer was correct, updates repetition dates, or marks
 * the word as mastered.
 *
 * @return \App\Models\WordProgress
 */
public function applyAnswer(WordProgress $progress, bool $isCorrect): WordProgress
{
    if ($isCorrect) {
        return $this->handleCorrectAnswer($progress);
    }

    return $this->handleWrongAnswer($progress);
}
```

```
/**
 * Handle a correct answer for the given word progress.
 *
 * Advances the progress to the next repetition level if available,
 * or marks the word as mastered when the final level is reached.
 *
 * @return \App\Models\WordProgress
 */
private function handleCorrectAnswer(WordProgress $progress): WordProgress
{
    $nextConfig = RepetitionConfig::query()
        ->where('level', '>', $progress->config->level)
        ->orderBy('level')
        ->first();

    // Немає наступної секції - слово вивчено
    if (! $nextConfig) {
        $progress->update([
            'status'          => 'mastered',
            'last_reviewed_at' => now(),
            'next_review_at'   => null,
        ]);

        return $progress->fresh('config');
    }

    $progress->update([
        'config_id'       => $nextConfig->id,
        'last_reviewed_at' => now(),
```

```
            'next_review_at'   => now()->addDays($nextConfig->interval_days),
        ]);

        return $progress->fresh('config');
    }

    /**
     * Handle an incorrect answer for the given word progress.
     *
     * Resets the progress to the first repetition level and schedules
     * the next review according to the initial interval.
     *
     * @return \App\Models\WordProgress
     */
    private function handleWrongAnswer(WordProgress $progress): WordProgress
    {
        $firstConfig = RepetitionConfig::orderBy('level', 'asc')->firstOrFail();

        $progress->update([
            'config_id'        => $firstConfig->id,
            'last_reviewed_at' => now(),
            'next_review_at'   => now()->addDays($firstConfig->interval_days),
        ]);

        return $progress->fresh('config');
    }
}
```